Methods in
Molecular Biology 2284

Springer Protocols

Ernesto Picardi *Editor*

RNA Bioinformatics

Second Edition

Humana Press

RNA Bioinformatics

Second Edition

Edited by

Ernesto Picardi

Dipartimento Di Bioscienze, Biotecnologie E Biofarmaceutica, Università degli Studi di Bari Aldo Moro, Bari, Italy

Humana Press

Editor
Ernesto Picardi
Dipartimento Di Bioscienze
Biotecnologie E Biofarmaceutica
Università degli Studi di Bari Aldo Moro
Bari, Italy

ISSN 1064-3745 ISSN 1940-6029 (electronic)
Methods in Molecular Biology
ISBN 978-1-0716-1306-1 ISBN 978-1-0716-1307-8 (eBook)
https://doi.org/10.1007/978-1-0716-1307-8

Cover Illustration Caption: Image created by Dr. Francesco Russo.

This Humana imprint is published by the registered company Springer Science+Business Media, LLC part of Springer Nature.
The registered company address is: 1 New York Plaza, New York, NY 10004, U.S.A.

Preface

In the last two decades, genomics and transcriptomics have been deeply transformed by massive sequencing technologies. In a few hours, researchers can profile nucleotide variants or quantify hundreds of genes in multiple experiments. In this era, dominated by Big Data and Omic Sciences, bioinformatics and computational biology play pivotal roles. Increasingly efficient algorithms are required to handle and store huge amounts of data and extract from them relevant biological information. Looking at main database resources for high-throughput sequencing (such as the Short Read Archive or the European Nucleotide Archive), it appears that a significant fraction of sequencing data is devoted to the study of RNA, a very versatile molecule with a plethora of functional roles, all indispensable to ensure cell homeostasis. Thanks to omics data and high performance servers, we are now able to explore entire transcriptomes of living organisms at unprecedented resolution and appreciate, one more time, the importance of investigating RNA molecules and the RNA world in general.

The aim of this book is to provide an overview of novel bioinformatics resources for exploring diverse aspects of RNA biology. Differently from the previous edition, here we focus on methods dealing with noncoding RNA (miRNAs, circRNAs, or lncRNAs), RNA modifications (m6A or RNA editing), single cell RNA-seq, and statistical models to handle count data from RNA-seq experiments. Nonetheless, the book also includes chapters based on the classical RNA bioinformatics methods, such as those for deciphering secondary and tertiary RNA structures, but revised taking into account deep sequencing data. Finally, we have inserted chapters describing methods to analyze RNA sequencing data from emerging third-generation sequencing technologies that could provide interesting insights into the transcriptional process and its regulation.

We really hope that this novel edition could fulfill the reader expectations. This book is the sum of different contributions from outstanding researchers in the field of RNA bioinformatics that I personally thank for their efforts. A special "thank you" is devoted to my double As (Angela and Adele) for their support and much patience.

Bari, Italy ***Ernesto Picardi***

Contents

Contributors

LUCA ALESSANDRÌ • *Department of Molecular Biotechnology and Health Sciences, University of Torino, Torino, Italy*
ERNESTO APARICIO-PUERTA • *Department of Genetics, University of Granada, Granada, Spain*
MADDALENA ARIGONI • *Department of Molecular Biotechnology and Health Sciences, University of Torino, Torino, Italy*
GABRIELE AUSIELLO • *Centre for Molecular Bioinformatics, Department of Biology, University of Rome Tor Vergata, Rome, Italy*
BŁAŻEJ BAGIŃSKI • *Institute for Research in Biomedicine (IRB Barcelona), The Barcelona Institute of Science and Technology, Barcelona, Spain*
BÉRÉNICE BATUT • *Bioinformatics Group, Department of Computer Science, Albert-Ludwigs-University Freiburg, Freiburg, Germany*
MARCO BECCUTI • *Department of Computer Science, University of Turin, Turin, Italy*
PIETRO BOCCALETTO • *Laboratory of Bioinformatics and Protein Engineering, International Institute of Molecular and Cell Biology in Warsaw, Warsaw, Poland*
VALENTINA BOUCHÉ • *Telethon Institute of Genetics and Medicine (TIGEM), Armenise/Harvard Laboratory of Integrative Genomics, Pozzuoli, Italy*
DAVIDE CACCHIARELLI • *Telethon Institute of Genetics and Medicine (TIGEM), Armenise/Harvard Laboratory of Integrative Genomics, Pozzuoli, Italy; Department of Translational Medicine, University of Naples "Federico II", Naples, Italy*
RAFFAELE A. CALOGERO • *Department of Molecular Biotechnology and Health Sciences, University of Turin, Turin, Italy*
ENRICA CALURA • *Department of Biology, University of Padova, Padova, Italy*
ANNAMARIA CARISSIMO • *Istituto per le Applicazioni del Calcolo "Mauro Picone", Consiglio Nazionale delle Ricerche, Naples, Italy*
TIZIANA CASTRIGNANÒ • *Department of Ecological and Biological Sciences (DEB), University of Tuscia, Viterbo, Italy*
KUNQI CHEN • *Department of Biological Sciences, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China; Institute of Ageing & Chronic Disease, University of Liverpool, Liverpool, UK*
MING CHEN • *Department of Bioinformatics, The State Key Laboratory of Plant Physiology and Biochemistry, College of Life Sciences, Zhejiang University, Hangzhou, China*
GIOVANNI CHILLEMI • *SCAI—Super Computing Applications and Innovation Department, CINECA, Rome, Italy; Department for Innovation in Biological, Agro-Food and Forest Systems, DIBAF, University of Tuscia, Viterbo, Italy*
UCIEL CHOROSTECKI • *Barcelona Supercomputing Centre (BSC-CNS), Barcelona, Spain; Institute for Research in Biomedicine (IRB), The Barcelona Institute of Science and Technology, Barcelona, Spain*
FRANCESCA CORDERO • *Department of Computer Science, University of Turin, Turin, Italy*
MATTIA D'ANTONIO • *SuperComputing Applications and Innovation Department, CINECA, Rome, Italy*
MICHELE DE BORTOLI • *Department of Clinical and Biological Sciences, University of Turin, Orbassano, Italy*

STEFANO DE PRETIS • *Center for Genomic Science of IIT@SEMM, Fondazione Istituto Italiano di Tecnologia (IIT), Milan, Italy*
MARIA A. DOYLE • *Research Computing Facility, Peter MacCallum Cancer Centre, Melbourne, VIC, Australia; Sir Peter MacCallum Department of Oncology, The University of Melbourne, Melbourne, VIC, Australia*
GIULIO FERRERO • *Department of Computer Science, University of Turin, Turin, Italy; Department of Clinical and Biological Sciences, University of Turin, Orbassano, Italy*
TIZIANO FLATI • *SCAI—Super Computing Applications and Innovation Department, CINECA, Rome, Italy*
BASTIAN FROMM • *Department of Molecular Biosciences, Stockholm University, Stockholm, Sweden*
MATTIA FURLAN • *Center for Genomic Science of IIT@SEMM, Fondazione Istituto Italiano di Tecnologia (IIT), Milan, Italy*
TONI GABALDON • *Barcelona Supercomputing Centre (BSC-CNS), Barcelona, Spain; Institute for Research in Biomedicine (IRB), The Barcelona Institute of Science and Technology, Barcelona, Spain; ICREA, Barcelona, Spain*
GENNARO GAMBARDELLA • *Telethon Institute of Genetics and Medicine (TIGEM), Armenise/Harvard Laboratory of Integrative Genomics, Pozzuoli, Italy; Department of Chemical Materials and Industrial Engineering, University of Naples "Federico II", Naples, Italy*
RAMYA GAMINI • *Pfizer Worldwide Research and Development, Cambridge, MA, USA*
XIJIN GE • *Department of Mathematics and Statistics, South Dakota State University, Brookings, SD, USA*
SILVIA GIOIOSA • *SCAI—Super Computing Applications and Innovation Department, CINECA, Rome, Italy*
CLAUDIO LO GIUDICE • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy*
ANTONIO GRIMALDI • *Telethon Institute of Genetics and Medicine (TIGEM), Armenise/Harvard Laboratory of Integrative Genomics, Pozzuoli, Italy*
MICHAEL HACKENBERG • *Department of Genetics, University of Granada, Granada, Spain*
MARC K. HALUSHKA • *Department of Pathology, Johns Hopkins University, Baltimore, MD, USA*
STEFAN HAMMER • *Department of Theoretical Chemistry, University of Vienna, Vienna, Austria; Bioinformatics Group, Interdisciplinary Center of Bioinformatics, University of Leipzig, Leipzig, Germany*
WEN HE • *Pfizer Worldwide Research and Development, Cambridge, MA, USA*
MANUELA HELMER-CITTERICH • *Centre for Molecular Bioinformatics, Department of Biology, University of Rome Tor Vergata, Rome, Italy*
DAHUI HU • *Department of Bioinformatics, The State Key Laboratory of Plant Physiology and Biochemistry, College of Life Sciences, Zhejiang University, Hangzhou, China*
YING HUANG • *Pfizer Worldwide Research and Development, Cambridge, MA, USA*
DANNY INCARNATO • *Department of Molecular Genetics, Groningen Biomolecular Sciences and Biotechnology Institute (GBB), University of Groningen, Groningen, The Netherlands*
MAXIMILIAN KRAUSE • *Computational Biology Unit, Department of Informatics, University of Bergen, Bergen, Norway; Sars International Centre for Marine Molecular Biology, University of Bergen, Bergen, Norway*
ADRIEN LEGER • *European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, Cambridge, UK*
TOMMASO LEONARDI • *Center for Genomic Science, Istituto Italiano di Tecnologia, Milano, Italy*

QIAN LI • *Health Informatics Institute, University of South Florida, Tampa, FL, USA*
PIETRO LIBRO • *Department of Ecological and Biological Sciences (DEB), University of Tuscia, Viterbo, Italy*
NICOLA LICHERI • *Department of Computer Science, University of Turin, Turin, Italy*
GEORGE C. LINDERMAN • *Department of Applied Mathematics, Yale University, New Haven, CT, USA*
HUI LIU • *School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, Jiangsu, China*
MICHAEL I. LOVE • *Department of Biostatistics, University of North Carolina-Chapel Hill, Chapel Hill, NC, USA; Department of Genetics, University of North Carolina-Chapel Hill, Chapel Hill, NC, USA*
JIANI MA • *School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, Jiangsu, China*
LUIGI MANSI • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy*
TYCHO MARINUS • *Department of Molecular Genetics, Groningen Biomolecular Sciences and Biotechnology Institute (GBB), University of Groningen, Groningen, The Netherlands*
PAOLO MARTINI • *Department of Biology, University of Padova, Padova, Italy; Department of Molecular and Translational Medicine, University of Brescia, Brescia, Italy*
JIA MENG • *Department of Biological Sciences, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China; Institute of Integrative Biology, University of Liverpool, Liverpool, UK*
REIKO NAKASHIMA • *Pfizer Worldwide Research and Development, Cambridge, MA, USA*
ADNAN M. NIAZI • *Computational Biology Unit, Department of Informatics, University of Bergen, Bergen, Norway*
MIHO OKA • *Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa-shi, Chiba, Japan; Ono Pharmaceutical Co., Ltd., Osaka, Japan*
FRANCESCO PANARIELLO • *Telethon Institute of Genetics and Medicine (TIGEM), Armenise/Harvard Laboratory of Integrative Genomics, Pozzuoli, Italy*
GIULIO PAVESI • *Dipartimento di Bioscienze, Università di Milano, Milan, Italy*
MATTIA PELIZZOLA • *Center for Genomic Science of IIT@SEMM, Fondazione Istituto Italiano di Tecnologia (IIT), Milan, Italy*
GRAZIANO PESOLE • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy; Institute of Biomembranes and Bioenergetics, National Research Council, Bari, Italy; National Institute of Biostructures and Biosystems (INBB), Rome, Italy; Consorzio Interuniversitario Biotecnologie, Trieste, Italy*
ERNESTO PICARDI • *Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy; Institute of Biomembranes and Bioenergetics, National Research Council, Bari, Italy; National Institute of Biostructures and Biosystems (INBB), Rome, Italy; Consorzio Interuniversitario Biotecnologie, Trieste, Italy*
MARCO PIETROSANTO • *Centre for Molecular Bioinformatics, Department of Biology, University of Rome Tor Vergata, Rome, Italy*
YANN PONTY • *LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France*
VLADIMIR REINHARZ • *Department of Computer Science, Université du Québec à Montréal, Montréal, QC, Canada*
DAVIDE RISSO • *Department of Statistical Sciences, University of Padova, Padova, Italy*
FRANCESCO RUSSO • *Danish Center for Newborn Screening, Department of Congenital Disorders, Clinical Metabolomics Lab, Statens Serum Institut, Copenhagen, Denmark*

Gabriele Sales • *Department of Biology, University of Padova, Padova, Italy*
Roman Sarrazin-Gendron • *School of Computer Science, McGill University, Montréal, QC, Canada*
Ester Saus • *Barcelona Supercomputing Centre (BSC-CNS), Barcelona, Spain; Institute for Research in Biomedicine (IRB), The Barcelona Institute of Science and Technology, Barcelona, Spain*
Masahide Seki • *Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa-shi, Chiba, Japan*
Shaked Slovin • *Telethon Institute of Genetics and Medicine (TIGEM), Armenise/Harvard Laboratory of Integrative Genomics, Pozzuoli, Italy*
Nicola Soranzo • *Earlham Institute, Norwich, UK*
Ayako Suzuki • *Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa-shi, Chiba, Japan*
Yutaka Suzuki • *Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa-shi, Chiba, Japan*
Eivind Valen • *Computational Biology Unit, Department of Informatics, University of Bergen, Bergen, Norway; Sars International Centre for Marine Molecular Biology, University of Bergen, Bergen, Norway*
Marius van den Beek • *Department of Biochemistry and Molecular Biology, Penn State University, University Park, PA, USA*
Jérôme Waldispühl • *School of Computer Science, McGill University, Montréal, QC, Canada*
Xuefeng Wang • *Department of Biostatistics and Bioinformatics, H. Lee Moffitt Cancer Center and Research Institute, Tampa, FL, USA*
Sebastian Will • *LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France; Bioinformatics Group, Interdisciplinary Center of Bioinformatics, University of Leipzig, Leipzig, Germany*
Jesse R. Willis • *Barcelona Supercomputing Centre (BSC-CNS), Barcelona, Spain; Institute for Research in Biomedicine (IRB), The Barcelona Institute of Science and Technology, Barcelona, Spain*
Liu Xu • *Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa-shi, Chiba, Japan*
Qingru Xu • *Department of Biological Sciences, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China*
Hua-Ting Yao • *LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France, School of Computer Science, McGill University, Montreal, QC, Canada*
Federico Zambelli • *Dipartimento di Bioscienze, Università di Milano, Milan, Italy*
Lin Zhang • *School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, Jiangsu, China*
Peijing Zhang • *Department of Bioinformatics, The State Key Laboratory of Plant Physiology and Biochemistry, College of Life Sciences, Zhejiang University, Hangzhou, China*
Ying Zhang • *Pfizer Worldwide Research and Development, Cambridge, MA, USA*
Shanrong Zhao • *Pfizer Worldwide Research and Development, Cambridge, MA, USA*

Chapter 1

Advanced Design of Structural RNAs Using RNARedPrint

Yann Ponty, Stefan Hammer, Hua-Ting Yao, and Sebastian Will

Abstract

RNA design addresses the need to build novel RNAs, e.g., for biotechnological applications in synthetic biology, equipped with desired functional properties. This chapter describes how to use the software RNARedPrint for the de novo rational design of RNA sequences adopting one or several desired secondary structures. Depending on the application, these structures could represent alternate configurations or kinetic pathways. The software makes such design convenient and sufficiently fast for practical routine, where it even overcomes notorious problems in the application of RNA design, e.g., it maintains realistic GC content.

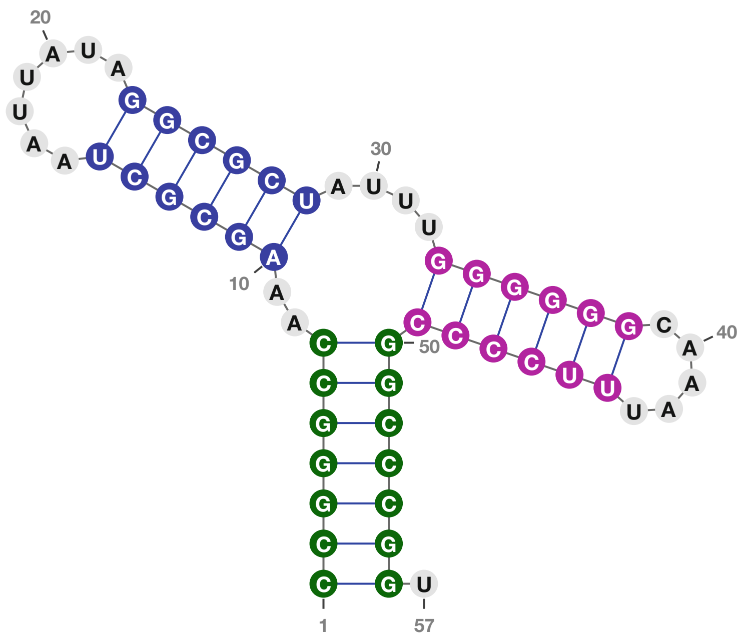
Key words RNA design, Kinetic landscapes, Riboswitches

1 Introduction

RNA design targets a wide diversity of biological functions and, as such, encompasses of a wide array of computational tasks. Two dominant paradigms dominate current computational approaches:

– **Negative design** focuses on the **specificity** of produced sequences for "design targets." This can comprise the attempt to avoid functions, interactions, structures, or other properties that differ from the targeted ones. Such tasks correspond to inverse combinatorial problems and can be computationally intractable (NP-hard) even when their direct version can be optimized in polynomial time [1].

In the context of structural RNA design, negative design is usually referred to as the **inverse folding** problem [9] and consists in producing nucleotide sequences that uniquely fold into the target structure with respect to the minimum free energy (MFE) criterion. Variants of the inverse folding problem consider the minimization of various notions of **defects** [5], notably including the **ensemble defect** [12], the expected base



```
CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU
(((((((..((((((.......))))))....((((((.....))))))))))))).
```

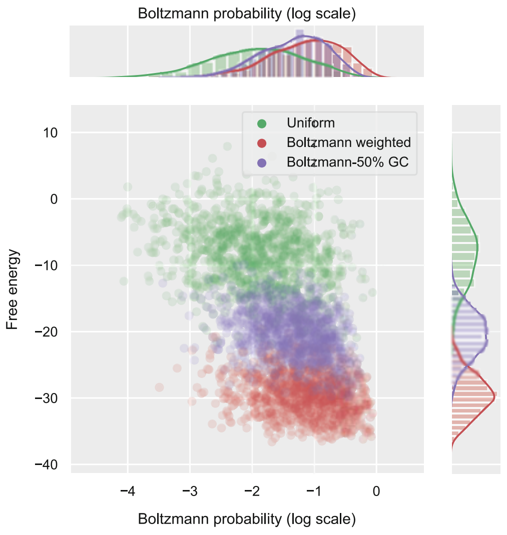
Fig. 1 The target RNA structure of our running example for single-target design, together with the finally designed sequence. We show its representation as as 2D plot (top, rendered using VARNA [3]) and dot-bracket string (below). The latter represents base pairs by balanced parentheses

pair distance between the target, and a random structure in the Boltzmann-Gibbs distribution.

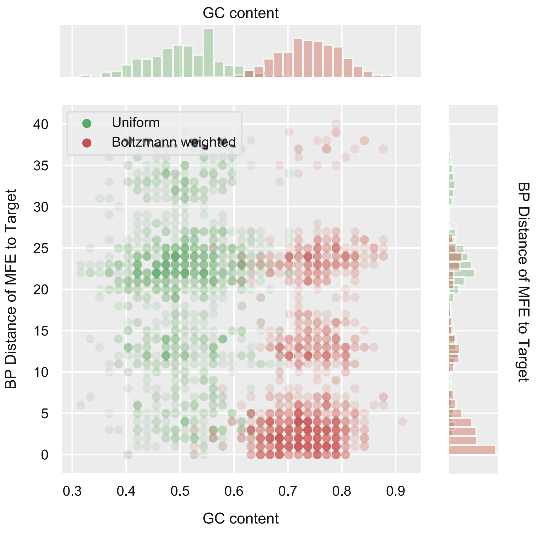
An example of RNA design for a single-target structure is provided in Fig. 1, showing the results from our first running example of the Methods section. As natural extension of single-structure design, we will moreover discuss RNA design for multiple structural targets (cf. Fig. 3).

– **Positive design** can be loosely defined as focusing on the **propensity** of produced RNAs to achieve a certain function. In a structural context, positive design usually involves generating one or several sequences having good affinity (i.e., high stability $\approx$ low free energy) for a targeted structure. Functionally, design constraints will also include the presence/absence of sequences motifs, a controlled affinity toward interactions with molecules, or a control of composition biases, such as the GC-content [10].

  Recently, the objectives of positive design have been extended to include sampling of sequences in a controlled distribution induced by the design objective [7, 8, 10]. Interestingly positive design approaches implementing a controlled sampling strategy can be used to empirically tackle negative design objectives, by coupling a random generation of



a) Boltzmann probability *vs* Free-energy of target



b) GC content *vs* distance of MF to target structure.

Fig. 2 (a) Boltzmann probability vs free energy of target (b) GC content vs distance of MFE to target structure. Negative design can be addressed by positive design in Boltzmann-weighted distributions on sequences. Distribution of Boltzmann probability, free energy of target (**a**), GC content, and distance of MFE to target structure (**b**). For a target structure (((((((...(((((.......))))).....(((((.......)))))))))))))., 1000 sequences were generated, either uniformly or in the Boltzmann distribution

sequences, filtered to only retain good candidates for the negative design. Indeed, as shown in Fig. 2, negative design objectives, such that the Boltzmann probability of the target structure, or the distance of the MFE to the target, tend to correlate well with positive design objectives, such as the free energy of the target structure.

1.1 Applications of RNA Design

From a molecular biology perspective, designing RNAs represents the **ultimate stress-test** of our understanding of how RNA folds and acts on its environment. In this setting, one designs RNA sequences expected to fold into a predefined structure with respect to a folding prediction model. Synthesizing the resulting sequences, and using experimental methods to verify the actual adoption of the desired structure, one either validates the model or reveals some of its flaws, fueling and prioritizing further developments. Indeed, misfolding designed RNAs reveal gaps in our energy models and descriptors of the conformation spaces used by predictive algorithms. A similar strategy can be more generally used to test functional hypotheses involving the structure of RNA.

Molecular design also represents one of the primitives of **synthetic biology**, and RNAs have been used in multiple roles, notoriously including biosensors [6], regulators, and nanomaterials. Some naturally occurring RNAs are sufficiently stable to fold in a modular fashion, enabling a *copy/paste approach* that simply combines existing RNAs into complete architectures. However, such a strategy is hindered, in an in vivo context, by the competition of artificial and endogenous RNAs and by the intrinsic difficulty to produce orthogonal constructs over a limited number of available architectures. A rational design, coupled with an experimental filtering phase, is thus likely to represent the method of choice for future endeavors in RNA-based synthetic biology.

At a (primarily) sequence-based level, design is essential for future developments of **RNA-based therapeutics**. For instance, the recent discovery of viable treatments based on RNA interference is fueled by an understanding of how small RNAs can interfere with selected messenger RNAs to activate or inhibit them. In this context, an optimization of the nucleotide sequence of small interfering RNAs, akin to a design task, is crucial to ensure its efficacy and selectivity, mitigating the risk of side effects for the drugs. Similarly, the specificity of genetic contents targeted by CRISPR-based editing, and thus its limited toxicity, is ensured by a redesign of guide RNAs. More generally, the adoption of a stable structure is very often a prerequisite for the interaction of RNAs with selected proteins [4] and is therefore crucial for the functionality of designed RNAs in a cellular context.

Design also helps in the **search for homologous RNAs**. Indeed, in many RNA families, selective pressure mostly applies at the structure level. This aggravates the discovery of new occurrences of given RNA genes within the same organism (paralogs) or across available genomes (orthologs). If a structural model is known, and if the number of identified homologs is limited, a natural approach is to enrich homologs with sequences designed as to fold into the shared structure, in order to cover a larger proportion of the (neutral) sequence space.

1.2 Overview

In this chapter, we describe how to install and apply RNARedPrint [8] to perform positive and negative RNA design. The method was designed to simultaneously account for the constraints induced by multiple RNA secondary structures. Its core capability is to generate sequences that achieve predefined thermodynamic stabilities for the target structures while controlling the GC content. Consequently, we start by discussing the design for single RNA structures and go on to the design of RNAs that fold into multiple structures. We demonstrate how to take advantage of RNARedPrint's versatility, for tackling a large variety of RNA design tasks.

2 Material

2.1 Installing RNARedPrint and Its Dependencies

RNARedPrint can be conveniently installed using the package manager Conda For Linux or macOS systems, we highly recommend this way of installing the software and provide detailed instructions below. For other systems or other types of installation, it is possible to install directly from the source code at https://github.com/yannponty/RNARedPrint following the provided instructions. In this chapter, we describe the release 0.3 of the software.

2.1.1 Package Manager Conda Installation

You may skip this section if Conda is already installed and set. Otherwise, one can install Conda by installing Miniconda from https://conda.io/en/latest/miniconda.html. This could also be done from command line for Linux users

```
wget \
https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
sh Miniconda3-latest-Linux-x86\_64.sh
```

or if you have macOS, use

```
wget \
https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh
sh Miniconda3-latest-MacOSX-x86_64.sh
```

It is moreover convenient to set up several Conda channels (i.e., repositories for Conda packages), by

```
conda config --add channels defaults
conda config --add channels conda-forge
conda config --add channels bioconda
```

We require the bioconda channel, a Conda channel dedicated to bioinformatics software, which contains the RNARedPrint package.

2.1.2 RNARedPrint Installation

After following the instructions above, RNARedPrint can be installed using the command

```
conda install rnaredprint
```

This will install the tool `RNARedPrint` and three complementary Python scripts, `design-energyshift.py`, `design-multistate.py`, and `calcprobs.py`. Make sure to activate the Conda environment before calling the programs:

```
conda activate
```

3 Methods

3.1 General Usage

For typical advanced design tasks, our software package provides easy-to-use Python scripts as high-level front ends to the computational engine of the software, implemented in the program `RNARedPrint`. It implements the package's core functionality of sampling RNA sequences from a specific distribution controlled by multiple targets.

The script `design-energyshift.py` generates RNA sequences with highly specific GC content and energies for the given secondary structures. For this purpose, it implements a multidimensional Boltzmann sampling strategy on top of `RNARedPrint`. Our second script `desing-multistate.py` samples start structures for further optimization according to negative design objectives like probability or ensemble defect. Such optimization can be performed, e.g., using the classical tool RNAinverse [9] (included in the already installed Vienna RNA package) or the recent software RNABluePrint [7]. Only the latter is equipped to handle the general and more complex case of multi-target design. To directly select sequences with high probabilities of the target sequences, a typical negative design criterion, we provide the script `calcprobs.py`. In the following, we provide concrete examples of the usage of all these tools in the two scenarios: first, the design for a single-target structure, and second, the design for multiple structural targets. All these tools can be invoked from the command line of a terminal. The command line interface lets the user flexibly control the various options of the tools and, for advanced usage, enables integrating them into larger workflows.

3.1.1 The Tool *`RNARedPrint`*

This workhorse of our software supports generating sequences from a multidimensional Boltzmann distribution controlled by weights of the GC content and for the energies of one or several RNA secondary structures.

Our high-level scripts that can be used to address typical design scenarios rely on this tool performing the main computation. Here, the tool serves us as quick test of the installation. Make sure that `RNARedPrint` is installed and found in your search path by running

```
RNARedPrint --version
```

If the software was installed successfully (if you install via Conda as described above, do not forget to activate Conda), it reports its version in the form

```
RNARedPrint 0.3
```

Generally, the tools of the package provide usage information via the command line argument –help. Please have a look at the output of the command

```
RNARedPrint --help
```

to get a brief, complete overview of the usage of the tool. We will demonstrate the main usage in the subsequent sections.

3.2 Designing for a Single-Target Structure

The most common case of RNA sequence design asks for sequences that fold well into a single RNA structure. While RNARedPrint is as well prepared to handle the more challenging task of multi-structure design, we will start with this simpler case to introduce the software. Thus, let us demonstrate how to design RNA sequences for our example target structure illustrated in Fig. 1.

Target secondary structures are given as a "dot-bracket string," where each pair of balanced parentheses describes a base pair. This is the typical secondary structure linearization prominently popularized by the Vienna RNA package. We will use this representation throughout our explications and in the input and output of our design tools.

A popular approach to RNA sequence design is to efficiently generate start sequences, also called seeds, that are subsequently optimized using local search strategies. Our software allows to design "good" seed sequences, where we have strong control over the GC content of the sequences and their energies for the target structure. Recall that the design of such start sequences is a case of *positive design*.

3.2.1 Positive Design for a Single-Target Structure with RNARedPrint

Let us start by calling RNARedPrint from the command line (again, from the installation directory) as

```
RNARedPrint --num 4 --weights 5 --gcw 0.5 \
  "((((((...(((((.......))))).....(((((.......)))))))))))."
```

With these arguments, we ask the tool to sample four sequences S (--num) with probabilities proportional to

$$5^{-E_{\mathrm{bp}}(S)} \cdot 0.5^{\#GC(S)}, \tag{1.1}$$

where $E_{bp}(S)$ denotes the energy of the sequence and the target structure in the simple base pair energy model (cf. Sect. 4.0.7) and $\#GC(S)$ denotes the number of bases G and C in the sequence. Note how the weights are set by options –gcw and –weights and affect the distribution due to Equation 1. Without such options, the tool would use default weights of 1.

A typical example of the output produced by the tool is

```
CGUCCACGAUACCCGUACGCUAUGGGUGGUGUGCCCCUCAAAAUGGGGCGUGGAUGA GC=0.59 E1=-22.87
CCGCGUCUGUAGCCCCGAUAACGGGUUUAUACAGGUUAUAUUAUGGCUUGACGUGGC GC=0.51 E1=-19.22
GUUGGUUUUGGUCUACGAAGCGUAGACAGGUACCCCAUAUGUAGUGGGGAACCAAUA GC=0.48 E1=-16.52
GGGGGCGAACGUCCAUAUCGGAUGGACUAUUUGCCCGUCUUAAGCGGGCCGCCUUUU GC=0.57 E1=-23.11
```

By changing the weights, the distributions and thereby the means of the energy and the GC content can be changed. Increasing the energy weight causes the generation of sequences with better energy. In tendency, due to RNA thermodynamics, these sequences have higher GC content. At the same time, lowering the GC-weight (–`gcw`) allows counteracting and keeping the GC content controlled. To observe this effect, e.g., compare the outputs of the following two calls

```
RNARedPrint --num 10 --weights 20 --gcw 0.5 \
  "(((((((...(((((.......))))).....(((((.......)))))))))))))."
```

and

```
RNARedPrint --num 10 --weights 20 --gcw 0.2 \
  "(((((((...(((((.......))))).....(((((.......)))))))))))))."
```

3.2.2 Targeting Specific GC Content and Free Energy

Playing around with different weights for `RNARedPrint` in an attempt to target specific (mean) GC content or specific energies quickly reveals that this is not at all trivial, since the different targets are not independent. For this purpose, we provide the script `design-energyshift.py`, which solves the optimization problem of finding weights, such that specific energies and GC content are targeted by multidimensional Boltzmann sampling.

Notably, the script supports targeting specific energies in the accurate *Turner energy model* [11], whereas the tool `RNARedPrint` samples based on energies in a simple model (by default, the base pair energy model).

This script allows specifying targets, e.g., GC content of 60 percent and (Turner) energy of −28 kcal/mol, for the designed sequences in a call like

```
design-energyshift.py --num 4 --gc 0.6 -e=-28 \
  <<<"(((((((...(((((.......))))).....(((((.......)))))))))))))."
```

By this call, the script produces output similar to

```
CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU GC=0.59 E1=-28.70
CGCCAGCUGCCCUCUAUACAUUAGAGGAAUUUGCGCGUAUCCGUCGCGCGCUGGCGA GC=0.59 E1=-27.00
GGCCCGUUCUGCGCCGAUAUUAGGCGCUCACACGCCGUGUUAGUCGGUGAUGGGCCA GC=0.62 E1=-27.70
CCGCCACAUAGCGCUCGUGAAUAGCGCAAGGAGCGCCAUCAAAAGGCGCGUGGCGGA GC=0.62 E1=-28.90
```

3.2.3 Negative Design for a Single Target

The RNA sequences generated by the script `design-energy-shift.py`, while targeting low energies, are typically good designs for the target structure even according to negative design criteria like MFE or even probability and ensemble defect (cf. Fig. 2). To quickly test this for concrete examples, one can fold the designs using `RNAfold`. Let us demonstrate this for the first of the above designs.

```
RNAfold \
  <<<CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU
```

RNAfold reports the MFE structure together with additional information about the RNA structure ensemble.

```
CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU
(((((((..((((((.......))))))....((((((.....))))))))))))). (-30.80)
```

We observe that the distance between MFE and target structure (i.e., the MFE defect) is only two base pairs; the structures look almost identical. The next design even yields optimal MFE defect of zero. This suggests that negative design can be performed rather effectively by screening through a series of (positive) designs by RNARedPrint. Even more effective is the use of such designs of start structures in a local optimization according to negative design objectives. Such optimization is possible using tools like RNAinverse or RNABluePrint.

To apply the former, we feed it with our target structure and, e.g., our first designed sequence

```
echo -e \
  "(((((((...(((((.......))))).....(((((.......)))))))))))).\n\
  CCGGGCCAAAGCGCUAAUUAUAGGCGCUAUUUGGGGGGCAAAUUUCCCCGGCCCGGU"\
  | RNAinverse -Fp
```

The given option causes RNAinverse to optimize the probability of the target structure. On this input, RNAinverse succeeds quickly, returning a very good design for the target structure.

```
UUAGAUCAAAUAGGAGUCGAUGUCCUGGGGGUACGUGAAACAAGCACGUGAUUUAGU   23
GGCCGGCGAAGGGGCGAAUAAAGCCCCAAAAACCCGCGAAAAAAGCGGGGCCGGCCA   47  (0.989619)
```

The finally designed sequence (last row) forms the target structure with a probability of 0.99. Notably, the GC content of the design—which used to be hard to control by classic design methods—is fairly close to the start sequence.

For harder designs than our running example, these observations suggest an automated two-step approach to negative design. First, a set of start sequences is systematically generated using positive design. Second, these sequences are optimized, e.g., using RNAinverse, as shown above. Similar strategies have been discussed in the literature, e.g., by the early approach INFO-RNA [2] or using the tool IncARNation [10]. The latter is implementing the exact the same ideas of Boltzmann sampling for positive RNA design, including the control of GC content. Due to our software RNARedPrint, which easily covers single-structure design as a special case, this approach to single-target design got even more accessible and flexible.

3.3 Multi-target Design

3.3.1 Positive Design by RNARedPrint

Let us demonstrate the use of RNARedPrint for multi-target design by running the tool for a small instance (sequence length 40) with three target structures. As preparation, we create the file `targets.txt` containing the target structures (one per line)

```
((((((((....))))))))((((((((....))))))))
((((((((..((((((((...))))))))...))))))))
(((((((((((((((((...)))))))))))))))))...
```

Then, we call the tool by

```
RNARedPrint --weights 1,2,5 --gcw 0.5 --num 5 $(cat targets.txt)
```

This call specifies three target secondary structures and asks for five sequences that have the same length as the structures. The call sets the weights for the three target structures (respectively 1, 2, and 5) and the weight for the GC content. The tool automatically seeds its random number generator, such that repeated calls produce different output. The output should look like

```
((((((((....))))))))((((((((....))))))))
((((((((..((((((((...))))))))...))))))))
(((((((((((((((((...)))))))))))))))))...
GGAGGGGGCCCCCUCCUUUUGGGGGGGGGCCCCUUCCUUC GC=0.73 E1=-18.29 E2=-21.07 E3=-28.83
GAGGGGGGCUCCCCCUCUUUGGGGGGGGGCCCCCUUUUC GC=0.73 E1=-19.51 E2=-22.29 E3=-28.49
GGGGGGGGCUCCCCCUCUCCGGGGGGGGGCCCCCCCUCC GC=0.86 E1=-25.96 E2=-26.30 E3=-31.29
GGGGGGGGUCCUUCCCUUUCGGGGGGGGGUCCCCCCCCU GC=0.77 E1=-23.86 E2=-23.86 E3=-27.98
GGGGGGGGCCCCCCCCCCUUGGGGGGGGGCCUCCCCCCUU GC=0.84 E1=-24.74 E2=-27.18 E3=-32.51
```

As in the single-target case, decreasing (or increasing) the weight '–`gcw`' will produce sequences with, in tendency, lower (or, respectively, higher) GC content. Similarly, decreasing (or increasing) the weight of some secondary structure generates sequences with higher (or lower, i.e., better) energy. Thus, changing the weights allows shifting the distribution means of the different features of the generated sequences (i.e., the GC content and energies for the different structures).

3.3.2 Targeting Multiple Specific Energies and GC Content

Targeting very specific values of several features again requires the support of advanced automatized methods—here, this is even more important than for single-target design. Again, the corresponding hard optimization problem is solved performing multidimensional Boltzmann sampling by the script `design-energyshift.py`.

In generalization of the single-target case, the script allows generating sequences with sharply defined values of the multiple targeted features. For example, we can ask for sequences with respective Turner energies of -18, -22, and -20 for the three previously introduced target structures. At the same time, we want aim at GC contents of about 65 percent.

To obtain five such sequences, one calls the script as

```
design-energyshift.py -i targets.txt --num 5 \
  --gc 0.65 --energies="-18,-22,-20"
```

which produces five designs like

```
GGGGGGGGUUCCUCUCUCUUGGAGGGGGGUUUUCUCCUCC GC=0.64 E1=-17.30 E2=-22.20 E3=-19.40
GGGGGGGGCUUCCUUUCCUUGGGGGGGGGUUUUCCCUCCC GC=0.68 E1=-19.00 E2=-21.90 E3=-20.40
GGGGGGGAUCCCUUCUCCUUGGGGGGGGGUUUUCUUCCCC GC=0.66 E1=-17.60 E2=-22.30 E3=-20.50
GGGAGGGGUUCCCUUCUCUUGGGGGGGGGUCUUUCCUCCC GC=0.66 E1=-17.20 E2=-22.60 E3=-19.70
GGGGGGAGUUCCCUUUCCUUGGGGGGGGGCUUCUUUCCCC GC=0.66 E1=-17.30 E2=-22.10 E3=-20.60
```

By default, the script tolerates deviations from the targets by as much as 1 kcal/mol. We can define the tolerance, e.g., more strictly as ± 0.5 kcal/mol using the additional argument `–tolerance 0.5`. Similarly, the tolerance of the GC content can be selected, and the operation of the tool can be fine-tuned in various other ways. When called with option `–help`, the script provides a full overview of available arguments.

3.3.3 Aiming at Negative Design Criteria

As we already discussed before, a major motivation for positive design is to produce sequences that perform well according to negative design criteria. From a sample of such sequences, one can either already satisfy negative design requirements or find good start sequences for further refinement, e.g., by stochastic local search.

A typical requirement for good designs in the case of single-target design can be expressed in slight simplification as "good stability, avoiding extreme GC contents." In the multi-structure case, we additionally aim at specific relations between the energies. This is possible with our tool `design-energyshift.py` by targeting energies for each structure.

The script `design-multistate.py` was written to design sequences, where all structures have highly similar energies, which is a common objective in RNA design. Let us demonstrate this for the previously introduced three target structures (in file `targets.txt`).

To produce five designs targeting similar energy for all three targets, while controlling the GC content at about 0.65, the script is called as

```
design-multistate.py -i targets.txt --num 5 --gc 0.65
```

It returns the designs together with their GC content and Turner energies:

```
GGGGGGGGUCUUCCCUUCUUAGGGGGGGGUUCUCCCCCCU GC=0.68 E1=-24.20 E2=-22.90 E3=-24.00
AGGGGGGGUUCUCCCCUCUUAGGGGGGAGUUUUCCCUCUU GC=0.59 E1=-22.30 E2=-23.80 E3=-22.70
GGGGGGGGUCUCCCUUUCUCGGAGGGGGGUUUCUCCCUCC GC=0.68 E1=-22.80 E2=-22.40 E3=-22.30
GGGGGGGAUUUCUCCCUCUCGGGGGGGGAUUUUCCUCUCC GC=0.64 E1=-23.10 E2=-23.60 E3=-22.70
GGGGGGGAUUCUUCCCUUCUAGGGGGGAGUUUUUCCCCCU GC=0.59 E1=-23.70 E2=-23.10 E3=-23.00
```

Similar to the script `design-energyshift.py`, further options provide additional control over the script operation, and an overview of arguments can be produced by running the script with the `–help` option.

3.3.4 Negative Design Due to Positive Design

Sequences as designed above can be used as start sequences for refinement due to negative design criteria, which cannot be directly targeted by the sampling engine.

Refinement by local optimization can be performed using third-party software. For the case of single-target design, we demonstrated the use of RNAinverse, as the “classic” tool for this purpose. However, different to the single-target case, corresponding software for multi-structure design is rare. One good choice would be the design software RNABluePrint.

In many cases, good solutions to negative design criteria can be found by screening a larger number of positive designs. This mainly requires us to re-evaluate sequences as generated by using our scripts according to negative criteria. We demonstrate this for the criterion of accumulated probability of the target structures. Figure 3 shows good negative designs for our three target sequences (as well as, for comparison, one single-target design) that we obtained following this strategy.

For the purpose of re-evaluating the positive designs, we provide the script `calcprobs.py`. It can be used to post-process the output of our design scripts. For each sequence in the output, it annotates the corresponding output line. For example, when reading the lines

```
AGGGGGAGUUUCCUCCCCCUGGGGGGGGACUUCUCCUCCU GC=0.66 E1=-26.90 E2=-27.10 E3=-27.10
AGGGGGGGUUUCCUCUCCCUGGGAGGGGACUUCCCCUCCU GC=0.66 E1=-26.90 E2=-27.10 E3=-27.20
```

the script calculates the minimum free energy (MFE) and the ensemble energy of the sequences (EE). Then it computes the probabilities of each target for the sequences (Pi) and their sum (Psum). Consequently, it extends the lines to report this information by the respective strings

```
MFE=-27.10 EE=-28.05 P1=0.16 P2=0.21 P3=0.21 Psum=0.58
MFE=-27.20 EE=-28.05 P1=0.15 P2=0.21 P3=0.25 Psum=0.62
```

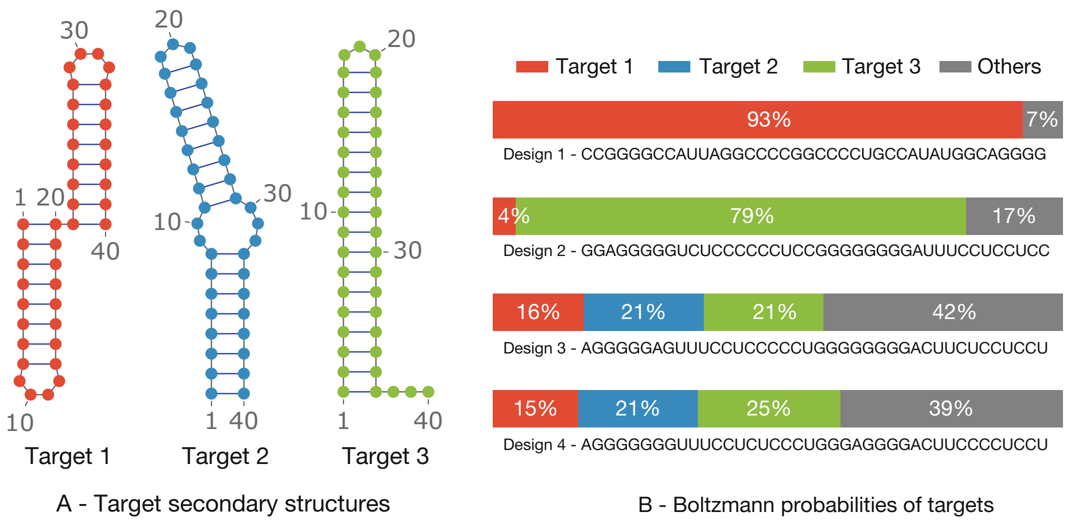


Fig. 3 Negative design for three target structures. For three target structures, inducing disjoint sets of base pairs (A – left), sequences are designed as to simultaneously optimize the Boltzmann probabilities (B – right) of the first (Design 1), third (Design 2), and second (Designs 3 and 4) target structures. For the two latter sequences, the probabilities of the three targets were designed to be comparable, e.g., in order to address the need for a switching behavior at the thermodynamic equilibrium

The above two lines correspond to the two last designs of Fig. 3. They were obtained as best negative designs, according to `Psum`, after generating and annotating 1000 samples by

```
design-multistate.py -i targets.txt --num 1000 \
  --gc 0.7 --energy="-27" --tolerance=0.2 \
  | calcprobs.py  -i targets.txt
```

where `calcprobs.py` is used in a pipeline with the design script.

4 Notes

4.1 Boltzmann Distributions as Generated by RNARedPrint

The fundamental computational task in the RNARedPrint is the generation of Boltzmann samples of sequences. This allows controlling the frequencies of feature values, namely, the GC content and the energies of the target structures (in a simple energy model). The distribution of the different features is furthermore controlled by weights for each of the features. Consequently, each sequence is generated with a probability that is proportional to the sum of “feature weight to the power of the feature value” over all features. In Eq. 1, we gave an example for the simple case of a single structure.

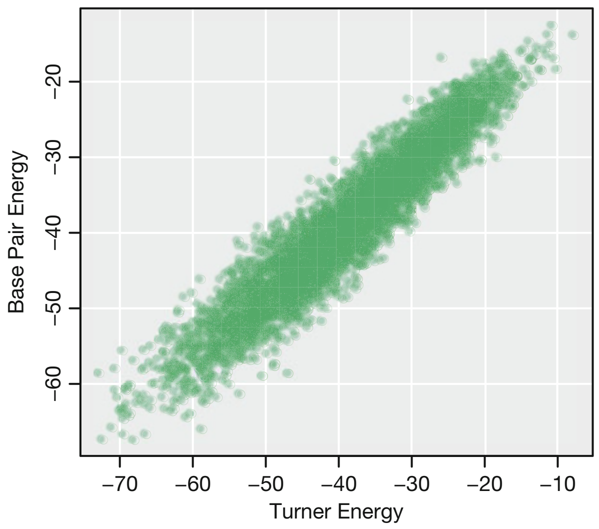


Fig. 4 Strong correlation of 0.95 between energies in the simple base pair model and Turner energies reported by the Vienna RNA package; determined for 5000 uniform random RNA sequences and their MFE structures

4.2 Uniform Sampling of Sequences

Uniform sampling is a special case of Boltzmann sampling. If we set all weights to 1, which is the default of RNARedPrint, then it produces a uniform sample of the sequences. Effectively, this turns off the influence of feature values (since 1^x equals 1, such that the proportionality factor is constant for all sequences).

4.3 Simple Energy Model as Efficient Proxy for the Turner Model

For efficient sampling in RNARedPrint, we utilize simple energy models that approximate the highly accurate nearest neighbor Turner RNA energy model. For this purpose, we trained parameters [8] such that the energies even in the base pair model show good linear correlation to the Turner energies (Fig. 4). The base pair model evaluates energies as a sum of energy terms for each base pair that depend only on the type of the base pair ("AU," "CG," or "GU") and furthermore distinguishes stacked and non-stacked base pairs. For details, see [8], which as well discusses the slightly more complex stacking energy model.

The strong correlation between energies in the two models allows us to much more efficiently target energies in the simple model for the purpose of finally targeting energies in the realistic Turner model. Utilizing the simple model as proxy for the realistic model is indeed a crucial aspect for the viability of our method.

References

1. Bonnet É, Rzażewski P, Sikora F (2018) Designing RNA secondary structures is hard. In: Raphael BJ (ed) Research in computational molecular biology – 22nd annual international conference, RECOMB 2018. Lecture Notes in Computer Science, Paris, vol 10812. Springer, pp 248–250

2. Busch A, Backofen R (2006) INFO-RNA—a fast approach to inverse RNA folding. Bioinformatics 22(15):1823–1831
3. Darty K, Denise A, Ponty Y (2009) VARNA: interactive drawing and editing of the RNA secondary structure. Bioinformatics 25 (15):1974–1975
4. de Groot NS, Armaos A, Graña-Montes R, Alriquet M, Calloni G, Martin Vabulas R, Tartaglia GG (2019) RNA structure drives interaction with proteins. Nat Commun 10(1)
5. Dirks RM, Lin M, Winfree E, Pierce NA (2004) Paradigms for computational nucleic acid design. Nucleic Acids Res 32(4):1392–1403
6. Findeiß S, Etzel M, Will S, Mörl M, Stadler PF (2017) Design of artificial riboswitches as biosensors. Sensors (Basel, Switzerland) 17(9): E1990
7. Hammer S, Tschiatschek B, Flamm C, Hofacker IL, Findeiß S (2017) RNAblueprint: flexible multiple target nucleic acid sequence design. Bioinformatics (Oxford, England) 33:2850–2858
8. Hammer S, Wang W, Will S, Ponty Y (2019) Fixed-parameter tractable sampling for RNA design with multiple target structures. BMC Bioinformatics 20(1):209
9. Hofacker IL, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, Schuster P (1994) Fast folding and comparison of RNA secondary structures. Monatshefte für Chemie/Chemical Monthly 125(2):167–188
10. Reinharz V, Ponty Y, Waldispühl J (2013) A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotide distribution. Bioinformatics 29(13):i308–i315
11. Turner DH, Mathews DH (2010) NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. Nucleic Acids Res 38(Database issue): D280–D282
12. Zadeh JN, Wolfe BR, Pierce NA (2011) Nucleic acid sequence design via efficient ensemble defect optimization. J Comput Chem 32(3):439–452

Chapter 2

Modeling and Predicting RNA Three-Dimensional Structures

Vladimir Reinharz, Roman Sarrazin-Gendron, and Jérôme Waldispühl

Abstract

Modeling the three-dimensional structure of RNAs is a milestone toward better understanding and prediction of nucleic acids molecular functions. Physics-based approaches and molecular dynamics simulations are not tractable on large molecules with all-atom models. To address this issue, coarse-grained models of RNA three-dimensional structures have been developed. In this chapter, we describe a graphical modeling based on the Leontis–Westhof extended base pair classification. This representation of RNA structures enables us to identify highly conserved structural motifs with complex nucleotide interactions in structure databases. We show how to take advantage of this knowledge to quickly predict three-dimensional structures of large RNA molecules and present the RNA-MoIP web server (http://rnamoip.cs.mcgill.ca) that streamlines the computational and visualization processes. Finally, we show recent advances in the prediction of local 3D motifs from sequence data with the BayesPairing software and discuss its impact toward complete 3D structure prediction.

Key words Tertiary structure, RNA motifs, Extended secondary structure, Base pair classification, Modeling, Prediction

1 Introduction

RNAs perform a broad range of catalytic and regulatory functions in cells, which often use the folding properties of these molecules. RNA structures are typically described at two levels of organization. The secondary structure, which enumerates the Watson–Crick and Wobble base pairs that create the backbone of the molecular structure, and the tertiary structure, which indicates the positions of each atom in the molecule.

The study of secondary structures and base pairing properties can reveal fundamental insights into the functional mechanisms of RNAs such as frameshift elements [1] and riboswitches [2]. However, the information provided by this representation is sometimes not sufficient to describe the complexity of inter- and intra- molecular interactions that govern RNA functions. A seminal example is the sarcin-ricin factor-binding loop, which possesses a sophisticated tertiary structure [3].

Classical approaches to predict and analyze three-dimensional molecular structures make use of molecular dynamic simulations [4]. This methodology has the potential to perform calculations on all-atom models, but suffers from high computational complexity. Straightforward applications are thus limited to small molecules (approximately 50 nt.) on a short period of time with very small body motion. Coarse-grained modeling of RNA structures enables us to overcome some of these limitations [5–7], but the practical impact and theoretical horizon of these technologies remain unchanged. In addition, molecular dynamic simulations often require fine-tuning and can be challenging to run and interpret for nonexperts.

Recently, several research groups developed alternate strategies to model and predict RNA three-dimensional structures. One of the most successful approach, applied in the MC-Fold|MC-Sym pipeline [8], RNA2D3D [9], or 3dRNA [10], consists in predicting a secondary structure first, and use it to build a three-dimensional model. Other programs have employed fragment-assembly [11, 12] or conditional random fields techniques [13]. It is worth noting that, as we saw in a previous chapter, comparative modeling techniques can also be applied to RNA molecules if one structural homolog has been already identified [14].

In this chapter, we introduce a versatile model for RNA structures, which enables us to describe essential features and fine-grained details of RNA three-dimensional structures while preserving the complexity of the representation to a minimum. This methodological advance is key to large-scale analysis and to better understanding of critical features of RNA molecular structures. In turn, this knowledge enables us to define new computational techniques to quickly and easily predict three-dimensional structures of large RNA molecules.

First, we describe the base pair classification at the base of this model [15], and present rnaview [16]—a program that annotates automatically RNA three-dimensional structures. Then, we introduce the concept of RNA structural motifs and present recent databases and online resources based on this definition [17, 18]. Finally, show how to use this knowledge to predict RNA three-dimensional structures. The pipeline described in this chapter works in two steps. First, we predict secondary structures with RNAsubopt [19] and expand the prediction to a full base-pairing interaction network with RNA-MoIP [20]. Next, we use this graphical representation to build the three-dimensional structure of the RNA with MC-Sym [8].

2 Material

2.1 Sequence and Structure Data

The Nucleic Acids Database [21] is a repository of experimentally determined three-dimensional RNA structures maintained at Rutgers University, which is available at http://ndbserver.rutgers.edu. This is a specialized version of the Protein Data Bank [22], which is available at http://www.rcsb.org/pdb. The structures are typically stored in files with the extension ".pdb". In particular, the PDB files store the spatial coordinates of each atom in the molecule. Plain sequences can also be downloaded separately under the FASTA format.

In this chapter, we illustrate our methods on the sequence and experimentally determined tertiary structures of tRNA(Cys) of *Archaeoglobus fulgidus* [23] (*see* **Note 1**). The PDB ID of this molecule is 2DU3.

2.2 Automatic Annotation of 3D Structures

The rnaview software is used to identify all base-pairing interactions that represent in a RNA tertiary structure [16]. This program can be downloaded at: http://ndbserver.rutgers.edu/ndbmodule/services/download/rnaview.html and is currently available for Linux, UNIX, SUN, and MAC systems.

2.3 RNA Secondary Structure Prediction Tools

The prediction of RNA secondary structures is performed with the RNAfold program available in the Vienna RNA package [19, 24]. The software suite is available at http://www.tbi.univie.ac.at/RNA/ and can runs under LINUX, UNIX, MAC, and WINDOWS systems. In this chapter, we used the version 2.1.2 of the Vienna RNA package. Web services are also available at http://rna.tbi.univie.ac.at/.

2.4 Insertion of RNA Motifs in Secondary Structure

We perform the insertion of RNA motifs inside predicted RNA secondary structures with the RNA-MoIP software [20] available at http://csb.cs.mcgill.ca/RNAMoIP/. Noticeably, this software requires installing the Gurobi solver (http://www.gurobi.com/), which is free for academic users.

2.5 Building RNA 3D Structures from a RNA Interaction Graphs

We use the MC-Sym software to build tertiary structure from a base-pairing interaction network [8]. MC-Sym is part of the MC-tools package available at: http://www.major.iric.ca/MajorLabEn/MC-Tools.html.

3 Methods

In this section, we will describe how to extract from a three-dimensional structure, the list of all base pairing interactions. Then, we will describe how to use databases of recurrent motifs to predict the tertiary structure of large RNA molecules.

3.1 Classification of Base Pairing Interactions

Watson–Crick (C-G and A-U) and Wobble (G-U) base pairs are the most common type of interactions. They create a scaffold for the tertiary structure. Nonetheless, the analysis of RNA crystal structures revealed a diversity of base pairing interactions that goes far beyond these canonical base pairs. In order to facilitate the description of RNA structures, Leontis and Westhof proposed a complete nomenclature of base pairing interactions [15] that aims to provide a better description of the complexity of the tertiary structure. Key to this model is to introduce a detailed representation of the base of nucleotides, which in turn enables us to define a more sophisticated classification of base-pairing interactions. Thereby, a base is abstracted with a right triangle modeling all edges of the molecules that can be potentially involved in an interaction with other nucleotides (*see* Fig. 1). The three interacting edges are: the Watson–Crick edge, the Hoogsteen edge, and the Sugar edge. The hypotenuse is associated with the Hoogsteen edge, and the vertex created by the Hoogsteen and Sugar edge represents the root of the base.

Base pairing interactions between nucleotides can now occur between any of these edges. In addition, a complete description of these interactions must also specify the relative orientation of the bases (i.e., the glycosidic bond orientation). Hence, we indicate if the bases are oriented in the same or in opposite directions. These configurations are respectively named *trans* and *cis* configurations.

These parameters enable us to define a complete nomenclature of all possible base-pairing interactions between nucleotides. This catalog is shown in Fig. 1 and includes all 12 possible base pairs that are found in RNA structures.

In secondary structure diagrams, base pairs are often represented with specific links. C-G base pairs are represented with two parallel line, A-U base pairs with a single line and G-U base pairs with a circle. Leontis and Westhof also assigned a symbol to each base pairing interaction of their classification. Watson–Crick edges are represented with a circle, Hoogsteen edges with a square and Sugar edges with a triangle. In addition, black symbols represent a *cis* orientation, and white symbols a *trans* orientation. This notation is illustrated at the bottom of each base pair in Fig. 1.

Using this nomenclature, RNA tertiary structures can be decomposed into a network of base-pairing interactions. Unlike classical secondary structures, nucleotides are now no longer restricted to interact with at most one other nucleotide. Instead,

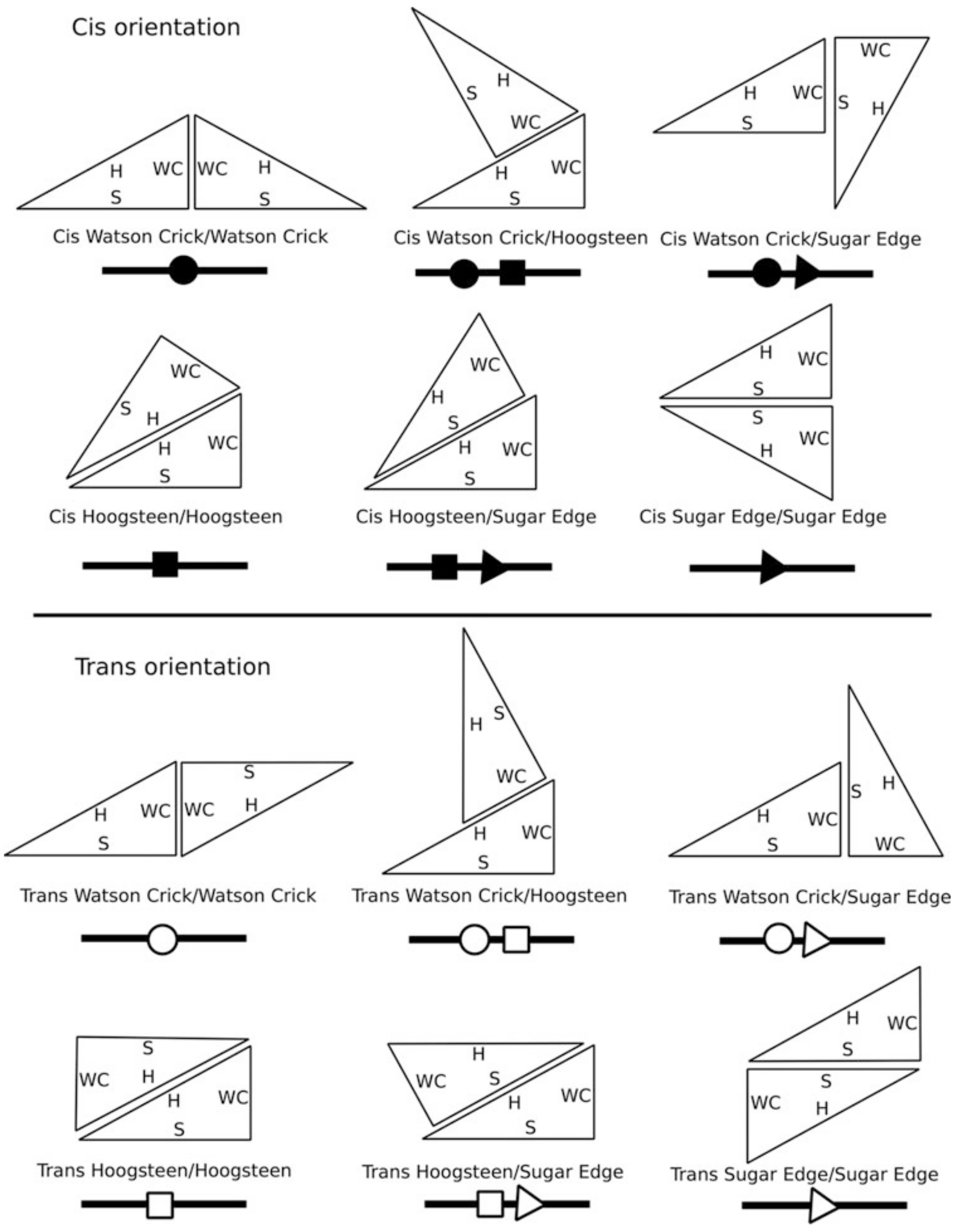

Cis orientation
Cis Watson Crick/Watson Crick
Cis Watson Crick/Hoogsteen
Cis Watson Crick/Sugar Edge
Cis Hoogsteen/Hoogsteen
Cis Hoogsteen/Sugar Edge
Cis Sugar Edge/Sugar Edge
Trans orientation
Trans Watson Crick/Watson Crick
Trans Watson Crick/Hoogsteen
Trans Watson Crick/Sugar Edge
Trans Hoogsteen/Hoogsteen
Trans Hoogsteen/Sugar Edge
Trans Sugar Edge/Sugar Edge
WC
H
S

Fig. 1 Base modeling and Leontis–Westhof base pair classification. The base of a nucleotide is represented with a right triangle. The hypotenuse represents the Hoogsteen edge (noted "H"), and the other sides are associated with the Watson–Crick edge (noted "W") and Sugar edge (noted "S"). This figure represents all 12 bp interactions with *cis* or *trans* orientation

they can be involved in multiple interactions and create, for instance, base triples. Crossing interactions are also permitted. While this modeling does not obviously encompass all details of three-dimensional atomistic structures, it appears to store most of the information that one needs to reconstruct full tertiary structures [8].

3.2 Annotation of Base-Pairing Interactions from RNA 3D Structures

The rnaview program annotates automatically of all base-pairing interactions found in RNA tertiary structure using the Leontis–Westhof classification [15]. It can also produce an image of the interaction graph. The command line to run the program is:

```
rnaview -p input.pdb
```

where "input.pdb" is your input three-dimensional structure and the "-p" flag is an optional parameter that indicates to the program to create a visualization of the annotated structure. Figure 2 shows the graphical output of rnaview on the tRNA (Cys) from *Archaeoglobus fulgidus.*

The output files are stored in the same directory as the input file. The list of base pairing interactions is stored in a new text file named after the input file and appended with the suffix ".out". Similarly, if you used the "-p" option, rnaview also creates a post-script file with a drawing of the base pairing interaction network.

The following code, calculated from the tRNA(Cys) from *Archaeoglobus fulgidus* (2DU3), illustrates a typical output of rnaview:

```
9_12, B: 9 U-G 12 B: S/W tran syn n/a
16_52, B: 16 U-A 52 B: W/H tran XXIV
18_51, B: 18 G-G 51 B: H/S tran n/a
18_53, B: 18 G-C 53 B: +/+ cis syn XIX
```

The first column gives the indices, separated by an underscore, of the two nucleotides involved in the base pair. These indices are calculated by rnaview and correspond to their positions in the sequence(s) entered in the program. They may eventually differ from the indices stored in the PDB file, which are given in the third and fifth columns.

The letters in the second and sixth columns indicate the chain of the two nucleotides. In our case, the RNA molecule has a single chain named "B" in the PDB file. The types of the nucleotides forming the base pair are indicated in the fourth column.

Finally, the seventh and eighth columns describe the edges involved in the base pair (Watson–Crick, Hoogtseen, or Sugar edge), followed by the orientation of their interaction (*cis* or *trans*). For instance, in the example above the first row says that "the nucleotide U at index 9, and the nucleotide G at position 12, form a *trans* base pair between the Sugar edge (S) of nucleotide U and Watson–Crick edge (W) of the nucleotide G". It is worth noting that classical Watson–Crick base pairs are annotated with +/+ (C-G base pair) or −/− (A-U base pair). More details on the base pair notation, including "marginal" interactions, can be found in [16] or [25].

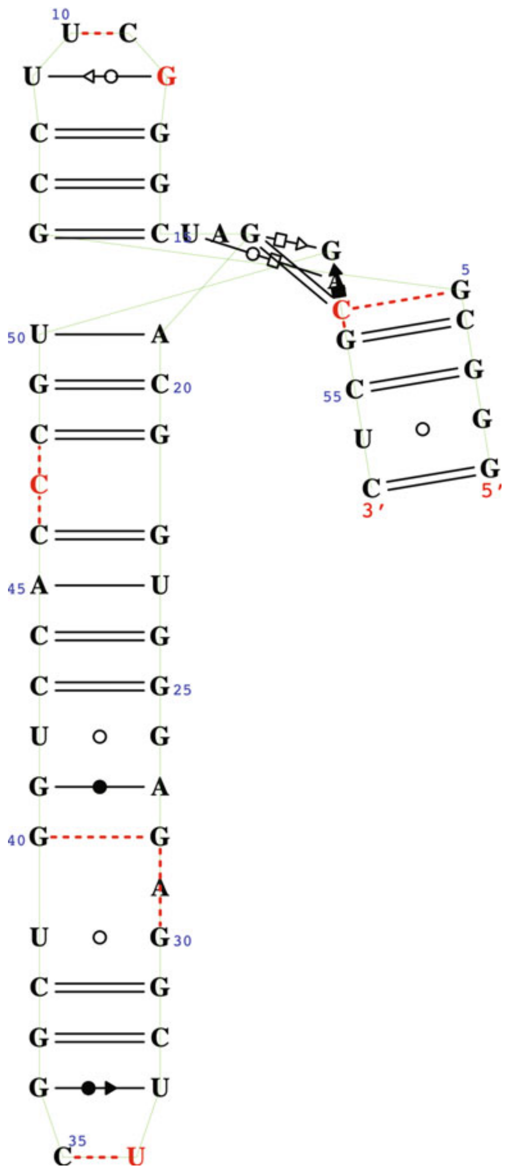


Fig. 2 rnaview annotation of a crystal structure of tRNA(Cys) from *Archaeoglobus fulgidus* [23]

As expected, most base pairs represented in Fig. 2 are Watson–Crick base pairs. Nonetheless, we note that noncanonical base pairs tend to get concentrated in specific regions of the secondary structure backbone and create sophisticated local motifs. In fact, this observation is recurrent in most annotated structures. It will give rise to the notion of RNA motifs introduced in the next section.

Finally, it is worth noting that we can alternatively use the program MC-Annotate [26] to perform similar annotations. However, the classification used by MC-Annotate differs slightly from the one used in the motif databases described further.

3.3 RNA Motifs

The modeling of RNA tertiary structures into networks of base-pairing interactions revealed recurrent motifs conserved across families of structures. These structural patterns form a base toward better understanding of complex organizations of nucleotides inside hairpins, internal loops and beyond (see previous section for a definition of the secondary structure elements). Several methodologies and databases have been developed to mine and store these data. Among the most popular repositories, we count the RNA 3D Hub maintained by the Bowling Green State University RNA group at http://rna.bgsu.edu/rna3dhub/ [18], and the RNA 3D Motif database developed at the university of Paris-Sud and available at http://rna3dmotif.lri.fr/ [17]. More recently, the international institute for molecular and cell biology in Warsaw introduced a novel motif database RNAbricks, available at http://iimcb.genesilico.pl/rnabricks/.

Beside databases of annotated structures (i.e., the RNA Structure Atlas) and recurrent RNA 3D motifs (RNA 3D Motif Atlas), the RNA 3D Hub offers large suite of online tools and web services. In particular, users can retrieve here local 3D motifs with WebFR3D (http://rna.bgsu.edu/main/webapps/webfr3d/) or align RNA tertiary structures with R3D Align (http://rna.bgsu.edu/main/webapps/webr3dalign/).

Each database developed its own local 3D motif format description, based on the Leontis–Westhof base pair classification. RNA-MoIP use the format introduced with RNA3Dmotif [17]. An example of an internal loop motif is provided below:

```
Bases: 26_G 27_A 28_G 29_A 30_G 39_U 40_G 41_G 42_U

     ( 39_U )--- C/C - ---( 40_G )

     ( 30_G )--- 5/5 s ---( 40_G )

     ( 30_G )--- +/+ c ---( 39_U )

     ( 40_G )--- C/C - ---( 41_G )

     ( 29_A )--- C/C - ---( 30_G )

     ( 28_G )--- 5/5 s ---( 41_G )

     ( 28_G )--- C/C - ---( 29_A )

     ( 41_G )--- C/C - ---( 42_U )

     ( 26_G )--- +/+ c ---( 42_U )

     ( 27_A )--- C/C - ---( 28_G )

     ( 26_G )--- C/C - ---( 27_A )
```

The first line starting with the key word "Bases" indicates the nucleotides involved in this motif. For each nucleotide, we display its index and its type separated by an underscore. Each following line describes a base pairing interaction in this motif. The syntax of a base pair is structured as follows. On both ends of the row, the two nucleotides (index and type) are shown between parentheses. Then, the type of the interaction is shown in the middle, surrounded by three dashes on each side. The first field indicates the edges involved and the second the base pair orientation ("c" for *cis*, "t" for *trans*, and "s" for a stacking). An interaction annotated "C/C" stands for a backbone connection (i.e., consecutive nucleotides) and thus may not be considered as a base pair in our discussion. More information can be found at http://rna3dmotif.lri.fr/help.html.

RNA-MoIP needs to decompose these motifs into components to permit their insertion. Components are largest sequences of contiguous nucleotides that belong to a motif. For instance, in the example above, we have two components ([26_G,27_A,28_G,29_A,30_G] and [39_U,40_G,41_G,42_U]). In fact, the definition of components follows the classical secondary structure loop classification. Hairpins have one single component, bulges and internal loops have two component and k-way junctions have k components. This definition has been introduced with RNA-MoIP to facilitate the description of integer programming equations.

Figure 3 illustrates the definition of motifs. In this example, we identify two motifs (a hairpin and an internal loop) inserted into a single stem. The figure shows the 3D structures and the base

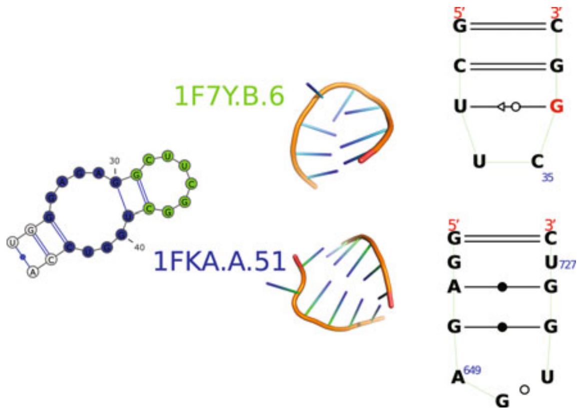


Fig. 3 Example of 3D RNA motif insertions in a secondary structure. In green, we show the position of the hairpin motif "1F7Y.B.6", and in blue we indicate the position of the internal loop motif "1FKA.A.51". On the left side of the motif IDs, we display a 3D structure of the motif together with its base pairing interaction graph

pairing interaction graphs of each motif. We observe that the hairpin has one single component (i.e., one single stranded region), while the internal loop has two.

3.3.1 Prediction of RNA Tertiary Structures from Sequence Data

Modeling RNA tertiary structure is the first step toward predicting RNA 3D structures. We will now describe how the knowledge accumulated in motif databases can be used together with RNA secondary structure prediction methods to predict the structure of large RNA molecules from sequence data only. This strategy presented here works in two steps. First, we predict a secondary structure using classical software such as RNAfold [19] and refine this prediction by inserting RNA 3D motifs and adding noncanonical base pairs inside the predicted secondary structure with RNA-MoIP [20]. Next, we use this extended secondary structure to reconstruct three-dimensional models of the molecule using the MC-Sym software [8].

3.4 Prediction of a Secondary Structure Scaffold

Our first objective is to create a base-pairing network from sequence data. Since the majority of base pairs in RNA structures are *cis* Watson–Crick base pairs, we first predict a secondary structure (without pseudoknots) that will be used to build a scaffold of the interaction graph. Secondary structures (without pseudoknot) can be deterministically predicted with RNAfold, or stochastically generated with RNAsubopt. The command line used to predict the minimum free energy (MFE) secondary structure with RNAfold is

```
RNAfold --noPS < input.fasta
```

where input.fasta is a text file (FASTA format recommended) that stores your input sequence. The --noPS flag is not mandatory, but it prevents the program from generating a postscript file drawing the predicted secondary structure. The program returns the input sequence with its MFE secondary structure in bracket format on the line below.

However, as discussed in previous chapters, single energy minimized structures do not always provide the best secondary structure prediction. Instead, it is recommended to perform a deeper exploration of the conformational space and to consider suboptimal structures [27, 28]. This approach to secondary structure prediction, originally implemented in mfold [27, 29] and Sfold [30], is available with the RNAsubopt program in the Vienna RNA package. The command line for running RNAsubopt is

```
RNAsubopt -e 3 < input.fasta
```

where the “-e” option specifies the depth of the suboptimal search. More specifically, this argument indicates the range (in kCal/mol) from the MFE, within which all suboptimal structures must be returned. Obviously, the values of that range dependent of the MFE of the sequence, and should be adjusted on a case-by-case basis. In our experiments, a value of 3 kCal/Mol generated 25 secondary structures; which appears to provide a good representation of the suboptimal conformational landscape.

The list of suboptimal secondary structures generated by RNA-subopt (available at http://csb.cs.mcgill.ca/RNAMoIP) provides us a description of the set of potential secondary structure backbones. It will be used as it in the next step.

It is worth noting that other software could have been used to generate the initial secondary structures. RNAstructure [31], mfold [29], or Sfold [30] make similar prediction than RNAsubopt. Further, recent software such as MC-Fold [8] and RNAwolf [32] have been developed to predict extended secondary structures (including all noncanonical base pairs) directly from sequence data.

3.5 Prediction of a Complete Base Pairing Interaction Graph with Motif Insertion

We describe how to use RNA-MoIP [20] to insert local motifs into secondary structures generated with RNAsubopt. By default, RNA-MoIP aims to inserts motifs from a repository build with RNA3Dmotif [17]. This repository of nonredundant motifs is included in the distribution of RNA-MoIP and named "No_Redondance_DESC". Nonetheless, advanced users can also either build themselves an up-to-date motif repository using RNA3Dmotif, or use databases available on the RNA 3D Hub [18].

RNA-MoIP is a flexible tool that allows modifications of the secondary structure to permit the insertion of motifs. In particular, the program has the capacity to remove a fixed amount of base pairs from the input secondary structure. This feature is particularly helpful if the predicted secondary structure has incorrectly predicted base pairs that prevent motifs to be inserted.

Let us assume that you work in a directory that contains the RNA-MoIP program (i.e., the python script named "RNAMoIP.py") and that Gurobi has been properly installed. The command line to insert motifs in the sequence and secondary structure with RNA-MoIP is:

```
gurobi.sh RNAMoIP.py -s <sequence> -ss <structure_list> -d
<path_to_repository>
```

Where the argument <sequence> is a string representing the primary structure of the RNA sequence, <structure_list> is the name of the file that stores the secondary structures in bracket format generated by RNAsubopt, and <path_to_repository> is the location of the motif repository. It indicates the path to the motif repository stored in the directory named "CATALOGUE", which is distributed with the RNA-MoIP package at http://csb.cs.mcgill.ca/RNAMoIP/. Assuming that the directory " CATA LOGUE " is in your current directory, the value of

<path_to_repository> is the string "./ CATALOGUE /No_Redondance_DESC/".

RNA-MoIP accepts an additional parameter to control the maximum number of base pairs that can be removed. This parameter can be adjusted with a float number (between 0 and 1) through the option –r. By default, RNA-MoIP allows up to 30% (i.e., –r 0.3) of base pairs to be removed. This is a reasonable choice as the base pair prediction positive predictive value (PPV) is roughly 60% for classical secondary structure predictors such as RNAfold and mfold [33]. Nonetheless, users can decrease this value if they are confident in their predicted secondary structure.

Let us now run a concrete example on the tRNA(Cys) sequence, and let us call "RNAsubopt.out" the file storing the output of RNAsubopt. Then, the RNA-MoIP command line is:

```
gurobi.sh RNAMoIP.py -s 'GCCAGGGUGGCAGAGGGGCUUUGCGGCGGACUGCA
GAUCCGCUUUACCCCGGUUCGAAUCCGGGCCCUGGC'-ss RNAsubopt.out -d './
CATALOGUE/No_Redondance_DESC/'
```

The program outputs first the usual output of Gurobi (the mathematical solver used by RNA-MoIP). We are interested by the output produced after the line starting with "Best objective". It contains the secondary structure used to insert the motifs, the IDs of the inserted motifs, the positions of the deleted base pairs, and the score of the solution. The RNA-MoIP command above will output the following results:

```
Solution for the secondary structure:
(((((((..((((((...)))))).(((((.......))))).....
(((((.......))))))))))))
Optimal solution nb: 1
Corrected secondary structure: ((((((...(((.........)))..
((((.......))))......((((.........)))).))))))
C-2DU6.D.2-12-22-1
C-3CUL.D.6-51-61-1
C-2DU3.D.3-30-38-1
C-2DU5.D.1-6-10-1
C-2DU5.D.1-24-27-2
C-2DU5.D.1-41-48-3
C-2DU5.D.1-64-66-4
D-15-19
D-14-20
D-13-21
D-26-42
D-52-60
D-7-65
```

The optimal solutions has as value:

```
-663.0
```

The first line starts with “Solution for the secondary structure” and returns the best secondary structure extracted from the list of suboptimal structure, which has been used to insert motifs. The next line (starting with “Optimal solution nb”) indicates how many suboptimal structures have been used. In our case, only one structure can be used to build the optimal solution.

The line starting with “Corrected secondary structure” shows the structure used by RNA-MoIP. Since RNA-MoIP may remove some base pairs to insert motifs, this structure can be different to the one returned on the first line.

Then, the program outputs information about the inserted motifs and deleted base pairs. The rows starting with a “D” are the positions of the deleted base pairs. Hence, here the output tells us that the base pairs (15,19), (14,20), (13,21), (26,42), (52,60) and (7,65) have been removed to insert the motifs. We highlight in red the deleted base pairs.

```
(((((((..((((((...)))))).(((((.......))))).....
(((((.......))))))))))))
```

As indicated in the third line, the secondary predicted (or updated) by RNA-MoIP is:

```
((((((...(((.........)))..((((.......))))......
((((.........)))).))))))
```

The rows starting with a “C” display information about the inserted motifs. Each row indicates where a component of a motif (i.e., contiguous sequence of a local motif) has been used. The syntax of these lines is

```
C-<ID>-<first_index>-<last_index>-<component_number>
```

where <ID> is the identifier (or name) of the inserted motif, <first_index> is the first position, and <last_index> is the last position of the insertion. The last value <component_number> indicates which component of the motif has been used.

In our example, four motifs have been inserted: Three hairpins and one 4-way junction. For each motif, Table 1 shows the ID, type, and positions at which each component has been inserted. Figure 4 illustrates these results and shows where the motifs and components have been inserted by RNA-MoIP in the secondary structure.

Table 1
Description of motifs inserted by RNA-MoIP in the suboptimal secondary structures generated by RNAsubopt for the tRNA(Cys) from *Archaeoglobus fulgidus*

Motif ID type indices of insertion sites		
2DU6.D.2	Hairpin	[12,22]
3CUL.D.6	Hairpin	[51,61]
2DU3.D.3	Hairpin	[30,38]
2DU5.D.1	4-Way junction	[6,10],[24, 27],[41,48],[64,66]

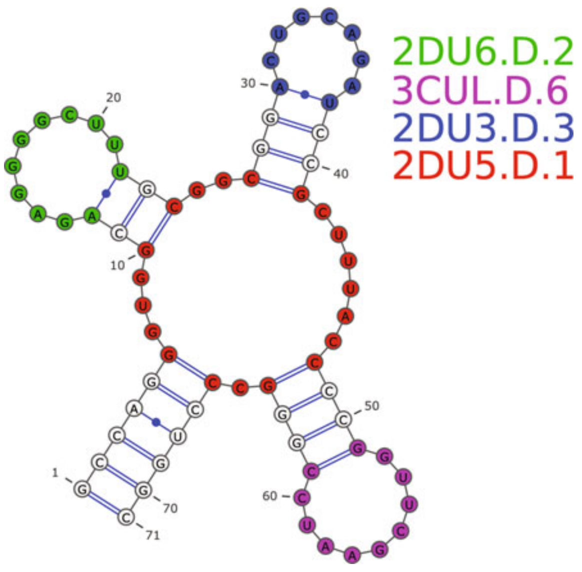


Fig. 4 Location of RNA motifs inserted by RNA-MoIP in the secondary structure of tRNA(Cys) from *Archaeoglobus fulgidus*. Three hairpins (2DU6.D.2 in green, 2CUL.D.6 in magenta, and 2DU3.D.3 in blue) and one four-way junction (2DU5.D.1 in red) have been inserted. We note the four components of the four-way junction, which correspond to the four single strands of the multiloop

The last line in the output returns the value of the objective function that has been optimized by RNA-MoIP. In our example, the value returned is −663.0. Details about this function can be found in [20].

3.6 Reconstruction of Tertiary Structures from Base-Pairing Interaction Networks

Now, we describe how to use a base pairing network to build three-dimensional structures with the MC-Sym software [8]. This operation requires writing scripts that will be used to run MC-Sym at http://www.major.iric.ca/MC-Sym/.

An MC-Sym script is composed of six parts. The first one provides a description of the sequence that is going to be modeled, and the second indicates the set of fragments to be used. The third part is the order in which the fragments will be merged. The fourth and fifth parts describe geometrical constraints to satisfy during the construction of the three-dimensional model. The last segment defines a set of rules for space exploration. Every line starting with '//' is a comment.

In this example, we show how to model the tRNA(Cys) molecule of *Archaeoglobus fulgidus*, a 71-nucleotides long RNA into which four motifs are inserted by RNAMoIP (see previous section). Each part of the script is described and explained separately.

3.6.1 Part 1: Sequence Definition

The first step defines the sequence we are modeling. The syntax is

```
// ==================== Sequence ====================

sequence( r A1
GCCAGGGUGGCAGAGGGGCUUUGCGGCGGACUGCAGAUCCGCUUUACCCCGGUUCGAAU
CCGGGCCCUGGC )

//
((((((...((((((...))))))..((((.......))))......(((((((...))
))))).))))))

//
12345678901234567890123456789012345678901234567890123456789
012345678901

// 1 2 3 4 5 6 7
```

The keyword "sequence" states that we are defining a sequence. The program requires three arguments that are indicated between the parentheses. First, "r" specifies that we are working with an RNA sequence. Next, "A1" sets as an identifier for the sequence "A" and defines the first position as "1." Finally, we end by the sequence and close the parenthesis. The two lines of comment that follows are just here to improve readability.

3.6.2 Part 2: Fragment Definition

This is probably the most important and delicate step of the script. In MC-Sym, the basic units are called cycles. The second step aims to define which cycles and motifs will be used to build the structure. Cycles, motifs and other structural elementary blocks used by MC-Sym are called fragments. The order in which they entered is not important, but we must ensure that every position is included in a fragment and that all fragments overlap.

In the following example, we illustrate how to insert a basic cycle (i.e., not a motif). Here, the cycle is a stack between two consecutive base pairs. This is the most common usage of cycles as most Watson–Crick and Wobble base pairs are predicted by secondary structure predictors and thus not covered by RNA motifs. The code below indicates how and where to map this cycle. In this case, a stacked base pairs at positions 1, 2, 70, and 71.

```
ncm_01 = library(
pdb( "MCSYM-DB/2_2/GCGC/*R20*.pdb.gz" )
#1:#2, #3:#4 <- A1:A2, A70:A71
rmsd( 0.1 sidechain && !( pse || lp || hydrogen ) ) )
```

We start with a unique identifier (i.e., "ncm_01"), followed by the keyword "library". It indicates to MC-Sym where to retrieve the basic fragments.

On the second line, we provide specifications of the files within that library. Here, our library is composed of PDB files, followed by the path "MCSYM-DB/2_2/GCGC/*_1.pdb.gz". MCSYM-DB is the location of all standard library fragments. MC-Sym fragments are sorted according to the number of consecutive nucleotides in each component. Here, we have two times two consecutive nucleotides (1–2 and 70–71). Therefore, we must look into the subfolder "2_2". Next, we must explicitly tell which sequence of nucleotides are involved, in the order 5' to 3'. Since the nucleotides at positions 1, 2, 70 and 71 are respectively 'G', 'C', 'G', 'C', the next path is "GCGC". We end by "*.pdb.gz" to consider all possible structures for this specific fragment. Specific subsets of those fragments can be chosen as specified in the MC-Sym documentation.

On the third line, we provide detailed information on the positions onto which the fragment will be mapped. The syntax is composed of two parts, separated by the symbol "<-". The left-hand side indicates the segments associated with each consecutive component. Segments are represented with an interval, and nucleotides are numbered sequentially according to their position in the motif. The right-hand side defines the specific positions (still as intervals or segments) onto which these components/segments must match. In our sample code, we have two stretches of two nucleotides. We write it "#1:#2, #3:#4", which also implies that the first part is composed of the two first nucleotides and the second one of the next two. On the right hand side, "A1:A2, A70:A71" describes which sequence must be mapped. The nucleotides at from position 1 to 2 for the first component, and the nucleotides at from position 70 to 71 for the second component. "A" is a unique identifier of the sequence to be used, as defined in the first part. The last line is a set of constraints for this specific fragment. Since this information is nonessential here, we kindly redirect the reader to the MC-Sym documentation for more details. It is also important to not forget to close the parenthesis opened after the keyword "library."

Now, we must specify how to declare the fragments inserted by RNAMoIP. The simplest one is a hairpin. Let us start with the hairpin “1DU6.D.2”, between positions “12” and “22.” The code is:

```
pdbs_hairpin_1 = library(
pdb ("/u/reinharz/RNAMOIP-DB-VIEW3D/2DU6.D.2/*.pdb")
#1:#11 <- A12:A22
)
```

The syntax is very similar to those of the previous example. The first line is the same as for the stacked base pair: a unique identifier followed by the keyword “library.” However, the path to the library is different. All RNA-MoIP motifs are contained in a repository that is accessible with the path “/u/reinharz/RNAMOIP-DB-VIEW3D/<Motif Identifier>”. The path must always end by “*.pdb” to indicate that we want to consider all possible structures in the folder. On the third line, we define the positions involved. In this case, the motif contains 11 consecutive nucleotides, thus the left hand side is “#1:#11”. Those nucleotides are at positions “12” to “22” in our sequence, defining the right hand side. We end by closing the parenthesis of “library.”

Finally, motifs with multiple stretches of consecutive nucleotides are a slightly more complicated to specify. We illustrate that case with the four-way junction "2DU5.D.1". Using the same syntax as above, we write the code as follows.

```
pdbs_4_way = library(
pdb ("/u/reinharz/RNAMOIP-DB-VIEW3D/2DU5.D.1/*.pdb")
#1:#5, #6:#9, #10:#17, #18:#20 <- A6:A10, A24:A27, A41:A48,
A64:A66
)
```

In this case, our motif is composed of four components, between positions “6” and “10,” “24” and “27,” “41” and “48,” and “64” and “66” in the input sequence. Each component of consecutive nucleotides needs to be described individually.

3.6.3 Part 3: Merging Fragments

Once all fragments are specified, we must define in which order they are going to be assembled together. One fragment must be selected to start the construction. All others will be sequentially added; with the constraint that each new fragment must overlaps with any of the previous one. The syntax to assemble the first two stacked base pairs, uses the unique names used in the fragment definitions (i.e., ncm_01 and ncm_02), and is written as follows.

```
structure = backtrack
(
ncm_01
merge( ncm_02 1.5 )
)
```

It is important to always start with the keywords “structure = backtrack”. The assembly is indicated between two parentheses. The starting point of our model is “ncm_01” and must be written on the first line. On the second line. we use the keyword “merge” followed, between parentheses, by an argument that indicates the next fragment, and another argument that specifies an “error” threshold that can be tolerated by the program to extend the structure. By default we use a value of “1.5”. More information on this parameter can be found in the MC-Sym documentation.

3.6.4 Parts 4 and 5: Constraints

MC-Sym requires several general parameters and constraints to perform the three-dimensional reconstruction. A standard set of values is provided in the following code and can be used as it is.

```
// =================== Backtrack Restraints
===================
clash
(
structure
1.5 !( pse || lp || hydrogen )
)
backtrack_rst
(
structure
width_limit = 25%,
height_limit = 33%,
method = probabilistic
)
// ================== Ribose Restraints ==================
ribose_rst
(
structure
method = ccm,
pucker = C3p_endo,
threshold = 2.0
)
```

3.6.5 Part 6: Space Exploration Parameters

Finally, the last step is to determine the boundaries of the space to explore. We also specify the time limit and the maximum number of structures to output. MC-Sym will stop as soon as one of those conditions is fulfilled. The other parameter options are standard and can be found in the documentation of MC-Sym.

```
// ================== Exploration Initialization =========
explore
(
structure
option(
model_limit = 1000,
time_limit = 30m,
seed = 3210 )
rmsd( 3.0 sidechain && !( pse || lp || hydrogen ) )
pdb( "structure" zipped )
)
```

Where, the key word "model_limit" specifies the maximal number of structure to generate, and "time_limit" the maximal amount of time allowed to run MC-Sym. Here, we use an upper bound of 30 min with a maximum of 1000 structures. The time needed to generate structures can vary. In our benchmark [20], we have seen that about half an hour of computation was often enough to produce solutions with molecules with sizes up to one hundred nucleotides. Within this time frame, the maximum number of structures created will rarely reach one thousand in that time. Importantly, motifs with a large number of components will highly restrict the exploration of the space, thus the time needed to explore the conformational space. In absence of any motif with three or more components, increasing the time limit to 2 days is a reasonable step. In that case, the number of structures could increase to five or even ten thousands, and we can expect much more diversity in the sample set.

It is difficult to determine in advance how many structure samples are necessary to provide a good representation of the conformational landscape. Numbers between 100 and 1000 are good starts, but more might be occasionally needed. We acknowledge that large numbers of structures are difficult to analyze. Fortunately, as we will see in the next section, the MC-Sym server offers several tools to facilitate these tasks.

The key word "seed" is a random number used to initialize MC-Sym. Its value has little importance in the context of this discussion, and users can modify it if they wish.

The complete source code of this MC-Sym script is available at http://csb.cs.mcgill.ca/RNAMoIP/.

3.7 Retrieving, Optimizing, Analyzing and Visualizing Structures

Once submitted, the script will run on MC-Sym servers and the URL of a result page will be returned to the user. This page offers multiple options to optimize, analyze and retrieve the results. To access these options, the user must access the web page "commands.html" located in the results directory.

The energy minimization of the MC-Sym structure is probably one of the most important options. We recommend any user to run it before analyzing or visualizing the results. The "steepest descent" option returns satisfactory results in a short time, but more sophisticated and slower techniques (e.g., simulated annealing) are also available.

Clustering of structures using the k-means algorithm is another useful option that enables the user to automatically identify the most significant structures in the set or structures returned by MC-Sym.

A PDB model of all predicted structures is available at the root of the directory. Each model can be visualized with a molecular viewer such as PyMOL or Jmol. Figure 5 show an example of a structure predicted with our RNA-MoIP and MC-Sym pipeline, aligned with the experimental structure [23].

3.8 Web Server

A web server is available with most functionalities at rnamoip.cs.mcgill.ca [34]. It has two objectives. The first one is to help the user to visualize the secondary structure and motif predictions of RNA-MoIP. The second is to facilitate the production of complete 3D models with the automated generation of a script for MC-Sym web server.

The basic input is an RNA sequence. For convenience, up to five sequences can be given in the same file, which should use the FASTA format. The user can adjust several parameters in the

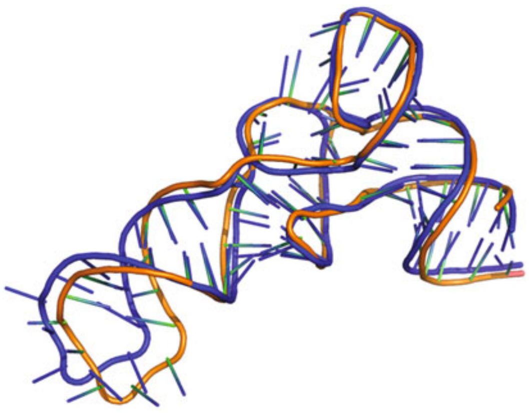


Fig. 5 3D structure of the tRNA(Cys) from *Archaeoglobus fulgidus* predicted with the RNA-MoIP+MC-Sym pipeline (in blue), aligned with its crystallographic model [23]

Advanced Options menu, including (1) the maximal fraction of base pairs that can be removed from the secondary structure (default 30%), (2) the complexity of junctions (by default only hairpin, interior loops, and three way junctions are allowed but the user can optionally choose to search for four way junctions), (3) the maximal number of solutions to output (default 1), and (4) the method to generate candidate secondary structures (i.e., the parameters to be used by RNAsubopt to sample the secondary structures or a custom list).

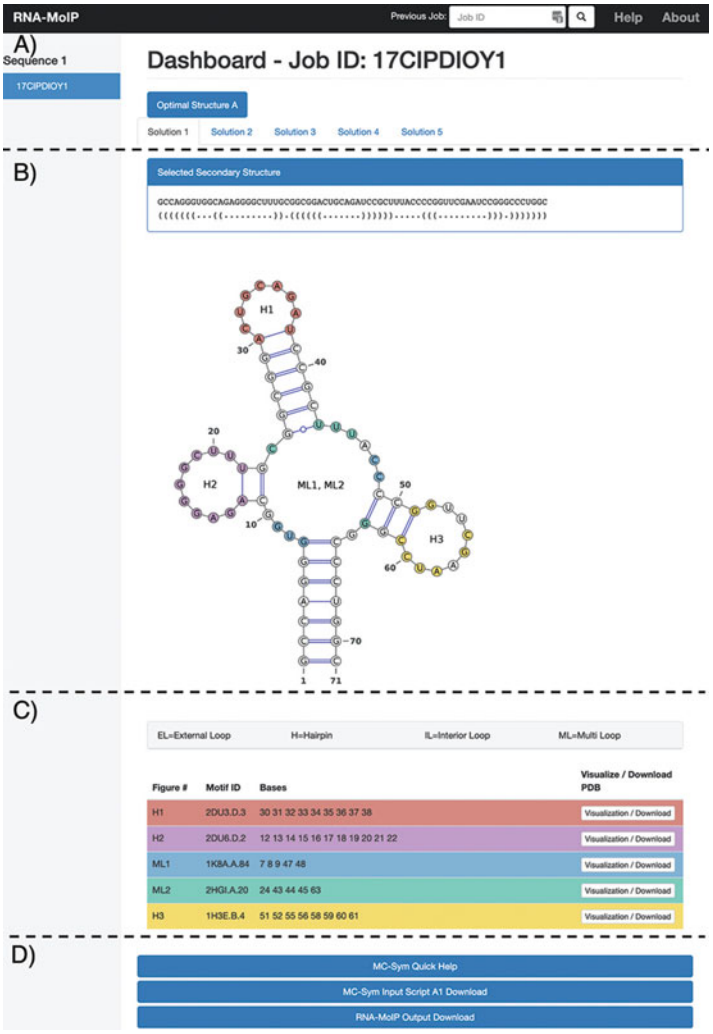
Figure 6 shows the main page of the web server output. Its URL is available for at least 2 weeks. There are four main elements in the output. The top bar (A) contains links to other optimal RNA-MoIP predictions (i.e., other candidate secondary structures with motifs insertions computed by RNA-MoIP). The middle frame (B) contains the sequence and secondary structure in dot-bracket notation as well as the graphical VARNA representation [35]. Below, a table (C) stores all inserted motifs with their position in the sequence and links to visualize them separately in an interactive 3D viewer. Finally, the footer (D) features links to download a MC-Sym script that can be used on the MC-Sym web server to generate 3D structures, and a tarball with the entire RNA-MoIP output.

3.9 Prediction of RNA 3D Modules from Sequence Data

In order to map an input sequence to a 3D structure, RNA-MoIP identifies subsequences that match exactly to a known 3D structure fragments. Since data must match with a 3D structure, the number of covered sequences remains relatively low. While very robust, this assumption has the downside of ignoring slightly different sequences which can fold into the same 3D structure, but have not been crystallized yet.

Alignments of RNA sequences with similar secondary structure and function have been assembled and curated, and include, for some motifs, hundreds of distinct sequences for which a crystal structure is not yet available but can confidently be expected to adopt the same three-dimensional structure. These alignments, referred to as family alignments, are available in the Rfam database [36]. RNA-MoIP cannot yet directly take advantage of this information. Yet, we developed BayesPairing [37, 38] with the task of scanning a sequence and identifying RNA 3D modules. It is a python package that comes with a pretrained database of modules, which should be sufficient to considerably enrich 3D structure prediction. The software is available as a web server and as a standalone tool, as long as a user manual, at bayespairing.cs.mcgill.ca. With the stand-alone version, we use the following commands to execute the program:

```
cd bayespairing/src
python parse_sequences.py -s input.fa -samplesize 1000
```


RNA-MoIP Previous Job: Job ID Help About

A)
Sequence 1
17CIPDIOY1

Dashboard - Job ID: 17CIPDIOY1

Optimal Structure A

Solution 1 Solution 2 Solution 3 Solution 4 Solution 5

B)

Selected Secondary Structure

```
GCCAGGGUGGCAGAGGGGCUUUGCGGCGGACUGCAGAUCCGCUUUACCCCGGUUCGAAUCCGGGCCCUGGC
(((((((...((.........)).((((((.......)))))).....(((.........))).)))))))
```

C)

EL=External Loop H=Hairpin IL=Interior Loop ML=Multi Loop

Figure #	Motif ID	Bases	Visualize / Download PDB
H1	2DU3.D.3	30 31 32 33 34 35 36 37 38	Visualization / Download
H2	2DU6.D.2	12 13 14 15 16 17 18 19 20 21 22	Visualization / Download
ML1	1K8A.A.84	7 8 9 47 48	Visualization / Download
ML2	2HGI.A.20	24 43 44 45 63	Visualization / Download
H3	1H3E.B.4	51 52 55 56 58 59 60 61	Visualization / Download

D)

MC-Sym Quick Help
MC-Sym Input Script A1 Download
RNA-MoIP Output Download

Fig. 6 RNAMoIP web server output on the *Archaeoglobus fulgidus* sequence (2DU3). In (A) there is the opportunity to toggle between different optimal structures and solutions. We can see here that there are five insertion patterns that are equivalent. In (B) there is the sequence and corrected secondary structure—after base pairs removal—as a 2D VARNA representation of that structure with the positions where modules are inserted colored. In (C) a table shows all inserted modules with their positions in the sequence, and provides a link to see the module in an interactive 3D window and download its PDB description. Finally in (D) there are links to download the full raw output of RNA-MoIP as the automatically generated script for MC-Sym

where “input.fa” is a fasta file containing the sequences to parse. BayesPairing then prints out predictions. If the secondary structure of the tested sequences is known or has been previously predicted by another software, it can be added to the input fasta file as an extra line after the sequence under each entry. A typical output looks like follows:

```
===========================================================
| MODULE    SCORE        MODULE POSITIONS         SEQUENCE
|
|  0         6.85       5-6,29-35,65-68
CA*UUUGAAC*GGAG        |
|  0         2.46       25-26,42-48,72-75
CA*UUUGAAC*GGAG        |
|  0         1.54       9-10,28-34,59-62
CA*UUUGAAC*GGAG        |
|  1         5.17       5-9,35-38,43-44,69-72
CAGGG*CCCU*AC*GGAG     |
=========================================================
```

where the first column contains the module ID in the default BayesPairing database. This default database includes motifs that can map to previously published module catalogues like Rna3dMotif [17], the RNA 3D Motif Atlas [39], the RNAbricks database [40], or caRNAval [41]. A description and graphical representation of each module is available with BayesPairing. The second column describes a probabilistic score, indicating how likely the presence of the module is. The third column includes the predicted position of the module in the input sequence, and the fourth column contains the sequence at these positions.

BayesPairing 2 performs predictions by sampling the secondary structure ensemble with RNAsubopt like RNA-MoIP, so both software can be expected to perform well on the same type of sequence. For sequences in the length range that can be accurately predicted by RNA-MoIP and MC-Sym, we recommend using a sample size of 1000 structures with the parameter `-samplesize`.

It is also possible to create a new model to scan for a specific module of interest with BayesPairing. The inputs required are a graph, which models the local 3D structure of the module (available with Rna3dmotif and the RNA 3D Motif Atlas), and a multiple sequence alignment, which represents the different nucleotides that can be adopted by the module at each position without significantly affecting the final 3D structure.

BayesPairing can already be used to enhance RNA-MoIP predictions. Indeed, if an important module is not immediately found by RNA-MoIP but is identified BayesPairing, a user can pass this information to RNA-MoIP through a DESC structure file with sequence positions. This feature, only available in the command line interface, allows RNA-MoIP to include any motifs identified with other tools. BayesPairing can thus be used to verify, enhance or complement module identifications performed by RNA-MoIP, eventually contributing to a better 3D structure determination by adding new constraints.

4 Notes

1. The motifs database used by RNA-MoIP includes motifs from the PDB model 2DU3 (tRNA(Cys) from *Archaeoglobus fulgidus*) database. In applications on other (new) molecules, it will be very unlikely to have motifs from the same molecule in the motif repository. Fortunately, even in the absence of motifs, the benchmark performed in [20] shows that the quality of the structure prediction remains high.

References

1. Bekaert M, Bidou L, Denise A et al (2003) Towards a computational model for -1 eukaryotic frameshifting sites. Bioinformatics 19:327–335. https://doi.org/10.1093/bioinformatics/btf868
2. Vitreschak AG, Rodionov DA, Mironov AA, Gelfand MS (2004) Riboswitches: the oldest mechanism for the regulation of gene expression? Trends Genet 20:44–50. https://doi.org/10.1016/j.tig.2003.11.008
3. Szewczak AA, Moore PB, Chang YL, Wool IG (1993) The conformation of the sarcin/ricin loop from 28S ribosomal RNA. Proc Natl Acad Sci U S A 90:9581–9585. https://doi.org/10.1073/pnas.90.20.9581
4. Šponer J, Otyepka M, Banáš P et al (2012) Molecular dynamics simulations of RNA molecules. In: Schlick T (ed) Innovations in biomolecular modeling and simulations: volume 2. RSC Biomolecular Science, London, pp 129–155
5. Ding F, Sharma S, Chalasani P et al (2008) Ab initio RNA folding by discrete molecular dynamics: from structure prediction to folding mechanisms. RNA 14:1164–1173
6. Jonikas MA, Radmer RJ, Laederach A et al (2009) Coarse-grained modeling of large RNA molecules with knowledge-based potentials and structural filters. RNA 15:189–199. https://doi.org/10.1261/rna.1270809
7. Poursina M, Bhalerao KD, Flores SC et al (2011) Strategies for articulated multibody-based adaptive coarse grain simulation of RNA. Methods Enzymol 487:73–98. https://doi.org/10.1016/B978-0-12-381270-4.00003-2
8. Parisien M, Major F (2008) The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data. Nature 452:51–55. https://doi.org/10.1038/nature06684
9. Martinez HM, Maizel JV Jr, Shapiro BA (2008) RNA2D3D: a program for generating, viewing, and comparing 3-dimensional models of RNA. J Biomol Struct Dyn 25:669–683. https://doi.org/10.1080/07391102.2008.10531240
10. Zhao Y, Huang Y, Gong Z et al (2012) Automated and fast building of three-dimensional RNA structures. Sci Rep 2:734
11. Das R, Baker D (2007) Automated de novo prediction of native-like RNA tertiary structures. Proc Natl Acad Sci U S A 104:14664–14669. https://doi.org/10.1073/pnas.0703836104
12. Das R, Karanicolas J, Baker D (2010) Atomic accuracy in predicting and designing noncanonical RNA structure. Nat Methods 7:291–294. https://doi.org/10.1038/nmeth.1433
13. Wang Z, Xu J (2011) A conditional random fields method for RNA sequence–structure relationship modeling and conformation sampling. Bioinformatics 27:i102–i110. https://doi.org/10.1093/bioinformatics/btr232
14. Rother M, Rother K, Puton T, Bujnicki JM (2011) ModeRNA: a tool for comparative

modeling of RNA 3D structure. Nucleic Acids Res 39:4007–4022. https://doi.org/10.1093/nar/gkq1320

15. Leontis NB, Westhof E (2001) Geometric nomenclature and classification of RNA base pairs. RNA 7:499–512. https://doi.org/10.1017/s1355838201002515
16. Yang H, Jossinet F, Leontis N et al (2003) Tools for the automatic identification and classification of RNA base pairs. Nucleic Acids Res 31:3450–3460. https://doi.org/10.1093/nar/gkg529
17. Djelloul M, Denise A (2008) Automated motif extraction and classification in RNA tertiary structures. RNA 14:2489–2497. https://doi.org/10.1261/rna.1061108
18. Leontis NB, Zirbel CL (2012) Nonredundant 3D structure datasets for RNA knowledge extraction and benchmarking. In: Leontis N, Westhof E (eds) RNA 3D structure analysis and prediction. Springer, Berlin, pp 281–298
19. Hofacker IL, Fontana W, Stadler PF et al (1994) Fast folding and comparison of RNA secondary structures. Chem Monthly 125:167–188
20. Reinharz V, Major F, Waldispühl J (2012) Towards 3D structure prediction of large RNA molecules: an integer programming framework to insert local 3D motifs in RNA secondary structure. Bioinformatics 28: i207–i214. https://doi.org/10.1093/bioinformatics/bts226
21. Berman HM, Olson WK, Beveridge DL et al (1992) The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids. Biophys J 63:751–759. https://doi.org/10.1016/S0006-3495(92)81649-1
22. Bernstein FC, Koetzle TF, Williams GJB et al (1978) The protein data bank: a computer-based archival file for macromolecular structures. Arch Biochem Biophys 185:584–591
23. Fukunaga R, Yokoyama S (2007) Structural insights into the first step of RNA-dependent cysteine biosynthesis in archaea. Nat Struct Mol Biol 14:272–279
24. Lorenz R, Bernhart SH, Siederdissen CH et al (2011) ViennaRNA package 2.0. Algorith Mol Biol 6:26
25. Waugh A, Gendron P, Altman R et al (2002) RNAML: a standard syntax for exchanging RNA information. RNA 8:707–717. https://doi.org/10.1017/s1355838202028017
26. Lemieux S, Major F (2002) RNA canonical and non-canonical base pairing types: a recognition method and complete repertoire. Nucleic Acids Res 30:4250–4263. https://doi.org/10.1093/nar/gkf540
27. Zuker M (1989) On finding all suboptimal foldings of an RNA molecule. Science 244:48–52. https://doi.org/10.1126/science.2468181
28. Ding Y, Chan CY, Lawrence CE (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. RNA 11:1157–1166. https://doi.org/10.1261/rna.2500605
29. Zuker M, Mathews DH, Turner DH (1999) Algorithms and thermodynamics for RNA secondary structure prediction: a practical guide. In: Barciszewski J, Clark BFC (eds) RNA biochemistry and biotechnology. Springer, Dordrecht, pp 11–43
30. Ding Y, Lawrence CE (2003) A statistical sampling algorithm for RNA secondary structure prediction. Nucleic Acids Res 31:7280–7301. https://doi.org/10.1093/nar/gkg938
31. Reuter JS, Mathews DH (2010) RNAstructure: software for RNA secondary structure prediction and analysis. BMC Bioinformatics 11:129. https://doi.org/10.1186/1471-2105-11-129
32. zu SCH, zu Siederdissen CH, Bernhart SH et al (2011) A folding algorithm for extended RNA secondary structures. Bioinformatics 27: i129–i136
33. Do CB, Woods DA, Batzoglou S (2006) CONTRAfold: RNA secondary structure prediction without physics-based models. Bioinformatics 22:e90–e98. https://doi.org/10.1093/bioinformatics/btl246
34. Yao J, Reinharz V, Major F, Waldispühl J (2017) RNA-MoIP: prediction of RNA secondary structure and local 3D motifs from sequence data. Nucleic Acids Res 45: W440–W444. https://doi.org/10.1093/nar/gkx429
35. Darty K, Denise A, Ponty Y (2009) VARNA: interactive drawing and editing of the RNA secondary structure. Bioinformatics 25:1974–1975. https://doi.org/10.1093/bioinformatics/btp250
36. Kalvari I, Argasinska J, Quinones-Olvera N et al (2018) Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. Nucleic Acids Res 46:D335–D342. https://doi.org/10.1093/nar/gkx1038
37. Sarrazin-Gendron R, Reinharz V, Oliver CG et al (2019) Automated, customizable and efficient identification of 3D base pair modules with BayesPairing. Nucleic Acids Res 47:3321–3332. https://doi.org/10.1093/nar/gkz102

38. Sarrazin-Gendron R, Yao H-T, Reinharz V et al (2019) Stochastic sampling of structural contexts improves the scalability and accuracy of RNA 3D modules identification. bioRxiv 2019:834762

39. Petrov AI, Zirbel CL, Leontis NB (2013) Automated classification of RNA 3D motifs and the RNA 3D motif atlas. RNA 19:1327–1340. https://doi.org/10.1261/rna.039438.113

40. Chojnowski G, Waleń T, Bujnicki JM (2014) RNA bricks—a database of RNA 3D motifs and their interactions. Nucleic Acids Res 42: D123–D131. https://doi.org/10.1093/nar/gkt1084

41. Reinharz V, Soulé A, Westhof E et al (2018) Mining for recurrent long-range interactions in RNA structures reveals embedded hierarchies in network families. Nucleic Acids Res 46:3841–3851

Chapter 3

Motif Discovery from CLIP Experiments

Marco Pietrosanto, Gabriele Ausiello, and Manuela Helmer-Citterich

Abstract

RNA primary and secondary motif discovery is an important step in the annotation and characterization of unknown interaction dynamics between RNAs and RNA-Binding Proteins, and several methods have been developed to meet the need of fast and efficient discovery of interaction motifs. Recent advances have increased the amount of data produced by experimental assays and there is no available method suitable for the analysis of all type of results. Here we present a simple workflow to help choosing the more appropriate method, depending on the starting situation, among the three algorithms that best cover the landscape of approaches. A detailed analysis is presented to highlight the need for different algorithms in different working settings. In conclusion, the proposed workflow depends on the nature of the starting data and on the availability of RNA annotations.

Key words Motif discovery, RNA secondary structure, CLIP-Seq, High-throughput assay, RNA

1 Identification of Structural Motifs in Ensembles of RNA Molecules

We will start with a possible (and widely used) description of the problem.

We have a set of RNA sequences, whose representation can be thought as a string (for primary or even secondary structure [1]), all of which are supposed to share a common feature (which may be a common function, a common interactor, similar roles). These RNAs are defined by their length L_i and by their number N. Moreover, in this problem, an assumption is used that requires an additional layer of management, to specify whether or not one expects to find the motif (whether in primary or secondary structure) in each RNA or only in a subset, and how many motif instances one expects to find in each RNA harbouring the motif. The ZOOPS model (that stands for Zero or One Occurrence Per Sequence) assumes that in every RNA the number of occurrences of the motif to be found is either 0 or 1. In other words, you may not have two occurrences of the same motif in one single RNA (more words will be spent about this later) and, more importantly, this super-problem requires the ability to find a subset of “correct” RNAs that really contain the functional region.

In numbers the search space number of states Ω can be written as

$$\Omega(N, \overline{L}) = \frac{2^{2N}}{N+1} \sum_{l=1}^{\overline{L}} l(\overline{L} - l + 1)$$

where $\overline{L}$ is the mean length of the analyzed RNAs.

Considering one section of the equation at a time: the problem of finding which RNA contains the motif creates a series of possible answers that are exactly 2^N, the number of possible ways of subsetting a set of N items.

We now have to take into account the fact that the motif length and its position in every RNA is unknown. This actually extends the search space by a factor of

$$\sum_{l=1}^{\bar{L}} l(\bar{L} - l + 1)$$

for each RNA, where $\bar{L}$ is the mean length along the input set of RNAs. However, this factor is understandably less problematic since its quadratic dependence. This aspect is one of the possible reasons why most methods focus on the research of patterns in inputs that are characterized by a relatively low number of RNA molecules (100–200), each one up to 2 kb long or more. In this region of the input space the computational complexity becomes feasible enough to think of efficient exhaustive searches.

On the other hand, moving toward the direction of more input RNAs exponentially amplifies the problem (by a factor that is proportional to 2^N), thus making the state space unexplorable in a complete way.

Let us clarify this with a plot: in Fig. 1 there is a sketch of the approximated search space dimension (in Log2) with the number of input RNAs (N) and the mean RNA sequence length ($\overline{L}$) as variables.

It is quite clear how the increase in N is the dominant term here. Existing methods for structural motifs, to the best of our knowledge, fall in the left side of the input space plot, ranging from lower left to upper left and they seldom overcome the limits of few hundreds of RNAs. If they do so they are bound to increase their needed time exponentially (CMfinder for example takes ~1 min to explore 100 RNAs with a mean length of 100 nt but ~20 min for a dataset of 500 RNAs of the same size), but that is because they have very efficient specialized algorithms that are not designed for those kinds of inputs.

The solution for the limitation of having only one occurrence (and not more) per molecule can be approximated by

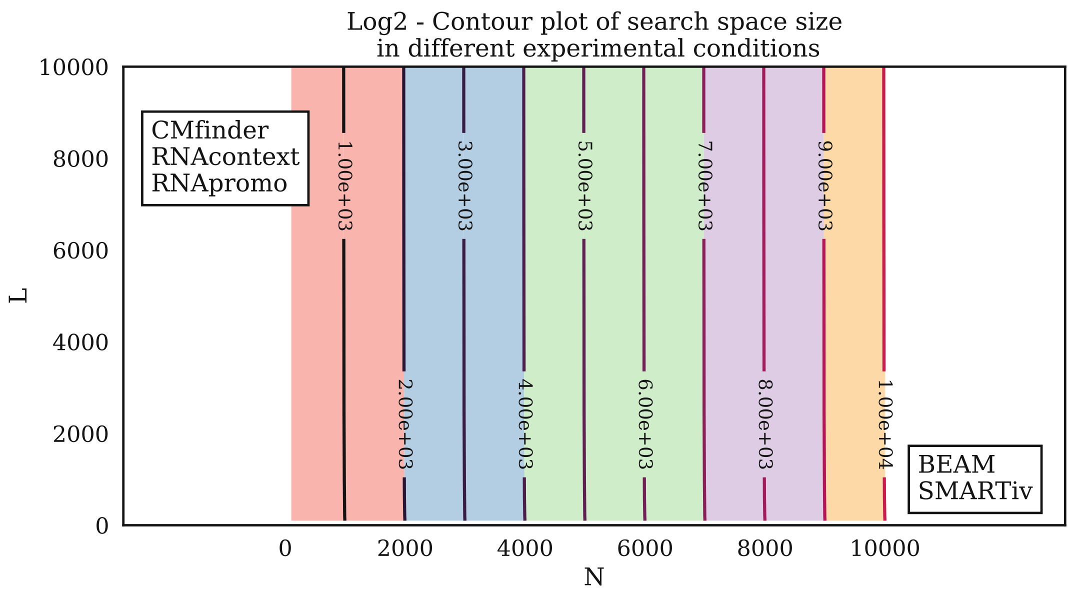


Fig. 1 Base-2 logarithm of the number of states allowed by the problem of motif finding as characterized by the ZOOPS assumption, with unknown motif Length and noise addition

superimposing another layer that can address this aspect afterward, for example by searching the derived motif model in the input set while dropping the ZOOPS assumption. Note that this does neither hinder nor slow the process since the exponential dependence of the search space from the number of RNAs (2^N as stated before) is only present during the construction of a motif model (that is the research of a common similar piece - sequence or structure, up to now it is equivalent because the part of the problem that depends on the structure has not been introduced—in a subset of the input set).

The subsequent search of the motif model in the input set, that can be used to refine the model or to catch all occurrences, is not affected by exponential growth of the state space. This kind of problem has in fact a number of states that have to be checked that is roughly proportional to

$$\left(\overline{L} - l\right) \times N$$

where l is the motif model length and $\overline{L}$ is the mean length of the input set of RNAs. Since looking for all the possible candidates is not exponential as it is in the model definition, it becomes feasible to simply scan all the input set looking for possible occurrences of the same motif.

2 Available Methods

Many methods have been developed for the problem of motif discovery (counting both primary and secondary structure focused methods there are more than 30, for a recent review see the excellent work from Sasse and colleagues [2]). Each offers a peculiar approach on the problem modeling, ranging from standard PSSMs to higher-order interactions (e.g., graphs, [3]), taking into account secondary structures at different levels, from a simple context managing [4] to complex structural dependencies encoded in the data representation [5–8]. Moreover, a subset of these methods is developed to work on strong assumptions of conserved RNAs [8], or can only be applied to in vitro RBP-RNA data [9, 10], or relies on secondary structure prediction for the structural aspect of their algorithm [6, 11]. The landscape is heterogeneous and no single method is able to cover any working setting. However, some methods have shown to cover more possibilities than others, while maintaining excellent performances. We are referring to CMfinder [8], SMARTIV [12], and BEAM [5], which will be the focus of this work.

CMfinder estimates covariance models for a set of aligned RNAs, assuming their conservation. It is at the core of the RFAM database [13]. The algorithm is highly efficient for long RNAs (since it does not rely on single structure predictions) but extremely slow for sets of RNAs with more than ~500 elements. Its algorithm is deterministic but does not provide an explicit way of managing noisy data. It does not rely on single secondary structure prediction and estimates a *consensus* secondary structure of the resulting covariance model that can be interpreted as a motif. This method has not been explicitly developed for motif discovery tasks but in practice it can be used with efficiency by searching for a suitable alignment of your input set (with *CMalign*) and then computing the covariance model with *CMfinder*.

SMARTIV explicitly uses data from in vivo binding techniques such as CLIP-Seq assays, by exploiting the binding affinity to cut the dataset into a target set (usually the highest 1000 scoring RNAs) and a background set (the others). It relies on secondary structure prediction (RNAfold by default [14]) to predict single RNA conformations and uses a combined sequence-structure encoding which maps each nucleotide into its secondary structure state (paired or unpaired). The analysis is carried out by enumerating enriched k-mers. Authors do not explicitly cover the noise managing but the assumption on the top ranking affinity score is solid.

BEAM has been built to work on high numbers of elements. The secondary structure is encoded with a high-order alphabet which takes into account secondary structure elements and lengths, and uses a secondary structure elements substitution matrix (developed in [1]) to perform secondary structure alignments. It runs parallel primary and secondary structure analyses by aligning local secondary contexts to a Position Frequency Matrix (representing the motif) which is built dynamically during the run and optimized by means of a simulated annealing procedure. The algorithm is however not deterministic and depending on the complexity of the dataset can produce different suboptimal solutions. It has been shown that it can manage noisy data making up to the 80% of the dataset.

3 Workflow

The standard starting point is having a set of supposedly homofunctional RNAs. If the data is produced by RBP-RNA binding assays such as in vivo PAR-CLIP/HITS-CLIP it may be annotated with RNAs estimated binding affinities. In that case SMARTIV is the best tool since it explicitly exploits that part and directly uses the BED file coming out from experimental assays. However, if this annotation is not present, the experimenter wants to use a different background data, or there is reason to ignore the affinity score, one should select a subset of data (e.g., by cutting at a certain score threshold, depending on the experimental conditions) and do as follows: both BEAM and CMfinder need explicit RNA sequences/structures, so there is need to extract the sequences from chromosomic intervals. This is best done by the bedtools suite software bedtofa [15] (Fig. 2). If using BEAM, chromosomic intervals coming out from these experiments are usually short (~30 nt), and should secondary structure analysis be needed, a longer

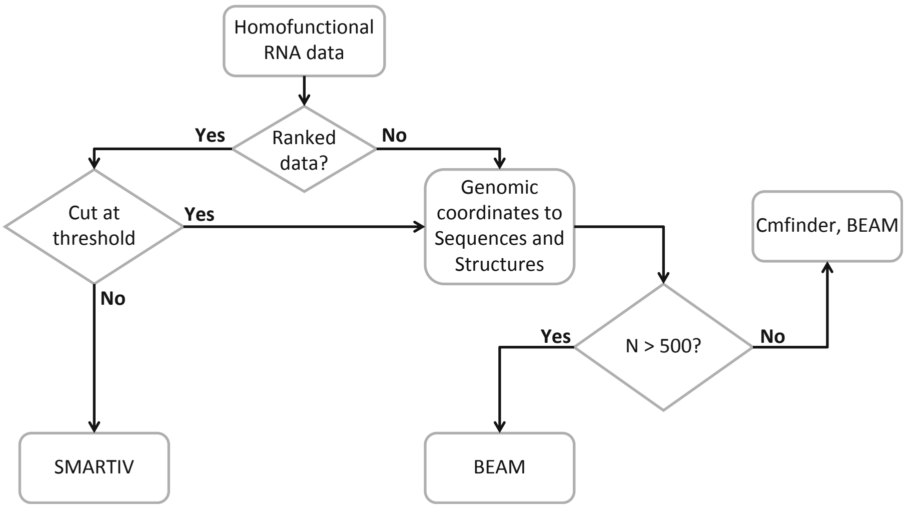


Fig. 2 BED to Fasta suggested pipeline

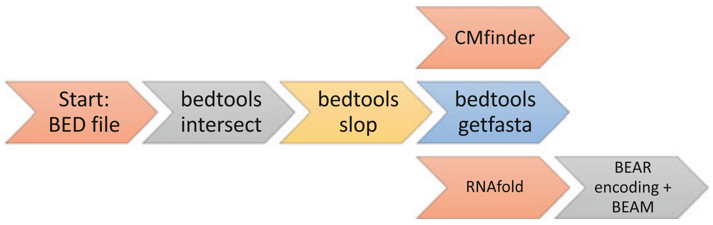


Fig. 3 Suggested workflow from raw data to motif discovery

sequence is preferred (for structural prediction). With this regard we suggest to extend chromosomic intervals by a fixed amount at both sides (the BEAM server provides this service automatically [16]) (Fig. 3). With this workflow one should not exceed ~500 nt for the final length of each RNA since secondary structure prediction tools are not efficient nor accurate over that threshold [17]. While CMfinder works best for smaller datasets sizes (it becomes computationally unfeasible for datasets bigger than ~500 elements), BEAM can work at any size provided (it has been tested up to 100.000 elements) [6].

4 Comparison with CMfinder on Small Datasets

The workflow is well separated for different working conditions, but a comparison between BEAM and CMfinder for smaller, non-ranked datasets has been made in [5].

The test is as follows: consider the datasets that contained a total of 100 structural inserts used to assess BEAM performances on large inputs (about 10^4 molecules, described in [5]—main and supplementary data—and available online at beam.uniroma2.it/data), without added noise (i.e., where the insert is put in the 100% of the RNAs and with one occurrence per RNA). Since CMfinder was designed for smaller RNA collections, sample 100 RNAs for every dataset 100 times (extraction with reinsertion) ending with 100 noiseless datasets for every insert.

Both RNAfold+BEAM and CMfinder were run requesting only one motif in output (using the following command lines: **CMfinder**: *perl -w cmfinder.pl -n 1 $file;* **BEAM**: *java -jar BEAM_-release_1.5.1.jar -f $file -M 1 -R 1*), and the output for every dataset was compared to the reference structure using a structural gapless alignment and the MBR substitution matrix as scoring function. The test score is therefore a *ratio* of the structural alignment score of the motif model in output against the reference structure with the reference structure against itself (the maximum alignment score possible). This score lies between 0 and 1 since negative scores due to very bad alignments (very rare, 1 for each method) were forced to 0. If the motif and the reference were of different lengths the

Table 1
Wall-clock run-time for CMfinder and BEAM at different input sizes

Input size (#RNA)	100	500	10,000
CMfinder (s)	~150	~3000	NA
BEAM (s)	~80	~100	~100

maximum scoring window was reported. The BEAM motif model was considered as the highest individually scoring RNA in the output. The CMfinder motif model was taken from the *.motif.* file.

The results show that for small datasets BEAM performances are very similar to those of CMfinder, with some outliers where BEAM has higher performances: tolerating at most 5% difference between scores, the test resulted in 50 times where BEAM went better and 25 "ties" (this holds for the case where every point is the mean of the insert-related dataset, that is the worst).

For what it regards wall-clock run-time, with inputs of about 100 RNAs both RNAfold plus BEAM and CMfinder alone take around $O(10^2)$ seconds to compute a single motif. To show however that BEAM could maintain performances while not affecting running time, the same test was repeated with bigger datasets (500 and 10^4 molecules). Performances were maintained for both procedures, however CMfinder times took an average of about 35 min (2100 s) per run with input size 500 while BEAM 100 s, revealing a faster execution time for extremely large datasets (such as CLIP data). For larger datasets CMfinder execution times became unfeasible and thus its usage in those conditions is discouraged (Table 1).

5 Output

Each suggested software has its own data format to provide as output. It is useful to know what to expect when choosing which one is more suitable for the available data.

SMARTIV reports a list of motifs as Position Weight Matrices (PWMs) with both primary sequence and their most probable paired/unpaired status, providing a sequence-focused Logo representation and the matrix itself.

CMfinder *.motif.* file contains the information about the motif model as a sequence PWM as well as files for the derived Covariance model. This is well integrated with other InfeRNAl software like CMsearch in order to provide further insights or analyses of your motif.

BEAM outputs a list of sequences that match the putative motif and PWMs for both primary and secondary structure composing the result along with a Logo representation of the secondary structure.

For both SMARTIV and BEAM there is currently no preconstructed follow-up software and further work should be done by analyzing the PWMs with in-house scripts.

References

1. Mattei E, Ausiello G, Ferrè F, Helmer-Citterich M (2014) A novel approach to represent and compare RNA secondary structures. Nucleic Acids Res 42:6146–6157. https://doi.org/10.1093/nar/gku283
2. Sasse A, Laverty KU, Hughes TR, Morris QD (2018) Motif models for RNA-binding proteins. Curr Opin Struct Biol 53:115–123. https://doi.org/10.1016/j.sbi.2018.08.001
3. Maticzka D, Lange SJ, Costa F, Backofen R (2014) GraphProt: modeling binding preferences of RNA-binding proteins. Genome Biol 15:R17. https://doi.org/10.1186/gb-2014-15-1-r17
4. Hiller M, Pudimat R, Busch A, Backofen R (2006) Using RNA secondary structures to guide sequence motif finding towards single-stranded regions. Nucleic Acids Res 34:e117. https://doi.org/10.1093/nar/gkl544
5. Pietrosanto M, Mattei E, Helmer-Citterich M, Ferrè F (2016) A novel method for the identification of conserved structural patterns in RNA: from small scale to high-throughput applications. Nucleic Acids Res 44:8600–8609. https://doi.org/10.1093/nar/gkw750
6. Adinolfi M, Pietrosanto M, Parca L et al (2019) Discovering sequence and structure landscapes in RNA interaction motifs. Nucleic Acids Res 47:4958–4969. https://doi.org/10.1093/nar/gkz250
7. Rabani M, Kertesz M, Segal E (2011) Computational prediction of RNA structural motifs involved in post-transcriptional regulatory processes. Methods Mol Biol 714:467–479
8. Yao Z, Weinberg Z, Ruzzo WL (2006) CMfinder--a covariance model based RNA motif finding algorithm. Bioinformatics 22. https://doi.org/10.1093/bioinformatics/btk008
9. Kazan H, Ray D, Chan ET et al (2010) RNAcontext: a new method for learning the sequence and structure binding preferences of RNA-binding proteins. PLoS Comput Biol 6:28. https://doi.org/10.1371/journal.pcbi.1000832
10. Cook KB, Vembu S, Ha KCH et al (2017) RNAcompete-S: combined RNA sequence/structure preferences for RNA binding proteins derived from a single-step in vitro selection. Methods 126:18–28. https://doi.org/10.1016/j.ymeth.2017.06.024
11. Heller D, Krestel R, Ohler U et al (2017) ssHMM: extracting intuitive sequence-structure motifs from high-throughput RNA--binding protein data. Nucleic Acids Res 45:11004–11018. https://doi.org/10.1093/nar/gkx756
12. Polishchuk M, Paz I, Kohen R et al (2017) A combined sequence and structure based method for discovering enriched motifs in RNA from in vivo binding data. Methods 118–119:73–81. https://doi.org/10.1016/j.ymeth.2017.03.003
13. Kalvari I, Argasinska J, Quinones-Olvera N et al (2017) Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. Nucleic Acids Res 46:335–342. https://doi.org/10.1093/nar/gkx1038
14. Gruber AR, Lorenz R, Bernhart SH et al (2008) The Vienna RNA websuite. Nucleic Acids Res 36:W70–W74. https://doi.org/10.1093/nar/gkn188
15. Quinlan AR (2014) BEDTools: the Swiss-army tool for genome feature analysis. Curr Protoc Bioinformatics 47:11.12.1–11.12.34. https://doi.org/10.1002/0471250953.bi1112s47
16. Pietrosanto M, Adinolfi M, Casula R et al (2018) BEAM web server: a tool for structural RNA motif discovery. Bioinformatics 34:1058–1060. https://doi.org/10.1093/bioinformatics/btx704
17. Will S, Grüning B, Backofen R et al (2017) Recent advances in RNA folding. J Biotechnol 261:97–104. https://doi.org/10.1016/j.jbiotec.2017.07.007

Chapter 4

Profiling of RNA Structure at Single-Nucleotide Resolution Using nextPARS

Uciel Chorostecki, Jesse R. Willis, Ester Saus, and Toni Gabaldon

Abstract

RNA molecules play important roles in almost every cellular process, and their functions are mediated by their sequence and structure. Determining the secondary structure of RNAs is central to understanding RNA function and evolution. RNA structure probing techniques coupled to high-throughput sequencing allow determining structural features of RNA molecules at transcriptome-wide scales. Our group recently developed a novel Illumina-based implementation of in vitro parallel probing of RNA structures called nextPARS.

Here, we describe a protocol for the computation of the nextPARS scores and their use to obtain the structural profile (single- or double-stranded state) of an RNA sequence at single-nucleotide resolution.

Key words RNA secondary structure, Genome-wide enzymatic probing, RNA structurome, RNA folding

1 Introduction

Knowledge of the secondary structure of RNAs both in vivo and in vitro is crucial for understanding the regulatory roles that RNAs exert in most cell functions, via characterizing their intramolecular interactions, and how they can change depending on external conditions, including interactions with other molecules [1]. Among several approaches that have been described during the last years to interrogate RNA structure using high throughput sequencing technologies, nextPARS [2] is an enzymatic-based technique that allows probing the secondary structure of RNAs in vitro at a genome-wide scale. It is an adaptation of the previously developed PARS [3] strategy to the Illumina sequencing platform, which allows for sample multiplexing and higher throughput than the original technique.

To obtain structural information of RNAs using nextPARS [2], two µg of total or polyA-selected RNA of the species of interest are first denatured, folded, and then enzymatically probed with RNase

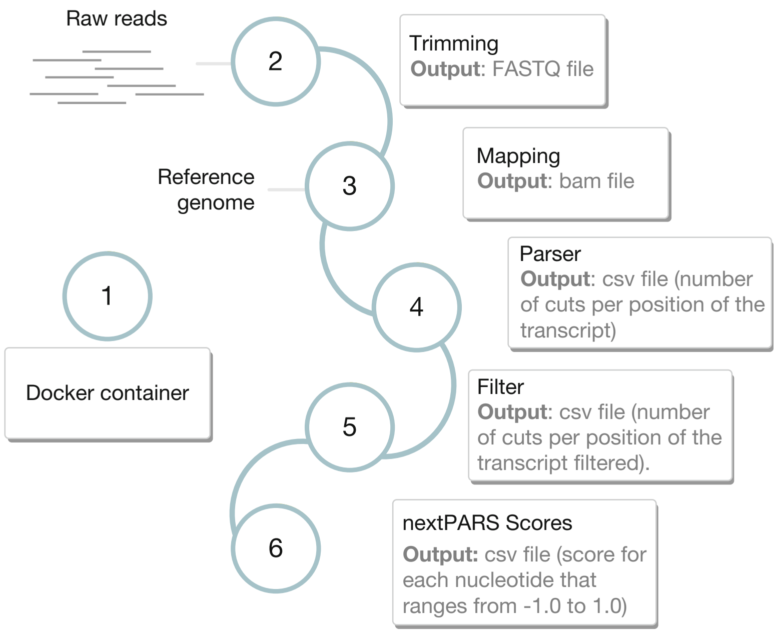

Raw reads
2
Trimming
Output: FASTQ file
Reference genome
3
Mapping
Output: bam file
1
Docker container
4
Parser
Output: csv file (number of cuts per position of the transcript)
5
Filter
Output: csv file (number of cuts per position of the transcript filtered).
6
nextPARS Scores
Output: csv file (score for each nucleotide that ranges from -1.0 to 1.0)

Fig. 1 nextPARS Pipeline. Flowchart showing the steps involved in the nextPARS protocol to go from sequencing reads to structural profiles of RNAs present in a sample

V1 to cleave double-stranded sites and with S1 nuclease in a separate sample to cut single-stranded conformations. Some additional RNA molecules of interest can be spiked in each of the samples, as possible positive controls with known structure, for example. Digested samples are library prepared using the TruSeq Small RNA Sample Preparation Kit (Illumina), reverse transcribed, PCR-amplified and finally sequenced in multiplex with Illumina platform. The reads obtained are mapped to the reference genome and the cutting points are determined at the 5′end of the reads. Further details on the experimental methodology can be found in the original nextPARS publication [2].

In this chapter, we briefly describe nextPARS methodology and report, using an illustrative example, how the nextPARS raw output data is analyzed in order to go from sequencing reads to structural profiles of RNAs present in a sample (Fig. 1).

2 Materials

2.1 Dataset and Overview

Let us briefly describe the samples and formats for this work (Fig. 1).

– *Raw sequencing data*: we obtained the sequencing data (raw reads) from the SRA database (accession number

PRJNA380612). The raw sequencing data is in standard Sanger FASTQ format. Details on the experimental methodology can be found in nextPARS publication [2].

– *Preprocessing*: we removed adapter sequences from sequencing reads. The input and output data is in FASTQ format.
– *Mapping of Illumina reads and determination of enzymatic cleavage points*. Illumina reads were aligned to the *S. cerevisiae* reference genome and we concatenated the sequences of spike-ins control molecules. We use S288C full chromosomes version R64-2-1, released 18 Nov 2014 (fasta file) and features file from the 16 nuclear chromosomes plus the mitochondrial genome (gff file). The output is in .bam format.
– *Parsing*. For each read alignment, we retrieved the 5′-end position in the reference genome and compared this to the genome annotation. The resulting digestion profile is stored as the number of cuts per position of the transcript. This is stored in csv, comma-separated format (values for each position are separated by semicolons).
– *nextPARS score*: To obtain the scores from nextPARS experiments we used as an input the csv files described above (number of cuts per position of the transcript), and we obtained a new csv file. This file contains a structural profile of an RNA transcript (single- or double-stranded state), with a score for each nucleotide that ranges from −1.0 (highest preference for single strand) to 1.0 (highest preference for double-strand).

2.2 Computational Hardware

We tested the pipeline on a computer with Intel(R) Xeon(R) CPU 3.30GHz, with four cores and 32 GB of RAM. We use Linux (Debian) on x86_64 Architecture. The size of the raw data, intermediate files and final results are about 30 GB for this example. Therefore, 100 GB of hard disk space is required for computing the example data.

2.3 Computational Software

Reproducibility facilitates peer review and ensures that the model, application, and analysis you build can run without worrying about details and software installation. Also, it facilitates collaboration and sharing. Therefore, we built a Docker container based on Ubuntu 18 for nextPARS software. Docker is a tool designed to make it easier to create and run applications by using containers. Containers allow developers to package up an application without the need to make custom builds for different environments. As a result, the application will run on any other Linux machine regardless of any custom settings the machine may have that could differ from the machine used to write and test the code.

This protocol assumes users have a Unix-like operating system (i.e., Linux or macOS X), with a bash shell or similar, thus all commands have to be run in a terminal. Although it is possible to follow this protocol with a Microsoft Windows machine (e.g., using the Unix-like Cygwin), the additional steps required are not discussed here.

3 Methods

To illustrate this example, we will use six samples from *S. cerevisiae* and additional RNA molecules from V1 and S1 nextPARS experiment. The data can be downloaded from the publicly SRA database (accession number PRJNA380612) (*see* **Note 1**). Most of the examples are shown using one of the samples (1_S1), so you have to repeat each step on the other five samples. In step 7, the scripts were adapted to process all samples together.

In order to go from the fastq outputs of the nextPARS experiments to a format that allows us to calculate scores, first map the reads in the fastq files to a reference genome using the program of your choice [4]. If you already did that you can omit steps 2 and 3. But, If you want to download the data from SRA with SRA toolkit, trim the sequences with cutadapt and map to the genome with STAR (see **Note 2** and Subheadings 3.2 and 3.3).

This table shows the correlation of each sample run from the SRA project and the sample name (V1's and S1's experiments).

```
# RUN        SAMPLE_NAME
# SRR5422921 1_V1.fastq
# SRR5422920 2_V1.fastq
# SRR5422919 3_V1.fastq
# SRR5422926 1_S1.fastq
# SRR5422925 2_S1.fastq
# SRR5422924 3_S1.fastq
```

You can download the source code and the sample data from GitHub (https://github.com/Gabaldonlab/nextPARS_docker) and the docker container from Docker Hub (https://hub.docker.com/nextpars).

3.1 Setup the Pipeline: TIMING <5 min

3.1.1 Clone the Git Repository

```
git clone https://github.com/Gabaldonlab/nextPARS_docker.git
```

Download the last image version from Docker Hub.

```
docker pull cgenomics/nextpars
```

3.1.2 Download and Run Docker Container

Start a docker interactive session using the `nextpars` container.

```
 docker run -it -v /path/to/nextpars/github/:/home/nextPARS
cgenomics/nextpars:latest bash
```

The `/path/to/nextpars/github` refers to the path from your local machine where you have the nextPARS directory (clone from github). Warning: Anything you do within the docker will be reflected in the data folder!

Now you are working inside the docker container. First, cd into `bin/scripts` directory and execute the configure bash script.

```
cd /home/nextPARS/bin/scripts
source configure.sh
```

3.1.3 Build Genome Indexes (Optional)

Step 1.3, 2, and 3 are optional. If you want to trim and map the raw reads with different software than cutadapt [5] and STAR [6], proceed directly to Step 4 (Fig. 1).

STAR requires a reference genome index for mapping. We use *S. cerevisiae* S288C as the reference genome that was downloaded from Saccharomyces Genome Database (SGD) [7]. We build the index by using the following command.

```
 cd $DATAPATH/DB/saccharomyces_cerevisiae
 STAR --runThreadN 4 \
 --runMode genomeGenerate \
 --genomeDir . \
 --genomeFastaFiles saccharomyces_cerevisiae.fasta \
 --sjdbGTFfile saccharomyces_cerevisiae.gff \
 --sjdbGTFfeatureExon gene \
 --sjdbGTFtagExonParentTranscript Parent
  # --runThreadN Number of threads to be used for genome
generation
 # --runMode genomeGenerate option directs STAR to run genome
indices generation job
 # --genomeDir specifies the path to the directory where the
genome indices are stored
 # genomeFastaFiles specified one or more FASTA files with the
genome reference sequences
 # specifies the path to the file with annotated transcripts in
the standard GTF format
 # --sjdbGTFfeatureExon tag name to be used as exons' parents
for building transcripts
 # --sjdbOverhang specifies the length of the genomic sequence
around the annotated junction to be used in constructing the
splice junctions database.
```

3.2 Trimming Sequence Adapters: TIMING <3 min per sample (Optional)

Raw sequencing reads likely to contain parts of the adapter sequence. Therefore, these sequences must be identified and trimmed. These adapters can be removed using a specialized adapter removal tool and there is a more than large choice for the appropriate published adapter trimming tools. Here, we use cutadapt for this purpose. In `fastq` directory you should have the 6 samples from *S. cerevisiae* nextPARS experiments.

```
 cd $DATAPATH
 cutadapt -a TGGAATTCTCGGGTGCCAAGGAACTCCAGTCAC -m 18 -j 4 -o
$DATAPATH/trimming/1_S1.fastq.qz $DATAPATH/fastq/1_S1.fastq
 # -a ADAPTER Regular 3′ adapter
 # -m LENGTH Discard processed reads that are shorter than
LENGTH.
 # -j N, where N is the number of cores to use.
 # -o output file (output file formats are FASTA and FASTQ,
with optional compression and the output file format is
recognized from the file name extension).
```

3.3 Aligning the Reads Against the Genome using STAR: TIMING <5 min per Sample (Optional)

Mapping reads to the genome: TIMING <5 min per sample

The trimmed reads should be aligned to the reference genome. Mapping results are in BAM format.

```
 cd $DATAPATH
 STAR --runThreadN 4 \
 --genomeDir $DATAPATH/DB/saccharomyces_cerevisiae/ \
 --readFilesIn $DATAPATH/trimming/1_S1.fastq.qz \
 --outFileNamePrefix $DATAPATH/mapping/1_S1.fastq \
 --outSAMtype BAM SortedByCoordinate \
 --outTmpDir $TMP_DIR/1_S1.fastq
 # --runThreadN Number of threads to use for mapping
 # --genomeDir path of the STAR index
 # ---readFilesIn name(s) (with path) of the files containing
the sequences to be mapped
 # --outFileNamePrefix Output files name prefix (full or
relative path).
 # --outSAMtype type of SAM/BAM output
 # --readFilesCommand Command line to execute for each of the
input files.
 # --outTmpDir path to a directory that will be used as
temporary by STAR.
```

Once you obtain the bam file, use nextPARSParser.py to count the number of reads at each position (which indicates a cut site for the enzyme in the file name).

3.4 Parser: Number of Reads Beginning at Each Position: TIMING <8 min per Sample

If you skipped steps 1.3, 2 and 3, we assume that you have the bam files in the `data/mapping` directory. As an alternative, you can use already generated bam files (subsets from the originals), that are in `data/mapping_subset` directory. To do so, you have to change `-b` argument by modifying `mapping` by `mapping_subset` on the following command.

```
 cd $BINPATH

 python nextPARSParser.py \
 -b $DATAPATH/mapping/1_S1.fastqAligned.sortedByCoord.out.bam
\
 -g $DATAPATH/DB/saccharomyces_cerevisiae/saccharomyces_cere-
visiae.gff \
 -o $DATAPATH/tab/1_S1.tab \
 -t gene

 # -b Path to the SAM/BAM file containing the mapped reads
 # -g Path to the GTF file containing the features
 # -o The name given to the output file in csv format (.tab
extension)
 # -t Feature type (3rd column in GTF file) to be used
(default, suitable for Ensembl GTF files: exon)
 # -a Skip all reads with MAPQ alignment quality lower than the
given minimum value (default: 10). MAPQ is the 5th column of a
SAM/BAM file and its usage depends on the software used to map
the reads.
```

The output of this script is a csv file containing the name of the molecule and the count values (number of inferred enzyme cuts) for each position, separated by semicolons.

3.5 Filtered Out Transcripts with Low Counts: TIMING <1 min per Sample

The reformat_PARSparser_output.py script filters out transcripts with low counts and produces output in csv format.

```
 cd $BINPATH
 python reformat_PARSparser_output.py -t $DATAPATH/tab/1_S1.
tab -m 20

 # -m Min average counts for a given transcript
 # -tab csv file to be reformatted
```

3.6 nextPARS Scores: TIMING <1 min per Sample

To obtain the final scores from nextPARS experiments (from .tab files), use the following command. For details on how to calculate the scores, *see* **Note 3**.

```
cd $BINPATH
python get_combined_score.py \
 -i U1 \
 -inDir ../data/tab \
 -f ../data/fasta/U1.fa \
 -o ../data/score/U1_score.RNN.tab

# -i to indicate the molecule for which you want scores (all available data files will be included in the calculations -- molecule name must match that in the data file names)
# -inDir to indicate the directory containing the .tab files with read counts for each V1 and S1 enzyme cuts
# -f to indicate the path to the fasta file for the input molecule
# -s to produce an output Structure Preference Profile (SPP) file. Values for each position are separated by semi-colons. Here 0 = paired position, 1 = unpaired position, and NA = position with a score too low to determine its configuration.
# -o to output the calculated scores, again with values for each position separated by semi-colons.
# --nP_only to output the calculated nextPARS scores before incorporating the RNN classifier, again with values for each position separated by semi-colons.
# {-V nextPARS} to produce an output with the scores that is compatible with the structure visualization program VARNA1
# {-V spp} to produce an output with the SPP values that is compatible with VARNA.
# -t to change the threshold value for scores when determining SPP values [default = 0.8, or -0.8 for negative scores]
# -c to change the percentile cap for raw values at the beginning of calculations [default = 95]
# -v to print some statistics in the case that there is a reference CT file available. If not, will still print nextPARS scores and info about the enzyme .tab files included in the calculations.
```

3.7 Automatization of Steps

In order to simplify steps 2–5 (Fig. 1), there are different bash scripts in the bin/scripts directory inside the container. These scripts will help to automate the pipeline and to use a set of V1 and S1 samples. We assume that you have the .fastq files in the `data/fastq` directory.

```
cd ~/bin/scripts
```

Trimming

```
./trimming.sh
```

Mapping reads. Use the following script to do the mapping. First, you have to generate the genome index (step 1.3).

```
./mapping.sh
```

If you want to start with the Parser step (count number of reads beginning at each position), we assumed that you have the bam files in the `data/mapping` directory. As an alternative, you can use already generated bam files (subsets from the originals), that are inside the `data/mapping_subset` directory. To do so, you have to modify the FILE variable (changing `mapping` by `mapping_-subset`) on the following script.

Parser

```
./nextPARSParser.sh
```

Filter out transcripts

```
./reformat_PARSParser.sh
```

4 Notes

1. *Download SRA sequences.*

   How to download sequence data files using SRA Toolkit is explained in detail here: https://www.ncbi.nlm.nih.gov/sra/docs/sradownload/.

2. *Trimming the data.*

   If you are not sure if you need to trim your data, fastp [8] is a tool that has implemented methods that automatically detect 5' or 3' adapters for both paired and single-end data.

3. *Computation of nextPARS scores.*

   We present here a summary of the main step of the next-PARS methodology, the computation of nextPARS scores from csv (.tab) files (number of reads at each position, which indicates a cut site for the enzyme). This is implemented in get_-combined_score.py script. A more detailed description can be found in [2].

   Phase I: scores from raw experimental data (Sprofile).

   There are five parts to the profile score (S_{profile}) calculation:

   (a) First, the digestion profiles are read from the .tab files. This gives the number of cuts at each position, and it should include one or more .tab file for both the V1 and S1 enzymes.

(b) These raw counts are capped at the maximum percentile of the value. By default, this is 95%, so that any position with more cuts than 95% of the other positions are set to this 95th percentile value. This helps to dampen the skew in cuts due to the few positions that may be preferentially cleaved by either V1 or S1 at rates orders of magnitudes greater than most other positions.

(c) The capped counts from each .tab file are then normalized to its average so that each .tab now has a mean of 1 cut per position. This corrects for (I) the different expression levels for each molecule, (II) different sequencing depth between runs of the nextPARS experiment, and (III) the different rates of cleavage between the V1 and S1 enzymes, the latter of which cuts more frequently. Since the final 50 positions do not have counts in a nextPARS experiment of this example (50 bases long reads were used), these are not included in the normalization.

(d) In the case that multiple replicates of the experiment were performed, a single list of cuts is generated separately for V1 and S1 by taking the average cuts at each position.

(e) Finally, a single combined *S* score is calculated to determine whether a position is likely to be paired or unpaired, using the following steps. (I) The lists of average normalized V1 and S1 cuts are each then normalized to a maximum of 1. (II) S1 values are subtracted from V1 values to know if a position tends to be cut more by one enzyme or the other. (III) The resulting positive values are then normalized to a maximum of +1 while the negative values are normalized to a minimum of -1, such that the range of values is always from -1 to +1. In this way, we have a fixed range to which we can apply threshold values for cuts per position to determine those which can be confidently called paired or unpaired.

Phase II: scores from a recurrent neural network (RNN) classifier (SRNN).

The $S_{profile}$ calculation can then be complemented by recurrent neural network (RNN) classifier score (S_{RNN}) which calculates the probability that a position is paired or unpaired by considering that nucleotide and its neighbors. The model is trained on a database of known RNA secondary structures and is constructed with a long term short memory (LTSM) layer and a dense neural network layer [9, 10]. For details on the training and implementation of the classifier model [2].

Use the RNN classifier separately.

The RNN classifier that already incorporated into the nextPARS scores can run separately, using a different experimental score input (in .tab format), it can be run like so: `python predict2.py -f molecule.fasta -p scoreFile.tab -o output.tab`

Acknowledgments

We thank Manuel Molina for his help with testing the pipeline. This work was supported by a grant from the Marie Skłodowska-Curie Actions [H2020-MSCA-IF-2017-793699] to U.C., and grants from the Spanish Ministry of Economy, Industry, and Competitiveness (MEIC) [PGC2018-099921-B-I00] cofounded by European Regional Development Fund (ERDF); from the CERCA Programme/Generalitat de Catalunya; from the Catalan Research Agency (AGAUR) SGR423, from the European Union's Horizon 2020 research and innovation program under the grant agreement ERC-2016-724173; and from INB (PT17/0009/0023—ISCIII-SGEFI/ERDF) to T.G.

References

1. Wan Y, Kertesz M, Spitale RC, Segal E, Chang HY (2011) Understanding the transcriptome through RNA structure. Nat Rev Genet 12 (9):641–655
2. Saus E, Willis JR, Pryszcz LP, Hafez A, Llorens C, Himmelbauer H, Gabaldón T (2018) nextPARS: parallel probing of RNA structures in Illumina. RNA 24(4):609–619
3. Kertesz M, Wan Y, Mazor E, Rinn JL, Nutter RC, Chang HY, Segal E (2010) Genome-wide measurement of RNA secondary structure in yeast. Nature 467(7311):103–107
4. Costa-Silva J, Domingues D, Lopes FM (2017) RNA-Seq differential expression analysis: an extended review and a software tool. PLoS One 12(12):e0190152
5. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet J 17:1
6. Dobin A, Davis C, Schlesinger F, Drenkow F, Zaleski C, Jha S, Batut S, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29(1):15–21
7. Cherry JM, Hong EL, Amundsen C, Balakrishnan R, Binkley G, Chan ET, Christie KR, Costanzo MC, Dwight SS, Engel SR, Fisk DG, Hirschman JE, Hitz BC, Karra K, Krieger CJ, Miyasato SR, Nash RS, Park J, Skrzypek MS, Simison M, Weng S, Wong ED (2012) Saccharomyces Genome Database: the genomics resource of budding yeast. Nucleic Acids Res 40(Database issue):D700–D705
8. Chen S, Zhou Y, Chen Y, Gu J (2018) fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics 34-17:i884–i890
9. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780
10. Sutskever I, Vinyals O, Le Q (2014) NIPS '14: proceedings of the 27th international conference on neural information processing systems, vol 2. Neural Information Processing Systems (NIPS), Montreal, QC

Chapter 5

RNA Framework for Assaying the Structure of RNAs by High-Throughput Sequencing

Tycho Marinus and Danny Incarnato

Abstract

RNA structure is a key player in regulating a plethora of biological processes. A large part of the functions carried out by RNA is mediated by its structure. To this end, in the last decade big effort has been put in the development of new RNA probing methods based on Next-Generation Sequencing (NGS), aimed at the rapid transcriptome-scale interrogation of RNA structures. In this chapter we describe RNA Framework, the to date most comprehensive toolkit for the analysis of NGS-based RNA structure probing experiments. By using two published datasets, we here illustrate how to use the different components of the RNA Framework and how to choose the analysis parameters according to the experimental setup.

Key words RNA structure, RNA probing, High-throughput sequencing, DMS, SHAPE

1 Introduction

NGS-based methods for RNA structure probing are rapidly becoming the standard for studying RNA structures under both in vitro and in vivo conditions. These approaches take advantage of chemicals that are either able to modify the Watson–Crick interface of single-stranded nucleobases (i.e., dimethyl sulfate, CMCT, kethoxal, EDC) or the 2′-OH of the ribose moiety of structurally flexible RNA residues (i.e., SHAPE reagents) [1].

The typical readout of these experiment is based on the detection of reverse transcription (RT) stop/drop-off events (due to the inability of most RT enzymes to read through these modified residues) [1]. More recently, mutational profiling (MaP) approaches have been devised to enable RT read-through at these modification sites [2–5], leading to their recording as mutations within the resulting cDNA molecule.

Although these experimental methods are now becoming widely employed to query RNA structures, no standard data analysis approach has yet been defined. We have recently introduced the RNA Framework as a generalized toolkit for the analysis of

NGS-derived RNA structure probing experiments [6]. Herein we describe a detailed procedure for the analysis of data derived from both RT stop-based and MaP approaches, from read mapping to RNA structure modeling, using RNA Framework. Particularly, we will exploit two datasets generated by in vivo probing of *Saccharomyces cerevisiae* with dimethyl sulfate (DMS), an alkylating reagent, that is able to readily permeate cell membranes, resulting in the rapid methylation of respectively the N1 and N3 of adenosine (A) and cytosine (C) residues.

2 Materials

2.1 RNA Framework

RNA Framework is implemented in Perl and tested on Linux (Fedora Core 21-30) and it can be obtained from our website (http://www.rnaframework.com/). It requires a computer with a 64-bit architecture running Linux, Mac OS X or any other UNIX-based operating system and Perl v5.12 (or greater), with ithreads support.

The following software and packages are required by RNA Framework:

- Bowtie v1.1.2 or greater (http://bowtie-bio.sourceforge.net/index.shtml), and/or Bowtie v2.2.7 or greater (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml).
- SAMTools v1.2 or greater (http://www.htslib.org/).
- BEDTools v2.0 or greater (https://github.com/arq5x/bedtools2/).
- Cutadapt v2.1 or greater (http://cutadapt.readthedocs.io/en/stable/index.html).
- ViennaRNA Package v2.4.0 or greater (http://www.tbi.univie.ac.at/RNA/).
- Perl non-CORE modules (http://search.cpan.org/):
  - DBD::mysql.
  - RNA (installed by the ViennaRNA package).
  - XML::LibXML.
  - Config::Simple.

To install RNA Framework, it is sufficient to clone it from the Git repository:

```
$ git clone https://github.com/dincarnato/RNAFramework
```

This will create the "RNAFramework" folder. Then, to add the RNA Framework executables to your PATH, simply type:

```
$ export PATH=$PATH:$(pwd)/RNAFramework
```

Table 1
List of datasets used in this chapter

Accession ID	Description	Reference
SRR815612	In vivo DMS-seq (*S. cerevisiae*, polyA+, DMS treated, biological replicate 1)	[7]
SRR815613	In vivo DMS-seq (*S. cerevisiae*, polyA+, DMS treated, biological replicate 2)	[7]
SRR815614	In vivo DMS-seq (*S. cerevisiae*, polyA+, DMS treated, biological replicate 3)	[7]
SRR815615	In vivo DMS-seq (*S. cerevisiae*, polyA+, DMS treated, biological replicate 4)	[7]
SRR3929621	In vivo DMS-MaPseq (*S. cerevisiae*, polyA+, DMS treated, biological replicate 1)	[4]
SRR3929622 SRR3929623	In vivo DMS-MaPseq (*S. cerevisiae*, polyA+, DMS treated, biological replicate 2)	[4]
SRR3929626	In vivo DMS-MaPseq (*S. cerevisiae*, polyA+, untreated control)	[4]

To obtain a detailed help, with a complete description of the allowed parameters, each tool can be invoked with the "-h" (or "--help") flag, for example:

```
$ rf-index -h
```

Alternatively, you can refer to the online manual (https://rnaframework.readthedocs.io).

2.2 Datasets

In order to walk the reader through the use of RNA Framework, here we are going to use two published datasets, DMS-seq [7] and DMS-MaPseq [4], obtained by in vivo probing of *S. cerevisiae* with dimethyl sulfate (DMS). Datasets can be retrieved from the NCBI Sequence Read Archive (https://www.ncbi.nlm.nih.gov/sra) and converted to FastQ format using the NCBI SRA Toolkit (available at https://www.ncbi.nlm.nih.gov/sra/docs/toolkitsoft/) and the accession IDs reported in Table 1:

```
$ fastq-dump -A <SRA accession ID>
```

This will generate a FastQ file named after the provided SRA file.

3 Methods

By default, RNA Framework relies on Bowtie v1 or Bowtie v2 [8, 9] for read mapping. It is however possible to use any other aligner. In case alignment has been already performed, skip the next two paragraphs and proceed directly with Subheading 3.3 ("Counting per-base DMS modifications").

3.1 Reference Index Creation

Bowtie v1 can only perform ungapped read alignment, thus it is only suitable for the analysis of RT stop-based experiments (i.e., DMS-seq [7], Structure-seq [10], SHAPE-seq [11], CIRS-seq [12]). It is rather advisable to use Bowtie v2 for the analysis of mutational profiling (MaP) experiments (i.e., SHAPE-MaP [2], DMS-MaPseq [4, 5]), as a substantial part of the mutational information of these experiments is recorded within sequencing reads in the form of insertions and deletions.

As we are going to illustrate the analysis of both DMS-seq and DMS-MaPseq data, we will generate both Bowtie v1 and v2 indexes for the *S. cerevisiae* transcriptome reference, using the **rf-index** tool. rf-index automatically generates Bowtie reference indexes by querying the UCSC genome database (https://genome.ucsc.edu) for a given genome assembly and gene annotation. A complete list of the available genome assemblies can be found at https://genome.ucsc.edu/FAQ/FAQreleases.html. For example, available gene annotations for the "sacCer3" *S. cerevisiae* genome assembly can be listed through the "-la" parameter:

```
$ rf-index -g sacCer3 -la
```

To build the reference transcriptome index using the NCBI RefSeq gene annotation, type:

```
$ rf-index -g sacCer3 -a ncbiRefSeq  # Bowtie v1 index
$ rf-index -b2 -g sacCer3 -a ncbiRefSeq  # Bowtie v2 index
```

rf-index will generate a folder named "sacCer3_ncbiRefSeq_bt" (or "sacCer3_ncbiRefSeq_bt2" if the "-b2" parameter was specified), containing the reference genome's FASTA file, the gene annotation BED file, the reference transcriptome FASTA file and the Bowtie index files.

Additionally, rf-index comes with a set of prebuilt indexes, that can be listed through the "-lp" flag.

3.2 Read Mapping

RNA Framework performs read mapping via the **rf-map** tool. rf-map provides a streamlined interface for adapter clipping and trimming of low-quality bases followed by read mapping. This step will result in the generation of a sorted BAM file for each processed FastQ file.

Optionally, prior to read mapping, it is advisable to inspect base qualities. This can be easily performed using FastQC (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/).

3.2.1 Mapping of DMS-Seq Reads

For the mapping of DMS-seq data we will use rf-map and Bowtie v1:

```
$ rf-map -ca3 TCGTATGCCGTCTTCTGCTTG -bi sacCer3_ncbiRef-
Seq_bt2/sacCer3_ncbiRefSeq -bnr -bc 3200 -ba -bm 20 -o
rf_map_seq SRR81561*.fastq
```

In this example, the four FastQ files belonging to four biological replicates can be simultaneously processed through a single rf-map command, as they are being mapped on the same reference.

The "-ca3" parameter defines the adapter sequence to clip at the 3′ end of reads. With paired-end experiments, a 5′ adapter sequence can also be provided via the "-ca5" parameter (this sequence will be automatically reverse-complemented).

By default, trimming of low-quality bases (Phred <20) is performed only from the 3′ end of reads (controlled by the "-cq3" parameter). Quality trimming of bases from the 5′ end (controlled by the "-cq5" parameter) must be avoided when analyzing RNA footprinting experiments based on the detection of RT stops (*see* **Note 1**).

These trimming/clipping steps are optional and can be skipped through the flags "-cqo", to disable adapter clipping, and "-cp", to disable both adapter clipping and quality trimming.

The "-bi" parameter specifies the reference index (*see* Subheading 3.1). It is worth pointing out that Bowtie indexes consist of multiple files. Only the basename common to all files must be provided to rf-map (in this example, "sacCer3_ncbiRefSeq").

The DMS-seq library prep strategy results in the sequencing of reads having the same sequence of the RNA transcripts they have originated from. As rf-index generates a transcriptome index, the "-bnr" parameter is needed to instruct Bowtie to only allow reads mapping to the forward reference strand.

By default, Bowtie v1 randomly reports a single mapping when multiple equally-scoring mapping locations are possible. Use of the "-ba" flag instructs Bowtie to report all these equally-scoring mapping positions. The "-bm" parameter sets the maximum allowed number of equally scoring positions for a read. If more than this number of mappings are possible, the read is discarded.

3.2.2 Mapping of DMS-MaPseq Reads

For the mapping of DMS-MaPseq data, we will use rf-map and Bowtie v2 (enabled by the "-b2" flag):

```
$ rf-map -b2 -cq5 20 -ca3 TCGTATGCCGTCTTCTGCTTG -mp '--very-
sensitive-local' -bi sacCer3_ncbiRefSeq_bt2/sacCer3_ncbiRef-
Seq -o rf_map_mapseq SRR392962*.fastq
```

In this case the structure information is encoded within sequencing reads in the form of mutations. Therefore, also quality trimming of 5′ end can be performed through the "-cq5" parameter.

Since we are using Bowtie v2 for read mapping, it is important to pick the right reference index folder (note the "_bt2" suffix).

Even though RNA Framework comes with a lot of built-in options, specific mapping parameters can be provided to Bowtie through the "-mp" parameter. In this example, we are directly invoking Bowtie v2 with the "--very-sensitive-local" preset, that causes Bowtie to extensively look for the top-scoring local alignment.

3.3 Counting Per-base DMS Modifications

Following mapping, **rf-count** calculates per-base RT stop/mutation counts and read coverage for each transcript.

Counting of RT stops is the default behavior of rf-count. It is worth remembering that it is essential to account for the eventual 5′ end read trimming that could have been performed during the mapping stage (*see* **Note 2**). DMS-seq samples are analyzed by:

```
$ rf-count -r -f sacCer3_ncbiRefSeq_bt/sacCer3_ncbiRefSeq.fa
-o rf_count_seq rf_map_seq/SRR81561*.bam
```

For the analysis of DMS-MaPseq samples, it is sufficient to enable the "-m" flag to make rf-count perform mutation counting:

```
$ rf-count -r -f sacCer3_ncbiRefSeq_bt2/sacCer3_ncbiRefSeq.fa
-m -o rf_count_mapseq rf_map_mapseq/SRR392962*.bam
```

Counting requires input SAM/BAM files to be sorted lexicographically by transcript ID, and numerically by position. This is the case when mapping is performed with rf-map. In this case, specifying the "-r" flag reduces the execution time by skipping BAM sorting.

rf-count will produce an RNA Count (RC) file for each processed BAM file. RC files are binary files optimized for fast random access. For the full format specification, please refer to the online documentation (https://rnaframework.readthedocs.io/en/latest/rf-count/#rc-rna-count-format). Further manipulation of RC files is made possible through the use of the **rf-rctools** utility (*see* **Note 3**).

3.4 Reactivity Normalization

Raw counts computed by rf-count need to be normalized in order to use them for data-driven RNA folding. This is performed by the **rf-norm** tool in two steps: calculation of raw scores, followed by normalization of base reactivities to values ranging from 0 to 1 (or greater, depending on the normalization method).

3.4.1 Reactivity Normalization of DMS-Seq

According to Rouskin et al. [7], DMS-seq is analyzed by performing 90% Winsorizing (values above the 95th percentile are set to the 95th percentile and every data point is divided by the value of the 95th percentile) of raw RT stop counts in sliding windows containing 50 A/C residues, by

```
$ for f in rf_count_seq/*.rc; do rf-norm -rb AC -sm 2 -nm 2 -dw
-ec 10 -n 10 -i rf_count_seq/index.rci -t $f; done
```

The “-sm” and “-nm” parameters respectively define the scoring and normalization methods to be used. In this example, scoring method “2” and normalization method “2” respectively correspond to the Rouskin et al., 2014 scoring scheme and 90% Winsorizing. For a complete list of the available scoring and normalization schemes, please refer to the online documentation (https://rnaframework.readthedocs.io/en/latest/rf-norm/).

The “-rb” parameter allows specifying reactive bases (in this case, only As and Cs can be modified by DMS). By default, choice of scoring method “2” enables windowed normalization with both a window size and an offset of 50 nucleotides (respectively controlled by the “-nw” and “-wo” parameters). As DMS can only modify A/C residues and these might not be evenly distributed along a transcript, this can result in windows containing very few reactive bases, leading to normalization artifacts. Use of the “-dw” flag prevents this by making rf-norm dynamically adjust the window size to include 50 A/C residues.

Normalization of lowly covered transcripts is skipped by setting a threshold on the median read coverage through the “-ec” parameter. Additionally, the “-n” parameter sets the coverage threshold for a base to be included in the reactivity profile (reactivities for bases below this coverage will be reported as NaNs). Reactivity profiles are reported in XML format.

3.4.2 Reactivity Normalization of DMS-MaPseq

For DMS-MaPseq experiments, raw reactivities are calculated as the ratio between the number of mismatches at each base, divided by the read coverage of the base [4], by:

```
$ for f in rf_count_mapseq/SRR392962*[^6].rc; do rf-norm -rb
AC -sm 4 -nm 2 -ec 1000 -n 1000 -i rf_count_seq/index.rci -t
$f; done
```

In this case, scoring method “4” is selected, corresponding to the Zubradt et al., 2017 scoring scheme. Also, thresholds for median read coverage and base coverage (“-ec” and “-n”) have been increased to 1000X, as this coverage has been previously proven to be necessary to obtain reliable reactivity profiles with mutational profiling experiments [2].

When an untreated control is available, this can be used to calculate background mutation frequencies, that will be then subtracted from mutation rates in the DMS treated sample. As the Zubradt et al., 2017 scoring scheme does not provide the possibility to account for an untreated control, the Siegfried et al., 2014 scoring scheme [2] (originally introduced for the analysis of SHAPE-MaP data) can be used:

```
$ for f in rf_count_mapseq/SRR392962*[^6].rc; do rf-norm -rb
AC -sm 4 -nm 3 -ec 1000 -n 1000 -i rf_count_seq/index.rci -t $f
-u rf_count_mapseq/SRR3929626.rc; done
```

With the Siegfried et al., 2014 scoring method ("-sm 3"), box-plot normalization is recommended ("-nm 3"). The RC file for the untreated control sample is provided through the parameter "-u".

Optionally, when available, a denatured control sample can also be provided to account for maximum per-base reactivities, through the parameter "-d".

Experiments composed of multiple replicates can be further compared (and combined) using the **rf-correlate** and **rf-combine** tools (*see* **Note 4**).

3.5 Data-Constrained RNA Structure Prediction

Once transcript reactivity profiles have been obtained, these can be used to perform data-driven RNA structure inference. This is usually accomplished by converting base reactivities into pseudo free energy contributions, that are then used to either reward or penalize certain base-pairs [13]. The extent of the contribution of base reactivities to free energies is determined by two parameters, namely the *slope* and the *intercept*. Optimal slope and intercept vary with the specific experimental setup (probing reagent used, library construction strategy, etc.) and therefore it is advisable to empirically determine them.

3.5.1 Grid Search (Jackknifing) of Optimal Folding Parameters

The process by which optimal slope/intercept values are determined is called grid search (or *jackknifing*) and it is performed with the **rf-jackknife** tool. Given experimental probing data and a reference RNA with a known (experimentally-validated) secondary structure, rf-jackknife will perform structure prediction by varying slope/intercept values in a user-defined range. For each predicted structure, the positive predictive value (PPV), sensitivity, and the geometric mean of the two are calculated with respect to the known structure. The slope–intercept pair yielding the structure with the highest PPV/sensitivity can be then used for all the other transcripts.

Here we will show the jackknifing process using the structure of 16S and 23S ribosomal RNAs of *E. coli*, queried by DMS-MaPseq (SRA ID: SRR8172706) [5]. Dataset was processed as follows:

```
$ rf-index -b2 -pb 5
$ rf-map -b2 -mp '--very-sensitive-local' -cq5 20 -bi Eco-
li_rRNA_bt2/reference SRR8172706.fastq
$ rf-count -r -m -f Ecoli_rRNA_bt2/reference.fa rf_map/
SRR8172706.bam
$ rf-norm -sm 4 -nm 3 -rb AC -ec 1000 -n 1000 -i rf_count/
index.rci -t rf_count/SRR8172706.rc
```

Jackknifing is performed by

```
$ rf-jackknife -r ecoli_rRNA.db -x -rp '-nlp -md 600'
SRR8172706_norm/
```

Reference RNA structures are passed through the “-r” parameter (*E. coli* rRNA reference structures can be downloaded from http://www.rnaframework.com/data/publications/Springer2020/ecoli_rRNA.db).

The “-x” parameter enables the use of more relaxed criteria for structure comparison. Specifically, when calculating PPV/sensitivity, a base-pair between nucleotides i and j is considered correctly predicted if the known structure contains a pair between i and j, $i + 1$ or $i - 1$ and j, or i and $j + 1$ or $j - 1$ [13].

rf-jackknife will iteratively call **rf-fold** (see next paragraph) with different slope–intercept value pairs. rf-fold parameters can be adjusted through the “-rp” parameter (in this example, “-nlp” disallows lonely base-pairs and “-md 600” sets the maximum base-pairing distance to 600 nucleotides). Besides reporting the best slope–intercept pair (slope: 2.4; intercept: -0.2), rf-jackknife will generate a set of CSV files containing the PPV/sensitivity (or their geometric mean) for each tested slope–intercept value pair.

The following R code can be used to generate a heatmap from the resulting CSV files (Fig. 1):

```
library(gplots)
library(RColorBrewer)
csv<-read.csv("geometric_mean.csv", sep = ";", check.names =
FALSE)
row.names(csv)<-csv$Mean
csv<-csv[,-1]
csv<-data.matrix(csv)
heatmap.2(csv[nrow(csv):1,], col = rev(brewer.pal(11, "Spec-
tral")), trace = "none", cellnote = round(csv[nrow(csv):1,],
digits = 2), notecol = "black", Rowv = FALSE, Colv = FALSE,
dendrogram = "none", xlab = "Intercept (kcal/mol)", ylab =
"Slope (kcal/mol)", key = FALSE)
```

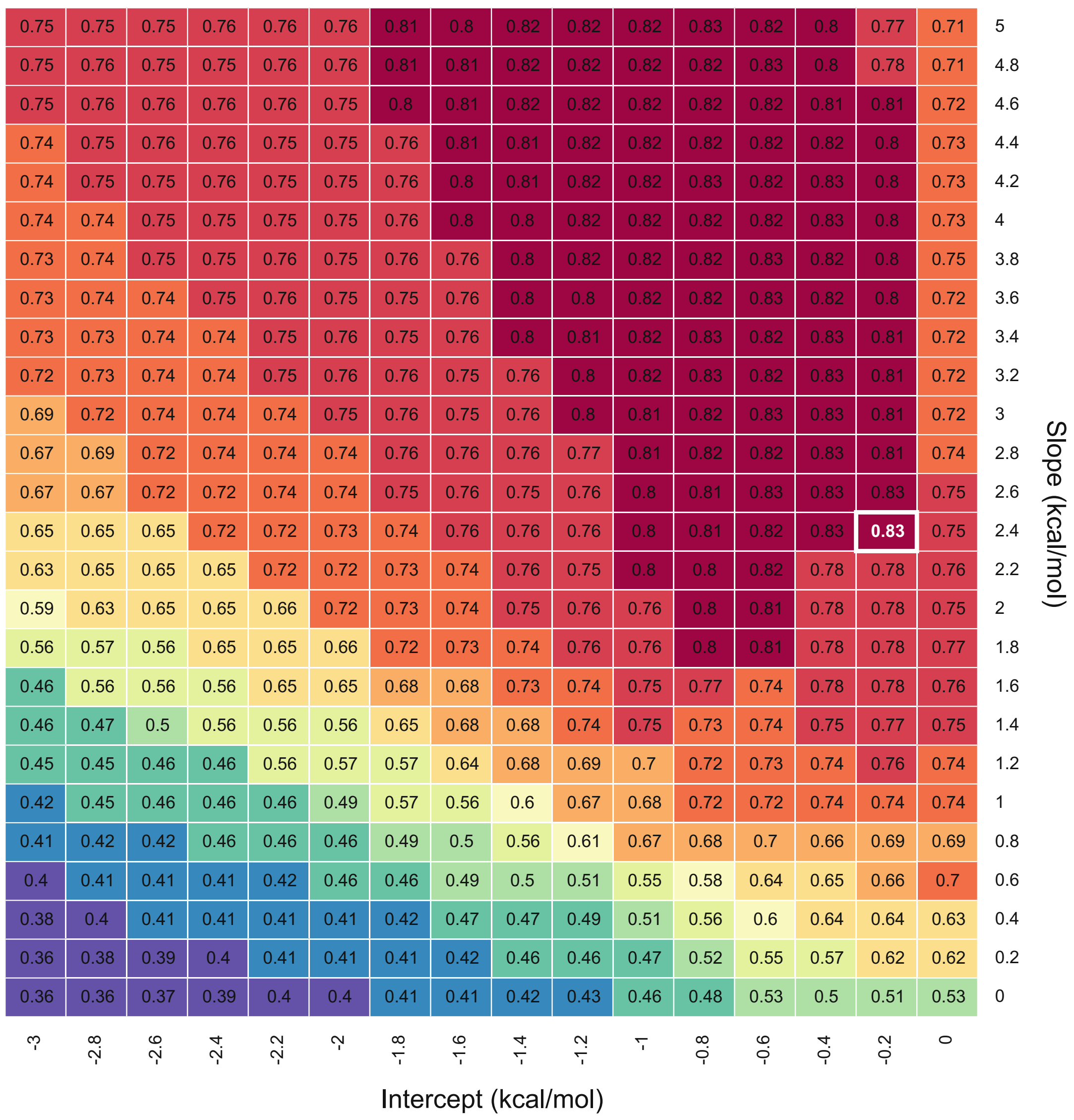
3.5.2 Transcriptome-Wide RNA Structure Inference

RNA secondary structure prediction is performed with the **rf-fold** tool. As an example, transcriptome-wide prediction of RNA structures for the DMS-MaPseq experiment can be performed by

```
$ rf-fold -sl 2.4 -in -0.2 -md 600 -nlp -dp -sh -g DMS-MaP-
seq_merge/
```

By default, rf-fold uses ViennaRNA as the algorithm for RNA structure prediction [14] (this can be changed to RNAstructure [15] through the “-m” parameter). Parameters “-sl” and “-in” respectively set the slope and the intercept values (in this example, the slope–intercept value pair found by jackknifing has been used).

The “-g” flag enables the generation of SVG files depicting base reactivities, base-pairing probabilities, Shannon entropies and the



	-3	-2.8	-2.6	-2.4	-2.2	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
5	0.75	0.75	0.75	0.76	0.76	0.76	0.81	0.8	0.82	0.82	0.82	0.83	0.82	0.8	0.77	0.71
4.8	0.75	0.76	0.75	0.75	0.76	0.76	0.81	0.81	0.82	0.82	0.82	0.82	0.83	0.8	0.78	0.71
4.6	0.75	0.76	0.76	0.76	0.76	0.75	0.8	0.81	0.82	0.82	0.82	0.82	0.82	0.81	0.81	0.72
4.4	0.74	0.75	0.76	0.76	0.75	0.75	0.76	0.81	0.81	0.82	0.82	0.82	0.82	0.82	0.8	0.73
4.2	0.74	0.75	0.75	0.76	0.75	0.75	0.76	0.8	0.81	0.82	0.82	0.83	0.82	0.83	0.8	0.73
4	0.74	0.74	0.75	0.75	0.75	0.75	0.76	0.8	0.8	0.82	0.82	0.82	0.82	0.83	0.8	0.73
3.8	0.73	0.74	0.75	0.75	0.76	0.75	0.76	0.76	0.8	0.82	0.82	0.82	0.83	0.82	0.8	0.75
3.6	0.73	0.74	0.74	0.75	0.76	0.75	0.75	0.76	0.8	0.8	0.82	0.82	0.83	0.82	0.8	0.72
3.4	0.73	0.73	0.74	0.74	0.75	0.76	0.75	0.76	0.8	0.81	0.82	0.83	0.82	0.83	0.81	0.72
3.2	0.72	0.73	0.74	0.74	0.75	0.76	0.76	0.75	0.76	0.8	0.82	0.83	0.82	0.83	0.81	0.72
3	0.69	0.72	0.74	0.74	0.74	0.75	0.76	0.75	0.76	0.8	0.81	0.82	0.83	0.83	0.81	0.72
2.8	0.67	0.69	0.72	0.74	0.74	0.74	0.76	0.76	0.76	0.77	0.81	0.82	0.82	0.83	0.81	0.74
2.6	0.67	0.67	0.72	0.72	0.74	0.74	0.75	0.76	0.75	0.76	0.8	0.81	0.83	0.83	0.83	0.75
2.4	0.65	0.65	0.65	0.72	0.72	0.73	0.74	0.76	0.76	0.76	0.8	0.81	0.82	0.83	**0.83**	0.75
2.2	0.63	0.65	0.65	0.65	0.72	0.72	0.73	0.74	0.76	0.75	0.8	0.8	0.82	0.78	0.78	0.76
2	0.59	0.63	0.65	0.65	0.66	0.72	0.73	0.74	0.75	0.76	0.76	0.8	0.81	0.78	0.78	0.75
1.8	0.56	0.57	0.56	0.65	0.65	0.66	0.72	0.73	0.74	0.76	0.76	0.8	0.81	0.78	0.78	0.77
1.6	0.46	0.56	0.56	0.56	0.65	0.65	0.68	0.68	0.73	0.74	0.75	0.77	0.74	0.78	0.78	0.76
1.4	0.46	0.47	0.5	0.56	0.56	0.56	0.65	0.68	0.68	0.74	0.75	0.73	0.74	0.75	0.77	0.75
1.2	0.45	0.45	0.46	0.46	0.56	0.57	0.57	0.64	0.68	0.69	0.7	0.72	0.73	0.74	0.76	0.74
1	0.42	0.45	0.46	0.46	0.46	0.49	0.57	0.56	0.6	0.67	0.68	0.72	0.72	0.74	0.74	0.74
0.8	0.41	0.42	0.42	0.46	0.46	0.46	0.49	0.5	0.56	0.61	0.67	0.68	0.7	0.66	0.69	0.69
0.6	0.4	0.41	0.41	0.41	0.42	0.46	0.46	0.49	0.5	0.51	0.55	0.58	0.64	0.65	0.66	0.7
0.4	0.38	0.4	0.41	0.41	0.41	0.41	0.42	0.47	0.47	0.49	0.51	0.56	0.6	0.64	0.64	0.63
0.2	0.36	0.38	0.39	0.4	0.41	0.41	0.41	0.42	0.46	0.46	0.47	0.52	0.55	0.57	0.62	0.62
0	0.36	0.36	0.37	0.39	0.4	0.4	0.41	0.41	0.42	0.43	0.46	0.48	0.53	0.5	0.51	0.53

Fig. 1 Heatmap showing the geometric mean of PPV/sensitivity for different slope–intercept value pairs, using DMS-MaPseq data for *E. coli* 16S/23S rRNAs. The optimal slope–intercept value pair is highlighted in white

secondary structure model for each analyzed transcript (Fig. 2). The "-dp" and "-sh" flags respectively enable the generation of base-pairing probability dot-plots and Wiggle tracks containing per-base Shannon entropies. These files can be further loaded into Integrative Genomics Viewer (http://software.broadinstitute.org/software/igv/) for visualization. Regions of low Shannon entropy, high base-pairing probability and low DMS reactivity can be used to identify high-confidence helices [2, 5].

Structures predicted by rf-fold are by default pseudoknot-free. Prediction of structures including pseudoknots can be enabled

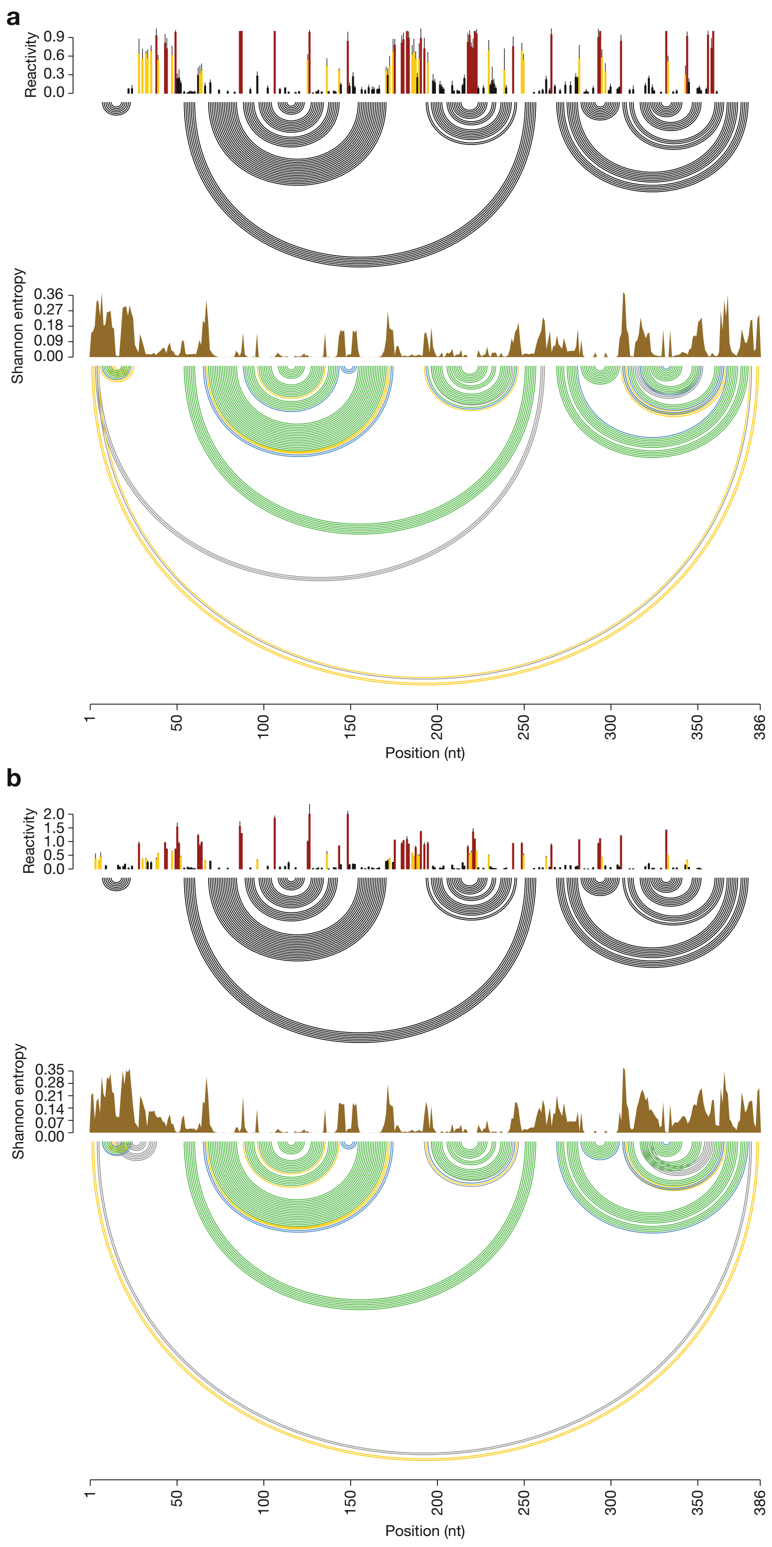


Fig. 2 Reactivity profile, secondary structure model, Shannon entropy and base-pairing probabilities for *S. cerevisiae* snoRNA snR37 (NR_132195). Structure predictions performed using either (**a**) DMS-seq or (**b**) DMS-MapSeq data are shown

through the "-pk" flag. It is worth noting that use of this option will result in considerably longer computation times.

rf-fold allows for extensive customization. In this example, only the essential parameters were presented. For a detailed list of all the available options, please refer to the online documentation (https://rnaframework.readthedocs.io/en/latest/rf-fold/).

4 Notes

1. When inspection by FastQC reveals the presence of low-quality bases at the 5′ end of reads, it might be beneficial to trim them in order to facilitate mapping. Quality trimming of bases controlled by the "-cq5" parameter is *dynamic*; thus, an arbitrary number of bases can be trimmed from each read. In RT stop-based experiments, the position immediately preceding the start mapping coordinate of a read corresponds to the site of modification by the probing reagent. It is therefore essential to keep track of the exact number of bases trimmed from the 5′ end of each read. This is however not possible when *dynamic* quality trimming is performed. To this end, a static trimming of a user-defined number of bases from the 5′ side of all reads can be performed using the "-b5" parameter.
2. If static 5′ end read trimming has been performed during read mapping (through the "-b5" parameter), it is necessary to specify to rf-count the number of trimmed bases, through the "-t5" parameter. Alternatively, only in case read mapping has been performed with Bowtie, it possible to use the "-fh" flag to make rf-count automatically detect the number of trimmed 5′ bases for each sample from the SAM/BAM header.
3. Inspection of RC files can be performed using **rf-rctools**, by

```
$ rf-rctools view <RC file>
```

rf-rctools also allows merging multiple RC files, by

```
$ rf-rctools merge <RC file #1> <RC file #2> ... <RC file #n>
```

In our example, DMS-MaPseq data for biological replicate #2 is provided in two independent datasets (SRR3929622 and SRR3929623). As these datasets belong to the same biological replicate, their RC files can be merged by

```
$ rf-rctools merge rf_count_mapseq/SRR392962[23].rc -o
rf_count_mapseq/SRR392962_2_3.rc
```

4. For experiments containing multiple biological replicates, transcript level (and experiment level) pairwise Pearson correlations can be assessed with the **rf-correlate** tool, by

```
$ rf-correlate -m 0.8 <XML folder #1> <XML folder #2>
```

The "-m" parameter sets the threshold for the minimum number of covered bases needed to evaluate the correlation between two replicates. When comprised between 0 and 1, this value is interpreted as a fraction of the number of reactive bases of a transcript.

Replicates can be further merged with **rf-combine**, by

```
$ rf-combine -m 0.8 <XML folder #1> <XML folder #2> ... <XML
folder #n>
```

A new folder containing merged XML reactivity profiles for transcripts present in all the provided experiments will be generated. Only transcripts whose correlations exceed a user-defined threshold (0.7 by default, controlled through the "-c" parameter) will be merged.

In our example, the four biological replicates of DMS-seq and the two biological replicates of DMS-MaPseq can be respectively merged by

```
$ rf-combine -m 0.8 -o DMS-seq_merge/ SRR81561*_norm/
$ rf-combine -m 0.8 -o DMS-MaPseq_merge/ SRR392962*_norm/
```

Acknowledgments

This work was supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO; OCENW.XS3.044) and by funding of the University of Groningen (Groningen, the Netherlands) and the Groningen Biomolecular Sciences and Biotechnology Institute (GBB) to D.I.

References

1. Incarnato D, Oliviero S (2017) The RNA epistructurome: uncovering RNA function by studying structure and post-transcriptional modifications. Trends Biotechnol 35:318–333
2. Siegfried NA, Busan S, Rice GM et al (2014) RNA motif discovery by SHAPE and mutational profiling (SHAPE-MaP). Nature Methods. 11:959–965
3. Homan PJ, Favorov OV, Lavender CA et al (2014) Single-molecule correlated chemical probing of RNA. Proc Natl Acad Sci USA 111:13858–13863
4. Zubradt M, Gupta P, Persad S et al (2017) DMS-MaPseq for genome-wide or targeted RNA structure probing in vivo. Nat Methods 14:75–82

5. Simon LM, Morandi E, Luganini A et al (2019) In vivo analysis of influenza A mRNA secondary structures identifies critical regulatory motifs. Nucleic Acids Res 36:3960
6. Incarnato D, Morandi E, Simon LM et al (2018) RNA framework: an all-in-one toolkit for the analysis of RNA structures and post-transcriptional modifications. Nucleic Acids Res 46:e97–e97
7. Rouskin S, Zubradt M, Washietl S et al (2014) Genome-wide probing of RNA structure reveals active unfolding of mRNA structures in vivo. Nature 505:701–705
8. Langmead B, Trapnell C, Pop M et al (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10:R25
9. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9:357–359
10. Ding Y, Tang Y, Kwok CK et al (2014) In vivo genome-wide profiling of RNA secondary structure reveals novel regulatory features. Nature 505:696–700
11. Lucks JB, Mortimer SA, Trapnell C et al (2011) Multiplexed RNA structure characterization with selective 2'-hydroxyl acylation analyzed by primer extension sequencing (SHAPE-Seq). Proc Natl Acad Sci USA 108:11063–11068
12. Incarnato D, Neri F, Anselmi F et al (2014) Genome-wide profiling of mouse RNA secondary structures reveals key features of the mammalian transcriptome. Genome Biol 15:491
13. Deigan KE, Li TW, Mathews DH et al (2009) Accurate SHAPE-directed RNA structure determination. Proc Natl Acad Sci 106:97–102
14. Lorenz R, Bernhart SH, Höner Zu Siederdissen C et al (2011) ViennaRNA Package 2.0. Algorith Mol Biol 6:26
15. Reuter JS, Mathews DH (2010) RNAstructure: software for RNA secondary structure prediction and analysis. BMC Bioinformatics 11:129

Chapter 6

Using RNentropy to Detect Significant Variation in Gene Expression Across Multiple RNA-Seq or Single-Cell RNA-Seq Samples

Federico Zambelli and Giulio Pavesi

Abstract

RNA-Seq has become the de facto standard technique for characterization and quantification of transcriptomes, and a large number of methods and tools have been proposed to model and detect differential gene expression based on the comparison of transcript abundances across different samples. However, state-of-the-art methods for this task are usually designed for pairwise comparisons, that is, can identify significant variation of expression only between two conditions or samples. We describe the use of RNentropy, a methodology based on information theory, devised to overcome this limitation. RNentropy can thus detect significant variations of gene expression in RNA-Seq data across any number of samples and conditions, and can be applied downstream of any analysis pipeline for the quantification of gene expression from raw sequencing data. RNentropy takes as input gene (or transcript) expression values, defined with any measure suitable for the comparison of transcript levels across samples and conditions. The output consists of genes (or transcripts) exhibiting significant variation of expression across the conditions studied, together with the samples in which they result to be over- or underexpressed. RNentropy is implemented as an R package and freely available from the CRAN repository. We provide a detailed guide to the functions and parameters of the package and usage examples to demonstrate the software capabilities, also showing how it can be applied to the analysis of single-cell RNA sequencing data.

Key words Next-generation sequencing, RNA-seq, Single cell RNA-seq, Transcriptome quantification, Differential gene expression, Tissue-specific genes, Marker genes

1 Introduction

Next generation sequencing technologies applied to RNA molecules (RNA-Seq) have become the de facto standard for the characterization and quantification of transcriptomes [1, 2], also at the single-cell level [3]. RNA-Seq permits to estimate expression levels either by first assembling transcripts and employing sequence reads for quantification of the abundance of each transcript [4, 5], or by mapping reads directly on a reference genome, providing expression estimates for available gene annotations [6, 7]. More recent approaches also bypass the need for mapping sequence reads on the genome, resulting in a significantly faster quantification step compared to alignment-based methods while maintaining comparable reliability [8, 9].

The comparison of the estimated expression levels across samples and replicates is then the next logical step, in order to identify those genes which, following a significant variation of abundance of the transcripts they produce, can lead to an explanation of the observed phenotypic differences among the conditions being investigated. The most challenging point of this step is to define and model correctly significant variation from a statistical point of view, since a certain degree of data variability has to be expected due both to the inherent nature of biological systems [10] and to the technical noise introduced by experimental procedures [11]. Thus, a fundamental challenge posed by RNA-seq data is how to normalize correctly (i.e., make comparable) the estimated expression levels in the samples, taking into account different transcript lengths, different library sizes, and biases introduced by the sequencing procedure [12], and then how to model the variability of normalized expression values. These issues have led to the proliferation of methods tackling normalization and/or differential expression analysis using different approaches and statistical frameworks [13–15].

In general, the most commonly used methods for differential gene expression analysis have been designed for pairwise comparisons. That is, their approach is tailored to detect significant variation of gene expression levels when contrasting data from two biological samples or conditions at a time [16]. Simultaneous comparisons on multiple samples and conditions are made problematic by the difficulty of conceiving a unique definition for *condition(s)-specific* genes. An example are tissue specificity metrics, that have been introduced to tackle this issue when comparing samples from different adult tissues for the identification of tissue-specific genes, that can be adapted to other multicondition comparisons [17]. However, these measures account only for relative variation of expression, thus potentially introducing biases for lowly expressed genes, that tend to show more intrinsic variability with respect to highly expressed ones. Furthermore, technical [18] and biological [19] variability still need to be correctly assessed across all the conditions.

For this task, we introduced RNentropy, an approach designed to tackle multisample RNA-Seq differential expression without relying on multiple pairwise comparisons [20]. RNentropy can work simultaneously on any number of conditions and replicates to detect genes whose expression level changes significantly across the samples, that is, genes that are under- or overexpressed in one or more conditions with respect to their average expression level. The input of RNentropy consists of a table of the normalized gene expression values associated with genes for each sample in the assay. The normalized values can be obtained using any of the several methods available for this task (e.g., [4–9]). The use of commonly accepted measures (e.g., FPKM, RPKM, TPM, TMM) for the normalization of gene expression values is recommended, and the performance of RNentropy seems to be robust in this respect, with only minor differences on the final output depending on the measure adopted [20] (*see* **Note 1**). The output of RNentropy reports for each gene the sample(s) where it can be considered over- or underexpressed, if any, together with the corresponding significance *p*-value(s). Thus, RNentropy represents a suitable tool for differential gene expression analysis for RNA-Seq assays involving multiple conditions, in particular when it is more appropriate to retrieve genes of interest considering all the samples simultaneously instead of resorting to the design of pairwise comparisons followed by intersections of differentially expressed gene sets. The ability to detect significant changes in expression across any number of samples makes RNentropy a perfectly suitable approach also for the analysis of single-cell RNA-Seq [21].

2 Materials

RNentropy's home page can be accessed at http://www.beacon-lab.it/RNentropy. Two implementations of RNentropy do exist. The first one is a C++ implementation: executables for 64-bit Linux platforms and source code can be downloaded directly from the home page together with an installation guide and manual with usage examples. The second implementation of RNentropy comes in the form of an R package available on the CRAN repository at https://cran.r-project.org/web/packages/RNentropy/index.html. For the sake of brevity and clarity, from here on, we introduce and describe the functions of the R implementation only, which is also more feature-rich. The user interested in using the more basic C++ implementation of RNentropy can easily map most of the concepts herein described to that version.

RNentropy requires R (release 2.10 or newer) and can be used on any operating system for which the R software environment is available (a wide variety of UNIX platforms, Windows and macOS). No other dependencies (e.g., other R packages) are required. Datasets presented here as usage examples are already included in the package.

3 Methods

3.1 The Algorithm

We present here a summary of the RNentropy algorithm. A more comprehensive description can be found in [20].

3.1.1 Shannon's Entropy

The approach to finding differentially expressed genes is inspired by information theory and is based on *Shannon's entropy* (from now on referred to simply as entropy). Its application to the problem of differential expression analysis was indeed first suggested in [22]. At a conceptual level, entropy can be defined as the "amount of information" contained in a random variable (*see* [23] for a gentle introduction). When a discrete random variable can have m possible outcomes, entropy is defined from the frequency with which each outcome is observed:

$$H = -\sum_{k=1}^{m} f_k \log f_k$$

where f_k is the frequency with which the random variable assumes the kth value in the observations. We can use e as the logarithm base. When the variable always assumes the same value k across all the observations, then $f_k = 1$ and its entropy will be 0. On the other extreme of the spectrum, all the m outcomes are observed with the same frequency $f_k = 1/m$ and the entropy is maximum.

3.1.2 Entropy and Gene Expression Levels: The Global Sample Specificity Test

Let T be a gene and $t_1 \ldots t_m$ its normalized expression levels in m samples. Let t_{cum} be the cumulative expression value of T given by the sum of all the values t_i with i going from 1 to m, and $f_i = t_i/t_{\mathrm{cum}}$. If we now apply the entropy formula to the f_i values, the entropy $H(T)$ will be 0 when all the RNA produced by T comes from a single sample, that is, the gene is expressed only in a single condition, while will reach its maximum value of 1 when T has the same expression value across all the samples.

Given an uniform expected value of $b_i = 1/m$, RNentropy evaluates the variability of gene T with a G-test (goodness of fit test) [24], based on the computation of relative entropy with respect to the b_i values. The test computes, given t_i and the random distribution with expected frequencies b_i, the probability to obtain the following G(T) value by chance:

$$G(T) = 2\sum_{i=1}^{m} t_i \log \frac{f_i}{b_i} = 2WRE(T).$$

$G(T)$ follows a chi-square distribution with m-1 degrees of freedom. We call this the *global sample specificity test*. For each gene, we assume a uniform background distribution $b_i = 1/m$, but, in principle, the same framework can be applied to identify significant deviations from any other distribution of b_i. The

Benjamini–Hochberg [25] correction is employed to correct the *p*-values obtained for multiple testing. Suitable thresholds of 0.01 or 0.05 for the corrected *p*-values can be employed to single out genes whose expression has a significant variation from the uniform background distribution.

3.1.3 Identification of Significant Samples: The Local Sample Specificity Test

Once the test just described shows that a gene T deviates significantly from the expected distribution of its expression levels across samples, we can single out in which samples it is over- or under-expressed using a variation of the same test, in which the expression of T in each sample is compared with its expression in the other $m - 1$ samples. That is, the sum will consist only of two terms, with expected values t_{cum}/m for the sample considered and $(m - 1)$ t_{cum}/m for the others. We call this the *local sample specificity test.*

Thus, a gene T that passes the global specificity test is considered to be significantly over- (sample expression higher than average expression) or under- (sample expression lower than average expression) expressed in those samples where the local specificity test yields a *p*-value lower than a given threshold.

3.1.4 Taking Replicates into Account

RNentropy does not attempt to explicitly estimate the variance of the expression values for the different replicates (technical or biological) of the same condition but performs a test on each replicate. That is, when n samples are defined as replicates of the same condition c, a gene will be considered over- or underexpressed in condition c if all the n replicates pass the local specificity test *p*-value threshold. The only difference is that, when performing the local specificity test for a replicate, the expected expression value is computed without taking into account the expression values of all the other replicates for the same condition. This approach is very stringent since it requires all replicates of the same condition to pass the local test, that is to be significantly expressed above (or below) the gene average. For the correctness of the results, thus, the definition of "replicates" is of the utmost importance. Our advice is to define replicates for RNentropy only for technical replicates or anyway cases where the variability of replicates is due mostly to experimental and not biological variation (*see* **Note 2**).

3.1.5 Evaluating Similarity Among Conditions

After genes displaying a significant variation of expression across samples have been identified, it becomes possible to compare the different conditions in order to single out those that are more similar and those that diverge the most in terms of gene expression. The approach of RNentropy to this task relies on *pointwise mutual information* (PMI) [26]. Given an assay with g genes and c conditions (samples), a matrix M with g rows and c columns is first built as follows: $M_{(x,y)} = 0$ if gene x is neither over- nor under-expressed in condition y, $M_{(x,y)} = 1$ if gene x is overexpressed in condition *y*, and $M_{(x,y)} = -1$ when gene *x* is underexpressed in condition *y* according to both global and local tests. PMI on overexpressed genes between any two conditions (columns) *i* and *j* can be defined as

$$\mathrm{PMI}(i,j) = \log_2 \frac{f(i,j)}{f(i)f(j)},$$

where *f(i)* is the fraction of rows with a value of *1* for the column *i* of *M*, *f(j)* the fraction of rows with a value of *1* for column *j* of *M*, and *f(i,j)* the fraction of rows with a value of *1* in both columns. Positive PMI values indicate that the number of shared overexpressed genes between the two conditions is greater than expected, vice versa, negative PMI values point to anticorrelation between the two conditions.

3.2 Using RNentropy

RNentropy is implemented as an R package containing both functions and sample data. We will describe how to interact with the software and employ the included data to provide examples. Basic knowledge of the R environment and functions is recommended in order to use the RNentropy package.

The *install.packages* function of R can be employed to install RNentropy on the local R library without the need for manually downloading the software package. From within the R console type:

```
install.packages("RNentropy")
```

The installation needs to be performed only once. When it ends (in a matter of seconds), it becomes possible to load the RNentropy package within the current R workspace using the following command:

```
library("RNentropy")
```

The same command can be used to load RNentropy in different R workspaces.

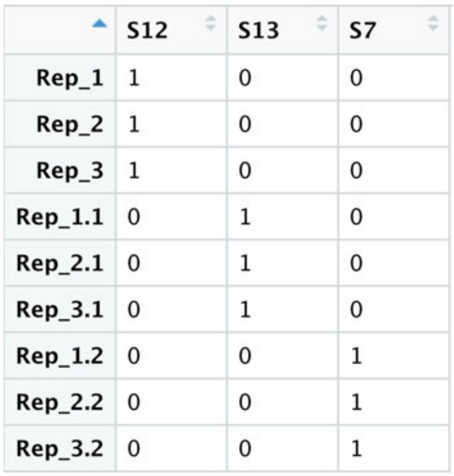
3.2.1 Input

The input of RNentropy consists of an R *data frame* object containing the normalized gene expression values (expression table from now on) and a binary *matrix* object describing the experimental design (design matrix from now on). The design matrix is a table with information about which condition each sample belongs to, that is, used to define which samples should be considered to be replicate experiments of the same condition. The expression table includes a row for each gene or transcript and a column for each sample. Univocal gene names or transcript IDs must be used to label each row. When using transcript IDs as labels, it is possible to have an additional column reporting the names (or symbols) of the genes to which each transcript belongs to. Expression levels can in principle be defined using any of the commonly used normalized metrics (e.g., FPKM, RPKM, TPM, TMM) since RNentropy seems to provide consistent results independently of the measure employed [20] (*see* **Note 1**). Raw counts should instead be avoided since RNentropy does not perform any normalization step by itself.

The design matrix has a row for each sample and a column for each condition. For each row, a value of *1* is assigned to the condition (column) for which the row is a replicate. All the other columns on the same row contain *0*'s. A simple way to check that no replicates have been assigned wrongly to less or more than one condition is to ensure that the sum of each row is precisely *1*. In the same way, the sum of the values in each column must match the number of replicates in the corresponding condition. Finally, the number of replicates should match the number of numeric columns in the expression table. If no replicates are defined (*see* also Subheading 3.1.4) (*see* **Note 2**), thus only one replicate for each condition is available, it is possible to omit the design matrix altogether. In this case, RNentropy will use a default square matrix where each sample is considered to represent a different condition.

The following command can be used to load the "Brain" illustrative dataset provided by the RNentropy package and its corresponding experimental design matrix (Fig. 1):

```
data("RN_Brain_Example_tpm", "RN_Brain_Example_design")
```



	S12	S13	S7
Rep_1	1	0	0
Rep_2	1	0	0
Rep_3	1	0	0
Rep_1.1	0	1	0
Rep_2.1	0	1	0
Rep_3.1	0	1	0
Rep_1.2	0	0	1
Rep_2.2	0	0	1
Rep_3.2	0	0	1

Fig. 1 An example of design matrix. The matrix shown for the "Brain" dataset of RNentropy describes three conditions constituted by three different individuals labelled S12, S13, and S7, with three replicates each. More details on this dataset can be found in [20] and accompanying supplementary material

RN_Brain_Example_tpm is an expression table with over 78,000 rows and 10 columns. Each row corresponds to a transcript in the UCSC gene annotation on the human genome (version hg19). The first column of the table reports the gene symbol associated with the transcripts of each row. The other nine columns contain expression values for samples coming from three different individuals, with three technical replicates each, defined as transcripts per million (TPM). The first step of the RNentropy workflow consists in running the global and local sample specificity tests on the input.

3.2.2 Running RNentropy: the RN_calc function

The *RN_calc* function is used to run both the global and the local sample specificity tests of RNentropy. The syntax is as follows:

```
RN_calc(expression_table, design_matrix)
```

Since the returned value (results container from now on) is an R *list* object composed of six elements, it is convenient to assign the return value to a new variable. For example, to run RNentropy on the "Brain" dataset type:

```
Brain_Results <- RN_calc(RN_Brain_Example_tpm, RN_Brain_Exam-
ple_design)
```

Depending on the number of samples (columns of the expression table), the processing time may vary from a few seconds to a few minutes. The following objects are found within the results container:

- **expr:** the original expression table.
- **design:** the original design matrix.
- **gpv:** a *numeric vector* reporting the $-\log_{10}$ of the *p*-values obtained by each gene (or transcript) from the global sample specificity test. Those genes for which it is not possible to compute this value since their expression is zero in all the conditions get a gpv of 0 (corresponding to a *p*-value of 1). Genes with high gpv values are the ones passing the global specificity test and therefore have an overall high variability of expression across conditions.
- **lpv:** a *data frame* reporting the $\log_{10}$ of the *p*-values obtained by each gene (or transcript) in each condition from the local sample specificity test. Again, in those genes for which it is not possible to compute gpv, lpv is set to zero in all the conditions. For any sample, the lpv value of a gene is negative when underexpressed in that sample w.r.t. its average expression (obtained excluding the other replicates of the same condition of the sample, *see* Subheading 3.1.4), positive otherwise.

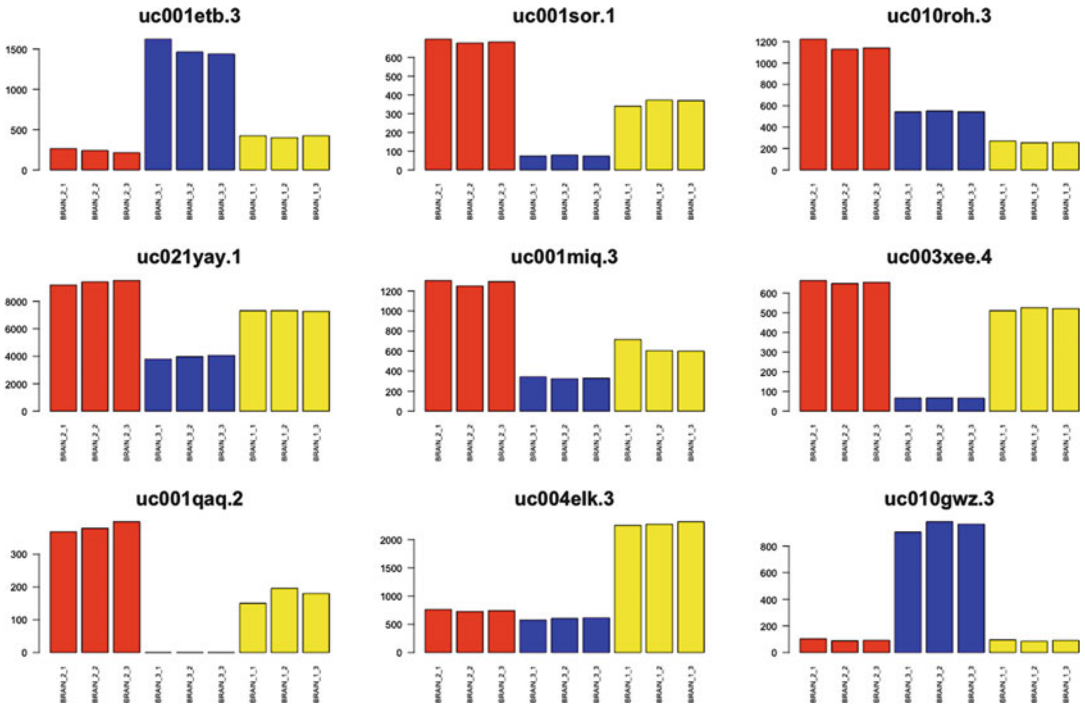


Fig. 2 Expression values (in TPM) of transcripts passing the global and local sample specificity tests of RNentropy in the "Brain" dataset. Red, blue, and yellow bars correspond to the three replicates for each of the three individuals studied

- **res:** the comprehensive results *data frame* obtained joining together expr, gpv, and lpv.
- **c_like:** is like res, but formatted as the output obtained from the C++ implementation of RNentropy. It can be safely ignored.

Genes obtaining high gpv values are those with the most significant variation of expression across conditions (Fig. 2). For each gene, the single samples that diverge the most from the average expression obtain in turn the highest lpv absolute values.

3.2.3 Selecting Significant Genes Differentially Expressed Across Conditions

The *RN_select* function can be employed to filter and output only those genes (or transcripts) obtaining significant values according to both gpv and lpv values. The syntax of *RN_select* is as follows:

```
RN_select(Results, gpv_t = 0.01, lpv_t = 0.01, method = "BH")
```

Results is the return value of the *RN_calc* function. Since the returned value consists of the original *Results* list now containing two additional objects, it is convenient to use it as in the following example:

```
Brain_Results <- RN_select(Brain_Results)
```

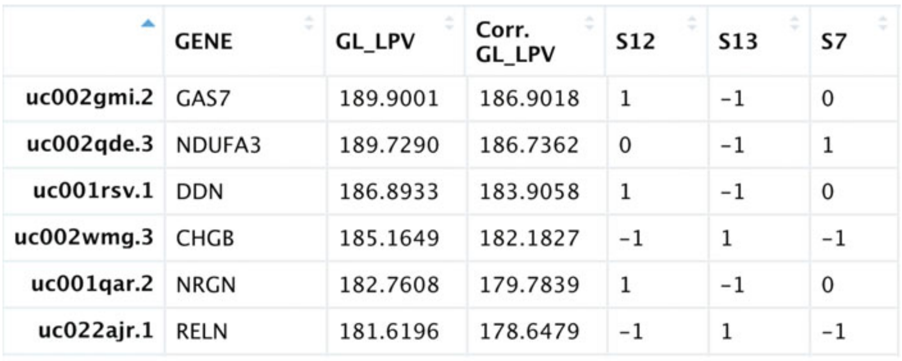
The *gpv_t* argument sets the threshold for the global sample specificity test *p*-value for a gene to be considered differentially expressed across samples and has a default value of 0.01. *The lpv_t* argument sets the threshold for the local sample specificity test *p*-value for a gene to be considered differentially expressed in a sample and also has a default value of 0.01. The *method* argument selects the *p*-value correction method for multiple testing that will be employed and defaults to Benjamini and Hochberg (FDR), that we recommend. Other available methods include *"bonferroni"* for the eponymous method and "*BY*" for Benjamini and Yekutieli. The complete list of multiple test correction methods is available consulting the manual for *p.adjust.methods* in the R documentation.

The following two additional objects can be found in the results container returned by *RN_select*:

- **gpv_bh:** is a *numeric vector* similar to **gpv** (*see* Subheading 3.2.2) but with values corrected for multiple testing. This means that nonzero gpv_bh values will be lower than their corresponding gpv values.
- **selected:** is a *data frame* and constitutes the main output of *RN_Select* (Fig. 3). Only genes (or transcripts) with a corrected global sample specificity test *p*-value lower than gpv_t are reported in this table. The two columns "GL_LPV" and "Corr. GL_LPV" report the gpv and gpv_bh values, respectively, for the selected genes. One additional column is then present for each of the conditions defined in the design matrix. Cells in those columns can assume one of four flag values: *1*,*0*, *−1* and *NA*. A value of *1* means that all the replicates for that condition pass the local specificity test *p*-value threshold and have expression values higher than the average expression; thus, the gene can be considered overexpressed in that condition. A value of *−1* means that all the replicates for that condition pass the local specificity test *p*-value threshold and have expression values lower than the average expression; thus, the gene can be considered underexpressed in that condition. A value of *0* means that at least one of the replicates does not pass the local specificity test *p*-value threshold, and therefore the condition cannot be considered over- and neither underexpressed. Finally, the *NA* flag means that the local specificity test *p*-values of the different replicates for that condition are not consistent. That is, at least one replicate seems to be overexpressed, and at least another one for the same condition results to be underexpressed.

3.2.4 Assessing Similarity Among Conditions Using RN_pmi

The *RN_pmi* function is used to compute the pointwise mutual information scores between every pair of conditions, starting from the genes over- or underexpressed in each. The function can be used with the following syntax:



	GENE	GL_LPV	Corr. GL_LPV	S12	S13	S7
uc002gmi.2	GAS7	189.9001	186.9018	1	-1	0
uc002qde.3	NDUFA3	189.7290	186.7362	0	-1	1
uc001rsv.1	DDN	186.8933	183.9058	1	-1	0
uc002wmg.3	CHGB	185.1649	182.1827	-1	1	-1
uc001qar.2	NRGN	182.7608	179.7839	1	-1	0
uc022ajr.1	RELN	181.6196	178.6479	-1	1	-1

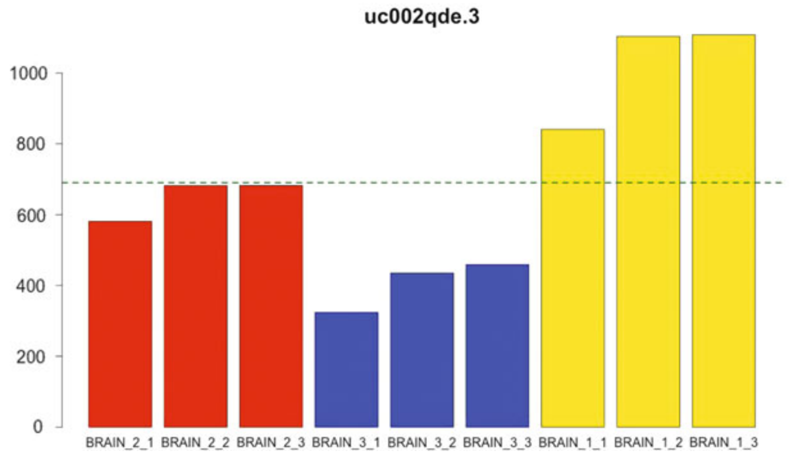


Fig. 3 Top: examples from the "selected" table returned by the *RN_Select* function applied on the "Brain" dataset. For each transcript, the table reports the transcript and gene IDs, the global sample specificity test *p*-value (GL_LPV), the multiple testing corrected global sample specificity test *p*-value (Corr. GL_LPV). An additional column for each condition, in this case, the three individuals S12, S13, and S7, flags the over- (1) or under- (−1) expression status of that condition. For example, transcript "uc002qde.3" results to be over-expressed in S7, underexpressed in S13, and nether over- or underexpressed in S12. Bottom: plot of the expression levels of transcript "uc002qde.3" (in TPM) across the three individuals. All the three replicates from the third individual (yellow bars) are consistently overexpressed, while replicates from the second individual (blue bars) are consistently underexpressed. The green horizontal line marks the uc002qde.3 average expression level

```
RN_pmi(Results)
```

Results is the object returned by *RN_calc* or *RN_select*. In the former case, *RN_select* will be automatically run on *Results*. For example, we can use the *RN_pmi* function to quickly check if the replicates for the same individual of the "Brain" dataset are consistent among them:

```
Brain_Results <- RN_pmi(RN_calc(RN_Brain_Example_tpm))
```

In this case, we are applying the *RN_calc* function without explicitly providing the experimental design matrix. That is, we are asking RNentropy to consider all the samples independently since we want to ascertain how the different replicates correlate among them. We expect replicates from the same individual to be more similar among them than when compared with replicates of the other two individuals.

The object returned by *RN_pmi* contains the same objects of the one returned by *RN_select* and the following two additional objects:

- **pmi:** a numeric matrix with pointwise mutual information for all the possible pairwise comparisons among conditions. Positive values point to correlation, negative values point to anticorrelation, values close to zero means the two conditions are independent.
- **npmi:** as above but using normalized pointwise mutual information. Values close to *1* point to strong correlation, values close to -1 point to strong anticorrelation, values close to *0* means the two conditions are independent.

Using *heatmap()* or a similar function of R or specialized plotting packages on pmi (or npmi) it is possible to cluster the conditions (or samples) and draw the resulting heatmaps (Fig. 4).

3.2.5 Usage Example: Characterization of Gene Expression Specificity in Mouse Oligodendrocytes

A thorough characterization of the transcriptome of several mouse brain cell types has been presented in [27] and the results collected in a user-friendly database. In particular, the study focuses on gene expression in different cell types: neurons, astrocytes, oligodendrocytes, microglia, and endothelial cells. Oligodendrocytes are in turn split into three subtypes corresponding to three different maturation stages, namely, precursor (OPC), newly formed (NFO), and myelinating (MO). Two biological replicates obtained from pooled purified cell populations were sequenced for each cell type, and their expression values used to compute confidence intervals for the expression levels, defined as FPKMs. Genes with unreliable expression estimates were then excluded from further processing. We will describe how RNentropy can be applied to this dataset (BarresLab from now on) to find overexpressed genes in oligodendrocytes and characterize the markers of the different stages of maturation.

Use the following command to load the BarresLab dataset into the R environment

```
data("RN_BarresLab_FPKM")
```

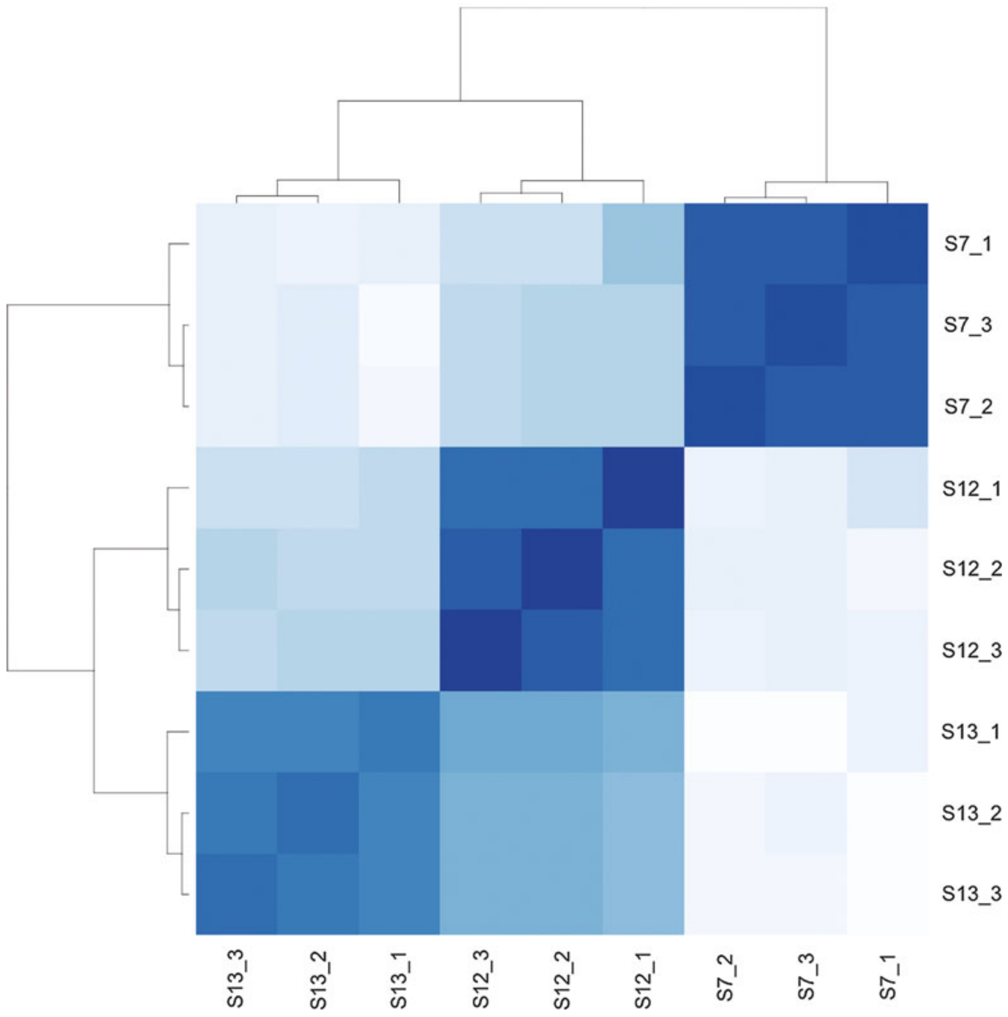


Fig. 4 Heatmap obtained using the R *heatmap* function on the pointwise mutual information matrix of the brain example dataset. The plot clearly shows the high correlation among the replicates of each individual

The dataset includes expression levels for genes with FPKM>0 in at least one sample (12,978 in all) in seven different cell types: Astrocyte, Neuron, OPC, NFO, MO, Microglia, and Endothelial. FPKM expression levels have been estimated in advance, merging two replicates for cell type. As a consequence, there is just one column for each cell type, and therefore the definition of replicates through an experimental design matrix for RNentropy is not necessary. Notice also that, since RNentropy does not need to estimate the variance of gene expression levels across replicates, there is no

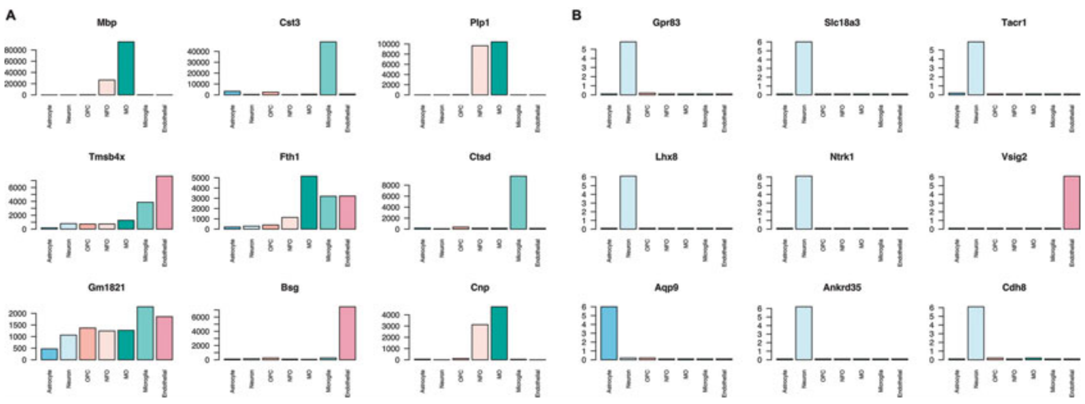


Fig. 5 Expression level (FPKM) of genes with (**a**) highest and (**b**) lowest expression identified by RNentropy as being overexpressed in at least one sample across the seven cell types of the BarresLab dataset. Genes have been ranked by average expression over the seven samples

particular problem in using directly these values instead of the original expression values for each replicate.

To use RNentropy on the data, run the *RN_calc* and *RN_select* functions as follows:

```
Barres_Results <- RN_calc(RN_BarresLab_FPKM)
Barres_Results <- RN_select(Barres_Results)
```

We can verify that 6875 genes pass the RNentropy global sample specificity *p*-value test with:

```
nrow(Barres_Results$selected)
```

In other words, those are the genes showing significant variation of expression in one or more cell types (Fig. 5).

To extract genes whose expression characterizes one or more of the oligodendrocyte subtypes, proceed as follows:

```
colnames(Barres_Results$selected)[3:9] <- colnames(Barres_Results$expr)
Oligodendrocyte_genes <- row.names(Barres_Results$selected[(Barres_Results$selected$OPC == 1
                      | Barres_Results$selected$NFO == 1
                      | Barres_Results$selected$MO == 1)
                      & Barres_Results$selected$Astrocyte != 1
                      & Barres_Results$selected$Neuron != 1
                      & Barres_Results$selected$Microglia != 1
                      & Barres_Results$selected$Endothelial != 1
                       ,])
```

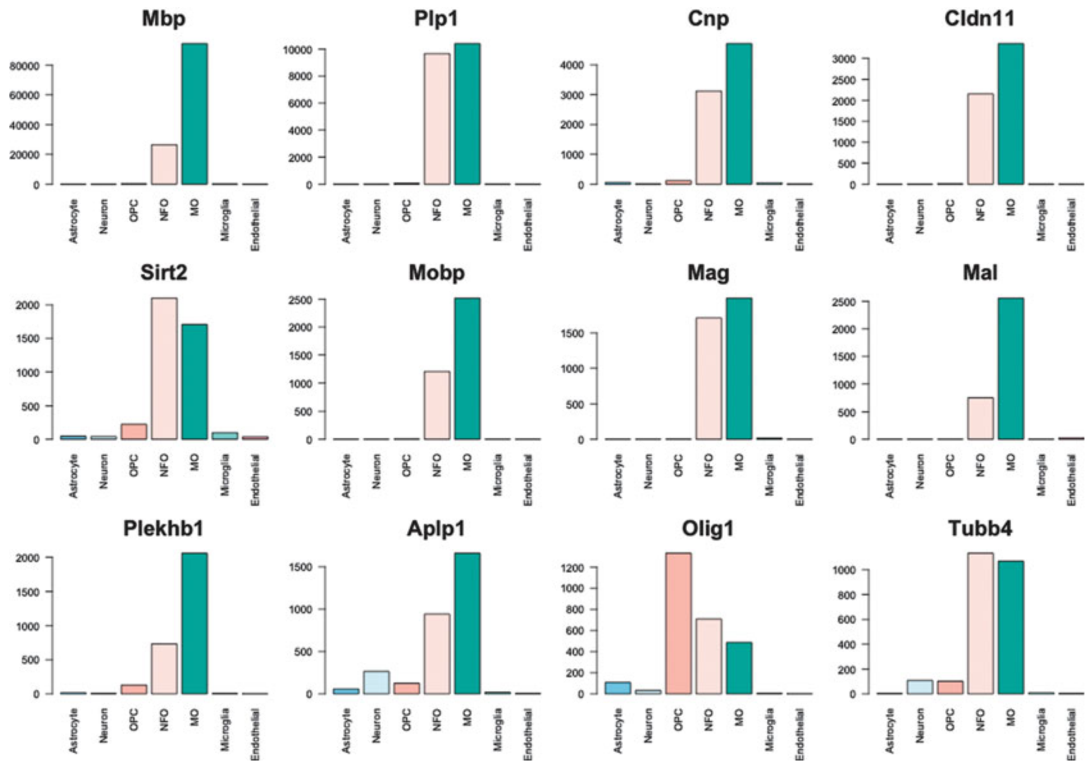


Fig. 6 Genes with highest expression values (FPKM) identified by RNentropy as being overexpressed in at least one of the oligodendrocyte subtypes

The first line of code sets the name of columns in the "selected" table to match those of the input samples. This is convenient to make the subsequent lines of code clearer. The second line of code stores in the *Oligodendrocyte_genes* variable the names of the 1227 genes that are, according to RNentropy, at the same time overexpressed in at least one of the oligodendrocyte subtypes and not overexpressed in any of the other four non-oligodendrocyte cell types (Fig. 6).

The final step consists in extracting, from the resulting oligodendrocyte-specific gene set, those genes whose are overexpressed in only one of the three oligodendrocyte maturation stages. Proceed as follows:

```
Selected_Oligodendrocytes <- Barres_Results$selected[Oligodendrocyte_genes,]
 OPC_Specific_Genes <- row.names(Selected_Oligodendrocytes[Selected_Oligodendrocytes$OPC== 1
                                 & Selected_Oligodendrocytes$NFO != 1
                                 & Selected_Oligodendrocytes$MO != 1
                                 ,])
 NFO_Specific_Genes <- row.names(Selected_Oligodendrocytes[Selected_Oligodendrocytes$OPC!= 1
                                 & Selected_Oligodendrocytes$NFO == 1
                                 & Selected_Oligodendrocytes$MO != 1
                                 ,])
MO_Specific_Genes <-row.names(Selected_Oligodendrocytes[Selected_Oligodendrocytes$OPC!= 1
                                 & Selected_Oligodendrocytes$NFO != 1
                                 & Selected_Oligodendrocytes$MO == 1
                                 ,])
```

The first line of code defines a new variable storing the “selected” table filtered for the oligodendrocyte-specific genes, useful to simplify the rest of the code. Then, the names of the OPC, NFO, and MO specific genes are stored respectively in the variables *OPC_Specific_Genes*, *NFO_Specific_Genes*, and *MO_Specific_Genes* (Fig. 7). RNentropy identified 379, 214, and 190 genes whose overexpression is significant and specific for only one of the three cell subtypes OPC, NFO, and MO, respectively. The remaining 444 oligodendrocyte-specific genes can then be considered overexpressed in at least two of the three subtypes.

3.3 Applying RNentropy to Single-Cell RNA-Seq Data

The computational and statistical framework RNentropy is based on makes it perfectly suitable also for the analysis of single-cell RNA-Seq data. In this case, every cell is a “condition” by itself, and no replicates have to be defined. The output will indicate, for each gene, if its expression changes in a significant way across all the cells studied, and the list of cells in which the gene is considered to be over- or underexpressed. Since the typical input for experiments of this kind is made of thousands of cells, the output of RNentropy over the whole dataset without no further pre- or postprocessing would be of little use. Our suggestion is to couple RNentropy with the commonly used strategies for scRNA-Seq analysis (for example in [21]), either as a preprocessing step for the selection of the most variable genes or as a final step after the individuation of cluster of

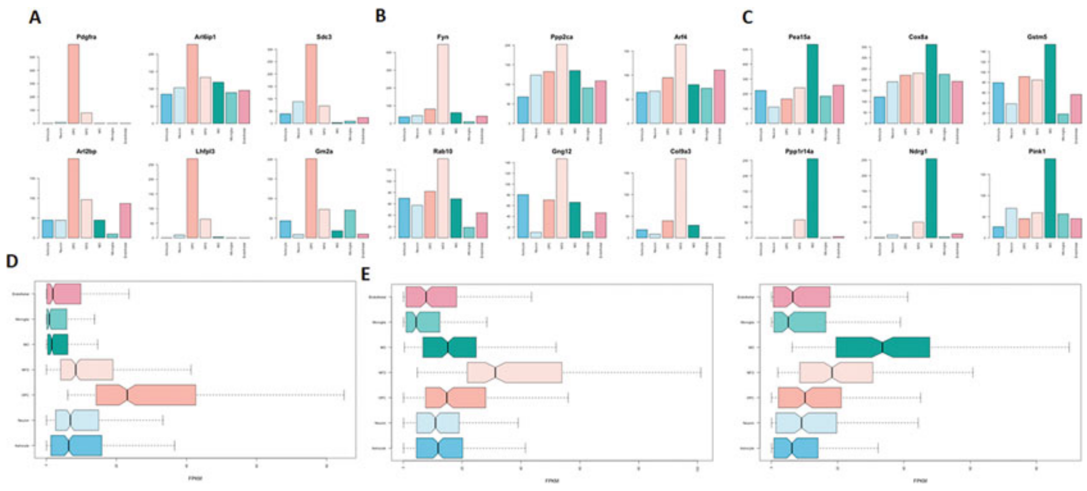


Fig. 7 Top panel: examples of expression variation in the seven cell types for genes overexpressed in (**a**) OPC only, (**b**) NFO only, and (**c**) MO only. Bottom panel: distribution of expression levels across the seven cell types for OPC (**d**), NFO (**e**), and MO (**f**) specific genes

cells defining putative subtypes, for the characterization of the marker genes of each cluster.

For preprocessing, it has been shown that in several cases better results can be obtained by considering only the most variable genes across the cells studied for dimensionality reduction and clustering [11]. This step is straightforward with RNentropy, by running the global specificity test, and selecting genes passing the test with a p-value lower than a given threshold or ranking genes according to increasing p-values and selecting the top n.

For postprocessing, let us suppose that n clusters have been determined by the analysis of a scRNA-Seq dataset after normalization, dimensionality reduction, and clustering. Let n_i be the number of cells assigned to cluster i, and N the overall number of cells (samples) studied. As we stated above, expression data after normalization can be processed as they are by RNentropy, considering each cell a condition by itself with no replicates. For any gene G passing the global specificity test, hence whose expression is significantly variable across the samples, the number of cells in each cluster in which the gene is overexpressed can be easily computed by examining the results of the local specificity test. The idea is that gene G can be considered to be a "marker" gene for a given cluster (cell subtype) if it results from the local specificity test to be overexpressed in a significant fraction of the cells of the cluster itself. In order to avoid the definition of arbitrary thresholds for this step, we can compute explicitly the probability of a gene to be found overexpressed in a given number of cells of a cluster by chance. Let u_i be the number of cells of cluster i in which G is overexpressed, and let U the total number of cells analysed in which the gene is overexpressed. Since the fraction of cells belonging to cluster i is defined by (n_i/N), the expected value for u_i can be defined as $U\ x\ (n_i/N)$. If $u_i > U\ x\ (n_i/N)$, then the probability of finding by chance u_i cells in cluster i where G is overexpressed can be calculated with Fisher's exact test:

$$P(u_i) = \frac{\binom{U}{u_i}\binom{N-U}{n_i-u_i}}{\binom{N}{n_i}}.$$

The same calculations can be repeated for every gene passing the global specificity test and each cluster, obtaining for each cluster the list of putative "marker" genes, with a p-value representing the significance of each of the latter. That is, if the p-value returned by the formula for a given gene G and a cluster i is lower than a typical threshold like 0.01 (possibly corrected by the numbers of genes and clusters tested), then G can be considered to be a "marker" for cluster i. The same rationale can also be applied to down-regulated genes, in order to single out genes whose expression seems to be silenced in each of the clusters identified.

4 Notes

1. As mentioned in Subheading 3.2.1 and demonstrated in [20], RNentropy shows consistent results independently of the normalization metrics employed to quantify gene expression levels. However, usually, TMM is considered to be more statistically sound when comparing expression levels in different conditions [14] and should be preferred, whenever possible, in typical RNA-Seq assays where only a minority of the genes are expected to be significantly differentially expressed among any two conditions. In this case, the input values for RNentropy are the "read per million" values after TMM normalization, that can be for example performed by using the edgeR package [14]. For single-cell RNA-Seq data, where metrics change according to the platform and sequencing strategy employed, we advise using the same values used in all the other analysis steps, that is, for dimensionality reduction and clustering.
2. Biological replicates are of critical importance to any RNA-Seq experiment [15]. However, the strict approach adopted by RNentropy in the selection of significantly differentially expressed genes (*see* Subheading 3.1.4) implies that the presence of just one outlier replicate in a condition could shadow the presence of genuinely differentially expressed genes in that condition, possibly increasing the ratio of false negatives. Also, when replicates present high biological variability (e.g., come from different individuals in a study designed on human or

mouse), none of them will likely present the same exact pattern of differentially expressed genes. To avoid this eventuality, it is always better to process data also by keeping "replicates" separate, also in order to verify the consistency of biological replicates and eventually to discard inconsistent replicates whenever possible. One approach to check the consistency of replicates is the one described in Subheading 3.2.4, which is making a preliminary run of RNentropy considering all the samples as different conditions and computing the pointwise mutual information table. Replicates for the same condition that do not cluster together should be considered outliers. Another way to perform the same check is to draw a principal component analysis plot (e.g., using the *prcomp* function of R) on the expression values of all the samples and, again, discard those replicates for the same condition that do not cluster together. When dealing with highly variable replicates of the same condition, differentially expressed genes can be defined downstream of the analysis, for example by selecting those that result differentially expressed in most of the replicates, according to the conditions studied.

References

1. Goodwin S, McPherson JD, McCombie WR (2016) Coming of age: ten years of next-generation sequencing technologies. Nat Rev Genet 17:333–351. https://doi.org/10.1038/nrg.2016.49
2. Wang Z, Gerstein M, Snyder M (2009) RNA-Seq: a revolutionary tool for transcriptomics. Nat Rev Genet 10:57–63. https://doi.org/10.1038/nrg2484
3. Kulkarni A, Anderson AG, Merullo DP, Konopka G (2019) Beyond bulk: a review of single cell transcriptomics methodologies and applications. Curr Opin Biotechnol 58:129–136. https://doi.org/10.1016/j.copbio.2019.03.001
4. Trapnell C, Williams BA, Pertea G et al (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat Biotechnol 28:511–515. https://doi.org/10.1038/nbt.1621
5. Grabherr MG, Haas BJ, Yassour M et al (2011) Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nat Biotechnol 29:644–652. https://doi.org/10.1038/nbt.1883
6. Li B, Dewey CN (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics 12:323. https://doi.org/10.1186/1471-2105-12-323
7. Anders S, Pyl PT, Huber W (2015) HTSeq-A Python framework to work with high-throughput sequencing data. Bioinformatics 31:166–169. https://doi.org/10.1093/bioinformatics/btu638
8. Bray NL, Pimentel H, Melsted P, Pachter L (2016) Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol 34:525–527. https://doi.org/10.1038/nbt.3519
9. Patro R, Duggal G, Love MI et al (2017) Salmon provides fast and bias-aware quantification of transcript expression. Nat Methods 14:417–419. https://doi.org/10.1038/nmeth.4197
10. Fay DS (2013) A biologist's guide to statistical thinking and analysis. WormBook, Pasadena, CA, pp 1–54. https://doi.org/10.1895/wormbook.1.159.1
11. Brennecke P, Anders S, Kim JK et al (2013) Accounting for technical noise in single-cell RNA-seq experiments. Nat Methods 10:1093–1098. https://doi.org/10.1038/nmeth.2645
12. Garber M, Grabherr MG, Guttman M, Trapnell C (2011) Computational methods for transcriptome annotation and quantification

using RNA-seq. Nat Methods 8:469–477. https://doi.org/10.1038/nmeth.1613
13. Costa-Silva J, Domingues D, Lopes FM (2017) RNA-Seq differential expression analysis: an extended review and a software tool. PLoS One 12:1–18. https://doi.org/10.1371/journal.pone.0190152
14. Robinson MD, McCarthy DJ, Smyth GK (2009) edgeR: a bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26:139–140. https://doi.org/10.1093/bioinformatics/btp616
15. Schurch NJ, Schofield P, Gierliński M et al (2016) How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use? RNA 22:839–851. https://doi.org/10.1261/rna.053959.115
16. Conesa A, Madrigal P, Tarazona S et al (2016) A survey of best practices for RNA-seq data analysis. Genome Biol 17:1–19. https://doi.org/10.1186/s13059-016-0881-8
17. Kryuchkova-Mostacci N, Robinson-Rechavi M (2017) A benchmark of gene expression tissue-specificity metrics. Brief Bioinform 18:205–214. https://doi.org/10.1093/bib/bbw008
18. Mcintyre LM, Lopiano KK, Morse AM et al (2011) RNA-seq : technical variability and sampling. BMC Genomics. https://doi.org/10.1186/1471-2164-12-293
19. Mccarthy DJ, Chen Y, Smyth GK (2012) Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Res 40:4288–4297. https://doi.org/10.1093/nar/gks042
20. Zambelli F, Mastropasqua F, Picardi E et al (2018) RNentropy: an entropy-based tool for the detection of significant variation of gene expression across multiple RNA-Seq experiments. Nucleic Acids Res 46(8):e46. https://doi.org/10.1093/nar/gky055
21. Bhattacherjee A, Djekidel MN, Chen R et al (2019) Cell type-specific transcriptional programs in mouse prefrontal cortex during adolescence and addiction. Nat Commun 10:4169. https://doi.org/10.1038/s41467-019-12054-3
22. Wang K, Phillips CA, Rogers GL et al (2014) Differential Shannon entropy and differential coefficient of variation: alternatives and augmentations to differential expression in the search for disease-related genes. Int J Comput Biol Drug Des 7:183–194. https://doi.org/10.1504/IJCBDD.2014.061656
23. Vajapeyam S (2014) Understanding Shannon's entropy metric for information. arXiv 1405:2061
24. McDonald JH (2014) Handbook of biological statistics, 3rd edn. Sparky House Publishing, Baltimore, MD
25. Benjamini Y, Hochberg Y (1995) Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. J R Stat Soc B 57:289–300. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x
26. Fano RM, Hawkins D (1961) Transmission of information: a statistical theory of communications. Am J Physiol 29:793–794. https://doi.org/10.1119/1.1937609
27. Zhang Y, Chen K, Sloan SA et al (2014) An RNA-sequencing transcriptome and splicing database of glia, neurons, and vascular cells of the cerebral cortex. J Neurosci 34:11929–11947. https://doi.org/10.1523/JNEUROSCI.1860-14.2014

Chapter 7

Statistical Modeling of High Dimensional Counts

Michael I. Love

Abstract

Statistical modeling of count data from RNA sequencing (RNA-seq) experiments is important for proper interpretation of results. Here I will describe how count data can be modeled using count distributions, or alternatively analyzed using nonparametric methods. I will focus on basic routines for performing data input, scaling/normalization, visualization, and statistical testing to determine sets of features where the counts reflect differences in gene expression across samples. Finally, I discuss limitations and possible extensions to the models presented here.

Key words Count data, DESeq2, Gene expression, RNA-seq

1 Introduction

Here I will describe how count data, as often arises in RNA sequencing (RNA-seq) experiments, can be modeled using count distributions, as well as how nonparametric methods can be used to analyze count data. The section will cover basic routines for performing data input, scaling/normalization, visualization, and statistical testing to determine sets of features where the counts reflect differences in expression across samples. The final section will cover limitations of the methods presented and extensions.

The code in this book includes the basic routines that can be found in software vignettes of various Bioconductor packages, including tximeta, DESeq2, and fishpond. Please see those package vignettes for further details. Any specific questions about Bioconductor software should be posted to the Bioconductor support site:

- https://support.bioconductor.org.

There are also two published workflows that are related to the analysis steps and packages described here, but which explore different directions. These workflows are hosted on the Bioconductor workflow site, and checked regularly to ensure they build correctly and without error:

- rnaseqGene—gene-level exploratory analysis and differential expression [1].
- rnaseqDTU—differential transcript usage [2].

Another related reference is Van den Berge et al. [3], which is a review of RNA-seq expression analysis, written by a collection of researchers who develop statistical models and software for RNA-seq data.

2 Quantification

One key initial step in analyzing RNA-seq data is to quantify, or estimate, the number of fragments in the experiment that can be assigned to each feature, whether a gene or a transcript (an isoform of a gene). Alongside quantification, it is strongly recommended to perform quality control (QC) checks on the sequence files. Reports spanning multiple samples can be generated with the MultiQC software [4]. Here I will not start with quality control checks, but instead move straight to quantification and import of estimated counts into R, although an example quality control report can be found in the extdata directory of the *airway2* package on GitHub (https://github.com/mikelove/airway2).

The data I will examine first in this section is from an experiment of the effect of knocking down OCT4 and BRG1 in mouse embryos [5]. In particular, I will examine the treatment effect of knocking down OCT4. The experiment had four groups of samples, each replicated in triplicate. The RNA-seq data from the experiment is available in the *oct4* Bioconductor package.

I will use the *Salmon* software for estimating transcript abundance [6]. Briefly, *Salmon* uses the sequenced reads and the reference transcripts, and constructs a generative model for the observed data, which includes modeling of various technical biases that are commonly observed in RNA-seq. *Salmon* then outputs the estimated counts for each transcript, and an "effective length" of the transcript, which is shorter than the full length of the transcript if it was biased to having fewer reads due to technical artifact, and longer than the full length of the transcript if it was biased to having more reads. For details about the Salmon method and software, refer to the Salmon website (https://combine-lab.github.io/salmon).

All 12 samples from the experiment were quantified using *Salmon*. The jobs for processing the reads with *Salmon* were executed via Snakemake [7], a convenient tool for scheduling and executing repetitive rule-based operations on input data. The exact lines of code can be seen in the Snakefile provided in the scripts directory of the *oct4* package.

Most of the packages shown in this online book live within the Bioconductor project [8]. Bioconductor objects are more complex than basic objects in R, for example *numeric*, *character*, or other simple objects, in that they often have attached metadata, such as additional information about rows and columns, or other metadata. We will see how to make use of the metadata throughout the various sections by examining, for example, `colData` for information about the columns of a matrix, or `mcols` for metadata columns.

I begin by loading some data in the Bioconductor package *oct4*. Note that this step is not useful for a typical RNA-seq workflow, as the data will not be contained in an R package, but contained in some directory on a server or compute cluster. So in lieu of the `system.file` command below, which is used here to locate a file within an R package, you could just specify the `dir` variable to be a path to the files, for example, `/path/to/data/dir`. Note also that this and other packages must be installed via the Bioconductor package installation instructions at https://bioconductor.org/install.

```
dir <- system.file("extdata", package="oct4", mustWork=TRUE)
list.files(dir)
```

```
## [1] "coldata.csv"     "list"            "quants"
## [4] "SraRunTable.txt"
```

I will use the *readr* and *dplyr* packages to read in a CSV file with information about the samples. Because we are typically working with "tall" count matrices, with the rows representing genomic features such as genes or transcripts, and columns representing samples, the sample information is referred to as the column data or `coldata`. For more information on dplyr, see the excellent dplyr online documentation.

```
library(readr)
coldata <- read_csv(file.path(dir,"coldata.csv"))
```

```
## Parsed with column specification:
## cols(
##   names = col_character(),
##   line = col_character(),
##   condition = col_character()
## )
```

I next set the `levels` of the `line` and `condition` factor variables so that `OCT4` and `untrt` are the reference levels, and I specify the path to the quantification files I want to read in.

In the last step of the `mutate` call, note I specify paths to the quantification files `quant.sf.gz`. Typically, these files are not gzipped, but I have compressed the `quant.sf` files to reduce the size of the data package. Note also that, although I point only to the quant.sf files, the entire directory containing that file is required for proper import of the quantification data. There are other metadata files that provide important information about the experiment, including uncertainty information. The metadata files enable automatic identification of the feature set, a computational reproducibility feature that will be described later in this section.

```
suppressPackageStartupMessages(library(dplyr))
coldata <- coldata %>%
  mutate(line=factor(line, levels=c("OCT4","BRG1")),
         condition=factor(condition, levels=c("untrt","trt")),
         files=file.path(dir, "quants", names, "quant.sf.gz"))
```

All the files exist at the locations I specified, which in this case have the pattern /<DIR>/quants/<NAMES>/quant.sf.gz.

```
all(file.exists(coldata$files))
## [1] TRUE
```

I will now use the Bioconductor package *tximeta* [9] to read in the quantification data, and create a *SummarizedExperiment* object. The *SummarizedExperiment* class is described by Huber et al. [8], in particular diagrammed in Figure 2. Here, the tximeta function performs a number of operations on behalf of the user, identifying the reference transcripts that were used to quantify the RNA-seq reads, automatically downloading (or loading) the relevant genomic locations of the transcripts, and attaching the relevant genomic context (the genome version and chromosome names and lengths). *tximeta* can perform these operations by leveraging the hash signature of the sequence of the transcripts, which is stored when processing bulk RNA-seq with *Salmon*, or when processing single cell RNA-seq with the *alevin* software [10] (distributed with *Salmon*).

A small note: I specify dropInfReps=TRUE, because I will be performing gene-level analysis, and I will not make use of the uncertainty information in this first analysis. I will show in a later section how to make use of the inferential replicates computed by *Salmon* during quantification.

```
library(tximeta)

suppressPackageStartupMessages(library(SummarizedExperiment))

se <- tximeta(coldata, dropInfReps=TRUE)

## importing quantifications

## reading in files with read_tsv

## 1 2 3 4 5 6 7 8 9 10 11 12
## found matching transcriptome:
## [ GENCODE - Mus musculus - release M20 ]
## loading existing TxDb created: 2020-03-25 23:26:23
## loading existing transcript ranges created: 2020-07-15 15:13:43
## fetching genome info for GENCODE
```

In the above output, we can see that `tximeta` identified that the mouse reference transcripts, release M20, were used from GENCODE [11]. This information would be automatically identified, even if the group that performed the quantification did not document this when uploading processed expression data to a public repository.

We can see the matrices of data that have been compiled, and examine the genomic locations of the features (rows):

```
assayNames(se)
```

```
## [1] "counts"    "abundance" "length"
```

```
rowRanges(se)
```

```
## GRanges object with 137271 ranges and 3 metadata columns:
##                           seqnames          ranges strand |
##                              <Rle>       <IRanges>  <Rle> |
##   ENSMUST00000193812.1     chr1 3073253-3074322      + |
##   ENSMUST00000082908.1     chr1 3102016-3102125      + |
##   ENSMUST00000162897.1     chr1 3205901-3216344      - |
##   ENSMUST00000159265.1     chr1 3206523-3215632      - |
##   ENSMUST00000070533.4     chr1 3214482-3671498      - |
##                    ...      ...             ...    ... .
##   ENSMUST00000082419.1     chrM     13552-14070      - |
##   ENSMUST00000082420.1     chrM     14071-14139      - |
##   ENSMUST00000082421.1     chrM     14145-15288      + |
##   ENSMUST00000082422.1     chrM     15289-15355      + |
##   ENSMUST00000082423.1     chrM     15356-15422      - |
##                            tx_id            gene_id
##                        <integer>    <CharacterList>
##   ENSMUST00000193812.1         1 ENSMUSG00000102693.1
##   ENSMUST00000082908.1         2 ENSMUSG00000064842.1
##   ENSMUST00000162897.1      4203 ENSMUSG00000051951.5
##   ENSMUST00000159265.1      4204 ENSMUSG00000051951.5
##   ENSMUST00000070533.4      4205 ENSMUSG00000051951.5
##                    ...       ...                  ...
##   ENSMUST00000082419.1    138833 ENSMUSG00000064368.1
##   ENSMUST00000082420.1    138834 ENSMUSG00000064369.1
##   ENSMUST00000082421.1    138825 ENSMUSG00000064370.1
##   ENSMUST00000082422.1    138826 ENSMUSG00000064371.1
##   ENSMUST00000082423.1    138835 ENSMUSG00000064372.1
##                                     tx_name
##                                 <character>
##   ENSMUST00000193812.1 ENSMUST00000193812.1
##   ENSMUST00000082908.1 ENSMUST00000082908.1
##   ENSMUST00000162897.1 ENSMUST00000162897.1
##   ENSMUST00000159265.1 ENSMUST00000159265.1
##   ENSMUST00000070533.4 ENSMUST00000070533.4
##                    ...                  ...
##   ENSMUST00000082419.1 ENSMUST00000082419.1
##   ENSMUST00000082420.1 ENSMUST00000082420.1
##   ENSMUST00000082421.1 ENSMUST00000082421.1
##   ENSMUST00000082422.1 ENSMUST00000082422.1
##   ENSMUST00000082423.1 ENSMUST00000082423.1
##   -------
##   seqinfo: 22 sequences (1 circular) from mm10 genome
```

Because the `tximeta` function has identified the correct reference transcripts that were used for quantification, and stored this as metadata in the *SummarizedExperiment* object, I can obtain additional information, such as the correspondence of transcripts to genes. I can then perform a summarization task, without having to look up this information manually.

```
gse <- summarizeToGene(se)

## loading existing TxDb created: 2020-03-25 23:26:23

## obtaining transcript-to-gene mapping from database

## loading existing gene ranges created: 2020-03-25 23:26:26

## summarizing abundance

## summarizing counts

## summarizing length
```

Gene-level summarization of transcript-level quantification data is described in the paper for *tximport* [12]. The `tximeta` function internally calls methods from the *tximport* package during its operation. Note that *tximport* provides similar functionality to *tximeta*, but instead of returning a rich Bioconductor object, *tximport* returns a simple list of matrices.

Below I show some examples of operations that can be easily performed because I have a *SummarizedExperiment* object with the appropriate gene ranges attached. I can easily subset the genes (rows) based on overlaps with a given genomic range using standard square bracket indexing. The range-based class system and set of methods for operating on genomic ranges is provided by the *GenomicRanges* package, which has useful help pages and vignettes [13].

```
rowRanges (gse)

## GRanges object with 53697 ranges and 1 metadata column:
##                        seqnames              ranges strand |
##                           <Rle>           <IRanges>  <Rle> |
##    ENSMUSG00000000001.4    chr3 108107280-108146146      - |
##   ENSMUSG00000000003.15    chrX   77837901-77853623      - |
##   ENSMUSG00000000028.15   chr16   18780447-18811987      - |
##   ENSMUSG00000000031.16    chr7 142575529-142578143      - |
##   ENSMUSG00000000037.16    chrX 161117193-161258213      + |
##                     ...     ...                 ...    ... .
##    ENSMUSG00000117651.1   chr17   32731010-32731806      + |
##    ENSMUSG00000117652.1   chr18     6910459-6936621      - |
##    ENSMUSG00000117653.1   chr17   94811611-94812922      - |
##    ENSMUSG00000117654.1   chr17   32863662-32877942      - |
##    ENSMUSG00000117655.1   chr19   57497390-57512784      + |
        ##                                            gene_id
        ##                                        <character>
        ##    ENSMUSG00000000001.4   ENSMUSG00000000001.4
        ##   ENSMUSG00000000003.15  ENSMUSG00000000003.15
        ##   ENSMUSG00000000028.15  ENSMUSG00000000028.15
        ##   ENSMUSG00000000031.16  ENSMUSG00000000031.16
        ##   ENSMUSG00000000037.16  ENSMUSG00000000037.16
        ##                     ...                    ...
        ##    ENSMUSG00000117651.1   ENSMUSG00000117651.1
        ##    ENSMUSG00000117652.1   ENSMUSG00000117652.1
        ##    ENSMUSG00000117653.1   ENSMUSG00000117653.1
        ##    ENSMUSG00000117654.1   ENSMUSG00000117654.1
        ##    ENSMUSG00000117655.1   ENSMUSG00000117655.1
        ##   -------
        ##   seqinfo: 22 sequences (1 circular) from mm10 genome
```

```
x <- GRanges("chr1", IRanges(10e6, 11e6))
gse[gse %over% x, ]
```

```
## class: RangedSummarizedExperiment
## dim: 18 12
## metadata(6): tximetaInfo quantInfo ... txomeInfo txdbInfo
## assays(3): counts abundance length
## rownames(18): ENSMUSG00000025916.10 ENSMUSG00000025917.9
##   ... ENSMUSG00000103448.1 ENSMUSG00000103810.1
## rowData names(1): gene_id
## colnames(12): SRX2236945 SRX2236946 ... SRX2236955
##   SRX2236956
## colData names(3): names line condition
```

I can likewise easily subset the samples by referring to the `colData` columns, for example, `colData(gse)$line`. A shortcut for referring to `colData` columns is to just use the dollar sign directly on the object, `gse$line`.

```
gse[, gse$line == "OCT4"]
```

```
## class: RangedSummarizedExperiment
## dim: 53697 6
## metadata(6): tximetaInfo quantInfo ... txomeInfo txdbInfo
## assays(3): counts abundance length
## rownames(53697): ENSMUSG00000000001.4
##   ENSMUSG00000000003.15 ... ENSMUSG00000117654.1
##   ENSMUSG00000117655.1
## rowData names(1): gene_id
## colnames(6): SRX2236945 SRX2236946 ... SRX2236949
##   SRX2236950
## colData names(3): names line condition
```

Finally, I demonstrate that it is easy to add alternative identifiers, making use of the organism annotation packages in

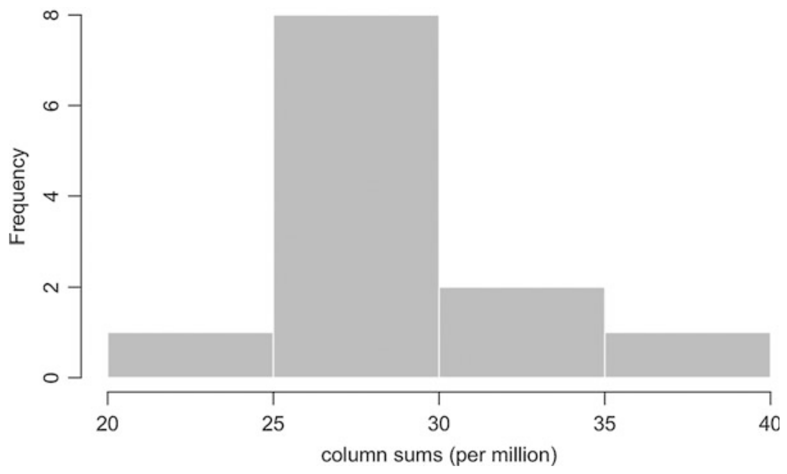


Fig. 1 Distribution of mapped fragments per sample in millions

Bioconductor. For example, to add gene symbols, I can use *tximeta*'s `addIds` function, and specify the `SYMBOL` column.

```
library(org.Mm.eg.db)
gse <- addIds(gse, column="SYMBOL")

## mapping to new IDs using 'org.Mm.eg.db' data package
## if all matching IDs are desired, and '1:many mappings' are reported,
## set multiVals='list' to obtain all the matching IDs

## it appears the rows are gene IDs, setting 'gene' to TRUE

## 'select()' returned 1:many mapping between keys and
## columns
```

To check all the available columns for an organism package, use the `columns` function:

```
columns(org.Mm.eg.db)

##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"
##  [5] "ENSEMBLTRANS" "ENTREZID"     "ENZYME"       "EVIDENCE"
##  [9] "EVIDENCEALL"  "GENENAME"     "GO"           "GOALL"
## [13] "IPI"          "MGI"          "ONTOLOGY"     "ONTOLOGYALL"
## [17] "PATH"         "PFAM"         "PMID"         "PROSITE"
## [21] "REFSEQ"       "SYMBOL"       "UNIGENE"      "UNIPROT"
```

3 Counts Modeling

3.1 A First Exploration of Counts

In this section, I will discuss the statistical models that are often used to analyze RNA-seq data, in particular gene-level count matrices. I will then use the *DESeq2* package to calculate scaling factors,

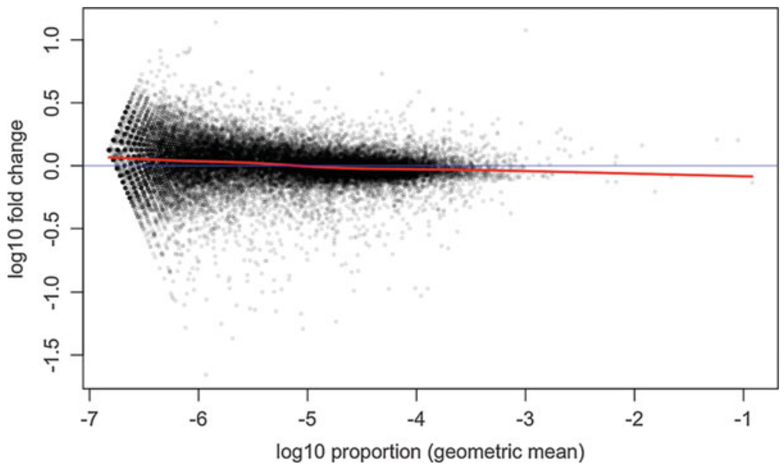


Fig. 2 Ratio of the proportions from samples 1 and 2, over the geometric mean of proportions (log10 scale). Each gene is plotted as a point

estimate biological dispersion within groups of samples, and perform differential testing per gene [14].

Some other popular Bioconductor packages for RNA-seq analysis include the *edgeR* package [15, 16] and the *limma-voom* method in the *limma* package [17]. The approach taken by *DESeq2* for estimation of dispersion is similar to the method proposed by Wu, Wang, and Wu [18] in the *DSS* Bioconductor package.

I will begin by investigating the estimated counts that were imported from the *Salmon* software, and comparing these counts across and within samples. I will note the varying precision that the counts offer for log ratio comparisons between samples. Finally, I will perform per-gene testing for differential expression using the *DESeq2* package, and multiple test correction, including the *IHW* method [19]. Note that the first part of this section includes code and plots that are not typically performed during RNA-seq analysis, but mostly for introducing the reader to some basic properties of RNA-seq count matrices.

First, it is useful to explore the varying number of fragments (pairs of reads) that have been assigned to the genes for each sample (Fig. 1). For a typical mammalian RNA-seq experiment, we might expect tens of millions of fragments per sample, which are distributed across tens of thousands of genes, although there is inevitably a range of sequencing depth for each sample:

```
assayNames(gse)

## [1] "counts"    "abundance" "length"

cs <- colSums(assay(gse, "counts"))
hist(cs/1e6, col="grey", border="white",
     main="", xlab="column sums (per million)")
```

Let us first consider just two samples, one from the OCT4 untreated group and one from the OCT4 treated group. I will make a plot examining the proportion of the total count for each

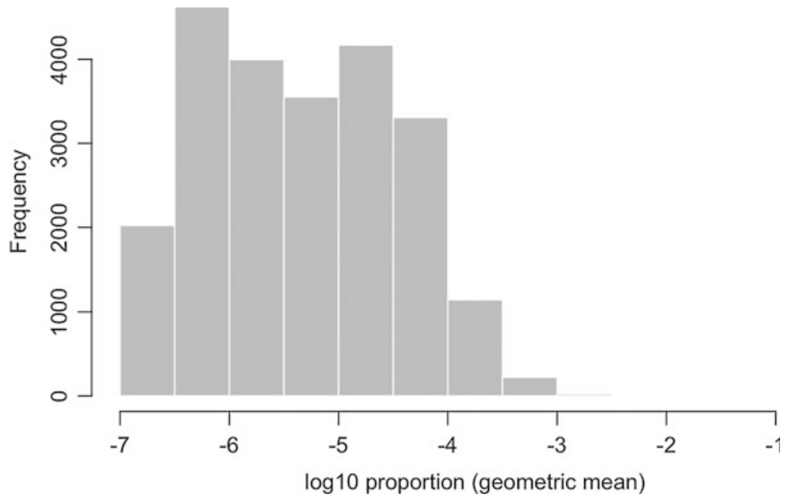


Fig. 3 Distribution of the mean of log10 proportions across all genes

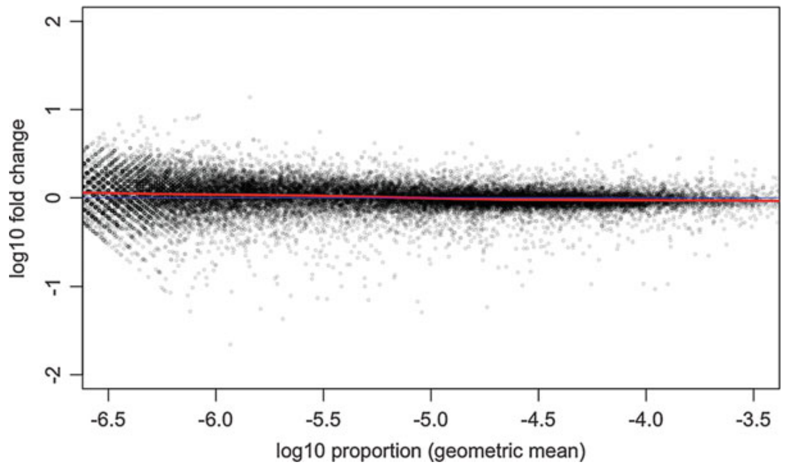


Fig. 4 Zoomed-in plot of the ratio of proportions over the geometric mean (log10 scale), between the range of −6.5 and −3.5

gene. I will first subset to only those genes where both samples have a count of 5 or more, to cut down on the number of points to plot. Here I use the total count, `colSums(cts)` to divide the counts for each sample, while later we will discuss a robust estimator for sequencing depth variation across samples.

Before I create the proportions, it is important to remember that, because genes with longer transcripts will produce more cDNA fragments, the proportion estimated here (without taking into account the length of the feature) does not estimate the proportion of *molecules*. The `abundance` assay of the `gse` object contains estimates of the proportions of the molecules, in transcripts per million (TPM). For more on how these estimates are computed, consult the *Salmon* paper [6].

```
cts <- assay(gse, "counts")[,c(1,4)]
idx <- rowSums(cts >= 5) == 2
cts <- cts[idx,]
p <- sweep(cts, 2, colSums(cts), "/")
```

We can examine the ratio of the proportions for each gene, over the geometric mean, using the `maplot` function from the *rafalib* package. We will plot the ratio and the geometric mean of proportions both on the log scale. `maplot` is a basic function for making an "MA-plot" which has been used in transcriptomics since at least the early 2000s when it was used for microarray datasets [20, 21]. Later I will show a specialized MA-plot function for RNA-seq data in *DESeq2* (Fig. 2).

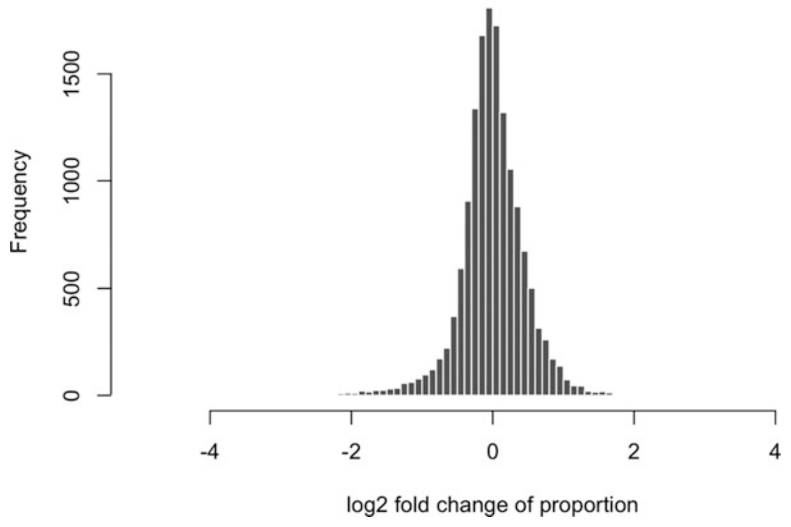


Fig. 5 Distribution of the log2 fold changes in proportion between sample 1 and 2, when the geometric mean proportion is between 1/1e6 and 1/1e4

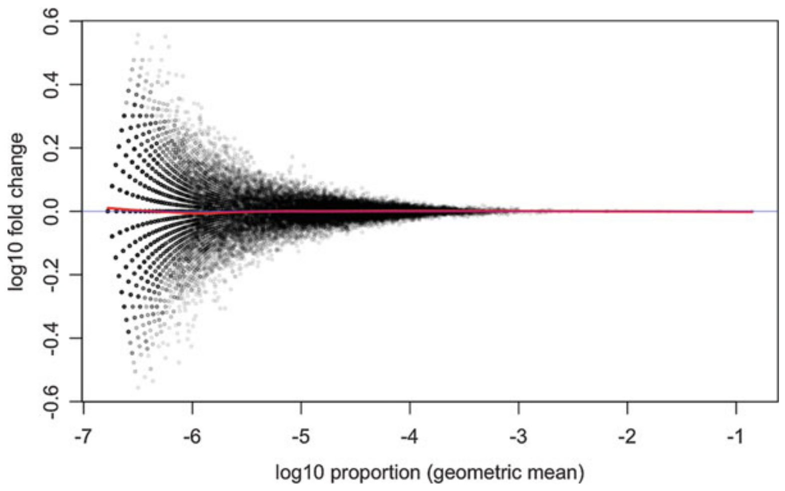


Fig. 6 MA-plot for simulated technical replicate samples, demonstrating the extent of expected variation from sampling fragments

```
library(rafalib)
maplot(log10(p[,1]), log10(p[,2]), n=nrow(p),
       cex=.3, col=rgb(0,0,0,.2),
       xlab="log10 proportion (geometric mean)",
       ylab="log10 fold change")
abline(h=0, col=rgb(0,0,1,.5))
```

The red line is a smooth curve through the log ratios (here $\log_{10}$, although we will later switch to $\log_2$ for more interpretable ratios). Note that the line is relatively flat across many orders of magnitude of the proportion. So whether the gene has 1 millionth of the total count (-6) or up to one thousandth of the total count (-3), the ratio between the two samples tends to fluctuate around 0, with most of the points somewhere between -0.1 and 0.1. The points further off the horizontal line may indicate differentially expressed genes, but we will better identify these using all of the replicates.

While just considering the two samples, I can also examine the histogram of the x-axis above, the mean of the $\log_{10}$ proportions (or equivalently the $\log_{10}$ of the geometric mean of proportions) (Fig. 3). We see that most genes that we are considering fall in the range from 1/10,000,000 (-7) to 1/10,000 (-4) of the total count (Fig. 4). That these ratios are very low is relevant for the choice of statistical distribution for the counts.

```
mean.log.p <- rowMeans(log10(p))
hist(mean.log.p,  col="grey", border="white",
     main="", xlab="log10 proportion (geometric mean)")

library(rafalib)
maplot(log10(p[,1]), log10(p[,2]), n=nrow(p),
       xlim=c(-6.5, -3.5), ylim=c(-2,2),
       cex=.3, col=rgb(0,0,0,.2),
       xlab="log10 proportion (geometric mean)",
       ylab="log10 fold change")
abline(h=0, col=rgb(0,0,1,.5))
```

I will zoom out on the *y*-axis and zoom in on the *x*-axis on the MA-plot to emphasize that, for the range containing most of the genes, the middle of the distribution of the ratio of proportions across the treatment is centered on 0.

The base of 10 is not a great choice for the logarithm, because it is not common to have a tenfold change, and then it becomes hard to interpret the meaningful changes in the range from 0 to 1. Much better is to use $\log_2$, as we can easily interpret 1 as doubling, 2 as quadrupling, etc. Now I calculate the $\log_2$ ratio of proportions and plot the histogram of these log ratios, for genes where the log10 mean proportion is between −6 and −4 (Fig. 5):

```
lfc <- log2(p[,2]) - log2(p[,1])
hist(lfc[between(mean.log.p, -6, -4)],
     breaks=seq(-10,10,by=.1),
     xlim=c(-5,5),
     col="grey50", border="white",
     main="", xlab="log2 fold change of proportion")
```

Again we see that it is rare for there to be an extreme change (more than doubling) comparing these two samples from different treatment groups. Most of the genes fall around 0. Exactly 0 implies no change in the proportion as calculated using the total count.

Before we begin modeling the counts from the RNA-seq experiment, I produce one more plot to give a sense of expected sampling variation with counts of a similar distribution to the ones we observed. Supposing that the proportions p was fixed for the first sample, and we draw 30,000,000 fragments according to these gene-wise proportions. We would then obtain a multinomial distribution for the counts per gene. When the number of observations is large, and the proportions are small, the count for any given gene is well approximated by a Poisson distribution. So I will create two simulated *technical replicate* samples, by creating two draws from a Poisson distribution along the genes. I then repeat the same code as above, to examine the MA-plot for the simulated counts (Fig. 6).

```
sim.cts <- matrix(rpois(nrow(p) * 2, 30e6 * p[,1]), ncol=2)
colSums(sim.cts)

## [1] 30006032 30009034

idx <- rowSums(sim.cts >= 5) == 2
sim.cts <- sim.cts[idx,]
sim.p <- sweep(sim.cts, 2, colSums(sim.cts), "/")
maplot(log10(sim.p[,1]), log10(sim.p[,2]), n=nrow(p),
       cex=.3, col=rgb(0,0,0,.2),
       xlab="log10 proportion (geometric mean)",
       ylab="log10 fold change")
abline(h=0, col=rgb(0,0,1,.5))
```

Note that, even after filtering out genes that do not have a count of 5 or more for both samples, there is still substantial variance for the ratio of proportions for the small count genes compared to the large count genes.

3.2 Modeling Counts with DESeq2

I will now demonstrate the use of the *DESeq2* package for estimating per-sample scaling factors, per-gene dispersion and fold changes per gene across the samples. As in *edgeR* and *limma*, *DESeq2* allows for the use of complex designs, leveraging R's *formula* syntax. For details about various design formula, first consult the *DESeq2* vignette and `?results` help page.

I create a *DESeqDataSet* object, by providing the gene-level *SummarizedExperiment* object, and specifying the `design`, which is a formula expressing how we wish to model the counts. Here, I specify a baseline term for each `line` (OCT4 or BRG1), and a condition effect specific to each line (so comparing treated vs untreated, specific to each line). The colon between `line` and

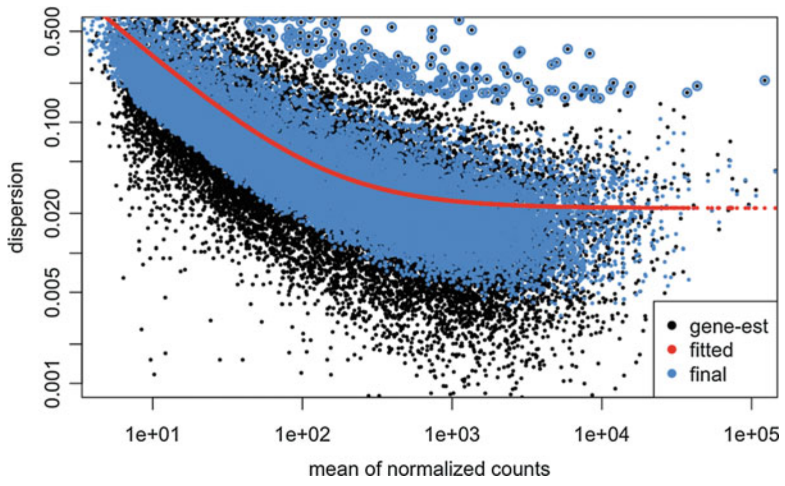


Fig. 7 Plot of dispersion estimates over the mean of scaled counts, as computed by DESeq2

`condition` specifies to form an interaction between those two terms.

```
library(DESeq2)
dds <- DESeqDataSet(gse, design=~line + line:condition)
## using counts and average transcript lengths from tximeta
```

After creating the dataset, I perform some minimal filtering that makes sense for bulk RNA-seq. I filter such that there must be at least three samples with a count of 10 or more to keep the gene in the dataset. This will reduce the size of the dataset (here, removing more than half of the genes) and the time needed to fit various per-gene parameters, such as dispersion and fold change estimates.

```
keep <- rowSums(counts(dds) >= 10) >= 3
table(keep)

## keep
## FALSE  TRUE
## 30173 23524

dds <- dds[keep,]
```

The parameters are estimated with a single call to the `DESeq` function. For details on all the steps performed by this function, check the help page `?DESeq`, as well as a section of the vignette called, "The DESeq2 model". Briefly, *DESeq2* computes a robust size factor which outperforms the total count in adjusting for differential sequence depth across libraries. Then *DESeq2* computes (iteratively) the coefficients of a generalized linear model (GLM), and a dispersion parameter that reflects the variation in addition to

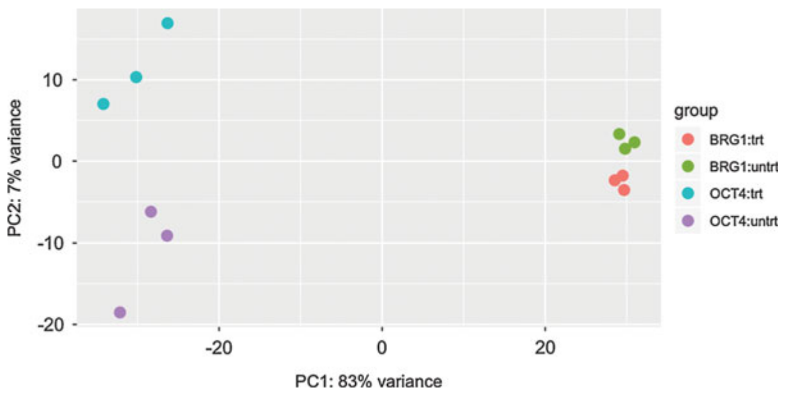


Fig. 8 Principal component analysis (PCA) plot on variance stabilized data using the VST

the Poisson variation, around the expected value for each sample conditioned on information in the design matrix.

```
dds <- DESeq(dds)
## estimating size factors
## using 'avgTxLength' from assays(dds), correcting for library size
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```

One of the key estimates is the dispersion for each gene. *DESeq2* uses a combination of methods to estimate the dispersion. First, the gene-wise estimate is produced using the methods proposed by *edgeR* in 2012 for a Negative Binomial generalized linear model (GLM) [16]. Briefly, the maximum adjusted profile likelihood estimate is calculated, where the adjustment of Cox and Reid [22] is used to avoid a downward bias on the dispersion estimate. This bias adjustment is similar in theory to the the use of $\frac{1}{n-1}$ in estimating the sample variance.

Below I plot the estimates over the mean of scaled counts for each gene. Note that many of the plots in *DESeq2* refer to "normalized counts"; here this just implies scaling the counts by the size factor, so that the differences affecting counts across samples are minimized.

There are two per-gene estimates, an initial estimate which looks only at the data for a single gene (`gene-est`, black points), and a final estimate that incorporates information from each gene, as well as sharing information across genes (`final`, blue points) (Fig. 7). The blue circles at the top of the plot represent genes with high dispersion relative to the rest of the dataset, and in these cases, only the `gene-est` estimate is used, without information from other genes.

```
plotDispEsts(dds, ylim=c(1e-3, .5), xlim=c(5,1e5))
```

Below I will continue with per-gene analysis, but first, I demonstrate how to examine differences across the 500 most variable genes using a PCA plot. Before I compute the principal components, I use the `vst` function to compute a variance stabilizing transformation (VST) [23] of the count data. This is similar to a log2 transform but avoids inflating the variance of the low count genes. For more details on the methods used here to compute the transformation, consult the *DESeq2* vignette or `?vst`. The specific VST used by *DESeq2* for RNA-seq counts was proposed by the first *DESeq* paper [24].

Another option for performing dimension reduction (as in PCA) on count data is to use the Poisson distance [25] or GLM-PCA [26]. These two alternatives are explored in the gene-level RNA-seq workflow hosted on Bioconductor [1].

Below I specify `blind=FALSE` which will use the sample grouping information when calculating the per-gene dispersion (Fig. 8). The transformation itself does not use the design, once the global dispersion trend has been fit.

```
vsd <- vst(dds, blind=FALSE)
plotPCA(vsd, intgroup=c("line","condition"))
```

We can see that the primary axis of variation among the most variable genes is the cell line difference (OCT4 vs. BRG1), and the second axis corresponds to treatment differences, though note that these are not consistent across cell line.

I can extract various results tables from the `dds` object, based on the design. Here I extract a table of results for the effect of condition in the OCT4 line. The `summary` function will provide a summary table across all genes. We observe more than a thousand genes up- and down-regulated with response to treatment for this comparison, for a false discovery rate (FDR) cutoff of 10%.

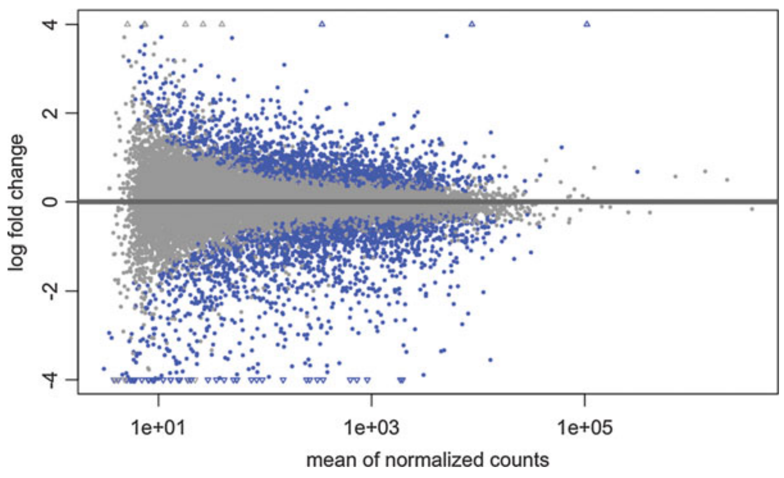


Fig. 9 MA-plot of the MLE log2 fold changes from DESeq2

```
resultsNames(dds)

## [1] "Intercept"              "line_BRG1_vs_OCT4"
## [3] "lineOCT4.conditiontrt" "lineBRG1.conditiontrt"

res <- results(dds, name="lineOCT4.conditiontrt")
summary(res)

##
## out of 23524 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1615, 6.9%
## LFC < 0 (down)     : 1932, 8.2%
## outliers [1]       : 3, 0.013%
## low counts [2]     : 0, 0%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

The genes with an adjusted *p*-value, `padj`, less than a threshold, say 0.1, provide a set that is expected to control its nominal FDR, for example no more than 10% of the genes in such a set on average should be false positives.

We can look at the top lines of the results table. Note that *DESeq2* does not sort the table, this must be done by the user. The top three lines correspond to the first three genes in the dataset unless the user performs an ordering operation.

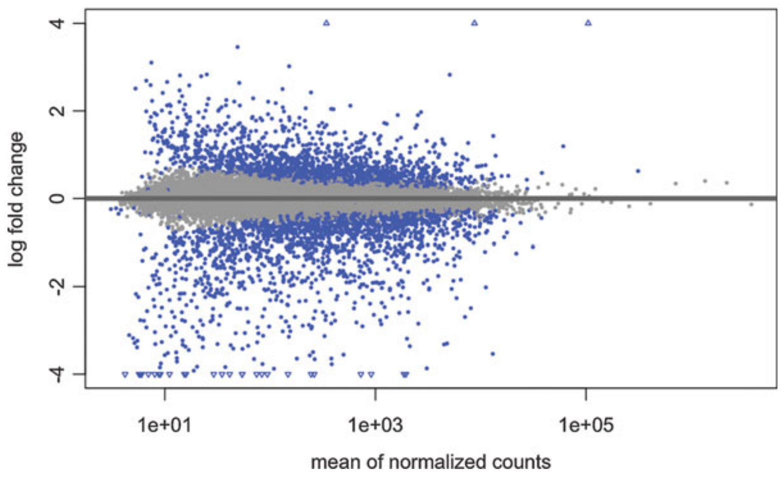


Fig. 10 MA-plot of the "shrunken" log2 fold changes from apeglm

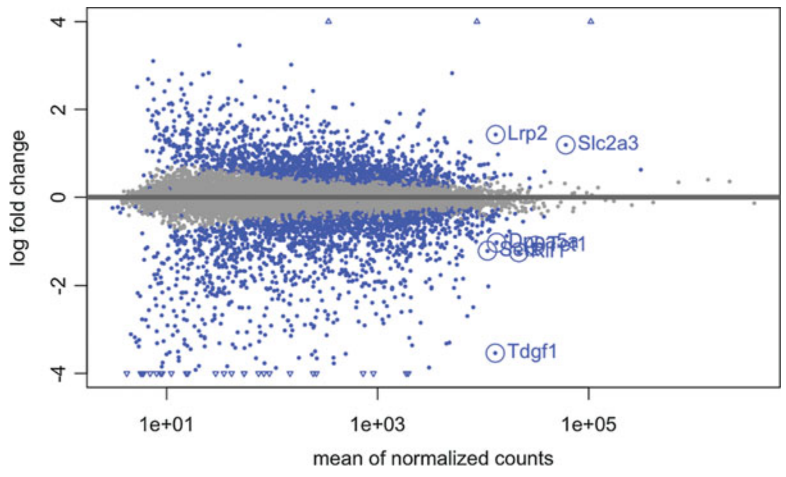


Fig. 11 MA-plot with gene symbols added using the text() function

```
head(res, 3)
```

```
## log2 fold change (MLE): lineOCT4.conditiontrt
## Wald test p-value: lineOCT4.conditiontrt
## DataFrame with 3 rows and 6 columns
##                         baseMean log2FoldChange     lfcSE
##                        <numeric>      <numeric> <numeric>
## ENSMUSG00000000001.4     2182.07    -0.00287371  0.205255
## ENSMUSG00000000028.15    1210.76     0.05177269  0.129113
## ENSMUSG00000000031.16    4641.53     1.28151815  0.620842
##                             stat    pvalue      padj
##                        <numeric> <numeric> <numeric>
## ENSMUSG00000000001.4  -0.0140007 0.9888295  0.996691
## ENSMUSG00000000028.15  0.4009890 0.6884283  0.888527
## ENSMUSG00000000031.16  2.0641608 0.0390025  0.195436
```

For example, I can order adjusted *p*-values from small to large (but then remember that this object is no long aligned with rows of `dds` for example).

```
res.ord <- res[order(res$padj),]
```

I will show how to examine the differences across all genes in an MA-plot. But first, I will compute a new estimate of fold change. The estimate in the results table above is the MLE, or maximum likelihood estimate, which is highly variable for low count genes (as we saw in the simulated example). Here I compute a Bayesian estimate for the fold change using methods in the *apeglm* package [27]. The *apeglm* functions are wrapped up in a *DESeq2* function called `lfcShrink`, which produces a table similar to `results` but

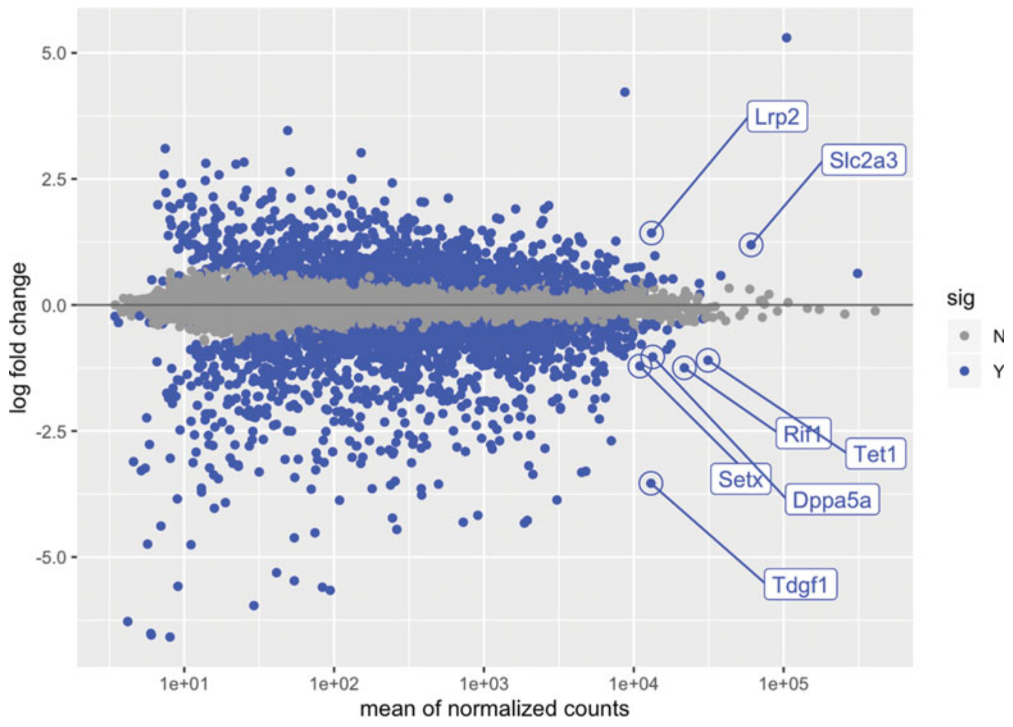


Fig. 12 MA-plot with gene symbols added using ggrepel

with shrunken LFC instead of MLE.

```
library(apeglm)
lfc <- lfcShrink(dds, coef="lineOCT4.conditiontrt", type="apeglm")

## using 'apeglm' for LFC shrinkage. If used in published research, please
cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior
distributions for
##     sequence count data: removing the noise and preserving large
differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

Another option for shrinkage is to specify `type="ashr"` which will make use of the *ashr* package for shrinkage of effect sizes, with methods described by Stephens [28].

We can now examine the differences between the MLE and the Bayesian estimate of fold change from *apeglm* (Fig. 9). First the MLE:

```
plotMA(res, ylim=c(-4,4),
       colNonSig="grey60", colSig="blue", colLine="grey40")
```

The genes passing a 10% FDR threshold are colored in blue.

Note the wide variability on the left side of the plot. This is mostly due to imprecision in our estimates.

After applying the Bayesian shrinkage procedure, the variability due to imprecision on the left side of the plot is reduced (Fig. 10). In the paper by Zhu et al. [27], it is demonstrated that the shrunken LFC are better suited for ranking genes by effect size.

```
plotMA(lfc, ylim=c(-4,4),
       colNonSig="grey60", colSig="blue", colLine="grey40")
```

For ranking genes by effect size, one would rank by `abs (log2FoldChange)` (with `decreasing=TRUE`), instead of by padj.

In the next series of code chunks, I demonstrate how to add additional identifiers, such as gene symbols, to the MA-plot. I will select a subset of genes to add labels, here filtering the results table to a set of genes based on `baseMean` and `log2FoldChange` (these choices are arbitrary, solely for demonstration). For a given experiment, it would make more sense to pick out relevant genes by both significance, effect size, and biological interpretation. Because I have not reordered the `lfc` results table, I can add the SYM to the results table, and the of interest (Fig. 11).

```
lfc$SYMBOL <- mcols(dds)$SYMBOL
tab <- lfc %>% as.data.frame %>%
         filter(between(baseMean, 1e4, 1e5),
                between(abs(log2FoldChange), 1, 4))
tab <- tab[complete.cases(tab),]
```

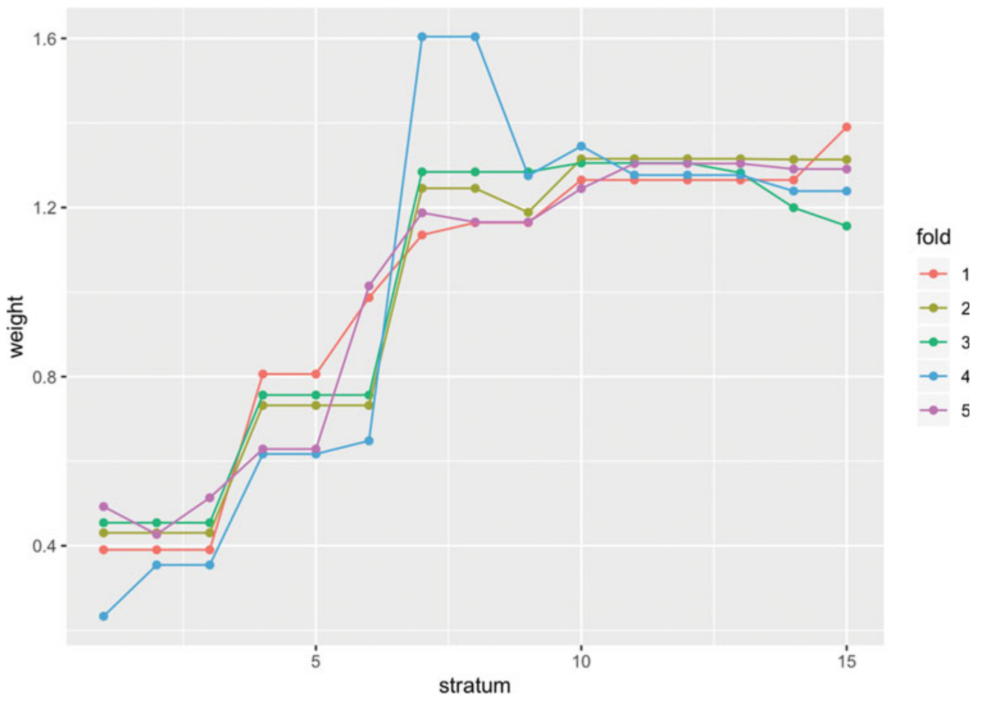


Fig. 13 Plot of the estimated hypothesis weights, over the mean of scaled counts, as computed by IHW

I now highlight the genes in `tab` using the `points` and `text` functions.

```
plotMA(lfc, ylim=c(-4,4),
       colNonSig="grey60", colSig="blue", colLine="grey40")
with(tab, {
        points(baseMean, log2FoldChange, cex=2, col="blue")
        text(baseMean, log2FoldChange, SYMBOL, pos=4, col="blue")
})
```

That worked, although it is not easy to see the labels for a set of overlapping points below the horizontal axis. I can make a nicer looking plot using *ggplot2*. First I create a data frame that I will pass to the `ggplot` function.

```
dat <- as.data.frame(lfc)
dat <- dat[complete.cases(dat),]
dat <- dat %>% mutate(sig = ifelse(padj < .1, "Y", "N"))
tab$sig <- "Y"
```

The following `ggplot` code chunk recreates the MA-plot that is built into *DESeq2*, but also uses the *ggrepel* package to make sure the point labels for our genes of interest do not overlap (Fig. 12):

```
library(ggplot2)
library(ggrepel)
ggplot(dat, aes(baseMean, log2FoldChange, col=sig, label=SYMBOL)) +
        geom_point() + scale_x_log10() +
        xlab("mean of normalized counts") +
        ylab("log fold change") +
        geom_hline(yintercept=0, col="grey40") +
        scale_color_manual(values=c("grey60", "blue")) +
        geom_point(data=tab, shape=1, size=5, show.legend=FALSE) +
        geom_label_repel(data=tab,
                         nudge_x = 1,
                         nudge_y = 2*sign(tab$log2FoldChange),
                         show.legend=FALSE)
```

3.3 Hypothesis Weighting

In the last series of code chunks, I will demonstrate the use of Independent Hypothesis Weighting (IHW) [19], in lieu of the more simplistic mean count filtering that is used in `results` by default. Instead of finding a threshold on the mean of scaled counts that optimizes the number of rejected hypotheses following Benjamini–Hochberg correction [29], the *IHW* package finds an optimal weighting of the hypotheses that maximizes power while still controlling the FDR.

To use *IHW* instead of mean count thresholding, I pass the `ihw` function to the `filterFun` argument of `results`. A similar number of genes are detected as differentially expressed in this case.

```
suppressPackageStartupMessages(library(IHW))

res.ihw <- results(dds, name="lineOCT4.conditiontrt", filterFun=ihw)
summary(res.ihw)

##
## out of 23524 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1630, 6.9%
## LFC < 0 (down)     : 1974, 8.4%
## outliers [1]       : 3, 0.013%
## [1] see 'cooksCutoff' argument of ?results
## see metadata(res)$ihwResult on hypothesis weighting
```

We can observe the weighting of hypotheses for various mean count ranges, across the cross-validation folds (Fig. 13). For more details on the methods used here, consult the *IHW* package vignette and the publication by Ignatiadis et al. [19].

```
ihw.obj <- metadata(res.ihw)$ihwResult
plot(ihw.obj)
```

Again, we can show that a similar set of genes were found in this case although in general the *IHW* procedure can outperform the simple filtering rules that `results` uses by default.

```
table(filter=res$padj < .05, IHW=res.ihw$padj < .05)

##          IHW
## filter   FALSE  TRUE
##    FALSE 20616   145
##    TRUE     95  2665
```

4 Transcript Expression

In the previous sections, I showed importing transcript-level data with *tximeta*, summarizing the data to gene-level counts, and modeling the gene-level counts with *DESeq2*. Now I return to transcript-level data and demonstrate how we can perform statistical testing on transcripts, that is, across all the isoforms of all the genes. There are two important aspects to consider when performing transcript-level analysis:

1. Uncertainty—Because the isoforms of a gene often have a considerable amount of sequence similarity resulting from shared exons, and because short read RNA-seq protocols involve generating fragments that do not span the entire transcript, there can be considerable uncertainty in assigning a given fragment to a particular transcript (see next section for

a discussion of long read protocols). The uncertainty is not constant across the transcripts, and depends on many factors, some inherent to the gene model, such as the size of the alternative exons, and some inherent to the experiment, such as the sequencing depth, fragment length, read length, and technical biases producing nonuniform coverage.

2. Isoform switching—We may perform testing for differential *expression* of each transcript (I will perform this analysis in this chapter), or we may also consider testing whether the *usage* of the isoforms within a gene changes across condition. The latter question is often termed *differential transcript usage* (DTU), and can be related to *differential transcript expression* (DTE), but they are not identical questions. For example, if all of the isoforms of a gene increase in their expression across condition with equal fold change, this is an example of differential gene expression (DGE) and DTE, but not DTU, as the proportions of the individual isoforms did not change.

Regarding isoform switching, one reference which explores Bioconductor packages that can be used to detect DTU is the *rnaseqDTU* Bioconductor workflow [2]. This workflow demonstrates optimal filtering techniques [30], how the methods *DEXSeq* [31] and *DRIMSeq* [32] can be applied to estimated transcript counts, and how *stageR* [33] can be utilized to detect which genes and which isoforms contain evidence of DTU while controlling overall error rates.

I will first introduce the experimental data, and then discuss various approaches used to analyze transcript-level data. I will load some processed RNA-seq data from an experiment by Alasoo et al. [34], a subset of which is available in the *macrophage* Biconductor package. The experiment involved measuring transcription in macrophage cell lines from a number of human donors, both untreated, as well as treated with IFNg, *Salmonella*, and IFNg combined with *Salmonella*. Here I will focus on the samples that were untreated and treated with IFNg. As each cell line was from a human donor, I will also control for a baseline donor effect when comparing across treatment.

The *macrophage* dataset has paired samples from six of the donors (all female), and has been quantified using *Salmon*. One unique aspect of *Salmon* is that it allows for GC bias correction at the fragment level during quantification, which is critical for reliable identification of the correct expressed isoform in experiments that have nonuniform coverage along the transcripts [6, 35].

Here I will perform differential transcript expression (DTE) analysis. A key aspect, compared to gene-level analysis, is that there is much more uncertainty in the assignments of fragments to transcripts. A number of statistical methods have been proposed to take this measurement uncertainty into account when performing downstream testing, including *BitSeq* [36], *mmdiff* [37], *IsoDE* [38], and *Sleuth* [39], the latter which leverages bootstrap quantification estimates from the *kallisto* [40] quantification method. These methods incorporate measurement uncertainty into parametric models for differential expression where biological variability is also modeled. The exception is *IsoDE* which compares bootstrap distributions of transcript expression for two samples at a time.

Here, I will use a nonparametric method that takes into account both inferential uncertainty of fragment assignments, as well as biological variability across samples, called *Swish* [41], which is available in the *fishpond* Bioconductor package. *Swish* stands for "SAMseq With Inferential Samples Helps," as it is based on the existing statistical method for differential gene expression, *SAMseq* [42]. The key idea is to make use of nonparametric testing methods such as the Mann-Whitney Wilcoxon statistic, which operate only on the ranks of the data across samples. The original *SAMseq* method performed resampling of the counts in order to account for sequencing depth differences. Here, *Swish* will make use of multiple values in each cell of the count matrix that were computed by the *Salmon* software, using a technique called Gibbs sampling. For more details on the Gibbs sampling procedure, consult the publication of *Salmon* [6] and *mmseq* [43]. Finally, the test statistics are averaged over the multiple versions (or "inferential replicates") of the counts matrix and q-values for false discovery rate estimation are computed via a permutation technique [44].

I begin by locating the files in the *macrophage* package. As before, this step is not useful for a typical RNA-seq workflow, as the data will not be contained in an R package, but contained in some directory on a server or compute cluster. In lieu of the `system.file` command below, you should just specify the `dir` variable to be a path to the files, for example, `/path/to/data/dir`.

```
dir <- system.file("extdata", package="macrophage")
list.files(dir)

## [1] "coldata.csv"
## [2] "errs"
## [3] "gencode.v29_salmon_0.12.0"
## [4] "gencode.v29.annotation.gtf.gz"
## [5] "PRJEB18997.txt"
## [6] "quants"
## [7] "supp_table_1.csv"
## [8] "supp_table_7.csv"
```

I then read in the sample table, and use *dplyr* to select certain columns, convert columns into factors, and add a new column pointing to the quantification files.

```
library(readr)
library(dplyr)
coldata <- read_csv(file.path(dir,"coldata.csv"))
```

```
## Parsed with column specification:
## cols(
##   names = col_character(),
##   sample_id = col_character(),
##   line_id = col_character(),
##   replicate = col_double(),
##   condition_name = col_character(),
##   macrophage_harvest = col_character(),
##   salmonella_date = col_character(),
##   ng_ul_mean = col_double(),
##   rna_extraction = col_character(),
##   rna_submit = col_character(),
##   library_pool = col_character(),
##   chemistry = col_character(),
##   rna_auto = col_double()
## )
```

```
lvls <- c("naive","IFNg","SL1344","IFNg_SL1344")
coldata <- coldata %>%
  dplyr::select(names, id=sample_id, line=line_id,
                condition=condition_name) %>%
  mutate(line=factor(line),
         condition=factor(condition, levels=lvls),
         files=file.path(dir, "quants", names, "quant.sf.gz"))
```

I will only consider for this demonstration the untreated and IFNg treated samples:

```
coldata <- coldata %>% filter(condition %in% c("naive","IFNg"))
coldata$condition <- droplevels(coldata$condition)
```

The `coldata` sample table now looks like:

```
head(coldata)
```

```
## # A tibble: 6 x 5
##   names     id     line  condition files
##   <chr>     <chr>  <fct> <fct>     <chr>
## 1 SAMEA103… diku_A diku… naive     /Library/Frameworks/R.framewo…
## 2 SAMEA103… diku_B diku… IFNg      /Library/Frameworks/R.framewo…
## 3 SAMEA103… eiwy_A eiwy… naive     /Library/Frameworks/R.framewo…
## 4 SAMEA103… eiwy_B eiwy… IFNg      /Library/Frameworks/R.framewo…
## 5 SAMEA103… fikt_A fikt… naive     /Library/Frameworks/R.framewo…
## 6 SAMEA103… fikt_B fikt… IFNg      /Library/Frameworks/R.framewo…
```

Test that all the files exist as I specified:

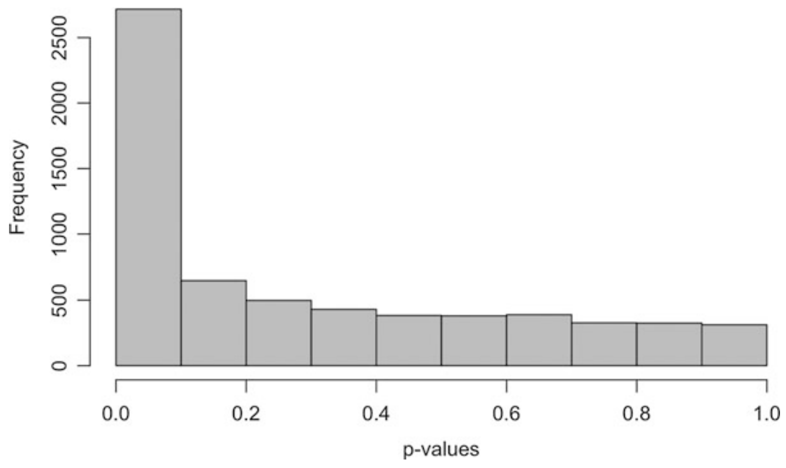


Fig. 14 Distribution of *p*-values from the Swish method for testing differential transcript expression

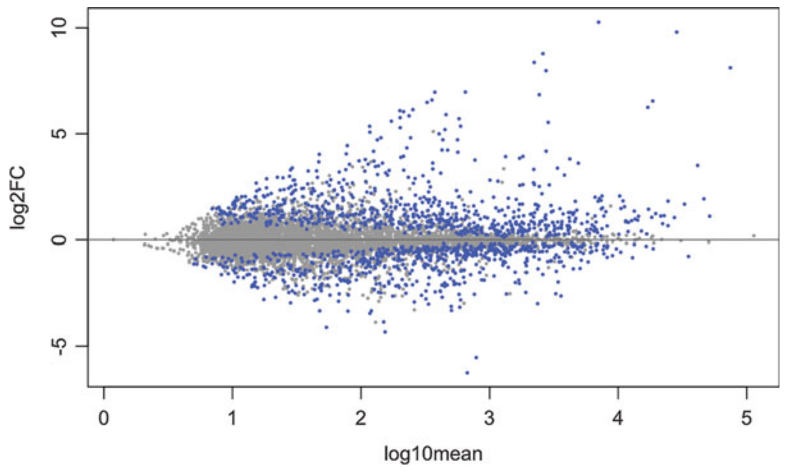


Fig. 15 MA-plot of the log2 fold changes over the mean of scaled counts, for differential transcript expression by Swish

```
all(file.exists(coldata$files))
## [1] TRUE
```

As before, I use *tximeta* to read in the quantification data. This time I do not set `dropInfReps=TRUE`, as I will need the inferential replicates created by *Salmon* to perform DTE with *Swish*. The inferential replicates allows the analysis to take into account the uncertainty of fragment assignment to transcripts.

```
library(tximeta)
suppressPackageStartupMessages(library(SummarizedExperiment))
y <- tximeta(coldata)
```

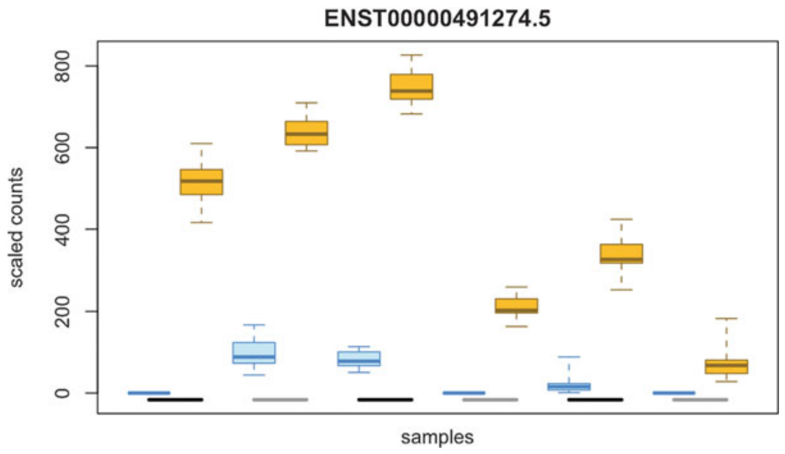


Fig. 16 Plot of the estimated counts from Salmon over samples, for a particular transcript with a *p*-value less than 0.05 and a log2 fold change greater than 4. The boxes represent the distribution of inferential replicate estimated counts for each sample

```
## importing quantifications
## reading in files with read_tsv
## 1 2 3 4 5 6 7 8 9 10 11 12
## found matching linked transcriptome:
## [ GENCODE - Homo sapiens - release 29 ]
## loading existing TxDb created: 2020-03-12 01:46:43
## loading existing transcript ranges created: 2020-03-12
01:48:11
## fetching genome info for GENCODE
```

For speed of the demonstration, I subset to only the transcripts on chromosome 1 (this would not be recommended for a typical analysis).

```
y <- y[seqnames(y) == "chr1",]
```

I load the *fishpond* package, which contains the methods for running *Swish*. There are three basic steps: scaling of the inferential replicates to make them comparable despite different sequencing depth, filtering out lowly expressed transcripts, and the testing itself. The scaling method by default uses the median ratio method of *DESeq* [24]. The `labelKeep` function by default will keep those transcripts with three or more samples with a count of 10 or higher. For scRNA-seq with UMI deduplication, it is recommended to lower the minimal count to a lower value such as 3. The minimal number of samples can be increased for experiments with many samples.

```
library(fishpond)
y <- scaleInfReps(y, quiet=TRUE)
y <- labelKeep(y)
y <- y[mcols(y)$keep,]
```

Because the method makes use of permutations, it is required to set a seed for computational reproducibility. I specify to test across the `condition` variable, while controlling for a pairing variable `line`. The `line` variable indicates which donor the cell line came from.

```
set.seed(1)
y <- swish(y, x="condition", pair="line", quiet=TRUE)
## note: less permutations are available than requested
## 64 are available
```

After running `swish`, all of the results are stored in the metadata columns (`mcols`) of the object `y`. I look to see how many transcripts have a small *q*-value (analogous to an adjusted *p*-value, this should provide a set with a nominal FDR control).

```
names(mcols(y))

##  [1] "tx_id"     "gene_id"   "tx_name"   "log10mean" "keep"
##  [6] "stat"      "log2FC"    "pvalue"    "locfdr"    "qvalue"

table(mcols(y)$qvalue < .05)

##
## FALSE  TRUE
##  5081  1329
```

One important aspect in testing across many features, in particular where the uncertainty level is so heterogeneous, is to consider if the *p*-value distribution is roughly uniform, with the exception of the rejected tests. Here *Swish* provides a roughly uniform distribution, with a spike on the left side representing the rejections of the null hypothesis (Fig. 14).

```
hist(mcols(y)$pvalue, col="grey",
     main="", xlab="p-values")
```

As with *DESeq2* I can make an MA-plot, with the differential transcripts highlighted in blue (here at 5% FDR) (Fig. 15).

```
plotMASwish(y, alpha=.05)
```

I can also examine individual transcripts with evidence of differential expression. As each sample is represented by a distribution of possible estimated counts from *Salmon*, *Swish* uses boxplots to represent the differences in expression across samples (Fig. 16):

```
idx <- with(mcols(y), which(pvalue < .05 & log2FC > 4))
plotInfReps(y, idx[1], x="condition", cov="line", xaxis=FALSE)
```

This section gave a basic introduction to DTE using nonparametric testing with the *Swish* method. For more details on transcript-level analysis, it is recommended to consult the *fishpond* Bioconductor package vignette, or the *rnaseqDTU* workflow on Bioconductor [2].

5 Limitations and Extensions

In this final section, I discuss limitations to the methods presented earlier, and extensions for analyzing high dimensional counts in contexts beyond what was previously covered.

1. *Single cell RNA-seq*—The *DESeq2* framework shown in the gene-level analysis section was designed for bulk RNA-seq, in which the Negative Binomial GLM assessing differences across samples was suitable both in terms of distribution and in terms of answering many biological questions of interest. In single cell RNA-seq, there are new considerations and questions of interest. One aspect is that, with UMI barcoding, there is a need for quantification methods that resolve errors and deduplicate the read data into molecule counts per cell. The *alevin* method [10], packaged within the *Salmon* software, can accomplish this UMI deduplication, and can resolve the increased rate of multimapping reads seen in $3'$ tagged sequencing, through an approach similar to that taken by *Salmon*. The quantification from *alevin* can be easily imported into R/Bioconductor using the *tximeta* software seen in the quantification section.

   After quantification, there are many choices regarding the analysis pipeline, I refer to Bioconductor's online book for single cell analysis, and recent reviews for systematic comparisons. Amezquita et al. [45] have recently published an overview and online book for performing analysis of scRNA-seq data using Bioconductor packages. Soneson and Robinson

[46] evaluates methods for detecting differences in expression across groups of cells. Sun et al. [47] evaluates methods for dimension reduction, which is often performed in the context of cell clustering and lineage reconstruction. Duo et al. [48] evaluates methods for clustering to recover sub-populations of cells. Finally, I note that the NB methods shown in the gene-level chapter can be combined with other statistical methods to add and model a zero component, in the case that the Negative Binomial is not a suitable distribution [49]. The zero component may not be needed for all scRNA-seq datasets, however, in particular if UMI deduplication is possible.

2. *Long reads*—The data presented in previous sections involved sequencing relatively short sequences of the cDNA fragments. They sequences are short in the sense that they do not come close to capturing the entire sequence of the transcript for most mammalian transcripts. However, new technologies have emerged in the past decade that allow for high-throughput sequencing of lengths that approach the entire transcript length. This necessitates new methods for alignment (the long sequences nevertheless have a higher error rate than the "short" reads). One of the most popular methods for aligning long reads is *minimap2* [50]. Following alignment, it is possible to again quantify expression using *Salmon* and import the data into R/Bioconductor using *tximeta*. A systematic evaluation of quantification using the Nanopore long read technology has been performed by Soneson et al. [51]. A pipeline for long read mapping with *minimap2* and quantification with *Salmon* has been recently published with an associated GitHub repository [52]. Finally, a review of bioinformatic pipelines for long read data analysis has recently been published by Amarasinghe et al. [53].
3. *Genetic variation*—An aspect not explored in the previous sections was genetic variation across the samples in the exonic sequence. One analysis of interest is to identify common genetic variants in the exonic sequence, and to quantify, among the samples that are heterozygous for a given exonic SNP, the expression of each allele. Best practices for allelic expression analysis have been presented by Castel et al. [54], and an evaluation of EM-based methods for assessing allelic expression have been proposed and compared by Raghupathy et al. [55]. Aside from interest in quantifying allelic expression in the presence of heterozygous exonic positions, Srivastava et al. [56] have examined the effect of genetic variation on transcript and gene expression quantification.
4. *Microbiome*—I have described here various methods for analyzing counts reflecting the abundance of RNA molecules across samples. Another type of high dimensional count dataset with

similar but distinct analysis considerations is that produced in a microbiome or metagenomic study, in which the counts reflect the abundance of certain taxa across samples. The count data is arranged in a similar format to gene expression, but with the taxa replacing the transcripts or genes on the rows of the matrix.

While many have considered using gene expression normalization and testing methods for analyzing this type of data, a number of the assumptions used in gene expression models may be invalid for particular microbiome datasets. In particular, I demonstrated in the first exploration of gene expression counts that there were thousands of features in which the changes from sample to sample were minimal. There was a clear center of the distribution of log fold changes that could be used to estimate the size factors for scaling normalization across samples. In particular microbiome studies, this assumption may not fit, as there may not be a group of taxa that can be assumed roughly equally abundant across all samples in a dataset. In addition, there may be too few taxa, such that the Poisson modeling assumption no longer makes sense, and so a compositional model may better capture the distributional properties [57]. A recent benchmarking effort compares compositional methods as well as single cell RNA-seq methods for analyzing microbiome datasets for differences in abundance of taxa [58].

Alternative pipelines for analyzing microbiome abundance data have been detailed by Callahan et al. [59]. There may be more interesting and relevant approaches to modeling the counts besides the GLM, and latent variable models are considered and applied to microbiome datasets recently by Sankaran and Holmes [60]. Finally, statistical considerations of various diversity measures for count-based microbiome studies have been explored recently by Willis [61].

6 Session Information

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.5
##
## Matrix products: default
## BLAS:   /System/Library/Frameworks/Accelerate.framework/Versions/A/Fram
eworks/vecLib.framework/Versions/A/libBLAS.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libR
lapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices datasets
## [7] utils     methods   base
##
## other attached packages:
##  [1] apeglm_1.11.0               GenomicFeatures_1.41.0
##  [3] bookdown_0.20               BiocParallel_1.23.0
##  [5] testthat_2.3.2              rmarkdown_2.3
##  [7] devtools_2.3.0              usethis_1.6.1
##  [9] fishpond_1.5.33             IHW_1.17.0
## [11] ggrepel_0.8.2               ggplot2_3.3.0
## [13] DESeq2_1.29.3               rafalib_1.0.0
## [15] org.Mm.eg.db_3.11.1         AnnotationDbi_1.51.0
## [17] SummarizedExperiment_1.19.2 DelayedArray_0.15.1
## [19] matrixStats_0.56.0          Biobase_2.49.0
## [21] GenomicRanges_1.41.1        GenomeInfoDb_1.25.0
## [23] IRanges_2.23.4              S4Vectors_0.27.5
## [25] BiocGenerics_0.35.4         tximeta_1.7.3
## [27] dplyr_0.8.5                 readr_1.3.1
##
## loaded via a namespace (and not attached):
##   [1] backports_1.1.6               AnnotationHub_2.21.0
      ##   [3] BiocFileCache_1.13.0          servr_0.17
      ##   [5] plyr_1.8.6                    lazyeval_0.2.2
      ##   [7] splines_4.0.0                 lpsymphony_1.17.0
      ##   [9] digest_0.6.25                 ensembldb_2.13.1
      ##  [11] htmltools_0.4.0               fansi_0.4.1
      ##  [13] magrittr_1.5                  memoise_1.1.0
      ##  [15] remotes_2.1.1                 Biostrings_2.57.0
      ##  [17] annotate_1.67.0               askpass_1.1
      ##  [19] bdsmatrix_1.3-4               prettyunits_1.1.1
      ##  [21] colorspace_1.4-1              blob_1.2.1
      ##  [23] rappdirs_0.3.1                xfun_0.13
      ##  [25] callr_3.4.3                   crayon_1.3.4
      ##  [27] RCurl_1.98-1.2                jsonlite_1.6.1
      ##  [29] tximport_1.17.0               genefilter_1.71.0
      ##  [31] survival_3.1-12               glue_1.4.0
      ##  [33] gtable_0.3.0                  zlibbioc_1.35.0
      ##  [35] XVector_0.29.0                pkgbuild_1.0.8
      ##  [37] abind_1.4-5                   scales_1.1.1
      ##  [39] mvtnorm_1.1-0                 DBI_1.1.0
      ##  [41] Rcpp_1.0.4.6                  xtable_1.8-4
      ##  [43] progress_1.2.2                emdbook_1.3.12
      ##  [45] bit_1.1-15.2                  httr_1.4.1
      ##  [47] RColorBrewer_1.1-2            ellipsis_0.3.0
      ##  [49] pkgconfig_2.0.3               XML_3.99-0.3
      ##  [51] farver_2.0.3                  dbplyr_1.4.3
      ##  [53] locfit_1.5-9.4                utf8_1.1.4
      ##  [55] reshape2_1.4.4                tidyselect_1.0.0
      ##  [57] labeling_0.3                  rlang_0.4.6
      ##  [59] later_1.0.0                   munsell_0.5.0
      ##  [61] BiocVersion_3.12.0            tools_4.0.0
      ##  [63] cli_2.0.2                     RSQLite_2.2.0
      ##  [65] fdrtool_1.2.15                evaluate_0.14
      ##  [67] stringr_1.4.0                 fastmap_1.0.1
      ##  [69] yaml_2.2.1                    processx_3.4.2
      ##  [71] knitr_1.28                    bit64_0.9-7
      ##  [73] fs_1.4.1                      purrr_0.3.4
      ##  [75] AnnotationFilter_1.13.0       mime_0.9
      ##  [77] slam_0.1-47                   biomaRt_2.45.0
      ##  [79] compiler_4.0.0                rstudioapi_0.11
      ##  [81] curl_4.3                      interactiveDisplayBase_1.27.0
      ##  [83] tibble_3.0.1                  geneplotter_1.67.0
      ##  [85] stringi_1.4.6                 ps_1.3.3
      ##  [87] desc_1.2.0                    lattice_0.20-41
      ##  [89] ProtGenerics_1.21.0           Matrix_1.2-18
      ##  [91] vctrs_0.2.4                   pillar_1.4.4
      ##  [93] lifecycle_0.2.0               BiocManager_1.30.10
      ##  [95] bitops_1.0-6                  qvalue_2.21.0
      ##  [97] httpuv_1.5.2                  rtracklayer_1.49.3
      ##  [99] R6_2.4.1                      promises_1.1.0
      ## [101] codetools_0.2-16              sessioninfo_1.1.1
      ## [103] MASS_7.3-51.6                 gtools_3.8.2
      ## [105] assertthat_0.2.1              pkgload_1.0.2
      ## [107] openssl_1.4.1                 rprojroot_1.3-2
      ## [109] withr_2.2.0                   GenomicAlignments_1.25.0
      ## [111] Rsamtools_2.5.0               GenomeInfoDbData_1.2.3
      ## [113] hms_0.5.3                     grid_4.0.0
      ## [115] coda_0.19-3                   bbmle_1.0.23.1
      ## [117] numDeriv_2016.8-1.1           shiny_1.4.0.2
      ## [119] tinytex_0.22
```

References

1. Love M, Anders S, Kim V, Huber W (2015) RNA-seq workflow: gene-level exploratory analysis and differential expression. F1000research 4:1070
2. Love M, Soneson C, Patro R (2018) Swimming downstream: statistical analysis of differential transcript usage following salmon quantification. F1000research 7:952
3. Van den Berge K, Hembach KM, Soneson C, Tiberi S, Clement L et al (2019) RNA sequencing data: Hitchhiker's guide to expression analysis. Ann Rev Biomed Data Sci 2(1):139–173
4. Ewels P, Magnusson M, Lundin S, Käller M (2016) MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics 32(19):3047–3048
5. King HW, Klose RJ (2017) The pioneer factor oct4 requires the chromatin remodeller brg1 to support gene regulatory element function in mouse embryonic stem cells. Elife 6:e22631
6. Patro R, Duggal G, Love M, Irizarry R, Kingsford C (2017) Salmon provides fast and bias-aware quantification of transcript expression. Nat Methods 14:417–419
7. Köster J, Rahmann S (2012) Snakemake—a scalable bioinformatics workflow engine. Bioinformatics 28(19):2520–2522
8. Huber W, Carey VJ, Gentleman R, Anders S, Carlson M et al (2015) Orchestrating high-throughput genomic analysis with Bioconductor. Nat Methods 12(2):115–121
9. Love MI, Soneson C, Hickey PF, Johnson LK, Pierce NT et al (2020) Tximeta: reference sequence checksums for provenance identification in RNA-seq. PLoS Comput Biol 16(2): e1007664
10. Srivastava A, Malik L, Smith TS, Sudbery I, Patro R (2019) Alevin efficiently estimates accurate gene abundances from dscRNA-seq data. Genome Biol 20:65
11. Frankish A, GENCODE-consoritum, Flicek P. (2018) GENCODE reference annotation for the human and mouse genomes. Nucleic Acids Res 47(D1):D766–D773
12. Soneson C, Love MI, Robinson M (2015) Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences. F1000research 4:1521
13. Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M et al (2013) Software for computing and annotating genomic ranges. PLoS Comput Biol 9(8):e1003118
14. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15(12):550
15. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26(1):139
16. McCarthy DJ, Chen Y, Smyth GK (2012) Differential expression analysis of multifactor RNA-seq experiments with respect to biological variation. Nucleic Acids Res 40:4288–4297
17. Law CW, Chen Y, Shi W, Smyth GK (2014) voom: precision weights unlock linear model analysis tools for RNA-seq read counts. Genome Biol 15(2):29
18. Wu H, Wang C, Wu Z (2012) A new shrinkage estimator for dispersion improves differential expression detection in RNA-seq data. Biostatistics 14(2):232–243
19. Ignatiadis N, Klaus B, Zaugg J, Huber W (2016) Data-driven hypothesis weighting increases detection power in genome-scale multiple testing. Nat Methods 13(7):577–580
20. Dudoit S, Yang YH, Callow MJ, Speed TP (2002) Statistical methods for identifying differentially expressed genes in replicated cDNA

microarray experiments. Stat Sin 12 (1):111–139

21. Roberts CJ, Nelson B, Marton MJ, Stoughton R, Meyer MR et al (2000) Signaling and circuitry of multiple mapk pathways revealed by a matrix of global gene expression profiles. Science 287(5454):873–880
22. Cox DR, Reid N (1987) Parameter orthogonality and approximate conditional inference. J R Stat Soc B 49(1):1–39
23. Tibshirani R (1988) Estimating transformations for regression via additivity and variance stabilization. J Am Stat Assoc 83:394–405
24. Anders S, Huber W (2010) Differential expression analysis for sequence count data. Genome Biol 11:R106
25. Witten DM (2011) Classification and clustering of sequencing data using a Poisson model. Annal Appl Stat 5(4):2493–2518
26. Townes FW, Hicks SC, Aryee MJ, Irizarry RA (2019) Feature selection and dimension reduction for single cell RNA-seq based on a multinomial model. Genome Biol 20:295
27. Zhu A, Ibrahim JG, Love MI (2018) Heavy-tailed prior distributions for sequence count data: removing the noise and preserving large differences. Bioinformatics 35(12):2084–2092
28. Stephens M (2016) False discovery rates: a new deal. Biostatistics 18(2):41
29. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc B 57:289–300
30. Soneson C, Matthes KL, Nowicka M, Law CW, Robinson MD (2016) Isoform prefiltering improves performance of count-based methods for analysis of differential transcript usage. Genome Biol 17(1):12
31. Anders S, Reyes A, Huber W (2012) Detecting differential usage of exons from RNA-seq data. Genome Res 22(10):2008–2017
32. Nowicka M, Robinson M (2016) DRIMSeq: a Dirichlet-multinomial framework for multivariate count outcomes in genomics. F1000research 5:1356
33. Van den Berge K, Soneson C, Robinson MD, Clement L (2017) stageR: a general stage-wise method for controlling the gene-level false discovery rate in differential expression and differential transcript usage. Genome Biol 18(1):151
34. Alasoo K, Rodrigues J, Mukhopadhyay S, Knights A, Mann A et al (2018) Shared genetic effects on chromatin and gene expression indicate a role for enhancer priming in immune response. Nat Genet 50:424–431
35. Love MI, Hogenesch JB, Irizarry RA (2016) Modeling of RNA-seq fragment sequence bias reduces systematic errors in transcript abundance estimation. Nat Biotechnol 34 (12):1287–1291
36. Glaus P, Honkela A, Rattray M (2012) Identifying differentially expressed transcripts from RNA-seq data with biological variation. Bioinformatics 28(13):1721–1728
37. Turro E, Astle WJ, Tavaré S (2013) Flexible analysis of RNA-seq data using mixed effects models. Bioinformatics 30(2):180–188
38. Al Seesi S, Temate-Tiagueu Y, Zelikovsky A, Măndoiu II (2014) Bootstrap-based differential gene expression analysis for RNA-seq data with and without replicates. BMC Genomics 15(Suppl 8):S2
39. Pimentel H, Bray NL, Puente S, Melsted P, Pachter L (2017) Differential analysis of RNA-seq incorporating quantification uncertainty. Nat Methods 14(7):687–690
40. Bray NL, Pimentel H, Melsted P, Pachter L (2016) Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol 34(5):525
41. Zhu A, Srivastava A, Ibrahim J, Patro R, Love M (2019) Nonparametric expression analysis using inferential replicate counts. Nucleic Acids Res 47(18):e105
42. Li J, Tibshirani R (2011) Finding consistent patterns: a nonparametric approach for identifying differential expression in RNA-Seq data. Stat Methods Med Res 22(5):519–536
43. Turro E, Su S-Y, Gonçalves Â, Coin LJ, Richardson S, Lewin A (2011) Haplotype and isoform specific expression estimation using multi-mapping RNA-seq reads. Genome Biol 12(2):R13
44. Storey J, Tibshirani R (2003) Statistical significance for genome-wide experiments. Proc Natl Acad Sci 100(16):9440–9445
45. Amezquita RA, Lun ATL, Becht E, Carey VJ, Carpp LN et al (2020) Orchestrating single-cell analysis with bioconductor. Nat Methods 17(2):137–145
46. Soneson C, Robinson MD (2018) Bias, robustness and scalability in single-cell differential expression analysis. Nat Methods 15 (4):255–261
47. Sun S, Zhu J, Ma Y, Zhou X (2019) Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. Genome Biology 20(1):269
48. Duo A, Robinson M, Soneson C (2018) A systematic performance evaluation of clustering methods for single-cell RNA-seq data. F1000research 7:1141

49. Van den Berge K, Perraudeau F, Soneson C, Love MI, Risso D et al (2018) Observation weights unlock bulk RNA-seq tools for zero inflation and single-cell applications. Genome Biol 19:24

50. Li H (2018) Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics 34 (18):3094–3100

51. Soneson C, Yao Y, Bratus-Neuenschwander A, Patrignani A, Robinson MD, Hussain S (2019) A comprehensive examination of nanopore native RNA sequencing for characterization of complex transcriptomes. Nat Commun 10 (1):3359

52. Cruz-Garcia L, O'Brien G, Sipos B, Mayes S, Love M et al (2019) Generation of a transcriptional radiation exposure signature in human blood using long-read nanopore sequencing. Radiat Res 193(2):143–154

53. Amarasinghe SL, Su S, Dong X, Zappia L, Ritchie ME, Gouil Q (2020) Opportunities and challenges in long-read sequencing data analysis. Genome Biol 21(1):30

54. Castel SE, Levy-Moonshine A, Mohammadi P, Banks E, Lappalainen T (2015) Tools and best practices for data processing in allelic expression analysis. Genome Biol 16(1):195

55. Raghupathy N, Choi K, Vincent MJ, Beane GL, Sheppard KS et al (2018) Hierarchical analysis of RNA-seq reads improves the accuracy of allele-specific expression. Bioinformatics 34(13):2177–2184

56. Srivastava A, Malik L, Sarkar H, Zakeri M, Almodaresi F et al (2019) Alignment and mapping methodology influence transcript abundance estimation. Genome Biol 21:239

57. Fernandes AD, Reid JN, Macklaim JM, McMurrough TA, Edgell DR, Gloor GB (2014) Unifying the analysis of high-throughput sequencing datasets: characterizing RNA-seq, 16S rRNA gene sequencing and selective growth experiments by compositional data analysis. Microbiome 2(1):15

58. Calgaro M, Romualdi C, Waldron L, Risso D, Vitulo N (2020) Assessment of single cell RNA-seq statistical methods on microbiome data. Genome Biol 21:191

59. Callahan B, Sankaran K, Fukuyama J, McMurdie P, Holmes S (2016) Bioconductor workflow for microbiome data analysis: from raw reads to community analyses. F1000research 5:1492

60. Sankaran K, Holmes SP (2018) Latent variable modeling for the microbiome. Biostatistics 20 (4):599–614

61. Willis AD (2019) Rarefaction, alpha diversity, and statistics. Front Microbiol 10:2407

Chapter 8

QuickIsoSeq for Isoform Quantification in Large-Scale RNA Sequencing

Ramya Gamini, Reiko Nakashima, Wen He, Ying Huang, Ying Zhang, and Shanrong Zhao

Abstract

RNA-sequencing (RNA-seq) is a powerful technology for transcriptome profiling. While most RNA-seq projects focus on gene-level quantification and analysis, there is growing evidence that most mammalian genes are alternatively spliced to generate different isoforms that can be subsequently translated to protein molecules with diverse or even opposing biological functions. Quantifying the expression levels of these isoforms is key to understanding the genes biological functions in healthy tissues and the progression of diseases. Among open source tools developed for isoform quantification, *Salmon*, *Kallisto*, and *RSEM* are recommended based upon previous systematic evaluation of these tools using both experimental and simulated RNA-seq datasets. However, isoform quantification in practical RNA-seq data analysis needs to deal with many QC issues, such as the abundance of rRNAs in mRNA-seq, the efficiency of globin RNA depletion in whole blood samples, and potential sample swapping. To overcome these practical challenges, *QuickIsoSeq* was developed for large-scale RNA-seq isoform quantification along with QC. In this chapter, we describe the pipeline and detailed the steps required to deploy and use it to analyze RNA-seq datasets in practice. The *QuickIsoSeq* package can be downloaded from https://github.com/shanrongzhao/QuickIsoSeq.

Key words RNA-seq, Isoform quantification, *QuickIsoSeq*, RNA-seq pipeline

1 Introduction

RNA-sequencing (RNA-seq) is a next-generation sequencing technique that allows an in-depth examination of the transcriptome. With decreasing sequencing cost, RNA-seq has become an attractive approach to profile gene expression levels or transcript abundance [1, 2]. Recent large genome-scale studies concluded that almost all human multiexon genes could be spliced into multiple transcript isoforms [3]. There are 58,037 annotated human genes and 198,093 isoforms in GENCODE v25 [4]. On average, there

Ramya Gamini and Reiko Nakashima contributed equally to this work.

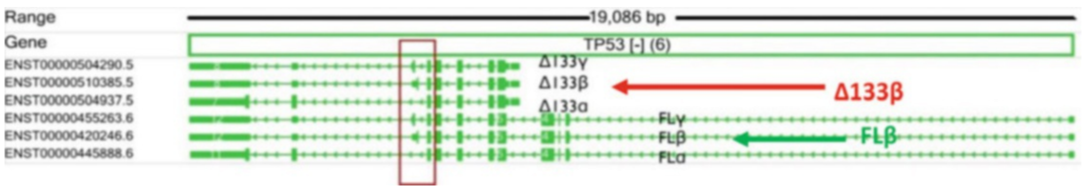


Fig. 1 TP53 has multiple isoforms. However, not all isoforms have the same role in tumor suppression. The Δ133β isoform inhibits apoptosis of tumor cells induced by the full-length FLβ isoform

are 3.4 annotated transcripts per human gene and if only protein-coding genes are considered, the ratio increases to 7:1 [4]. Isoforms from the same gene can be involved in distinct processes or even play opposite roles. The *TP53* gene, also known as tumor protein p53, is a case in point. The *TP53* gene is well studied and has a central role in the regulation of DNA-damaged cells [5–7]. However, not all *TP53* isoforms have the same role in tumor suppression (Fig. 1). For instance, the roles of Δ133β and full-length FLβ isoforms are opposite to each other. The Δ133β isoform inhibits apoptosis of tumor cells induced by the FLβ isoform [5]. In such cases, it is essential to obtain accurate quantification of expression at the transcript level to understand the function of each isoform.

Previously, we developed the *QuickRNASeq* pipeline [8, 9] for RNA-seq gene-level quantification, in which *featureCounts* [10] was chosen as the counting tool due to its simplicity and popularity. All multiple mapping reads and those reads mapped to gene overlapping regions are excluded from counting due to the ambiguity of read origins. As a result, the expression levels for some genes are significantly underestimated. For instance, the human gene *IL3RA* is encoded on both chrX and chrY, and its expression level is always zero when quantified by *featureCounts* because multiple mapping reads are excluded. More importantly, more insights can be gained from isoform-level RNA-seq data analyses over standard gene level analysis, as demonstrated previously [11]. Isoform quantification not only detects isoform-switching events that are masked by gene-level analysis, but also improves gene-level quantification accuracy by aggregating the transcript-level quantification results [12]. Thus, isoform quantification is recommended for all RNA-seq data analyses.

2 Materials and Methods

Although a number of open source tools for isoform quantification have been developed recently [13, 14], in practical RNA-seq data analysis, isoform quantification is more complicated than selecting a computational tool and running it. *QuickIsoSeq* is an integrated pipeline developed for large-scale RNA-seq projects.

2.1 Overview of QuickIsoSeq

QuickIsoSeq is built upon multiple best-in-class open source tools. In addition to isoform quantification, this application generates many useful QC (Quality Control) metrices to support practical RNA-seq data analysis. It is particularly designed to be run in a high-performance computing cluster (HPC) environment for large-scale RNA-seq projects. It offers a unifying interface to the underlying isoform quantification algorithms. The goal of *QuickIsoSeq* is to streamline the process of isoform quantification and improve efficiency and reproducibility in RNA-seq data analysis.

2.1.1 Computational Algorithms for Isoform Quantification

A number of packages have been developed to quantify expression at the transcript level including *RSEM* [15], *eXpress* [16], *TIGAR2* [17], and *Cufflinks* [18]. Most Recently, ultrafast alignment-free methods, such as *Sailfish* [19], *Salmon* [20] and *Kallisto* [21] have been developed to exploit the idea that precise alignments are not required to assign reads to their origins. *Salmon* is the most flexible tool with two modes of quantification. It can either process raw sequencing reads or take transcriptome-mapped BAM files as inputs. After a comprehensive evaluation of seven packages for isoform quantification [13], we found that alignment-free methods, such as *Salmon*, *Sailfish* and *Kallisto*, require less computational time while achieving similar or better accuracies compared with other methods. *Cufflinks* and *eXpress*, two alignment-dependent algorithms, have inferior accuracy performance. *TIGAR2* has overall good performance, but the run time and memory requirements render the tool less popular use. Considering both the accuracy and computational resources needed, *Salmon*, *Kallisto*, and *RSEM* were incorporated into the *QuickIsoSeq* package.

2.1.2 Challenges in Isoform Quantification and Functionalities of QuickIsoSeq

Based on our experience with in-house analyses of multiple RNA-seq datasets of varying sizes using open source tools, the main challenges, gaps, and bottlenecks for large-scale RNA-seq isoform quantification can be summarized as follows.

- It is hard to make the best choice of software packages and set software-specific parameters, as it often requires both an in-depth understanding of the algorithms and thorough benchmarking.
- It is challenging to make different open source tools work seamlessly in a pipeline, since quite often, the inputs and outputs of different algorithms are not compatible with each other. Additionally, most algorithms are implemented to process an individual sample. Consequently, the results of primary data analyses have to be further processed.
- *RSEM*, *Salmon*, and *Kallisto* require different inputs and command line parameters and generate output in different formats.

Take the strandness of a sample sequenced using Illumina's *TruSeq Stranded mRNA* as an example. The corresponding parameter in a RSEM command line is "*--forward-prob=0*"; while for *Salmon* and *Kalliso*, it is "*-l ISR*" and "*--rf-stranded*", respectively. Therefore, addition "bridge" scripts must be developed to handle the input/output difference in *RSEM*, *Salmon*, and *Kallisto*.

- It is common that some samples have low quality and often substitute samples are not available, especially for RNA-seq of clinical specimens. Such samples can cause bias in analysis and lead to misinterpretation of results. Therefore, it is necessary to establish stringent RNA-seq QC metrics to identify sample outliers that should be excluded from further downstream data analysis.
- For large-scale RNA-seq studies in which hundreds or even thousands of RNA samples are sequenced, it is common that some samples are mishandled and appear to be swapped or even sequenced more than once. Such errors can become a serious problem for downstream interpretation of results, especially for longitudinal sample analyses.

The *QuickIsoSeq* pipeline is implemented to meet the challenges above, as it performs more than just isoform quantification. The majority of *QuickIsoSeq* functionalities was developed to address a variety of issues in sample QC and integration. It can detect potential sample swapping, identify samples with issues in rRNA depletion and globin RNA (for blood samples) depletion, and flag those samples with low quality in an automatic fashion. The main output files from *QuickIsoSeq* are as follows:

- The library size, summary of read mapping and counts for individual samples.
- The breakdown of sequence reads: rRNA, globin RNA, and other.
- SNP concordance among samples to detect potential sample swapping.
- Merged counts table from different isoform quantification algorithms.
- The number of expressed genes or transcripts at different TPM cutoffs.
- The top highly expressed genes and the percentage of genes from mitochondria.
- The correlation of expression profiles among all samples and potential outliers.
- MultiQC report for aggregation results across many samples.

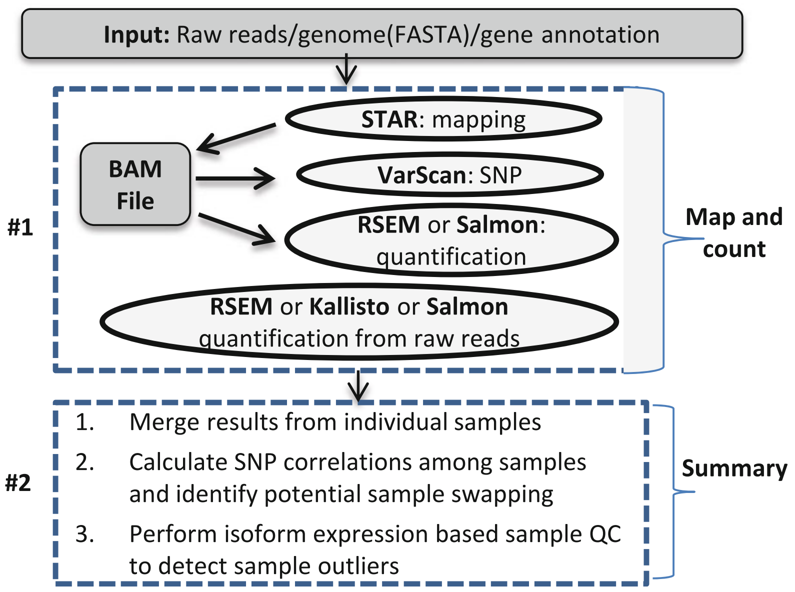

Input: Raw reads/genome(FASTA)/gene annotation
#1
BAM File
STAR: mapping
VarScan: SNP
RSEM or Salmon: quantification
RSEM or Kallisto or Salmon quantification from raw reads
Map and count
#2
1. Merge results from individual samples
2. Calculate SNP correlations among samples and identify potential sample swapping
3. Perform isoform expression based sample QC to detect sample outliers
Summary

Fig. 2 Overview of the *QuickIsoSeq* pipeline. Step #1 is computationally intensive, and processes individual samples independently. Step #2 integrates RNA-seq data analysis results from the individual samples in Step #1 and generates comprehensive QC metrices

2.1.3 The Architecture of QuickIsoSeq

QuickIsoSeq (Fig. 2) is implemented with the same design principles and user interface used in *QuickRNAseq* [8, 9]. The inputs to the pipeline are raw sequence reads and reference genome/transcriptome. **Step #1** performs RNA-seq read mapping, SNP calling, and isoform quantification. This step is computationally intensive and processes each sample independently. The independence of samples means they can be processed in parallel using an HPC. **Step #2** merges the analysis results from individual samples and generates QC metrics. All QC metrices are available in both tab delimited text files and in plots generated by the R library ggplot2 (https://ggplot2.tidyverse.org/).

2.2 Deployment of QuickIsoSeq

2.2.1 Installation of the Third-Party Open Source Tools

After a user downloads and unpacks the *QuickIsoSeq* package from GitHub, the environment variable **QuickIsoSeq** needs to be set to the root folder where the package is installed. All required third-party tools are listed in the Table 1 and can be installed by following the instructions from the individual tool's website. Alternatively, a user can run the **install-tools.sh** script which automates the installation of all the tools except for *MultiQC*. Full installation should take <10 min to complete. You can install the MultiQC python package by "`pip install multiqc`". Additionally, the *QuickIsoSeq* pipeline comes with many R scripts for data post-processing. In order to run them successfully, R version 3.2 or higher is required, and the *ggplot2* and *reshape2* R libraries should be installed.

Table 1
Required third-party tools

Tool	Function	Source
STAR	Read alignment/mapping	https://github.com/alexdobin/STAR
FeatureCounts	Count reads	http://subread.sourceforge.net/
VarScan	Variant calling	https://github.com/dkoboldt/varscan
Samtools	Manipulate read alignments	https://github.com/samtools/samtools
RSEM	Isoform quantification	https://github.com/deweylab/RSEM/
Salmon	Isoform quantification	https://github.com/COMBINE-lab/salmon
Kallisto	Isoform quantification	https://github.com/pachterlab/kallisto
Bowtie	Read alignment	https://sourceforge.net/projects/bowtie-bio/
gffread	Extract transcript sequences	http://ccb.jhu.edu/software/stringtie/dl/
FastQC	Raw fastq file QC	https://www.bioinformatics.babraham.ac.uk/projects
Multiqc	Aggregate results across many samples into a single report	https://multiqc.info/

By default, **install-tools.sh** will install all tools under **$Quick-IsoSeq/Tools** folder and generate a file **tools_path.txt** to record the locations of all the installed third-party packages. The contents in **tools_path.txt** are shown below and can be appended into a **run.config** file later.

```
APPLICATION_ROOT=${QuickIsoSeq}/Tools
STAR=$APPLICATION_ROOT/STAR_2.7.3a/bin/Linux_x86_64_static
FEATURECOUNTS =$ APPLICATION _ROOT/subread-2.0.0/bin
VARSCAN_JAR=$APPLICATION_ROOT/VarScan.v2.4.0.jar
SAMTOOLS=$APPLICATION_ROOT/samtools-1.9/bin
RSEM=$APPLICATION_ROOT/RSEM-1.3.1
SALMON=$APPLICATION_ROOT/salmon-1.1.0/bin
KALLISTO=$APPLICATION_ROOT/kallisto
GFFREAD=$APPLICATION_ROOT/gffread-0.11.4.Linux_x86_64
BOWTIE=$APPLICATION_ROOT/bowtie-1.2.3-linux-x86_64
FASTQC=$APPLICATION_ROOT/FastQC
export PATH=$STAR:$FEATURECOUNTS:......:$BOWTIE:$FASTQC:$PATH
```

2.2.2 Preparation of Index Files for Different Species

A reference genome in FASTA format and an isoform annotation file in GTF format are needed to create index files for isoform quantification. To simplify this step, an example script **create_indexes.GRCh38_Genecode30.sh** is provided, which creates all the required index files corresponding to human genome GRCh38 and GENCODE Release version 30. It takes about 1hr to run this script in an HPC cluster. You can change input files (i.e., the reference genome and gene annotation files) and create index files for other species and annotations as well. The main functions of this script are summarized as follows:

1. Download a genome file in fasta format, unzip and rename it to **genome.fa**.
2. Download a gene annotation file in GTF format, unzip and rename it to **gene.gtf**.
3. Extract sequences for all transcripts in **gene.gtf**, and generate **transcript.fa**.
4. Parse **gene.gtf** to get the corresponding annotations **gene.annot** and **transcript.annot**.
5. Create index files for *STAR*, *RSEM*, *Salmon*, and *Kallisto*, respectively.
6. Create *bowtie* indexes for rRNA and globin transcripts, respectively.
7. Define the genomic region **CHR_REGION** for SNP calls. In the human genome, chromosome 6 is recommended where the MHC (Major Histocompatibility Complex) region is located. For mouse, it is chromosome 17.

The file **${INDEX_ROOT}/ indexes_path.txt** records the locations of all index files created by **create_indexes.GRCh38_Genecode30.sh**. The contents of this file are shown below and can be cut-and-pasted into the **run.config file** later.

```
SAMPLE_SPECIES=human
INDEX_ROOT=${QuickIsoSeq}/Indexes/GRCh38_Genecode30
GENOME_FASTA=$INDEX_ROOT/genome.fa
GTF_FILE=$INDEX_ROOT/gene.gtf
TRANSCRIPT_FASTA=$INDEX_ROOT/transcript.fa
 TRANSCRIPT _ANNOTATION=$INDEX_ROOT/tran-
script.annot
 GENE_ANNOTATION=$INDEX_ROOT/gene.annot
 CHR_REGION=chr6:1-170805979
 STAR_INDEX=${INDEX_ROOT}/STAR_100
 RSEM_INDEX=${INDEX_ROOT}/rsem/rsem
 SALMON_INDEX=${INDEX_ROOT}/salmon
 KALLISTO_INDEX=${INDEX_ROOT}/kallisto/index
 rRNA_BWT_INDEX=${INDEX_ROOT}/bowtie_rRNA/rRNA
 hgRNA_BWT_INDEX=${INDEX_ROOT}/bowtie_hgRNA/hgRNA
```

2.3 Running QuickIsoSeq to Analyze RNA-Seq Datasets

After the above steps are completed, users can proceed to analyze their own RNA-seq datasets. To help a user to get started quickly, *QuickIsoSeq* comes with a **test_run** example project using the same 48 GTEx samples from five donors as in the original *QuickRNASeq* publication [9]. It is recommended to copy the four files in the **test_run** folder to a working folder and then tailor them accordingly.

1. **allIDs.txt**: sample identifiers to be processed.
2. **sample.annotation.txt**: an annotation file for RNA samples.
3. **run.config**: run configuration file.
4. **master-cmd.sh**: command lines to run *QuickIsoSeq*. These are provided for convenience so that a user can cut-and-paste and run the command lines, instead of typing them.

2.3.1 Preparing Sample Annotation, Sample ID File and run.config

Sample annotation is optional, but it is strongly recommended to prepare a meaningful annotation file to capture important meta data for RNA samples. A proper annotation file should be in tab delimited text format. The first and second columns correspond to sample and subject identifiers, respectively. The **sample.annotation.txt** file in the **test_run** directory has the columns "sample_id", "subject_id", "histological_type", and "sex". The second column is used to check potential sample swapping as all RNA samples from a same subject should have a very high SNP concordance, whereas the SNP concordance is lower for a pair of samples from different subjects. The file **allIDs.txt** contains one unique sample identifier per line. There is no column header. The **allIDs.txt** file in the **test_run** directory lists all 48 samples in this demo project. For an RNA-seq project, only those samples in the sample ID file are processed by *QuickIsoSeq*.

The **run.config** file controls the execution of *QuickIsoSeq* and decouples the dependency of the workflow on third-party open source tools and gene annotations. The **run.config** consists of three parts: **Part #1** contains RNA-seq project specific information; **Part #2** provides species-specific indexes, reference genome and gene annotation, generated previously; and **Part #3** sets the locations of tools and software specific parameters. For the same given species, **Parts #2** and **#3** in **run.config** usually remain the same across different RNA-seq projects, but **Part #1** varies from project to project, and those corresponding parameters must be set properly. The most important parameters in **Part #1** are:

- ***FASTQ_DIR***: the directory where the fastq files are located.
- ***FASTQ_SUFFIX***: a fastq file typically ends with fastq.gz, fq.gz, fastq, or fq.
- ***STRAND***: if non-stranded:0; if first read is forward strand:1; if first read is reverse strand:2.

- ***SEQUENCE_TYPE***: "pair" for pair-ended reads; "single" for single-ended sequencing
- ***SEQUENCE_DEPTH***: set it to "regular" if <80 million reads; otherwise set it to "deep".
- ***ISOFORM_ ALGORITHM*** : set it to *RSEM*, *KALLISTO*, *SALMON_ALN*, *SALMON*, or ALL.

2.3.2 Processing and Analyzing Individual Samples

Under a project root folder, invoke mapping, quantitation, QC, and SNP calling for each sample by running **run-isoseq.sh**. Because this step is computationally intensive, it is advised to run this command in an HPC cluster. The **run-isoseq.sh** can be run off the shelf in an HPC that uses platform load sharing facility (or simply LSF) as the job scheduler. Otherwise, for a cluster using a job scheduler other than LSF, **run-isoseq.sh** needs to be modified. A separate result folder will be created for each sample under the project folder. Below is the example command line.

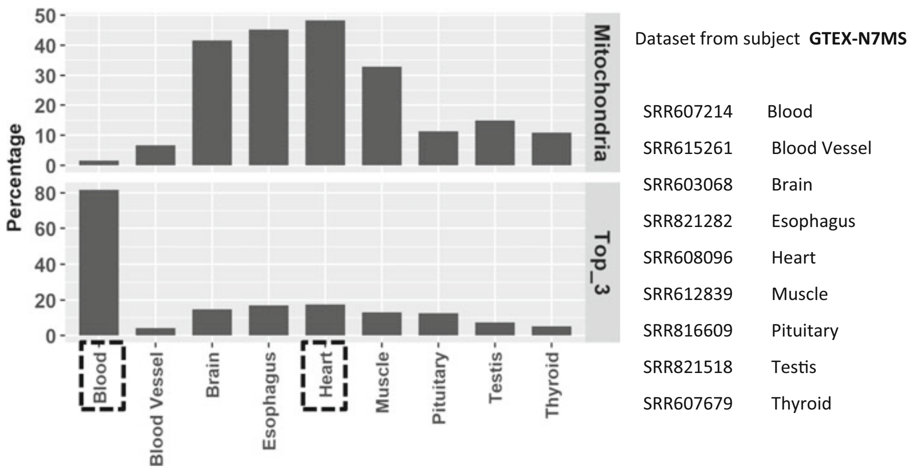
```
export QuickIsoSeq=<Your QuickIsoSeq Install Folder>
export PATH=$QuickIsoSeq:$PATH
run-isoseq.sh allIDs.txt run.config
```

2.3.3 Postprocessing: Merging Results from Individual Samples

As in the previous step, this step also runs under the project root directory. The post-processing step can only be run after all jobs processing individual samples have completed. This step not only merges results from individual samples and generates combined results such as a counts table but also generates a variety of QC metrics for stringent sample QC. All QC outputs are available in both plain tab delimited text files and as plots. Below are commands used to generate the summary results and QC metrices. The default output folder is **Results/Summary** unless a different and optional output folder name is provided as the third parameter in the command line when running **merge-isoseq.sh**.

```
export QuickIsoSeq=<Your QuickIsoSeq Install Folder>
//Make sure a R environment is available
merge-isoseq.sh allIDs.txt run.config &> Results.log
```

The QC metrices generated by *QuickIsoSeq* can help pinpoint sequencing issues in an RNA-seq project quickly, such as sample swapping, outliers, and the top highly expressed genes (Fig. 3). In Fig. 3, the percentage of transcripts from mitochondria and the three most abundant genes calculated from *Kallisto* are shown for nine tissue types from the same subject GTEX-N7MS in the Genotype-Tissue Expression (GTEx) project [22]. In heart, 48.3% of sequenced transcripts are from mitochondria, while in blood this percentage drops to as low as 1.5%. In heart, the three



Dataset from subject **GTEX-N7MS**

SRR607214	Blood
SRR615261	Blood Vessel
SRR603068	Brain
SRR821282	Esophagus
SRR608096	Heart
SRR612839	Muscle
SRR816609	Pituitary
SRR821518	Testis
SRR607679	Thyroid

Fig. 3 The percentages of transcripts from mitochondria, and the top three most abundant transcripts, in different tissue samples of the same subject (GTEX-N7MS) from the GTEx project. In heart, 48.3% of sequenced transcripts are from mitochondria, while in blood this percentage drops to as low as 1.5%. In blood, the top three highly expressed genes (HBA2, HBB, and HBA1) constitute as high as 81.8% of sequenced transcripts, indicating no globin clear is performed

highest expressed genes (MT-ATP6, MT-ATP8, and MT-CO3) represent a total of 17.4% of transcripts. In blood, the top three genes (HBA2, HBB, and HBA1) constitute as high as 81.8% of sequenced transcripts. Clearly, no globin reduction was performed for this blood sample. Considering the sequenced RNA repertoires differ so dramatically, direct comparison of TPM values of transcripts between blood and heart should be cautiously done.

3 Conclusions

By combing the best open source tool sets, we implemented the *QuickIsoSeq* pipeline that significantly reduces the effort involved in primary RNA-seq data analyses including QC. The workflow configuration file contains project, species, and software related parameters, and thus improves the reproducibly in RNA-seq data analyses. *RSEM*, *Salmon*, and *Kallisto* have different input requirements and output formats, and *QuickIsoSeq* takes care of wrapping between tools to perform analysis seamlessly for end users. The *QuickIsoSeq* pipeline has been already applied to multiple in-house large-scale RNA-seq projects, and its current version is stable and mature for public release and adoption.

Acknowledgments

The authors would like to thank Robert Stanton for his critical reading of the draft manuscript.

References

1. Mortazavi A et al (2008) Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nat Methods 5(7):621–628
2. Stark R, Grzelak M, Hadfield J (2019) RNA sequencing: the teenage years. Nat Rev Genet 20(11):631–656
3. Wang ET et al (2008) Alternative isoform regulation in human tissue transcriptomes. Nature 456:470–476
4. Harrow J et al (2012) GENCODE: the reference human genome annotation for the ENCODE project. Genome Res 22:1760–1774
5. Aoubala M et al (2011) p53 directly transactivates Delta133p53alpha, regulating cell fate outcome in response to DNA damage. Cell Death Differ 18:248–258
6. Kim S, An SS (2016) Role of p53 isoforms and aggregations in cancer. Medicine (Baltimore) 95:e3993
7. Mondal AM et al (2013) p53 isoforms regulate aging- and tumor-associated replicative senescence in T lymphocytes. J Clin Invest 123:5247–5257
8. He W et al (2018) QuickRNASeq: guide for pipeline implementation and for interactive results visualization. Methods Mol Biol 1751:57–70
9. Zhao S et al (2016) QuickRNASeq lifts large-scale RNA-seq data analyses to the next level of automation and interactive visualization. BMC Genomics 17:39
10. Liao Y, Smyth GK, Shi W (2014) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics 30:923–930
11. Zhang C et al (2018) Computational identification and validation of alternative splicing in ZSF1 rat RNA-seq data, a preclinical model for type 2 diabetic nephropathy. Sci Rep 8(1):7624
12. Zhao S, Xi L, Zhang B (2015) Union exon based approach for RNA-Seq gene quantification: to be or not to be? PLoS One 10(11): e0141910
13. Zhang C et al (2017) Evaluation and comparison of computational tools for RNA-seq isoform quantification. BMC Genomics 18 (1):583
14. Zhang C et al (2016) Bioinformatics tools for RNA-seq gene and Isoform quantification. Next Gen Sequence Appl 3:3
15. Li B, Dewey CN (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics 12:323
16. Roberts A, Pachter L (2013) Streaming fragment assignment for real-time analysis of sequencing experiments. Nat Methods 10:71–73
17. Nariai N et al (2014) TIGAR2: sensitive and accurate estimation of transcript isoform expression with longer RNA-Seq reads. BMC Genomics 15:S5
18. Trapnell C et al (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat Biotechnol 28:511–515
19. Patro R, Mount SM, Kingsford C (2014) Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. Nat Biotechnol 32:462–464
20. Patro R et al (2017) Salmon provides fast and bias-aware quantification of transcript expression. Nat Methods 14:417–419
21. Bray NL et al (2016) Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol 34:525–527
22. Carithers LJ, Moore HM (2015) The Genotype-Tissue Expression (GTEx) Project. Biopreserv Biobank 13(5):307–308

Chapter 9

Summarizing RNA-Seq Data or Differentially Expressed Genes Using Gene Set, Network, or Pathway Analysis

Enrica Calura and Paolo Martini

Abstract

The main purpose of pathway or gene set analysis methods is to provide mechanistic insight into the large amount of data produced in high-throughput studies. These tools were developed for gene expression analyses, but they have been rapidly adopted by other high-throughput techniques, becoming one of the foremost tools of omics research.

Currently, according to different biological questions and data, we can choose among a vast plethora of methods and databases. Here we use two published examples of RNAseq datasets to approach multiple analyses of gene sets, networks and pathways using freely available and frequently updated software. Finally, we conclude this chapter by presenting a survival pathway analysis of a multiomics dataset. During this overview of different methods, we focus on visualization, which is a fundamental but challenging step in this computational field.

Key words Pathway, Networks, Gene set analysis, Topological pathway analysis, Multiomics data integration

1 Introduction

In the last 20 years, there has been an explosion of genome wide data, especially in relation to the quantification of mRNA expression, occurring first with microarrays and then with RNAseq experiments. This generated an overwhelming quantity of expression data, with a single experiment able to detect expression of tens of thousands of genes. The complexity of these results prompted the development of new analysis systems to help in summarizing these data and in extracting biologically meaningful information.

Currently, gene set or pathway analysis is the most widely used tool to investigate genes by groups, when comparing two conditions in an experiment. However, with the advent of single sample gene sets, as well as time-course and survival gene set analysis, researchers now have more choices available.

Ernesto Picardi (ed.), *RNA Bioinformatics*, Methods in Molecular Biology, vol. 2284, https://doi.org/10.1007/978-1-0716-1307-8_9,
© Springer Science+Business Media, LLC, part of Springer Nature 2021

A gene set can be any group of genes that are related in some way. For example, the Gene Ontology (GO) provides groups of genes describing cellular components, biological processes, and molecular functions [1, 2]. Another database, MSigDB, groups genes by molecular signatures inferred from perturbation experiments, allowing the researcher to have sets of genes that are coexpressed in response to a specific stimulus [3].

Gene sets can also be derived from pathways, which are detailed maps of cell processes. Pathway data contain interactions and relationships among the involved genes, and by exploiting this detailed information (known as pathway topology), we can have topological pathway analyses.

In this chapter, we describe a general workflow to study the biological processes that emerge from the downstream analysis of RNAseq gene expression data. Additionally, the last section will be dedicated to describing a new approach for dealing with multiomics data integration using pathways and their topology.

This chapter is not meant to cover all available knowledge on pathway analysis but rather to give some hints and practical advice on how to explore expression data.

2 Materials

2.1 Gene Sets and Pathways

A variety of databases containing information about biologically related gene sets and cell signaling pathways have been developed over the past years.

Gene sets are groups of genes that are biologically related, that is, genes with a common function or localization, or genes coexpressed in a specific disease, condition, or treatment. The most common source of gene sets is the Gene Ontology (GO; http://geneontology.org/) [2]. The GO knowledgebase is the world's largest source of information on the functions of genes. The current release (2020–01–01) has 44,700 GO terms (gene sets) containing 1,351,824 gene products in 4591 species. Another widely used database of gene sets is the Molecular Signatures Database (MSigDB) which hosts seven collections of related gene sets: hallmark, positional, curated, motif, computational, GO, oncogenic, and immunologic [3]. These gene sets, originally available for the Homo sapiens, have also been translated for mouse, rat, pig, fly, and yeast [4].

Pathway databases serve as repositories of manually curated maps representing current knowledge regarding cell processes, signaling, and metabolism. To help navigate this information, the Pathguide resource serves as a good overview of current pathway databases [5]. It lists more than 700 pathway repositories, over 250 of which are dedicated to *Homo sapiens*. Among these are KEGG [6] and Reactome [7], two of the most famous and widely used pathway repositories. While all these initiatives are successful results of the joint effort of a wider community, different databases are characterized by different annotations and only a part of the interactions are confirmed by all the repositories. Unlike GO, which can simply be retrieved in an R environment within gene annotations, pathway data require more attention. Independently of the specific pathway databases used, all pathway maps are comprised of a myriad of different interaction types across multiple different biological entities. In fact, these maps are often too rich to be represented as networks (sets of one-to-one connections across genes). In particular, challenges have been posed by the presence of chemical compounds that may be mediating interactions (nodes that usually are not quantified in genomic experiments) and by different types of gene groups (e.g., protein complexes or gene families) that are usually represented as single nodes in pathway maps even though they represent multiple genes.

Thus, in this tutorial we have exploited the R Bioconductor package *graphite*, which was developed to solve these issues regarding the translation of pathway maps into networks. *graphite* acts as a pathway provider in an R environment, providing a bridge between the different pathway data and pathway analyses, thus offering useful functions for pathway and network data management [8].

2.2 Two RNA-Seq Datasets

The tutorial described herein has been built around two datasets. However, the code can obviously be accommodated for your dataset of interest.

The first dataset, hereafter called IRF6KO, is related to the paper "The RIPK4-IRF6 signalling axis safeguards epidermal differentiation and barrier function" [9]. In the paper the authors analyzed the differential expression of mouse skin specimens: three samples from IRF6 knockout mice (IRF6KO) and three from wild-type mice (WT). They reported that pathways of lipid metabolism are perturbed in the KO mouse model.

The second dataset is an ovarian cancer dataset, provided by The Cancer Genome Atlas Consortium, hereafter called the OV-TCGA. We have compared two subtypes of ovarian cancer tumors, one called "differentiated," which is characterized by a gene expression profile commonly seen in well differentiated cancers with low malignant potential, and another called "proliferative," which is characterized by the expression of genes associated with cell division [10].

2.3 Data Retrieval

The "IRF6KO" dataset can be found in the GEO database under the accession code GSE124048. Raw counts can be downloaded at "ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE124nnn/GSE124048/suppl/GSE124048_RAW.tar". Final count data matrix was obtained by combining the six samples. The list of differentially expressed genes (DEGs) was obtained using *edgeR*, comparing WT (WT mouse skin samples) *versus* IRF6KO (Irf6 knockout mouse skin samples).

The OV-TCGA dataset is freely available (level 2 or level 3 data) from the GDC Data portal "https://portal.gdc.cancer.gov/". At "https://cavei.github.io/example-datasets/" you can find the *R Data* with annotations and data matrices for expression, methylation, mutations, and copy number variations for the multiomics OV-TCGA dataset. The list of DEGs was obtained by comparing proliferative versus differentiated samples using *edgeR*.

The procedure starts by assuming that all these steps have already been performed and all these data are already computed. The reproducible commands on how to preprocess the two datasets can be found at "https://cavei.github.io/springer_prep/".

2.4 Software

Data processing and analyses can be entirely carried out using specific packages with the R programming language, which is the preferred environment for statistical computing and graphics ("https://www.r-project.org/"). Pathway and gene set databases can also be accessed through R packages. Table 1 lists all the annotation and R packages used in this chapter.

More elaborated visualization can be achieved also using a web-based application ("https://reactome.org/") or the *Cytoscape* software ("https://cytoscape.org/").

3 Methods

3.1 Outline of Methods

Generally, pathway analysis can be carried out as follows:

1. Identification of the preferred gene set or pathway source.
2. Identification of the appropriate test for gene set or pathway analysis.
3. Visualization of the results.

Commonly, the latter is the most underestimated step, but it is of fundamental importance in order to understand the results and the biological processes under study.

In the following section, we analyze an RNA-seq dataset with two conditions and few replicates per condition (less than 10 per condition). The number of replicates is an important parameter that must be taken carefully into account when choosing the appropriate test for gene set/pathway analysis. Here, when analyzing the IRF6KO dataset (3 KO vs. 3 WT), the pathways were treated as mere gene sets; thus, we performed an enrichment analysis starting from the list of Differentially Expressed Genes (DEGs). Finally, the results were explored graphically.

Following this, we used the second dataset, OV-TGCA, which has a higher number of replicates, to further elaborate the analyses.

Table 1
R packages used in the chapter

Package name	Description	References
aracne.networks	ARACNe-inferred gene networks from TCGA tumor datasets (ver. 1.10.0)	[11]
clipper	Gene Set Analysis Exploiting Pathway Topology (ver. 1.24.0)	[12]
clusterProfiler	statistical analysis and visualization of functional profiles for genes and gene clusters (ver. 3.12.0)	[13]
edgeR	Empirical Analysis of Digital Gene Expression Data in R (ver. 3.26.8)	[14]
graphite	GRAPH Interaction from pathway Topological Environment (ver. 1.30.0)	[8]
ggplot2	Create Elegant Data Visualisations Using the Grammar of Graphics (ver. 3.2.1)	[15]
GSVA	Gene Set Variation Analysis for microarray and RNA-seq data (ver. 1.32.0)	[16]
limma	Linear Models for Microarray Data (ver. 3.40.6)	[17]
msigdbr	MSigDB Gene Sets for Multiple Organisms in a Tidy Data Format (ver. 7.0.1)	[4]
MOSClip	Multi Omics Survival Clip (ver. 0.4.1)	[18]
org.Hs.eg.db	Genome wide annotation for Human (ver. 3.8.2)	[19]
org.Mm.eg.db	Genome wide annotation for Mouse (ver. 3.8.2)	[20]
pathview	a tool set for pathway based data integration and visualization (ver. 1.24.0)	[21]
viper	Virtual Inference of Protein-activity by Enriched Regulon analysis (ver. 1.18.1)	[22]

Using this dataset, we (1) replicated the enrichment analysis, (2) applied single sample analysis, which is also usable for small size datasets, (3) inferred protein activity, and (4) performed topological pathway analysis. At the end of this section, we (5) show an example of multiomics data integration using survival pathway analysis.

We would like to stress that these different tools are not mutually exclusive. Whenever it is possible and biological meaningful, these tools can be combined to strengthen the data interpretation.

3.2 Analyses of IRF6 Knockout Dataset

The dataset IRF6KO was processed as follows: (1) per sample raw count data was collapsed in a matrix composed of 25,065 rows (genes) and 6 columns (samples), (2) below-threshold expressed genes were filtered out, and only genes with 0.5 CPM in at least two samples (15,221 genes) were kept, (3) sample annotations of the two groups (WT and IRF6KO) were retrieved, and (4) differentially expressed genes (DEGs) were computed with *edgeR* (WT vs. IRF6KO).

We defined DEGs as those genes with a corrected *p*-value < =0.05 and an absolute log fold change > =1. With these settings we have 1040 upregulated genes (with more expressed in the WT samples) and 476 downregulated genes (with more expressed in IRF6KO samples).

3.2.1 Gene Set Enrichment Pathway Analysis

Firstly, we performed the simplest yet most used pathway analysis approach: pathway enrichment through a hypergeometric test. This method tests whether the pathway of interest contains more DEGs (than expected by chance) compared to those outside the pathway, that is, the hypergeometric test tests whether the pathway is enriched/overrepresented within the DEGs. This test is included in many pathway analysis tools, and here we use the function enricher from the R package *clusterProfiler* [13].

To perform enrichment analysis we needed gene sets: for this we used the Bioconductor R package *graphite* as the pathway provider, retrieving the mouse pathways from the KEGG database [6].

```
library(graphite)
mmu_kegg_sym <- pathways("mmusculus", "kegg")
mmu_kegg_sym <- convertIdentifiers(mmu_kegg_sym, "SYMBOL")
```

In the package *graphite*, each pathway is a "*Pathway*" class object. Using the *nodes* method, we can retrieve genes from each pathway and store them as *data.frame* (using the *ldply* function from *plyr* library). The obtained *data.frame* is two columns: pathway name and gene symbol. We renamed the columns to "*gs_name*" and "*gene_symbol*". Lastly, we removed the suffix "SYMBOL:" and transformed gene set names from factor to character. In this way, the *graphite* pathways were transformed into *TERM2GENE* objects required by the *clusterProfiler* package.

```
library(plyr)
kegg_t2g <- ldply(lapply(mmu_kegg_sym, nodes), data.frame)
names(kegg_t2g) <- c("gs_name", "gene_symbol")
kegg_t2g$gene_symbol <- gsub("SYMBOL:", "", as.character(kegg_t2g$gene_symbol))
kegg_t2g$gs_name <- as.character(kegg_t2g$gs_name)
```

Now, with our list of DEGs and the *TERM2GENE* object, it is possible to run the *enricher* function as follows:

```
library(clusterProfiler)
eKEGG <- enricher(gene = IRF6_dataset$up_down_regulated$up, TERM2GENE = kegg_t2g,
                  universe = row.names(IRF6_dataset$lnorm_data),
                  pAdjustMethod = "BH",
                  pvalueCutoff = 0.1, qvalueCutoff = 0.05,
                  minGSSize = 10, maxGSSize = 500)
```

In the above command, the *gene* argument was the gene list of DEGs (in our case only the upregulated subset), *TERM2GENE* was the KEGG pathways for mouse, and the *universe* was the set of the genes for which we have expression data. Additionally, we set the *p*-value correction method as Benjamini & Hochberg (*pAdjust-Method* = *"BH"*), *p*-value and *q*-value cutoff to 0.1 and 0.05 respectively, and minimal and maximal size of the tested gene set equal to 10 and 500 genes, respectively.

The command produces an *enrichResult* object that can be transformed into a *data.frame* and saved into a file. We chose to save the file in csv format, which can also be easily imported into any chosen favorite spreadsheet program. The file contains the enriched pathways by row, and nine columns including *"ID"*, *"Description"*, *"GeneRatio"*, *"BgRatio"*, *"p-value"*, *"padjust"*, *"qvalue"*, *"geneID"*, and *"Count"*. See the *clusterProfiler* manual for more details.

```
UP_eKEGG_df <- as.data.frame(eKEGG)
write.table(UP_eKEGG_df, file = "enrichResult_upreg_genes_wt_vs_irf6KO.csv",
            sep = ",", row.names = F)
```

The *enrichResult* object can also be plotted in many ways. As an example, we created a bar plot with the 15 enriched pathways as shown in Fig. 1a: the *y*-axis shows the pathway, while the *x*-axis reports the number of DEGs in the pathway, and the color represents the adjusted *p*-value.

```
barplot(eKEGG, showCategory = 15)
```

We can also calculate the enrichment for downregulated DEGs. Since downregulated genes in our data set were fewer than upregulated genes, for visualization purposes, we have raised the thresholds for both the *pvalueCutoff* and *qvalueCutoff* parameters.

```
eKEGG_down <- enricher(IRF6_dataset$up_down_regulated$down, TERM2GENE = kegg_t2g,
                       universe = row.names(IRF6_dataset$lnorm_data),
                       pAdjustMethod = "BH",
                       pvalueCutoff = 0.2, qvalueCutoff = 0.2,
                       minGSSize = 10, maxGSSize = 500)
```

To visualize downregulated genes, we used the *cnetplot* function, which represents the network of enriched pathways connected with their genes (Fig. 1b); this visualization allows the user to rapidly understand the genes that are shared by multiple pathways.

```
cnetplot(eKEGG_down, colorEdge = T)
```

As you can *see* in Fig. 1b, a simple enrichment approach can recapitulate all of the results from the original paper [9].

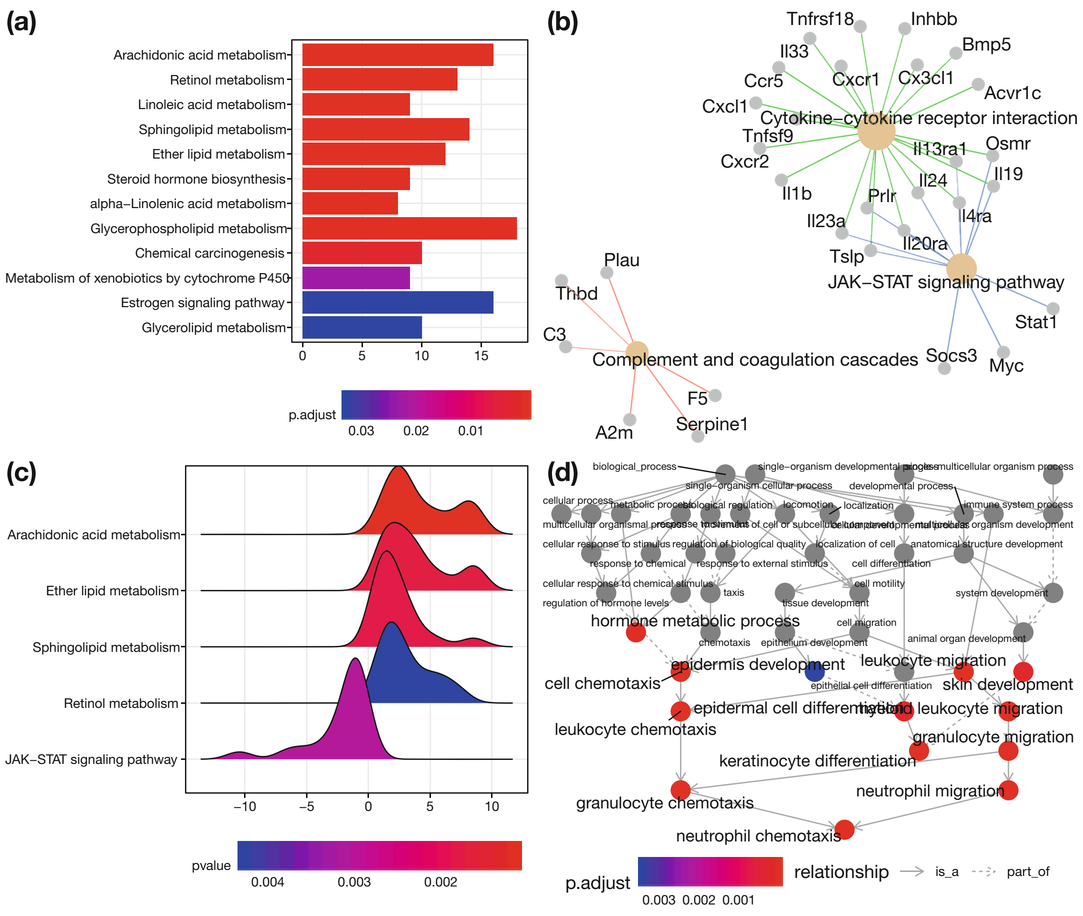


Fig. 1 Enrichment plots related to dataset IRF6KO. (**a**) Upregulated DEGs found in each enriched KEGG pathway set are colored by corrected *p*-value. (**b**) Downregulated enriched pathway sets with their downregulated DEGs. (**c**) Ridge plot of top five enriched categories identified with GSEA and colored by *p*-value. (**d**) GO BP hierarchy graph induced from downregulated enriched GO BP gene sets

3.2.2 GSEA Method

GSEA is a method that does not rely on strict cutoff thresholds defining the DEGs but rather on all the genes being ranked by their fold change [3]. To perform this analysis, we have used the *GSEA* function from *clusterProfiler*. With the following commands, we prepared the data (genes were ranked by their fold change).

```
table <- IRF6_dataset$DE_tables
gene_ranks <- table[order(table$logFC, decreasing = T), "logFC", drop = F]
gene_ranked <- gene_ranks$logFC
names(gene_ranked) <- row.names(gene_ranks)
```

Then, we ran the GSEA.

```
set.seed(1)
kegg_gsea <- GSEA(gene_ranked, TERM2GENE = kegg_t2g,
                  nPerm = 10000, minGSSize = 10, maxGSSize = 500,
                  pvalueCutoff = 1, pAdjustMethod = "BH",
                  seed = T)
```

Following this, we can produce a *ridge plot* which shows the distribution of the gene log fold change in the top five analyzed pathways with their *p*-value.

```
ridgeplot(kegg_gsea, showCategory = 5, fill = "pvalue")
```

GSEA analysis also puts the focus on lipid metabolism (Fig. 1c).

3.2.3 Gene Ontology Enrichment Analysis

As previously stated, another popular source of gene sets is the gene ontology (GO; http://geneontology.org/) [2]. In our dataset, we have used the mouse GO terms through the *enrichGO* function (from *clusterProfiler*) which automatically extracts GO information from Bioconductor packages (*org.Mm.eg.db*).

When working with organisms where the GO has not been packed into an *AnnotationDBI* package, it is still possible to download the GO from any source, pack it into a *TERM2GENE* object and use the *enricher* function, as described previously. We have provided an example of this in Subheading 4 (*see* **Note 1**).

Using the *enrichGO* function, we need three additional arguments: (1) the *AnnotationDBI* package of the desired species (in our case "*org.Mm.eg.db*"), (2) the type of identifier used for the genes (in our case "*SYMBOL*"), and (3) the GO ontology, which can be one of the following: Biological Processes (BP), Molecular Function (MF), or Cellular Components (CC). In the following example, we have used BP ontology. To test all of them at the one time, the user just needs to set *ont="ALL"*.

```
eGOBP <- enrichGO(IRF6_dataset$up_down_regulated$up,
                  OrgDb = "org.Mm.eg.db", keyType = "SYMBOL", ont = "BP",
                  pvalueCutoff = 0.1, qvalueCutoff = 0.05,
                  minGSSize = 10, maxGSSize = 500)
```

With the following commands, we can produce the *barplot* and the *emapplot* of the upregulated GO categories (plot not shown). Enrichment map plots (*emaplot*) are useful because they allow for the organization of enriched sets into networks where edges reflect the overlap between gene sets, so that overlapping gene sets tend to cluster together.

```
barplot(eGOBP, showCategory = 20)
emapplot(eGOBP, showCategory = 20)
```

To manually check the results, we can create the *data.frame* from the enriched GO BP (*p*-value corrected < =0.05). In Table 2

Table 2
Top ten enriched terms and their corrected *p*-values

Description	*p*.adjust
Skin development	5.075*e*–27
Keratinocyte differentiation	7.835*e*–24
Epidermal cell differentiation	5.435*e*–20
Epidermis development	5.029*e*–19
Peptide cross-linking	6.382*e*–16
Multicellular organismal water homeostasis	1.236*e*–13
Regulation of water loss via skin	2.748*e*–13
Water homeostasis	9.508*e*–13
Establishment of skin barrier	4.205*e*–12
Sphingolipid metabolic process	1.329*e*–08

below, we have reported the top ten enriched terms and their corrected *p*-values.

```
eGOBP_df <- as.data.frame(eGOBP)
sig_eGOBP_df <- eGOBP_df[eGOBP_df$p.adjust <= 0.05, ]
head(sig_eGOBP_df[,c("Description", "p.adjust")], 10)
```

Looking at the enriched GO BP from the upregulated genes we saw that the IRF6KO samples were missing lipid related processes, which were impacting skin barrier functionality. Indeed, this process is lost in the IRF6KO mice, as reported by the authors [9].

We ran the same analysis for downregulated genes, and we plotted the results. Since we are using GO, aside from the already presented *barplot*, *cnetplot*, and *emapplot*, we can also use another plot function from *clusterProfiler* specifically devised for GO called *goplot*. This produces the enriched GO induced graph.

```
eGOBP_down <- enrichGO(IRF6_dataset$up_down_regulated$down,
                       OrgDb = "org.Mm.eg.db", keyType = "SYMBOL", ont = "BP",
                       pvalueCutoff = 0.1, qvalueCutoff = 0.05,
                       minGSSize = 10, maxGSSize = 500)

goplot(eGOBP_down)
```

As shown by the *goplot* in Fig. 1d, the enriched GO BP showed immune system impairment in the IRF6KO samples and confirmed that IRF6KO impacts “lipid metabolism”, “epidermis development”, and the “establishment of skin barrier”.

3.2.4 KEGG Pathway Map Representation

Using the *pathview* R package [21], we can map DEGs on specific KEGG pathways. It requires NCBI entrez gene IDs as the input, thus we had to convert all the gene symbols into entrez gene ids and then call *pathview*, specifying the species ("mmu") and the pathway ("mmu04530", i.e., the "Tight junctions" pathway).

```
library("pathview")
id_table_conversion <- AnnotationDbi::select(org.Mm.eg.db,
                                   keys = IRF6_dataset$up_down_regulated$up,
                                   columns = c("SYMBOL", "ENTREZID"),
                                   keytype = "SYMBOL")

mmu04530 <- pathview(gene.data = id_table_conversion$ENTREZID,
                     pathway.id = "mmu04530", species = "mmu")
```

The function outputs a "png" image with the KEGG "Tight junctions pathway" where the DEGs are mapped in red (Fig. 2). This function is extremely useful to see the position of the DEGs in a pathway map.

3.3 Analyses of the Ovarian Cancer (OV-TCGA) Dataset

Preprocessing of the OV-TCGA dataset followed the procedure described in [18]. Briefly, we downloaded the sample annotations and we selected those patients with follow-up information (progression-free survival) and TCGA subtypes annotation as reported in [23]. In total, we have at our disposal 259 patients divided in four TCGA subtypes: 73 differentiated, 61 immunoreactive, 63 mesenchymal, and 62 proliferative. For all patients, we downloaded expression, methylation, mutation, and CNV profiles. Comparing proliferative *vs* differentiated subtypes, we obtained the list of DEGs to be used for our tutorial (corrected p-value $< =0.05$ and an absolute log fold change $> =1$). With these settings we have 739 upregulated genes (more highly expressed in the proliferative patients) and 549 downregulated genes (more highly expressed in the differentiated patients).

3.3.1 Gene Set Enrichment Pathway Analyses

The following section describes the study of DEG enrichment in KEGG and REACTOME pathways. As above, we retrieved the pathways though the *graphite* package. In the TCGA dataset, the genes are annotated as *ENTREZID*, so we needed to retrieve the Homo sapiens pathways and convert identifiers into *ENTREZID*. The following code describes how to retrieve the KEGG data. To perform the same task for Reactome it will be necessary to substitute "kegg" with "reactome" in the code.

```
library(graphite)
hsapiens_kegg_eg <- pathways("hsapiens", "kegg")
hsapiens_kegg_eg <- convertIdentifiers(hsapiens_kegg_eg, "ENTREZID")
```

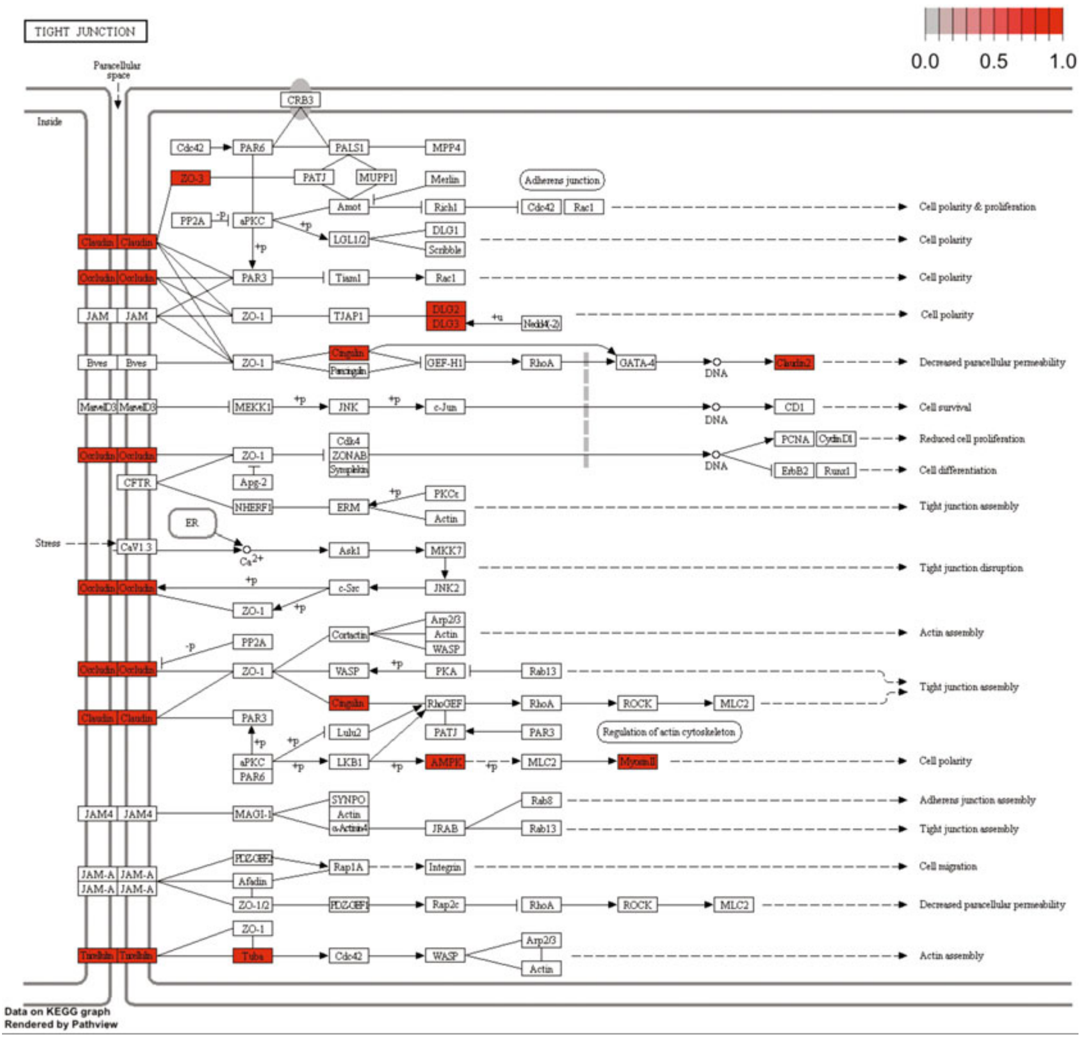


Fig. 2 Upregulated DEGs mapped on the "Tight Junction" pathway through the *pathview* R package

Following this, we converted the *pathway* list of *graphite* into a *TERM2GENE* object to be used by the *clusterProfiler enricher* function (see the IRF6KO analysis for a detailed explanation).

```
library(plyr)

gset_hsapiens_kegg_eg <- ldply(lapply(hsapiens_kegg_eg, nodes), data.frame)
names(gset_hsapiens_kegg_eg) <- c("gs_name", "entrez")

gset_hsapiens_kegg_eg$entrez <- gsub("ENTREZID:", "",
                              as.character(gset_hsapiens_kegg_eg$entrez))
gset_hsapiens_kegg_eg$gs_name <- as.character(gset_hsapiens_kegg_eg$gs_name)
```

At this point, rather than calculating upregulated and down-regulated gene enrichment in separate plots as shown for the

IRK6KO dataset, we instead compared up and downregulated genes in the same plot. With this goal we introduce the *compareCluster* function (from *clusterProfiler*). This function allows the user to specify an arbitrary number of gene lists through a named list. This allows for the creation of an *enrichment* object that takes into account all the gene lists and directly compares different enriched pathways in a plot. In the following code, we created a list with a first element called "Prolif" that contained the upregulated genes and a second element called "Dif" that contained the downregulated genes.

```
comparison_list <- list(Prolif = TCGA_OV_deg$up_down_regulated$up,
                        Dif = TCGA_OV_deg$up_down_regulated$down)
```

Following this we call the function *compareCluster*. In this case, *enricher* becomes an argument of the function *compareCluster*, thus we have used the function *enricher* with the listed parameters in each element of the *comparison_list*.

```
compareKEGG <- compareCluster(comparison_list,
                              fun = "enricher",
                              TERM2GENE = gset_hsapiens_kegg_eg,
                              universe = row.names(TCGA_OV_deg$lnorm_data),
                              pAdjustMethod = "BH",
                              pvalueCutoff = 0.1, qvalueCutoff = 0.1,
                              minGSSize = 10, maxGSSize = 500)
```

To visualize the results from the *compareCluster* function, we have used the *dotplot* function which represents pathways as dots; the size of each dot follows the gene ratio (DEGs/total number of genes in the set), and the color represents the *p*-value. With the following command we asked for a representation of the top ten pathways.

```
dotplot(compareKEGG, showCategory=10)
```

As you can *see* in Fig. 3a, the *dotplot* provides a clear overview of the pathways that are specific or in common with the two conditions. In this example, we can appreciate that differentiated samples showed pathways related to inflammation, while in proliferative we have other pathways such as "ECM-receptor interaction" and "Wnt and TGF-beta signaling".

The following section shows what DEGs look like using another source of pathways, that is, the Reactome database [7] from the *graphite* R package. The procedure to retrieve the Reactome database is the same as was used for KEGG; however, it is necessary to substitute "kegg" with "reactome" in the code.

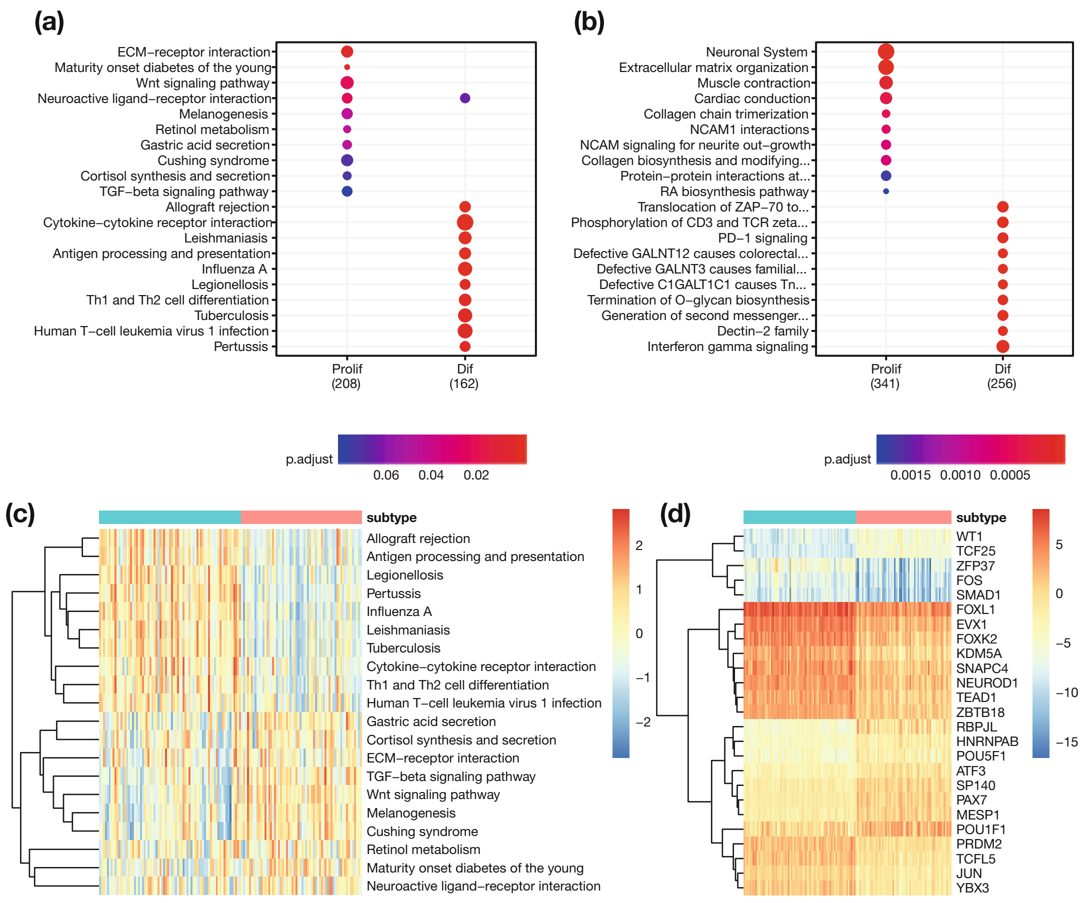


Fig. 3 Plots from gene set/network analyses for the OV-TCGA dataset. (**a**, **b**) Comparisons of the top ten upregulated (Prolif) and top ten downregulated (Dif) enriched KEGG and Reactome pathways, respectively. (**c**) Single sample gene set analysis: KEGG enriched pathways in Fig. 2c are summarized for each patient with Gene Set Variation Analysis (GSVA). (**d**) Transcription factors with significant differences in protein activity between proliferative and differentiated groups (*p*-value $< =$0.01, absolute mean differencies $> =$2)

Firstly, we created the *TERM2GENE* object:

```
gset_hsapiens_reactome_eg <- ldply(lapply(hsapiens_reactome_eg, nodes),
data.frame)
names(gset_hsapiens_reactome_eg) <- c("gs_name", "entrez")

gset_hsapiens_reactome_eg$entrez <- gsub("ENTREZID:", "",
                              as.character(gset_hsapiens_reactome_eg$entrez))
gset_hsapiens_reactome_eg$gs_name <- as.character(
                                       gset_hsapiens_reactome_eg$gs_name)
```

Then, we used the *compareCluster* function on the *comparison_list* object.

```
compareReactome <- compareCluster(comparison_list,
                         fun = "enricher",
                         TERM2GENE = gset_hsapiens_reactome_eg,
                         universe = row.names(TCGA_OV_deg$lnorm_data),
                         pAdjustMethod = "BH",
                         pvalueCutoff = 0.1, qvalueCutoff = 0.1,
                         minGSSize = 10, maxGSSize = 500)
```

Following this we called the *dotplot* function and compared the top ten proliferative *vs* differentiated enriched gene sets for Reactome.

```
dotplot(compareReactome, showCategory=10)
```

The Reactome database analysis confirmed that the proliferative group still has "Extracellular matrix organization" as the most highly expressed pathway, while the differentiated group has immuno-related processes in the top position (Fig. 3b). Nevertheless, Reactome can add more detail regarding the different processes extending what can be achieved with KEGG. For example, from differentiated samples it was seen that the "PD-1 signaling" pathway has more DEGs (downregulated in proliferative patients) than what would have been expected by chance. This pathway can be particularly interesting, because the PD-1/PD-L1 axis can be impacted by immune checkpoint inhibitor-based cancer therapies [24].

3.3.2 Single Sample Analysis

There are some scenarios where it could be useful to summarize gene expression in pathway deregulation sample by sample, that is, to provide single patient classification.

In the following section, we explored single sample gene set analysis (ssGSA) using the *GSVA* R package. To run the commands it was necessary to convert a gene set *data.frame* (aka *TERM2-GENE* object) into a named list, where each pathway/gene set name is associated with the list of its genes. To do so we started from the *TERM2GENE* object built in the previous analysis:

```
gset_list_hsapiens_kegg_eg <- lapply(tapply(gset_hsapiens_kegg_eg$entrez,
                                            gset_hsapiens_kegg_eg$gs_name,
                                            identity),
                                     identity)
```

To call *gsva*, we needed to provide a normalized expression matrix (in our case *lnorm_data*), the list of gene sets (the gene set list from the KEGG database), and the method for the summary (either "gsva", "ssgsea", "zscore", or "plage"). Since our expression data are log CPM, we used the gaussian kernel to estimate the cumulative distribution function of expression levels across samples. We limited the analysis to gene sets larger than 10 genes but smaller than 500 genes.

```
library(GSVA)
gsva_summary <- gsva(TCGA_OV_deg$lnorm_data, gset_list_hsapiens_kegg_eg,
                     method = "gsva", kcdf = "Gaussian",
                     min.sz = 10, max.sz = 500)
```

The *gsva* function returns a matrix-like object, with pathways in rows and patients in columns. With the following commands, we extracted both differentiated and proliferative samples from the matrix.

```
select_columns <- multiOmicDataset$patients_subtype$subtype=="OVCA.Differentiated"
                 | multiOmicDataset$patients_subtype$subtype=="OVCA.Proliferative"
```

We then created a gene set expression matrix by selecting the desired patients, and we produced heatmaps of the GSVA analysis. We focused on the pathways shown in the KEGG comparison dotplot (Fig. 2a; the list of pathways was saved into *interesting_-kegg_pathways*)

```
annotation_columns <- multiOmicDataset$patients_subtype[select_columns, , drop=F]
annotation_columns <- annotation_columns[order(annotation_columns$subtype), ,
                                         drop=F]
gsva_interesting_kegg <- gsva_summary[int eresting_kegg_pathways,
                                      row.names(annotation_columns),
                                      drop=F]

pheatmap(gsva_interesting_kegg,
         annotation_col = annotation_columns,
         cluster_cols = F, scale="row", show_colnames = F)
```

As can be seen in Fig. 3c, we can reconfirm what we saw in the enrichment analysis for KEGG. We can appreciate that the "ECM-receptor interaction" pathway is more activated in proliferative patients and that the "Cytokine-cytokine receptor interaction" pathway is more active in differentiated samples. It is also worth mentioning that the GSVA output matrix can be used to perform other statistical tests of pathway differential expression analysis or cluster analysis that has not been covered in this protocol.

3.3.3 Gene Set Enrichment Analysis with MSigDB

In the following section we attempt to summarize the DEGs from our dataset using molecular signatures like those hosted in MSigDB [25, 26] (https://www.gsea-msigdb.org/gsea/msigdb/collections.jsp). In the MSigDB we found different types of gene sets, including Hallmark genes, which are genes that display coordinate expression as a hallmark of a specific process, as well as

Transcription Factor Targets (TFT). These data were collected from the R package *msigdbr*. In the following piece of code, we created the *HALLMARK* and *TFT* sets for Homo sapiens from the *msigdbr* R package and converted the *TERM2GENE* object into a named list, in which each element is a gene set containing its genes.

```
library(msigdbr)

hallmark_t2g <- msigdbr(species = "Homo sapiens", category = "H")
                %>% select(gs_name, entrez_gene)

hallmark_t2g$gs_name <- gsub("HALLMARK_", "", hallmark_t2g$gs_name)
hallmark_t2g$entrez_gene <- as.character(hallmark_t2g$entrez_gene)

tft_t2g <- msigdbr(species = "Homo sapiens", category = "C3",
                   subcategory = " TFT:TFT_Legacy")
           %>% select(gs_name, entrez_gene)
tft_t2g$entrez_gene <- as.character(tft_t2g$entrez_gene)

hallmark_gene_set_list <- tapply(hallmark_t2g$entrez_gene,
                                 hallmark_t2g$gs_name, identity, simplify = F)
hallmark_gene_set_list <- lapply(hallmark_gene_set_list, identity)

tft_gene_set_list <- tapply(tft_t2g$entrez_gene, tft_t2g$gs_name, identity,
simplify = F)
tft_gene_set_list <- lapply(tft_gene_set_list, identity)
```

Following this, we can run *compareCluster* with the *TFT* gene set list.

```
compareTFT <- compareCluster(comparison_list, fun = "enricher",
                             TERM2GENE = tft_t2g,
                             universe = row.names(TCGA_OV_deg$lnorm_data),
                             pAdjustMethod = "BH",
                             pvalueCutoff = 0.1, qvalueCutoff = 0.1,
                             minGSSize = 10, maxGSSize = 500)
```

We created a tabular output to inspect the results and to extract only the TFT gene sets that have a corrected p-value $< =0.1$. With the *table* function we counted how many TFT gene sets are in the proliferative group and how many are in the differentiated group.

```
tft_df <- as.data.frame(compareTFT)
sig_tft_df <- tft_df[tft_df$p.adjust <= 0.1, ]
table(sig_tft_df$Cluster)
```

Subtype	#sig TFT gene set
Prolif	172
Dif	0

As can be seen from the result summary, no TFT gene sets were detected in the differentiated samples. We then asked ourselves what would happen if we redid the same analysis using hallmarks.

```
compareHallmark <- compareCluster(comparison_list, fun = "enricher",
                              TERM2GENE = hallmark_t2g
                              universe = row.names(TCGA_OV_deg$lnorm_data),
                              pAdjustMethod = "BH",
                              pvalueCutoff = 0.1, qvalueCutoff = 0.1,
                              minGSSize = 10, maxGSSize = 500)
```

We created a tabular output to inspect the results and to extract only the hallmark gene sets that had a corrected *p*-value $< = 0.1$. With the *table* function we counted how many hallmark gene sets were in the proliferative group and how many were in the differentiated samples.

```
hallmark_df <- as.data.frame(compareHallmark)
sig_hallmark_df <- hallmark_df[hallmark_df$p.adjust <= 0.1, ]
table(sig_hallmark_df$Cluster)
```

subtype	#sig H gene set
Prolif	6
Dif	12

As can be seen above, 18 different hallmark processes were found, which characterize the two groups of patients. With the following code we show their subtype specificity, their names and their corrected *p*-values (Table 3).

Again, the analysis highlights specific differences between the two groups under study. Nonetheless, the results are consistent with all the previous analyses, that is, we found changes in immune-related gene sets in the differentiated samples and WNT related pathways in the proliferative subtypes.

The following code which shows the *dotplot* function can be used to create a plot of the expression differences (plot not shown).

```
dotplot(compareHallmark, showCategory=20)
```

3.3.4 The Computational Inference of Protein Activity Through a Network Approach

So far, we have seen how different analyses can help in understanding the processes occurring in the samples under evaluation. In the following section, we describe the analysis of protein activity. Briefly, we use gene expression regulated by a given protein, for example, the targets of a transcription factor (TF) or of signal transduction proteins, which act as reporters of the protein activity of the regulator. We use the *viper* R package (Virtual Inference of Protein-activity by Enriched Regulon analysis), which allows the

Table 3
Subtype specificity, names, and their corrected *p*-values

Subtype	H Gene set	*p*.adjust
Prolif	EPITHELIAL_MESENCHYMAL_TRANSITION	0.0757
Prolif	SPERMATOGENESIS	0.0757
Prolif	HEDGEHOG_SIGNALING	0.0757
Prolif	PANCREAS_BETA_CELLS	0.0757
Prolif	WNT_BETA_CATENIN_SIGNALING	0.0757
Prolif	APICAL_JUNCTION	0.0757
Dif	TNFA_SIGNALING_VIA_NFKB	1.29e-05
Dif	ALLOGRAFT_REJECTION	2.23e-05
Dif	INFLAMMATORY_RESPONSE	0.000229
Dif	INTERFERON_GAMMA_RESPONSE	0.000513
Dif	COMPLEMENT	0.00456
Dif	APOPTOSIS	0.00527
Dif	COAGULATION	0.00527
Dif	ESTROGEN_RESPONSE_LATE	0.00527
Dif	ESTROGEN_RESPONSE_EARLY	0.00558
Dif	KRAS_SIGNALING_UP	0.0481
Dif	MYOGENESIS	0.0481
Dif	KRAS_SIGNALING_DN	0.0481

computational inference of protein activity to be calculated from gene expression profile data [22]. Thus, with *viper* we can study the regulon, which is a context-specific coregulatory network. The Ovarian Cancer regulon has been provided by the R package *aracne.networks* where different regulons for the pancancer datasets are stored.

In the following code, we (1) import the Ovarian Cancer regulon, (2) create an *ExpressionSet* (eset) with the normalized data and the phenotype data (i.e., patient subtypes), and finally (3) call *viper*.

```
library(viper)
library(aracne.networks)
library(Biobase)

data(regulonov)

select_columns <- multiOmicDataset$patients_subtype$subtype=="OVCA.Differentiat
ed"
                   | multiOmicDataset$patients_subtype$subtype=="OVCA.Proliferati
ve"

pheno=droplevels(multiOmicDataset$patients_subtype[select_columns, ,drop=F])
pheno <- pheno[order(pheno$subtype), , drop=F]

TCGA_OV_matrix <- as.matrix(TCGA_OV_deg$lnorm_data[,row.names(pheno),drop=F])
eset <- ExpressionSet(assayData = TCGA_OV_matrix, phenoData = AnnotatedDataFram
e(pheno))

proteins_activity <- viper(eset, regulonov, verbose = FALSE)
activity_inferred <- exprs(proteins_activity)
```

We found 5708 genes whose protein activity has been inferred from the Ovarian Cancer regulon. With this information we can search for the activity of known TFs. Using the Homo sapiens annotation package, we can retrieve TFs (e.g., using GO terms related to TFs) and then search in the *viper* results.

```
library(org.Hs.eg.db)

GO_2_KEEP_TF <- c("GO:0003700", "GO:0004677", "GO:0030528", "GO:0004677")

gene.list <- as.list(org.Hs.egGO2EG)
regulator_TF <- unname(unlist(gene.list[GO_2_KEEP_TF]))
symbols <- AnnotationDbi::select(org.Hs.eg.db, keys = regulator_TF,
                                 columns = "SYMBOL")
sym <- symbols$SYMBOL
names(sym) <- symbols$ENTREZID
tfs <- intersect(row.names(activity_inferred), symbols$ENTREZID)
```

Using the above code, we retrieved 469 TFs with their estimated protein activity. We then proceed to extract TF activity.

```
tfs_activity <- activity_inferred[tfs,]
tfs_symbols <- sym[tfs]
```

Following this, we wanted to see the TFs that were differentially activated between the proliferative and differentiated subtypes. For each TF, we computed the difference between the mean activity in proliferative and differentiated subtypes. Then we performed a t-test between the two and extracted the TFs with a *p*-value $< = 0.01$ and an absolute mean difference $> = 2$. This will represent the subtype-specific TFs active at the protein level.

```
tfs_activity_diff <-
  rowMeans(tfs_activity[,pheno$subtype==levels(pheno$subtype)[1]])-
  rowMeans(tfs_activity[,pheno$subtype==levels(pheno$subtype)[2]])

tfs_activity_tt <- apply(tfs_activity, 1, function(profile) {
  x <- profile[pheno$subtype==levels(pheno$subtype)[1]]
  y <- profile[pheno$subtype==levels(pheno$subtype)[2]]
  t.test(x,y)$p.value
})

selected <- tfs_activity_tt <= 0.01 & abs(tfs_activity_diff) >= 2

tfs_activity_top <- tfs_activity[selected,]
tfs_symbols_top <- sym[selected]
```

A total of 26 TFs were detected to be differentially activated in the two subtypes. To visualize them, we suggest creating a heatmap with their expression activity, as we did in Fig. 3d.

```
pheatmap(tfs_activity_top, cluster_cols = F, annotation_col = pheno,
         labels_row=tfs_symbols_top, show_colnames = F)
```

3.3.5 Topological Pathway Analysis

In the sections above we have described how to study gene expression data using gene sets. Now we would like to show a type of pathway analysis which is able to exploit the pathways in the form of networks: topological pathway analysis. Contrary to gene set analyses, few topological methods have been published. The first and probably the most famous one is SPIA [27]. In this section, we demonstrate the Bioconductor R package *clipper*.

This method, based on Gaussian Graphical Models, has the interesting and biologically meaningful ability to decompose a pathway, which are sometimes too large and too general. This method can then extract the subnetwork—part of the original pathway—including the strongest differences between the two categories under study [12]. In Fig. 4, a general scheme on how

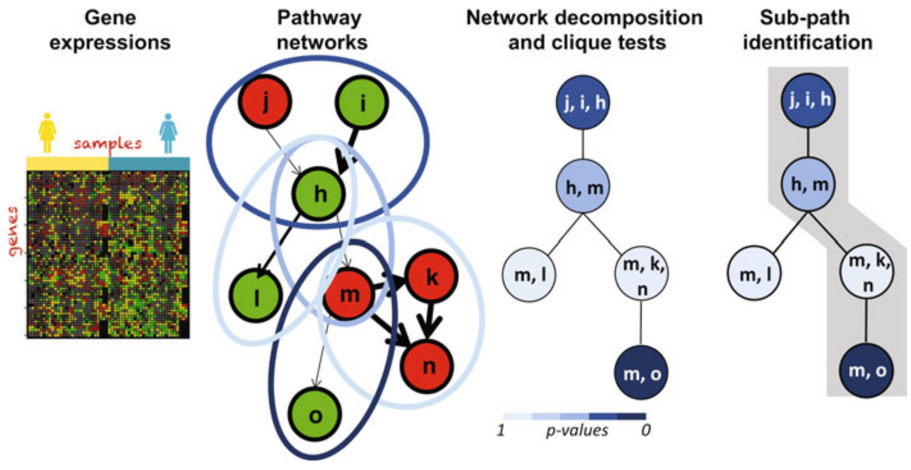


Fig. 4 Schematic overview of *clipper* approach

clipper works is shown. *clipper* transforms the pathway-network into a junction tree (a tree of cliques, which are groups of fully connected genes) and explores the tree clique by clique. It does so by comparing the gene expression in the two categories in terms of mean (changes in gene expression magnitude) and variance (changes in the strength of the relationship between connected genes) of expression values. In contrast to other methods, *clipper* allows the identification of small but coordinated differences between the two classes. *clipper* analysis needs pathways in the form of networks as provided by the *graphite* R package [8].

In the following section, we demonstrate a *clipper* analysis which was performed on a set of Reactome pathways related to "PD-1 signaling".

Reactome has a hierarchical structure (similar to those seen for the GO) where small and well-focused pathways are contained within bigger and more generic pathways. The hierarchy is browsable through the Reactome Pathway Browser at "https://reactome.org/PathwayBrowser/". In the following code, we downloaded the Reactome hierarchy using the *downloadPathwayRelationFromReactome* function from the *houseOfClipUtility* R package (from "https://github.com/cavei/houseOfClipUtility"). Then, we selected the Reactome pathway IDs and associated the pathway names.

```
pathHierarchy <- houseOfClipUtility::downloadPathwayRelationFromReactome()

ids=sapply(hsapiens_reactome_eg, function(x) x@id)
id2path <- names(ids)
names(id2path) <- ids
```

With the following commands, we extracted the direct children pathways of "Adaptive Immune System" and, with a second round of selection, those located two steps further (i.e., the children of the children, which include "PD-1 signaling"). The resulting vector contains the names of the pathways of interest. To conclude the preparation, the pathways extracted as *Pathway* objects from *graphite* were converted into *graphNEL* objects.

```
adaptive_imm_system_childs <- pathHierarchy[pathHierarchy$parent=="R-HSA-128021
8", "child"]
selected_childs <- pathHierarchy$parent %in% adaptive_imm_system_childs
adaptive_imm_system_2nd_gen <- pathHierarchy[selected_childs, "child"]

interesting_reactome_pathways <- c(id2path[adaptive_imm_system_childs],
                                   id2path[adaptive_imm_system_2nd_gen])

reactomeGraph <- lapply(hsapiens_reactome_eg[interesting_reactome_pathways], pa
thwayGraph)
```

Following this, we performed *clipper* analysis using the list of *graphNEL* pathways, the normalized expression matrix with proliferative and differentiated samples (stored as *lnrom_prol_diff*), and the sample class annotations (stored as *annotation_prol_diff*).

The analysis was performed by cycling over the *reactomeGraph* object which contains the pathways. For each pathway graph *g*, we call *clipper* with the method "mean" over the expression matrix with the subtype classes.

```
all_clipped <- lapply(reactomeGraph, function(g) {
  set.seed(1234)
  clipped <- clipper(lnrom_prol_diff,
                     classes = as.numeric(annotation_prol_diff$subtype),
                     graph = g, method = "mean")
})
```

After the analysis we ordered the resulting pathways by *maxScore* and collapsed the results in a *data.frame*.

```
collapsed_first_results <- do.call(
  rbind, lapply(all_clipped, function(x) {
    sort_x <- x[order(as.numeric(x$maxScore), decreasing = T), , drop=F]
    sort_x[1, , drop=F]
    }))

sorter <- order(as.numeric(collapsed_first_results$maxScore), decreasing = T)
ordered_collapsed_first_results <- collapsed_first_results[sorter, ]
```

In the following lines we have reported the top ten pathways according to the *clipper maxScore* (Table 4).

For visualization purposes, we extracted from the result table the column *length*, which reports how many cliques are involved.

Table 4
Top ten pathways according to the *clipper maxScore*

	Length	maxScore	Impact
TCR signaling	9	54.92	0.69
Antigen processing-cross presentation	5	34.54	0.5
Costimulation by the CD28 family	4	27.63	0.57
B Cell activation	4	27.63	0.33
Downstream TCR signaling	4	27.63	0.4
CTLA4 inhibitory signaling	4	27.63	1
Antigen activates B cell receptor (BCR) leading to generation of second messengers	4	27.63	0.67
CD28 co-stimulation	3	20.72	0.6
Downstream signaling events of B cell receptor (BCR)	3	19.63	0.5
Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell	2	13.82	0.11

The higher the number of cliques, the longer the signaling path. Furthermore, the score (*maxScore*, which indicates the max cumulative score along the path) together with the *impact* tells us how much of the pathway is affected by the perturbation.

```
ordered_collapsed_first_results[1:10, c(4,5,8)]
```

Then, we decided to focus on the tenth path, which is part of the pathway "Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell," which has two cliques that seem to drive class differences, thus impacting 11% of the pathway. In the following code, we extracted the genes involved and evaluated the expression differences with a heatmap of gene expression.

```
selected_egenes <- unique(unlist(strsplit(ordered_collapsed_first_results[10,
"involvedGenes"], ";")))
selected_genes <- gsub("ENTREZID:", "", selected_egenes)
symbols <- AnnotationDbi::select(org.Hs.eg.db, keys = selected_genes,
                                  columns = "SYMBOL")

Imm_expression <- lnrom_prol_diff[selected_egenes, , drop=F]
row.names(Imm_expression) <- symbols$SYMBOL
breaksl <- seq(-1,10)
palette <- colorRampPalette(rev(RColorBrewer::brewer.pal(n = 7, name =
"RdYlBu")))(length(breaksl))
library(RColorBrewer)

pheatmap(Imm_expression, cluster_cols = F, annotation_col =
annotation_prol_diff,
         clustering_distance_cols = "correlation",
         clustering_distance_rows = "correlation",
         show_colnames = F,
         breaks = breaksl,
         color = palette)
```

From the heatmap in Fig. 5a, we can say that there is a small but highly coordinated signal in the differentiated samples.

The involved genes in the original pathway "Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell" can be seen by going to the Reactome home page (https://reactome.org/) and selecting the "Analyze data" section. In the dedicated box the list of entrez gene IDs can be entered, and the pathway of interest can be browsed. Alternatively, the network representation of the pathway can be plotted in *Cytoscape* through the *graphite* function. In *Cytoscape*, a world of aesthetic customization is available, from network layouts to node and edge styles. In the following code, we show how to plot a *graphite pathway* object in *Cytoscape* and how to pass the *clipper* analysis to the *Cytoscape* session. First, we selected the pathway that we wanted to plot, then we extracted all the genes, and converted them into symbols using the annotation package.

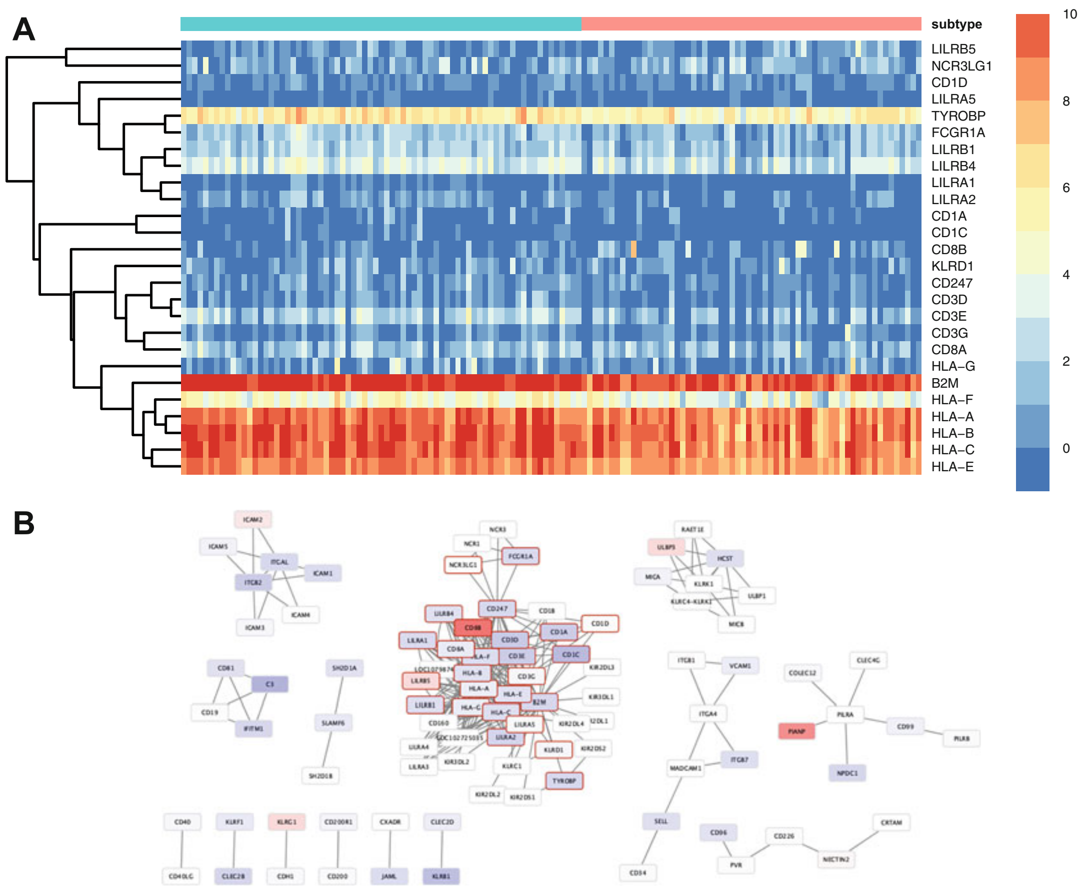


Fig. 5 *clipper* analysis plots. (**a**) Gene expression of the portion of the "Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell" pathway found by *clipper* as the one which better characterizes the proliferative and differentiated phenotypes. (**b**) Network representation of the "Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell" pathway in *Cytoscape*. Nodes are colored according to their log fold change in proliferative *vs* differentiated groups (blue represents differentiated, red represents proliferative). The node borders of the genes of the most important pathway portions found by *clipper* are shown in red (genes in panel **a**)

```
pathway_name <- "Immunoregulatory interactions between a Lymphoid and a non-
Lymphoid cell"
path_graph <- reactomeGraph[[pathway_name]]
allgene <- gsub("ENTREZID:", "", nodes(path_graph))
symbols <- AnnotationDbi::select(org.Hs.eg.db, keys = allgene, columns =
"SYMBOL")
```

Using the *graphite* function *cytoscapePlot*, we can plot the pathway in *Cytoscape*. It is necessary to have *Cytoscape* version 3.7.1 or later open on the computer before running the following codes, as they generate the pathway directly into the *Cytoscape* session.

```
library(graphite)
cytoscapePlot(hsapiens_reactome_eg[[pathway_name]])
```

The next step is to color the genes that best discriminate the two subtypes according to *clipper* analysis. First, we identified the genes that compose the path (*selected_genes*). In *Cytoscape*, coloring selected genes requires the creation of an attribute *data.frame* containing: (1) the node ID, to map attributes to the respective nodes, (2) symbols, which will be used as node labels and (3) a 1–0 vector, indicating if the gene is relevant or not according to the analysis. (4) We also included the log fold change computed from *edgeR* to color the genes in the pathway.

```
clippeg_logical <- symbols$ENTREZID %in% selected_genes
lfc <- TCGA_OV_deg$DE_tables[nodes(path_graph), "logFC"]
lfc[is.na(lfc)] <- 0.0
attrs <- data.frame(id = nodes(path_graph), SYMBOL = symbols$SYMBOL,
                    clippeg = ifelse(clippeg_logical, 1, 0), lfc = lfc)
```

The new attributes were loaded to the *Cytoscape* session with the *loadTableData* function (*RCy3* R package) using the "id" column as a bridge to match the id in the node table.

```
library(RCy3)
loadTableData(attrs, data.key.column = "id", table = "node")
```

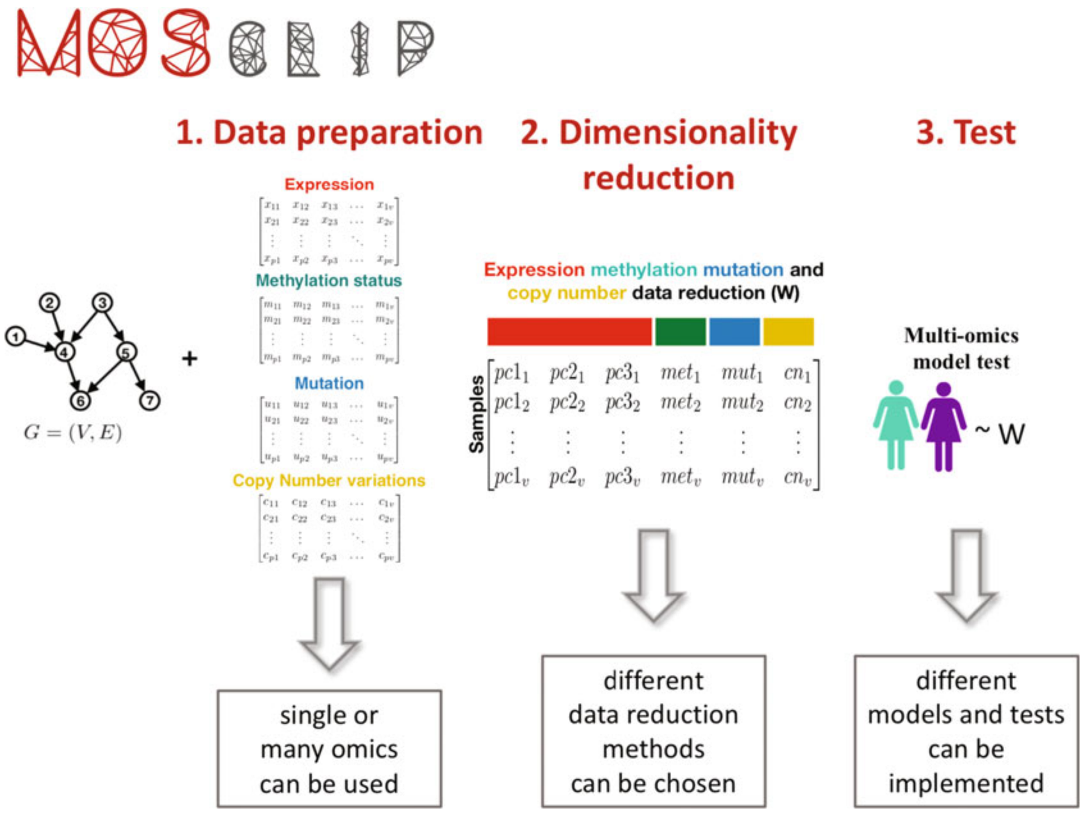
With the new attributes loaded, it is possible to modify the graph aesthetics (see RCy3 or Cytoscape manual).

```
setNodeLabelMapping("SYMBOL")

column <- 'lfc'
control.points <- c (-3.0, 0.0, 3.0)
colors <- c ("#686de0", "#FFFFFF", "#eb4d4b")
setNodeColorMapping(column, control.points, colors)

column <- 'clippeg'
control.points <- c (0, 1)
colors <- c ('#dcdde1', '#c23616')
setNodeBorderWidthDefault(4)
setNodeBorderColorMapping(column, control.points, colors)
```

Figure 5b shows the network exported from *Cytoscape*. We set the node color according to the fold change (red for proliferative and blue for differentiated samples), the node border in red for gene results from *clipper* and grey for the others.

3.4 Survival Analyses of a Multiomics Dataset Using Pathways

Finally, in this last section we would like to demonstrate survival pathway analysis of a multiomic dataset using MOSClip (https://github.com/cavei/MOSClip). In Fig. 6, an overview of the *MOSClip* strategy is shown. In brief, for each of the omics to be analyzed, a data reduction strategy guided by pathway topology is



MOSClip
1. Data preparation
2. Dimensionality reduction
3. Test
Expression
Methylation status
Mutation
Copy Number variations
G = (V, E)
+
Expression methylation mutation and copy number data reduction (W)
Samples
Multi-omics model test
~ W
single or many omics can be used
different data reduction methods can be chosen
different models and tests can be implemented

Fig. 6 *MOSClip* approach overview

performed. We devised an omic summarization that can be tested for survival on a per pathway or per modules (the pathway cliques) basis. In this way the survival analyses gain power from the analysis of multiple pathway genes in multiple omics.

Shown below is the code in which we used the full multiomic OV-TCGA dataset composed of expression, mutations, CNVs and methylation for 259 ovarian cancer patients, as used in [18]. Data retrieval, data preprocessing and analysis are fully explained by the *MOSClip* tutorial available with the R package at "https://cavei.github.io/MOSClip/". Here as an example, we chose to investigate the Reactome "Adaptive Immune System" related pathways.

We needed to build a *multiOmic* dataset object by feeding it with the expression data (normalized logCPM), the methylation data (organized in clusters), the CNV matrix and the mutations as event matrices. To create a *multiOmic* dataset, we needed to specify the four omics matrices, the gene summarization method for each data matrix, and other arguments related to the summarization method chosen. For example, in the following commands, we set that the expression needed to be summarized with the function *summarizeWithPca* which would then be run with parameters listed by *pcaArgs*, and so on for the other matrices.

```
library(MOSClip)
multiOmics <- Omics(data = list(expr = expression,
    met = methylation$MET_Cancer_Clustered,
    mut = mutation, cnv = cnv),
    methods = c("summarizeWithPca",
                "summarizeInCluster",
                "summarizeToNumberOfEvents",
                "summarizeToNumberOfDirectionalEvents"),
    specificArgs = list(
      pcaArgs = list(name = "exp", shrink = "FALSE",
                     method = "sparse", maxPCs = 3),
      clusterArgs = list(name = "met", dict = methylation$eMap,
                         max_cluster_number = 3),
      countEvent = list(name = "mut", min_prop = 0.05),
      cnvAgv = list(name = "cnv", min_prop = 0.05)))
```

Then, we selected the pathways to be analyzed. We used the list of "interesting Reactome pathways" which contains the "Adaptive Immune System" and all of its children as described above in the paragraph relating to *clipper*.

```
ractome_from_degs <- hsapiens_reactome_eg[interesting_reactome_pathways]
```

Following these steps, *MOSClip* pathway analysis is now ready to be run. With the "lappy" function, we cycled over the selected pathways (*reactome_from_degs*) and performed a pathway multiomic survival test (*multiOmicsSurvivalPathwayTest* function) using the *multiOmics* object, the *g*th pathway, and the progression-free survival annotation (thus limiting the usable genes to those with mRNA expression profiles (saved on the *geneToConsider* object). A more in-depth version of the loop containing progress through the analyzed pathway can be found by uncommenting "*print (g@title)*".

```
genesToConsider <- row.names(expression)
multiOmicsPathway <- lapply(ractome_from_degs, function(g) {
        # print(g@title)
        set.seed(1234)
        fcl = multiOmicsSurvivalPathwayTest(
          multiOmics, g, survAnnot, useThisGenes = genesToConsider)
        fcl
    })
```

At this point, all results have been stored into a *multiOmicsPathway* object. For a global view of the pathway analysis results, we explored the multipathway report plot. Using the function *plotMultiPathwayReport* from *MOSClip* we created a heatmap ordered by overall *p*-value with pathways in rows and the *p*-value for each omic in the columns for the first top ten pathways (figure not shown).

Table 5
First five lines of a tabular report

	p-Value	expPC1	expPC2	expPC3	met2k2	met3k2	met3k3	mut	cnvNEG	cnvPOS
CD28 co-stimulation	0.02	0.41	0.02	0.67	0	NA	NA	0.63	0.11	0.29
Costimulation by the CD28 family	0.02	0.09	0.85	0.03	0	NA	NA	0.38	0.79	0.91
B Cell activation	0.07	0.4	0.36	0	0.05	NA	NA	0.49	0.81	0.92
CD22 mediated BCR regulation	0.07	0.01	0.74	NA	0.3	NA	NA	NA	NA	0.34
Generation of second messenger molecules	0.12	0.29	0.99	0.36	0.25	NA	NA	0.02	NA	0.88

```
plotMultiPathwayReport(multiOmicsPathway,
                       MOcolors = c(exp = "red", met = "green",
                                    mut = "blue", cnv = "yellow"),
                       top = 10, fontsize = 10, fontsize_number = 10,
                       fontsize_row = 8, fontsize_col = 8)
```

Alternatively, the following line of code allow for the creation of a tabular report. In Table 5, we have shown the first five lines of such a report.

```
multiPathwayreport <- multiPathwayReport(multiOmicsPathway)
```

As you can see from the top five results, out of the 21 pathways analyzed, two have an overall *p*-value lower than 0.05. Focusing on the first pathway, “CD28 co-stimulation” we can see that the survival associated components are expression (*expPC2*) and methylation (*met2k*). To further investigate the pathway, we can plot the pathway heatmap where the most highly significant genes of each omics are shown. We ran the *plotPathwayHeat* function from *MOSClip*, asking it to sort patients by methylation (“met2k”) and by expression (“expPC2”). The other parameters were used to set the colors of the omics (*paletteNames*), the ratio aspect of the heatmaps (*nrowsHeatmaps*) and to remove the sample names from the plot ($withSampleNames = F$).

```
plotPathwayHeat(multiOmicsPathway[["CD28 co-stimulation"]],
                sortBy = c("met2k", "expPC2"),
                paletteNames = c(exp = "red", met = "green",
                  mut="blue", cnv = "yellow"),
                nrowsHeatmaps = 2, withSampleNames = F)
```

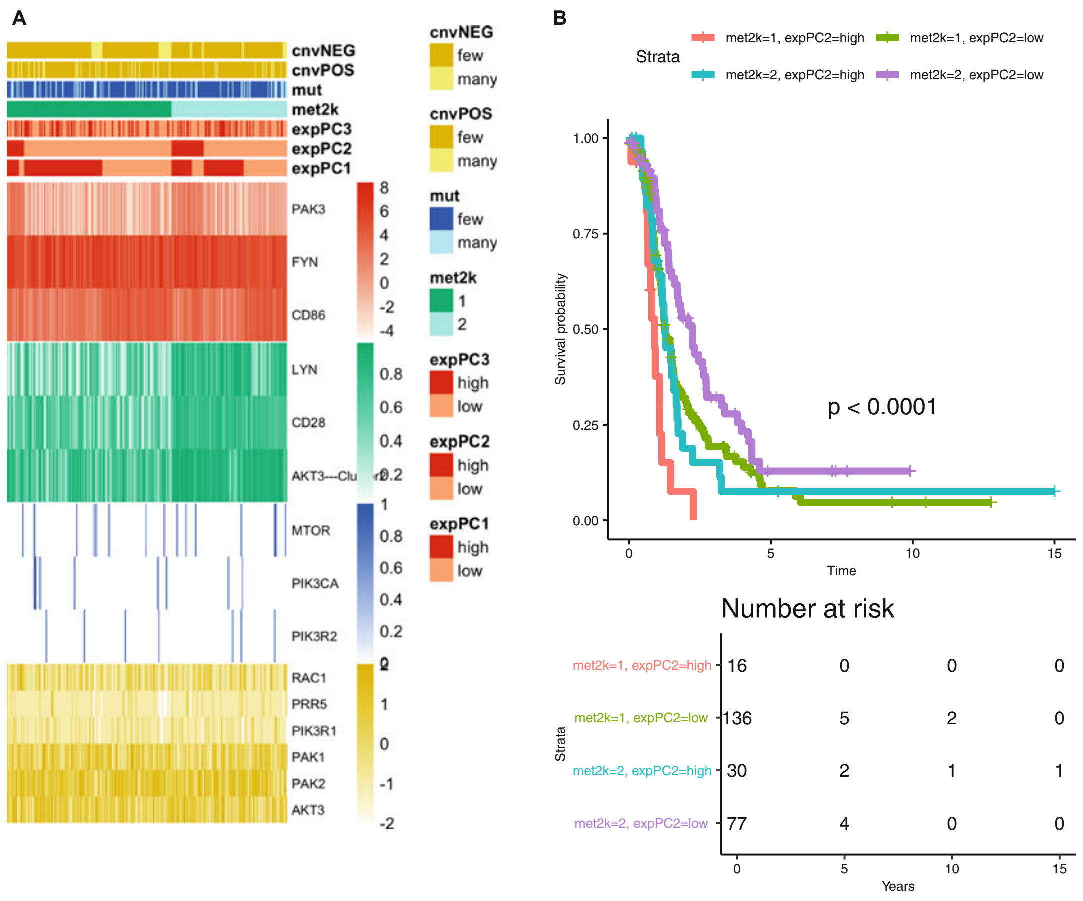


Fig. 7 *MOSClip* analysis plots. (**a**) “CD28 co-stimulation” visualization of the most relevant gene for each omics. Patients are sorted according to methylation (met2k) plus expression (expPC2). (**b**) Kaplan-Meier survival curves combining classes from methylation and expression

As seen in Fig. 7a, the methylation status on the promoter of the genes LYN, CD28 and AKT3 seem to be the major points responsible for the separation of the pathway methylation profile in the two classes. The expression of the gene PAK3 is the main contributor to the PC2. Furthermore, the plot also offers insight into the mutations and CNV omics in the pathway. While this plot shows us the best gene candidates for association with survival, we can also show the Kaplan-Meier curves obtained by dividing patients into groups by using *met2k* and *expPC2*. Thus, we can evaluate if methylation and expression profiles of the pathway genes can stratify patients into classes with different survival rates. We used the *plotPathwayKM* function from *MOSClip,* which takes the Survival formula where we specified that we want to correlate survival status and time (days) to the combination of *met2k* and *expPC2* covariates.

```
plotPathwayKM(multiOmicsPathway[["CD28 co-stimulation"]],
              formula = "Surv(days, status) ~ met2k + expPC2")
```

Figure 7b shows the four classes of patients, demonstrating significant differences in survival time (p-value < 0.0001).

In the above example, we focused only on selected pathways; however, in [18], a more complete analysis was proposed. In this study, the authors collapsed all resulted genes into a network. This network was used to separate the patients in poor survival, middle survival and high survival, thus identifying a new pathway that included all survival-associated genes (in a multiomics fashion) that are able to predict a patient's outcome.

Figure 8 shows an example of what this network could look like. Genes have been colored according to their most important omic and their association with poor survival (e.g., increased or decreased expression, high or low methylation). This example shows how topologically aware tools and appropriate visualization systems can also assist in getting mechanistic insight into survival mechanisms.

That is all. We hope that all the gene expression analyses presented in this chapter would help the researchers to study the cell activities occurring in their samples.

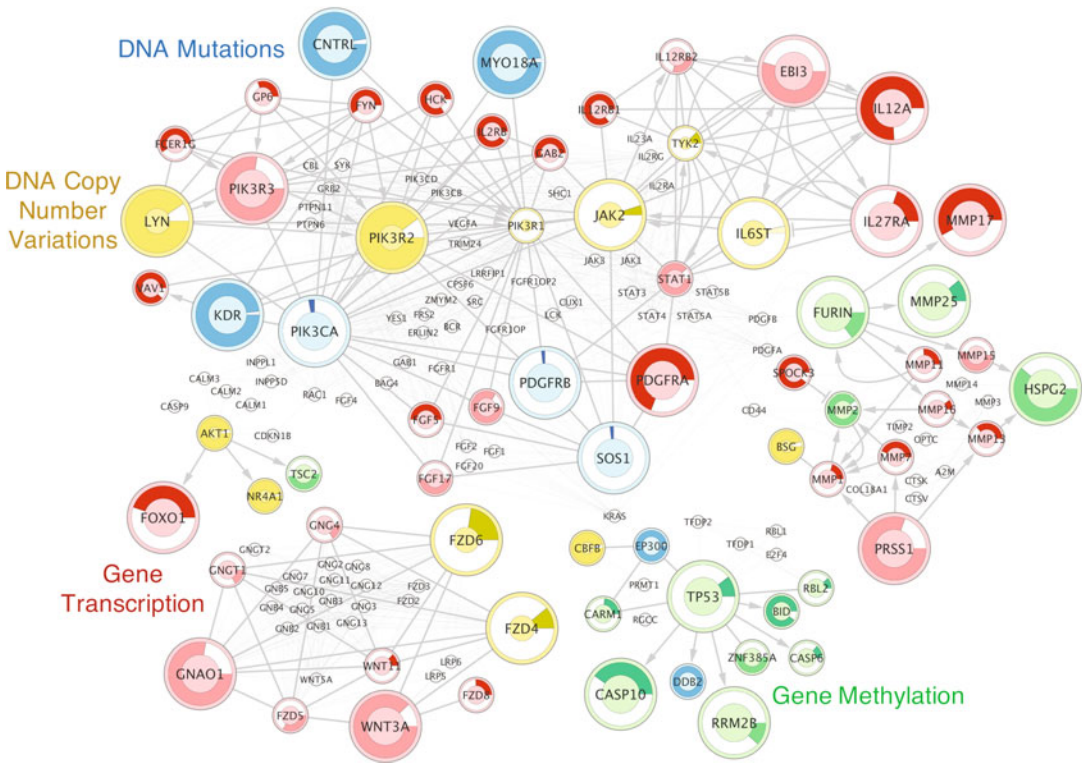


Fig. 8 Bubble network samples from *MOSClip* analysis

And as said by a professor of our university: “Measure what is measurable, and make measurable what is not so.” Galileo Galilei 1564–1642.

4 Notes

1. When working with nonmodel organisms or organisms with no “AnnotationDBI,” one solution could be to compile a GO database from an external source, transform it into a *TERM2-GENE* data frame and run enrichment analysis with *enricher*. The following chunk of code shows how to perform this task, starting from the GO annotated into *MSigDB* within the *msigdbr* R package.

```
library(msigdbr)
mouse_df <- msigdbr(species = "Mus musculus")
gobp_t2g <- msigdbr(species = "Mus musculus", category = "C5", subcategory = "BP")
%>% select(gs_name, gene_symbol)
head(gobp_t2g)

alt_eGOBP <- enricher(IRF6_dataset$up_down_regulated$up,
                      TERM2GENE = gobp_t2g,
                      universe = row.names(IRF6_dataset$lnorm_data),
                      pAdjustMethod = "BH",
                      pvalueCutoff = 0.1, qvalueCutoff = 0.05,
                      minGSSize = 10, maxGSSize = 500)
barplot(alt_eGOBP)
```

Acknowledgments

This work was supported by the “My First AIRC grant” to Enrica Calura (MFAG 2019, Grant N. 23522), the IG grant to Chiara Romualdi (Grant N. IG 21837) both provided by Italian Association for Cancer Research (AIRC) and the European Molecular Biology Organization (EMBO) Short-Term Fellowship [8517] to Paolo Martini.

References

1. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT et al (2000) Gene ontology: tool for the unification of biology. Nat Genet 25:25–29
2. GeneOntologyConsortium (2019) The gene ontology resource: 20 years and still going strong. Nucleic Acids Res 47:D330–D338
3. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES et al (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci 102:15545–15550
4. Dolgalev I (2019) Msigdbr: MSigDB gene sets for multiple organisms in a tidy data format
5. Bader GD, Cary MP, Sander C (2006) Pathguide: a pathway resource list. Nucleic Acids Res 34:D504–D506

6. Kanehisa M, Goto S (2000) KEGG: Kyoto encyclopedia of genes and genomes. Nucleic Acids Res 28:27–30
7. Jassal B, Matthews L, Viteri G, Gong C, Lorente P, Fabregat A, Sidiropoulos K, Cook J, Gillespie M, Haw R et al (2020) The reactome pathway knowledgebase. Nucleic Acids Res 48:D498–D503
8. Sales G, Calura E, Cavalieri D, Romualdi C (2012) G raphite-a bioconductor package to convert pathway topology to gene network. BMC Bioinformatics 13:20
9. Oberbeck N, Pham VC, Webster JD, Reja R, Huang CS, Zhang Y, Roose-Girma M, Warming S, Li Q, Birnberg A et al (2019) The ripk4–irf6 signalling axis safeguards epidermal differentiation and barrier function. Nature 574:249–253
10. Network CGAR et al (2011) Integrated genomic analyses of ovarian carcinoma. Nature 474:609
11. Giorgi FM (2019) Aracne.networks: ARACNe-inferred gene networks from tcga tumor datasets
12. Martini P, Sales G, Massa MS, Chiogna M, Romualdi C (2013) Along signal paths: an empirical gene set approach exploiting pathway topology. Nucleic Acids Res 41:e19–e19
13. Yu G, Wang L-G, Han Y, He Q-Y (2012) ClusterProfiler: an r package for comparing biological themes among gene clusters. Omics 16:284–287
14. Robinson MD, McCarthy DJ, Smyth GK (2010) EdgeR: a bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26:139–140
15. Wickham H (2016) Ggplot2: elegant graphics for data analysis. Springer, New York, NY
16. Hanzelmann S, Castelo R, Guinney J (2013) GSVA: gene set variation analysis for microarray and rna-seq data. BMC Bioinformatics 14:7
17. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015) Limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res 43:e47–e47
18. Martini P, Chiogna M, Calura E, Romualdi C (2019) MOSClip: multi-omic and survival pathway analysis for the identification of survival associated gene and modules. Nucleic Acids Res 47:e80–e80
19. Carlson M (2019) Org.Hs.eg.db: genome wide annotation for human
20. Carlson M (2019) Org.Mm.eg.db: genome wide annotation for mouse
21. Luo W, Brouwer C (2013) Pathview: an r/bioconductor package for pathway-based data integration and visualization. Bioinformatics 29:1830–1831
22. Alvarez MJ, Shen Y, Giorgi FM, Lachmann A, Ding BB, Ye BH, Califano A (2016) Functional characterization of somatic mutations in cancer using network-based inference of protein activity. Nat Genet 48:838
23. Liu J, Lichtenberg T, Hoadley KA, Poisson LM, Lazar AJ, Cherniack AD, Kovatich AJ, Benz CC, Levine DA, Lee AV et al (2018) An integrated tcga pan-cancer clinical data resource to drive high-quality survival outcome analytics. Cell 173:400–416
24. Darvin P, Toor SM, Nair VS, Elkord E (2018) Immune checkpoint inhibitors: recent progress and potential biomarkers. Exp Mol Med 50:1–11
25. Liberzon A, Birger C, Thorvaldsdóttir H, Ghandi M, Mesirov JP, Tamayo P (2015) The molecular signatures database hallmark gene set collection. Cell Syst 1:417–425
26. Liberzon A, Subramanian A, Pinchback R, Thorvaldsdóttir H, Tamayo P, Mesirov JP (2011) Molecular signatures database (msigdb) 3.0. Bioinformatics 27:1739–1740
27. Tarca AL, Draghici S, Khatri P, Hassan SS, Mittal P, J-s K, Kim CJ, Kusanovic JP, Romero R (2009) A novel signaling pathway impact analysis. Bioinformatics 25:75–82

Chapter 10

Computational Analysis of circRNA Expression Data

Giulio Ferrero, Nicola Licheri, Michele De Bortoli, Raffaele A. Calogero, Marco Beccuti, and Francesca Cordero

Abstract

Analysis of circular RNA (circRNA) expression from RNA-Seq data can be performed with different algorithms and analysis pipelines, tools allowing the extraction of heterogeneous information on the expression of this novel class of RNAs. Computational pipelines were developed to facilitate the analysis of circRNA expression by leveraging different public tools in easy-to-use pipelines. This chapter describes the complete workflow for a computationally reproducible analysis of circRNA expression starting for a public RNA-Seq experiment. The main steps of circRNA prediction, annotation, classification, sequence reconstruction, quantification, and differential expression are illustrated.

Key words Circular RNAs, RNA sequencing, Noncoding RNAs

1 Introduction

Circular RNAs (circRNAs) are an emerging class of RNAs arising from Back-Splicing (BS) events, noncollinear splicing events involving the acceptor splice site of downstream exons with a donor splice site of an upstream one [1]. These events create circular RNA transcripts characterized by high overall stability and peculiar molecular functions mediated by microRNAs and RNA binding protein interactions. The identification of circRNAs was revolutionized by the spreading of RNA-Seq experiments and the implementation of new computational approaches focused on the identification of unmapped reads and linear splicing junctions which supports BS events [2]. A large variety of tools was developed for circRNA prediction and their performance was extensively evaluated [3]. Furthermore, specific algorithms were designed for the circRNA postprediction analyses including the analysis of circRNA internal splicing events (CIRI-AS [4]), differential expression (circTest [5]), prediction of protein-coding potential (CircCode [6]), or visualization by circRNA dedicated browser (CircView [7]).

In the last few years, different groups provided computational solutions to collect circRNA analysis tools into a comprehensive analysis pipeline, including CirCompara [8], circtools [9], and Ulacirc [10]. To facilitate the computational analysis of circRNA expression with particular attention on the result reproducibility, our group developed the Docker4Circ pipeline [11]. In this pipeline, public algorithms for circRNA prediction, classification, annotation, quantification, and sequence reconstruction are embedded into different Docker images to facilitate the execution, porting, and reproduction of a whole circRNA analysis pipeline starting from RNA-Seq data. The users can easily run the pipeline through R functions designed in accordance with the guidelines of the Reproducible Bioinformatic Project (RBP [12]) or through an ad hoc Java Graphical User Interface [13]. The pipeline can be applied on data generated on human samples as well as a model organism and it can be easily applied on circRNAs predicted with eleven different tools.

In this chapter is described Docker4Circ a basic analysis workflow suitable for the prediction of circRNAs and their expression analysis from an RNA-Seq experiment. Exemplary data and analysis steps for their analysis are available at https://www.protocols.io/view/protocol-for-a-reproducible-circrna-analysis-using-9vmh646.

The Docker4Circ workflow is summarized in Fig. 1 The analysis is organized in four modules: (1) circRNA prediction with one or more algorithms starting from the fastq files; (2) circRNA classification and annotation based on the information stored in public databases; (3) analysis of the circRNA sequences in terms of alternative splicing events and reconstruction of the BS sequence; (4) quantification of the circRNA expression in independent RNA-Seq datasets and identification of deregulated circRNAs in tumor samples.

2 Materials

2.1 Exemplary Datasets

The exemplary RNA-Seq dataset **TNBC_circ** is available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE113230. This dataset was generated using 150 bp paired-end Illumina sequencing of total RNA extracted from three triple-negative breast cancer tissues and adjacent normal tissues surgically removed and flash frozen on dry ice. The generation and analysis of the dataset were described in [14].

The second exemplary dataset, named **BC_circ**, is available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE52194 and it is composed of 17 total RNA-Seq data of primary breast cancer samples and three normal breast organoids.

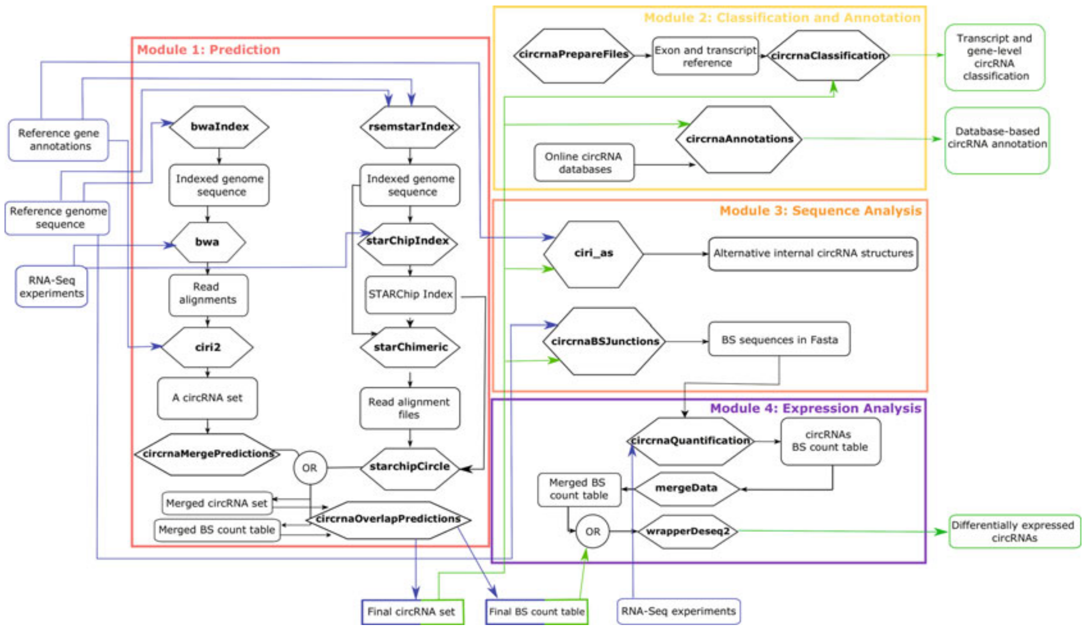


Fig. 1 Docker4Circ workflow. Four modules characterized the pipeline of analysis. Module 1 is dedicated to the circRNA prediction, Module 2 is focused on the classification and annotation, Module 3 to the sequence analysis and Module 4 for the expression analysis. Blue boxes and arrows are used to highlight the input data, while green boxes and arrows are used for the output data

This dataset is used to test the quantification module of the Docker4Circ pipeline.

The reference human genome selected for the analysis was hg38 which sequence can be retrieved from ftp://ftp.ensembl.org/pub/release-98/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz. The GTF file storing the Ensembl v98 annotations can be retrieved from http://ftp.ensembl.org/pub/release-98/gtf/homo_sapiens/Homo_sapiens.GRCh38.98.gtf.gz.

2.2 Computational Hardware

The computing hardware required for the analysis depends on part of pipeline considered and on the number of datasets analyzed. With exclusion of the *circrnaQuantification* function, the whole analysis can be run on standard workstation because the only requirement is 32 Gb of RAM available if the STAR Chimeric analysis is performed. In [11], the whole analysis pipeline was run successfully on Intel NUC6I7KYK mini-PC with eight threads and it required in total from six to ten hours based on the tool selected for the circRNA prediction.

2.3 Computational Software

The described workflow requires a UNIX or a MacOS operating systems, the installation of docker daemon (https://docs.docker.com/install/), the installation of R (https://cran.r-project.org/) version >3.00, the installation in R of *devtools* (install.packages ("devtools")), and *docker4circ* (library(devtools); install_github ("kendomaniac/docker4seq", ref="master")) libraries. In case Docker4Circ is used via graphical interface, Oracle JAVA version 8 has to be installed (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) and 4SeqGUI (https://github.com/mbeccuti/4SeqGUI) needs to be downloaded on the local computer.

3 Methods

3.1 Preparation of the Analysis Folders

The Docker4Circ analysis pipeline requires the indication of specific folders for the analysis: (1) a *data* folder storing in different subfolders, the fastq files related to each sample analyzed; (2) a *reference* folder in which the reference genome and the reference transcriptome annotations are stored; (3) a *scratch* folder in which the temporary analysis files are stored.

3.2 CircRNA Prediction

The first step of a circRNA analysis pipeline is the prediction of a circRNA set starting from the RNA-Seq datasets. Generally, circRNA prediction can be divided into an initial alignment phase in which the RNA-Seq fastq files are aligned against a reference genomic sequence with a selected set of gene annotations. Then, the reads neither aligned on the exons of reference annotations and on the linear splicing junctions are analyzed for candidate BS events. Several circRNA prediction tools were developed which differ by the chosen reads alignment algorithm and by the computational approach designed for the detection of the BS supporting reads [2, 3]. In Docker4Circ, two different circRNA prediction tools were included: CIRI2 [15] and STARChip [16]. Furthermore, the function *circrnaOverlapResults* is provided to overlap the list of circRNAs predicted with 11 different algorithms.

3.2.1 CIRI2 Prediction

In the CIRI2 prediction pipeline, the RNA-Seq reads are initially aligned on the genomic reference using the BWA algorithm. Then, the first step of the pipeline is the creation of the index of the selected reference genome sequence. The *bwaIndex* included in the Docker4Circ pipeline allows the creation of the genome index of a reference starting from a fasta file downloaded from a URL provided by the user. The function requires the following parameters: *genome.url*, the URL of the reference genomic sequence (in fasta format); *genome.folder*, the path to the folder where the indexed reference genome for BWA will be located; the argument *mode* set to *general*.

Then, using the *wrapperCiri* function it is possible to run the complete CIRI2 analysis pipeline for a specific sample. The analysis includes the FastQC quality control analysis of the RNA-Seq dataset, the BWA alignment, and the CIRI2 circRNA prediction. The function requires the following mandatory parameters: *data.folder*, the path to the folder where gzip fastq files are located; *genome.file*, the path to the fasta file of the reference genomic sequence (it should be the same reference indexed for the BWA alignment); *seq.type*, a string indicating if the experiment is single-end (se) and paired-end (pe) sequencing. Additional parameters include *sample.id*, a string indicating the id to be associated with the alignment bam that will be created; *threads*, a number indicating the number of cores to be used for the analysis; *annotation.file*, the path to the GTF/GFF file reporting the reference gene annotations; *max.span*, an integer indicating the maximum spanning distance of a circRNA (default = 200,000 bp); and *stringency.value*, related to the stringency level of the analysis. Three possible options are available: "high" (default), in which CIRI2 only provides circRNAs supported by more than two distinct paired chiastic clipping signals (PCC, specific partial alignment in the BWA output supporting a BS junction); "low" (low stringency), CIRI2 only provides circRNAs supported by more than two BS reads; "zero," CIRI2 provides all circRNAs regardless junction read counts or PCC signals; *quality.threshold,* integer indicating the threshold for mapping quality of each segment of junction reads (default = 10).

The output of the three analysis steps will be stored into the folder of each sample and it will consist of: (1) a files with extension *fastqc.zip* and *fastqc.html* storing the results of the FastQC analysis; (2) a file named *aligned_reads.sam* storing the result of the BWA alignment; (3) a tab-delimited file with extension *ciri2* reporting the list of predicted circRNAs.

3.2.2 STARChip Prediction

The second circRNA prediction pipeline included in Docker4Circ involved the STARChip algorithms. Since this tool processes the chimeric alignment from the STAR algorithm, the first step of the pipeline is the creation of the STAR index of the reference genome and transcriptome annotations. The STAR index can be generated using the function *rsemstarIndex* with the following mandatory parameters: *genome.folder*, the path to the folder where the indexed reference genome for STAR will be located; *ensembl.urlgenome*, the URL from ENSEMBL ftp for the unmasked genome sequence of interest; *ensembl.urlgtf*, the URL from ENSEMBL ftp for the GTF for genome of interest. Another function named *starChipIndex* must be applied in order to create the bed file required by STARChip analysis. The function requires as mandatory parameter, the path to the reference genome folder (*genome.folder*).

After the generation of the reference files, the analysis can be performed using the function *wrapperSTARChip* which sequentially executes the FastQC reads quality control (*fastqc* function), the STAR chimeric read alignment (*starChimeric* function), and the circRNA prediction using STARChip (*starchipCircle*). The mandatory parameters of the wrapper functions are: *genome.folder*, the path to the folder where the indexed reference genome for STAR is located; *samples.folder*, the path to the folder where are located all the subfolders of the samples processed with starChimeric; *threads*, the number of threads to be used; *reads.cutoff*, the minimum number of back-splicing reads required; *min.subject.limit,* the minimum number of samples with readsCutoff reads; *do.splice*, a boolean value indicating if the splices within the circRNA be detected and reported. Linear splices are searched within each circRNA in each sample. Any linear splice with > =60% of the read count of the circRNA is considered a splice within the circRNA. Two files are then created, ".consensus" with the most common splice pattern, and ".allvariants" with all reported splice patterns; *annotation*, boolean value indicating if the circRNAs will be provided with the gene annotations. Additional parameters include *chimSegmentMin*, indicating the minimal length of the overlap of a read to the chimeric element (default = 0); *chimJunctionOverhangMin*, a number indicating the minimum overhang for a chimeric junction (default = 15); *cpm.cutoff*, the minimum number of log2(CountsPerMillion) reads supporting the circRNAs (default = 0); and *subjectCPM.cutoff*, a value representing the lower limit for number of samples required to have the circRNA expressed at a value higher than *cpmCutoff*.

The output of the function are four files with different extension reporting respectively: the table with the circRNA expression in the analysed samples (file with extension *countmatrix*); the STARChip annotation of the identified circRNAs (extension *annotated*); the information of the genes (extension *genes*) or the specific exons in the BS events (extension *investigate.consensus*) involved.

3.2.3 Overlap of Multiple circRNA Predictions

Since a list of circRNAs can be predicted specifically for each sample, Docker4Circ includes the *circrnaMergePredictions* function allowing the merge of multiple prediction into a single matrix. The merging process uses prediction files obtained using the same prediction tool, which are recognized by the filename extension (for example all CIRI2 predictions must be named with the suffix .ciri2). Supported tools include CFS (suffix cfs), CIRI (ciri), CIRI2 (ciri2), Find_Circ2 (findcirc2), CIRCexplorer (circexplorer), CIRCexplorer2 (circexplorer2), DCC (dcc), KNIFE (knife), STARChip (starchip), Uroborus (uroborus), and circRNA_Finder (circrnafinder). *circrnaMergePredictions* executes also a filter based on BS reads detected in each experiment and the average reads computed among replicates of the same biological condition. The mandatory parameters of the function are: *data.folder*, the path to folder where output files are located; *samples.list*, a vector indicating the identifiers of the samples; *covariates.list,* a vector indicating the classes of the samples; *covariate.order*, a character vector indicating the order of covariates in the output file; *used.tool*, the name of the tool used for the prediction. Additional parameters include *min_-reads,* the minimum number of reads supporting a circRNA (default = 2); *min_reps*, the minimum number of replicates with at least min_reads (default = 0); and *min_avg*, the average number of BS reads across biological replicates of the same experimental condition (default = 10).

The output of *circrnaMergePredictions* is a table containing the detected circRNAs associated with the number of BS reads obtained in each input sample.

Furthermore, Docker4Circ includes the function *circrnaOver-lapResults* list of circRNAs predicted by different algorithms. The function automatically searches within a user-defined folder, each circRNA prediction table named with the suffix "tool.txt" where tool is the name of the tool used for the prediction. Supported tools are ACFS (suffix cfs), CIRI (ciri), CIRI2 (ciri2), Find_Circ2 (find-circ2), CIRCexplorer (circexplorer), CIRCexplorer2 (circex-plorer2), DCC (dcc), KNIFE (knife), STARChip (starchip), Uroborus (uroborus), and circRNA_Finder (circrnafinder). The mandatory parameters for the *circrnaOverlapResults* function are: *input.folder*, the path of the folder containing the predictions to overlap; *output.folder*, string indicating the path of the output folder; *min_support*, the minimum number of algorithms who detected the circRNAs.

The output of *circrnaOverlapResults* is a tab-delimited file named with the suffix *detection.txt* reporting the list of circRNAs with the indication of their genomic coordinates, the number and the name of the tools in which they were predicted. A second output file named *circRNA_list.txt* will report the list of circRNAs detected. This list will be generated considering the minimum number of tools chosen by the user.

3.3 circRNA Classification

Attribution of a circRNA to a specific gene annotation is not trivial since many overlapping transcripts can share the same BS exons. In Docker4Circ is included a function named *circrnaClassification* designed to classify each predicted circRNA based on the genomic coordinates of the BS exons and those from a set of Ensembl transcripts annotations. The function requires an initial preprocessing of the Ensembl annotation using the function *circrnaPrepar-eFiles* which generates two files reporting respectively the exons and the transcript isoform annotations based on the Ensembl annotations. The mandatory parameters required by this function are: *data.folder,* the path to the folder where the output files will be

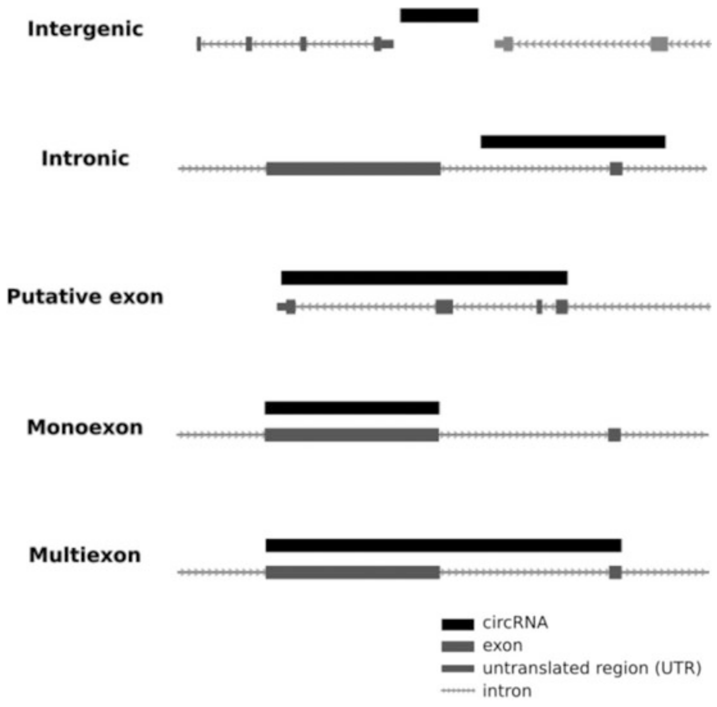


Fig. 2 Schematic representation of the five circRNA classifications available in Docker4Circ

saved; *assembly*, the identifier of the species analysed. The compatible assemblies are hg19 (default), hg18, hg38, mm9, mm10, rn6, dm6, and ce11. In addition, the parameter *version* can be provided indicating the specific version of Ensembl used for the analysis. Then, using these reference annotations, the *circrnaClassification* can be applied providing the following parameters: *circrna.data*, the path to the list of circRNAs; *exon.data*, the path to the exon annotation file; *isoform.data*, string indicating the path to the isoform annotation file; *assembly*, the reference human genome assembly analyzed.

The output of the function will be two files: *circRNA_classification.txt* reporting for each circRNA, the classification computed with respect to each transcript overlapping the genomic regions involved in the circularization; *circRNA_univocal_classification.txt* reporting the gene level classification obtained by considering the classification of the canonical isoform of each gene. As reported in Fig. 2, five classifications can be determined based on the genomic position involved in the BS event: (1) i*ntergenic*, one or both BS boundaries are mapped outside a genic region; (2) *intronic*, one or both BS boundaries are mapped within and intronic region; (3) *putative exon*, the BS event involves a one or more exons but one or both the BS boundaries do not coincide with the exon boundaries and are mapped within the exon; (4) *monoexon*, only one exon involved and (5) *multiexon*, multiple exons are involved.

Table 1
List of databases used for the circRNA annotation

Database	Main features available in Docker4Circ
circBase	circRNA genomic information and classification and expression in public data
CSCD2	circRNA expression in normal and cancer tissues
exoRBase	circRNA expression in blood exosomal samples
circ2Disease	Information on circRNA expression in specific disease phenotype, prediction of miRNA and RNA binding protein interaction
CircFunBase	Result of circRNA expression in public differential expression analyses
TSCD	circRNA expression in adult and fetal tissues, prediction of miRNA and RNA binding protein interaction

The second column reports the main information stored and used by Docker4Circ

3.4 circRNA Annotation

Accumulating data of circRNA expression in cell lines models or primary tissues are becoming available thanks manually curated databases. With Docker4Circ, using the function *circrnaAnnotations* it is possible to annotate a list of circRNAs with the data from six databases circBase, TSCD, CSCD v2, ExoRBase, Circ2Disease, and CircFunBase. *circrnaAnnotations* allows the annotation of circRNAs detected in multiple versions of the human (hg18, hg19, hg38) and mouse genome (mm9, mm10). The circRNA genomic coordinates can be converted between different genome versions using the *pyliftover* Python package. The required parameters of the function are *circrna.file*, the list of predicted circRNAs; *assembly*, the reference genome assembly; *annotation. sources*, a list of databases id to analyze. Compatible databases: circbase, cscd, exorbase, circ2disease, circfunbase, and tscd. Table 1 reports the main features that can be retrieved from these databases using Docker4Circ.

The output of the function consists of multiple tab-delimited files named with the name of the database and the suffix *anno* as extension.

3.5 Analysis of the circRNA Sequence

Alternative internal splicing events occurring within the circRNA structure can be predicted using Docker4Circ function *ciri_as*. This function applies the CIRI-AS algorithm [4] using as input the list of circRNAs detected by CIRI2 and the alignment files from BWA. *ciri_as* also provides a file named *structure_AS.list* reporting the list of predicted alternative splicing events and associated dPSI. The mandatory parameters required are: *sam.file*, the path to the RNA-Seq alignment SAM/BAM file from BWA; *ciri.file*, the path to the list of circRNAs predicted by CIRI2; *genome.file*, the path to the Fasta file of the reference genomic sequence used for the BWA alignment; *annotation.file*, the path to the GTF/GFF file reporting the reference gene annotations.

The reconstruction of the sequence of a BS event is mandatory to design the primers for qRT-PCR validation. Then, Docker4Circ includes the function *circrnaBSJunctions* that starting from a list of circRNAs and a reference genome assembly is able to reconstruct the BS sequence considering ±35 bp regions centered on the BS position. The mandatory parameters of this function are: *circrna.data*, the path to the list of circRNAs; *exon.data*, the path to an Ensembl exon annotation file that can be obtained using the function *circrnaPrepareFiles*; *assembly*, the name of the reference genome assembly. The output of the function is a fasta file reporting for each circRNA the reconstructed BS sequence.

3.6 Direct circRNA Quantification in RNA-Seq Datasets

Given a list of circRNAs, Docker4Circ allows for the quantification of their expression directly from the reads of RNA-Seq experiments. For this purpose, *circrnaQuantification* was designed applying the HashCirc algorithm [17] using as input the RNA-Seq reads and the fastq of the reconstructed BS junctions. HashCirc applies a two-step approach composed of an initial rapid read filtering based the hash table and a second step focused on the direct alignment between the BS sequences and the surviving read from the first step. The mandatory parameters are *rnaseq.data*, the path to the RNA-Seq fastq file; *backsplicing_junctions.data*, the path to the fasta file storing the circRNA BS sequences; and *hc.params*, vector of six numeric parameters (the k-mer size, the thread number, the dimension of the hash table, the dimension of the collision list; the number of k-mers that must be matched between the reads and a BS sequence, the number of perfect matches to consider). The output of the function is a tab-delimited files reporting the identifier of the circRNA and the number of reads aligned to its BS sequence.

3.7 Differential Expression Analysis

The comparison of the circRNA expression level between two or more experimental conditions can be performed using the functions *mergeData* and *wrapperDeseq2*.

The *mergeData* function is able to merge different files characterized by the same extension. Given a folder and a vector of sample name, the function will iterate over each sample folder joining the files characterized by the indicated extension. A covariate vector can be also provided to group each sample into a specific class. The output of this function is table in which each column represents the level of expression of the circRNAs obtained in each experiment. The name of each column reports also the name of the class in which each sample belongs to.

The results of the mergeData function can be used as input of the *wrapperDeseq2* function which applies a differential expression analysis based on the DESeq2 algorithm [18]. The function requires the following parameters: *output.folder*, the path to the folder where the tables generated by *mergeData* are located and were results will be placed; *experiment.table*, the path to the counts table generated with *mergeData* with addition of covariates; *ref. covar*, covariate to be used as reference; *batch*, boolean value indicating whether the batch effect should be considered in the analysis. Additional parameters include *log2fc*, selected absolute log2fc threshold for differentially expressed genes (default = 1); and *fdr*, False Discovery Rate (FDR) threshold selected for defining the differentially expressed genes (default = 0.05). The output of the function is the full table of differentially expressed circRNAs (prefix *DEfull*), the filtered table of differentially expressed circRNAs (prefix *DEfiltered*) based on author selected thresholds and the normalized counts table (prefix *normalized*).

3.8 Application of Docker4Circ

We applied Docker4Circ on **TNBC_circ** dataset. On average of 27,839 circRNAs have been predicted by CIRI2 (https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_CIRI2_circRNA_predictions.xlsx). The application of *circrnaMergePredictions* function on the circRNAs predicted by CIRI2 generate a list of 5,583 circRNAs (https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_merged_CIRI2_predictions.xlsx). Conversely, the prediction analysis with *STARChip* resulted in 14,118 circRNAs. The predicted circRNAs can be retrieved from https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_STARChip_circRNA_predictions.xlsx. A final set of 4,500 circRNAs predicted by both CIRI2 and STARChip algorithms was considered for further analyses (https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_Overlapped_circRNAs_list.xlsx).

The circRNAs reported in the final list were classified as multiexon (85.3%), followed by monoexon (5.08%), putative exons (4.62%), intronic (4.21%), and intergenic (0.80%). The table reporting the circRNA classification can be retrieved at https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_circRNAs_classification.xlsx, the circRNA annotations are available at https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_circRNAs_annotation.xlsx, and the fasta file reporting the BS sequences at https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/circRNA_backsplicing_sequences.fasta.

The sequence analysis reveals 1,039 splicing events reported in https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_CIRI-AS_results.xlsx.

The differential expression analysis comparing the expression of circRNAs between normal and cancer tissues evidenced 54 circRNAs significantly deregulated. The result of this analysis can be retrieved from https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_DE_analysis_results.xlsx.

Moreover, we also performed an expression analysis using the **BC_circ** dataset. We directly quantified the expression of the 4,500 circRNAs using the *circrnaQuantification* function. In total, 3,471 were detected in this dataset and the differential expression analysis between triple-negative breast tumors and normal breast organoids reveals 312 deregulated circRNAs (adjusted p-value <0.05 and base mean >2). Among them, four circRNAs were detected as differentially expressed also in the **TNBC_circ** datasets and they showed the same expression trend in the two datasets.

The results of this analysis can be retrieved from https://github.com/cursecatcher/biodocker/blob/master/docker4circ/example/Table_analysis_BC_circ_dataset.xlsx.

References

1. Salzman J (2016) Circular RNA expression: its potential regulation and function. Trends Genet 32:309–316
2. Szabo L, Salzman J (2016) Detecting circular RNAs: bioinformatic and experimental challenges. Nat Rev Genet 17:679–692
3. Hansen TB (2018) Improved circRNA identification by combining prediction algorithms. Front Cell Dev Biol 6:20
4. Gao Y, Wang J, Zheng Y et al (2016) Comprehensive identification of internal structure and alternative splicing events in circular RNAs. Nat Commun 7:12060
5. Cheng J, Metge F, Dieterich C (2016) Specific identification and quantification of circular RNAs from sequencing data. Bioinformatics 32:1094–1096
6. Sun P, Li G (2019) CircCode: a powerful tool for identifying circRNA coding ability. Front Genet 10:981
7. Feng J, Xiang Y, Xia S et al (2018) CircView: a visualization and exploration tool for circular RNAs. Brief Bioinform 19:1310–1316
8. Gaffo E, Bonizzato A, Kronnie GT, Bortoluzzi S (2017) CirComPara: a multi-method comparative bioinformatics pipeline to detect and study circRNAs from RNA-seq data. Noncoding RNA 3. https://doi.org/10.3390/ncrna3010008
9. Jakobi T, Uvarovskii A, Dieterich C (2019) circtools-a one-stop software solution for circular RNA research. Bioinformatics 35:2326–2328
10. Humphreys DT, Fossat N, Demuth M et al (2019) Ularcirc: visualization and enhanced analysis of circular RNAs via back and canonical forward splicing. Nucleic Acids Res 47: e123–e123
11. Ferrero G, Licheri N, Tarrero LC et al (2019) Docker4Circ: a framework for the reproducible characterization of circRNAs from RNA-seq data. Int J Mol Sci 21:293
12. Kulkarni N, Alessandrì L, Panero R et al (2018) Reproducible bioinformatics project: a community for reproducible bioinformatics analysis pipelines. BMC Bioinformatics 19:349
13. Beccuti M, Cordero F, Arigoni M et al (2018) SeqBox: RNAseq/ChIPseq reproducible analysis on a consumer game computer. Bioinformatics 34:871–872
14. Zeng K, He B, Yang BB et al (2018) The pro-metastasis effect of circANKS1B in breast cancer. Mol Cancer 17:160
15. Gao Y, Zhang J, Zhao F (2018) Circular RNA identification based on multiple seed matching. Brief Bioinform 19:803–810
16. Akers NK, Schadt EE, Losic B (2018) STAR chimeric post for rapid detection of circular RNA and fusion transcripts. Bioinformatics 34:2364–2370
17. Coscujuela Tarrero L, Ferrero G, Miano V et al (2018) Luminal breast cancer-specific circular RNAs uncovered by a novel tool for data analysis. Oncotarget 9:14580–14596
18. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15:550

Chapter 11

Differential Expression Analysis of Long Noncoding RNAs

Qian Li and Xuefeng Wang

Abstract

Long noncoding RNA (lncRNA) expression data have been increasingly used in identifying diagnostic and prognostic biomarkers in clinical studies. Low-expression genes are commonly observed in lncRNA and need to be effectively accommodated in differential expression analysis. In this chapter, we describe a protocol based on existing R packages for lncRNA differential expression analysis, including lncDIFF, ShrinkBayes, DESeq2, edgeR, and zinbwave, and provide an example application in a cancer study. In order to establish guidelines for proper application of these packages, we also compare these tools based on the implemented core algorithms and statistical models. We hope that this chapter will provide readers with a practical guide on the analysis choices in lncRNA differential expression analysis.

Key words Long noncoding RNA, Differential expression analysis, Zero-inflated counts

1 Introduction

Long noncoding RNAs (lncRNAs) have traditionally been understudied in the analysis of RNAseq data because only a small fraction of lncRNAs have been functionally characterized. Further, most downstream analysis (e.g., pathway analyses and gene ontology (GO) analysis) can only be performed on protein-coding genes. However, considerable attention has been directed to this unexplored genomic space in recent years. The transcriptomic research community increasingly recognized the critical regulatory roles of lncRNAs in various pathways, biological processes, and immune-related mechanisms. Yearly lncRNA-related publications in PubMed have increased from 149 in the year 2010 to 5655 in the year 2019. In the most recent GENCODE v32 catalog, there are around 18,000 human lncRNA genes defined. This implies that a large number of lncRNAs can be retrieved from existing RNAseq data collected from previous studies. No alignment step will be needed and one only needs to update gene expression quantification by reprocessing the mapped BAM files, that is, using HTSeq count with the corresponding GENCODE GTF files. Given its essential functions and large volume, lncRNA has emerged as a key strategy for biomarker discovery in cancer research. Although total RNA protocol is preferred in such secondary analysis, a surprisingly large number of lncRNA genes can still be recalled from ploy(A)-based libraries, such as the RNAseq data in The Cancer Genome Atlas (TCGA) project [1].

Alongside the rapidly growing noncoding gene database that will facilitate the large-scale evaluation of lncRNAs, it is important to consider the specific analytical challenges that still need to be tackled. Although most computational tools and statistical methods for analyzing RNA-seq data can be readily applied to lncRNAs, new guidelines and strategies are required to take into account the specific characteristics of lncRNA data (lower read counts and higher variability across genes/samples). To ensure validity and reliability of any downstream analysis like differential expression, we recommend to first filter out low-expression genes with 50th-percentile RPKM (Reads Per Kilobase Million mapped reads) equal to 0 as well as genes with 90th-percentile RPKM less than 0.1. In our experience, around two-thirds of lncRNAs will be excluded after this screening procedure in most analyses. Interestingly, we found that excess zeros or low expression values remained an issue in the prescreened dataset [2], which motivated the use of zero-inflated statistical models for the differential analysis. Depending on the study design and the anticipated use of the biomarkers, one may consider a more aggressive screening scheme such as the removal or downweighting antisense lncRNAs, which were often found poorly quantified by alignment-based quantitative methods [3]. Particular attention should be paid to the data in which the number of expressed long intergenic noncoding RNAs (lincRNA) is disproportionately lower than antisense lncRNAs. In this chapter, we describe a detailed procedure of using existing packages such as DESeq, edgeR, and lncDIFF for the differential expression analysis of lncRNAs.

2 Materials

2.1 Annotation, Alignment and Quantification

lncRNA transcripts abundance can be profiled by RNA sequencing output files and the well-known bioinformatics preprocessing tools or pipelines. Annotation of transcripts or genes can be extracted based on either full transcriptome or lncRNA reference GTF files, available at GENCODE https://www.gencodegenes.org/. We also assume that RNA reads are aligned to the reference genome based on tools such as STAR [4], HISAT2 [5], and bowtie [6]. Commonly used quantification methods for RNA-Seq samples are Kallisto [7], Salmon [8], HTSeq [9], featureCounts [10], and RSEM [11]. Kallisto and Salmon adopt pseudoalignment methods and do not align sequencing reads to the reference genome. These

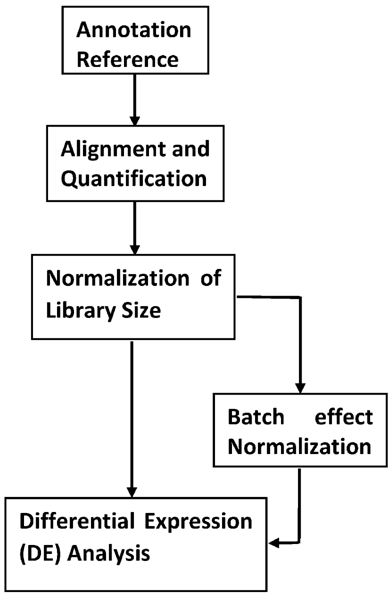

Annotation Reference
Alignment and Quantification
Normalization of Library Size
Batch effect Normalization
Differential Expression (DE) Analysis

Fig. 1 Preprocessing from RNA-Seq fastq files: annotation, alignment, quantification, normalization, and differential expression analysis

tools assign reads to a set of transcripts by the expectation-maximization algorithm. On the other hand, HTSeq, featureCounts and RSEM are alignment-based methods that require aligning and mapping of the sequencing reads to transcriptome or genome. A recent review [3] of lncRNA preprocessing tools suggested that full transcriptome annotation with either pseudoalignment methods or alignment-based method RSEM outperforms the combinations of other tools, generating relatively higher counts at either transcript or gene level. Figure 1 shows a typical procedure of generating lncRNA counts from raw (FASTQ) sequencing files.

2.2 Normalization

Prior to any statistical analyses of lncRNA expression values are performed, it is important to know how the corresponding raw read counts have been processed and normalized. There are multiple methods commonly used in RNA-Seq library size normalization before differential expression (DE) analysis, that is, RPKM, FPKM (Fragments Per Kilobase Million) for pair-end, TMM (Trimmed Mean of *M*-values), and UQ (Upper Quartile) [12]. In addition to library size normalization, there are also approaches removing known or unknown confounding artifacts before DE analysis on the primary factor of interest, for example, ComBat, SVD, and RUV. These methods aim to address unwanted batch effects on transcriptomic data, although researchers can alternatively skip this step and set the batches as covariates in a DE analysis model, as illustrated in Fig. 1.

2.3 An Example lncRNA Counts

In this section, we use an example of normalized lncRNA counts (FPKM) downloaded from a public data portal for cancer studies TANRIC [1] at https://ibl.mdanderson.org/tanric/_design/basic/main.html. To ensure proper detection reliability, a common practice is to filter out lncRNA genes of extremely low average FPKM, for example, <0.3. We recommend using the two-step filter proposed in a previous study [13], which is eliminating the genes with 50th-percentile FPKM as 0, and then keeping the genes with 90th-percentile FPKM over 0.1. FPKM of lncRNA genes passing the two-step filter are available in R package lncDIFF, which can be loaded by running the following R script.

```
# Load data sets from a cancer study
install.packages('lncDIFF')
library(lncDIFF)
data('hnsc.edata')
# 'hnsc.edata' is FPKM for 1000 genes and 40 pairs of 1:1 tumor-
    normal matched tissue samples.
# Preview data
# Normal tissue samples
head(hnsc.edata[,1:3])
#                        Normal 1      Normal 2
    Normal 3
#   ENSG00000005206.12   0.04021539    0.1438404
    0.1142951
#   ENSG00000100181.17   2.16004722    3.2080241
    0.2434748
#  ENSG00000126005.11    5.85834207    14.5885636
    11.2034642
#  ENSG00000130600.11    5.65611346    240.844692
    452.0361821
#             ENSG00000131484.3    0.15085461
    0.2900211      0.2188895
#             ENSG00000142396.6    1.51961695
    2.6278220      2.4907083
    # Tumor tissue samples
head(hnsc.edata[,41:43])
#                        Tumor 1      Tumor 2
    Tumor 3
#     ENSG00000005206.12   0.33760      0.20922
    0.18174
```

```
# ENSG00000100181.17 0.29766 0.45109
  0.05831
# ENSG00000126005.11 10.36660 7.85668
  8.62536
# ENSG00000130600.11 23.82804 167.32558
  14.60800
# ENSG00000131484.3 0.64983 0.58489
  0.16850
# ENSG00000142396.6 1.92266 2.78181
  2.07949
```

3 Methods

3.1 Distribution of lncRNA Counts

In order to understand and choose from algorithms developed for lncRNA analysis, researchers should first look into and interpret the statistical patterns of lncRNA counts. Using the above loaded example of lncRNA FPKM data, gene-wise coefficient of variation (CV, i.e., the ratio of standard deviation over mean $CV = \frac{\sigma}{\mu}$) in Fig. 2 reveals a latent statistical distribution in most low expression lncRNA genes. A large proportion of genes (i.e., circles with log2 (mean)<0 and $-0.5<\log2(CV) < 0.5$ in Fig. 2) show CV values being approximately equal to 1 and not changing along with the mean expression level. This pattern naturally leads to an assumption of Exponential distribution ($\mu = \sigma$, $CV = 1$) for lncRNA low counts (*see* **Note 1**). Meanwhile, the other genes in Fig. 2 display a drop of CV value as mean expression level increases, which coincides with a pattern in Negative Binomial (or Gamma-Poisson mixture) distribution. Therefore, most of lncRNA differential expression analyses are conducted by assuming Exponential or Negative Binomial distribution. The following algorithms and tools were developed based on distinct statistical assumptions and addressed different patterns of lncRNA counts for DE analysis.

3.2 R Packages for lncRNA DE Analysis

3.2.1 lncDIFF

lncDIFF is a recently developed tool designed for low expression genes or transcripts in noncoding RNA, which is implemented in the R package lncDIFF. The core algorithm in lncDIFF adopts the generalized linear model with zero-inflated exponential quasi-likelihood to estimate group effect on low abundance features. This tool is only applicable to data already processed by standard RNA-Seq preprocessing and normalization pipelines. The following examples illustrate the typical DE analysis between tumor and normal tissue samples in a cancer study.

Example #1:

```
library(lncDIFF)
```

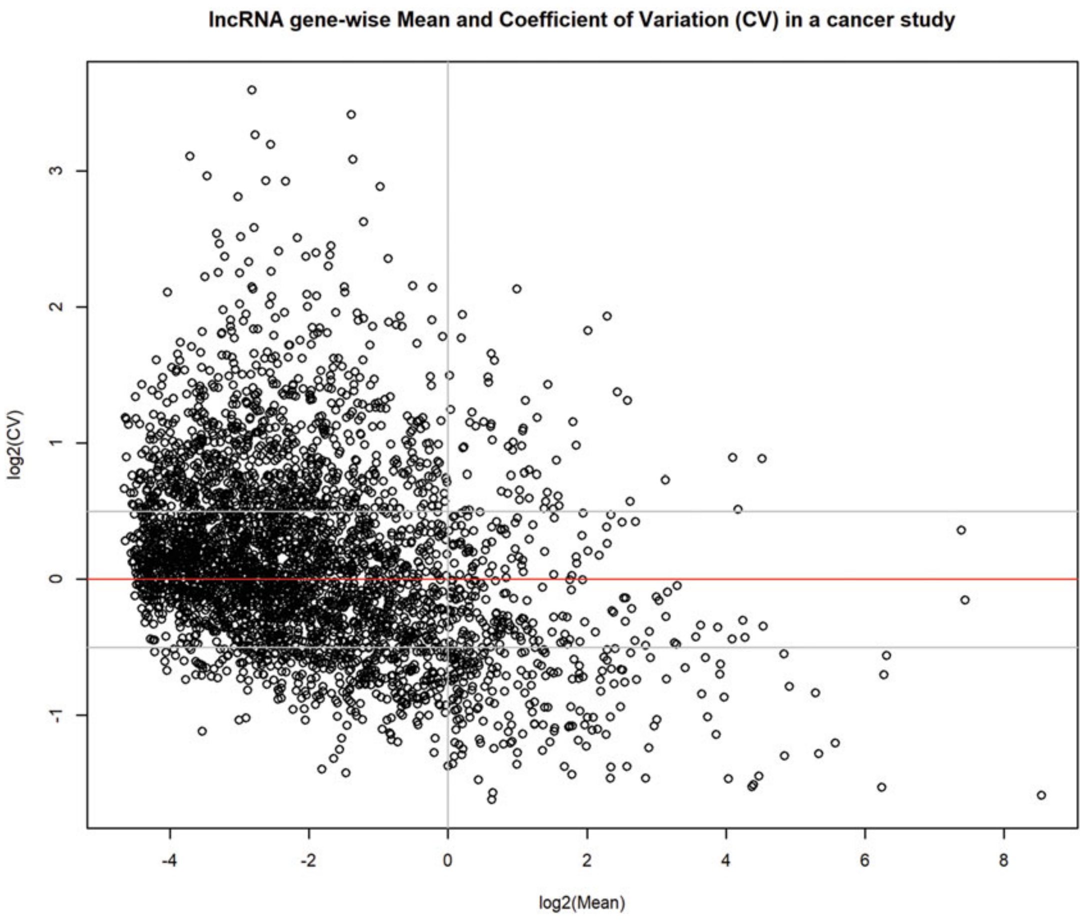


Fig. 2 Relation between gene-wise mean and coefficient of variation in lncRNA counts

```
data('hnsc.edata','tissue','cov')
# 'tissue' is a vector of tissue type for 80 samples, i.e., tumor or
    normal.
# 'cov' is a matrix of covariates, i.e., batch of sequencing.
lncDIFF.result<-
+ lncDIFF(edata=hnsc.edata,group=tissue,covariate=cov)
# DE genes
DE.lncDIFF<-
+ rownames(hnsc.edata )[lncDIFF.result$DE.gene=='Yes']
```

Example #2:

```
# Perform DE analysis on a subset of groups
new.group=paste(tissue, rep(1:2,40),sep='_')
# Sample distribution by new.group
```

```
table(new.group)
# new.group
# normal_1 normal_2 tumor_1 tumor_2
#   20         20            20            20
# DE analysis on normal_1 vs tumor_1 only
lncDIFF.result.subset<-
+ lncDIFF(edata=hnsc.edata,group=new.group,covariate=cov,
+ CompareGroups=c('normal_1', 'tumor_1')
```

ShrinkBayes

ShrinkBayes [14] is a Bayesian approach that utilizes the zero-inflated negative binomial (ZI-NB) model and joint shrinkage of dispersion-related parameters for handling any counts in RNA-Seq DE analysis. This method provides a high detection rate of DE genes, especially for low counts genes. Due to its Bayesian nature, it is computationally demanding compared to other tools and often requires manual parameter tuning to avoid random numerical errors.

Example #3:

```
install.packages(c("devtools","sp","pixmap",           "snowfall",
     "VGAM",  "mclust",  "logcondens",  "Iso","XML","rgl"),
     repos="http://cran.r-project.org")
source("http://www.math.ntnu.no/inla/givemeINLA.R")
library("devtools")
install_github("markvdwiel/ShrinkBayes")
data('hnsc.edata','tissue','cov')
tissue<-as.factor(tissue)
batch<-as.factor(cov)
form = ~ 1 + tissue+batch
# Note: ShrinkBayes requires variables specified in the full
# model as either factor or numeric
ShrinkBayes.result<- ShrinkBayesWrap(hnsc.edata,form,
+ ntag=c(500,1000), paramtotest="tissue")
# Note: if ntag is not specified, computation time may be longer
# DE genes
DE.ShrinkBayes<- ShrinkBayes.result$FDRs$BFDR_tissuetumor
```

3.2.2 edgeR

The tool edgeR [15] examines DE of any transcripts or genes by assuming Negative Binomial (or overdispersed Poisson) distribution to account for multiple factors (i.e., biological and technical). Empirical Bayes method was adopted to control the degree of overdispersion across features, leading to robust and reliable estimation. The input data for the edgeR package can be positive integers (i.e., raw counts) or decimals (i.e., normalized counts).

Example #4:

```
> data('hnsc.edata','tissue','cov')
> dgList=DGEList(counts= hnsc.edata,
+ genes=rownames(hnsc.edata))
> pdata=as.data.frame(cbind(tissue,cov))
# Note: groups and covariates must be a data frame, not a matrix
> design=model.matrix(~tissue+batch,data =pdata)
> dgList <- estimateGLMCommonDisp(dgList, design=design)
> dgList <- estimateGLMTrendedDisp(dgList, design=design)
> dgList <- estimateGLMTagwiseDisp(dgList, design=design)
> fit <- glmFit(dgList, design)
> edgeR.results <- topTags(glmLRT(fit, coef = 2), n = nrow
  (data))
# DE genes
> DE.edgeR<- rownames(hnsc.edata)[edgeR.results$tables
  $FDR<0.05
```

3.2.3 DESeq2

DESeq2 [16] is one of the most commonly used differential analysis tools for nonzero RNA-Seq data, which provides stable estimates by using shrinkage estimation for dispersions and fold changes. As noted by the developer, this package analyzes and interprets the differentially expressed features quantitatively rather than the mere detection of DE features as a binary outcome. However, the input counts data for DESeq2 must be a nonnegative integer value. Hence, FPKM of lncRNAs must be rounded to integer before running DESeq2. In addition to DE analysis, DESeq2 implements multiple library-size normalization methods by estimating size factors, which can be found in the package vignette (*see* **Note 2**).

Example #5:

```
> install.packages("BiocManager")
> BiocManager::install("DESeq2")
> library(DESeq2)
> pdata=cbind(tissue,cov)
> des<-DESeqDataSetFromMatrix(countData =
+ round(hnsc.edata)+1,design = ~tissue+batch, colData = pdata)
> desseq<-DESeq(des,test = 'LRT',full = ~ tissue+batch,reduced
  = + ~ batch)
```

```
> DESeq.result<-results(desseq,independentFiltering=F)
# DE genes
> DE.DESeq<- dataobject@NAMES[DESeq.result$padj<0.05
```

3.2.4 zinbwave+DESeq2

The single-cell RNA-Seq (scRNA-seq) tool zinbwave [17] is another DE tool that must be used with DESeq2. The method accounts for the zero inflation typically observed in scRNA-seq, and thus can also be applied to address the similar issue in lncRNA data. This tool adopts a so-called ZINB-based Wanted Variation Extraction (ZINB-WaVE) approach and requires most samples having counts >4.

Example #6:

```
> install.packages("BiocManager")
> BiocManager::install("DESeq2","zinbwave")
> pdata=cbind(tissue,cov)
> round.data<-as.matrix(round(hnsc.edata,0))
> dataobject<-
+  SummarizedExperiment(round.data,rowData=data.frame(row-
   names(round.data)),
+ colData = pdata)
> zinb <- zinbwave(Y=dataobject,X=~ tissue+batch, K = 0,
+ BPPARAM=SerialParam(), epsilon=1e12)
> dds <- DESeqDataSet(zinb, design=~ tissue+batch)
> dds <- DESeq(dds, test="LRT", reduced=~batch,
+ sfType="poscounts")
> zinbwavedeseq.result=results(dds,independentFiltering=F)
# DE genes
> DE.zinbwavedeseq<- dataobject@NAMES[zinbwavedeseq.
   result$padj<0.05]
```

3.3 Technical Classification of DE Methods and Tools

In this section, we summarize and classify methods implemented in the aforementioned DE analysis tools based on statistical assumptions in the generalized linear model (GLM) (*see* **Notes 3** and **4**). Some readers may skip this section if they only want to learn an application overview. The algorithms are grouped as zero-inflated (ZI)-exponential (i.e., lncDIFF), negative binomial (i.e., edgeR and DESeq2), and ZI-NB (i.e., zinbwave and ShrinkBayes). Here and below, we denote Υ_{ij} the lncRNA library-size normalized counts for gene i in sample j, belonging to each phenotype or treatment group k, $k = 1, \ldots, K$.

GLM ZI-Exponential

The ZI-Exponential density function models counts Υ_{ij} as

$$f\left(\Upsilon_{ij}\right) = (1 - \pi_i)\delta_0\left(\Upsilon_{ij}\right) + \pi_i F_{\text{Exp}}$$

Here, δ_0 is the Dirac function and F_{Exp} is the density function of Exponential distribution. The expected nonzero abundance per gene is λ_{ij}. Let w_{jk} and β_{ik} be design matrix elements and coefficients for groups in DE analysis, and v_{jm} and γ_m ($m = 1, \ldots, M$) being the covariates and corresponding coefficients. The parameters of interest in DE analysis are β_{ik} and can be linked to λ_{ij} as

Identity link: $\lambda_{ij} = \sum_{k=1}^{K} \beta_{ik} w_{jk} + \sum_{m=1}^{M} \gamma_m v_{jm}$

Logarithmic link: $\log\left(\lambda_{ij}\right) = \sum_{k=1}^{K} \beta_{ik} w_{jk} + \sum_{m=1}^{M} \gamma_m v_{jm}$

The likelihood function based on logarithmic link is

$$L^*(\pi_i, \beta_i, \gamma) = \sum_{j=1}^{N} l_j^*(\pi_i, \beta_i, \gamma)$$

$$l_j^*(\pi_i, \beta_i, \gamma) = I_{\left(\Upsilon_{ij}=0\right)} \log(1 - \pi_i) + I_{\left(\Upsilon_{ij}>0\right)} (2 \cdot \log(\pi_i) - \frac{\pi_i \Upsilon_{ij}}{\sum_{k=1}^{K} \beta_{ik} w_{jk}} - \log\left(\sum_{k=1}^{K} \beta_{ik} w_{jk} + \sum_{m=1}^{M} \gamma_m v_{jm}\right)$$

The exponential likelihood estimate for mean gene expression is the maximizer of $L(\beta_i, \gamma)$, that is, $\left(\widehat{\beta}_i, \widehat{\gamma}\right) = argmax\ L(\beta_i, \gamma)$. lncDIFF employs a likelihood ratio test (LRT) based on the ZI-Exponential distribution, in which the test statistic asymptotically follows χ^2 distribution. The p-values from LRT are adjusted for multiple testing using the Benjamini and Hochberg procedure for false discovery rate [18].

3.3.1 GLM with Negative Binomial

R packages edgeR and DESeq2 assume most RNA-Seq raw counts follow negative binomial (NB) distribution, that is $\Upsilon_{ij} \sim \text{NB}(M_j p_{ij}, \varphi_i)$. Here M_j is the library size, φ_i is the dispersion parameter and p_{ik} is the relative abundance of gene i in group k. Similar to lncDIFF, the parameter of interest β_{ik} in DE analysis is linked to p_{ij} by a logarithmic function as [16]

$$\log\left(p_{ij}\right) = \sum_{k=1}^{K} \beta_{ik} w_{jk} + \sum_{m=1}^{M} \gamma_m v_{jm}.$$

The likelihood function for model fitting and estimation is derived from NB density function [17, 19] and the above logarithmic link function. edgeR uses a conditional likelihood weighted on gene-wise total counts to estimate all parameters with an empirical Bayes procedure to shrink the dispersion. Based on estimated parameters, edgeR detected DE genes by Fisher's exact test adapted for over-dispersion [20]. On the other hand, DESeq2 first estimates and normalizes library size factors by various methods, and then estimates other abundance-related parameters by maximum likelihood. Empirical Bayes shrinkage for the dispersion parameter is also adopted in DESeq2 to account for gene-wise dependence of dispersion on mean expression. These tools are ideal for analyzing genes with intermediate or high counts in RNA-Seq, although low count data are also allowed.

3.3.2 GLM with ZI-Negative Binomial

In order to address an excessive proportion of zeros in RNA-Seq counts, ShrinkBayes and zinbwave consider ZI-NB distribution with point mass probability at zero counts. The ZI-NB density function for counts Υ_{ij} is

$$f\left(\Upsilon_{ij}\right) = (1-\pi_i)\delta_0\left(\Upsilon_{ij}\right) + \pi_i F_{\mathrm{NB}}$$

4 Notes

1. It is worthwhile to note that the distribution of lncRNA low counts may deviate from an Exponential distribution in some applications, which does not significantly impact the performance of lncDIFF on DE gene detection. The reason is that lncDIFF is a pseudo or quasi-likelihood [20] approach rather than a "true" likelihood method for lncRNA DE analysis. To illustrate the parameter estimation performance of lncDIFF, we generated lncRNA counts for a single gene in two or three biological groups (i.e., groups A, B, C) by model-based sampling from ZI-Exponential and ZI-NB distributions, respectively, with 1000 replicates for each [2]. The average and median of group effect estimation is approximately equal to the true value of group effect, indicating that the presence of ZINB low counts did not change the estimation power of group effect.
2. In order to confirm whether choice of library-size normalization method has an impact on the performance of DE features detection, we have applied lncDIFF DE analysis to different types of normalized counts, that is, FPKM, TMM, and UQ, using low abundance mRNA in a large-scale cancer study ($n = 546$) [2]. The Pearson correlation of log10 adjusted p-values between the three normalization methods were FPKM vs. TMM 0.82, FPKM vs. UQ 0.92, TMM vs. UQ 0.96, implying similar DE analysis results.
3. Computation time of different tools were also compared and illustrated in [2], showing that lncDIFF, DESeq2 and edgeR were the fastest in computation and zinbwave was relatively slower. ShrinkBayes was powerful with INLA [14]

Table 1
Compare multiple R packages for lncRNA differential expression analysis

R packages	Zero inflation	Noninteger counts	Computation efficiency	Detection power (counts level)
lncDIFF	Yes	Yes	High	High (low counts)
zinbwave + DESeq2	Yes	No	Median	High (high counts)
DESeq2	No	No	High	High (nonzero high counts)
ShrinkBayes	Yes	Yes	Low	High (any counts)
edgeR	No	Yes	High	Median (any counts)

incorporated, however, it required much more computation time and space. Running ShrinkBayes might occasionally receive computational errors, which can be addressed by tuning parameters.

4. A summary of the aforementioned DE analysis tools is shown in Table 1. DESeq2 and zinbwave are popular and powerful tools for median or high counts features DE analysis enabling estimation of dispersion parameters, while lncDIFF has better performance in detecting low expression DE features compared to other existing tools. ShrinkBayes also effectively accounts for the low counts in lncRNA data with detection power close to lncDIFF; however, the computation time of ShrinkBayes is less appealing, especially for large samples. As a best practice, we recommend to use lncDIFF for low expression and limited dispersion (e.g., genes with FPKM mean < 2 and CV < 1.5), and to use zinbwave+DESeq2 for intermediate/high expression or overdispersion (e.g., genes with FPKM mean > 2 or CV > 1.5) for lncRNA DE analysis.

Acknowledgments

We thank Denise Kalos for providing helpful comments and editing for this chapter. This work has been supported in part by the Biostatistics and Bioinformatics Core at the H. Lee Moffitt Cancer Center & Research Institute, a comprehensive cancer center designated by the National Cancer Institute and funded in part by Moffitt's Cancer Center Support Grant (P30-CA076292). This work was also supported in part by the Environmental Determinants of Diabetes in the Young (TEDDY) study, funded by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK).

References

1. Li J, Han L, Roebuck P, Diao L, Liu L, Yuan Y, Weinstein JN, Liang H (2015) TANRIC: an interactive open platform to explore the function of lncRNAs in cancer. Cancer Research 75 (18):3728–3737. https://doi.org/10.1158/0008-5472.can-15-0273
2. Li Q, Yu X, Chaudhary R, Slebos RJ, Chung CH, Wang X (2018) lncDIFF: a novel distribution-free method for differential expression analysis of long non-coding RNA. bioRxiv. https://doi.org/10.1101/420562
3. Zheng H, Brennan K, Hernaez M, Gevaert O (2019) Benchmark of long non-coding RNA quantification for RNA sequencing of cancer samples. Gigascience 8(12):giz145. https://doi.org/10.1093/gigascience/giz145
4. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29(1):15–21. https://doi.org/10.1093/bioinformatics/bts635
5. Kim D, Langmead B, Salzberg SL (2015) HISAT: a fast spliced aligner with low memory requirements. Nat Methods 12(4):357–360. https://doi.org/10.1038/nmeth.3317
6. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9 (4):357–359. https://doi.org/10.1038/nmeth.1923
7. Bray NL, Pimentel H, Melsted P, Pachter L (2016) Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol 34 (5):525–527. https://doi.org/10.1038/nbt.3519
8. Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C (2017) Salmon provides fast and bias-aware quantification of transcript expression. Nat Methods 14(4):417–419. https://doi.org/10.1038/nmeth.4197
9. Anders S, Pyl PT, Huber W (2015) HTSeq—a Python framework to work with high-throughput sequencing data. Bioinformatics 31(2):166–169. https://doi.org/10.1093/bioinformatics/btu638
10. Liao Y, Smyth GK, Shi W (2013) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics 30(7):923–930. https://doi.org/10.1093/bioinformatics/btt656
11. Li B, Dewey CN (2011) RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics 12(1):323. https://doi.org/10.1186/1471-2105-12-323
12. Abbas-Aghababazadeh F, Li Q, Fridley BL (2018) Comparison of normalization approaches for gene expression studies completed with high-throughput sequencing. PLoS One 13(10):e0206312. https://doi.org/10.1371/journal.pone.0206312
13. Yan X, Hu Z, Feng Y, Hu X, Yuan J, Zhao SD, Zhang Y, Yang L, Shan W, He Q (2015) Comprehensive genomic characterization of long non-coding RNAs across human cancers. Cancer Cell 28(4):529–540
14. van de Wiel MA, Neerincx M, Buffart TE, Sie D, Verheul HMW (2014) ShrinkBayes: a versatile R-package for analysis of count-based sequencing data in complex study designs. BMC Bioinformatics 15(1):116. https://doi.org/10.1186/1471-2105-15-116
15. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26 (1):139–140. https://doi.org/10.1093/bioinformatics/btp616
16. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology 15(12):550. https://doi.org/10.1186/s13059-014-0550-8
17. Risso D, Perraudeau F, Gribkova S, Dudoit S, Vert J-P (2018) A general and flexible method for signal extraction from single-cell RNA-seq data. Nat Commun 9(1):284. https://doi.org/10.1038/s41467-017-02554-5
18. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc B 57:289–300
19. Li Q, Noel-MacDonnell JR, Koestler DC, Goode EL, Fridley BL (2018) Subject level clustering using a negative binomial model for small transcriptomic studies. BMC Bioinformatics 19(1):474. https://doi.org/10.1186/s12859-018-2556-9
20. Robinson MD, Smyth GK (2008) Small-sample estimation of negative binomial dispersion, with applications to SAGE data. Biostatistics 9(2):321–332. https://doi.org/10.1093/biostatistics/kxm030

Chapter 12

Micro-RNA Quantification, Target Gene Identification, and Pathway Analysis

Gabriele Sales and Enrica Calura

Abstract

RNA sequencing has become a powerful tool for profiling the expression level of small RNAs from both solid tissues and liquid biopsies. In conjunction with pathway analysis, it offers exciting possibilities for the identification of disease specific biomarkers. In this chapter, we describe a workflow for processing this type of sequencing data. We start by removing technical sequences (adapters) and by performing quality control, a critical task that is necessary to identify possible issues caused by sample preparation and library sequencing. We then describe read alignment and gene-level abundance estimation. Building on these results, we normalize expression profiles and compute differentially expressed microRNAs between sample groups of interest. We conclude by showing how to employ pathway analysis to identify molecular signatures corresponding to biological processes that are significantly altered by the action for microRNAs.

Key words Micro-RNA, Sequencing, Quantification, Target genes, Target predictions, Pathways

1 Introduction

1.1 What Is a microRNA?

MicroRNAs (miRNAs) are fundamental regulatory elements of gene expression in animals and plants. They are 17–24 nucleotide long and regulate eukaryotic gene expression posttranscriptionally. Primary miRNA (pri-miRNA) transcripts with stem–loop regions are usually produced by RNA polymerase II, but occasionally by RNA polymerase III. The stem–loop precursor, pre-miRNA, is released by a cleavage event, which is catalyzed by the nuclear Microprocessor complex that contains the RNase III Drosha and Pasha. Pre-miRNAs are actively exported from the nucleus to the cytoplasm by the Exportin 5. A distinct RNase III, Dicer, subsequently produces 22 base-pair duplex RNA, that is the mature miRNA. In miRNA duplexes, usually the strand with the weakest 5′-end base pairing is selected and loaded into RISC (RNA-Induced Silencing Complex) which contains the Argonaute (Ago) protein. miRNAs use base-pairing to guide RISC to specific messenger RNAs (mRNAs) with fully or partially complementary

Ernesto Picardi (ed.), *RNA Bioinformatics*, Methods in Molecular Biology, vol. 2284, https://doi.org/10.1007/978-1-0716-1307-8_12,
© Springer Science+Business Media, LLC, part of Springer Nature 2021

sequences located especially in 3′ untranslated regions (UTRs). Watson–Crick base-pairing of 2–7 miRNA nucleotides, called “seed,” is crucial for the targeting. The miRNA–mRNA pair enables translational inhibition or exonucleolytic mRNA decay. Unfortunately, the factors that govern the prevalence of one specific mechanism remain unknown.

Each miRNA regulates numerous target genes and a lot of computational algorithms were developed to predict the resulting expression downregulation. The development of algorithms goes hand in hand with the understanding of miRNA mode of action and function [1, 2].

miRBase is the reference database of published miRNA sequences and annotations (http://www.mirbase.org/). At the time of writing, the current release of miRBase (version 22) contains miRNA sequences from 271 organisms, 38 589 hairpin precursors and 48 860 mature miRNAs [3]. These numbers are increasing day by day far exceeding the forecasts [4, 5].

miRNAs are involved in a large number of processes and the deregulation of their expression could lead to dysfunctions and diseases, such as cancer. Multiple aspects of cancer are regulated by miRNAs and they are aberrantly expressed in a high number of different cancers [6]. It has been demonstrated that miRNAs are considered ideal candidates as diagnostic and prognostic markers to distinguish between type of even better than mRNAs. Moreover, they are attractive therapeutic targets because they can be easily overexpressed or inhibited [7]. Despite the general low expression of miRNAs in cancer samples compared to normal tissues miRNAs can act both as oncomirs, causing tumors, or oncosuppressors, protecting against tumors.

miRNA aberrant expression can occur through several mechanisms, such as (1) chromosomal abnormalities like deletions, duplications, and translocations, (2) single nucleotide polymorphisms (SNPs) either in miRNA locus or in binding site for miRNA, or (3) alternative splicing.

1.2 Experimental Study of miRNA Mechanism and Identification of Target Genes

We can divide experimental identification of miRNA targets in direct and indirect methods [8]: (1) direct methods allow the validation of miRNA–mRNA pairs and are often based on the quantification of a reporter construct; (2) indirect methods do not test the binding between miRNA and mRNA but screen all the interactions suggesting a set of candidates. The latter use high-throughput techniques such as immunoprecipitation of Ago complex and the HITS-CLIP approach. The first is based on the isolation of functional Ago-miRNA–mRNA complexes using antibody anti-Ago followed by either microarray or sequencing analysis of the two sets of RNAs. HITS-CLIP uses the UV light to cross-link Ago protein with the associated miRNA and mRNA that will be subsequently identified by sequencing. Among proteome analyses we can find SILAC techniques that use stable light and heavy isotope labeled amino acids in cell culture to distinguish protein level variations after miRNA overexpression or inhibition. Proteins are identified using mass spectrometry and the protein quantity is considered proportional to the protein peak intensity. The transcriptome analyses can be also performed to study the miRNA targets forcing miRNA up- or downregulation with subsequent measure of genome-wide expression changes. A third category of transcriptome analysis is the use of miRNA–mRNA matched measurement on the same biological samples. This technique depicts a more natural situation without experimental artifacts and it is suitable for studies in which a high number of patient samples is required.

1.3 In Silico Identification of miRNA Target Genes

Despite the increase of experimentally validated miRNA targets the majority of them remain unknown. In silico prediction is the only solution to investigate large amount of data rapidly.

Several algorithms have been developed using strategies like the sequence alignment between 3′ regions of genes and the seed sequence of miRNAs; the sequence conservation through species; the target site accessibility and the binding stability.

Some of the most popular target prediction algorithms include DIANA-microT [9], ElMMO [10], miRSVR [11], Pictar [12], PITA [13], RNA22 [14], and TargetScan [15]. All these algorithms differ for the features considered and the strategy adopted to perform predictions. Evaluating and comparing these tools presents several difficulties given the absence of the clear definition of true positives.

Alexiou et al. [16], in one of the most complete reviews, compares the results of each algorithm with data retrieved from miRNA overexpression experiments, and with a collection of experimentally validated targets. The authors concluded that, despite some features are more useful and some programs, like DIANA-MicroT and TargetScan, are more accurate than others, in general all the programs fail to identify most of the targeted genes. The biggest issue is that we do not know the proportion of miRNAs that follow the rules used by the predictors. When we make comparisons with experimental data we can eliminate false-positive predictions, but the total amount of the false negatives remains unknown [8].

1.4 miRNA–Target Dedicated Databases

An increasing number of databases, which collect information about miRNAs and their targets, exists. These databases contain information coming from literature, concerning both in silico and experimental approaches in physiological and disease conditions. miRbase [3] is the most important web resource, especially for nomenclature and sequences. HMDD (Human MicroRNA-associated Disease Database) [17], miR2Disease [18], and PhenomiR [19] are dedicated to miRNAs in diseases. miRGator [20], miRGen [21], and Argonaute [22] contain in silico target gene predictions. miRecords [23] and Tarbase [24] contain validated miRNA targets and information about the experimental validation methodologies.

1.5 miRNAs in Signaling Pathways

Cells express multiple miRNAs at the same time. A single gene can be targeted by different miRNAs and a miRNA can target multiple genes.

Especially for their rapid action and multigene regulatory capacity, miRNAs are the best candidates to play pivotal role in the modulation of the signal transductions in time and space. A signal transduction is a mechanism that converts a signal (stimulus) in a change of behavior of the cell, that is, alteration of metabolism, proliferation or apoptosis, regulation of transcription of genes, cell commitment, and so on.

Although miRNAs have the role to downregulate gene expression, their function is not only repressive. Depending on pathway topology, there are examples of miRNA involvement in both activation and repression of signal transduction [25].

miRNAs also play their role by amplifying or repressing the response, so that the signal can or cannot pass the sensitivity threshold of the system. This explains how the cell is able to perceive quantitatively the signal generating a response tailored to the intensity and the duration of the stimulus [25].

Moreover, the temporal difference to produce miRNAs and proteins (miRNAs are processed faster) allows miRNAs to affect gene expression more rapidly than what is done by transcription factors. In this way miRNAs are fundamental elements of signaling pathway conferring temporal, as well as quantitative precision.

2 Materials

2.1 Deep Sequencing Dataset

As a running example for this chapter we are going to use the dataset described in [26]. In this study, the authors profile miRNA and mRNA expression in nasopharyngeal carcinoma (NPC) and normal nasopharyngeal mucosal specimens. Total RNA from 11 samples was ribo-depleted and sequenced on an Illumina HiSeq 2500 system, producing paired-end 150-bp long reads. A small RNA library was built enriching RNA fragments with lengths from 18 to 30 nucleotides. It was then sequenced using standard protocols on an Illumina HiSeq 4000 instrument.

2.2 Hardware Requirements

The analyses we describe can be completed on commodity hardware. Specifically, the entire workflow will require no more than 16 GB of RAM and 500 GB of free disk space. The availability of multiple processors could be used to reduce the total processing time, but this is not strict requirement.

2.3 Software Requirements

We make use of a number of common utilities for processing RNA-seq datasets, including *sra-tools* to download the raw reads; *cutadapt* to trim adapters; *FastQC* to evaluate the read qualities; and *hisat2*, *featureCounts*, and *gffread* to align the reads and produce gene-level summaries.

Gene and miRNA abundances are then loaded in the R environment, where we rely on a number of packages for data manipulation (*tidyverse*), differential expression (*edgeR*), retrieval of miRNA–target relationships (*multiMiR*), Gene Ontology enrichment (*clusterProfiler*), and pathway analysis (*graphite*). We plot pathway graphs with *Cytoscape*.

3 Methods

3.1 Example Dataset Retrieval

We can access the entire collections of sequences described in [26] at the *Sequence Read Archive*, searching first the project code PRJNA486528 and then looking for the run IDs (they all start with the "SRR" prefix; e.g., SRR7707733). The fasterq-dump command (part of the *sra-tools* [27]) downloads the raw read data in the FASTQ format. We invoke the command using the code of a run.

```
fasterq-dump --progress SRR7707733
```

There is a problem with the approach above, though. We have to patiently wait the completion of the command and then manually issue another to download the next library, 22 times in total. A more efficient approach makes use of shell scripting to automate the entire procedure. To begin with, let us define two variables, one listing the identifiers of all mRNA libraries and the other for miRNAs.

```
    MRNA_RUNS="SRR7707733 SRR7707734 SRR7707737 SRR7707739
SRR7707743
   SRR7707736 SRR7707738 SRR7707741 SRR7707735 SRR7707740
   SRR7707742"
MIRNA_RUNS="SRR7707744 SRR7707746 SRR7707748 SRR7707751
SRR7707754
   SRR7707750 SRR7707745 SRR7707747 SRR7707749 SRR7707752
   SRR7707753"
```

There are two things to note about the expressions above. Each group of IDs has to be written on a single line and enclosed in a pair of double quotes; if you forget them, you get a rather dense error message (for instance, bash: SRR7707733: command not found). The definition of the MRNA RUNS and MIRNA RUNS variables is only temporary: if you close your shell or if you switch to another one already open, they will not be available. In that case, you should just type those two lines again.

Now we can *reference* the variables and execute multiple commands one after the other using a for loop.

```
    for run in $MRNA_RUNS $MIRNA_RUNS; do
    fasterq-dump --progress $run
done
```

At line 1 we introduce a new variable run (notice the singular form) which will hold one SRR identifier at a time. Later (line 2) we use it to compose the appropriate fasterq-dump command.

3.2 Initial Processing

The size of a small RNA transcript such as a miRNA is typically 22 nucleotides. As the reads generated by the sequencer are longer, they will inevitably include the 3′ end adapter introduced during the library preparation. Our first task will thus be that of searching such technical sequences and to remove them (a procedure usually referred to as *trimming*).

The content of the adapter obviously depends on the library preparation kit used in the experiment. You should consult the documentation of the manufacturer to recover the exact sequence. In our example dataset, the authors have used the "NEBNext Multiplex Small RNA Library Prep Set" whose adapter is:

```
AGATCGGAAGAGCACACGTCT
```

cutadapt [28] is an easy to use software that scans FASTQ files and removes adapter instances. The program looks for the provided sequence from the 3′ end of each read and is even capable to identify (and remove) partial matches. For the moment, we are going to use it on a single run to understand the effect of trimming.

```
cutadapt -a AGATCGGAAGAGCACACGTCT \
   -o trimmed.fastq \
   SRR7707744.fastq
```

The file trimmed.fastq generated by the command above contains only the portion of each read that passed the filter. (In case you were wondering, the "\" at the end of lines 1 and 2 allows us to split a long command over multiple lines while still running the entire expression as a single unit.)

At the end of its run, cutadapt prints on the screen some diagnostic information: in particular, the number of reads that were actually trimmed by the software. As expected, the fraction is close to 100% for our sample.

Total reads processed:	13,848,958
Reads with adapters:	13,789,346 (99.6%)

Our read preprocessing is not yet completed. If we scan the file produced by cutadapt, we might notice that some reads look odd. For instance:

```
@SRR7707744.148898 148898 length=49
+
```

In this case, the procedure trimmed the entire read. This usually happens when the sequenced molecule is an adapter dimer or some degraded RNA, rather than a bona fide miRNA. As no usable information is present, we should drop the read altogether.

The -m option instructs cutadapt to discard all reads shorter than a given length (we are going to choose 18 nucleotides in our case).

To complete the trimming configuration, we should consider one more signal: the base-calling quality, also known as the *Phred quality.* For each base in a read the sequencing instrument provides a score Q representing the probability of an erroneous call. Indeed, Q is linked to the error probability P by the following equation:

$$Q = -10\log_{10} P \qquad (1)$$

In a FASTQ file each quality value is compactly encoded by one letter. Below you will find Phred scores in the last line, the one following the "+" header.

```
@SRR7707744.1 1 length=49
GCGGGTGATGCGAACTGGAGTCTGAGC
+
AAAFF7<FJJJJJJJJJJJJFF<FJJJ
```

The character "A" corresponds to $Q = 32$, or an error probability of $P \sim 0.06\%$; "#", on the other hand, would represents $Q = 2$ and $P \sim 63\%$. Large error probabilities might reduce our trust in the base reported by the instrument. Luckily cutadapt can trim low-quality bases from the 3′ end of our reads when we provide (-q option) a minimum acceptable score.

When we take all the steps above together, we obtain the following command.

```
   for run in $MIRNA_RUNS; do
         cutadapt -a AGATCGGAAGAGCACACGTCT \
         -m 18 \
         -q 24 \
         -o $run.trimmed.fastq \
         $run.fastq
done
```

The diagnostic output informs us on the number of reads that have been removed by the different filters we have configured.

```
   Total reads processed:    13,848,958
  Reads with adapters: 13,787,544 (99.6%)
  Reads that were too short: 165 (0.0%)
  Reads written (passing filters): 13,848,793 (100.0%)
  Total basepairs processed:    678,598,942 bp
  Quality-trimmed: 1,459,916 bp (0.2%)
  Total written (filtered): 309,390,389 bp (45.6%)
```

3.3 Quality Control

Before we proceed further, we should take some time to evaluate the quality of the reads we have selected and transformed with trimming. Specifically, we will consider two metrics:

- The posttrimming length of each read.
- The calling quality of remaining bases.

The FastQC [29] program computes both of those. The tool can be used through an interactive graphical interface, but here we call it from the command-line to generate one report for each sample. To keep our files organized, we store all produced files in a separate directory.

```
   mkdir fastqc
for run in $MIRNA_RUNS; do
     fastqc -o fastqc $run.trimmed.fastq
done
```

FastQC has now created one HTML report for each run. We can open them in any web browser of our liking.

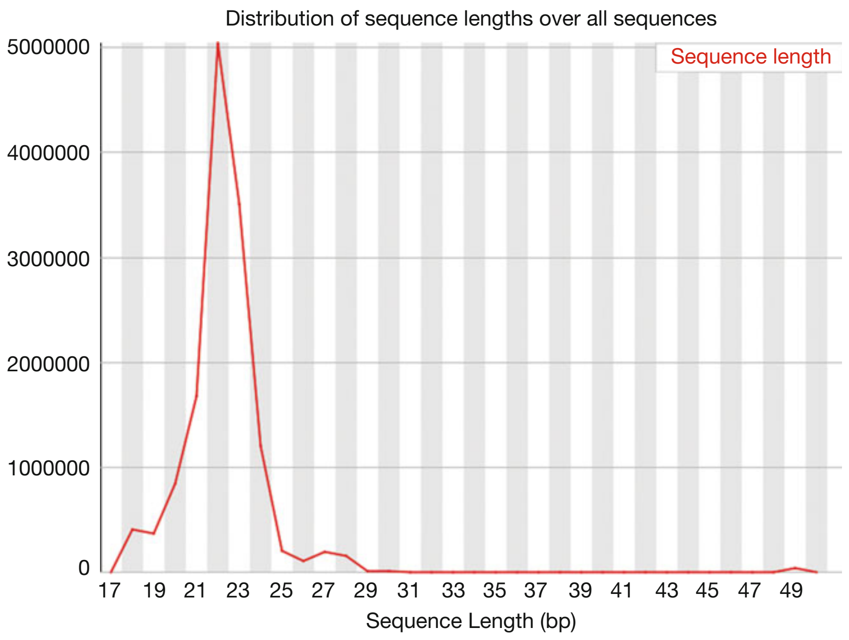


Fig. 1 Read length distribution, after trimming

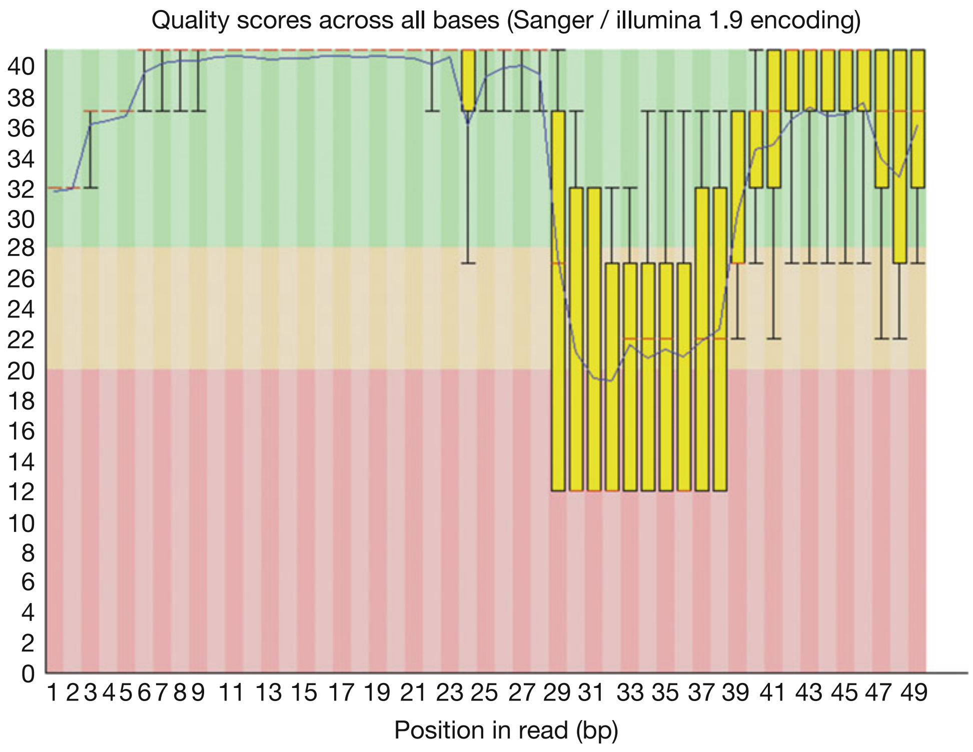


Fig. 2 Quality distribution per base, after trimming

The distribution of read lengths (Fig. 1) prominently displays one peak. It appears in the 21–23 nucleotide range and directly corresponds to processed miRNAs. A low abundance of reads in that range would signal some problem in the sample preparation, such as incorrect small RNA isolation, size selection of fragments or degradation of the RNA.

We now turn our attention to base quality. cutadapt should have discarded all low-quality bases, leaving only those with a high Phred score. The plot of the distribution of such scores over the length of each read (Fig. 2) should confirm that and provides an internal consistency check for our preprocessing procedure.

The plot shows a reduction in base quality after position 27, but this should not be cause for alarm. We have seen that the largest fraction of reads was trimmed *before* that position. The observed quality reduction thus affects only a small minority of all the sequences present in our dataset. Attentive readers might have also noticed that the minimum Phred represented by the boxplots is below $Q = 24$, our selected cutoff.

This is not a cutadapt error: it depends on the details of the trimming algorithm which you can find described in [30].

3.4 Alignment and Quantification

As a preliminary step toward abundance quantification, we should link each read to a putative RNA molecule of origin and match it with the appropriate annotation. We obtain this correspondence by *sequence alignment*, that is, comparing the content of each read with a set of well-known *reference sequences*. Here we have two possibilities: we can limit our search to small RNAs, for instance those listed in domain-specific databases such as miRBase [31]; alternatively, we can rely on the entire genome sequence for our species of interest. Clearly, considering only small RNAs reduces computational costs: the search will be faster and will use less memory. On the other hand, when we align reads against the entire genome we gain the possibility of characterizing novel transcripts that are not yet annotated, a clear advantage—for instance—for nonmodel organisms. In this chapter, we are going to concentrate on the latter strategy: among other advantages, it offers us a uniform approach for processing both mRNA and miRNA libraries.

We start by downloading both genome sequences and gene annotations from the website of the GENCODE project [32].

```
   curl -o genome.fa.gz \
     'ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/
release_33/GRCh38.primary_assembly.genome.fa.gz'
gzip -d genome.fa.gz
```

```
curl -o genes.tf.z \
     'ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/
release_33/gencode.v33.annotation.gtf.gz'
gzip -d genes.gtf.gz
```

Both files are stored in a compressed format, as you can notice by the ".gz" suffix. We invoke the gzip utility to extract the original (uncompressed) content.

hisat2 [33] is a fast and sensitive alignment program for mapping sequencing reads. We will use it first to process mRNA libraries.

```
hisat2-build genome.fa genome

      for run in $MRNA_RUNS; do
        hisat2 -x genome -1 ${run}_1.fastq -2 ${run}_2.fastq |
           samtools view -b -o $run.bam
done
```

The actual procedure consists of two phases. First (line 1) we use the hisat2-build program to create the *genome index*, a data structure representing the entire collection of reference sequences in a form designed to make successive searches extremely fast. We will exploit it for the analysis of mRNA and miRNA libraries; indeed, the genome sequence is exactly the same for both.

Starting at line 3 we run hisat2 in a loop, once for each sample. As messenger RNAs were sequenced with paired-end reads, we provide to the software the two files generated for each library (options "-1" and "-2").

The hisat2 program outputs useful diagnostic information as it progresses. In particular, it prints the *alignment rates*, that is, the fraction of reads that it was able to map onto the genome. Here is an example.

```
   24,797,025 reads; of these:
 24,797,025 (100.00%) were paired; of these:
   3,138,374 (12.66%) aligned concordantly 0 times
   20,441,936 (82.44%) aligned concordantly exactly 1 time
   1,216,715 (4.91%) aligned concordantly >1 times ----
   3,138,374 pairs aligned concordantly 0 times; of these:
    577,274 (18.39%) aligned discordantly 1 time ----
             2,561,100 pairs aligned 0 times concordantly or
discordantly; of these:
    5,122,200 mates make up the pairs; of these:
     3,231,467 (63.09%) aligned 0 times
     1,681,181 (32.82%) aligned exactly 1 time
     209,552 (4.09%) aligned >1 times
93.48% overall alignment rate
```

The specific numbers are obviously going to change for each library, but in general you should check that the fraction of "aligned concordantly exactly 1 time" is above 60%. Smaller values may indicate poor RNA quality or a wrong choice of reference sequences.

Each of our alignments links a pair of reads to some genomic coordinates, but this is not yet sufficient to attribute them to specific genes or transcripts. featureCounts [34] performs this last step of the expression quantification.

```
   for run in $MRNA_RUNS; do
          featureCounts -t exon -g gene_id -p \
      -a genes.gtf -o $run.counts $run.bam
done
```

3.5 MicroRNA Libraries

The estimation of the abundance of miRNAs follows a similar set of steps. First, we run hisat2, taking care of providing as input the trimmed reads. Since miRNA libraries are single-end, we use the "-U" option.

```
   for run in $MIRNA_RUNS; do hisat2 -x genome -U ${run}.
trimmed.fastq | samtools view -b -o $run.bam
done
```

miRBase maintains an up-to-date collection of all miRNA annotations, which we fetch from the project website.

```
curl -o mirnas.gff \
    'ftp://mirbase.org/pub/mirbase/CURRENT/genomes/hsa.gff3'
gffread -T mirnas.gff -F |
     grep MIMAT >mirnas.gtf
```

The command on line 3 is necessary because of a technical reason: featureCounts requires a GTF input, while miRBase only offers GFF3 files. The two formats are similar, yet not completely interchangeable. We use the gffread utility [35] to perform the conversion (line 3), followed by grep (line 4) to keep only the records for mature miRNA forms.

The actual quantification command is almost identical to the one we used earlier for mRNAs.

```
    for run in $MIRNA_RUNS; do
         featureCounts -t transcript -g Name \
             -a mirnas.gtf -o $run.counts $run.bam
done
```

3.6 Data Normalization and Differential Expression

From this point on, we will rely on the R programming environment [36] to complete the analysis. One possibility is to directly run the code you will find below from the command line, invoking the interpreter with the “R” command. If you would rather use a graphical interface, we suggest you install RStudio which, among other things, offers the possibility of preparing interactive notebooks to share your analyses and results with other researchers.

Either way, our first need is to load a few packages which will simplify the following data manipulation and statistical testing.

```
    library(tidyverse)
library(edgeR)
```

```
mrna_samples <- tribble(
 ~ run, ~disease,
 "SRR7707733", "NPC",
 "SRR7707734", "NPC",
 "SRR7707735", "NPC",
 "SRR7707736", "NPC",
 "SRR7707737", "NPC",
 "SRR7707738", "NPC",
 "SRR7707739", "NPC",
 "SRR7707740", "Control",
 "SRR7707741", "Control",
 "SRR7707742", "Control",
 "SRR7707743", "Control
)
```

The samples in our study are subdivided into two groups: control samples and nasopharyngeal carcinoma (NPC) samples. We prepare a data.frame to appropriately subdivide them.

To load gene abundances into R, we rely on a loop which iteratively reads information about one sample at a time.

```
mrna_counts <- list()
for (run in mrna_samples$run) {
    tbl <- read_tsv(paste0(run, ".counts"), skip = 1)
    last_col <- tbl[[ncol(tbl)]]
    last_col <- setNames(last_col, tbl[["Geneid"]])
    mrna_counts[[run]] <- last_col
}
mrna_counts <- do.call(cbind, mrna_counts)
```

At line 1, we define an empty list. A for loop (line 2) repeats the same procedure for each sample: we read the entire featureCounts table (line 3); we select the last column only (abundance levels, line 4), which we label with the corresponding gene names (line 5); we finally store the resulting abundance vector back into our list (line 6), under the sample name. The cbind function at line 8 converts this list of vectors into a proper data.frame.

The gene abundances we have gathered cannot be readily used. They are still affected by two main sources of technical variation: sequencing depth and RNA composition. We refer the interested reader to the edgeR [37] manual for a discussion of both (Subheading 3.7, "Normalization"). Here we simply note that they have an impact on the actual values we observe, yet they do not correspond to any meaningful biological signal. Consequently, we want to normalize the raw counts to reduce these undesired effects.

```
    dl <- DGEList(counts = mrna_counts, group = mrna_samples
$disease)
  keep <- filterByExpr(dl)
dl <- dl[keep, , keep.lib.sizes=FALSE]
dl <- calcNormFactors(dl)
```

We create (line 1) an object of type DGEList, which edgeR will use to track raw counts and sample-group associations. The filterByExpr function implements a filter for dropping genes (lines 2–3) whose abundance is so low to be insufficient for a reliable statistical analysis. We finally apply (line 4) the trimmed mean of M-values (TMM) normalization [38].

```
design <- model.matrix(~ mrna_samples$disease)
dl <- estimateDisp(dl, design)
fit <- glmQLFit(dl, design)
qlf <- glmQLFTest(fit, coef = 2)
```

```
mrna_de <- topTags(qlf, n = Inf, p.value = 0.05)
```

To identify differentially expressed genes in the comparison between the two sample groups, we build a model matrix from sample metadata (line 1, code below); we estimate over-dispersion and fit a quasi-likelihood (QL) negative binomial generalized log-linear model to counts (lines 2–3); and finally we perform genewise statistical tests (line 4). To complete the procedure, line 5 extracts the topmost differentially expressed genes, imposing a significance cutoff on the *p*-value of 0.05.

The analysis of miRNAs is quite similar, so we list all the code together.

```
mirna_samples <- tribble(
 ~ run, ~disease,
 "SRR7707744", "NPC",
 "SRR7707745", "NPC",
 "SRR7707746", "NPC",
 "SRR7707747", "NPC",
 "SRR7707748", "NPC",
 "SRR7707749", "NPC",
 "SRR7707750", "NPC",
 "SRR7707751", "Control",
 "SRR7707752", "Control",
 "SRR7707753", "Control",
 "SRR7707754", "Control"
)
mirna_counts <- list()
for (run in mirna_samples$run) {
     tbl <- read_tsv(paste0("mirna_count/", run, ".counts"),
skip = 1)
      mirna_counts[[run]] <- setNames(tbl[[ncol(tbl)]], tbl
[["Geneid"]])
}
mirna_counts <- do.call(cbind, mirna_counts)
       mirna_dge <- DGEList(counts = mirna_counts, group =
mirna_samples$ disease)
  keep <- filterByExpr(mirna_dge)
  mirna_dge <- mirna_dge[keep, , keep.lib.sizes=FALSE]
  mirna_dge <- calcNormFactors(mirna_dge)
design <- model.matrix(~ mirna_samples$disease)
mirna_dge <- estimateDisp(mirna_dge, design)
fit <- glmQLFit(mirna_dge, design)
qlf <- glmQLFTest(fit, coef = 2)
mirna_top <- topTags(qlf, n = Inf, p.value = 0.05)
```

3.7 Identification of miRNA Target Genes

To retrieve miRNA–target relationships, we use the R package *multiMiR*, which is a comprehensive collection of predicted and validated miRNA–target interactions. This software simplifies the search of validated target genes for specific miRNAs. By default, the search is performed for human sequences.

Here we download a table with information about the targets of a specific miRNA, including the type of experimental validation. That can be further filtered, for example to select "Luciferase" assays those providing the most accurate evidence about direct interactions.

```
    library(multiMiR)
mir200Lux <- get_multimir(mirna = 'hsa-miR-200c-3p',
summary = TRUE) %>%
                        pluck("data") %>%
                        as_tibble() %>%
                       filter(grepl("Luciferase", experiment))
```

We can interrogate the software in the opposite direction just as easily: starting from a gene, we retrieve all miRNAs that may influence its expression.

```
   mirnasOfZEB1 <- get_multimir( org = "hsa", target = "ZEB1",
                                                           table =
"predicted", summary = TRUE,
                                                predicted.cutoff =
35, predicted.cutoff.type = "p",
                                                  predicted.site =
"all")
```

To inform the rest of our analysis, we collect into a single data. frame all miRNA–target relationships predicted by the TargetScan software.

```
    mirna_top_names <- rownames(mirna_top)
mirnas_targets <- get_multimir(org = 'hsa', mirna = mirna_-
top_names,
                                 table = 'targetscan', summary
= TRUE)
```

We found that the 53 differentially expressed miRNAs are potentially able to target 2256 genes.

On this list we can do a gene set analysis with the objective of understanding which cellular processes might be controlled by the deregulated miRNAs.

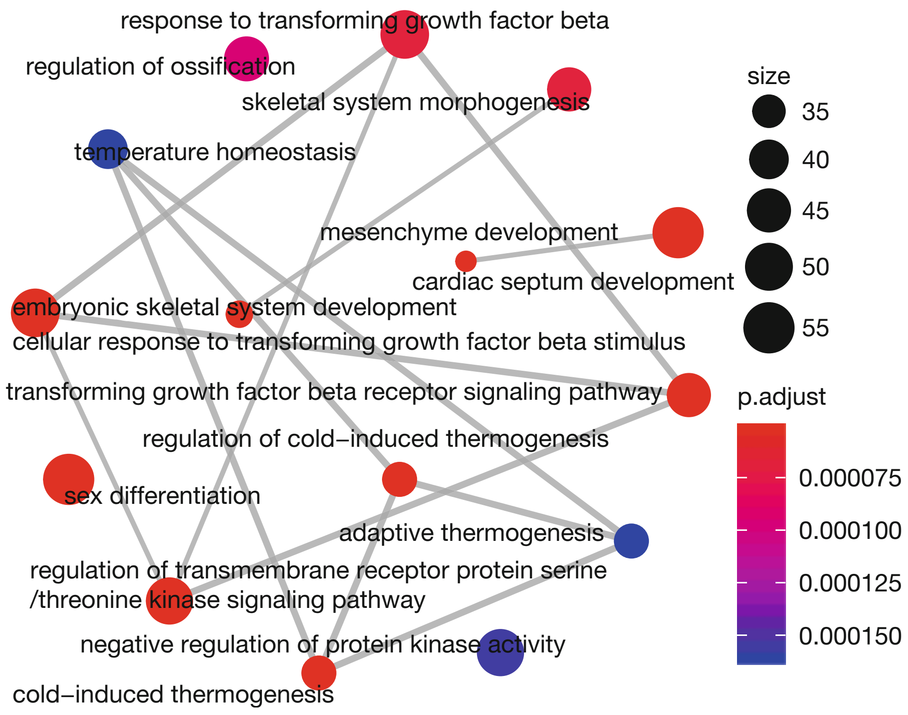


Fig. 3 GO analysis representation

```
    library(clusterProfiler)
targets_GOBP  <-  enrichGO(mirnas_targets@data[,"target_en-
sembl"],
                              OrgDb="org.Hs.eg.db", keyType =
"ENSEMBL", ont="BP",
                           pvalueCutoff = 0.001, qvalueCutoff
= 0.001, minGSSize = 100,
                               maxGSSize = 300)
emapplot(targets_GOBP, showCategory=15)
```

The output of the emapplot function is shown in Fig. 3.

3.8 Correlation Analysis

The dataset we are working on includes matched libraries: that means each sample was sequenced twice, once measuring long RNAs and once for short RNAs. Having this information at our disposal, we can exploit correlations in the expression patterns of genes and miRNAs to identify authentic interactions.

We start by retrieving predictions related to expressed genes and miRNAs.

```
    mrna_exp <- rownames(mrna_counts_norm)
  mirna_exp <- rownames(mirna_counts_norm)
  mirnas_targets <- get_multimir(org = "hsa",
   mirna = mirna_exp, target = mrna_exp, table = targetscan",
   summary=TRUE, predicted.cutoff.type="p",
   predicted.cutoff = 10, use.tibble = TRUE)

      predictions <- as.data.frame(
         mirnas_targets@data[,c("mature_mirna_id","target_en-
sembl")]
  )
  colnames(predictions) <- c("mirna", "gene)
```

Many common statistical analyses to explore multidimensional datasets work best if measurements are homoskedastic, that is, if all samples display the same variance. This assumption is violated by RNA-seq, though: the variance of read counts is expected to increase with the mean. We can correct this with a transformation of common usage: we compute the logarithm of the counts plus 1 (to avoid infinities). The resulting measures can then be used for exploratory data analyses, such as clustering or principal component analysis.

log_gene_counts	<- log2(mrna_counts_norm+1)
log_mirna_counts	<- log2(mirna_counts_norm+1)

We rely on the cor.test() function to compute the correlation among gene and miRNA profiles.

```
    corr_signif <- predictions %>%
                filter(mirna %in% rownames(log_mirna_counts),
                      gene %in% rownames(log_gene_counts)) %>%
    unique() %>%
    pmap(function(mirna, gene) {
        mirna_expr <- log_mirna_counts[mirna, ]
   gene_expr <- log_gene_counts[gene, ]
   res <- cor.test(mirna_expr, gene_expr, method = "pearson",
                             na.action = na.omit)
          list(mirna = mirna, gene = gene, cor = res$estimate,
                 pvalue = res$p.value)
        }) %>%
    transpose() %>%
    simplify_all() %>%
    as_tibble() %>%
    mutate(padj = p.adjust(pvalue, method="BH")) %>%
    filter(abs(cor) >= 0.8, padj < 0.05) %>%
    arrange(-abs(cor))
```

The analysis found 47 miRNA–target pairs with correlated expressions. The Pearson correlation coefficient is used to measure the strength of a linear association between two variables, where the value $r = 1$ means a perfect positive correlation and the value $r = -1$ means a perfect negative correlation.

3.9 Pathway Analysis

The effect of miRNAs on gene expression can be further elucidated by considering their impact on well-known pathways.

Usually, miRNA and gene circuits are identified through the combination of binding predictions and expression correlation. The latter, however, are based on one-to-one relationships among miRNAs and genes. They ignore the biological context of cell signaling in which miRNAs and their targets are characterized by many-to-many relationships and they should be considered as part of a much more complex system of cellular interactions.

Pathway maps might help in this context to understand the multiple miRNA relations, but in practice the number of annotated miRNAs is small. That is why in the following command we shown how to insert miRNAs into pathways to have a picture of the many-to-many relationships occurring in cell circuits.

We will exploit *graphite*, another Bioconductor package designed to retrieve pathway information. First, we need to choose the pathway set of our interest. The pathwayDatabases() function prints out the list of all available species and annotation databases. In our case, we are going to select the KEGG pathways for *Homo sapiens*.

```
    library(graphite)
hsapienskegg <- pathways("hsapiens", "kegg")
pathwayDBensembl <- convertIdentifiers(hsapienskegg, "EN-
SEMBL")
```

In the last line, we convert all genes to ENSEMBL identifiers to match the annotation of our dataset.

We are now going to extend KEGG pathways with miRNA–target interactions, following the *micrographite* pipeline (Calura et al, NAR 2014). Specifically, we will include two types of miRNA–target interactions:

– Interactions validated with reporter assays.
– Interactions predicted in silico and filtered by correlation of expression.

```
  # Predicted interactions
  predicted <- tibble(src_type = "MIR",
```

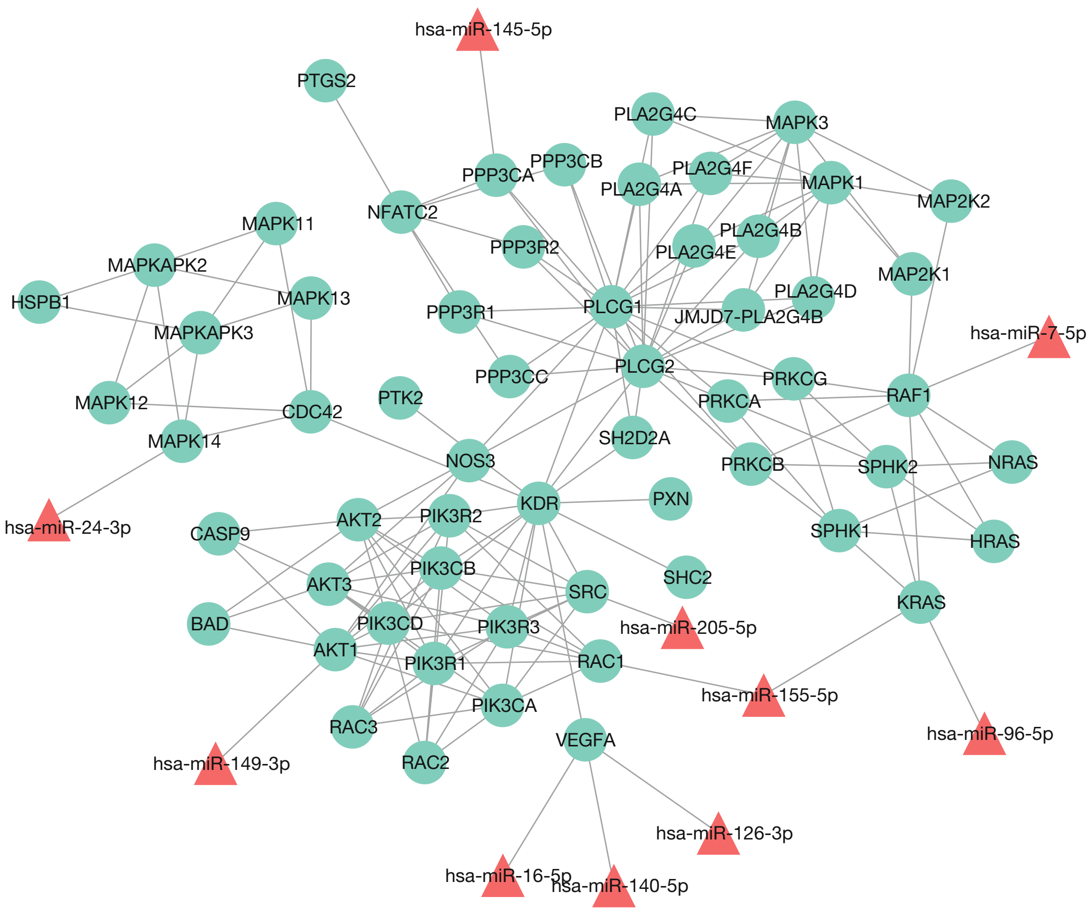


Fig. 4 VEGF signaling pathway extended with miRNAs

```
                                        src = corr_signif$mirna,
                                        dest_type = "ENSEMBL",
                                        dest = corr_signif$gene,
                                        direction = "directed",
                                        type = "MiR_predicted")

  # Validated interactions
  validated <- get_multimir(org = "hsa",
             mirna = mirna_exp, target = mrna_exp, table =
"tarbase", summary = TRUE) %>%
        pluck("data") %>%
        as_tibble() %>%
        filter(grepl("Reporter assay", experiment)) %>%
        transmute(src_type = "MIR",
             src = mature_mirna_id, dest_type = "ENSEMBL",
            dest = target_ensembl, direction = "directed",
            type = "MiR_validated")
mirna_interactions <- rbind(predicted, validated)
```

```
mirpath <- pathwayDBensembl$'VEGF signaling pathway'

   newInteractionsPath <- mirna_interactions %>%
      filter(dest %in% gsub("^(.*):", "", nodes(mirpath))) %>%
      unique()

   newPathwayWithMirs <- buildPathway(
       pathwayId(mirpath),
       pathwayTitle(mirpath),
       pathwaySpecies(mirpath),
       paste0(pathwayDatabase(mirpath), "-micrographite"),
         proteinEdges = rbind(edges(mirpath), newInteraction-
sPath))
```

Once collected the miRNA–target interactions, we can enrich a pathway wiring the miRNAs to the pathway genes, as follows:

If Cytoscape [39] is installed on your system, you can now obtain a graphical representation of the network (*see* Fig. 4) with one command:

```
    cytoscapePlot(convertIdentifiers(mirpath, "symbol"))
```

Once we have the pathway enriched with miRNAs, we can analyze it with many tools of pathway and network analyses. Many of these tools were described in the Chapter dedicated to pathway analyses. Additionally, among them, we can find an extension of the topological pathway analysis *clipper* - extensively presented in the chapter dedicated to pathway analysis—dedicated to the study of miRNAs in pathway called *micrographite* [40]. *Micrographite* is a pipeline to integrate pathway information with predicted and validated miRNA–target interactions and to perform integrated topological analyses of miRNA and gene expression profiles to identify miRNA–gene circuits. *Micrographite* is available as set of R functions. Code and guidelines are available at http://romualdi.bio.unipd.it/micrographite and a complete analysis in [40].

Acknowledgments

This work was supported by the “My First AIRC grant” provided by Italian Association for Cancer Research to Enrica Calura (MFAG 2019, number 23522).

References

1. Bartel DP (2004) MicroRNAs: genomics, biogenesis, mechanism, and function. Cell 116 (2):281–297
2. Bartel DP (2009) MicroRNAs: target recognition and regulatory functions. Cell 136 (2):215–233
3. Kozomara A, Birgaoanu M, Griffiths-Jones S (2019) miRBase: from microRNA sequences to function. Nucleic Acids Res 47(D1): D155–D162
4. Bentwich I, Avniel A, Karov Y, Aharonov R, Gilad S, Barad O, Barzilai A, Einat P, Einav U, Meiri E et al (2005) Identification of hundreds of conserved and nonconserved human microRNAs. Nat Genet 37(7):766–770
5. Berezikov E, Guryev V, van de Belt J, Wienholds E, Plasterk RH, Cuppen E (2005) Phylogenetic shadowing and computational identification of human microrna genes. Cell 120(1):21–24
6. Hayes J, Peruzzi PP, Lawler S (2014) MicroRNAs in cancer: biomarkers, functions and therapy. Trends Mol Med 20(8):460–469
7. Lee YS, Dutta A (2009) Micrornas in cancer. Annu Rev Pathol Mech Dis 4:199–227
8. Ørom UA, Lund AH (2010) Experimental identification of microRNA targets. Gene 451 (1–2):1–5
9. Maragkakis M, Reczko M, Simossis VA, Alexiou P, Papadopoulos GL, Dalamagas T, Giannopoulos G, Goumas G, Koukis E, Kourtis K et al (2009) DIANA-microT web server: elucidating microRNA functions through target prediction. Nucleic Acids Res 37(Suppl 2): W273–W276
10. Gaidatzis D, van Nimwegen E, Hausser J, Zavolan M (2007) Inference of miRNA targets using evolutionary conservation and pathway analysis. BMC Bioinformatics 8(1):69
11. Betel D, Koppal A, Agius P, Sander C, Leslie C (2010) mirsvr predicted target site scoring method: Comprehensive modeling of microRNA targets predicts functional non-conserved and non-canonical sites. Genome Biol 11:R90
12. Lall S, Grün D, Krek A, Chen K, Wang YL, Dewey CN, Sood P, Colombo T, Bray N, MacMenamin P et al (2006) A genome-wide map of conserved microRNA targets in C. elegans. Current biology 16(5):460–471
13. Kertesz M, Iovino N, Unnerstall U, Gaul U, Segal E (2007) The role of site accessibility in microRNA target recognition. Nat Genet 39 (10):1278–1284
14. Miranda KC, Huynh T, Tay Y, Ang YS, Tam WL, Thomson AM, Lim B, Rigoutsos I (2006) A pattern-based method for the identification of microRNA binding sites and their corresponding heteroduplexes. Cell 126 (6):1203–1217
15. Friedman RC, Farh KKH, Burge CB, Bartel DP (2009) Most mammalian mrnas are conserved targets of microRNAs. Genome Res 19 (1):92–105
16. Alexiou P, Maragkakis M, Papadopoulos GL, Reczko M, Hatzigeorgiou AG (2009) Lost in translation: an assessment and perspective for computational microrna target identification. Bioinformatics 25(23):3049–3055
17. Li Y, Qiu C, Tu J, Geng B, Yang J, Jiang T, Cui Q (2014) Hmdd v2. 0: a database for experimentally supported human microRNA and disease associations. Nucleic Acids Res 42(D1): D1070–D1074
18. Jiang Q, Wang Y, Hao Y, Juan L, Teng M, Zhang X, Li M, Wang G, Liu Y (2009) mir2disease: a manually curated database for microRNA deregulation in human disease. Nucleic Acids Res 37(Suppl 1):D98–D104
19. Ruepp A, Kowarsch A, Schmidl D, Buggenthin F, Brauner B, Dunger I, Fobo G, Frishman G, Montrone C, Theis FJ (2010) PhenomiR: a knowledgebase for microRNA expression in diseases and biological processes. Genome Biol 11(1):R6
20. Nam S, Kim B, Shin S, Lee S (2007) miRGator: an integrated system for functional annotation of microRNAs. Nucleic Acids Res 36 (Suppl_1):D159–D164
21. Megraw M, Sethupathy P, Corda B, Hatzigeorgiou AG (2007) miRGen: a database for the study of animal microRNA genomic organization and function. Nucleic Acids Res 35(suppl 1):D149–D155
22. Shahi P, Loukianiouk S, Bohne-Lang A, Kenzelmann M, Küffer S, Maertens S, Eils R, Gröne HJ, Gretz N, Brors B (2006) Argonaute—a database for gene regulation by mammalian microRNAs. Nucleic Acids Res 34 (Suppl 1):D115–D118
23. Xiao F, Zuo Z, Cai G, Kang S, Gao X, Li T (2009) miRecords: an integrated resource for microRNA–target interactions. Nucleic Acids Res 37(Suppl 1):D105–D110
24. Vergoulis T, Vlachos IS, Alexiou P, Georgakilas G, Maragkakis M, Reczko M, Gerangelos S, Koziris N, Dalamagas T, Hatzigeorgiou AG (2012) TarBase 6.0: capturing the exponential growth of miRNA targets

with experimental support. Nucleic Acids Res 40(D1):D222–D229

25. Inui M, Martello G, Piccolo S (2010) Micro-RNA control of signal transduction. Nat Rev Mol Cell Biol 11(4):252–263
26. Lin C, Zong J, Lin W, Wang M, Xu Y, Zhou R, Lin S, Guo Q, Chen H, Ye Y et al (2018) EBV-miR-BART8-3p induces epithelial-mesenchymal transition and promotes metastasis of nasopharyngeal carcinoma cells through activating NFκb and Erk1/2 pathways. J Exp Clin Cancer Res 37(1):283
27. SRA Toolkit Development Team (2019) SRA Tools. https://github.com/ ncbi/sra-tools
28. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet J 17(1):10–12
29. Andrews S et al. (2010) FastQC: a quality control tool for high throughput sequence data
30. Martin M (2019) Cutadapt quality trimming algorithm. https://cutadapt.readthedocs.io/en/stable/algorithms.html#quality-trimming-algorithm
31. Kozomara A, Birgaoanu M, Griffiths-Jones S (2019) Mirbase: from microrna sequences to function. Nucleic Acids Res 47(D1):D155–D162
32. Frankish A, Diekhans M, Ferreira AM, Johnson R, Jungreis I, Loveland J, Mudge JM, Sisu C, Wright J, Armstrong J et al (2019) Gencode reference annotation for the human and mouse genomes. Nucleic Acids Res 47(D1):D766–D773
33. Kim D, Langmead B, Salzberg SL (2015) HISAT: a fast spliced aligner with low memory requirements. Nat Methods 12(4):357–360
34. Liao Y, Smyth GK, Shi W (2014) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics 30(7):923–930
35. Pertea G (2019) GFF/GTF utility providing format conversions, region filtering, FASTA sequence extraction and more. https://github.com/gpertea/gffread
36. R Development Core Team R, et al (2011) R: A language and environment for statistical computing
37. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26 (1):139–140
38. Robinson MD, Oshlack A (2010) A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biol 11 (3):R25
39. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res 13(11):2498–2504
40. Calura E, Martini P, Sales G, Beltrame L, Chiorino G, D'Incalci M, Marchini S, Romualdi C (2014) Wiring miRNAs to pathways: a topological approach to integrate miRNA and mRNA expression profiles. Nucleic Acids Res 42(11):e96–e96

Chapter 13

In Silico Analysis of Micro-RNA Sequencing Data

Ernesto Aparicio-Puerta, Bastian Fromm, Michael Hackenberg, and Marc K. Halushka

Abstract

High-throughput sequencing for micro-RNAs (miRNAs) to obtain expression estimates is a central method of molecular biology. Surprisingly, there are a number of different approaches to converting sequencing output into micro-RNA counts. Each has their own strengths and biases that impact on the final data that can be obtained from a sequencing run. This chapter serves to make the reader aware of the trade-offs one must consider in analyzing small RNA sequencing data. It then compares two methods, miRge2.0 and the sRNAbench and the steps utilized to output data from their tools.

Key words Micro-RNA, Small RNA sequencing, Alignment, Bowtie, MirGeneDB, miRBase, isomiR

1 Introduction

A current area of interest in biological research is the relative expression patterns of RNA species across tissues, cells, organisms, and diseases. RNA expression levels can be characterized by using one of two main sequencing methods. For mRNA and lncRNAs, traditional full-length RNA sequencing (RNA-seq) is utilized. Small RNA sequencing methods are used for micro-RNAs, tRNA halves and fragments, snoRNAs, piRNAs, and other smaller species. Among these smaller RNA species, micro-RNAs have historically been the moiety of most interest, although there has been an increasing interest of tRNA halves and fragments [1, 2].

Micro-RNAs are small regulatory RNAs (~22 bp) known to cause mRNA repression and translational repression [3]. In this role, micro-RNAs are tuners of protein expression whose expression levels change relative to embryologic development, cell maturation, cell stressors, and diseases such as neoplasia [4, 5]. The three main approaches to ascertaining micro-RNA levels in a sample are hybridization arrays, small RNA-seq, and quantitative PCR (qPCR). While each method has its advantages and disadvantages, this chapter will focus on the sequencing approaches to micro-RNAs.

Two of the main advantages of small RNA-seq for micro-RNAs are the ability to capture all micro-RNAs in a single assay and the cost per micro-RNA with highly multiplexed approaches. We and others have found that one million micro-RNA reads for a given sample is sufficient sampling to confidently determine levels of all functionally relevant micro-RNAs [6]. However the amount of reads needed per sample can be much higher if micro-RNAs represent a smaller percentage of total reads, as seen in plasma or with some library preparation kits. With current sequencing systems offering 300 million or more reads per lane, significant multiplexing can greatly reduce costs per sample. Some disadvantages of the approach are a lack of agreed upon normalization approach and methodologic biases (particularly ligation bias) impacting on specific micro-RNAs.

1.1 Unique Features of Micro-RNAs

To properly identify micro-RNAs in sequencing data it is important to understand some basic features of micro-RNAs. Most importantly, micro-RNAs are not a solitary sequence. Rather, due to inexact cleavage events of DROSHA/DICER, 3′ nucleotide additions, rare SNP sites, and rarer A-to-I editing changes, each micro-RNA can be thought of as a family of reads, or isomiRs [7]. In fact, the most abundant read of any given micro-RNA only averages 45% of all reads assigned to that micro-RNA [8]. Even further, although each micro-RNA has an official canonical sequence reported in a micro-RNA repository (MirGeneDB 2.0 or miRBase release 22.1), that sequence may or may not be the most abundant version of the micro-RNA which can vary across cells and tissues [9, 10]. This collection of isomiRs can be thought of as comprising these three groups:

- Sequence variants: the read shows at least one variation compared to the canonical sequence. The variation can be caused by sequencing errors, SNPs, somatic mutations or RNA editing.
- Length variants: The read is longer or shorter that the canonical sequence, but the extending nucleotides correspond to the genomic template, that is, the pri-micro-RNA sequence.
- Nontemplated additions: Several nucleotidyl-transferases can uridylate or adenylate RNA molecules. These changes normally manifest through added nucleotides that do not match the reference sequence and cause mismatches in the alignment.

As a result of this wide diversity, it is important to approach micro-RNA sequence alignment in a thoughtful manner.

1.2 Alignment Options and Considerations for Micro-RNAs

A number of different alignment methods exist for the analysis, discovery and profiling of micro-RNAs. Forty-seven are listed in the useful website https://tools4mirs.org/ under "Sequencing Data Analysis" [11]. Two of these, miRge and sRNAbench, will be described in greater detail below. A number of other groups forgo these alignment tools and create their own one-off methods [12, 13].

A major decision point of alignment software is the library to which the alignment will occur. Some (mostly older and one-off) methods align to the species' entire genome, typical for mRNA methods. Other (mostly newer) methods align to RNA expression libraries. Based on the diversity and complexity of isomiRs, an approach that utilizes tight alignments with one to no mismatches will reduce the number of reads being identified and assigned as micro-RNAs. Allowing for more mismatches will capture more isomiRs, but can increase the number of false positive alignments. However, if aligning to the genome, the search space becomes so large that these relatively short reads (~16–27 bp) are likely to align to multiple locations unrelated to the true micro-RNA. Therefore, we strongly support methods that align to specific RNA sequence libraries and allow for the capture of isomiRs as we describe below. Not only do different search strategies affect the number of micro-RNAs detected, these differences can affect the isomiRs that can be detected. Recent publications suggest that some isomiRs may be functionally relevant [14] and a consortium, miRTOP, has developed a consensus method to report isomiRs consistently using a GFF3 format [15].

Another point of comparison between micro-RNA and mRNA alignment methods is the complexity of the read data. Despite the large number of isomiRs seen for micro-RNAs, a few reads for each micro-RNA tend to capture the majority of the isomiR signal. Extrapolating this across all of the micro-RNAs present, a few hundred to a few thousand reads might sum to >90% of all of the sequenced reads needing to be aligned. This is due to the entire micro-RNA being sequenced in each read. For mRNAs, where the RNA is sheared, multiple different fragments exist for each gene with noncomplementary start and end sites. Thus a few thousand of the most abundant reads would not represent a large percentage of all reads to be aligned. For micro-RNAs, alignment tools can take advantage of this overlap by first combining all identical reads before sending each read through an alignment step (Fig. 1). This can be a considerable time saver for processing small RNA-seq datasets.

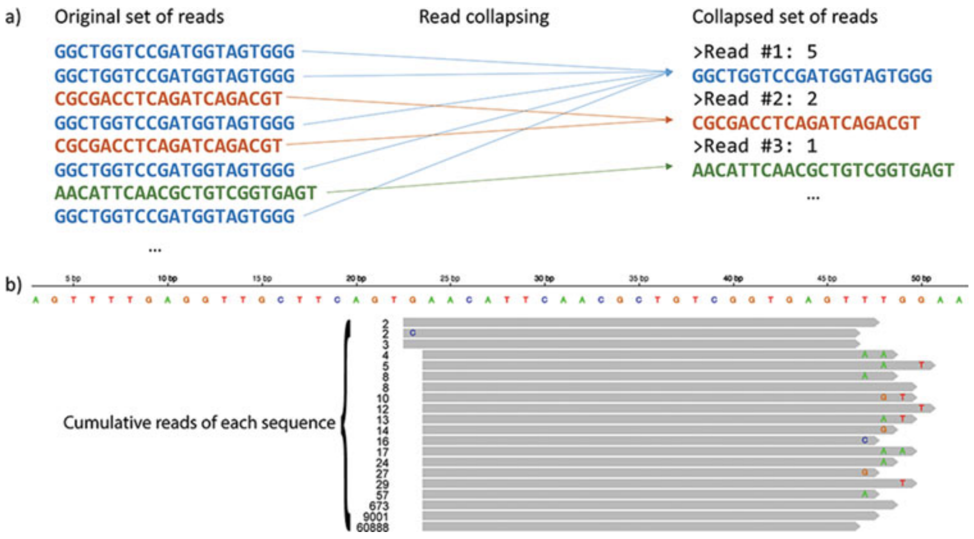


Fig. 1 The ability to sequence the entire miRNA read makes it advantageous to collapse identical reads together before the alignment step. (**a**) Example of duplicate sequences being collapsed together. (**b**) Integrated Genome Viewer (IGV) screen capture demonstrating how different isomiRs all align to the same miRNA (miR-181a). The hypothetical numbers ranging from 2 to 60,888 demonstrate the vast disparity of read counts of different isomiRs

1.3 The Importance of Reference Libraries

Although micro-RNAs have been among the most intensively studied molecules of the past 25 years, determining what is and what is not a micro-RNA has previously not been straightforward. Today, however, based on the knowledge of their unique and very restrictive biogenesis, canonical micro-RNAs are very well characterized by a distinctive suite of features not seen in other types of small RNAs [3, 16–18]. Regrettably, recognition and implementation of these clear and mechanistically well-understood features is not common practice and, for years, a major concern in micro-RNA research has been the quality of the central online repository miRBase [19, 20]. Estimates of 2/3 false-positive entries in miRBase are, unfortunately, in line with other, independent, extreme overestimations of the human micro-RNA complement [21, 22]. Given the aforementioned fundamental roles of micro-RNAs, especially in diseases, it is imperative that bona fide micro-RNAs are clearly distinguished from non-micro-RNAs and correctly identified, annotated, and profiled using consistent criteria to avoid spurious conclusions about diseases [23, 24].

For animals, including human, MirGeneDB represents a manually curated alternative to miRBase [9, 10]. In its second version more than 11,000 micro-RNAs of 45 metazoan organisms are represented and a comparison of these micro-RNA complements has not only confirmed a large number of false positive entries in miRBase, but found also a surprisingly high number of missing

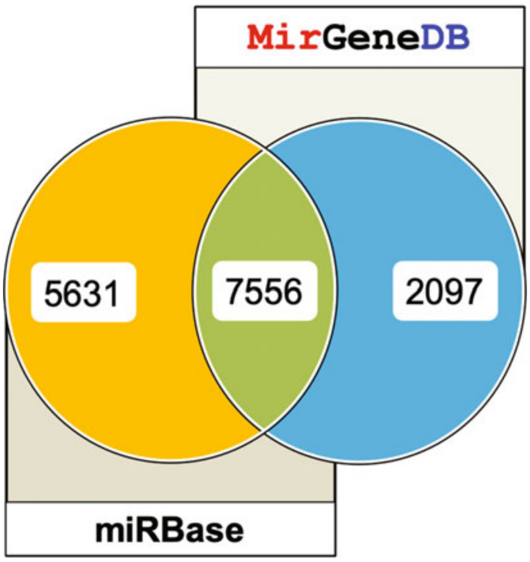


Fig. 2 The overlap of micro-RNAs between miRBase v18 and MirGeneDB2.0 across all species

entries, an important problem for alignment-based profiling of micro-RNAs - again highlighting the multiple issues there are with a community and submission-based repository (Fig. 2).

We stress that many of the miRBase entries rejected as bona fide micro-RNAs might still be functional small RNAs, such as tRNA, snoRNA fragments and miRTrons, but are not part of the highly conserved and efficiently shaped micro-RNAome. Typically, however, they are found at low endogenous levels representing transcriptional noise rather than biological signal. Therefore, both presented tools have the option to still use miRBase as a reference for alignment, but we strongly recommend using MirGeneDB to identify biologically functional micro-RNAs similar to other leading micro-RNA laboratories [3, 25–27].

2 Materials

2.1 Features and Rationale of the Micro-RNA Aligner miRge

miRge and miRge 2.0 were designed for stand-alone use as ultrafast aligners of FASTQ files in bulk [6, 28]. miRge 2.0, the current version of the tool, inputs an unlimited number of FASTQ files (machine RAM dependent) for consolidated workflow. A detailed workflow is given below, but the general rationale of the tool is described here. The first step removes 3′ adapter sequences and poor quality reads using Cutadapt v1.16 in preparation of alignment [29]. The second step collapses all of the identical reads together for accelerated alignment through the Bowtie steps. These initial steps are repeated for all of the submitted FASTQ files before proceeding through the alignment.

The reduced search parameter size, after sequence collapse allows for ultrafast alignments. miRge takes advantage of this by performing multiple separate alignment steps. The first step is an exact match to a modified micro-RNA sequence library using Bowtie [30]. This library has two 5′ nucleotide and six 3′ nucleotide genomic extensions of the mature micro-RNAs and SNPs included. This will allow for all length variants (genomic match) to be captured in the first step. The library is crafted to use either the MirGeneDB 2.0 or miRBase set of micro-RNAs. This step establishes sequence reads that match the definitive micro-RNA in the small RNA library. These sequences are then removed from miRge searches going forward. miRge then performs additional searches against other RNA libraries (tRNA, mRNA, snoRNA, rRNA, etc.), allowing a single mismatch. This step is used to capture sequences that align to non-micro-RNAs. At this point, the remaining reads consist of nontemplated isomiRs (with 3′ extensions or internal modifications) and nonalignable RNA reads (contamination, poor sequencing, etc.). miRge takes a broad approach at this step to try and capture as many isomiRs as possible from this remaining data file by performing a final Bowtie alignment using loose search parameters (skipping the first and last three nucleotides and allowing up to 3 misaligned base pairs). After all of these steps, miRge consolidates each of the searches into a unified set of reports. Within the reports, a key step is the amalgamating of micro-RNA results.

It is known that within an isomiR family of a given micro-RNA, the majority of reads are templated (exactly match the genomic sequence). The use of two alignment steps for micro-RNAs (an exact match and a loose match) allows miRge to evaluate this ratio for each micro-RNA. miRge takes advantage of this by rejecting micro-RNAs in which too high a percentage of the total reads are nontemplated reads. The inability to control for this in other methods results in problems. Micro-RNAs with sequences nearly identical to repeat elements (e.g., miR-3168 shares high sequence homology with a MIR SINE element) can be substantially overcalled using traditional alignment approaches that allow mismatches. To avoid this, some alignment methods take a very conservative approach to alignment (e.g., no or one mismatch), but this causes many nontemplated isomiRs to be skipped and micro-RNAs to be underreported.

Another challenge of micro-RNAs is the near perfect matching of micro-RNAs within the same family. For example, hsa-let-7a and hsa-let-7c are identical except for an A/G variant at the 19th nucleotide position. An isomiR of hsa-let-7c that ends at position 18 and has an A nucleotide added as a 3′ extension will exactly match an hsa-let-7a isomiR that is 19 nucleotides long. Therefore, miRge has uniquely approached these highly similar micro-RNAs (with identical seed sequences, but a nucleotide difference at positions 19 or higher) by combining their read counts in the final reporting. This makes an assumption that the identical seed sequences will result in overlapping functionality.

2.2 Features and Rationale of sRNAbench

sRNAbench is the successor program of miRanalzyer which was the first web-server available for the analysis of micro-RNA high-throughput sequencing data [31]. In 2011, the first stand-alone version was released [32]. Although from the beginning, other RNA species were analyzed as well, the initial focus was clearly on the expression profiling of known micro-RNAs and the prediction of novel micro-RNAs. Since then, new bioinformatics challenges have arisen due to novel protocols, the apparent importance of micro-RNA variants (isomiRs) or tRNA fragments. In 2014 miR-analyzer was reimplemented from scratch to better correspond to this new complexity in small RNA sequencing data renaming it to sRNAbench. sRNAbench is now a part of sRNAtoolbox which comprises several tools for RNA research [33, 34]. In order to conduct a differential expression analysis, all samples need to be first analyzed with *sRNAbench*, calculating differential expression in a second step by means of *sRNAde*, the differential expression tool that is also part of sRNAtoolbox.

sRNAbench was successfully used on metazoan and plant data but also in other study designs like virus–host or parasite–host interactions. It allows simultaneous analysis of an unlimited number of species, which is important if the genetic material in the experiment can come from different sources like in the case of virus infection. Nowadays, miRNA-seq data can be easily generated by virtually all molecular biology labs. However, a trained bioinformatician is not always available to analyze the data. Therefore, sRNAtoolbox is available as a user-friendly web-server, virtual machine, Docker, and stand-alone version. While the web-server might be the choice of users with little or no bioinformatics background, the command line usage gives access to many parameters and options not available in the web-server.

In theory, these tools are not limited to predefined species, however, to allow for species that are not in miRBase/MirGeneDB, a local database needs to be generated. To aid with this task we developed several helper tools that are described in the manual.

2.3 Main Features of sRNAtoolbox

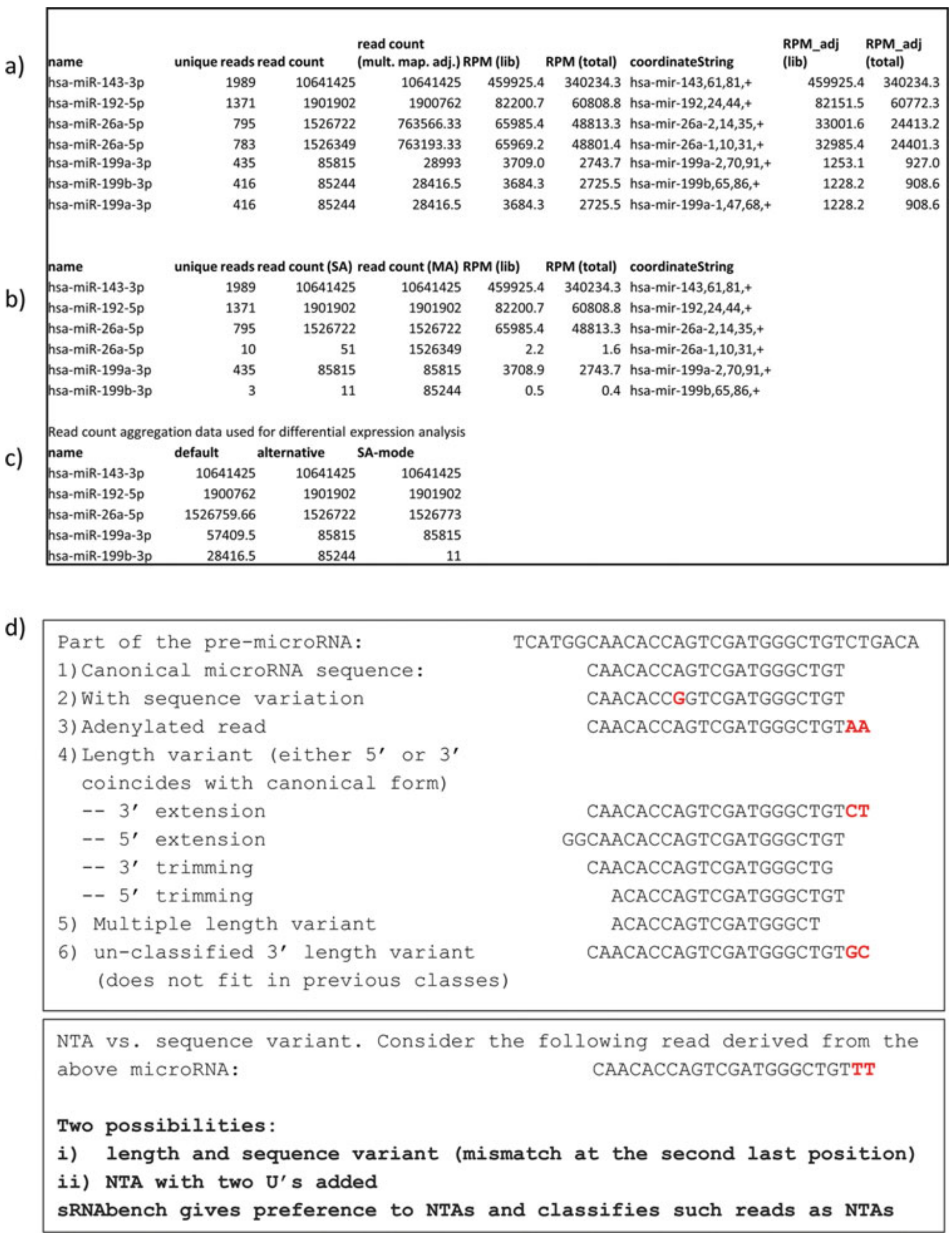
- Profiling of miRBase, MirGeneDB, or Pmiren micro-RNAs or any other source with a coherent nomenclature for precursor and mature sequences.
- Detection and classification of micro-RNA variants (isomiRs).
- Prediction of novel micro-RNAs with models for animals and plants.
- Differential expression using five different statistical methods. Statistically significant differences can be explored also for

isomiRs, or other properties like read length distributions or RNA species content.

- Multimapping is addressed in two different ways: i) simple adjustment of the read counts dividing it by the number of mappings and ii) single assignment, that is, each read is only assigned once to the loci with most overall mappings.
- Complete preprocessing including adapter trimming, quality control and read collapsing. All major library processing protocols (Illumina, NEBnext, NEXTflex, or Qiagen) can be used including different UMI designs.
- Reads can be filtered out providing "filter libraries," that is, if ribosomal RNA should be filtered out before (when predicting novel micro-RNAs for example).
- Spike-in sequences can be provided and additional, spike-in normalized output files are provided.
- Guess-protocols: sRNAbench provides both, the automatic detection of the used library preparation protocol and the species.

2.4 Multiple Mapping and Differential Expression in sRNAbench

In many micro-RNA studies, the ultimate goal is the detection of differentially expressed sequences. Micro-RNAs are frequently transcribed from several loci in the genome and the sequences of different members of the same family are highly similar. As a consequence, a read can map more than once to the genome or a given reference library and a decision needs to be made on how to deal with these multiple mapping reads. sRNAbench estimates read counts (i.e., the number of reads assigned to a reference sequence) in three different ways: (1) assigning the full read count to all loci, (2) dividing the read count by the number of mappings (adjusted read count) and (3) perform a single assignment (SA), that is, each read is assigned only once, to the loci or reference sequence with the highest number of assigned total read count. This is illustrated in Fig. 3a and b. If a micro-RNA has more than one locus, it will appear several times in the "*mature_sense.grouped*" output file. In column 6, however, it can be determined to which loci or reference sequence this mature micro-RNA belongs. When calculating differential expression, the abundance of the mature sequence is considered and therefore a decision needs to be made on how a final expression value is assigned to each mature micro-RNA. Figure 3c shows the three strategies implemented into sRNAde: (1) summing up the adjusted read counts (default method), (2) assigning the expression of the loci with the highest RC, (3) performing a single assignment, that is, assigning each read only once to the loci with the overall highest read count over all samples. Finally, sRNAde calculates the differential expression by means of four third-party methods (DESeq, DESeq2, edgeR, and NoiSeq) and a Student's *t*-test using read-per-million (RPM) normalized values.



a)

name	unique reads	read count	read count (mult. map. adj.)	RPM (lib)	RPM (total)	coordinateString	RPM_adj (lib)	RPM_adj (total)
hsa-miR-143-3p	1989	10641425	10641425	459925.4	340234.3	hsa-mir-143,61,81,+	459925.4	340234.3
hsa-miR-192-5p	1371	1901902	1900762	82200.7	60808.8	hsa-mir-192,24,44,+	82151.5	60772.3
hsa-miR-26a-5p	795	1526722	763566.33	65985.4	48813.3	hsa-mir-26a-2,14,35,+	33001.6	24413.2
hsa-miR-26a-5p	783	1526349	763193.33	65969.2	48801.4	hsa-mir-26a-1,10,31,+	32985.4	24401.3
hsa-miR-199a-3p	435	85815	28993	3709.0	2743.7	hsa-mir-199a-2,70,91,+	1253.1	927.0
hsa-miR-199b-3p	416	85244	28416.5	3684.3	2725.5	hsa-mir-199b,65,86,+	1228.2	908.6
hsa-miR-199a-3p	416	85244	28416.5	3684.3	2725.5	hsa-mir-199a-1,47,68,+	1228.2	908.6

b)

name	unique reads	read count (SA)	read count (MA)	RPM (lib)	RPM (total)	coordinateString
hsa-miR-143-3p	1989	10641425	10641425	459925.4	340234.3	hsa-mir-143,61,81,+
hsa-miR-192-5p	1371	1901902	1901902	82200.7	60808.8	hsa-mir-192,24,44,+
hsa-miR-26a-5p	795	1526722	1526722	65985.4	48813.3	hsa-mir-26a-2,14,35,+
hsa-miR-26a-5p	10	51	1526349	2.2	1.6	hsa-mir-26a-1,10,31,+
hsa-miR-199a-3p	435	85815	85815	3708.9	2743.7	hsa-mir-199a-2,70,91,+
hsa-miR-199b-3p	3	11	85244	0.5	0.4	hsa-mir-199b,65,86,+

c)

Read count aggregation data used for differential expression analysis

name	default	alternative	SA-mode
hsa-miR-143-3p	10641425	10641425	10641425
hsa-miR-192-5p	1900762	1901902	1901902
hsa-miR-26a-5p	1526759.66	1526722	1526773
hsa-miR-199a-3p	57409.5	85815	85815
hsa-miR-199b-3p	28416.5	85244	11

d)

```
Part of the pre-microRNA:          TCATGGCAACACCAGTCGATGGCTGTCTGACA
1)Canonical microRNA sequence:           CAACACCAGTCGATGGCTGT
2)With sequence variation                CAACACCGGTCGATGGCTGT
3)Adenylated read                        CAACACCAGTCGATGGCTGTAA
4)Length variant (either 5' or 3'
  coincides with canonical form)
  -- 3' extension                        CAACACCAGTCGATGGCTGTCT
  -- 5' extension                      GGCAACACCAGTCGATGGCTGT
  -- 3' trimming                         CAACACCAGTCGATGGCTG
  -- 5' trimming                           ACACCAGTCGATGGCTGT
5) Multiple length variant                 ACACCAGTCGATGGGCT
6) un-classified 3' length variant       CAACACCAGTCGATGGGCTGTGC
   (does not fit in previous classes)
```

```
NTA vs. sequence variant. Consider the following read derived from the
above microRNA:                           CAACACCAGTCGATGGGCTGTTT

Two possibilities:
i)  length and sequence variant (mismatch at the second last position)
ii) NTA with two U's added
sRNAbench gives preference to NTAs and classifies such reads as NTAs
```

Fig. 3 Example output of sRNABench. Output files (**a**) mature_sense.grouped and (**b**) mature_senseSA.grouped list all loci with the same mature micro-RNA sequence. For example, hsa-miR-26a-5p originates from two genomic loci and therefore it appears twice. The second column shows the full read count (each read is fully assigned to all positions) and in the third column the read count is divided by the number of locations that it maps to (multiple mapping adjusted). In the single assignment (SA) file no adjusted read counts exist as each read is only assigned once (**b**). If the second and third column have the exact same value, this indicates no multiple mapping reads are assigned to this micro-RNA (ex. 143-3p in row 1). (**c**) In the expression matrix, each micro-RNA sequence appears once. This data is used by sRNAde. (**d**) The isomiR classification schema and how some "special" cases are treated

2.5 sRNAbench isomiR Detection and Classification

sRNAbench approaches the challenges of nontemplated additions for isomiRs by using the Bowtie seed alignment option which scores only the first L nucleotides ($L = 19$ by default). In this way, mismatches caused by nontemplated additions are not taken into account for the reporting of an alignment. In general, a read can fall into more than one category of isomiRs, that is, it can be both, a length and sequence variant. By default, sRNAbench applies a hierarchical classification assigning each read to only one group (Fig. 3d).

3 Methods

3.1 miRge 2.0 Hardware and Software

miRge 2.0 is best installed on a linux system with robust RAM and CPUs. miRge 2.0 is installed via bioconda and detailed instructions on installation are located at https://github.com/mhalushka/miRge. A number of standard Python dependencies are needed as well as the stand-alone programs Cutadapt (currently compatible to v1.16) and Bowtie.

3.2 Running miRge 2.0

miRge 2.0 is run at the command line. Two main uses for miRge 2.0 exist. The standard use of miRge 2.0 is to annotate a FASTQ file for known micro-RNAs. The second use is to both annotate and predict novel micro-RNAs from a FASTQ file.

For pure annotation, the simplest usage would be as follows:

```
$miRge2.0 annotate -pb [path_to_bowtie] -lib [path_to_align-
ment_libraries] -sp [species] -ad [adaptor_options] -s [se-
quence_file_name(s).fastq]
```

With actual information, that command line might look like this:

```
$miRge2.0 annotate -pb /usr/local/bowtie-1.2.1/bowtie -lib /
home/libraries/miRge.Libs/ -sp human -ad illumina -s
SRR649562.fastq SRR649564.fastq
```

In this example, miRge2.0 will remove the standard Illumina TruSeq Small RNA library adaptor and use the human miRge libraries on two FASTQ files.

The full options available in miRge 2.0 for annotations are available in Table 1.

Table 1
Annotation mode options for miRge 2.0

`-h, --help`	show this help message and exit
`-s [sample <required> [sample <required> ...]]`	Two options: 1. A file where each row represents one sample name; 2. *.fastq *.fastq ...
`-o <dir>`	The directory of the outputs (default: Current
`-d <string required>`	The miRNA database (default: miRBase. MirGeneDB is optional)
`-pb <dir required>`	The path to the systems bowtie binary
`-lib <dir required>`	The path to the miRge libraries
`-sp <string required>`	The species can be human, mouse, fruit fly, nematode, rat, and zebrafish (novel miRNA detection is confined to human and mouse)
`-ex <float>`	The threshold of the proportion of canonical reads for the miRNAs to determine whether keeping them or not when counting. Users can set it between 0 and 0.5 (default: 0.1)
`-ad <string>`	The adapter need to be removed which could be illumina, ion or a defined sequence (default: none)
`-phred64`	phred64 format (default: 33)
`-spikeIn`	Switch to annotate spike-ins if the bowtie index files are located at the path of bowties index files (default: off)
`-tcf`	Switch to write trimmed and collapsed fasta file (default: off)
`-di`	Switch to calculate of isomirs entropy (default: off)
`-cpu <int>`	The number of processors to use for trimming, qc, and alignment (default: 1)
`-ai`	Switch to calculate of A to I editing (default: off)
`-gff`	Switch to output results in gff format (default: off)
`-trf`	Switch to analyze tRNA fragment (default: off)
`--version`	Show programs version number and exit

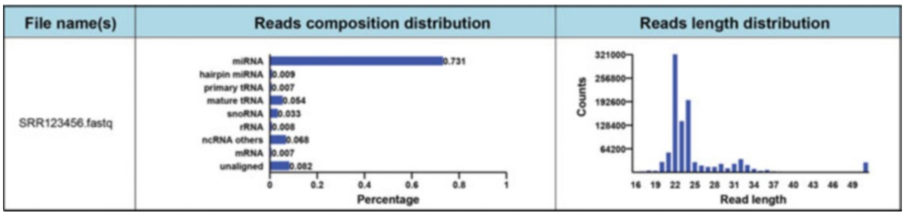


Fig. 4 miRge 2.0 annotation report showing the percent of different RNA species and the length distribution of reads after adaptor trimming

At the completion of the run, miRge outputs five or more reports, depending on the options chosen. An overall annotation report in both html and csv styles shows the distribution of reads across 14 categories including the total number of unique micro-RNAs detected and the total number of micro-RNA reads. The html version graphs the read length distribution and the percent of micro-RNAs and other species relative to all other reads (Fig. 4). Separate micro-RNA counts and RPM files report the abundance of each micro-RNA type after the cleaning strategy described above. Files of mapped or unmapped reads are given to allow a user to identify the assignment (or lack of) for any given sequencing read. Other optional reporting can give detailed information on tRNA fragments utilizing a new clustering approach to this challenging RNA species.

miRge 2.0 can also be used to predict novel micro-RNAs. While essentially all bona fide micro-RNAs have been detected in standard tissues of major species, there is a limited opportunity to detect organ or cell specific micro-RNAs from poorly characterized samples. To that end, miRge 2.0 has a robust, yet conservative micro-RNA prediction algorithm based on key features of real micro-RNAs.

For micro-RNA prediction and annotation, the simplest usage would be as follows:

```
$mirge2.0 predict -pb [path_to_bowtie] -lib [path_to_alignmen-
t_libraries] -ps [path_to_SAM_tools] -pr [path_to_RNAfold] -sp
[species] -ad [adaptor_options] -s [sequence_file_name(s).
fastq]
```

With input information, this command line might look like:

```
$miRge2.0 predict -pb /usr/local/bowtie-1.2.1/bowtie -lib /
home/libraries/miRge.Libs/ -ps /usr/local/bin -pr /usr/local/
bin -sp human -ad ion -s SRR649562.fastq SRR649564.fastq
```

Running miRge2.0 in predict mode can replicate the steps of the annotation mode, but greatly increases the time to process the samples due to the lengthy analysis needed to identify novel microRNAs. The full options available in miRge 2.0 for prediction mode are available in Table 2.

Table 2
Predict mode options for miRge 2.0

`-h, --help`	show this help message and exit
`-s [sample <required> [sample <required> ...]]`	Two options: 1. A file where each row represents one sample name; 2. *.fastq *.fastq ...
`-o <dir>`	The directory of the outputs (default: current
`-d <string required>`	The miRNA database (default: miRBase. MirGeneDB is optional)
`-pb <dir required>`	The path to the systems bowtie binary
`-lib <dir required>`	The path to the miRge libraries
`-sp <string required>`	The species can be human, mouse, fruitfly, nematode, rat and zebrafish (novel miRNA detection is confined to human and mouse)
`-ps <dir required>`	The path to the systems samtools binary
`-pr <dir required>`	The path to the systems rnafold binary
`-ex <float>`	The threshold of the proportion of canonical reads for the miRNAs to determine whether keeping them or not when counting. Users can set it between 0 and 0.5 (default: 0.1)
`-ad <string>`	The adapter need to be removed which could be illumina, ion or a defined sequence (default: none)
`-phred64`	phred64 format (default: 33)
`-spikeIn`	Switch to annotate spike-ins if the bowtie index files are located at the path of bowties index files (default: off)
`-tcf`	Switch to write trimmed and collapsed fasta file (default: off)
`-di`	Switch to calculate of isomirs entropy (default: off)
`-cpu <int>`	The number of processors to use for trimming, qc, and alignment (default: 1)

(continued)

Table 2
(continued)

`-ai`	`Switch to calculate of A to I editing (default: off)`
`-gff`	`Switch to output results in gff format (default: off)`
`-trf`	`Switch to analyze tRNA fragment (default: off)`
`-ws <file>`	`The file containing the overall samples to analysis for novel miRNA prediction. No header, just a list of *.fastq file names in a column. Names of files can be to your choosing (e.g., filestochecknovel.txt).`
`-minl <int>`	`The minimum length of the retained reads for novel miRNA detection (default: 16)`
`-maxl <int>`	`The maximum length of the retained reads for novel miRNA detection (default: 25)`
`-cc <int>`	`The maximum read count of the retained reads for novel miRNA detection (default: 2)`
`-ml <int>`	`The maximum number of mapping loci for the retained reads for novel miRNA detection (default: 3)`
`-sl <int>`	`The seed length when invoking Bowtie for novel miRNA detection (default: 25)`
`-olc <int>`	`The length of overlapped sequence when joining reads into longer sequences based on the coordinate on the genome for novel miRNA detection (default: 14)`
`-clc <int>`	`The maximum length of the clustered sequences for novel miRNA detection (default: 30)`
`--version`	`Show programs version number and exit`

3.3 sRNAbench Hardware and Software

To run sRNAbench 8Gbs of RAM are recommended and parallelization can be increased with more CPUs. sRNAbench is implemented in Java as a stand-alone command line tool and installation and user manuals, Docker images or change-logs can be accessed through this link: https://bioinfo2.ugr.es/srnatoolbox/. A web-server was also implemented as a solution to users with little or no bioinformatics background or no access to computing resources.

Although fewer analyses and parameters are available on the web, it is user-friendly and relatively fast even when a large number of jobs are submitted.

3.4 Using sRNAbench and sRNAde

By means of sRNAtoolbox webserver (currently hosted at https://arn.ugr.es/srnatoolbox) users can run sRNAbench to profile one or several (using batch mode) micro-RNA sequencing files (Fig. 5a). Using the sidebar menu, one can navigate to the desired sRNAbench flavor. Once inside sRNAbench webpage, input reads can be provided as an uploaded FASTQ file, a link, or an SRA Run accession ID. Then, one or several species can be selected to customize sample analysis according to experimental design, as well as genome or library mode. Finally users should provide protocol information for preprocessing of the input reads (the five most common micro-RNA-seq protocols are preloaded but any parameter combination can be produced using custom options). Several other parameters that deal with quality control and micro-RNA prediction are also available but not required. Example data and instructions can be found on the web.

After job(s) completion, a results page including different sections will be displayed. Sections include summaries, visualization and detailed profiles information (Fig. 5b).

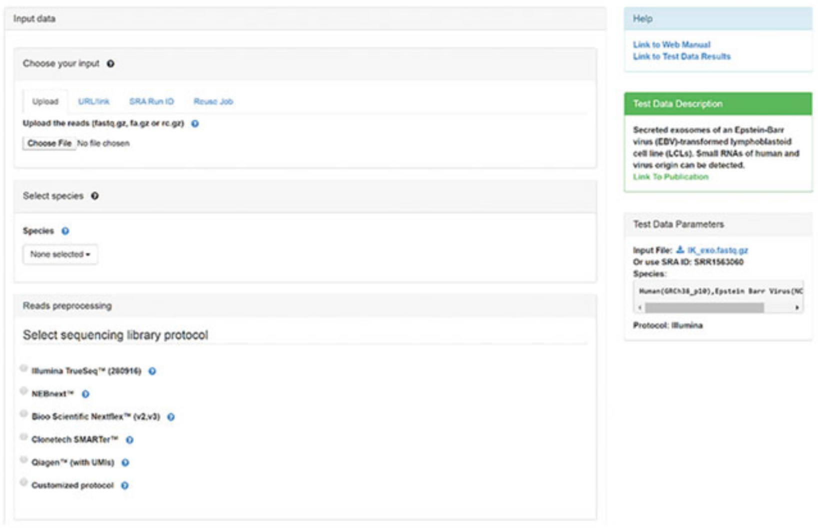
Frequently, different conditions are considered in experimental designs to quantify differences among groups of samples. To assess these differences, several algorithms have been proposed [35–38]. Once sRNAbench jobs are completed users can easily pipe their data into a differential expression tool (sRNAde) that will run five different algorithms. To do so, one will input all sRNAbench jobIDs and assign them to a group following onscreen instructions. The Results page includes quality control summary, consensus and per method differential expression, variant analysis and several figures including heatmaps [39] and UpSet plots [40] (Fig. 5c).

As mentioned above, sRNAbench and sRNAde can also be run as command line tools. These versions will surely be preferable for users who can access computing clusters and have some scripting skills. Example lines are available on the online manual https://bioinfo2.ugr.es/srnatoolbox/manual/.

The simplest general line would look like this:

```
$sRNAbench input=[path_to_input] output=[path_to_output] protocol=[library_protocol] microRNA=[species_miRBase_short_name] minRC=2
```

a)



Input data

Choose your input

Upload the reads (fastq.gz, fa.gz or rc.gz)

Choose File No file chosen

Select species

Species

None selected ▾

Reads preprocessing

Select sequencing library protocol

- ○ Illumina TrueSeq™ (280916)
- ○ NEBnext™
- ○ Bioo Scientific Nextflex™ (v2,v3)
- ○ Clonetech SMARTer™
- ○ Qiagen™ (with UMIs)
- ○ Customized protocol

Help

Link to Web Manual
Link to Test Data Results

Test Data Description

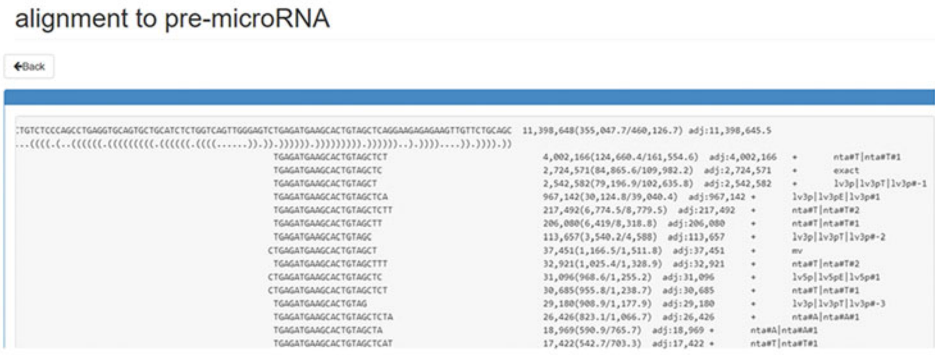
Secreted exosomes of an Epstein-Barr virus (EBV)-transformed lymphoblastoid cell line (LCLs). Small RNAs of human and virus origin can be detected.
Link To Publication

Test Data Parameters

Input File: IK_exo.fastq.gz
Or use SRA ID: SRR1563060
Species:
Human(GRCh38_p10),Epstein Barr Virus(NC

Protocol: Illumina

b)



alignment to pre-microRNA

```
:TGTCTCCCAGCCTGAGGTGCAGTGCTGCATCTCTGGTCAGTTGGGAGTCTGAGATGAAGCACTGTAGCTCAGGAAGAGAGAAGTTGTTCTGCAGC  11,398,648(355,047.7/460,126.7) adj:11,398,645.5
...((((.(..((((((.(((((((((.((((((.((((......)).)).)))))).))))))))).))))))..).))))....)).)))).))
                                                        TGAGATGAAGCACTGTAGCTCT          4,002,166(124,660.4/161,554.6)  adj:4,002,166   •       nta#T|nta#T#1
                                                        TGAGATGAAGCACTGTAGCTC           2,724,571(84,865.6/109,982.2)  adj:2,724,571   •       exact
                                                        TGAGATGAAGCACTGTAGCT            2,542,582(79,196.9/102,635.8)  adj:2,542,582   •       lv3p|lv3pT|lv3p#-1
                                                        TGAGATGAAGCACTGTAGCTCA          967,142(30,124.8/39,040.4)  adj:967,142 •       lv3p|lv3pE|lv3p#1
                                                        TGAGATGAAGCACTGTAGCTCTT         217,492(6,774.5/8,779.5)  adj:217,492   •       nta#T|nta#T#2
                                                        TGAGATGAAGCACTGTAGCTT           206,080(6,419/8,318.8)  adj:206,080     •       nta#T|nta#T#1
                                                        TGAGATGAAGCACTGTAGC             113,657(3,540.2/4,588)  adj:113,657     •       lv3p|lv3pT|lv3p#-2
                                                       CTGAGATGAAGCACTGTAGCT            37,451(1,166.5/1,511.8)  adj:37,451     •       mv
                                                        TGAGATGAAGCACTGTAGCTTT          32,921(1,025.4/1,328.9)  adj:32,921     •       nta#T|nta#T#2
                                                       CTGAGATGAAGCACTGTAGCTC           31,096(968.6/1,255.2)  adj:31,096       •       lv5p|lv5pE|lv5p#1
                                                       CTGAGATGAAGCACTGTAGCTCT          30,685(955.8/1,238.7)  adj:30,685       •       nta#T|nta#T#1
                                                        TGAGATGAAGCACTGTAG              29,180(908.9/1,177.9)  adj:29,180       •       lv3p|lv3pT|lv3p#-3
                                                        TGAGATGAAGCACTGTAGCTCTA         26,426(823.1/1,066.7)  adj:26,426       •       nta#A|nta#A#1
                                                        TGAGATGAAGCACTGTAGCTA           18,969(590.9/765.7)  adj:18,969 •       nta#A|nta#A#1
                                                        TGAGATGAAGCACTGTAGCTCAT         17,422(542.7/703.3)  adj:17,422 •       nta#T|nta#T#1
```

c)

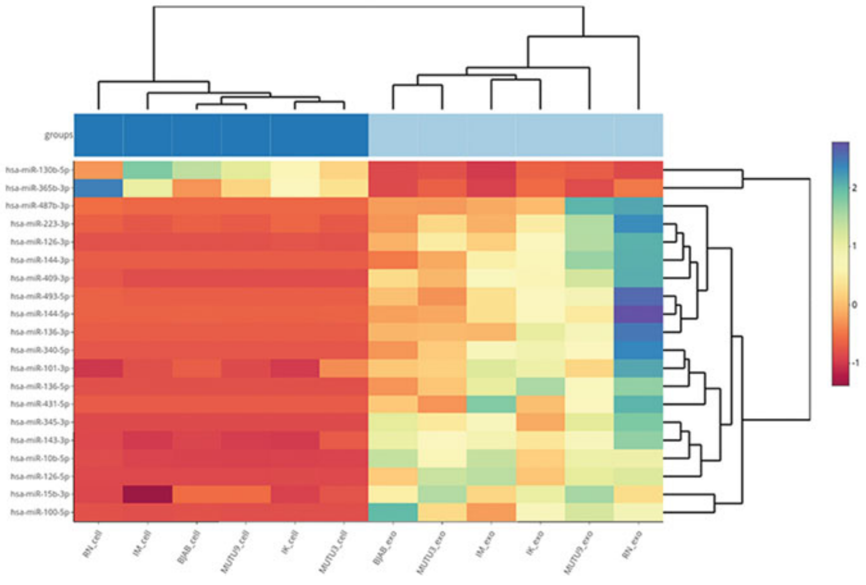


Fig. 5 (**a**) sRNAbench input form. Users can easily provide all necessary data to launch their job that will automatically be queued in the system. (**b**) For all detected micro-RNAs, the alignment of the reads to the

Using a specific SRA Run accession (SRR649562):

```
$sRNAbench input=SRR649562 output=output_folder protocol=Il-
lumina microRNA=hsa minRC=2
```

sRNAbench can also predict new micro-RNAs from sequencing data:

```
$sRNAbench input=[path_to_input] output=[path_to_output] pro-
tocol=[library_protocol] species=[genome_assembly] predict=-
true alignType=v minReadLength=18 maxReadLength=26
```

Using the previous example run:

```
$sRNAbench input=SRR649562 output=output_folder protocol=Il-
lumina species= GRCh38_p10_mp predict=true alignType=v minRea-
dLength=18 maxReadLength=26
```

3.5 Running Datasets Through sRNAbench and miRge Using MirGeneDB and miRBase

To demonstrate the functionality of miRge and sRNAbench, we ran two public samples (SRR649562 and SRR8393494) through both methods and report their output. For both options we aligned to both MirGeneDB and miRBase libraries. Table 3 reports the key parameters from each method, and Fig. 6 shows pairwise correlations across the methods. These results demonstrate how different choices made by the aligners can result in some remarkably different values including how many micro-RNAs were detected. However, the output of these two tools are quite robust and ultimately the micro-RNA RPM values are essentially the same across the methods.

Fig. 5 (continued) precursor sequence can be visualized (top expressed hsa-miR-143 is shown). A clear stack of reads corresponds to the mature sequences, that is, the canonical sequence and isomiRs. To the right of the mature sequences, the total read count and the adjusted read count are given. The numbers between the brackets correspond to the total RPM and library RPM of the read counts. In this example, the most frequent mature sequence does not correspond to the canonical sequence ("exact" label at the right at the second most frequent sequence) but to a monouridylated version of this sequence (nta#T#1). (**c**) A heatmap displaying expression values of differentially expressed micro-RNAs produced using sRNAde online

Table 3
A comparison of sRNAbench and miRge analysis of samples SRR649562 (human colon) and SRR8393494

	SRR649562[a]				SRR8393494[b]			
	miRge		sRNAbench		miRge		sRNAbench	
	MirGeneDB	miRBase	MirGeneDB	miRBase	MirGeneDB	miRBase	MirGeneDB	miRBase
Total input reads	37,373,347	37,373,347	37,373,347	37,373,347	4,376,562	4,376,562	4,376,562	4,376,562
Trimmed reads	34,264,948	34,264,948	33,555,912	33,555,912	4,183,021	4,183,021	3,263,483	3,263,483
Unique Reads[c]	1,412,840	1,412,840	314,241	314,241	189,348	189,348	33,880	33,880
Aligned micro-RNA reads	24,833,903	24,874,635	24,775,814	24,809,235	3,086,645	3,093,167	3,039,143	3,047,819
Unique micro-RNAs	615	925	627	1010	447	574	459	608
Aligned non-micro-RNA reads	7,972,544	7,966,888	8,441,253	8,358,844	926,747	926,350	207,982	199,235
Unaligned reads	1,443,614	1,359,998	338,845	387,833	168,200	157,878	16,358	16,429

[a]Human control colon tissue with Illumina TruSeq Small RNA library
[b]Human papillary thyroid carcinoma with NEXTflex sequencing kit v3
[c]The sRNAbench webserver applies a read count threshold of 2, filtering out singleton reads, whereas miRge does not

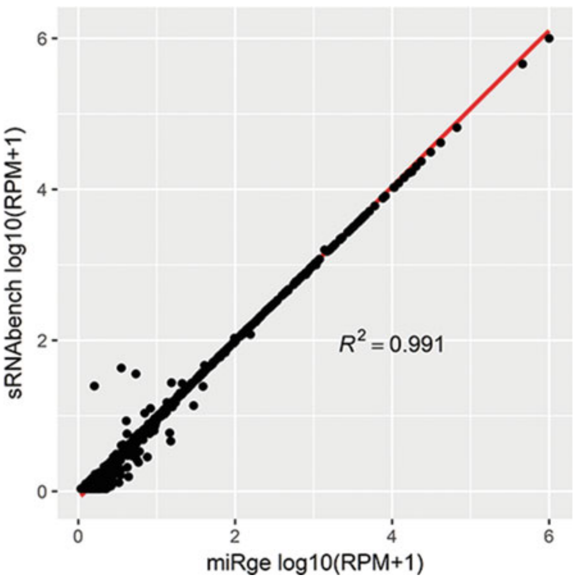


Fig. 6 Comparison of sRNAbench and miRge 2.0 for sample SRR8393494. Output is log10 RPM based on alignment to MirGeneDB micro-RNAs. Each dot represents one micro-RNA. There is strong agreement across the two methods

References

1. Halushka MK, Fromm B, Peterson KJ, McCall MN (2018) Big strides in cellular micro-RNA expression. Trends in genetics : TIG 34 (3):165–167. https://doi.org/10.1016/j.tig.2017.12.015
2. Pliatsika V, Loher P, Magee R, Telonis AG, Londin E, Shigematsu M, Kirino Y, Rigoutsos I (2018) MINTbase v2.0: a comprehensive database for tRNA-derived fragments that includes nuclear and mitochondrial fragments from all The Cancer Genome Atlas projects. Nucleic Acids Res 46(D1):D152–D159. https://doi.org/10.1093/nar/gkx1075
3. Bartel DP (2018) Metazoan Micro-RNAs. Cell 173(1):20–51. https://doi.org/10.1016/j.cell.2018.03.006
4. Mendell JT, Olson EN (2012) Micro-RNAs in stress signaling and human disease. Cell 148 (6):1172–1187. https://doi.org/10.1016/j.cell.2012.02.005
5. Calin GA, Croce CM (2006) Micro-RNA-cancer connection: the beginning of a new tale. Cancer Res 66(15):7390–7394. https://doi.org/10.1158/0008-5472.CAN-06-0800
6. Baras AS, Mitchell CJ, Myers JR, Gupta S, Weng LC, Ashton JM, Cornish TC, Pandey A, Halushka MK (2015) miRge – a multiplexed method of processing small RNA-Seq data to determine micro-RNA entropy. PLoS One 10(11):e0143066. https://doi.org/10.1371/journal.pone.0143066
7. Neilsen CT, Goodall GJ, Bracken CP (2012) IsomiRs—the overlooked repertoire in the dynamic micro-RNAome. Trends Genet 28 (11):544–549. https://doi.org/10.1016/j.tig.2012.07.005
8. McCall MN, Kim MS, Adil M, Patil AH, Lu Y, Mitchell CJ, Leal-Rojas P, Xu J, Kumar M, Dawson VL, Dawson TM, Baras AS, Rosenberg AZ, Arking DE, Burns KH, Pandey A, Halushka MK (2017) Toward the human cellular micro-RNAome. Genome Res 27 (10):1769–1781. https://doi.org/10.1101/gr.222067.117
9. Fromm B, Domanska D, Hoye E, Ovchinnikov V, Kang W, Aparicio-Puerta E, Johansen M, Flatmark K, Mathelier A, Hovig E, Hackenberg M, Friedlander MR, Peterson KJ (2020) MirGeneDB 2.0: the metazoan micro-RNA complement. Nucleic Acids Res 48(D1):D132–D141. https://doi.org/10.1093/nar/gkz885
10. Kozomara A, Birgaoanu M, Griffiths-Jones S (2019) miRBase: from micro-RNA sequences to function. Nucleic Acids Res 47(D1): D155–D162. https://doi.org/10.1093/nar/gky1141
11. Lukasik A, Wojcikowski M, Zielenkiewicz P (2016) Tools4miRs – one place to gather all

the tools for miRNA analysis. Bioinformatics 32(17):2722–2724. https://doi.org/10.1093/bioinformatics/btw189

12. Sakaram S, Craig MP, Hill NT, Aljagthmi A, Garrido C, Paliy O, Bottomley M, Raymer M, Kadakia MP (2018) Identification of novel DeltaNp63alpha-regulated miRNAs using an optimized small RNA-Seq analysis pipeline. Sci Rep 8(1):10069. https://doi.org/10.1038/s41598-018-28168-5
13. Kang W, Eldfjell Y, Fromm B, Estivill X, Biryukova I, Friedlander MR (2018) miRTrace reveals the organismal origins of micro-RNA sequencing data. Genome Biol 19(1):213. https://doi.org/10.1186/s13059-018-1588-9
14. Yang A, Bofill-De Ros X, Shao TJ, Jiang M, Li K, Villanueva P, Dai L, Gu S (2019) 3′ uridylation confers miRNAs with non-canonical target repertoires. Mol Cell 75(3):511–522. e514. https://doi.org/10.1016/j.molcel.2019.05.014
15. Desvignes T, Loher P, Eilbeck K, Ma J, Urgese G, Fromm B, Sydes J, Aparicio-Puerta-E, Barrera V, Espin R, Thibord F, Ros XB, Londin E, Telonis AG, Ficarra E, Friedlander MR, Postlethwait JH, Rigoutsos I, Hackenberg M, Vlachos IS, Halushka MK, Pantano L (2019) Unification of miRNA and isomiR research: the mirGFF3 format and the mirtop API. Bioinformatics 36:698. https://doi.org/10.1093/bioinformatics/btz675
16. Wang X, Liu XS (2011) Systematic curation of miRBase annotation using integrated small RNA high-throughput sequencing data for C. elegans and drosophila. Front Genet 2:25. https://doi.org/10.3389/fgene.2011.00025
17. Tarver JE, Donoghue PC, Peterson KJ (2012) Do miRNAs have a deep evolutionary history? BioEssays 34(10):857–866. https://doi.org/10.1002/bies.201200055
18. Taylor RS, Tarver JE, Hiscock SJ, Donoghue PC (2014) Evolutionary history of plant micro-RNAs. Trends Plant Sci 19(3):175–182. https://doi.org/10.1016/j.tplants.2013.11.008
19. Chiang HR, Schoenfeld LW, Ruby JG, Auyeung VC, Spies N, Baek D, Johnston WK, Russ C, Luo S, Babiarz JE, Blelloch R, Schroth GP, Nusbaum C, Bartel DP (2010) Mammalian micro-RNAs: experimental evaluation of novel and previously annotated genes. Genes Dev 24(10):992–1009. https://doi.org/10.1101/gad.1884710
20. Ludwig N, Becker M, Schumann T, Speer T, Fehlmann T, Keller A, Meese E (2017) Bias in recent miRBase annotations potentially associated with RNA quality issues. Sci Rep 7(1):5162. https://doi.org/10.1038/s41598-017-05070-0
21. Londin E, Loher P, Telonis AG, Quann K, Clark P, Jing Y, Hatzimichael E, Kirino Y, Honda S, Lally M, Ramratnam B, Comstock CE, Knudsen KE, Gomella L, Spaeth GL, Hark L, Katz LJ, Witkiewicz A, Rostami A, Jimenez SA, Hollingsworth MA, Yeh JJ, Shaw CA, McKenzie SE, Bray P, Nelson PT, Zupo S, Van Roosbroeck K, Keating MJ, Calin GA, Yeo C, Jimbo M, Cozzitorto J, Brody JR, Delgrosso K, Mattick JS, Fortina P, Rigoutsos I (2015) Analysis of 13 cell types reveals evidence for the expression of numerous novel primate- and tissue-specific micro-RNAs. Proc Natl Acad Sci U S A 112(10):E1106–E1115. https://doi.org/10.1073/pnas.1420955112
22. Alles J, Fehlmann T, Fischer U, Backes C, Galata V, Minet M, Hart M, Abu-Halima M, Grasser FA, Lenhof HP, Keller A, Meese E (2019) An estimate of the total number of true human miRNAs. Nucleic Acids Res 47(7):3353–3364. https://doi.org/10.1093/nar/gkz097
23. Hou XS, Han CQ, Zhang W (2018) MiR-1182 inhibited metastasis and proliferation of ovarian cancer by targeting hTERT. Eur Rev Med Pharmacol Sci 22(6):1622–1628. https://doi.org/10.26355/eurrev_201803_14569
24. Zhang D, Xiao YF, Zhang JW, Xie R, Hu CJ, Tang B, Wang SM, Wu YY, Hao NB, Yang SM (2015) miR-1182 attenuates gastric cancer proliferation and metastasis by targeting the open reading frame of hTERT. Cancer Lett 360(2):151–159. https://doi.org/10.1016/j.canlet.2015.01.044
25. de Rie D, Abugessaisa I, Alam T, Arner E, Arner P, Ashoor H, Astrom G, Babina M, Bertin N, Burroughs AM, Carlisle AJ, Daub CO, Detmar M, Deviatiiarov R, Fort A, Gebhard C, Goldowitz D, Guhl S, Ha TJ, Harshbarger J, Hasegawa A, Hashimoto K, Herlyn M, Heutink P, Hitchens KJ, Hon CC, Huang E, Ishizu Y, Kai C, Kasukawa T, Klinken P, Lassmann T, Lecellier CH, Lee W, Lizio M, Makeev V, Mathelier A, Medvedeva YA, Mejhert N, Mungall CJ, Noma S, Ohshima M, Okada-Hatakeyama M, Persson H, Rizzu P, Roudnicky F, Saetrom P, Sato H, Severin J, Shin JW, Swoboda RK, Tarui H, Toyoda H, Vitting-Seerup K, Winteringham L, Yamaguchi Y, Yasuzawa K, Yoneda M, Yumoto N, Zabierowski S, Zhang PG, Wells CA, Summers KM, Kawaji H, Sandelin A, Rehli M, Consortium F, Hayashizaki Y, Carninci P, Forrest ARR, de Hoon MJL (2017) An integrated expression

atlas of miRNAs and their promoters in human and mouse. Nat Biotechnol 35(9):872–878. https://doi.org/10.1038/nbt.3947

26. Kleaveland B, Shi CY, Stefano J, Bartel DP (2018) A network of noncoding regulatory RNAs acts in the mammalian brain. Cell 174 (2):350–362. e317. https://doi.org/10.1016/j.cell.2018.05.022
27. Kim B, Jeong K, Kim VN (2017) Genome-wide mapping of DROSHA cleavage sites on primary micro-RNAs and noncanonical substrates. Mol Cell 66(2):258–269. e255. https://doi.org/10.1016/j.molcel.2017.03.013
28. Lu Y, Baras AS, Halushka MK (2018) miRge 2.0 for comprehensive analysis of micro-RNA sequencing data. BMC Bioinformatics 19 (1):275. https://doi.org/10.1186/s12859-018-2287-y
29. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnetjournal 17(1):10–12. https://doi.org/10.14806/ej.17.1.200
30. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10(3):R25. https://doi.org/10.1186/gb-2009-10-3-r25
31. Hackenberg M, Sturm M, Langenberger D, Falcon-Perez JM, Aransay AM (2009) miRanalyzer: a micro-RNA detection and analysis tool for next-generation sequencing experiments. Nucleic Acids Res 37(Web Server issue):W68–W76. https://doi.org/10.1093/nar/gkp347
32. Hackenberg M, Rodriguez-Ezpeleta N, Aransay AM (2011) miRanalyzer: an update on the detection and analysis of micro-RNAs in high-throughput sequencing experiments. Nucleic Acids Res 39(Web Server issue): W132–W138. https://doi.org/10.1093/nar/gkr247
33. Rueda A, Barturen G, Lebron R, Gomez-Martin C, Alganza A, Oliver JL, Hackenberg M (2015) sRNAtoolbox: an integrated collection of small RNA research tools. Nucleic Acids Res 43(W1):W467–W473. https://doi.org/10.1093/nar/gkv555
34. Aparicio-Puerta E, Lebron R, Rueda A, Gomez-Martin C, Giannoukakos S, Jaspez D, Medina JM, Zubkovic A, Jurak I, Fromm B, Marchal JA, Oliver J, Hackenberg M (2019) sRNAbench and sRNAtoolbox 2019: intuitive fast small RNA profiling and differential expression. Nucleic Acids Res 47(W1): W530–W535. https://doi.org/10.1093/nar/gkz415
35. Anders S, Huber W (2010) Differential expression analysis for sequence count data. Genome Biol 11(10):R106. https://doi.org/10.1186/gb-2010-11-10-r106
36. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15(12):550. https://doi.org/10.1186/s13059-014-0550-8
37. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26 (1):139–140. https://doi.org/10.1093/bioinformatics/btp616
38. Tarazona S, Furio-Tari P, Turra D, Pietro AD, Nueda MJ, Ferrer A, Conesa A (2015) Data quality aware analysis of differential expression in RNA-seq with NOISeq R/bioc package. Nucleic Acids Res 43(21):e140. https://doi.org/10.1093/nar/gkv711
39. Galili T, O'Callaghan A, Sidi J, Sievert C (2018) Heatmaply: an R package for creating interactive cluster heatmaps for online publishing. Bioinformatics 34(9):1600–1602. https://doi.org/10.1093/bioinformatics/btx657
40. Conway JR, Lex A, Gehlenborg N (2017) UpSetR: an R package for the visualization of intersecting sets and their properties. Bioinformatics 33(18):2938–2940. https://doi.org/10.1093/bioinformatics/btx364

Chapter 14

RNA Editing Detection in HPC Infrastructures

Claudio Lo Giudice, Luigi Mansi, Tiziano Flati, Silvia Gioiosa, Giovanni Chillemi, Pietro Libro, Tiziana Castrignanò, Graziano Pesole, and Ernesto Picardi

Abstract

RNA editing by A-to-I deamination is a relevant co/posttranscriptional modification carried out by ADAR enzymes. In humans, it has pivotal cellular effects and its deregulation has been linked to a variety of human disorders including neurological and neurodegenerative diseases and cancer. Despite its biological relevance, the detection of RNA editing variants in large transcriptome sequencing experiments (RNAseq) is yet a challenging computational task. To drastically reduce computing times we have developed a novel REDItools version able to identify A-to-I events in huge amount of RNAseq data employing High Performance Computing (HPC) infrastructures.

Here we show how to use REDItools v2 in HPC systems.

Key words HPC, MPI, RNA editing, A-to-I editing, Deep sequencing, Bioinformatics, Genomics, Transcriptomics, RNAseq, DNAseq

1 Introduction

RNA editing is a widespread posttranscriptional phenomenon that alters primary RNA sequences through the insertion/deletion or modification of specific nucleotides [1] and takes place in a variety of organisms including prokaryotes [2], animals [3], plants [4], and viruses [5].

RNA editing by base substitution mainly occurs in plant organelles and especially in mitochondria where specific cytidines (C) are modified in uridines (U) by deamination. In humans, it affects primary transcripts by A-to-I [6] and, to a lesser extent, C-to-U modifications [7].

A-to-I conversion is carried out by members of adenosine deaminases that acts on RNA (ADAR) family [8]. Such enzymes perform the adenosine deamination in double-stranded (ds) RNAs by means of dsRNA binding domains (dsRDBs) and a conserved

C-terminal catalytic domain. To date, numerous A-to-I events have been reported in both coding and noncoding transcript regions [9].

Since inosine is commonly interpreted as guanosine by translation and splicing machineries (other than sequencing enzymes), A-to-I modifications can alter codon identity or base-pairing interactions within higher-order RNA structures. As a result, A-to-I RNA editing can increase the proteome diversity or regulate gene expression at the RNA level. Moreover, editing within pre-mRNAs can generate or destroy splice sites, modulate alternative splicing and influence the dynamics of constitutive splice sites.

The best studied editing events in coding RNAs are those related to neurotransmitter receptors such as the glutamate receptor subunit B (GluR-B) and the serotonin 2C receptor (5-HT2cR). In GluR-B, two editing sites lead to amino-acid substitution with functional consequences. Editing at R/G site regulates the desensitization kinetics of the receptor, whereas the Q/R site regulates the calcium permeability of the ion channel. R/G site editing affects the nucleotide at position −2 of the 5′ splice site of the exon 13 and influences mutually exclusive flop and flip incorporation of exon 14 and exon 15, respectively.

Editing at human neurotransmitter receptors has serious physiological implications. Indeed, its deregulation has been linked to several nervous and neurodegenerative diseases such as epilepsy, schizophrenia, depression, Alzheimer's, and amyotrophic lateral sclerosis. In addition, the functional importance of this mechanism was established by showing that mice lacking ADARs die in utero or soon after weaning [10]. Recently, editing alterations have also been associated to a variety of human cancers. Under-editing at GluR-B Q/R site, for instance, is implicated in malignant gliomas. In addition, RNA editing events can also occur in mature microRNAs and corresponding precursors influencing every step in the miRNA pathway.

Current technologies for massive transcriptome sequencing such as RNAseq (sequencing of entire transcriptomes) are providing accurate maps of transcriptional dynamics occurring in complex eukaryotic genomes as in humans.

The de novo detection of posttranscriptional RNA editing modifications in RNAseq data is a challenging and computationally intensive task. To this aim, a few years ago, we developed the REDItools package. It has been written in the portable python programming language and makes use of the Pysam module (including methods and functions to handle read alignments in SAM/BAM format), a wrapper of widely used SAMtools.

To discover A-to-I changes, REDItools explore all genomic positions supported by RNAseq reads, site by site, invoking the "mpileup" function of the Pysam module. Since this function is executed million times depending on the size of the input genomic regions, it represents the main computational bottleneck.

To speed up the identification of RNA editing changes, REDItools have been revised and optimized to run in HPC infrastructures. In particular, this novel REDItools release (v2) overcomes the mpileup bottleneck, maintaining alignments in memory dynamically, during the traversing of the genome. In this way, only a small set of reads are kept in memory at a given time, avoiding the loading of the same read multiple times. In addition, the new implementation of REDItools is based on the MPI (Message Passing Interface) paradigm with a central MPI dispatcher which iteratively assigns subtasks (genomic intervals) to MPI slave processes, waits for their completion, and finally collects the output into a single tabular file. Optimal genomic intervals are calculated through an ad hoc preprocessing step in order to reduce computational peaks.

REDItools v2 package is freely available at https://github.com/BioinfoUNIBA/REDItools2.

Our tests indicate that this version is on average eight times faster than the previous one on a single sample and the code shows a very good scalability up to the tested 300 cores with multiple samples (Fig. 1).

Hereafter, we describe a bioinformatics protocol to execute the HPC version of REDItools v2 on a human RNAseq experiment.

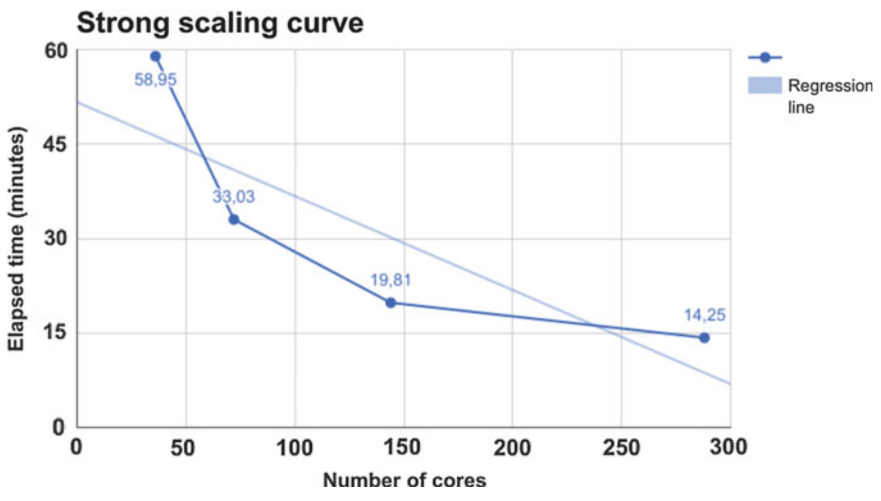


Fig. 1 Relationship between the execution time (in minutes) and the number of cores

2 Materials

REDItools v2 are implemented in Python and require Python 2.7 (*see* **Note 1**) as well as a few modules such as pysam, sortedcontainers, mpi4py, psutil, and netifaces.

2.1 Hardware

The software is compatible with any SLURM-based HPC infrastructure running Linux or UNIX-based operating systems. We recommend each computing node to be equipped with at least 128 GB of RAM for best performances. Several GBs of free disk space are also recommended, according to the size of the samples to be analyzed.

2.2 External Packages and Libraries

Some external packages and libraries are required before the installation of REDItools2:

- htslib (https://github.com/samtools/htslib/).
- SAMtools (https://github.com/samtools/samtools/).
- MPI (REDItools v2 has been tested with OpenMPI (https://www.open-mpi.org/) but other MPI libraries can also be used, such as Intel MPI).
- SRA Toolkit (https://github.com/ncbi/sra-tools).
- FASTP (https://github.com/OpenGene/fastp).
- STAR (https://github.com/alexdobin/STAR).

2.3 Installation of REDItools v2

The latest version of REDItools v2 can be downloaded from https://github.com/BioinfoUNIBA/REDItools2 and installed in the following way:

```
$ git clone https://github.com/BioinfoUNIBA/REDItools2
$ cd reditools2.0
$ pip install -r requirements.txt --user
```

The third command is required because, in addition to the external libraries listed in Subheading 2.2, REDItools v2 require a few Python modules to be installed on the user's machine (pysam, sortedcontainers, mpi4py, etc.). By using the flag `--user` these dependencies will be downloaded and installed only for the current user (usually in its home directory); without the flag, the changes will be at system level (this might require sudo rights).

The Reditools v2 repository also includes two scripts, `prepare_test.sh` and `serial_test.sh` that can be used to run functional tests of the package on a small dataset (chromosome

21, hg19 assembly). First script, downloads both the fasta and the index file of chromosome 21 into a test directory, while the second script runs the software on the reads mapping on it.

2.4 Dataset

REDItools2 accept prealigned RNAseq reads in the standard Binary Alignment Map (BAM) format [11]. In our description we will use an RNAseq dataset (11 GB) and DNAseq dataset (325 GB) from NA12878 cell line that can be obtained from the ENA database under the accession IDs SRR1258218 and ERR262997 respectively.

3 Methods

3.1 Input Data and Quality Check

Download SRR1258218 RNAseq data by Fastq-dump:

```
$ fastq-dump --split-files SRR1258218
```

Check the data quality by FastQC (it calculates a set of quality checks on raw sequence data that are useful for finding potential problems or biases):

```
$ fastqc ../SRR1258218_1.fastq.gz ../SRR1258218_2.fastq.gz
```

Trim RNAseq data by FASTP:

```
$ fastp -i../SRR1258218_1.fastq.gz -I../SRR1258218_2.fastq.gz
-o
out_SRR1258218_1.fastq.gz -O out_SRR1258218_2.fastq.gz -q
25 -u 10 -l
50 -y -x -w 4
```

where `-i` [read1 input file name], `-I` [read1 input file name], `–o` [read1 output file name], `-O` [read2 output file name], `–q` [INT] base phred quality (e.g., 25), `–u` [INT] percentage of bases allowed to be unqualified (e.g., 10, means 10%), `–l` [INT] minimum length, reads below this threshold will be discarded (e.g., 50), `–y` low complexity filter, the complexity is defined as the percentage of a certain base that is different from its next base (base[i]!= base[i +1]), `–x` enable polyX trimming in 3′ ends and `-w` number of working threads.

See **Note 2** for further details about reads trimming and quality check.

3.2 RNAseq Alignment

REDItools require aligned reads in the standard BAM format.

Although a plethora of mapping programs exist, we obtained optimal and reproducible results [12] with STAR [13] (*see* **Note 3** for basic installation of STAR under Linux/Unix systems).

Genome mapping of RNAseq reads is a crucial step because it allows to infer the genomic origin of the reads (*see* **Notes 4–6**). Other tools to perform the alignment of RNA reads onto the reference genome could be used (e.g., GSNAP [14] or HISAT [15]), even though the selected mapping software may affect the RNA editing detection [16]. Additionally, input BAM files need be coordinate-sorted and indexed. These operations can be performed by SAMtools [11], in the following way:

```
$ samtools sort test/$INPUT_BAM_FILE > test/$INPUT_BAM_FILE_-
SORTED
$ samtools index test/$INPUT_BAM_FILE_SORTED
```

REDItools v2 also require the reference sequence of the human genome. It can be downloaded, uncompressed, and indexed by typing:

```
$ wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_hu-
man/release_30/GRCh37_mapping/GRCh37.primary_assembly.genome.
fa.gz
 $ gunzip GRCh37.primary_assembly.genome.fa.gz
$ samtools faidx GRCh37.primary_assembly.genome.fa
```

See **Note 7** for further details about the reference file.

3.3 Running REDItools v2 in Serial Mode

REDItools v2 can be invoked in serial mode by the following command:

```
$ python ./src/cineca/reditools.py -f $INPUT_BAM_FILE -r $REF
ERENCE -o $OUTPUT_FILE
```

If required, the analysis can be restricted only to a certain region (e.g., only chr21) or a specific chromosomal interval (e.g., chr1:1000–2000), by means of the `-g` option:

```
$ mkdir Reditool_DNA_RNA_chr21
$ cd Reditool_DNA_RNA_chr21
$ python ./src/cineca/reditools.py -f $INPUT_BAM_FILE -r PATH/
TO/GRCh37.primary_assembly.genome.fa -o editing_chr21 -g chr21
```

```
$ python ./src/cineca/reditools.py -f $INPUT_BAM_FILE -r PATH/
TO/GRCh37.primary_assembly.genome.fa -o editing_chr21_speci-
fic_region -g chr1:1000-2000
```

where `-f` is the input BAM file, `–r` is the reference in FASTA file, `–o` is the output file, and `-g` is the genomic region to analyze in the format chr:start-end.

For a complete list of all REDItools v2 options, their usage and meaning, please refer to Table 1.

By using REDItools v2 in its basic mode, the user will benefit only from the code optimizations introduced with REDItools v2. While being significantly faster (about a 8x factor), such mode does not take advantage of the possibility of splitting the computational load among multiple CPU cores.

Table 1
List of REDItools2 parameters

Parameter	Parameter	Description
-h	--help	Show this help message and exit
-f	--file	The bam file to be analyzed
-o	--output-file	The output statistics file
-S	--strict	Activate strict mode: only sites with edits will be included in the output
-s	--strand	Strand: this can be 0 (unstranded), 1 (secondstrand oriented), or 2 (firststrand oriented)
-a	--append-file	Appends results to file (and creates if not existing)
-r	--reference	The reference FASTA file
-g	--region	The region of the bam file to be analyzed
-m	--omopolymeric-file	The file containing the omopolymeric positions
-c	--create-omopolymeric-file	Whether to create the omopolymeric span
-os	--omopolymeric-span	The omopolymeric span
-sf	--splicing-file	The file containing the splicing sites positions
-ss	--splicing-span	The splicing span
-mrl	--min-read-length	The minimum read length. Reads whose length is below this value will be discarded
-q	--min-read-quality	The minimum read quality. Reads whose mapping quality is below this value will be discarded

(continued)

Table 1
(continued)

Parameter	Parameter	Description
-bq	--min-base-quality	The minimum base quality. Bases whose quality is below this value will not be included in the analysis
-mbp	--min-base-position	The minimum base position. Bases which reside in a previous position (in the read) will not be included in the analysis
-Mbp	--max-base-position	The maximum base position. Bases which reside in a further position (in the read) will not be included in the analysis
-l	--min-column-length	The minimum length of editing column (per position). Positions whose columns have length below this value will not be included in the analysis
-men	--min-edits-per-nucleotide	The minimum number of editing for events each nucleotide (per position). Positions whose columns have bases with less than min-edits-per-base edits will not be included in the analysis
-me	--min-edits	The minimum number of editing events (per position). Positions whose columns have bases with less than “min-edits-per-base edits” will not be included in the analysis
-Men	--max-editing-nucleotides	The maximum number of editing nucleotides, from 0 to 4 (per position). Positions whose columns have more than “max-editing-nucleotides” will not be included in the analysis
-d	--debug	REDItools is run in DEBUG mode
-T	--strand-confidence	Strand inference type 1:maxValue 2:useConfidence [1]; maxValue: the most prominent strand count will be used; useConfidence: strand is assigned if over a prefixed frequency confidence (-TV option)
-C	--strand-correction	Strand correction. Once the strand has been inferred, only bases according to this strand will be selected
-Tv	--strand-confidence-value	Strand confidence [0.70]
-V	--verbose	Verbose information in stderr
-H	--remove-header	Do not include header in output file
-N	--dna	Run REDItools 2.0 on DNA-Seq data
-B	--bed_file	Path of BED file containing target regions
Other parameter in parallel mode		
-G	--coverage-file	The coverage file of the sample to analyze
-D	--coverage-dir	The coverage directory containing the coverage file of the sample to analyze divided by chromosome
-t	--temp-dir	The temp directory where to store temporary data for this sample.
-Z	--chromosome-sizes	The file with the chromosome sizes

3.4 Running REDItools in Parallel Mode

This modality exploits the existence of multiple cores (also on multiple nodes) offered by High Performance Computing (HPC) facilities. It differs from the previous one since it requires a little bit more system setup, but the setting time is repaid in terms of faster computational performances.

In the parallel mode, REDitools v2 include four additional parameters:

`-G` is the coverage file of the sample to analyze.

`-D` is the coverage directory containing the coverage file of the sample to analyze divided by chromosome.

`-t` is the temp directory where to store temporary data.

`-Z` is the file with the chromosome sizes.

In order to run REDItools v2 in parallel mode, the first step consists in calculating the coverage data for all the samples by means of the `extract_coverage.sh` script:

```
$ mkdir temp
$ mkdir coverage
$ cd coverage
$ mkdir genome_hg19
$ cd genome_hg19
$  wget  ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_hu-
man/release_30/GRCh37_mapping/GRCh37.primary_assembly.genome.
fa.gz
 $ gunzip GRCh37.primary_assembly.genome.fa.gz
$ samtools faidx GRCh37.primary_assembly.genome.fa
$ cd ../../
$ ./extract_coverage.sh $INPUT_BAM_FILE_SORTED \
 coverage/genome_hg19/GRCh37.primary_assembly.genome.fa.fai
$ mpirun -np 2 src/cineca/parallel_reditools.py \
 -f SRR1258218_Aligned.sortedByCoord.out.bam \
 -r genome_hg19/GRCh37.primary_assembly.genome.fa \
 -t temp/ \
 -Z genome_hg19/GRCh37.primary_assembly.genome.fa.fai \
 -G coverage/SRR1258218.cov -D coverage/
```

Finally, in order to obtain a unique output, run the. /merge,sh script:

```
$. /merge.sh temp/ parallel_table.txt.gz 2
```

3.5 Running REDItools v2 in HPC Mode

The configuration required for running REDItools v2 in a HPC environment is slightly more complicated. The script parallel_slurm.sh contained in the GitHub repository provides a template which can be filled to run REDItools v2 in a SLURM-based HPC environment.

The parallel script can be schematized in five subsections as shown in Fig. 2:

1. Parameters settings (input BAM file, coverage dir, etc.).
2. Modules loading (e.g., module load samtools).
3. Coverage analysis (by executing the extract_coverage_dynamic.sh script).

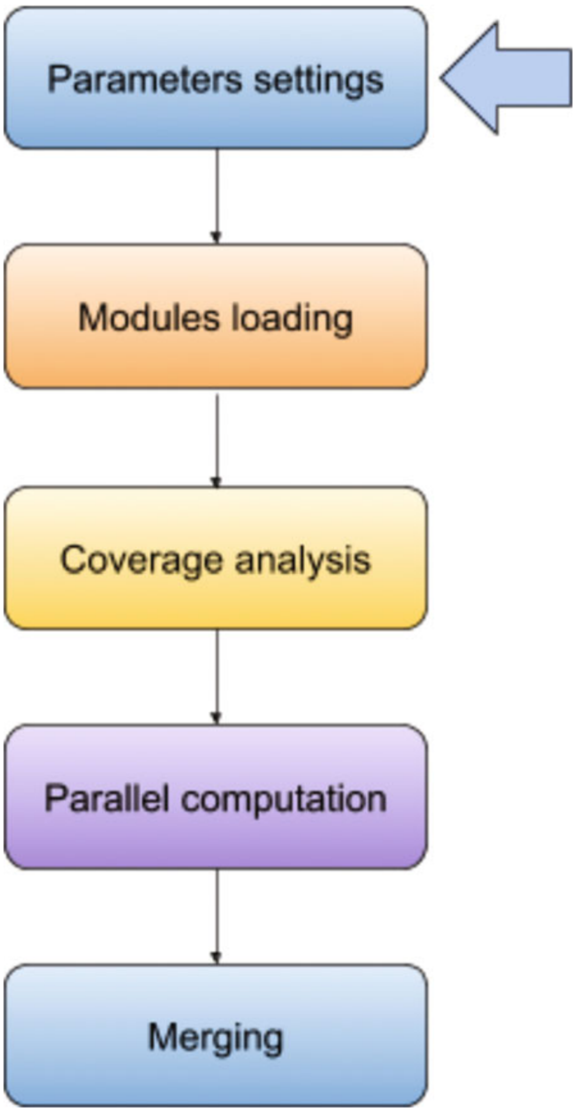


Fig. 2 Scheme of the main five subsections of the script parallel_slurm.sh. The blue arrow indicates the only subsection that should be configured by the user

4. Parallel computation (by executing the parallel_reditools.py script).
5. Merging (by executing the merge.sh script).

Before launching REDItools v2 in HPC mode, the user needs to configure the "Parameter settings" subsection of script parallel_-slurm.sh.

Below, we report its code as well as comments about the meaning of required variables.

```
#########################################################
######## Parameters setting ########
#########################################################

##SAMPLE_ID is the basename of the input sample
SAMPLE_ID="SRR1258218"




##input BAM file
SOURCE_BAM_FILE="PATH/TO/SRR1258218.bam"




##reference chromosome or genome
REFERENCE ="PATH/TO/genome_hg19/GRCh37.pri-
mary_assembly.genome.fa
REFERENCE_DNA=$(basename "$REFERENCE")

##fasta index file created by samtools
SIZE_FILE="PATH/TO/genome_hg19/GRCh37.primary_assembly.gen-
ome.fa.fai"




##number of utilized cores
NUM_CORES=2

##set the output file and the temporary directory
OUTPUT_FILE="results/output/parallel_table.txt.gz"
TEMP_DIR="results/temp/"

##set the coverage file
COVERAGE_FILE="results/coverage/SRR1258218.cov"
```

```
##set the coverage directory
COVERAGE_DIR="results/coverage/"

##set the output directory
OUTPUT_DIR=$(basename "$OUTPUT_FILE")

#############################################################-
#################
```

.

Once all needed parameters have defined and saved, the parallel script can be launched as a bash script in the HPC environment by the following command:

```
$ sbatch. /parallel_slurm.sh
```

3.6 REDItools 2.0 in Combination with Whole Genome Sequencing (Optional)

REDItools v2 is able to handle RNAseq data alone or combine RNAseq and mapped genomic reads from DNAseq experiments to minimize the false discovery rate due to single nucleotide polymorphisms (SNPs).

In order to use DNAseq data with REDItools v2 two preliminary steps are required.

The first one consists in analyzing RNAseq data (e.g., file *rna.bam*) and obtaining the corresponding output table (e.g., rna_table.txt or gzipped rna_table.txt.gz) as previously described. The second step, instead, requires the processing of prealigned genomic reads (e.g., dna.bam) with REDItools v2, providing as input:

```
-the DNAseq BAM file (dna.bam) (e.g., option -f dna.bam);
```

See **Note 8** for details about how to obtain the DNAseq BAM file using BWA;

```
-the RNA-table output from the previous step (e.g., option -B
rna_table.txt);
```

Last step will produce an intermediate output table (e.g., dna_-table.txt). Finally, annotate the RNAseq table by means of the DNAseq table by running REDItools v2 annotator script with the two tables as input (e.g. *rna_table.txt* and *dna_table.txt*).

```
$ src/cineca/annotate_with_DNA.py -r $rna_table.txt -d $dna_-
table.txt -r $reference.fai [-Z]
```

where.

`-r` is the RNA-editing events table to be annotated.

`-d` is the RNA-editing events table as obtained from DNAseq data.

`-R` is the .fai file of the reference genome.

`-Z` is an optional flag to skip positions with multiple changes in DNA-Seq.

The resulting table (e.g., *final_table.txt*), can therefore be used for the downstream analyses.

3.7 Output

REDItools v2 output a table comprises the following fields:

- **Region**: is the genomic region according to the reference genome.
- **Position**: is the exact genomic coordinate (1-based).
- **Reference**: is the nucleotide base in the reference genome.
- **Strand**: is the strand info with notation 1 for + strand, 0 for – strand and 2 for unknown or not defined strand.
- **Coverage-qxx**: is the depth per site at a given xx quality score.
- **MeanQ**: is the mean quality score per site.
- **BaseCount**[A,C,G,T]: is the base distribution per site in the order A, C, G and T.
- **AllSubs**: is the list of observed substitutions at a given site, separated by a space. A character "-" is included in case of invariant sites.
- **Frequency**: is the observed frequency of substitution. In case of multiple substitutions, it refers to the first in the AllSubs field.
- **gCoverage-qxx**: is the depth per site at a given xx quality score in DNA-Seq.
- **gMeanQ**: is the mean quality score per site in DNA-Seq.
- **gBaseCount**[A,C,G,T]: is the base distribution per site in the order A, C, G, and T.
- **gAllSubs**: is the list of observed substitutions at a given site, separated by a space. A character "-" is included in case of invariant sites.
- **gFrequency**: is the observed frequency of substitution. In case of multiple substitutions, it refers to the first in the gAllSubs field.

An example of output table is shown in Table 2.

Table 2
Example of a REDItool output table

Region	Position	Reference	Strand	Coverage-q25	MeanQ	BaseCount [A,C,G,T]	AllSubs	Frequency	gCoverage-q25	gMeanQ	gBaseCount [A,C,G,T]	gAllSubs	gFrequency
chr21	15412990	A	1	18	37.22	[3, 0, 15, 0]	AG	0.83	18	37.22	[3, 0, 15, 0]	AG	0.83
chr21	15415901	A	1	13	37.15	[2, 0, 11, 0]	AG	0.85	13	37.15	[2, 0, 11, 0]	AG	0.85
chr21	15423330	A	1	11	38.27	[4, 0, 7, 0]	AG	0.64	11	38.27	[4, 0, 7, 0]	AG	0.64
chr21	15425640	A	1	8	36.12	[0, 0, 8, 0]	AG	1.00	8	36.12	[0, 0, 8, 0]	AG	1.00
chr21	15456434	T	1	90	34.96	[0, 6, 1, 83]	TC TG	0.07	90	34.96	[0, 6, 1, 83]	TC TG	0.07
chr21	15461406	A	1	83	37.27	[73, 0, 10, 0]	AG	0.12	83	37.27	[73, 0, 10, 0]	AG	0.12
chr21	15461417	A	1	90	36.26	[72, 0, 18, 0]	AG	0.20	90	36.26	[72, 0, 18, 0]	AG	0.20
chr21	15461444	A	1	64	37.22	[26, 0, 38, 0]	AG	0.59	64	37.22	[26, 0, 38, 0]	AG	0.59
chr21	15461479	A	1	70	36.96	[66, 0, 4, 0]	AG	0.06	70	36.96	[66, 0, 4, 0]	AG	0.06
chr21	15461486	A	1	68	37.06	[61, 0, 7, 0]	AG	0.10	68	37.06	[61, 0, 7, 0]	AG	0.10
chr21	15461503	A	1	76	37.26	[69, 0, 7, 0]	AG	0.09	76	37.26	[69, 0, 7, 0]	AG	0.09
chr21	15461511	A	1	81	37.68	[55, 0, 26, 0]	AG	0.32	81	37.68	[55, 0, 26, 0]	AG	0.32

3.8 Downstream Analyses

RNA editing candidates reported in REDItools tables can be further analyzed using the ad hoc protocol described in [12] and [16]. In addition, REDItools tables from different experimental conditions can also be compared to detect differential RNA editing using accessory scripts described in [12].

4 Notes

1. REDItools are designed to work with python versions 2.7 and superior. You can check the python installation on your computer as well as the release version using the command:

```
$ python -version
```

2. Quality check and trimming are strongly recommended, since they increase the mapping rate and the alignment quality, two factors that directly influence the yield and accuracy of the final RNA editing results.
3. In order to install STAR on Unix/Linux systems check the latest version from https://github.com/alexdobin/STAR and execute the following commands:

```
$ wget https://github.com/alexdobin/STAR/archive/2.7.3a.tar.
gz
$ tar -xzf 2.7.3a.tar.gz
$ cd STAR-2.7.3a
```

   Alternatively, STAR source can be obtained from its GitHub repository by

```
$ git clone https://github.com/alexdobin/STAR.git
```

   and compiled with

```
$ cd STAR/source
$ make STAR
```

4. A list of known annotated transcript can be obtained from GENCODE website at http://www.gencodegenes.org/.

   For the hg19 genome assembly, the annotation table can be retrieved from ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_19/gencode.v19.annotation.gtf.gz.

and uncompressed with

```
$ gunzip gencode.v19.annotation.gtf.gz
```

5. STAR indexes for human genome hg19 can be generated using the command:

```
$ mkdir STAR_genome_index
$ STAR --runMode genomeGenerate --genomeDir STAR_genome_index
--genomeFastaFiles PATH/TO/genome_hg19/GRCh37.primary_assem-
bly.genome.fa
--sjdbGTFfile PATH/TO/gencode.v19.annotation.gtf
```

6. Align RNASeq reads to the reference genome with STAR:

```
$ mkdir Alignment
$ cd Alignment/
$ STAR --runThreadN 4 --genomeDir PATH/TO/STAR_genome_index --
outFileNamePrefix SRR1258218_ --outSAMtype BAM SortedByCoor-
dinate --outSAMattributes All --readFilesCommand zcat --read-
FilesIn PATH/TO/out_SRR1258218_1.fastq.gz PATH/TO/
out_SRR1258218_2.fastq.gz
```

7. Before launching REDItools script, check that chromosome/region names in the reference file are the same contained in the header of the input BAM file.
8. In order to obtain the DNAseq BAM file described in Subheading 3.6, download NA12878 WGS reads:

```
$ mkdir WGS_ERR262997
$ cd WGS_ERR262997
$ fastq-dump --split-files ERR262997
```

Download and install BWA:

```
$ git clone https://github.com/lh3/bwa
$ cd bwa; make
```

Using BWA is not mandatory; other packages such as BOWTIE [17], GSNAP [14] or SOAP2 [18] can perform the same task with similar results.

Build a reference genome index for BWA:

```
$ bwa index PATH/TO/genome_hg19/GRCh37.primary_assembly.gen-
ome.fa
```

The reference sequence to be indexed must be the same downloaded in the Subheading 3.2.

Align NA12878 WGS reads to the reference genome with BWA-MEM:

```
$ bwa mem -t 24 PATH/TO/genome_hg19/GRCh37.primary_assembly.
genome.fa -Y ERR262997_1.fastq ERR262997_2.fastq > ERR262997.
sam
```

where

`-t` [INT] is the "Number of threads" on the user's machine.

`-Y` indicates the "Use soft clipping for supplementary alignments."

By default, BWA-MEM uses soft clipping for the primary alignment and hard clipping for supplementary alignments.

BWA outputs alignments in SAM format; thus, a supplementary step by SAMtools is needed to convert them in a sorted BAM file compatible with REDItools v2:

```
$ samtools view -b ERR262997.sam > ERR262997.bam
$ samtools index ERR262997.bam
```

Acknowledgments

This work was supported by ELIXIR-IIB and PRACE projects 2016163924 and 2018194670.

References

1. Gott JM, Emeson RB (2000) Functions and mechanisms of RNA editing. Annu Rev Genet 34:499–531. https://doi.org/10.1146/annurev.genet.34.1.499
2. Bar-Yaacov D, Pilpel Y, Dahan O (2018) RNA editing in bacteria: occurrence, regulation and significance. RNA Biol 15:863–867. https://doi.org/10.1080/15476286.2018.1481698
3. Porath HT, Knisbacher BA, Eisenberg E, Levanon EY (2017) Massive A-to-I RNA editing is common across the Metazoa and correlates with dsRNA abundance. Genome Biol 18:185. https://doi.org/10.1186/s13059-017-1315-y
4. Takenaka M, Zehrmann A, Verbitskiy D, Härtel B, Brennicke A (2013) RNA editing in plants and its evolution. Annu Rev Genet 47:335–352. https://doi.org/10.1146/annurev-genet-111212-133519
5. Pfaller CK, Donohue RC, Nersisyan S, Brodsky L, Cattaneo R (2018) Extensive editing of cellular and viral double-stranded RNA

structures accounts for innate immunity suppression and the proviral activity of ADAR1p150. PLoS Biol 16:e2006577. https://doi.org/10.1371/journal.pbio.2006577

6. Picardi E, Manzari C, Mastropasqua F, Aiello I, D'Erchia AM, Pesole G (2015) Profiling RNA editing in human tissues: towards the inosinome atlas. Sci Rep 5:1–17. https://doi.org/10.1038/srep14941
7. Lerner T, Papavasiliou FN, Pecori R (2018) RNA editors, cofactors, and mRNA targets: an overview of the C-to-U RNA editing machinery and its implication in human disease. Genes 10. https://doi.org/10.3390/genes10010013
8. Nishikura K (2010) Functions and regulation of RNA editing by ADAR Deaminases. Annu Rev Biochem 79:321–349. https://doi.org/10.1146/annurev-biochem-060208-105251
9. Nishikura K (2016) A-to-I editing of coding and non-coding RNAs by ADARs. Nat Rev Mol Cell Biol 17:83–96. https://doi.org/10.1038/nrm.2015.4
10. Vesely C, Tauber S, Sedlazeck FJ, Tajaddod M, von Haeseler A, Jantsch MF (2014) ADAR2 induces reproducible changes in sequence and abundance of mature microRNAs in the mouse brain. Nucleic Acids Res 42:12155–12168 . https://doi.org/10.1093/nar/gku844
11. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R (2009) The sequence alignment/map format and SAMtools. Bioinformatics 25:2078–2079. https://doi.org/10.1093/bioinformatics/btp352
12. Giudice CL, Tangaro MA, Pesole G, Picardi E (2020) Investigating RNA editing in deep transcriptome datasets with REDItools and REDIportal. Nat Protoc 15:1098–1131. https://doi.org/10.1038/s41596-019-0279-7
13. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29:15–21. https://doi.org/10.1093/bioinformatics/bts635
14. Wu TD, Reeder J, Lawrence M, Becker G, Brauer MJ (2016) GMAP and GSNAP for genomic sequence alignment: enhancements to speed, accuracy, and functionality. Methods Mol Biol 1418:283–334. https://doi.org/10.1007/978-1-4939-3578-9_15
15. Kim D, Langmead B, Salzberg SL (2015) HISAT: a fast spliced aligner with low memory requirements. Nat Methods 12:357–360. https://doi.org/10.1038/nmeth.3317
16. Lo Giudice C, Silvestris DA, Roth SH, Eisenberg E, Pesole G, Gallo A, Picardi E (2020) Quantifying RNA editing in deep Transcriptome datasets. Front Genet 11:194. https://doi.org/10.3389/fgene.2020.00194
17. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol 10:R25. https://doi.org/10.1186/gb-2009-10-3-r25
18. Li R, Yu C, Li Y, Lam T-W, Yiu S-M, Kristiansen K, Wang J (2009) SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics 25:1966–1967. https://doi.org/10.1093/bioinformatics/btp336

Identification of Genes Post-Transcriptionally Regulated from RNA-seq: The Case Study of Liver Hepatocellular Carcinoma

Stefano de Pretis, Mattia Furlan, and Mattia Pelizzola

Abstract

The field of transcriptional regulation generally assumes that changes in transcripts levels reflect changes in transcriptional status of the corresponding gene. While this assumption might hold true for a large population of transcripts, a considerable and still unrecognized fraction of the variation might involve other steps of the RNA lifecycle, that is the processing of the premature RNA, and degradation of the mature RNA. Discrimination between these layers requires complementary experimental techniques, such as RNA metabolic labeling or block of transcription experiments. Nonetheless, the analysis of the premature and mature RNA, derived from intronic and exonic read counts in RNA-seq data, allows distinguishing between transcriptionally and post-transcriptionally regulated genes, although not recognizing the specific step involved in the post-transcriptional response, that is processing, degradation, or a combination of the two. We illustrate how the INSPEcT R/Bioconductor package could be used to infer post-transcriptional regulation in TCGA RNA-seq samples for Hepatocellular Carcinoma.

Key words RNA-seq, Post-transcriptional regulation, Transcriptional regulation, Cancer, Hepatocellular carcinoma

1 Introduction

High-throughput sequencing is nowadays considered the standard routine for profiling RNA transcription processes. Generally, transcripts abundance is regarded as a direct measurement of the transcriptional activity of a gene, and changes in RNA levels among different conditions are attributed to changes in the promoter activity. In practice, RNA transcripts undergo a continuous process of synthesis, processing, and decay that collectively determine the levels of premature and mature RNAs. As a consequence, changes in premature RNA occur due to a change in either the synthesis or the processing rates, while changes in the mature RNA result from the regulation of either synthesis or degradation rates [1]. While assuming that changes in transcripts levels reflect changes in the transcriptional status might hold true for a large population of transcripts, a considerable and still unrecognized fraction of the variation might occur due different layers involved in the RNA life cycle. In order to measure gene activity at this level of resolution, RNA-seq needs to be complemented with further experiments, such the profiling of nascent RNA or block of transcription assays, which directly measure the rates of synthesis and degradation of RNAs, respectively. These techniques are considered technically difficult and substantially increase sequencing costs. For these reasons, different methods have been devised to increase the information obtained from standard RNA-seq experiments by integrating the expression of mature transcripts with premature, quantified through the RNA-seq signal associated to introns. By introducing different assumptions or simplifications, those techniques bring up the information of the life cycle of RNAs within expression analysis. In particular, assuming that processing rates are invariant among conditions, EISA-seq estimates changes in the transcriptional activity of a gene from changes in intronic expression and changes in the degradation rates as the difference between the variation of the intronic and the exonic parts [2]. Another tool, Rembrandts, modified this approach by imposing a coupling factor between the variation in synthesis and processing rates that reflects a hypothetical saturation of the splicing machinery [3]. INSPEcT-, which was recently proposed by us [4], identifies post-transcriptionally regulated genes without imposing a-priori any limit to the variation of the processing rates. At steady state, INSPEcT- relies on the fact that the ratio between the mature and the premature RNA expression is indicative of the ratio between the rates of processing and degradation rates (post-transcriptional ratio, PT ratio), and that a change in the rate of processing or degradation will result into a change of the PT-ratio. Moreover, INSPEcT- increases the performance in the identification of truly post-transcriptionally regulated genes by modeling the dependence between the PT-ratio and the gene expression in a dataset specific fashion. In this chapter we illustrate how we applied this approach to TCGA RNA-seq samples relative to Hepatocellular Carcinoma (HCC) dataset, in order to identify genes whose post-transcriptional regulation might be involved in the carcinogenesis process. Briefly, we downloaded RNA-seq coverage information for 424 HCC samples and applied INSPEcT- to quantify the intronic and exonic signals, to estimate sample specific PT-ratio, and to identify post-transcriptionally regulated genes between different samples. We observed that normal samples have higher PT-ratios compared to primary tumor samples, reflecting either a more efficient splicing in the normal tissues or an increased degradation of the mature RNAs in the tumor samples.

Additionally, we identified several post-transcriptionally regulated genes that are able to stratify between normal and tumor samples and possibly between different tumor subtypes.

2 Materials

2.1 RNA-Seq Dataset

Intronic and exonic counts for 424 TCGA RNA-seq samples of Hepatocellular Carcinoma (HCC) were derived from BigWig files generated by the Recount project [5]. The Recount project offers the reanalysis of human RNA-seq datasets available in SRA, GTEx, and TCGA, ensuring that the same, traceable pipeline is used for all the analyzed samples. The TCGA HCC dataset is composed by 50 normal, 371 primary tumors, and 3 recurrent tumors samples. Mapped read counts per sample range from 50 to 300 million reads, with first and third quartiles assessed at 107 and 140 million reads, respectively. Reads were aligned to the human reference genome GRCh38.

2.2 Computational Hardware

The demanding part of the computational analysis concerns the download of the TCGA BigWig files and the quantification of intronic and exonic counts, which might take hours of computational time. The amount of data required to be downloaded from the Recount servers amounts to ~50 GB, which is required to be stored in the local disk. The time of download depends upon the network resources and about 2 h with cable internet connection (*see* **Note 1**). The quantification of intronic and exonic counts from the 424 BigWig takes about 4 h of computational time on a single processor but could be speed up with parallelization. The requirements in terms of memory usage increase with the number of cores used for the parallelization.

2.3 Computational Software

The complete analysis is performed within the R/Bioconductor environment. R (version $\geq$ 4.0.0) should be downloaded from the CRAN website (https://cran.r-project.org/) and installed on the local machine. Bioconductor (version $\geq$ 3.11) can be installed from the R shell using the following commands:

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.11")
```

For the sake of the analysis it is required to install specific packages, such as ‘recount’ for the raw-data and meta-data retrieval, ‘TxDb.Hsapiens.UCSC.hg38.knownGene’ for the genomic annotation of the human genome, ‘INSPEcT’ for the quantification of the intronic and exonic gene features and the post-transcriptional regulation analysis, and ‘pheatmap’ for results visualization.

```
BiocManager::install(c('recount','TxDb.Hsapiens.UCSC.hg38.knownGene',
                       'INSPEcT','pheatmap'))
```

For the survival analysis it is required to install the packages ‘survival’ and ‘survminer’.

```
BiocManager::install(c('survival','survminer'))
```

For regulatory factors enrichment analysis via the AURA database [6], the package ‘org.Hs.eg.db’ is required for the conversion of the gene names from Entrez ID to Gene Symbols.

```
BiocManager::install('org.Hs.eg.db')
```

3 Methods

3.1 Download Metadata and Raw Sequencing Data Relative to TCGA Samples

In order to download metadata and BigWig files relative to TCGA samples, we take advantage of the R/Bioconductor package “recount.” The Recount project reanalyzed several human RNA-seq datasets, including the entire TCGA, using a standardized pipeline. Results are readily available from their webserver or from the associated R/Bioconductor package, in the form of read counts or raw data. Read counts have been computed on several features, namely, exons, junctions, transcripts, or genes. For each study, associated metadata is also reported. For our purpose, we also need the counts associated to introns. For this reason, we start our analysis from raw dataset, which Recount made available in the form of BigWig files, containing RNA-seq reads coverage throughout the whole genome. The function “download_study” from the recount package can be used to download the desired data type from a specific project. In this case, the argument “type” is set to “sample,” pointing to BigWig files, and “project” is set to “TCGA.” We also set “download” to FALSE, in order to retrieve only the URLs were the BigWig files are stored. In this way, we are able to download only specific TCGA samples, and not the whole TCGA RNA-seq raw dataset (~ 2 TB of data).

```
library(recount)
metadata <- all_metadata(subset = "tcga", verbose = TRUE)
samples_url <- download_study(project = 'TCGA',
                              type = 'samples',
                              download = FALSE)
```

We decided to focus on TCGA samples relative to a specific tumor type, Liver Hepatocellular Carcinoma (HCC). Using the metadata field named “cgc_file_disease_type”, we select the URLs relative to 424 Liver HCC samples and retrieve the relative TCGA metadata.

```
hcc_samples <- which(metadata$cgc_file_disease_type ==
                       'Liver Hepatocellular Carcinoma')
hcc_urls <- samples_url[hcc_samples]
hcc_metadata <- metadata[hcc_samples,]
```

We then download the samples from the selected URLs using the built-in R function "download.file". First of all, we suggest creating a local directory that will contain all the BigWig files. Additionally, we create the BigWig destination file names using the basename of the URLs to be downloaded.

```
dir.create('BigWigFiles', showWarnings = FALSE)
hcc_destfiles <- file.path('BigWigFiles', basename(hcc_urls))
```

In the case of liver HCC samples, the amount of data to be downloaded amounts to about 50 GB. We suggest ensuring the a-vailability of such space on the local disk before starting the download. It is possible to download each file individually, or the entire set in parallel. The latter case is fast but might originate errors in the download of individual files with poor control over the samples that failed. For this reason, we implemented a code that download chunks of files and repeat the download of individual chunks in case of error in the retrieval of at least one file. We suggest using a chunk size of 10–50 samples. This procedure might take up to 1–2 h of time.

```
chunk_size <- 10
download_chunks <- ceiling(seq(hcc_urls)/chunk_size)
download_outcomes <- rep(1, max(download_chunks))
while( any(download_outcomes != 0) ) {
  for( i in which(download_outcomes != 0) ) {
    chunk_indexes <- which(download_chunks==i)
    download_outcomes[i] <- download.file(hcc_urls[chunk_indexes],
                                          hcc_destfiles[chunk_indexes],
                                          quiet=TRUE)
  }
}
```

3.2 Quantification of Intronic and Exonic Expressions from Raw Sequencing Data

Once BigWig files are stored locally, the R/Bioconductor package INSPEcT can be used to quantify read counts and expressions at the level of exonic and intronic annotations via the routine "quantifyExpressionsFromBWs" (*see* **Note 2**). The function requires a genome-annotation database (in the form of a TxDb object), the paths to the BigWig files, and the experimental design, that is a vector that defines the samples belonging to the same experimental condition. In this case we want to analyze each sample separately, therefore the experimental design reflects the names of the samples.

The TxDb object must be specific to the genome assembly used to create the BigWig files, in this case "hg38" (*see* **Note 3**).

```
library(INSPEcT)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
sampleNames <- sub('.bw$','',basename(hcc_urls))
liverExprs <- quantifyExpressionsFromBWs(txdb = txdb,
                                          BWfiles = hcc_destfiles,
                                          experimentalDesign = sampleNames)
saveRDS(liverExprs, file='liverExprs.rds')
```

This procedure takes about 4 h of computational time on machine with a 2.6 GHz 6-Core Intel Core i7 processor. In order to reduce the computational time, it is possible to parallelize the procedure by providing the argument "BPPARAM" to the function "quantifyExpressionsFromBWs." For example, to run the procedure with two threads, the argument 'BPPARAM' must be set to 'MulticoreParam(2)'. The more threads are used, the higher resources in terms of memory are consumed. In fact, each thread loads the raw file relative to the sample under analysis, therefore multiple threads reflect in multiple raw files loaded on the memory.

3.3 Analysis of the Post-transcriptional Ratio in Different Sample Types

The package INSPEcT also provides specific routines to analyze the post-transcriptional regulation patterns among the different samples. First of all, we use the "newINSPEcT" function to create an object of class INSPEcT starting from the quantified intronic and exonic expression of HCC samples. The routine checks the input data for some basic requirements. In particular, genes without intronic quantification are filtered out, because the intronic signal is a requisite for the following analyses. Following this, genes that in at least one condition had an intronic quantification greater than the exonic one are discarded due to inconsistency of the data. Lastly, lowly expressed gens with zero counts in more than 2/3 of the samples are discarded. Noteworthily, the latter filter can be suppressed by setting the "genesFilter" argument to FALSE, while all other filters are performed automatically by the routine.

```
inspectLiver <- newINSPEcT(sampleNames, matureExpressions = liverExprs)
```

As a first analysis we investigate if the median ratio between premature and mature RNA is varying among the HCC TCGA samples. In RNA-seq experiments, the expression levels represent the average of many (typically asynchronous) cells, which can be collectively considered at steady state. In this condition, premature RNA levels for individual genes correspond to the ratio between their average rate of synthesis (i.e., rate of transcription) and their average rate of processing. Analogously, mature RNA levels correspond to the ratio between the average rate of transcription and the average rate of degradation. As a consequence, the ratio between the mature and the premature RNA levels equals the ratio between the processing and the degradation rates (i.e., post-transcriptional ratio, or PT-ratio). Thus, a change in the ratio between mature and premature RNAs reflects a change in the rates of either one or the other process, which might have important physiological and clinical implications.

```
pt_ratio <- PTratio(inspectLiver)
```

In the following analysis, we will stratify the results based on the sample type. The TCGA HCC dataset includes 50 normal, 371 primary, and 3 recurrent tumors samples. We rearrange sample types provided by the TCGA in a “factor” variable with levels sorted according to their supposed temporal order, that is, normal, primary tumor, and recurrent tumor (*see* **Note 4**). This will facilitate the following data visualization.

```
sample_type <- hcc_metadata$cgc_sample_sample_type
sample_type <- factor(sample_type,
                      levels = c('Solid Tissue Normal',
                                 'Primary Tumor',
                                 'Recurrent Tumor'))
table(sample_type)
```

We calculate the median PT-ratio per each sample and group them by sample type. The boxplot and the trend line evaluated on the medians of the groups show a linear decrease of the PT-ratio during the progression from normal to primary tumor, and later to recurrent tumor (Fig. 1).

```
pt_ratio_median <- apply(pt_ratio, 2, median, na.rm=TRUE)
pt_ratio_median_per_type <- split(pt_ratio_median, sample_type)
names(pt_ratio_median_per_type) <-
  gsub(' ', '\n', names(pt_ratio_median_per_type))
boxplot(pt_ratio_median_per_type, varwidth=T, notch=T, las=2,
        ylab = 'post-transcriptional ratio',
        main='Sample medians, grouped by type')
abline(lm(sapply(pt_ratio_median_per_type, median) ~ seq(1,3)), col='red')
```

To assess a statistical difference among the groups, we compare the distributions by means of the ‘Two-sample Wilcoxon test’. This test does not assume the normality of the distributions and can be correctly applied to our data. The following barplot shows that the greatest statistical difference is given when normal and primary tumor samples are compared (Fig. 2). Comparisons that involve recurrent tumors have lower statistical power due to the limited

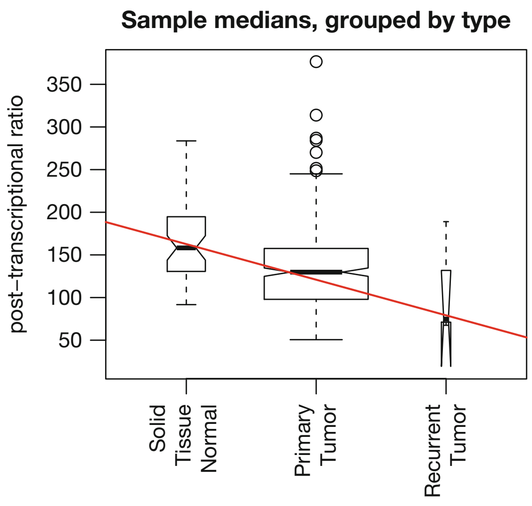


Fig. 1 Boxplot representing the median PT-ratio by sample, grouped by the TCGA sample type. The trend line, in red, is evaluated based on the group medians

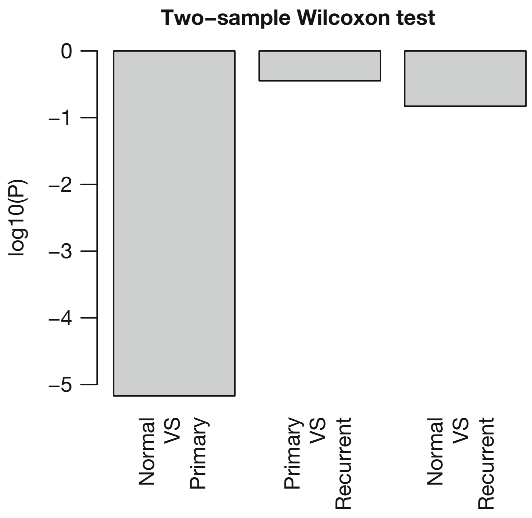


Fig. 2 Results of the 'Two-sample Wilcoxon test' that compared the median PT-ratio by sample grouped by sample type. The comparisons are named in the x-axis of the barplot, which depicts log10 transformed p-value of the tests results

sample size ($N = 3$).

```
barplot(log10(c(
  'Normal\nVS\nPrimary'   = wilcox.test(pt_ratio_median_per_type[[1]],
pt_ratio_median_per_type[[2]])$p.value,
  'Primary\nVS\nRecurrent'= wilcox.test(pt_ratio_median_per_type[[2]],
pt_ratio_median_per_type[[3]])$p.value,
  'Normal\nVS\nRecurrent' = wilcox.test(pt_ratio_median_per_type[[1]],
pt_ratio_median_per_type[[3]])$p.value
)), ylab = 'log10(P)', main = 'Two-sample Wilcoxon test', las=2)
```

It is also possible to exploit the metadata provided by the recount package in order to perform a survival analysis. In particular, we show that individuals with higher median PT-ratio have a worst prognosis compared to individuals with lower median PT-ratio (Fig. 3).

```
library(survival)
library(survminer)
hcc_survdata <- data.frame(
  time = apply(cbind(hcc_metadata$xml_days_to_last_followup,
                     hcc_metadata$xml_days_to_death), 1, sum, na.rm=T),
  status = as.numeric(hcc_metadata$xml_vital_status),
  high_pt_ratio = pt_ratio_median > 155
)
ggsurvplot(
  fit = survfit(Surv(time, status) ~ high_pt_ratio, data = hcc_survdata),
  xlab = "Days",
  ylab = "Overall survival probability",
  pval = TRUE)
```

3.4 Identification of Post-Transcriptionally Regulated Genes

The package INSPEcT is also able to identify individual genes that strongly differ from a null model that is expected when the sole change of the rate of transcription occurs. This task is performed by the method "compareSteadyNoNascent," which identifies the expected trend as the linear model that better explains the relationship between premature and mature RNA gene medians within the dataset under analysis. Genes that deviate from this trend at gene level, in specific samples are highlighted as post-transcriptionally regulated. The inferred trend can be visualized using the method "plotPMtrend" (Fig. 4). Alternatively, it is possible to impose the trend (the slope of the model) between premature and mature RNA with the argument "trivialAngle." Other arguments define the expression threshold that premature and mature RNA should pass for the gene to be considered expressed ("expressionThreshold"), the log2 fold change threshold from the null model that a gene should pass to be defined post-transcriptionally regulated ("log2FCThreshold"), and the condition to be used as reference ("referenceCondition"). The reference condition is used to define, gene by gene, the intercept of the linear model. In case no reference condition is defined, the median premature and mature RNA levels

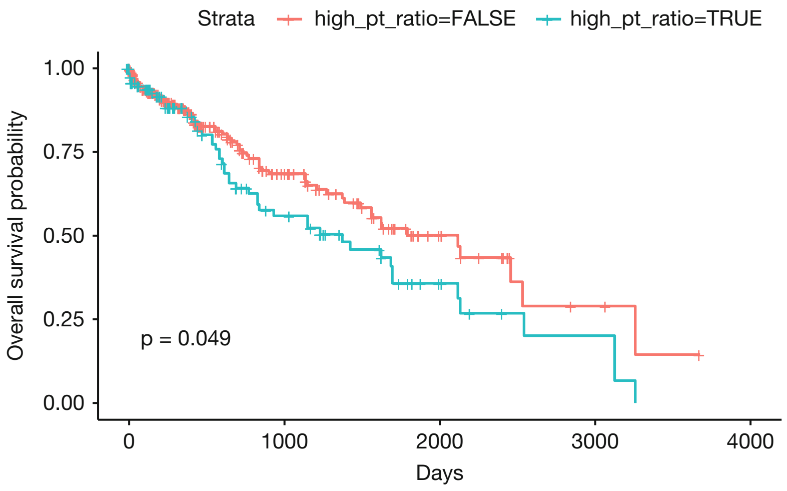


Fig. 3 Kaplan–Meier plot showing the survival analysis of patients diagnosed with HCC stratified for median high or low median PT-ratio

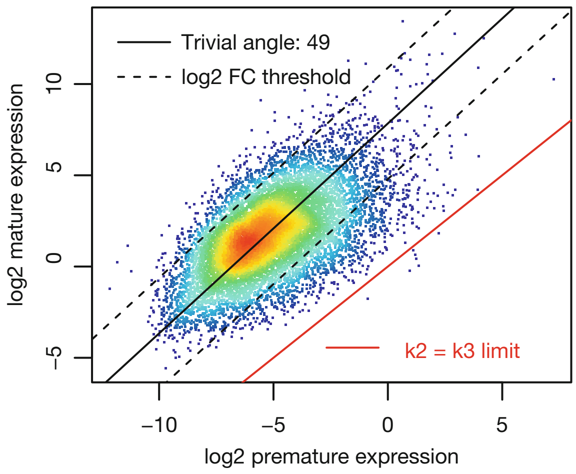


Fig. 4 Dotplot representing the premature and mature median gene expression among the 424 HCC samples. The points are colored according to their density in the plot, in a scale from blue to yellow to red. The black lines represent the estimated trend between mature and premature medians, which will be used to infer post-transcriptionally regulated genes. The red line represents the part of the graph where premature equal mature expressions, and therefore the processing rates (k2) equal degradation rates (k3). Above the red line k2 $>$ k3, which is the common case

among samples is used.

```
inspectLiver <- compareSteadyNoNascent(inspectLiver, expressionThreshold =
.01)
plotPMtrend(inspectLiver)
```

The results of ‘compareSteadyNoNascent’, which can be accessed through the “PTreg” method, consist of a matrix where rows represent genes and columns represents conditions (in this case the samples). 0 indicates that the gene in that condition is expressed but not post-transcriptionally regulated, 1 indicates that the gene is expressed and post-transcriptionally regulated, and NA indicates that the gene has either premature or mature RNA below the expression threshold. We briefly summarize the results of the “compareSteadyNoNascent” method on the Liver HCC dataset by determining the number of regulated genes per sample and the number of regulated samples for gene (Fig. 5). This analysis revealed that the vast majority of the samples have between 0 and 600 post-transcriptionally regulated genes, with the majority of them lying below 200 regulated genes. Conversely, most of the genes were found to be regulated in less than 20 samples.

```
pt_reg <- PTreg(inspectLiver)
reg_genes_per_sample <- colSums(1*pt_reg, na.rm=TRUE)
reg_samples_per_gene <- rowSums(1*pt_reg, na.rm=TRUE)
par(mfrow=c(1,2), mar=c(5,4,1,2))
hist(reg_genes_per_sample, breaks=50,
     xlab='Regulated genes per sample', main = '')
hist(reg_samples_per_gene, breaks=50,
     xlab='Regulated samples per gene', main = '')
```

As in the analysis of the post-transcriptional ratio, we observe a trend between the number of post-transcriptionally regulated genes and the sample type (Fig. 6). In this case, the trend is positive, meaning that the number of post-transcriptionally regulated genes increases with the progression of the disease.

```
reg_genes_per_type <- split(reg_genes_per_sample, sample_type)
names(reg_genes_per_type) <- gsub(' ', '\n', names(reg_genes_per_type))
reg_genes_per_type_lm <- lm(sapply(reg_genes_per_type, median) ~ seq(1,3))
boxplot(reg_genes_per_type, varwidth=T, notch=T, outline=FALSE, las=2,
        ylab = 'regulated genes per sample',
        main='Grouped by sample type')
abline(reg_genes_per_type_lm, col='red')
```

In order to visualize the results of the analysis in a more comprehensive way, we use the heatmap functionality as implemented by the package “pheatmap”. The matrix that is output by the “compareSteadyNoNascent” method is clustered on both rows and columns in order to facilitate the visualization. The original matrix is composed by 0, 1, and NAs. The presence of missing values might impair the estimation of the similarity among genes and

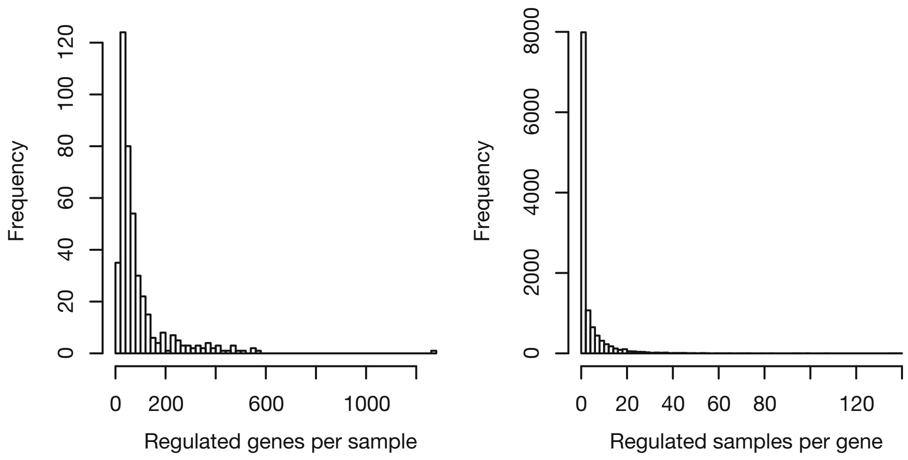


Fig. 5 Histograms representing the number of regulated genes per sample and the number of regulated samples per gene, as estimated by the method "compareSteadyNoNascent"

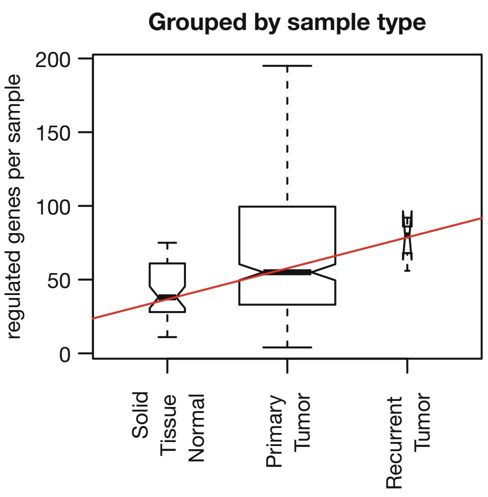


Fig. 6 Boxplot representing the number of regulated genes per sample, grouped by the TCGA sample type. The trend line, in red, is evaluated based on the group medians

samples; therefore, we transform the NAs to 0, the 0 to 1 and 1 to 2. Following this, nonexpressed genes are represented by 0 (color coded in blue), expressed and not differentially post-transcriptionally regulated by 1 (yellow), and expressed and differentially post-transcriptionally regulated by 2 (red).

```
pt_reg_heat <- 1*pt_reg + 1
pt_reg_heat[is.na(pt_reg_heat)] <- 0
```

We complement the columns of the post-transcriptional regulation matrix with selected metadata. For reasons of simplicity, we plot only the information relative to the subset of genes that carried the greatest information in terms of post-transcriptional regulation, that is, those genes found to be regulated in at least 20 samples. Additionally, we use the k-means clustering to identify groups among genes and samples that share similar post-transcriptional patterns (Fig. 7).

```r
library(pheatmap)
annotation_col <- data.frame(
  sample_type = sample_type,
  tumor_stage = sub('[a-c]$',
  '',hcc_metadata$gdc_cases.diagnoses.tumor_stage),
  gender = hcc_metadata$gdc_cases.demographic.gender,
  race = hcc_metadata$gdc_cases.demographic.race,
  row.names = colnames(pt_reg)
)
ann_colors <- list(
  tumor_stage = c('not reported'='grey', 'stage i'='lightgoldenrod',
                  'stage ii'='orange', 'stage iii'='violet',
                  'stage iv'='purple'),
  race = c('american indian or alaska native'='brown',
           'asian'='lightgoldenrod',
           'black or african american'='black',
           'not reported'='grey', 'white'='lightpink'),
  gender = c('female'='lightpink', 'male'='lightskyblue'),
  sample_type = c('Solid Tissue Normal'='orange', 'Primary Tumor'='violet',
                  'Recurrent Tumor'='purple')
)
pt_reg_heat_subset <- pt_reg_heat[reg_samples_per_gene>=20,]
set.seed(0)
row_clustering <- kmeans(pt_reg_heat_subset, 22,
                         iter.max = 100, nstart = 10)
columns_clustering <- kmeans(t(pt_reg_heat_subset), 6,
                             iter.max = 100, nstart = 10)
pheatmap(pt_reg_heat_subset[order(row_clustering$cluster),
                            order(columns_clustering$cluster)],
         cluster_cols = FALSE, cluster_rows = FALSE,
         gaps_row = which(diff(sort(row_clustering$cluster))==1),
         gaps_col = which(diff(sort(columns_clustering$cluster))==1),
         show_rownames = FALSE, show_colnames = FALSE,
         annotation_col = annotation_col, annotation_colors = ann_colors
)
```

Noteworthily, the first cluster among the columns is able to collect almost all the normal samples. Other clusters, which are composed mostly by primary tumors samples, show very different sets of post-transcriptionally regulated genes. In order to better dissect the information contained among clustered rows, it is possible to search for specific enrichments among genes that clustered together. For instance, the AURA database [6] collects cis- and trans- elements experimentally associated to the 5′ and 3’ UTRs of individual genes. It is possible to query the database online (aura.science.unitn.it) and use the batch mode to find the enrichments of

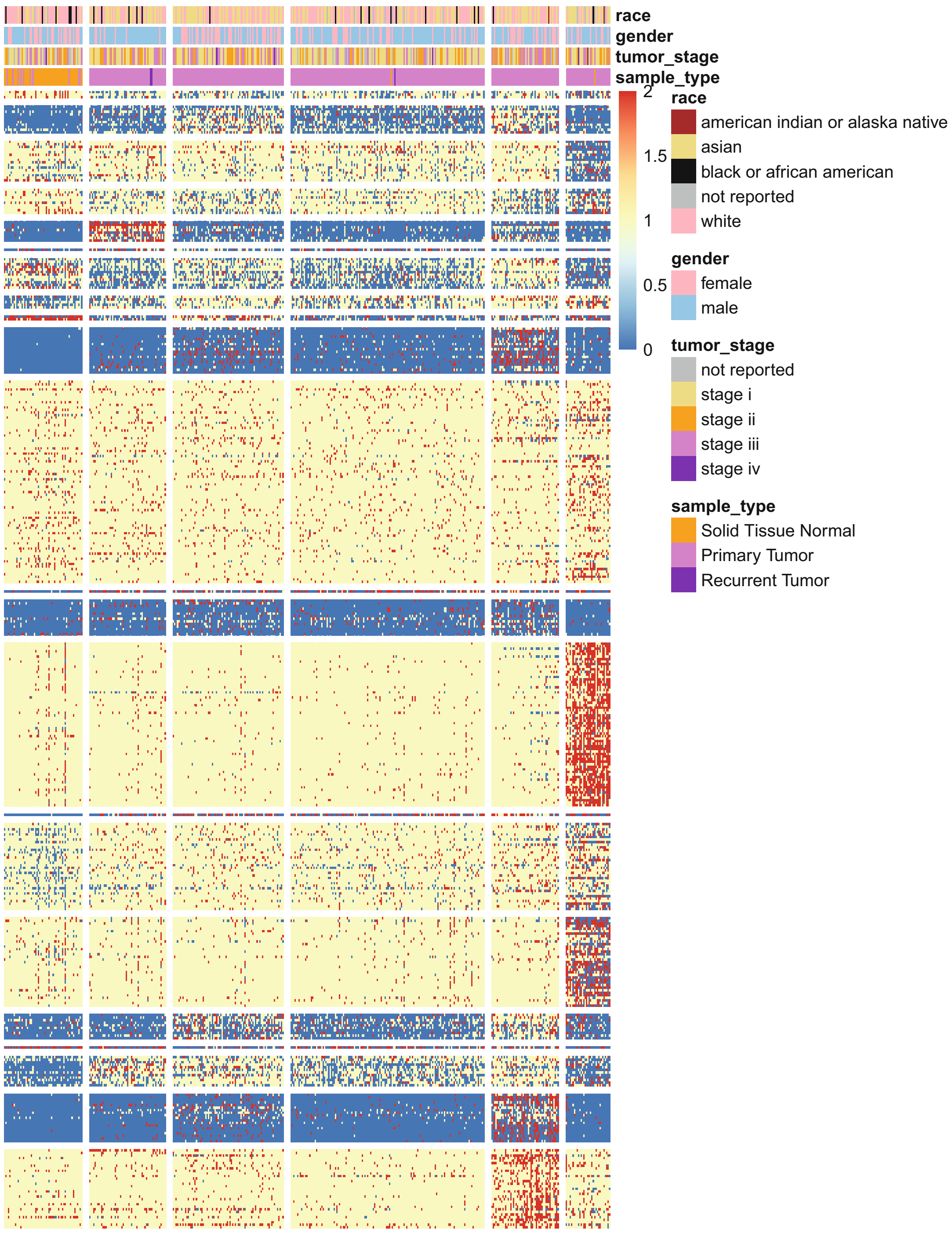


Fig. 7 Heatmap representing post-transcriptionally regulated genes, as identified by INSPEcT. Blue dots are genes expressed below the threshold set during the analysis, yellow dots are expressed genes with no evidence of post-transcriptional regulation, and red dots are post-transcriptionally regulated genes. Rows, representing genes, are clustered using k-means with 22 centers. Columns, representing samples, are clustered using k-means with six centers. Race, gender, tumor stage, and sample type are extracted from TCGA metadada and reported as color-bars on the top of the heatmap

regulatory elements. Alternatively, it is possible to download a light version of the AURA database and perform the enrichment tests locally. As an example, we show the code to download the database and perform the enrichment analysis on the trans-factors that bind on the genes of cluster n. 17, which shows high levels of post-transcriptional regulation. The results of this analysis highlight statistical enrichment for more than 60 trans-factors, after correction of the p-values with the Benjamini–Hochberg procedure and using 0.05 as a threshold. Among the most enriched trans-factors it is possible to find AKAP8L, DROSHA, FTO, MTPAP, NKRF, RPS11, and TBRG4.

```
## download the AURA database
download.file(
  url = 'http://aura.science.unitn.it/site_media/download/AURAlight.tar.gz',
  destfile = 'AURAlight.tar.gz')
untar('AURAlight.tar.gz', exdir = 'AURAlight')
aura_trans_factors <-
  read.table('AURAlight/AURAlight_trans-factors.txt',
             sep='\t', header=TRUE, stringsAsFactors = FALSE)
## add the entrez id column
library(org.Hs.eg.db)
aura_trans_factors$ENTREZID <-
  mapIds(org.Hs.eg.db, keys = aura_trans_factors$Target.Gene,
         column = 'ENTREZID', keytype = 'SYMBOL')
aura_trans_factors_eid <-
  aura_trans_factors[!is.na(aura_trans_factors$ENTREZID),]
## split regulator factors by entrez id
class_aura_trans <- split(aura_trans_factors_eid$Regulatory.factor,
                          aura_trans_factors_eid$ENTREZID)
class_aura_trans_unique <- lapply(class_aura_trans, unique)
## enrichment analysis for cluster 17
cluster_number <- 17
cluster_members <- names(which(row_clustering$cluster == cluster_number))
regulators_all_genes <- factor(unlist(class_aura_trans_unique))
regulators_cluster_members <-
  factor(unlist(class_aura_trans_unique[cluster_members]),
         levels = levels(regulators_all_genes))
regulators_table <- cbind(table(regulators_all_genes),
                          table(regulators_cluster_members))
transcript_number <- 29345
background_model <- c(transcript_number, length(cluster_members_symbols))
fisher_test_pvalues <- sapply(which(regulators_table[,2]>0), function(i)
  fisher.test(unname(rbind(background_model-regulators_table[i,],
regulators_table[i,])),
              alternative = 'greater')$p.value)
fisher_test_padjusted <- p.adjust(fisher_test_pvalues, method='BH')
```

4 Notes

1. The analysis presented here may potentially run within SciServer (www.sciserver.org). SciServer provides access both to the counts and BigWig files of the Recount project, and the computational tools to analyze them, that is an RStudio environment preloaded with the recount package. As a consequence, the steps and computational time spent to download the raw expression data could be skipped. Nevertheless, at the time of this writing, Sciserver runs R-3.5.1, while the R/Bioconductor package INSPEcT requires R $\geq$ 4.0.0. As soon as the Recount compute image within SciServer will be updated to a newer version of R, the pipeline described in the Methods part of this chapter could be run within SciServer with small adaptations.
2. In case the raw data of the dataset under analysis are provided in the form of BAM files, the "INSPEcT" package implemented a function that is analogous to "quantifyExpressionsFromBWs" and specifically treats BAM files. The name of the routine is "quantifyExpressionsFromBAMs."
3. At the time of writing of this script, the only precompiled annotation available within R/Bioconductor relative to hg38 is the UCSC annotation (i.e., the package 'TxDb.Hsapiens.UCSC.hg38.knownGene'). This annotation contains the information about 25,221 genes, most of them are protein-coding. Noteworthily, TxDb objects relative to different annotations can be created via the "GenomicFeatures" package. In particular, the more comprehensive Ensembl annotation can be generated using the function "makeTxDbFromEnsembl." More generally, any user-defined annotation can be created starting from a GFF/GTF file using the function "makeTxDbFromGFF." Here we report the code to generate the Ensembl annotation relative to hg38 genome; in particular, we generate the annotation relative to version number 98, which was released in September 2019 and contains the annotation of 67′946 genes. In order to make the function "makeTxDbFromEnsembl" working properly, the R/CRAN package 'RMariaDB' and a mysql client must be installed on the local machine.

```
BiocManager::install('RMariaDB')
library(GenomicFeatures)
txdbEnsemblGRCh38 <- makeTxDbFromEnsembl(organism="Homo sapiens", release=98)
```

4. In case other TCGA datasets are analyzed, they may contain different sample types, such as "metastatic" or additional terms. In that case the levels should be redefined accordingly.

References

1. de Pretis S, Kress T, Morelli MJ, Melloni GE, Riva L, Amati B et al (2015) INSPEcT: a computational tool to infer mRNA synthesis, processing and degradation dynamics from rna-and 4sU-seq time course experiments. Bioinformatics 31:2829–2835
2. Gaidatzis D, Burger L, Florescu M, Stadler MB (2015) Analysis of intronic and exonic reads in rna-seq data characterizes transcriptional and post-transcriptional regulation. Nat Biotechnol 33:722
3. Alkallas R, Fish L, Goodarzi H, Najafabadi HS (2017) Inference of rna decay rate from transcriptional profiling highlights the regulatory programs of Alzheimer's disease. Nat Commun 8:1–11
4. Furlan M, Galeota E, Gaudio N del, Dassi E, Caselle M, de Pretis S, et al (2019) Genome-wide dynamics of RNA synthesis, processing and degradation without RNA metabolic labeling. bioRxiv. https://doi.org/10.1101/520155
5. Collado-Torres L, Nellore A, Kammers K, Ellis SE, Taub MA, Hansen KD et al (2017) Reproducible rna-seq analysis using recount2. Nat Biotechnol 35:319–321
6. Dassi E, Re A, Leo S, Tebaldi T, Pasini L, Peroni D et al (2014) AURA 2: empowering discovery of post-transcriptional networks. Translation 2: e27738

Chapter 16

Computational Analysis of Single-Cell RNA-Seq Data

Luca Alessandrì, Francesca Cordero, Marco Beccuti, Maddalena Arigoni, and Raffaele A. Calogero

Abstract

Single-cell RNAseq data can be generated using various technologies, spanning from isolation of cells by FACS sorting or droplet sequencing, to the use of frozen tissue sections retaining spatial information of cells in their morphological context. The analysis of single cell RNAseq data is mainly focused on the identification of cell subpopulations characterized by specific gene markers that can be used to purify the population of interest for further biological studies. This chapter describes the steps required for dataset clustering and markers detection using a droplet dataset and a spatial transcriptomics dataset.

Key words Single cell RNA sequencing, Droplet, Spatial transcriptomics, Clustering, Cell markers, Bioinformatics

1 Introduction

Single-cell sequencing [1] is becoming a very powerful instrument to understand the heterogeneity of cells populations. Nowadays, single-cell data can be generated using a variety of technologies [1]. Full-transcript single-cell sequencing (i.e., each cell is isolated from a pool using FACS sorting [2] or specific microfluidic devices [3], and sequence is done using a RNAseq highly sensitive library preparation method, providing full-length transcript coverage [4]), although being the most sensitive available method, is rarely used today, because of its limited throughput. Instead, majority of single cell sequencing is done by droplet-based RNAseq 3′ end sequencing methods [5, 6], which are available as commercial solutions (Biorad, 10XGenomics, 1Cellbio, BD). Recently, it also became possible to obtain single-cell sequencing data from frozen tissues, without the need of dissociating the cells, using a technology called spatial transcriptomics [7], commercialized by 10XGenomics. Single cell sequencing technology delivers data made by extremely large and sparse matrices, that is, from thousands to hundreds of thousands of features with less than 3% of entries which are non-zero. Aggregating cells on the basis of transcriptomics represents one of the challenges of single cell data analysis. In this context a mandatory aspect is the availability of dedicated bioinformatics workflows. To the best of our knowledge, there are only few computational workflows providing analysis flexibility and achieving at the same time functional and computational reproducibility [8] from counts generation to cell subpopulation identification. Among these tools, we developed rCASC [8], a modular workflow providing integrated analysis environment for single cell data. All the computational tools in rCASC are embedded in docker images stored in a public repository on docker hub. Parameters are delivered to docker containers via a set of R functions, part of rCASC R github package (https://github.com/kendomaniac/rCASC). rCASC is specifically designed to provide an integrated analysis environment for cell subpopulation discovery. rCASC allows the direct analysis of fastq files, generated with 10XGenomics, inDrop, and BD Rhapsody platforms, or counts tables. rCASC provides raw data preprocessing, subpopulation discovery via different clustering approaches and cluster-specific gene-signature detection. In this chapter, we will describe a basic workflow suitable for the analysis of droplet and spatial transcriptomics datasets. Exemplary data and the analysis steps for their analysis are available in a dedicated github repository called droplet-spatial (https://github.com/kendomaniac/droplet-spatial).

The workflow described in this chapter is summarized in Fig. 1.

It is important to note that any analysis workflow needs to be adapted to the dataset under analysis. However, some basic steps are generally applicable and represent a starting point for an exploratory analysis (Fig. 1): (1) conversion of fastq files in a counts table; (2) inspection of the cells sequencing data characteristics; (3) removal of mitochondrial/ribosomal protein genes and cells with little information content, that is, few detectable genes; (4) selection of a user defined number of the most variant and most expressed genes; (5) selection of the most effective clustering procedure; (6) detecting cluster-specific marker genes.

2 Materials

2.1 Exemplary Datasets

The exemplary dataset, from droplet-based technology (10XGenomics), available in droplet-spatial github, was generated using the sequencing data from FACS sorted cell types, published by Zheng [9]. This dataset, called **SetA**, was also used as exemplary dataset in rCASC paper [8]. The counts table is made of a total of 500 cells, including 100 cells randomly selected from each of the following cell types: B-cells (B), Monocytes (M), Hematopoietic Stem cells (SC), Natural Killer cells (NK), and Naive T-cells (NT).

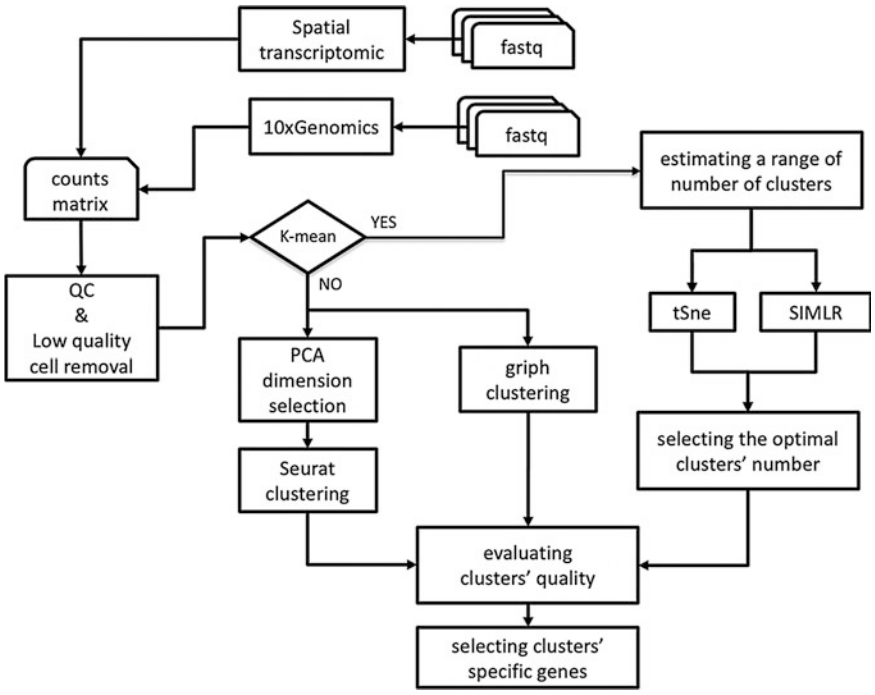

Spatial transcriptomic
fastq
10xGenomics
fastq
counts matrix
QC & Low quality cell removal
K-mean
YES
NO
estimating a range of number of clusters
tSne
SIMLR
PCA dimension selection
griph clustering
Seurat clustering
selecting the optimal clusters' number
evaluating clusters' quality
selecting clusters' specific genes

Fig. 1 Exploratory data analysis workflow for single cell RNAseq

The exemplary dataset for spatial transcriptomics is the human lymph node dataset available at 10XGenomics data repository. The count table is available at droplet-spatial github with the name **limphoST**.

An exemplary fastq dataset, **SChs1m**, for 10XGenomics technology, is available in the droplet-spatial github. This dataset contains one million reads and was generated only to learn how to use the fastq to counts conversion software implemented in rCASC tool. This dataset is from human and requires the 10Xgenomics human reference genome (https://support.10xgenomics.com/single-cell-gene-expression/software/downloads/latest).

A dataset for Spatial Transcriptomics technology, is available at 10XGenomics data repository (https://support.10xgenomics.com/spatial-gene-expression/datasets/1.0.0/V1_Mouse_Kidney). This dataset is from mouse and requires mouse spatial transcriptomics reference genome (https://support.10xgenomics.com/spatial-gene-expression/software/downloads/latest).

2.2 Computational Hardware

The computing hardware requirements depends on the number of cells under analysis and on the clustering tool. Up to 4000 cells can be analyzed in few hours with a computer equipped with at least an i7 intel CPU (3.5 GHz, 8 threads), 32 GB RAM, and 1 TB SSD.

2.3 Computational Software

The described workflow requires a UNIX or a MacOS operating system, the installation of docker daemon (https://docs.docker.com/install/) and the installation of R (https://cran.r-project.org/) version >3.00. Furthermore, the installation in R of devtools

```
install.packages("devtools")
```

and rCASC

```
library(devtools)
install_github("kendomaniac/rCASC", ref="master")
```

is also needed. In case rCASC is used via graphical interface, Oracle JAVA, at least version 8 has to be installed (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) and 4SeqGUI (https://github.com/mbeccuti/4SeqGUI) needs to be downloaded on the local computer.

3 Methods

3.1 From Fastq to Count Table

The demultiplexing of single cell data produce three files with the following extensions: I1_XXX.fastq.gz, R1_XXX.fastq.gz ad R2_XXX.fastq.gz, where XXX is a progressive number, identical for all three files; for example, 001. I1 is the sample index present in the sequenced fragment, R1 is fragment forward sequence, including cell barcode and UMI (Unique Molecular Identifier), R2 is the reverse sequence including the sequence of the 3′ end of the transcript (Fig. 2).

3.1.1 10XGenomics from Fastq to Counts

rCASC function *cellrangerCount* executes the *cell ranger* software of 10XGenomics and converts fastq files in a counts table, where columns are cells and row are genes. The function requires the following mandatory parameters: *transcriptome.folder*, which is the folder where genome index was downloaded; *fastq.folder*, which is the folder where the fastq files are located; *scratch.folder*, which is the folder where temporary files are generated (an example of code for the use of *cellrangerCount* function is available at https://github.com/kendomaniac/droplet-spatial/blob/master/droplet-spatial.pdf).

The output counts table is saved as *results_cellranger.csv*, this table represents the starting point for data analysis. In the subfolder *outs* there is the file *web_summary.html*, which contains statistics on the sequenced data, that is, Number of sequenced cells, mean reads/cell, median number of genes/cell, sequencing statistics, mapping statistics, and cell statistics.

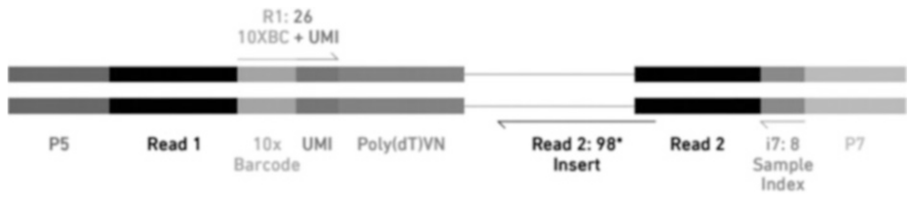


Fig. 2 10XGenomics library structure

3.1.2 Spatial Transcriptomics from Fastq to Counts

In the case of spatial transcriptomics (ST) data, the function for the generation of a counts table is *stpipeline*, and the mandatory parameters are: *scratch.folder*; *data.folder*, which is the folder where results are saved; *genome.folder*, which is the folder where genome index was downloaded; *fastqPathFolder*, which is the folder where the fastq files are located. *ID*, a string indicating the name of the sample; *imgNameAndPath*, the path and name of tiff image of the tissue on ST slide; *slide*, the identification number of the slide, which is provided by 10XGenomics; *area*, value from the sequenced area, which are four in each slide. Ones the counts table is generated the analysis steps are the same for all single cell data (Fig. 1).

3.2 Counts Table QC

The first step of the exploratory analysis is the evaluation of the UMI distribution within each cell and the evaluation of cells quality. The above estimations are done by the function *mitoRiboUmi*, which requires the following mandatory parameters: *file*, which is the count table with the full path; *scratch.folder* (*see* above); *gtf.name*, the name of the ENSEMBL GTF file corresponding to the reference genome used for mapping, which must be located in the same folder where the count table is. *mitoRiboUmi* function generates two plots, one showing the number of detected genes plotted for each cell with respect to the total number of UMI/reads in that cell (*genes.umi.pdf*, Fig. 3a) and the other (*Ribo_mito.pdf*, Fig. 3b) showing the percentage of mitochondrial protein genes plotted with respect to percentage of ribosomal protein genes (an example of code for the use of *mitoRiboUmi* function is available at https://github.com/kendomaniac/droplet-spatial/blob/master/droplet-spatial.pdf).

3.3 Annotation and Filtering

The function *scannobyGtf* offers the possibility to add gene symbol to ENSEMBL gene ids and to performs a set of filters (*see* **Notes 1–3**). The mandatory parameters are: *file*, which is the count table with the full path; *gtf.name*, the name of the ENSEMBL GTF file corresponding to the reference genome used for reads mapping and located in the same folder where the count table is. *mt*, a Boolean defining if mitochondrial genes have to be removed, FALSE means that mt genes are removed. *Ribo.proteins*, a Boolean defining if ribosomal proteins have to be removed, FALSE means that ribosomal proteins (gene names starting with rpl or rps) are removed.

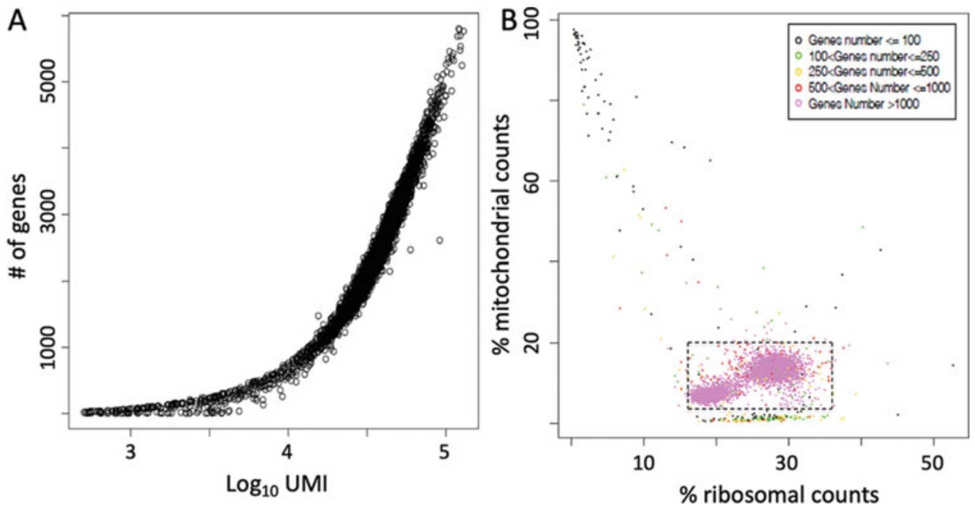


Fig. 3 Cells QC: (**a**) Cells represented by the number of genes called present versus log$_{10}$UMI. (**b**) Cells represented as fraction of mitochondrial genes expressed with respect to the fraction of expressed ribosomal genes

riboStart.percentage, lower bound for ribosomal percentage, cells below this value are discarded. *riboEnd.percentage*, upper bound for ribosomal percentage, cells above this value are discarded. *mitoStart.percentage*, lower bound for mitochondrial percentage, cells below this value are discarded. *mitoEnd.percentage*, upper bound for mitochondrial percentage, cells above this value are discarded. *thresholdGenes*, an integer number indicating the minimal number of genes called present required to retain a cell for further analysis. The annotated and filtered table has the prefix *filtered_annotated_*, the effects of the filtering is shown in a pdf having as postfix of the counts table *_annotated_genes.pdf*. A summary of the filtering is available in the file *filteredStatistics.txt*.

3.4 Selecting Top Ranked Genes

The function *topx*, allows to select a defined number of genes ranked on the basis of variance or expression (*see* **Note 4**). The mandatory parameters are *file*, which is the count table with the full path; *threshold*, an integer indicating the number of genes to be selected; *type*, which has two possible values: *expression*, referring to the selection of the top expressed genes and *variance* referring to the selection of the top variable genes. In Fig. 4, it is shown an exemplary result of topx. In Fig. 4b 10,000 most varying genes are selected from the full set of genes (Fig. 4a), out of them 5000 most expressed genes are selected, Fig. 4c. (An example of code for the use of *topx* function is available at https://github.com/kendomaniac/droplet-spatial/blob/master/droplet-spatial.pdf).

The filtered table and a pdf showing the distribution of reads in the cells before and after filtering have the prefix *filtered_expression_* or *filtered_variance_* depending on the filtering option defined in *type* parameter.

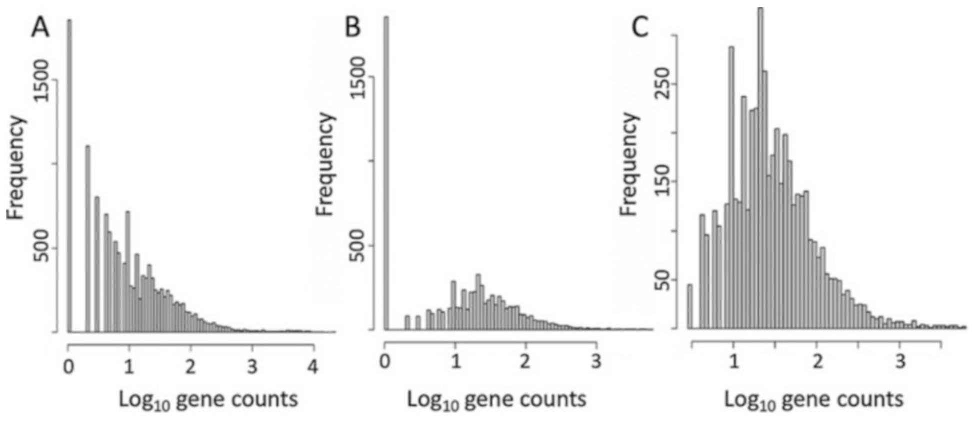


Fig. 4 Filtering genes: (**a**) full dataset, (**b**) selecting from A the 10,000 most varying genes, and (**c**) selecting 5000 most expressed genes out of (**b**)

3.5 Clustering

rCASC was developed for the characterization of cells heterogeneity. Subpopulations organization is depicted by rCASC using clustering methods. However, since there is no a priori way to define which is the optimal clustering method for a specific sample, we have implemented in rCASC four clustering tools: tSne [10], SIMLR [11], griph [12], and Seurat [13]. Furthermore, in rCASC we have developed a specific metric call Cell Stability Score (CSS), which provides information on the stability of cells in a cluster and on the relative cell type homogeneity of a cluster, for more information on CSS please refer to rCASC paper [8]. The outputs of the analysis are saved in a folder Results containing a folder with the same name of the counts table. In this folder are present: (1) a file called *_Stability_Violin_Plot.pdf*, providing an overview of the CSS in all clusters and (2) a folder named with the number of clusters used in the analysis. In the latter folder are present: (1) a pdf file with extension *_Stability_Plot.pdf*, which provides a plot of the detected clusters and the CSS for each cell; (2) a file with the extension *_clustering.output*; and (3) a file with extension *_scoreSum*. The data at 2 and 3 are used to build the *_Stability_Plot.pdf*.

3.5.1 Clustering with tSne

*tsneBootstra*p wrapper function allow to perform tSne analysis and calculating CSS metrics. The mandatory parameters are: *scratch.folder* (*see* previous functions); *file*, the path of the file, with file name and extension included; *nPerm*, number of permutations to be executed; *permAtTime*, number of permutations computed in parallel; *percent*, percentage of randomly selected cells removed in each permutation. *range1*, lower bound of the range of clusters to be investigated; *range2*, upper bound of the range of clusters to be investigated; *perplexity*, number of close neighbors for each point (*see* **Note 5**).

3.5.2 Clustering with SIMLR

simlrBootstrap wrapper function allows to perform SIMLR analysis and calculating CSS (*see* **Note 6**). The mandatory parameters are the same for the function *tsneBootstra*p, but *perplexity.* SIMLR identifies a distance metric that better fits the structure of the data by combining multiple Gaussian kernels [11].

3.5.3 Clustering with Griph

griphBootstrap wrapper function executes SIMLR analysis and calculating CSS. The parameters are the same for *simlrBootstrap*. However, since griph detects the optimal number of clusters, *range*1 and 2 parameters are not required. The griph algorithm is closer to agglomerative clustering methods, because every node is initially assigned to its own community and communities are subsequently built by iterative merging [12].

3.5.4 Clustering with Seurat

Before performing clustering with Seurat, it is required to run *seuratPCAEval*, which estimates the range of principal components (PC) to be used in Seurat clustering, with each PC essentially representing a metagene that combines information across a correlated gene set. *seuratPCAEval* returns a plot called *PCE_bowPlot.pdf* (*see* **Note 7**) Fig. 5.

The function required to run Seurat clustering is *seuratBootstrap*. The parameters for *seuratBootstrap* are the same for *griphBootstrap*, unless for *pcaDimension*, which represents the PC threshold defined with *seuratPCAEval.* Seaurat clustering is based on the Louvain modularity optimization algorithm (*see* **Note 8**).

(example of code for the use of *tsneBootstra*p, *simlrBootstrap*, *griphBootstrap*, *seuratPCAEval*, and *seuratBootstrap* functions are available at https://github.com/kendomaniac/droplet-spatial/blob/master/droplet-spatial.pdf, *see* **Note 9**).

3.6 Depicting Cluster-Specific Genes

Genes, detected as specific for each cluster, can be used as target for antibodies to trace or purify subpopulations identified by single-cell RNAseq. In rCASC we have implemented both statistical and machine-learning algorithms to detect cluster-specific gene markers. The function *anovaLike*, is a wrapper for the edgeR anova-like [14], which can be used also for single-cell data. This statistic method allows the comparison of multiple conditions with respect to a reference. Thus, it can be applied to single cells, if a cluster can be used a reference. The mandatory parameters for this analysis are: *file*, a character string indicating the counts table file with the path of the file. *cluster.file*, a character string indicating the *_clustering.output.txt* file (*see* **Note 10**) of interest, generated by any of the clustering methods described in Subheading 3.5. *ref.cluster* is a number indicating the cluster to be used a reference for anova-like comparison with the other clusters. *logFC.threshold* indicates the minimal $\log_2$ fold change which should be detected in at least one of the comparisons with respect to reference covariate. *FDR.threshold* is the minimal adjusted-pvalue detected in at least one of

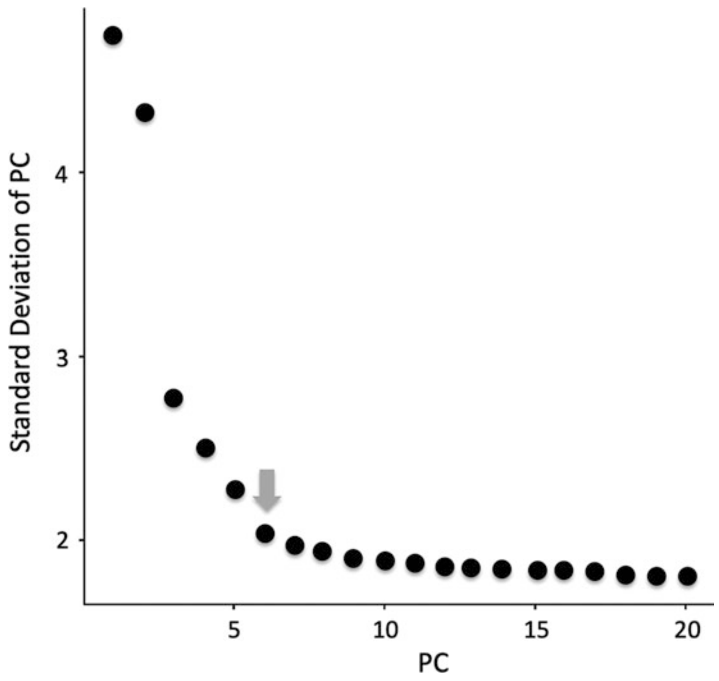


Fig. 5 PCE_bowPlot

the comparisons with respect to reference covariate. *logCPM.threshold*, is the minimal average abundance of the gene expressed as $\log_{10}$ CPMs. *plot* is Boolean value (TRUE, FALSE). When it is set to TRUE a plot of the differentially expressed genes is generated.

The function *seuratPrior* selects from the output of *seuratBootstrap* the set of genes that play the major role in separating cells in clusters. The mandatory parameters are: *scratch.folder*, already described in above. *file*, *see* above functions. *PCADim*, which is the number of principal components used in *seuratBootstrap*. *geneNumber* is the expected number of cluster specific genes. *nCluster* indicates the total number of clusters detected by *seuratBootstrap*.

The function *cometsc* controls a computational framework for the identification of candidate gene markers consisting of one or more genes for each cell populations identified with single-cell RNA-seq data [15]. The mandatory parameters are: *file*, *see* above functions; *scratch.folder*, *see* above functions. *X* which is the argument for XL-mHG, where default is 0.15, for more info *see cometsc* manual (https://hgmd.readthedocs.io/en/latest/manual.html). *K* is the number of gene combinations to be considered. The possible values are from 2 to 4 and default is 2. *nCluster* is the number of clusters generated by the clustering methods used, *see* above paragraph. The output are the folders *outputvis* and *outputdata*, containing, for each cluster, respectively plots for the possible markers and tables with ranking statistics (*see* **Note 11**), for more info *see cometsc* manual.

4 Notes

1. We suggest to filter out cells with less than 250 called genes (Fig. 3a), since their transcriptome is too small to provide sufficient information for subpopulation-specific clustering.
2. Furthermore, percentage of mitochondrial protein genes greater than 10% in human and 5% in mouse (Fig. 3b) is an indication of stressed cells, which are probably undergoing to apoptosis [16]. At the same time cells which have less than 5% mitochondrial protein genes (Fig. 3b) are also quite unexpected. In Fig. 3b, cells show more than 35% ribosomal protein genes indicating the presence of duplets, that is, two cells in a single droplet. Thus, for this specific example, we suggest to apply a filter, removing cells having less than 5% of mitochondrial protein genes, cells having more than 10% (human) and 5% (mouse) of mitochondrial protein genes. At the same time, we suggest to remove cells with more than 35% ribosomal protein genes.
3. It is important to note that filters suggested in **Notes 1** and **2** are not generally applicable thresholds, since thresholds are affected by cell type and library prep chemistry. The QC plots are useful to define for each specific experiment the optimal filtering thresholds. For example, it has to be noted that some cell types might have more than 35% of ribosomal counts. Since duplets are relatively rare, we suggest to filter cells with very high ribosomal counts percentage with respect to the majority of the cells in the experiment.
4. Since single cell RNAseq tables are zero-inflated, filtering for variance first and expression after select genes that probably are the most useful for clustering. We suggest to retain 10,000 most variant genes (Fig. 4b), and then retaining from them the top 5000 most expressed (Fig. 4c). This filtering approach has the extra advantage of reducing the size of the counts table.
5. If the overall quality of the tSne clustering is poor it is necessary to trim the perplexity parameter to see if it is possible to improve the overall CSS of the clustering.
6. As described in rCASC paper [8] the clustering time required by SIMLR is strongly affected by the number of cells in the dataset. We suggest to run SIMLR only for datasets below 2000 cells.
7. To define the optimal number of principal components to be used by Seurat, we suggest to look at the PCE_bowPlot.pdf and select as threshold the last PC, before PCs became little informative (Fig. 5, grey arrow).

8. Although griph and Seurat are based on Louvain modularity optimization algorithm, the different ways they normalize the data and the use of meta-features by Seurat, make the two clustering approaches similar but not identical.
9. There is no a priori way to identify the optimal clustering approach. Thus, in rCASC we have implemented four different methods in order to select the one that provide the best solution. CSS is the instrument that we used to detect the best cell partition approach, since it provides a measurement of how much cells are stably associated to a cluster given a data perturbation, that is, removal of a subset of cells [8]. An example is given in Fig. 6. The violin plots generated for each clustering method, on the same dataset, indicate that Seurat presents, in this specific example, the higher number of stable cells. Figure 7 shows a comparison between the Seurat clustering results (A) and griph clustering (B) results. All clusters but one in griph clustering (Fig. 7b) are characterized by a poor CSS, as instead in Seurat clustering (Fig. 7a), there are only two clusters with a CSS below 50÷75%. Furthermore, it is possible that the two unstable clusters (Fig. 7a) are in reality a unique cluster, since cells shuffle between one and the other cluster upon minimal perturbations of the dataset.
10. This file must be located in the same folder where counts table is placed.

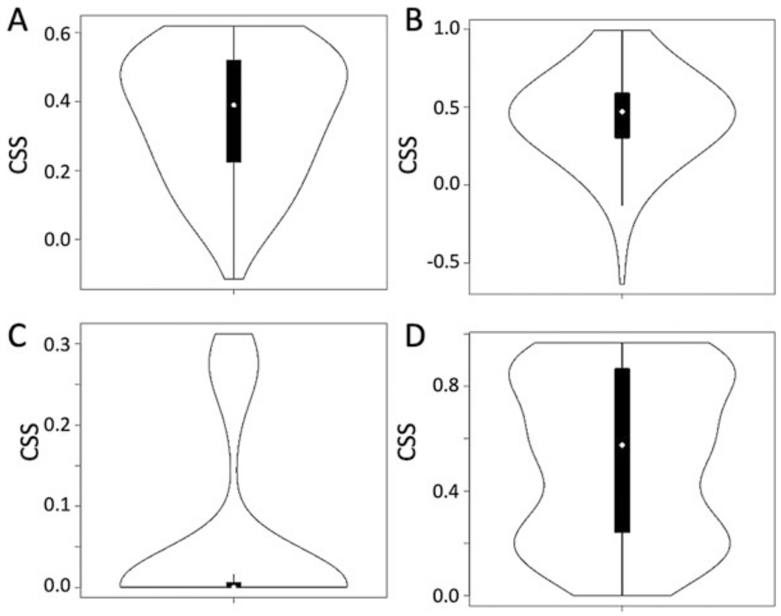


Fig. 6 CSS calculated using 80 permutations, in which 10% of the cells were randomly removed: (**a**) tSne, (**b**) SIMLR, (**c**) griph, and (**d**) Seurat

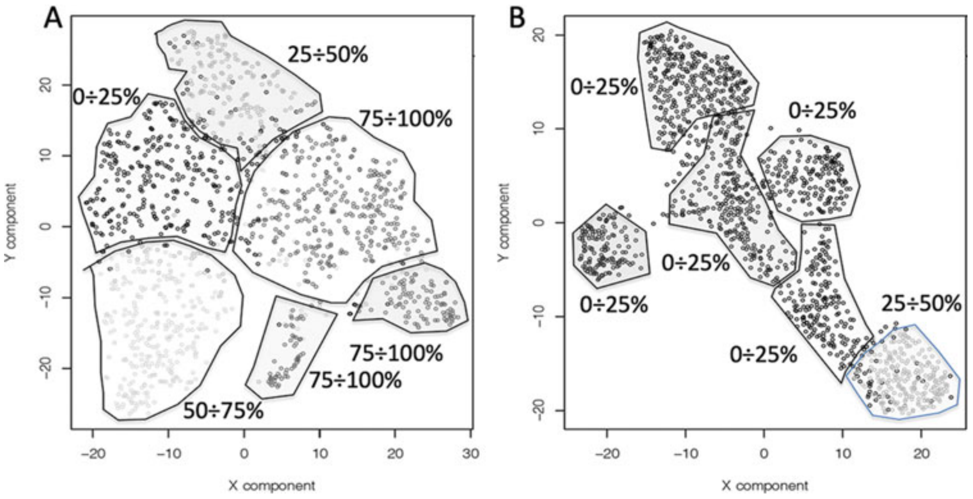


Fig. 7 (**a**) Seurat clustering results. (**b**) Griph clustering results. CSS is indicated for each cluster

11. A peculiar feature of cometsc is the identification of pairs, triplets and quadruplets of genes that, taken together, are able to discriminate one cluster from the other. This is particularly interesting as feature, since it might be used to identify surface markers suitable to identify and select specific cell subpopulation, belonging to a cluster, by FACS analysis.

References

1. Ziegenhain, C., et al., (2017) Comparative analysis of single-cell RNA sequencing methods. Mol Cell, 2017. 65(4): 631–643.e4
2. Gross A et al (2015) Technologies for Single-Cell Isolation. Int J Mol Sci 16 (8):16897–16919
3. Lecault V et al (2012) Microfluidic single cell analysis: from promise to practice. Curr Opin Chem Biol 16(3–4):381–390
4. Picelli S et al (2014) Full-length RNA-seq from single cells using smart-seq2. Nat Protoc 9 (1):171–181
5. Macosko EZ et al (2015) Highly parallel genome-wide expression profiling of individual cells using Nanoliter droplets. Cell 161 (5):1202–1214
6. Klein AM et al (2015) Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell 161(5):1187–1201
7. Salmen F et al (2018) Barcoded solid-phase RNA capture for spatial transcriptomics profiling in mammalian tissue sections. Nat Protoc 13(11):2501–2534
8. Alessandri L et al (2019) rCASC: reproducible classification analysis of single-cell sequencing data. Gigascience 8(9)
9. Zheng GX et al (2017) Massively parallel digital transcriptional profiling of single cells. Nat Commun 8:14049
10. Pezzotti N et al (2017) Approximated and user steerable tSNE for progressive visual analytics. IEEE Trans Vis Comput Graph 23 (7):1739–1752
11. Wang B et al (2017) Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. Nat Methods 14 (4):414–416
12. Serra D et al (2019) Self-organization and symmetry breaking in intestinal organoid development. Nature 569(7754):66–72
13. Butler A et al (2018) Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol 36 (5):411–420
14. McCarthy DJ, Chen Y, Smyth GK (2012) Differential expression analysis of multifactor

RNA-Seq experiments with respect to biological variation. Nucleic Acids Res 40 (10):4288–4297

15. Delaney C et al (2019) Combinatorial prediction of marker panels from single-cell transcriptomic data. Mol Syst Biol 15(10):e9005

16. Ordonez-Rueda D et al (2019) Apoptotic cell exclusion and bias-free single-cell selection are important quality control requirements for successful single-cell sequencing applications. Cytometry A

Chapter 17

Normalization of Single-Cell RNA-Seq Data

Davide Risso

Abstract

Normalization is an important step in the analysis of single-cell RNA-seq data. While no single method outperforms all others in all datasets, the choice of normalization can have profound impact on the results. Data-driven metrics can be used to rank normalization methods and select the best performers. Here, we show how to use R/Bioconductor to calculate normalization factors, apply them to compute normalized data, and compare several normalization approaches. Finally, we briefly show how to perform downstream analysis steps on the normalized data.

Key words RNA-seq, Single cell, Normalization, Exploratory data analysis, Quality control, Gene expression, Transcriptomics

1 Introduction

Normalization is an important yet often neglected step of RNA-seq data analysis, both for bulk [1] and single-cell data [2]. Recent benchmark studies have identified normalization as the choice that had the largest impact on performance of downstream analyses [3].

While many normalization methods have been proposed for bulk RNA-seq, a direct application of such methods to single-cell RNA-seq (scRNA-seq) data is not always appropriate. In fact, single-cell data are typically characterized by more heterogeneity and sparsity [2].

For this reason, several normalization strategies specifically designed for scRNA-seq have been published, mostly falling into two distinct classes: stand-alone normalizations and model-based approaches.

Stand-alone approaches include global scaling methods, such as *scran* [4] and *Census* [5], and nonlinear normalization, such as *SCnorm* [6] and quantile normalization [7]. The main advantage of these methods is their modularity. Once a normalized count matrix is defined, all downstream analysis steps can be applied directly to it.

Model-based approaches, on the other hand, start with the specification of a statistical model and include normalization as one of its parameters. Notable examples are *BASiCS* [8], *ZINB-WaVE* [9], and *GLM-PCA* [10].

Here, we focus on stand-alone methods implemented in Bioconductor [11]. We shall note that there are many alternative approaches, and a comprehensive review and benchmarking of normalization methods is outside the scope of this chapter. For a review of Bioconductor-based methods, *see* [12] and its associated e-book (http://bioconductor.org/books/release/OSCA/).

2 Materials

2.1 Computational Hardware

The workflow presented in this Chapter can be run on any computing system, since it uses the R/Bioconductor software suite, available for Windows, Linux, and Mac.

The example dataset is small enough that can be analyzed on a laptop or desktop computer. For larger datasets (more than 100,000 cells), we recommend using a machine with a large amount of memory, although solutions exist to work with out-of-memory data [13].

The use of systems with several CPUs is very advantageous, as many of the steps described below can be parallelized.

The analyses presented here were run on a MacBook Pro 2.9 GHz Intel Core i5 with 16 GB of RAM.

2.2 Computational Software

The results presented in this Chapter were run using R version 3.6.1 and Bioconductor version 3.10.

The following R commands can be used to install Bioconductor and all the packages used in this chapter.

```r
install.packages("BiocManager")

BiocManager::install(c("scater", "scone", "zinbwave", "scRNAseq",

                       "scran", "edgeR", "monocle", "igraph"),

                     version = "3.10")
```

The above command will install the specified packages and all the required dependencies.

To ensure that the package versions are consistent with each other, the `valid()` function can be used.

```
BiocManager::valid()
```

Finally, we load all the required packages.

```
library(scRNAseq)

library(scater)

library(scone)

library(scran)

library(edgeR)

library(monocle)

library(igraph)
```

2.3 Example Dataset

The analyses presented here make use of an example dataset, available in the form of a `SingleCellExperiment` object as part of the `scRNAseq` package (*see* **Note 1**). To access the data, we need to load the package and specify the dataset to use.

```
sce <- ReprocessedAllenData(assays = "tophat_counts")
```

Here, we use a subset of 379 cells from the mouse visual cortex from Tasic et al. [14]. However, by simply changing the code chunk above, readers can explore different examples. For a full list of datasets available through the `scRNAseq` package, *see* the package vignette at https://bioconductor.org/packages/scRNAseq.

The chosen dataset is described in detail in [14]. Briefly, the cells were isolated from the mouse visual cortex and sequenced using the SMARTer protocol.

As observed in the recent literature, full-length RNA protocols (such as SMARTer) and droplet-based methods differ in the statistical properties of the aggregated read count data [10, 15]. While the results reported here refer primarily to SMARTer data, many of the same considerations are also valid for droplet data (e.g., Dropseq or 10X Genomics). Readers who want to test the proposed workflow on 10X Genomics data are encouraged to look at the `TENxPBMCData` Bioconductor package (https://bioconductor.org/packages/TENxPBMCData), which contains several publicly available datasets.

3 Methods

3.1 The `SingleCellExperiment` Class

The data are stored as an object of the `SingleCellExperiment` class. The advantage of this class is that it is used across many Bioconductor packages, so that one can seamlessly integrate different packages in the same workflow.

The `SingleCellExperiment` class contains one or more matrices of expression data (stored in the `assays` slot). At the minimum, one matrix, typically containing the raw count data, is present. Optionally, other matrices containing transformed and/or normalized data are stored in the object as well.

In addition to the expression data, the object includes all the metadata associated with the cells (`colData`) and with the genes (`rowData`). Finally, the `altExp` component contains additional data not directly associated with the endogenous genes' RNA-seq profiles. This slot can be used to store information on the spike-in genes (as done here), or on surface protein expression (as in CITE-seq data [16]). For more information on the class, *see* [12].

It is worthwhile to look at the specific dataset in more detail.

```
sce
## class: SingleCellExperiment
## dim: 20816 379
## metadata(2): SuppInfo which_qc
## assays(1): tophat_counts
## rownames(20816): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3
## rowData names(0):
## colnames(379): SRR2140028 SRR2140022 ... SRR2139341
SRR2139336
## colData names(22): NREADS NALIGNED ... Animal.ID pas-
ses_qc_checks_s
## reducedDimNames(0):
## spikeNames(0):
## altExpNames(1): ERCC
```

Simply typing the name of the object displays useful information. We can see the number of genes (20,816) and cells (379) in the dataset, as well as additional information, like row names, column names, and the presence of metadata and ERCC spike-ins.

There is only one matrix of counts, called `tophat_counts`, no information on the genes other than the name (`rowData`) and 22 variables annotating the cells (`colData`).

The `colData()` and `rowData()` methods can be used to visualize all the information available at the cell level and at the gene level, respectively. The `assay()` method can be used to access the expression matrix, and the `altExp()` method to access information on the spike-ins.

To more easily use some of the functions below, we rename the assay of the object.

```
assayNames(sce)[1] <- "counts"

assayNames(altExp(sce))[1] <- "counts"
```

3.2 Quality Control and Filtering

Prior to normalization, it is advisable to remove poor quality cells and lowly expressed genes. To this end, a set of quality control metrics can be computed. Our example dataset already includes a set of such metrics in its column data, for example,

```
colData(sce)[,1:3]

## DataFrame with 379 rows and 3 columns
##                NREADS  NALIGNED    RALIGN
##             <numeric> <numeric> <numeric>
## SRR2140028  13743900  13011100   94.6681
## SRR2140022  14078700  13521900   96.0454
## SRR2140055   5842930   5135980   87.9008
## SRR2140083  16784400  15585800   92.8587
## SRR2139991  11558600  10864300   93.9929
## ...              ...       ...       ...
## SRR2139325  12875700  11307000   87.8172
## SRR2139373   9699400   8964140   92.4196
## SRR2139379   6175660   5728080   92.7526
## SRR2139341  28038500  26320000   93.8711
## SRR2139336   7878700   7467200   94.7772
```

However, it is easy to compute a set of QC metrics using the `scater` package [17].

```
stats <- perCellQCMetrics(sce)

stats[,1:3]


## DataFrame with 379 rows and 3 columns

##             sum  detected   percent_top_50

##       <numeric> <integer>        <numeric>

## 1       4949215      6708 16.5146796007044

## 2       6258794      6611 17.7734081038615

## 3       2181009      4857 14.8873755220634

## 4       4925535      7140 12.6828253174528

## 5       4479434      6412 17.0757734124445

## ...         ...       ...              ...

## 375     4046011      5068 16.5968406907445

## 376     3313703      7285 16.6980565246795

## 377     2492480      6352 17.3386747335987

## 378    13533062      8246 18.9948217188394

## 379     4674829      5572 19.3193376698912
```

In order to remove low quality cells, we can use the `metric_sample_filter()` function from the `scone` package [18] (*see* **Notes 2** and **3**). This function takes as input a list of "common genes", for example, genes that are highly expressed in at least a quarter of the cells.

```
num_reads <- quantile(counts(sce)[counts(sce) > 0], probs = .75)

num_cells <- .25 * ncol(sce)

is_common <- rowSums(counts(sce) >= num_reads ) >= num_cells
```

We are now ready to filter the cells.

```
mfilt <- metric_sample_filter(counts(sce),
                              nreads = colData(sce)$NREADS,
                              ralign = colData(sce)$RALIGN,
                              gene_filter = is_common,
                              zcut = 3, mixture = FALSE,
                              plot = FALSE)
```

With the specified arguments, this function computes a set of three metrics, used to determine whether a sample is of poor quality. The metrics are: the total number of reads per cell (`nreads`), the proportion of aligned reads (`ralign`) and the number of detected genes among the ones specified in `gene_filter`.

The argument `zcut` and `mixture` control the way in which the filtering is done: the former specifies the threshold according to which considering a cell of poor quality (in this case, if worse than three times the standard deviation from the mean); the latter instructs the function on whether to use a mixture model in the definition of the outliers. By setting `plot = TRUE`, the function returns a plot with the distribution of the metrics and the thresholds for the filtering.

We can then use these thresholds to remove low-quality cells.

```
keep_genes <- !apply(simplify2array(mfilt[!is.na(mfilt)]), 1, any)
```

Since the example dataset contains ERCC spike-ins [19], we can use them to further filter those cells that have too many reads mapped to them. This can be due to less viable cells or to a problem with the dilution of the spike-ins.

```
keep_ercc <- stats$altexps_ERCC_percent <= 5

table(keep_genes, keep_ercc)


##             keep_ercc

## keep_genes FALSE TRUE

##      FALSE    18   17

##      TRUE     46  298


sce <- sce[, keep_genes & keep_ercc]
```

Finally, we want to remove lowly expressed genes. To do so, we can use the `filterByExpr()` function of the `edgeR` package [20]. This function determines which genes have sufficiently large expression in at least one group. Here, we use each mouse as a different group.

```
keep <- filterByExpr(counts(sce), group = sce$Animal.ID)

table(keep)


## keep

## FALSE  TRUE

##  7128 13688


sce <- sce[keep,]
```

We do the same for the spike-ins, to remove spike-in genes with too few reads.

```
keep_spikes <- filterByExpr(counts(altExp(sce)), group = sce$Animal.ID)

altExp(sce) <- altExp(sce)[keep_spikes,]

sce

## class: SingleCellExperiment

## dim: 13688 298

## metadata(2): SuppInfo which_qc

## assays(1): counts

## rownames(13688): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3

## rowData names(0):

## colnames(298): SRR2140028 SRR2140022 ... SRR2139341 SRR2139336

## colData names(22): NREADS NALIGNED ... Animal.ID passes_qc_checks_s

## reducedDimNames(0):

## spikeNames(0):

## altExpNames(1): ERCC
```

Our filtered dataset has 13,688 genes and 298 cells.

3.3 Computing Normalization Factors

As noted in Section 1, several methods exist for the normalization of scRNA-seq data. Here, we show how to compute and apply normalization factors using the `scran` approach [4]. In Section 3.3, we will show how to compare different normalizations and select the best performing one.

Scran is a global scaling approach, meaning that it scales the count of all genes in each sample by a global factor. Hence, it can be thought of as a two-step procedure: (1) compute the normalization factors; (2) apply the normalization factors to obtain normalized data.

The `scran` approach uses pools of cells to compute size factors that are then deconvolved to individual cells. It is beneficial if the pooled cells are similar to each other [4]. Hence, a quick clustering step is suggested prior to normalization. Note that this only needs to be a rough grouping of similar cells and should not be thought of as a final, biologically meaningful, clustering of the data.

```
clust <- quickCluster(sce)

sce <- computeSumFactors(sce, cluster=clust, min.mean=0.1)
```

Note how the `computeSumFactors()` function takes a `SingleCellExperiment` object both as input and output. The output object contains information on the `sizeFactors`.

```
head(sizeFactors(sce))

## [1] 1.0334050 1.2082307 0.4291021 0.4578487 0.9734242 2.2759324
```

These size factors are a sample-specific constant used to scale the data to obtain the normalized counts.

We then apply the normalization factors and obtain the log-normalized counts.

```
sce <- logNormCounts(sce)

sce


## class: SingleCellExperiment

## dim: 13688 298

## metadata(2): SuppInfo which_qc

## assays(2): counts logcounts

## rownames(13688): 0610007P14Rik 0610009B22Rik ... Zzef1 Zzz3

## rowData names(0):

## colnames(298): SRR2140028 SRR2140022 ... SRR2139341 SRR2139336

## colData names(22): NREADS NALIGNED ... Animal.ID passes_qc_checks_s

## reducedDimNames(0):

## spikeNames(0):

## altExpNames(1): ERCC
```

Note how the `sce` object now contains a new `assay`, called `logcounts`, that contains the log-normalized data. To avoid problem with the log of zero, a "pseudo-count" of one is added to the counts prior to the log transformation.

3.4 Comparing and Selecting Normalization Methods

Normalization is an inherently difficult problem, and no single method outperforms all others in all scenarios. Instead, the methods' performances are data-dependent and it is essential to be able to rank normalization methods based on performance metrics calculated on the dataset at hand.

To this end, the `scone` package provides a set of data-driven metrics that can be used to rank normalization methods and choose the best performing approach [18].

Scone follows a modular approach. After a scaling step, it considers two methods to remove "unwanted variation," that is, the variability coming from technical rather than biological sources. To do so, `scone` employs either the QC metrics discussed above or a set of negative control genes [21]. *See* [18] for details.

One of the assumptions of the latter approach is the existence of a set of negative control genes, for example, genes whose expression does not change across cells. Since these data include spike-ins, we can use them as negative controls. In the absence of spike-ins, one can use prior knowledge (or prior data) to identify negative controls (*see* [21] for details).

To run `scone`, we first need to create a `SconeExperiment` object, which includes some useful information about the experiment. In addition to the gene expression data, we can optionally specify a number of the following details: biological origin of the cells (e.g., cell types, if known), QC metrics (scaled), a set of negative controls, a set of positive controls (e.g., known cell type markers), any known batch effects (e.g., day of the experiment) (*see* **Note 4**). Scone will use this information in the ranking of normalization methods. *See* the `scone` manual page for all the arguments that can be passed to the function.

```
# Expression data (note that we concatenate gene and spike-in

expression)

exprs <- rbind(counts(sce), counts(altExp(sce)))
```

```
# Scaled QC metrics

qc <- colData(sce)[,metadata(sce)$which_qc]

ppq <- scale(qc[,apply(qc,2,sd) > 0],center = TRUE, scale = TRUE)
```

```
# Negative controls

negcon <- rownames(altExp(sce))
```

```
# Positive controls: here we use a few neuronal markers

poscon <- intersect(rownames(sce), c("Gad1", "Gad2", "Slc32a1",

"Slc17a7", "Lamp5", "Ndnf", "Sncg", "Vip", "Sst", "Chodl", "Pvalb",

"Cux2", "Rorb", "Fezf2"))
```

```
se <- SconeExperiment(exprs,
                      qc=ppq,
                      negcon_ruv = rownames(exprs) %in% negcon,
                      poscon = rownames(exprs) %in% poscon
)
```

The second step of the workflow is to decide which normalization methods to compare. Scone includes a few wrapper functions for popular normalization methods.

```
scaling <- list(none=identity, # Identity: do nothing
             sum = SUM_FN,
             tmm = TMM_FN,
             deseq = DESEQ_FN,
             scran = SCRAN_FN)
```

In this case, we have used the simple scaling by total number of reads (sum), the TMM approach proposed in [22], the DESeq approach proposed in [23], and the scran approach [4]. Users can also define their own method. To illustrate this process, we add the Census transformation [5]. This approach is implemented in the `monocle` Bioconductor package. All we have to do is to create a wrapper function around it; the wrapper should take as input a matrix of raw counts and return a matrix with normalized data.

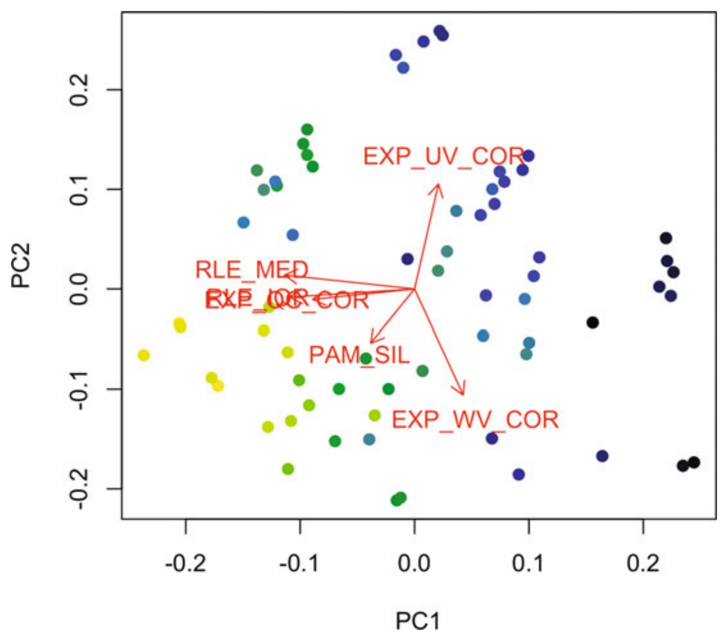


Fig. 1 Biplot of the first two principal components of scone's performance metrics. Each point represents a normalization method and is colored according to the average score ranking (the lighter the better). The arrows represent the metrics (*see* description in the text)

```
CENSUS_FN <- function(mat) {

  tpm <- calculateTPM(mat)

  cds <- newCellDataSet(tpm)

  census <- relative2abs(cds)

  return(census)

}



scaling$census <- CENSUS_FN
```

The third and final step is to run the `scone` function, which applies all selected normalizations to the data and rank them according to a set of evaluation metrics.

```
se <- scone(se,

            scaling = scaling,

            zero = "postadjust")
```

The `zero = "postadjust"` option ensures that the genes with zero reads in a given cell are left at zero and not transformed by the normalization procedure.

Scone computes a set of scores (in a variable number depending on the options passed to the `scone` function) and orders the normalization approaches from the best to the worst performer, based on the average rank of the scores.

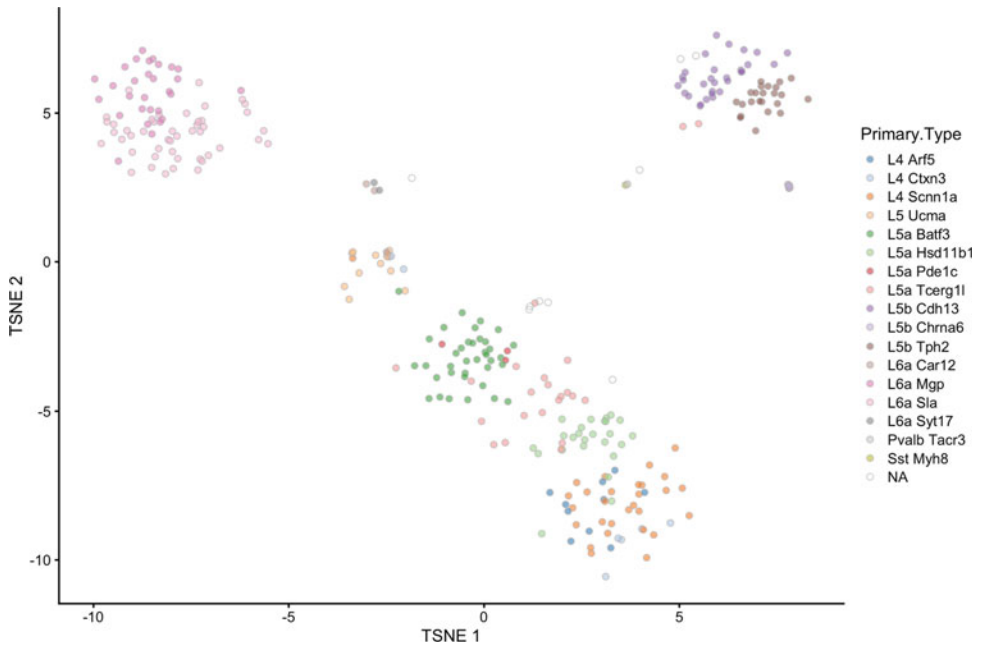


Fig. 2 t-SNE plot of the log-normalized data, colored by cell type

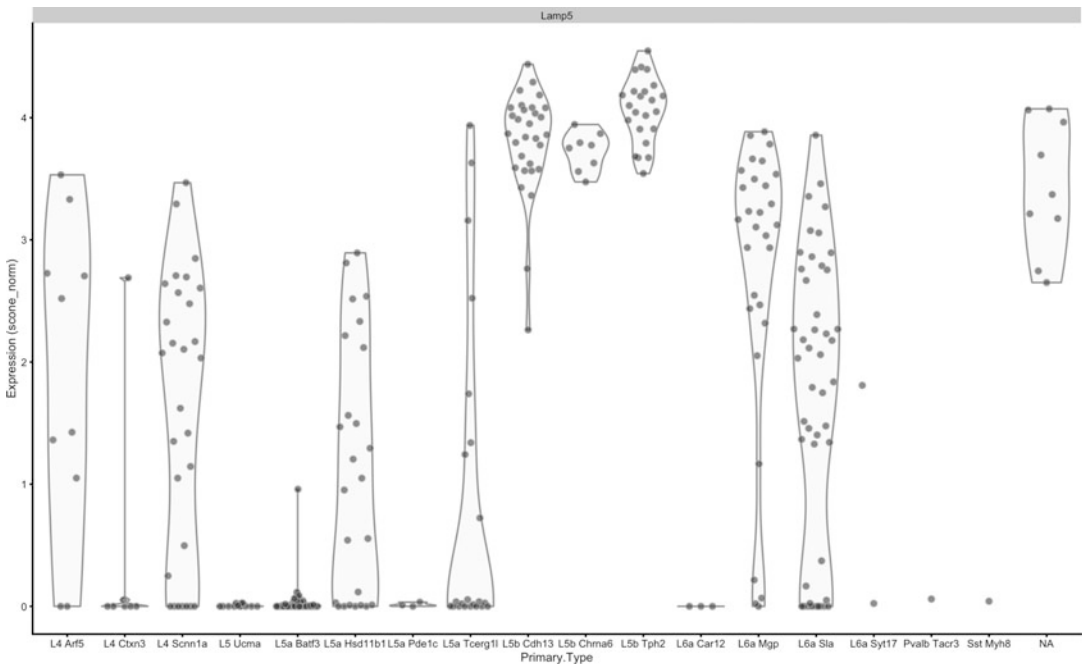


Fig. 3 Violin plot of Lamp5 log-normalized expression, grouped by cell type

```
head(round(get_scores(se), 3))

##                                      PAM_SIL  EXP_QC_COR  EXP_UV_COR
EXP_WV_COR
## none,census,qc_k=2,no_bio,no_batch   0.523      -0.304      -0.043
0.315
## none,census,qc_k=4,no_bio,no_batch   0.526      -0.285      -0.043
0.309
## none,census,qc_k=5,no_bio,no_batch   0.529      -0.256      -0.050
0.308
## none,census,qc_k=3,no_bio,no_batch   0.530      -0.291      -0.043
0.308
## none,tmm,qc_k=4,no_bio,no_batch      0.562      -0.256      -0.041
0.309
## none,tmm,qc_k=5,no_bio,no_batch      0.563      -0.235      -0.048
0.309
##                                      RLE_MED  RLE_IQR
## none,census,qc_k=2,no_bio,no_batch   0.000   -0.011
## none,census,qc_k=4,no_bio,no_batch   0.000   -0.011
## none,census,qc_k=5,no_bio,no_batch   0.000   -0.010
## none,census,qc_k=3,no_bio,no_batch   0.000   -0.011
## none,tmm,qc_k=4,no_bio,no_batch     -0.002   -0.037
## none,tmm,qc_k=5,no_bio,no_batch     -0.001   -0.035
```

In this case, the computed metrics are as follows:

1. PAM_SIL: the average silhouette width of the clusters obtained with Partition Around Medoids (PAM) [24]. The higher the score the better the normalization.
2. EXP_QC_COR: the weighted coefficient of determination for the regression of log-count principal components on all principal components of the QC metrics. This roughly corresponds to a correlation between normalized gene expression and sample quality. The lower the score the better the normalization.
3. EXP_UV_COR: same as 2, but for factors of unwanted variation derived from negative controls (*see* [18]). The lower the score the better the normalization.
4. EXP_WV_COR: same as 2, but for factors of wanted variation derived from positive controls (*see* [18]). The higher the score the better the normalization.
5. RLE_MED: Mean squared median relative log-expression (RLE).
6. RLE_IQR: Variance of interquartile range (IQR) of RLE.

For our example dataset, the best normalization seems to be Census [5], followed by regression of two QC factors (*see* [18] for details).

We can use a biplot to visualize the results (Fig. 1).

```
pc_obj <- prcomp(apply(t(get_scores(se)),1,rank),

                center = TRUE,scale = FALSE)

biplot_color(pc_obj, y = -get_score_ranks(se), expand = .6)
```

In the biplot shown in Fig. 1, the points are colored according to their average score ranks (the lighter the better). From the plot, we can see which metrics most influence the choice of the best normalization and whether several normalizations are performing similarly.

An interactive version of the plot can be created by using the `biplot_interactive()` function, which allows the user to click on the points and visualize the name of the normalization method.

Once we have decided which normalization to choose, we can obtain the normalized counts via the `get_normalize()` function.

Note that, for memory efficiency, `scone` does not save all normalized matrices, but computes the requested one only when needed.

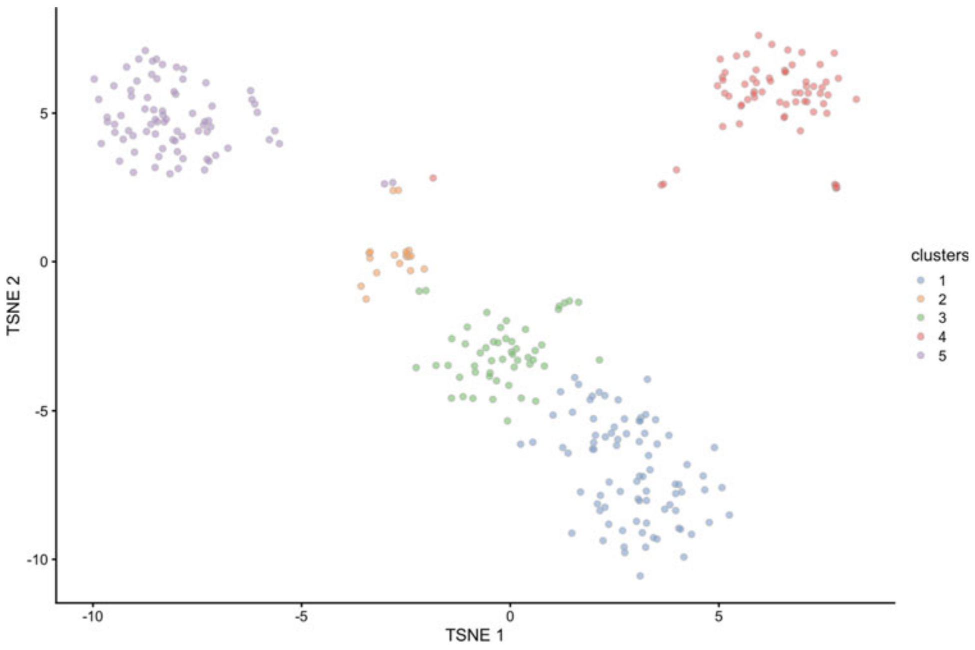


Fig. 4 t-SNE plot of the log-normalized data, colored by cluster

```r
norm <- get_normalized(se, method = rownames(get_scores(se))[1], log =

TRUE)
```

We can add the normalized matrix to the `SingleCellExperiment` object with the following command.

```r
assay(sce, "scone_norm") <- norm[rownames(sce),]
```

We can then use `scater` to visualize the data, for example, with a t-SNE plot [25] (Fig. 2).

```r
sce <- runTSNE(sce, exprs_values = "scone_norm")

plotTSNE(sce, colour_by = "Primary.Type")
```

The normalized data can be used directly for visualization, for example, in heatmaps or violin plots. Figure 3 shows how the gene Lamp5 is distributed among the cell types.

```
plotExpression(sce, "Lamp5", exprs_values = "scone_norm", x =

"Primary.Type", theme_size = 5)
```

3.5 Downstream Analyses

One of the most important applications of scRNA-seq is the identification of cell types (or states). In statistical terms, this corresponds to clustering.

Generally, we want to perform a dimensionality reduction step prior to clustering. One possibility is to compute the principal components (PCs) of the log-normalized data, and then run a graph-based clustering method, such as Louvain clustering [26], in PC space. Figure 4 shows the clustering results in a t-SNE plot.

```
sce <- runPCA(sce, exprs_values = "scone_norm")

g <- buildSNNGraph(sce, k=10, use.dimred = "PCA")

clust <- cluster_louvain(g)

sce$clusters <- as.factor(clust$membership)



plotTSNE(sce, colour_by = "clusters")
```

The most important parameter of this analysis is the number ($k = 10$) of nearest neighbors to consider in the creation of the graph. Decreasing such value will increase the number of clusters.

Finally, we can use the normalized data to identify gene markers for each cluster. This can be easily achieved via the `findMarkers()` function of the `scran` package.

```
genes <- findMarkers(sce, groups = sce$clusters,
                     test.type = "wilcox", assay.type = "scone_norm")
```

This function will return a list of five elements, one per each cluster. For instance, to look at marker genes for cluster 1, we can look at the first element of the list.

```
head(genes[[1]])

## DataFrame with 6 rows and 7 columns
##                Top              p.value                  FDR
AUC.2
##          <integer>            <numeric>            <numeric>
<numeric>
## Bcl6             1 4.96400402008946e-27 9.70675528956926e-24
0.487282463186078
## Ptn              1 6.18270344437476e-28 6.14214190955068e-24
0.77376171352075
## Tmsb10           1 5.02432753142714e-19  6.4273827336612e-17
0.506024096385542
## Ucma             1 1.58358003844407e-14 7.96913366405235e-13
0.166666666666667
## Dscaml1          2 6.01858545551281e-27 1.02977997143824e-23
0.463186077643909
## Foxp2            2 1.13559565130905e-27 6.14214190955068e-24
0.528781793842035
##                      AUC.3              AUC.4              AUC.5
##                  <numeric>          <numeric>          <numeric>
## Bcl6     0.499881880463029 0.0191199580932425  0.548192771084337
## Ptn      0.687219466099693  0.980443513183167                  1
## Tmsb10  0.0326009922041106  0.308887724812293  0.271944922547332
## Ucma     0.490196078431373  0.485507246376812                0.5
## Dscaml1  0.430191353649894 0.0303823991618648    0.2026286966046
## Foxp2    0.526340656744626  0.454688318491357 0.0225316851822875
```

Here, we used the nonparametric Wilcoxon rank sum test (`test.type = "wilcox"`) to test all pairwise differences between the groups. We should note that while it is reasonable to use the *p*-values to rank the genes and select the top ones as markers, these p-values do not have any inferential meaning, as the compared groups are defined from the same data (*see* [27] for a discussion).

We have shown one possible approach to cluster the normalized data and retrieve gene markers for each cluster. However, many different approaches exist for the dimensionality reduction, clustering, and differential expression of scRNA-seq data, and a comparison of such approaches is outside of the scope of this Chapter. *See* Soneson and Robinson [28] for a systematic evaluation of differential expression methods, Sun et al. [29] for a review and comparison of dimensionality reduction methods, and Duò et al. [30] for a benchmark of clustering methods.

4 Notes

1. *Differences between UMI and read count data.* The example dataset that we used here did not make use of Unique Molecular Identifiers (UMIs). It has been shown that UMIs effectively reduce much of the amplification bias observed in scRNAseq [2]. In addition the use of UMIs decreases the need to model zero inflation [10]. However, the use of UMIs does not eliminate the need for normalization [2] and much of what is shown here applies to UMI data as well.
2. *Computational complexity.* Some of the steps highlighted in this Chapter are computationally intensive. In particular, `scone` computes several normalizations and may be slow for large datasets. Enabling parallel computations helps (*see* below), but for very large datasets, we suggest to run scone on a subset of data and apply to the full dataset only the chosen normalization.
3. *Parallel computing.* Most functions presented here can leverage parallel computing to increase speed for large datasets. They all depend on the `BiocParallel` package that allows the user to set a backend and the number of threads to use. By default parallel computing is turned off. To enable it, one can use the following command at the beginning of the workflow (where `ncores` should be substituted with the number of cores to use).

```
## Available only for unix systems (including Mac)

BiocParallel::register(BiocParallel::MulticoreParam(ncores))



## OR



## Also available for Windows

BiocParallel::register(BiocParallel::SnowParam(ncores))
```

4. *Dealing with batch effects.* When combining data from different runs, protocols, or laboratories, the data are often affected by so-called batch effects. The methods presented above, for example, regressing out QC metrics, can ameliorate these effects, but a more direct approach may be beneficial. The `batchelor` package implements several strategies to remove known batch effects, for example, the Mutual Nearest Neighbor (MNN) method of [31]. Alternatively, scone allows the use of a "batch covariate" that can be included in the normalization model to adjust for batch effects.

References

1. Bullard JH, Purdom E, Hansen KD, Dudoit S (2010) Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. BMC Bioinformatics 11:94. https://doi.org/10.1186/1471-2105-11-94
2. Vallejos CA, Risso D, Scialdone A, Dudoit S, Marioni JC (2017) Normalizing single-cell RNA sequencing data: challenges and opportunities. Nat Methods 14(6):565–571. https://doi.org/10.1038/nmeth.4292
3. Vieth B, Parekh S, Ziegenhain C, Enard W, Hellmann I (2019) A systematic evaluation of single cell RNA-seq analysis pipelines. Nat Commun 10(1):4667. https://doi.org/10.1038/s41467-019-12266-7
4. Lun AT, Bach K, Marioni JC (2016) Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. Genome Biol 17:75. https://doi.org/10.1186/s13059-016-0947-7
5. Qiu X, Hill A, Packer J, Lin D, Ma YA, Trapnell C (2017) Single-cell mRNA quantification and differential analysis with Census. Nat Methods 14(3):309–315. https://doi.org/10.1038/nmeth.4150
6. Bacher R, Chu LF, Leng N, Gasch AP, Thomson JA, Stewart RM, Newton M, Kendziorski C (2017) SCnorm: robust normalization of single-cell RNA-seq data. Nat Methods 14(6):584–586. https://doi.org/10.1038/nmeth.4263
7. Townes FW, Irizarry RA (2020) Quantile normalization of single-cell RNA-seq read counts without unique molecular identifiers. Genome Biol 21:160 https://doi.org/10.1186/s13059-020-02078-0
8. Vallejos CA, Marioni JC, Richardson S (2015) BASiCS: Bayesian analysis of single-cell

sequencing data. PLoS Comput Biol 11(6): e1004333. https://doi.org/10.1371/journal.pcbi.1004333

9. Risso D, Perraudeau F, Gribkova S, Dudoit S, Vert JP (2018) A general and flexible method for signal extraction from single-cell RNA-seq data. Nat Commun 9(1):284. https://doi.org/10.1038/s41467-017-02554-5
10. Townes FW, Hicks SC, Aryee MJ, Irizarry RA (2019) Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. Genome Biol 20(1):295. https://doi.org/10.1186/s13059-019-1861-6
11. Huber W, Carey VJ, Gentleman R, Anders S, Carlson M, Carvalho BS, Bravo HC, Davis S, Gatto L, Girke T, Gottardo R, Hahne F, Hansen KD, Irizarry RA, Lawrence M, Love MI, MacDonald J, Obenchain V, Oleś AK, Pagès H, Reyes A, Shannon P, Smyth GK, Tenenbaum D, Waldron L, Morgan M (2015) Orchestrating high-throughput genomic analysis with Bioconductor. Nat Methods 12(2):115–121. https://doi.org/10.1038/nmeth.3252
12. Amezquita RA, Lun ATL, Becht E, Carey VJ, Carpp LN, Geistlinger L, Marini F, Rue-Albrecht K, Risso D, Soneson C, Waldron L, Pagès H, Smith ML, Huber W, Morgan M, Gottardo R, Hicks SC (2020) Orchestrating single-cell analysis with Bioconductor. Nat Methods 17(2):137–145. https://doi.org/10.1038/s41592-019-0654-x
13. Lun ATL, Pagès H, Smith ML (2018) beachmat: a Bioconductor C++ API for accessing high-throughput biological data from a variety of R matrix types. PLoS Comput Biol 14(5): e1006135. https://doi.org/10.1371/journal.pcbi.1006135
14. Tasic B, Menon V, Nguyen TN, Kim TK, Jarsky T, Yao Z, Levi B, Gray LT, Sorensen SA, Dolbeare T, Bertagnolli D, Goldy J, Shapovalova N, Parry S, Lee C, Smith K, Bernard A, Madisen L, Sunkin SM, Hawrylycz M, Koch C, Zeng H (2016) Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. Nat Neurosci 19(2):335–346. https://doi.org/10.1038/nn.4216
15. Svensson V (2020) Droplet scRNA-seq is not zero-inflated. Nat Biotechnol 38(2):147–150. https://doi.org/10.1038/s41587-019-0379-5
16. Stoeckius M, Hafemeister C, Stephenson W, Houck-Loomis B, Chattopadhyay PK, Swerdlow H, Satija R, Smibert P (2017) Simultaneous epitope and transcriptome measurement in single cells. Nat Methods 14(9):865–868. https://doi.org/10.1038/nmeth.4380
17. McCarthy DJ, Campbell KR, Lun AT, Wills QF (2017) Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. Bioinformatics 33(8):1179–1186. https://doi.org/10.1093/bioinformatics/btw777
18. Cole MB, Risso D, Wagner A, DeTomaso D, Ngai J, Purdom E, Dudoit S, Yosef N (2019) Performance assessment and selection of normalization procedures for single-cell RNA-Seq. Cell Syst 8(4):315–328.e318. https://doi.org/10.1016/j.cels.2019.03.010
19. Jiang L, Schlesinger F, Davis CA, Zhang Y, Li R, Salit M, Gingeras TR, Oliver B (2011) Synthetic spike-in standards for RNA-seq experiments. Genome Res 21(9):1543–1551. https://doi.org/10.1101/gr.121095.111
20. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26(1):139–140. https://doi.org/10.1093/bioinformatics/btp616
21. Risso D, Ngai J, Speed TP, Dudoit S (2014) Normalization of RNA-seq data using factor analysis of control genes or samples. Nat Biotechnol 32(9):896–902. https://doi.org/10.1038/nbt.2931
22. Robinson MD, Oshlack A (2010) A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biol 11(3):R25. https://doi.org/10.1186/gb-2010-11-3-r25
23. Anders S, Huber W (2010) Differential expression analysis for sequence count data. Genome Biol 11(10):R106. https://doi.org/10.1186/gb-2010-11-10-r106
24. Kaufman L, Rousseeuw PJ (2009) Finding groups in data: an introduction to cluster analysis, vol 344. John Wiley & Sons, Hoboken, NJ
25. Maaten Lvd HG (2008) Visualizing data using t-SNE. J Mach Learn Res 9(Nov):2579–2605
26. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):P10008
27. Zhang JM, Kamath GM, Tse DN (2019) Valid post-clustering differential analysis for single-cell RNA-Seq. Cell Syst 9(4):383–392.e386. https://doi.org/10.1016/j.cels.2019.07.012
28. Soneson C, Robinson MD (2018) Bias, robustness and scalability in single-cell differential expression analysis. Nat Methods 15(4):255–261. https://doi.org/10.1038/nmeth.4612

29. Sun S, Zhu J, Ma Y, Zhou X (2019) Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. Genome Biol 20(1):269

30. Duò A, Robinson MD, Soneson C (2018) A systematic performance evaluation of clustering methods for single-cell RNA-seq data. F1000Res 7

31. Haghverdi L, Lun ATL, Morgan MD, Marioni JC (2018) Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nat Biotechnol 36(5):421–427. https://doi.org/10.1038/nbt.4091

Chapter 18

Dimensionality Reduction of Single-Cell RNA-Seq Data

George C. Linderman

Abstract

Dimensionality reduction is a crucial step in essentially every single-cell RNA-sequencing (scRNA-seq) analysis. In this chapter, we describe the typical dimensionality reduction workflow that is used for scRNA-seq datasets, specifically highlighting the roles of principal component analysis, t-distributed stochastic neighborhood embedding, and uniform manifold approximation and projection in this setting. We particularly emphasize efficient computation; the software implementations used in this chapter can scale to datasets with millions of cells.

Key words scRNA-seq, Dimensionality-reduction, Visualization, pca, t-SNE, umap

1 Introduction

Recent advances in single-cell RNA-sequencing (scRNA-seq) technology have allowed researchers to study cell-to-cell heterogeneity at unprecedented resolution. While bulk RNA-sequencing averages the expression of all cells in a sample, scRNA-seq allows for quantification of gene expression at the single-cell level. The scale of scRNA-seq experiments has been growing exponentially [1, 2], with recent technology allowing for the expression of millions of cells to be quantified in a single experiment. The inherent complexity of gene expression, combined with the massive amounts of data being generated, makes subsequent analysis challenging. New methods continue to be developed for analysis of scRNA-seq data, and best practices are only beginning to be established.

Despite the diversity of methods used to analyze scRNA-seq data, essentially every analysis pipeline is built upon dimensionality reduction. Gene expression data is inherently high dimensionality: it involves measuring the expression of the ~20,000 genes in the human genome. Each cell is thus a point in the ~20,000 dimensional "gene-space." Many approaches to data analysis and visualization suffer from the curse of dimensionality [3], namely, that they do not scale to high dimensions. For this reason, it is common to first reduce the dimensionality of the dataset, representing each cell by a point in a lower dimensional space (e.g., 50–100 dimensions) prior to proceeding with the subsequent analysis. The overall goal is to reduce the dimension of the dataset while preserving as much structure as possible.

In this chapter, we present the dimensionality reduction methods that are most commonly used in the analysis of scRNA-seq data. We will first center our attention on principal component analysis [4], which is the first step in most scRNA-seq analyses. Then we will discuss nonlinear methods for dimensionality reduction, with a focus on t-SNE [5] and UMAP [6], both of which have become standard tools for visualization of scRNA-seq data.

2 Materials

2.1 scRNA-Seq Dataset

The input data for our analysis will be the expression matrix obtained from a scRNA-seq experiment after alignment. In this expression matrix, rows correspond to genes and columns to cells. Most published scRNA-seq studies deposit this expression matrix on a public repository, such as the Gene Expression Omnibus (GEO), as required by major journals. For demonstration purposes, we will analyze a dataset from Hrvatin et al. [7], with GEO accession number GSE102827, which contains 65,539 mouse cortical cells. The expression matrix GSE102827_MATRIX.csv.gz can be directly downloaded from GEO at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE102827 and subsequently extracted to a comma-separated values (CSV) file containing the expression values. In that paper, the authors assigned each cell a label corresponding to its putative cell type. These cell type assignments are contained in the file GSE102827_cell_type_assignments.csv.gz that can be downloaded from the above link.

2.2 Computational Hardware

Analysis of scRNA-seq datasets can be computationally intensive. The most common computational bottleneck is memory, as simply storing a full scRNA-seq dataset in memory can be prohibitive for personal computers. If the matrix is stored as doubles (8 bytes), then the required gigabytes of memory to simply store the matrix is 8(number of cells)(number of genes)(10^{-9}). For example, the Hrvatin et al. dataset used here requires 13.2 GB of memory to load with R (*see* **Note 1**). CPU performance is not as critical, but can significantly speed up the analysis at certain key steps. In particular, multithreaded implementations of the dimensionality reduction tools used in this chapter can significantly speed up the analysis.

2.3 Software

Both R and Python are widely used in the scRNA-seq community, and the tools discussed in this chapter are implemented for both environments. PCA, t-SNE, and UMAP will be discussed in detail later; here we provide links to where they can be obtained.

The standard function for computing PCA in R, princomp(), does not scale to datasets of the size frequently encountered in scRNA-seq datasets. Either Lanczos methods (e.g., PROPACK [8]) or randomized methods [9] for fast PCA are commonly used instead. An R implementation of Lanczos methods for PCA is included in the CRAN package irlba [10], and an implementation of randomized PCA is available from the CRAN package rsvd [11]. Python implements both in the scikit-learn [12] package as sklearn.decomposition.PCA.

The original t-SNE implementation also did not scale beyond several thousand points, so essentially all scRNA-seq analyses use a fast t-SNE implementation. We recently developed FFT-accelerated, Interpolation-based t-SNE (FIt-SNE) [12], which is an order of magnitude faster than the previously fastest implementation (Barnes-Hut t-SNE) [13]. FIt-SNE has R and Python wrappers (https://github.com/KlugerLab/FIt-SNE), and it is also available on PyPI as part of the opentsne [14] package. We also note that several GPU implementations of t-SNE have been recently developed (e.g., [15]), which can provide even further speed-up when run on the appropriate hardware.

UMAP has recently become widely popular for visualization of scRNA-seq datasets [6, 16]. The original Python version is available on PyPI as umap-learn which has an R wrapper available on CRAN as umap. A pure R version called uwot is also available on CRAN.

There are also several toolkits for analyzing scRNA-seq data which aim to streamline the analysis pipeline. The most popular of these is Seurat [17, 18], which can be downloaded from CRAN and includes most of the previously mentioned dimensionality reduction methods. For Python users, scanpy [19] is also becoming popular. In this chapter, we perform the dimensionality reduction using the native tools (i.e., not using any wrappers).

3 Methods

3.1 Preprocessing

Quality control, normalization, batch effect correction, and other preprocessing steps are crucial parts of any scRNA-seq analysis pipeline that are typically performed prior to dimensionality reduction. Although these steps can dramatically impact the results of subsequent analysis, their discussion is beyond the scope of this chapter. We mention them only briefly and refer the reader to [20] for more details.

The first step of most single-cell analysis pipelines is quality control, where "low quality" cells are removed. The most common measure of cell quality is the number of reads or genes detected, where cells above or below certain thresholds are removed from subsequent analysis. For example, Hrvatin et al. removed cells with fewer than 700 or more than 15,000 read counts, shrinking the dataset from 114,601 cells to the 64,539 cells that we downloaded above.

Due to the inherently stochastic nature of the sequencing process, identical cells can have different count depth. Various normalization approaches have been proposed to remove this variation. The most common is to normalize each cell to have the same number of counts. Following Hrvatin et al., we normalize the dataset to contain 10,000 transcripts.

```
A <- read.csv('GSE102827_MATRIX.csv', header =T, row.names=1)
totalReadsPerCell <- colSums(A)
A_norm<-sweep(A,2,totalReadsPerCell,'/')*mean(totalReadsPerCell)
```

Next, a variance-stabilizing transformation (VST) is typically applied to the data. A VST is typically used when the features follow a distribution where the variance is dependent on the mean, as this can be problematic for PCA. For example, a gene that is expressed at high levels in all cells will necessarily have a high variance, even if in reality all cells express that gene at almost the same level. Assuming scRNA-seq data is generated from a negative binomial distribution, the log-transform can be shown to approximately remove the dependence of the variance on the mean. A "pseudo-count" of 1 is typically added to the expression values prior to applying the log-transform, as log(0) is not defined (*see* **Notes 2** and **3**).

```
A_norm <- log(A_norm + 1)
```

3.2 Principal Component Analysis

Each cell in a single-cell RNA-seq dataset is represented as a point in ~20,000-dimensional space, where each dimension corresponds to single gene. Biologically, genes do not act independently, but instead form groups of highly correlated genes often referred to as "gene modules" or "pathways." Given this intricate correlation structure, gene expression is highly redundant: a much smaller number of dimensions can capture the biological variability in the dataset. Principal component analysis (PCA) is used to construct a small number of uncorrelated variables (principal components or PCs) that capture the maximum amount of variability in the original dataset. PCA is the workhorse of essentially any high dimensional analysis, as it reduces the number of dimensions to a number that is manageable for downstream analysis.

The principal components of a dataset are defined as the eigenvectors of the data covariance matrix, and they are typically computed using the singular value decomposition (SVD). However, standard implementations of SVD are memory and time intensive because they compute all of the singular vectors and hence do not scale to the size of datasets encountered in scRNA-seq datasets. There are two major classes of fast methods for SVD that can be used to efficiently compute the leading singular vectors: Lanczos-based approximations (e.g., irlba in R) and randomized algorithms (e.g., rsvd). A comparison of these methods is beyond the scope of this chapter; we will use a randomized algorithm here (*see* **Note 4**).

To compute PCA, we first center the rows, and then compute the leading 100 singular vectors (*see* **Note 5**).

```
library(rsvd)
row_gene_means <- rowMeans(A_norm)
A_norm_c <- sweep(A_norm, 1, row_gene_means, '-')
svdout <- rsvd(A_norm_c, k=100)
obsPCA <- svdout$v %*% diag(svdout$d)
```

Plotting the first few singular vectors (Fig. 1), we see that while cells of the same type are generally close together, the clusters are not clearly separated.

```
library(ggplot2)
ct <- read.csv('GSE102827_cell_type_assignments.csv',
       header = TRUE, row.names = 1, stringsAsFactors = FALSE)
df <- data.frame(obsPCA[,1:2], Y= as.factor(ct$celltype))
df <- df[!is.na(df$Y),] #Only keep cells with assigned cell type
(g <- ggplot(df) +
       geom_point(aes(x=X1, y=X2, color=Y), size=0.5)
+ theme(legend.position="none") + labs(x="PC 1", y="PC 2"))
```

To see why, we can plot the "scree plot," or the variance captured by each principal component (Fig. 2).

```
total_var <- sum(rowMeans(A_norm_c^2))
component_var <- svdout$d^2/(ncol(A_norm_c)-1)
fraction_var_explained <- component_var/total_var

(g <- qplot(x=1:100,y=fraction_var_explained) +
           labs( y='Proportion of variance', x="Index") )
```

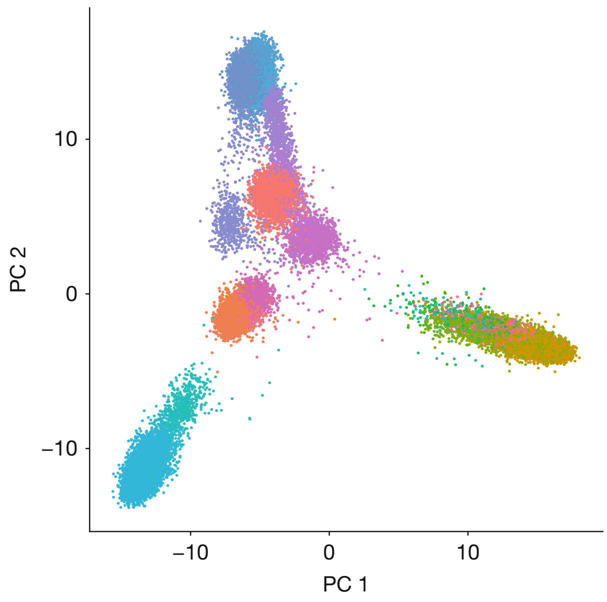


Fig. 1 The first and second PCs of the Hrvatin et al. dataset. Each color corresponds to a different cell type, as assigned by the original authors. The cell types are generally clustered together, but not distinct from each other

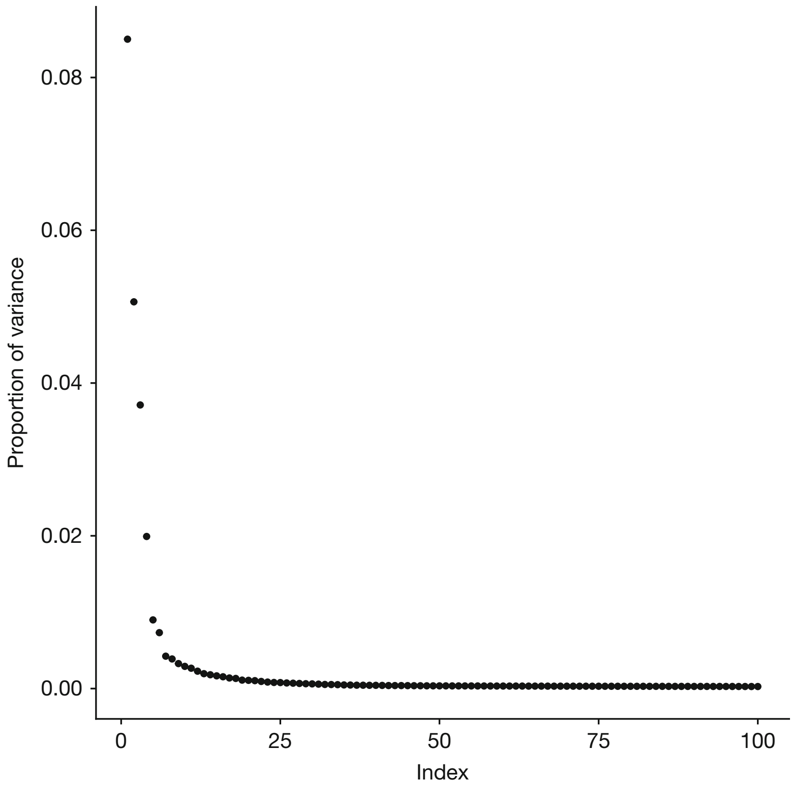


Fig. 2 The singular values of the centered matrix can be normalized to obtain the proportion of variance captured by each singular vector

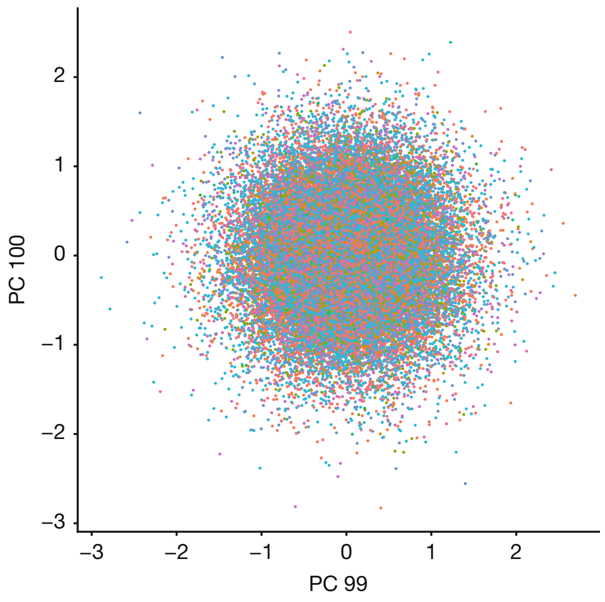


Fig. 3 The 99th and 100th PCs of the Hrvatin et al. dataset

The first two PCs only capture 14% of the variance. Clearly, the subsequent PCs contain important biological variability that cannot be ignored. But we do not want to retain all of the PCs, because eventually they are just noise. For example, the 99th and 100th singular vectors are clearly meaningless (Fig. 3), and including the noise may negatively impact subsequent analysis.

For this reason, the number of PCs to retain is an important consideration when doing PCA, and it is the subject of continued research. The goal is to identify the point at which the PCs no longer correspond to biological variability. One common approach is to look for a gap or “elbow” in the scree plot, but as can be seen in Fig. 2, this can be fairly arbitrary. Another common approach is to use a permutation method, like parallel analysis [21] or the related “jackstraw” method [22]. The general idea of these approaches is to independently shuffle the entries of each column to obtain “fake” data matrices. Next, the SVD of each “fake” matrix is computed, obtaining a null distribution for each singular value. Finally, each singular value of the observed matrix is compared with the corresponding null distribution, and only the singular values that exceed a given percentile are retained. The intuition is that the shuffled matrices contain the same values as the observed matrix, but the correlation structure was “destroyed” by the permutation process and hence approximate the noise inherent of the data.

3.3 Two-Dimensional Visualization

As observed in the previous section, more than two PCs are generally necessary to capture the biological variation in scRNA-seq datasets. Many subsequent analysis steps (e.g., clustering) can be performed in the 30–100-dimensional space formed by the leading PCs. However, much of scRNA-seq analysis is driven by visualization, making further dimensionality reduction necessary. T-distributed stochastic neighborhood embedding (t-SNE) and uniform manifold approximation and projections (UMAP) have become widely used for this purpose. Intuitively, these nonlinear methods force the data into a two or three-dimensional space by preserving the short-range (i.e., local) distances between points, often at the cost of the longer-range (i.e., global) distances.

The optimization underlying t-SNE produces an embedding where points that were close in the (high-dimensional) original space are also close in the (low-dimensional) embedding space. To do so, the algorithm minimizes the divergence between a Gaussian distribution modeling affinities between points in the original space and a Student's t-distribution modeling affinities in the embedding space.

The embedding is achieved using gradient descent, where points start in some initial configuration (by default, random) and then are iteratively updated to minimize the divergence. UMAP also uses a variant of gradient descent, but optimizes a different cost function.

Crucially, parameters suggested in the original t-SNE publications (which are default in some implementations) are not optimal for embedding large scRNA-seq datasets [23]. In particular, t-SNE's preservation of global structure can be dramatically improved by initializing the embedding with the first two PCs (which are typically rescaled to have standard deviation 0.0001). Furthermore, the default learning rate of 200 is too small for embedding large datasets (e.g., $n > 100{,}000$), so we follow [23] and use a learning rate of $\sim n/12$ in those settings. We refer to their article for more sophisticated applications of t-SNE to scRNA-seq data. We now embed the dataset using FIt-SNE, with PCA initialization (Fig. 4).

```
source('<path to FIt-SNE>/fast_tsne.R', chdir=T)
init <- 0.0001*(obsPCA[,1:2]/sd(obsPCA[,1]))
k <- 30 # Number of PCs chosen by Hrvatin et al.
fitsneout <- fftRtsne( obsPCA[,1:k],
                        initialization = init, rand_seed=3)
```

We can also embed the data using UMAP, as follows (Fig. 5).

```
library(umap)
umapout <- umap(obsPCA[,1:k])
```

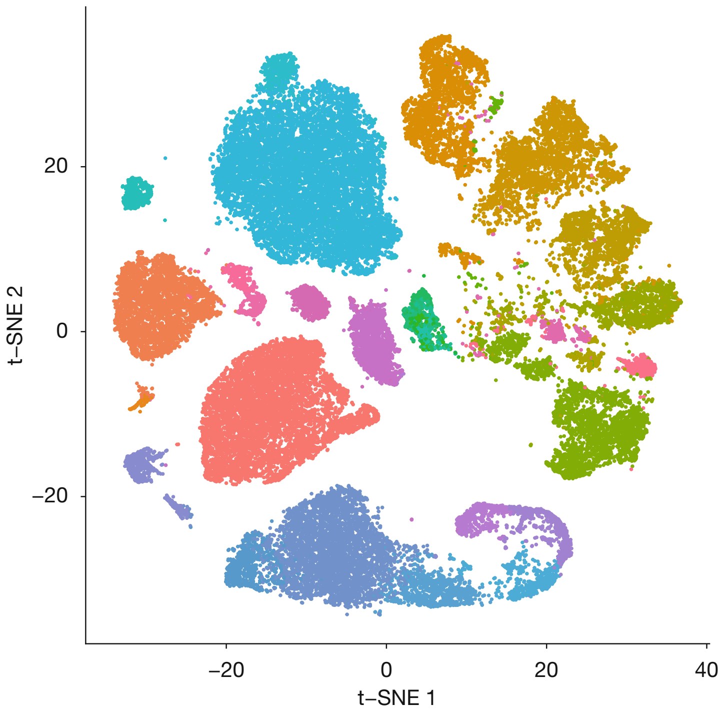


Fig. 4 t-SNE of the top 30 PCs of the Hrvatin et al. dataset

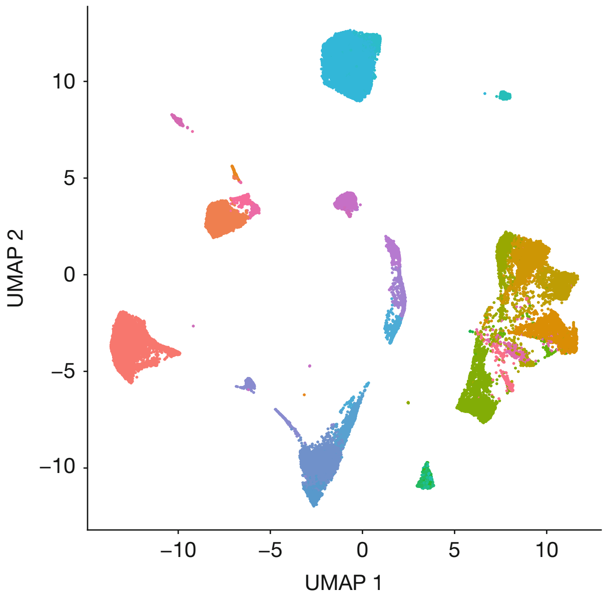


Fig. 5 UMAP of the top 30 PCs of the Hrvatin et al. dataset

The resulting embedding is generally consistent with t-SNE, although the UMAP embedding has markedly more white space. In general, further research is necessary to determine which algorithm more faithfully represents scRNA-seq data. The FIt-SNE implementation of t-SNE is at least as fast as UMAP, and when initialized using PCA, it preserves global distances as well as UMAP does [24].

UMAP and t-SNE are the two most popular methods used for visualization of scRNA-seq data, but they are generic methods that are not specifically designed for scRNA-seq datasets. In contrast, a large number of dimensionality reduction methods have been developed specifically for scRNA-seq datasets. We refer the reader to [25–27] and the references therein for more details.

4 Notes

1. Given that the vast majority of values in the scRNA-seq dataset are zero, the data can be stored more compactly as a sparse matrix. Unlike a dense matrix which stores every value, a sparse matrix stores only the nonzero values. For example, the 10X cellranger pipeline uses the Market Exchange Format (MEX) for sparse matrices. On machines which have limited RAM, it is possible to keep the matrix in sparse format, as the preprocessing steps outlined above (normalization, log-transform) do not make the matrix dense. However, an SVD implementation that can be run on a sparse matrix and that supports row/column centering (e.g., irlba() in R) should be used. In these implementations, the SVD of the centered matrix is computed without actually forming the centered matrix (which would be dense). After the top PCs are computed, the subsequent analysis can proceed using dense matrices, since the number of PCs retained is much smaller than the number of genes.
2. If the reads are assumed to be generated from a Poisson distribution, the variance-stabilizing transformation is a square-root. This transformation is sometimes used in place of the log-transform.
3. The log-transform and the associated "pseudo-count" as described above are common preprocessing steps but were recently shown by the authors of [28] to bias the transformed data. Instead, they developed a method for generalized principal component analysis (GLM-PCA) based on a multinomial sampling model, which can be used directly on the count data. An R implementation called glmpca is available on CRAN and may be considered an alternative to the standard PCA-based pipeline.

4. The basic operation underlying randomized SVD is matrix multiplication, which is highly parallelizable. However, the rsvd package used here is single-threaded. Dramatic speed improvements can be obtained using multithreaded implementations, but they are not available in R (e.g., [29] in MATLAB).
5. It is widespread practice to retain only the 1000–5000 most variable genes for subsequent analysis. When using standard tools for PCA, this feature selection step is often necessary in order to make the dataset more computationally manageable. The fast methods described in this chapter make this step optional, as they can be run on the full dataset. However, the feature selection step may have an important denoising effect on some analyses, and hence it may still be included for that reason. This can be done using Seurat's VariableGenes() function.

Acknowledgments

Many thanks to Dmitry Kobak for helpful comments on a draft of this chapter.

References

1. Svensson V, da Veiga Beltrame E, Pachter L (2020) A curated database reveals trends in single-cell transcriptomics. Database 2020
2. Svensson V, Vento-Tormo R, Teichmann SA (2018) Exponential scaling of single-cell RNA-seq in the past decade. Nat Protoc 13 (4):599–604
3. Bellman RE (1961) Adaptive control processes: a guided tour. Princeton University Press, Princeton, N.J
4. Jolliffe IT (1986) Principal component analysis and factor analysis. In: Principal component analysis. Springer, New York, pp 115–128
5. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
6. Becht E, McInnes L, Healy J, Dutertre C-A, Kwok IW, Ng LG, Ginhoux F, Newell EW (2019) Dimensionality reduction for visualizing single-cell data using UMAP. Nat Biotechnol 37(1):38
7. Hrvatin S, Hochbaum DR, Nagy MA, Cicconet M, Robertson K, Cheadle L, Zilionis R, Ratner A, Borges-Monroy R, Klein AM (2018) Single-cell analysis of experience-dependent transcriptomic states in the mouse visual cortex. Nat Neurosci 21(1):120
8. Larsen RM (1998) Lanczos bidiagonalization with partial reorthogonalization. DAIMI Rep Ser 27(537)
9. Halko N, Martinsson P-G, Tropp JA (2011) Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev 53 (2):217–288
10. Baglama J, Reichel L, Lewis B (2017) irlba: Fast truncated singular value decomposition and principal components analysis for large dense and sparse matrices. R package version 2 (1)
11. Erichson NB, et al. (2019) Randomized matrix decompositions using R. J Stat Softw 89 (1):1–48
12. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12(Oct):2825–2830
13. Van Der Maaten L (2014) Accelerating t-SNE using tree-based algorithms. J Mach Learn Res 15(1):3221–3245
14. Policar PG, Strazar M, Zupan B (2019) openTSNE: a modular Python library for t-SNE

dimensionality reduction and embedding. BioRxiv:731877
15. Chan DM, Rao R, Huang F, Canny JF (2019) GPU accelerated t-distributed stochastic neighbor embedding. J Parallel Distrib Comput 131:1–13
16. McInnes L, Healy J, Melville J (2018) Umap: uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:180203426
17. Butler A, Hoffman P, Smibert P, Papalexi E, Satija R (2018) Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol 36 (5):411–420
18. Stuart T, Butler A, Hoffman P, Hafemeister C, Papalexi E, Mauck IIIWM, Hao Y, Stoeckius M, Smibert P, Satija R (2019) Comprehensive integration of single-cell data. Cell 177(7):1888–1902.e1821
19. Wolf FA, Angerer P, Theis FJ (2018) SCANPY: large-scale single-cell gene expression data analysis. Genome Biol 19(1):15
20. Luecken MD, Theis FJ (2019) Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol 15(6)
21. Horn JL (1965) A rationale and test for the number of factors in factor analysis. Psychometrika 30(2):179–185
22. Chung NC, Storey JD (2015) Statistical significance of variables driving systematic variation in high-dimensional data. Bioinformatics 31 (4):545–554
23. Kobak D, Berens P (2019) The art of using t-SNE for single-cell transcriptomics. Nat Commun 10(1):1–14
24. Kobak D, Linderman GC (2019) UMAP does not preserve global structure any better than t-SNE when using the same initialization. bioRxiv
25. Moon KR, Stanley JS III, Burkhardt D, van Dijk D, Wolf G, Krishnaswamy S (2018) Manifold learning-based methods for analyzing single-cell RNA-sequencing data. Curr Opin Syst Biol 7:36–46
26. Sun S, Zhu J, Ma Y, Zhou X (2019) Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis. Genome Biol 20(1):269
27. Çakır B, Prete M, Huang N, van Dongen S, Pir P, Kiselev VY (2020) Comparison of visualization tools for single-cell RNAseq data. NAR Genomics and Bioinformatics 2(3):lqaa052. https://doi.org/10.1093/nargab/lqaa052
28. Townes FW, Hicks SC, Aryee MJ, Irizarry RA (2019) Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. Genome Biol 20(1):1–16
29. Li H, Linderman GC, Szlam A, Stanton KP, Kluger Y, Tygert M (2017) Algorithm 971: an implementation of a randomized algorithm for principal component analysis. ACM Trans Math Softw 43(3):1–14

Chapter 19

Single-Cell RNA Sequencing Analysis: A Step-by-Step Overview

Shaked Slovin, Annamaria Carissimo, Francesco Panariello, Antonio Grimaldi, Valentina Bouché, Gennaro Gambardella, and Davide Cacchiarelli

Abstract

Thanks to innovative sample-preparation and sequencing technologies, gene expression in individual cells can now be measured for thousands of cells in a single experiment. Since its introduction, single-cell RNA sequencing (scRNA-seq) approaches have revolutionized the genomics field as they created unprecedented opportunities for resolving cell heterogeneity by exploring gene expression profiles at a single-cell resolution. However, the rapidly evolving field of scRNA-seq invoked the emergence of various analytics approaches aimed to maximize the full potential of this novel strategy. Unlike population-based RNA sequencing approaches, scRNA seq necessitates comprehensive computational tools to address high data complexity and keep up with the emerging single-cell associated challenges. Despite the vast number of analytical methods, a universal standardization is lacking. While this reflects the fields' immaturity, it may also encumber a newcomer to blend in.

In this review, we aim to bridge over the abovementioned hurdle and propose four ready-to-use pipelines for scRNA-seq analysis easily accessible by a newcomer, that could fit various biological data types. Here we provide an overview of the currently available single-cell technologies for cell isolation and library preparation and a step by step guide that covers the entire canonical analytic workflow to analyse scRNA-seq data including read mapping, quality controls, gene expression quantification, normalization, feature selection, dimensionality reduction, and cell clustering useful for trajectory inference and differential expression. Such workflow guidelines will escort novices as well as expert users in the analysis of complex scRNA-seq datasets, thus further expanding the research potential of single-cell approaches in basic science, and envisaging its future implementation as best practice in the field.

Key words Single-cell RNA-seq, Experimental workflow, Data analysis tutorial, Computational pipelines, Clustering, Monocle, Seurat, gf-icf, Scanpy

Shaked Slovin, Annamaria Carissimo and Francesco Panariello contributed equally with all other contributors.

1 Introduction

Throughout the last decade, population-based RNA sequencing approaches (aka bulk RNA-seq) have played a significant role in deciphering genome-wide transcriptome variations across a broad range of fields, including cancer biology, developmental biology, and cellular homeostasis [1–3]. However, as bulk RNA-seq data represents an average of gene expression across individual cells, it may mask the transcriptional trends of distinct subpopulations with the most abundant cell types or states (Simpson's paradox [4]).

Single-cell RNA sequencing (scRNA-seq) bridged over this hurdle, providing unprecedented opportunities for exploring gene expression profiles at a single-cell resolution. Since its first introduction in 2009 [5, 6], scRNA-seq opened a new avenue to uncover the underlying cellular heterogeneity of composite systems. However, the practical procedures were arduous, time-consuming, cost-intensive, and heavily relied on a single-sourced set of equipment. At present, with the emergence of efficient and low-cost technologies (Table 1 [7]), a typical lab bench suffices for building sequencing libraries amounting to thousands of cells [8–10], thus encouraging the use of single-cell technology as a standard procedure.

These technical advancements enabled the discovery of novel cell types [11, 12] and the study of cellular dynamic processes at a previously unattainable spatial and temporal resolution [13–16], featuring in-cell variation such as gene interaction, allelic expression, and novel RNA processing in the field of molecular cell biology [17, 18]. Moreover, scRNA-seq became a key ingredient in the rapidly evolving field of precision medicine [19, 20]. The profound amount of new information obtained with scRNA-seq holds the potential of reshaping our understanding of developmental biology, gene regulation, and cell heterogeneity in health and disease.

2 The Laboratory Workflow of scRNA-seq

At present, all scRNA-seq laboratory methods rely on six main steps (Fig. 1): (*I*) preparation of a viable single-cell suspension, (*II*) assessment of cell viability, (*III*) lysed cell removal, (*IV*) individual transcriptome barcoding, (*V*) cDNA generation, and (*VI*) sequencing library generation [21]. As for instrument implementation, one of the most popular sequencing platforms is the Illumina® series due to its cost-effectiveness and high-quality outputs. A relatively new introduction in the field is the BGI sequencing portfolio, which allows equipotential sequencing results even in single-cell studies [22].

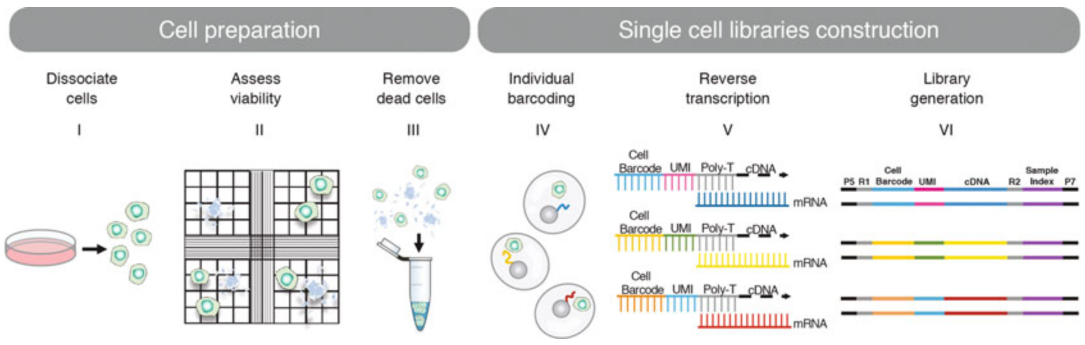


Fig. 1 Single-cell RNA sequencing workflow. The scRNA-seq procedure consists of six key steps. (*I*) Samples are dissociated into a single-cell suspension. (*II*) As lysed cells might bias the data and cause high noise interference, it is essential to maximize the quality of the input material and assess cell viability. (*III*) If the viability is lower than 90%, dead cells should be filtered either by centrifugation (i.e., density gradient) or immunodepletion (*i.e* FACS or magnetic sorting). (*IV*) Single cells are captured and isolated in different ways, depending on the technique of choice. Microfluidics-based scRNA-seq technologies encapsulate single cells within water-in-oil droplets together with unique primers attached to microparticle surface and lysis buffer. Then, each lysed cell's mRNA content is captured by the poly-A tail domain of a single primer and labeled with UMI and cell-specific barcodes. Several errors can occur during this step, like multiple cells or microparticles captured in a single droplet (i.e., multiplets), and sub-Poisson loading trade-offs, such as empty barcoded drops. (*V*) Captured mRNA transcripts from droplets are then collected, reverse-transcribed, and (*VI*) amplified in pools to be used for standard sequencing platforms. During library construction, cDNA molecules are tagged with sample-specific indexes allowing multiplexing of different captures in the same sequencing run. Further computational demultiplexing will use such barcode information to sort samples, cells, and transcripts

Barcoding the transcriptome of individual cells is a key step in all available single-cell protocols, and exemplifies the main difference from bulk RNA-seq. Two barcoding strategies are suggested, either (1) the addition of a cell-specific barcode to each transcriptome following cell isolation, or alternatively (2) the addition of a unique index combination to each cell transcriptome without physical partitioning (e.g., split-Seq [23]). Both strategies can be further classified into subcategories with different advantages and drawbacks (Table 1, Supplementary). However, all scRNA-seq strategies rely on high-quality input material, requiring the optimization of any dissociation and thawing protocol to maximize cell viability [24–26].

Among the more recent advancements in the field, microfluidic-based scRNA-seq technologies have gained popularity due to their cost-effectiveness, high efficiency, and moderate data size requirements for preserving data integrity and coherence [27, 28]. Generally, microfluidic technologies, such as Chromium [10], inDrop [9], and Drop-seq [8], rely on passive coflow of cells, microparticles (i.e., beads) and a lysis buffer that produces water-in-oil droplets, thus encapsulating precisely one cell and one bead. The transcriptional content of each cell is captured and amplified by unique primers attached to the surface of a single microparticle.

Such primers share the same underlying three-tuple structure, including (1) a cellular barcode, a short sequence common to all primers on a single microparticle, with the purpose of identifying all transcripts belonging to the same cell; (*2*) a unique molecular identifier (UMI), a molecular transcript-specific tag which secures read's integrity by identifying PCR duplicates [29]; (iii) a poly-T tail, for the capture and amplification of the 3′END of each transcript.

Ideally, each droplet should encapsulate a single cell and a single bead. However, as in practice the encapsulation step follows a Poisson distribution, the capture rate of one bead and one cell within a single droplet follows a double Poisson distribution. Ergo, many droplet-based approaches yielded large numbers of empty droplets and inefficient data assemblage.

The limitation of Poisson statistics has been tackled by inDrop and Chromium technologies. Through close-packed ordering of deformable particles, both methods instrument a sub-Poisson distribution [30], thus achieving controllable encapsulated particle quantities, with a single bead occupancy of about 80%. Hence, the main differences among the three platforms, inDrop, Chromium, and Dop-seq, is their respective capture efficiency, largely dependent on beads types at use [31]. While Drop-seq, inDrops, and Chromium capture about 5–12%, 75%, and 65% of the input cells, they also require $>2 \times 10^5$, 2×10^3–10^4, and $>10^3$ input cells, respectively.

Hence, choosing the appropriate technique is crucial, and pends on a particular field of study and research requirements. When investigating highly heterogeneous samples, like tumors and tissues, high-throughput methods are advisable. Nevertheless, high-sensitivity strategies are best suited either when analyzing low expressed genes or classifying rare cell populations [32].

However, if on one side scRNA-seq allows to dissect cellular heterogeneity at high-resolution, it also carries two key drawbacks. The first one is a low gene retrieval yield, with usually a 1–5% of transcripts per cell representing highly expressed genes (about 5000 genes per cell), thus leading to significant observational uncertainty. This drop-out effect introduces a high cell-to-cell variability and low signal-to-noise ratio (SNR) [33]. A further drawback is evidently the cost burden of scRNA-seq commercial technologies, while noncommercial platforms (inDrop, Drop-seq) require considerable operator expertise. Consequently, the implementation of scRNA-seq techniques is still not broadly accessible for many laboratories in the field [34, 35]. However, these obstacles shall not hold the promise of scRNA-seq to expand beyond the genomic research frontier, as overcoming the current challenges will widen the future outlooks for medicine and biological studies.

3 The Computational Workflow of scRNA-seq

Unlike previous genome-wide transcriptomic assays, scRNA-seq necessitates innovative analysis tools to address the emerging single-cell associated challenges, including large-scale data and high levels of noise interference due to dropout events [33, 35]. Indeed, more than 600 standalone tools are available to analyze and explore single-cell transcriptomic data [36], but with the lack of universal standardization [37, 38], a newcomer to blend in.

The challenge in achieving standardized pipelines stems from several reasons, including the relative immaturity of the field. Depending on the platform of choice, individual procedural steps may be processed differently, resulting in inconsistent downstream analysis outputs for the same entry dataset [6]. In addition, the choice of a specific analytic tool is largely swayed by a programming language preference such as R or Python, and thus restricting their usage to a narrower audience specialized in a specific programming language. A further, and probably the most significant hurdle, is the need to find a common analytic strategy that could fit various biological data types (cell lines, cancer cells, stem cells, etc.). However, due to their high diversity and distinct biological inquiries at hand, ad hoc computational strategies might be needed.

In this review we aim to address all the above-mentioned challenges, outlining a standardized workflow that will guide the reader through the key steps of scRNA-seq data analysis, regardless of specific tools and different biological data types. Herein, we propose four ready-to-use computational pipelines, which include raw counts normalization, feature selection, dimensionality reduction, and clustering (Fig. 2). Completing these steps enables the users to analyze their respective data without any loss of information. The proposed pipelines cover both R and Python programming languages, and employ Seurat (R) [39], Scanpy (Python) [40], Monocle (R) [41, 42], and gf-icf (R) [43] platforms, which are all easily accessible for a newcomer.

A case study employing the four proposed pipelines is demonstrated using a subset of Tabula Muris [44] public dataset retrieved by the Chromium technology, outlining the different steps with plots and command lines, all available on github [45]. As the proposed pipelines might be permissive or too restrictive for a given assay, we offer guidelines for tailoring the analytic settings to meet user's data requisites.

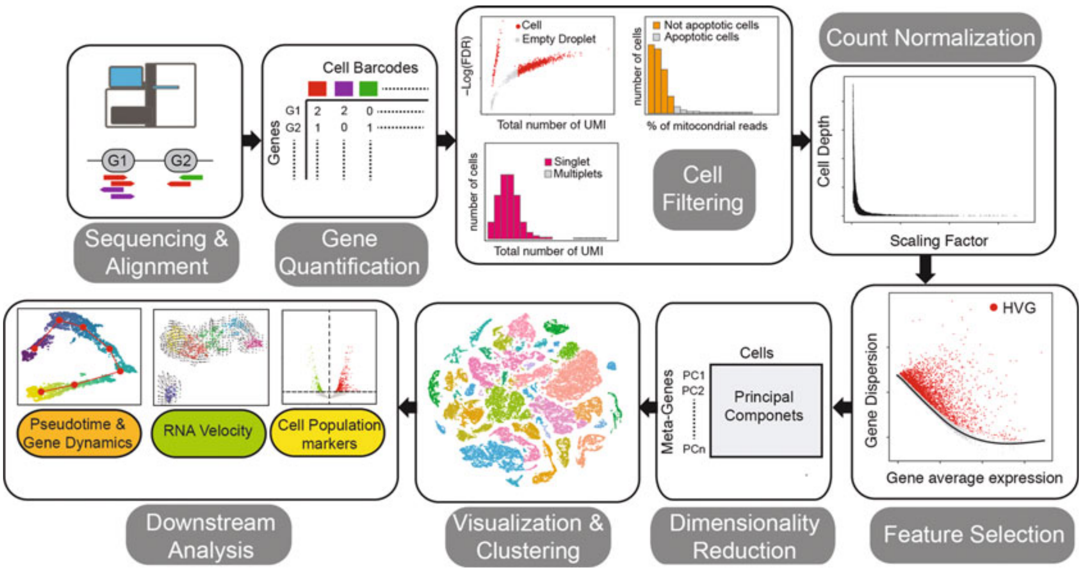


Fig. 2 Computational analysis of single-cell RNA sequencing. ScRNA-seq analysis embraces six underlying steps, including raw-data preprocessing, filtering via QC covariants, normalization, feature selection, linear dimensional reduction, visualization, and clustering: (*I*) Raw reads are processed and quantified to generate gene/barcode matrices. (*II*) Cells in the count matrix are then filtered to avoid misinterpretation of ambient gene expression, apoptotic cells, and multiplets. (*III*) Count reads normalization is required, as the analysis is disrupted by low input and weak SNR, following which data is primed for downstream analysis. (*IV*) A lesser number of highly variable features are selected for the purpose of realizing a faster and accurate procedure. (*V*) Based on the designated genes, a PCA is performed to lower data dimensionality. (*VI*) Clustering and nonlinear dimensionality reduction steps utilize a subset of significant principal components to overcome data noisiness. Subsequently, cells are clustered and visualized based on their PCA scores

4 Raw Reads Demultiplexing, Alignment, and Expression Quantification

Captured transcript fragments that are processed by sequencers, termed "reads," are stored into a text-based format called FASTQ [46]. FASTQ files contain both nucleotide sequence reads and their corresponding quality scores encoded as ASCII (American Standard Code for Information Interchange) characters.

While the majority of bulk approaches are suitable for the preprocessing of full-length scRNA-seq datasets, 3′-end scRNA-seq protocols require distinctive analytic tools. The preprocessing workflow of 3′-end scRNA-seq raw data includes three steps, (1) assigning captured RNA fragments to their associated sample and store them in FASTQ files (i.e., demultiplexing); (2) aligning the reads to a reference genome; (3) quantifying UMI per gene and assigning them to their associated barcode (i.e., cell). Eventually, each sample compiles into gene/barcode matrices that can be further filtered and analyzed.

4.1 Demultiplexing

Herein, we employ a commonly used subpipeline of the CellRanger platform [10], namely, **mkfastq**, exclusively designed for preprocessing raw data obtained from the 10x-Genomics® platform. Although CellRanger offers additional analytical tools for clustering and gene expression analysis, we have narrowed its use only to the preprocessing steps.

As input, CellRanger **mkfastq** uses raw sequencer's reads in the form of BCL files. Providing sample index sequences, **mkfastq** will demultiplex the raw data into sample-specific FASTQ files using the sample indexes.

4.2 Mapping and Expression Quantification

Before quantifying gene expression, the raw reads are first aligned to a reference genome, grouped by genes, and assigned to their original cellular barcode. These steps can be applied either by CellRanger-count for data retrieved via the 10×-Genomic® platform, or through the STARsolo tool [47, 48] for all other protocols.

Both tools require the raw FASTQ files obtained by the demultiplex step as input, and perform: (1) error correction of cell barcodes using a predefined whitelist; (2) mapping using STAR aligner; (3) correction and deduplication of UMI, and finally (4) quantification of gene expression per cell by counting the number of unique UMI per gene (i.e., transcripts).

Through the mapping step, read alignment assigns raw sequences to the most proper position in a reference genome. Although the alignment can employ a transcriptional reference, it is preferable to use a whole-genome reference, as it allows easier removal of "off-target" captured sequences that are not forced to be aligned on a transcriptional reference, but filtered out (*see* **Notes 1** and **2**).

Next, inconsistent cell barcodes and UMIs are filtered to avoid data misrepresentation. During this step, the presence of each barcode is verified in a predefined list of known cell barcode sequences provided by the single-cell platform. Accordingly, incompatible cell barcodes are either discarded or corrected by the most abundant barcode separated with a single editing distance. Similarly, CellRanger and STARsolo will assess the quality of UMIs and correct a single mismatch to a higher count UMI sequence if they both share a cell barcode and gene sequence.

Both CellRanger and STARsolo output two count matrices, filtered and unfiltered, so the user can choose which to include in the downstream analysis. The filtered count matrix consists of barcodes/identifiers that represent genuine cells and the expression levels for each gene. Differently from STARsolo, last CellRanger versions (above 3.1) employ a statistical method called EmptyDrop [49] to distinguish cells from empty barcoded drops. In this review, we will demonstrate how to apply EmptyDrop autonomously using the unfiltered count matrix, as it is common to both STARsolo and Cellranger outputs.

5 Quality Control and Cell Filtering: How to Identify Viable Cells

Current limitations of scRNA-seq are mainly related to low capture efficiency that can result in an increased level of technical noise. As of today, even the highly sensitive scRNA-seq protocols produce a small portion of low-quality barcodes due to lysed or apoptotic cells. Therefore, before proceeding with the downstream analysis, cellular barcodes that do not correspond to viable cells must be filtered out. These cells are usually recognized by detecting outliers in the distribution of QC covariates and filtered out by thresholding (*see* **Notes 3** and **4**). This step is common to all scRNA-seq pipelines and based on the analysis of three QC covariates distribution: (1) the number of captured genes per cell barcode; (2) the fraction of mitochondrial reads per barcode to identify dying cells; and (3) the number of unique UMIs per barcode (i.e., coverage depth of a cell).

5.1 Identify Empty Barcoded Drops

It is common to have empty drops when using droplet-based technology, as cells are highly diluted in order to yield a single-cell scaling. Empty drops might be contaminated with free RNA molecules, also called "ambient" RNA [37], that originated from cell lysis, which can be wrongly considered as cell-specific transcripts. To avoid misleading results, empty barcoded drops should not be included in downstream analysis. A recent method for identifying and filtering out empty drops is through the aforementioned emptyDrops function, provided by the DropletUtils package [49]. EmptyDrops is a function designed to test how significantly the barcode expression profile deviates from the ambient one using a Dirichlet-multinomial model. As input, it takes an unfiltered feature-barcode matrix and returns a data frame, where each barcode is associated with a p-value, obtained by permutation testing, and its relative FDR correction. Putting a threshold to this latter parameter allows the identification of ambient profiles with a significant deviation from cell-containing droplets, which are then considered as genuine cells. Here we show how to read data generated from the cellRanger count pipeline and detect empty droplets in the case of Tabula Muris dataset (Fig. 3a), where read data have been originally generated with the cellRanger count pipeline. Notably, since significance is retrieved by using permutations, a seed needs to be set.

5.2 Multiplet Identification

Multiplets occur when two or more cells are captured in a single drop and thus assigned to the same cell barcode [50]. This error may be misinterpreted as higher gene counts in an individual cell. Thereby, doublets can be simply filtered by identifying outliers in the count depth distribution. In the case of datasets generated by the aggregation of different samples and with different depth of

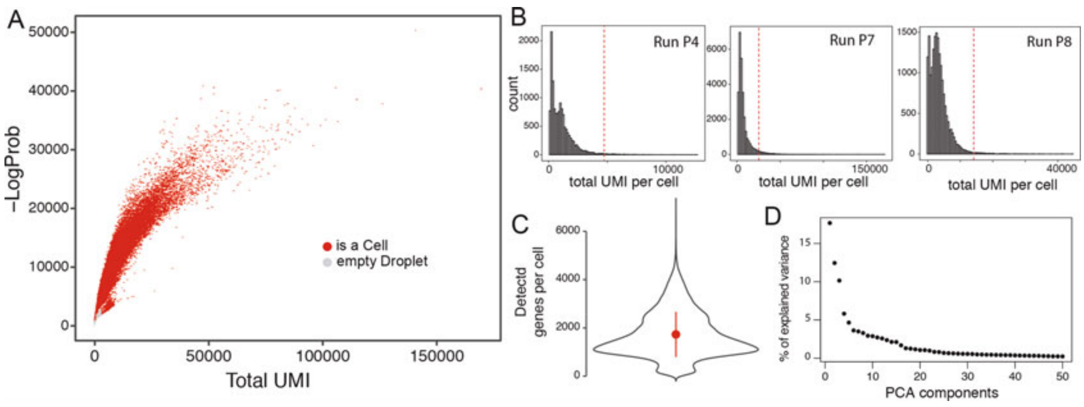


Fig. 3 Cell QC on Tabula Muris dataset. (**a**) Detection of empty droplet by using emptyDrop function from DropUtils R package on Tabula Muris dataset. (**b**) Identification of cell multiplets in each independent run of Tabula Muris Dataset. (**c**) Distribution number of detected genes across the cells in the Tabula muris dataset. (**d**) PCA components as a function of their percentage of explained variance on Tabula Muris dataset (elbow plot)

coverage among them, it is essential to perform multiplets filtered separately (Fig. 3b, *see* **Note 5**). Although a thresholding approach is usually sufficient to identify cell multiplets, new specific tools have been developed recently, offering more elegant and potentially better solutions [50–52].

5.3 Cells Lysis

Cell barcodes associated with transcripts originated by lytic cells are usually characterized by low count depth with few detected genes (Fig. 2c) and a high fraction of mitochondrial reads. In this case, unlike cytoplasmic RNA, most mitochondrial RNA is conserved thanks to undamaged mitochondrial membranes. Hence, it is acceptable to filter out barcodes with more than 10% of mitochondrial-associated reads. However, when setting a threshold, the biological property of the dataset should always be considered, therefore the threshold for an acceptable percentage of mitochondrial reads may vary according to the biological model of study. For cancer cells or specific cell types with increased respiratory or metabolic processes, the high levels of mitochondrial RNAs are inherent to the model itself [53] (*see* **Note 6**).

6 Start Working with the Scanpy, Seurat, Monocle, and gf-icf Pipelines

With the outburst of single-cell sequencing technologies, numerous statistical methods have been developed to address distinct steps of scRNA-seq analysis. Different toolkits like Seurat, Monocle 3, Scanpy, and gf-icf assembled these standalone methods to offer a single workflow. One of the most popular code-based platforms is Seurat, which offers a wide range of tutorials and analytical tools

[54]. An additional well-adapted platform is Monocle, which largely facilitated the trajectory inference field since its first introduction [41]. The latest version, Monocle 3, provides both pseudotemporal ordering and the basic scRNA-seq clustering pipeline for user convenience [42]. Scanpy, a relatively new addition in the field, allows for analyzing large size datasets up to one million cells and more, as it has improved the computational scaling. Here we also tested a recently introduced method named gf-icf, which is based on a data transformation model called term frequency-inverse document frequency (TF-IDF) that has been extensively used in the field of text mining, where sparse and zero-inflated data are common [55]. For downstream analysis, each pipeline employs either R or Python programming language. In order to interpret outputs and understand the basics of the analytical tools, each step will be examined and compared in all four pipelines.

7 Gene Filtering: How to Remove "Noisy" Genes

A scRNA-seq dataset often includes over 25,000 genes measured across thousands of cells, many of which might be uninformative as they mostly contain zero counts, and should be filtered out before starting the downstream analysis. Gene filtering can help to speed up data processing by dipping its dimensions and reducing the excess of zeros counts, consequently improving the data normalization step and all downstream analysis. Usually, a fixed threshold is defined, whereby genes detected in a small number of cells are removed (*see* **Notes** 7 and **8**).

8 Data Normalization: How to Make Gene Expression Comparable Across Individual Cells

Data normalization addresses the unwanted biases arisen by count depth variability while preserving true biological differences. The quantity of mRNA captured from each cell may diversify due to either biological variability or technical effects inherited throughout the scRNA-seq procedure, including single-cell preparation, library construction, and sequence steps [56–58]. With normalization, the expression of each gene is rescaled, considering the abundance of mRNA molecules that have been captured for each cell, in order to make gene expression comparable across individual cells. The way in which this scaling factor is estimated for each cell mainly differs across the plethora of currently available normalization methods.

As discussed above, scRNA-seq data is usually sparse due to both biological and technical reasons (dropouts). Hence, normalization methods adopted from bulk RNA-seq, such as TMM [59] and DESeq [60], might be biased by zero inflation. To address this issue, single-cell normalization procedures have evolved in recent years [61, 62]. However, at present, the most commonly used method for scRNA-seq data normalization is count per million (CPM), a linear global scaling approach that has been inherited from bulk RNA-seq.

An additional source of variation not related to the biological process under study can result from handling samples in different batches. The batch effect may arise when an experiment with identical cells is repeated independently, for example, by different operators or sampling different experimental time-lines. Standard normalization procedures do not correct for batch effect, compromising the analysis of the real biological effects. Several methods have been recently developed to account for the batch effects in scRNA-seq data [63], although ComBat [64], a method originally developed for microarray data, performs well also for single-cell experiments of low-to-medium complexity [65].

All four pipelines proposed here account for the normalization step through the CPM method. Seurat, Monocle, and Scanpy use log transformation of the CPM to reduce cell depth variability (*see* **Notes 9–11**) and few advanced options to rescale the data for some sources of variation, including the effect cell cycle [39]. With the gf-icf pipeline, genes are rescaled by their inverse cell frequency and cells are rescaled to have Euclidean norm equal to one (L2 normalization), in order to account for cell depth variability.

9 Feature Selection: How to Discard "Uninformative" Genes

A large-scale scRNA-seq dataset can easily include over 25,000 genes measured across more than 10,000 cells, with many of these genes being uninformative because mostly containing zero counts.

Feature selection aims to detect genes with relevant biological information, while excluding the uninformative ones. ScRNA-seq data dimensions can remain quite high, with a large number of genes (>10,000) still retained even after gene filtering. Feature selection can largely speed up the processing as it reduces data dimensionality by filtering "uninformative" genes. This is usually accomplished by selecting a limited number of highly variable genes (HVG) to direct further analysis. HVG are highly informative as they have a significant impact on the data configurations, and therefore allow to preserve the integrity and reproducibility of the data. Usually, 1000–5000 HVG are selected depending on the size of the assay (*see* **Note 12**). Each pipeline implements a unique method for the detection and selection of HGVs. Using Scanpy, genes are binned by their mean expression, and genes with the highest variance-to-mean ratio are selected as HVGs in each bin.

Seurat, on the other hand, first modeled the mean-variance relationship using a local polynomial regression function. Then, given the expected variance by the fitted curve and the observed mean, the feature values are standardized, and for each gene, the variance across all cells is computed [66] (*see* **Note 13**). Unlike Seurat and Scanpy, Monocle does not cover this step, while gf-icf feature selection is performed only when differentially expressed genes across clusters need to be identified. Although with few differences, also in gf-icf the feature selection is performed by modeling the mean/variance relationship as proposed by Chen et al. [67]. The feature selection in gf-icf pipeline is built in the normalization step when gene expression is rescaled by their inverse cell frequency [43], and the total number of filtered genes is considered for the dimensionality reduction step.

10 Dimensionality Reduction: How to Summarize and Visualize scRNA-seq Data

10.1 Linear Dimensional Reduction: For the Summarization of scRNA-seq Data

Dimensionality reduction aims to condense the complexity of the data into a lower-dimensional space by optimally preserving its key properties. Dimensionality reduction methods are essential for clustering, visualization, and summarization of scRNA-seq data. Linear dimensionality reduction methods are commonly used as a preprocessing step for nonlinear dimensionality reduction methods. The most popular linear dimensionality reduction algorithm is the PCA (Principal Component Analysis) [68]. Usually, 10–50 significant principal components are selected and later used as input for nonlinear dimensionality reduction methods. Principle components are highly indicative of primary sources of heterogeneity in the dataset.

PCA is used to summarise a dataset throughout the top N principal components (*see* **Note 14**). The number of PCA to use is usually determined by manually inspecting the elbow plot (Fig. 3d), in which principal components are plotted as a function of the variability they account for, and the number of PCA to use is determined by the point in which an "elbow" is observed. Additional methods can be used, including jackstraw [69] and heat maps of leading genes in each principal component. However, when choosing the significant principal components to use, it is better to err on the higher side to avoid information loss.

10.2 Nonlinear Dimensionality Reduction for the Visualization of scRNA-seq Data

Dimensionality reduction for visualization of scRNA-data uses methods that capture the nonlinearity of the scRNA-seq data, avoiding the overcrowding of the representation (*see* **Note 15**). The two most commonly used methods are the t-Distributed Stochastic Neighbor Embedding (t-SNE) [70] and the Uniform Manifold Approximation and Projection (UMAP) [71]. t-SNE is a stochastic method that efficiently highlights local data structure in low dimensions, representing cell populations as distinct clusters. However, t-SNE is not able to preserve the global structure, so the distance between clusters is meaningless. UMAP is a more recent nonlinear dimensionality-reduction technique, that is instead able to preserve both local and global structure of the data outperforming t-SNE also with a shorter run time for really large-scale scRNA-datasets.

Several additional methods exist for both linear and nonlinear data dimensionality reduction, but it is out of the scope of this tutorial to review all the existing methods, while we prefer to focus on best practices and methods currently accepted by the scRNA-seq community. However, a detailed review of linear and nonlinear methods for dimensionality reduction of single-cell transcriptomic data can be found in Moon et al. [72].

11 Clustering Analysis: How to Identify Cellular Subpopulations

As transcriptionally distinct populations of cells usually correspond to distinct cell types, a key goal of scRNA-seq consists in the identification of cell subpopulations based on their transcriptional similarity [73]. Thus, organizing cells into groups (i.e., clusters) can allow for de novo detection of cell types or identification of different subpopulations in a single cell state (*see* **Note 16**).

Clustering is an old unsupervised machine learning problem, which aims to determine the intrinsic grouping in a set of unlabelled objects by knowing their similarity score (i.e., distance). A plethora of distance measures has been proposed in the literature to compute similarity scores among objects of interest, including Euclidean distance, Cosine distance, and correlation-based distances.

Several unsupervised clustering methods have been applied to partition single-cell data and can be further divided into three groups: (1) *k*-means, (2) hierarchical clustering, and (3) community detection approaches. For single-cell data analysis, all methods are applied after feature selection and data dimensionality reduction on the PC-reduced space. The identified clusters of cells are then overlaid onto the visualization space.

The k-means algorithm uses an iterative approach to partition cells into a predefined number of clusters (*k*). At each iteration, cells are assigned to the closest centroid using the Euclidean distance. Alternative distances, like correlation-based or cosine distances, can also be used for single-cell data analysis [74]. The position of the centroids is recomputed at the end of each iteration, and since the starting position of centroids is randomly selected, it is common to run the *k*-means algorithm multiple times [75]. Although fast, *k*-

means requires to know the initial number of clusters (k) in which to partition cells, which is usually unknown and must be settled with additional complex analyses.

Hierarchical clustering is a partitioning method that seeks to build a hierarchy of clusters, and it is generally divided into two types, namely agglomerative or divisive. An agglomerative hierarchical clustering technique follows the "bottom-up" approach, where initially each cell represents an individual cluster, and gradually similar clusters are merged until getting a unique cluster. On the other hand, a divisive hierarchical clustering follows the "top-down" approach, where all cells start from a single cluster and are then progressively split. Hierarchical clustering produces a dendrogram where clusters are obtained by cutting the tree at a predetermined distance that can heuristically be settled using bootstrap approaches [76]. Examples of the application of hierarchical clustering in scRNA-seq data can be found in CIDR [77], SINCERA [78], and pcaReduce [79]. However, hierarchical clustering methods generally work slower than k-means, and do not perform well on a large-scale scRNA-seq dataset.

Community detection techniques are scalable clustering approaches, which are appropriate for large-scale graphs and can be used to cluster a hundred thousand or even millions of cells efficiently. By definition, a graph $G = (V,E)$ consists of a collection of nodes V (i.e., cells) and edges representing the degree of similarity between pairs of cells. This graph of cells can be built using the K-Nearest Neighbors (KNN) algorithm [80] applied on the PC-reduced space, where each cell is connected to its K most similar cells. Then, edge weight between any two cells is refined by Jaccard similarity, by using the proportion of neighbors they share.

Finding communities means gathering cells into groups, with a higher density of edges within groups than between them [81]. A measure of the community structure of a graph is modularity [82], namely, the fraction of edges that fall within the given groups minus the expected fraction if edges were randomly distributed. Modularity is based on the idea that a random graph is not expected to have a cluster structure. The most popular detection algorithm based on modularity is Louvain, which was introduced by PhenoGraph and also used by Seurat, Scanpy, and gf-icf.

When running a graph-based clustering, it is necessary to set the resolution parameter for the community detection algorithm based on modularity optimization. The resolution parameter is correlated to the scale of observing communities. In particular, the higher is the resolution parameter, the larger is the number of smaller communities. In our pipelines, we set the resolution parameter to 0.5, which usually represents a good trade-off.

12 Differential Expression: How to Annotate Cell Populations

Characterization and annotation of the groups of cells identified by a clustering algorithm can be managed by identifying marker genes (i.e. cluster gene signature) via differential expression analysis. Marker genes are identified by comparing cells of every single cluster to all other cells. Some differentially expressed testing methods have been developed specifically for handling the presence of dropout elements in scRNA-seq data, including the Bayesian approach [83] and MAST [84], but they are not computationally efficient when considering large-scale scRNA-seq datasets. Hence, faster tests are used for detecting differentially expressed genes, like Wilcoxon rank-sum test implemented in Seurat, Scanpy, and gf-icf, while Monocle uses a generalized additive model (VGAM). Additional complex tests are also provided by Seurat, Scanpy, and Monocle. Once gene signatures of each cluster have been identified, additional analysis including Gene Ontology Enrichment Analysis (GOEA) and Gene Set Enrichment Analysis (GSEA) [85] can be used to identify the biological processes active in each cell's cluster.

13 Results Evaluation and Comparison Among the Implemented Pipelines

To evaluate the performance of the four pipelines in identifying groups of cells (Fig. 4a–d), we calculated the agreement across clusters produced by the different methods, by using the average Jaccard coefficient [86] among each pair of clusters (Fig. 4e). We then used the retrieved clusters from each method to hierarchically cluster cell types (Fig. 4f), and showed that the different methods produce biologically meaningful partitions. We also observed that cells in the same cluster belong to the same lineage but with different levels of granularity, which can be tuned by changing the resolution parameter used to identify cell clusters.

14 Additional Analyses: How to Reconstruct Cell Transcriptional Dynamics

Depending on the biological question to address, one may think to investigate further single-cell data leveraging other existing tools that may provide other levels of information. Biological mechanisms are highly dynamic processes and thus cannot always be well described by using a discrete approach, such as clustering. Cells can transit across several transcriptional states governed by environmental changes and external perturbations. Thus, to model such continuance biological systems, including developmental processes, a new class of computational methods, called trajectory inference, have been developed in the last few years. These methods

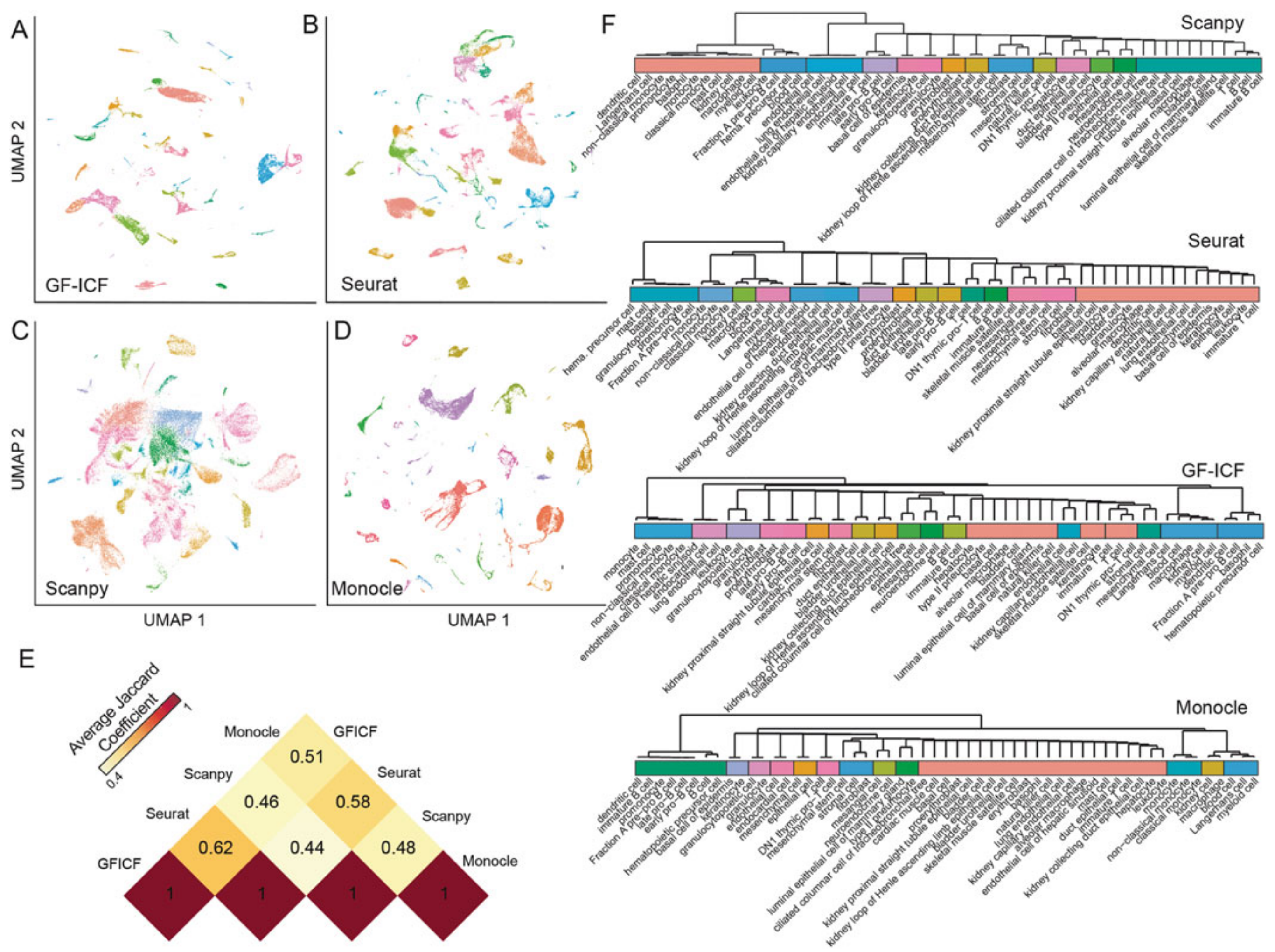


Fig. 4 Cell clustering, comparison and evaluation of the implemented pipelines. (**a**–**d**) UMAP visualization produced by each one of the four implemented pipelines where cells are colored according to the cluster in which they fall. (**e**) Agreement of identified cell clusters among the four implemented pipelines. (**f**) Cluster results of each independent pipeline is used to hierarchically cluster cell types and reconstruct cell lineage

use scRNA-seq data generated from a population of cells underlying a biological process that were collected at different time-points, and try to computationally order them along an evolutionary trajectory, which can have different topologies (i.e. linear, bifurcating or even more complex graph structure). Once cells have been ordered, gene expression patterns throughout the inferred trajectories can be used to identify key regulator genes governing cell fate decisions.

We first introduced the concept of pseudotime with Monocle as a robust methodology to describe developmental systems [41]. Since Monocle, which is at its third version now [42], the number of available methods has grown exponentially [87]. Recently, a newly proposed method to infer developmental trajectories that substantially differ from others, was modeling cellular processes using the optimal transport problem [88]. Interestingly, to date, more than 100 methods have been developed to infer cell trajectory [87].

Once trajectories have been reconstructed, RNA velocities [89] can be overlaid onto the inferred trajectory to add directionality to the reconstructed dynamical process.

15 Discussion and Future Directions

With the outburst of scRNA-seq technology, an increasing number of analytic methods have been introduced to the scientific community. Despite the wide range of analytic options, the absence of standardization leads to high entry barriers. In the present review, we propose four ready-to-use pipelines for the analysis of scRNA-seq data that could fit various biological data types. With a novice in mind, these computational pipelines provide an effective and simple workflow, including normalization of raw counts, feature selection, dimensionality reduction, and data clustering. The proposed pipelines cover both R and Python programming languages, and employ Seurat (R), Scanpy (Python), Monocle (R), and gf-icf (R) platforms.

As it is important to have the ability to interpret outputs in order to ensure data coherence, we reviewed the key steps of scRNA-seq analysis. We also highlighted guidelines and offered standardized values to filter and reduce data dimensionality. It is the user's responsibility to carefully assess the output of their analysis, and if necessary, adjust the pipeline default settings to fit source data. Furthermore, as the field evolves rapidly, this review might lag behind the up-to-date tools. Therefore, we recommend referring to this review as a basic workflow guideline while keeping in line with innovations in the field.

As single-cell sequencing is no longer limited to transcriptional experiments but allows for capturing also other data types, including DNA, ChIP, and ATAC, we presume that future pipelines must be able to cope with multiomics data integration. Single-cell multiomics will simultaneously allow gaining information on all levels of the living cells, including DNA, RNA, proteins, and epigenetic modification [90–92]. Integration of these different "omics" information into a single dimension will allow having a more comprehensive understanding of the cell fate regulations and phenotypes.

Interestingly, another new technology that necessitates high scaling computational tools is the spatial transcriptomics, which allows to identify the cell type spatial composition of tissues [93, 94]. This approach may help to increase the accuracy of the investigated system by adding another guiding dimension to the data. By positionally annotating the cells, it would be possible to precisely cluster different subpopulations in highly heterogeneous systems, such as organoids, and track the spatiotemporal dynamics between them. Therefore, the ability to conserve the spatial position will provide a better perception of tissue organization, functionality, and development.

An additional perspective is the use of high-throughput scRNA-seq technology for personalized medicine. Several efforts have been made to screen different cell types and tissues via scRNA-seq to tailor appropriate medical treatment to patients' individual characteristics [95, 96]. Developing new tools that incorporate machine learning approaches may increase the advancement in the field of precision medicine, and bring it closer to clinical usage. We believe that innovative tools occupying the aforementioned properties will stand at the forefront of science.

16 Notes

1. Before proceeding with further analyses, it is recommendable to go through sequencing and mapping statistics, which are often provided by the bioinformatic tool used for preprocessing. For instance, less than 70% of barcode-associated reads might suggest high levels of ambient RNA (due to a significant level of lysed cells or insufficient washes after tissue dissociation).
2. Reads confidently mapped to the genome should exceed 80% of the total.
3. QC-based outlier detection, that is, multiplet and lytic cell filtering, should be performed taking these covariates concomitantly.
4. The threshold for filtering outlier cells should be as permissive as possible to avoid excessive dropout effect. It could be further

adjusted once downstream analyses have been performed to better interpret data.

5. As transcripts coverage may differ between samples, it is essential to set the threshold for each one separately.
6. When setting a threshold, the biological property of the dataset should be considered, as increased respiratory or metabolic processes may also cause high mitochondrial reads.
7. The selected threshold should be as permissible as possible to avoid a dropout effect or removal of a rare cell population.
8. An acceptable guideline is to adjust the threshold to the smallest cluster size or to the number of genes expressed in more than 1–5% in the dataset.
9. Despite the normalization method of choice, data transformation (e.g., log transformation) should always be applied since most tools for downstream analyses expect normally distributed data.
10. Try to avoid correcting biological batches, unless you want to infer trajectories and such correction does NOT mask other biological information of interest.
11. When performing batch correction on technical as well as biological covariates, it should be done simultaneously.
12. The choice of HVGs may influence downstream analysis, although it has been shown that choosing between 200 and 2400 HVGs does not affect representation in lower dimensions (i.e., PCA space) [9].
13. Feature selection based on mean and variance cannot be performed on data scaled to zero mean and unit variance.
14. Principal components can also be used to inspect the effect of technical covariates on data [65], or to address the role of specific genes across the dataset [69].
15. Nonlinear dimensionality reduction methods are a powerful tool for data visualization, NOT summarization.
16. Downstream analyses require summarized data, for example, PCA or diffusion maps.

Acknowledgments

We thank Chiara Reggio and Bashir Sadet (10× Genomics) for their feedbacks. This work was supported by Fondazione Telethon Core Grant, Armenise-Harvard Foundation Career Development Award, European Research Council (grant agreement 759154, CellKarma), the Rita-Levi Montalcini program from MIUR (to D.C.) and by the STAR (Sostegno Territoriale alle Attività di Ricerca) grant of University of Naples Federico II and My First

AIRC Grant (MFAG, 23162) (to G.G.). Conflict of interest – Gennaro Gambardella declares that he has no conflict of interest. Davide Cacchiarelli is the founder, shareholder, and consultant of Next Generation Diagnostic srl. Figures were created with BioRender software,©biorender.com.

References

1. Wang ET, Sandberg R, Luo S et al (2008) Alternative isoform regulation in human tissue transcriptomes. Nature 456:470–476
2. Cacchiarelli D, Trapnell C, Ziller MJ et al (2015) Integrative analyses of human reprogramming reveal dynamic nature of induced pluripotency. Cell 162:412–424
3. Mitelman F, Johansson B, Mertens F (2007) The impact of translocations and gene fusions on cancer causation. Nat Rev Cancer 7:233–245
4. Trapnell C (2015) Defining cell types and states with single-cell genomics. Genome Res 25:1491–1498
5. Hedlund E, Deng Q (2018) Single-cell RNA sequencing: technical advancements and biological applications. Mol Asp Med 59:36–46
6. Hwang B, Lee JH, Bang D (2018) Single-cell RNA sequencing technologies and bioinformatics pipelines. Exp Mol Med 50(96)
7. Supplementary Table 1: https://github.com/gambalab/scRNAseq_chapter/blob/master/tables/table1.xlsx
8. Macosko EZ, Basu A, Satija R et al (2015) Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell 161:1202–1214
9. Klein AM, Mazutis L, Akartuna I et al (2015) Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell 161:1187–1201
10. Zheng GXY, Terry JM, Belgrader P et al (2017) Massively parallel digital transcriptional profiling of single cells. Nat Commun 8:14049
11. Plasschaert LW, Žilionis R, Choo-Wing R et al (2018) A single-cell atlas of the airway epithelium reveals the CFTR-rich pulmonary ionocyte. Nature 560:377–381
12. Suo S, Zhu Q, Saadatpour A et al (2018) Revealing the critical regulators of cell identity in the mouse cell atlas. Cell Rep 25:1436–1445.e3
13. Velasco S, Kedaigle AJ, Simmons SK et al (2019) Individual brain organoids reproducibly form cell diversity of the human cerebral cortex. Nature 570:523–527
14. Fischer DS, Fiedler AK, Kernfeld EM et al (2019) Inferring population dynamics from single-cell RNA-sequencing time series data. Nat Biotechnol 37:461–468
15. Liu Z, Wang L, Welch JD et al (2017) Single-cell transcriptomics reconstructs fate conversion from fibroblast to cardiomyocyte. Nature 551:100–104
16. Cacchiarelli D, Qiu X, Srivatsan S et al (2018) Aligning single-cell developmental and reprogramming trajectories identifies molecular determinants of myogenic reprogramming outcome. Cell Syst 7:258–268.e3
17. van Dijk D, Sharma R, Nainys J et al (2018) Recovering gene interactions from single-cell data using data diffusion. Cell 174:716–729.e27
18. Hayashi T, Ozaki H, Sasagawa Y et al (2018) Single-cell full-length total RNA sequencing uncovers dynamics of recursive splicing and enhancer RNAs. Nat Commun 9:619
19. Savas P, Virassamy B, Ye C et al (2018) Single-cell profiling of breast cancer T cells reveals a tissue-resident memory subset associated with improved prognosis. Nat Med 24:986–993
20. Moghe I, Loupy A, Solez K (2018) The human cell atlas project by the numbers: relationship to the Banff classification. Am. J. Transplant 18:1830
21. Ziegenhain C, Vieth B, Parekh S et al (2017) Comparative analysis of single-cell RNA sequencing methods. Mol Cell 65:631–643.e4
22. Senabouth A, Andersen S, Shi Q et al (2020) Comparative performance of the BGI and Illumina sequencing technology for single-cell RNA-sequencing. NAR Genom Bioinform 2. https://doi.org/10.1093/nargab/lqaa034
23. Rosenberg AB, Roco CM, Muscat RA et al (2018) Single-cell profiling of the developing mouse brain and spinal cord with split-pool barcoding. Science 360:176–182
24. Tasic B, Yao Z, Graybuck LT et al (2018) Shared and distinct transcriptomic cell types across neocortical areas. Nature 563:72–78

25. Guillaumet-Adkins A, Rodríguez-Esteban G, Mereu E et al (2017) Single-cell transcriptome conservation in cryopreserved cells and tissues. Genome Biol 18:45
26. Wohnhaas CT, Leparc GG, Fernandez-Albert F et al (2019) DMSO cryopreservation is the method of choice to preserve cells for droplet-based single-cell RNA sequencing. Sci Rep 9:10699
27. Baran-Gale J, Chandra T, Kirschner K (2018) Experimental design for single-cell RNA sequencing. Brief Funct Genomics 17:233–239
28. Salomon R, Kaczorowski D, Valdes-Mora F et al (2019) Droplet-based single cell RNAseq tools: a practical guide. Lab Chip 19:1706–1727
29. Islam S, Zeisel A, Joost S et al (2014) Quantitative single-cell RNA-seq with unique molecular identifiers. Nat Methods 11:163–166
30. Abate AR, Chen C-H, Agresti JJ, Weitz DA (2009) Beating Poisson encapsulation statistics using close-packed ordering. Lab on a Chip 9:2628
31. Zhang X, Li T, Liu F et al (2019) Comparative analysis of droplet-based ultra-high-throughput single-cell RNA-Seq systems. Mol Cell 73:130–142.e5
32. Brazovskaja A, Treutlein B, Camp JG (2019) High-throughput single-cell transcriptomics on organoids. Curr Opin Biotechnol 55:167–171
33. Stegle O, Teichmann SA, Marioni JC (2015) Computational and analytical challenges in single-cell transcriptomics. Nat Rev Genet 16:133–145
34. Haque A, Engel J, Teichmann SA, Lönnberg T (2017) A practical guide to single-cell RNA--sequencing for biomedical research and clinical applications. Genome Medicine 9
35. Lähnemann D, Köster J, Szczurek E et al (2020) Eleven grand challenges in single-cell data science. Genome Biol 21:31
36. scRNA-tools table page. https://www.scrna-tools.org/. Accessed 22 June 2020
37. Luecken MD, Theis FJ (2019) Current best practices in single-cell RNA-seq analysis: a tutorial. Mol Syst Biol 15
38. Neu KE, Tang Q, Wilson PC, Khan AA (2017) Single-cell genomics: approaches and utility in immunology. Trends Immunol 38:140–149
39. Butler A, Hoffman P, Smibert P et al (2018) Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol 36:411–420
40. Wolf FA, Angerer P, Theis FJ (2018) SCANPY: large-scale single-cell gene expression data analysis. Genome Biol 19:15
41. Trapnell C, Cacchiarelli D, Grimsby J et al (2014) The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nat Biotechnol 32:381–386
42. Cao J, Spielmann M, Qiu X et al (2019) The single-cell transcriptional landscape of mammalian organogenesis. Nature 566:496–502
43. Gambardella G, di Bernardo D (2019) A tool for visualization and analysis of single-cell RNA-Seq data based on text mining, Front Genet:10
44. Tabula Muris Consortium, Overall Coordination, Logistical Coordination, et al (2018) Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature 562:367–372
45. scRNAseq_chapter. Github. https://github.com/gambalab/scRNAseq_chapter
46. Cock PJA, Fields CJ, Goto N et al (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Res 38:1767–1771
47. Dobin A, Davis CA, Schlesinger F et al (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29:15–21
48. Du Y, Huang Q, Arisdakessian C, Garmire LX (2020) Evaluation of STAR and Kallisto on single cell RNA-Seq data alignment. G3 10:1775–1783
49. Lun ATL, participants in the 1st Human Cell Atlas Jamboree, Riesenfeld S, et al (2019) EmptyDrops: distinguishing cells from empty droplets in droplet-based single-cell RNA sequencing data. Genome Biol 20
50. Wolock SL, Lopez R, Klein AM (2019) Scrublet: computational identification of cell doublets in single-cell transcriptomic data. Cell Syst 8:281–291.e9
51. DePasquale EAK, Schnell DJ, Van Camp P-J et al (2019) DoubletDecon: deconvoluting doublets from single-cell RNA-sequencing data. Cell Rep 29:1718–1727.e8
52. McGinnis CS, Murrow LM, Gartner ZJ (2019) DoubletFinder: doublet detection in single-cell RNA sequencing data using artificial nearest neighbors. Cell Syst 8(4):329–337.e4
53. Rogalinska M (2016) The role of mitochondria in cancer induction, progression and changes in metabolism. Mini Rev Med Chem 16:524–530
54. Düchting H, Seurat G (2000) Seurat. Taschen
55. Robertson SE, Jones KS (1976) Relevance weighting of search terms. J Am Soc Inf Sci 27:129–146
56. Marinov GK, Williams BA, McCue K et al (2014) From single-cell to cell-pool transcriptomes: Stochasticity in gene expression and RNA splicing. Genome Res 24:496–510

57. Grün D, Kester L, van Oudenaarden A (2014) Validation of noise models for single-cell transcriptomics. Nat Methods 11:637–640
58. Wu AR, Neff NF, Kalisky T et al (2014) Quantitative assessment of single-cell RNA-sequencing methods. Nat Methods 11:41–46
59. Robinson MD, Oshlack A (2010) A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biol 11:R25
60. Anders S, Huber W (2010) Differential expression analysis for sequence count data. Genome Biol 11(10):R106
61. Lun ATL, Bach K, Marioni JC (2016) Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. Genome Biol 17:75
62. Vallejos CA, Risso D, Scialdone A et al (2017) Normalizing single-cell RNA sequencing data: challenges and opportunities. Nat Methods 14:565–571
63. Tran HTN, Ang KS, Chevrier M et al (2020) A benchmark of batch-effect correction methods for single-cell RNA sequencing data. Genome Biol 21:12
64. Johnson WE, Li C, Rabinovic A (2007) Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics 8:118–127
65. Büttner M, Miao Z, Wolf FA et al (2019) A test metric for assessing single-cell RNA-seq batch correction. Nat Methods 16:43–49
66. Stuart T, Butler A, Hoffman P et al (2019) Comprehensive integration of single-cell data. Cell 177:1888–1902.e21
67. Chen H-IH, Jin Y, Huang Y, Chen Y (2016) Detection of high variability in gene expression from single-cell RNA-seq profiling. BMC Genomics 17(Suppl 7):508
68. Pearson K (1901) LIII. On lines and planes of closest fit to systems of points in space. London Edinburgh Dublin Philos Mag J Sci 2:559–572
69. Chung NC, Storey JD (2015) Statistical significance of variables driving systematic variation in high-dimensional data. Bioinformatics 31:545–554
70. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
71. McInnes L, Healy J, Melville J (2018) UMAP: Uniform Manifold Approximation and Projection for dimension reduction. arXiv [stat.ML]
72. Moon KR, Stanley JS, Burkhardt D et al (2018) Manifold learning-based methods for analyzing single-cell RNA-sequencing data. Curr Opin Syst Biol 7:36–46
73. Andrews TS, Hemberg M (2018) Identifying cell populations with scRNASeq. Mol Aspects Med 59:114–122
74. Kim T, Chen IR, Lin Y et al (2019) Impact of similarity metrics on single-cell RNA-seq data clustering. Brief Bioinform 20:2316–2326
75. Kiselev VY, Kirschner K, Schaub MT et al (2017) SC3 – consensus clustering of single-cell RNA-Seq data. Nat Methods 14 (5):483–486
76. Langfelder P, Zhang B, Horvath S (2008) Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. Bioinformatics 24:719–720
77. Lin P, Troup M, Ho JWK (2017) CIDR: Ultrafast and accurate clustering through imputation for single-cell RNA-seq data. Genome Biol 18:59
78. Guo M, Wang H, Potter SS et al (2015) SINCERA: a pipeline for single-cell RNA-Seq profiling analysis. PLoS Comput Biol 11: e1004575
79. Žurauskienė J, Yau C (2016) pcaReduce: hierarchical clustering of single cell transcriptional profiles. BMC Bioinformatics 17:140
80. Levine JH, Simonds EF, Bendall SC et al (2015) Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. Cell 162:184–197
81. Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E Stat Nonlin Soft Matter Phys 70:066111
82. Newman MEJ (2006) Modularity and community structure in networks. Proc Natl Acad Sci U S A 103:8577–8582
83. Kharchenko PV, Silberstein L, Scadden DT (2014) Bayesian approach to single-cell differential expression analysis. Nat Methods 11:740–742
84. Finak G, McDavid A, Yajima M et al (2015) MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. Genome Biol 16:278
85. Subramanian A, Tamayo P, Mootha VK et al (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci U S A 102:15545–15550
86. Jaccard P (1901) Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull Soc Vaud sci nat 37:547–579
87. Saelens W, Cannoodt R, Todorov H, Saeys Y (2019) A comparison of single-cell trajectory inference methods. Nat Biotechnol 37:547–554

88. Schiebinger G, Shu J, Tabaka M et al (2019) Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. Cell 176:1517
89. La Manno G, Soldatov R, Zeisel A et al (2018) RNA velocity of single cells. Nature 560:494–498
90. Argelaguet R, Clark SJ, Mohammed H et al (2019) Multi-omics profiling of mouse gastrulation at single-cell resolution. Nature 576:487–491
91. Angermueller C, Clark SJ, Lee HJ et al (2016) Parallel single-cell sequencing links transcriptional and epigenetic heterogeneity. Nat Methods 13:229–232
92. Han KY, Kim K-T, Joung J-G et al (2018) SIDR: simultaneous isolation and parallel sequencing of genomic DNA and total RNA from single cells. Genome Res 28:75–87
93. Moncada R, Barkley D, Wagner F et al (2020) Integrating microarray-based spatial transcriptomics and single-cell RNA-seq reveals tissue architecture in pancreatic ductal adenocarcinomas. Nat Biotechnol 38:333–342
94. Ståhl PL, Salmén F, Vickovic S et al (2016) Visualization and analysis of gene expression in tissue sections by spatial transcriptomics. Science 353:78–82
95. Valdes-Mora F, Handler K, Law AMK et al (2018) Single-cell transcriptomics in cancer immunobiology: the future of precision oncology. Front Immunol 9:2582
96. Shalek AK, Benson M (2017) Single-cell analyses to tailor treatments. Sci Transl Med 9. https://doi.org/10.1126/scitranslmed.aan4730

Chapter 20

RNA-Seq Data Analysis in Galaxy

Bérénice Batut, Marius van den Beek, Maria A. Doyle, and Nicola Soranzo

Abstract

A complete RNA-Seq analysis involves the use of several different tools, with substantial software and computational requirements. The Galaxy platform simplifies the execution of such bioinformatics analyses by embedding the needed tools in its web interface, while also providing reproducibility. Here, we describe how to perform a reference-based RNA-Seq analysis using Galaxy, from data upload to visualization and functional enrichment analysis of differentially expressed genes.

Key words Galaxy, Workflow, Visualizations, Quality control, Sequence mapping, Differential gene expression, Functional enrichment

1 Introduction

In recent years, RNA sequencing (in short RNA-Seq) has become a very widely used technology to analyze the continuously changing cellular transcriptome, that is, the set of all RNA molecules in one cell or a population of cells. One of the most common aims of RNA-Seq is the profiling of gene expression by identifying genes or molecular pathways that are differentially expressed (DE) between two or more biological conditions.

The computational workflow for the detection of DE genes and pathways from RNA-Seq data requires the use of several command-line tools and substantial computational resources that most users may not have access to.

Galaxy [1] is a powerful and easy to use web-based platform for scientific data analysis. Steps in an analysis are executed by running Galaxy tools, which describe how to translate parameters for command-line software into a user-friendly web interface.

The graphical web interface and a large amount of high-quality, community-developed and maintained tools and training materials enable rapid interactive analyses for novices and expert users alike.

For each step in an analysis, Galaxy captures several metadata (e.g., tool identifier and version, inputs, and parameters) enabling reproducibility. Galaxy also allows users to easily share their workflows and data.

Galaxy's backend architecture can interface with various cloud and high-performance computing (HPC) environments, thereby providing the necessary computing resources to run computationally demanding analyses, while the end users only need access to a web browser.

Galaxy is free, open source software and can be installed locally or used on more than 120 publicly available servers. Galaxy is supported by a large community of users and developers. An important community-maintained resource is the Galaxy Training Material (available at https://training.galaxyproject.org) [2], which hosts a wide range of step-by-step hands-on tutorials for common bioinformatic analysis tasks. In particular, this chapter is based on the "Reference-based RNA-Seq data analysis" tutorial (https://training.galaxyproject.org/topics/transcriptomics/tutorials/ref-based/tutorial.html), and we defer the reader to additional explanations there.

In this chapter we will use a selection of Galaxy tools to show step-by-step how to find differentially expressed genes, from data upload to functional enrichment analysis, using real experimental data.

2 Materials

2.1 RNA-Seq Dataset

In the study of [3], the authors identified genes and pathways regulated by the *pasilla* (*ps*) gene (the *Drosophila melanogaster* homologue of the mammalian splicing regulators Nova-1 and Nova-2 proteins) using RNA-Seq data. They depleted the *ps* gene in *D. melanogaster* by RNA interference (RNAi). Total RNA was then isolated and used to prepare both single-end and paired-end RNA-Seq libraries for treated (*ps*-depleted) and untreated samples. These libraries were sequenced to obtain RNA-Seq reads for each sample. The RNA-Seq data for the treated and untreated samples can be compared to identify the effects of the *ps* gene depletion on gene expression.

In this chapter, we illustrate the analysis of the gene expression data step by step using seven of the original datasets:

- Four untreated samples: GSM461176, GSM461177, GSM461178, GSM461182.
- Three treated samples (*ps* gene depleted by RNAi): GSM461179, GSM461180, GSM461181.

In the first part of this chapter, we will use the files for two out of the seven samples to demonstrate how to calculate read counts (a measure of the gene expression) from FASTQ (https://en.wikipedia.org/wiki/FASTQ_format) files.

2.2 Computational Resources

The entire analysis described in this article can be conducted efficiently on any Galaxy server which has the required tools and reference genome; a list can be found in the "Available on these Galaxies" menu on the "Reference-based RNA-Seq data analysis" tutorial webpage mentioned above. However, to be sure, the authors recommend using the Galaxy Europe server (https://usegalaxy.eu).

3 Methods

This chapter provides a detailed workflow for the detection of DE genes and gene ontologies from raw RNA-Seq data using Galaxy (Fig. 1). The tutorial starts from quality control of the reads using **FastQC** and **Cutadapt** [4]. The reads are then mapped to a reference genome using **STAR** [5] and checked using the Integrative Genomics Viewer (**IGV**) [6] and other tools. From the mapped sequences, the number of reads per annotated genes are counted using **featureCounts** [7]. For each step, quality reports are aggregated using **MultiQC** [8]. **DESeq2** [9] is then used on the read counts to normalize them and extract the differentially expressed genes. **Heatmap2** and **Volcano Plot** are used to visualize DE genes and finally, functional enrichment analysis of the DE genes is performed using **goseq** [10] to extract interesting Gene Ontologies.

3.1 Upload FASTQ to Galaxy

RNA-Seq analysis usually starts with raw data from the sequencing machine in FASTQ format. Therefore, we first need to upload the FASTQ files for two out of the seven samples into Galaxy.

The Galaxy user interface is split up into four main areas:

- The top panel for navigating different modes (Analysis, Workflows, Library, Shared Data, User Preferences).
- The left hand side contains a searchable menu, called the Toolbox, which is used to find and select Tools in the Analysis and Workflow mode.
- The center panel, whose content changes during the different parts of an analysis. When preparing to run a Galaxy tool or workflow, the user can see and change tool parameters, while it may also be used to show information and metadata for a dataset or its content.
- The right hand side, called the History, which in the analysis mode shows the list of datasets uploaded or created by previously executed and currently executing tools.

Please login or register for a free account at the Galaxy server you are using to run the tutorial (e.g., https://usegalaxy.eu).

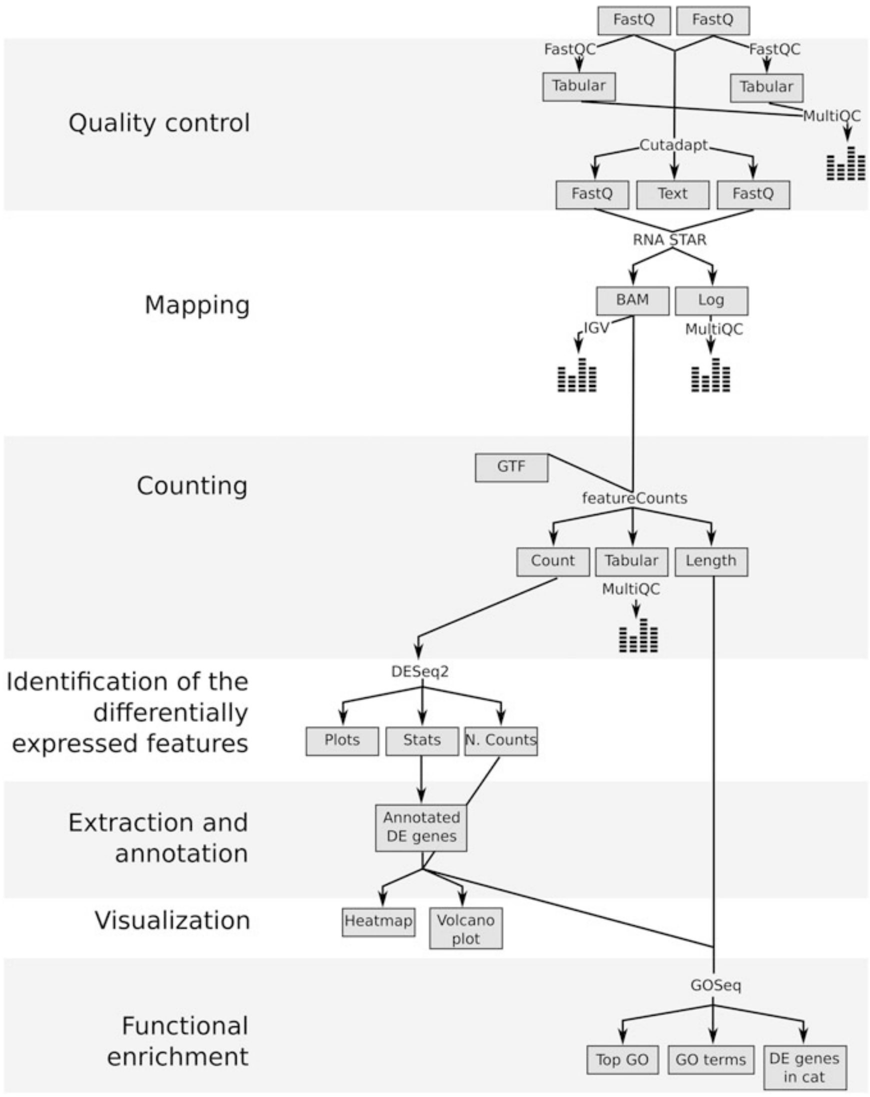


Fig. 1 Overview of the analysis pipeline used

Hands-on: Data Upload

1. Create a new history for this RNA-Seq exercise:
   (a) Click the **+** icon at the top of the history panel.
   (b) Click on **Unnamed history**.
   (c) Write a proper name, for example, Reference-based RNA-seq data analysis.
2. Import the FASTQ file pairs from the Shared data library:
   (a) Go into **Shared Data** (top panel) then **Data Libraries**.
   (b) Click on **GTN—Material** then **Transcriptomics**, **Reference-based RNA-seq data analysis**, and https://doi.org/10.5281/zenodo.1185122.

   (c) In the search box type *fastq* to see just the FASTQ files.
   (d) Select the following files:
      - https://zenodo.org/api/files/1d804082-4153-47f1-a320-4ac261ce091d/GSM461177_1.fastqsanger
      - https://zenodo.org/api/files/1d804082-4153-47f1-a320-4ac261ce091d/GSM461177_2.fastqsanger
      - https://zenodo.org/api/files/1d804082-4153-47f1-a320-4ac261ce091d/GSM461180_1.fastqsanger
      - https://zenodo.org/api/files/1d804082-4153-47f1-a320-4ac261ce091d/GSM461180_2.fastqsanger
   (e) Click on the **Export to History** button near the top and select **as Datasets** from the drop-down menu.
   (f) In the pop-up window, select the history you want to import the files to (or create a new one).
   (g) Click on **Import**.
   (h) Click the green pop-up box or **Analyze Data** in the top panel to move to the analysis page.

3. Rename each dataset according to the sample id (e.g., GSM461177_1):
   (a) Click on the **pencil icon** for the dataset to edit its attributes.
   (b) In the central panel, change the **Name** field.
   (c) Click the **Save** button.
4. Check that the datatype (i.e., format) of each dataset is fastqsanger, **not** fastq (if needed, click on the dataset name to expand the box to see). If it is not, please change the datatype to fastqsanger.
   (a) Click on the **pencil icon** for the dataset to edit its attributes.
   (b) In the central panel, click on the **Datatypes** tab on the top.
   (c) Select fastqsanger.
   (d) Click the **Change datatype** button.
5. Add to each dataset a tag corresponding to the name of the sample (#GSM461177 or #GSM461180):
   (a) Click on the dataset.
   (b) Click on the **Edit dataset tags** icon.
   (c) Add a tag starting with #. Tags starting with # will be automatically propagated to the outputs of tools using this dataset.
   (d) Check that the tag is appearing below the dataset name.

The reads are raw data from the sequencing machine without any preprocessing. They first need to be assessed for their quality.

3.2 Quality Control and Trimming

During sequencing, errors are introduced, such as incorrect nucleotides being called. These are due to the technical limitations of each sequencing platform. Sequencing errors might bias the analysis and can lead to a misinterpretation of the data. Adapters may also be present if the reads are longer than the fragments sequenced and trimming these may improve the number of reads mapped.

Sequence quality control is therefore an essential first step in every analysis. We recommend to use tools such as **FastQC** (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/) to create a report of sequence quality, **MultiQC** [8] to aggregate generated reports, and **Cutadapt** [4] to improve the quality of sequences via trimming and filtering (*see* **Note 1** for alternative tools). Note that to find a tool in Galaxy, you can search for it in the search box at the top of the tool panel on the left. To run a tool after selecting the parameters, just click the Execute button on the tool form.

Hands-on: Quality Control

1. **FastQC**:
   (a) For the "*Short read data from your current history*" input:
   - Click on the **Multiple datasets** button.
   - Select all the input datasets you have uploaded by keeping the *Ctrl* (or *COMMAND*) key pressed and clicking on the various datasets.
2. **MultiQC** with the following parameters to aggregate the FastQC reports:
   (a) In "*Results*".
   - "*Which tool was used generate logs?*": FastQC.
   - In "*FastQC output*".
     – "*Type of FastQC output?*": Raw data.
     – "*FastQC output*": the 4 RawData files (output of **FastQC**).
3. Inspect the web page output from **MultiQC** for each FASTQ dataset.

The aggregate report shows that everything seems good for three of the files, but in one file (reverse reads of GSM461180) the quality decreases quite a lot at the end of the sequences (Fig. 2).

We should trim the reads to get rid of bases that were sequenced with high uncertainty (i.e. low quality bases) at the read ends, and also remove reads of overall bad quality.

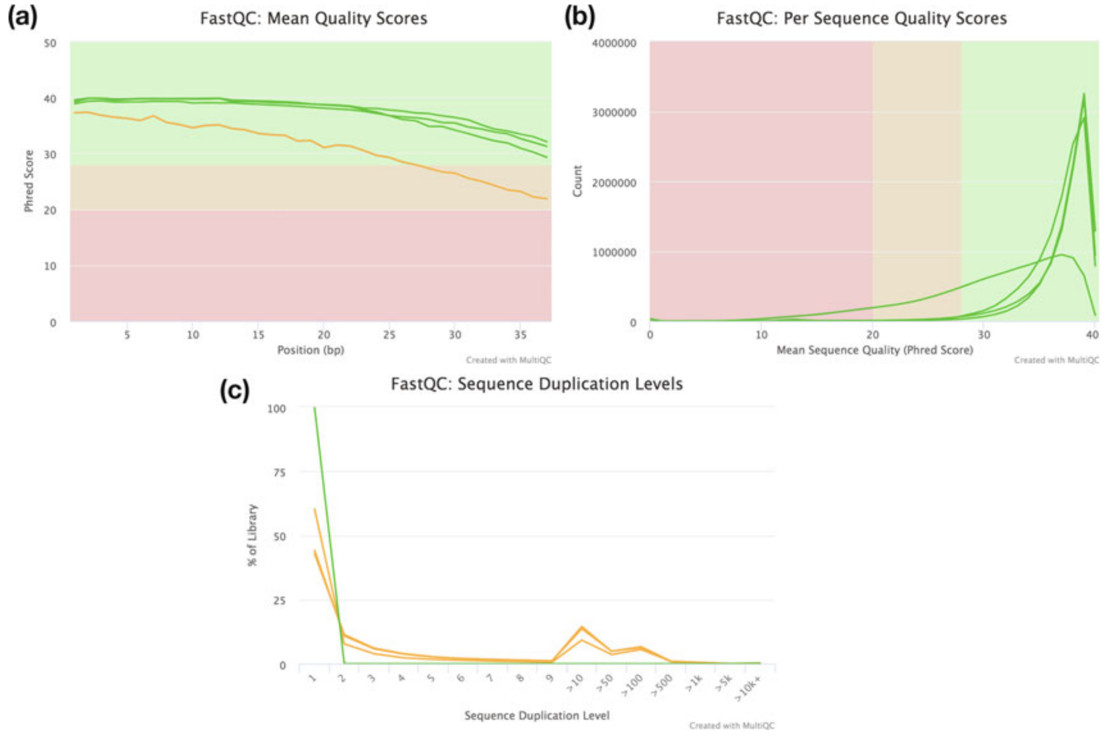


Fig. 2 The “Per base sequence quality” (**a**) is globally good with a slight decrease at the end of the sequences. For reverse reads of GSM461180, the decrease is quite large. The mean quality score over the reads (**b**) is quite high, but the distribution is slightly different for reverse reads of GSM461180. All except reverse reads of GSM461180 have a high proportion of duplicated reads (**c**, expected in RNA-Seq data)

Hands-on: Read Trimming and Filtering

1. **Cutadapt** with the following parameters to trim low quality sequences:
   (a) "*Single-end or Paired-end reads?*": Paired-end.
   - "*FASTQ/A file #1*": both fastqsanger datasets ending with "_1", selected using the multiple datasets option.
   - "*FASTQ/A file #2*": both fastqsanger datasets ending with "_2", selected using the multiple datasets option.

   (b) In the "*Filter Options*" section:
   - "*Minimum length*": 20

   (c) In the "*Read Modification Options*" section:
   - "*Quality cutoff*": 20

   (d) In the "*Output Options*" section:
   - "*Report*": Yes.
2. Inspect the generated Report datasets in your history.

For GSM461177, 5,072,810 bp has been trimmed for the forward reads (read 1) and 8,648,619 bp for the reverse (read 2) because of quality. For GSM461180, 10,224,537 bp for forward and 51,746,850 bp for the reverse. It is not a surprise: we saw that at the end of the reads the quality was dropping more for the reverse reads than for the forward reads, especially for GSM461180.

3.3 Mapping

To make sense of the reads, we need to first figure out where the sequences originated from in the genome, so we can then determine to which genes they belong. When a reference genome for the organism is available, this process is known as aligning or "mapping" reads to the reference.

In this study, the authors used *D. melanogaster* cells. We should map the quality-controlled sequences to the reference genome of *D. melanogaster* [11], i.e. the set of nucleic acid sequences assembled as a representative example of the species' genetic material.

With eukaryotic transcriptomes most reads originate from processed mRNAs lacking introns, therefore they cannot be simply mapped back to the genome as we normally do for DNA data (Fig. 3). Instead, several splice-aware mappers (e.g., TopHat [12], HISAT2 [13, 14], STAR [5]) have been developed to efficiently map transcript-derived reads against a reference genome. Here we will map our reads to the *D. melanogaster* genome using **STAR**.

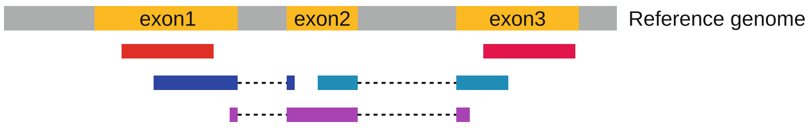


Fig. 3 The types of RNA-Seq reads (adapted from Fig. 1a from [13]), reads that mapped entirely within an exon (in red), reads spanning over two exons (in blue), read spanning over more than two exons (in purple)

Hands-on: Spliced Mapping

1. Import the Ensembl gene annotation for *D. melanogaster* (Drosophila_melanogaster.BDGP6.87.gtf) from the Shared Data library into your current Galaxy history.
   (a) Rename the dataset if necessary.
   (b) Verify that the datatype is gtf and not gff, and that the database is **dm6**. If not, click on the pencil icon and edit its attributes.
2. **RNA STAR** to map the reads from both samples on the reference genome:
   (a) "*Single-end or paired-end reads*": Paired-end (as individual datasets)
      - "*RNA-Seq FASTQ/FASTA file, forward reads*": "Read 1 Output" for both samples (outputs of **Cutadapt**), selected using the multiple datasets option.
      - "*RNA-Seq FASTQ/FASTA file, reverse reads*": "Read 2 Output" for both samples (outputs of **Cutadapt**), selected using the multiple datasets option.

   (b) "*Custom or built-in reference genome*": Use a built-in index.
      - "*Reference genome with or without an annotation*": "use genome reference without builtin gene-model".
        - "*Select reference genome*": Fly (Drosophila Melanogaster): "dm6 Full".
        - "*Gene model (gff3,gtf) file for splice junctions*": the imported Drosophila_melanogaster.BDGP6.87.gtf.
        - "*Length of the genomic sequence around annotated junctions*": 36 (This parameter should be length of reads—1).
3. **MultiQC** to aggregate the STAR logs:
   (a) In "*Results*".
      - "*Which tool was used generate logs?*": STAR.
      - In "*STAR output*".
        - "*Type of STAR output?*": Log.
        - "*STAR output*": log files (outputs of **RNA STAR**).

The **MultiQC** report reveals that around 80% of reads for both samples are mapped exactly once to the reference genome. Percentages below 70% should be investigated for potential contamination, so here we can safely proceed with the analysis. Both samples have a low (less than 10%) percentage of reads that mapped to multiple locations on the reference genome. This is in the normal range for Illumina short-read sequencing, but the range expected may be lower for long-read sequencing datasets that can span larger repeated regions in the reference sequence.

The main output of **STAR** is a BAM (https://en.wikipedia.org/wiki/Binary_Alignment_Map) file. The BAM files contain mapping information for all our reads, making it difficult to inspect and explore in text format. A powerful tool to visualize the content of BAM files is the Integrative Genomics Viewer (**IGV**).

Hands-on: Inspection of Mapping Results

1. Install IGV from https://software.broadinstitute.org/software/igv/download (if not already installed).
2. Start IGV locally.
3. Expand the mapped.bam file (output of **RNA STAR**) for GSM461177.
4. Click on the local in display with IGV local *D. melanogaster* (dm6) to load the reads into the IGV browser.
5. **IGV**: Zoom to chr4:540,000–560,000 (Chromosome 4 between 540 kb to 560 kb) (Fig. 4a).
6. **IGV**: Inspect the splice junctions using a **Sashimi plot** (Fig. 4b).
   (a) Right click on the BAM file (in IGV).
   (b) Select **Sashimi Plot** from the menu.

After the mapping, we have now the information on where the reads are located on the reference genome and how well they were mapped (*see* **Note 2** for more quality checks). The next step in RNA-Seq data analysis is quantification of the number of reads

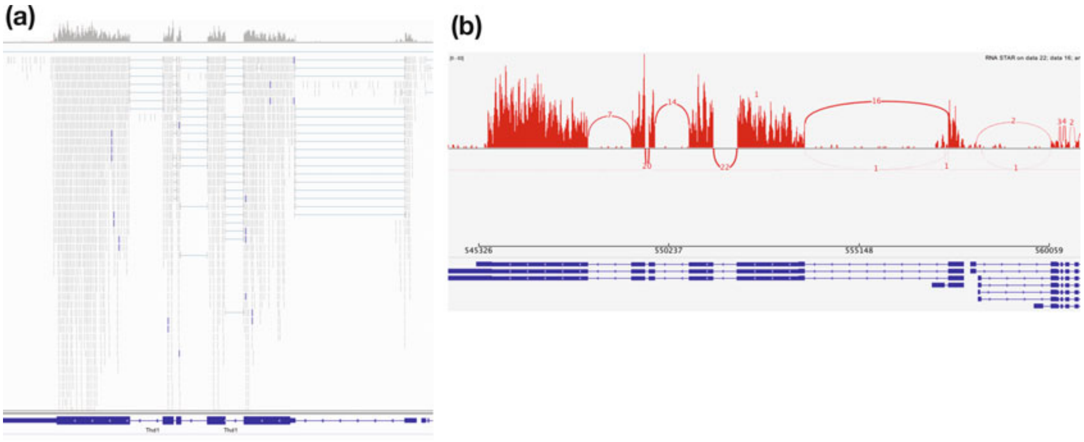


Fig. 4 Inspection of BAM file with IGV on chromosome 4. (**a**) On the top, the coverage plot shows the sum of mapped reads at each position as grey peaks. In the middle, each read is displayed where it maps. The blue lines indicate the junction events (or splice sites), that is, reads that are mapped across an intron. On the bottom, the reference genome with its genes is represented. (**b**) Sashimi plot showing the coverage in red with the arcs representing the splice junctions. The numbers refer to the number of reads spanning the junctions. On the bottom, the different groups of linked boxes represent the different transcripts from the genes at this location that are present in the GTF file

mapped to genomic features (genes, transcripts, exons, . . .). Here we will focus on the genes as we would like to identify the ones that are differentially expressed because of the *pasilla* gene knockdown.

3.4 Count the Number of Reads per Annotated Genes

To compare the expression of single genes between different conditions (e.g., with or without *ps* depletion), an essential first step is to quantify the number of reads per gene, or more specifically the number of reads mapping to the exons of each gene. A fast and efficient tool for this task is **featureCounts** [7] (*see* **Note 3** for alternative tools).

Hands-on: Counting the Number of Reads per Annotated Gene

1. **featureCounts** to count the number of reads per gene:
   (a) "*Alignment file*": mapped.bam files (outputs of **RNA STAR**).
   (b) "*Specify strand information*": Unstranded (*see* **Note 4**).
   (c) "*Gene annotation file*": in your history.
   - "*Gene annotation file*": Drosophila_melanogaster.BDGP6.87.gtf.

   (d) "*Output format*": Gene-ID "\t" read-count (MultiQC/DESeq2/edgeR/limma-voom compatible).
   (e) "*Create gene-length file*": Yes.
   (f) In "*Options for paired-end reads*":
   - "*Count fragments instead of reads*": Enabled; fragments (or templates) will be counted instead of reads.

   (g) In "*Advanced options*":
   - "*GFF feature type filter*": exon.
   - "*GFF gene identifier*": gene_id.
   - "*Allow reads to map to multiple features*".
   - "*Minimum mapping quality per read*": 10
2. **MultiQC** to aggregate the report:
   (a) In "*Results*":
   - "Which *tool was used generate logs?*": featureCounts.
   - "Output *of FeatureCounts*": Summary files (outputs of **featureCounts**).

The main output of **featureCounts** is a table with the number of reads (or fragments in the case of paired-end reads) mapped to each gene (in rows, with their ID in the first column) in the provided annotation. **FeatureCounts** can also generate a file with the length of each gene, a file we will need later for the functional enrichment analysis.

3.5 Identification of Differentially Expressed Genes

To be able to identify differential gene expression induced by *ps* depletion, all datasets (three treated and four untreated) must be analyzed following the same procedure. To save time, we have run the previous steps for you and generated seven files with the counts for each gene of *D. melanogaster* for each sample.

Hands-on: Import all Count Files

1. Create a new empty history.
2. Import the seven count files from the same Shared Data library: GSM461176_untreat_single.counts, GSM461177_untreat_paired.counts, GSM461178_untreat_paired.counts, GSM461179_treat_single.counts, GSM461180_treat_paired.counts, GSM461181_treat_paired.counts, GSM461182_untreat_single.counts.
3. Rename the datasets to the names above (i.e., remove the path prefix).

We would like now to calculate the extent of differential gene expression. **DESeq2** [9] is a tool for differential analysis of count data which uses negative binomial generalized linear models (*see* **Note 5** for alternative tools). DESeq2 takes read count files from different samples, combines them into a big table (with genes in the rows and samples in the columns) and applies normalization for sequencing depth and library composition. Gene length normalization does not need to be accounted for because we are comparing the counts between sample groups for the same gene.

DESeq2 also runs the differential gene expression analysis, whose two basic tasks are as follows:

- Estimate the biological variance using the replicates for each condition (*see* **Note 6** about replicates).
- Estimate the significance of expression differences between any two conditions.

Multiple factors can be incorporated in the analysis describing known sources of variation (e.g., treatment, tissue type, gender, batches), with two or more levels representing the conditions for each factor. After normalization we can compare the response of the expression of any gene to the presence of different levels of a factor in a statistically reliable way.

In our example, we have samples with two varying factors that can contribute to differences in gene expression: Treatment (either treated or untreated) and Sequencing type (paired-end or single-end). Here, treatment is the primary factor that we are interested in. The sequencing type is further information we know about the data that might affect the analysis. Multifactor analysis allows us to assess the effect of the treatment, while taking the sequencing type into account too.

Hands-on: Determine Differentially Expressed Features

1. **DESeq2** with the following parameters:
   (a) "*how*": Select datasets per level.
   - In "1: Factor".
     - "*Specify a factor name*": Treatment
     - In "*Factor level*":
       1. In "*1: Factor level*":
          (a) "*Specify a factor level*": treated.
          (b) "*Counts file(s)*": the three gene count files with "treat" in their name.
       2. In "*2: Factor level*":
          (a) "*Specify a factor level*": untreated
          (b) "*Counts file(s)*": the four gene count files with "untreat" in their name.
   - Click on "*Insert Factor*" (not on "Insert Factor level").
   - In "2: Factor".
     - "*Specify a factor name*": Sequencing
     - In "*Factor level*":
       1. In "*1: Factor level*":
          (a) "*Specify a factor level*": PE
          (b) "*Counts file(s)*": the four gene count files with "paired" in their name.
       2. In "*2: Factor level*":
          (a) "*Specify a factor level*": SE
          (b) "*Counts file(s)*": the three gene count files with "single" in their name.

   (b) "*Files have header?*": No.
   (c) "*Visualising the analysis results*": Yes.
   (d) "*Output normalized counts table*": Yes.

DESeq2 generated three outputs. The first output is the table with the normalized counts for each gene (rows) in the samples (columns). The second output is a graphical summary of the results, useful to evaluate the quality of the experiment:

- Plot with the first two dimensions from a principal component analysis (PCA) run on the normalized counts of the samples (Fig. 5a). It shows the samples in the 2D plane spanned by their first two principal components. Each replicate is plotted as an individual data point. This type of plot is useful for visualizing the overall effect of experimental covariates and batch effects.
- Heatmap of the sample-to-sample distance matrix (with clustering) based on the normalized counts (Fig. 5b). The heatmap

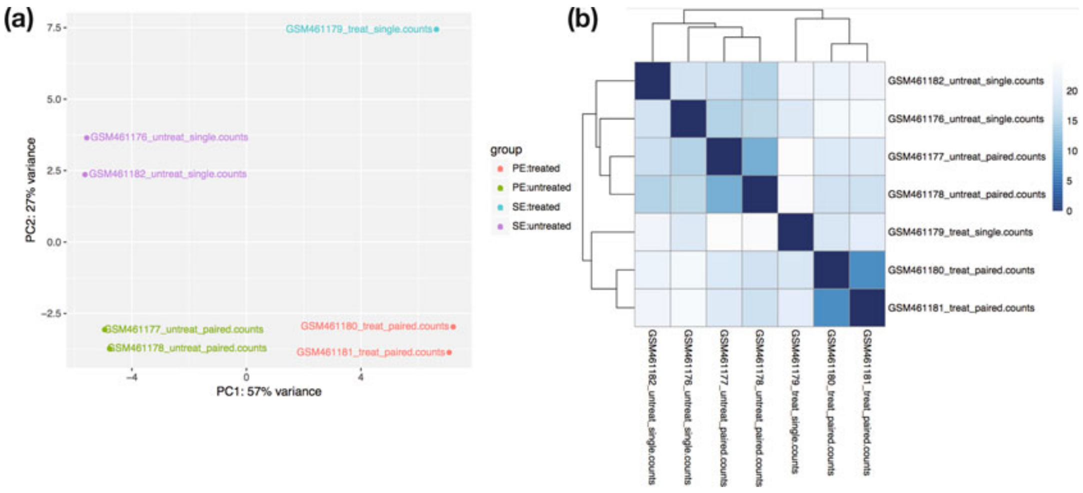


Fig. 5 Graphical summary of DESeq2 results. (**a**) Plot with the first 2 dimensions from a principal component analysis (PCA) run on the normalized counts of the samples. The first dimension is separating the treated samples from the untreated samples and the second dimension the single-end datasets from the paired-end datasets. The datasets are grouped following the levels of the two factors. No hidden effect seems to be present on the data. If there is unwanted variation present in the data (e.g., batch effects) it is always recommended to correct for this, which can be accommodated in DESeq2 by including in the design any known batch variables. (**b**) Heatmap of the sample-to-sample distance matrix (with clustering) based on the normalized counts. The samples are first grouped by the treatment (the first factor) and secondly by the sequencing type (the second factor), as in the PCA plot

gives an overview of similarities and dissimilarities between samples: the color represents the distance between the samples. Dark blue means shorter distance, that is, closer samples given the normalized counts.

- Dispersion estimates: gene-wise estimates (black), the fitted values (red), and the final maximum a posteriori estimates used in testing (blue). This dispersion plot is typical, with the final estimates shrunk from the gene-wise estimates toward the fitted estimates. Some gene-wise estimates are flagged as outliers and not shrunk toward the fitted value. The amount of shrinkage can be more or less than seen here, depending on the sample size, the number of coefficients, the row mean and the variability of the gene-wise estimates.
- Histogram of *p*-values for the genes in the comparison between the two levels of the first factor.
- MA plot. It displays the global view of the relationship between the expression change of conditions (log ratios, *M*), the average expression strength of the genes (average mean, *A*), and the ability of the algorithm to detect differential gene expression. The genes that passed the significance threshold (adjusted *p*-value <0.1) are colored in red.

The main output of DESeq2 is a summary file with the following values for each gene:

- Gene identifier.
- Mean normalized counts, averaged overall samples from both conditions.
- Fold change in log2 (logarithm base 2). The log2 fold changes are based on the primary factor level 1 vs factor level 2, hence the input order of factor levels is important. Here, DESeq2 computes fold changes of 'treated' samples against 'untreated' from the first factor 'Treatment', i.e. the values correspond to up- or downregulation of genes in treated samples (*see* **Note 7** for details about other factors and levels comparisons).
- Standard error estimate for the log2 fold change estimate.
- Wald statistic.
- *p*-Value for the statistical significance of this change,
- *p*-Value adjusted for multiple testing with the Benjamini–Hochberg procedure, which controls the false discovery rate (FDR).

For example, the gene FBgn0003360 is differentially expressed because of the treatment: it has a significant adjusted *p*-value (4.0×10^{-178}, much less than 0.05) and it is less expressed (− in the log2FC column) in treated samples compared to untreated samples, by a factor ~8 ($2^{|\text{log2FC}|}$).

Some of the tools we will use in the rest of the chapter require a header row in the DESeq2 result file so we will add column names before going further.

Hands-on: Add Column Names

1. **Create a new file** from the following (header line of the DESeq2 output) by pasting the line below into the Galaxy upload file **Paste/Fetch data** box:

   GeneID Base-mean log2(FC) StdErr Wald-Stats *P*-value P-adj

   (a) In **Type**, select "Tabular".

   (b) In **Settings**, click on "Convert spaces to tabs".

2. **Concatenate datasets** to add the header line to the annotated genes.

   (a) "*Concatenate*": the Pasted entry dataset.

   (b) "*Dataset*": the **DESeq2** result file.

We would like to extract the most differentially expressed genes due to the treatment and with an absolute fold change >2 (equivalent to an absolute $\log_2\text{FC} > 1$).

Hands-on: Extract the most Differentially Expressed Genes

1. **Filter data on any column using simple expressions** to extract genes with a significant change in gene expression (adjusted *p*-value below 0.05) between treated and untreated samples:
   (a) "*Filter*": output of **Concatenate**.
   (b) "*With following condition*": c7 < 0.05
   (c) "*Number of header lines to skip*": 1
2. Rename the output "Genes with significant adj *p*-value".
3. **Filter data on any column using simple expressions** to extract genes with an absolute fold change (FC) > 2.
   (a) "*Filter*": Genes with significant adj *p*-value.
   (b) "*With following condition*": abs(c3) > 1
   (c) "*Number of header lines to skip*": 1

We now have a table with 131 lines corresponding to the most differentially expressed genes. For each gene, we have its ID, its mean normalized counts (averaged overall samples from both conditions), its log2FC and other information.

The ID for each gene is something like FBgn0003360, which is an ID from the corresponding database, here Flybase [15]. These IDs are unique but sometimes we prefer to have the gene symbols, even if they may not reference a unique gene (e.g., duplicated after reannotation), as they may hint already to a function or they help you to search for desired candidates. We would also like to display the location of these genes on the genome. We can extract such information from the annotation file which we used for mapping and counting.

Hands-on: Annotate the Differentially Expressed Genes

1. Using **View all histories,** drag and drop the Ensembl gene annotation for *D. melanogaster* (Drosophila_melanogaster.BDGP6.87.gtf) from the previous history into this history.
2. **Annotate DESeq2/DEXSeq output tables** with:
   (a) "*Tabular output of DESeq2/edgeR/limma/DEXSeq*": output of the last **Filter**.
   (b) "*Input file type*": DESeq2/edgeR/limma.
   (c) "*Reference annotation in GFF/GTF format*": Drosophila_melanogaster.BDGP6.87.gtf.

The generated output is an extension of the previous file: (1) Gene identifiers, (2) Mean normalized counts overall samples, (3) Log2 fold change, (4) Standard error estimate for the log2 fold change estimate, (5) Wald statistic, (6) *p*-value for the Wald statistic, (7) *p*-value adjusted for multiple testing with the Benjamini–Hochberg procedure for the Wald statistic, (8) Chromosome, (9) Start, (10) End, (11) Strand, (12) Feature, (13) Gene name.

With this extra information, we can see that FBgn0025111, one of the most significantly overexpressed genes, is located on the reverse strand of chromosome X, between 10,778,953 bp and 10,786,907 bp, and is also named Ant2, that is, that it corresponds to adenine nucleotide translocase 2.

3.6 Visualization

We can visualize the differentially expressed results with volcano plots and heatmaps.

We can generate a heatmap of expression for the top differentially expressed genes in the different samples. To do this we need the normalized counts for these genes. To extract the normalized counts for the interesting genes, we join the normalized count table generated by **DESeq2** with the table of the top differentially expressed genes that we just generated. We can then use **heatmap2** to create the heatmap. In **heatmap2** we will select to scale the data by row (genes), which converts the expression values to z-scores and prevents highly expressed genes from dominating the plot. However, note that **heatmap2** performs clustering before scaling, so if you want to view the clustering after scaling, use the **Table Compute** tool to compute Z-scores before creating the heatmap.

Hands-on: Create an Expression Heatmap for the Top Differentially Expressed Genes

1. **Join two Datasets side by side on a specified field** to keep only the most differentially expressed genes in the DESeq2 normalized counts file:
   (a) "*Join*": the DESeq2 normalized counts file.
   (b) "*using column*": Column: 1.
   (c) "*with*"*:* output from **Annotate DESeq2/DEXSeq**.
   (d) "*and column*": Column: 1.
   (e) "*Keep lines of first input that do not join with second input*"*:* No.
   (f) "*Keep the header lines*"*:* Yes.
2. **Cut columns from a table** to extract the columns with the gene IDs and normalized counts:
   (a) "*Cut columns*": c1-c8
   (b) "*Delimited by*": Tab.
   (c) "*From*": the output of the previous **Join**.
3. **heatmap2** to create a heatmap:
   (a) "*Input should have column headers - these will be the columns that are plotted*"*:* the file from the previous **Cut**.
   (b) "*Plot title*": Top differentially expressed genes
   (c) "*Data transformation*": Log2(value) transform my data.

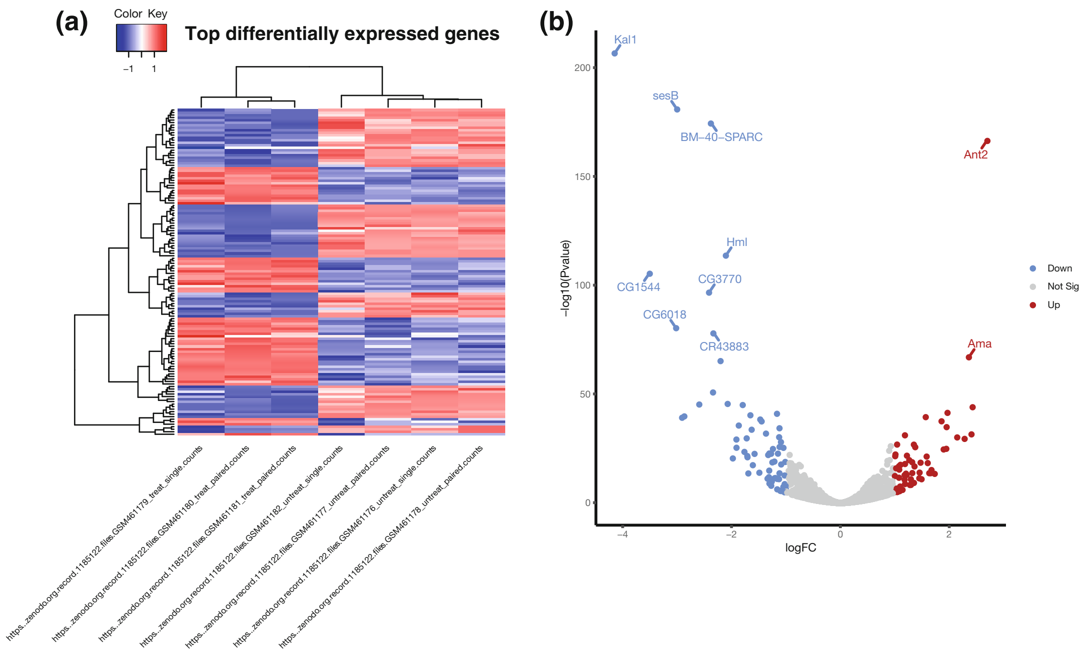


Fig. 6 Visualization of the expression results. (**a**) Heatmap of normalized expression (*z*-scores) for the top 130 differentially expressed genes in the 7 samples. Blue indicates relatively low expression in a sample, red indicates high. (**b**) Volcano plot for the comparison between treated and untreated samples, showing all genes, with $\log_2$FC on the *X*-axis and $-\log_{10}$ of the *P*-value on the *Y*-axis. The points in grey correspond to genes that are not significantly differentially expressed (using a threshold of 0.05 on the adjusted p-value and absolute $\log_2$FC of 1), in red the significantly overexpressed genes ($\log_2\text{FC} > 1$) and in blue the significantly under-expressed genes ($\log_2\text{FC} < -1$)

(d) "*Enable data clustering*": Yes.

(e) "*Clustering columns and rows*": Cluster rows and columns.

(f) "*Labeling columns and rows*": Label columns and not rows.

(g) "*Coloring groups*": Blue to white to red.

(h) "*Data scaling*": Scale my data by row.

Based on the normalized counts for the 130 top differentially expressed genes, the samples (in columns) are clustered primarily by treatment (Fig. 6a). We can see that the samples within each treatment type (treated and untreated) tend to have similar expression patterns for these genes (low expression is blue and high expression is red), which is good. We can see also clusters of genes based on their expression.

Volcano plots are commonly used to display the results of RNA-Seq or other omics experiments. A volcano plot is a type of scatterplot that shows statistical significance (P value) versus magnitude of change (fold change). It enables quick visual identification of genes with large fold changes that are also statistically significant. These may be the most biologically significant genes. In a volcano plot, the most upregulated genes are toward the right, the most downregulated genes are toward the left, and the most statistically significant genes are toward the top. We will make a volcano plot showing the names of the top ten most differentially expressed genes.

Hands-on: Creating a Volcano Plot

1. **Filter data on any column using simple expressions** to remove genes with NAs from the DESeq2 result:
   (a) "*Filter*": output from **Concatenate**.
   (b) "*With following condition*": c7!='NA'
   (c) "*Number of header lines to skip*": 1
2. **Join two Datasets side by side on a specified field** to add the gene names for the most differentially expressed genes to the DESeq2 results file:
   (a) "*Join*": output from the previous **Filter**.
   (b) "*using column*": Column: 1.
   (c) "*with*"*:* output from **Annotate DESeq2/DEXSeq**.
   (d) "*and column*": Column: 1.
   (e) "*Keep lines of first input that do not join with second input*"*:* Yes.
   (f) "*Keep the header lines*"*:* Yes.
3. **Volcano Plot** to create a volcano plot:
   (a) "*Specify an input file*"*:* output of the previous **Join**.
   (b) "*FDR (adjusted P value)*"*:* Column: 7.
   (c) "*P value (raw)*"*:* Column: 6.
   (d) "*Log Fold Change*"*:* Column: 3.
   (e) "*Labels*"*:* Column: 20.
   (f) "*Significance threshold*"*:* 0.05
   (g) "*LogFC threshold to colour*"*:* 1.0
   (h) "*Points to label*"*:* Significant.
   - "*Only label top most significant*"*:* 10
   (i) In "*Plot options*":
   - "*Label* boxes"*:* No.

Figure. 6b shows a volcano plot for this dataset. The significantly differentially expressed genes (using thresholds of adjusted *p*-value 0.05 and absolute $\log_2$FC of 1) are colored red if they are upregulated and blue if they are downregulated. The top ten most significantly differentially expressed genes by P value are labeled. In this plot we can see that the most significantly upregulated gene is

Ant2, the most downregulated is Kal1, and that the top ten genes are mostly downregulated (8/10 genes).

3.7 Functional Enrichment Analysis

We have extracted genes that are differentially expressed in treated (*ps* gene-depleted) samples compared to untreated samples. Now, we would like to know if the differentially expressed genes are enriched transcripts of genes which belong to more common or specific categories in order to identify biological functions that might be impacted. Gene Ontology (GO) analysis is widely used to reduce complexity and highlight biological processes in genome-wide expression studies. To perform the GO analysis of RNA-Seq data, we will use the **goseq** tool [10]. **goseq** provides methods for performing GO analysis of RNA-Seq data while taking gene length bias into account. **Goseq** could also be applied to other category-based tests of RNA-Seq data, such as KEGG pathway analysis.

goseq needs two files as inputs:

- A tabular file with differentially expressed genes from all genes assayed in the RNA-Seq experiment with two columns: the Gene IDs (unique within the file), in uppercase letters; a Boolean indicating whether the gene is differentially expressed or not ("True" if differentially expressed and "False" if not).
- A file with information about the length of a gene to correct for potential length bias in differentially expressed genes.

Hands-on: Prepare the Datasets for Goseq

1. **Compute an expression on every row** with.
   (a) "*Add expression*": bool(c7 < 0.05)
   (b) "*as a new column to*": the **DESeq2** result file.
2. **Cut** with.
   (a) "*Cut columns*": c1,c8
   (b) "*Delimited by*": Tab.
   (c) "*From*": the output of the **Compute**.
3. **Change Case** with.
   (a) "*From*": the output of the previous **Cut**.
   (b) "*Change case of columns*": c1
   (c) "*Delimited by*": Tab.
   (d) "*To*": Upper case.
4. Rename the output to "Gene IDs and differentially expression".
5. Drag and drop one of the feature length datasets generated by **featureCounts** into this history using the **View all histories**.
6. **Change Case** with.
   (a) "*From*": the feature lengths (output of **featureCounts**).

   (b) "*Change case of columns*": c1
   (c) "*Delimited by*": Tab.
   (d) "*To*": Upper case.
7. Rename the output to "Gene IDs and length".

We now have the two required input files for **goseq**.

Hands-on: Perform GO Analysis

1. **goseq** with.
   (a) "*Differentially expressed genes file*": Gene IDs and differentially expression.
   (b) "*Gene lengths file*": Gene IDs and length.
   (c) "*Gene categories*": Get categories.
      - "*Select a genome to use*": Fruit fly (dm6).
      - "*Select Gene ID format*": Ensembl Gene ID.
      - "*Select one or more categories*": GO: Cellular Component, GO: Biological Process, GO: Molecular Function.
   (d) In "*Output Options*".
      - "*Output Top GO terms plot?*": Yes.
      - "*Extract the DE genes for the categories (GO/KEGG terms)?*": Yes.

The main output of **goseq** is a table ("Ranked category list - Wallenius method") with the following columns for each GO term:

1. GO category ("category").
2. *p*-Value for overrepresentation of the term in the differentially expressed genes ("over_rep_pval").
3. *p*-Value for underrepresentation of the term in the differentially expressed genes (under_rep_pval).
4. Number of differentially expressed genes in this category ("numDEInCat").
5. Number of genes in this category ("numInCat").
6. Details about the term.
7. Ontology with MF for "Molecular Function" (molecular activities of gene products), CC for "Cellular Component" (where gene products are active), BP for"Biological Process" (pathways and larger processes made up of the activities of multiple gene products).
8. *p*-Value for overrepresentation of the term in the differentially expressed genes, adjusted for multiple testing with the Benjamini–Hochberg procedure ("p.adjust.over_represented").

9. *p*-Value for underrepresentation of the term in the differentially expressed genes, adjusted for multiple testing with the Benjamini–Hochberg procedure ("p.adjust.under_represented").

From this table we can extract the 31 overrepresented GO terms (using the **Filter** tool on column 8) and the 83 underrepresented terms (using the **Filter** tool on column 9), and then group them (using **Group data** tool on column 7 and count on column 1) to identify that over the 31 overrepresented GO terms, 20 are BP, 3 CC, and 8 MF.

goseq generates also a graph with the top ten overrepresented GO terms and a table with differentially expressed genes (from the list we provided) associated to the GO terms (DE genes for categories (GO/KEGG terms)).

In this chapter, we covered only GO enrichment analysis with **goseq**, but other gene set enrichment analysis can be done with Galaxy (*see* **Note 8**).

3.8 Sharing the Results

Using Galaxy to perform this analysis makes it is both reusable and shareable. In fact, it is possible to simply extract a workflow from a Galaxy history that describes each step of the analysis (tool with parameters used, connections to previous and following steps). This workflow can then be applied on the same or different data, guaranteeing reproducibility.

Moreover, Galaxy histories and workflows can be effortlessly shared with other selected users (via their Galaxy user account or a link), or made publicly available to anyone. *See* the Galaxy 101 tutorial (https://training.galaxyproject.org/training-material/topics/introduction/tutorials/galaxy-intro-101/tutorial.html) for more details.

4 Conclusion

In this tutorial, we have used the Galaxy platform to perform a complex reference-based RNA-Seq analysis through a web interface in a reproducible and easily shareable way. We extracted meaningful information from the RNA sequencing data, such as which genes are up or downregulated by the depletion of the *pasilla* gene, and also which GO terms are enriched.

5 Notes

1. As alternative to Cutadapt, the **Trim Galore!** or **Trimmomatic** tools can be used.

2. In addition to checking the mapping percentage and quick visual check using IGV, the quality of the data and mapping can be checked further:
   (a) *Duplicate reads*: Duplicate reads can come from highly expressed genes, therefore they are usually kept in RNA-Seq differential expression analysis. But a high percentage of duplicates may indicate an issue, for example, overamplification during PCR of low complexity library. **MarkDuplicates** from the Picard suite (http://broadinstitute.github.io/picard/) can examine aligned records from a BAM file to locate duplicate reads, that is, reads mapping to the same location (based on the start position of the mapping). In general, up to 50% of duplication can be considered normal to obtain. So both our samples are good.
   (b) *Number of reads mapped to each chromosome*: To assess the sample quality (e.g., excess of mitochondrial contamination), check the sex of samples, or see if any chromosome have highly expressed genes, we can check the numbers of reads mapped to each chromosome using **IdxStats** from the **Samtools** suite.
   (c) *Gene body coverage*: The gene body is the different regions of a gene. It is important to check if reads coverage is uniform over gene body or if there is any 5′–3′ bias. For example, a bias toward the 5′ end of genes could indicate degradation of the RNA. Alternatively, a 3′ bias could indicate that the data is from a 3′ assay. To assess this, we can use the **Gene Body Coverage** tool from the RSeQC tool suite [16]. This tool scales all transcripts to 100 nucleotides (using a provided annotation file) and calculates the number of reads covering each nucleotide position.
   (d) *Read distribution across features*: With RNA-Seq data, we expect most reads to map to exons rather than introns or intergenic regions. Before going further in counting and differential expression analysis, it may be interesting to check the distribution of reads across known gene features (exons, CDS, 5′ UTR, 3′ UTR, introns, intergenic regions). For example, a high number of reads mapping to intergenic regions may indicate the presence of DNA contamination. We can use the **Read Distribution** tool from the RSeQC tool suite, which uses the annotation file to identify the position of the different gene features.
3. As an alternative to featureCounts, the **HTSeq-count** [17] tool can be used.
4. RNAs that are typically targeted in RNA-Seq experiments are single stranded (e.g., mRNAs) and thus have polarity (5′ and 3′

ends that are functionally distinct). During a typical RNA-Seq experiment, the information about strandness is lost after both strands of cDNA are synthesized, size selected, and converted into a sequencing library. However, this information can be quite useful for the read counting step, especially for reads located on the overlap of two genes that are on different strands.

Some library preparation protocols create so called *stranded* RNA-Seq libraries that preserve the strand information. This information can be estimated using a tool called **Infer Experiment** from the RSeQC [16] tool suite. This tool takes the BAM files from the mapping, selects a subsample of the reads and compares their genome coordinates and strands with those of the reference gene model (from an annotation file). Based on the strand of the genes, it can gauge whether sequencing is strand-specific, and if so, how reads are stranded (forward or reverse).

5. Alternative tools that could be used instead of DESeq2 are **edgeR** and **limma-voom**.
6. The expression analysis is estimated from read counts and attempts are made to correct for variability in measurements using replicates, which are absolutely essential for accurate results. For your own analysis, we advise you to use at least 3, but preferably 5, biological replicates per condition. It is possible to have different numbers of replicates per condition.

   A technical replicate is an experiment which is performed once but measured several times (e.g., multiple sequencing of the same library). A biological replicate is an experiment performed (and also measured) several times.

   In our data, we have four biological replicates (here called samples) without treatment and three biological replicates with treatment (*pasilla* gene depleted by RNAi).

   We recommend to combine the count tables for different technical replicates (but not for biological replicates) before a differential expression analysis.
7. DESeq2 in Galaxy returns the comparison between the different levels for the first factor, after correction for the variability due to the second factor. In our current case, treated against untreated for any sequencing type. To compare sequencing types, we should run DESeq2 again switching factors: factor 1 (treatment) becomes factor 2 and factor 2 (sequencing) becomes factor 1.

   To compare the effect of two factors, for example to see if there is a difference in the treatment effect detected with paired vs. single end data, we should run DESeq2 another time but with only one factor with the following four levels: treated-PE, untreated-PE, treated-SE, untreated-SE. By selecting "*Output all levels vs all levels of primary factor (use when you have >2 levels for primary factor)*" to "Yes," we can then compare treated-PE vs treated-SE.

8. **goseq** can also be used to identify interesting pathways by replacing GO terms with KEGG pathways. The KEGG database is a collection of pathway maps representing the current knowledge on the molecular interaction, reaction and relation networks. A map can integrate many entities including genes, proteins, RNAs, chemical compounds, glycans, and chemical reactions, as well as disease genes and drug targets.

   From the **goseq** output, we could investigate which genes are involved in which pathways by looking at the second file generated by **goseq**. This can be less cumbersome if the pathways are represented as a diagram: **Pathview** [18] can help to generate automatically pathway representation with information about the genes (e.g., expression).

   Other gene set enrichment tools available for Galaxy include **fgsea** [19] and **EGSEA** [20]. fgsea (fast gene set enrichment analysis) takes a ranked list of genes and some gene sets to test, such as from the Molecular Signatures Database (MSigDB), and identifies enriched gene sets. It produces a table of enriched gene sets and barcode plots showing the ranking of the gene set. MSigDB only provide gene sets for human, but if you are using another species you could first map the nonhuman gene ids to human. EGSEA (Ensemble of Gene Set Enrichment Analyses) is another gene set enrichment tool that takes a table of counts and built-in gene sets, including MSigDB, and runs a number of enrichment algorithms. It produces a table of enriched gene sets and different types of plots, such as KEGG diagrams. EGSEA provides built-in gene sets for human, mouse and rat, including those from MSigDB.

Acknowledgments

This book chapter is based upon the online tutorial "Reference-based RNA-Seq data analysis" (https://training.galaxyproject.org/training-material/topics/transcriptomics/tutorials/ref-based/tutorial.html) © 2019 Bérénice Batut, Mallory Freeberg, Mo Heydarian, Anika Erxleben, Pavankumar Videm, Clemens Blank, Maria Doyle, Nicola Soranzo, Peter van Heusden, licensed under the Creative Commons Attribution 4.0 International Public License (https://creativecommons.org/licenses/by/4.0/).
The authors would like to kindly thank all contributors of the Galaxy Training Material.

References

1. Afgan E, Baker D, Batut B et al (2018) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. Nucleic Acids Res 46:W537–W544
2. Batut B, Hiltemann S, Bagnacani A et al (2018) Community-driven data analysis training for biology. Cell Syst 6:752–758.e1
3. Brooks AN, Yang L, Duff MO et al (2011) Conservation of an RNA regulatory map between Drosophila and mammals. Genome Res 21:193–202
4. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet.journal 17:10–12
5. Dobin A, Davis CA, Schlesinger F et al (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29:15–21
6. Robinson JT, Thorvaldsdóttir H, Winckler W et al (2011) Integrative genomics viewer. Nat Biotechnol 29:24–26
7. Liao Y, Smyth GK, Shi W (2014) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics 30:923–930
8. Ewels P, Magnusson M, Lundin S, Käller M (2016) MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics 32:3047–3048
9. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15:550
10. Young MD, Wakefield MJ, Smyth GK, Oshlack A (2010) Gene ontology analysis for RNA-seq: accounting for selection bias. Genome Biol 11: R14
11. dos Santos G, Schroeder AJ, Goodman JL et al (2015) FlyBase: introduction of the *Drosophila melanogaster* Release 6 reference genome assembly and large-scale migration of genome annotations. Nucleic Acids Res 43: D690–D697
12. Trapnell C, Pachter L, Salzberg SL (2009) TopHat: discovering splice junctions with RNA-Seq. Bioinformatics 25:1105–1111
13. Kim D, Langmead B, Salzberg SL (2015) HISAT: a fast spliced aligner with low memory requirements. Nat Methods 12:357–360
14. Kim D, Paggi JM, Park C et al (2019) Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. Nat Biotechnol 37:907–915
15. Thurmond J, Goodman JL, Strelets VB et al (2019) FlyBase 2.0: the next generation. Nucleic Acids Res 47:D759–D765
16. Wang L, Wang S, Li W (2012) RSeQC: quality control of RNA-seq experiments. Bioinformatics 28:2184–2185
17. Anders S, Pyl PT, Huber W (2015) HTSeq—a Python framework to work with high-throughput sequencing data. Bioinformatics 31:166–169
18. Luo W, Brouwer C (2013) Pathview: an R/Bioconductor package for pathway-based data integration and visualization. Bioinformatics 29:1830–1831
19. Korotkevich G, Sukhov V, Sergushichev A (2019) Fast gene set enrichment analysis. bioRxiv 060012
20. Alhamdoosh M, Ng M, Wilson NJ et al (2017) Combining multiple tools outperforms individual methods in gene set enrichment analyses. Bioinformatics 33:414–424

Chapter 21

RAP: A Web Tool for RNA-Seq Data Analysis

Mattia D'Antonio, Pietro Libro, Ernesto Picardi, Graziano Pesole, and Tiziana Castrignanò

Abstract

Since 1950 main studies of RNA regarded its role in the protein synthesis. Later insights showed that only a small portion of RNA codes for proteins where the rest could have different functional roles. With the advent of Next Generation Sequencing (NGS) and in particular with RNA-seq technology the cost of sequencing production dropped down. Among the NGS application areas, the transcriptome analysis, that is, the analysis of transcripts in a cell, their quantification for a specific developmental stage or treatment condition, became more and more adopted in the laboratories. As a consequence in the last decade new insights were gained in the understanding of both transcriptome complexity and involvement of RNA molecules in cellular processes. For what concerns computational advances, bioinformatics research developed new methods for analyzing RNA-seq data. The comparison among transcriptome profiles from several samples is often a difficult task for nonexpert programmers. Here, in this chapter, we introduce RAP (RNA-Seq Analysis Pipeline), a completely automated web tool for transcriptome analysis. It is a user-friendly web tool implementing a detailed transcriptome workflow to detect differential expressed genes and transcript, identify spliced junctions and constitutive or alternative polyadenylation sites and predict gene fusion events. Through the web interface the researchers can get all this information without any knowledge of the underlying High Performance Computing infrastructure.

Key words HPC, Bioinformatics, Genomics, Transcriptomics, RNA-Seq, Alternative splicing sites, Fusion transcripts

1 Introduction

The analysis of transcriptome through next generation sequencing (NGS) technology is considered today a golden standard and it is completely replacing the expression profiling based on the microarray technology, that dominated the field for more than a decade. With the advent of massively sequencing, more than 10 years ago, the RNA sequencing (RNA-seq) was grown as a technological tool in molecular biology [1–3] to gain a better comprehension of the gene expression process, estimating the expressed mRNAs [4] through the sequencing of a complete transcriptome in any cell/ tissue type and condition. The importance of this sequencing technology is due to the investigation of many aspects of molecular biology, such as the ability of studying mRNA splicing [5], the regulation of gene expression by noncoding RNAs [6] and enhancer RNAs [7]. Over 100,000 alternative isoforms were detected with RNA-seq experiments [5], but identifying functional transcripts is still challenging. In a NGS experiment a standard laboratory workflow begins with RNA extraction and ends with the preparation of the sequencing library. The library is then sequenced generating several million short reads typically hundreds of bases in length. The primary computational application of RNA-seq is to determine the quantitative changes in expression levels between experimental groups (e.g., expression at gene and/or transcript level). Other computational strategies are also investigated (alternative splicing events, alternative polyadenylation sites, fusion events, etc.) depending on the biological questions. However, as the throughput of experimental data continues to grow and bioinformatics is de facto entered in the era of big data [8], interpreting the results of the comparisons between the various RNA-seq samples becomes increasingly complex. Parallel to the technological developments of NGS, huge primary repositories of raw sequencing data (Sequence Read Archive—SRA [9], Tumor Cancer Genome Atlas—TCGA [10], Genotype-Tissue Expres sion—GTex [11], Cancer Cell Line Encyclopedia—CCLE [12]) have been populated with incredible speed and the data deluge. These archives contain, among different kind of omics data, a huge amount of RNA-seq samples available for new analyses [13–15] and reuse of data for in silico comparisons and validations [16].

In the last decade, several pipeline tools have been implemented for RNA-seq data analysis [17–21]. In this context, we have developed RAP, a web tool implemented in cloud on Cineca HPC infrastructure, that allows users to analyze big transcriptomic data in main model organisms [22]. A great benefit of web applications, such as RAP, is the possibility to analyze RNA-Seq data without the need of IT competences nor the knowledge about the underlined High Performance Computing infrastructure (computational resources are completely transparent to the users). In addition, a very intuitive web interface allows to customize the analysis and data results are often provided in tabular fashion allowing to further filter subsets of results. The execution of this pipeline is totally automated and optimized on HPC infrastructure. Through RAP the user is able to quickly perform a very complex transcriptome analysis identifying the differential expressed genes, transcripts, splice junctions, polyadenylation sites, and fusion events, thanks to the use of a simple and intuitive interface. The web interface is based on a cloud architecture that completely hides the underlined HPC infrastructure. Both the dedicated hardware and the software environment is described in the following Subheadings 1.1 and 1.2.

1.1 Computational Hardware

The Galileo supercomputer, whose nodes are also dedicated to transcriptome analysis, has the purpose of enabling new classes of "BigData" bioinformatic applications. It can manage and process large amount of raw data, coming from both experiments or data reuse. Galileo is composed by 1022 nodes made of 2 × 18-cores Intel Xeon E5-2697 v4 at 2.30 GHz. Therefore each node, with 128 GB of RAM, has 36 cores, for a total of 36.792 cores. Galileo was designed to optimize density and performance, allowing to analyze large data repositories in CINECA. The storage area accessed by Galileo nodes is composed by high-throughput GPFS disks for a total amount of about 2 PB. Such a storage is also connected with a large capacity tape library for a total actual amount of 12 PB.

1.2 Computational Software

RAP has been developed as an ensemble of modules interconnected through their dependencies and can be schematized as a direct acyclic graph. Its architecture allows each module to run independently, using data stored in MYSQL relational databases. The program which manages RAP modules is completely written in PHP Object Oriented, while mysql libraries are used for database interactions. RAP integrates several open source third-party analysis tools as well as in-house developed python and bash scripts into one single completely automated pipeline. All required computational tasks are managed and distributed on nodes of the clusters depending on the computational needs.

All the analysis, from the raw sequence data uploading to the achievement of final results can be handled and visualized by using an interactive, web-based graphical user interface (GUI). The RAP web-based GUI is written in PHP: Hypertext Preprocessor (PHP) language using HyperText Markup Language (HTML) and JQuery for a better user interaction. The web interface is based on Foundation 4, an advanced responsive CSS front-end framework. RAP is available at the following web address: https://bioinformatics.cineca.it/rap.

1.3 Computational Bioinformatic Workflow

The bioinformatic analysis workflow (Fig. 1) can be divided in six branches, each focused on specific biological entities:

- A main branch, mandatory, for reads mapping and expression profiling. This branch can be completed with differential analysis at transcript level (branch A) and gene level (branch B).
- Detection of fusion transcripts (branch C). The activation of this branch is optional.
- Detection and quantification of splice junctions. This branch is optional and can be completed by a differential analysis of observed junctions (branch D).

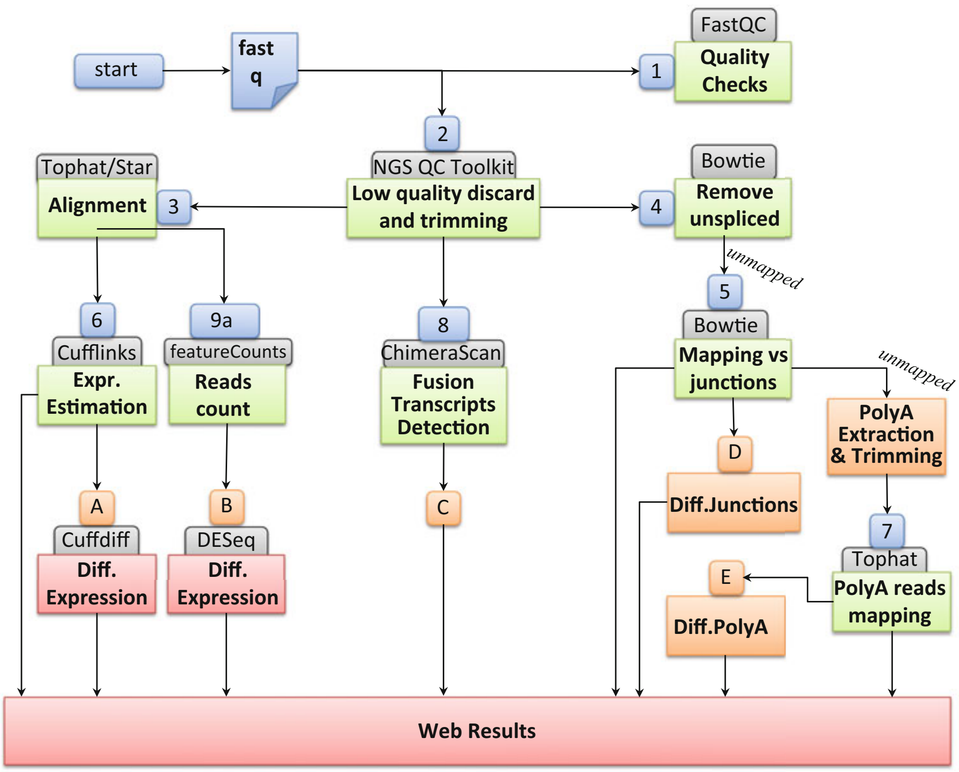


Fig. 1 Completely automated bioinformatic workflow of RAP

- Extraction of polyadenylation sites. This branch is optional and can be completed by a differential analysis of observed poly (A) sites (branch E).

We introduce now with more level of detail the single steps of the pipeline.

Quality checks. The first step of the pipeline provides several quality control checks on raw sequence data by running FastQC tool [23] (Fig. 1, step 1). It produces a set of quality results formatted in HTML report pages, which give a quick overview of whether user data has any problems and therefore how to consider the rest of the downstream analysis. Some samples of the dataset could be indeed biased and contaminated.

Law quality discard and trimming. The action of filtering out low quality reads to increase the overall dataset quality, is performed by the execution of NGS QC Toolkit [24]. It allows to process in the next steps of the pipeline only high-quality reads and therefore provide more robust results in the next mapping phase. This trimming step is launched in parallel with the previous step of quality control (no dependency is applied between the steps 1 and 2). In addition both FastQC and NGSQCtoolkit are optimized to handle several samples at a time and developed to work in a multi-threaded fashion, distributing the computational load on the available HPC resources.

Read mapping. A very crucial phase for analysis consists of aligning the reads against the reference genome and transcriptome (when an annotation is provided). A tool able to perform both kinds of alignment is TopHat2 [25] (Fig. 1, step 3) that includes, as mapping engine, Bowtie2 [26]. TopHat2 is able to use during the analysis the full-length transcripts defined by annotations in order to improve both sensitivity and accuracy. The computational strategy implemented in TopHat2 is able to align the reads with true indels (insertions and deletions), also taking advantage of Bowtie2 ability to detect short indels very accurately. The first step of the method consists in transcriptome mapping of the reads and, in a second step, those unmapped or poorly aligned are mapped against the reference genome in order to detect those reads entirely within exons. Also in this phase some reads, the multiexon spanning reads, are unmapped and, in the next step, they are split into smaller segments that mapped against the reference genome. Fragments are then aligned to the junction (splice site) flanking sequences and stitched together to shape whole read alignments. The last step consists in realigning the reads minimally overlapping the introns against the exons.

As an alternative alignment workflow RAP allows to replace Tophat2 with STAR [27] for faster but still sensitive analyses.

Gene/isoform expression quantification and differential analysis. The building of the transcriptome assembly and the evaluation of the expression level of all detected isoforms is a specific task of Cufflink, which takes as input the results of Tophat2 mapping (Fig. 1, step 6). Cufflink builds the transcript assembly starting by a gene reference annotation. In this phase the user has the opportunity to choose between two assembler algorithms: Cufflinks and RABT [28] assembler.

The first module assembles aligned RNA-Seq reads into a parsimonious set of transcripts, then estimates the abundances of the transcripts considering how many reads support each one.

The second allows to include in the calculus both reference transcripts and novel assembled genes and isoforms. This second assembler is particularly useful for those organisms where a deep annotation does not already exist. The differential analysis at transcript-level resolution of RNA-seq experiments and controls, based on expression levels calculated by Cufflinks, is performed by Cuffdiff2 (branch A). It both provides more accurate transcript-resolution estimates of changes in gene expression, performing statistical computations at isoform-level resolution, and considers the variability in measurements across biological replicates of an experiment.

At gene expression level, another parallel branch (branch B) in the workflow is executed to estimate the raw-count. The third-party software launched is HTSeq [29] that models each gene as the union of all its exons. In particular htseq-count is the script of HTSeq designed for RNA-Seq data analysis: taken a GTF file as input from the previous step, it counts for each gene how many aligned reads overlap exactly its exons whereas the reads overlapping with more than one gene are discarded. These counts will be used for gene-level differential expression analyses performed by the next step.

Fusion events detection and annotation. A further optional branch of RAP workflow, branch C, can be activated with the aim of detecting chimeric transcripts (Fig. 1, step 8), made of exons from two different genes that encode novel putative proteins. The workflow integrates ChimeraScan [30], a software built over Bowtie aligner to identify putative fusion breakpoints. The reads mapping is performed against a mixture of genome-transcriptome reference. Read pairs not aligned concordantly are split into smaller segments (default = 25 bp) and realigned. Reads that align to distinct references or distant genomic locations of the same reference are added in a list of putative 5–3 transcript pairs that could be chimera candidates. A new reference index from the list of sequences is built and candidate junction-spanning reads are realigned against this index. In order to reduce false-positive chimeras, the incorporated spanning reads are filtered, discarding those supported by few reads or those with fragment sizes greater than the range of the distribution.

Splice junction detection. The analysis of splice junctions is performed by the execution of branch E. High Quality Reads, derived from NGS QC Toolkit, are mapped with Bowtie against the reference genome and are discarded from the initial dataset (Fig. 1, step 4). The unmapped reads, that may potentially contain a splicing site, are mapped again by using always Bowtie to a custom-built splice junction library (Fig. 1, step 5). This reference is built starting from a gene annotation model in GTF format [31]. It includes two different categories of splice junctions: known junctions derived from RefSeq [32] and novel junctions, obtained through a combinatorial exon skipping procedure by considering all compatible exon skipping patterns.

Polyadenylation site detection. Another optional branch (E) is executed in cascade from the previous branch D and concerns the analysis of Polyadenylation sites. Residual reads still unmapped from alignments against genome, transcriptome and junctions may contain information about polyadenylation sites (Fig. 1, step 7). In this phase, Poly(A) tags (reads containing a stretch of A at the end of the sequence) are extracted, trimmed, and aligned to the genome. Another spliced alignment with Tophat2 is applied to verify if the sequence also contains a splice junction related to the final exon. As final step, a parsing procedure is applied to annotate the concurrent occurrence of polyadenylation signal (PAS) sequences. In addition to the canonical polyadenylation signal (AAUAAA) a total of 10 known variants are considered [33].

Differential expression analysis. The correct identification of differentially expressed biological entities (such as, in RAP workflow, the identification of differential expressed genes, transcripts, polyadenylation sites, and splice junctions) between specific conditions, each eventually represented by more replicates, is a key in the understanding phenotypic variation. The branches deputed to this kind of analysis are respectively, branch A, B, D, and E in Fig. 1. In particular, RAP detects differentially expressed genes by using DESeq [29] taking as input raw counts calculated by HTSeq (Fig. 1, step B). According to the DESeq algorithm the variance is the sum of a term of raw variance (derived from biological variability) and end of gunshot noise (from counts uncertainty). This method allows to process data without or with very few replicates, putting genes together with similar expression levels.

Cuffdiff2 [34] is the tool used to estimate the differential expressed transcripts from transcript abundances determined in the previous step by Cufflinks (Fig. 1, step A). It calculates differential analysis at transcript-level and controls the variability across replicates and the uncertainty in abundance expression estimates caused by ambiguously mapped reads. In case of incorrect rejections of a true null hypothesis (false positives) it introduces also the Benjamini–Hochberg correction [35] for multiple testing of differential expression (false discovery rate, FDR).

The differential expression analysis of junctions and polyadenylation sites, performed in branches D, E are managed by two home-made PHP parser scripts specific to the output format results from the respectively previous steps 7 and 5 in Fig. 1.

1.4 Reference Genomes and Transcripts Included in RAP

RAP supports the analysis of RNA-seq reads from several organisms. At present, the workflow is available for *Homo sapiens* (genomes hg18, hg19, and hg38), *Mus musculus* (genomes mm9 and mm10), *Rattus norvegicus* (genome rn4), *Drosophila melanogaster* (genome dm3), *Saccharomyces cerevisiae* (genome sacCer3), *Equus ferus caballus* (genome equCab2), *Danio rerio* (genome danRer10), *Zea mays* (genomes maize3 and Mo17_v1), and *Arabidopsis thaliana* (genome tair10). Additional reference genomes and annotations can be added on the HPC cloud becoming available for further analysis, upon users' request.

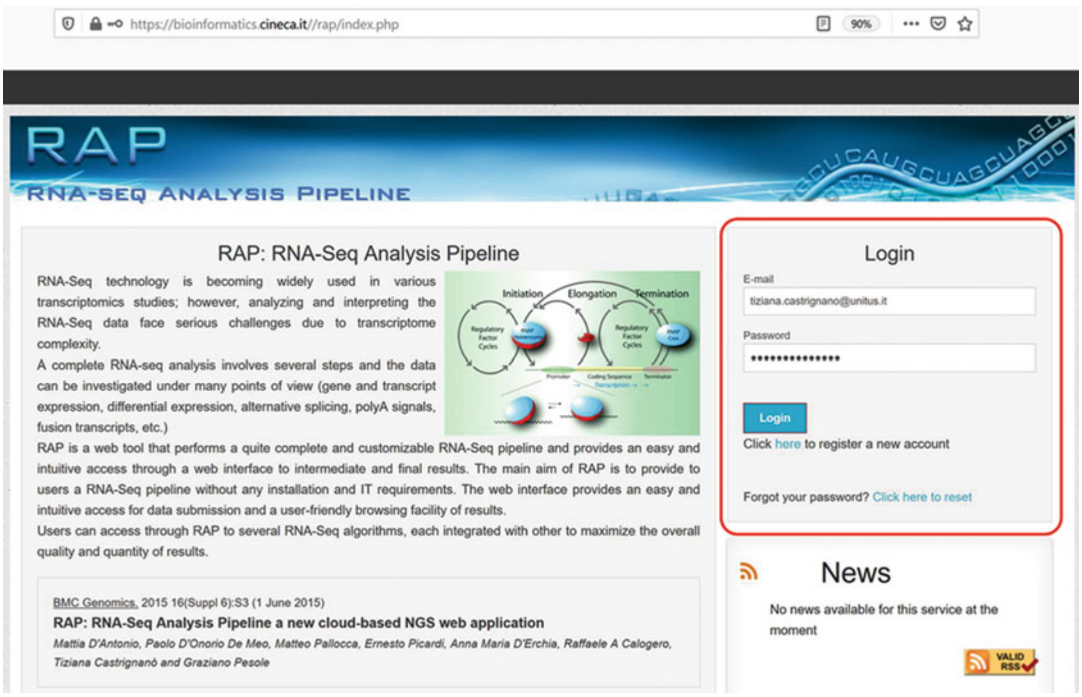
2 Material

To access the RAP pipeline and results it is not required any special hardware or software, apart from a common web browser (Chrome, Firefox, Internet Explorer, and Safari are mostly all supported).

Each account is granted for a 60 days period with a limitation of 2 projects, 2 analyses per projects and 12 files per analysis. Upload of files via web is limited to 2 GB but a FTP account can be requested to be able to upload files without any size limitation. Files can be uploaded as compressed archives (zip, gz) so a compression software can be very useful to limit the bandwidth.

3 Methods

To ensure the confidentiality of data, the use of RAP requires a personal account and by default only the data owner can access to uploaded files and obtained results. All academic users who provide an institutional email address can request for an account via the registration module on the website (the logging form for the registration and authentication is highlighted in red square in Fig. 2).



RAP: RNA-Seq Analysis Pipeline

RNA-Seq technology is becoming widely used in various transcriptomics studies; however, analyzing and interpreting the RNA-Seq data face serious challenges due to transcriptome complexity.

A complete RNA-seq analysis involves several steps and the data can be investigated under many points of view (gene and transcript expression, differential expression, alternative splicing, polyA signals, fusion transcripts, etc.)

RAP is a web tool that performs a quite complete and customizable RNA-Seq pipeline and provides an easy and intuitive access through a web interface to intermediate and final results. The main aim of RAP is to provide to users a RNA-Seq pipeline without any installation and IT requirements. The web interface provides an easy and intuitive access for data submission and a user-friendly browsing facility of results.

Users can access through RAP to several RNA-Seq algorithms, each integrated with other to maximize the overall quality and quantity of results.

BMC Genomics, 2015 16(Suppl 6):S3 (1 June 2015)

RAP: RNA-Seq Analysis Pipeline a new cloud-based NGS web application

Mattia D'Antonio, Paolo D'Onorio De Meo, Matteo Pallocca, Ernesto Picardi, Anna Maria D'Erchia, Raffaele A Calogero, Tiziana Castrignanò and Graziano Pesole

Login

E-mail: tiziana.castrignano@unitus.it

Password: ••••••••••••••

Login

Click here to register a new account

Forgot your password? Click here to reset

News

No news available for this service at the moment

Fig. 2 RAP Login Form

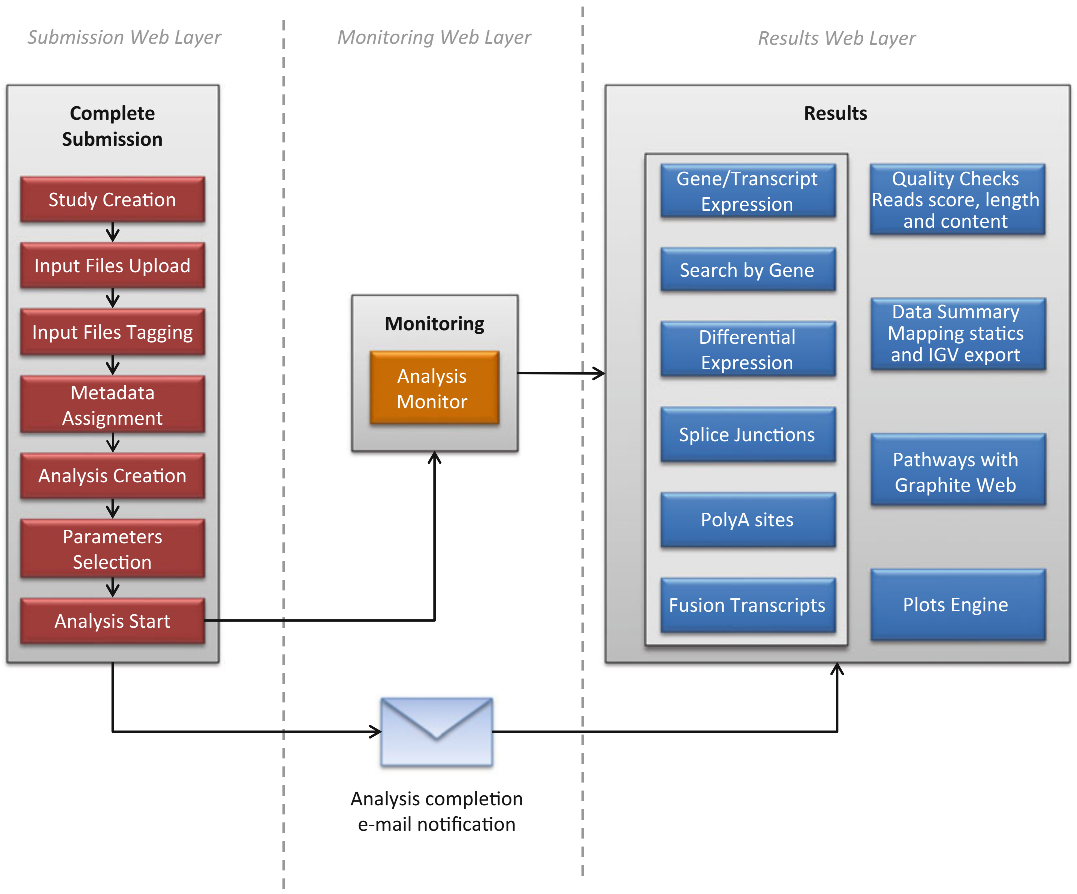


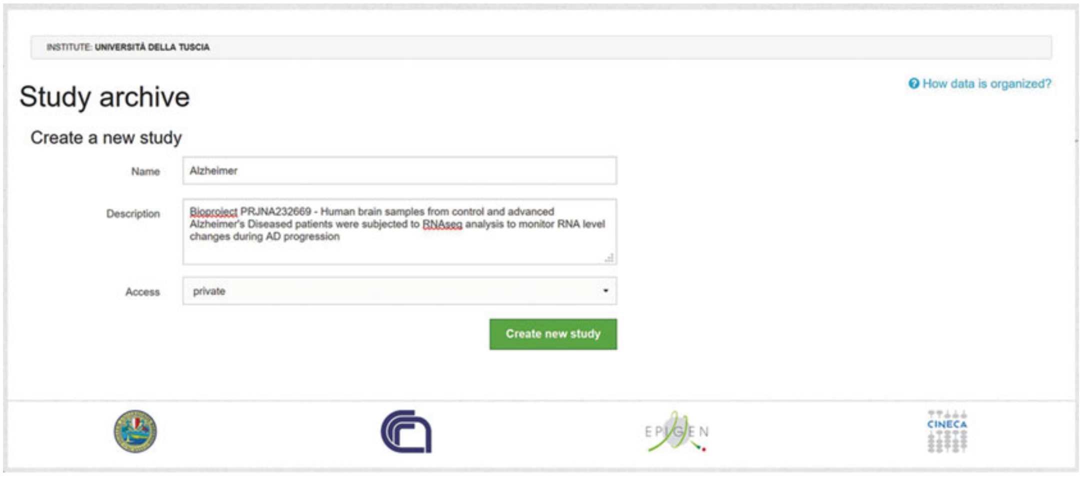
Fig. 3 RAP Web User Interface workflow

A workflow scheme of RAP web user interface is shown in Fig. 3.

The first step to submit a dataset to RAP is the creation of a new study. RAP implements a data architecture inspired by the data format standard proposed by the European Nucleotide Archive (ENA) curated by the European Bioinformatics Institute (EBI). According to ENA guidelines, a study is a homogenous collection of data about a single sequencing project. The creation of a new study in RAP only requires little information such as a title, a description and an access level (private, group, or public).

A private study can be accessed only by the owner while a public study will be accessed by everyone. After the creation (Fig. 4a), the user can open the study by clicking its name or using the view icon. The owner can also edit the given information or delete own studies (Fig. 4b).

After the Study creation, the web user interface shows a workflow composed of three steps: “Add Files,” “Process Files,” and “Analyze Files” (Fig. 5).


INSTITUTE: UNIVERSITÀ DELLA TUSCIA

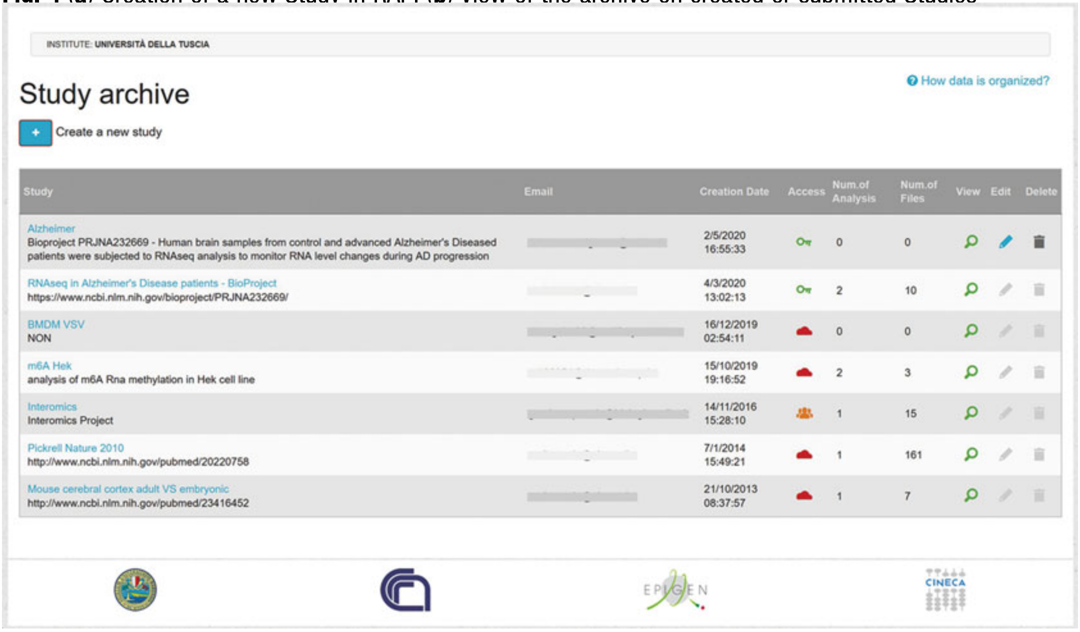
Study archive

How data is organized?

Create a new study

Name	Alzheimer
Description	Bioproject PRJNA232669 - Human brain samples from control and advanced Alzheimer's Diseased patients were subjected to RNAseq analysis to monitor RNA level changes during AD progression
Access	private

Fig. 4 (**a**) Creation of a new Study in RAP. (**b**) View of the archive on created or submitted Studies


INSTITUTE: UNIVERSITÀ DELLA TUSCIA

Study archive

How data is organized?

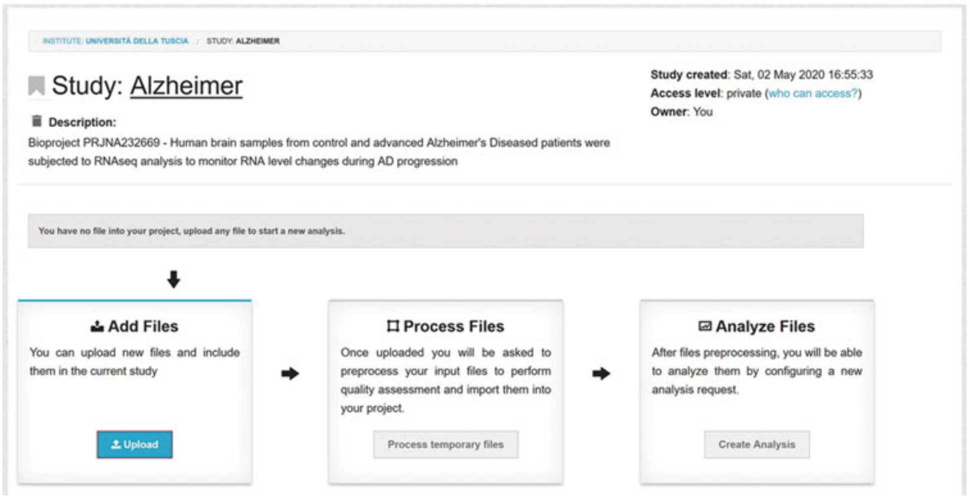
Create a new study

Study	Email	Creation Date	Access	Num.of Analysis	Num.of Files	View	Edit	Delete
Alzheimer Bioproject PRJNA232669 - Human brain samples from control and advanced Alzheimer's Diseased patients were subjected to RNAseq analysis to monitor RNA level changes during AD progression	[redacted]	2/5/2020 16:55:33	🔑	0	0	🔍	✎	🗑
RNAseq in Alzheimer's Disease patients - BioProject https://www.ncbi.nlm.nih.gov/bioproject/PRJNA232669/	[redacted]	4/3/2020 13:02:13	🔑	2	10	🔍	✎	🗑
BMDM VSV NON	[redacted]	16/12/2019 02:54:11	☁	0	0	🔍	✎	🗑
m6A Hek analysis of m6A Rna methylation in Hek cell line	[redacted]	15/10/2019 19:16:52	☁	2	3	🔍	✎	🗑
Interomics Interomics Project	[redacted]	14/11/2016 15:28:10	👥	1	15	🔍	✎	🗑
Pickrell Nature 2010 http://www.ncbi.nlm.nih.gov/pubmed/20220758	[redacted]	7/1/2014 15:49:21	☁	1	161	🔍	✎	🗑
Mouse cerebral cortex adult VS embryonic http://www.ncbi.nlm.nih.gov/pubmed/23416452	[redacted]	21/10/2013 08:37:57	☁	1	7	🔍	✎	🗑

Fig. 4 (continued)

Before starting any analysis, the user has to upload one or more input files. The upload engine offers several options for the submission of the input files: Web Upload, Web Link and FTP protocol (Fig. 6).

The Web Upload is implemented by using JavaScript and allows the user to upload several files at once, but the size of the single file is limited to 2 Gb. The user can verify the upload progress and interact with the system by adding or removing files also during the transfer. Nevertheless, any web based upload widget has limitations, for example during the upload process the page cannot be


INSTITUTE: UNIVERSITÀ DELLA TUSCIA / STUDY: ALZHEIMER

Study: Alzheimer

Study created: Sat, 02 May 2020 16:55:33
Access level: private (who can access?)
Owner: You

Description:
Bioproject PRJNA232669 - Human brain samples from control and advanced Alzheimer's Diseased patients were subjected to RNAseq analysis to monitor RNA level changes during AD progression

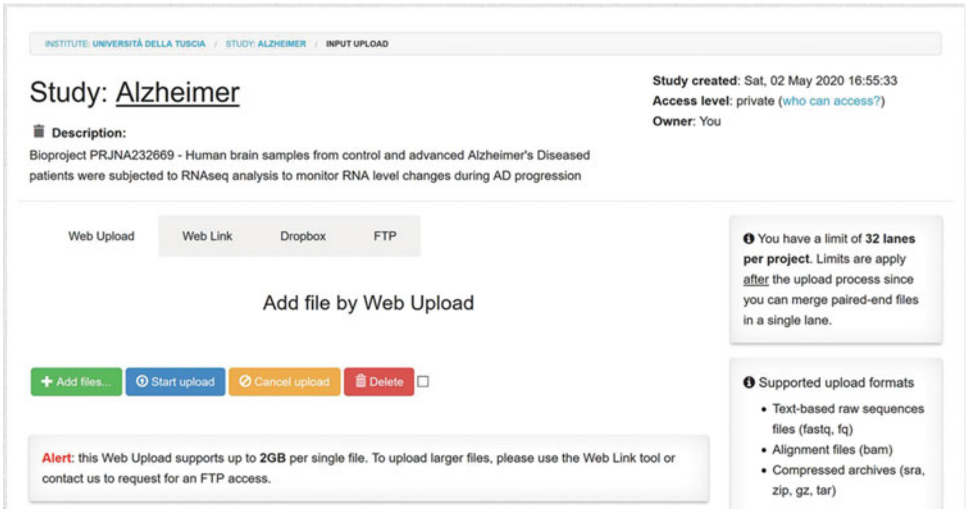
You have no file into your project, upload any file to start a new analysis.

Add Files
You can upload new files and include them in the current study

Process Files
Once uploaded you will be asked to preprocess your input files to perform quality assessment and import them into your project.

Analyze Files
After files preprocessing, you will be able to analyze them by configuring a new analysis request.

Fig. 5 Workflow of the three main steps: "Add Files," "Process Files," and "Analyze Files"


INSTITUTE: UNIVERSITÀ DELLA TUSCIA / STUDY: ALZHEIMER / INPUT UPLOAD

Study: Alzheimer

Study created: Sat, 02 May 2020 16:55:33
Access level: private (who can access?)
Owner: You

Description:
Bioproject PRJNA232669 - Human brain samples from control and advanced Alzheimer's Diseased patients were subjected to RNAseq analysis to monitor RNA level changes during AD progression

Add file by Web Upload

You have a limit of 32 lanes per project. Limits are apply after the upload process since you can merge paired-end files in a single lane.

Supported upload formats
- Text-based raw sequences files (fastq, fq)
- Alignment files (bam)
- Compressed archives (sra, zip, gz, tar)

Alert: this Web Upload supports up to 2GB per single file. To upload larger files, please use the Web Link tool or contact us to request for an FTP access.

Fig. 6 Uploading options for input files

closed, otherwise the upload is interrupted. To overcome these limitations as well as the maximum allowed file size the user can provide one or more links and the system will handle the download in batch. The user will be notified by email when downloads are completed. As a third option, the user can access the RAP user space by using a FTP client.

Several input formats are supported such as text-based raw sequences produced by Illumina sequencing platforms (i.e., FASTQ), prealigned data (i.e., BAM and SAM), and compressed reads (i.e., SRA archives). The user can also upload these files in a compressed archive to speed up the uploading process (several common compressed formats are supported, such as zip, tar, gzip, and bz2).

At the end of the upload if the files are produced through a paired-end sequencing protocol, the user will be allowed to specify each pair of files that comes from the same lane, using a simple drag and drop utility provided by the GUI. The same utility can also be used to reorder lanes, if needed. After that, the user has to assign to each file (or pair of files for PE read data) a unique label (file tagging). A good label should be short and explanatory enough to recognize the input since it will be used in the results pages as replacement for the file raw name. The user can also associate one or more samples, adding information about the sequenced material. This metadata information (i.e., organism, tissue, cellular line, phenotype, and strain) can be useful to describe the input files and are especially important in the case of public experiments.

After the metadata assignment, the uploaded files are imported into the project and can be used to start a new analysis by selecting one or more inputs. The user can choose between two different workflows: one based on Tophat2 and the other based on STAR alignment algorithm. Before starting the analysis the user can accept a set of default parameters to perform a standard workflow or can customize the parameters to tailor the analysis on own data. Parameters are divided into six categories: Common parameters, Quality check and filtering, Genome spliced alignment, Transcript assembly and abundance estimation, Determination of polyA reads, Gene fusions detections in paired-end RNA-Seq datasets.

The first category contains parameters common to all or many pipelines modules, such as the reference database and the Reference-GTF. If input files have been associated with an organism during the annotation phase, only database for such organism are reported here, simplifying the analysis customization and thus preventing potential errors. A set of flags (search-junctions, search-polya, search-chimeric) enable or disable the optional branches.

With the Quality check and filtering parameters the user can modify the behavior of quality control and trimming module. With the quality and length parameters is respectively possible to modify the cutoff value for the PHRED quality score for high-quality filtering (default value is 20) and the percentage of read length that should be of given quality (default value is 70%). This module can also remove primers and adaptors by selecting one of provided libraries (Genomic DNA/Chip-Seq Library, Paired End DNA Library, DpnII gene expression Library, NlaIII gene expression Library, Small RNA Library, Multiplexing DNA Library) or uploading a file containing user defined sequences (one per line).

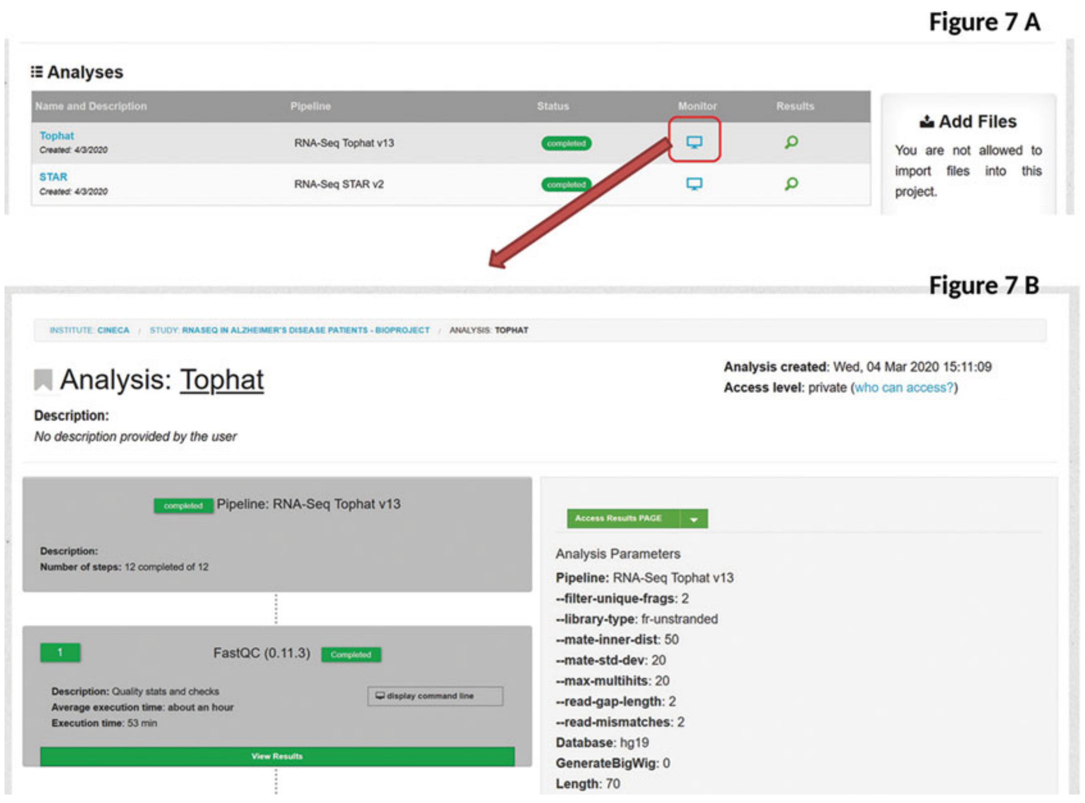
With Genome alignment parameters, the user can customize the mapping phase performed by TopHat2 or STAR (based on the selected workflow). An additional option (GenerateBigWig) allows the user to request the automatic conversion of alignments into

BigWig format. It is a convenient file format for display dense data to be loaded into the University of California, Santa Cruz (UCSC) Genome Browser.

In the Transcript assembly and abundance estimation section the user can enable the reconstruction of novel-transcripts or provide a GTF list (MaskFile) containing transcripts to be ignored during the assembly.

Finally, in the Determination of poly(A) reads and Gene fusions detections in paired-end RNA-Seq datasets categories, the user can customize the poly(A) extraction step and chimeric transcripts determination step, respectively.

Once configured, the analysis is automatically submitted on a queue system based on torque Resource Manager. The user can follow the analysis progress through a monitor page (that can be opened by clicking on the monitor icon, highlighted with a red square in Fig. 7a; a list of all steps are displayed (Fig. 7b) with corresponding running status (to be done, queued, running, skipped, completed, error), the used software and parameters, the intermediate output results if completed.



Analyses

Name and Description	Pipeline	Status	Monitor	Results
Tophat Created: 4/3/2020	RNA-Seq Tophat v13	completed		
STAR Created: 4/3/2020	RNA-Seq STAR v2	completed		

Add Files

You are not allowed to import files into this project.

INSTITUTE: CINECA / STUDY: RNASEQ IN ALZHEIMER'S DISEASE PATIENTS - BIOPROJECT / ANALYSIS: TOPHAT

Analysis: Tophat

Description:
No description provided by the user

Analysis created: Wed, 04 Mar 2020 15:11:09
Access level: private (who can access?)

completed Pipeline: RNA-Seq Tophat v13

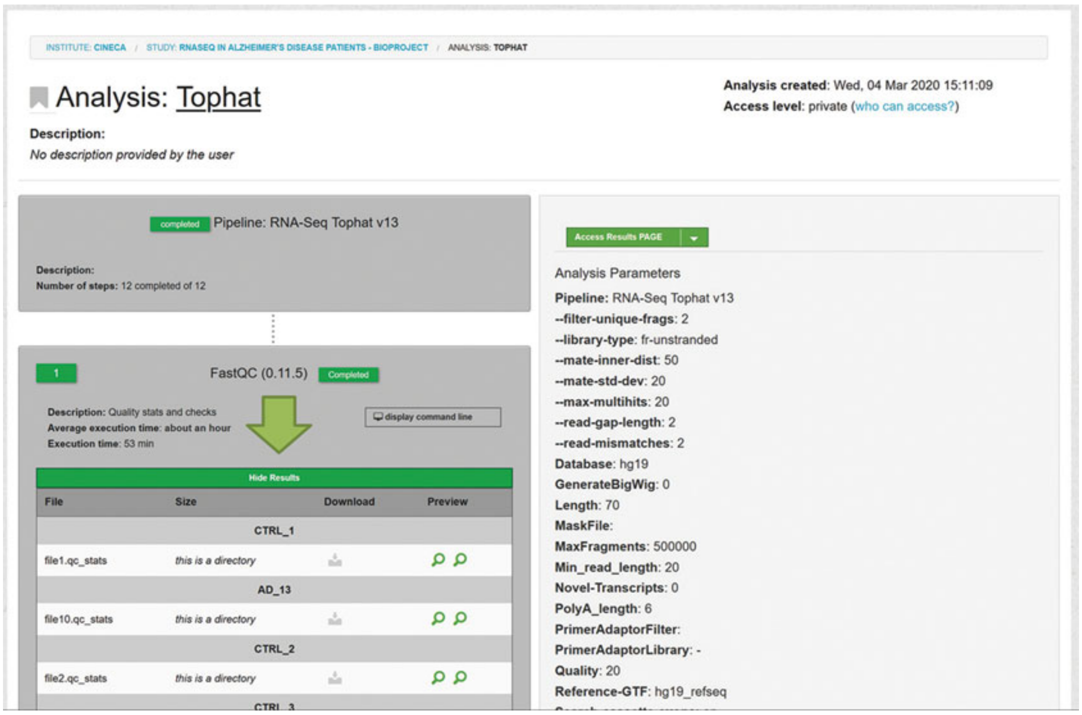
Description:
Number of steps: 12 completed of 12

1 FastQC (0.11.3) Completed

Description: Quality stats and checks
Average execution time: about an hour
Execution time: 53 min

Analysis Parameters
Pipeline: RNA-Seq Tophat v13
--filter-unique-frags: 2
--library-type: fr-unstranded
--mate-inner-dist: 50
--mate-std-dev: 20
--max-multihits: 20
--read-gap-length: 2
--read-mismatches: 2
Database: hg19
GenerateBigWig: 0
Length: 70

Fig. 7 (**a**) list of the launched analyses (based on Tophat and Star aligners respectively with monitor icon squared in red); (**b**) a screenshot of the monitoring page associated to Tophat analysis



INSTITUTE: CINECA / STUDY: RNASEQ IN ALZHEIMER'S DISEASE PATIENTS - BIOPROJECT / ANALYSIS: TOPHAT

Analysis: Tophat

Description:
No description provided by the user

Analysis created: Wed, 04 Mar 2020 15:11:09
Access level: private (who can access?)

completed Pipeline: RNA-Seq Tophat v13

Description:
Number of steps: 12 completed of 12

1 FastQC (0.11.5) Completed

Description: Quality stats and checks
Average execution time: about an hour
Execution time: 53 min

display command line

Hide Results

File	Size	Download	Preview
	CTRL_1		
file1.qc_stats	*this is a directory*		
	AD_13		
file10.qc_stats	*this is a directory*		
	CTRL_2		
file2.qc_stats	*this is a directory*		
	CTRL_3		

Access Results PAGE

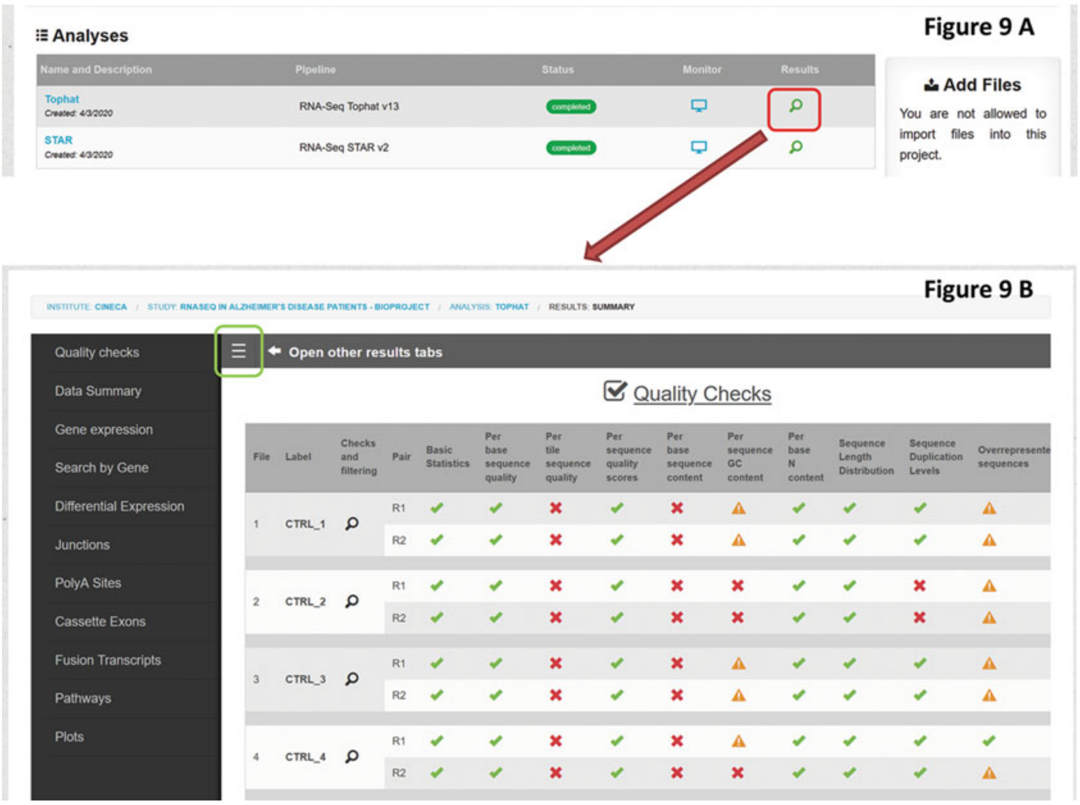
Analysis Parameters
Pipeline: RNA-Seq Tophat v13
--filter-unique-frags: 2
--library-type: fr-unstranded
--mate-inner-dist: 50
--mate-std-dev: 20
--max-multihits: 20
--read-gap-length: 2
--read-mismatches: 2
Database: hg19
GenerateBigWig: 0
Length: 70
MaskFile:
MaxFragments: 500000
Min_read_length: 20
Novel-Transcripts: 0
PolyA_length: 6
PrimerAdaptorFilter:
PrimerAdaptorLibrary: -
Quality: 20
Reference-GTF: hg19_refseq

Fig. 8 Example of list of result files produced by the FastQC step; the results are available after the completion of the step and can be visualized by clicking on the green bar (pointed by the big green arrow into the Monitor Page)

After the analysis completion of each step the result files can be accessed from the monitoring page (*see* Fig. 8) and the analysis summary table in the project page. Analysis results can be accessed from the analysis summary table (*see* Fig. 9a) and the output from Quality checks is shown by default. Further results can be visualized by enabling the side menu (*see* Fig. 9b).

For a better comprehension about the output organization and visualization of RAP, real data set have been used from project "RNA seq in Alzheimer's Disease patients" https://www.ncbi.nlm.nih.gov/bioproject/PRJNA232669. The output summary results of the analysis of the bioprojet PRJNA232669 are available for users at the following URL:

- https://bioinformatics.cineca.it/rap/results.php?a=7282#data_summary for a summary of RNA-seq metrics.
- https://bioinformatics.cineca.it//rap/results.php?a=7282&gene=%09RPPH1&submit=Search#gene_expression for "Gene and transcript expression summary".
- https://bioinformatics.cineca.it//rap/results.php?a=7282&gene=%09RPPH1&submit=Search#differential_expression for "Differential gene and transcript expression".



Name and Description	Pipeline	Status	Monitor	Results
Tophat Created: 4/3/2020	RNA-Seq Tophat v13	completed		
STAR Created: 4/3/2020	RNA-Seq STAR v2	completed		

File	Label	Checks and filtering	Pair	Basic Statistics	Per base sequence quality	Per tile sequence quality	Per sequence quality scores	Per base sequence content	Per sequence GC content	Per base N content	Sequence Length Distribution	Sequence Duplication Levels	Overrepresented sequences
1	CTRL_1		R1	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠
			R2	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠
2	CTRL_2		R1	✓	✓	✗	✓	✗	✗	✓	✓	✗	⚠
			R2	✓	✓	✗	✓	✗	✗	✓	✓	✗	⚠
3	CTRL_3		R1	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠
			R2	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠
4	CTRL_4		R1	✓	✓	✗	✓	✗	⚠	✓	✓	✓	✓
			R2	✓	✓	✗	✓	✗	✗	✓	✓	✓	⚠

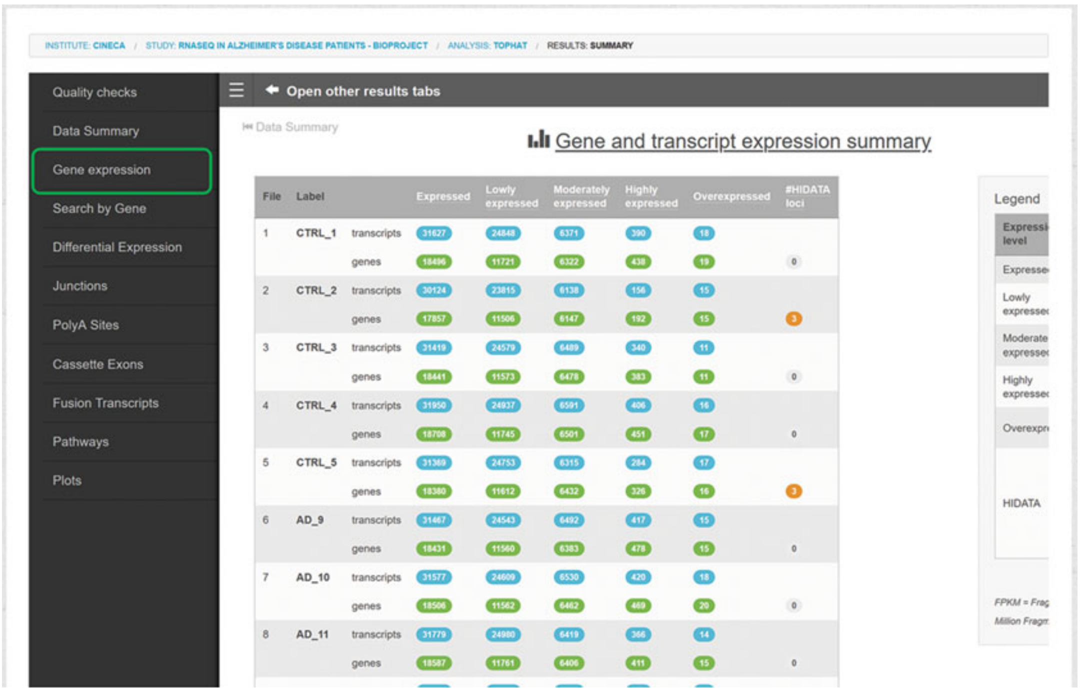
Fig. 9 (**a**) List of results in case of Tophat and Star aligners. (**b**) Result summary page for Quality Checks opened by clicking on the lent icon in the list

- https://bioinformatics.cineca.it//rap/results.php?a=7282#junctions for a "Summary of splice junction".
- https://bioinformatics.cineca.it//rap/results.php?a=7282#fusion_transcripts for "Fusion transcripts".

In this analysis samples named CTRL_* are "control" samples whereas samples named AD_* are Alzheimer's Disease samples.

Each of the URLs, loading a summary page report, allow users to go in details of results of interest. By clicking on each summary count number in the summary table a page opens visualizing the result of interest (genes, transcripts, fusion transcripts, junctions, etc).

Other kind of results ("Gene Expression," "Search by Gene," "Differential Expression," and so on) can be consulted by the user, clicking on the menu item "Open other results tabs," in the top-left corner of the gray frame. For instance, if the user is interested in "Gene Expression" results tab, clicking on the specific tab, the system will show the below navigable results page (Fig. 10):



File	Label		Expressed	Lowly expressed	Moderately expressed	Highly expressed	Overexpressed	#HIDATA loci
1	CTRL_1	transcripts	31627	24848	6371	390	18	
		genes	18496	11721	6322	438	19	0
2	CTRL_2	transcripts	30124	23815	6138	156	15	
		genes	17857	11506	6147	192	15	3
3	CTRL_3	transcripts	31419	24579	6489	340	11	
		genes	18441	11573	6478	383	11	0
4	CTRL_4	transcripts	31950	24937	6591	406	16	
		genes	18708	11745	6501	451	17	0
5	CTRL_5	transcripts	31369	24753	6315	284	17	
		genes	18380	11612	6432	326	16	3
6	AD_9	transcripts	31467	24543	6492	417	15	
		genes	18431	11560	6383	478	15	0
7	AD_10	transcripts	31577	24609	6530	420	18	
		genes	18506	11562	6462	469	20	0
8	AD_11	transcripts	31779	24980	6419	366	14	
		genes	18587	11761	6406	411	15	0

Fig. 10 Gene and Transcript Expression summary page for the analyzed samples

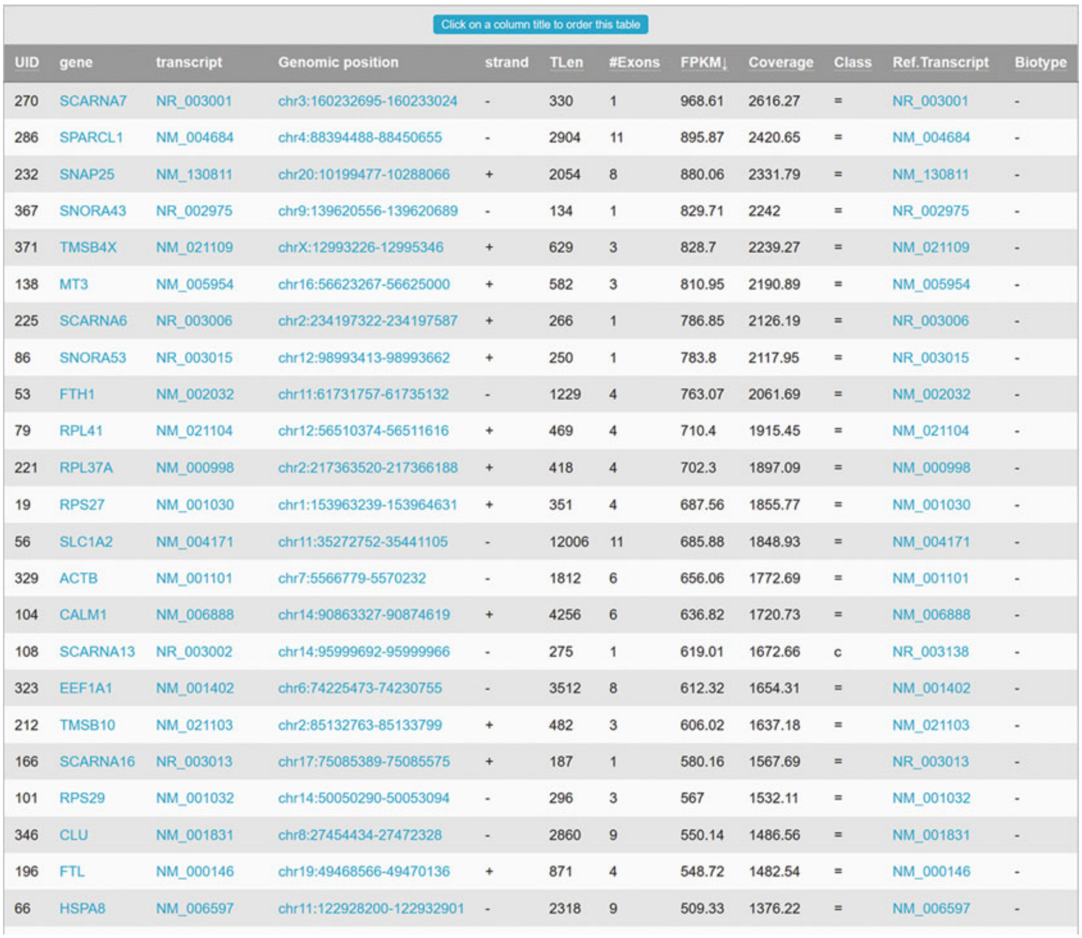
In case the user is interested about "Highly expressed transcripts" in sample named "CTRL_1", by clicking on the blue icon with the number "390" in the first row of Fig. 10, a page opens with the list of the transcripts of interest (genes, transcripts, genomic position, strand, transcript length, number of exons, FPKM, associated Coverage, class compared to reference annotation, ID of reference transcript, and biotype based on gencode annotation) (Fig. 11).

3.1 Analysis Results and Differential Expression

After the completion of the analysis, all output files are parsed into a MySQL database for ease of visualization from the web interface. Results are divided into several sections: Quality checks, Data Summary, Gene Expression, Search by Gene, Differential Expression, Junctions, PolyA Sites, Fusion Transcripts, Pathways, and Plots.

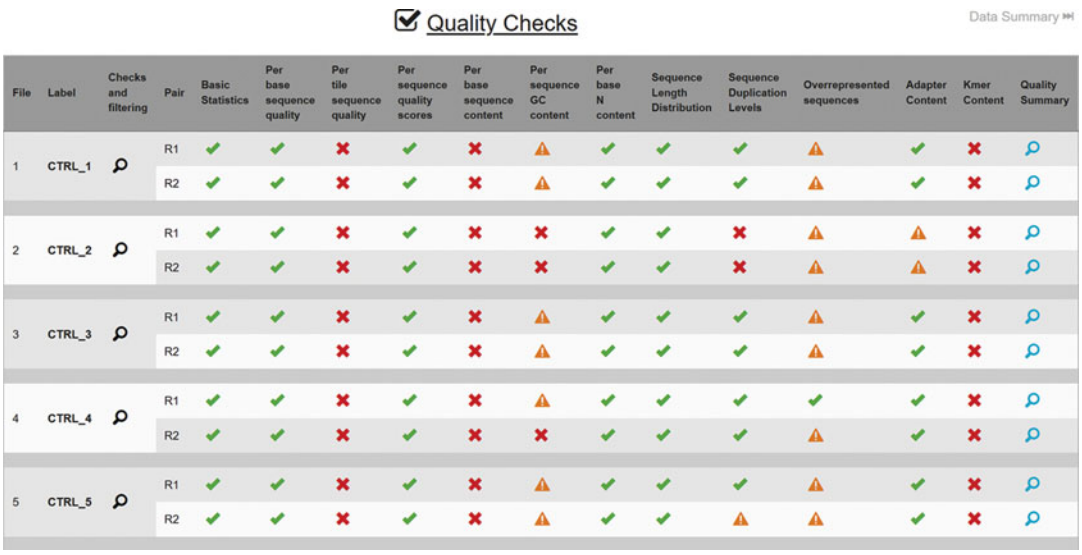
The first section (Quality checks) reports the output provided by FastQC and NGS QC Toolkit, arranged in a comprehensive summary table with color-coded labels to give to the user a prompt quality overview (Fig. 12). Green labels indicate passed filters, red labels point to failed filters and the orange is the color used to report filters warnings. Each label can be explored visualizing the corresponding FastQC output.

The Data Summary section reports a quantitative analysis of obtained results with metrics such as the total amount of short reads (both raw and high-quality reads as filtered by NGS QC


Click on a column title to order this table

UID	gene	transcript	Genomic position	strand	TLen	#Exons	FPKM↓	Coverage	Class	Ref.Transcript	Biotype
270	SCARNA7	NR_003001	chr3:160232695-160233024	-	330	1	968.61	2616.27	=	NR_003001	-
286	SPARCL1	NM_004684	chr4:88394488-88450655	-	2904	11	895.87	2420.65	=	NM_004684	-
232	SNAP25	NM_130811	chr20:10199477-10288066	+	2054	8	880.06	2331.79	=	NM_130811	-
367	SNORA43	NR_002975	chr9:139620556-139620689	-	134	1	829.71	2242	=	NR_002975	-
371	TMSB4X	NM_021109	chrX:12993226-12995346	+	629	3	828.7	2239.27	=	NM_021109	-
138	MT3	NM_005954	chr16:56623267-56625000	+	582	3	810.95	2190.89	=	NM_005954	-
225	SCARNA6	NR_003006	chr2:234197322-234197587	+	266	1	786.85	2126.19	=	NR_003006	-
86	SNORA53	NR_003015	chr12:98993413-98993662	+	250	1	783.8	2117.95	=	NR_003015	-
53	FTH1	NM_002032	chr11:61731757-61735132	-	1229	4	763.07	2061.69	=	NM_002032	-
79	RPL41	NM_021104	chr12:56510374-56511616	+	469	4	710.4	1915.45	=	NM_021104	-
221	RPL37A	NM_000998	chr2:217363520-217366188	+	418	4	702.3	1897.09	=	NM_000998	-
19	RPS27	NM_001030	chr1:153963239-153964631	+	351	4	687.56	1855.77	=	NM_001030	-
56	SLC1A2	NM_004171	chr11:35272752-35441105	-	12006	11	685.88	1848.93	=	NM_004171	-
329	ACTB	NM_001101	chr7:5566779-5570232	-	1812	6	656.06	1772.69	=	NM_001101	-
104	CALM1	NM_006888	chr14:90863327-90874619	+	4256	6	636.82	1720.73	=	NM_006888	-
108	SCARNA13	NR_003002	chr14:95999692-95999966	-	275	1	619.01	1672.66	c	NR_003138	-
323	EEF1A1	NM_001402	chr6:74225473-74230755	-	3512	8	612.32	1654.31	=	NM_001402	-
212	TMSB10	NM_021103	chr2:85132763-85133799	+	482	3	606.02	1637.18	=	NM_021103	-
166	SCARNA16	NR_003013	chr17:75085389-75085575	+	187	1	580.16	1567.69	=	NR_003013	-
101	RPS29	NM_001032	chr14:50050290-50053094	-	296	3	567	1532.11	=	NM_001032	-
346	CLU	NM_001831	chr8:27454434-27472328	-	2860	9	550.14	1486.56	=	NM_001831	-
196	FTL	NM_000146	chr19:49468566-49470136	+	871	4	548.72	1482.54	=	NM_000146	-
66	HSPA8	NM_006597	chr11:122928200-122932901	-	2318	9	509.33	1376.22	=	NM_006597	-

Fig. 11 Details on highly expressed transcripts found for '*CTRL_1*'


Quality Checks

Data Summary ⏭

File	Label	Checks and filtering	Pair	Basic Statistics	Per base sequence quality	Per tile sequence quality	Per sequence quality scores	Per base sequence content	Per sequence GC content	Per base N content	Sequence Length Distribution	Sequence Duplication Levels	Overrepresented sequences	Adapter Content	Kmer Content	Quality Summary
1	CTRL_1	🔍	R1	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠	✓	✗	🔍
			R2	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠	✓	✗	🔍
2	CTRL_2	🔍	R1	✓	✓	✗	✓	✗	✗	✓	✓	✗	⚠	⚠	✗	🔍
			R2	✓	✓	✗	✓	✗	✗	✓	✓	✗	⚠	⚠	✗	🔍
3	CTRL_3	🔍	R1	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠	✓	✗	🔍
			R2	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠	✓	✗	🔍
4	CTRL_4	🔍	R1	✓	✓	✗	✓	✗	⚠	✓	✓	✓	✓	✓	✗	🔍
			R2	✓	✓	✗	✓	✗	✗	✓	✓	✓	⚠	✓	✗	🔍
5	CTRL_5	🔍	R1	✓	✓	✗	✓	✗	⚠	✓	✓	✓	⚠	✓	✗	🔍
			R2	✓	✓	✗	✓	✗	⚠	✓	✓	⚠	⚠	✓	✗	🔍

Fig. 12 The comprehensive summary table of Quality Check results

Toolkit), mapping statistics, junction alignment metrics, information from poly(A) extraction phase, the distributions of mapped reads across functional gene regions and transcripts coverage evaluation. Furthermore this section contains a utility to load data into the Integrative Genomics Viewer (IGV), a free Java desktop application supporting interactive exploration of large-scale genomic data sets on standard desktop computers.

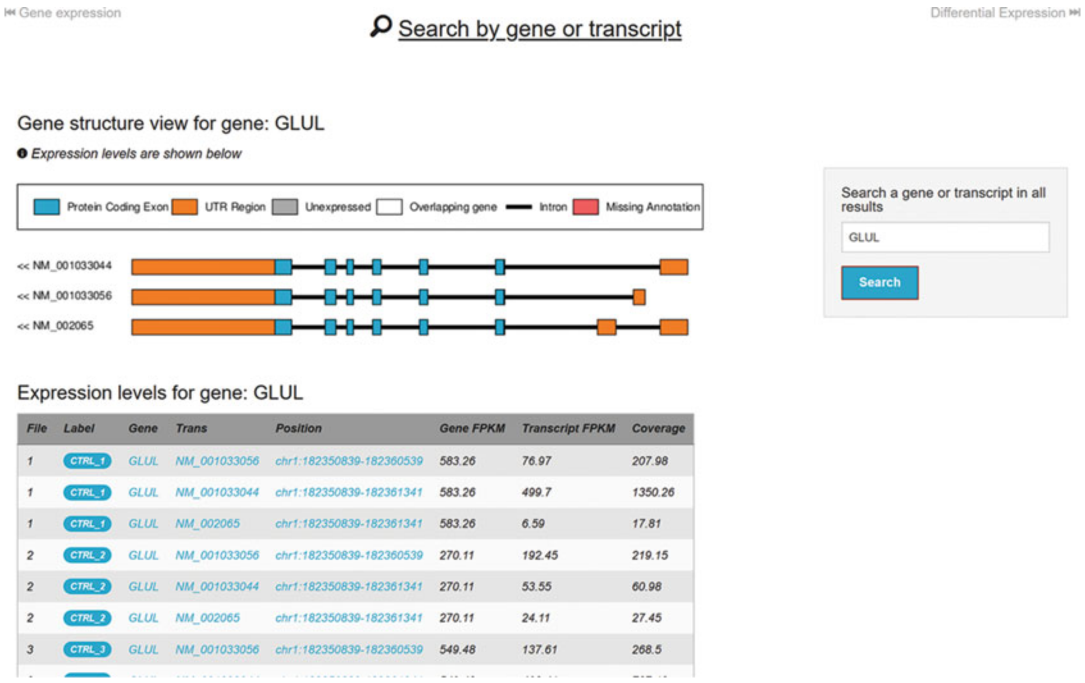
The Gene Expression section reports expression values as estimated by Cufflinks. The summary table reports, for each lane, colored-boxed numbers of both expressed genes (in green) and transcripts (in blue). Four different default FPKM cutoffs are proposed, to give a comprehensive idea about the number of low- or high-expressed genes and transcripts (Fig. 10). Each colored-boxed can be clicked to open the expression overview, a detailed list of all expressed genes (or transcripts) in a given sample with detailed information associated (Fig. 11). This set of results (as any other overview table describe below) can be filtered using customizable thresholds to facilitate the identification of functionally significant variants. Every column can be used to filter results and filters can be combined to produce complex queries. The output tables, as reported after the application of a set of filters, can be exported as textual/Excel files for offline downstream analyses.

The Search by Gene section allows to query simultaneously all expression results. With this form the user can retrieve the expression values of a given gene or transcript obtaining as result a table reporting, for each lane and each isoform the genomic position, the gene FPKM, the transcript FPKM and the transcript coverage. In case of resulting transcripts belong to the same a graphical gene structure is also displayed (Fig. 13).

The Junctions section reports all results obtained by the mapping on the splice junctions library (Fig. 14).

The detail page lists the whole set of observed junctions, each annotated by many information. Both gene and transcript common names are dynamically linked to NCBI, respectively to Gene and Nucleotide databases. Then, several coordinates identify the precise junction by reporting chromosome, start and end position of junction, upstream exon and downstream exon (Fig. 15).

The Poly(A) Sites Section reports a summary table of the number of observed event and coupled with a graphical view of the chromosomal distribution. The user can access to the sortable, filterable, downloadable details page. The frequency of polyadenylation sites (PAS) hexamers is also shown. The canonical PAS (AAUAAA) is clearly expected to overcome the other known variants but, with a sufficient number of sites, the expected relative frequencies should be observed. In the case of bioproject PRJNA232669 the list polyadenylation sites is not available since the library was built without enriching fragments that carry poles (A) tail.



Gene structure view for gene: GLUL

Expression levels are shown below

Expression levels for gene: GLUL

File	Label	Gene	Trans	Position	Gene FPKM	Transcript FPKM	Coverage
1	CTRL_1	GLUL	NM_001033056	chr1:182350839-182360539	583.26	76.97	207.98
1	CTRL_1	GLUL	NM_001033044	chr1:182350839-182361341	583.26	499.7	1350.26
1	CTRL_1	GLUL	NM_002065	chr1:182350839-182361341	583.26	6.59	17.81
2	CTRL_2	GLUL	NM_001033056	chr1:182350839-182360539	270.11	192.45	219.15
2	CTRL_2	GLUL	NM_001033044	chr1:182350839-182361341	270.11	53.55	60.98
2	CTRL_2	GLUL	NM_002065	chr1:182350839-182361341	270.11	24.11	27.45
3	CTRL_3	GLUL	NM_001033056	chr1:182350839-182360539	549.48	137.61	268.5

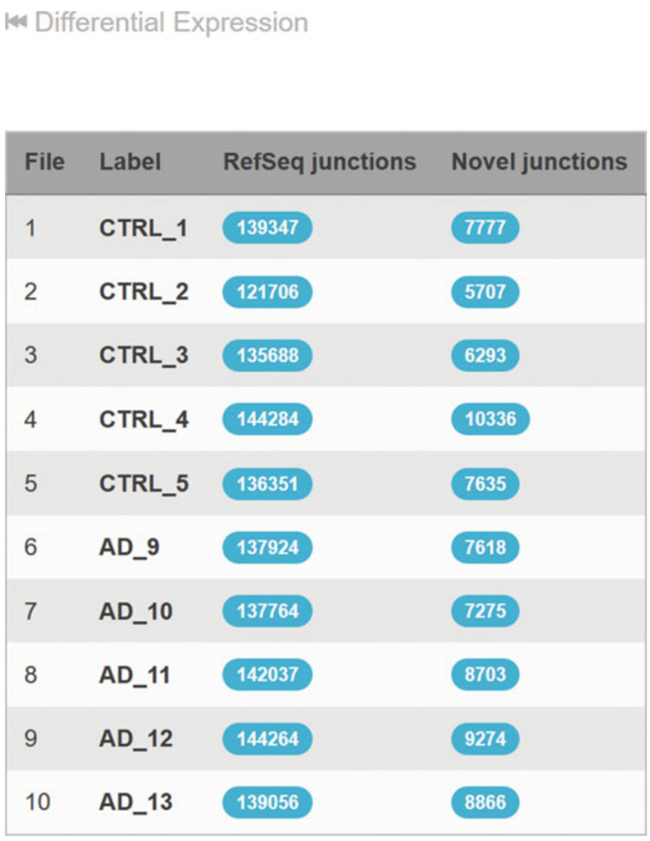
Fig. 13 Result of Search by Gene by querying GLUL. The gene structure and expression levels of the three isoforms are shown

The Fusion Transcripts section reports chimeric isoforms as detected by ChimeraScan. The information schema is the same already proposed in the previous sections: a summary table reports the number of events for each input (Fig. 16) and a details page gives information and annotations for each chimera (Fig. 17). Several information is associated to each chimeric transcript such as the coordinates and gene and transcript ids of both 5′ partner and 3′ partner; the number of supporting reads; and the fusion type (read-through, intrachromosomal, or interchromosomal).

The Pathways section is a utility page used to link results obtained by RAP to an external service, Graphite Web, allowing for pathway analyses and network visualization by using expressed genes as input.

The Plot section imports data obtained from both expression and differential expression sections exploiting the filtering engine and allows the user to created several plots by leveraging the CummeRbund package. Typical plots included in the suite are PCA (principal component analysis), MDS-plot (MultiDimensional Scaling plot), boxplots, scatterplots, density plots, heatmaps, and sashimi plots.

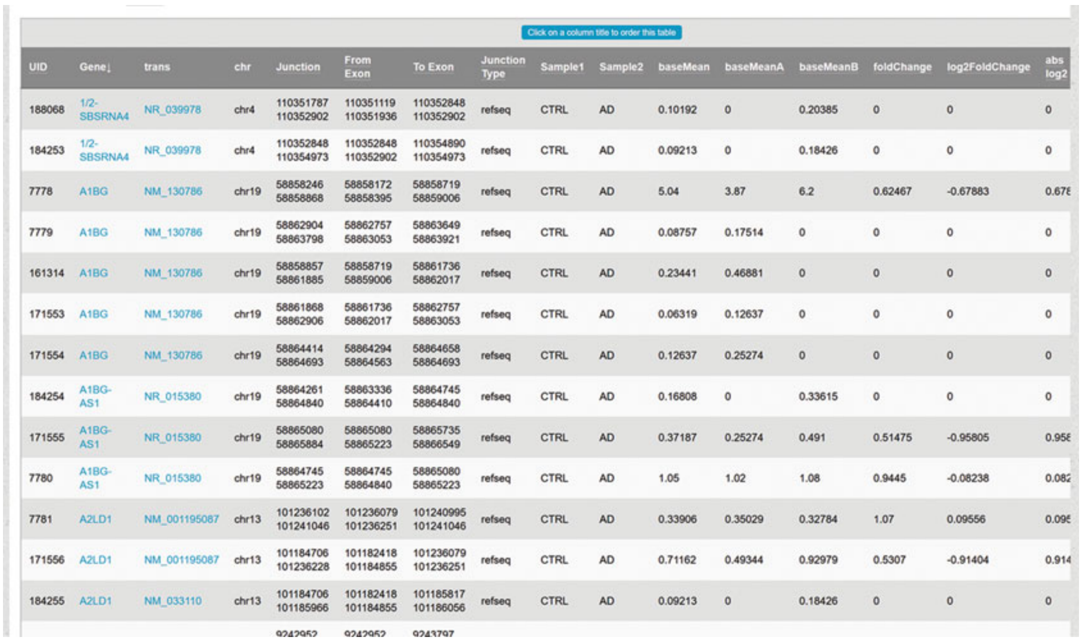
RAP implements several differential expression (DE) analyses, such as transcripts expression, gene expression, Poly(A) site usage, and alternative exon skipping events. RAP allows for the execution



⏮ Differential Expression

File	Label	RefSeq junctions	Novel junctions
1	CTRL_1	139347	7777
2	CTRL_2	121706	5707
3	CTRL_3	135688	6293
4	CTRL_4	144284	10336
5	CTRL_5	136351	7635
6	AD_9	137924	7618
7	AD_10	137764	7275
8	AD_11	142037	8703
9	AD_12	144264	9274
10	AD_13	139056	8866

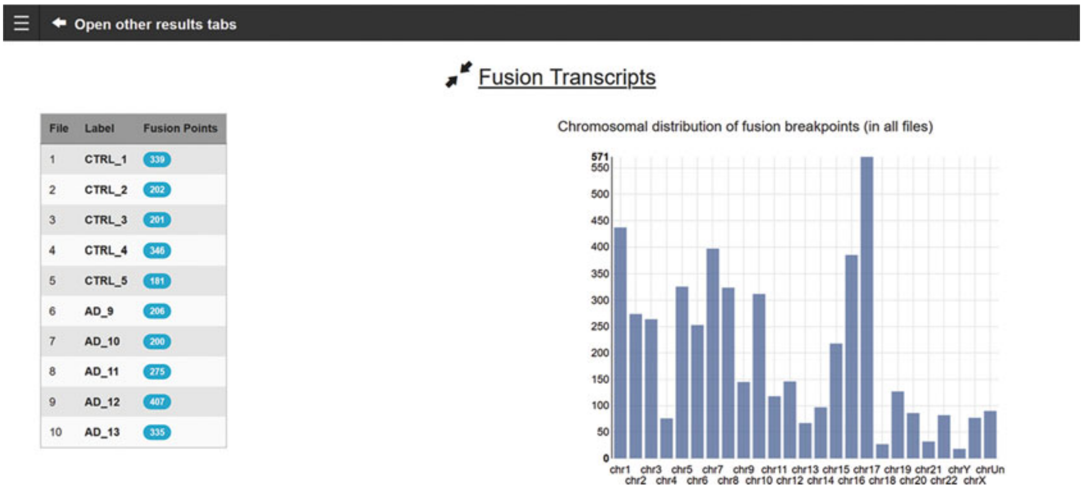
Fig. 14 Summary results of detected junctions



Click on a column title to order this table

UID	Gene↓	trans	chr	Junction	From Exon	To Exon	Junction Type	Sample1	Sample2	baseMean	baseMeanA	baseMeanB	foldChange	log2FoldChange	abs log2
188068	1/2-SBSRNA4	NR_039978	chr4	110351787 110352902	110351119 110351936	110352848 110352902	refseq	CTRL	AD	0.10192	0	0.20385	0	0	0
184253	1/2-SBSRNA4	NR_039978	chr4	110352848 110354973	110352848 110352902	110354890 110354973	refseq	CTRL	AD	0.09213	0	0.18426	0	0	0
7778	A1BG	NM_130786	chr19	58858246 58858868	58858172 58858395	58858719 58859006	refseq	CTRL	AD	5.04	3.87	6.2	0.62467	-0.67883	0.67
7779	A1BG	NM_130786	chr19	58862904 58863798	58862757 58863053	58863649 58863921	refseq	CTRL	AD	0.08757	0.17514	0	0	0	0
161314	A1BG	NM_130786	chr19	58858857 58861885	58858719 58859006	58861736 58862017	refseq	CTRL	AD	0.23441	0.46881	0	0	0	0
171553	A1BG	NM_130786	chr19	58861868 58862906	58861736 58862017	58862757 58863053	refseq	CTRL	AD	0.06319	0.12637	0	0	0	0
171554	A1BG	NM_130786	chr19	58864414 58864693	58864294 58864563	58864658 58864693	refseq	CTRL	AD	0.12637	0.25274	0	0	0	0
184254	A1BG-AS1	NR_015380	chr19	58864261 58864840	58863336 58864410	58864745 58864840	refseq	CTRL	AD	0.16808	0	0.33615	0	0	0
171555	A1BG-AS1	NR_015380	chr19	58865080 58865884	58865080 58865223	58865735 58866549	refseq	CTRL	AD	0.37187	0.25274	0.491	0.51475	-0.95805	0.95
7780	A1BG-AS1	NR_015380	chr19	58864745 58865223	58864745 58864840	58865080 58865223	refseq	CTRL	AD	1.05	1.02	1.08	0.9445	-0.08238	0.08
7781	A2LD1	NM_001195087	chr13	101236102 101241046	101236079 101236251	101240995 101241046	refseq	CTRL	AD	0.33906	0.35029	0.32784	1.07	0.09556	0.09
171556	A2LD1	NM_001195087	chr13	101184706 101236228	101182418 101184855	101236079 101236251	refseq	CTRL	AD	0.71162	0.49344	0.92979	0.5307	-0.91404	0.914
184255	A2LD1	NM_033110	chr13	101184706 101185966	101182418 101184855	101185817 101186056	refseq	CTRL	AD	0.09213	0	0.18426	0	0	0
				9242952	9242952	9243797									

Fig. 15 The report page of observed junctions, each annotated by many information



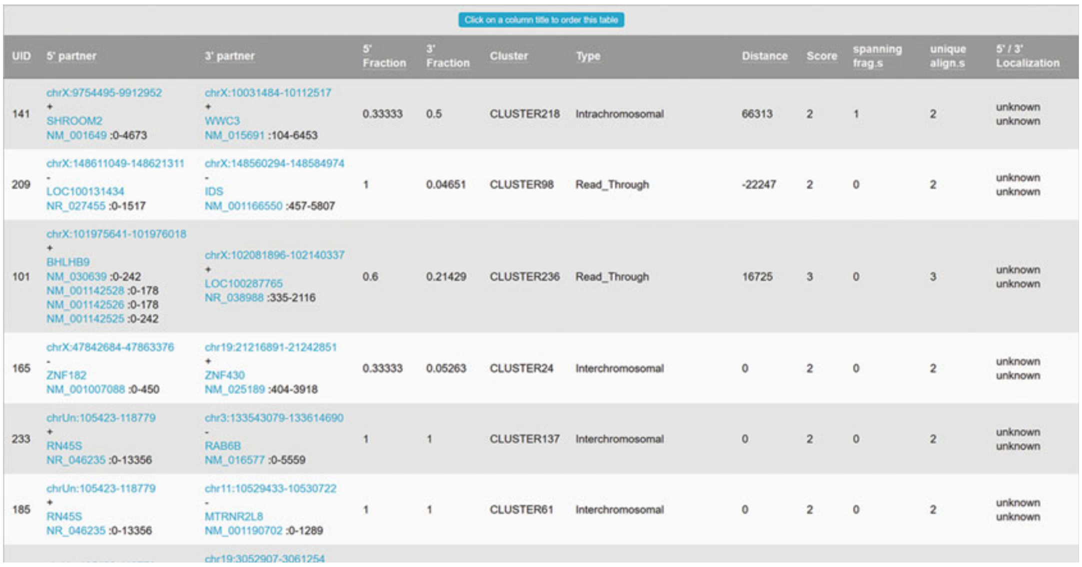
☰ Open other results tabs

Fusion Transcripts

File	Label	Fusion Points
1	CTRL_1	339
2	CTRL_2	202
3	CTRL_3	201
4	CTRL_4	346
5	CTRL_5	181
6	AD_9	206
7	AD_10	200
8	AD_11	275
9	AD_12	407
10	AD_13	335

Chromosomal distribution of fusion breakpoints (in all files)
571
550
500
450
400
350
300
250
200
150
100
50
0
chr1 chr3 chr5 chr7 chr9 chr11 chr13 chr15 chr17 chr19 chr21 chrY chrUn
chr2 chr4 chr6 chr8 chr10 chr12 chr14 chr16 chr18 chr20 chr22 chrX

Fig. 16 Summary table of the number of gene fusion events for each input and dynamical chromosomal distribution of fusion breakpoints



Click on a column title to order this table

UID	5' partner	3' partner	5' Fraction	3' Fraction	Cluster	Type	Distance	Score	spanning frag.s	unique align.s	5' / 3' Localization
141	chrX:9754495-9912952 + SHROOM2 NM_001649 :0-4673	chrX:10031484-10112517 + WWC3 NM_015691 :104-6453	0.33333	0.5	CLUSTER218	Intrachromosomal	66313	2	1	2	unknown unknown
209	chrX:148611049-148621311 - LOC100131434 NR_027455 :0-1517	chrX:148560294-148584974 - IDS NM_001166550 :457-5807	1	0.04651	CLUSTER98	Read_Through	-22247	2	0	2	unknown unknown
101	chrX:101975641-101976018 + BHLHB9 NM_030639 :0-242 NM_001142528 :0-178 NM_001142526 :0-178 NM_001142525 :0-242	chrX:102081896-102140337 + LOC100287765 NR_038988 :335-2116	0.6	0.21429	CLUSTER236	Read_Through	16725	3	0	3	unknown unknown
165	chrX:47842684-47863376 - ZNF182 NM_001007088 :0-450	chr19:21216891-21242851 + ZNF430 NM_025189 :404-3918	0.33333	0.05263	CLUSTER24	Interchromosomal	0	2	0	2	unknown unknown
233	chrUn:105423-118779 + RN45S NR_046235 :0-13356	chr3:133543079-133614690 - RAB6B NM_016577 :0-5559	1	1	CLUSTER137	Interchromosomal	0	2	0	2	unknown unknown
185	chrUn:105423-118779 + RN45S NR_046235 :0-13356	chr11:10529433-10530722 - MTRNR2L8 NM_001190702 :0-1289	1	1	CLUSTER61	Interchromosomal	0	2	0	2	unknown unknown
		chr19:3052907-3061254									

Fig. 17 Detailed information and annotations for each gene fusion event

of differential analyses only starting from a completed main analysis, through the result sections, because DE is based on a statistical comparison of results obtained from two or more independents inputs.

A specific section of results (named Differential Expression) is devoted to configure and visualize differential expression operations. At first, this section only contains the list of input files, the user can request for a differential expression by selecting inputs to be included in the operation and assigning them to custom groups.

The user can select the type of differential expression operation by choosing between transcript level (based on Cuffdiff2), gene level (based on DESeq2) and both. Once configured the requested operation, the section is enriched by the differential expression request by reporting the list of selected files with corresponding assigned groups as well as the job execution status. After the operation completion, the *expand results* button allows to visualize results. This button opens an operation matrix reporting all analyzed pairs and a summary of configured parameters.

The differential usage of Poly(A) sites can be requested starting from the PolyA Sites section. The input selection and grouping strategy is the same already described above. Differential usage of polyadenylation sites is computed by a combination of custom scripts and DESeq2.

Acknowledgments

This work was supported by ELIXIR-IIB.

References

1. Emrich SJ, Barbazuk WB, Li L, Schnable PS (2007) Gene discovery and annotation using LCM-454 transcriptome sequencing. Genome Res 17:69–73
2. Nagalakshmi U et al (2008) The transcriptional landscape of the yeast genome defined by RNA sequencing. Science 320:1344–1350
3. Mortazavi A, Williams BA, Mccue K, Schaeffer L, Wold B (2008) Mapping and quantifying mammalian transcriptomes by RNA- Seq. Nat Methods 5:621–628
4. Wang Z, Gerstein M, Snyder M (2009) RNA-Seq: a revolutionary tool for transcriptomics. Nat Rev Genet 10(1):57–63
5. Wang ET et al (2008) Alternative isoform regulation in human tissue transcriptomes. Nature 456:470–476
6. Morris KV, Mattick JS (2014) The rise of regulatory RNA. Nat Rev Genet 15:423–437
7. Li W, Notani D, Rosenfeld MG (2016) Enhancers as non-coding RNA transcription units: recent insights and future perspectives. Nat Rev Genet 17:207–223
8. Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ et al (2015) Big data: astronomical or genomical? PLoS Biol 13(7): e1002195. https://doi.org/10.1371/journal.pbio.1002195
9. Leinonen R, Sugawara H, Shumway M, International Nucleotide Sequence Database Collaboration (2011) The sequence read archive. Nucleic Acids Res 39:D19–D21
10. TCGA, Tumor Fusion Gene Data Portal @ONLINE. http://54.8 4.12.177/PanCanFusV2//. Aug 2017
11. GTEx Consortium (2013) The Genotype-Tissue Expression (GTEx) project. Nat Genet 45(6):580–585. https://doi.org/10.1038/ng.2653
12. CCLE, Broad Institute portal—CCLE Repository. https://portals. broadinstitute.org/ccle/home. Oct 2016
13. Picardi E, Manzari C, Mastropasqua F, Aiello I, D'Erchia AM, Pesole G (2015) Profiling RNA editing in human tissues: towards the inosinome Atlas. Sci Rep 5:14941
14. Licht K, Kapoor U, Amman F et al (2019) A high resolution A-to-I editing map in the mouse identifies editing events controlled by pre-mRNA splicing. Genome Res 29 (9):1453–1463
15. Cirilli M, Flati T, Gioiosa S, Tagliaferri I, Ciacciulli A, Gao Z, Gattolin S, Geuna F, Maggi F, Bottoni P, Rossini L, Bassi D, Castrignanò T, Chillemi G (2018) PeachVar-DB: a curated collection of genetic variations for the interactive analysis of Peach Genome Data. Plant Cell Physiol 59:1–9. ISSN: 0032-0781

16. Gioiosa S, Bolis M, Flati T, Massini A, Garattini E, Chillemi G, Fratelli M, Castrignanò T (2018) Massive NGS data analysis reveals hundreds of potential novel gene fusions in human cell lines. GIGASCIENCE 7:1–8. ISSN: 2047-217X
17. Gatto A, Torroja-Fungairino C, Mazzarotto F, Cook SA, Barton PJ, Sanchez-Cabo F, Lara-Pezzi E (2014) FineSplice, enhanced splice junction detection and quantification: a novel pipeline based on the assessment of diverse RNASeq 17. alignment solutions. Nucleic Acids Res 42(8):e71
18. Kalari KR, Nair AA, Bhavsar JD, O'Brien DR, Davila JI, Bockol MA, Nie J, Tang X, Baheti S, Doughty JB et al (2014) MAP-RSeq: Mayo Analysis Pipeline for 18. RNA sequencing. BMC Bioinformatics 15(1):224
19. Boria I, Boatti L, Pesole G, Mignone F, Hong D, Rhie A, Park SS, Lee J, Ju YS, Kim S, Yu SB, Bleazard T, Park HS, Rhee H et al (2012) FX: an RNA-Seq analysis 19. Tool on the cloud. Bioinformatics 28(5):721–723
20. RNA Bioinformatics (ed) (2015) Editor Ernesto Picardi. "Exploring the RNA editing potential of RNA-seq data by ExpEdit". Mattia D'Antonio, Ernesto Picardi, Tiziana Castrignanò, Anna Maria D'Erchia, and Graziano Pesole. Methods Mol Biol 1269:365–378
21. Picardi E, D'Antonio M, Carrabino D, Castrignanò T, Pesole G (2011) ExpEdit: a web server to explore human RNA editing in RNA-Seq experiments. Bioinformatics 27 (9):1311–1312
22. D'Antonio M, D'Onorio De Meo P, Pallocca M, Picardi E, D'Erchia AM, Calogero R, Castrignanò T, Pesole G (2015) RAP: RNA-Seq Analysis Pipeline, a new cloud-based NGS web application. BMC Genomics 16:S3
23. FastQC: A quality control tool for high throughput sequence data. http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
24. Patel RK, Jain M (2012) NGS QC Toolkit: a toolkit for quality control of next generation sequencing data. PLoS One 7(2):e30619
25. Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, Salzberg SL (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. Genome Biol 14(4):R36. https://doi.org/10.1186/gb-2013-14-4-r36
26. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nat Methods 9 (4):357–359. https://doi.org/10.1038/nmeth.1923
27. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29(1):15–21. https://doi.org/10.1093/bioinformatics/bts635. Epub 2012 Oct 25
28. Roberts A, Pimentel H, Trapnell C, Pachter L (2011) Identification of novel transcripts in annotated genomes using RNA-Seq. Bioinformatics 27(17):2325–2329
29. Anders S, Huber W (2010) Differential expression analysis for sequence count data. Genome Biol 11(10):R106. https://doi.org/10.1186/gb-2010-11-10-r106
30. Iyer MK, Chinnaiyan AM, Maher CA (2011) ChimeraScan: a tool for identifying chimeric transcription in sequencing data. Bioinformatics. 27(20):2903–2904. https://doi.org/10.1093/bioinformatics/btr467
31. GFF/GTF file format—Definition and supported options. http://www.ensembl.org/info/website/upload/gff.html
32. Pruitt KD, Tatusova T, Maglott DR (2007) NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Res 35 (Database):D61–D65
33. Beaudoing E, Freier S, Wyatt JR, Claverie JM, Gautheret D (2000) Patterns of variant polyadenylation signal usage in human genes. Genome Res 10(7):1001–1010
34. Trapnell C, Hendrickson DG, Sauvageau M, Goff L, Rinn JL, Pachter L (2013) Differential analysis of gene regulation at transcript resolution with RNA-seq. Nat Biotechnol 31 (1):46–53. https://doi.org/10.1038/nbt.2450. Epub 2012 Dec 9
35. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate – a practical and powerful approach to multiple testing. J R Stat Soc B Met 57(1):289–300

Chapter 22

iDEP Web Application for RNA-Seq Data Analysis

Xijin Ge

Abstract

RNA sequencing (RNA-seq) has become a routine method for transcriptomic profiling. We developed a user-friendly web app called iDEP (integrated differential expression and pathway analysis) to help biologists interpret read counts or other types of expression matrices derived from read mapping. With iDEP, users can easily conduct exploratory data analysis, identify differentially expressed genes, and perform pathway analysis. Due to its intuitive user interface and massive annotation database, iDEP is being widely adopted for interactive analysis of RNA-seq data. Using a public dataset on the effect of heat shock on mouse with and without functional *Hsf1*, we demonstrate how users can prepare data files and conduct in-depth analysis. We also discuss the importance of critical interpretion of results (avoid p-hacking and rationalizing) and validation of significant pathways by using different methods and independent annotation databases.

Key words iDEP, Integrated differential expression and pathway analysis, RNA-seq, Bioconductor, Differential gene expression, Transcriptomic analysis, Pathway analysis, Gene set enrichment analysis, DESeq2

1 Introduction

As sequencing cost gets cheaper, the lack of access to experienced bioinformaticians becomes a major barrier for many biologists to take full advantages of the sequencing technologies. This barrier is especially severe in developing countries or smaller institutions with limited resources. Many high-quality command-line tools have been developed, but most are out of reach for busy biologists. To empower biologists across diverse fields, we created a user-friendly, interactive tool for in-depth, reproducible analysis of RNA-Seq data called iDEP (integrated differential expression and pathway analysis) [1]. It enables biologists to effortlessly conduct visualization, statistical analyses, and in-depth pathway analysis using read-counts and other gene-level expression data.

Prior to using iDEP, the raw reads need to be processed and mapped, as iDEP requires summarized expression data as input. Focusing on such data enabled us to leverage many existing R

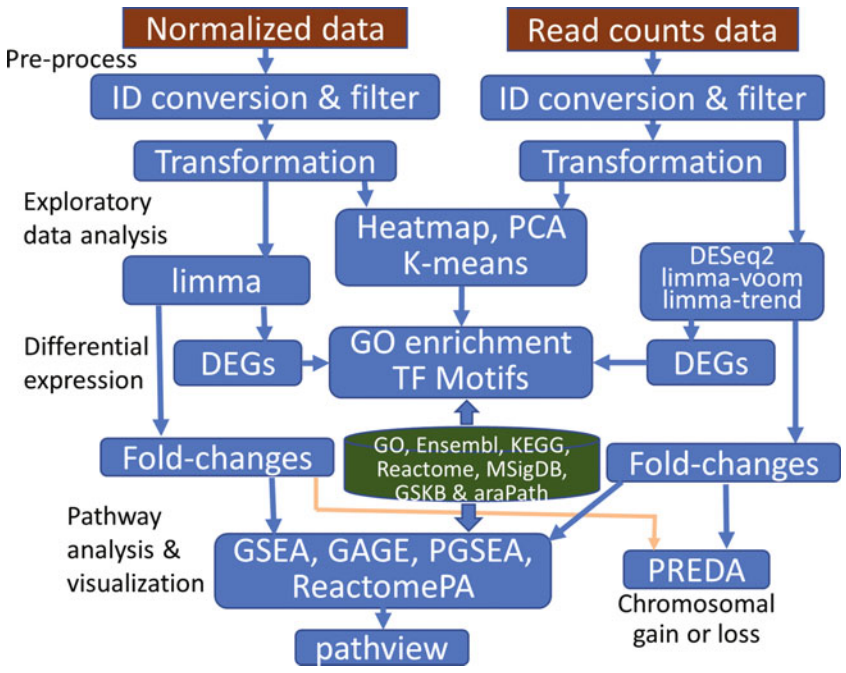


Fig. 1 Flowchart and functional modules of iDEP

packages and the power of the Rstuio Shiny platform to greatly simplify and streamline the bioinformatics workflow. Integrating over 60 R and Bioconductor packages with a massive annotation and pathway database for over 2000 species, iDEP provides functionalities for (1) high-quality static and interactive graphics, (2) common statistical analysis like preprocessing, data transformation, hierarchical clustering, *k*-Means clustering, principal component analysis (PCA), t-SNE, and other dimension reduction methods, (3) differential gene expression analysis (limma, DESeq2) and pathway analysis (GSEA, GAGE, PGSEA, ReactomePA), biclustering, and coexpression networks. The main workflow is illustrated in Fig. 1. Through an example, we demonstrate how this web application democratizes access to high-throughput technologies and help biologists pinpoint molecular pathways from large data.

2 Materials

This demonstration uses iDEP v0.91, accessed on May 3, 2020 at http://bioinformatics.sdstate.edu/idep/.

Data files used are available on Zenodo (https://doi.org/10.5281/zenodo.3783838).

A video tutorial is also available on YouTube (https://youtu.be/Hs5SamHHG9s).

3 Methods

3.1 Acquire and Prepare Input Files

iDEP provides access to a large compendium of published, uniformly processed RNA-seq data. As shown in Table 1, we have collected 5470 human and mouse RNA-seq data from ARCHS4 [2] database. We also provided an API interface to DEE2 [3], which can be used to access12,960 datasets from nine species (human, mouse, rat, zebrafish, worm, fly, *E. coli*, and Arabidopsis).

Users can start searching for public data by clicking on the "**Analyzing public RNA-seq datasets for 9 species**" button in the entry page of iDEP. By choosing **ARCHS4_mouse** and entering the keyword "heat shock," we identified several datasets. Selecting the row for GSE95602, we obtained metadata about this dataset. More information about this experiment is obtained by searching for GSE95602 in Google, which leads us to the NCBI's Gene Expression Omnibus (GEO) page (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE95602) with the associated publication [4]. The entire dataset contains expression data for three genotypes under four conditions. For heat shock treatment, the core body temperature of mice was raised to 41.5 °C for 15 min. RNA samples were harvested from muscle tissues and sequenced 4 h afterward. A subset of the data is used in our analysis to make a typical 2 × 2 design, namely two genotypes (wild type and Hsf1 knockout) with and without heat shock treatment.

Processed read count file is also available on the GEO page. We download the ARCHS4 dataset by clicking the **Gene-level counts**

Table 1
Processed public RNA-seq data currently accessible in iDEP

Source	Species	#Datasets	#Samples
ARCHS4	Human	2861	65,429
	Mouse	2609	72,363
DEE2	Arabidopsis	622	18,875
	E. coli	61	847
	Fly	507	11,804
	Human	4979	162,197
	Mouse	5682	198,903
	Rat	259	5744
	Worm	258	5868
	Yeast	302	6352
	Zebrafish	290	9130

Table 2
Renaming samples and constructing of design file

Original sample titles	Shortened	Genotype	Treatment
GSM2517903 Hsf1 −/− control heat shock rep1	Hsf1_ctrl_1	Hsf1	Ctrl
GSM2517904 Hsf1 −/− control heat shock rep2	Hsf1_ctrl_2	Hsf1	Ctrl
GSM2517905 Hsf1 −/− heat shock rep1	Hsf1_heat_1	Hsf1	Heat
GSM2517906 Hsf1 −/− heat shock rep2	Hsf1_heat_2	Hsf1	Heat
GSM2517911 wild type control heat shock rep1	wt_ctrl_1	wt	Ctrl
GSM2517912 wild type control heat shock rep2	wt_ctrl_2	wt	Ctrl
GSM2517913 wild type heat shock rep1	wt_heat_1	wt	Heat
GSM2517914 wild type heat shock rep2	wt_heat_2	wt	Heat

button, as it uses relatively new genome and annotation (GRCm38 with the GRCm38.88 annotation file) [4]. The downloaded CSV file is opened in **Microsoft Excel**. We delete the samples on R6/2 genotype and the HSP90 inhibition treatments. The columns names are then changed into a shorter, more informative names so that the plots are easily interpretable. It is often convenient to copy the column names and paste them transposed into a separate Excel file (Table 2), edit the column names, and then paste them back to the original file transposed.

After the expression data file is edited, we recommend users create an experimental design file. Such files are unnecessary for experiments with only one factor as iDEP automatically put samples into groups. Factorial design is widely adopted in expression analysis, even though many do not realize it when conducting bioinformatics analysis. As shown in Table 2, the two factors are clear in this experiment: treatment and genotype. The last three columns of Table 2 can be transposed and saved as a design file. Note that the sample names in the design file must match the sample names in the data matrix. For detailed explanation and example please consult the documentation of iDEP (https://idepsite.wordpress.com/data-format/).

Note that it is well-documented that Excel can automatically format some gene symbols into dates [5–7]. For example, "SEPT2," the symbol for Septin 2 gene, is converted to 2-Sept by default. In iDEP, we solved this issue by adding a space character in front of gene names. Users should be careful when gene symbols are used as the only identifier and edited files are saved for downstream analysis. One way of overcoming this is to use other stable gene IDs as the main identifiers. If we use the Ensembl genome sequences and annotations in the read mapping stage, we will obtain summarized expression data that identified by Ensembl gene IDs, which are natively supported by iDEP.

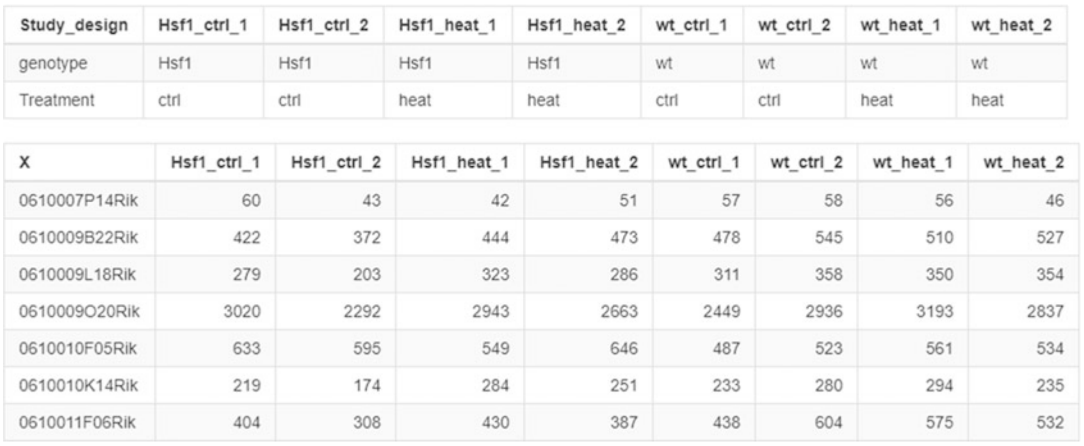
Table 3
Different types of RNA-seq data

Read counts	The numbers of reads mapped to a gene. The data matrix are integers. Outputs from alignment-free methods such as kallisto [26] could also be decimals
Counts per million (CPM)	Reads counts are scaled so that the total for each library are one million
RPKM	Reads Per Kilobase of transcript per million mapped reads. Read counts is normalized by library size and transcript length
FPKM	Fragments Per Kilobase of transcript per million mapped reads. The key difference is that a fragment is defined as two reads uniquely mapped to the same transcript when the library is generated by paired-end sequencing
Other normalized data	Data could be obtained using DNA microarray or proteomics. Note that data needs to be normalized first
Fold-change and *P* values	This could be several pairs of fold-change and *P* values derived from other methods such as cuffdiff. Users can conduct hierarchical clustering, *k*-means clustering and pathway analysis

3.2 Uploading the Input Files to iDEP

We start the analysis by first uploading the expression file. As iDEP can guess the species by mapping gene IDs, we leave the species drop-down menu as "**Best matching species**". Since this is read counts data, we use the default "**Read counts data (recommended)**" option. We recommend users upload read counts data whenever possible. iDEP also accepts normalized expression data in various formats (Table 3). We first browse and upload the expression file and then upload the experimental design file. The uploading process seems slow, because iDEP immediately applies the default filter to remove lowly expressed gene. All gene IDs passed the filter are then compares against all IDs known to iDEP, over 200 million in total, to help determine what the types of gene ID and species best match the uploaded identifiers. This is one of the many ways we make iDEP user friendly. The bottom left of the **Load Data** tab shows the species ranked by the numbers of matched genes in parentheses, such as "Mouse (13709)". This number is how many genes passed the filter and matched the database. As iDEP correctly guessed mouse as the organism, we can move on to the next step. Otherwise, users can manually choose their organisms. The design matrix and the top 20 rows of the expression matrix are also printed out for users to double check (Fig. 2).

Errors encountered in this step often arise due to mismatched sample names between the design matrix and the expression matrix. We recommend users to first copy-and-paste columns names from the read counts file into Excel and construct a file like Table 2. Then the shortened file names can be copy-and-pasted back to both the expression matrix and the design file. Another possible error is that



Study_design	Hsf1_ctrl_1	Hsf1_ctrl_2	Hsf1_heat_1	Hsf1_heat_2	wt_ctrl_1	wt_ctrl_2	wt_heat_1	wt_heat_2
genotype	Hsf1	Hsf1	Hsf1	Hsf1	wt	wt	wt	wt
Treatment	ctrl	ctrl	heat	heat	ctrl	ctrl	heat	heat

X	Hsf1_ctrl_1	Hsf1_ctrl_2	Hsf1_heat_1	Hsf1_heat_2	wt_ctrl_1	wt_ctrl_2	wt_heat_1	wt_heat_2
0610007P14Rik	60	43	42	51	57	58	56	46
0610009B22Rik	422	372	444	473	478	545	510	527
0610009L18Rik	279	203	323	286	311	358	350	354
0610009O20Rik	3020	2292	2943	2663	2449	2936	3193	2837
0610010F05Rik	633	595	549	646	487	523	561	534
0610010K14Rik	219	174	284	251	233	280	294	235
0610011F06Rik	404	308	430	387	438	604	575	532

Fig. 2 Uploaded files are shown in iDEP for confirmation

multiple gene ID columns. iDEP only allows the first column to contain gene IDs. The rest of the columns are treated as expression data.

3.3 Preprocessing

The main challenge in analyzing transcriptomic data is to overcome the inherent noise at the low end of the detection limit. The function of a type of tissue often requires a subset of all proteins encoded in the genome. Thus, in a typical RNA-seq dataset, as much as 20–50% of the annotated genes in the genome are nearly undetectable with close to zero read counts across all samples. Removing these genes will increase the signal-to-noise ratio in downstream analysis. In the preprocessing step, we filter out genes that are expressed at very low levels across all samples. The default is that we only keep genes expressed at least 0.5 counts per million (CPM) in at least one library. *See* Table 4 for detailed explanation of all the parameters in iDEP. In our dataset, this criterion means that we remove genes expressed less than 10 reads in all samples, as the libraries have around 20 million reads each. In addition, we can increase the **n libraries** cutoff to require a certain level be reached in two or more samples. With the default filter, only 13,845 (42%) of the 32,544 genes passed the filter. We decrease the **Min. CPM** filter to 0.2 and got 15,611 genes. Among these, 13,501 is converted to Ensembl IDs, a central gene ID used in the annotation and pathway databases by iDEP. The 2110 unrecognized genes remain in the dataset using original IDs, and will be used for clustering and DEG analysis, but not in enrichment analysis. Unconverted gene IDs could be due to mismatches in genome versions and different annotation files used in the processing of raw reads. We recommend the use of reference genome sequences and annotation files from Ensembl to ensure that most of the genes be recognized by iDEP.

Table 4
Parameters in iDEP

Parameter	Default	Options and reasons for adjustment in this dataset
Preprocessing		
Min. CPM	0.5	A gene needs to have 0.5 counts per million (CPM) in one or more (specified by the **n libraries** below) samples. Otherwise removed from further analysis. In our dataset, this eliminates 58% of genes, so it is reduced to 0.2
n libraries	1	Sometimes we require a minimal level be detected in two or more genes. This is used together with the Min. CPM to weed out lowly expressed genes
Transformation	EdgeR	**EdgeR: log2(CPM + c)**. Read count are first normalized to CPM, and then a pseudo count is added to all counts before log-transformation. This is set as default because it is faster and generally does a good job **rlog**: The regularized log in the DESeq2 package. This is a more robust method but is slow. Here we used rlog as our data is small **VST**: Variance stabilizing transformation described in the DESeq2 package
Heatmap		
Most variable genes	1000	Genes are ranked by standard deviation(SD) across all samples. The top 1000 is used in heatmap. Based on the "Gene SD distribution," we increase it to 2000, as more genes are differentially expressed
Distance	Correlation	**Correlation**: 1 − PCC, where PCC is the Pearson's correlation coefficient. **Euclidian:** Euclidian distance. **AbsolutePCC:** 1 − |PCC|. The absolute values are used to treat positive and negative correlation the same
Linkage	Average	Hierarchical clustering trees can be built with many methods such as average, single, complete, median, centroid, and Mcquitty [27]. For most cases, we recommend the default average linkage
Cutoff *Z* score	4	The heatmap.2 function uses the largest and smallest numbers in the data matrix to determine a range for color mapping. To avoid the situation where a few extreme values skew the color range, values outside 4 SDs from the median is trimmed. In this example data set, we set this to 3. Smaller cutoff will visually enhance the contrast
Center genes	TRUE	Mean is subtracted from each gene before hierarchical clustering

(continued)

Table 4
(continued)

Parameter	Default	Options and reasons for adjustment in this dataset
Normalize genes	FALSE	After the mean centering, we could further normalize the data by dividing each row with its SD. This will make the variation uniform across genes. This is rarely the case in real expression data but can help scale the genes so that we focus on the pattern
Center samples	FALSE	The mean is subtracted from each column (sample)
Normalize samples	FALSE	Each column is divided by its SD, so all columns have SD of 1. This is typically done after the centering to standardize the columns
k-Means clustering		
Normalization	Mean center	**Mean center**: Genes are centered by subtracting the mean. **Standardization**: Genes are mean centered and then divided by its SD so that all genes have mean of zero and SD of one. **L1 normalization**: Each row in the matrix is divided by the sum of the absolute values of each of values
Remove redundant genesets	Yes	When showing enrichment results, if two gene sets overlap more than 90% of the genes, then only the gene set with the most significant FDR values are shown
Rerun		As a heuristic algorithm, *k*-means initiates randomly and updates cluster memberships iteratively. Thus, different random initiation can lead to different clusters
DEG1		
Method	DESeq2	We recommend users upload raw read counts and use the DESeq2 method to identify DEGs. In addition to the power of DESeq2 package, we also have tested it more thoroughly. Methods like voom and limma-trend [28] are also have great performance. When the number of total reads vary greatly between libraries limma-voom have better performance than limma-trend. When users upload normalized expression data, limma is used by default and this box is hidden
FDR cutoff	0.1	This is the cutoff for false discovery rate (FDR) for DEGs. If there are too many (several thousands) or too few (only several dozens) of DEGs, users can adjust this cutoff
Fold-change	2	Genes must have more than twofold increase or decrease. This cutoff needs to be adjusted based on the magnitude of expression change in a specific experiment
Pathway analysis		

(continued)

Table 4
(continued)

Parameter	Default	Options and reasons for adjustment in this dataset
Methods	GAGE	GAGE (generally applicable gene set enrichment) [22] is a parametric method for pathway analysis that conducts *t*-tests and meta-tests on fold-changes. It is default method as it is fast. For the popular GSEA (gene set enrichment analysis) [21], we adopted a novel implementation provided by the fgsea package [29] to analyze genes preranked by the fold-change. The standalone GSEA programs are more powerful and flexible as it can analyze normalized expression values through permutations. Based on the parametric analysis of gene set enrichment (PAGE) algorithm [14], the PGSEA package [13] can be used to visualize pathway that are differently activated among sample groups
Select a comparison		If there are more than one comparison, a list of comparisons will be displayed
Geneset size: Min.	15	After merging the annotation database with fold-change, if a pathway has less than 15 genes, this pathway is filtered out. This can be reduced to as few as 5, if some pathway of interest only has a small number of genes. But smaller genesets introduce noise
Geneset size: Max.	2000	If a pathway contains too many genes, it is not used in pathway analysis
Pathway significance cutoff (FDR)	0.2	This significance level can be adjusted. Note that only a limited number of pathways are displayed depending on the choice below
Number of top pathways to show	30	Pathways are filtered first by FDR and then ranked by the test statistic
Use absolute values of fold-change	FALSE	When enabled, this converts original fold-changes to the absolute values of fold-changes. Essentially, we treat the up- and downregulations as the same. By ignoring the directionality of expression change, we can identify most affected pathways. Some pathways could be regulated by upregulating some while downregulating other genes in the pathway
Remove genes with big FDR before pathway analysis:	1	Before conducting pathway analysis, users can eliminate some genes with very large FDRs (e.g., >0.5) in the differential expression analysis, thus increasing the signal to noise ratio. This value should not be too small like 0.05, as it will drastically reduce the number of genes

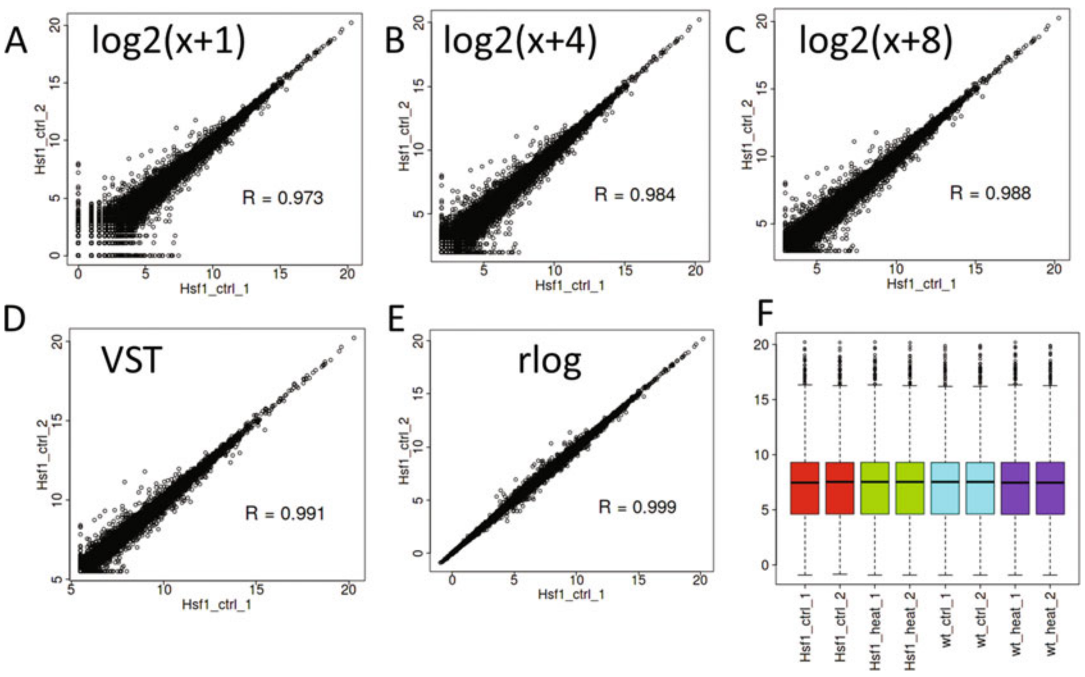


Fig. 3 Effects of various transformations on data distribution

We then apply transformations so that lowly expressed genes do not introduce as much noise in exploratory data analysis (EDA, *see* Subheadings 3.4–3.6). As shown in Fig. 3, various methods have different effects in reducing the variability of expression scores. The rlog transformation is the most aggressive but is slower. Since we only have eight samples, we choose **rlog**. You can see the effects of the transformation by observing the scatter plots and boxplots. We want to make sure that we have similar distribution in the density plot and boxplot across samples. Note that for read counts data transformations only affect EDA, as the identification of differentially expressed genes (DEGs) is based on the modeling of the read counts directly using discrete distributions. If users uploaded normalized expression data, the transformed data is used in all subsequent analyses (*see* Fig. 1).

3.4 Hierarchical Clustering Heat Map

EDA is an essential part of the bioinformatics analysis of RNA-seq data. Even though not directly related to the identification of DEGs, EDA enables researchers to examine, observe, and interact with the data. This could help reveal potential issues in the dataset. Potential errors could arise from many of the previous steps: the original biological experiment, RNA extraction, removal of ribosomal RNA, library construction, sequencing, adapter removal, alignment, and quantification. We should always be skeptical, especially when some of the steps are outsourced. Sample mislabeling happens in as much as 4% of the RNA-seq datasets [8] and can have devastating effect on a study. In our own experience, we once accidently found that a purchased RNA sample for normal mouse testis shows the typical expression profiles of uterus, after a graduate student had worked on the data for over a year. Another benefit of EDA is that we can gain insights into the trend in expression change. Intuitive understanding of the data forms a foundation for downstream analyses. To encourage biologist to conduct EDA, we put several methods in front of the DEG and pathway analysis tabs.

As one of the most direct and effective methods for EDA, heatmap and hierarchical clustering can be customized in many ways in iDEP. One trivial adjustment is to make the **browser window narrower**, which forces iDEP to rerender the heatmap to achieve a desired aspect ratio. The only option we changed for this dataset is the **Cut-off Z score** from the default value of 4 to 3. This enhances the contrast of the heatmap, as values deviating from the median by more than 3 standard deviations are replaced with these upper or lower bounds.

As shown in Fig. 4, heatmaps give us a big picture of the overall patterns in gene expression. Using 2000 genes with the most variation across all samples, we observe that the two genotypes differ substantially in the expression profiles of many genes. Replicates are grouped together as expected. We manually marked some of the gene groups (Fig. 4). Genes in Groups 1 and 3 are up- and downregulated genes by Hsf1 knockout compared to wild type, respectively. While genes in Group 2 are upregulated by heat shock in both genotypes, those in Group 4 are induced only in wild type. With tools like iDEP, users can obtain several dozen plots quickly, it is essential to carefully examine and critically interpret these results. Starting with the heatmap, users should begin to obtain intuitive impression of the overall quality of the data, and the trend of gene expression changes.

Data normalization before clustering can be done at different levels. The least aggressive is the mean centering by genes, which is often recommended for most expression studies. Mean centering enables us to observe relative fold-changes as the most highly upregulated and downregulated genes will be marked by bright red and green in the default color scheme. On top of this centering we can further divided each row by its standard deviation (SD). Sometimes referred to as standardization, each row now has mean of zero and SD of one. This can be useful in visualizing the trend of changes, but the drawback is that we lost the information on the absolute fold-changes and can amplify noise if genes with minimal changes are included. Similarly, centering and standardization can be applied to the columns (samples).

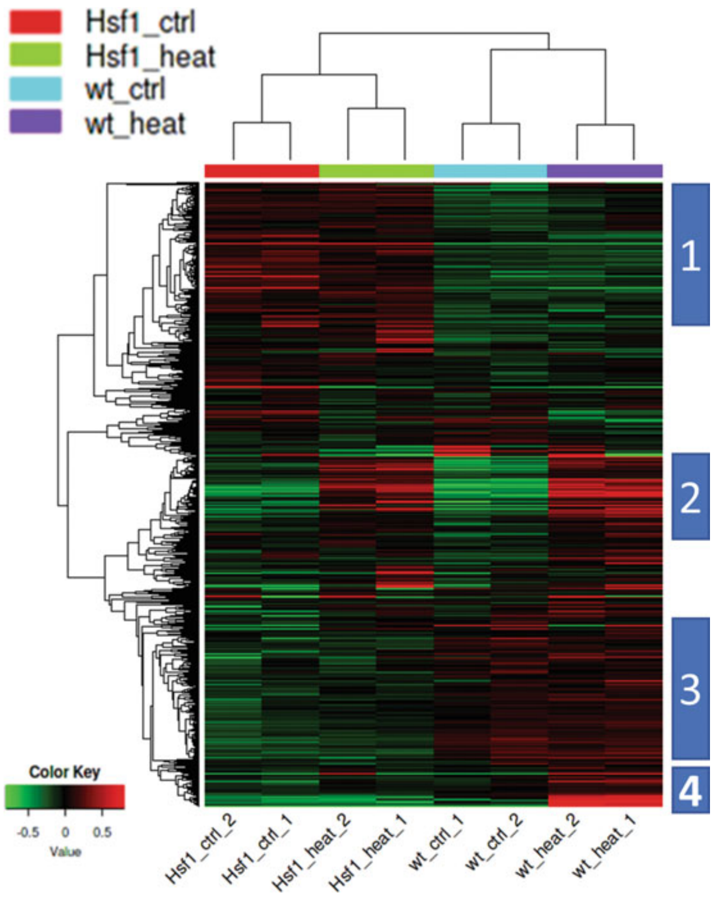


Fig. 4 Hierarchical clustering of the top 2000 genes reveals patterns of gene expression. Several groups of genes are marked

Some experiments induce drastic changes in the transcription of thousands of genes. For example, the heat shock experiment in this example. Other experiments, however, has minimal effects on most genes. Users should respect whatever the data is telling them and select an appropriate number of genes with expression change. The **Gene SD distribution** chart can guide the user in deciding how many genes should be included in hierarchical clustering. We typically examine a few hundred to a few thousand genes with the most variations in expression characterized by SD.

3.5 k-Means Clustering

With *k*-Means clustering, we can explicitly divide genes into groups and carry out enrichment analysis to gain insights on what types of genes are responding to treatment. First, users can take a quick look at the **Gene SD distribution** to determine how many genes should be included for clustering, like what we did in the Heatmap tab. To decide on a proper *k*, the number of clusters, users can look at the plot of within-group sum of squares (WSS) by clicking on the

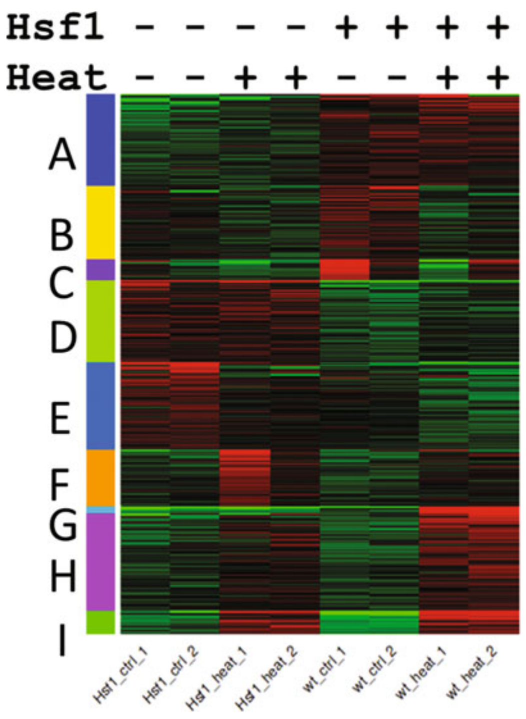


Fig. 5 With *k*-means clustering, the top 2000 genes are divided into nine groups with distinct expression patterns

button labeled "**How many clusters**?" We should choose a number where an additional cluster does not lead to substantially decrease in WSS. This is often subjective and for this dataset, $k = 9$ seems appropriate. Users are encouraged to try different choices. In general, if the dataset contains more biological samples, we need a larger k, which will give us more refined grouping. When we start to see many small clusters with no coherent enrichment results, we need to decrease k.

In this dataset, we settled on $k = 9$ (Fig. 5). Genes in clusters A and D are up- and downregulated by Hsf1 knockout, respectively. These differences are not responsive to the heat shock treatment. Induced by heat shock in both genotypes, cluster I genes are related to response to stress, based on the enrichment results that were conducted automatically. Genes in cluster H are more strongly upregulated in the wide type animals and are also related to cellular response to stimulus. Cluster G contains a small group of 24 genes, of which 12 are related to protein folding, including many heat shock proteins (HSPs) involved in Hsf1 mediated heat response. The names of the genes in each cluster can be obtained by downloading the file by clicking the **Enrichment details** button.

Users should avoid over-interpret results to fit a "story". In addition to Gene Ontology (GO), there are many other types of pathway databases. Many of the enriched terms with moderately significant P values like 10^{-3} are often noise. We recommend that users focus on the most significantly enriched terms. Based on the hypergeometric test, the most significant enrichment is fatty acid metabolic process with false discovery rate (FDR) of 3.9×10^{-18} in cluster C. This is unexpected and this group of genes are highly expressed in wild-type animals regardless of treatment. Searching the literature on the link between Hsf1 and fatty acid metabolism, we found the work of Jin et al. [9], who demonstrated that loss of Hsf1 in mice results in the strong downregulation of adipose-specific genes like PPARG, which is included in Cluster C. Therefore, this result is consistent with the observation that Hsf1 is essential for lipid metabolism. The second most significantly enriched term is protein folding ($\text{FDR} < 1.17 \times 10^{-17}$) in cluster G. As much as 12 (50%) of the 24 genes in this cluster are related to this function. Eight are clearly HSPs based on their names. These genes are induced by heat shock only in wild-type animals, suggesting that their induction is dependent on Hsf1. The Hsf1-independent responses in Cluster H are enriched with response to chemical stimulus ($\text{FDR} < 4.0 \times 10^{-12}$). The fourth most significant group of GO Biological Process (GOBP) terms are related to development, especially circulatory system development ($\text{FDR} < 4.7 \times 10^{-10}$), overrepresented in Cluster E. There are strong evidence that Hsf1 plays a role in development [10]. A hierarchical clustering tree of enriched pathways can be generated with the **Visualize enrichment** button. By interacting with the data, users can start to understand the data and form hypothesis on the molecular pathways.

3.6 Principal Component Analysis (PCA)

PCA is a dimensionality reduction method that enables the mapping of RNA-seq samples on a scatter plot that captures overall variability in expression profiles. From Fig. 6, we can see that technical replicates are plotted close to each other as expected. While the four knockout samples are located on the top left, the wild-type samples are on the lower right. Other methods such as multidimensional scaling (MDS) and t-SNE are nonlinear and maybe more useful for complex datasets.

The principal components (PCs) represents directional changes in gene expression and can be meaningful. As shown in Fig. 6, each of the samples has coordinates on each of PCs. We can conduct an ANOVA on these projections to see if each of PCs are correlated with the factors in the design matrix. If correlation is detected, P values will be printed out. We do not detect significant ($P < 0.05$) correlation for this dataset. But for the demo data built in iDEP [11] which studies the effect of whole-body ionizing radiation (IR) on the mouse with or without p53, the first PC is correlated with ionization radiation. In addition, each of the PCs has loadings on each of the genes. Called metagenes by some researchers [12], these PCs can be correlated with a certain

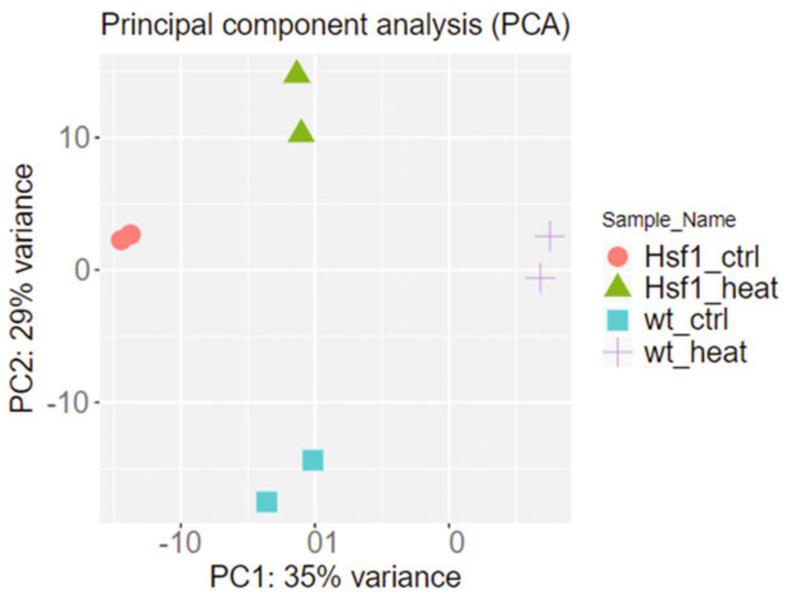


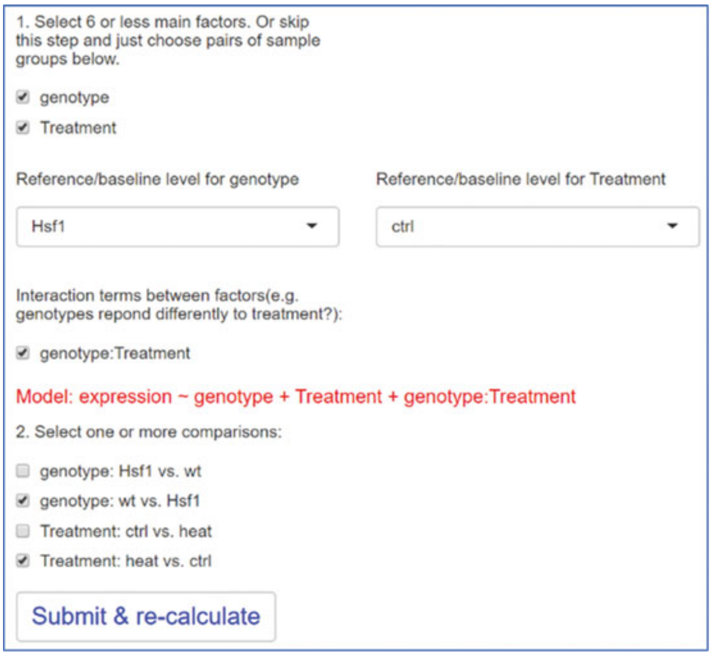
Fig. 6 Principal component analysis (PCA) of the eight samples

pathways. We can treat these loadings vectors as fold-changes and conduct pathway analysis using PGSEA [13, 14]. For this dataset, we can see that the first two PCs are related to mitochondrion organization and cell migration, respectively. The fourth and fifth PCs are associated with fatty acid metabolism and immune response, respectively. This is one of the innovations iDEP introduced to combine PCA with pathway analysis.

3.7 Differentially Expressed Genes (DEGs)

We recommend users to upload read counts instead of FPKM or RPKM data, which loses information through normalization. Read counts will enable us to take advantage of the power of DESeq2 package [15] to directly model discrete distributions. In addition, iDEP has been tested more thoroughly for read count data. By default, the DEG1 tab conducts an all-vs-all comparison for all sample groups. For simple experiments, this might be enough. In this case we have an 2 × 2 factorial design, so we need to define our model by clicking on the **Select factors & comparisons** button.

For this experiment, we choose both the treatment and genotype as factors (Fig. 7). To make the results easily interpretable, *it is imperative to set reference (base) levels for each of the factors.* By default, it is set alphabetically. For the treatment, we select "ctrl" as the reference level. For genotype, we choose the Hsf1 knockout as reference as it does not have a functional HSF1 protein. This will enable us to conveniently observe the additional effect of introducing Hsf1 in the wild type during heat stress. We also select the interaction term as we are interested in the differential response between the genotypes.



1. Select 6 or less main factors. Or skip this step and just choose pairs of sample groups below.

☑ genotype
☑ Treatment

Reference/baseline level for genotype: Hsf1

Reference/baseline level for Treatment: ctrl

Interaction terms between factors(e.g. genotypes repond differently to treatment?):

☑ genotype:Treatment

Model: expression ~ genotype + Treatment + genotype:Treatment

2. Select one or more comparisons:

☐ genotype: Hsf1 vs. wt
☑ genotype: wt vs. Hsf1
☐ Treatment: ctrl vs. heat
☑ Treatment: heat vs. ctrl

Submit & re-calculate

Fig. 7 Interface to select factors and make models

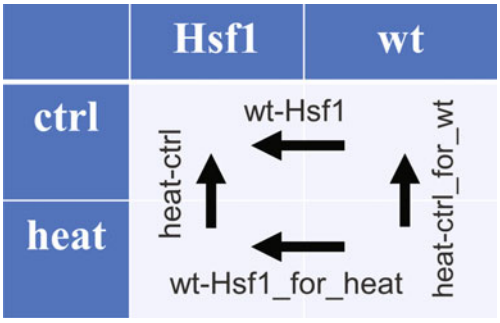


Fig. 8 How comparisons are coded in iDEP. It is important to set reference levels for factors. The reference levels are omitted in the names of the comparisons. For example, "wt-Hsf1" means wt vs. Hsf1 under "ctrl" condition. A positive fold-change number will mean genes highly expressed in wild type compared with Hsf1 knockout under control condition. "wt-Hsf1_for_heat" means the difference of gene expression in wild type compared with Hsf1 knockout both with heat treatment. If a gene has a positive fold-change, it is more highly induced by heat in wild type than in Hsf1 knockout

After setting up the statistical model, we have the option to decide the direction of the comparisons. For genotype, we choose to compare wild type vs. Hsf1 mutant, while for treatment we are interested in heat treatment vs. control (Fig. 7). This is to ensure that we can easily interpret the fold-change values. Figure 8 illustrates how all the four possible comparisons are named and

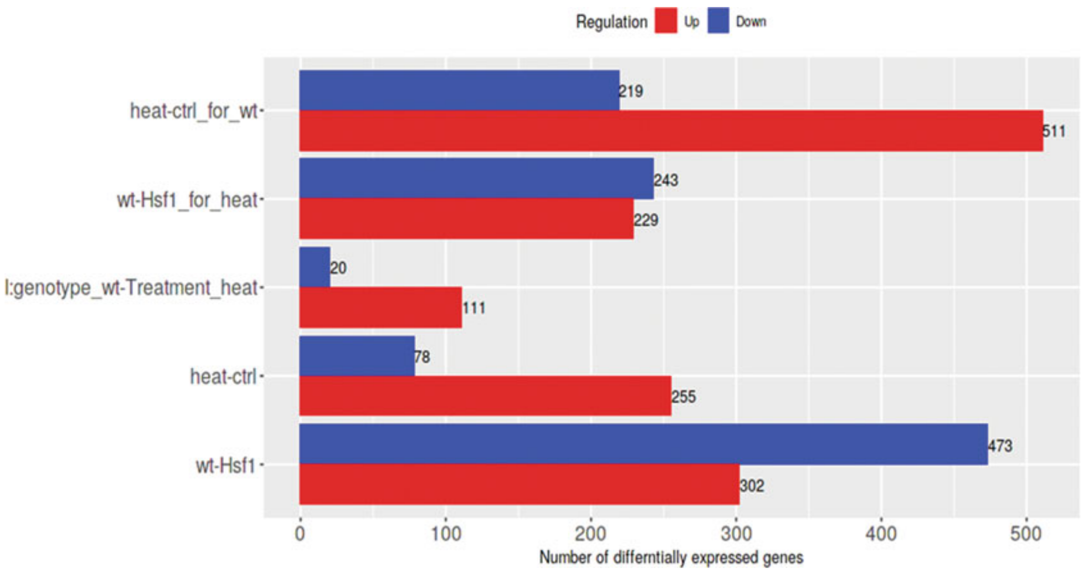


Fig. 9 The numbers of DEGs for all comparisons

interpreted across the four biological samples. Once we submit the models, DESeq2 is rerun and a table and a bar plot of the number of DEGs are shown for each comparison. Here, since the numbers of DEGs are small, we decrease the **Fold-change cutoff** from 2 to 1.5. The FDR cutoff is set at the default of 0.1. As shown in Fig. 9, heat shock induces 511 genes in the wild type, much more than the 255 in Hsf1 knockout. Also, under control conditions, Hsf1 knockout downregulated 473 genes, suggesting that Hsf1 is essential in normal physiological conditions outside heat shock. Associated with the interaction term, 111 genes are specifically upregulated by heat stress in wild-type animals, constituting an Hsf1-dependent response.

Using the **Venn Diagram** button, we can compare these lists of DEGs. For example, Fig. 10a shows that 186 genes are upregulated in both genotypes, but heat shock induces more genes in wild type than those in Hsf1 knockout. While Hsf1 plays an essential role in heat shock response, there are Hsf1-independent responses. The 186 genes are part of a Hsf1-independent response.

The **DEG2** tab enables detailed examination of the individual comparisons. For each of the comparison, iDEP can produce static and interactive versions of volcano plots, MA plots, and scatter plots. We start with the response to heat stress in wild-type animals by choosing the "**heat-ctrl_for_wt**" comparison. The scatter plot in Fig. 10b indicates the drastic upregulation of Hspa1b. Figure 11a shows the enriched GOBP terms arranged into groups based on overlapping genes. Upregulated genes are divided into several themes, for example, stimulus response, development, and metabolisms. For model organisms, iDEP provides a large database

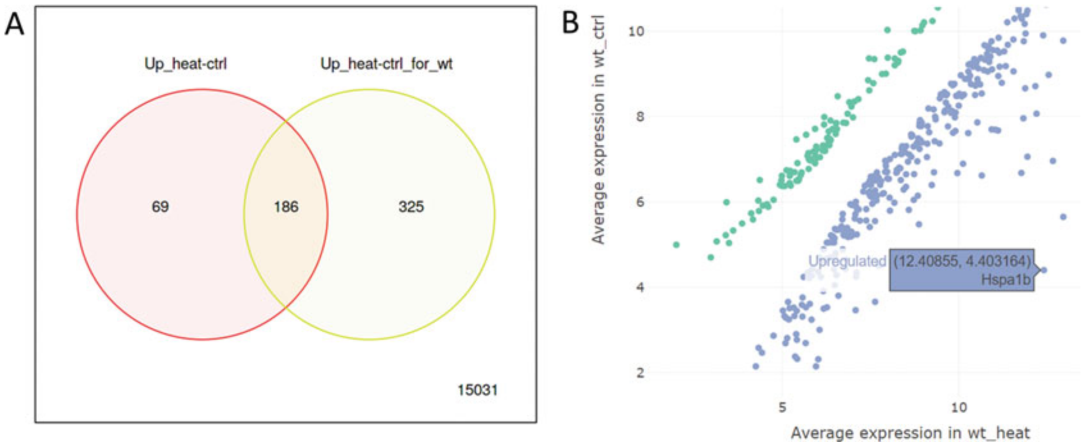
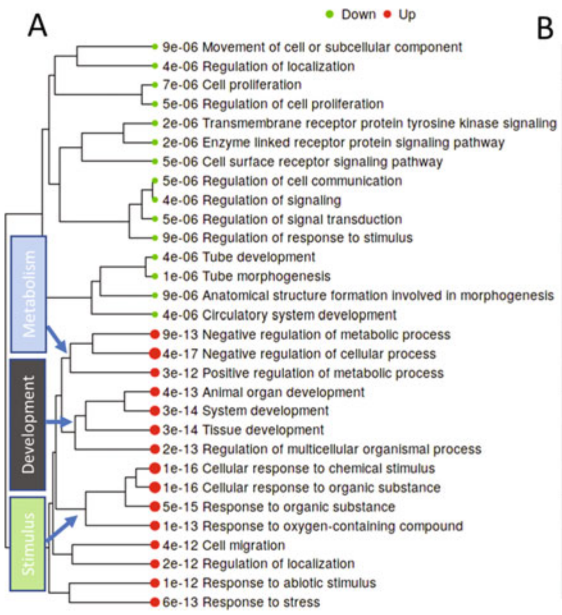
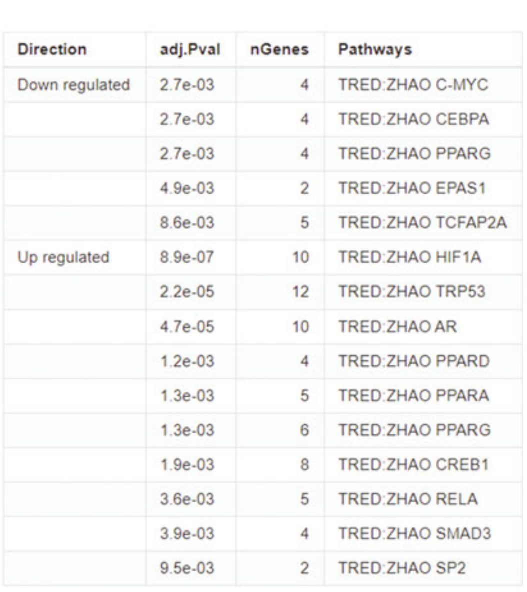


Fig. 10 (**a**) Venn diagrams comparing genes induced by heat in wild type and Hsf1 knockout. (**b**) Interactive scatter plot comparing the average expression of DEGs in treated vs untreated wild-type animals

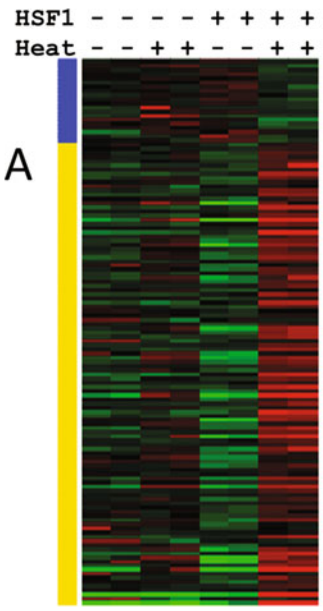

A
● Down ● Up
9e-06 Movement of cell or subcellular component
4e-06 Regulation of localization
7e-06 Cell proliferation
5e-06 Regulation of cell proliferation
2e-06 Transmembrane receptor protein tyrosine kinase signaling
2e-06 Enzyme linked receptor protein signaling pathway
5e-06 Cell surface receptor signaling pathway
5e-06 Regulation of cell communication
4e-06 Regulation of signaling
5e-06 Regulation of signal transduction
9e-06 Regulation of response to stimulus
4e-06 Tube development
1e-06 Tube morphogenesis
9e-06 Anatomical structure formation involved in morphogenesis
4e-06 Circulatory system development
9e-13 Negative regulation of metabolic process
4e-17 Negative regulation of cellular process
3e-12 Positive regulation of metabolic process
4e-13 Animal organ development
3e-14 System development
3e-14 Tissue development
2e-13 Regulation of multicellular organismal process
1e-16 Cellular response to chemical stimulus
1e-16 Cellular response to organic substance
5e-15 Response to organic substance
1e-13 Response to oxygen-containing compound
4e-12 Cell migration
2e-12 Regulation of localization
1e-12 Response to abiotic stimulus
6e-13 Response to stress
Metabolism
Development
Stimulus

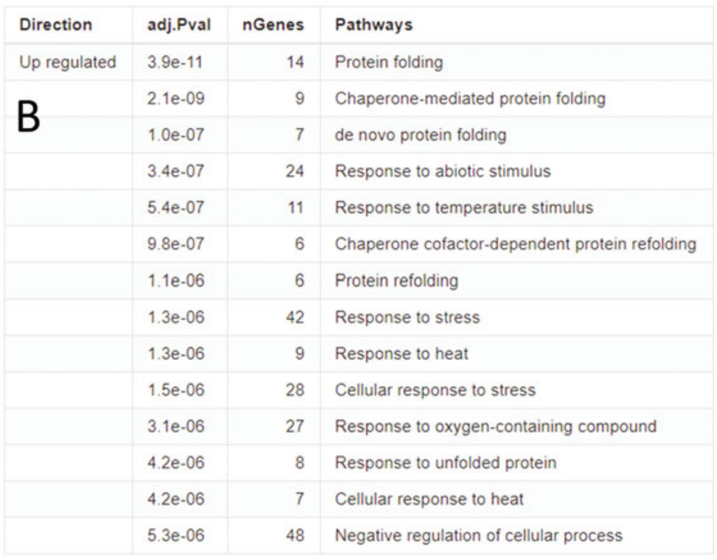

B

Direction	adj.Pval	nGenes	Pathways
Down regulated	2.7e-03	4	TRED:ZHAO C-MYC
	2.7e-03	4	TRED:ZHAO CEBPA
	2.7e-03	4	TRED:ZHAO PPARG
	4.9e-03	2	TRED:ZHAO EPAS1
	8.6e-03	5	TRED:ZHAO TCFAP2A
Up regulated	8.9e-07	10	TRED:ZHAO HIF1A
	2.2e-05	12	TRED:ZHAO TRP53
	4.7e-05	10	TRED:ZHAO AR
	1.2e-03	4	TRED:ZHAO PPARD
	1.3e-03	5	TRED:ZHAO PPARA
	1.3e-03	6	TRED:ZHAO PPARG
	1.9e-03	8	TRED:ZHAO CREB1
	3.6e-03	5	TRED:ZHAO RELA
	3.9e-03	4	TRED:ZHAO SMAD3
	9.5e-03	2	TRED:ZHAO SP2

Fig. 11 Overrepresented GO Biological Process terms (**a**) and transcription factor target genes (**b**) in treatment-induced DEGs in wild-type animals

of pathways including GO, KEGG, many specialized pathway databases, transcription factor (TF) and microRNA target genes, and drug associations. Switching the database from GOBP to TRED (TF.Target.TRED) [16], we get Fig. 11b, which suggests a significant overrepresentation of 10 HIF1 target genes in this list (FDR $< 3.9 \times 10^{-18}$). The upregulated genes also include 12 target genes of p53, which is part of the stress response, perhaps




B

Direction	adj.Pval	nGenes	Pathways
Up regulated	3.9e-11	14	Protein folding
	2.1e-09	9	Chaperone-mediated protein folding
	1.0e-07	7	de novo protein folding
	3.4e-07	24	Response to abiotic stimulus
	5.4e-07	11	Response to temperature stimulus
	9.8e-07	6	Chaperone cofactor-dependent protein refolding
	1.1e-06	6	Protein refolding
	1.3e-06	42	Response to stress
	1.3e-06	9	Response to heat
	1.5e-06	28	Cellular response to stress
	3.1e-06	27	Response to oxygen-containing compound
	4.2e-06	8	Response to unfolded protein
	4.2e-06	7	Cellular response to heat
	5.3e-06	48	Negative regulation of cellular process

Fig. 12 (**a**) The expression profiles of DEGs associated with the interaction term contains genes specifically up- or downregulated with functional HSF1. (**b**) These genes are related to protein folding

independent of HSF1. Other factors include androgen receptor (AR) and several regulators of lipid metabolism (PPARA, PPARD and PPARG).

Similar analysis can be done on the "heat-ctrl" comparison, representing the response to heat in knockout animals. The GOBP terms also contain the stress response, but also includes cytokines and programmed cell death (data not shown). The upregulated genes are enriched with target genes of RELA, ATF1, and TP53. As RELA is part of the NF-κB, a strong inflammatory response is triggered by heat shock in animals without the Hsf1. There is a complex interplay between HSF1 an p53 pathways [17]. Our results suggest that p53 might be part of the heat response independent of Hsf1.

We then can examine DEGs linked to the interaction terms, which represent the additional heat response in wild-type animals. The expression patterns of these genes are shown in Fig. 12a, which indicates a group of genes only upregulated by heat treatment in wild type. The top two most upregulated (over 100×) genes are Hspa1b and Hspa1a, two members of the Hsp70 family. The most significantly downregulated gene is Mmrn1 (Multimerin 1). These groups of genes are clearly related to protein folding and heat response based on overrepresented GOBP terms (Fig. 12b). Many other databases can be explored to gain further insights. For example, using Comparative Toxicogenomics Database (CTD) [18], we found that the interaction term is highly correlated (FDR $< 5.7 \times 10^{-10}$) with gedunin, which is a tetranortriterpenoid

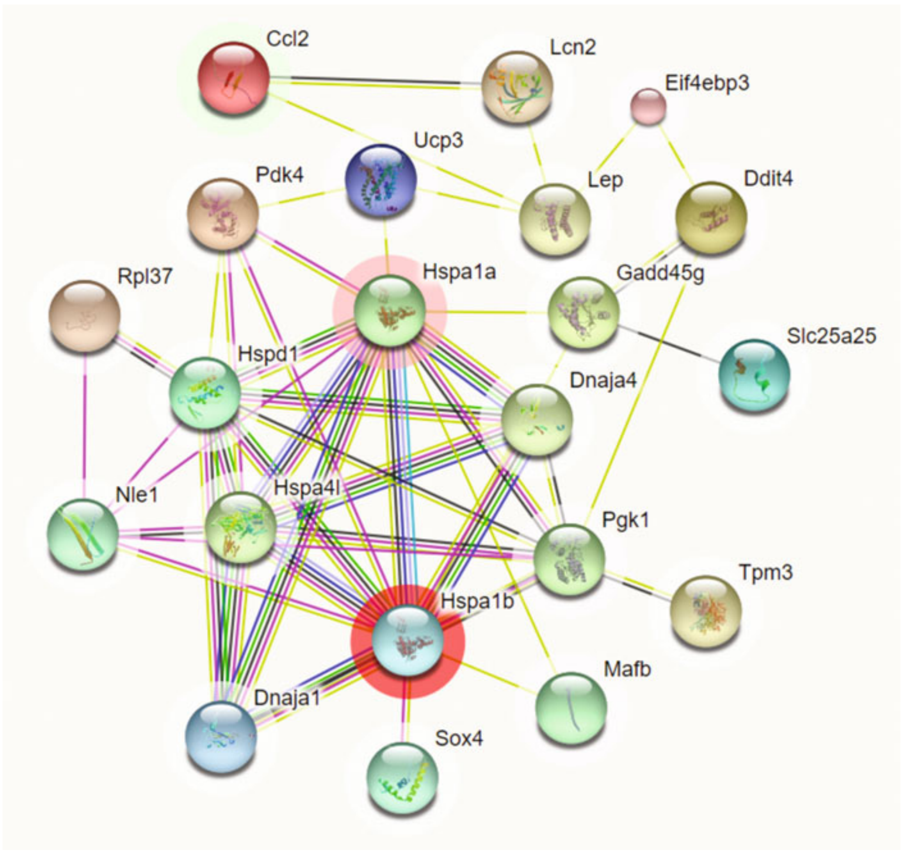


Fig. 13 Protein–protein interactions of the top 30 genes associated with the interaction term. This is rendered by API access to STRING-db

isolated from the Indian neem tree. Used to treat malaria and shown to have anticancer activity, gedumin modulates Hsp90 [19] to activate the heat-shock response. Based on the data here, the effect of gedumin is dependent on Hsf1. Therefore, for human and mouse, it is possible to identify drugs associated with expression signatures.

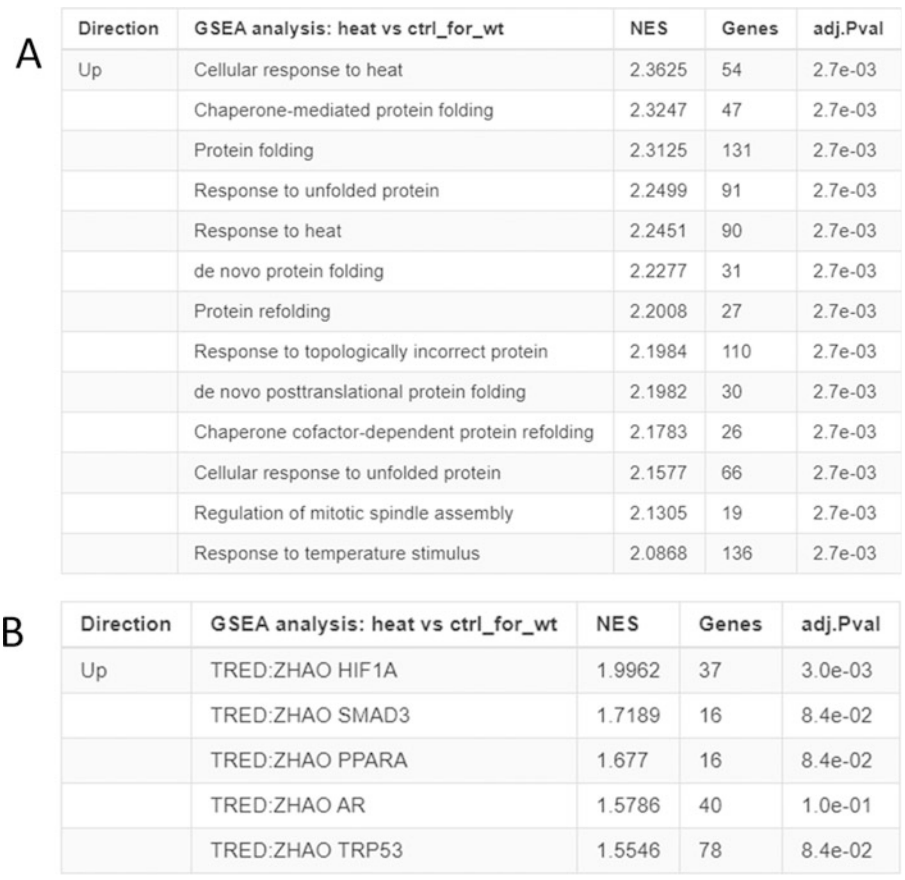
Users can employ the API (application program interface) access to STRING-db web server [20] to double-check the enrichment results. The results are based on analysis by the STRINB-db server, independent of iDEP. STRING-db has the additional capacity of enrichment analysis of protein family and domains as well as the retrieval of protein–protein interaction (PPI) networks. For example, Fig. 13 shows interactive PPI among the top 30 highly expressed genes in the interaction term. This plot is generated by API access to STRING-db with links to underlying literature. We can see many heat shock proteins interacting with each other.

3.8 Pathway Analysis

The list of DEGs are sensitive to thresholds of FDR and fold-change. Subsequently, the enrichment analysis based on these gene lists can also vary depending on user's choices. One solution is to analyze the expression data of all genes to detect pathways whose member genes are coherently regulated, even below the DEG threshold. In iDEP, pathway analysis is implemented through the analysis of fold-changes of all genes. Some pathway analysis methods, such as the standalone GSEA [21], use expression levels of all samples directly, which retain more information, to enable permutations of sample groupings. Therefore, pathway analysis in iDEP has its limitations due to the use of fold-changes. We recommend users try different tools. Even though sensitive to arbitrary cutoffs, the enrichment analysis on DEGs (*see* Subheading 3.7) can sometimes be more powerful than pathway analysis. Thus, users should combine results from both types of analyses.

With the combination of various pathway databases and the four different types of pathway analysis methods, iDEP can produce a lot of results quickly. Many of them might be merely **noise**. The danger is to try all the different combinations and search for the results we want. This is called **P-hacking** or **selective reporting**. It is a serious issue that we as developers worry about. Instead of picking the pathways, users should use the different combinations to corroborate and double check if the results are robust (*see* **Note 1**).

First, we focus on the heat response in wild-type animals, which is represented by fold-changes labeled with "heat-ctrl_for_wt". The default method GAGE [22] fails to identify any pathways using the GOBP. Switching to GSEA [21], we detect pathways (Fig. 14a) consistent with what we observed in enrichment analyses of *k*-Means clusters and DEGs. The results from GSEA precisely pinpoints the "Cellular response to heat" pathway that involves 54 genes. Their expression patterns are conveniently displayed below the pathway table. The adjusted P values are dependent on the number of permutations in the fgsea implementation of GSEA (*see* Table 4). With the 10,000 permutations as the current setting, the smallest unadjusted P value is 1×10^{-4}. That is why we see the same for all top pathways. Users should look at the test statistic to see the relative effect size of different pathways. That is how significant pathways are ranked. Users should pay attention to the pathways with the highest the NES (normalized enrichment scores) in GSEA results. As shown in Fig. 14a, the top pathways identified by GSEA are directly related to heat stress in wild-type animals. By switching pathway database to TRED, we can again detect the upregulation of 37 HIF1A target genes(Fig. 14b), in agreement with the results from the enrichment analysis of DEGs (Fig. 11b). The difference is only 10 of the 37 genes are among the DEGs as we are using fold-changes of all genes. Other TFs listed in (Fig. 14b) maybe involved in heat response, namely androgen receptor (AR),


A

Direction	GSEA analysis: heat vs ctrl_for_wt	NES	Genes	adj.Pval
Up	Cellular response to heat	2.3625	54	2.7e-03
	Chaperone-mediated protein folding	2.3247	47	2.7e-03
	Protein folding	2.3125	131	2.7e-03
	Response to unfolded protein	2.2499	91	2.7e-03
	Response to heat	2.2451	90	2.7e-03
	de novo protein folding	2.2277	31	2.7e-03
	Protein refolding	2.2008	27	2.7e-03
	Response to topologically incorrect protein	2.1984	110	2.7e-03
	de novo posttranslational protein folding	2.1982	30	2.7e-03
	Chaperone cofactor-dependent protein refolding	2.1783	26	2.7e-03
	Cellular response to unfolded protein	2.1577	66	2.7e-03
	Regulation of mitotic spindle assembly	2.1305	19	2.7e-03
	Response to temperature stimulus	2.0868	136	2.7e-03

B

Direction	GSEA analysis: heat vs ctrl_for_wt	NES	Genes	adj.Pval
Up	TRED:ZHAO HIF1A	1.9962	37	3.0e-03
	TRED:ZHAO SMAD3	1.7189	16	8.4e-02
	TRED:ZHAO PPARA	1.677	16	8.4e-02
	TRED:ZHAO AR	1.5786	40	1.0e-01
	TRED:ZHAO TRP53	1.5546	78	8.4e-02

Fig. 14 Results of pathway analysis of heat stress in wild-type animals using GSEA method on GO Biological Process database (**a**) and the transcription factor target genes (**b**)

SMAD3, PPARA, and TP53. By switching from GOBP to KEGG pathways, we can identify the regulated KEGG pathways. Figure 15 shows the change in gene expression on pathway diagrams of the top KEGG pathway, "Protein processing in endoplasmic reticulum (ER)." Heat shock causes the accumulation of misfolded proteins in the ER; HSP are known to regulate ER [23].

Second, we examined the pathways associated with heat response in animals without Hsf1. The fold-changes are labeled "Heat-ctrl". As the knockout genotype is the reference level, it is skipped in the comparison name. Without Hsf1, heat shock induces inflammatory response (Fig. 16). The upregulated GOBP terms forms two big clusters. One is related with inflammatory response, especially iterleukin-1 (IL1) induced. The other is related to migration of various immune cells (macrophage, monocyte, neutrophil). The representation of enriched GO terms as hierarchical clustering trees is a novel feature introduced in iDEP to aid the interpretation of GO terms, which are often redundant and overlapping. These enriched terms are different from what we observed in the wild-

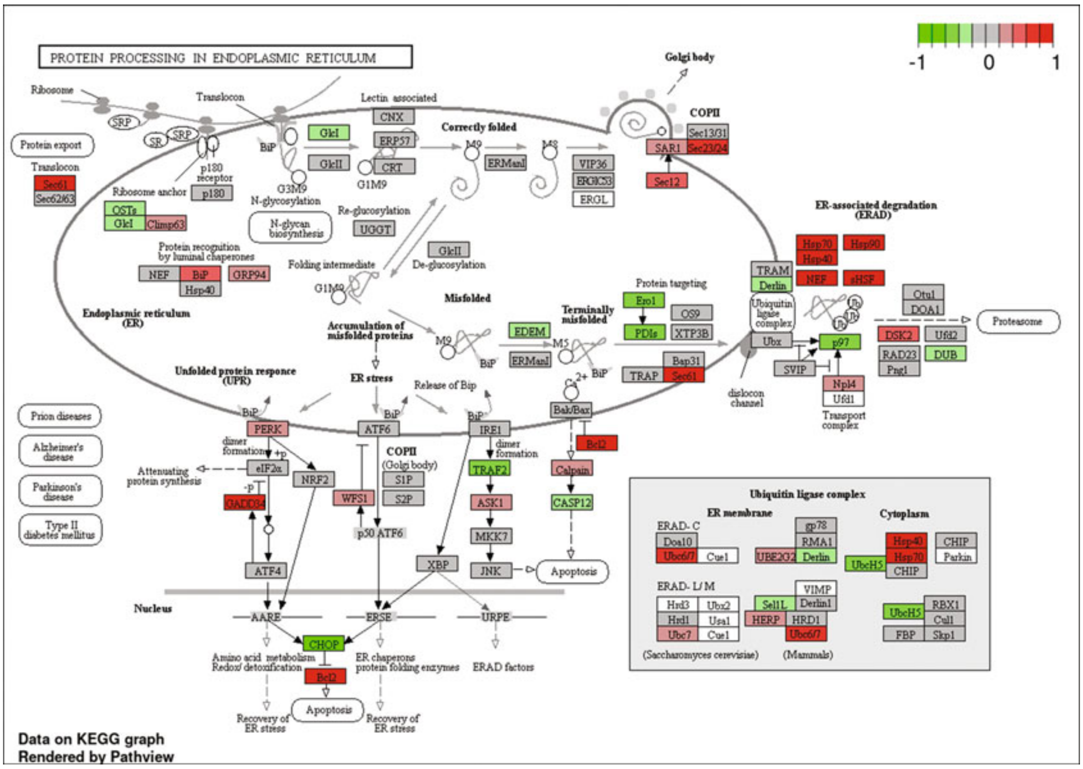


Fig. 15 KEGG pathway showing the expression profiles of genes related to protein processing in the endoplasmic reticulum. The fold-changes are heat response in wild-type animals

type animals, where the most of the significantly upregulated pathways are related to chaperone-mediated protein folding and heat response (Fig. 14a). This result is consistent with findings that HSF1 suppresses interleukin 1β (IL1B) by directly binding to its promoter [24].

Third, we can directly study the difference in response by analyzing the interaction term “I:genotype_wt-Treatment_heat”. The FCs are calculated by subtracting the treatment-induced FCs in knockout animals from those in the wild-type animals. The interaction term thus measures the additional effect of heat on animals with a functional Hsf1 gene. Applying the GSEA on GOBP pathways, we observe the upregulation of chaperone-mediated heat response and the downregulation of immune cell migration related terms (data not shown). This agrees with our observation above. To visualize pathway activity, we use the PGSEA method (Fig. 17a), which shows that the chaperone-mediated protein folding pathway is specifically activated in wild-type animals after treatment. We can select this pathway to confirm the expression profiles of the genes. Part of the heat map is shown in Fig. 17a, indicating strong upregulation of heat shock factors, specifically in wild-type animals with treatment. Finally, we can use the

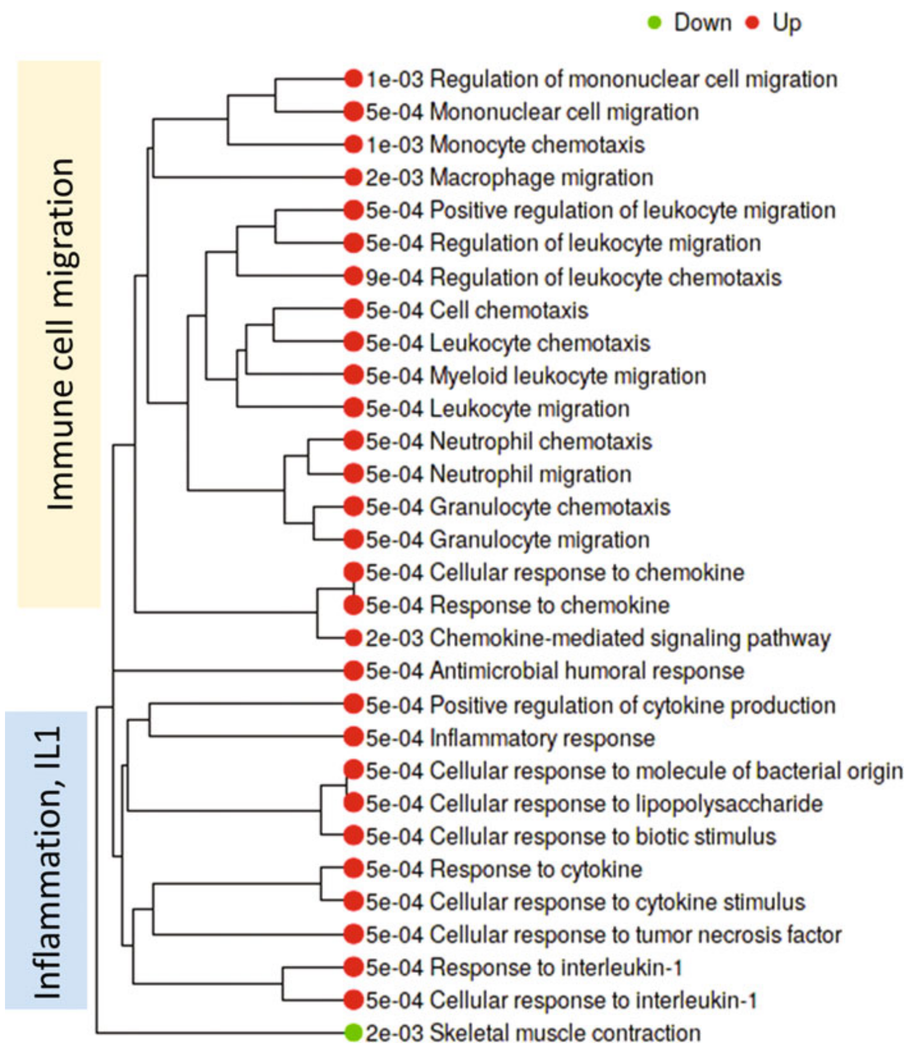
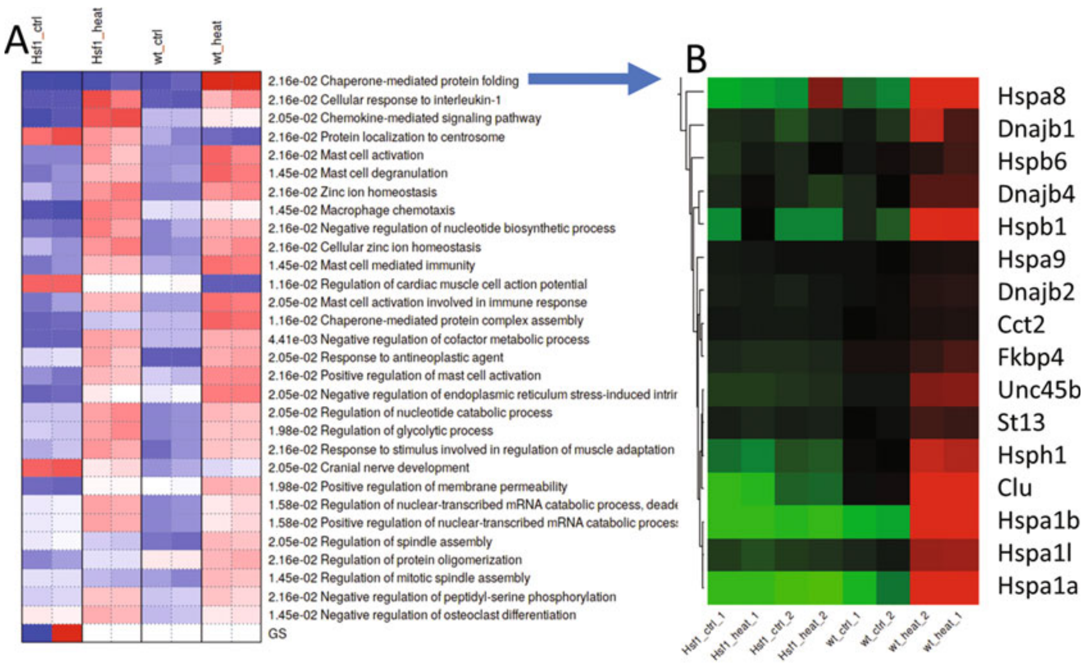


Fig. 16 Pathway analysis using GSEA on GO Biological Processes for the heat response in animals without HSF1

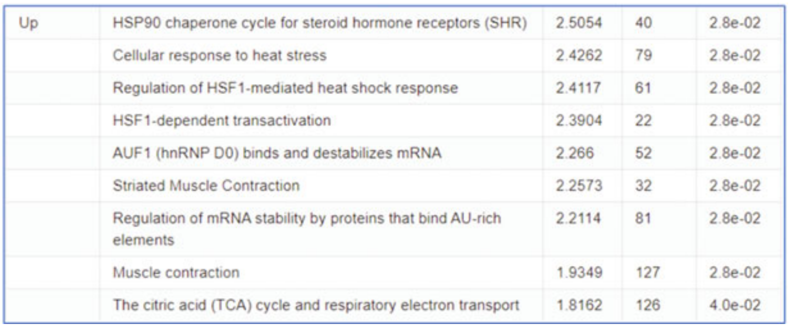
ReactomePA package [25] to detect significant pathways independent of the iDEP database. As shown in Fig. 17c, the HSF1-dependent transactivation pathway is detected alongside many heat response related pathways. As ReactomePA uses an independent database and calculation, this serves as a confirmation. Truly significant pathways should be robustly detected.

3.9 Reanalyzing Public Data

iDEP also offers access to many uniformly processed public RNA-seq data. For example, using "heat shock" as key words, we can find many similar datasets. We found a dataset (GSE81520) in worm that has similar design (Hsf1 knockout) with and without heat treatment. The files are available at Zenodo (https://doi.org/10.5281/zenodo.3783838). Analyzing this dataset with iDEP, we can



C



Up	HSP90 chaperone cycle for steroid hormone receptors (SHR)	2.5054	40	2.8e-02
	Cellular response to heat stress	2.4262	79	2.8e-02
	Regulation of HSF1-mediated heat shock response	2.4117	61	2.8e-02
	HSF1-dependent transactivation	2.3904	22	2.8e-02
	AUF1 (hnRNP D0) binds and destabilizes mRNA	2.266	52	2.8e-02
	Striated Muscle Contraction	2.2573	32	2.8e-02
	Regulation of mRNA stability by proteins that bind AU-rich elements	2.2114	81	2.8e-02
	Muscle contraction	1.9349	127	2.8e-02
	The citric acid (TCA) cycle and respiratory electron transport	1.8162	126	4.0e-02

Fig. 17 Pathways analysis using the fold-changes associated with the interaction term. (**a**) PGSEA using GOBP visualizes the activities of the pathways in the sample groups. The chaperone-mediated protein folding is only active in wild-type animals with heat treatment. The expression pattern of the associated genes are shown in (**b**). (**c**) Results using ReactomePA method which confirms the upregulation of HSF1 pathway and the subsequent heat response

also identify the enrichment of protein-folding and heat-stress induced genes in wild type, but not in knockout animals. With iDEP, users can easily analyze multiple datasets.

4 Notes

1. **Critical interpretation**: iDEP is developed thus far by a tiny team without much funding. As a result, it has not been thoroughly tested and evaluated. We thus welcome feedbacks or bug reports either through email or through our GitHub repository. We have

recently secured funding to thoroughly test and address bugs. To check the robustness of pathway analysis, users can:

1. Compare with enrichment analysis from *k*-Means clusters.
2. Compare with enrichment analysis of DEGs.
3. Use different pathway databases (many are overlapping).
4. Compare results using different pathway analysis methods.
5. Use independent data sources and methods accessible via STRING-db and ReactomePA.

iDEP also provides functionalities to visualize fold-changes on chromosomes. For larger datasets, users can also run biclustering and coexpression networks. These modules are still in development stage and need further enhancement.

References

1. Ge SX, Son EW, Yao R (2018) iDEP: an integrated web application for differential expression and pathway analysis of RNA-Seq data. BMC Bioinformatics 19(1):534. https://doi.org/10.1186/s12859-018-2486-6
2. Lachmann A, Torre D, Keenan AB, Jagodnik KM, Lee HJ, Wang L, Silverstein MC, Ma'ayan A (2018) Massive mining of publicly available RNA-seq data from human and mouse. Nat Commun 9(1):1366. https://doi.org/10.1038/s41467-018-03751-6
3. Ziemann M, Kaspi A, El-Osta A (2019) Digital expression explorer 2: a repository of uniformly processed RNA sequencing data. Gigascience 8 (4). https://doi.org/10.1093/gigascience/giz022
4. Neueder A, Gipson TA, Batterton S, Lazell HJ, Farshim PP, Paganetti P, Housman DE, Bates GP (2017) HSF1-dependent and -independent regulation of the mammalian in vivo heat shock response and its impairment in Huntington's disease mouse models. Sci Rep 7 (1):12556. https://doi.org/10.1038/s41598-017-12897-0
5. Mallona I, Peinado MA (2017) Truke, a web tool to check for and handle excel misidentified gene symbols. BMC Genomics 18(1):242. https://doi.org/10.1186/s12864-017-3631-8
6. Zeeberg BR, Riss J, Kane DW, Bussey KJ, Uchio E, Linehan WM, Barrett JC, Weinstein JN (2004) Mistaken identifiers: gene name errors can be introduced inadvertently when using Excel in bioinformatics. BMC Bioinformatics 5:80. https://doi.org/10.1186/1471-2105-5-80
7. Ziemann M, Eren Y, El-Osta A (2016) Gene name errors are widespread in the scientific literature. Genome Biol 17(1):177. https://doi.org/10.1186/s13059-016-1044-7
8. Zych K, Snoek BL, Elvin M, Rodriguez M, Van der Velde KJ, Arends D, Westra HJ, Swertz MA, Poulin G, Kammenga JE, Breitling R, Jansen RC, Li Y (2017) reGenotyper: detecting mislabeled samples in genetic data. PLoS One 12(2):e0171324. https://doi.org/10.1371/journal.pone.0171324
9. Jin X, Moskophidis D, Mivechi NF (2011) Heat shock transcription factor 1 is a key determinant of HCC development by regulating hepatic steatosis and metabolic syndrome. Cell Metab 14(1):91–103. https://doi.org/10.1016/j.cmet.2011.03.025
10. Li J, Labbadia J, Morimoto RI (2017) Rethinking HSF1 in stress, development, and organismal health. Trends Cell Biol 27 (12):895–905. https://doi.org/10.1016/j.tcb.2017.08.002
11. Tonelli C, Morelli MJ, Bianchi S, Rotta L, Capra T, Sabo A, Campaner S, Amati B (2015) Genome-wide analysis of p53 transcriptional programs in B cells upon exposure to genotoxic stress in vivo. Oncotarget 6 (28):24611–24626. https://doi.org/10.18632/oncotarget.5232
12. Brunet JP, Tamayo P, Golub TR, Mesirov JP (2004) Metagenes and molecular pattern discovery using matrix factorization. Proc Natl Acad Sci U S A 101(12):4164–4169. https://doi.org/10.1073/pnas.0308531101
13. Furge K, Dykema K (2012) PGSEA: parametric gene set enrichment analysis. R package version 1480

14. Kim SY, Volsky DJ (2005) PAGE: parametric analysis of gene set enrichment. BMC Bioinformatics 6:144. https://doi.org/10.1186/1471-2105-6-144
15. Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol 15(12):550. https://doi.org/10.1186/s13059-014-0550-8
16. Zhao F, Xuan Z, Liu L, Zhang MQ (2005) TRED: a transcriptional regulatory element database and a platform for in silico gene regulation studies. Nucleic Acids Res 33(Database issue):D103–D107. https://doi.org/10.1093/nar/gki004
17. Toma-Jonik A, Vydra N, Janus P, Widlak W (2019) Interplay between HSF1 and p53 signaling pathways in cancer initiation and progression: non-oncogene and oncogene addiction. Cell Oncol (Dordr) 42(5):579–589. https://doi.org/10.1007/s13402-019-00452-0
18. Davis AP, Grondin CJ, Johnson RJ, Sciaky D, McMorran R, Wiegers J, Wiegers TC, Mattingly CJ (2019) The comparative toxicogenomics database: update 2019. Nucleic Acids Res 47(D1):D948–D954. https://doi.org/10.1093/nar/gky868
19. Lamb J, Crawford ED, Peck D, Modell JW, Blat IC, Wrobel MJ, Lerner J, Brunet JP, Subramanian A, Ross KN, Reich M, Hieronymus H, Wei G, Armstrong SA, Haggarty SJ, Clemons PA, Wei R, Carr SA, Lander ES, Golub TR (2006) The Connectivity Map: using gene-expression signatures to connect small molecules, genes, and disease. Science 313(5795):1929–1935. https://doi.org/10.1126/science.1132939
20. Szklarczyk D, Franceschini A, Wyder S, Forslund K, Heller D, Huerta-Cepas J, Simonovic M, Roth A, Santos A, Tsafou KP, Kuhn M, Bork P, Jensen LJ, von Mering C (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. Nucleic Acids Res 43(Database issue):D447–D452. https://doi.org/10.1093/nar/gku1003
21. Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci U S A 102(43):15545–15550. https://doi.org/10.1073/pnas.0506580102
22. Luo W, Friedman MS, Shedden K, Hankenson KD, Woolf PJ (2009) GAGE: generally applicable gene set enrichment for pathway analysis. BMC Bioinformatics 10:161. https://doi.org/10.1186/1471-2105-10-161
23. Liu Y, Chang A (2008) Heat shock response relieves ER stress. EMBO J 27(7):1049–1059. https://doi.org/10.1038/emboj.2008.42
24. Cahill CM, Waterman WR, Xie Y, Auron PE, Calderwood SK (1996) Transcriptional repression of the prointerleukin 1beta gene by heat shock factor 1. J Biol Chem 271(40):24874–24879
25. Yu G, He QY (2016) ReactomePA: an R/Bioconductor package for reactome pathway analysis and visualization. Mol Biosyst 12(2):477–479. https://doi.org/10.1039/c5mb00663e
26. Bray NL, Pimentel H, Melsted P, Pachter L (2016) Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol 34(5):525–527. https://doi.org/10.1038/nbt.3519
27. Clifford H, Wessely F, Pendurthi S, Emes RD (2011) Comparison of clustering methods for investigation of genome-wide methylation array data. Front Genet 2:88. https://doi.org/10.3389/fgene.2011.00088
28. Law CW, Chen Y, Shi W, Smyth GK (2014) voom: precision weights unlock linear model analysis tools for RNA-seq read counts. Genome Biol 15(2):R29. https://doi.org/10.1186/gb-2014-15-2-r29
29. Sergushichev AA (2016) An algorithm for fast preranked gene set enrichment analysis using cumulative statistic calculation. bioRxiv

Chapter 23

Exploring Noninvasive Biomarkers with the miRandola Database: A Tool for Translational Medicine

Francesco Russo

Abstract

Noninvasive biomarkers are required for addressing crucial clinical needs. The ideal biomarker should be easily accessible and provide a unique characteristic for a healthy status or a pathological condition. In the last years, microRNAs (miRNAs) have been proposed as promising tissue-based biomarkers for several diseases such as cancer and cardiovascular diseases. Recently, miRNAs have shown great potential as novel noninvasive biomarkers, due to their high stability in human body fluids such as serum, plasma, and urine. Furthermore, many other noncoding RNAs (ncRNAs) such as long noncoding RNAs (lncRNAs) and circular RNAs (circRNAs) have shown to be novel biomarkers as well. The aim of this exciting research field is to offer novel tools, allowing translational scientists to develop new strategies for diagnosis, screening, and monitoring of diseases. In this book chapter, the miRandola database and its applications will be introduced. The database offers the possibility to explore information on ncRNAs as noninvasive biomarkers, manually extracted from scientific literature and public available resources.

Key words MicroRNAs, ncRNAs, Biomarkers, Database, Body fluids

1 Introduction

The miRandola database is a compendium of RNA classes found in several body fluids and released by different cell lines [1, 2]. The database represents an important tool for the translational medicine community, since it offers comprehensive information about the potential role of RNAs as noninvasive biomarkers in several biological contexts and pathological conditions.

The main RNA class in the database consists of miRNAs, which are small ncRNAs (21–23 nt long) that regulate gene expression by binding mRNAs (and other RNAs) [3]. Usually, miRNAs downregulate their targets, but it has been reported that they can also cause the upregulation of genes [4]. miRNAs bind at the 3′-UTR of transcripts but other locations have been reported as well [5].

The database reports data about other RNA classes, including lncRNAs and circRNAs. LncRNAs, are ~200 nt long and have several functions since they can interact with other RNAs or proteins. It has been reported that lncRNAs can act like "sponges" for miRNAs, by reducing the gene expression of one or more miRNAs at the same time. The sponge effect has downstream consequences in the regulation of miRNA targets, through complex loops of regulations [6, 7]. LncRNAs can also bind proteins [8], demonstrating their multifunctional role in the regulation of biological processes and pathological conditions.

CircRNAs are one of the most recent RNA classes. Like the linear lncRNAs, they can also act as sponges for miRNAs having high accessibility for miRNA binding sites in the circular sequence [9–11]. Some circRNAs have also been involved in development and in the transcriptional or posttranscriptional gene regulation of their host genes. For example, the CDR1as circRNA may promote the expression of CDR1 sense mRNA, but the mechanism behind it is still unknown [12].

These RNA classes have been found to be very stable in several body fluids, complexed in membrane-bound vesicles such as exosomes, which offer them protection from RNAses [13, 14]. miRNAs are also found with Argonaute 2 (Ago2) protein [15], part of the RNA-induced silencing complex (RISC) responsible of the RNA silencing process. Furthermore, they can be complexed with high-density lipoprotein (HDL) and actively transported into target cells [16].

In this book chapter, the functionalities of miRandola will be presented giving the users a practical guide for the use of the database as important resource in translational medicine.

2 Material

The miRandola database is freely accessible at http://mirandola.iit.cnr.it/. The full database or individual files can be downloaded and used for noncommercial purposes at http://mirandola.iit.cnr.it/download.php. The individual tab delimited text files include information about the list of papers reported in the database, data for miRNAs, lncRNAs, and circRNAs.

3 Methods

Most of the data in miRandola are based on searches in PubMed (https://pubmed.ncbi.nlm.nih.gov/). In the first versions of the database, articles reporting studies in the field of ncRNAs as noninvasive biomarkers were retrieved using keywords such as "microRNA," "circulating," and "extracellular." In the newest version, a text mining–assisted curation was introduced for helping the human biocurators to identify and prioritize papers for manual curation by applying the tagger software [17]. Then, the relevant information is manually extracted, such as the extracellular RNA forms, RNA type, experimental procedures, associated diseases and sample type.

The database also includes some studies from two publicly available resources, ExoCarta [18] and Vesiclepedia [19], manually curated databases on extracellular vesicles. Overall, miRandola contains 314 studies, 14 organisms, 3283 entries, 1002 miRNAs, 12 lncRNAs, 8 circRNAs, 7 extracellular RNA forms, 47 sample types, and 25 drugs having an effect on the release of ncRNAs into the bloodstream. Data are organized in a MySQL database running on an Apache server and Bootstrap (https://getbootstrap.com/) is used as front-end framework. The website is very dynamic, having several links between pages and it offers different options for searching and browse the entries in the database.

In the following sections, the main characteristics of the website will be introduced showing the different options for searching the database and some of the results. All the options are accessible through the user-friendly menu bar.

3.1 Home Page

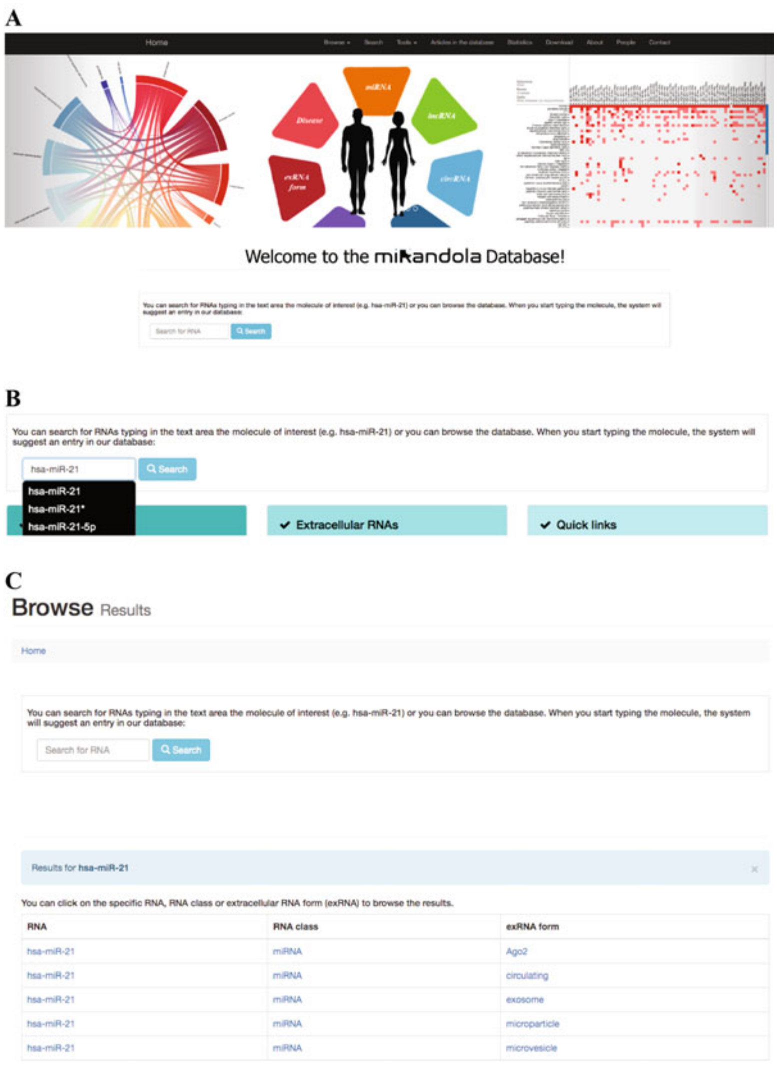
The home page of the website is very intuitive (Fig. 1a). The menu bar on the top offers all the main options including *Browse*, *Search*, *Tools*, *Articles in the database*, *Statistics*, and *Download*.

A text area is also available (Fig. 1b) for a quick search for RNA molecules of interest (e.g., "hsa-miR-21"). When the user starts typing, the autocomplete function will suggest an entry in the database, facilitating the identification of the term of interest. After clicking on the RNA of interest, users will have an overview of the results, reporting the specific RNA class and extracellular forms (Fig. 1c). The links between web pages, allow the user to navigate the results by clicking on the RNA term (in this example "hsa-miR-21"), RNA class ("miRNA"), or the extracellular RNA form ("Ago2," "circulating," "exosome," "microparticle," or "microvesicle").

3.2 Browsing the Database

The menu bar offers several options. The *Browse* option is probably the best way to explore the database and having an overview of the data types (Fig. 2). Users can browse for *microRNAs*, *long non-coding RNAs*, *circular RNAs*, *Diseases*, *exRNA forms*, *Samples*, *Drugs*, and *Organisms*.

Upon selection of the search-term of interest, users will be able to inspect the results. It is possible to choose for three RNA classes: *microRNAs*, *long non-coding RNAs*, and *circular RNAs*. Figure 3 shows, as example, the result for *microRNA* which represents the largest RNA class in the database. The result table contains five fields: *Mature miRNA ID from literature* (as reported in the original research paper), *miRBase ID* (the official miRNA ID extracted from the miRBase database [20]), *miRBase accession* (official accession number), *miRBase family*, and *Organism*.



Welcome to the miRandola Database!

You can search for RNAs typing in the text area the molecule of interest (e.g. hsa-miR-21) or you can browse the database. When you start typing the molecule, the system will suggest an entry in our database:

Search for RNA

You can search for RNAs typing in the text area the molecule of interest (e.g. hsa-miR-21) or you can browse the database. When you start typing the molecule, the system will suggest an entry in our database:

hsa-miR-21

Browse Results

Home

You can search for RNAs typing in the text area the molecule of interest (e.g. hsa-miR-21) or you can browse the database. When you start typing the molecule, the system will suggest an entry in our database:

Search for RNA

Results for **hsa-miR-21**

You can click on the specific RNA, RNA class or extracellular RNA form (exRNA) to browse the results.

RNA	RNA class	exRNA form
hsa-miR-21	miRNA	Ago2
hsa-miR-21	miRNA	circulating
hsa-miR-21	miRNA	exosome
hsa-miR-21	miRNA	microparticle
hsa-miR-21	miRNA	microvesicle

Fig. 1 Webpage of the database and search bar. (**a**) Home page; (**b**) search bar with autocomplete function; (**c**) overview of the result of the search

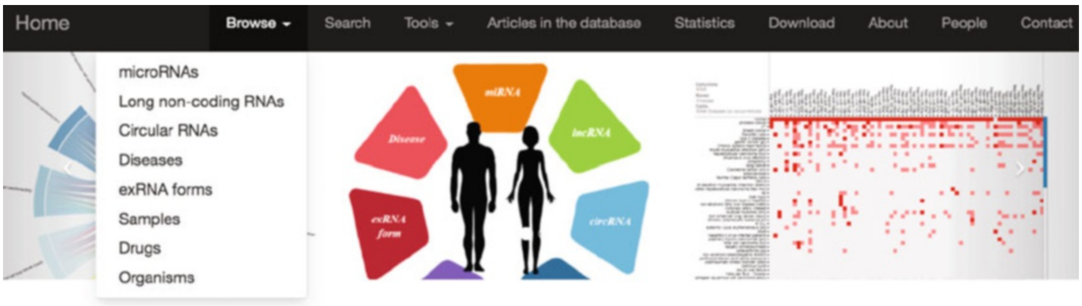
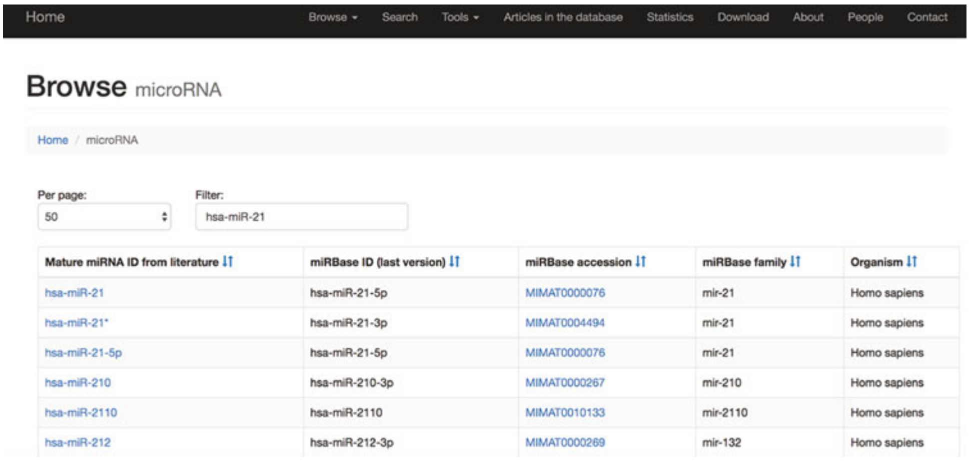


Fig. 2 Menu bar and Browse option. Users can browse for *microRNAs*, *long non-coding RNAs*, *circular RNAs*, *Diseases*, *exRNA forms*, *Samples*, *Drugs*, and Organisms



Browse microRNA

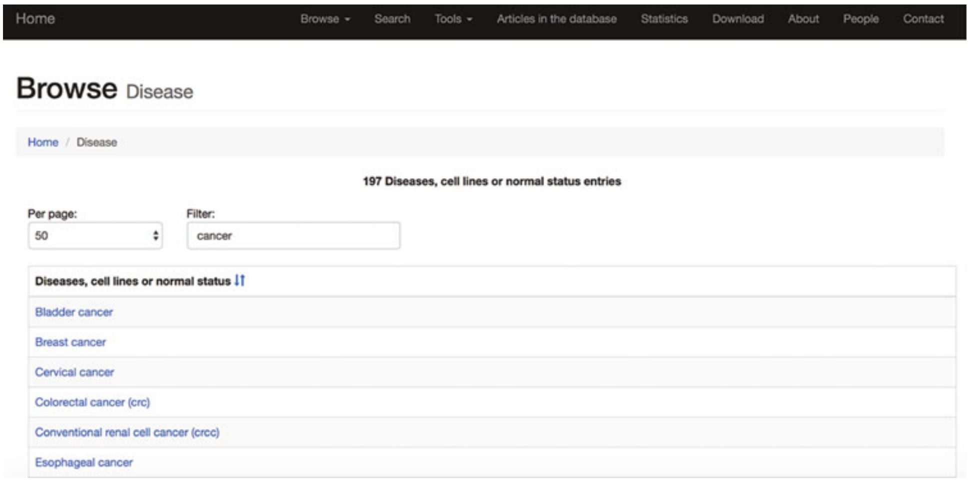
Home / microRNA

Per page: 50

Filter: hsa-miR-21

Mature miRNA ID from literature	miRBase ID (last version)	miRBase accession	miRBase family	Organism
hsa-miR-21	hsa-miR-21-5p	MIMAT0000076	mir-21	Homo sapiens
hsa-miR-21*	hsa-miR-21-3p	MIMAT0004494	mir-21	Homo sapiens
hsa-miR-21-5p	hsa-miR-21-5p	MIMAT0000076	mir-21	Homo sapiens
hsa-miR-210	hsa-miR-210-3p	MIMAT0000267	mir-210	Homo sapiens
hsa-miR-2110	hsa-miR-2110	MIMAT0010133	mir-2110	Homo sapiens
hsa-miR-212	hsa-miR-212-3p	MIMAT0000269	mir-132	Homo sapiens

Fig. 3 Result page for the Browse option (miRNAs). The result page for the Browse option, offers an overview of the data types in the database. In this example, the overview for miRNAs is given. It is possible to filter for each field of the table (in this example, we filter for “hsa-miR-21”)



Browse Disease

Home / Disease

197 Diseases, cell lines or normal status entries

Per page: 50

Filter: cancer

Diseases, cell lines or normal status
Bladder cancer
Breast cancer
Cervical cancer
Colorectal cancer (crc)
Conventional renal cell cancer (crcc)
Esophageal cancer

Fig. 4 Result page for the Browse option (diseases). Pathologies and normal conditions in which the RNAs were found. It also contains cell lines who released the molecules into the bloodstream

The result page offers the option of filtering for each field of the table (in this example, we filter for “hsa-miR-21”). Upon typing, the table is updated by showing the results related to the term we are looking for.

The result table for *Diseases* shows pathologies and normal conditions in which the RNAs were found (Fig. 4). It also contains cell lines who released the molecules into the bloodstream. By applying the filter, we are able to look at specific terms of interest (in this example, “cancer”). The terms are listed as reported in the literature.

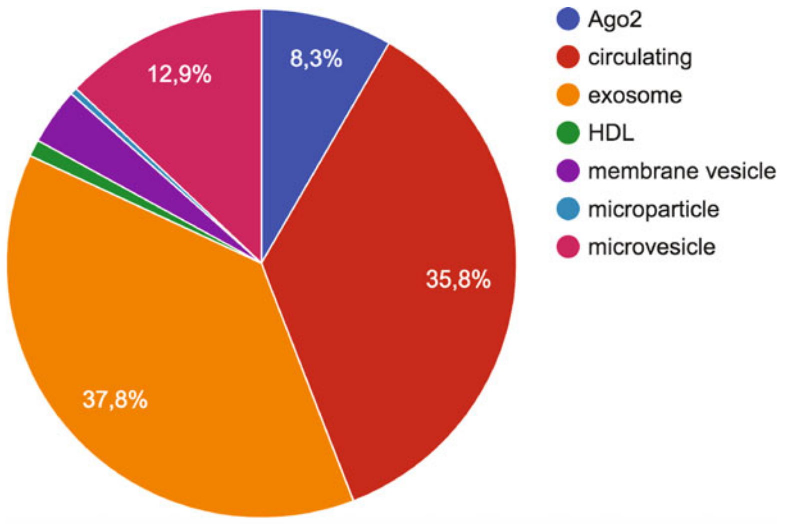


Fig. 5 Distribution of extracellular RNA forms. The majority of the extracellular RNAs in the database are found to be in exosomes but the second largest group consists of the category "circulating," used when the authors do not distinguish between all the other forms

The section of the extracellular RNA forms (*exRNA forms*) consists of seven entries based on the different types of complexes: "Ago2," "HDL," "membrane vesicle," "microparticle," "microvesicle," and "exosome." An additional category defined in miRandola is "circulating." This term is used when the other extracellular forms are not specified in the research articles, which is the case for more than 35% of the RNAs in the database (Fig. 5).

The database includes 47 types of samples, with serum and plasma contributing to the largest groups (Fig. 6). The variety of samples shows how research increased over the years, exploring the presence of extracellular RNAs in several body fluids and their potential role as noninvasive biomarkers.

The last two sections regard *Drugs* and *Organisms*. The drug effects on the release of RNAs in the bloodstream, is one of the newest research topics in this exciting field. The database includes 25 drugs used in different diseases such as cancer and cardiovascular conditions. For example, it has been reported that aspirin prevents the transfer of miR-126 from the platelet to the plasma compartment in type 2 diabetes [21]. Another example is given by the use of miR-21 as marker to indicate the transformation to hormone-refractory prostate cancer, and potential predictor for the efficacy of docetaxel-based chemotherapy [22].

3.3 Search Function in miRandola

The search function allows the user to look for RNAs (e.g., hsa--miR-21), diseases (e.g., prostate cancer), or samples of interest (e.g., plasma) by typing in a text area (Fig. 7). It is also possible

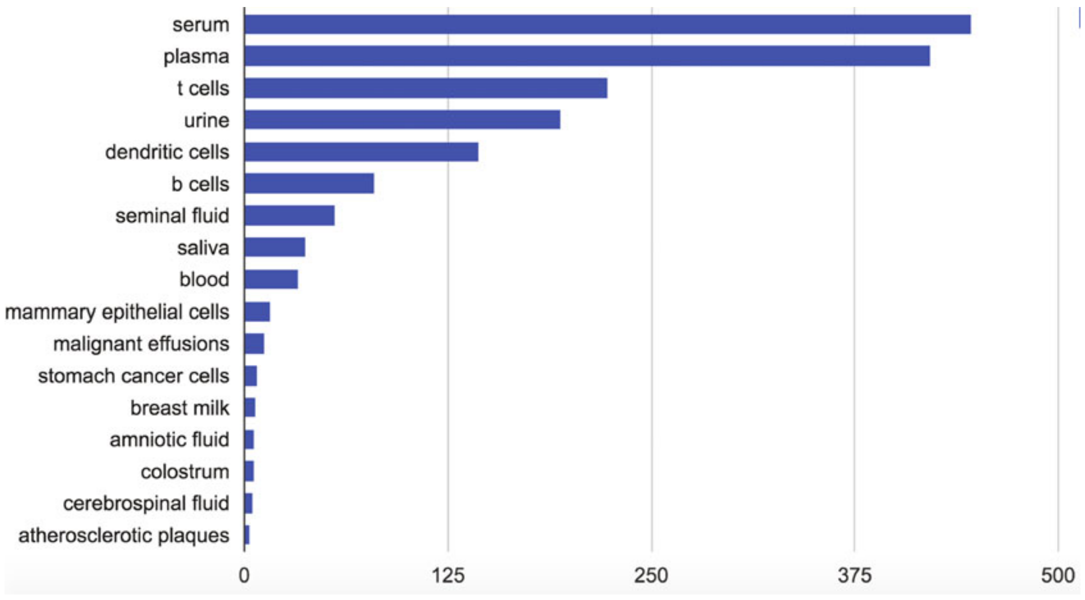
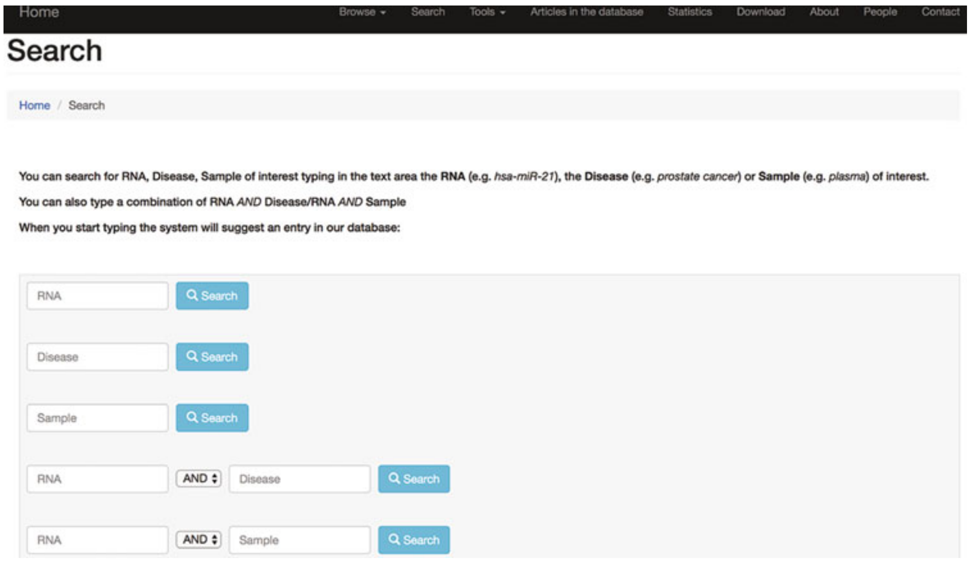


Fig. 6 Main samples in the database. Serum, plasma, and urine are the largest categories



Search

Home / Search

You can search for RNA, Disease, Sample of interest typing in the text area the **RNA** (e.g. *hsa-miR-21*), the **Disease** (e.g. *prostate cancer*) or **Sample** (e.g. *plasma*) of interest.

You can also type a combination of RNA *AND* Disease/RNA *AND* Sample

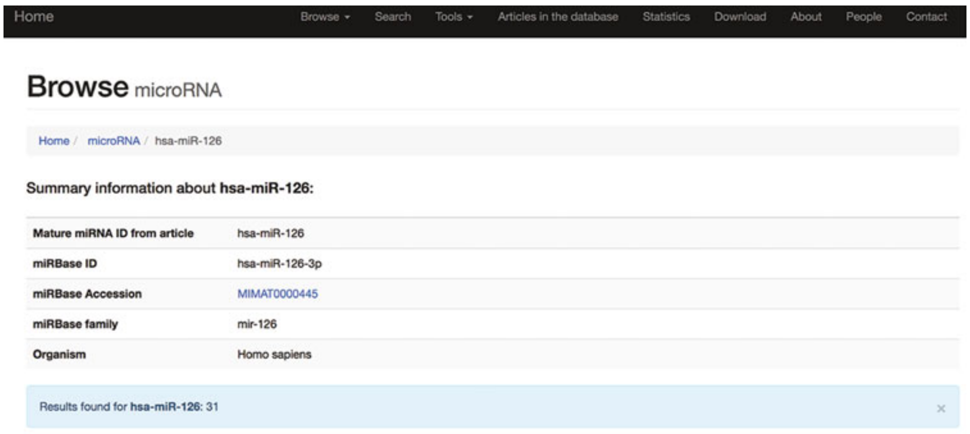
When you start typing the system will suggest an entry in our database:

Fig. 7 Search function. The database offers a convenient Search option, to look for specific RNA, disease or sample of interest, or a combination of them

to type a combination of RNA *AND/OR* disease or sample. The implementation of the autocomplete function makes it easier to find the term of interest.

3.4 Result Table

The result table contains the main content of the database, which is reachable from the *Browse* or *Search* options. The top part of the main content for *microRNAs* (Fig. 8), shows summary information



Browse microRNA

Home / microRNA / hsa-miR-126

Summary information about hsa-miR-126:

Mature miRNA ID from article	hsa-miR-126
miRBase ID	hsa-miR-126-3p
miRBase Accession	MIMAT0000445
miRBase family	mir-126
Organism	Homo sapiens

Results found for hsa-miR-126: 31

Fig. 8 Top part of the result table. It shows summary information about the miRNA of interest

about the specific miRNA of interest (in this example hsa-miR-126), reporting the same fields of the *Browse* section (i.e., *Mature miRNA ID from literature*, *miRBase ID*, *miRBase accession*, *miRBase family*, and *Organism)* and the number of results for the specific search.

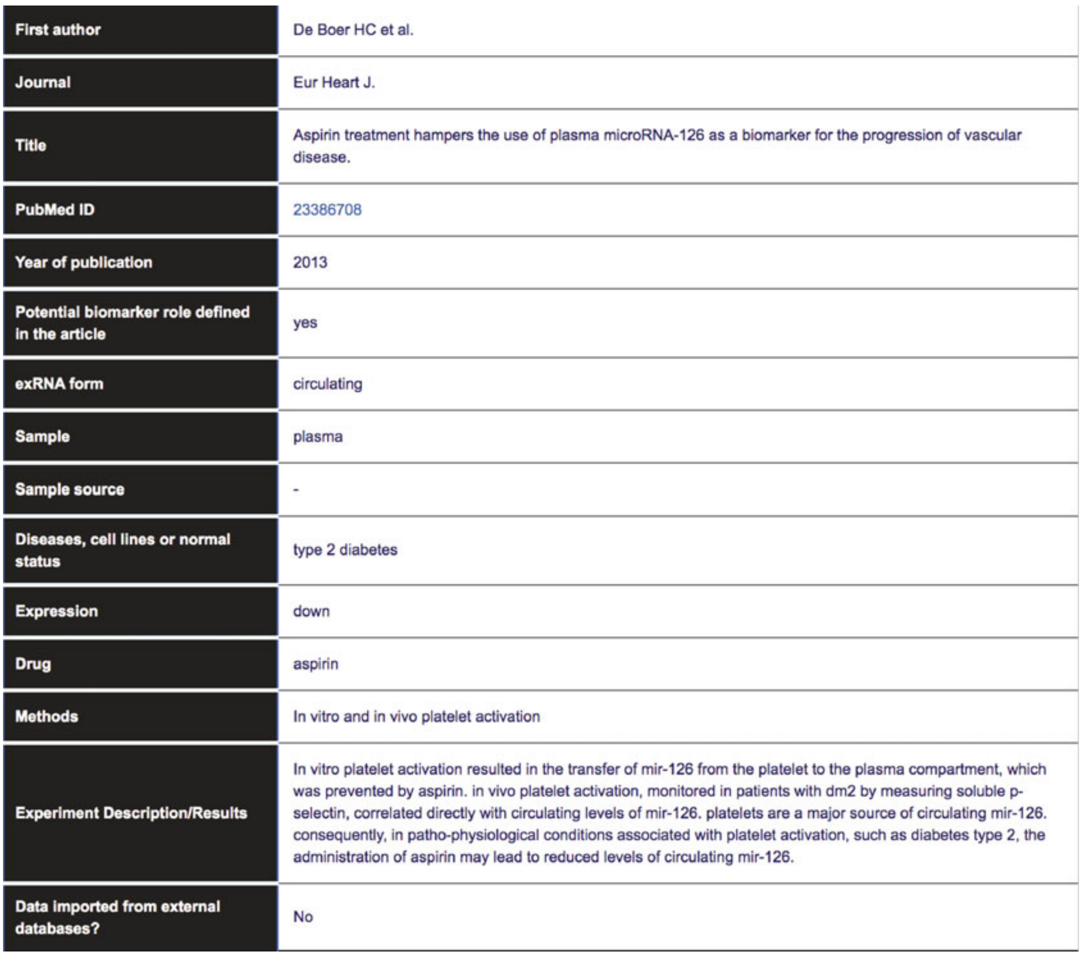
The second part of the result table, contains several fields manually extracted from the research articles (Fig. 9), including publication id (PubMed ID), title of the research article, publication year, first author, and journal. The associations to diseases, sample types, extracellular RNA type, RNA expression level, and the drug used in the experiment are also reported. Furthermore, the table includes the methods used to verify the expression or other lab techniques, and a brief description of the results summarizing the main findings. The "Potential biomarker role defined in the article," is also reported, which indicates whether the selected RNA has a potential role as biomarker, or its role is still unknown.

3.5 Tools

miRandola includes a *Visualization* section consisting of a graphical representation of the different data types in the database, a circos plot showing the disease network representing the RNAs in common for the most representative tumor types and a heat map showing the RNA–Disease co-occurrences. Together with the *Statistics* section, these simple visualizations allow users to have a general overview of the data in the database.

3.6 Conclusions

miRandola was established in 2012 as first database of its kind, in the early stage of this research field when few scientific articles were available. Initially, the main focus of the database was to collect data regarding human miRNAs as potential noninvasive biomarkers. Later, the number of publications increased and the database started to include additional ncRNAs, organisms, diseases, and drugs. The database is manually curated, which entails the



First author	De Boer HC et al.
Journal	Eur Heart J.
Title	Aspirin treatment hampers the use of plasma microRNA-126 as a biomarker for the progression of vascular disease.
PubMed ID	23386708
Year of publication	2013
Potential biomarker role defined in the article	yes
exRNA form	circulating
Sample	plasma
Sample source	-
Diseases, cell lines or normal status	type 2 diabetes
Expression	down
Drug	aspirin
Methods	In vitro and in vivo platelet activation
Experiment Description/Results	In vitro platelet activation resulted in the transfer of mir-126 from the platelet to the plasma compartment, which was prevented by aspirin. in vivo platelet activation, monitored in patients with dm2 by measuring soluble p-selectin, correlated directly with circulating levels of mir-126. platelets are a major source of circulating mir-126. consequently, in patho-physiological conditions associated with platelet activation, such as diabetes type 2, the administration of aspirin may lead to reduced levels of circulating mir-126.
Data imported from external databases?	No

Fig. 9 Bottom part of the result table. It reports all the manually extracted information about the miRNA (or other RNA classes) of interest

involvement of several human biocurators and a time-consuming process for updating it. Recently, a text mining–assisted curation was included to support the biocurators and speed up the updates. Future directions will consider to use more text-mining approaches to support the curation and to have more and updated data on noninvasive biomarkers.

Acknowledgments

The database has been developed in collaboration with several scientists over the last years that made possible to realize this idea. I would like to thank my first mentor Prof. Alfredo Ferro (University of Catania, Italy) and his lab to believe in this project in 2012, Dr. Marco Pellegrini (National Research Council, Italy), Prof. Lars J Jensen, and Prof. Søren Brunak (University of Copenhagen, Denmark).

I was supported by the Novo Nordisk Foundation grant agreement no. NNF14CC0001 and by the BRIDGE—Translational Excellence Programme (bridge.ku.dk), Faculty of Health and Medical Sciences, University of Copenhagen, funded by the Novo Nordisk Foundation (grant agreement no. NNF18SA0034956).

References

1. Russo F, Di Bella S, Nigita G, Macca V, Lagana A, Giugno R, Pulvirenti A, Ferro A (2012) miRandola: extracellular circulating microRNAs database. PLoS One 7(10): e47786. https://doi.org/10.1371/journal.pone.0047786
2. Russo F, Di Bella S, Vannini F, Berti G, Scoyni F, Cook HV, Santos A, Nigita G, Bonnici V, Lagana A, Geraci F, Pulvirenti A, Giugno R, De Masi F, Belling K, Jensen LJ, Brunak S, Pellegrini M, Ferro A (2018) miRandola 2017: a curated knowledge base of non-invasive biomarkers. Nucleic Acids Res 46(D1):D354–D359. https://doi.org/10.1093/nar/gkx854
3. Bartel DP (2009) MicroRNAs: target recognition and regulatory functions. Cell 136 (2):215–233. https://doi.org/10.1016/j.cell.2009.01.002
4. Vasudevan S, Tong Y, Steitz JA (2007) Switching from repression to activation: microRNAs can up-regulate translation. Science 318 (5858):1931–1934. https://doi.org/10.1126/science.1149460
5. Lytle JR, Yario TA, Steitz JA (2007) Target mRNAs are repressed as efficiently by microRNA-binding sites in the 5′ UTR as in the 3′ UTR. Proc Natl Acad Sci U S A 104 (23):9667–9672. https://doi.org/10.1073/pnas.0703820104
6. Shan Y, Ma J, Pan Y, Hu J, Liu B, Jia L (2018) LncRNA SNHG7 sponges miR-216b to promote proliferation and liver metastasis of colorectal cancer through upregulating GALNT1. Cell Death Dis 9(7):722. https://doi.org/10.1038/s41419-018-0759-7
7. Salmena L, Poliseno L, Tay Y, Kats L, Pandolfi PP (2011) A ceRNA hypothesis: the Rosetta Stone of a hidden RNA language? Cell 146 (3):353–358. https://doi.org/10.1016/j.cell.2011.07.014
8. Ferre F, Colantoni A, Helmer-Citterich M (2016) Revealing protein-lncRNA interaction. Brief Bioinform 17(1):106–116. https://doi.org/10.1093/bib/bbv031
9. Zhang X, Wang S, Wang H, Cao J, Huang X, Chen Z, Xu P, Sun G, Xu J, Lv J, Xu Z (2019) Circular RNA circNRIP1 acts as a microRNA-149-5p sponge to promote gastric cancer progression via the AKT1/mTOR pathway. Mol Cancer 18(1):20. https://doi.org/10.1186/s12943-018-0935-5
10. Han D, Li J, Wang H, Su X, Hou J, Gu Y, Qian C, Lin Y, Liu X, Huang M, Li N, Zhou W, Yu Y, Cao X (2017) Circular RNA circMTO1 acts as the sponge of microRNA-9 to suppress hepatocellular carcinoma progression. Hepatology 66(4):1151–1164. https://doi.org/10.1002/hep.29270
11. Kulcheski FR, Christoff AP, Margis R (2016) Circular RNAs are miRNA sponges and can be used as a new class of biomarker. J Biotechnol 238:42–51. https://doi.org/10.1016/j.jbiotec.2016.09.011
12. Hansen TB, Wiklund ED, Bramsen JB, Villadsen SB, Statham AL, Clark SJ, Kjems J (2011) miRNA-dependent gene silencing involving Ago2-mediated cleavage of a circular antisense RNA. EMBO J 30(21):4414–4422. https://doi.org/10.1038/emboj.2011.359
13. Zhang J, Li S, Li L, Li M, Guo C, Yao J, Mi S (2015) Exosome and exosomal microRNA: trafficking, sorting, and function. Genomics Proteomics Bioinformatics 13(1):17–24. https://doi.org/10.1016/j.gpb.2015.02.001
14. Huang X, Yuan T, Tschannen M, Sun Z, Jacob H, Du M, Liang M, Dittmar RL, Liu Y, Liang M, Kohli M, Thibodeau SN, Boardman L, Wang L (2013) Characterization of human plasma-derived exosomal RNAs by deep sequencing. BMC Genomics 14:319. https://doi.org/10.1186/1471-2164-14-319
15. Turchinovich A, Weiz L, Langheinz A, Burwinkel B (2011) Characterization of extracellular circulating microRNA. Nucleic Acids Res 39(16):7223–7233. https://doi.org/10.1093/nar/gkr254
16. Vickers KC, Palmisano BT, Shoucri BM, Shamburek RD, Remaley AT (2011) MicroRNAs are transported in plasma and delivered to recipient cells by high-density lipoproteins. Nat Cell Biol 13(4):423–433. https://doi.org/10.1038/ncb2210
17. Pafilis E, Frankild SP, Fanini L, Faulwetter S, Pavloudi C, Vasileiadou A, Arvanitidis C,

Jensen LJ (2013) The SPECIES and ORGANISMS resources for fast and accurate identification of taxonomic names in text. PLoS One 8 (6):e65390. https://doi.org/10.1371/journal.pone.0065390

18. Keerthikumar S, Chisanga D, Ariyaratne D, Al Saffar H, Anand S, Zhao K, Samuel M, Pathan M, Jois M, Chilamkurti N, Gangoda L, Mathivanan S (2016) ExoCarta: a web-based compendium of exosomal cargo. J Mol Biol 428(4):688–692. https://doi.org/10.1016/j.jmb.2015.09.019
19. Kalra H, Simpson RJ, Ji H, Aikawa E, Altevogt P, Askenase P, Bond VC, Borras FE, Breakefield X, Budnik V, Buzas E, Camussi G, Clayton A, Cocucci E, Falcon-Perez JM, Gabrielsson S, Gho YS, Gupta D, Harsha HC, Hendrix A, Hill AF, Inal JM, Jenster G, Kramer-Albers EM, Lim SK, Llorente A, Lotvall J, Marcilla A, Mincheva-Nilsson L, Nazarenko I, Nieuwland R, Nolte-'t Hoen EN, Pandey A, Patel T, Piper MG, Pluchino S, Prasad TS, Rajendran L, Raposo G, Record M, Reid GE, Sanchez-Madrid F, Schiffelers RM, Siljander P, Stensballe A, Stoorvogel W, Taylor D, Thery C, Valadi H, van Balkom BW, Vazquez J, Vidal M, Wauben MH, Yanez-Mo M, Zoeller M, Mathivanan S (2012) Vesiclepedia: a compendium for extracellular vesicles with continuous community annotation. PLoS Biol 10(12):e1001450. https://doi.org/10.1371/journal.pbio.1001450
20. Kozomara A, Birgaoanu M, Griffiths-Jones S (2019) miRBase: from microRNA sequences to function. Nucleic Acids Res 47(D1): D155–D162. https://doi.org/10.1093/nar/gky1141
21. de Boer HC, van Solingen C, Prins J, Duijs JM, Huisman MV, Rabelink TJ, van Zonneveld AJ (2013) Aspirin treatment hampers the use of plasma microRNA-126 as a biomarker for the progression of vascular disease. Eur Heart J 34 (44):3451–3457. https://doi.org/10.1093/eurheartj/eht007
22. Zhang HL, Yang LF, Zhu Y, Yao XD, Zhang SL, Dai B, Zhu YP, Shen YJ, Shi GH, Ye DW (2011) Serum miRNA-21: elevated levels in patients with metastatic hormone-refractory prostate cancer and potential predictive factor for the efficacy of docetaxel-based chemotherapy. Prostate 71(3):326–331. https://doi.org/10.1002/pros.21246

Chapter 24

Database Resources for Functional Circular RNAs

Dahui Hu, Peijing Zhang, and Ming Chen

Abstract

Circular RNA (or circRNA) is a type of single-stranded covalently closed circular RNA molecule and play important roles in diverse biological pathways. A comprehensive functionally annotated circRNA database will help to understand the circRNAs and their functions. CircFunBase is such a web-accessible database that aims to provide a high-quality functional circRNA resource including experimentally validated and computationally predicted functions. CircFunBase provides visualized circRNA–miRNA interaction networks. In addition, a genome browser is provided to visualize the genome context of circRNA. In this chapter, we illustrate examples of searching for circRNA and getting detailed information of circRNA. Moreover, other circRNA related databases are outlined.

Key words CircRNA, CircRNA function, CircRNA interaction, CircFunBase, Database

1 Introduction

Circular RNAs are a special group of endogenous RNAs characterized by covalent closed loops [1]. CircRNA are reported to be expressed widespread both in animals and plants [2–4]. Thought expression levels of most of circRNAs are low [5], increasingly evidence shows that circular RNAs play important roles in regulation of biological pathways [6]. CircRNA can perform its function in a variety of ways, such as acting as miRNA sponges to regulate the target genes of miRNA [7], acting as RNA binding proteins sponges to regulate proteins function indirectly [8], working as protein scaffold to facilitate the colocalization of enzymes and their substrates to influence the reaction kinetics [9]. Some circRNAs have been proved to function as templates and translate to peptides [10]. As a new member of RNA world, circRNAs have greatly increased the complexity of transcriptional regulation. Recent reviews have overviewed the biogenesis and function of circRNAs [11, 12].

Together with high-throughput sequencing technologies and circRNA detection pipeline like CIRI2 [13], KNIFE [14], and Circmarker [15], thousands of circRNAs in both animals and plants have been identified and integrated into the circRNA databases, such as circBase [16], circAtlas [17], and PlantCircNet [18] (Table 1). Although these circRNA databases provide the tissue information, the functions of circRNAs remains unclear.

To bridge the gap between circRNAs and their functions, we collect current findings about function of circRNA from literature and build a functional circRNA knowledgebase (CircFunBase, http://bis.zju.edu.cn/CircFunBase).

Table 1
Resources of circRNA databases

Name	Description	URL	Year
Circ2Traits [21]	CircRNAs potentially associated with disease and traits	http://gyanxet-beta.com/circdb	2013
circBase [16]	A unified circRNA data set from previous publications	http://www.circbase.org/	2014
CircInteractome [22]	Circular RNAs and Their Interacting Proteins and microRNAs	https://circinteractome.nia.nih.gov/	2016
CircNet [23]	CircRNA–miRNA–mRNA interaction networks	http://circnet.mbc.nctu.edu.tw/	2016
circRNADb [24]	CircRNAs with protein-coding potential	http://202.195.183.4:8000/circrnadb	2016
TSCD [25]	Tissue-Specific circRNAs in human and mouse tissues	http://gb.whu.edu.cn/TSCD	2016
PlantcircBase [26]	The first plant circular RNA database that derived from public datasets	http://ibi.zju.edu.cn/plantcircbase/	2017
PlantCircNet [18]	Providing plant circRNA–miRNA–gene regulatory networks, as well as circRNA information and circRNA expression profiles	http://bis.zju.edu.cn/plantcircnet/index.php	2017
AtCircDB [27]	Tissue-specific Arabidopsis circular RNA database	http://genome.sdau.edu.cn	2017
CSCD [28]	Cancer specific circRNAs	http://gb.whu.edu.cn/CSCD/	2018
exoRBase [29]	CircRNAs derived from RNA-seq data analyses of human blood exosomes	http://www.exoRBase.org	2018
CircR2Disease [30]	Experimentally supported circRNA and disease associations database	http://bioinfo.snnu.edu.cn/CircR2Disease/	2018

(continued)

Table 1 (continued)

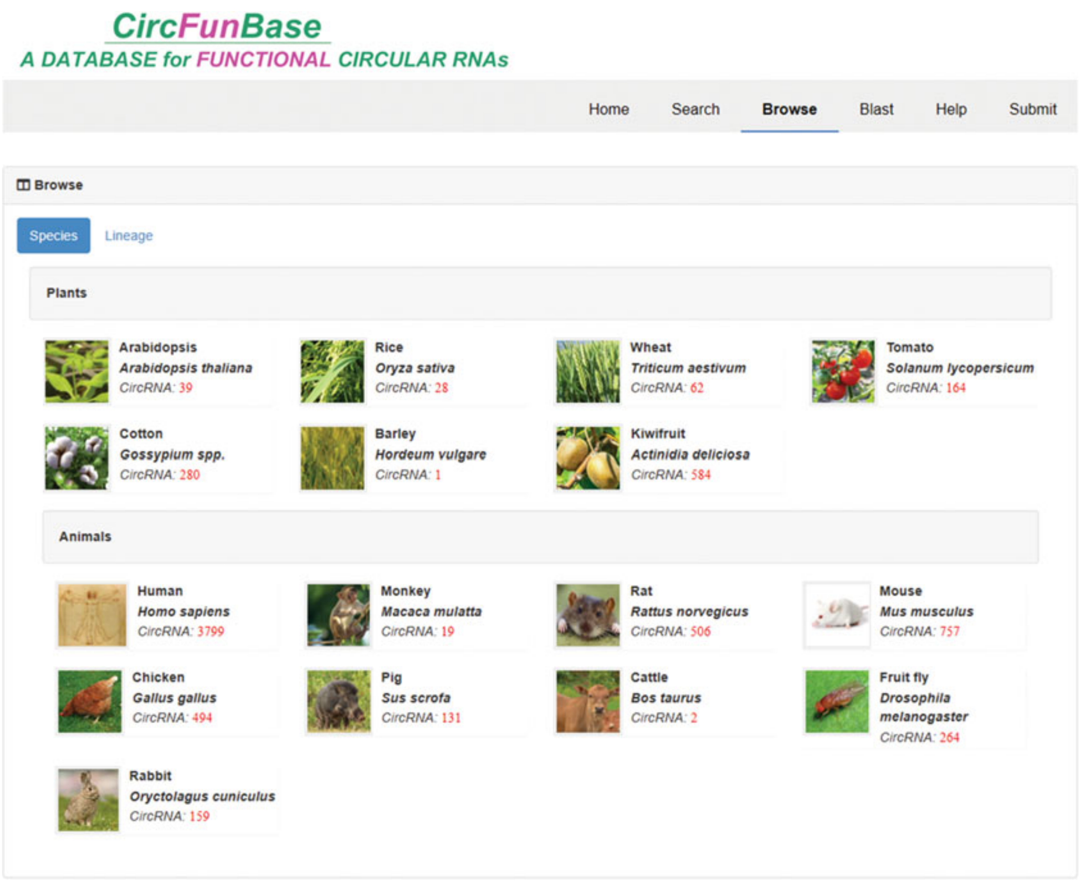
Name	Description	URL	Year
CIRCpedia [31]	CircRNA annotation from over 180 RNA-seq datasets across six different species	http://www.picb.ac.cn/rnomics/circpedia	2018
TRCirc [32]	Transcriptional regulation information of circular RNAs	http://www.licpathway.net/TRCirc	2018
circAtlas [17]	Integrating circRNAs and their expression and functional profiles across six species	http://159.226.67.237:8080/new/index.php	2019
CircFunBase [33]	Providing a high-quality functional circRNA resource including experimentally validated and computationally predicted functions	http://bis.zju.edu.cn/CircFunBase	2019

2 Materials

The CircFunBase database is implemented using HTML and PHP languages with relational database MySQL. CircFunBase provides user-friendly interface and includes flexible JQuery plugins to enhance interactivity. Cytoscape.js was used to visualize the circRNA-associated networks, while Dalliance [19] was used to view the context of genome. The BLAST module was implemented through SequenceServer [20]. The API module was provided by PhalAPI service. CircFunBase is freely available at: http://bis.zju.edu.cn/CircFunBase, updated Firefox or Chrome browsers are recommended. At the moment, CircFunBase deposits 7289 circRNAs from 7 plants and 9 animals (Fig. 1).

In the chapter, we provide guidelines to use the CircFunBase website. For the sequence-based module, the sequence of hsa_circ_0001946 which is also known as CDR1as will be used as example. This sequence can also be found on http://bis.zju.edu.cn/CircFunBase/detail.php?name=hsa_circ_0001946.

```
>hsa_circ_0001946
GGGTTTCCGATGGCACCTGTGTCAAGGTCTTCCAACAACTCCGGGTCTTCCAGCGACTT-
CAAGTCTTCCAATAATCTCAAGGTCTTCCAGATAATCCTGAGCTTCCAGAAAATCCA-
CATCTTCCAGACAATCCATGTCTTCCGGACAATCCATGTCTTCCAAGAAGCTC-
CAAGTCTTCCAGTAAATCAAGTCTTCCAGCAAATCCAGTCTTCCAGCAAT-
TACTGGTCTTCCACCAAATCCAGATCTTCCAGGAAAATCCACGTCTTCCAGGAAATC-
CATGTCTTCCAATAATTTCAAGGTCTTCCATCAAATACAGATCTTCCAGCTAATC-
CATGTCTTCCAGAAAAATCTGTGTCTTCCACCAAATCCAAGTCTTCCAGTAAATC-
TAGTTCTTCCAGAAAAATCTAGATCTTCCAGTCAATCAGTGTCTTCCAGAAAGAAATC-
CAGGTCTTCCAGTCAATCAGTGTCTTCCAGAAAGAAATCCAGGTCTTCCAGTCAGT-
```



Plants

Arabidopsis *Arabidopsis thaliana* CircRNA: 39

Rice *Oryza sativa* CircRNA: 28

Wheat *Triticum aestivum* CircRNA: 62

Tomato *Solanum lycopersicum* CircRNA: 164

Cotton *Gossypium spp.* CircRNA: 280

Barley *Hordeum vulgare* CircRNA: 1

Kiwifruit *Actinidia deliciosa* CircRNA: 584

Animals

Human *Homo sapiens* CircRNA: 3799

Monkey *Macaca mulatta* CircRNA: 19

Rat *Rattus norvegicus* CircRNA: 506

Mouse *Mus musculus* CircRNA: 757

Chicken *Gallus gallus* CircRNA: 494

Pig *Sus scrofa* CircRNA: 131

Cattle *Bos taurus* CircRNA: 2

Fruit fly *Drosophila melanogaster* CircRNA: 264

Rabbit *Oryctolagus cuniculus* CircRNA: 159

Fig. 1 CircFunBase web interface and species

```
CAGTGTCTTCCAGAAAAATCTACGTCTTCCACCAAATCCAGGTCTTCCAGTCAATCCA-
CATCTTCCGGAAAAAATCCAGGTCTTCCAGCCAATATATGTCTTCCTGAAGATC-
CACGTCTTCCAGAAAATCCATGTCTTCCAGAAAATCCATGTCTTCCAGTAACCTCC-
CAGTCTTCCAGAAAATCCACGTCTTCCCAACAATCCAAGTCTTCCGGA-
TAATTTGGGTCTTCCTGAAAATCTACGTCTTCCAAAAAAGCCATGTCTTCCAGAAAATC-
CACATCTTCCAATGGCCTCCAGGTCTTCCAGACTATCCATGTCTTCCA-
GAAAATCCTTGTCTTCCCTTAAATCTATAGCTTCCAAAAAATCCGGGTCTTCCAG-
GAAATCCGTGTCTTCCAGCAAGTCCACGTCTTCCAACAAAGCCATGTCTTCCAGACTATC-
CATGTCTTCCAGAAAATCCTTGTCTTCCCTCAAATCCATAGCTTCCGAAAAATC-
CAGGTCTTCCAGGAAATCCGTGTCTTCCAGCAAATCCACGTCTTCCAACAAAGC-
CATGTCTTCCATCAAATTAATGTCTTCCAGCCTACTTGTGTCTTCCAACAAAGG-
TACGTCTTCCAACAAAGGTACGTCTTCCAACAAAGGTATGTCTTCCAACAAAGG-
TACGTCTTCCAGAAAATCCACGTCTTCCAACCAAGCCATGTCTTCCAGAAAATC-
CACGTCTTCCAGAAAATATATGTCTTCCAACTAAGCTACGTCTTCCAACAAATC-
CATGTCTTCCTATATCTCCAGGTCTTCCAGCATCTCCAGGGCTTCCAG-
CATCTGCTCGTCTTCCAACATCTCCACGTCTTCCAGCATCTCTGTGTCTTCCAGCATCTT-
CATGTCTTCCAACAACTACCCAGTCTTCCATCAACTGGCTCAATATCCATGTCTTC-
CAACGTCTCCAGTGTGCTGATCTTCTGACATTCAGGTCTTCCAGTGTCTGCAATATCCAG
```

3 Methods

Figure 2 shows the overview of CircFunBase. User guidelines for each module are introduced in the following sections.

3.1 Search Form

There are multiple ways to access data in CircFunBase. CircFunBase can be queried through two different search forms: (1) the simple search form allowing basic queries on fields, such as circRNA name; (2) the BLAST form permitting sequence queries on the whole circRNAs nucleotide databases.

3.1.1 Keyword Search

Users can visit the main search page using the Search link in the navigation bar, which locates at the top of page. CircRNAs can be retrieved by circRNA name, location, parental gene symbol and other keywords. Users can also select the interested species by clicking the red colored text. And autofill input form is implemented by typehead.js. Users can type part of keywords and a list of candidates are provided. Examples are available in the Examples

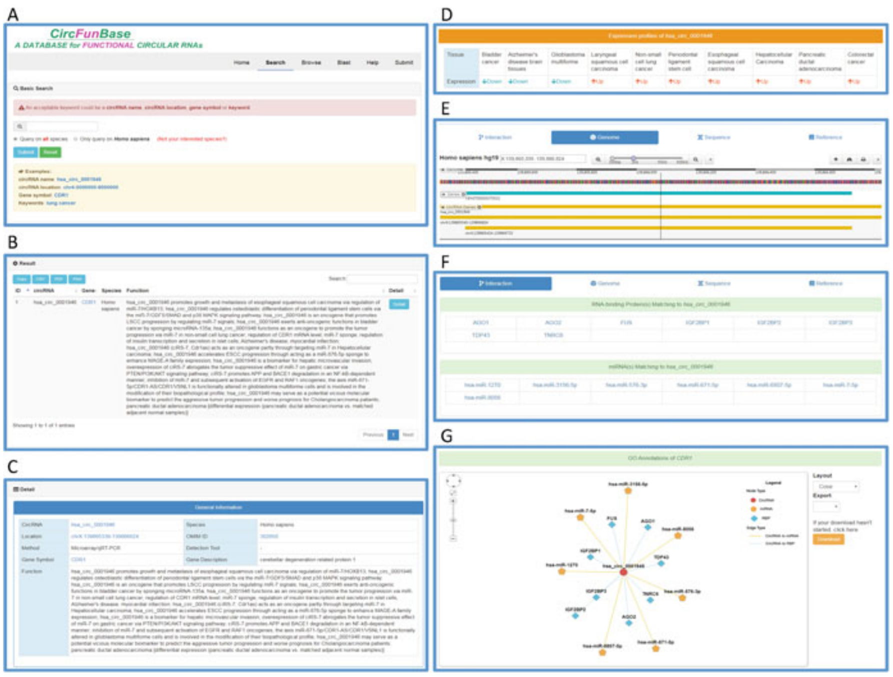


Fig. 2 Overview of CircFunBase

box. For instance, users can click the circRNA name has_-circ_0001946, then press the Submit button. Basic circRNA information table is listed in the Result panel, including circRNA name, parental gene, species, and functions. Clicking the Detail button can redirect to the detail page of circRNA.

Users can also copy or download the search result in CSV/PDF format. Results can also be sorted by columns, like circRNA name and parental gene. If no entry is available, the table will show "No data available in table" message.

3.1.2 BLAST Search

Rather than keyword-based search form, users can access the sequence search (BLAST search) page by clicking the BLAST link in the navigation bar. Users can paste query nucleotide sequence(s) or uploading file containing query sequence(s) in FASTA format. Next, choose databases to blast against, multiple selection is allowed. Then, users can specify the advanced parameters or skip this step by using the default parameters. Detailed parameters are listed by clicking the question mark.

For instance, paste the example hsa_circ_0001946 sequence into the query box and choose *Homo sapiens*, *Mus musculus* and *Rattus norvegicus* nucleotide databases. Then press BLAST button and results will be returned within a few seconds. The results consist of number of hits and details of hits.

Our example sequence returns hits that matches to two circRNAs except itself; they are chrX:139865340–139,866,824 and mmu_circ_0001878 (Fig. 3). chrX:139865340–139,866,824 is actually located in the almost same position with hsa_circ_0001946 while its start position is 1 nt apart from hsa_circ_0001946. We believe that they are actually the same sequence but with two names. mmu_circ_0001878 has 72.3% identities in *Mus musculus*, which suggests hsa_circ_0001946 may be conserved in human and mouse and may have functional similarities to some extent.

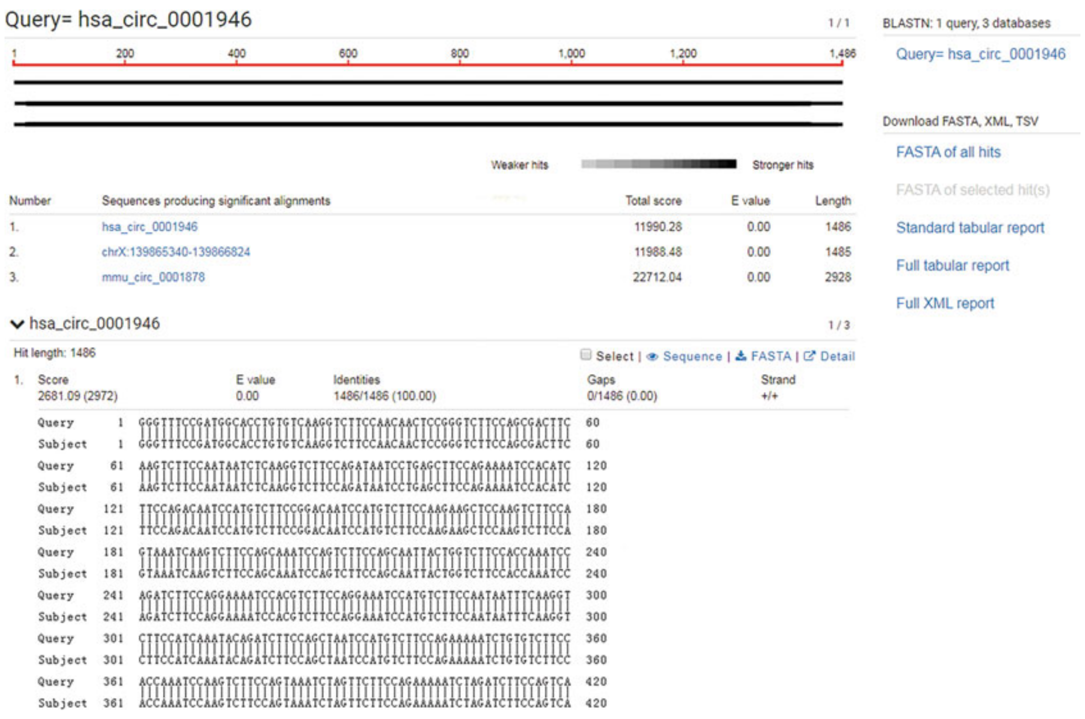
After running a search, users can redirect to the corresponding detail page by clicking the Detail button. Users can also select interested hits and download the BLAST results by clicking the buttons on the right in FASTA, tabular, or XML format.

3.2 Browse and Download

As an alternative to directed searching, users can click the Browse link in the navigation toolbar to view all circRNAs listed under different species. Users can click the icon of species or phylogenetic tree of species to get all circRNAs in the specific organism. Users can also download all the circRNAs entries via download section in Help page.

3.3 Detail

The Detail page documents the full information of functional circRNA entry, which can be accessed from the results of Search or Browse action. The detailed information of circRNA consists of several parts. First part is the basic information of circRNA, including circRNA name, detection method, and circRNA function.


Query= hsa_circ_0001946 1 / 1

BLASTN: 1 query, 3 databases

Query= hsa_circ_0001946

Download FASTA, XML, TSV

FASTA of all hits

FASTA of selected hit(s)

Standard tabular report

Full tabular report

Full XML report

Number	Sequences producing significant alignments	Total score	E value	Length
1.	hsa_circ_0001946	11990.28	0.00	1486
2.	chrX:139865340-139866824	11988.48	0.00	1485
3.	mmu_circ_0001878	22712.04	0.00	2928

hsa_circ_0001946 1 / 3

Hit length: 1486

	Score	E value	Identities	Gaps	Strand
1.	2681.09 (2972)	0.00	1486/1486 (100.00)	0/1486 (0.00)	+/+

```
Query      1  GGGTTTCCGATGGCACCTGTGTCAAGGTCTTCCAACAACTCCGGGTCTTCCAGCGACTTC  60
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subject    1  GGGTTTCCGATGGCACCTGTGTCAAGGTCTTCCAACAACTCCGGGTCTTCCAGCGACTTC  60
Query     61  AAGTCTTCCAATAATCTCAAGGTCTTCCAGATAATCCTGAGCTTCCAGAAAATCCACATC  120
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subject   61  AAGTCTTCCAATAATCTCAAGGTCTTCCAGATAATCCTGAGCTTCCAGAAAATCCACATC  120
Query    121  TTCCAGACAATCCATGTCTTCCGGACAATCCATGTCTTCCAAGAAGCTCCAAGTCTTCCA  180
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subject  121  TTCCAGACAATCCATGTCTTCCGGACAATCCATGTCTTCCAAGAAGCTCCAAGTCTTCCA  180
Query    181  GTAAATCAAGTCTTCCAGCAAATCCAGTCTTCCAGCAATTACTGGTCTTCCACCAAATCC  240
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subject  181  GTAAATCAAGTCTTCCAGCAAATCCAGTCTTCCAGCAATTACTGGTCTTCCACCAAATCC  240
Query    241  AGATCTTCCAGGAAAATCCACGTCTTCCAGGAAATCCATGTCTTCCAATAATTTCAAGGT  300
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subject  241  AGATCTTCCAGGAAAATCCACGTCTTCCAGGAAATCCATGTCTTCCAATAATTTCAAGGT  300
Query    301  CTTCCATCAAATACAGATCTTCCAGCTAATCCATGTCTTCCAGAAAAATCTGTGTCTTCC  360
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subject  301  CTTCCATCAAATACAGATCTTCCAGCTAATCCATGTCTTCCAGAAAAATCTGTGTCTTCC  360
Query    361  ACCAAATCCAAGTCTTCCAGTAAATCTAGTTCTTCCAGAAAAATCTAGATCTTCCAGTCA  420
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Subject  361  ACCAAATCCAAGTCTTCCAGTAAATCTAGTTCTTCCAGAAAAATCTAGATCTTCCAGTCA  420
```

Fig. 3 BLAST results of hsa_circ_0001946

1. circRNA name, mostly based on the name that literature used, if it is human circRNA, hyperlink to circBase is provided.
2. Species, where the circRNA is identified.
3. Location, including chromosome, start position, and end position. If it is from human, hyperlink to USCS genome browser is offered.
4. Parental gene, parent gene of circRNA with NCBI Gene link. If it's from tomato, the Ensembl Gene link is provided.
5. Parental gene function.
6. Gene OMIM, circRNA parental gene associated OMIM ID.
7. Detection tool, bioinformatics tools to identify circRNAs.
8. Method, experimental validation methods, including RNA-seq, microarray, qRT-PCR, and northern blotting.
9. Function, description of circRNA function.
10. Expression pattern, circRNA may display different expression patterns in different issues, which can help us understand complexity of RNA world. The red colored arrow means upregulated expression; the green colored arrow means downregulated expression.

The second part is the visualization part, which includes circRNA associated interaction network visualization and genome browser. The network shows the interaction of circRNA with miRNA and circRNA with RBP. The red circle indicates the circRNA, the blue diamond indicates the RNA binding protein and the orange pentagon symbol indicates the miRNA. The interaction information is also listed in tables. For example, except for the miR-7 and miR-671 that have been validated to be interacted with has_circ_0001946, other miRNA–circRNA interactions are also predicted to give an insight to functions of circRNA. Users can select the layout style and download the image by choosing the export format.

Genome browser can be accessed by clicking the Genome tab. Users can zoom in and out to view the context of circRNA. Users can relocate by enter the genomic location. Users can also change the track style and download the image through the toolbar provided by biodalliance.js.

The last part is sequences of circRNAs and references. The sequences of circRNAs are extracted through scripts, based on hypothesis that circRNAs share the canonical splice sites with linear transcripts. The references that identified and validated the circRNA can be linked to the PubMed to view the full text.

3.4 API

We provide an API interface to enable users to fetch data programmatically with machine-read format. There are three kinds of services, circRNA detailed information retrieval, miRNA interaction information retrieval, and RNA binding proteins interaction information retrieval. Take Python for example:

```
import requests
url = "http://bis.zju.edu.cn/CircFunBase/Api/Public/circfuna-
pi/?service=CircRNA.getinfo&circRNAname=%s"
circName = "hsa_circ_0001946"
res = requests.get(url % circName )
print(res.json())
```

Full documentation and field description are available via our help pages.

3.5 Submission

Directed submission by the researchers is supported. Users can submit one circRNA entry by filling the form and can also submit a file included multiple entries. Subsequent data correction is maintained by our administrators. We would appreciate that users can assist us maintaining and updating CircFunBase.

References

1. Wilusz JE (2018) A 360° view of circular RNAs: from biogenesis to functions. Wiley Interdiscip Rev RNA 9
2. Wang PL, Bao Y, Yee MC et al (2014) Circular RNA is expressed across the eukaryotic tree of life. PLoS One 9
3. Ye C-Y, Chen L, Liu C et al (2015) Widespread noncoding circular RNAs in plants. New Phytol 208:88–95
4. Salzman J, Gawad C, Wang PL et al (2012) Circular RNAs are the predominant transcript isoform from hundreds of human genes in diverse cell types. PLoS One 7
5. Vo JN, Cieslik M, Zhang Y et al (2019) The landscape of circular RNA in cancer. Cell 176:869–881
6. Meng X, Li X, Zhang P et al (2017) Circular RNA: an emerging key player in RNA world. Brief Bioinform 18:547–557
7. Piwecka M, Glažar P, Hernandez-Miranda LR et al (2017) Loss of a mammalian circular RNA locus causes miRNA deregulation and affects brain function. Science 357
8. Ashwal-Fluss R, Meyer M, Pamudurti NR et al (2014) CircRNA biogenesis competes with pre-mRNA splicing. Mol Cell 56:55–66
9. Du WW, Fang L, Yang W et al (2017) Induction of tumor apoptosis through a circular RNA enhancing Foxo3 activity. Cell Death Differ 24:357–370
10. Legnini I, Di Timoteo G, Rossi F et al (2017) Circ-ZNF609 is a circular RNA that can be translated and functions in myogenesis. Mol Cell 66:22–37
11. Patop IL, Wüst S, Kadener S (2019) Past, present, and future of circ RNA s. EMBO J 38
12. Kristensen LS, Andersen MS, Stagsted LVW et al (2019) The biogenesis, biology and characterization of circular RNAs. Nat Rev Genet 20:675–691
13. Gao Y, Zhang J, Zhao F (2018) Circular RNA identification based on multiple seed matching. Brief Bioinform 19:803–810
14. Szabo L, Morey R, Palpant NJ et al (2015) Statistically based splicing detection reveals neural enrichment and tissue-specific induction of circular RNA during human fetal development. Genome Biol 16
15. Li X, Chu C, Pei J et al (2018) CircMarker: a fast and accurate algorithm for circular RNA detection. BMC Genomics 19:572
16. Glažar P, Papavasileiou P, Rajewsky N (2014) CircBase: a database for circular RNAs. RNA 20:1666–1670
17. Ji P, Wu W, Chen S et al (2019) Expanded expression landscape and prioritization of circular RNAs in mammals. Cell Rep 26:3444–3460
18. Zhang P, Meng X, Chen H et al (2017) PlantCircNet: a database for plant circRNA-miRNA-mRNA regulatory networks. Database 2017
19. Down TA, Piipari M, Hubbard TJP (2011) Dalliance: interactive genome viewing on the web. Bioinformatics 27:889–890
20. Priyam A, Woodcroft BJ, Rai V et al (2019) Sequenceserver: a modern graphical user Interface for custom BLAST databases. Mol Biol Evol 36:2922–2924
21. Ghosal S, Das S, Sen R et al (2013) Circ2Traits: a comprehensive database for circular RNA potentially associated with disease and traits. Front Genet 4
22. Dudekula DB, Panda AC, Grammatikakis I et al (2016) Circinteractome: a web tool for exploring circular RNAs and their interacting proteins and microRNAs. RNA Biol 13:34–42
23. Liu YC, Li JR, Sun CH et al (2016) CircNet: a database of circular RNAs derived from transcriptome sequencing data. Nucleic Acids Res 44:D209–D215
24. Chen X, Han P, Zhou T et al (2016) CircRNADb: a comprehensive database for human circular RNAs with protein-coding annotations. Sci Rep 6:34985
25. Xia S, Feng J, Lei L et al (2017) Comprehensive characterization of tissue-specific circular RNAs in the human and mouse genomes. Brief Bioinform 18:984–992
26. Chu Q, Zhang X, Zhu X et al (2017) PlantcircBase: a database for plant circular RNAs. Mol. Plant 10:1126–1128
27. Ye J, Wang L, Li S et al (2019) AtCircDB: a tissue-specific database for Arabidopsis circular RNAs. Brief Bioinform 20:58–65
28. Xia S, Feng J, Chen K et al (2018) CSCD: a database for cancer-specific circular RNAs. Nucleic Acids Res 46:D925–D929
29. Li S, Li Y, Chen B et al (2018) ExoRBase: a database of circRNA, lncRNA and mRNA in human blood exosomes. Nucleic Acids Res 46: D106–D112
30. Fan C, Lei X, Fang Z et al (2018) CircR2Disease: a manually curated database for experimentally supported circular RNAs associated with various diseases. Database 2018
31. Dong R, Ma XK, Li GW, Yang L (2018) CIRCpedia v2: an updated database for

comprehensive circular RNA annotation and expression comparison. Genom Proteom Bioinf 16:226–233

32. Tang Z, Li X, Zhao J et al (2019) TRCirc: a resource for transcriptional regulation information of circRNAs. Brief Bioinform 20:2327–2333

33. Meng X, Hu D, Zhang P et al (2019) CircFunBase: a database for functional circular RNAs. Database 2019

Chapter 25

Databases for RNA Editing Collections

Claudio Lo Giudice, Luigi Mansi, Graziano Pesole, and Ernesto Picardi

Abstract

A-to-I RNA editing in humans plays a relevant role since it can influence gene expression and increase proteome diversity. In addition, its deregulation has been linked to a variety of human diseases, including neurological disorders and cancer.

In the last decade, massive transcriptome sequencing through the RNAseq technology has dramatically improved the investigation of RNA editing at single nucleotide resolution. Nowadays, different bioinformatics resources to discover and/or collect A-to-I events have been released. Hereafter, we initially provide an overview of the state-of-the-art RNA editing databases and, then, we focus on REDIportal, the largest collection of A-to-I events with more than 4.5 million sites from 2660 humans GTEx samples.

Key words RNA editing, A-to-I editing, Transcriptomics, RNAseq, Database, REDIportal

1 Introduction

RNA editing is a global term to describe a set of co-/posttranscriptional modifications occurring in a wide range of organisms that lead to differentiate transcripts nucleotide sequences from their corresponding genomic templates [1]. These modifications affect both coding and noncoding regions, thus contributing to increase the transcriptome [1, 2] and proteome diversity [3].

In mammals and, in particular, in humans, there are two main types of RNA editing events: adenine-to-inosine (A-to-I) and cytosine-to-uracil (C-to-U).

As inosine is generally interpreted as guanosine (G) by translation and splicing machineries, A-to-I editing is often referred to as A-to-G editing. While A-to-I accounts for >95% of all RNA editing events in the human genome [4], only few transcripts have been described undergoing C-to-U editing [5]. Biologically, RNA editing has a pivotal role in preserving the cellular homeostasis, as demonstrated by the fact that its deregulation in humans seems to

Ernesto Picardi (ed.), *RNA Bioinformatics*, Methods in Molecular Biology, vol. 2284, https://doi.org/10.1007/978-1-0716-1307-8_25,
© Springer Science+Business Media, LLC, part of Springer Nature 2021

Table 1
Database name, organism(s) stored, last update, PMID (PubMed IDentifier)

Database	Organism(s)	Last update	PMID
Darned	Human, fruit fly (dm3), mouse	2013	20547637
RADAR	Human (hg19), fruit fly (dm3), mouse (mm9)	2014	24163250
REDIdb	Plants organellar genomes	2018	29696033
REDIportal	Human, mouse	2019	27587585
PED	Plants	2019	30364952
EDK	Editome disease knowledgebase	2018	30357418
TCEA	Cancer editome	2018	31015229

be involved in aging [6], neurological [7, 8], autoimmune [9], cardiovascular diseases [10], and cancer [11].

Novel RNA editing events can be easily discovered by comparing cDNA sequences with their corresponding genomic loci, followed by the identification of A-to-G mismatches [12]. In order to minimize false events arising from somatic mutations, the gold standard is to analyze both DNA and RNA data from the same individual [13].

Thanks to the great advances in sequencing technologies, thousands of RNA editing changes have been detected and collected in freely available specialized databases, whose main characteristics are reported in Table 1.

While REDIdb [14] and PED [15] are devoted to plants organellar editomes, RADAR [16] and DARNED [17] provide information on A-to-I changes for human, mouse, and fruit fly.

DARNED has not been updated since 2013 and does not include editing levels, while RADAR, which has been dismissed in 2019, annotates A-to-I events in the same organisms contained by DARNED with editing levels available for 38% of stored positions only. DARNED content is based on a limited number of RNAseq samples, most of them deriving from LCL cell lines, that are thought to be not optimal for RNA editing studies [18].

Emerging evidences indicate that A-to-I editing is also present in fungi where is involved in the sexual reproduction [19, 20]. A specialized database of fungal A-to-I RNA editing, housing a collection of A-to-I editing sites in six filamentous fungal species together with extensive annotations for each editing site, has been recently released [21].

The proven association between editome alterations and human diseases has also led to the the development of specific databases integrating molecular and clinical data. In this sense an example is offered by the Editome Disease Knowledgebase (EDK)

[22], a manually curated database of RNA editing events in mRNAs, miRNAs, lncRNAs, viruses, and RNA editing enzymes, known from the literature to be associated with human diseases. EDK integrates data on diseases from over 200 publications and biological information about each editing event from RADAR [16] and REDIportal [23]. Another relevant clinical database is The Cancer RNA Editome Atlas (TCEA) [24] that is specifically devoted to the exploration of RNA editing in cancer. TCEA integrates RNA editing events with cancer stage or survival data, allowing clinician to better link editing events with tumor onset/ progression. In TCEA as well as in EDK, RNA editing annotations derive from REDIportal.

Historically, the RNA Editing ATLAS, comprising more than three millions of A-to-I events identified in six tissues from three healthy individuals, was the first human inosinome atlas to be published [25]. To date, REDIportal, representing an extension of the RNA Editing ATLAS, is the only active resource answering functional questions about A-to-I RNA editing, enabling the inspection and browsing of editing levels in a variety of human samples, tissues, and body sites. Indeed, it collects more than 4.5 millions of A-to-I events in 55 body sites of 150 healthy individuals from the GTEx project [26].

In REDIportal, RNA Editing sites can be searched by genomic region or gene name. Query results are reported in sortable and downloadable tables including relevant characteristics of resulting events such as their genomic context, the reference and edited nucleotide, the strand, the number of edited samples, the potential amino acid change and the conservation across vertebrates. Some table cells are colored and interactive with hyperlinks to external resources.

Additionally, REDIportal provides coverage values per site at both RNA (by RNAseq data) and DNA (by WGS data) level as well as the RNA editing levels.

In the following section, we describe the REDIportal database and provide guidelines to efficiently query it.

2 Materials

The use of REDIportal does not require specific bioinformatics skills. Indeed, results are organized in tables for easy and intuitive download and visualization. REDIportal can be reached at http://srv00.recas.ba.infn.it/atlas/.

2.1 Required Software

Standard desktop or laptop computer with a stable Internet connection and an updated browser (e.g., Internet Explorer, Chrome, Firefox, or Safari).

To use the REDIportal API, the Curl library is needed. For Windows and UNIX-like operating systems, it can be easily downloaded from https://curl.haxx.se/download.html.

3 Methods

3.1 Hands-on: REDIportal Web Interface

Searching into REDIportal is quite straightforward and also users with no bioinformatics skills can perform extensive searches across the database (Fig. 1).

RNA editing sites are classified according to their genomic positions and can be retrieved providing a genomic locus ("Genomic Region" field) or a known gene symbol ("Gene Name" field). Both fields are mutually exclusive (Fig. 2). As an example, in Fig. 2 we show how to search for A-to-I changes in one of the genes coding for subunits of glutamate receptors.

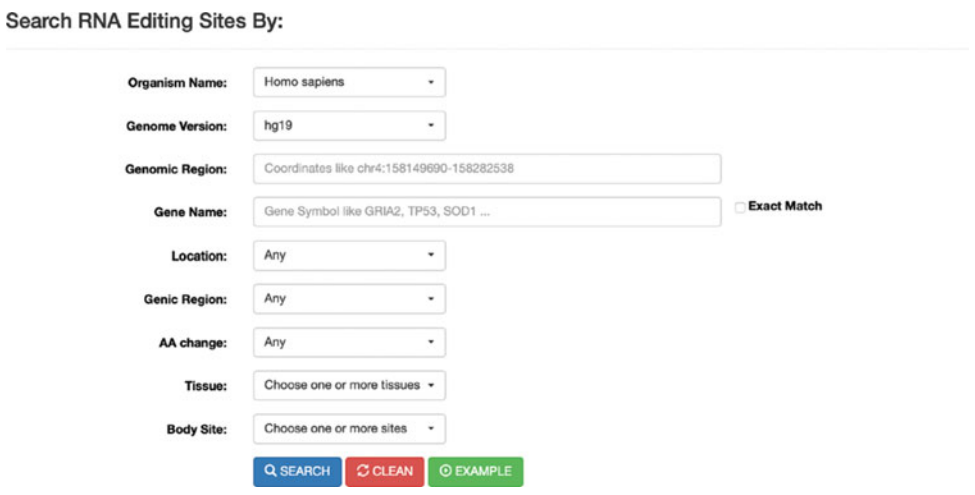
Furthermore, it is possible to refine the original query by using additional criteria:

– "**Location**," which allows the selection of RNA editing sites residing in Alu elements (ALU) or in repetitive elements non-Alu (REP) or in nonrepetitive regions (NONREP).

– "**Genic Region**," which allows the selection of RNA editing sites residing in regions such as untranslated regions (UTR) or intronic regions or coding/noncoding exons or intergenic regions (following the ANNOVAR [27] nomenclature).

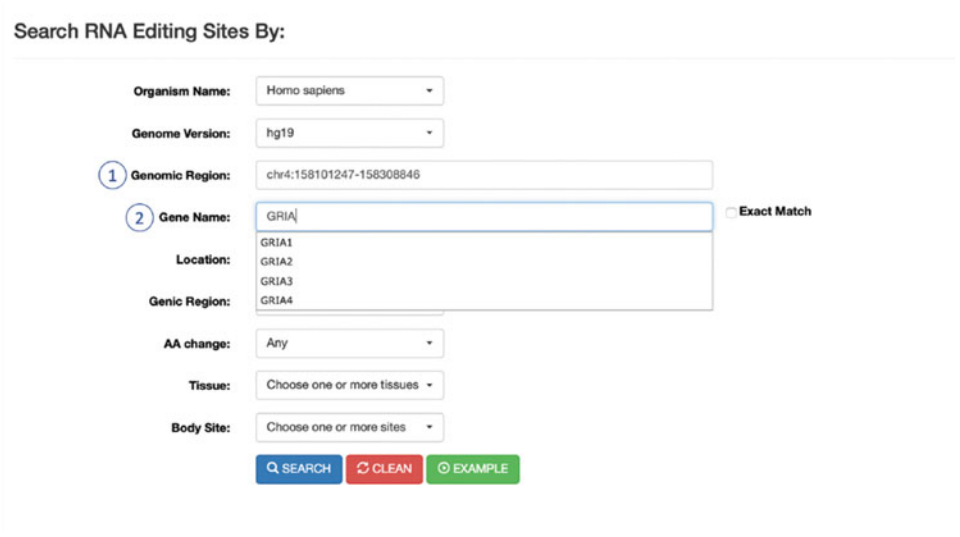
– "**AA change**," which allows the selection of RNA editing sites residing in protein-coding regions and affecting codon integrity.

– "**Tissue**," which allows the selection of RNA editing sites residing in specific human tissues. More than one tissue can be selected for each search. Tissue names are according to GTEx [26].


Search RNA Editing Sites By:

Organism Name:	Homo sapiens
Genome Version:	hg19
Genomic Region:	Coordinates like chr4:158149690-158282538
Gene Name:	Gene Symbol like GRIA2, TP53, SOD1 ... ☐ Exact Match
Location:	Any
Genic Region:	Any
AA change:	Any
Tissue:	Choose one or more tissues
Body Site:	Choose one or more sites

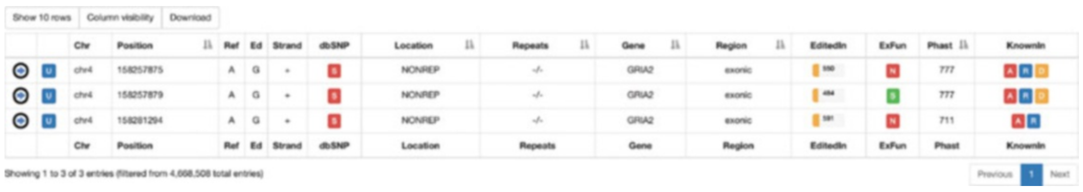
Fig. 1 REDIportal search page



Search RNA Editing Sites By:

Organism Name:	Homo sapiens
Genome Version:	hg19
1 Genomic Region:	chr4:158101247-158308846
2 Gene Name:	GRIA
Location:	
Genic Region:	
AA change:	Any
Tissue:	Choose one or more tissues
Body Site:	Choose one or more sites

Fig. 2 (1) Genomic loci can be interrogated entering chromosome coordinates in the format Chr:start-end (for example chr4:158101247–158308846). (2) RNA editing events in known genes can be retrieved entering the gene symbol in the "Gene Name" field. This field embeds an autocomplete function to facilitate the selection of the right gene



		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
	U	chr4	158257875	A	G	+	S	NONREP	-/-	GRIA2	exonic	550	N	777	A R D
	U	chr4	158257879	A	G	+	S	NONREP	-/-	GRIA2	exonic	494	S	777	A R D
	U	chr4	158281294	A	G	+	S	NONREP	-/-	GRIA2	exonic	591	N	711	A R
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn

Showing 1 to 3 of 3 entries (filtered from 4,668,508 total entries)

Fig. 3 Query results

– "**Body Site**," which allows the selection of RNA editing sites residing in specific human body sites. More than one body site can be selected for each search. Body site names are according to GTEx [26].

Once a search has been performed, results will be displayed in a table (Fig. 3).

Where:

– "**Ref**" is the nucleotide on Reference.

– "**Ed**" is the Edited Nucleotide.

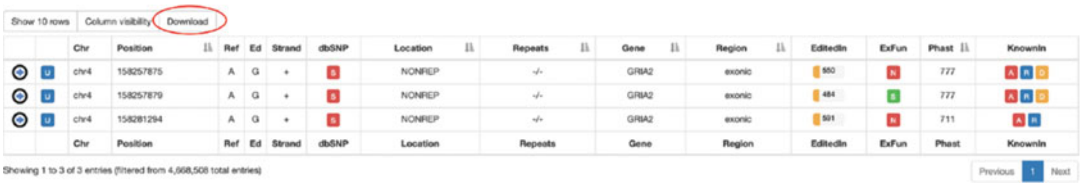
– "**dbSNP**" indicates the presence of a SNP in dbSNP (green flag for the match).

– "**Location**" indicates whether RNA Editing is in repetitive or nonrepetitive regions.

– "**Repeats**" indicates the class and family of repeat including the RNA editing position.

– “**Gene**” is the Gene Symbol according to Gencode [28].
– “**Region**” is the Genic Region (according to ANNOVAR).
– “**EditedIn**” is the number of samples in which the specific position appears to be edited.
– “**ExFun**” is the exonic function and is limited to synonymous and nonsynonymous positions. A colored flag is used to indicate if a site is synonymous (green) or nonsynonymous (red).
– “**Phast**” is the PhastCons conservation score calculated for multiple alignments of 45 vertebrate genomes to the human genome. It ranges from 0 (no conservation) to 1000 (max conservation). Values derive from UCSC phastCons46way table [29].
– “**KnownIn**” indicates if a site is present in other known databases (A: ATLAS, R: RADAR, D: DARNED).

It is also possible to download the information in the table by clicking on the appropriate button (Fig. 4) and choosing the columns of interest (Fig. 5). An example of a downloaded table is reported in Fig. 6.

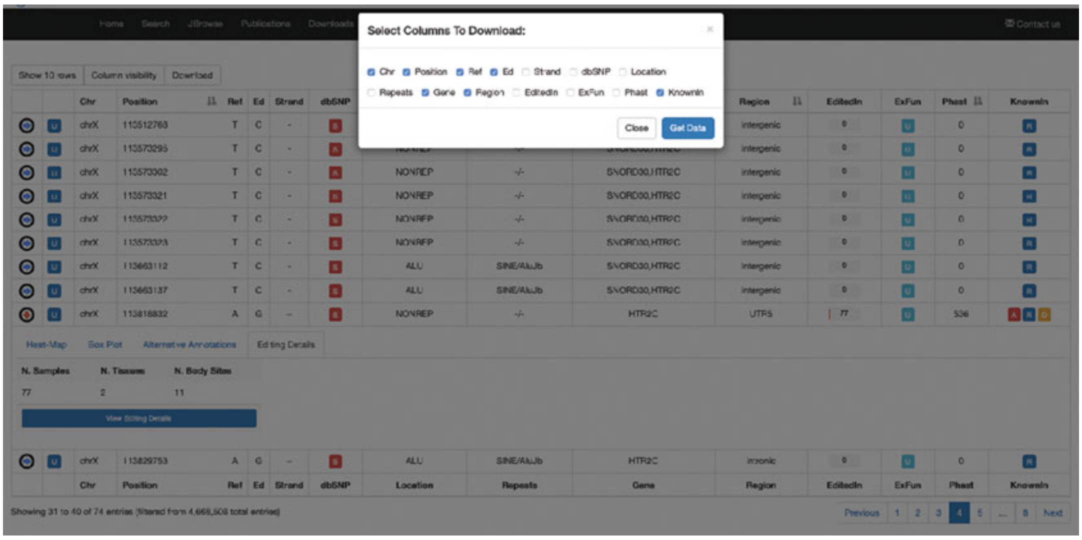


		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
⊕	U	chr4	158257875	A	G	+	S	NONREP	-/-	GRIA2	exonic	550	N	777	A R D
⊕	U	chr4	158257879	A	G	+	S	NONREP	-/-	GRIA2	exonic	484	S	777	A R D
⊕	U	chr4	158281294	A	G	+	S	NONREP	-/-	GRIA2	exonic	501	N	711	A R
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn

Showing 1 to 3 of 3 entries (filtered from 4,668,508 total entries)

Previous 1 Next

Fig. 4 Download button in red circle



Select Columns To Download:

[✓ Chr] [✓ Position] [✓ Ref] [✓ Ed] Strand dbSNP Location
Repeats [✓ Gene] [✓ Region] EditedIn ExFun Phast [✓ KnownIn]

		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
⊕	U	chrX	113512763	T	C	-	S				intergenic	0	U	0	R
⊕	U	chrX	113573295	T	C	-	S				intergenic	0	U	0	R
⊕	U	chrX	113573302	T	C	-	S	NONREP	-/-	SNORD30,HTR2C	intergenic	0	U	0	R
⊕	U	chrX	113573321	T	C	-	S	NONREP	-/-	SNORD30,HTR2C	intergenic	0	U	0	R
⊕	U	chrX	113573322	T	C	-	S	NONREP	-/-	SNORD30,HTR2C	intergenic	0	U	0	R
⊕	U	chrX	113573323	T	C	-	S	NONREP	-/-	SNORD30,HTR2C	intergenic	0	U	0	R
⊕	U	chrX	113663112	T	C	-	S	ALU	SINE/AluJb	SNORD30,HTR2C	intergenic	0	U	0	R
⊕	U	chrX	113663137	T	C	-	S	ALU	SINE/AluJb	SNORD30,HTR2C	intergenic	0	U	0	R
⊖	U	chrX	113818832	A	G	-	S	NONREP	-/-	HTR2C	UTR5	77	U	536	A R D

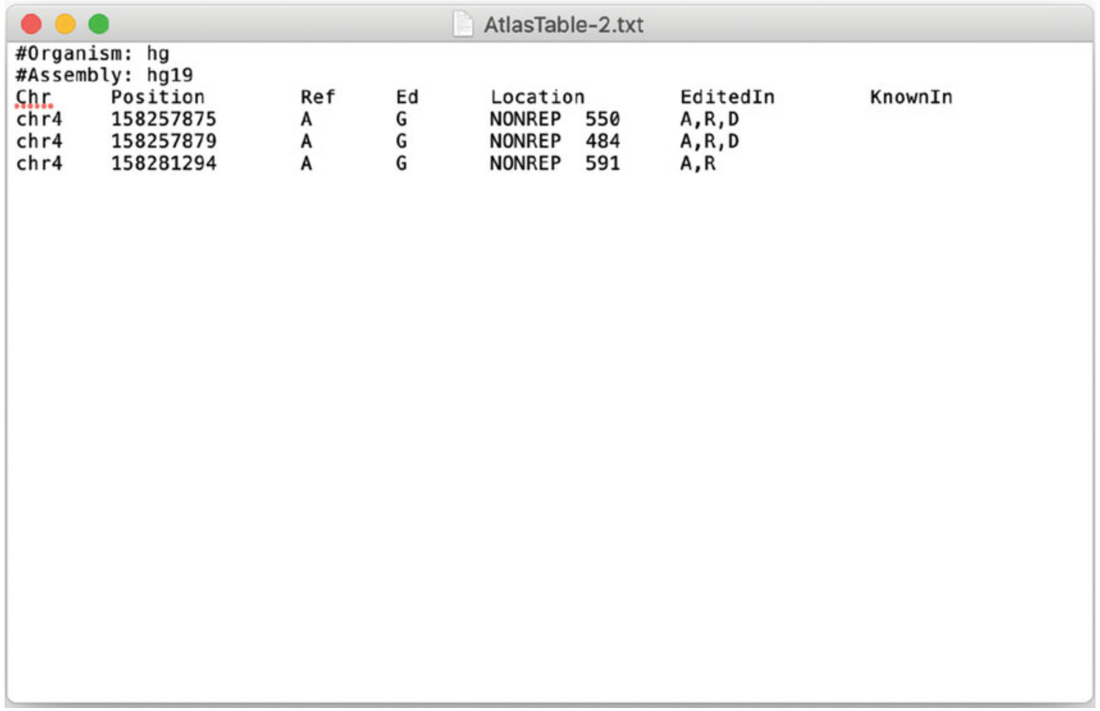
N. Samples	N. Tissues	N. Body Sites
77	2	11

		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
⊕	U	chrX	113829753	A	G	-	S	ALU	SINE/AluJb	HTR2C	intronic	0	U	0	R
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn

Showing 31 to 40 of 74 entries (filtered from 4,668,508 total entries)

Previous 1 2 3 4 5 ... 8 Next

Fig. 5 Download section

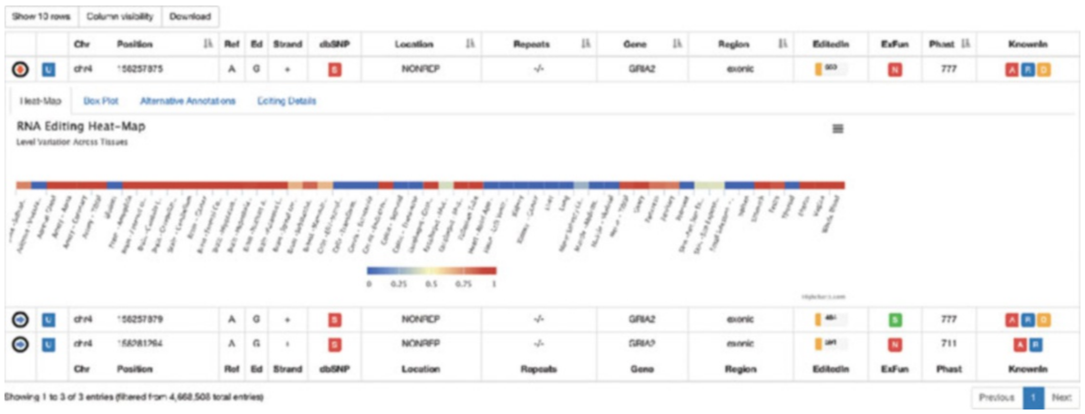


```
#Organism: hg
#Assembly: hg19
Chr      Position        Ref     Ed      Location         EditedIn        KnownIn
chr4     158257875       A       G       NONREP   550     A,R,D
chr4     158257879       A       G       NONREP   484     A,R,D
chr4     158281294       A       G       NONREP   591     A,R
```

Fig. 6 Downloaded table in plain text format. Column names are those selected in the download section

For each position, REDIportal provides additional info that can be displayed by clicking the blue arrow in the first column. This will cause the opening of four tabs:

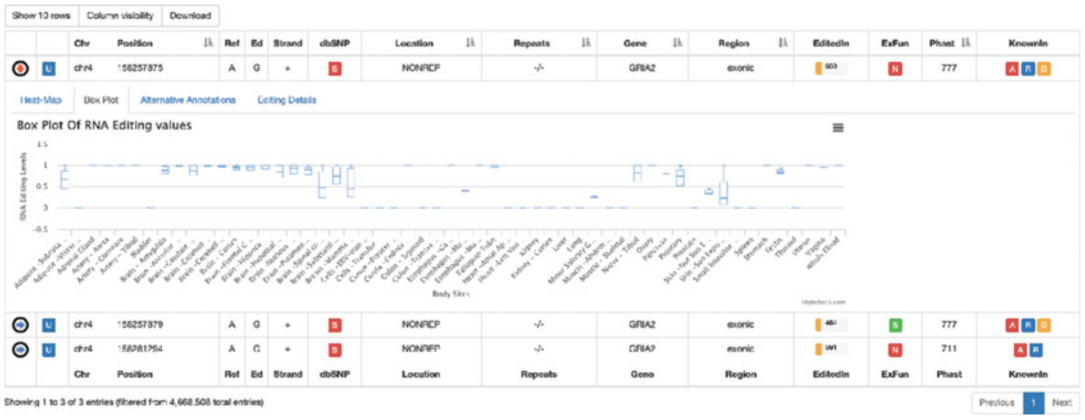
– "**Heat-Map**" (Fig. 7) displays an RNA Editing heat-map in which the mean editing level per body site is reported.
– "**Box Plot**" (Fig. 8) displays RNA Editing levels per each body site by means of box plots.
– "**Alternative Annotations**" (Fig. 9) displays a table with gene/transcript annotations from RefSeq database and UCSC KnownGene table.
– "**Editing Details**" (Fig. 10) displays the number of samples, tissues and body sites in which the position appears to be edited. Clicking on the "View Editing Details" button will cause the opening of new windows with a table including editing levels for each experiment. (Fig. 11).



		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
	U	chr4	158257875	A	G	+	S	NONREP	-/-	GRIA2	exonic	[illegible]	N	777	A R D
	U	chr4	158257879	A	G	+	S	NONREP	-/-	GRIA2	exonic	[illegible]	S	777	A R D
	U	chr4	158281294	A	G	+	S	NONREP	-/-	GRIA2	exonic	[illegible]	N	711	A R
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn

Showing 1 to 3 of 3 entries (filtered from 4,668,508 total entries)

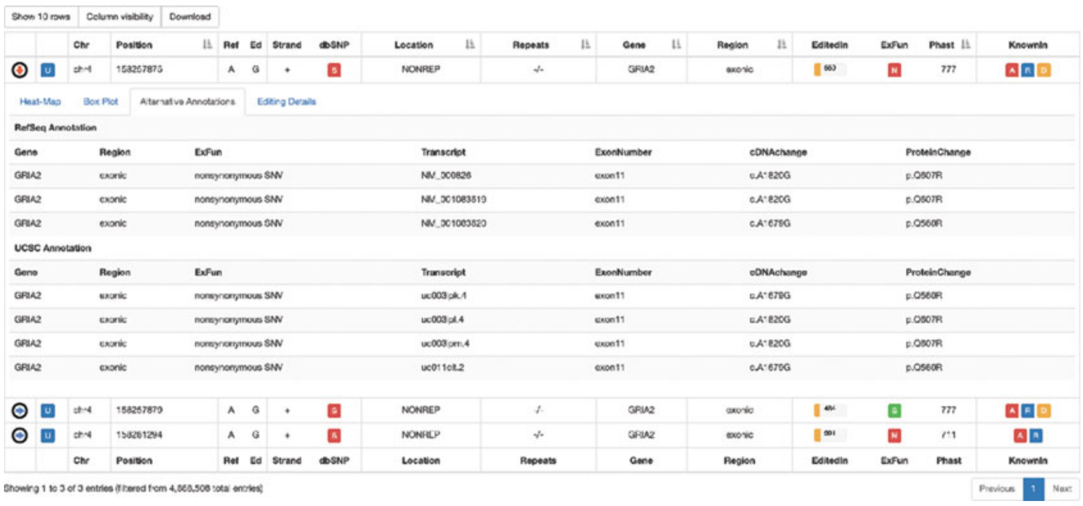
Fig. 7 Heat-map. Mouse over each body site to open a tooltip showing the average editing level



		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
	U	chr4	158257875	A	G	+	S	NONREP	-/-	GRIA2	exonic	[illegible]	N	777	A R D
	U	chr4	158257879	A	G	+	S	NONREP	-/-	GRIA2	exonic	[illegible]	S	777	A R D
	U	chr4	158281294	A	G	+	S	NONREP	-/-	GRIA2	exonic	[illegible]	N	711	A R
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn

Showing 1 to 3 of 3 entries (filtered from 4,668,508 total entries)

Fig. 8 Box PLot. Relevant values are available by mousing over each box plot



		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
	U	chr4	158257875	A	G	+	S	NONREP	-/-	GRIA2	exonic	550	N	777	A R D

RefSeq Annotation

Gene	Region	ExFun	Transcript	ExonNumber	cDNAchange	ProteinChange
GRIA2	exonic	nonsynonymous SNV	NM_000826	exon11	c.A1820G	p.Q607R
GRIA2	exonic	nonsynonymous SNV	NM_001083619	exon11	c.A1820G	p.Q607R
GRIA2	exonic	nonsynonymous SNV	NM_001083620	exon11	c.A1679G	p.Q560R

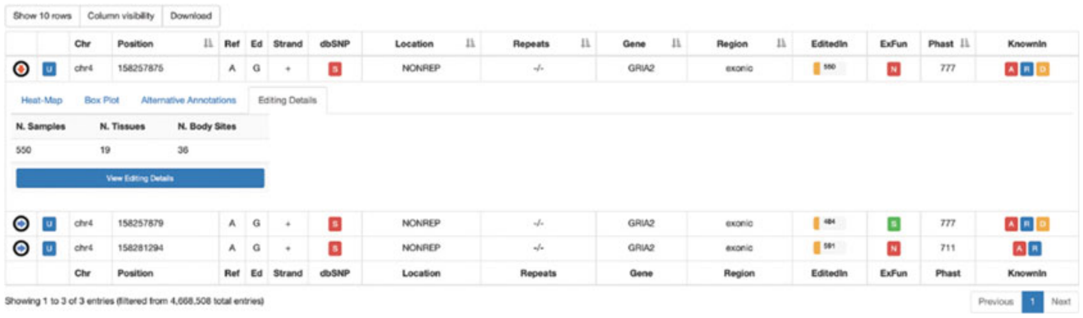
UCSC Annotation

Gene	Region	ExFun	Transcript	ExonNumber	cDNAchange	ProteinChange
GRIA2	exonic	nonsynonymous SNV	uc003ipk.4	exon11	c.A1679G	p.Q560R
GRIA2	exonic	nonsynonymous SNV	uc003ipl.4	exon11	c.A1820G	p.Q607R
GRIA2	exonic	nonsynonymous SNV	uc003ipm.4	exon11	c.A1820G	p.Q607R
GRIA2	exonic	nonsynonymous SNV	uc011cll.2	exon11	c.A1679G	p.Q560R

		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
	U	chr4	158257879	A	G	+	S	NONREP	-/-	GRIA2	exonic	484	S	777	A R D
	U	chr4	158281294	A	G	+	S	NONREP	-/-	GRIA2	exonic	591	N	711	A R
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn

Showing 1 to 3 of 3 entries (filtered from 4,668,508 total entries)

Fig. 9 Alternative annotations



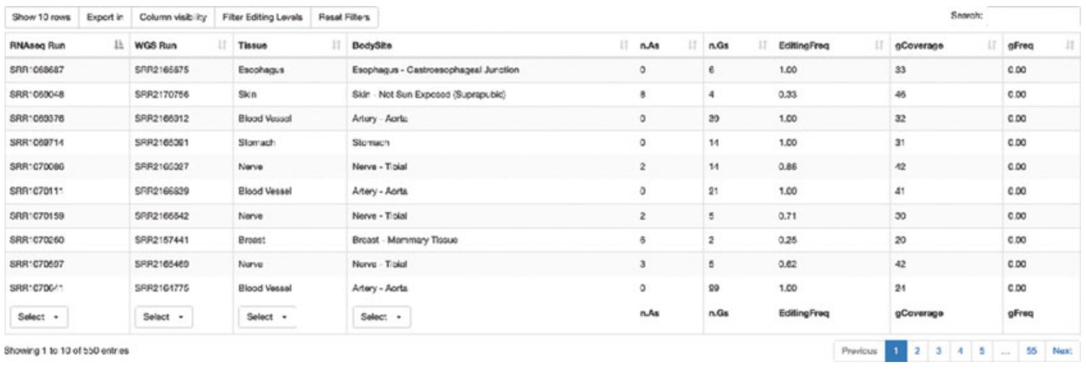
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
	U	chr4	158257875	A	G	+	S	NONREP	-/-	GRIA2	exonic	550	N	777	A R D

N. Samples	N. Tissues	N. Body Sites
550	19	36

		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn
	U	chr4	158257879	A	G	+	S	NONREP	-/-	GRIA2	exonic	484	S	777	A R D
	U	chr4	158281294	A	G	+	S	NONREP	-/-	GRIA2	exonic	591	N	711	A R
		Chr	Position	Ref	Ed	Strand	dbSNP	Location	Repeats	Gene	Region	EditedIn	ExFun	Phast	KnownIn

Showing 1 to 3 of 3 entries (filtered from 4,668,508 total entries)

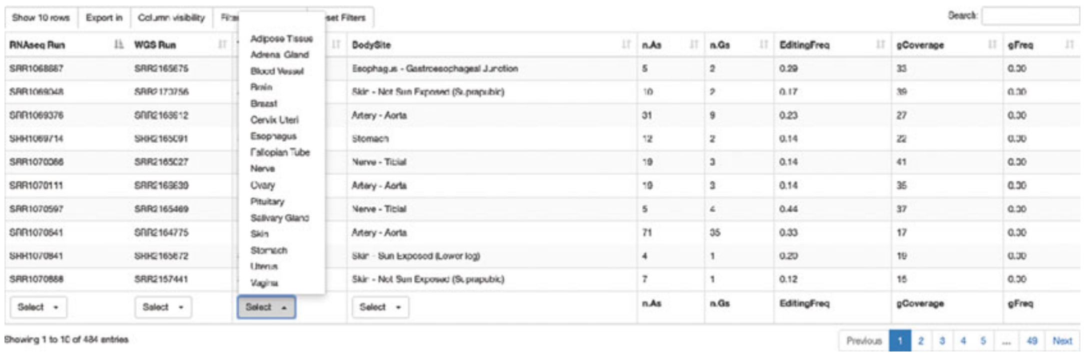
Fig. 10 Editing details



RNAseq Run	WGS Run	Tissue	BodySite	n.As	n.Gs	EditingFreq	gCoverage	gFreq
SRR1068687	SRR2165875	Esophagus	Esophagus - Gastroesophageal Junction	0	6	1.00	33	0.00
SRR1069048	SRR2170756	Skin	Skin - Not Sun Exposed (Suprapubic)	8	4	0.33	46	0.00
SRR1069376	SRR2166312	Blood Vessel	Artery - Aorta	0	39	1.00	32	0.00
SRR1069714	SRR2165391	Stomach	Stomach	0	14	1.00	31	0.00
SRR1070086	SRR2165327	Nerve	Nerve - Tibial	2	14	0.86	42	0.00
SRR1070111	SRR2166839	Blood Vessel	Artery - Aorta	0	21	1.00	41	0.00
SRR1070159	SRR2166542	Nerve	Nerve - Tibial	2	5	0.71	30	0.00
SRR1070260	SRR2157441	Breast	Breast - Mammary Tissue	6	2	0.25	20	0.00
SRR1070507	SRR2165469	Nerve	Nerve - Tibial	3	5	0.62	42	0.00
SRR1070641	SRR2161775	Blood Vessel	Artery - Aorta	0	99	1.00	24	0.00
Select	Select	Select	Select	n.As	n.Gs	EditingFreq	gCoverage	gFreq

Showing 1 to 10 of 550 entries

Fig. 11 Focus on “Editing Details”



RNAseq Run	WGS Run		BodySite	n.As	n.Gs	EditingFreq	gCoverage	gFreq
SRR1068887	SRR2165675		Esophagus - Gastroesochageal Junction	5	2	0.29	33	0.30
SRR1069048	SRR2173756		Skin - Not Sun Exposed (Suprapubic)	10	2	0.17	39	0.30
SRR1069376	SRR2163612		Artery - Aorta	31	9	0.23	27	0.30
SRR1069714	SRR2165091		Stomach	12	2	0.14	22	0.30
SRR1070066	SRR2165027		Nerve - Tibial	19	3	0.14	41	0.30
SRR1070111	SRR2163630		Artery - Aorta	19	3	0.14	35	0.30
SRR1070597	SRR2165469		Nerve - Tibial	5	4	0.44	37	0.30
SRR1070641	SRR2164775		Artery - Aorta	71	35	0.33	17	0.30
SRR1070841	SRR2165872		Skin - Sun Exposed (Lower leg)	4	1	0.20	19	0.30
SRR1070888	SRR2157441		Skin - Not Sun Exposed (Suprapubic)	7	1	0.12	15	0.30
Select	Select	Select	Select	n.As	n.Gs	EditingFreq	gCoverage	gFreq

Showing 1 to 10 of 484 entries

Fig. 12 Select “**Tissue**” in “Editing Details”

Individual run or tissue or body sites can be selected by using the “Select” button below each column (Fig. 12).

Further details can be found at http://srv00.recas.ba.infn.it/atlas/help.html.

3.2 REDIportal API

RNA editing sites can also be retrieved by a dedicated API (Application Programming Interface). Differently from the web-browsing approach, API does not leverage as much bandwidth and can also be interfaced with third-party programs able to manage outputs in JSON format. API service is freely accessible at **srv00.recas.ba.infn.it:5000/position/**. The results displayed in JSON format are the same contained in the corresponding REDIportal webpage. JSON output can be displayed in the web browser (Fig. 13) or in the UNIX shell. To submit a query in the web browser is sufficient to add the API address to the chromosome coordinates in the formats “Chr:start-end” (e.g., chr4:158101247–158308846).

http://srv00.recas.ba.infn.it:5000/position/chr4:158101247-158308846.

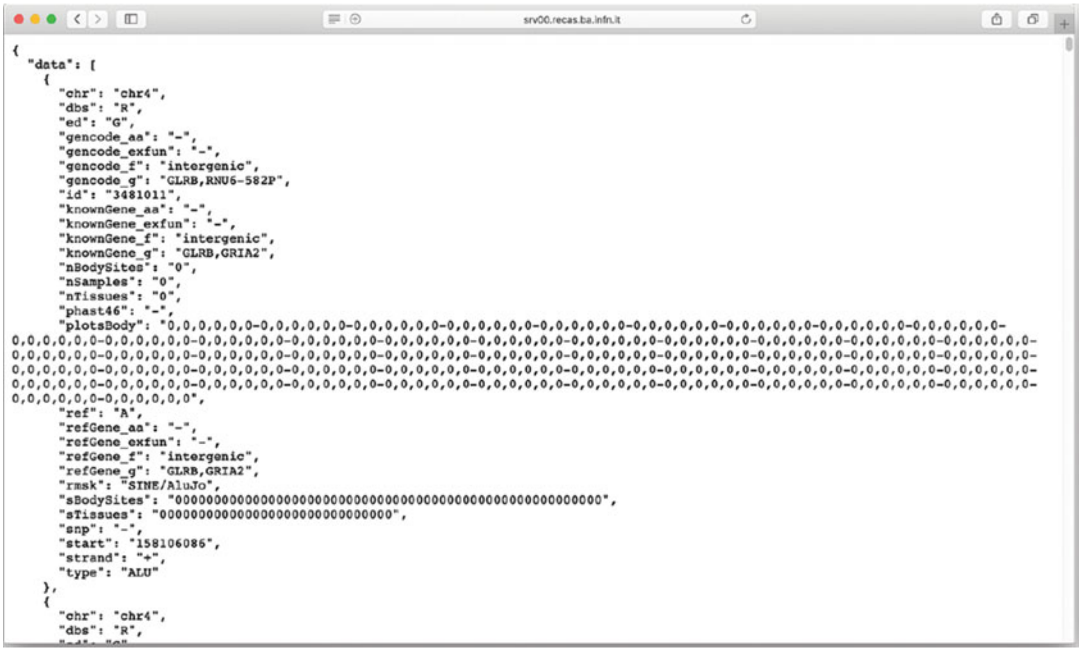
To obtain the same result in the UNIX commands shell (Fig. 14) the syntax is:

```
$ curl srv00.recas.ba.infn.it:5000/position/chr4:158101247-
158308846
```

3.3 Browse RNA Editing Sites in JBrowse

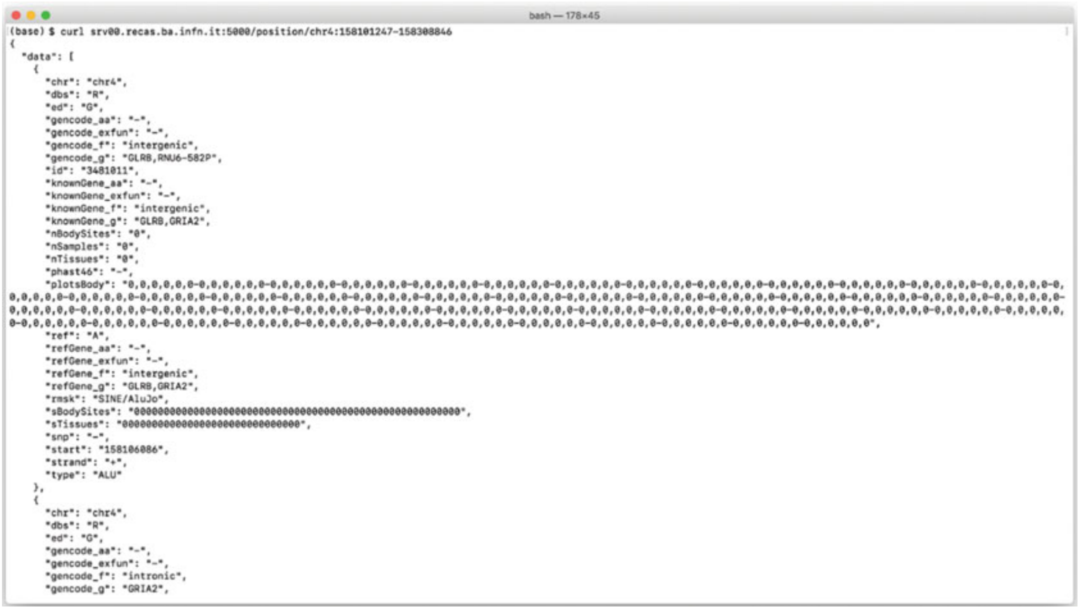
All RNA editing events stored in REDIportal can be explored in their genomic context through JBrowse [30], a fast genome browser based on JavaScript and HTML5. This browser is embedded in REDIportal and allows the visualization of basic tracks such as individual RNA editing sites, SNPs, RefSeq gene annotations, Alu elements, and LINEs (Fig. 15).

Main REDIportal tables are downloadable in the tab-delimited text format. Tables with RNA editing details are exportable in PDF or Excel formats. The download can be done by single


srv00.recas.ba.infn.it

```
{
  "data": [
    {
      "chr": "chr4",
      "dbs": "R",
      "ed": "G",
      "gencode_aa": "-",
      "gencode_exfun": "-",
      "gencode_f": "intergenic",
      "gencode_g": "GLRB,RNU6-582P",
      "id": "3481011",
      "knownGene_aa": "-",
      "knownGene_exfun": "-",
      "knownGene_f": "intergenic",
      "knownGene_g": "GLRB,GRIA2",
      "nBodySites": "0",
      "nSamples": "0",
      "nTissues": "0",
      "phast46": "-",
      "plotsBody": "0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-
0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-
0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-
0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-
0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-
0,0,0,0,0,0-0,0,0,0,0,0",
      "ref": "A",
      "refGene_aa": "-",
      "refGene_exfun": "-",
      "refGene_f": "intergenic",
      "refGene_g": "GLRB,GRIA2",
      "rmsk": "SINE/AluJo",
      "sBodySites": "0000000000000000000000000000000000000000000000000000000",
      "sTissues": "000000000000000000000000000000",
      "snp": "-",
      "start": "158106086",
      "strand": "+",
      "type": "ALU"
    },
    {
      "chr": "chr4",
      "dbs": "R",
```

Fig. 13 REDIportal API output (e.g., srv00.recas.ba.infn.it:5000/position/chr4:158101247-e158308846)


bash — 178×45

```
(base) $ curl srv00.recas.ba.infn.it:5000/position/chr4:158101247-158308846
{
  "data": [
    {
      "chr": "chr4",
      "dbs": "R",
      "ed": "G",
      "gencode_aa": "-",
      "gencode_exfun": "-",
      "gencode_f": "intergenic",
      "gencode_g": "GLRB,RNU6-582P",
      "id": "3481011",
      "knownGene_aa": "-",
      "knownGene_exfun": "-",
      "knownGene_f": "intergenic",
      "knownGene_g": "GLRB,GRIA2",
      "nBodySites": "0",
      "nSamples": "0",
      "nTissues": "0",
      "phast46": "-",
      "plotsBody": "0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,
0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-
0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,
0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0-0,0,0,0,0,0",
      "ref": "A",
      "refGene_aa": "-",
      "refGene_exfun": "-",
      "refGene_f": "intergenic",
      "refGene_g": "GLRB,GRIA2",
      "rmsk": "SINE/AluJo",
      "sBodySites": "0000000000000000000000000000000000000000000000000000000",
      "sTissues": "000000000000000000000000000000",
      "snp": "-",
      "start": "158106086",
      "strand": "+",
      "type": "ALU"
    },
    {
      "chr": "chr4",
      "dbs": "R",
      "ed": "G",
      "gencode_aa": "-",
      "gencode_exfun": "-",
      "gencode_f": "intronic",
      "gencode_g": "GRIA2",
```

Fig. 14 REDIportal API terminal output

chromosome or whole dataset. Relevant metadata such as samples info, header main table, and header levels table are also available for download.

Browse RNA Editing Sites in JBrowse:

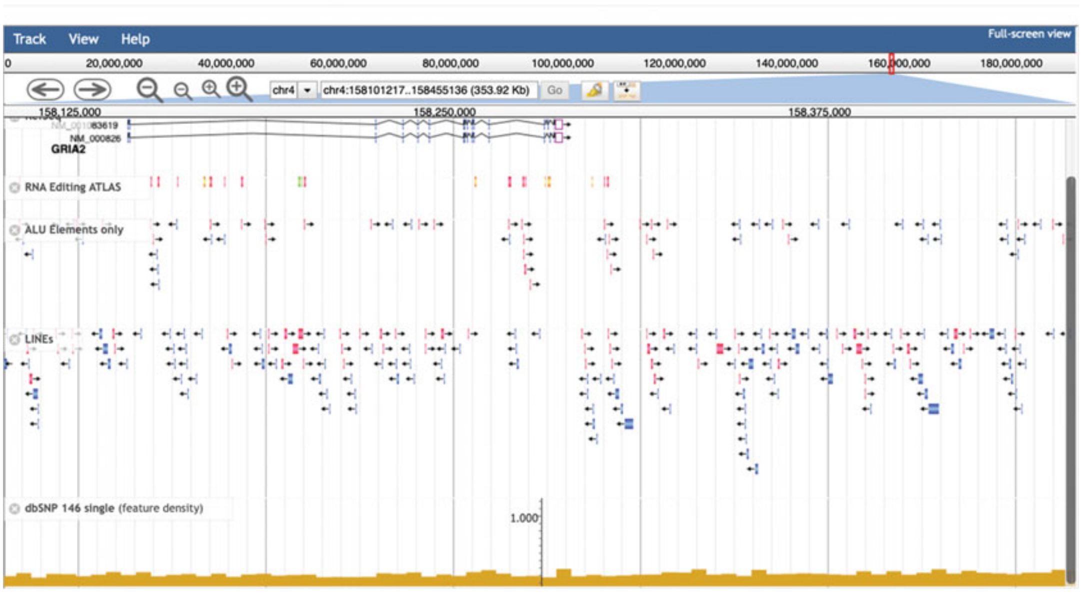


Fig. 15 Genomic intervals can be inspected by entering chromosome coordinates in the JBrowse search box in the format “chr:start-end” (e.g., chr4:158101247–158308846). Alternatively, the JBrowse search box accepts gene symbols and allows the autocomplete function to easily suggest gene names during the typing

To date, REDIportal is the reference resource devoted to RNA editing investigations and future plans include its expansion with additional RNAseq experiments from large scale projects (GTEx or TGCA or PsyENCODE) and further features such as hyper-edited regions and the Alu editing index for enabling the comparison of whole RNA editing profiles across human tissues.

Although REDIportal has been designed for human samples, it also houses mouse A-to-I events and will collect events from other organisms including primates, vertebrates, and invertebrates.

Acknowledgments

This work was supported by ELIXIR-IIB and PRACE projects 2016163924 and 2018194670.

References

1. Eisenberg E, Levanon EY (2018) A-to-I RNA editing - immune protector and transcriptome diversifier. Nat Rev Genet 19:473–490. https://doi.org/10.1038/s41576-018-0006-1
2. Barak M, Levanon EY, Eisenberg E, Paz N, Rechavi G, Church GM, Mehr R (2009) Evidence for large diversity in the human transcriptome created by Alu RNA editing. Nucleic Acids Res 37:6905–6915. https://doi.org/10.1093/nar/gkp729
3. Pullirsch D, Jantsch MF (2010) Proteome diversification by adenosine to inosine RNA-editing. RNA Biol 7:205–212. https://doi.org/10.4161/rna.7.2.11286

4. Bahn JH, Lee J-H, Li G, Greer C, Peng G, Xiao X (2012) Accurate identification of A-to-I RNA editing in human by transcriptome sequencing. Genome Res 22:142–150. https://doi.org/10.1101/gr.124107.111
5. Rosenberg BR, Hamilton CE, Mwangi MM, Dewell S, Papavasiliou FN (2011) Transcriptome-wide sequencing reveals numerous APOBEC1 mRNA-editing targets in transcript 3′ UTRs. Nat Struct Mol Biol 18:230–236. https://doi.org/10.1038/nsmb.1975
6. Nicholas A, de Magalhaes JP, Kraytsberg Y, Richfield EK, Levanon EY, Khrapko K (2010) Age-related gene-specific changes of A-to-I mRNA editing in the human brain. Mech Ageing Dev 131:445–447. https://doi.org/10.1016/j.mad.2010.06.001
7. Khermesh K, D'Erchia AM, Barak M, Annese A, Wachtel C, Levanon EY, Picardi E, Eisenberg E (2016) Reduced levels of protein recoding by A-to-I RNA editing in Alzheimer's disease. RNA 22:290–302. https://doi.org/10.1261/rna.054627.115
8. Hwang T, Park C-K, Leung AKL, Gao Y, Hyde TM, Kleinman JE, Rajpurohit A, Tao R, Shin JH, Weinberger DR (2016) Dynamic regulation of RNA editing in human brain development and disease. Nat Neurosci 19:1093–1099. https://doi.org/10.1038/nn.4337
9. Roth SH, Danan-Gotthold M, Ben-Izhak M, Rechavi G, Cohen CJ, Louzoun Y, Levanon EY (2018) Increased RNA editing may provide a source for autoantigens in systemic lupus erythematosus. Cell Rep 23:50–57. https://doi.org/10.1016/j.celrep.2018.03.036
10. Uchida S, Jones SP (2018) RNA editing: unexplored opportunities in the cardiovascular system. Circ Res 122:399–401. https://doi.org/10.1161/CIRCRESAHA.117.312512
11. Kung C-P, Maggi LBJ, Weber JD (2018) The role of RNA editing in cancer development and metabolic disorders. Front Endocrinol 9. https://doi.org/10.3389/fendo.2018.00762
12. Pinto Y, Levanon EY (2019) Computational approaches for detection and quantification of A-to-I RNA-editing. Methods 156:25–31. https://doi.org/10.1016/j.ymeth.2018.11.011
13. Ramaswami G, Lin W, Piskol R, Tan MH, Davis C, Li JB (2012) Accurate identification of human Alu and non-Alu RNA editing sites. Nat Methods 9:579–581. https://doi.org/10.1038/nmeth.1982
14. Lo Giudice C, Pesole G, Picardi E (2018) REDIdb 3.0: a comprehensive collection of RNA editing events in plant organellar genomes. Front Plant Sci 9:482. https://doi.org/10.3389/fpls.2018.00482
15. Li M, Xia L, Zhang Y, Niu G, Li M, Wang P, Zhang Y, Sang J, Zou D, Hu S, Hao L, Zhang Z (2019) Plant editosome database: a curated database of RNA editosome in plants. Nucleic Acids Res 47:D170–D174. https://doi.org/10.1093/nar/gky1026
16. Ramaswami G, Li JB (2014) RADAR: a rigorously annotated database of A-to-I RNA editing. Nucleic Acids Res 42:D109–D113. https://doi.org/10.1093/nar/gkt996
17. Kiran A, Baranov PV (2010) DARNED: a DAtabase of RNa EDiting in humans. Bioinformatics 26:1772–1776. https://doi.org/10.1093/bioinformatics/btq285
18. Ozgyin L, Horvath A, Hevessy Z, Balint BL (2019) Extensive epigenetic and transcriptomic variability between genetically identical human B-lymphoblastoid cells with implications in pharmacogenomics research. Sci Rep 9. https://doi.org/10.1038/s41598-019-40897-9
19. Liu H, Li Y, Chen D, Qi Z, Wang Q, Wang J, Jiang C, Xu J-R (2017) A-to-I RNA editing is developmentally regulated and generally adaptive for sexual reproduction in Neurospora crassa. Proc Natl Acad Sci U S A 114: E7756–E7765. https://doi.org/10.1073/pnas.1702591114
20. Bian Z, Ni Y, Xu J-R, Liu H (2019) A-to-I mRNA editing in fungi: occurrence, function, and evolution. Cell Mol Life Sci 76:329–340. https://doi.org/10.1007/s00018-018-2936-3
21. Liu J, Wang D, Su Y, Lang K, Duan R, Wu Y, Ma F, Huang S (2019) FairBase: a comprehensive database of fungal A-to-I RNA editing. Database (Oxford) 2019. https://doi.org/10.1093/database/baz018
22. Niu G, Zou D, Li M, Zhang Y, Sang J, Xia L, Li M, Liu L, Cao J, Zhang Y, Wang P, Hu S, Hao L, Zhang Z (2019) Editome disease knowledgebase (EDK): a curated knowledgebase of editome-disease associations in human. Nucleic Acids Res 47:D78–D83. https://doi.org/10.1093/nar/gky958
23. Picardi E, D'Erchia AM, Lo Giudice C, Pesole G (2017) REDIportal: a comprehensive database of A-to-I RNA editing events in humans. Nucleic Acids Res 45:D750–D757. https://doi.org/10.1093/nar/gkw767

24. Lin C-H, Chen SC-C (2019) The cancer editome atlas: a resource for exploratory analysis of the adenosine-to-inosine RNA editome in cancer. Cancer Res 79:3001–3006. https://doi.org/10.1158/0008-5472.CAN-18-3501

25. Picardi E, Manzari C, Mastropasqua F, Aiello I, D'Erchia AM, Pesole G (2015) Profiling RNA editing in human tissues: towards the inosinome Atlas. Sci Rep 5:1–17. https://doi.org/10.1038/srep14941

26. Aguet F, Barbeira AN, Bonazzola R, Brown A, Castel SE, Jo B, Kasela S, Kim-Hellmuth S, Liang Y, Oliva M, Parsana PE, Flynn E, Fresard L, Gaamzon ER, Hamel AR, He Y, Hormozdiari F, Mohammadi P, Muñoz-Aguirre M, Park Y, Saha A, Segrć AV, Strober BJ, Wen X, Wucher V, Das S, Garrido-Martín D, Gay NR, Handsaker RE, Hoffman PJ, Kashin S, Kwong A, Li X, MacArthur D, Rouhana JM, Stephens M, Todres E, Viñuela A, Wang G, Zou Y, Consortium TGte, Brown CD, Cox N, Dermitzakis E, Engelhardt BE, Getz G, Guigo R, Montgomery SB, Stranger BE, Im HK, Battle A, Ardlie KG, Lappalainen T (2019) The GTEx Consortium atlas of genetic regulatory effects across human tissues. bioRxiv 787903. 10.1101/787903

27. Wang K, Li M, Hakonarson H (2010) ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. Nucleic Acids Res 38:e164–e164. https://doi.org/10.1093/nar/gkq603

28. Frankish A, Diekhans M, Ferreira A-M, Johnson R, Jungreis I, Loveland J, Mudge JM, Sisu C, Wright J, Armstrong J, Barnes I, Berry A, Bignell A, Carbonell Sala S, Chrast J, Cunningham F, Di Domenico T, Donaldson S, Fiddes IT, García Girón C, Gonzalez JM, Grego T, Hardy M, Hourlier T, Hunt T, Izuogu OG, Lagarde J, Martin FJ, Martínez L, Mohanan S, Muir P, Navarro FCP, Parker A, Pei B, Pozo F, Ruffier M, Schmitt BM, Stapleton E, Suner M-M, Sycheva I, Uszczynska-Ratajczak B, Xu J, Yates A, Zerbino D, Zhang Y, Aken B, Choudhary JS, Gerstein M, Guigó R, Hubbard TJP, Kellis M, Paten B, Reymond A, Tress ML, Flicek P (2019) GENCODE reference annotation for the human and mouse genomes. Nucleic Acids Res 47:D766–D773. https://doi.org/10.1093/nar/gky955

29. Haeussler M, Zweig AS, Tyner C, Speir ML, Rosenbloom KR, Raney BJ, Lee CM, Lee BT, Hinrichs AS, Gonzalez JN, Gibson D, Diekhans M, Clawson H, Casper J, Barber GP, Haussler D, Kuhn RM, Kent WJ (2019) The UCSC genome browser database: 2019 update. Nucleic Acids Res 47:D853–D858. https://doi.org/10.1093/nar/gky1095

30. Buels R, Yao E, Diesh CM, Hayes RD, Munoz-Torres M, Helt G, Goodstein DM, Elsik CG, Lewis SE, Stein L, Holmes IH (2016) JBrowse: a dynamic web platform for genome visualization and analysis. Genome Biol 17:66. https://doi.org/10.1186/s13059-016-0924-1

Chapter 26

MODOMICS: An Operational Guide to the Use of the RNA Modification Pathways Database

Pietro Boccaletto and Błażej Bagiński

Abstract

MODOMICS is an established database of RNA modifications that provides comprehensive information concerning chemical structures of modified ribonucleosides, their biosynthetic pathways, the location of modified residues in RNA sequences, and RNA-modifying enzymes. This chapter covers the resources available on MODOMICS web server and the basic steps that can be undertaken by the user to explore them. MODOMICS is available at http://www.genesilico.pl/modomics.

Key words RNA, Ribonucleoside modification, Epitranscriptomics, Systems biology, Metabolic pathways, Enzymes, Database

1 Introduction

Modifications of ribonucleotides are post-synthesis changes to the chemical composition of RNA molecules, that have the potential to alter the stability and function of the entire polymer. RNA modifications occur in all living organisms and are one of the most conserved evolutionary features of RNAs [1]. Their chemical modifications have been known since 1950, but recently, since the discovery of key regulatory roles they play in gene expression and the advent of new technologies such as massive parallel sequencing of RNA (RNA-Seq), their popularity as a research subject has increased [2]. Various modifications are introduced, transformed, or removed by enzymes that interact with the nucleotides of the RNA chain by catalyzing diverse chemical reactions [3]. RNA modifications influence cell biology deeply, in a reversible and dynamic way. Modifications can perform a large number of functions: such as stability, folding, and decoding in tRNA [4, 5], or mRNA [6, 7]. They are also able to change their distribution and levels in response to stress factors, such as heat shock and nutrient deprivation, modulating cellular response [8]. Human diseases, such as cancer, diabetes, reduced fertility, or developmental delay, are directly or indirectly linked to the presence of specific modified nucleosides in certain positions, especially in the tRNA chains [9]. Many RNA chemical modifications, their pathways, functions, and enzymes are yet to be discovered. The systematic study of RNA modifications will allow for a better understanding of complex mechanisms that regulate the organisms, and to design new therapies for their related diseases.

The MODOMICS database (http://www.genesilico.pl/modomics) is a resource that collects and organizes up-to-date knowledge on RNA nucleosides chemical modifications, the chemical structures of modified ribonucleosides, their biosynthetic pathways, the location of modified residues in RNA sequences, and RNA-modifying enzymes [10]. It contains also browsable, by product of interest, collection of chemical synthesis protocols. Users will find information organized in pages, accessible through the left menu bar. The database finds usage in the diverse areas of knowledge, where RNA modifications are of interest—among others are cancer research [11–13], tRNA modifications [14–16], and viral research [17], and is described in numerous review articles in the field. An analog to MODOMICS, a database of covalently modified DNA nucleobases, DNAmod, has also been created [18].

MODOMICS is designed with its users in mind, with every update performed by a commission of RNA specialists, supported by highly valuable user's feedback. To date, its dedicated articles have been cited more than 1300 times by RNA researchers.

This chapter describes the MODOMICS database in an operative way. For clarity reasons, examples in this text will focus on the *N2,N2-dimethylguanosine* (m2,2G, or 22G), http://www.genesilico.pl/modomics/modifications/N2,N2-dimethylguanosine/), one of the many RNA modifications present in the database, to explain the various functionalities of MODOMICS. Bold font indicates terms as they appear in database. A link to the help page is located at the very top of the home page.

2 Materials

2.1 Web Server

The Web site backend is written in Python 3 (https://www.python.org/) and Django (https://www.djangoproject.com/), and the frontend is made in HTML, javascript, and bootstrap (https://getbootstrap.com/). The server is reachable at the address: http://www.genesilico.pl/modomics. MODOMICS relies also on third-party javascript libraries: JSmol (http://jmol.sourceforge.net/) allows us to visualize the 3D chemical structures of the modified ribonucleosides in the browser, cytoscape.js (https://js.cytoscape.org/) is used to plot the pathways graphs and datatables (https://datatables.net/) makes tables interactive adding the ability to filter by field, search, sort, print, and export them in different formats like PDF or CSV.

2.2 Data Collection

MODOMICS is a manually curated database, the data is collected, selected, and inserted from published articles and from the private collaboration between laboratories. A list of supporting publications can be found in the "Publications" tab, in the navigation panel.

3 Methods

The methods are described as per their order in the user menu (navigation panel) that is visible on the left of every page of the MODOMICS website.

3.1 Modifications

The "Modifications" page (http://www.genesilico.pl/modomics/modifications) stores a collection of naturally occurring modified RNA nucleotides. This can be seen as two different sections, one where all the modifications are ordered in a table (the overview page) and another, accessible through clicking on the name of the modification, as the description page of each modification (the detail page). At the time of writing, it contains 172 records.

3.1.1 Overview Page

The Overview Page lists all the modifications present in the MODOMICS database. It is possible to filter the table by using the drop-down lists at the top of the page—both by the base from which the modified is derived (by selecting the option of interest from the "Originating Base" list), as well as by their chemical type of modification (analogically). Both filters can be applied at the same time.

Following search filters are available for the nucleobase that the modified one originates from:

– Four standard nucleoside residues present in the transcribed RNA (**Adenosine**, **Citidine**, **Guanosine**, **Uridine**).
– **Queuosine**, which is synthesized first as a free modified base, and only then attached to the ribose of RNA by a transglycosylation reaction and eventually further hypermodified.
– The 5′ end of the **nascent transcript**, which represents the starting point for RNA capping.

The example options of the second filter, "Chemical type of modification" are as listed:

– Inorganic residues (e.g., **ammonia**, **hydroxyl**).
– Organic residues (e.g., **amino acid**, **methyl group**, **nucleobase**).
– Atoms (e.g., **heavy atom**, **hydrogen**, **sulphur**).
– As well as more general **ring modification**, and **none**.

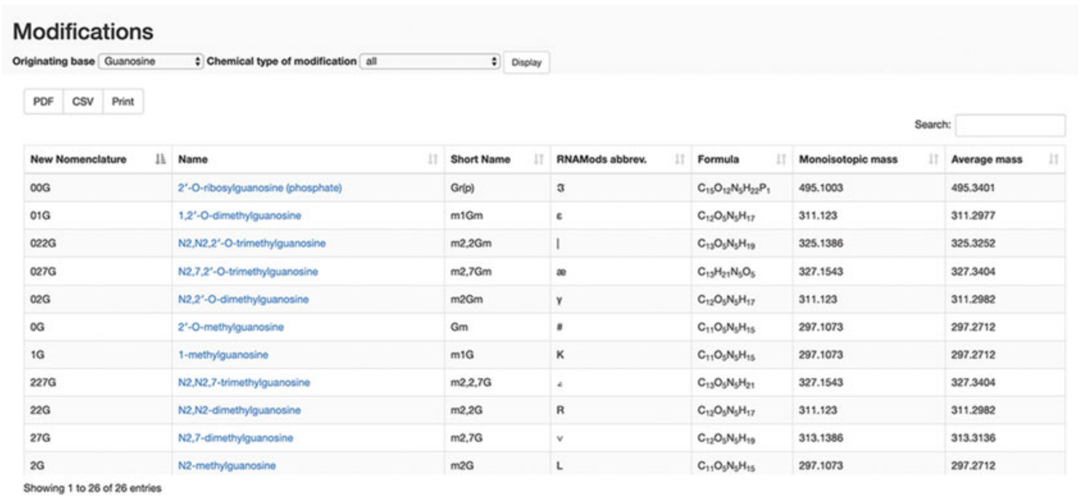
For each of them, following information are directly presented: **full name** (e.g., N2,N2-dimethylguanosine), **short name** (e.g., m2,2G), **new nomenclature** (e.g., 22G), naming convention that uses only digits and standard A, G, C, U letters, (designed to be more compatible with automated data processing that previous, containing special characters, short names), machine-friendly **MODOMICS-specific single character abbreviation** (e.g., R), a symbol that is used in RNA sequences to identify univocally the modification, molecular **formula** (e.g., $C_{12}O_5N_5H_{17}$), and two **molecular masses**: **monoisotopic** (e.g., 311.123) and **average** (311.2982). At the same time, obtained results can be sorted, ascendingly and descendingly, by clicking on the appropriate header.

As an example, we would like to display all those modifications whose originating base is guanosine.

1. Go on the modifications page, that being the second button in the navigation menu, or visit directly: http://www.genesilico.pl/modomics/modifications.
2. On the "Originating base" drop-down menu select "Guanosine." Chemical type of modification should be left to "all."
3. Press the Display button.

Result: 26 RNA modifications originated from the guanosine base will be displayed in the table, as seen in Fig. 1.

As an example, If a user would like to narrow the results and learn which of those previous results are modified by the addition or removal of a methyl group, the procedure would be as follows:



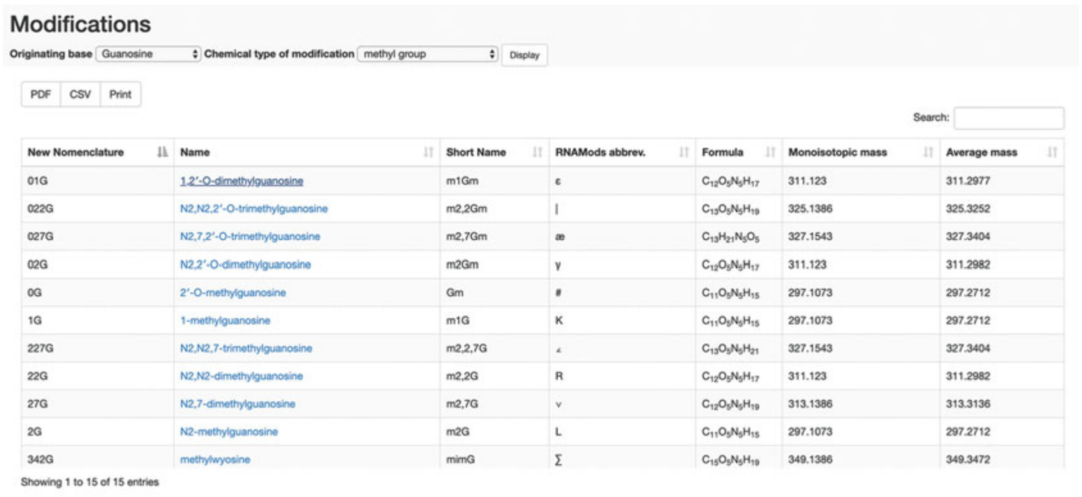
Modifications

Originating base: Guanosine Chemical type of modification: all

New Nomenclature	Name	Short Name	RNAMods abbrev.	Formula	Monoisotopic mass	Average mass
00G	2′-O-ribosylguanosine (phosphate)	Gr(p)	ℨ	$C_{15}O_{12}N_5H_{22}P_1$	495.1003	495.3401
01G	1,2′-O-dimethylguanosine	m1Gm	ε	$C_{12}O_5N_5H_{17}$	311.123	311.2977
022G	N2,N2,2′-O-trimethylguanosine	m2,2Gm	|	$C_{13}O_5N_5H_{19}$	325.1386	325.3252
027G	N2,7,2′-O-trimethylguanosine	m2,7Gm	æ	$C_{13}H_{21}N_5O_5$	327.1543	327.3404
02G	N2,2′-O-dimethylguanosine	m2Gm	γ	$C_{12}O_5N_5H_{17}$	311.123	311.2982
0G	2′-O-methylguanosine	Gm	#	$C_{11}O_5N_5H_{15}$	297.1073	297.2712
1G	1-methylguanosine	m1G	K	$C_{11}O_5N_5H_{15}$	297.1073	297.2712
227G	N2,N2,7-trimethylguanosine	m2,2,7G	∠	$C_{13}O_5N_5H_{21}$	327.1543	327.3404
22G	N2,N2-dimethylguanosine	m2,2G	R	$C_{12}O_5N_5H_{17}$	311.123	311.2982
27G	N2,7-dimethylguanosine	m2,7G	∨	$C_{12}O_5N_5H_{19}$	313.1386	313.3136
2G	N2-methylguanosine	m2G	L	$C_{11}O_5N_5H_{15}$	297.1073	297.2712

Showing 1 to 26 of 26 entries

Fig. 1 A result of guanosine derivative modification filtering applied to the modification page


Modifications
Originating base Guanosine
Chemical type of modification methyl group
Display
PDF
CSV
Print
Search:

New Nomenclature	Name	Short Name	RNAMods abbrev.	Formula	Monoisotopic mass	Average mass
01G	1,2′-O-dimethylguanosine	m1Gm	ε	$C_{12}O_5N_5H_{17}$	311.123	311.2977
022G	N2,N2,2′-O-trimethylguanosine	m2,2Gm	|	$C_{13}O_5N_5H_{19}$	325.1386	325.3252
027G	N2,7,2′-O-trimethylguanosine	m2,7Gm	æ	$C_{13}H_{21}N_5O_5$	327.1543	327.3404
02G	N2,2′-O-dimethylguanosine	m2Gm	γ	$C_{12}O_5N_5H_{17}$	311.123	311.2982
0G	2′-O-methylguanosine	Gm	#	$C_{11}O_5N_5H_{15}$	297.1073	297.2712
1G	1-methylguanosine	m1G	K	$C_{11}O_5N_5H_{15}$	297.1073	297.2712
227G	N2,N2,7-trimethylguanosine	m2,2,7G	∠	$C_{13}O_5N_5H_{21}$	327.1543	327.3404
22G	N2,N2-dimethylguanosine	m2,2G	R	$C_{12}O_5N_5H_{17}$	311.123	311.2982
27G	N2,7-dimethylguanosine	m2,7G	∨	$C_{12}O_5N_5H_{19}$	313.1386	313.3136
2G	N2-methylguanosine	m2G	L	$C_{11}O_5N_5H_{15}$	297.1073	297.2712
342G	methylwyosine	mimG	∑	$C_{15}O_5N_5H_{19}$	349.1386	349.3472

Showing 1 to 15 of 15 entries

Fig. 2 A result of filtering for guanosine derivative altered with methyl group addition/removal

4. From the menu "Chemical type of modification" drop-down menu and select "methyl group."
5. Press the Display button.

Result: 15 nucleotide modifications, which originated from a guanosine base and were further altered by the addition or removal of a methyl group have been displayed in the table, as seen in Fig. 2.

3.1.2 Detail Page

For every result of the search presented in the Overview Page, a Detail Page, containing further information, can be accessed by clicking on modification's name. It can also be accessed without going through the Overview Page, by generating URL in the following manner:

http://www.genesilico.pl/modomics/modifications/modification_name/.

The Detail Page contains extensive information about each modification:

– A link to the Pathway Page of the modification's base of origin (described in Subheading 3.2).
– A graphical representation of its molecular structure.
– A summary table, again providing **full** and **short name**, **RNA-Mods abbreviation**, as well as MODOMICS **new nomenclature**. In addition, it presents the list of known **enzymes** synthesizing it, their names also serve as links to the respective Protein Summary (further described in Subheading 3.5). The summary table also lists the kingdom in which it occurs (based on the list of organisms known to possess enzymes synthesizing

it), RNA types in which it occurs (based on MODOMICS collection of modified sequences), **PubChem ID** and a **SMILES** 1D structure.

– A table of physicochemical parameters of the molecule, useful for liquid chromatography and mass spectrometry (titled "LC-MS Information"): molecular formula, set of **monoisotopic**, **average**, and **protonated masses**, with protonated fragments ions generated from the precursor mass **[M + H] +** (product ions). Where available, **normalized LC elution time** and **LC elution order/characteristics** are provided with bibliographic references below. However, this data is available just for a small portion ("number") of the modifications.
– Structural information: interactive, 3D molecule representation of the molecule, a list of chemical groups that are modified in the nucleobase of origin, as well as address codes of Protein Data Bank entries, of known structures containing this modification.
– Enzymatic information: providing direct links to the MODOMICS detailed reaction page (described in 3. "Reactions") of enzymatic reactions for which the modification is either a substrate or product.

As an example, we would like to display detailed information about N2,N2-dimethylguanosine modification.

1. In the table of filtered search results, obtained in Subheading 3.1.1, **step 1**, Example of use **step 5**, click the name of the modification of interest, or add the name of modification "N2, N2-dimethylguanosine" after following URL template:

http://www.genesilico.pl/modomics/modifications/modification_name/

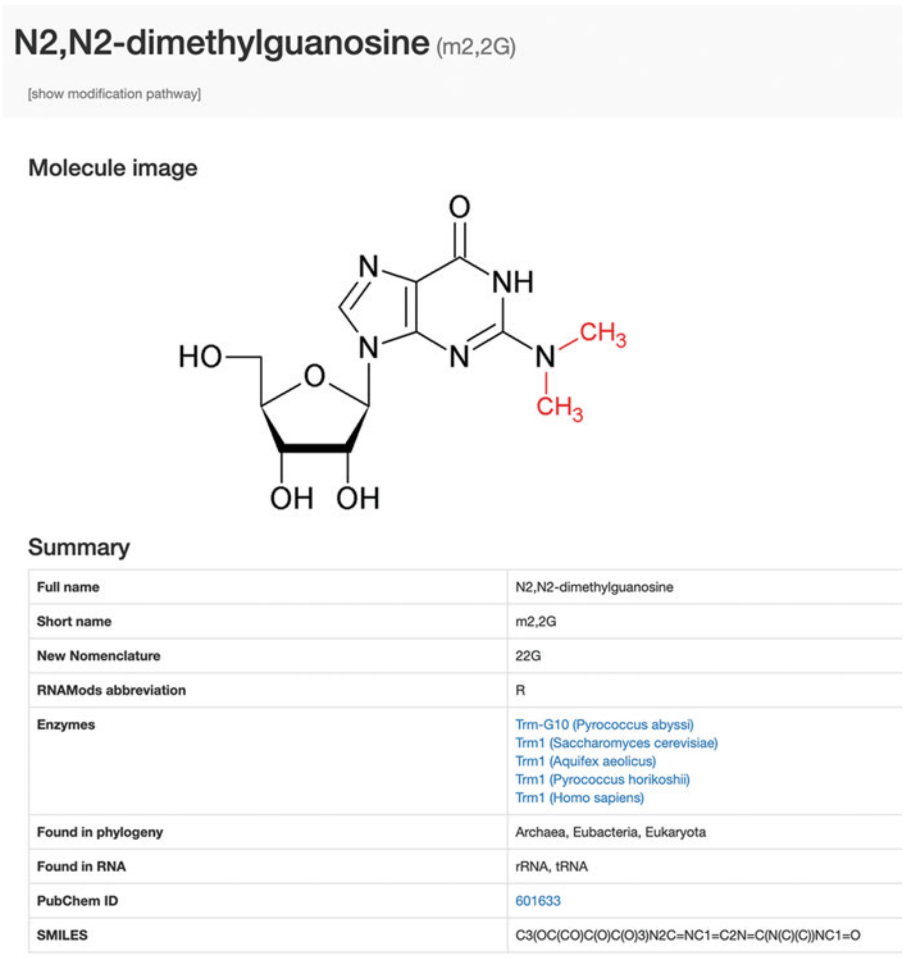
Obtaining the following direct URL:

http://www.genesilico.pl/modomics/modifications/N2,N2-dimethylguanosine/.

Result: the user is presented with the Detail Page of N2,N2-dimethylguanosine modification (Fig. 3).

3.2 Pathways

Pathways page (http://www.genesilico.pl/modomics/pathways) contains the graphical depiction of every known (documented) reaction, plus many speculated, that modifies the chemical structure of any part of nucleotide, traced back to the nucleobase of origin. Two additional enzymatic pathways are presented to the user: queuosine pathway, originating at preQ (7-cyano-7-deazaguanine) and originating from 5′ triphosphate end modifications.


N2,N2-dimethylguanosine (m2,2G)

[show modification pathway]

Molecule image

Summary

Full name	N2,N2-dimethylguanosine
Short name	m2,2G
New Nomenclature	22G
RNAMods abbreviation	R
Enzymes	Trm-G10 (Pyrococcus abyssi) Trm1 (Saccharomyces cerevisiae) Trm1 (Aquifex aeolicus) Trm1 (Pyrococcus horikoshii) Trm1 (Homo sapiens)
Found in phylogeny	Archaea, Eubacteria, Eukaryota
Found in RNA	rRNA, tRNA
PubChem ID	601633
SMILES	C3(OC(CO)C(O)C(O)3)N2C=NC1=C2N=C(N(C)(C))NC1=O

Fig. 3 A preview of the detailed modification page for N2,N2-dimethylguanosine

Following pathway from its base of origin to the modification of interest allows the user to understand how modifications emerge from the different unmodified residues in precursor RNA, what are the enzymes that can catalyze the reactions and what is the reaction types that modify the nucleosides.

Enzymatic reactions are structured in six pathway graphs representing the four nucleobases, the queuosine, and the 5′ end. Selecting one of the nucleobases from the main page will open a related graph, presenting a hierarchical structure with the unmodified residue on the top, branching into a pathway of its derivatives. The graphs are interactive and it is possible to move and zoom them, as well to change the location of their elements, by dragging them around on the display window. Displayed graphs can be downloaded in various format files (such as pictures, pdf or XML) by right-clicking on the white background of the graph.

Every modification is displayed as a box, labeled with its short name and MODOMICS new nomenclature. Clicking a particular box links the user to modification's Detail Page (described in Subheading 3.1.2). Arrows connecting two modifications are colored according to the type of reaction (such as exchange, removal or transfer of the groups, or certain bonds formation), with the dashed arrows indicating putative reactions. Their color codes are explained in the graph legend. All arrows are also clickable and link to the reaction-dedicated web page (described in Subheading 3.3.2).

As an example, we would like to investigate the origin of N2, N2-dimethylguanosine modification, and its role in N2,N2,2′-*O*-trimethylguanosine (m2,2Gm, 022G) formation.

1. Go on the Pathways Page, that being the third button in the navigation menu, or visit directly: http://www.genesilico.pl/modomics/pathways.
2. Select the Guanine base of origin.
3. Search on the graph for the m2,2G/22G modification.

Result: the user is presented with the known pathways map originating from guanine (Fig. 4). N2,N2-dimethylguanosine (m2,2G and 22G), is denoted by a blue (reactions of methylation) arrow, to be a product of methylation of N2-methylguanosine (m2G, 2G), and a suspected precursor to the N2,N2,2′-*O*-trimethylguanosine (m2,2Gm, 022G). Further details about reaction can be obtained by clicking on the arrow (reaction summary page being explained in 3.3.2). Detail Information (Subheading 3.1.2) page can be viewed by clicking on the modification's name.

3.3 Reactions

"Modification reactions" section of MODOMICS (http://www.genesilico.pl/modomics/reactions) catalogs both known and putative enzyme driven nucleotide alterations, while providing an atomistic level of details. Analogically to Modifications tab (Subheading 3.1.1), this information is divided into two different sections, the Overview Page, where all the reactions are ordered in a table, and another, accessible through clicking on the name of the reaction, as the description page of each reaction (the Reaction Summary page). At the time of writing, it contains 192 entries.

3.3.1 Overview Page

This page lists all the reactions present in the MODOMICS database. It is possible to filter the table by using the drop-down lists at the top of the page—by the base from which the modified is derived, by selecting the option of interest from the "Originating Base" list (described in Subheading 3.1.1), as well as by the type of catalyzed reaction (**isomerization**, **group removal**, **group exchange**, **group addition**). Both filters can be applied at the same time.

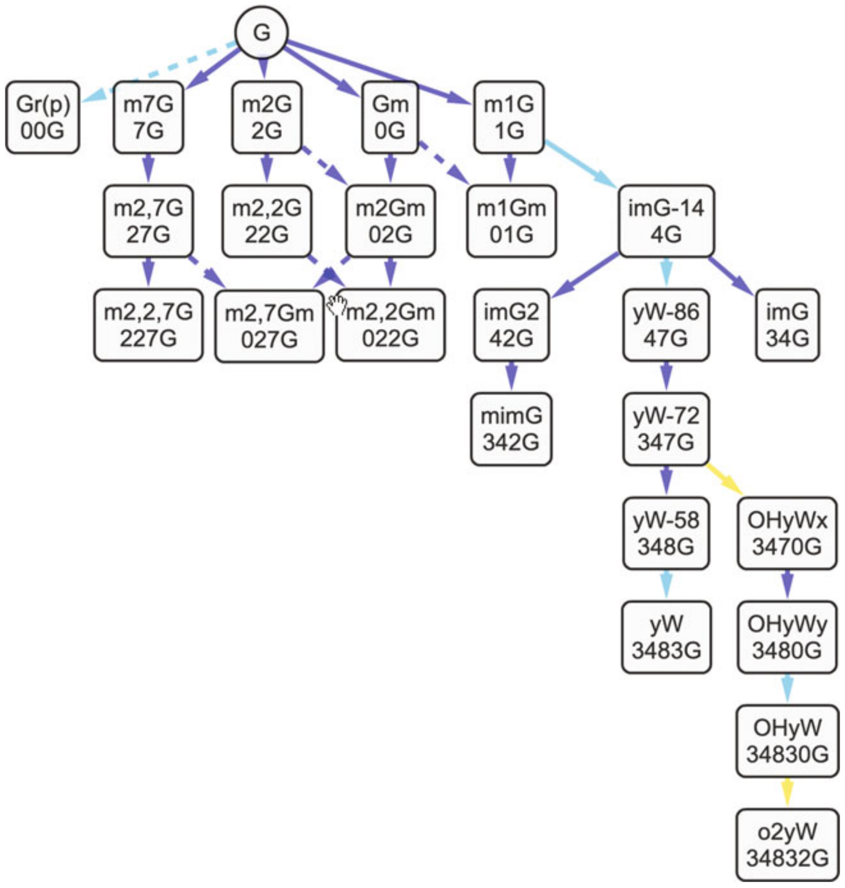
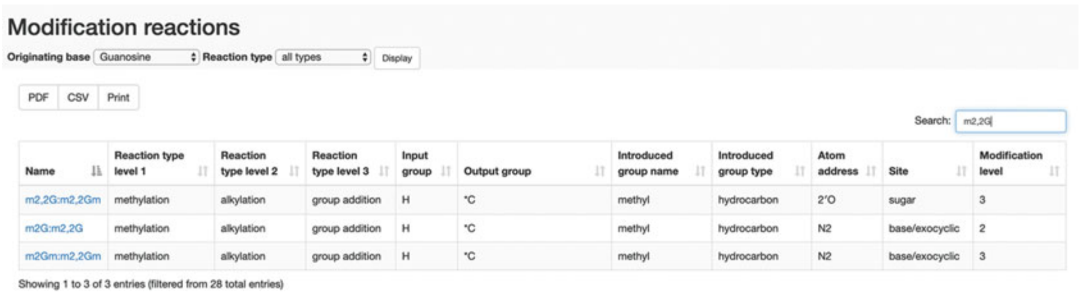


Fig. 4 Pathways map originating from guanine.

The user will then be presented with the overview table, listing the following information: **name**, consisting of standard "substrate_short_name:product_short_name" format (e.g., m2,2G: m2,2Gm), three levels of **reaction type** (**one**: ex. methylation, hydroxylation, **two**: ex. alkylation, acylation, **three**: group addition, group exchange), machine-friendly empirical notation of **input group** being modified (e.g., H, O) and resulting **output group** (e.g., C, C(=O)C). The alteration process itself (enzymatic reaction) is defined by **introduced group name** (e.g., methyl, guanyl), **introduced group type** (e.g., Hydrocarbon, nucleotide) and placed structurally by **atom address** (e.g., 2′O, N6) at the specified **site** (e.g., phosphate, base/exocyclic). **The modification level** is also designated. At the same time, obtained results can be sorted, ascendingly and descendingly, by clicking on the appropriate header.

As an example, we would like to know the details of the reaction leading to the formation of N2,N2-dimethylguanosine (m2,2G).



Name	Reaction type level 1	Reaction type level 2	Reaction type level 3	Input group	Output group	Introduced group name	Introduced group type	Atom address	Site	Modification level
m2,2G:m2,2Gm	methylation	alkylation	group addition	H	˚C	methyl	hydrocarbon	2'O	sugar	3
m2G:m2,2G	methylation	alkylation	group addition	H	˚C	methyl	hydrocarbon	N2	base/exocyclic	2
m2Gm:m2,2Gm	methylation	alkylation	group addition	H	˚C	methyl	hydrocarbon	N2	base/exocyclic	3

Showing 1 to 3 of 3 entries (filtered from 28 total entries)

Fig. 5 Three enzymatic reactions involving N2,N2-dimethylguanosine (m2,2G), in one of those denoted as a product

1. Go on the Reactions tab, that being the fourth button in the navigation menu, or visit directly: http://www.genesilico.pl/modomics/reactions.
2. On the "Originating base" drop-down menu select "Guanosine." Chemical type of modification should be left to "all." Press the Display button.
3. Name is composed of the substrate:product short names. If the user does not remember it, then it can be obtained by going back to MODOMICS "Modifications" tab and typing its chemical name into the search field. In this example, it is the "m2,2G".

The modification reactions list can be scrolled until encountering entry of interest, or "m2,2G" search phrase can be typed into a search box.

Result: A table entry containing information of the m2G:m2,2G reaction is displayed, as seen in Fig. 5.

3.3.2 Reaction Summary Page

This page is available for every reaction included in the database. It can also be accessed without going through the Overview Page (Subheading 3.3.1), by generating URL in the following manner:

http://www.genesilico.pl/modomics/reaction/reaction_name/.

It consists of three main elements:

– A convenient repetition of all the information described in the Subheading 3.3.1.

– Graphical representations (skeletal formulas) of both substrate and product of the reaction. Elements (atoms, residues, fragments of molecules) undergoing change are highlighted in red.

– A table containing a list of known enzymes, denoting their **acronym** (e.g., Trm-G10), **full name** (e.g., tRNA (guanine (10)-N2)-dimethyltransferase) and **organism** of origin (e.g., Pyrococcus abyssi). The acronym is also a direct link to the Protein Summary Page (described in Subheading 3.5.2). Putative enzymes are not indicated.

Detailed information on each reaction comprises enzymes that have been experimentally proven to catalyze it, chemical structures of the substrate(s) and product(s), information about cofactors, and other information in the free text format.

As an example, we would like to know what enzymes are mediating the formation of N2,N2-dimethylguanosine (m2,2G).

1. On the list of results obtained in Subheading 3.3.1 Example of use, **step 3**, click the reaction's name. A direct access link is generated by adding known reaction name "m2G:m2,2G" at the end of the template URL:

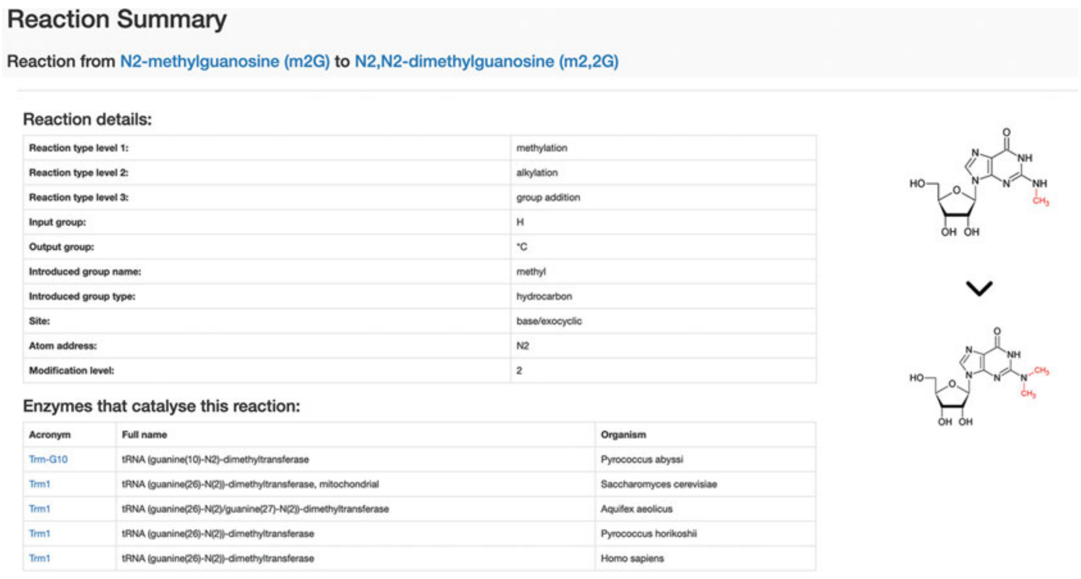
http://www.genesilico.pl/modomics/reaction/reaction_name/

Obtaining a direct URL to the desired summary page:

http://www.genesilico.pl/modomics/reaction/m2G:m2,2G/.

This page can also be accessed by clicking on the appropriate arrow, on the graphical representation of modification enzymatic pathways, described in Subheading 3.2.

Result: The obtained page includes known enzymes catalyzing this reaction (Trm-G10 and Trm1), listing their names and organism(s) of origin. Multiple homologs of Trm1 are known, resulting in multiple listings of that enzyme, with each name being a link to further enzyme's description (Subheading 3.5.2). It is presented in Fig. 6.



Reaction Summary

Reaction from N2-methylguanosine (m2G) to N2,N2-dimethylguanosine (m2,2G)

Reaction details:

Reaction type level 1:	methylation
Reaction type level 2:	alkylation
Reaction type level 3:	group addition
Input group:	H
Output group:	*C
Introduced group name:	methyl
Introduced group type:	hydrocarbon
Site:	base/exocyclic
Atom address:	N2
Modification level:	2

Enzymes that catalyse this reaction:

Acronym	Full name	Organism
Trm-G10	tRNA (guanine(10)-N2)-dimethyltransferase	Pyrococcus abyssi
Trm1	tRNA (guanine(26)-N(2))-dimethyltransferase, mitochondrial	Saccharomyces cerevisiae
Trm1	tRNA (guanine(26)-N(2)/guanine(27)-N(2))-dimethyltransferase	Aquifex aeolicus
Trm1	tRNA (guanine(26)-N(2))-dimethyltransferase	Pyrococcus horikoshii
Trm1	tRNA (guanine(26)-N(2))-dimethyltransferase	Homo sapiens

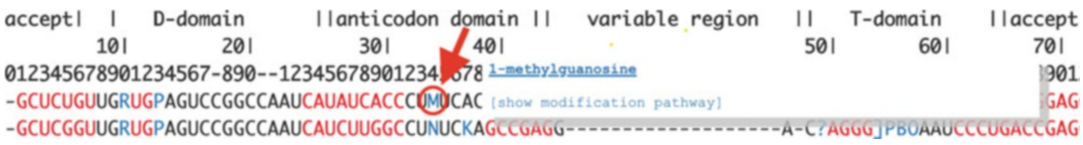
Fig. 6 Page containing detailed information of N2-methylguanosine (m2G) to N2,N2-dimethylguanosine (m2,2G) conversion

3.4 RNA Sequences

The RNA sequences page (http://www.genesilico.pl/modomics/sequences) contains a collection of tRNA, rRNA, snRNA, and snoRNA sequences that are known to be modified at multiple positions. During opening the page no results are shown, but instead the user is asked to select at least an RNA type from the above drop-down menu, and eventually filter the results by "subtype" and "organism." Once the button "Display" is pressed, a table with the available list of aligned RNA sequences is displayed. Depending on the type of the selected RNA type the table acquires different headers to adapt to the different features and available information of the sequences. The full list of headers contains of RNA "Type" and "Subtype"," Genbank" [19] code, MINTbase [20] link, "Amino acid type" and "Anticodon," "Organism," and the "Organellum" where the sequence was identified and the "numbering scheme for the alignment." For tRNA the numbering scheme above the alignment is based on the *E. coli* sequences. For all other RNA types, it refers to the first sequence in the alignment. It is possible to select a numbering based on one of the other sequences of the alignment clicking on their GenBank or type name. For tRNA sequences the Helical regions are colored in red, loop and bulge regions in black. Modifications within sequences are indicated in blue and marked with one-letter abbreviations, a unique symbol that identifies the modification in MODOMICS. Hovering the mouse cursor over a modified base within a sequence allows overviewing the full name of the modification and a link to its pathway page. Upon clicking on the full name or the symbol of a given modified base within a sequence, the corresponding detailed page of the Modification is shown (described in Subheading 3.1.2). The "Draw modification profile" (available for rRNA and tRNA sequences) allows displaying the mapping of the modified positions on secondary structure diagrams of RNA molecules. The mapping is done based on the sequence alignments. For rRNAs a reference structure of *E. coli* SSU and LSU rRNAs is used, while for tRNAs a consensus secondary structure diagram is used. It is possible to map information from a user-selected set of sequences, available in MODOMICS, onto the diagram. In such a case, the percentage of modified ribonucleosides of any type in each alignment position is calculated and displayed. The resulting diagrams can be downloaded as image files.

We would like to see all the transfer RNAs (tRNA) sequences that contain the amino acid glutamic acid from *Haloferax volcanii*.

1. Go on the RNA Sequence page: http://www.genesilico.pl/modomics/sequences.
2. On the "RNA type" drop-down menu select "tRNA."
3. On the "Subtype" drop-down menu select "Glu."



```
accept|   |   D-domain        ||anticodon domain ||   variable region    ||   T-domain      ||accept
        10|         20|          30|        40|                               50|        60|        70|
01234567890123456 7-890--12345678901234 678 1-methylguanosine                                        901
-GCUCUGUUGRUGPAGUCCGGCCAAUCAUAUCACCCUMUCAC [show modification pathway]                              GAG
-GCUCGGUUGRUGPAGUCCGGCCAAUCAUCUUGGCCUNUCKAGCCGAGG------------------A-C?AGGG]PBOAAUCCCUGACCGAG
```

Fig. 7 Hoovering above an RNA modification symbol on the RNA sequence unravels its name and links to its pathway graph and detail page

4. Select "*Haloferax volcanii*" on the "Organism" drop-down menu.
5. Press Display.

Results: As we can see two sequences with mentioned characteristics are available in the database. At position 34 of their anticodon domain, both have an "M" modified base.

6. Hover above the modification at position 34 on one of the sequences.
7. Click on "N4-acetylcytidine" to open the detailed page of the modification.

Results: We explored the modification at position 34 of the two tRNA Glu of *Haloferax volcanii* (Fig. 7).

There is also a possibility to see the modification profile. For this purpose.

8. Click on the "Draw Modification Profile" button.

Result: The modification profile can be seen on a new page (Fig. 8). We can define how to display the modification scale by selecting any of the options on the "display" drop-down menu.

3.5 Proteins

3.5.1 Overview Page

The "protein" page (http://www.genesilico.pl/modomics/proteins) contains a collection of both functional enzymes and protein cofactors necessary for multiprotein enzymatic activities that are involved in the RNA modification process. These proteins alter the behavior of the RNAs catalyzing different chemical reactions such as deamidation, methylation, and reduction. 340 proteins are available in the proteins table, which can be filtered by species, enzyme type (e.g., methyltransferase, pseudouridine synthase) or by type of RNA involved in the reaction. The table contains different information, the **Traditional Name** is the most often used or recommended acronym (e.g., "Ceg1"), the **Full name** (e.g., "GTP-RNA guanylyltransferase"), **Synonym** is a list of alternative names assigned to the protein (e.g., "GTase, Mce1"), **GI number** is the number by which the protein is recognized in the NCBI protein database [21] (e.g., "1,322,697"), **ORF name** (e.g., "CEG1"), **COG** that is the "Clusters of Orthologous Genes" code the protein has in the COG database [21] (e.g., "COG5226"), **UniProt ID** is the code of the protein in the UniProt database [22]

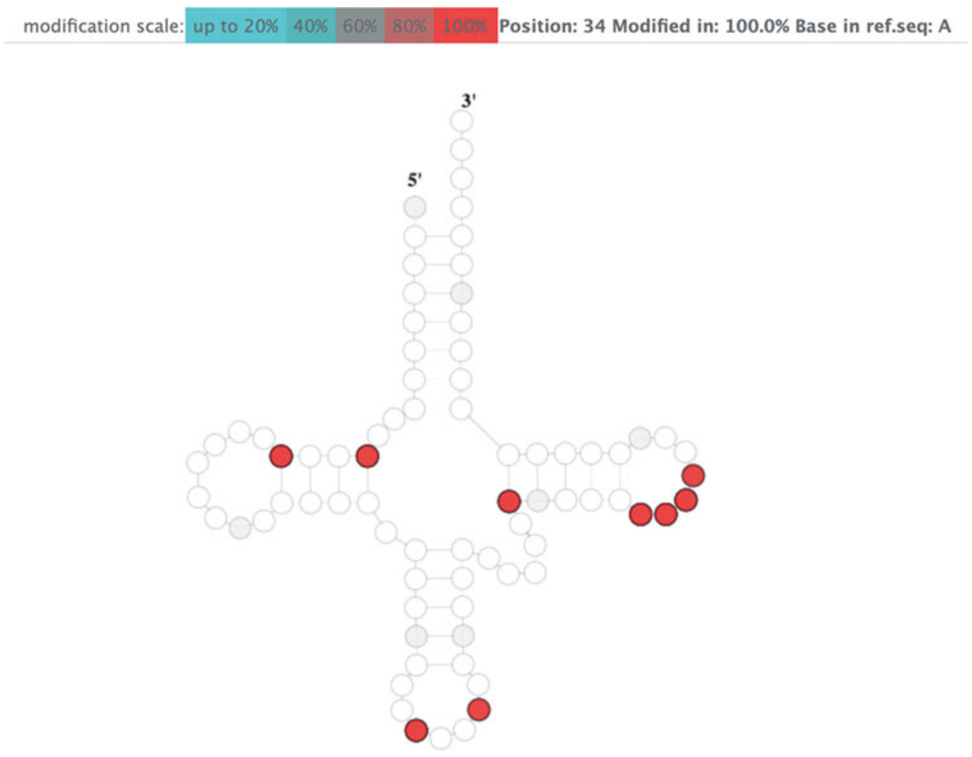

modification scale: up to 20% 40% 60% 80% 100% Position: 34 Modified in: 100.0% Base in ref.seq: A
3'
5'

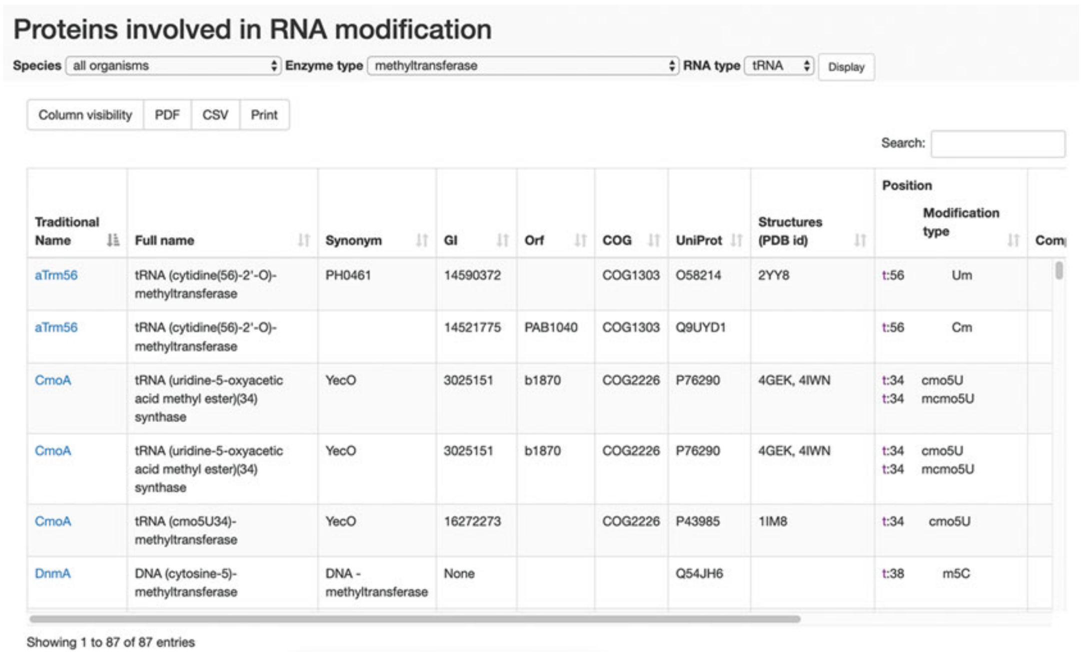
Fig. 8 Modification profile generated from the alignment of two tRNA Glutamic acid of *Haloferax volcanii*

(ex."Q01159"), **Structures (PDB ID)** can contain a list structures of the protein in the form of Protein Data Bank [23] codes (e.g.," 3KYH"), **Position/Modification type** gives the position (e.g., "m:0″) and the modification (ex."m7Gpp(pN)") and refers to the final modification, **Complex** only present if the protein works as a part of a well-characterized complex, **Enzyme type** is the type of reaction catalyzed by the enzyme (ex."guanylyltransferase"), **Organism** is the organism species the protein was found in (e.g., "*Saccharomyces cerevisiae*").

As an example, we would like to display all those proteins that catalyze the transfer of a methyl group on nucleobases of tRNA.

1. Go on the "proteins" page, that being the sixth button in the navigation menu, or visit directly: http://www.genesilico.pl/modomics/proteins.
2. On the "Enzyme type" drop-down menu, select "methyltransferase" and "RNA type" set to "tRNA."
3. Press the Display button.

Result: 87 methyltransferase proteins that operate on tRNA will be displayed in the table, as seen in Fig. 9.



Proteins involved in RNA modification

Species: all organisms · Enzyme type: methyltransferase · RNA type: tRNA · Display

Column visibility · PDF · CSV · Print

Search:

Traditional Name	Full name	Synonym	GI	Orf	COG	UniProt	Structures (PDB id)	Position	Modification type	Com
aTrm56	tRNA (cytidine(56)-2'-O)-methyltransferase	PH0461	14590372		COG1303	O58214	2YY8	t:56	Um	
aTrm56	tRNA (cytidine(56)-2'-O)-methyltransferase		14521775	PAB1040	COG1303	Q9UYD1		t:56	Cm	
CmoA	tRNA (uridine-5-oxyacetic acid methyl ester)(34) synthase	YecO	3025151	b1870	COG2226	P76290	4GEK, 4IWN	t:34 t:34	cmo5U mcmo5U	
CmoA	tRNA (uridine-5-oxyacetic acid methyl ester)(34) synthase	YecO	3025151	b1870	COG2226	P76290	4GEK, 4IWN	t:34 t:34	cmo5U mcmo5U	
CmoA	tRNA (cmo5U34)-methyltransferase	YecO	16272273		COG2226	P43985	1IM8	t:34	cmo5U	
DnmA	DNA (cytosine-5)-methyltransferase	DNA - methyltransferase	None			Q54JH6		t:38	m5C	

Showing 1 to 87 of 87 entries

Fig. 9 A result of proteins summary filtered by Enzyme type "methyltransferase" and RNA type "tRNA"

3.5.2 Summary Page

Each protein has its own dedicated page that gives the user more information about the details of each entry. The summary page has the title as the name of the protein and organisms in which it was identified. It consists of six elements:

- A convenient repetition of the information of all the elements described in Subheading 3.5.1.
- The "comments" section with a short text characterizing the protein.
- The "Protein sequence" section with the amino acid sequence of the protein.
- A table listing the "enzymatic activities" of the protein with links to the summary reaction page, the substrate of the reaction, the type of RNA involved and the position of the reaction in the RNA sequence.
- A table listing the publications related to the protein and its activity.
- Few links to additional data sources ex. Wikipedia, relevant PubMed search, Saccharomyces Genome Database (for yeast proteins), EcoCyc (for *E. coli* proteins).

As an example, we would like to find the protein sequence of the "CmoA" protein.

1. From the results obtained in Subheading 3.5.1, **step 3**, click the name of the protein of interest.
2. Scroll to the section “Protein Sequence.”

Result: The user finds the protein sequence into the Overview Page of the “CmoA” protein.

```
MSHRDTLFSAPIARLGDWTFDERVAEVFPDMIQRSVPGYSNIISMIGMLAERFVQPGTQ-
VYDLGCSLGAA TLSVRRNIHHDNCKIIAIDNSPAMIERCRRHIDAYKAPTPVDVIEGDIR
D I A I E N A S M V V L N F T L Q F L E P                   S
ERQALLDKIYQGLNPGGALVLSEKFSFEDAKVGELLFNMHHDFKRANGYSELEISQKRSM
LENVMLTDS VETHKARLHNAGFEHSELWFQCFNFGSLVALKAEDAA
```

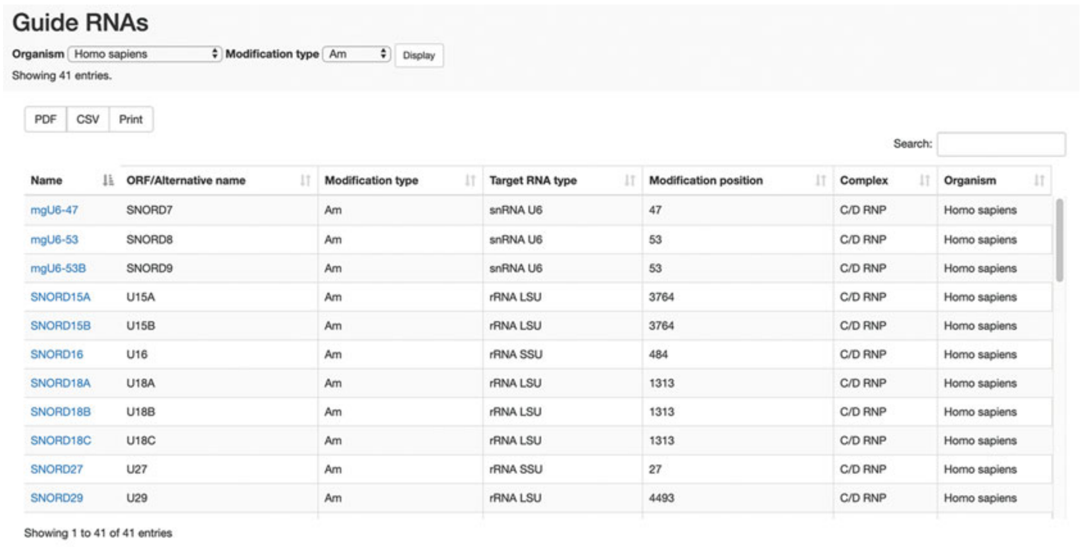
3.6 Guide RNAs

Small nucleolar RNAs or snoRNAs are a class of small RNA molecules that primarily guide chemical modifications of other RNAs like rRNAs, tRNAs, and snRNAs. Two main classes of snoRNAs exist, the C/D box snoRNAs, involved in methylation and the H/ACA box snoRNAs, involved in pseudouridylation [24]. MODOMICS “Guide RNAs page” (http://www.genesilico.pl/modomics/snornas/) contains a collection of RNAs from human and yeast snoRNAs that are involved in RNA-guided RNA modification by the C/D box and H/ACA box ribonucleoproteins, linked to the corresponding modification sites in human and yeast RNAs. Information about guide RNA allows the user to understand how modifications emerge from precursor RNA, how to design engineered gRNAs that can participate in posttranscriptional modifications, and how to use gRNAs in gene regulation methods such as RNA mutagenesis. The page contains a list of 276 snoRNAs with the following information, if available: **name** (e.g., “SNORD16”), linked to appropriate entries in HGNC database [25] or The yeast snoRNA database [26] for human and yeast snoRNAs, respectively, **ORF/Alternative name** (e.g., “U16”), **Modification type** (e.g., “Am”), **Modified position** (e.g., “484”), **Target RNA type** (e.g., “rRNA SSU”), **Complex** (e.g., “C/D RNP”) and **Organism** (e.g., “*Homo sapiens*”). The list of Guide RNAs can be filtered by the organism and/or type of modification that is found in the target position.

As an example, we would like to see all those guide RNAs in humans whose modification type is “Am.”

1. Go on the Guide RNAs page: http://www.genesilico.pl/modomics/snornas/.
2. On the “Organism” drop-down menu select “*Homo sapiens*.”
3. On the “Modification type” drop-down menu select “Am.”
4. Press Display.

Result: 41 Guide RNA in *Homo sapiens* with modification type Am are now displayed in the table (Fig. 10).


Guide RNAs

Organism: Homo sapiens · Modification type: Am · Display

Showing 41 entries.

PDF · CSV · Print

Search:

Name	ORF/Alternative name	Modification type	Target RNA type	Modification position	Complex	Organism
mgU6-47	SNORD7	Am	snRNA U6	47	C/D RNP	Homo sapiens
mgU6-53	SNORD8	Am	snRNA U6	53	C/D RNP	Homo sapiens
mgU6-53B	SNORD9	Am	snRNA U6	53	C/D RNP	Homo sapiens
SNORD15A	U15A	Am	rRNA LSU	3764	C/D RNP	Homo sapiens
SNORD15B	U15B	Am	rRNA LSU	3764	C/D RNP	Homo sapiens
SNORD16	U16	Am	rRNA SSU	484	C/D RNP	Homo sapiens
SNORD18A	U18A	Am	rRNA LSU	1313	C/D RNP	Homo sapiens
SNORD18B	U18B	Am	rRNA LSU	1313	C/D RNP	Homo sapiens
SNORD18C	U18C	Am	rRNA LSU	1313	C/D RNP	Homo sapiens
SNORD27	U27	Am	rRNA SSU	27	C/D RNP	Homo sapiens
SNORD29	U29	Am	rRNA LSU	4493	C/D RNP	Homo sapiens

Showing 1 to 41 of 41 entries

Fig. 10 Guide RNA page after the filter "*Homo sapiens*" and modification type "Am" are applied

3.7 Publications

The "publications" page (http://www.genesilico.pl/modomics/papers), contains all the organized bibliography from which MODOMICS sources its data. These publications are used to cite the data about modifications, enzymes, reactions, that are present in MODOMICS. The Overview Page contains a list of all the publications utilized to build MODOMICS and to cite its data. For every line in the list, the information about the first author, the title of the article, and PubMed ID (or DOI) is given. It is possible to access the detailed page for each publication by clicking on the name of the author or the title of the article. The detailed page relative to a publication is very simple. It shows the information regarding its title, the authors, the journal and the abstract. If present also the related elements of the database that refers to it.

3.8 Building Blocks

This subset of database provides a browsable catalogue of publications (protocols), designed to facilitate chemical synthesis of naturally occurring modified nucleosides. The substrate for such reaction is referred in MODOMICS as a "building block," and is used as a base for the search/filtering engine, available at:

http://www.genesilico.pl/modomics/blocks.

Detailed information such as chemical, structural, or a protocol itself, are provided across three progressive tiers of descriptions, for each possible precursor of particular modification (building block) separately.

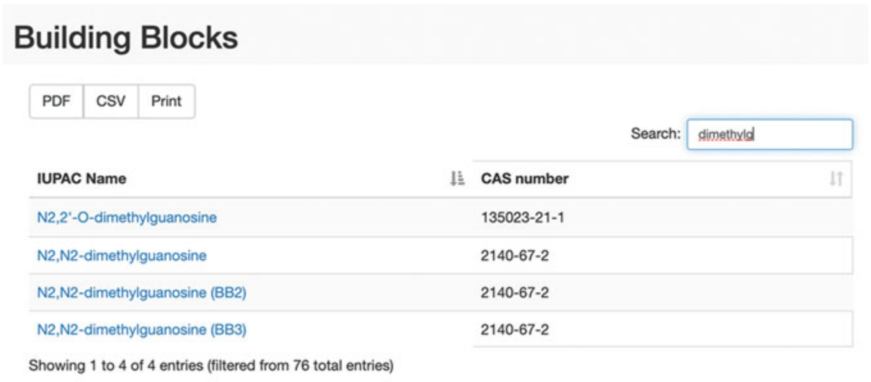
3.8.1 Overview Page

This page lists names of MODOMICS building blocks (reaction substrates), together with the CAS number identifying the modified nucleotide (reaction product). For clarity purposes, the name of a particular building block (reaction substrate) is kept the same as the name of resulting nucleotide modification (reaction product). If a multiple method of synthesis of a particular modified RNA monomer are known, originating from different substrates, than those would be indicated by a progressive number added to the block's name (e.g., BB2, BB3). User can filter the results, and a click on the building blocks name would lead to its detailed page.

As an example, user would like to know if there are any existing protocols for chemical synthesis of N2,N2-dimethylguanosine (m2,2G) present in MODOMICS.

1. Go on the Building blocks tab, that being the third button from the bottom of the navigation menu, or visit directly: http://www.genesilico.pl/modomics/blocks.
2. A list of all building blocks would be available. By default, it is sorted ascendingly by block's name. A real-time search bar is also available. Scroll down to the appropriate section. Alternatively, typing "dimethylg" in the search bar would narrow the list to the desired entries (as seen in Fig. 11).

Result: three different publications describing the chemical synthesis of N2,N2-dimethylguanosine (m2,2G and 22G) from three different substrates are present in the MODOMICS database. Further details can be obtained by clicking on the building block's name, leading to the following subpages (described in Subheading 3.8.2 and 3.8.3).


Building Blocks

IUPAC Name	CAS number
N2,2'-O-dimethylguanosine	135023-21-1
N2,N2-dimethylguanosine	2140-67-2
N2,N2-dimethylguanosine (BB2)	2140-67-2
N2,N2-dimethylguanosine (BB3)	2140-67-2

Showing 1 to 4 of 4 entries (filtered from 76 total entries)

Fig. 11 list of MODOMICS names of substrates for chemical synthesis of N2,N2-dimethylguanosine (m2,2G). In this example, three building blocks of interest are present in the database

3.8.2 Building Block Summary

This page contains the majority of information about, and is available for every building block described in MODOMICS. It can be accessed directly, by generating URL according to the formulas:

For an only known building block for particular modification, or the first discovered (i.e., the building block's name is the same as modification's):

http://genesilico.pl/modomics/block/<block_name>/.

For subsequent building blocks for the same modification:

http://genesilico.pl/modomics/block/<block_name>%20(BB<subsequent_number>)/.

This page contains following information:

– **Building block's name** (in certain cases, can be identical as modification's name), together with **IUPAC name** and **CAS number** of particular RNA modification.
– Table with graphical structure of the modification (**Naturally occurring structure**), and an analogical structure for currently viewed building block (**Building block structure**). From those, parts of importance (functional groups) are highlighted as a **building block characteristics**. Their names, structures and short description of chemical properties (e.g., acid labile group, or nonnucleophilic base) are provided, as well as "details" button, leading to the corresponding Reagent and protecting group Summary subpage (described further in Subheading 3.8.3).
– A detailed list of publications referring to the currently viewed building block, with interactive links: button "details" directs to the page dedicated fully to that publication (described in Subheading 3.7), **PubMed Id** number leads to article's entry in the PubMed database, with DOI number, analogically, linking to the article itself.

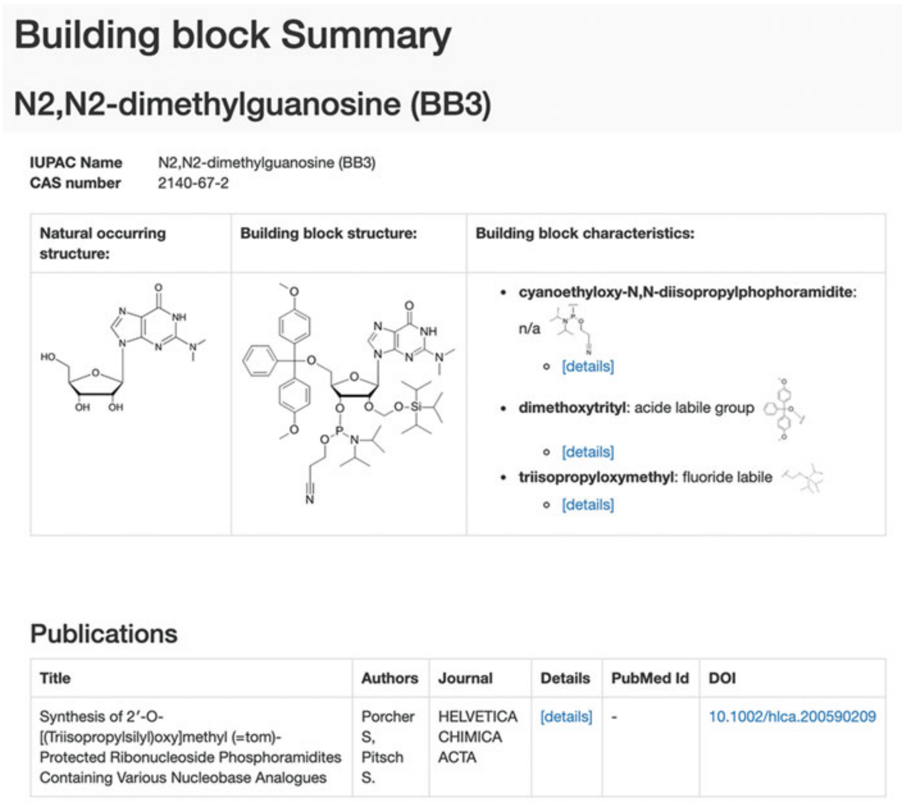
User would like to verify the feasibility of N2,N2-dimethylguanosine (m2,2G) synthesis in their laboratory.

1. On the list of results obtained in Subheading 3.8.1, **step 2**, investigate all three results of search for N2,N2-dimethylguanosine building blocks. An example of Summary page is presented in Fig. 12.

URL addresses can be used to switch between building blocks summary pages in a more automated way, according to the formulas described above. In that case, the addresses for all three pages of interest would be as follows:

http://www.genesilico.pl/modomics/block/N2,N2-dimethylguanosine/.

http://www.genesilico.pl/modomics/block/N2,N2-dimethylguanosine%20(BB2)/.



Building block Summary

N2,N2-dimethylguanosine (BB3)

IUPAC Name	N2,N2-dimethylguanosine (BB3)
CAS number	2140-67-2

Natural occurring structure:	Building block structure:	Building block characteristics:
		• cyanoethyloxy-N,N-diisopropylphophoramidite: n/a ∘ [details] • dimethoxytrityl: acide labile group ∘ [details] • triisopropyloxymethyl: fluoride labile ∘ [details]

Publications

Title	Authors	Journal	Details	PubMed Id	DOI
Synthesis of 2′-O-[(Triisopropylsilyl)oxy]methyl (=tom)-Protected Ribonucleoside Phosphoramidites Containing Various Nucleobase Analogues	Porcher S, Pitsch S.	HELVETICA CHIMICA ACTA	[details]	-	10.1002/hlca.200590209

Fig. 12 A summary page providing details about the third precursor (building block), present in MODOMICs, for N2,N2-dimethylguanosine synthesis

http://www.genesilico.pl/modomics/block/N2,N2-dimethylguanosine%20(BB3)/.

Result: user is presented with chemical and structural information about compounds required for the synthesis, their functional groups are described, and further details can be found in the source publication (described in Subheading 3.7).

3.8.3 Reagent and Protecting Group Summary

This page compiles information about the building block's functional groups, and can be accessed by clicking the corresponding "details" button on the Building block Summary page (Subheading 3.8.2). It is distributed into three main areas:

– Functional group's structure and **IUPAC Name** (e.g., 2-(dansyl)ethoxy carbonyl) with information aiding synthesis planning process, such as **Protecting group** (e.g., 5′-ribose), **Deprotection** method (e.g., with DBU/nonnucleophilic base) and its **Synthesis reagent** (e.g., nitrophenylethyl chloride).

– A list of building blocks which include that functional group, with the name being a hyperlink to block's summary page (Subheading 3.8.2), together with CAS numbers of their corresponding RNA modification.

– A listing of publications referring to above mentioned building blocks, with interactive links described in corresponding entry of the previous section (Subheading 3.8.2).

3.9 Search

The search page (http://genesilico.pl/modomics/search/advance/) is designed to facilitate finding information in the entire MODOMICS database. The page is organized into three tabs that can be used to reach different types of insights. The "**keyword**" tab allows the user to search by using a textual keyword, this is able to return all the result that corresponds with the name or short-name of modifications, the name of modifying enzymes or papers that contains the search term in its title or abstract. This keyword-based search is also the default search that will run when using the search bar present at the top right corner of the home page. The other two searches are more selective. As their name suggests "**Protein sequences**" tab can be used to search for Protein by inputting an amino acid sequence while "**Nucleic sequences**" is used to search for RNA sequences by inputting a nucleic acid sequence. Both work running BLAST (BLASTP for Amino acid and BLAST for nucleotides) on our database, the results are expressed in hits.

3.9.1 Keyword Search

It can be convenient sometimes to search directly for a modification instead of browsing through the list of all the known modifications.

We read in an article that position 34 of the human tRNA sequence can be modified with "cm5U" but we know little about this modification. We can search it with the keyword search of MODOMICS:

1. Go on the search page: http://genesilico.pl/modomics/search/advance/.
2. Enter the text "cm5u" in the textbox present under the keyword tab.
3. Press on the magnifying glass or press enter to start the search.

Result: The search tool found five modification matches that contained the keyword cm5U, and five articles as well. We see that 5-carboxymethyluridine (cm5U) is listed as third result. We can click on the full name of the modification to enter its detailed page and explore it.

3.9.2 Protein Sequences Search

If we have an amino acid sequence it can be useful to try to find if it is a known protein to catalyze an RNA-modifying reaction. The search page returns the output of the BLASTP search with all the sequence names linked to the detail page of the proteins.

We have the following amino acid sequence.

```
>unknown
AGVAGVVYLGRGRGLGPYYLARSGVEVVEVHPDEPLGYDPVDRLDVLLTFGGNPYLTEED-
VAAR VYCLLTGRGFDADIAPAPENLSGRVEIMVTRGDPDEAV
```

1. Open the search page: http://genesilico.pl/modomics/search/advance/.
2. Go to the tab "Protein sequences."
3. Insert a title for your search and paste your sequence in FASTA or raw format.
4. Click submit and wait for the results. This will also generate a unique link you can bookmark to later return back to the results.

Result: From the search we get to a page where all the sequences that produced a significant alignment have been listed. We notice that the first hit "238537840|CDAT8|Methanopyrus kandleri" has an *E*-value of 3e-68 and identity/positives of 100% and 0% gaps.

```
> 238537840|CDAT8|Methanopyrus kandleri
Length=278

 Score = 541 bits (1394), Expect = 0.0, Method: Compositional
matrix adjust.
 Identities = 278/278 (100%), Positives = 278/278 (100%), Gaps
= 0/278 (0%)
```

Opening the page linked to the name of the sequence we reach the detailed page of the protein, where we can see that our sequence is just a fragment of a larger sequence composing this protein.

3.9.3 Nucleic Sequences Search

This search must be used to explore the presence of an RNA sequence in the MODOMICS database. This works exactly as a Protein search, but it uses BLASTN instead of BLASTP.

We have the following sequence

```
>unknown
GGCGCGUAAACAAAGCGGAAAUGUAGCGGAUUGCAUAUCCGUCUAGUCCGGUUCGA-
CUCCGGAACGCGCCUCCA
```

1. Open the search page: http://genesilico.pl/modomics/search/advance/.
2. Go to the tab "Nucleic sequences."
3. Insert a title for your search and paste your sequence in FASTA or raw format.

4. Click submit and wait for the results. This will also generate a unique link you can bookmark to later return back to the results.

Result: We are brought to a page where one sequence that produced a significant alignment has been listed. "9|Escherichia coli" has *E*-value of 1e−28 and identity of 95% and 0% gaps. At this moment this kind of search can only reveal if a similar sequence is present on the MODOMICS database, but it is not able to point the user to the sequence itself.

3.10 Others

In the left menu of the website, we can also find Links, Downloads, Help, and Contact pages.

These pages can be very useful. The Links page contains a collection of links to other resources like RNA Databases, Auxiliary databases and tools, ontology and the affiliations. The downloads page contains a collection of flat files, modifications images, and 3D structures extracted from the MODOMICS database, it also contains references to the articles wrote about MODOMICS. The Help page is a very useful resource that sums up the content of the various sections of the database with few explanations on how to use it. The contact page has a list of current and past database developers and curators with their emails, as well it has the contacts of the Laboratory of Bioinformatics and Protein Engineering where MODOMICS was designed, developed and hosted.

4 Conclusions

MODOMICS has been in the years a useful tool for researchers which serves as a valuable resource for RNA modification and RNA-modifying enzymes. MODOMICS can offer an overall view of all modifications, quick information of pathways, reactions, proteins, and publications, visualization of the 3D structure of modified nucleosides. This can help researchers gain a proper perspective of the RNA modifications and the molecular mechanisms behind it. A constant database update process is necessary to include all the new recent discoveries in the field of RNA modifications and in the future will also be extended to include new functions.

Acknowledgments

We want to thank all the people who during the years were involved in the MODOMICS project from which we were inspired while writing this chapter. A special thanks go to Prof. Janusz Bujnicki from whose idea MODOMICS was developed.

References

1. Li S, Mason CE (2014) The pivotal regulatory landscape of RNA modifications. Annu Rev Genomics Hum Genet 15:127–150. https://doi.org/10.1146/annurev-genom-090413-025405
2. Limbach PA, Paulines MJ (2017) Going global: the new era of mapping modifications in RNA. Wiley Interdiscip Rev RNA 8:e1367. https://doi.org/10.1002/wrna.1367
3. Jonkhout N, Tran J, Smith MA et al (2017) The RNA modification landscape in human disease. RNA 23:1754–1769. https://doi.org/10.1261/rna.063503.117
4. El Yacoubi B, Bailly M, de Crécy-Lagard V (2012) Biosynthesis and function of posttranscriptional modifications of transfer RNAs. Annu Rev Genet 46:69–95. https://doi.org/10.1146/annurev-genet-110711-155641
5. Helm M (2006) Post-transcriptional nucleotide modification and alternative folding of RNA. Nucleic Acids Res 34:721–733. https://doi.org/10.1093/nar/gkj471
6. Ranjan N, Leidel SA (2019) The epitranscriptome in translation regulation: mRNA and tRNA modifications as the two sides of the same coin? FEBS Lett 593:1483–1493. https://doi.org/10.1002/1873-3468.13491
7. Shi H, Wei J, He C (2019) Where, when, and how: context-dependent functions of RNA methylation writers, readers, and erasers. Mol Cell 74:640–650. https://doi.org/10.1016/j.molcel.2019.04.025
8. Nachtergaele S, He C (2017) The emerging biology of RNA post-transcriptional modifications. RNA Biol 14:156–163. https://doi.org/10.1080/15476286.2016.1267096
9. de Crécy-Lagard V, Boccaletto P, Mangleburg CG et al (2019) Matching tRNA modifications in humans to their known and predicted enzymes. Nucleic Acids Res 47:2143–2159. https://doi.org/10.1093/nar/gkz011
10. Boccaletto P, Machnicka MA, Purta E et al (2018) MODOMICS: a database of RNA modification pathways. 2017 update. Nucleic Acids Res 46:D303–D307. https://doi.org/10.1093/nar/gkx1030
11. Chen X-Y, Zhang J, Zhu J-S (2019) The role of m6A RNA methylation in human cancer. Mol Cancer 18:103. https://doi.org/10.1186/s12943-019-1033-z
12. Hu B-B, Wang X-Y, Gu X-Y et al (2019) N6-methyladenosine (m6A) RNA modification in gastrointestinal tract cancers: roles, mechanisms, and applications. Mol Cancer 18:178. https://doi.org/10.1186/s12943-019-1099-7
13. Su Y, Huang J, Hu J (2019) m6A RNA methylation regulators contribute to malignant progression and have clinical prognostic impact in gastric cancer. Front Oncol 9:1038. https://doi.org/10.3389/fonc.2019.01038
14. Takakura M, Ishiguro K, Akichika S et al (2019) Biogenesis and functions of aminocarboxypropyluridine in tRNA. Nat Commun 10:5542. https://doi.org/10.1038/s41467-019-13525-3
15. Gkatza NA, Castro C, Harvey RF et al (2019) Cytosine-5 RNA methylation links protein synthesis to cell metabolism. PLoS Biol 17: e3000297. https://doi.org/10.1371/journal.pbio.3000297
16. Li J, Li H, Long T et al (2019) Archaeal NSUN6 catalyzes m5C72 modification on a wide-range of specific tRNAs. Nucleic Acids Res 47:2041–2055. https://doi.org/10.1093/nar/gky1236
17. Šimonová A, Svojanovská B, Trylčová J et al (2019) LC/MS analysis and deep sequencing reveal the accurate RNA composition in the HIV-1 virion. Sci Rep 9:8697. https://doi.org/10.1038/s41598-019-45079-1
18. Sood AJ, Viner C, Hoffman MM (2019) DNAmod: the DNA modification database. J Cheminform 11:30. https://doi.org/10.1186/s13321-019-0349-4
19. Benson DA, Cavanaugh M, Clark K et al (2013) GenBank. Nucleic Acids Res 41. https://doi.org/10.1093/nar/gks1195
20. Pliatsika V, Loher P, Magee R et al (2018) MINTbase v2.0: a comprehensive database for tRNA-derived fragments that includes nuclear and mitochondrial fragments from all The Cancer Genome Atlas projects. Nucleic Acids Res 46:D152–D159. https://doi.org/10.1093/nar/gkx1075
21. Resource Coordinators NCBI (2018) Database resources of the National Center for Biotechnology Information. Nucleic Acids Res 46: D8–D13. https://doi.org/10.1093/nar/gkx1095
22. The UniProt Consortium (2017) UniProt: the universal protein knowledgebase. Nucleic Acids Res 45:D158–D169. https://doi.org/10.1093/nar/gkw1099
23. Burley SK, Berman HM, Bhikadiya C et al (2019) RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. Nucleic

Acids Res 47:D464–D474. https://doi.org/10.1093/nar/gky1004

24. Bachellerie JP, Cavaillé J, Hüttenhofer A (2002) The expanding snoRNA world. Biochimie 84:775–790. https://doi.org/10.1016/s0300-9084(02)01402-5

25. Braschi B, Denny P, Gray K et al (2019) Genenames.org: the HGNC and VGNC resources in 2019. Nucleic Acids Res 47:D786–D792. https://doi.org/10.1093/nar/gky930

26. Piekna-Przybylska D, Decatur WA, Fournier MJ (2007) New bioinformatic tools for analysis of nucleotide modifications in eukaryotic rRNA. RNA 13:305–312. https://doi.org/10.1261/rna.373107

Chapter 27

MeT-DB V2.0: Elucidating Context-Specific Functions of N6-Methyl-Adenosine Methyltranscriptome

Hui Liu, Jiani Ma, Jia Meng, and Lin Zhang

Abstract

N6-methyladenosine (m^6A) is the most prevalent posttranscriptional modification in eukaryotes and plays a pivotal role in various biological processes. A knowledge base with the systematic collection and curation of context specific transcriptome-wide methylations is critical for elucidating their biological functions as well as for developing bioinformatics tools. In this chapter, we present a comprehensive platform MeT-DB V2.0 for elucidating context-specific functions of N6-methyl-adenosine methyltranscriptome. Met-DB V2.0 database contains context specific m6A peaks and single-base sites predicted from 185 samples for 7 species from 26 independent studies. Moreover, it is also integrated with a new database for targets of m6A readers, erasers and writers and expanded with more collections of functional data. The Met-DB V2.0 web interface and genome browser provide more friendly, powerful, and informative ways to query and visualize the data. More importantly, MeT-DB V2.0 offers for the first time a series of tools specifically designed for understanding m6A functions. The MeT-DB V2.0 web server is freely available at: http://compgenomics.utsa.edu/MeTDB and www.xjtlu.edu.cn/metdb2.

Key words Epitranscriptome, RNA modifications, N6-methyladenosine (m^6A), MicroRNA, Single-nucleotide polymorphisms (SNPs), Splicing factors

1 Introduction

Over a hundred fifty types of chemical modifications have been identified in messenger RNAs (mRNAs) and noncoding RNAs (ncRNAs) [1]. Among them, N6-methyladenosine (m^6A) is characterized as the most abundant and reversible RNA modification [2, 3]. Increasing studies suggest that m^6A has emerged as a critical regulator of posttranscriptional gene expression programs and involves with many cellular activities including splicing [4], translation efficiency [5], regulation, and gene expression. In addition, protein "writers", "erasers", and "readers" of m6A have been discovered to mediate the biological function of m^6A [6]. Moreover, dysregulation of m^6A can contribute to many complex human diseases [7, 8]. However, to our knowledge, m^{6}A modifications and epigenomic data have not been curated well together.

With the development of the next-generation sequencing (NGS) technologies, MeRIP-seq [9, 10] has developed to explore the distributions and quantitative features of m6A modifications across the entire transcriptome, paving the path for understanding their biological functions. The availability of large transcriptome-wide data sets for m^{6}A modification have stimulated the need to develop novel tools and databases for exploring the prevalence, mechanism, and function of m6A modifications.

Here, we present a comprehensive platform MeT-DB V2.0 [11] for annotating, visualizing, analyzing and discovering the m6A modification from large-scale modification sequencing data. MeT-DB V2.0 includes a comprehensive collection of m^{6}A sites predicted from 185 samples for 7 species from 26 independent studies. Moreover, it is also integrated with a new database for targets of m6A readers, erasers and writers and expanded with more collections of functional data. To enhance its utility, MeT-DB V2.0 also integrates functional data such as micro-RNA target sites, Single nucleotide polymorphisms (SNPs), binding-sites of splicing factor as well as RNA-binding protein, and information about cancer genes. In addition, the redesigned Met-DB V2.0 web interface and genome browser provide more friendly, powerful, and informative ways to query and visualize the data. More importantly, MeT-DB V2.0 offers for the first time a series of tools specifically designed for understanding m^{6}A functions.

2 Material

2.1 Hardware

Linux, Unix, Windows, or Macintosh workstation with an Internet connection.

2.2 Software

We recommend that you always use the latest version of Chrome as browser and set the resolution greater than 1280 $\times$ 720 for better browsing experience. If you use other browsers, you might notice that some functions and features would not working properly.

3 Methods

The methods presented here describe how to annotate, visualize, and analyze the MeRIP-Seq data, the targets of m^{6}A readers, erasers, and writers, as well as other functional data associated with m6A modification sites (*see* Fig. 1).

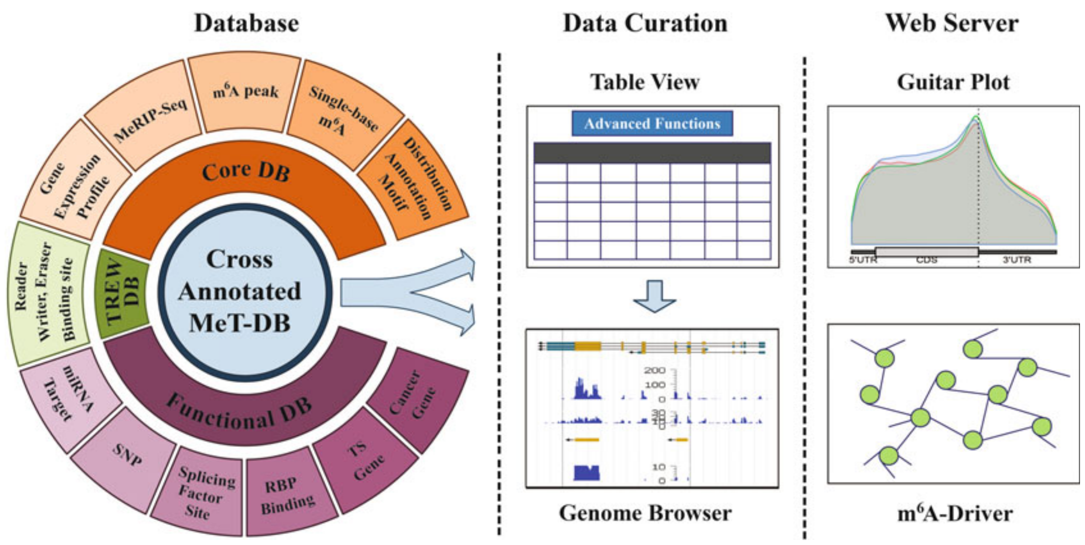


Fig. 1 Overall design of MeT-DB V2.0 database. MeT-DB V2.0 is composed of the database and web interface. The MeT-DB V2.0 database includes the core database that contains context-specific m^{6}A peaks and single-base sites, the TREW database that contains target sites of m^{6}A readers, writers and erasers, and the functional database such as micro-RNA target sites, binding sites of RNA binding proteins and information about cancer genes. There are three functional modules in the web interface: table view facilitates researcher to explore and search the data in detail, the genome browser helps the user visualize and compare m^{6}A peaks and functions data, and the tool module includes two useful web servers for investigating the functions of m^{6}A methyltranscriptome

3.1 Data Processing in MeT-DB V2.0 Database

The MeT-DB database consists of three subdatabases, core database, TREW database and functional database. In this section, we summarized all data we need to construct the MeT-DB V2.0 database.

3.1.1 Identifying Context-Specific m^{6}A Peaks and Single-Base Sites for Core Database

High-throughput m6A-seq or MeRIP-seq technology provides a powerful way to identify the context-specific m^{6}A peaks and single-base sites. In this section, we summarized the features and workflow used to identify context-specific m^{6}A peaks and single-base sites from MeRIP-seq data in core database (*see* Fig. 2).

1. Quality control. Sequencing data quality was first evaluated by FASTQC (v0.11.4). Adaptors or low-quality nucleotides were removed by Trim Galore (v0.4.2) according to the evaluation results of FastQC.
2. Mapping processed m^{6}A-seq reads to reference genome. Reads in the IP/Input FASTQ files (*see* **Note 1**) were aligned to the genome by Tophat2 (v2.1.0) [12] with default options to generate IP/Input BAM Files (*see* **Note 2**). BAM files were subsequently converted to bigwig files for visualization.
3. Identifying context-specific m^{6}A peaks. Peak calling was performed on the input and IP BAM files by exomePeak [13]. For

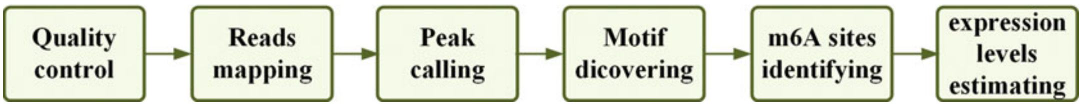


Fig. 2 The workflow to identify context-specific m6A peaks and single-base sites from MeRIP-seq data in core database. The workflow is divided into several main stages, including quality control, reads mapping, peak calling, motif enrichment, dentification of m^{6}A sites with consensus motif and transcriptome-wide expression levels calculating

each predicted m^{6}A peak, its chromosomal location including start/end position, strand information, *P*-value, fold enrichment, and q-value (FDR) were reported.

4. Discovering motif from m^{6}A peaks. For each sample, sequence motifs of predicted m^{6}A peaks were obtained using the MEME (v4.11.2) [14] suite and the peak distribution at a transcript level was also plotted by the Guitar package [15].
5. Identifying single-base m^{6}A sites. Single-base m6A sites were also predicted by searching the RRACH motif in peak[9]s identified by exomePeak. Transcript sequences of the peak region that contain only exons were first extracted, from which the location of RRACH motifs was identified. The genome positions of "A" in the identified motifs were annotated as single-base m^{6}A sites.
6. Calculating transcriptome-wide expression levels. Transcriptome-wide expression levels for each sample were also calculated based on the aligned BAM files of the MeRIP-seq input samples; cufflinks (v4.11.2) [16] with default settings was employed to calculate the gene/isoform expression fragments per kilobase of transcript per million mapped reads (FPKM) values and the reads counts generated by HTSeq (v0.6.1) [17] were also provided to facilitate further analysis.

3.1.2 Identifying m^{6}A Targets of m^{6}A Readers, Erasers, and Writers for TREW Database

TREW or the target of m^{6}A readers, erasers, and writers is our newly constructed database about the binding sites of m^{6}A methyltransferases (METTL3, WTAP, METTL14, and KIAA1429), demethylases (FTO and ALKBH5) and readers (YTH family proteins). Like the workflow talked in the section above, ParCLIP-seq data were retrieved directly from original publications, where the raw data were first processed with Trim Galore and FASTXToolkit (v0.0.13) for quality control, and then aligned to human hg19 or mouse mm10 reference genome respectively with Tophat2. Also, differential m6A analysis was performed with exomePeak and QNB [18] packages under the default setting on MeRIP-seq data of m^{6}A methylase or demethylase perturbation. The significant differential m^{6}A peaks after perturbation were determined to the target peaks. Information including overlapped binding sites of m^{6}A readers, writers, and erasers is provided in the TREW column in the web-interface.

3.1.3 Collecting Relevant Functional Datasets to Discover the Association Between m^{6}A Modification Sites and Them

To help understand the regulatory roles of m^{6}A, we also integrated the following six relevant functional datasets into the Met-DB v2.0 database.

1. miRNA target sites. Predicted miRNA target sites from TargetScan (version 7.1) [19] for human and mouse as well as from miRanda for human, mouse, and fly (August 2010 release) [20] were included. Furthermore, experimentally validated miRNA and target genes interaction pair information for human, mouse, zebrafish, and fly were downloaded from miRTarBase [21].
2. Splicing factor binding sites. A total of 655 and 125 binding sites of human and mouse splicing factors, respectively, were obtained from SpliceAidF (v1.1 03/2013) [22]. Each site includes the name of the binding splicing factor and the genome location of the binding site. SNP. 40 627 human literature-derived collected SNP-trait associations of 30 044 SNPs obtained from GWAS (All associations v1.0.1) [23] were included.
3. RBP-binding site. PAR-CLIP and HITS-CLIP detected binding locations of 24 human and 5 mouse RBPs were retrieved from StarBase version 2.0 [24] and included in the database.
4. *Cancer related genes.* A total of 761 human and 628 mouse tumor suppressor genes were obtained from TSGene database version 2.0 [25]. Also, 576 cancer genes were downloaded from COSMIC (v82) [26].

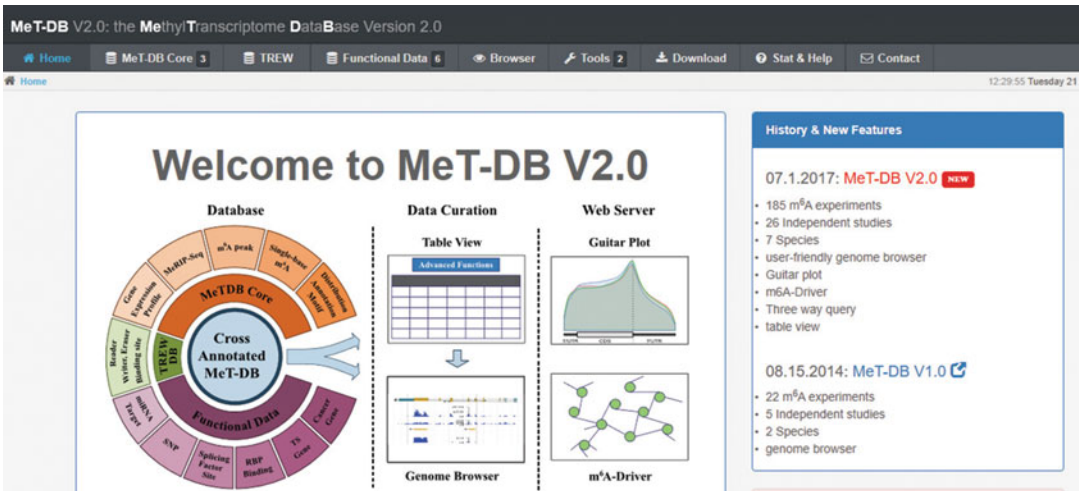
These relevant functional data are displayed as part of the query output of the core Met-DB database. Each functional dataset can also be downloaded from the website.

3.2 Obtaining the Desired m6A Data by Exploring the Table View Interface and Querying the Database

A MeT-DB V2.0 website (www.xjtlu.edu.cn/metdb2 and http://compgenomics.utsa.edu/MeTDB) was developed to support direct query of MeT-DB V2.0 database (*see* Fig. 3).

3.2.1 Table View

A table view (*see* Fig. 4) presents information of every genomic features of an m^{6}A peak/site as a row, thus simultaneously offering a large amount of information about m^{6}A. It can also show the potential interactions between a particular m^{6}A peak/site with other transcriptome features and functional data. Presenting this rich information can provide multifaceted perspectives of m^{6}A methylation and help generate hypotheses of their potential biological functions.



07.1.2017: MeT-DB V2.0 NEW

- 185 m^6A experiments
- 26 Independent studies
- 7 Species
- user-friendly genome browser
- Guitar plot
- m6A-Driver
- Three way query
- table view

08.15.2014: MeT-DB V1.0

- 22 m^6A experiments
- 5 Independent studies
- 2 Species
- genome browser

Fig. 3 MeT-DB V2.0 web server

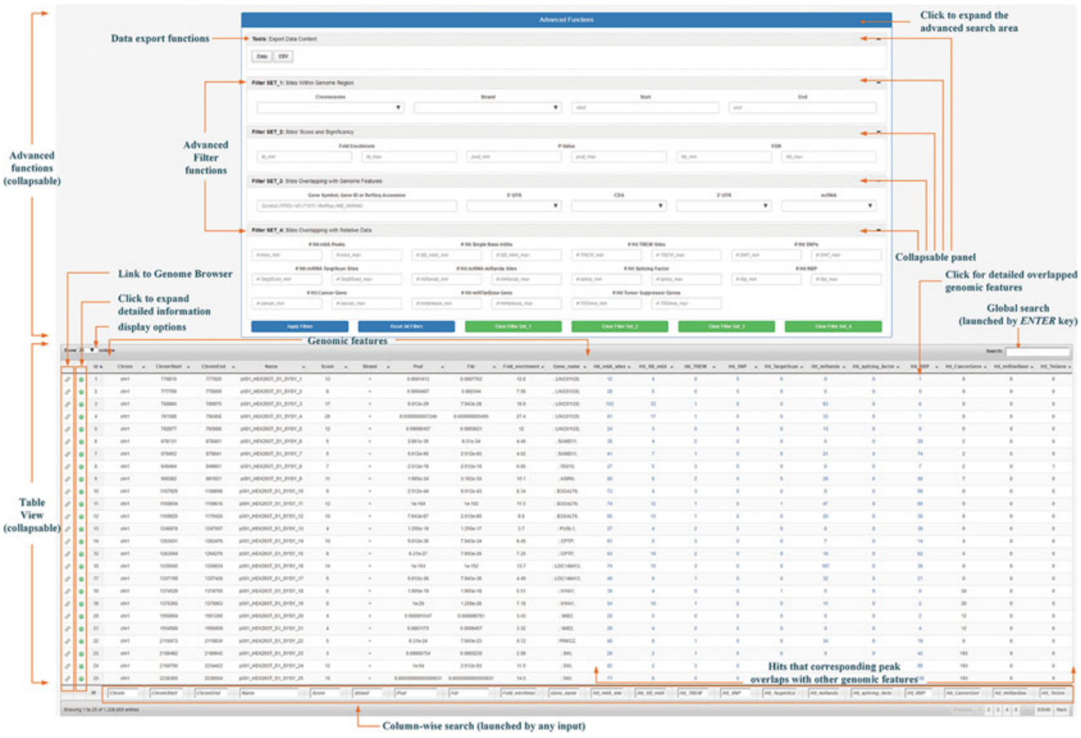

Advanced Functions
Data export functions
Tools: Export Data Content
Copy
CSV
Filter SET_1: Sites Within Genome Region
Filter SET_2: Sites' Score and Significancy
Filter SET_3: Sites Overlapping with Genome Features
Filter SET_4: Sites Overlapping with Relative Data
Advanced functions (collapsable)
Advanced Filter functions
Link to Genome Browser
Click to expand detailed information display options
Genomic features
Click to expand the advanced search area
Collapsable panel
Click for detailed overlapped genomic features
Global search (launched by ENTER key)
Table View (collapsable)
Hits that corresponding peak overlaps with other genomic features
Column-wise search (launched by any input)

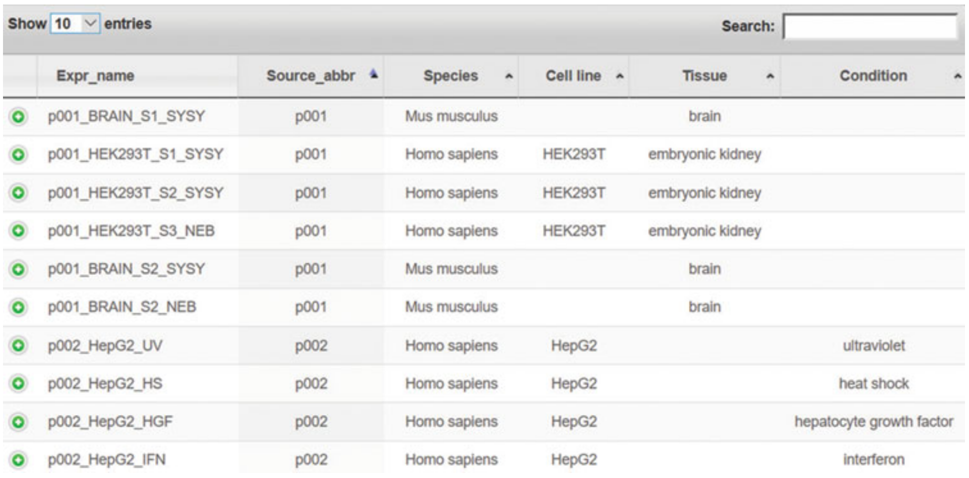
Fig. 4 Illustration of table view. Table view is capable of displaying background data clearly and providing several search methods. To be more specific, global search can query input information within entire table. Column specific search, located in the footer of table, can perform search within corresponding column. More important, we designed variety of advance search functions for different tables to further assistance users to screen out most valuable elements among huge information stored in background database. Users can export data by clicking export buttons under any search conditions for following investigate. Besides, users are able to view detail information in genome browser of a specific entry by clicking the genome browser icon at the very left end of each row

3.2.2 Querying the Database from a Global Perspective to a Single-Nucleotide Resolution

Three different ways of querying the database are made available through the MeT-DB V2.0 web page, namely by samples, by m^{6}A peaks and by single-base m^{6}A sites, providing information about m^{6}A methyl transcriptome at different scales ranging from a global perspective to a single nucleotide resolution.

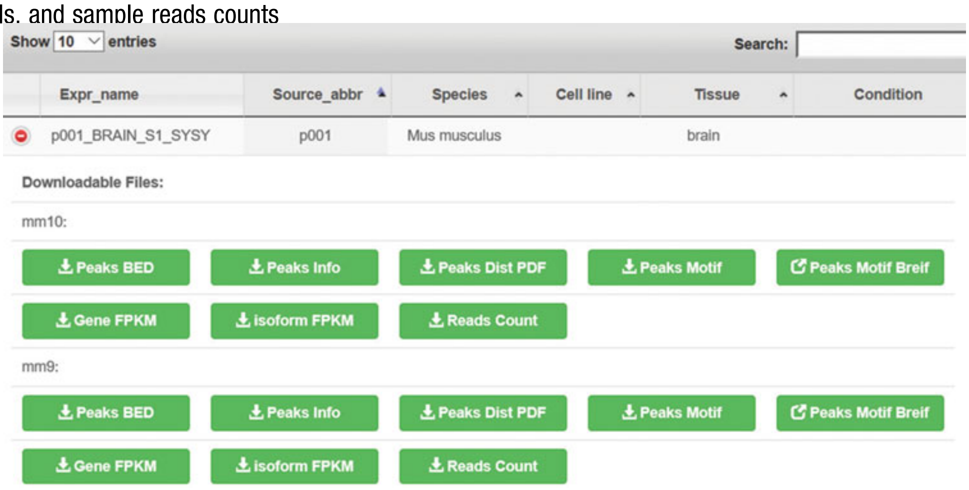
Querying by Samples

The search-by-sample function aims to provide the user with a comprehensive view of context specific m^{6}A methyltranscriptome. Click on" MeT-DB Core" → "by Samples" to query the database by samples. The query page and result page are shown in Fig. 5.



	Expr_name	Source_abbr	Species	Cell line	Tissue	Condition
⊕	p001_BRAIN_S1_SYSY	p001	Mus musculus		brain	
⊕	p001_HEK293T_S1_SYSY	p001	Homo sapiens	HEK293T	embryonic kidney	
⊕	p001_HEK293T_S2_SYSY	p001	Homo sapiens	HEK293T	embryonic kidney	
⊕	p001_HEK293T_S3_NEB	p001	Homo sapiens	HEK293T	embryonic kidney	
⊕	p001_BRAIN_S2_SYSY	p001	Mus musculus		brain	
⊕	p001_BRAIN_S2_NEB	p001	Mus musculus		brain	
⊕	p002_HepG2_UV	p002	Homo sapiens	HepG2		ultraviolet
⊕	p002_HepG2_HS	p002	Homo sapiens	HepG2		heat shock
⊕	p002_HepG2_HGF	p002	Homo sapiens	HepG2		hepatocyte growth factor
⊕	p002_HepG2_IFN	p002	Homo sapiens	HepG2		interferon

Fig. 5 Querying by samples. (**a**) The query takes sample ID, experiments, species, cell line or tissue as input and returns transcriptome-wide information about m^{6}A. (**b**) For each returned sample, the user can investigate the m^{6}A peak distribution at a transcript level and m^{6}A peak sequence motif and download the BED file, detailed information of the transcriptome-wide predicted m6A peaks, the gene and isoform expression FPKMs, and sample reads counts



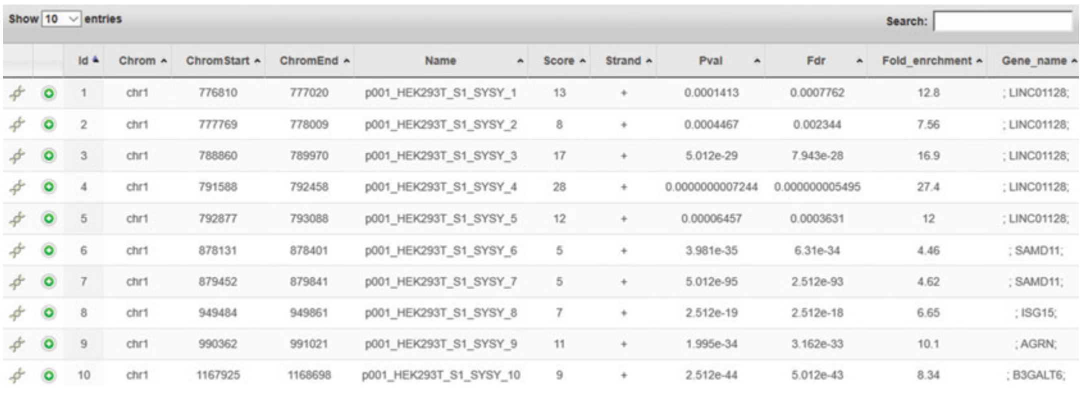
	Expr_name	Source_abbr	Species	Cell line	Tissue	Condition
⊖	p001_BRAIN_S1_SYSY	p001	Mus musculus		brain	

Downloadable Files:

mm10:

mm9:

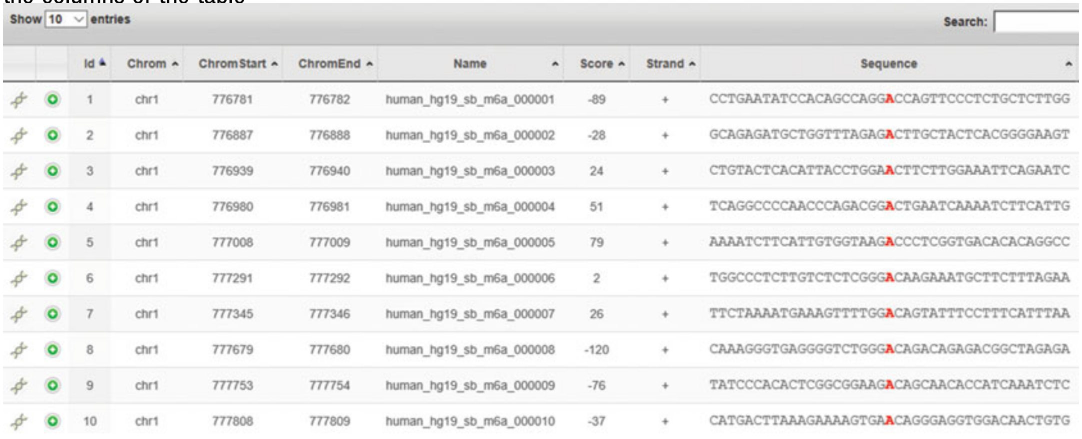
Fig. 5 (continued)



Show 10 entries | Search:

		Id	Chrom	ChromStart	ChromEnd	Name	Score	Strand	Pval	Fdr	Fold_enrchment	Gene_name
		1	chr1	776810	777020	p001_HEK293T_S1_SYSY_1	13	+	0.0001413	0.0007762	12.8	; LINC01128;
		2	chr1	777769	778009	p001_HEK293T_S1_SYSY_2	8	+	0.0004467	0.002344	7.56	; LINC01128;
		3	chr1	788860	789970	p001_HEK293T_S1_SYSY_3	17	+	5.012e-29	7.943e-28	16.9	; LINC01128;
		4	chr1	791588	792458	p001_HEK293T_S1_SYSY_4	28	+	0.0000000007244	0.000000005495	27.4	; LINC01128;
		5	chr1	792877	793088	p001_HEK293T_S1_SYSY_5	12	+	0.00006457	0.0003631	12	; LINC01128;
		6	chr1	878131	878401	p001_HEK293T_S1_SYSY_6	5	+	3.981e-35	6.31e-34	4.46	; SAMD11;
		7	chr1	879452	879841	p001_HEK293T_S1_SYSY_7	5	+	5.012e-95	2.512e-93	4.62	; SAMD11;
		8	chr1	949484	949861	p001_HEK293T_S1_SYSY_8	7	+	2.512e-19	2.512e-18	6.65	; ISG15;
		9	chr1	990362	991021	p001_HEK293T_S1_SYSY_9	11	+	1.995e-34	3.162e-33	10.1	; AGRN;
		10	chr1	1167925	1168698	p001_HEK293T_S1_SYSY_10	9	+	2.512e-44	5.012e-43	8.34	; B3GALT6;

Fig. 6 (**a**) Querying result by m^{6}A peaks. (**b**) Querying result by single-base m6A sites. For each peak/site, the genomic location, associated gene name/ID, peak enrichment fold change, prediction confidence and the number of other peaks/sites from other samples that overlap with this queried m6A peak/site are displayed in the columns of the table



Show 10 entries | Search:

		Id	Chrom	ChromStart	ChromEnd	Name	Score	Strand	Sequence
		1	chr1	776781	776782	human_hg19_sb_m6a_000001	-89	+	CCTGAATATCCACAGCCAGGACCAGTTCCCTCTGCTCTTGG
		2	chr1	776887	776888	human_hg19_sb_m6a_000002	-28	+	GCAGAGATGCTGGTTTAGAGACTTGCTACTCACGGGGAAGT
		3	chr1	776939	776940	human_hg19_sb_m6a_000003	24	+	CTGTACTCACATTACCTGGAACTTCTTGGAAATTCAGAATC
		4	chr1	776980	776981	human_hg19_sb_m6a_000004	51	+	TCAGGCCCCAACCCAGACGGACTGAATCAAAATCTTCATTG
		5	chr1	777008	777009	human_hg19_sb_m6a_000005	79	+	AAAATCTTCATTGTGGTAAGACCCTCGGTGACACACAGGCC
		6	chr1	777291	777292	human_hg19_sb_m6a_000006	2	+	TGGCCCTCTTGTCTCTCGGGACAAGAAATGCTTCTTTAGAA
		7	chr1	777345	777346	human_hg19_sb_m6a_000007	26	+	TTCTAAAATGAAAGTTTTGGACAGTATTTCCTTTCATTTAA
		8	chr1	777679	777680	human_hg19_sb_m6a_000008	-120	+	CAAAGGGTGAGGGGTCTGGGACAGACAGAGACGGCTAGAGA
		9	chr1	777753	777754	human_hg19_sb_m6a_000009	-76	+	TATCCCACACTCGGCGGAAGACAGCAACACCATCAAATCTC
		10	chr1	777808	777809	human_hg19_sb_m6a_000010	-37	+	CATGACTTAAAGAAAAGTGAACAGGGAGGTGGACAACTGTG

Fig. 6 (continued)

Querying by m6A Peaks and Single-Base m^{6}A Sites

The search-by-peak and search-by-site functions instead deliver information about predicted m^{6}A peaks and single-base site integrated from all the samples and display it in the table view interface. Click on "MeT-DB Core" → "by m^{6}A peaks" to query the database by peaks. Similarly, click on" MeT-DB Core" → "by Single-base m^{6}A sites" to query the database by single-base m^{6}A sites. In this section, we take hg19 as an example, the query results are separately shown in the Fig. 6a, b.

3.2.3 Exploring the Functional Data Using Functional Database

To further facilitate functional discovery, additional functional information including overlapped binding sites of m^{6}A readers, writers, and erasers; binding sties of RNA-binding proteins and slicing factors, miRNA target sites; and SNPs and status of its association with a tumor suppressor or a cancer gene is also provided in the "Functional Data" column of the table view.

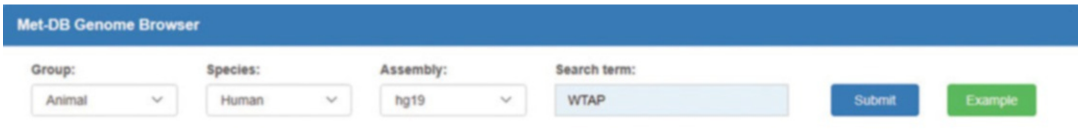


Fig. 7 MeT-DB Genome Browser. (**a**) The location and feature information about WTAP. Users can click the "+" or "−" button at the top to shrink or extend on the center of the annotation tracks window. Users can open the track selection panel by clicking "Select Tracks" button located on the upper-left corner and choose different datasets derived from various cell lines or treatments in MeT-DB. Users can open the track selection panel by clicking "Select Tracks" button located on the upper-left corner and choose different datasets derived from various cell lines or treatments in MeT-DB

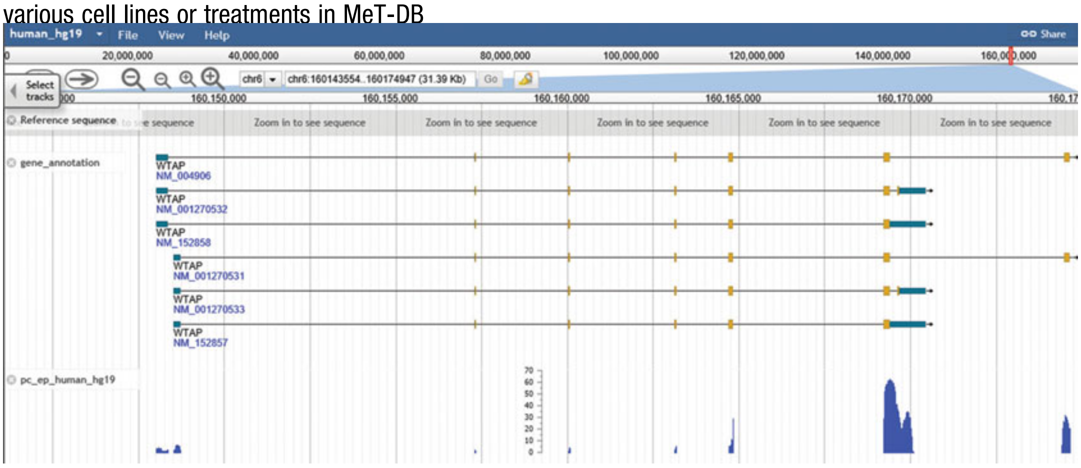


Fig. 7 (continued)

3.3 Genome Browser

Met-DB Genome Browser is built with JBrowse, which is a fast, smooth scrolling, and zooming genome browser. Met-DB Genome Browser can be used to browse m^{6}A peak, TREW target sites, and other functional data along genome.

1. Click "Browser" to open the Genome browser page.
2. Select the Group, Species, Assembly and the gene you interest, for example WTAP (*see* Fig. 7a). Then the users can gain the genome browser of interest (*see* Fig. 7b).

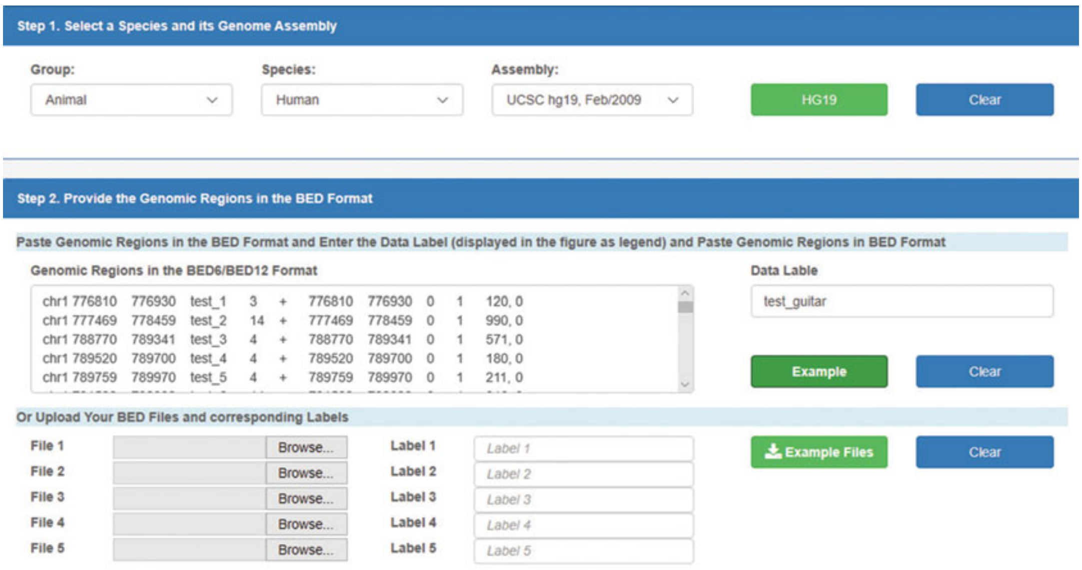
Moreover, by clicking the very left DNA icon of each record in table view interface, users can also take a detail look of the information of the specific record under genome browser view.

3.4 Online Bioinformatics Tools

MeT-DB v2.0 offers two Bioinformatics tools specifically designed for understanding m^{6}A functions.

3.4.1 Guitar Plot

Guitar plot is a tool designed for visualizing the transcript level m^{6}A distribution. It can also be used to visualize distributions of any other type of genome features and transcriptome. Methylations stored in BED files (*see* **Note 3**). MeT-DB V2.0 includes a web server for Guitar to generate the plots of m6A distribution from



Step 1. Select a Species and its Genome Assembly

Group: Animal

Species: Human

Assembly: UCSC hg19, Feb/2009

HG19 | Clear

Step 2. Provide the Genomic Regions in the BED Format

Paste Genomic Regions in the BED Format and Enter the Data Label (displayed in the figure as legend) and Paste Genomic Regions in BED Format

Genomic Regions in the BED6/BED12 Format

```
chr1 776810  776930  test_1   3   +   776810  776930  0   1   120, 0
chr1 777469  778459  test_2   14  +   777469  778459  0   1   990, 0
chr1 788770  789341  test_3   4   +   788770  789341  0   1   571, 0
chr1 789520  789700  test_4   4   +   789520  789700  0   1   180, 0
chr1 789759  789970  test_5   4   +   789759  789970  0   1   211, 0
```

Data Lable: test_guitar

Example | Clear

Or Upload Your BED Files and corresponding Labels

File 1	Browse...	Label 1	Label 1
File 2	Browse...	Label 2	Label 2
File 3	Browse...	Label 3	Label 3
File 4	Browse...	Label 4	Label 4
File 5	Browse...	Label 5	Label 5

Example Files | Clear

Fig. 8 Guitar plot web interface

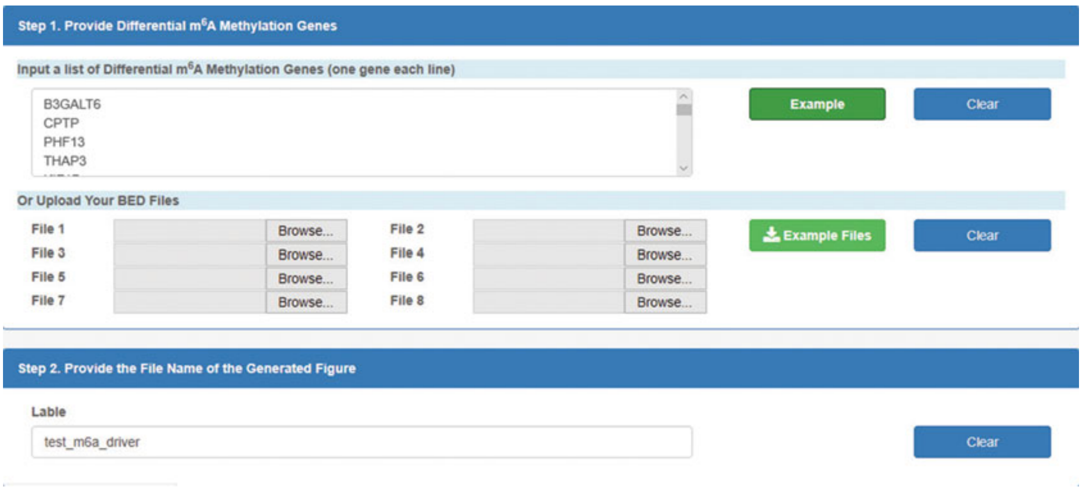
user custom data. The web server brings the full capability of Guitar to generate not only the distribution plot of a single sample but also the plot that compares distributions from multiple samples. The plot is in PDF format and is publication ready. We take hg19 as an example and describe how to use the MeT-DB V2.0 website to browse the detailed information of modification sites.

1. Click "Tools" → "Guitar plot" to open the Guitar Plot web server.
2. Select a Species and its Genome Assembly. For example, choose the "Animal" → "Human" → "UCSC hg19,Feb/2009."
3. Provide the Genomic Regions in the BED Format.
4. Leave Your Email, Submit, and Wait around 10 min to Receive the Results, then the Result.log and Result.pdf can be reached once the job is done (*see* Fig. 8).

3.4.2 m^{6}A-Driver

m^{6}A-Driver is a tool for predicting m^{6}A-driven genes and associated networks, whose functional interactions are likely to be actively modulated by m^{6}A methylation under a particular condition (e.g., disease or gene knockout).

1. Click "Tools" → " m^{6}A-Driver" to open the m^{6}A-Driver web server.
2. Input txt files of lists of official gene symbols. The txt files represent a replicate sample of context-specific m^{6}A targeted genes under a case–control condition obtained from, for instance, differential m6A analysis using exomePeak or Met-Diff. m^{6}A-Driver predicts m^{6}A driven genes from the provided


Step 1. Provide Differential m^{6}A Methylation Genes

Input a list of Differential m^{6}A Methylation Genes (one gene each line)

B3GALT6
CPTP
PHF13
THAP3

Or Upload Your BED Files

Step 2. Provide the File Name of the Generated Figure

Lable

test_m6a_driver

Fig. 9 m^{6}A-driver web interface

gene lists by assessing their topological and biological significance using a random walk with restart algorithm applied to the protein–protein interaction network (*see* Fig. 8).

3. The output of m^{6}A-Driver web server contains two files; one is a text that contains m^{6}A driven genes and the edges of underlying m^{6}A driven gene network and the other file is the figure of the network (Fig. 9).

4 Notes

1. The format of raw sequencing read is FASTQ. Each read records consists of four lines. Line 1 begins with a "@" character and is followed by a sequence identifier. Line 2 is the raw sequence letters. Line 3 begins with a "+" character and is often followed by the same sequence identifier. Line 4 encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as.
2. BAM format of alignments. BAM is the binary format of Sequence Alignment/Map file. Please visit the following website (http://genome.sph.umich.edu/wiki/SAM) to see the detailed information.
3. The annotation data is in UCSC BED and BED12 format. The upload file with the BED format includes chromosome, start position, end position, name, score, and strand direction in web-based modTool server. It should be noted that all start coordinates are 0-based in the RMBase platform. Moreover, the BED12 format includes 12 items. The detailed information is as follows. URL: http://genome.ucsc.edu/FAQ FAQformat.html#format1.

References

1. Boccaletto P et al (2018) MODOMICS: a database of RNA modification pathways. 2017 update. Nucleic Acids Res 46(D1): D303–D307
2. Yang Y et al (2018) Dynamic transcriptomic m (6)A decoration: writers, erasers, readers and functions in RNA metabolism. Cell Res 28 (6):616–624
3. Shi H, Wei J, He C (2019) Where, when, and how: context-dependent functions of RNA methylation writers, readers, and erasers. Mol Cell 74(4):640–650
4. Louloupi A et al (2018) Transient N-6-methyladenosine transcriptome sequencing reveals a regulatory role of m6A in splicing efficiency. Cell Rep 23(12):3429–3437
5. Wang X et al (2015) N-6-methyladenosine modulates messenger RNA translation efficiency. Cell 161(6):1388–1399
6. Fu Y et al (2014) Gene expression regulation mediated through reversible m(6)A RNA methylation. Nat Rev Genet 15(5):293–306
7. Du T et al (2015) An association study of the m6A genes with major depressive disorder in Chinese Han population. J Affect Disord 183:279–286
8. Li L et al (2017) Fat mass and obesity-associated (FTO) protein regulates adult neurogenesis. Hum Mol Genet 26 (13):2398–2411
9. Dominissini D et al (2012) Topology of the human and mouse m(6)A RNA methylomes revealed by m(6)A-seq. Nature 485 (7397):201–U84
10. Meyer KD, Jaffrey SR (2014) The dynamic epitranscriptome: N-6-methyladenosine and gene expression control. Nat Rev Mol Cell Biol 15(5):313–326
11. Liu H et al (2018) MeT-DB V2.0: elucidating context-specific functions of N-6-methyl-adenosine methyltranscriptome. Nucleic Acids Res 46(D1):D281–D287
12. Kim D et al (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. Genome Biol 14(4)
13. Meng J et al (2013) Exome-based analysis for RNA epigenome sequencing data. Bioinformatics 29(12):1565–1567
14. Bailey TL et al (2009) MEME SUITE: tools for motif discovery and searching. Nucleic Acids Res 37:W202–W208
15. Cui X et al (2016) Guitar: an R/Bioconductor package for gene annotation guided transcriptomic analysis of RNA-related genomic features. Biomed Res Int 2016:8367534
16. Roberts A et al (2011) Improving RNA-Seq expression estimates by correcting for fragment bias. Genome Biol 12(3)
17. Anders S, Pyl PT, Huber W (2015) HTSeq-a Python framework to work with high-throughput sequencing data. Bioinformatics 31(2):166–169
18. Liu L et al (2017) QNB: differential RNA methylation analysis for count-based small-sample sequencing data with a quad-negative binomial model. BMC Bioinformatics 18
19. Lewis BP, Burge CB, Bartel DP (2005) Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets. Cell 120 (1):15–20
20. Betel D et al (2010) Comprehensive modeling of microRNA targets predicts functional non-conserved and non-canonical sites. Genome Biol 11(8)
21. Chou C-H et al (2016) miRTarBase 2016: updates to the experimentally validated miRNA-target interactions database. Nucleic Acids Res 44(D1):D239–D247
22. Giulietti M et al (2013) SpliceAid-F: a database of human splicing factors and their RNA-binding sites. Nucleic Acids Res 41 (D1):D125–D131
23. MacArthur J et al (2017) The new NHGRI-EBI catalog of published genome-wide association studies (GWAS catalog). Nucleic Acids Res 45(D1):D896–D901
24. Li J-H et al (2014) starBase v2.0: decoding miRNA-ceRNA, miRNA-ncRNA and protein-RNA interaction networks from large-scale CLIP-Seq data. Nucleic Acids Res 42(D1): D92–D97
25. Zhao M et al (2016) TSGene 2.0: an updated literature-based knowledgebase for tumor suppressor genes. Nucleic Acids Res 44(D1): D1023–D1031
26. Forbes SA et al (2017) COSMIC: somatic cancer genetics at high-resolution. Nucleic Acids Res 45(D1):D777–D783

WHISTLE: A Functionally Annotated High-Accuracy Map of Human m^{6}A Epitranscriptome

Qingru Xu, Kunqi Chen, and Jia Meng

Abstract

N^6-Methyladenosine (m^{6}A) is the most prevalent posttranscriptional modification in eukaryotes and plays a pivotal role in various biological processes, such as splicing, RNA degradation, and RNA–protein interaction. Accurately identification of the location of m^{6}A is essential for related downstream studies. In this chapter, we introduce a prediction framework WHISTLE, which enables us to acquire so far the most accurate map of the transcriptome-wide human m^{6}A RNA-methylation sites (with an average AUC: 0.948 and 0.880 under the full transcript or mature messenger RNA models, respectively, when tested on independent datasets). Besides, each individual m^{6}A site was also functionally annotated according to the "guilt-by-association" principle by integrating RNA methylation data, gene expression data and protein–-protein interaction data. A web server was constructed for conveniently querying the predicted RNA methylation sites and their putative biological functions. The website supports the query by genes, by GO function, table view, and the download of all the functionally annotated map of predicted map of human m^{6}A epitranscriptome. The WHISTLE web server is freely available at: www.xjtlu.edu.cn/biologicalsciences/whistle and http://whistle-epitranscriptome.com.

Key words Epitranscriptome, RNA modifications, *N*^{6}-Methyladenosine (m^{6}A), Guilt-by-association, Machine learning

1 Introduction

More than 150 types of abundant RNA modifications have been revealed and investigated in the human epitranscriptome [1, 2]. Among them, *N*^{6}-methyladenosine (m^{6}A) is the most prevalent modification type present on eukaryotic messenger RNA (mRNA), and has been confirmed to be actively involved in various essential biological processes [3–9].

With the developing of the high-throughput sequencing method, m^{6}A-seq (or MeRIP-seq), the first whole-transcriptome m^{6}A profiling technique, was invented in 2012 [10, 11] and can detect m^{6}A sites with a resolution of around 100 nt [12, 13]. The resolution may be further narrowed down to single base by searching for the m^{6}A motif "RRACH" within the peaks called from m^{6}A-seq data. Since m^{6}A-seq is the mostly widely used approach for profiling m6A epitranscriptome, most existing bioinformatics databases, such as MeT-DB and RMBase [14, 15], heavily relied on this procedure. However, because of the randomly occurring RRACH motif, these databases are likely to contain large number of false positive m^{6}A sites. Meanwhile, true single-based resolution techniques, like miCLIP and m^{6}A-CLIP, were also developed [16, 17]. Unfortunately, limited by the laborious experimental procedures, these technologies have not been very widely adopted, and revealed only limited coverage of the epitranscriptome.

Here, with a high-accuracy map of the human m^{6}A epitranscriptome obtained from our WHISTLE prediction platform [18], we provide a web interface for directly query or download this data as well as the functional annotation information of each individual m^{6}A site. The information could be useful for further exploration of the downstream studies of this RNA modification.

2 Materials

2.1 Hardware

Computers or smartphones with Internet connection.

2.2 Software

The R programming language may be necessary to train the predictive model from the provided data. R for Linux, Mac OS, or Windows can be downloaded from https://www.r-project.org. The IDE of R language RStudio can be installed from https://www.rstudio.com. Those websites provide installation instructions.

2.3 Packages

The following R packages may be required: GenomicFeatures, Randomforest, and Caret.

2.4 Data Sources

The training and benchmarking data enrolled in this study, including six base-resolution miCLIP (or m^{6}A-CLIP) datasets were retrieved from the gene expression omnibus (GEO) as well as the supplementary data of the original references. Performance evaluations were conducted on these datasets as well as the dataset from a different technology, m^{6}A-seq with improved protocol. The search space of the WHISTLE framework [18] is the m^{6}A sites recorded in MeTDB database for the mature mRNA model, or RMBase for the full transcript model. The original base-resolution data has also been deposited into the WHISTLE website.

2.5 Web Browsers

An up-to-date Internet browser, such as Google Chrome (http://www.google.com/chrome), Safari (http://www.apple.com/safari), Internet Explorer (http://www.microsoft.com/windows/internet-explorer/worldwide-sites.aspx), or Firefox (http://www.mpzilla.org/firefox).

3 Methods

We describe here how the WHISTLE prediction framework was built, and how to explore web-based WHISTLE platform to obtain the desired m^{6}A data as well as the corresponding annotation information. The general workflow of WHISTLE framework is shown in Fig. 1.

3.1 Predicting High-Accuracy Map of the Human m^{6}A Epitranscriptome Using a Machine Learning Approach

In this section, we summarized the workflow of WHISTLE framework to predict the m^{6}A sites.

1. Preparing the training and testing datasets. Six single-base resolution m^{6}A experiments obtained from five cell types based on miCLIP (or m^{6}A-CLIP) were used, and two of the hg18 samples (CD8T and A549 m^{6}A-CLIP) were lifted to hg19 using UCSC liftOver tool (http://genome.ucsc.edu/cgi-bin/hgLiftOver).
2. Features for m^{6}A site prediction. Besides the conventional sequence-derived features (*see* **Note 1**), a total of 35 additional genomic-derived features (*see* **Note 2**) were generated for the prediction purposes.

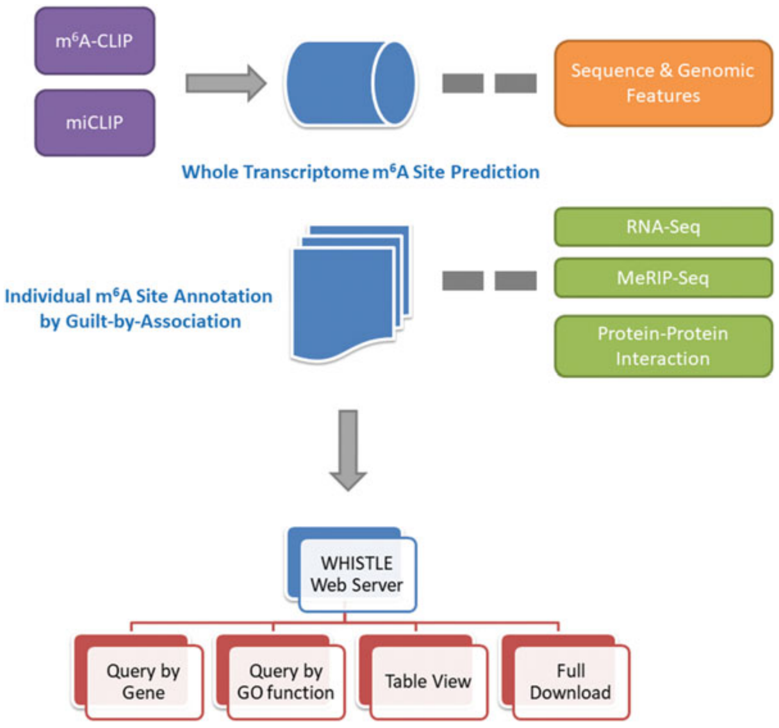

m6A-CLIP
miCLIP
Sequence & Genomic Features
Whole Transcriptome m6A Site Prediction
RNA-Seq
MeRIP-Seq
Protein-Protein Interaction
Individual m6A Site Annotation by Guilt-by-Association
WHISTLE Web Server
Query by Gene
Query by GO function
Table View
Full Download

Fig. 1 Basic framework for obtaining the prediction m^{6}A sites on human transcriptome, and functionally annotating them ^{6}A sites. All results generated by this framework are stored in MySQL database and are displayed in the web browser

3. Selecting the top *N* most significant genomic-derived features with greedy search. By employing the Perturb method [19] in the R caret package, top *N* most important genomic features were selected and retained for further prediction analysis to avoid the overfitting problem.
4. Training and evaluating datasets. For training the SVM and random forest [20] classifiers, a fivefold cross-validation was conducted. Dataset level Leave-one-out experiments, where five samples were used as the training datasets and the remaining one was used as the independent testing, were conducted. To test the performance of WHISTLE framework, it was assessed not only on these six independent datasets but also on the dataset generated from a different technology, m^{6}A-seq with improved protocol [21]. To evaluate the performance of our approach, the area under the ROC (receiver operating characteristic) curve (AUROC) values was calculated and used as the main performance evaluation metric.
5. Generating a map of human m^{6}A epitranscriptome and calculating the posterior probability of m^{6}A site. We applied the WHISTLE framework to the entire transcriptome to search for all the putative m6A sites. The predicted probability (or likelihood) of a particular site to be a real m^{6}A site under a specific model can be obtained for the mature mRNA model and the full transcript model, respectively. Furthermore, by including the prior probability (*see* **Note 3**), more reliable posterior probability is also calculated. The whole predicted map of human m^{6}A epitranscriptome data was deposited into the WHISTLE website (Table 1).

3.2 Obtaining the Desired m^{6}A Data by Exploring the WHISTLE Website

A WHISTLE website (http://whistle-epitranscriptome.com) was developed to support direct query of predicted human RNA methylation sites, their putative functions and potential association to other methylation sites or genes, providing the requisite data for the further epitranscriptome studies.

1. Querying m^{6}A modification sites by function or gene. In the home page of the server, users can select searching type by function or name, and then enter the corresponding search term (Fig. 2a).
2. Click on the "GO" button. Users can gain a list of m^{6}A modification sites of interest (Fig. 2b, c).
3. Table view of m^{6}A modification sites. In the table page of the server, users can view the detailed information in table format (Fig. 3a).
4. Click on an individual m^{6}A site, users can obtain each individual's information as well as its gene ontology annotation (Fig. 3b, c). The Gene Ontology (GO) is a major

Table 1
The 35 genome-derived features used for m^{6}A site prediction

ID	Name	Description	Note
1	UTR5	5′ UTR	Dummy variables indicating whether the site is overlapped to the topological region on the major RNA transcript
2	UTR3	3′ UTR	
3	Stop_codons	Stop codons flanked by 100 bp	
4	Start_codons	Start codons flanked by 100 bp	
5	TSS	Downstream 100 bp of TSS	
6	TSS_A	Downstream 100 bp of TSS on A	
7	exon_stop	Exons containing stop codons	
8	alternative_exon	Alternative exons	
9	constitutive_exon	Constitutive exons	
10	internal_exon	Internal exons	
11	long_exon	long exons (exon length $\geq$ 400 bp)	
12	last_exon_400bp	5′ 400 bp of the last exons	
13	last_exon_sc400	5′ 400 bp of the last exons containing stop codons	
14	pos_UTR5	Relative position on 5′ UTR	Relative position on the region
15	pos_UTR3	Relative position on 3′ UTR	
16	pos_exons	Relative position on exon	
17	length_UTR5	5′ UTR length	The region length in bp
18	length_UTR3	3′ UTR length	
19	length_gene_ex	Mature transcript length	
20	dist_sj_5_p2000	Distance to the 5′ splicing junction	Nucleotide distances toward the splicing junctions or the nearest neighboring sites
21	dist_sj_3_p2000	Distance to the 3′ splicing junction	
22	dist_nearest_p200	Distance to the closest neighbor truncated at 200 bp	
23	PC_1bp	phastCons scores of the nucleotide	Scores related to evolutionary conservation
24	PC_101bp	Average phastCons scores within the flanking 50 bp region	
25	FC_1bp	fitCons scores of the nucleotide	
26	FC_101bp	Average fitCons scores within the flanking 50 bp region	

(continued)

Table 1 (continued)

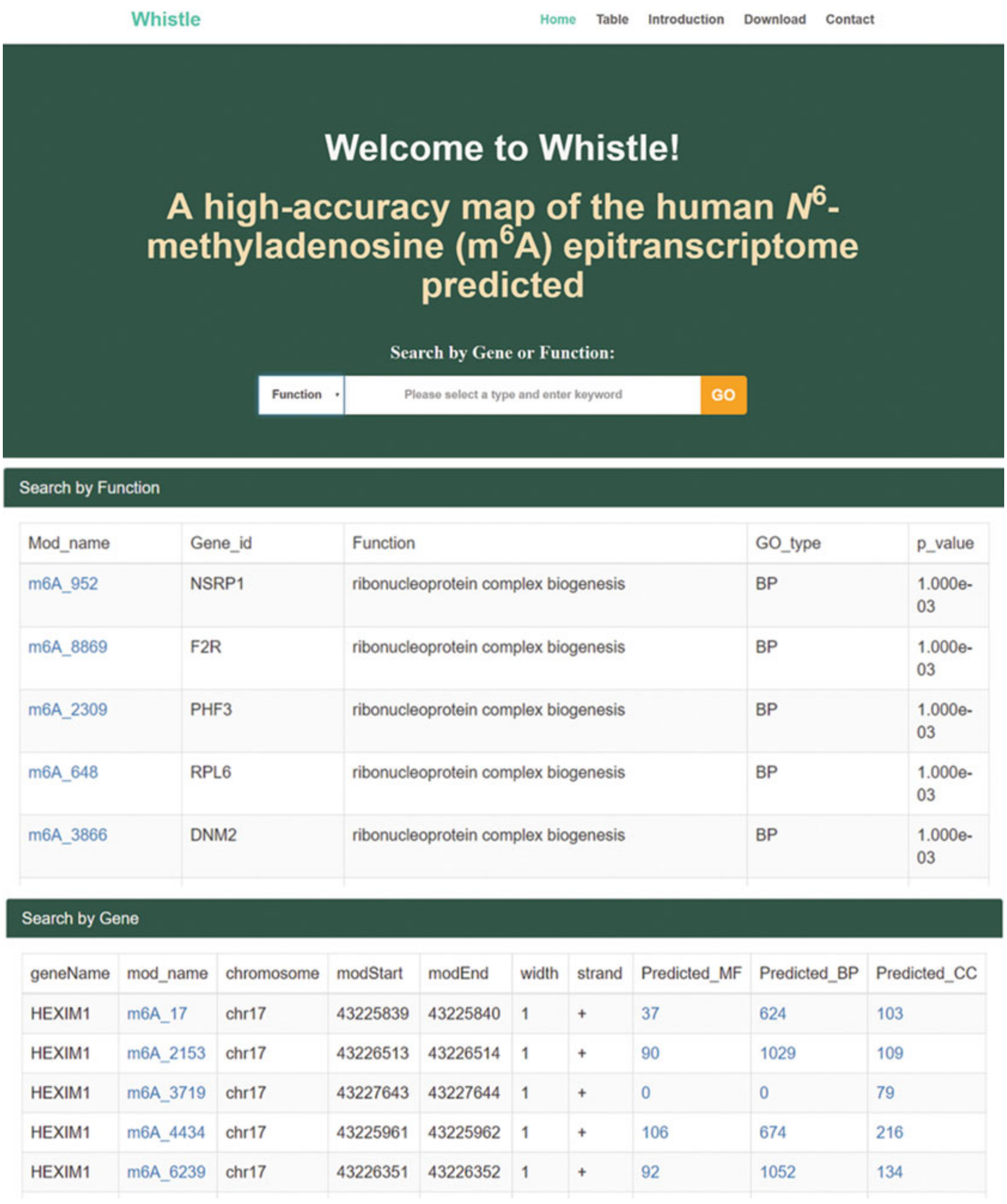
ID	Name	Description	Note
27	struc_hybridize	Predicted RNA hybridized region	RNA secondary structures
28	struc_loop	Predicted RNA loop region	
29	sncRNA	sncRNA	Attributes of the genes or transcripts
30	lncRNA	lncRNA	
31	HK_genes	hoUsekeeping genes	
32	miR_targeted_genes	miRNA targeted genes	
33	HNRNPC_eCLIP	eCLIP data of HNRNPC RNA binding sites	RNA annotations related to m^6A biology
34	Verified_miRtargets	miRNA targeted sites verified by experiment	
35	TargetScan	Predicted miRNA targeted sites by TargetScan	

bioinformatics initiative to unify the representation of gene and gene product attributes across all species; BP strands for biological process, MF strands for molecular function, CC strands for cellular component. Please refer to [18] for detailed methodology of the functional annotation.

5. In the download center of the website, WHISTLE support various data to be downloaded (Table 2), including the original single-base resolution data generated by miCLIP and m^6A-CLIP (#1) or m^6A-seq with improved protocol technology (#2), the predicted m6A sites under full transcript (#3, #4) with high confidence (*see* **Note 4**) and mature mRNA m6A sites (#5, #6) with high confidence, predicted m^6A function (#7) as well as the source code of the project code.

4 Notes

1. Sequence-derived features. This encoding method around the RRACH motif has also been applied by m6Apred and MethyRNA [22, 23] and achieved good performance. In detail, three distinct structural chemical properties: ring structures, functional groups, and hydrogen bonds were used. Specifically, adenine and guanine have two ring structures, while cytosine and uracil have only one ring; adenine and cytosine contain the amino group, while guanine and uracil contain the keto group; adenine and uracil can form two hydrogen bonds during hybridization, whereas guanine and cytosine can form three hydrogen bonds. Based on the three structural chemical



Welcome to Whistle!

A high-accuracy map of the human N^6-methyladenosine (m^{6}A) epitranscriptome predicted

Search by Gene or Function:

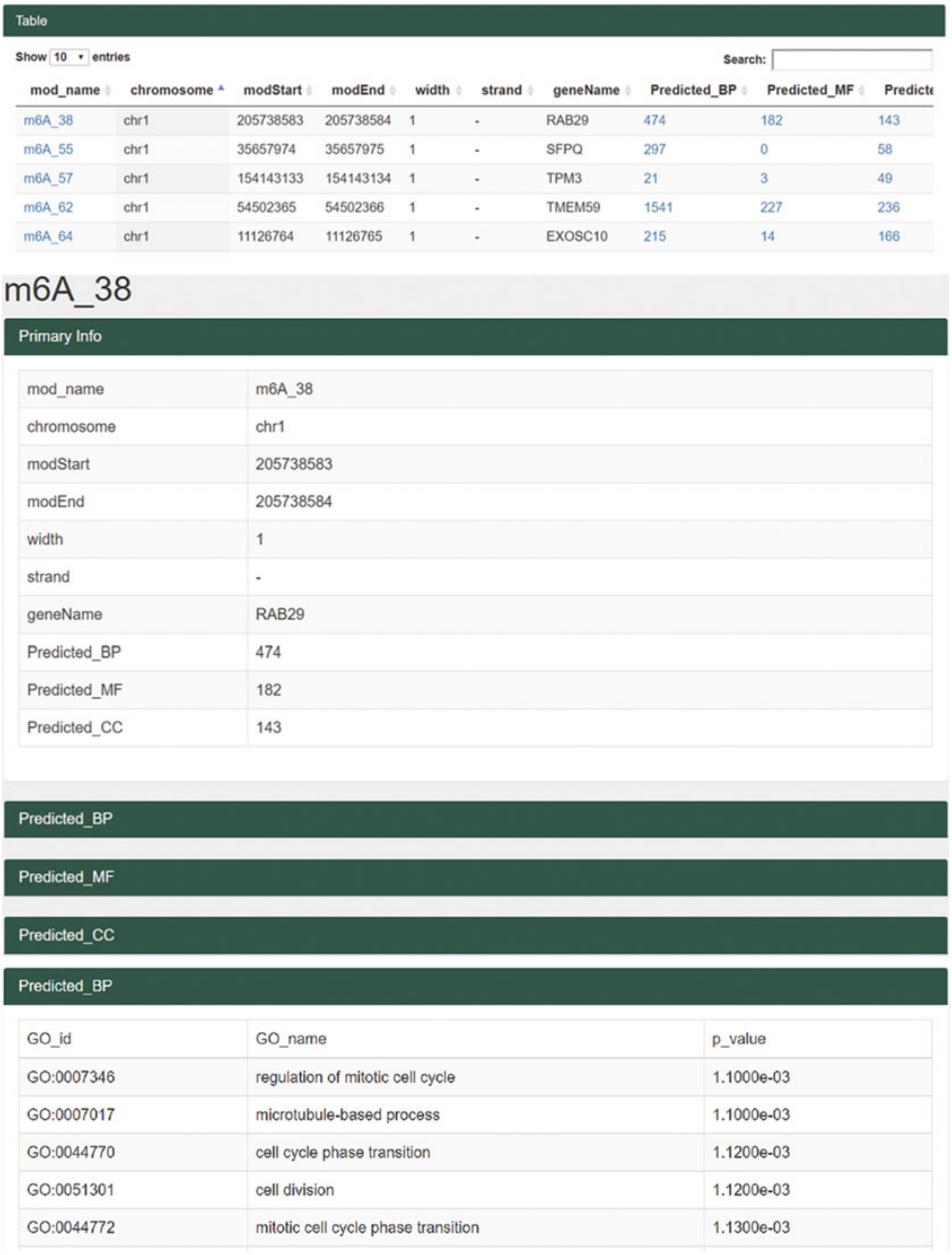
Search by Function

Mod_name	Gene_id	Function	GO_type	p_value
m6A_952	NSRP1	ribonucleoprotein complex biogenesis	BP	1.000e-03
m6A_8869	F2R	ribonucleoprotein complex biogenesis	BP	1.000e-03
m6A_2309	PHF3	ribonucleoprotein complex biogenesis	BP	1.000e-03
m6A_648	RPL6	ribonucleoprotein complex biogenesis	BP	1.000e-03
m6A_3866	DNM2	ribonucleoprotein complex biogenesis	BP	1.000e-03

Search by Gene

geneName	mod_name	chromosome	modStart	modEnd	width	strand	Predicted_MF	Predicted_BP	Predicted_CC
HEXIM1	m6A_17	chr17	43225839	43225840	1	+	37	624	103
HEXIM1	m6A_2153	chr17	43226513	43226514	1	+	90	1029	109
HEXIM1	m6A_3719	chr17	43227643	43227644	1	+	0	0	79
HEXIM1	m6A_4434	chr17	43225961	43225962	1	+	106	674	216
HEXIM1	m6A_6239	chr17	43226351	43226352	1	+	92	1052	134

Fig. 2 Querying m^{6}A modification sites by function or gene from WHISTLE website. (**a**) Select the type of function or gene, and then enter name of interest. (**b**) The search by function results of m^{6}A websites. (**c**) The search by gene results of m^{6}A web sites



Table

Show 10 entries

Search:

mod_name	chromosome	modStart	modEnd	width	strand	geneName	Predicted_BP	Predicted_MF	Predicte
m6A_38	chr1	205738583	205738584	1	-	RAB29	474	182	143
m6A_55	chr1	35657974	35657975	1	-	SFPQ	297	0	58
m6A_57	chr1	154143133	154143134	1	-	TPM3	21	3	49
m6A_62	chr1	54502365	54502366	1	-	TMEM59	1541	227	236
m6A_64	chr1	11126764	11126765	1	-	EXOSC10	215	14	166

m6A_38

Primary Info

mod_name	m6A_38
chromosome	chr1
modStart	205738583
modEnd	205738584
width	1
strand	-
geneName	RAB29
Predicted_BP	474
Predicted_MF	182
Predicted_CC	143

Predicted_BP

Predicted_MF

Predicted_CC

Predicted_BP

GO_id	GO_name	p_value
GO:0007346	regulation of mitotic cell cycle	1.1000e-03
GO:0007017	microtubule-based process	1.1000e-03
GO:0044770	cell cycle phase transition	1.1200e-03
GO:0051301	cell division	1.1200e-03
GO:0044772	mitotic cell cycle phase transition	1.1300e-03

Fig. 3 (**a**) Table view of m^{6}A modification sites. (**b**) Primary information about an individual m^{6}A site. (**c**) The biological processes (BPs) predicted to be involving the m^{6}A site

Table 2
Data types could be downloaded from WHISTLE

1	Base-resolution m^{6}A sites (miCLIP and m6A-CLIP)
2	Base-resolution m^{6}A sites (m^{6}A-seq with improved protocol)
3	Predicted m^{6}A sites—full transcript (complete version)
4	Predicted m^{6}A sites—full transcript (high confidence)[a]
5	Predicted m^{6}A sites—mature mRNA (complete version)
6	Predicted m^{6}A sites—mature mRNA (high confidence)[a]
7	Predicted m^{6}A functions
8	Project source codes

[a]The selection criteria of the predicted high-confidence human m6A sites are as follows: For each gene, up to five most probable m^{6}A sites whose methylation probability is greater than 0.9 are selected
For genes that has less than three sites with methylation probability greater than 0.9, we report up to two most probable sites. (We want the predicted m^{6}A sites to cover a larger proportion of the transcriptome)
For genes that have no reported m^{6}A sites conforming the RRACH motif in MeTDB/RMBase, no sites will be reported. (MeTDB and RMBase collected hundreds of samples, and their data processing pipelines are less likely to make false negative prediction compared to false positive prediction)

properties defined above, the *i*th nucleotide from sequence Scan been coded by a vector $S_i = (x_i, y_i, z_i)$:

$$x_i = \begin{cases} 1 & \text{if } s_i \in \{A, G\} \\ 0 & \text{if } s_i \in \{C, U\} \end{cases}, \quad y_i = \begin{cases} 1 & \text{if } s_i \in \{A, C\} \\ 0 & \text{if } s_i \in \{G, U\} \end{cases}, \quad z_i = \begin{cases} 1 & \text{if } s_i \in \{A, U\} \\ 0 & \text{if } s_i \in \{C, G\} \end{cases} \tag{1}$$

Therefore, the A, C, G, and U can be encoded as a vector of three features (1,1,1), (0,1,0), (1,0,0), and (0,0,1), respectively. Additionally, a feature of the cumulative nucleotide frequency is calculated for each nucleotide position in the sequence. The density of the *i*th nucleotide di is defined as the sum of all the instances of the *i*th nucleotide before the $i + 1$ position. The nucleotide frequency *fi* is defined by the following formula: $f_i = d_i/i$. Using the sequence "AUGGACACU" as an example, the cumulative frequency for adenine is 1.00 (1/1), 0.40 (2/5), and 0.43(3/7) at the first, fifth and seven position, respectively; while the frequency for uracil is 0.50 (1/2) and 0.11(1/9) at the second and ninth respective position.

2. Genomic features. The description of genomic features considered in the prediction was summarized in Table 1.

3. When calculating the posterior probability of RNA methylation, we defined the prior probability as the average number of m^6A sites from the 6 base-resolution databases divided by the number of occurrences of transcriptome RRACH motifs that are supported by at least one m^6A record in MeTDB or RMBase. The likelihood ratio was extracted from the predictive model.
4. The selection criteria of the predicted high-confidence human m^6A sites are as following:
   - For each gene, up to five most probable m^6A sites whose methylation probability is greater than 0.9 are selected;
   - For genes that have less than three sites with methylation probability greater than 0.9, we report up to two most probable sites. (We want the predicted m^6A sites to cover a larger proportion of the transcriptome.)
   - For genes that have no reported m^6A sites conforming to the RRACH motif in MeTDB/RMBase, no sites will be reported. RRACH motifs not captured in MeTDB/RMBase databases were not considered, this is because that MeTDB and RMBase collected hundreds of MeRIP-seq samples, and their data processing pipelines are less likely to make false negative prediction compared to false positive prediction.

References

1. Boccaletto P, Machnicka MA, Purta E, Piątkowski P, Bagiński B, Wirecki TK, de Crécy-Lagard V, Ross R, Limbach PA, Kotter AJ (2017) MODOMICS: a database of RNA modification pathways. 2017 update. Nucleic Acids Res 46(D1):D303–D307
2. Roundtree IA, Evans ME, Pan T, He C (2017) Dynamic RNA modifications in gene expression regulation. Cell 169(7):1187–1200
3. Alarcón CR, Lee H, Goodarzi H, Halberg N, Tavazoie SF (2015) N 6-methyladenosine marks primary microRNAs for processing. Nature 519(7544):482
4. Fustin J-M, Doi M, Yamaguchi Y, Hida H, Nishimura S, Yoshida M, Isagawa T, Morioka MS, Kakeya H, Manabe IJ (2013) RNA-methylation-dependent RNA processing controls the speed of the circadian clock. Cell 155 (4):793–806
5. Liu N, Dai Q, Zheng G, He C, Parisien M, Pan TJ (2015) N 6-methyladenosine-dependent RNA structural switches regulate RNA–protein interactions. Nature 518(7540):560
6. Meyer KD, SRJ J (2014) The dynamic epitranscriptome: N 6-methyladenosine and gene expression control. Nat Rev Mol Cell Biol 15 (5):313
7. Wang X, Lu Z, Gomez A, Hon GC, Yue Y, Han D, Fu Y, Parisien M, Dai Q, Jia GJ (2014) N 6-methyladenosine-dependent regulation of messenger RNA stability. Nature 505 (7481):117
8. Xiang Y, Laurent B, Hsu C-H, Nachtergaele S, Lu Z, Sheng W, Xu C, Chen H, Ouyang J, Wang SJN (2017) RNA m 6 A methylation regulates the ultraviolet-induced DNA damage response. Nature 543(7646):573
9. Zhou J, Wan J, Gao X, Zhang X, Jaffrey SR, Qian S-BJ (2015) Dynamic m 6 A mRNA methylation directs translational control of heat shock response. Nature 526(7574):591
10. Dominissini D, Moshitch-Moshkovitz S, Schwartz S, Salmon-Divon M, Ungar L, Osenberg S, Cesarkas K, Jacob-Hirsch J, Amariglio N, Kupiec MJ (2012) Topology of the human and mouse m 6 A RNA methylomes revealed by m 6 A-seq. Nature 485(7397):201
11. Meyer KD, Saletore Y, Zumbo P, Elemento O, Mason CE, Jaffrey SRJ (2012) Comprehensive analysis of mRNA methylation reveals

enrichment in 3′ UTRs and near stop codons. Cell 149(7):1635–1646

12. Dominissini D, Moshitch-Moshkovitz S, Salmon-Divon M, Amariglio N, Rechavi GJ (2013) Transcriptome-wide mapping of N 6-methyladenosine by m 6 A-seq based on immunocapturing and massively parallel sequencing. Nat Protoc 8(1):176
13. Meng J, Lu Z, Liu H, Zhang L, Zhang S, Chen Y, Rao MK, Huang YJ (2014) A protocol for RNA methylation differential analysis with MeRIP-Seq data and exomePeak R/Bioconductor package. Methods 69(3):274–281
14. Liu H, Wang H, Wei Z, Zhang S, Hua G, Zhang S-W, Zhang L, Gao S-J, Meng J, Chen X Jr (2017) MeT-DB V2. 0: elucidating context-specific functions of N 6-methyl-adenosine methyltranscriptome. Nucleic Acids Res 46(D1):D281–D287
15. Xuan J-J, Sun W-J, Lin P-H, Zhou K-R, Liu S, Zheng L-L, Qu L-H, Yang J-HJ (2017) RMBase v2. 0: deciphering the map of RNA modifications from epitranscriptome sequencing data. Nucleic Acids Res 46(D1):D327–D334
16. Bik HM, Dove ADM, Goldstein MC, Helm RR, MacPherson R, Martini K, Warneke A, McClain C (2015) Ten simple rules for effective online outreach. PLoS Comput Biol 11(4): e1003906. https://doi.org/10.1371/journal.pcbi.1003906
17. Linder B, Grozhik AV, Olarerin-George AO, Meydan C, Mason CE, Jaffrey SRJ (2015) Single-nucleotide-resolution mapping of m6A and m6Am throughout the transcriptome. Nat Methods 12(8):767
18. Chen K, Wu X, Zhang Q, Wei Z, Rong R, Lu Z, Meng J, de Magalhães JP, Su J, Rigden DJ (2019) WHISTLE: a high-accuracy map of the human N6-methyladenosine (m6A) epitranscriptome predicted using a machine learning approach. Nucleic Acids Res https://doi.org/10.1093/nar/gkz074
19. Gevrey M, Dimopoulos I, Lek SJ (2003) Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecol Modell 160(3):249–264
20. Liaw A, Wiener M (2002) Classification and regression by randomForest. Forest 2(3):18–22
21. Schwartz S, Mumbach MR, Jovanovic M, Wang T, Maciag K, Bushkin GG, Mertins P, Ter-Ovanesyan D, Habib N, Cacchiarelli DJ (2014) Perturbation of m6A writers reveals two distinct classes of mRNA methylation at internal and 5′ sites. Cell Rep 8(1):284–296
22. Chen W, Tang H, Lin HJ (2017) MethyRNA: a web server for identification of N6-methyladenosine sites. J Biomol Struct Dyn 35(3):683–687
23. Kundaje A, Meuleman W, Ernst J, Bilenky M, Yen A, Heravi-Moussavi A, Kheradpour P, Zhang Z, Wang J, Ziller MJJ (2015) Integrative analysis of 111 reference human epigenomes. Nature 518(7539):317

Chapter 29

Transcript Identification Through Long-Read Sequencing

Masahide Seki, Miho Oka, Liu Xu, Ayako Suzuki, and Yutaka Suzuki

Abstract

RNA-seq using long-read sequencing, such as nanopore and SMRT (Single Molecule, Real-Time) sequencing, enabled the identification of the full-length structure of RNA molecules. Several tools for long-read RNA-seq were developed recently. In this section, we introduce an analytical pipeline of long-read RNA-seq for isoform identification and the estimation of expression levels using minimap2, TranscriptClean, and TALON. We applied this pipeline to the public direct RNA-seq data of the HAP1 and HEK293 cell lines to identify transcript isoforms which can be detected only using long-read RNA-seq data.

Key words Long-read sequencing, Direct RNA sequencing, Transcript isoform, Alternative splicing, Expression estimation

1 Introduction

RNA sequencing (RNA-seq) using short-read sequencers, such as NovaSeq6000 (Illumina), leads to fragmented cDNA reads [1]. Therefore, it is difficult to identify the entire sequence of RNA molecules and an accurate estimation of the expression levels of isoforms. The emergence of long-read sequencers, such as Sequel (Pacific Biosciences) and MinION (Oxford Nanopore Technologies), allowed for the sequencing of entire single molecules of full-length cDNA as a single read [2, 3]. These methods, called Iso-Seq (using PacBio sequencers) and cDNA-seq (using Nanopore sequencers), sequence full-length cDNA synthesized by Smart-seq or its derivatives [2, 3]. These methods were applied to not only the sequencing of RNA purified from bulk cells, but also single-cell RNA-seq [4, 5]. Direct sequencing of mRNA molecules, called direct RNA-seq, is also possible using nanopore sequencing [6]. Direct RNA-seq yields not only the entire sequence of RNAs, but also information about base modifications and secondary structure [7].

By utilizing long-read RNA-seq data, we can especially obtain information on full-length transcript isoforms caused with alternative transcription start/termination sites and alternative splicing [8]. Transcript isoforms ensure variation of protein function and their aberrations are often observed in various disease cells. Especially in cancer cells, genomic mutations induce aberrant transcript structures [9]. Mutations disrupting or creating splice sites occasionally cause aberrant RNA splicing, and structural variants including chromosomal rearrangements are known to be causes of fusion transcripts [10, 11]. It is important to understand comprehensive variation of transcript isoforms which can be associated with fundamental cellular systems and diseases.

Most of the tools used for next-generation sequencing are optimized for short reads and are not applicable to long reads, for which they exhibit a high error rate. For the alignment of long reads of RNA, splice-aware aligners, such as minimap2, deSALT, Graphmap2, and LAST, can be used [12–15]. Because of the relative high error rate of long-reads, splice junction after alignment is error prone. In nanopore sequencing, the indels caused by the sequence context, such as homopolymers, are another problem of this approach. Therefore, it is preferable to correct the splice junctions using short-read data and/or the junctions of reference transcript models. FLAIR and TranscriptClean can correct splice junctions using the junctions of short reads and/or reference transcript models [16]. For isoform detection, FLAIR, Stringtie, TALON, Mandalorion, and Pinfish are also available [5, 16–19].

In this section, we introduce an analytical pipeline of long-read RNA-seq for isoform identification and the estimation of expression levels using minimap2, TranscriptClean, and TALON. We applied this pipeline to the public direct RNA-seq data of the HAP1 and HEK293 cell lines.

2 Materials

2.1 Dataset

We used the public datasets of short-read RNA-seq and direct RNA-seq of the chronic myelogenous leukemia cell line HAP1 and the human embryonic kidney cell line HEK293 [20, 21]. These datasets are available from the Sequence Read Archive and the European Nucleotide Archive (Table 1). The libraries of short RNA-seq of HAP1 and HEK293 were prepared using a TruSeq Stranded mRNA kit (Illumina) and a NEBNext Ultra RNA Library prep Kit (New England BioLabs) and sequenced using HiSeq4000 and HiSeq2500, respectively. The library of direct RNA-seq was prepared using a Direct RNA Sequencing Kit (SQK-RNA001) and sequenced using the R9.4 flowcell of MinION.

Table 1
Datasets used in this chapter

Cell line	Method	Accession Number
HAP1	Short-read RNA-seq MinION direct RNA-seq	ERR3218280 ERR3218372 ERR3218374
HEK293	Short-read RNA-seq MinION direct RNA-seq	SRR8181090 ERR3218376 ERR3218379

2.2 Reference Genome and Transcript Model

The human hg38 reference genome was downloaded from the UCSC Genome browser (http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/analysisSet/hg38.analysisSet.chroms.tar.gz). We used the FASTA file that concatenated the FASTA files of chr1–22, X, Y, M, and EBV, with the exception of the files of chrUn_*.fa and chr*_random.fa. The comprehensive gene annotation of GENCODE release 33 was downloaded from the website of GENCODE (ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_33/gencode.v33.annotation.gtf.gz) and used as the reference transcript model [22].

2.3 Software

We installed the following tools on the virtual environment of miniconda3 v 4.5.11: python v3.7.6, minimap2 v2.17 (r941), Trimgalore v0.6.4, STAR v2.7.3, Bedtools v2.29.0, Samtools v1.9, pybedtools v0.8.0, and pyfasta v0.5.2. We downloaded TranscriptClean v2.0.2 (https://github.com/dewyman/TranscriptClean.git) and TALON v4.4.2 (https://github.com/dewyman/TALON.git) from GitHub. We installed TALON by pip command on the virtual environment. For visualization of alignment and gene annotation, we used the Integrative Genomics Viewer (IGV) [23].

3 Methods

We employed the pipeline via minimap2 for the alignment of direct RNA-seq reads, TranscriptClean for error correction, and TALON for isoform identification and read counting (Fig. 1). We also used short-read RNA-seq data to prepare the database of splice junctions using STAR.

3.1 Alignment of Direct RNA-seq Reads

Minimap2 is one of the most frequently used alignment software for long-read sequencing data [12]. The parameters optimized for direct RNA-seq and cDNA nanopore sequencing, as well as for Iso-seq using SMRT sequencing, are available in the tutorial of minimap2 (https://github.com/lh3/minimap2). The *-uf* option

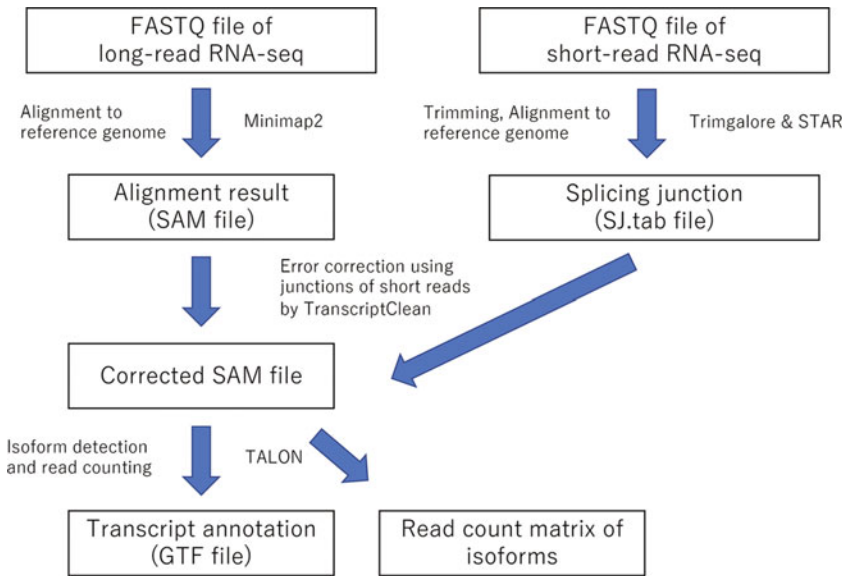

FASTQ file of long-read RNA-seq
FASTQ file of short-read RNA-seq
Alignment to reference genome
Minimap2
Trimming, Alignment to reference genome
Trimgalore & STAR
Alignment result (SAM file)
Splicing junction (SJ.tab file)
Error correction using junctions of short reads by TranscriptClean
Corrected SAM file
Isoform detection and read counting
TALON
Transcript annotation (GTF file)
Read count matrix of isoforms

Fig. 1 Workflow of the data analysis of long-read RNA-seq

of minimap2 considers only forward-strand reads, such as direct RNA-seq. The *-k14* option is optimized for direct RNA-seq reads with a higher error rate vs. DNA reads. Because long reads are error prone, misalignment around junctions occurs occasionally. Since its 2.17 version, minimap2 supports the *--junc-bed* option. This option allows the input of a splicing junction database. Using this option, the junction in the database is selected when there are multiple candidates of splice alignment (Fig. 2a). The splicing database can be generated from the GTF file of the transcript annotation using *paftools.js gff2bed*.

The SAM file is generated by the alignment of reads and contains information pertaining to reads and alignment results [24]. For visualization on the IGV, it is necessary to convert the SAM file to a sorted BAM file and generate and index file of the sorted BAM file (bam.bai file) using samtools.

1. (Optional) Convert the GTF file of gene annotation to the bed12 format using paftools.js of minimap2.

```
$paftools.js gff2bed gencode.gtf > gencode.bed
```

2. Align direct RNA-seq reads to the genome using minimap2.

```
$minimap2 -ax splice -uf -k14 --secondary=no --junc-bed
gencode.bed reference_genome.fa dRNA.fq > dRNA.sam
```

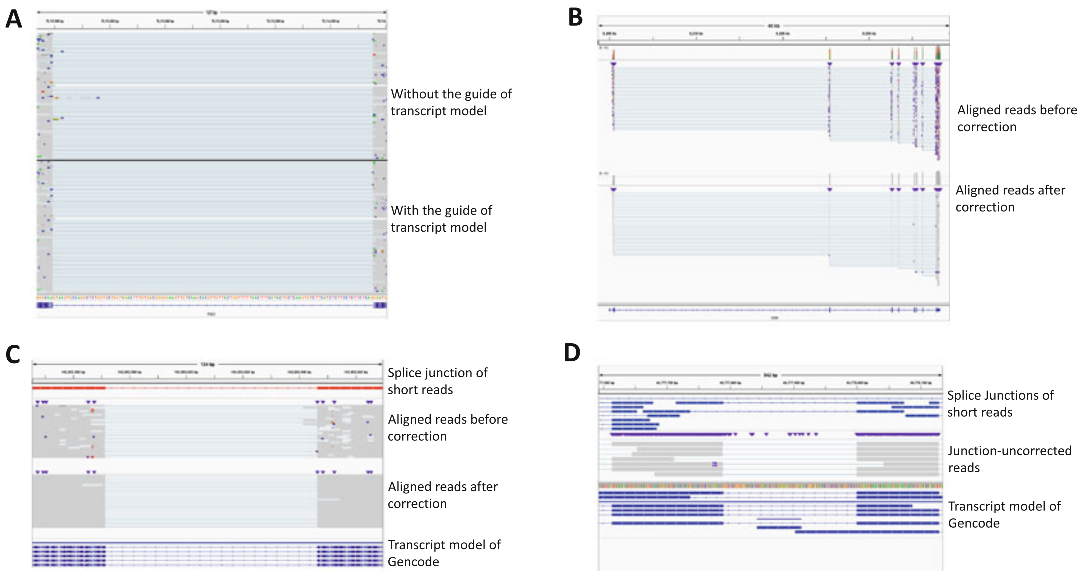

A
Without the guide of transcript model
With the guide of transcript model
B
Aligned reads before correction
Aligned reads after correction
C
Splice junction of short reads
Aligned reads before correction
Aligned reads after correction
Transcript model of Gencode
D
Splice Junctions of short reads
Junction-uncorrected reads
Transcript model of Gencode

Fig. 2 Alignment and error correction of direct RNA-seq reads. (**a**) The direct RNA-seq reads of HAP1 cells aligned using minimap2 without and with the guide of GENCODE annotation. (**b**) The aligned reads before and after error correction using TranscriptClean are shown. (**c**) The splice junction detected by short-read RNA-seq, the aligned reads before and after correction are shown. (**d**) The junction-uncorrected reads are shown

3. (Optional) For the visualization of alignment results on the IGV, convert the SAM file to the BAM format and generate an index file.

```
$samtools view -bS dRNA.sam -o dRNA.bam
$samtools sort dRNA.bam -o dRNA.bam
$samtools index dRNA.bam
```

3.2 (Optional) Alignment of Illumina RNA-seq Reads

In cases in which the cDNA insert is shorter than the length of the reads, sequences containing the cDNA and adapter sequences are read. The bases of the 3′ prime end of reads occasionally show a lower base quality. To obtain accurate alignment, it is preferable to trim the adapter sequence and the low-quality bases. We used the trimming software Trim Galore!, with comprises the adapter-trimming software Cutadapt and a FASTQ quality-check software FastQC [25–27]. Other software, such as Trimmomatic, can also be used for the same purpose [28].

STAR is an ultrafast alignment software for short-read RNA--seq [29]. First, reference index files are generated from a reference genome FASTA file and a gene annotation file using the *–runMode genomeGenerate* option of STAR. Second, the FASTQ files are aligned to the reference genome by STAR. We employed the parameters of STAR used in a previous study [17]. After alignment, SAM file, SJ.out.tab file, and log files were generated. The SJ.out.tab file contained splice junction information in a tab-delimited format. The SJ.out.tab file is usable as the database of splice junctions for TranscriptClean.

1. Perform adapter and quality trimming of Illumina reads.

```
    $trim_galore --illumina --paired read1.fastq.gz read2.
fastq.gz --basename trimed_out
```

2. Prepare the genome indexes.

```
    $STAR --runMode genomeGenerate --runThreadN 10 -
genomeFastaFiles reference_genome.fa --sjdbGTFfile gencode.
gtf --genomeDir output
```

3. Align the Illumina reads to the reference genome.

```
    $STAR --runThreadN 4 --genomeDir ../reference/star_genome
--readFilesIn trimmed_out_val_1.fq.gz trimmed_out_val_2.fq.gz
--readFilesCommand gunzip -c --outFileNamePrefix Illumina_ --
outFilterType BySJout --outFilterMultimapNmax 20 --alignSJo-
verhangMin 8 --alignSJDBoverhangMin 1 --outFilterMismatchNmax
999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20 --
alignIntronMax 1000000 --alignMatesGapMax 1000000 --outSAMat-
tributes NH HI NM MD jM jI --outSAMtype SAM
```

3.3 Correction of Errors and Splice Junctions Using TranscriptClean

As mentioned above, the sequencing and alignment errors of long reads can lead to misidentification of isoforms. TranscriptClean is a tool that is used for error correction of the SAM file of long reads (Fig. 2b, c). The FASTA file of the reference genome can be used for base correction, and the SJ.out.tab file obtained from STAR can be used for splice junction correction. Although we did not perform this analysis, it is able to perform variant-aware correction by also using a VCF file (*−-variants/−v* option), including variant information such as polymorphism and mutation. Variant-aware correction is advantageous for the detection of variants on transcripts and for haplotype phasing of reads using heterozygous SNPs. TranscriptClean outputs the corrected SAM file, corrected FASTA file, Transcript error log file (TE.log), and Transcript log file (.log). The transcript error log file contains information for each potential error regarding read name, position, error type, size (bp), with/without correction, and the reason for the absence of correction when the error was not corrected in the tab-delimited format. The transcript log file contains information per read about mapping type (primary, secondary, and unmapped) and the number of errors that were corrected or remained uncorrected among each error type. Because nanopore sequencing is error prone, it might be better to remove the reads containing uncorrected splicing junctions from the SAM file, for a more confident isoform identification (Fig. 2d).

1. Execute TranscriptClean.py to correct the SAM file of direct RNA-seq reads.

```
    $python TranscriptClean.py --sam dRNA.sam --genome
reference_genome.fa --outprefix dRNA --spliceJns Illumina_R-
NA_SJ.out.tab
```

2. (Optional) For the visualization of alignment results on the IGV, convert the SAM file to a sorted BAM format and generate an index file.

```
$samtools view -bS dRNA_clean.sam -o dRNA_clean.bam
$samtools sort dRNA_clean.bam -o dRNA_clean.bam
$samtools index dRNA_clean.bam
```

3.4 Isoform Detection and Abundance Estimation

TALON can detect transcript isoforms by comparing transcript models [17]. TALON is intended for strand-specific RNA-seq data. The use of data without strand specificity allows the recognition of the reads harboring the same splice junction as different isoforms if their strands are different. TALON classifies the reads into six categories: known, incomplete splice match (ISM), novel in catalog (NIC), novel not in catalog (NNC), antisense, and genomic transcripts. ISM is a truncation form of transcript. NIC is a novel isoform harboring only the splice junction annotated in transcript models. NNC is a novel isoform harboring more than one non-annotated splice junction. First, the database is prepared from the GTF file of GENCODE using the *talon_initialize_database* command. The config file containing the details and path to the SAM files of samples is prepared by the text editor. Using the *talon* command, the mapping results of SAM files and transcript annotation of GENCODE are compared, and the result of the comparison is added to the database file of TALON. QC.log file and talon_read_annot.tsv are also generated. The QC.log file contains information about the aligned base fraction and the identity per read. TALON uses these criteria for the quality check of reads. The default parameters of aligned base fraction (coverage) (-c/--cov option) and identity (-i/--identity) were 0.9 and 0, respectively. A coverage of 0.9 is too strict for the libraries of sequenced transcripts with an adapter sequence. In the presence of reads with an adapter sequence, a coverage threshold lower than that of the default parameter should be used, or adapter trimming of long reads should be performed. The talon_read_annot.tsv file contains the information of the transcript annotation of each read, including transcript ID and novelty. To remove artifacts, *talon_filter_transcripts* command can create the list of the novel transcripts that are commonly detected in at least two samples within replicates and the known transcript that are detected in at least one sample. The list obtained by *talon_filter_transcripts* can be used as a white list for subsequent steps by using the *--whitelist* option. The *talon_abundance* command can estimate the read counts of each isoforms. Unlike the expression value normalized to both read count and transcript length, such as RPKM (reads per kilobase of transcript per million mapped reads), used in short-read RNA-seq, the value normalized only by read count, such as RPM (Reads Per Million mapped reads), is used in long-read RNA-seq. The *Talon_create_GTF* command can create a GTF file of the identified transcripts.

1. Prepare the config file regarding the input SAM files for TALON. The config file is in csv format, and the following information is required: "dataset name," "sample description," "platform," "full path to sam file."

```
(Example of config file)
HAP1_1,HAP1,MinION_dRNA-seq,HAP1_1.sam
HAP1_2,HAP1,MinION_dRNA-seq,HAP1_2.sam
HEK293_1, HEK293,MinION_dRNA-seq, HEK293_1.sam
HEK293_2, HEK293,MinION_dRNA-seq, HEK293_2.sam
```

2. Prepare the database for TALON.

```
    $talon_initialize_database --f gencode.v33.annotation.
gtf --a gencode_v33 --g hg38 --o output
```

3. Execute TALON.

```
    $talon --f config.csv --db talon.db --build hg38 --o
output
```

4. Create the summary file of the detected transcripts.

```
$talon_summarize --db talon.db --v --o output
```

5. Prepare the paring list of experimental replicates in csv format.

```
(Example of paring file)
HAP1_1,HAP1_2
HEK293_1,HEK293_2
```

6. Prepare the transcript list for filtering.

```
$talon_filter_transcripts --db talon.db -a gencode_v33 -p
paring.csv --o filter_out.csv
```

7. Estimate the abundance of the transcript.

```
$talon_abundance --db talon.db --whitelist filter_out.csv
-a gencode_v33 --build hg38 --o output
```

8. Create GTF file of the detected isoforms.

```
$talon_create_GTF --db talon.db --whitelist filter_out.
csv -a gencode_v33 --build hg38 --o output
```

3.5 Analysis of the Direct RNA-seq Data of HAP1 and HEK293

We analyzed the direct RNA-seq data of HAP1 and HEK293 via short-read RNA-seq using this pipeline. We used the duplicate data of each cell line. We detected 21,292 known, 14,018 ISM, 6364 NIC, 524 NCC isoforms, 214 antisense, and 222 intergenic transcripts (Fig. 3a). Although 11,353 ISM transcripts were suffix type and lacked the 5′ end of the existing transcripts, only 140 of those were prefix type and lacked the 3′ end of the existing transcript. This indicates that the majority of ISM transcripts originated from artifacts of depredated RNA and disrupted sequencing, as previously suggested previously [17]. We calculated the RPM value from the read count matrix obtained from TALON, with the exception of mitochondrial genes and the suffix type of ISM transcript. We compared the expression levels between duplicates. The existence of a strong Pearson correlation between duplicates ($R = 0.99$) was shown in both HAP1 and HEK293 cells (Fig. 3b, c). We also compared the average transcript expression of the duplicates between the two cell lines and observed a strong correlation ($R = 0.93$) (Fig. 3d). We exemplified the genes that were expressed differentially between HAP1 and HEK293 cells in Fig. 3e. For the UQCRH gene, the isoform exhibited skipping of the third exon and was newly detected by TALON. This isoform showed a higher expressed in HAP1 compared with HEK293 cells.

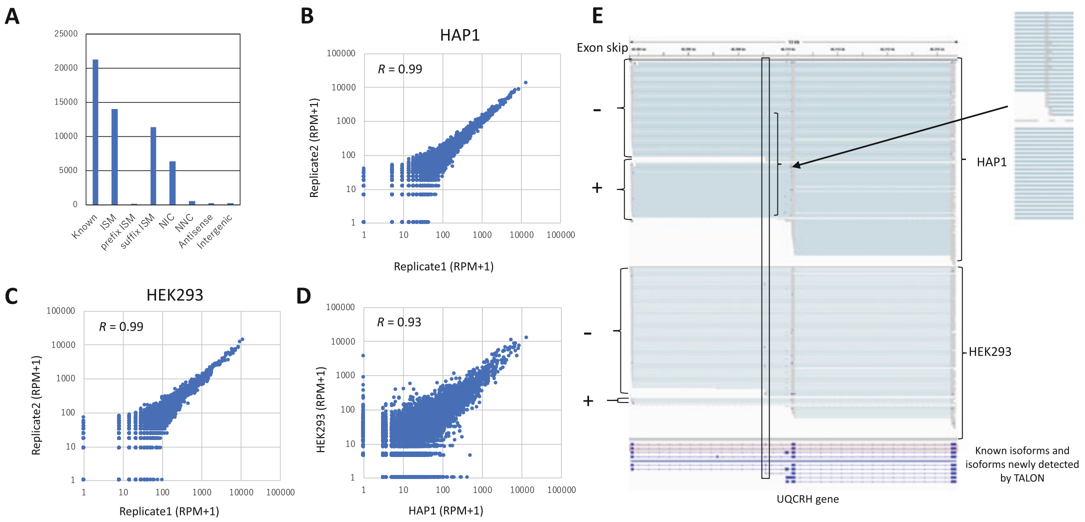


Fig. 3 Comparison of the expression levels of isoforms between HAP1 and HEK293 cells. (**a**) Number of detected isoforms. (**b**, **c**) The comparisons of the duplicates of HAP1 (**b**) and HEK293 (**c**) cells were shown, respectively. We calculated RPM as expression values. The Pearson correlation between the cell lines was shown in the graph. (**d**) We also compared the average expression levels of duplicate between HAP1 and HEK293. (**e**) An example of differential expressed gene between HAP1 and HEK293 was shown. The corrected reads of HAP1 and HEK293, and the isoforms detected by TALON were shown. The isoform of UQCRH gene with skipping of boxed third exon was newly detected by TALON. In HAP1, this boxed isoform showed higher expression than that in HEK293

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 16H06279 (PAGS), 19K16108, and 19K16792.

References

1. Stark R, Grzelak M, Hadfield J (2019) RNA sequencing: the teenage years. Nat Rev Genet 20:631–656. https://doi.org/10.1038/s41576-019-0150-2
2. Sharon D, Tilgner H, Grubert F, Snyder M (2013) A single-molecule long-read survey of the human transcriptome. Nat Biotechnol 31:1009–1014. https://doi.org/10.1038/nbt.2705
3. Oikonomopoulos S, Wang YC, Djambazian H et al (2016) Benchmarking of the Oxford Nanopore MinION sequencing for quantitative and qualitative assessment of cDNA populations. Sci Rep 6:31602. https://doi.org/10.1038/srep31602
4. Gupta I, Collier PG, Haase B et al (2018) Single-cell isoform RNA sequencing characterizes isoforms in thousands of cerebellar cells. Nat Biotechnol 36:1197–1202. https://doi.org/10.1038/nbt.4259
5. Byrne A, Beaudin AE, Olsen HE et al (2017) Nanopore long-read RNAseq reveals widespread transcriptional variation among the surface receptors of individual B cells. Nat Commun 8:16027. https://doi.org/10.1038/ncomms16027
6. Garalde DR, Snell EA, Jachimowicz D et al (2018) Highly parallel direct RN A sequencing on an array of nanopores. Nat Methods 15:201–206. https://doi.org/10.1038/nmeth.4577
7. Workman RE, Tang AD, Tang PS, et al (2018) Nanopore native RNA sequencing of a human

poly(A) transcriptome. bioRxiv 459529. https://doi.org/10.1101/459529

8. Seki M, Katsumata E, Suzuki A et al (2019) Evaluation and application of RNA-Seq by MinION. DNA Res 26:55–65. https://doi.org/10.1093/dnares/dsy038
9. Calabrese C, Davidson NR, Demircioğlu D et al (2020) Genomic basis for RNA alterations in cancer. Nature 578:129–136. https://doi.org/10.1038/s41586-020-1970-0
10. Shiraishi Y, Kataoka K, Chiba K et al (2018) A comprehensive characterization of cis-acting splicing-associated variants in human cancer. Genome Res 28:1111–1125. https://doi.org/10.1101/gr.231951.117
11. Mitelman F, Johansson B, Mertens F (2004) Fusion genes and rearranged genes as a linear function of chromosome aberrations in cancer. Nat Genet 36:331–334
12. Li H (2018) Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics 34:3094–3100. https://doi.org/10.1093/bioinformatics/bty191
13. Liu B, Liu Y, Li J et al (2019) deSALT: fast and accurate long transcriptomic read alignment with de Bruijn graph-based index. Genome Biol 20:274. https://doi.org/10.1186/s13059-019-1895-9
14. Marić J, Sović I, Križanović K, et al (2019) Graphmap2—splice-aware RNA-seq mapper for long reads. bioRxiv 720458. https://doi.org/10.1101/720458
15. Frith MC, Kawaguchi R (2015) Split-alignment of genomes finds orthologies more accurately. Genome Biol 16:106. https://doi.org/10.1186/s13059-015-0670-9
16. Tang AD, Soulette CM, van Baren MJ, et al (2018) Full-length transcript characterization of SF3B1 mutation in chronic lymphocytic leukemia reveals downregulation of retained introns. bioRxiv 410183. https://doi.org/10.1101/410183
17. Wyman D, Balderrama-Gutierrez G, Reese F, et al (2019) A technology-agnostic long-read analysis pipeline for transcriptome discovery and quantification bioRxiv 672931. https://doi.org/10.1101/672931
18. Kovaka S, Zimin AV, Pertea GM et al (2019) Transcriptome assembly from long-read RNA-seq alignments with StringTie2. Genome Biol 20:278. https://doi.org/10.1186/s13059-019-1910-1
19. Oxford Nanopore Technologies Pinfish. https://github.com/nanoporetech/pinfish. Accessed 24 Feb 2020
20. Soneson C, Yao Y, Bratus-Neuenschwander A et al (2019) A comprehensive examination of Nanopore native RNA sequencing for characterization of complex transcriptomes. Nat Commun 10:3359. https://doi.org/10.1038/s41467-019-11272-z
21. Sun Z, Xue S, Xu H et al (2019) Effects of *NSUN2* deficiency on the mRNA 5-methylcytosine modification and gene expression profile in HEK293 cells. Epigenomics 11:439–453. https://doi.org/10.2217/epi-2018-0169
22. Frankish A, Diekhans M, Ferreira A-M et al (2019) GENCODE reference annotation for the human and mouse genomes. Nucleic Acids Res 47:D766–D773. https://doi.org/10.1093/nar/gky955
23. Thorvaldsdóttir H, Robinson JT, Mesirov JP (2013) Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. Brief Bioinform 14:178–192. https://doi.org/10.1093/bib/bbs017
24. Li H, Handsaker B, Wysoker A et al (2009) The sequence alignment/map format and SAMtools. Bioinformatics 25:2078–2079. https://doi.org/10.1093/bioinformatics/btp352
25. Krueger F Trim Galore! http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/. Accessed 13 Mar 2020
26. Martin M (2011) Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet.journal 17:10. https://doi.org/10.14806/ej.17.1.200
27. Andrews Simon FastQC a quality control tool for high throughput sequence data. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/. Accessed 13 Mar 2020
28. Bolger AM, Lohse M, Usadel B (2014) Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics 30:2114–2120. https://doi.org/10.1093/bioinformatics/btu170
29. Dobin A, Davis CA, Schlesinger F et al (2013) STAR: ultrafast universal RNA-seq aligner. Bioinformatics 29:15–21. https://doi.org/10.1093/bioinformatics/bts635

Chapter 30

Transcript Isoform-Specific Estimation of Poly(A) Tail Length by Nanopore Sequencing of Native RNA

Adnan M. Niazi, Maximilian Krause, and Eivind Valen

Abstract

The poly(A) tail is a homopolymeric stretch of adenosine at the 3′-end of mature RNA transcripts and its length plays an important role in nuclear export, stability, and translational regulation of mRNA. Existing techniques for genome-wide estimation of poly(A) tail length are based on short-read sequencing. These methods are limited because they sequence a synthetic DNA copy of mRNA instead of the native transcripts. Furthermore, they can identify only a short segment of the transcript proximal to the poly (A) tail which makes it difficult to assign the measured poly(A) length uniquely to a single transcript isoform. With the introduction of native RNA sequencing by Oxford Nanopore Technologies, it is now possible to sequence full-length native RNA. A single long read contains both the transcript and the associated poly(A) tail, thereby making transcriptome-wide isoform-specific poly(A) tail length assessment feasible. We developed *tailfindr*—an R-based package for estimating poly(A) tail length from Oxford Nanopore sequencing data. In this chapter, we describe in detail the pipeline for transcript isoform-specific poly(A) tail profiling based on native RNA Nanopore sequencing—from library preparation to downstream data analysis with *tailfindr*.

Key words Poly(A) tail, Nanopore sequencing, Native RNA, *tailfindr*, R, Transcriptomics

1 Introduction

A poly(A) tail is formed by the nontemplated addition of a stretch of adenosines to the 3′-end of messenger RNA (mRNA) during RNA processing in the nucleus [1]. It mediates the transfer of processed RNA from nucleus into the cytoplasm in eukaryotes [2]. Furthermore, it is known to stabilize or destabilize the mRNA depending on its length: relatively long poly(A) tails inhibit degradation of mRNA by 3′-exonucleases and 5′-cap hydrolysis, whereas short poly(A) tails mark the mRNA for degradation by the exosome [3]. Additionally, the length of the poly(A) tail can, under certain conditions, influence the translational efficiency of the

Adnan M. Niazi and Maximilian Krause contributed equally with all other contributors.

mRNA [4–6]. Measuring isoform-specific poly(A) tail length over the whole transcriptome is therefore important in understanding its role in regulation of mRNA localization, mRNA half-life and translation regulation.

Existing methods for transcriptome-wide estimation of poly (A) tail length—which are primarily based on Illumina short-read sequencing technology [5, 7, 8]—have numerous limitations. First, RNA in its native form cannot be sequenced using Illumina sequencing: the RNA must first be reverse transcribed into cDNA, and subsequently amplified with PCR cycles to form clusters on the flow cell that are sequenced by synthesis. The conversion of RNA into cDNA results in loss of information; for example, the occurrence of native RNA modifications might be interesting to study along with the poly(A) tail length. Second, the repeated PCR cycles may introduce artefacts in the homopolymer regions that may cause errors in poly(A) tail length estimation [9–11]. Third, most of these methods estimate poly(A) tail length indirectly by inferring cDNA poly(A) or poly(T) segments using elaborate library preparation steps or custom-designed software for processing raw images of the sequencing clusters.

This renders these methods not only time-consuming but also technically challenging. Lastly, as Illumina sequencing is a short-read sequencing technology, a sequenced read from these methods contains only a small segment reflecting parts of the transcript proximal to the poly(A) tail. With such partial transcript fragments, transcript isoform-specific poly(A) tail assignment is hard, and in many instances impossible. This is because a read may align equally well to two or more transcript isoforms, making it impossible to decipher as to which transcript the read—and its associated poly (A) tail measurement—belongs to (*see* Fig. 1). Until recently it was therefore impossible to address whether different transcript isoforms have different poly(A) tail lengths.

With the advent of long read sequencing methods it recently became possible to sequence full length transcripts and their associated poly(A) tails [12–14]. In addition to offering long read sequencing only limited by the molecules integrity [15], Oxford Nanopore Technologies (ONT) novel sequencing approach also allows to sequence native RNA molecules without the conversion into cDNA [16]. This new technology has the potential to address isoform-specific poly(A) length measurements and RNA modification detection in a single assay [14, 17].

In this chapter, we will explain how ONT's sequencing approaches allow direct poly(A) measurement of native RNA (Subheading 2), describe the necessities for efficient Nanopore library preparation (Subheading 3), and how to process the data generated using *tailfindr* to perform transcriptome-wide isoform-specific poly(A) tail profiling (Subheading 4).

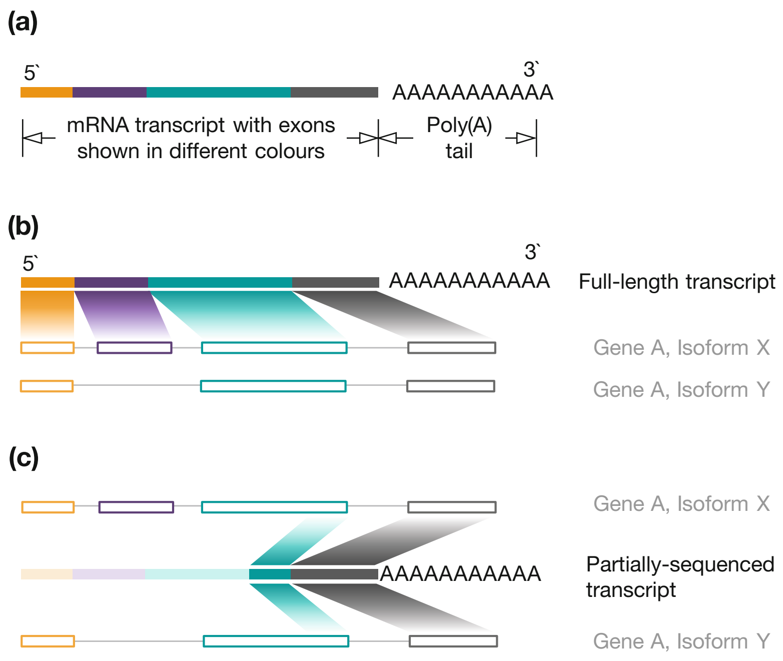


Fig. 1 (**a**) A poly(A)-tailed mRNA. (**b**) A full-length transcript uniquely and unambiguously maps to the isoform that it originated from. In the illustrated case, the read perfectly aligns to isoform X of gene A, and the measured poly (A) tail length can be uniquely attributed to isoform X. (**c**) A partially sequenced transcript can map equally well to multiple transcript isoforms, making it impossible to decipher from which of the many possible isoforms the read originated from. In this case, the partially sequenced transcript aligns equally well to both isoform X and isoform Y of gene A. Thus transcript-isoform specific poly(A) tail length assignment is not possible

2 Nanopore Sequencing

In ONT sequencing approaches, a protein nanopore is suspended in a hydrophobic material (membrane) that separates two buffer-filled wells [18]. A cross-membrane voltage of −180 mV is applied such that the *trans* side of the membrane is set at a positive potential compared to the *cis* side (*see* Fig. 2). This causes a constant ionic current to flow through the pore. The molecule to be sequenced, which can be either DNA or RNA, is located on the *cis* side of the membrane. Under the influence of the applied voltage, the negatively charged nucleotide strand threads through the pore. To ensure a homogeneous translocation rate (450 bps for DNA and 70 bps for RNA [19]), and to minimize the influence of secondary structure or DNA duplex binding energy, the DNA or RNA is fed into the pore by the ratcheting action of a motor protein. The nucleotides located in the constriction of the pore—5–6 nucleobases at any given time—modulate the current passing through the

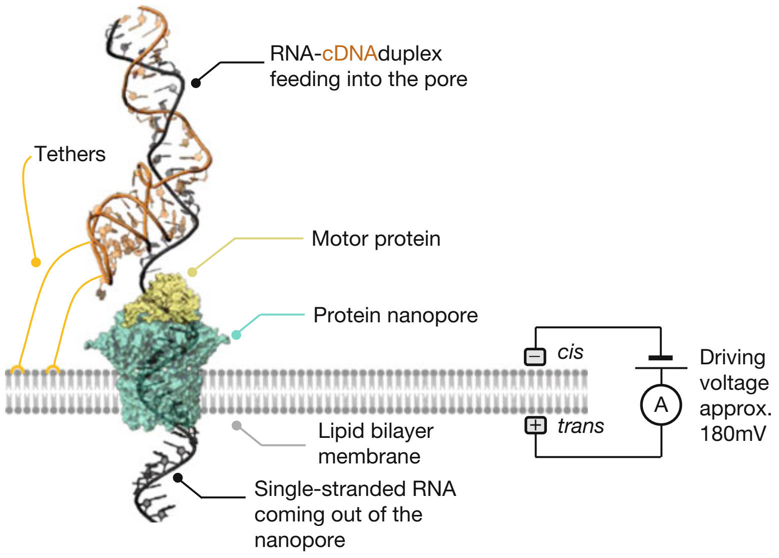


Fig. 2 RNA sequencing using ONT Direct-RNA sequencing**.** The RNA to be sequenced is first reverse-transcribed to make an RNA–cDNA duplex; this step removes RNA secondary structure that may otherwise cause pore blockage. The RNA–cDNA duplex, along with the ligated adaptor that contains the motor protein, is initially located on the *cis* side of the membrane. The tethers attached to the DNA adaptor have an affinity for the lipid membrane and help anchor the RNA–cDNA duplex to it. Under the influence of the applied voltage, the duplex shifts toward the pore, and eventually the RNA part of the duplex threads through the pore. The motor protein unwinds the RNA–cDNA duplex, and ratchets the RNA through the nanopore one base at a time. The fluctuations in the pore current as the RNA strand translocates through the pore are recorded

pore, thereby creating sequence-specific modulations in the current. This current is sampled at a rate of 3012 samples/second and saved as an array in a `.fast5` file by MinKNOW—the data acquisition and experiment management software provided by ONT. The resulting signal trace—the so-called squiggle—thus contains the information of the contiguous nucleotide strand and possible RNA modifications and should be stored as "raw data files." The raw data files are then used by a basecaller to predict the original sequence.

In the special case of ONT native RNA sequencing, the motor protein is added at the 3′-end of the molecule by poly(A)-guided ligation. Reverse transcription is optional, as the synthesized cDNA strand will not be sequenced at any time. Nevertheless, it is recommended to perform reverse transcription, as the resulting RNA–cDNA heteroduplex is devoid of secondary structure that potentially interferes with pore translocation. Furthermore, the RNA–cDNA heteroduplex is more stable than single-stranded RNA toward degradation by RNases (*see* **Note 1**). The added motor

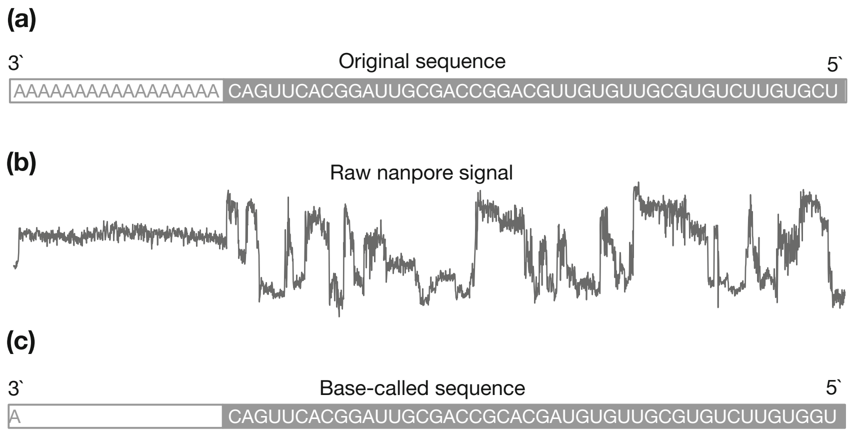

(a)
3`
Original sequence
5`
AAAAAAAAAAAAAAAAA
CAGUUCACGGAUUGCGACCGGACGUUGUGUUGCGUGUCUUGUGCU
(b)
Raw nanpore signal
(c)
3`
Base-called sequence
5`
A
CAGUUCACGGAUUGCGACCGCACGAUGUGUUGCGUGUCUUGUGGU

Fig. 3 Current basecalling algorithms underestimate poly(A) tail length. (**a**) A full-length mRNA with a 17-nt long poly(A) tail. (**b**) Raw signal generated by ONT sequencing when the sequence shown in (**a**) passes through the nanopore. Notice that the signal corresponding to the poly(A) tail is low-variance and monotonous. (**c**) Sequence predicted by the basecaller. Notice that the basecaller predicts only one adenosine in the poly(A) tail whereas the original sequence has 17 adenosines in the poly(A) region. This shows that although the raw signal for poly(A) tail is captured using Nanopore sequencing, it is not basecalled properly, preventing poly (A) tail length estimation directly from basecalling. N.B.: The sequences shown in this figure represent exemplified data

protein threads the RNA through the pore from its 3′-end to the 5′-end. The resulting current signal thus contains in this order: signal for the adaptor sequence that initially carried the motor protein, the poly(A) tail and the full-length transcript.

Although in theory it should be possible to infer the length of the poly(A) tail from the basecalled sequence alone, in practice this is not the case. When the raw signal is basecalled, the number of adenosines (reflected as A in sequence) called by the current basecallers in the poly(A) tail region is far lower than the actual number of adenosines in the poly(A) tail of the original RNA sequence (*see* Fig. 3). This is because the raw signal corresponding to a homopolymeric stretch of adenosine is a monotonous current devoid of any detectable transition from one adenosine to the next [20, 21]. The basecaller cannot decide where the signal of one adenosine ends and the next one starts; the entire poly(A) tail signal is therefore treated as a single adenosine base that got stalled in the nanopore for a long time. Thus, the poly(A) tail length currently cannot be faithfully estimated from basecalled sequences directly as it will often underestimate the actual poly(A) tail length. To accurately estimate the poly(A) tail length from Nanopore sequencing data, we developed an R package—*tailfindr* [12]. The software uses basecalled `.fast5` files and annotates the reads with poly(A) tail estimates (for more details refer to Subheading 4).

In the following sections, we will describe how to successfully perform library preparation for native RNA sequencing using Nanopore, and how to use *tailfindr* to obtain isoform-specific poly(A) tail measurements from the obtained data.

3 Library Preparation

ONT sequencing provides single-molecule long-read sequencing applications for RNA for the first time. However, the quality of the produced data and—most importantly—the quantity of data output directly depends on the quantity and quality of the provided RNA. It is therefore essential to make sure that enough RNA of good quality can be achieved prior to planning the experiment. Any RNA degradation not only affects the read length of the data obtained, but also makes library preparation inefficient, as it is based on poly(A)-dependent ligation of DNA adapters (Fig. 4). Therefore, all experimental procedures upstream of sequencing should be reviewed for forces that could degrade molecules, such as vigorous shaking or pipetting. Furthermore, RNA should be extracted as fresh as possible, or alternatively stored at -80C in RNA storage medium (TRI reagent or RNALater). Extraction should be chosen to avoid any contaminants, as these could be detrimental to the sequencing chemistry. In our experience, silica-column based purification strategies not only degrade RNA by physical force, but also retain Guanidine-hydrochloride contamination. We thus recommend the use of phenol-chloroform extraction methods, such as the use of TRI reagent. These are more time-consuming, but in our hands yield higher quality RNA with minimal contaminant carry-over. An example workflow for the use of TRI reagent for purification, as well as poly(A) enrichment based on the Poly(A)Purist MAG Kit, is described in an exemplary protocol at the end of this section.

Enriching for poly(A)-containing RNA is necessary in current ONT protocols, as the adapters are added specifically to the poly (A) tail. The addition of adapters happens through RNA ligation. However, the presence of high amounts of nonpolyadenylated RNA (such as rRNA) can significantly impact the efficiency of adapter ligation by titrating the enzyme or adapters by nonproductive binding events. Poly(A) enrichment based on magnets is the gold-standard experimental approach, but any other strategies that do not involve physical forces—such as vortexing, vigorous pipetting, or column-based purification—would work as well.

The efficiency of library preparation solely depends on the efficiency of DNA–RNA ligation procedures. A schematic workflow of Nanopore Library preparation is provided in Fig. 4. Any contaminant that reduces ligation efficiency will impact the sequencing performance of the library. It is thus important to follow the

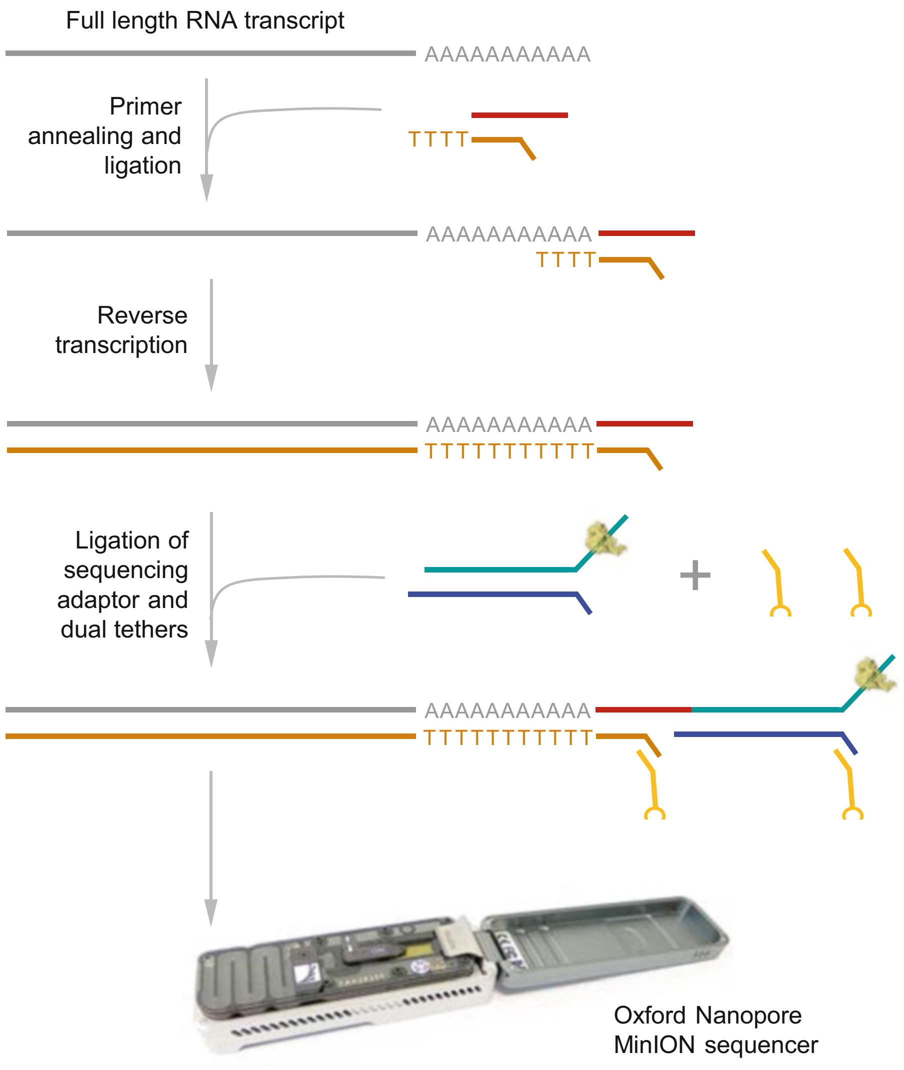

Full length RNA transcript
AAAAAAAAAAA
Primer annealing and ligation
TTTT
AAAAAAAAAAA
TTTT
Reverse transcription
AAAAAAAAAAA
TTTTTTTTTTT
Ligation of sequencing adaptor and dual tethers
+
AAAAAAAAAAA
TTTTTTTTTTT
Oxford Nanopore MinION sequencer

Fig. 4 Representation of ONT Direct-RNA library preparation protocol. The reverse transcription adaptor containing a T-overhang is ligated to the full-length RNA (shown in grey). This adaptEr can only bind at the 3′-end of the transcript and initiates reverse transcription. The reverse transcription creates a DNA strand (shown in orange). In this way, an RNA–cDNA duplex is formed. Next, a sequencing adaptor containing the motor protein is ligated to the RNA–cDNA duplex along with dual tethers. During sequencing on a ONT MinION sequencer, these tethers anchor the DNA strand to the lipid bilayer membrane, which helps to efficiently feed the RNA strand through the pore

recommendations given in the Nanopore protocols (nanoporetech.com) for RNA quality and quantity measures. The only exception are ligation and bead purification incubation times, which we routinely double. A longer incubation time at room temperature might increase the risk of RNA degradation, yet also increases the chance of successful ligation or DNA binding or elution events to beads, which leads to a more efficient library preparation. Finally, it is crucial to proceed quickly from the final ligation to actual sequencing and avoid harsh chemicals and temperatures with the final library, as an active protein has been added whose function is essential for sequencing. An example protocol for library preparation including total RNA extraction and poly(A) enrichment together with notes arising from our library preparation experience can be found at dx.doi.org/10.17504/protocols.io.9cjh2un.

4 Bioinformatics Analysis

To accurately estimate the poly(A) tail length from Nanopore native RNA sequencing data, we developed an R package—*tailfindr* [12]. Briefly, *tailfindr* estimates poly(A) tail length by first locating the monotonous stretch of current signal corresponding to the poly(A) tail within the raw signal, and then calculating its duration in samples (*see* Fig. 5). Next, a read-specific translocation rate is computed; it specifies the average of samples per nucleotide translocation. After estimating this translocation rate, it is used to normalize the tail length in samples found earlier to yield tail length in nucleotides. During all these steps, *tailfindr* only needs basecalled FAST5 files to estimate the poly(A) tail length, making it independent of downstream data processing and thus implementable in real-time data analysis pipelines. The following paragraphs will give you detailed instructions on how to use *tailfindr* toward obtaining isoform-specific poly(A) measurements from Nanopore native RNA sequencing.

4.1 Requirements

4.1.1 Test Dataset

We extracted RNA from Zebrafish (*Danio rerio*) using the protocol described above, and sequenced it on a MinION sequencer. A subset of reads from this experiment can be downloaded from tiny.cc/polya_rna_data. We will now demonstrate the various steps involved in transcript isoform-specific poly(A) tail length assessment using this example dataset, but you can use your own dataset as well.

4.1.2 Hardware Requirements

The example dataset can be processed on any laptop or desktop computer running a UNIX-based operating system with at least 3GB of free disk space. For a large real-world dataset, it is recommended that the pipeline is run on a Linux cluster, or a powerful workstation. For accelerating the basecalling speed, GPUs can be used. For more details on which GPUs are compatible with the current basecaller, please refer to this link: https://community.nanoporetech.com/posts/guppy-3-0-gpu-recommendati (requires community login).

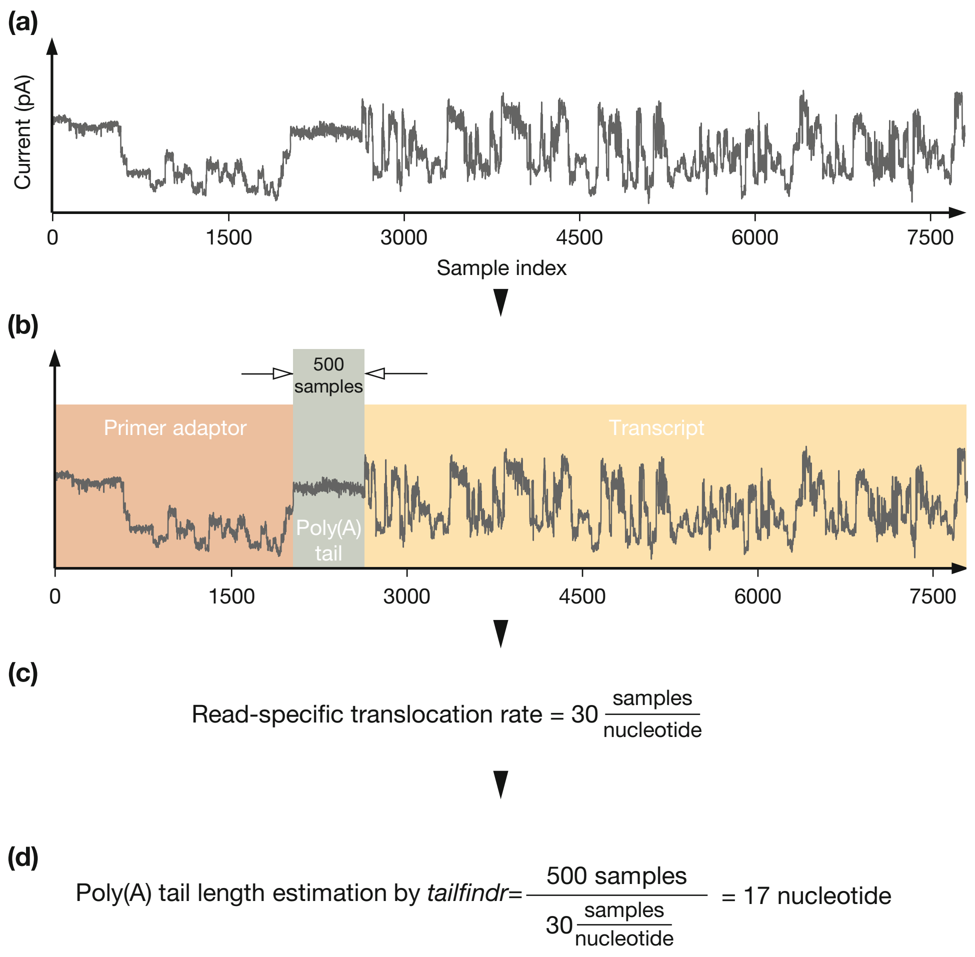

(a)
Current (pA)
0
1500
3000
4500
6000
7500
Sample index
(b)
500 samples
Primer adaptor
Transcript
Poly(A) tail
0
1500
3000
4500
6000
7500
(c)
Read-specific translocation rate = 30 samples/nucleotide
(d)
Poly(A) tail length estimation by *tailfindr* = 500 samples / 30 samples/nucleotide = 17 nucleotide

Fig. 5 A simplified view of how *tailfindr* estimates poly(A) tail length. (**a**) Complete raw signal corresponding to an RNA transcript translocating through the pore. The signal consists of a series of current samples measured in picoAmperes (pA). (**b**) *tailfindr* first locates the monotonous signal corresponding to the poly(A) tail (highlighted in brown). In this example, the segment is 500 samples long. (**c**) Next, *tailfindr* estimates the read-specific translocation rate, that is, the average number of samples generated per nucleotide in a given read. (**d**) Poly(A) length is calculated by dividing the tail length in samples by the read-specific translocation rate

4.1.3 Software Requirements

The following software should be installed on the analysis computer:

- Python 3 environment.
- R (version 3.5.3 or greater).
- Git.

4.2 Data Analysis Pipeline

There are various steps involved in going from raw reads produced by ONT sequencing to transcript isoform-specific poly(A) tail length assignment, as shown in Fig. 6. We will now describe each of these steps in detail.

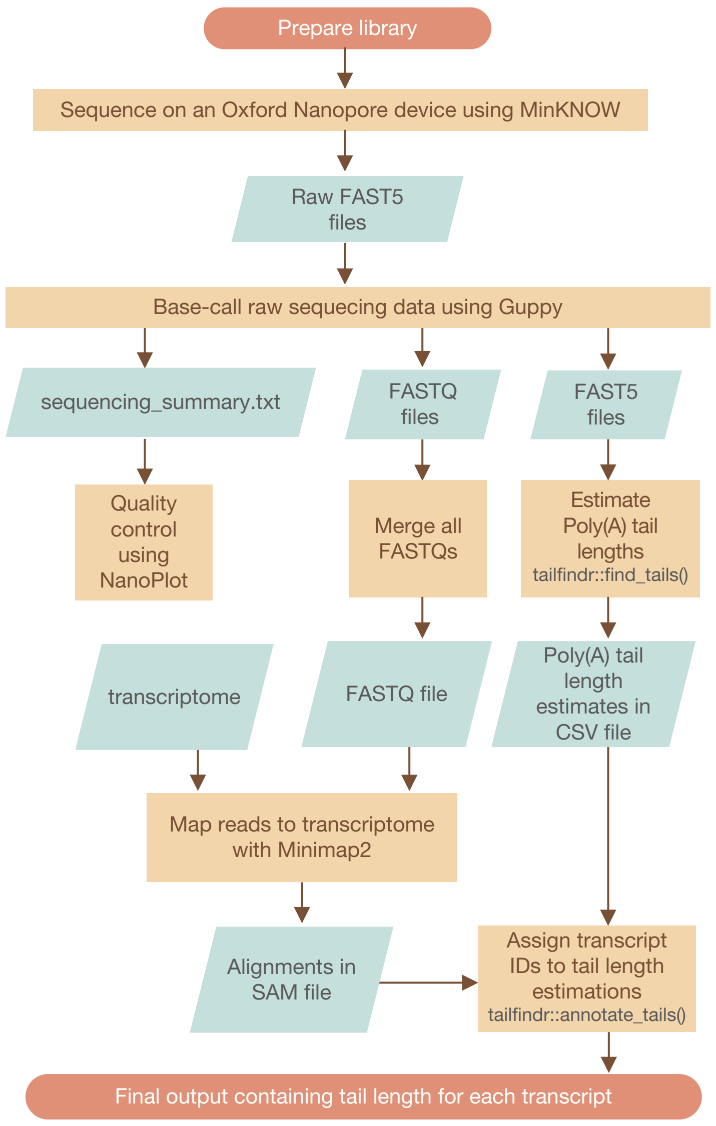

Prepare library
Sequence on an Oxford Nanopore device using MinKNOW
Raw FAST5 files
Base-call raw sequecing data using Guppy
sequencing_summary.txt
FASTQ files
FAST5 files
Quality control using NanoPlot
Merge all FASTQs
Estimate Poly(A) tail lengths tailfindr::find_tails()
transcriptome
FASTQ file
Poly(A) tail length estimates in CSV file
Map reads to transcriptome with Minimap2
Alignments in SAM file
Assign transcript IDs to tail length estimations tailfindr::annotate_tails()
Final output containing tail length for each transcript

Fig. 6 Flowchart for poly(A) tail length estimation using Nanopore sequencing and *tailfindr*

4.2.1 Basecalling

Nanopore sequencing produces raw FAST5 files that record the current signal through the pore as an RNA molecule translocates through it (*see* Fig. 7a). The first step is to basecall this raw signal to find the nucleotide sequence corresponding to the recorded current. There are many basecallers that can do this; please refer to [22] for a review on this topic. Some of the basecallers have been developed by ONT, while others are developed by Nanopore users. Albacore is a widely used basecaller developed by ONT. Guppy—a recently released basecaller, also developed by ONT—has now replaced Albacore because it has better basecalling performance

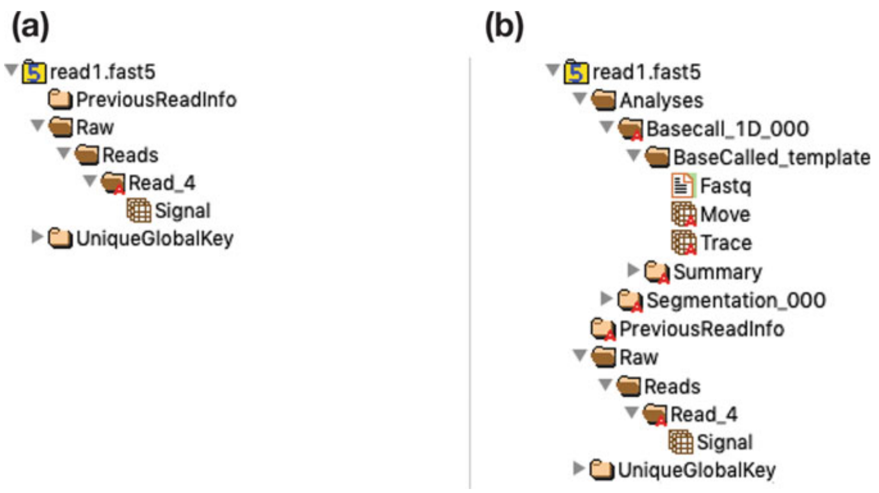


Fig. 7 Structure of a FAST5 file as displayed by HDFView software. (**a**) File structure of a raw FAST5 file generated by MinKNOW. (**b**) The same file as in (**a**) after basecalling by Guppy. During basecalling a new FAST5 is generated that contains not only raw signal data, but also additional basecalling information. Notice how additional levels of information (`Analyses`, `Basecall_1D_000` etc.) have now been added in this new file

and is faster than its predecessor. We recommend using the latest version of Guppy which can be downloaded from the ONT Community website https://community.nanoporetech.com/downloads.

Basecalling a raw FAST5 file using Guppy will add a `Basecall_1D_000` group to the FAST5 file hierarchy (*see* Fig. 7b). This basecall group contains a `Move` table (`Events` table in case of Albacore) which is used by *tailfindr* to compute the read-specific translocation rate. The structure of a FAST5 file—raw or basecalled—can be easily explored by opening it in HDFView (https://www.hdfgroup.org/downloads/hdfview/).

Guppy has both CPU and GPU versions. If you have access to an Nvidia GPU, then install and use the GPU version of Guppy, as it is faster to basecall on GPUs compared to CPUs. Here, we will demonstrate basecalling using the CPU version of Guppy (*see* **Note 2** on where to get the latest version of Guppy). Assuming that you have a Quad Core processor (with two threads per processor; eight threads in total) and 16GB of RAM, basecalling can be done by executing the following on the command line:

```
guppy_basecaller \
--config rna_r9.4.1_70bps_hac.cfg \
--input_path \path\to\raw\reads\folder \
--recursive \
--save_path \path\to\save\basecalled\data\to \
--fast5_out \
--trim_strategy none \
--num_callers 1 \
--cpu_threads_per_caller 8 \
2>&1 | tee logfile.txt
```

Parameter Description

`--config` specify the model configuration to be used during basecalling. In this case, we have chosen the "high accuracy" (hac) RNA model for pore version 9.4.1. The hac models yield more accurate basecalls at the cost of basecalling speed.

Refer to:

- **Note 3** for choosing a faster basecalling model.
- **Note 4** for selecting an appropriate config file for your experiment in case you are not sure.
- **Note 5** if your data is from a legacy RNA kit.

`--input_path` specify the path of the folder containing raw FAST5 files produced by the ONT sequencing platform. When using the example dataset, extract it first, and then specify the path of the extracted directory here.

`--recursive` specifies that the `input_path` directory should be recursively searched to discover all raw FAST5 files within any subfolders.

`--save_path` specify the path of the directory where basecalled files should be stored.

`--fast5_out` specifies that in addition to the FASTQ files, the basecaller should also output FAST5 files. Basecalled files containing FAST5 output is essential for *tailfindr* to calculate the read-specific translocation rate for normalizing the poly(A) tail length.

`--trim_strategy` should be set to `none` so that the basecaller does not trim off the adaptor sequence that was added to the 3′ end of the poly(A) + RNA.

`--num_callers` specifies how many basecallers to use in parallel.

`--cpu_threads_per_caller` specifies how many threads should be used per basecaller. In general, `num_callers` * `cpu_threads_per_caller` should not exceed the total number of threads available on the machine. Furthermore, there must be at least `4GB` + `1GB` * `num_callers` RAM available. In our case, both these criteria are satisfied for the machine that we are using. For more exhaustive information on these settings, please refer to the document "Guppy basecaller and Guppy basecaller server" (https://community.nanoporetech.com/protocols/Guppy-protocol/v/gpb_2003_v1_revm_14dec2018/guppy-basecaller-and-guppy-basecaller-server) on the Nanopore Community.

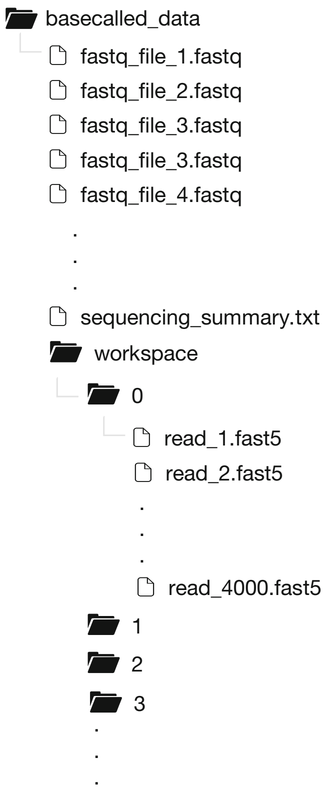


Fig. 8 Structure of the output directory produced by the Guppy basecaller. Each FASTQ file in the output of the basecaller contains sequence and quality scores for 4000 (default) reads. The `sequencing_summary.txt` file contains a summary of useful basecalling information, which is used by tools such as NanoPlot. The `workspace` folder contains numbered subfolders, each of which contain 4000 basecalled FAST5 files, which are used by tools such as *tailfindr*

`2>&1 | tee logfile.txt` specifies that the output, and any errors produced by the command, should be saved in a text file in addition to being displayed in the terminal. It is a good practice to do this for troubleshooting in case of a computer crash, power failure etc.

After successfully running the above, the basecalled FASTQ and FAST5 file can be found in the directory as specified in `save_-path`. The structure of this directory is depicted in Fig. 8. Please refer to **Note 6** to find how the structure of this directory changes when multi-fast5 files are basecalled.

4.2.2 Quality Control After Basecalling

Running quality control checks after basecalling is an optional but recommended step as it can reveal important information about the sequencing run such as the length of the reads (Fig. 9a), sequencing performance over time (Fig. 9b), and the quality of the reads

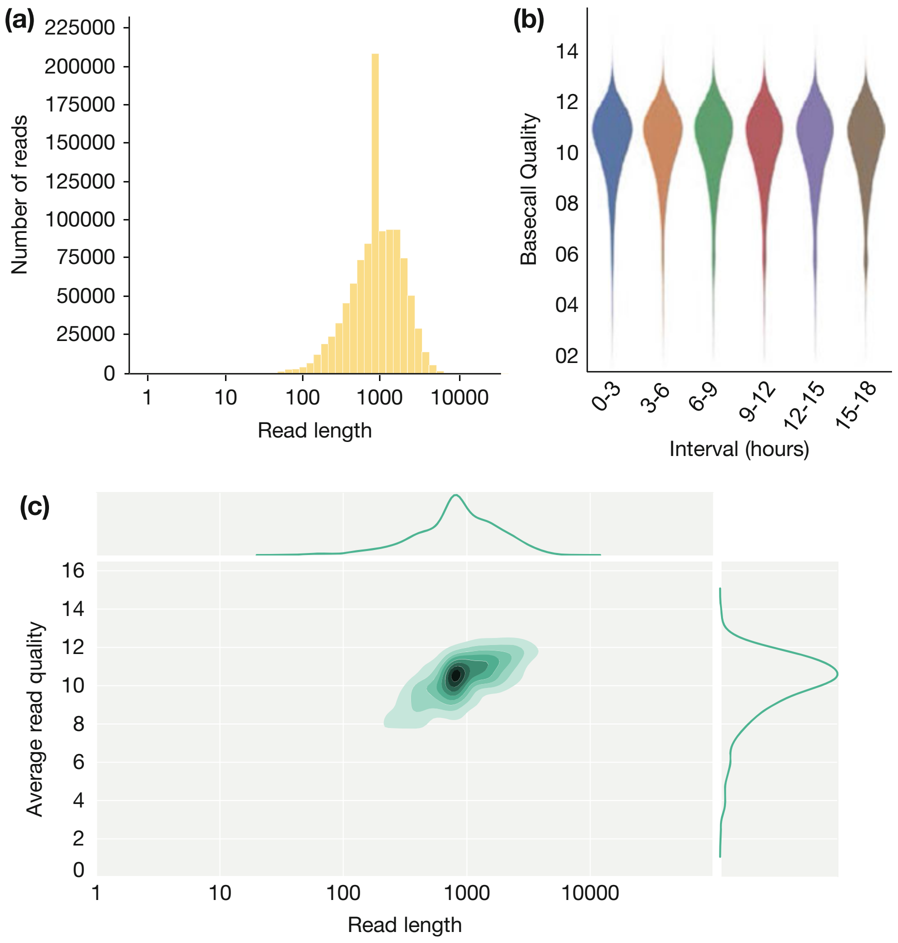


Fig. 9 A subset of figures generated by NanoPlot. (**a**) Read length histogram. This plot can be useful in understanding if RNA degradation significantly affected the sample. This particular histogram was generated for a sequencing run in which Zebrafish RNA was spiked with a synthetic GFP RNA construct of approx. 800 bp in length. The spike in the histogram around 800 represents these GFP reads, and the background represents the read length distribution for the Zebrafish transcriptome. (**b**) Basecall quality vs. time of sequencing. This plot is useful in assessing if the sequencing chemistry—which might degrade over time—is having an adverse effect on the quality of the reads. Ideally, the basecalling quality should not drop dramatically during the sequencing run. (**c**) Read length vs. average read quality plot. It is useful in understanding how the read quality varies over read length. In a good sequencing run, the read quality for the majority of the reads should be around 8–14 for RNA (9–20 for DNA). Higher reads quality are good, and lower read qualities for majority of reads might warrant revisiting the library preparation steps and figuring out what might have gone wrong

(Fig. 9c). There are many tools to perform QC on Nanopore data, but the ones that produce the most informative plots are NanoPlot (https://github.com/wdecoster/NanoPlot) and PycoQC (https://a-slide.github.io/pycoQC/) [23, 24].

Here, we will use NanoPlot to perform quality control checks on the basecalled data. NanoPlot requires only the `sequencing_summary.txt` file produced by Guppy. To run NanoPlot, first activate a Python 3 environment, and then run the following in the command line:

```
NanoPlot \
--summary \path\to\sequencing_summary.txt \
--outdir \output\path \
--loglength
```

Parameter Description

`--summary`, path of the summary file generated by Guppy.

`--outdir`, path of the directory where NanoPlot output should be saved.

`--loglength`, specifies that the read lengths should scaled logarithmically in the plots.

The output of NanoPlot is an HTML file that can be viewed in any browser of your choice.

4.2.3 *Installing and Running* Tailfindr

We are now ready to estimate poly(A) tail lengths in the basecalled data using *tailfindr*. Please refer to its documentation (https://github.com/adnaniazi/tailfindr) to learn how to install it. After installing *tailfindr*, poly(A) tail lengths can be estimated by using the following commands in R:

```
library(tailfindr)
df <- find_tails(fast5_dir = '/path/to/basecalled_data',
 save_dir = '/path/to/save/folder/',
 csv_filename = 'rna_tails.csv',
 num_cores = 2)
```

tailfindr discovers all FAST5 files recursively within the `fast5_dir`. The resulting CSV file—as specified in the `csv_filename` parameter—is saved in the `save_dir`. `num_cores` specifies the number of physical cores on the machine to be used when running *tailfindr*.

Please refer to:

- **Note 7** if you are running *tailfindr* on MinKNOW Live-basecalled data.
- **Note 8** if you want to generate plots highlighting the poly (A) tail region in the raw current data.
- **Note 9** on how to use *tailfindr* for estimating poly(A)/(T) length in cDNA data.

The output of *tailfindr* is CSV file contain six columns as described in Table 1.

Table 1
Description of columns in the CSV output of *tailfindr*

Column name	Column type	Description
read_id	Character	Read ID as given in the FAST5 file
tail_start	Numeric	Sample index of start site of the tail in raw data
tail_end	Numeric	Sample index of end site of the tail in raw data
samples_per_nt	Numeric	Read-specific translocation rate in terms of samples per nucleotide
tail_length	Numeric	Tail length in nucleotides. It is the difference between tail_end and tail_start divided by samples_per_nt
file_path	Character	Absolute path of the FAST5 file

Now that we have the poly(A) tail length for each read in the CSV file, it is possible to perform quality control checks of this data. For example, a distribution of poly(A) tail lengths can be plotted to see if it aligns with the expected distribution of poly(A) tail lengths. Furthermore, a distribution of the translocation rate `sample-s_per_nt` can also be plotted. Ideally, this distribution should be unimodal with no skew (*see* **Note 9**).

4.2.4 Concatenate FASTQ Files

During the basecalling step, Guppy produced both FASTQ and FAST5 files. By default, each FASTQ file contains sequences of 4000 reads (*see* Fig. 8). Downstream processing software, such as the mapper *Minimap2* [25], require only a single FASTQ file as input. Therefore, all FASTQ files produced by Guppy should be concatenated. Execute the following script in command line to combine all FASTQ file into one:

```
BASECALLED_DATA_PATH=/directory/containing/basecalled/data
OUTPUT_PATH=/directory/where/concatenated/fastq/is/to/stored
# Do not edit the code below this line
cd $BASECALLED_DATA_PATH
find ${ BASECALLED_DATA_PATH} -name '*.fastq' | cat > ${OUT-
PUT_PATH}/filenames.txt
{ xargs cat < ${OUTPUT_PATH}/filenames.txt ; } > ${OUTPUT_-
PATH}/all_reads.fq
```

The above shell script searches `BASECALLED`_DATA_PATH directory for all files with `.fastq` extension and produces the following two files in the `OUTPUT_PATH` directory:

1. `filenames.txt` file that contains the names of all FASTQ files that were found in `BASECALLED`_DATA_PATH directory, and will be concatenated.

2. `all_reads.fq` file that contains the concatenated FASTQ sequences from all the FASTQ files recorded in the `file-names.txt` file.

4.2.5 Alignment of Data to Transcriptome

Although we have estimated poly(A) tail lengths for all reads, we still do not know which transcript each of these reads originated from. To find the transcript identities, the reads must be mapped to the transcriptome of the organism from which the RNA was extracted (please refer to **Note 10** if no reliable transcriptome is present and data should be mapped to a reference genome). The alignment information can then be merged with *tailfindr* output to associate the poly(A) tail length estimations to their respective transcript IDs.

To map the data to the transcriptome, we will use Minimap2 (https://github.com/lh3/minimap2) [25]. Minimap2 needs a single FASTQ file containing all the reads to be aligned. Run the following command in command line to invoke Minimap2:

```
minimap2 \
 -ax map-ont \
 /path/to/reference.fa \
 /path/to/all_reads.fq > /path/to/alignments.sam
 2>&1 | tee logfile.txt
```

Parameter Description

Here is a description of the parameters used in the above command:

`-a` specifies that CIGAR string and output alignments should be produced in the SAM format.

`-x` use predefined settings for mapping. As each of these sequencing technologies differ in their insertion, deletion and error rates, there are a number of presets available in Minimap2 to choose from; `map-ont` is one of them. It specifies that Minimap2 should use alignment parameters fine-tuned for ONT sequencing data. This is because Minimap2 can align reads from Illumina, PacBio, and ONT sequencing.

4.2.6 Annotating Tailfindr Output with Transcript IDs

Now that we have the poly(A) tail length estimates from *tailfindr* in a CSV file, and the alignment information in a SAM file, we are ready to merge them together. This will annotate each read with its corresponding transcript ID. To do this, invoke *tailfindr's* built-in convenience function `annotate_tail()` in R:

```
df_annotate <-
annotate_tails(
sam_file = "/path/to/sam/file.sam",
 tails_csv_file = "/path/to/tails.csv",
 output_file = "/path/to/annotated_tails.csv"
)
```

Table 2
Description of columns added to the *tailfindr* CSV output by merging SAM information

Column name	Column type	Description
transcript_id	Character	Transcript ID from the transcriptome
mapping_quality	Numeric	Mapping quality of the transcript
sam_flag	Numeric	SAM flag

This command will add three more columns to the input CSV file as described in Table 2.

We now have the tail length and the corresponding transcript IDs in the `annotated_tails.csv` file. Thus we have successfully annotated each read with a transcript-isoform ID and a corresponding poly(A) tail length.

4.2.7 What Next?

Now that we have transcript-specific poly(A) tail lengths, we can do a number of things. For example, we can plot the distribution of poly(A) tail length of our dataset. We can also annotate the poly (A) tail length of a transcript with additional features such as gene name, gene length and its function. These steps are beyond the scope of this chapter, however, the reader should note that they can be easily done within R using the *biomaRt* Bioconductor package [26]. With gene name annotations, we can for instance generate a scatter plot of poly(A) tail length vs. gene length to see if there is any interesting relationship between the two. Additionally it is possible to plot poly(A) tail distributions from transcript isoforms of the same genes. Many further possibilities for data analysis exist, and implementation depends on the particular research question. The here described *tailfindr*-based pipeline provides the first step towards exploring these possibilities enabling the study of isoform-specific poly(A) tail-dependent regulation.

5 Conclusion

We have here demonstrated how long-read ONT native RNA sequencing in combination with *tailfindr* can be used for transcriptome-wide isoform-specific poly(A) tail profiling. This method simplifies isoform-specific poly(A) tail measurements and avoids common caveats from short-read based sequencing approaches, namely (1) the possible introduction of amplification artefacts, (2) transcript isoform quantification based on statistical analysis of short reads spanning exon borders, and (3) elaborate and time-consuming sequencing sample preparation.

The portability and low investments for ONT sequencers, coupled with its ability to basecall and analyze sequencing data in real-time, enables anyone to sequence anything, anywhere. Additionally, the ability of direct RNA sequencing to detect any epigenetic modification in native RNA alleviates the need for separate assays for detecting each RNA modification. This enables future studies—both in the field and in a laboratory settings—to assay poly(A) tail length and RNA modifications in a single experiment. Such a transcriptome-wide holistic approach would provide a valuable insight in understanding RNA biology—one long molecule at a time.

6 Notes

1. Reverse-transcribing RNA into an RNA–cDNA duplex is an optional but recommended step. Without performing this step, the throughput will be about 30% lower and basecalling quality scores will also be slightly lower. Most likely this is caused by secondary RNA structure affecting pore translocation, making current signal more variable. Additionally, RNA degradation causes the average read length to be shorter. We recommend that you perform this step unless you have a very good reason not to.
2. We demonstrated how to basecall reads using the latest basecaller at the time of this writing provided by ONT—Guppy v3.2.4. However, it should be noted that the basecalling technology is constantly evolving. Always check ONT's Software Download section (https://community.nanoporetech.com/downloads) to read about the latest version of the basecaller and how to use it, as these might significantly increase basecalling accuracy and thus transcript isoform assignment.
3. If basecalling speed is more important than basecalling accuracy, then use the fast model configuration file `rna_r9.4.1_70bps_fast.cfg`.

   At the time of this writing, the fast models are approximately 5–8 times faster than the high accuracy model. Table 3 shows a comparison between raw read accuracy of fast and high accuracy models.
4. If your experiment uses a pore version other than 9.4.1, then ensure that you specify a configuration file that matches the version of the pore used. You can find a list of all available configuration files for every flow cell and sequencing kit by executing the following command:

Table 3
A comparison of raw read accuracies between fast and high-accuracy basecalling models

Sample type	Model name	Raw read accuracy
DNA	Fast basecalling High-accuracy basecalling	92.1% 95.0%
RNA	Fast Basecalling High-accuracy basecalling	88.6% 93.9%

```
guppy_basecaller --print_workflows
```

If you are still unsure as to which configuration file to use, then, instead of specifying the configuration file, you can also let Guppy choose the appropriate configuration file for you. In this case, however, you have to specify the `flowcell` and `kit` arguments. Assuming if the flow cell and kit used in the experiment are FLO-MIN106 and SQK-RNA001, respectively, then use the following command in command line to invoke Guppy:

```
guppy_basecaller \
 --flowcell FLO-MIN106 \
 --kit SQK-RNA001 \
 --input_path \path\to\raw\reads\folder \
 --recursive \
 --save_path \path\to\save\basecalled\data\to \
 --fast5_out \
 --trim_strategy none \
 --num_callers 1 \
 --cpu_threads_per_caller 8 \
 2>&1 | tee logfile.txt
```

5. The use of *tailfindr* is compatible with any RNA kit—including legacy kits—as all of these kits sequence both the transcript and the poly(A) tail. Thus, you can use *tailfindr* to find poly(A) tail lengths on any older RNA dataset where the initial aim of the study was something entirely different. For *tailfindr* to work, the only requirement is the availability of FAST5 files—either raw or basecalled; *tailfindr* cannot be used if the only file remaining from past experiments are FASTQ files. We recommend that you always re-basecall the old previously basecalled FAST5 files before using *tailfindr* on it, and specify an appropriate value for `basecall_group` parameter when invoking *tailfindr*.

6. If the raw FAST5 files produced by MinKNOW have only one read per FAST5 file, then reads within the workspace folder are arranged in numbered subfolders such that each folder contains 4000 FAST5 reads, as depicted in Fig. 8. However, if the raw FAST5 files, produced by the sequencer, have multiple reads per FAST5 file, then there are no subfolders within workspace folder, and each basecalled FAST5 file in `workspace` folder will contain multiple reads (default is 4000) inside them.
7. MinKNOW—the data acquisition software used during sequencing on ONT sequencers—can basecall while the raw data is being acquired. This feature is called "MinKNOW Live Basecalling." Currently, *tailfindr* does not support MinKNOW live basecalled data because these FAST5 files do not contain Event/Move table (*see* Fig. 10a). The Event/Move table is required by *tailfindr* to compute a read-specific translocation rate in order to normalize the poly(A) tail length in samples to yield poly(A) tail length in nucleotides.

   To circumvent this problem, please basecall MinKNOW live-basecalled data again using standalone Guppy or Albacore. This will add an additional Basecall group (`Basecall_1D_001`) in the file structure of the FAST5 file (*see* Fig. 10b). When using *tailfindr* on these re-basecalled reads, you must correctly specify the Basecall group containing the Event/Move table. For example, the read shown in Fig. 10, the Event/Move table in the re-basecalled file is in the `Basecall_1D_001` in the FAST5 file structure hierarchy. *Tailfindr*, in the case, should be invoked in R as shown below:

```
df <- find_tails(fast5_dir = '/path/to/basecalled_data',
 save_dir = '/path/to/save/folder/',
 basecall_group = 'Basecall_1D_001',
 csv_filename = 'rna_tails.csv',
 num_cores = 2)
```

   The default value of basecall_group is `Basecall_1D_000`, which in the command above, has been changed to `Basecall_1D_001`.
8. *tailfindr* allows you to generate plots that show the tail location in the raw squiggle (*see* Fig. 11). You can save these plots as interactive .html files by using `'rbokeh'` as the `plotting_library`. You can then interactively zoom in on the tail region in the raw squiggle and see the exact location of the tail. To generate these plots, execute the following command in R:

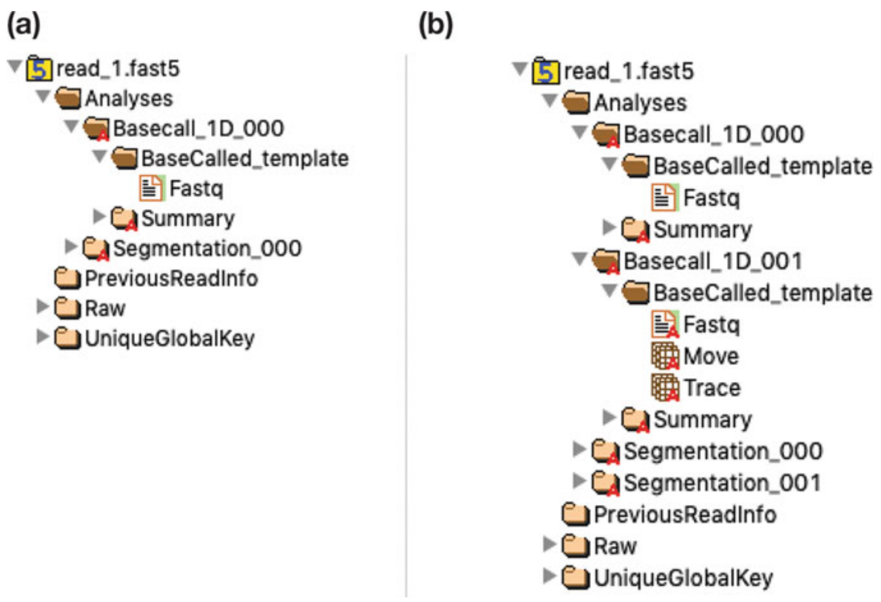


Fig. 10 Hierarchy of contents within basecalled FAST5 files as viewed through the HDFView software. (**a**) Contents of a MinKNOW live Basecalled read. Notice that under the Basecall_1D_000 group, there is no Move table, which is required by *tailfindr* to find the read-specific translocation rate (**b**) Contents of the read shown in (**a**) after it has been basecalled again using standalone Guppy. Notice the addition of Basecall_1D_001 group in the FAST5 file hierarchy, which now contains Move table. *Tailfindr* should now be invoked with basecall_group parameter set to ‘Basecall_1D_001’ to ensure that it can find the Move table

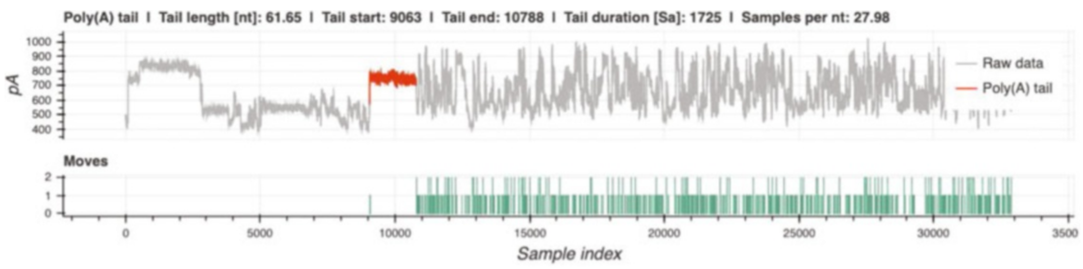


Fig. 11 A plot generated by *tailfindr*. The poly(A) tail is highlighted in red in the current trace. Each spike in the bottom panel shows the locations in the current trace where the basecaller has detected a nucleotide transition. Notice how the poly(A) tail region is devoid of any base transition. This is because the basecaller cannot distinguish when one adenosine base in the poly(A) tail ended and the next one started. It can detect a nucleotide transition only if a more diverse sequence is encountered

```
df <- find_tails(fast5_dir = '/path/to/basecalled_data',
 save_dir = '/path/to/save/folder/',
 csv_filename = 'rna_tails.csv',
 save_plots = TRUE,
 plotting_library = 'rbokeh',
 num_cores = 2)
```

Generating plots can slow down the performance of *tailfindr*. We recommend that you generate these plots only for a small subset of your reads.

9. Although we have demonstrated how to perform poly(A) tail profiling using Nanopore sequencing of native RNA, it is also possible to perform poly(A)/(T) profiling using complementary DNA (cDNA) sequencing data produced by Nanopore sequencing. Sequencing cDNA instead of RNA has many advantages:

   (a) cDNA is more stable compared to RNA which can degrade quickly if not handled very carefully at every step of library preparation protocol,

   (b) cDNA sequencing requires less starting material compared to RNA sequencing,

   (c) cDNA sequencing on Nanopore devices produces ten times more data per flowcell compared to RNA sequencing because of the faster motor protein, and,

   (d) poly(A) tail length estimates in DNA are more robust compared to RNA because the motor protein used in DNA sequencing ratchets the DNA at a more controlled speed compared to the motor protein used in RNA sequencing (*see* Fig. 12).

   For more information on poly(A)/(T) profiling in cDNA, please refer to the *tailfindr* paper [12] and documentation on GitHub.

10. Reads from RNA sequencing can be mapped either to a reference transcriptome, or to a genome. The latter is more cumbersome but could yield the identification of new transcript

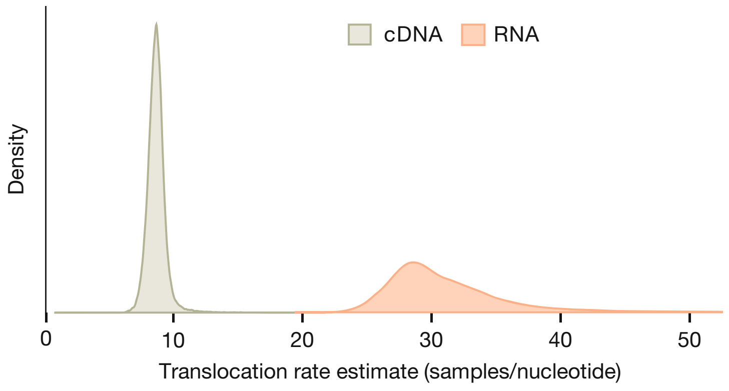


Fig. 12 Comparison of cDNA and RNA translocation rate estimates. RNA translocates at a slower speed compared to DNA. Furthermore, the spread in RNA translocation rate is greater than that of cDNA. This in turn translates to more spread in RNA poly(A) tail lengths compared to cDNA poly(A)/(T) tail lengths

isoforms. This is especially useful if the reference transcriptome is known to be erroneous and is being assessed for the first time by long-read sequencing. For aligning reads to a genome with Minimap2, use the following command:

```
minimap2 \
 -ax splice -uf -k14 \
 /path/to/reference_genome.fa \
 /path/to/all_reads.fq > /path/to/alignments.sam
 2>&1 | tee logfile.txt
```

Parameter Description

Here is a description of additional parameters in the above command:

`-splice` Specifies that spliced alignment should be done.

`-uf` By default, spliced alignment assumes the read orientation relative to the transcript strand is unknown and therefore it tries two rounds of alignment to infer the read orientation. This flag forces Minimap2 to consider only the forward transcript strand during mapping.

`-k14` For noisy Nanopore Direct RNA-seq reads, it is recommended to use a smaller k-mer size for increased sensitivity to the first or the last exons. Default value of k-mer size is 15.

Acknowledgments

Adnan M. Niazi and Maximilian Krause contributed equally to this work.

References

1. Bardwell VJ, Zarkower D, Edmonds M, Wickens M (1990) The enzyme that adds poly(A) to mRNAs is a classical poly(A) polymerase. Mol Cell Biol 10:846–849. https://doi.org/10.1128/mcb.10.2.846
2. Huang Y, Carmichael GG (1996) Role of polyadenylation in nucleocytoplasmic transport of mRNA. Mol Cell Biol 16:1534–1542. https://doi.org/10.1128/mcb.16.4.1534
3. Meyer S, Temme C, Wahle E (2004) Messenger RNA turnover in eukaryotes: pathways and enzymes. Crit Rev Biochem Mol Biol 39:197–216. https://doi.org/10.1080/10409230490513991
4. Beilharz TH, Preiss T (2007) Widespread use of poly(A) tail length control to accentuate expression of the yeast transcriptome. RNA 13:982–997. https://doi.org/10.1261/rna.569407
5. Subtelny AO, Eichhorn SW, Chen GR et al (2014) Poly(A)-tail profiling reveals an embryonic switch in translational control. Nature 508:66–71. https://doi.org/10.1038/nature13007
6. Lima SA, Chipman LB, Nicholson AL et al (2017) Short poly(A) tails are a conserved feature of highly expressed genes. Nat Struct Mol Biol 24:1057–1063. https://doi.org/10.1038/nsmb.3499
7. Chang H, Lim J, Ha M, Kim VN (2014) TAIL-seq: genome-wide determination of poly(A) tail length and 3′ end modifications. Mol Cell 53:1044–1052. https://doi.org/10.1016/j.molcel.2014.02.007

8. Woo YM, Kwak Y, Namkoong S et al (2018) TED-Seq identifies the dynamics of poly (A) length during ER stress. Cell Rep 24:3630–3641.e7. https://doi.org/10.1016/j.celrep.2018.08.084
9. Hite JM, Eckert KA, Cheng KC (1996) Factors affecting fidelity of DNA synthesis during PCR amplification of d(C-A)n•d(G-T)n microsatellite repeats. Nucleic Acids Res 24:2429–2434. https://doi.org/10.1093/nar/24.12.2429
10. Murray EL, Schoenberg DR (2008) Assays for determining poly(A) tail length and the polarity of mRNA decay in mammalian cells. Methods Enzymol 448:483–504. https://doi.org/10.1016/S0076-6879(08)02624-4
11. Hommelsheim CM, Frantzeskakis L, Huang M, Ülker B (2014) PCR amplification of repetitive DNA: a limitation to genome editing technologies and many other applications. Sci Rep 4:5052. https://doi.org/10.1038/srep05052
12. Krause M, Niazi AM, Labun K et al (2019) tailfindr: alignment-free poly(A) length measurement for Oxford Nanopore RNA and DNA sequencing. RNA 25:1229. https://doi.org/10.1261/rna.071332.119
13. Legnini I, Alles J, Karaiskos N et al (2019) FLAM-seq: full-length mRNA sequencing reveals principles of poly(A) tail length control. Nat Methods 16:879–886. https://doi.org/10.1038/s41592-019-0503-y
14. Workman RE, Tang AD, Tang PS et al (2019) Nanopore native RNA sequencing of a human poly(A) transcriptome. Nat Methods 16:1297. https://doi.org/10.1038/s41592-019-0617-2
15. Byrne A, Cole C, Volden R, Vollmers C (2019) Realizing the potential of full-length transcriptome sequencing. Philos Trans R Soc Lond Ser B Biol Sci 374:20190097. https://doi.org/10.1098/rstb.2019.0097
16. Garalde DR, Snell EA, Jachimowicz D et al (2018) Highly parallel direct RNA sequencing on an array of nanopores. Nat Methods 15:201–206. https://doi.org/10.1038/nmeth.4577
17. Liu H, Begik O, Lucas MC et al (2019) Accurate detection of m6A RNA modifications in native RNA sequences. Nat Commun 10:4079
18. Butler TZ, Pavlenok M, Derrington IM et al (2008) Single-molecule DNA detection with an engineered MspA protein nanopore. Proc Natl Acad Sci U S A 105:20647–20652. https://doi.org/10.1073/pnas.0807514106
19. Cherf GM, Lieberman KR, Rashid H et al (2012) Automated forward and reverse ratcheting of DNA in a nanopore at 5-Å precision. Nat Biotechnol 30:344–348. https://doi.org/10.1038/nbt.2147
20. Rang FJ, Kloosterman WP, de Ridder J (2018) From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. Genome Biol 19:90. https://doi.org/10.1186/s13059-018-1462-9
21. Jain M, Fiddes IT, Miga KH et al (2015) Improved data analysis for the MinION nanopore sequencer. Nat Methods 12:351–356. https://doi.org/10.1038/nmeth.3290
22. Wick RR, Judd LM, Holt KE (2019) Performance of neural network basecalling tools for Oxford Nanopore sequencing. Genome Biol 20:129. https://doi.org/10.1186/s13059-019-1727-y
23. De Coster W, D'Hert S, Schultz DT et al (2018) NanoPack: visualizing and processing long-read sequencing data. Bioinformatics 34:2666–2669. https://doi.org/10.1093/bioinformatics/bty149
24. Leger A, Leonardi T (2019) pycoQC, interactive quality control for Oxford Nanopore Sequencing. J Open Source Softw 4:1236. https://doi.org/10.21105/joss.01236
25. Li H (2018) Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics 34:3094–3100. https://doi.org/10.1093/bioinformatics/bty191
26. Durinck S, Spellman PT, Birney E, Huber W (2009) Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Nat Protoc 4:1184–1191. https://doi.org/10.1038/nprot.2009.97

Chapter 31

Nanopore RNA Sequencing Analysis

Tommaso Leonardi and Adrien Leger

Abstract

The recent advent of Nanopore sequencing allows for the sequencing of full-length RNA or cDNA molecules. This new type of data introduces new challenges from the computational point of view, and requires new software as well as dedicated analysis pipelines. In this chapter, we guide the reader through the typical analysis steps required to process the raw data produced by the instrument into a table of counts suitable for downstream analyses. We first describe the procedure to convert raw direct RNA-Seq and cDNA-Seq data into sequences in fastq format. We then outline how to perform quality control and filtering steps and how to map the filtered long reads to a reference transcriptome or genome.

Key words Nanopore, Transcriptomics, Direct RNA sequencing, cDNA sequencing, Bioinformatics

1 Introduction

The last few years have witnessed the adoption of new sequencing technologies for nucleic acids based on arrays of nanopores. The first company to commercialize a sequencing platform based on this type of technology was Oxford Nanopore Technologies (ONT), which has released methodologies for DNA sequencing as well as RNA sequencing [1]. The RNA sequencing protocols currently available are based on two fundamentally different approaches: the first one entails the retrotranscription of RNA into cDNA prior to sequencing (with an optional step of PCR amplification), whereas the second allows the direct sequencing of RNA molecules. Both approaches have remarkable advantages over previous methods: they allow for sequencing of full-length transcripts, they do not require PCR amplification and allow for the detection of alternative isoforms and splicing variants. Additionally, direct RNA sequencing is also free from biases induced by retrotranscription and allows for the detection of RNA modifications, at the expense of a higher error rate, higher input material requirements and lower throughput.

The analysis of Nanopore dRNA/cDNA sequencing experiments is highly dependent on the specific experimental setting and the biological questions of interest. Nevertheless, most analytical workflows will include raw data basecalling (i.e., conversion of the electric current recordings into a sequence), mapping and basic quality control steps.

This protocol will provide a general workflow to conduct these steps, in particular focusing on the following:

- Interpreting run quality metrics.
- Basecalling.
- Mapping (transcriptome/genome).
- Data quality control.
- Estimating transcript counts.

2 Materials

2.1 Operating System and Hardware Requirements

This protocol assumes all commands are run in a Bash shell on the GNU/Linux operating system. However, they should all function with little or no modification under other shells, other Unix-like operating systems (e.g., MacOS) as well as in the Windows Subsystem for Linux.

In terms of hardware requirements, the basecalling step is the most demanding in this analysis workflow, as it requires a machine with an i7 or Xeon CPU (with at least four cores), 16 GB of RAM and at least 1 TB of space on an SSD drive. Although not required, it is also highly recommended to have a CUDA-compatible NVIDIA GPU with NVIDIA compute version 6.1 or higher.

2.2 Fast5 Files

The main output of a Nanopore sequencing run is a folder (or multiple folders) containing a set of fast5 files. This format is a specification over HDF5 (Hierarchical Data Format) and these files contain the raw current measurements for each read. Each fast5 file can contain data for a single read or for multiple reads (multifast5 format), depending on the parameters specified at the beginning of the sequencing run. To simplify programmatic access and manipulation of fast5 file ONT has released a public API in python (https://github.com/nanoporetech/ont_fast5_api).

2.3 Reference Genome Sequence and Transcriptome Annotation

Mapping Nanopore reads requires the sequence of the reference genome in fasta format as well as an annotation of the reference transcriptome in GTF format. For most species, these files can be easily downloaded from the Ensembl FTP server (https://www.ensembl.org/info/data/ftp/index.html).

2.4 Software

Running a basic Nanopore sequencing analysis pipeline requires a computing environment with the following software installed (*see* **Note 1**):

- Bedparse (MIT license, optional) [2].
- PycoQC (GPL-3.0 license) [3].
- Bedtools (GPL-2.0 license) [4].
- Samtools (MIT license) [5].
- Minimap2 (MIT license) [6].
- NanoCount (MIT license).
- pyBioTools (GPL-3.0 license, optional).
- Guppy (proprietary software, freely available on the ONT website after registration).

The examples reported in this chapter assume that the commands are executed in Bash.

3 Methods

3.1 Interpreting Run Quality Metrics

The sequencing run is controlled by a software called MinKNOW, which runs on the host computer (i.e., a laptop in the case of MinION sequencers or the instrument's inbuilt computer in case of GridION). MinKNOW saves the sequencing data into fast5 files in a user specified folder, and optionally can also automatically perform basecalling. Along with the data files, MinKNOW also generates a report file in PDF format with diagnostic graphs about the run. Of particular interest for quality assessment and troubleshooting purposes are the total number of reads sequenced as well as the graph showing the pore occupancy over time, which is an indicator of the quality of library and yield of the sequencing run.

3.2 Basecalling

The first step in the analysis of Nanopore sequencing data is the basecalling of raw fast5 files into fastq format. There are multiple software that implement this functionality, developed both by ONT as well as by independent research groups. At the time of writing the *de facto* standard for basecalling is Guppy (*see* **Note 2**), the official algorithm developed by ONT and based on a Long short-term memory (LSTM) artificial recurrent neural network (RNN).

Guppy takes as input the fast5 files generated by MinKNOW and does the following:

- Generates a fastq file for each fast5 file containing the basecalled sequences and Phred qualities.

- Generates basecalled fast5 files that contain basecalled sequences in a dedicated slot (optional).
- Classifies fastq and fast5 files into pass/fail folders according to the average quality score of each read (optional).
- Generates summary files for the run.

3.2.1 Example Command

The guppy basecaller program has numerous command-line options to tune its default settings, and we encourage the users to read very carefully the help page and the descriptions of each option, which can be obtained by invoking *`guppy_basecaller --help`*.

3.2.2 cDNA-Seq

The following is an example command suitable for basecalling a standard cDNA sequencing experiment. This examples assumes that the MinKNOW output folder is located at /data/fast5_files. The parameters used have the following meanings:

- -i /data/fast5_files *indicates the path of the fast5 files to be basecalled.*
- -s /data/guppy *path where to save the output files.*
- --fast5_out *instructs guppy to save basecalled fast5 files in addition to fastq files (optional).*
- --recursive *looks for fast5 files recursively inside the path specified by -i.*
- --num_callers *number of basecalled threads to use.*
- --flowcell *Version of the flowcell used for sequencing.*
- --kit *version of the library preparation kit used.*

```
> guppy_basecaller \
-i /data/fast5_files \
-s /data/guppy \
--fast5_out \
--recursive \
--num_callers 5 \
--flowcell FLO-MIN106 \
--kit SQK-DCS109
```

In this example sequencing was done on a flowcell of version FLO-MIN106, while the library preparation kit was SQK-DCS109. For a full list of supported flowcell/kit versions users can invoke guppy with the `--print_workflows` option.

3.2.3 dRNA-Seq

A direct RNA sequencing run would instead require additional options, due to the reversed direction of the sequencing ($3' \rightarrow 5'$ as opposed to $5' \rightarrow 3'$), the presence of uracil instead of thymine, as well as the need for an optimized strategy for trimming the adapter's raw signal.

```
> guppy_basecaller \
 -i /data/fast5_files \
 -s /data/guppy \
 --fast5_out \
 --recursive \
 --num_callers 5 \
 --flowcell FLO-MIN106 \
 --kit SQK-RNA002 \
 --reverse_sequence true \
 --u_substitution true \
 --trim_strategy rna
```

3.2.4 Output Description

Guppy typically produces the following output files and folders:

- fast5_pass: contains the fast5 files that pass the qscore threshold.
- fast5_fail: contains the fast5 files that do not pass the qscore threshold.
- fastq_pass: contains the fastq files that pass the qscore threshold.
- fastq_fail: contains the fastq files that do not pass the qscore threshold.
- final_summary.txt: log file containing useful metadata on the sequencing run (instrument id, flowcell id, etc.).
- [run_id]_sequencing_summary.txt: run summary file reporting per-read quality metrics.
- report.md: MinKNOW report in Markdown format.
- report.pdf: MinKNOW report in PDF format.

The subdivision of fast5 and fastq files in pass and fail groups is controlled by the `--qscore_filtering` flag and by the `--min_qscore` parameter, which specifies the threshold below which reads are categorized as failed.

3.3 Mapping

After basecalling, the reads can be mapped to the reference transcriptome and/or genome to produce alignments in SAM/BAM format.

3.3.1 Generation of a Reference Transcriptome Fasta File

Several downstream applications (Tombo, Nanopolish, etc.) are unable to process spliced alignments. Due to this, it is often useful to map both to the reference genome of the species of interest as well as to the reference transcriptome. The sequence of the reference transcriptome can be easily obtained from a transcriptome annotation file (in gtf format) and the genome's sequence (in fasta format). For most species both files can be downloaded from Ensembl (under Downloads → Databases); for example, for the annotation file for the human transcriptome would be Homo_sapiens.GRCh38.98.gtf, whereas the genome sequence file would be Homo_sapiens.GRCh38.dna.primary_assembly.fa. Once these file have been downloaded, the sequence for the reference transcriptome can be easily obtained with bedtools:

```
#       To preserve transcript names in the fasta it's
convenient                    to                    convert
#                 the gtf file to bed12 format bedparse
gtf2bed /data/references/Homo_sapiens.GRCh38.98.gtf \
    >    /data/references/reference_transcriptome.bed
bedtools                 getfasta                     \
       -fi
/data/references/Homo_sapiens.GRCh38.dna.primary_assembly. fa \
       -s                                             \
       -split                                         \
       -name                                          \
       -bed /data/references/reference_transcriptome.bed     \
       > /data/references/Homo_sapiens.GRCh38.98.fa
```

In the command above the `-s` option ensures that a transcript's strand is taken into account when extracting its sequence, while the `-split` flag instructs bedtools to omit intronic regions, that is, to report the sequence of spliced transcripts. Alternatively, the cDNA sequences in fasta format for most species can be downloaded directly from Ensembl.

Finally, in both cases the resulting fasta file should be indexed with `samtools faidx`:

```
samtools faidx /data/references/Homo_sapiens.GRCh38.98.fa
```

3.3.2 Concatenate Fastq Files

Guppy produces multiple fastq files depending on the number of initial fast5 files and the number of threads used. Those files have to be concatenated into a single fastq. This can be easily achieved with the GNU Coreutils program `cat`. Alternatively, this step can also be carried out with the `FastqFilter` command of `pyBioTools`, which allows to remove duplicated reads, to filter based on sequence quality and length and to save the output in a compressed format:

```
pyBioTools Fastq Filter \
 -i data/guppy/fastq_pass/ \
 -o data/guppy/basecalled.fastq.gz \
 --remove_duplicates \
 --min_len 100 \
 --min_qual 7
```

3.3.3 Mapping to the Genome

Mapping to the genome in a splicing-aware fashion is the ideal strategy when the user is interested in novel genes/transcripts or when a good-quality annotation of the reference transcriptome is not available for the species of interest.

```
minimap2 -a -x splice -k14 -uf \
 /data/references/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
 /data/guppy/basecalled.fastq > /data/minimap/minimap_genome.sam
```

3.3.4 Mapping to the Transcriptome

Basecalled reads can also be aligned to the reference transcriptome, therefore not requiring that the alignment is performed in a splicing-aware fashion.

This is the recommended option for modification detection as the current resquiggling algorithms are not splice aware and hence might introduce errors at the splicing junctions. It also makes sense as transcripts are the natural setting for RNA modifications.

The following mapping commands are suitable for mapping direct RNA sequencing data, whereas the parameters need to be adjusted when mapping cDNA sequencing data.

```
minimap2 -a -x map-ont -k14 --for-only \
 /data/references/Homo_sapiens.GRCh38.98.fa \
 /data/guppy/basecalled.fastq \
 > /data/minimap/minimap_transcriptome.sam
```

The `--for-only` flag instructs minimap2 to only map to the forward strand of the reference sequences, whereas the `-uf` option limits the search for GT-AG canonical splicing sites to the transcript's sense strand only and not its reverse complement.

3.3.5 Alignment Filtering

Bam files have to be filtered to remove any reads that would be unmapped, secondary and supplementary as well as reads mapped on the reverse strand (SAM flag 2324). Depending on the downstream applications reads can also be filtered based on their alignment score.

```
samtools view /data/minimap/minimap_transcriptome.sam \
 -bh \
 -t /data/references/Homo_sapiens.GRCh38.98.fa \
 -F 2324 > /data/minimap/minimap_transcriptome.filt.bam
```

An alternative and more flexible option is to use the `Alignment Filter` command of `pyBioTools`, which also allows to filter based on alignment length and sequence identity between read and reference. This also automatically generates a Bam index file.

Finally, the alignments can be sorted and indexed.

```
pyBioTools Alignment Filter \
 -i /data/minimap/minimap_transcriptome.bam \
 -o /data/minimap/minimap_transcriptome.filt.bam \
 --skip_unmapped \
 --skip_supplementary \
 --skip_secondary \
 --min_align_len 100 \
 --orientation "+" \
 --verbose

samtools sort /data/minimap/minimap_transcriptome.filt.bam \
 -o /data/minimap/minimap_transcriptome.filt.sort.bam
samtools index /data/minimap/minimap_transcriptome.filt.sort.
bam
```

3.4 Data Quality Control

pycoQC is a lightweight tool to generate an interactive quality control report from datasets obtained with Oxford Nanopore base-callers. pycoQC uses the sequencing summary file generated by Guppy containing important prealignment metrics. This file can be found at the root of Guppy output directory, together with the fastq files. In addition, one can also generate a postalignment QC report by providing a BAM/SAM file obtained by Minimap2 or with another aligner.

```
# Basic usage for pre-alignment QC only
pycoQC -f sequencing_summary.txt -o pycoQC_report.html
# Usage for pre and post alignment QC
pycoQC -f sequencing_summary.txt -a aligned_reads.bam -o py-
coQC_report.html
```

3.5 Estimating Transcripts Count

To obtain a table of raw read counts per transcript users can simply count the occurrences of each transcript in the filtered SAM file resulting from the alignments to the transcriptome:

```
samtools view /data/minimap/minimap_transcriptome.filt.sort.
bam \
 | cut -f 3 | sort | uniq -c
```

For more accurate estimates of transcript abundance users can use dedicated algorithms that take into account mapping uncertainty. For example, NanoCount estimates transcript abundance from dRNA-Seq or cDNA-Seq experiments using an expectation–maximization approach, akin to what tools like RSEM, Kallisto, and Salmon do for short read data.

```
NanoCount -i /data/minimap/minimap_transcriptome.filt.sort.
bam \
 -o /data/counts.txt
```

4 Notes

1. Analysis workflows like the one outlined in these pages often depend on several software packages. Their installation can be a tedious and time-consuming process, which could also be complicated by version incompatibilities between the dependencies of each packages to be installed. Furthermore, it is often hard to keep track of the version of each program used in an analysis when they software is installed manually. To solve these issues and facilitate the execution of clean and reproducible analyses workflows, we encourage users to run these analyses inside software-managed environments (e.g., with Conda) or inside containers (e.g. Docker, Singularity). These tools greatly simplify software installation and allow each execution environments to be programmatically defined through text files (e.g., Dockerfiles or conda environment files).
2. The basecalling step with Guppy is usually the computational bottleneck in this type of analyses. Whenever possible we would encourage users to perform live basecalling during sequencing by activating the corresponding option in MinKNOW prior to starting the run. When this is not feasible (e.g., due to the hardware limitations of the computer running MinKNOW) it is possible to run Guppy in a low-accuracy mode, where the computing time is significantly reduced at the cost of a higher

basecalling error rate. The Guppy documentation and the ONT community forum will provide up-to-date instructions on how to achieve this.

References

1. Garalde DR, Snell EA, Jachimowicz D et al (2018) Highly parallel direct RNA sequencing on an array of nanopores. Nat Methods 15:201–206. https://doi.org/10.1038/nmeth.4577
2. Leonardi T (2019) Bedparse: feature extraction from BED files. J Open Source Softw 4:1228. https://doi.org/10.21105/joss.01228
3. Leger A, Leonardi T (2019) pycoQC, interactive quality control for Oxford Nanopore Sequencing. J Open Source Softw 4:1236. https://doi.org/10.21105/joss.01236
4. Quinlan AR (2014) BEDTools: the Swiss-Army tool for genome feature analysis: BEDTools: the Swiss-Army tool for genome feature analysis. Curr Protoc Bioinforma 47:11.12.1–11.12.34. https://doi.org/10.1002/0471250953.bi1112s47
5. Li H, Handsaker B, Wysoker A et al (2009) The sequence alignment/map format and SAMtools. Bioinformatics 25:2078–2079. https://doi.org/10.1093/bioinformatics/btp352
6. Li H (2018) Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics 34:3094–3100. https://doi.org/10.1093/bioinformatics/bty191

Index