UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science
Information Technology Curriculum

Henry Mägi

RNA-seq data analysis pipeline

Bachelor's Thesis (6 ECTS)

Supervisor:	Hedi Peterson, MSc
Supervisor:	Kersti Jääger, PhD

Tartu 2014

RNA-seq andmete analüüsi töövoog

Lühikokkuvõte: Kuna bioloogid viivad läbi suurel hulgal ülegenoomseid geeniekspressiooni eksperimente, on tekkinud vajadus töövoo jaoks, millega saaks töödelda ning analüüsida RNA-seq andmeid. Selline töövoog koosneb erinevatest arvutuslikest tööriistadest ning sisendfaili tüüpidest, mis teeb ühtse töövoo arenduse raskeks ülesandeks, kuid teeks teadlastele andmete analüüsi ja tulemuste tõlgendamise palju lihtsamaks. Kohandatud töövoogu on lihtsam rakendada, kuid see nõuab, et kasutaja oleks tuttav kõikide arvutuslike tööriistadega, millest töövoog koosneb. Käesoleva töö eesmärk oli kirjeldada detailselt RNA-seq andmete analüüsi töövoo loomist ning rakendamist. Saadud tulemustest võib järeldada, et ühtse töövoo tarkvara iRAP vajab veel edasiarendust. Lisaks sellele aitavad tulemused paremini mõista erinevate tööriistade funktsioonidest ning nende potentsiaalsetest parandustest.

Märksõnad: RNA-seq, geeniekspressioon, töövoog, TopHat, Cufflinks, iRAP

RNA-seq data analysis pipeline

Abstract: The vast amount of large-scale gene expression experiments carried out by biologists has created the need for a pipeline to process and analyse RNA-seq data. The pipeline consists of different computational tools and data input types which makes developing an integrated pipeline a challenging task but would make the use of the workflow much easier for researchers. A customized pipeline, on the other hand, is easier to implement but needs the user to be familiar with all of the computational tools that the pipeline consists of. The aim of this thesis was to provide good knowledge on creating and running a typical RNA-seq data analyis pipeline. The results obtained allow to conclude that the integrated pipeline iRAP still needs development. Also, the results create a better understanding of the functions and potential improvements of different tools.

Keywords: RNA-seq, gene expression, pipeline, TopHat, Cufflinks, iRAP

Dictionary:

- alignment - arrangement of sequence reads in correct positions on the reference genome
- annotation - biological explanation or function of a gene collected into a database
- cDNA - complementary DNA
- exon-exon junction or splice junction - marks a position in mRNA from which protein non-coding regions called introns have been removed by a biological process called splicing
- gene - a fragment of DNA that carries a defined biological function
- genome - the entire genetic library of an organism
- mapping - assigning already aligned reads to transcripts
- microarray - microscale chip covered with known short nucleotide sequences called probes
- mRNA - messenger RNA; a type of RNA that is translated into protein in the cell
- NGS - next generation sequencing
- nucleotides - building blocks of DNA and RNA
- read - a digitalized raw sequence
- RNA - ribonucleic acid
- sequencing - chemical detection of the nucleotide order of DNA and RNA
- transcript - single RNA molecule encoded from a gene; often mRNA
- transcriptome - the entire collection of transcripts encoded from the genome

Contents

1 Introduction

The goal of this thesis is to give an overview of the computational analysis steps of RNA-sequencing (RNA-seq) data and describe a typical RNA-seq analysis pipeline. The need for creating this pipeline has arisen from the vast amount of large-scale gene expression experiments carried out by biologists. In technical terms, RNA-seq is a method to measure the abundance of transcripts by counting the sequenced reads that map to certain regions in the genome. Biologically, RNA-seq enables researchers to detect the expression levels of different genes. The computational part of the analysis pipeline begins with getting sequenced reads (raw data) from machines specifically engineered for high-throughput DNA sequencing and ends with the generation of a gene expression matrix followed by a statistical analysis of the data. The author of this thesis has:

1. processed relevant literature to give a detailed overview of the RNA-seq data analysis pipeline;
2. worked with and analyzed different computational tools;
3. assembled these tools into a single working pipeline;
4. tested and analyzed an implementation of an integrated pipeline named iRAP;
5. tested the RNA-seq data analyisis pipeline with two different datasets.

Motivation section (Section 2) briefly introduces the principle of RNA-seq technique and its applications in biology and medicine. Also, RNA-seq and microarray technology, another widely used but older high-throughput gene expression profiling technique, are compared.

The main section (Section 3) describes a typical RNA-seq data analysis pipeline. The section gives a short overview of steps done in the experimental design and sample preparation followed by a thorough explanation of processing the raw reads, mapping the processed reads and analysing the mapped data.

Details of the computational methods section (Section 4) focuses on different types of input files, computational tools and necessary computational resources to run these tools. Also, a computational pipeline implementation, iRAP, is compared to a customized pipeline protocol.

Lastly, example datasets section (Section 5) describes the public datasets ran through the pipeline: *Homo sapiens* (human) and *Canis familiaris* (dog) - with associated comments and results from these datasets.

2 Motivation

Biological systems are complex and require computational means to resolve the questions about the life of organisms. The aim of this thesis is to describe a step by step approach for RNA-seq data analysis. RNA-seq is a method that allows to measure genome-wide gene expression levels and thereby answer questions about the function of genes and cells. This technique enables researchers to identify the genes that are expressed at specific time in distinct conditions on a genome-wide scale. More and more RNA-seq data is being generated both in basic research and in clinical setting which is why the demand for coherent data analysis workflows is increasing. Another demand is for a computational pipeline software acting as a whole to simplify the implementation of the pipeline. A potential pipeline, iRAP (see Section 4.3), is created for this purpose, and testing iRAP has been one part of my work.

2.1 RNA-seq technique

RNA-seq experiment starts in the wet-lab by lysing (breaking) the cells of interest and isolating the RNA. A fraction of isolated RNA named mRNA (originally encoded from the DNA within the cell) represents the active genes of the cell. Upon multiple sample preparation steps in the lab, the isolated RNA is converted to short fragments of complementary DNA (cDNA). This collection of fragments called 'cDNA library' is then applied to high-throughput sequencing. The sequencing reaction results in hundreds of millions of 'reads' - short DNA fragments - that can be computationally mapped back to the genome for transcript (that is, mRNA) identification. By counting the reads that map to the same region in the genome, transcript abundance (that is, gene expression level) can be calculated.

Gene expression is tightly linked with cell function and identity. RNA-seq captures a so-called snapshot of currently active genes in a certain time and condition. By knowing what genes are activated in a cell in response to applied stimuli or to natural changes in cellular environment, researchers can identify, for example, genes responding to these changes. Also, RNA-seq gene expression data is often used to compare normal and cancer cells to identify which genes could be related to or even cause cancer. New diagnostic tools and cancer therapies could be developed based on this information. Additionally, RNA-seq gene expression profiling is used to study cell differentiation - movement of a cell from one functional state to another - to understand how to turn stem cells into a desired cell type for cell replacement therapy applications. In this thesis, one example RNA-seq dataset describes gene expression profile of human liver cells that were derived from human stem cells [1].

2.2 RNA-seq versus microarray

Prior to RNA-seq, researchers used microarray technology for genome-wide gene expression profiling. However, due to multiple advantages of sequencing over microarray technique, RNA-seq is shaping into a more preferred method for high-throughput gene expression analysis. [2]

Microarray technology uses a chemically prepared microchip with probes complementary to known sequences attached to the chip. This evidently means that only those genes that are already known, can be analyzed using microchip-based technique, whereas RNA-seq generates read output from both known and unknown genes. Therefore, RNA-seq enables *de novo* (meaning "from the beginning") discovery of new genes and alignment of new sequences with base-pair (nucleotide) resolution, allowing to reanalyze the data once there is more information available about the structure of genomes. Microarray technique is based on a chemical hybridization reaction (binding between the sample of interest and the probe based on complementarity), which may result in high background signal due to cross-binding. This limits the signal detection level of microarray technique whereas RNA-seq exhibits a large dynamic range and can be applied to detect both highly expressed genes and rare transcripts.

3 Typical RNA-seq data analysis pipeline

Computational stages of the pipeline can be very complex because of the amount of possible arguments needed to receive a high-quality result and therefore a good overview is necessary. This section covers the stops of a typical RNA-seq data analysis pipeline with special focus on the computational stages of the analysis (see Figure 3). Since the details of the experimental design and generation of cDNA libraries are outside the scope of this thesis, only a short overview of these parts is provided.

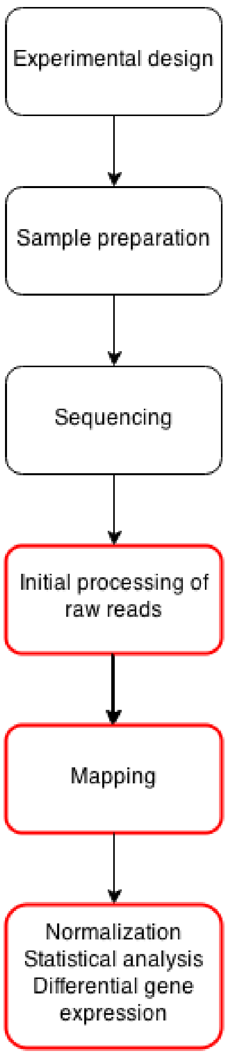


Figure 1: Typical RNA-seq data analysis pipeline. Stages marked in red are the focus of this thesis.

3.1 Experimental design

RNA-seq data analysis pipeline starts with designing of an experiment. Experimental design should be done carefully since a poorly designed experiment can result in useless data with a lot of time and money gone to waste. The main thing a researcher should decide is whether the objective is to gain qualitative or quantitative information from RNA sequencing. Qualitative data provides information on annotation and requires deeper sequencing of individual samples, while quantitative data enables to perform differential gene expression analysis. Therefore, more experimental replicates to compare are needed.

Accurate annotation results in identification of expressed transcripts, exon/intron boundaries and transcriptional start sites. Differential gene expression, on the other hand, measures differences in expression, alternative splicing and alternative transcriptional start sites between two or more groups. [3]

3.2 Generation of cDNA library by sample preparation

After the experiment is designed, a chemical cDNA library preparation will be done starting from harvesting the RNA. In order to get the highest quality reads, the quality of cDNA libraries should be validated and the libraries should be quantified before sequencing. Next generation sequencing (NGS) can be done by a variety of sequencing platforms. The most popular ones for RNA-seq are Illumina and SOLiD machines which provide deep sequencing and suitable read length for alignment to either reference genome or for transcriptome assembly.

3.3 Initial processing of raw reads: quality control

The computational stage of RNA-seq data analysis pipeline starts with controlling the quality of raw sequence data output from sequencing. FastQC [4] is one tool that can process BAM, SAM and FASTQ formats, preview the data for potential problems, create graphs and tables for summarization purposes and generate HTML reports from the results. FastQC is compatible with integration into a custom pipeline which makes it a valuable tool. In addition to the previous tool's functions, there is a need for removing sequencing adapters, filtering and trimming of reads and converting sequences from RNA to DNA or vice versa. FASTX [5] is one of the programs that can perform these functions. In case the raw sequence data is from the Sequence Read Archive (SRA, `http://www.ncbi.nlm.nih.gov/sra`), the data must be converted to a format supported by the alignment program. An official tool, SRA Toolkit [6], has been created for such purposes. It is very important to know whether the data is in the form of single-end or paired-end reads since converting paired-end data from SRA format as single-end data will result in failed alignment. The difference between single-end and paired-end reads comes from sequencing: single-end reads are DNA fragments derived from one end only while paired-end reads represent both ends of the same DNA fragment, resulting in two files for raw read data instead of just one.

3.4 Alignment of processed reads to reference sequence

Before mapping the processed reads, a reference genome (FASTA file) and transcripts annotation (GTF/GFF file) should be downloaded for the appropriate species. Ensembl [7] and National Center for Biotechnology Information [8] are two example sites that provide public genome and annotation data. It is not recommended to mix data from different databases since many alignment tools are not compatible with it. If the reference genome is a collection of FASTA files (e.g. chr1.fa, chr2.fa, ...) then they should be merged into a single FASTA file because it is more easily manageable.

Properly processed reads are then mapped by using an alignment program. TopHat2 (see Section 4.2.2), for example, is a good alignment program for a typical RNA-seq data analysis pipeline because it is well-documented, often updated and improved and has a wide selection of functions. TopHat2 requires specific indexes to start mapping. These indexes can be built with Bowtie (see Section 4.2.1). Preferrably, BT2 format indexes should be built because it is an improved format. An alternative is to download pre-built indexes from Bowtie2's website which are in EBWT format (Bowtie1). Building the indexes requires a reference genome. By default, without supplying TopHat2 with transcript annotation file, TopHat2 aligns the reads only to the genome. To align reads to the transcriptome, the annotation file must be provided. TopHat2 will use the annotation file to create a separate Bowtie index. If necessary, TopHat2 supplies an option to map reads only to the transcriptome without mapping to the genome. Another important argument to provide for TopHat2 is regarding the type of library. There are three different options for that: unstranded library (default), first strand library or second strand library.

In cases where TopHat2 run fails due to external problems (for example, when the system runs out of memory or disk space), a resume function is possible. Also, TopHat2 supports multithreading which makes the aligning much faster on systems using multi-core processor(s). After TopHat2 finishes, five different files can be found in the output directory that is separate for each sample. The default is "./tophat_out" but it can be changed. The most useful output file is "accepted_hits.bam". It contains aligned reads and is used by analysis tools. The other files, "junctions.bed", "insertions.bed" and "deletions.bed", are reports generated to represent found junctions and performed insertions/deletions.

3.5 Calculating transcript abundances: raw and normalized read counts

In order to calculate transcript abundances, the mapped data needs to be processed. Cufflinks2 (see Section 4.2.3), being a well supported and actively developed software, is built on processing mainly TopHat2 reads and therefore is a good fit for the pipeline. Firstly, the mapped reads in SAM or BAM format need to be ran with *cufflinks*. It is a good practice to additionally supply an output folder location for each BAM file (for example, "experiment1_clout", "experiment2_clout", ..., "experimentN_clout"). The accuracy of *cufflinks* can be improved by providing the reference genome in MultiFASTA file format with "-b" option (MultiFASTA needs to be converted from FASTA format). Additionally, if isoform expression estimation is needed, the reference transcript file should be provided with the "-g" option. After *cufflinks* assembles the transcripts, "transcript.gtf" files can be found in the output folders. Each processed BAM file has its own transcripts' assembly file.

Assemblies need to be merged with *cuffmerge* by providing it a text file with locations of the assembly files (if following the good practice example, the text file would consist of "experiment1_clout/transcripts.gtf", "experiment2_clout/transcripts.gtf", ..., "experimentN_clout/transcripts.gtf", each location being on a separate line). Optionally, a reference transcripts file can be provided with the "-g" option which *cuffmerge* will assemble with the transcript assembly files. The output of *cuffmerge* is a single file, "merged.gtf", that is an assembly of the input files.

The single assembly file, "merged.gtf", can be used with *cuffquant* that calculates the transcript abundances. Alignments not being structurally compatible with the reference transcript will be ignored. The input of *cuffquant* is an aligned read file in BAM format produced by TopHat2 with the "merged.gtf" file produced by *cuffmerge*, therefore *cuffquant* needs to be run separately for each experiment. A good practice is yet again to provide output folder locations (if the previous good practice suggestions are followed then these are "experiment1_clout", "experiment2_clout", ..., "experimentN_clout"). The output of *cuffquant* is "abundances.cxb" file for each quantified experiment. Transcript abundance estimates accuracy can be improved by providing the same MultiFASTA file, as discussed above, with the "-b" option.

At this point, there are two options: produce normalized expression levels for each gene, transcript, transcriptional start site group and coding sequence group with *cuffnorm* or perform differential expression analysis with *cuffdiff*. Both tools take the "merged.gtf" and "abundances.cxb" files produced as an input. The tool *cuffdiff* also accepts the MultiFASTA file mentioned above with the "-b" option to improve the accuracy of transcript abundance estimates. Both tools produce

FPKM values that hold the read counts. In addition to FPKM values, *cuffnorm* produces files containing information about samples and each gene, transcript, transcriptional start site group and coding sequence group as text files delimited by tab. The tool *cuffdiff* produces additionally count tracking files, read group tracking files, differential expression and splicing tests, differential coding output, differential promoter usage and read group information.

It should be noted that there are multiple advanced options for each tool in Cufflinks2 pipeline and even these do not cover all applications of the RNA-seq method. Also, RNA-seq as a method being constantly developed from which we can conclude that Cufflinks2 has many more functions to be developed.

4 Details of the computational methods

4.1 Input files

The standard of files used as an input in different stages of RNA-seq data analysis is not yet fully developed but there are multiple file formats that can be considered as a default. The main file formats used in the mapping stage are: GTF/GFF, FASTA and FASTQ. GFF and GTF formats are used for known gene annotations and transcripts data. FASTA format, representing DNA or protein sequences is widely used for reference genome data. FASTQ format, based on FASTA, is developed for sequences and corresponding quality scores. Mapping stage is followed by the analysis stage which generally accepts the following formats: SAM/BAM, GFF/GTF and BED. SAM and BAM formats are used for storing alignment data where SAM is text-based and BAM is a binary file. BED format is used to define genomic regions. In addition to the previous formats, SRA format is also used which is an archive for raw data.

The general feature format [9] (GFF) is a widely used plain text format by bioinformaticians to represent genomic features. The latest version (as of 06.05.2014) is GFF version 3. The gene transfer format [10] (GTF) is identical to GFF version 2 [11] and is another common format for representing genomic data. This format consists of rows where each row holds 9 columns. The fields are separated by tab and are defined below.

1. <seqid> - the identifier of the feature.
2. <source> - the source of this feature.
3. <type> - the type of the feature.
4. <start> - the starting position of this feature.
5. <end> - the ending position of this feature.
6. <score> - a floating point value.
7. <strand> - the value is '+' for forward strand, '-' for reverse strand, '.' for not stranded features and '?' for relevant, but unknown strands.
8. <phase> - the phase which indicates where the features begins with reference to the reading frame.
9. [attributes] - a list of attributes about the feature.

FASTA [12] format is used to represent either nucleotide or peptide sequences with describing nucleotides and amino acids as single-letter codes [13]. A sequence data begins with a single line describing the sequence and it is distinguished with the symbol '>'. The first word following the symbol identifies the sequence and the rest describes it. All other lines before the next greater-than symbol are considered to be either nucleotides' or amino acids' data. Blank lines are not allowed and it is recommended that the lines' length does not exceed 80 characters. RNA is a nucleic acid and the supported basic codes for it are provided in Table 1.

Nucleotide	**Code**
Adenosine	A
Cytidine	C
Guanine	G
Thymidine	T
Uridine	U
Any nucleotide	N

Table 1: Basic codes for nucleic acids in FASTA format

FASTQ format [14] is used to store a biological sequence together with its quality scores. It is a text-based format and generally uses 4 lines per read. The first line, beginning with a '@' character, represents the identifier of a sequence with an optional description. The second line contains the raw sequence letters [13]. The third line begins with a '+' character and optionally the identifier together with more information. The fourth line contains quality scores for each letter in the sequence. The quality scores, also known as PHRED scores, describe the sequencing quality which estimates the probability of error. FASTQ files from RNA-seq experiments are usually in Sanger or Solexa/Illumina format. The differences between the two formats are provided in Table 2. In context of Sanger or Solexa/Illumina formats, PHRED score is represented by an ASCII character which has an integer value. The reason for using this error estimation format is because using one character instead of numbers and spaces is more compact and readable by a human and it gives a very broad range of error probabilities (in case of Sanger format, the range is from 1.0, a wrong read, to $10^{-9.3}$, a very precise read).

Format	**ASCII interval**	**PHRED interval**
Sanger	33-126	0 to 93
Solexa/Illumina	64-126	0 to 62

Table 2: Basic codes for nucleic acid in FASTA format

Sequence alignment/map [15] (SAM) format is used for representing aligned reads data. Binary alignment/map (BAM) format is simply the binary version of this format. SAM files are in the form of tab separated text (except lines starting with "@CO") consisting of headers and alignments. Headers are lines starting with "@" symbol and hold fields, separated by tab, which are designed as "TAG:VALUE" where "TAG" field describes the "VALUE" field's content and format. Alignments generally hold the linear alignment data, consisting of reference sequence name, mapping quality, alignment position, aligned sequence and other information, of certain segments from a raw read.

Browser extensible data (BED) format is meant for defining annotation track's data without having very specific rules. It is a tab-delimited text file which has three required fields on each line: the name of the chromosome (e.g. "chr1" or "1"), the starting position of a feature the chromosome and the ending position of a feature in the chromosome). There are nine additionally usable fields for more specific information about each annotation track. For example, a BED file can be used in the UCSC Genome Browser to view the data graphically [16].

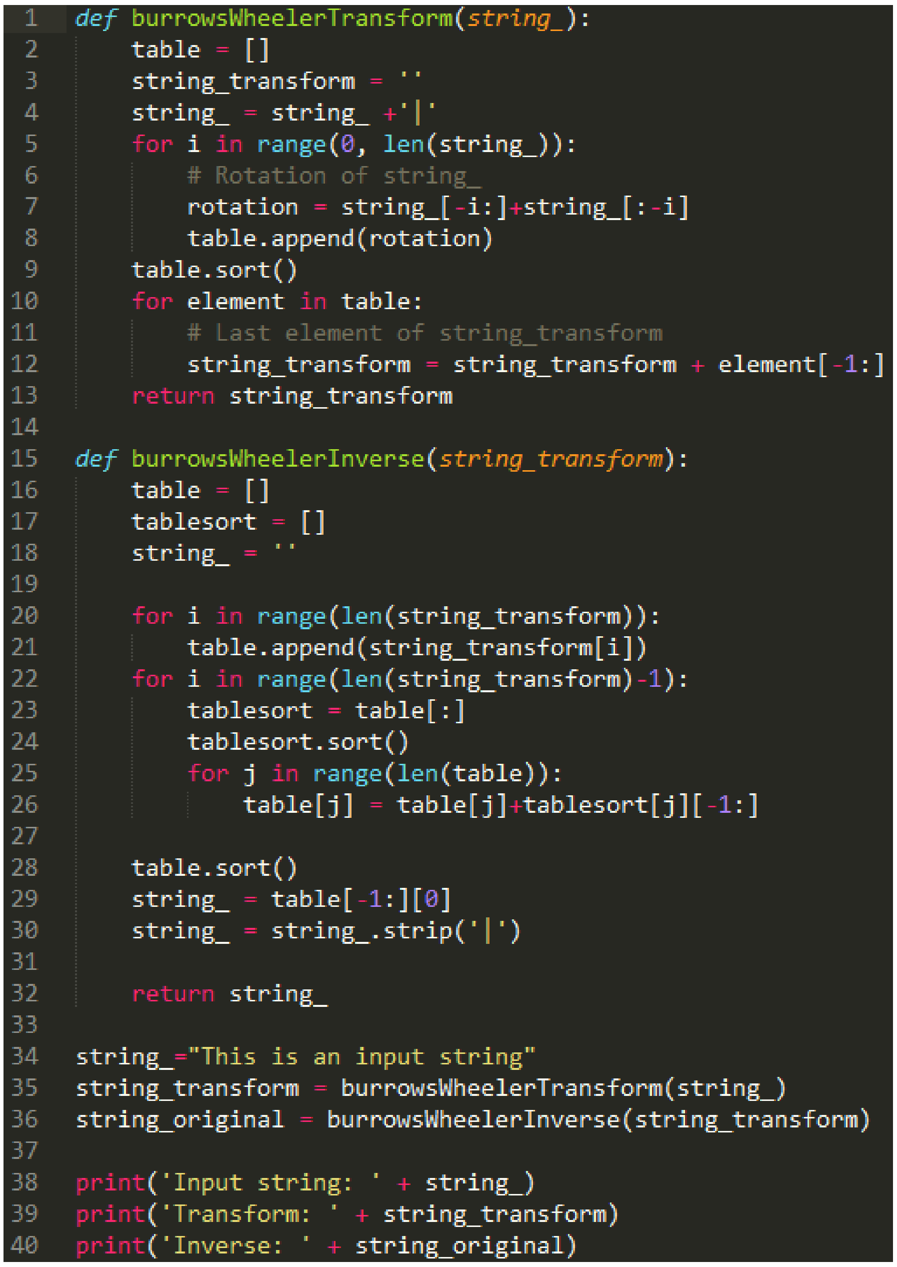
Sequence read archive (SRA) format is generally used to compress raw sequencing reads data and store them in archives like DNAnexus [17] or The Sequence Read Archive [18]. The accepted formats for conversion to SRA format are: BAM, SFF, HDF5, SOLiD, FASTQ, SRF and Illumina native.

4.2 Computational analysis tools

This section focuses on specific computational methods of RNA-seq data analysis. Namely, the methods focused on are Bowtie [19], TopHat [20] and Cufflinks [21]. They each serve a separate purpose in the pipeline. Bowtie is an open-source alignment tool for aligning DNA sequencing reads to long genomes and it is written in C++. TopHat aligns RNA-seq reads to a genome with the aim to identify exon-exon splice junctions and it is built on Bowtie as a Python script. Cufflinks, on the other hand, is also an open-source tool but it is meant for assembling aligned RNA-seq reads into transcripts, quantifying gene expression and testing for differential expression and regulation.

4.2.1 Bowtie

Bowtie was created with the purpose of having a faster and more memory-efficient alignment program than the ones existing at that time. Bowtie implements extended full-text minute-space (FM) index [22] which is based on Burrows-Wheeler transform (BWT) [23] (see Figure 2). The extension is necessary because the original algorithm does not allow sequencing errors or genetic variations. The trade-off to achieve Bowtie's high mapping speed is accuracy but it is reasonable because of high computing costs. Bowtie also supplies an option to increase the sensitivity resulting in a greater accuracy but that causes the computation time to be longer.



```python
def burrowsWheelerTransform(string_):
    table = []
    string_transform = ''
    string_ = string_ +'|'
    for i in range(0, len(string_)):
        # Rotation of string_
        rotation = string_[-i:]+string_[:-i]
        table.append(rotation)
    table.sort()
    for element in table:
        # Last element of string_transform
        string_transform = string_transform + element[-1:]
    return string_transform

def burrowsWheelerInverse(string_transform):
    table = []
    tablesort = []
    string_ = ''

    for i in range(len(string_transform)):
        table.append(string_transform[i])
    for i in range(len(string_transform)-1):
        tablesort = table[:]
        tablesort.sort()
        for j in range(len(table)):
            table[j] = table[j]+tablesort[j][-1:]

    table.sort()
    string_ = table[-1:][0]
    string_ = string_.strip('|')

    return string_

string_="This is an input string"
string_transform = burrowsWheelerTransform(string_)
string_original = burrowsWheelerInverse(string_transform)

print('Input string: ' + string_)
print('Transform: ' + string_transform)
print('Inverse: ' + string_original)
```

Figure 2: Example Burrows Wheeler transform implementation in Python (appendix named "bwt.py")

Over time, sequencing technology has been moving onto higher sequencing throughput and read length. This caused the developers of Bowtie to create Bowtie2 [24]. The latest release (as of 27.04.2014) is Bowtie 2.2.2 [25]. Bowtie2 consists of the main program *bowtie2* and three tools: *bowtie2-align*, *bowtie2-build* and *bowtie2-inspect*. The main program *bowtie2* takes an index file, which is created with *bowtie2-build*, and sequencing read files from an experiment and creates a set of alignments in SAM format [15] with *bowtie2-align*. The tool *bowtie2-inspect* is used to get information about Bowtie2's index and reference files that were used to create the index. In our RNA-seq data analysis pipeline, however, we only need to use *bowtie2-build* and *bowtie2-inspect* because these tools produce the necessary files and information in order to use TopHat2 (see Section 4.2.2).

4.2.2 TopHat

TopHat is an alignment program specifically created for RNA-seq experiments. It is built on Bowtie and it is written in C++ and Python. The release of Bowtie2 caused the developers of TopHat to create TopHat2 [26]. The most recent version (as of 30.04.2014) is TopHat 2.0.11 [27]. TopHat is designed to map RNA-seq reads to a genome with the purpose of finding exon-exon splice junctions without having to rely on a reference annotation. The software was originally developed to work with the reads from Illumina machines.

The workflow of TopHat is a two-phased operation. The first phase is an initial mapping of the reads to the genome using Bowtie or Bowtie2. Reads which do not map to the genome are classified as 'initially unmapped reads' (IUM) and are used in the final stages of TopHat's pipeline to search for span junctions. The second phase is building a database of possible splice junctions using Maq [28] followed by a final mapping of the reads against the newly built database using TopHat's own algorithm.

4.2.3 Cufflinks

Cufflinks is a program that assembles transcripts, calculates transcript abundances and tests for differential expression and regulation transcriptome-wide. Because Cufflinks relies on TopHat, Cufflinks2 was developed to be consistent with TopHat2. The latest version (as of 4.05.2014) is Cufflinks 2.2.0 [29].

Cufflinks2 consists of the main program *cufflinks* and additional tools *cuffmerge*, *cuffquant*, *cuffdiff* and *cuffnorm*. The main program assembles transcripts by using reads mapped by TopHat2 as an input. The next tool in the pipeline is *cuffmerge* which performs final transcriptome assembly and takes the output of *cufflinks*. This is followed by *cuffquant* which takes *cuffmerge*'s output together with reads mapped by TopHat2. The tool *cuffquant* outputs files in CXB format which are an input to either *cuffdiff* and/or *cuffnorm*. The tool *cuffdiff* performs differential expression while *cuffnorm* creates normalized expression and count tables. Differential expression results can be used by, for example, CummeRbund [30] to create expression plots. Normalized expression and count tables can be used with R [31], MATLAB [32] etc. Typical workflow of Cufflinks2 (version 2.2) is provided in Figure 3.

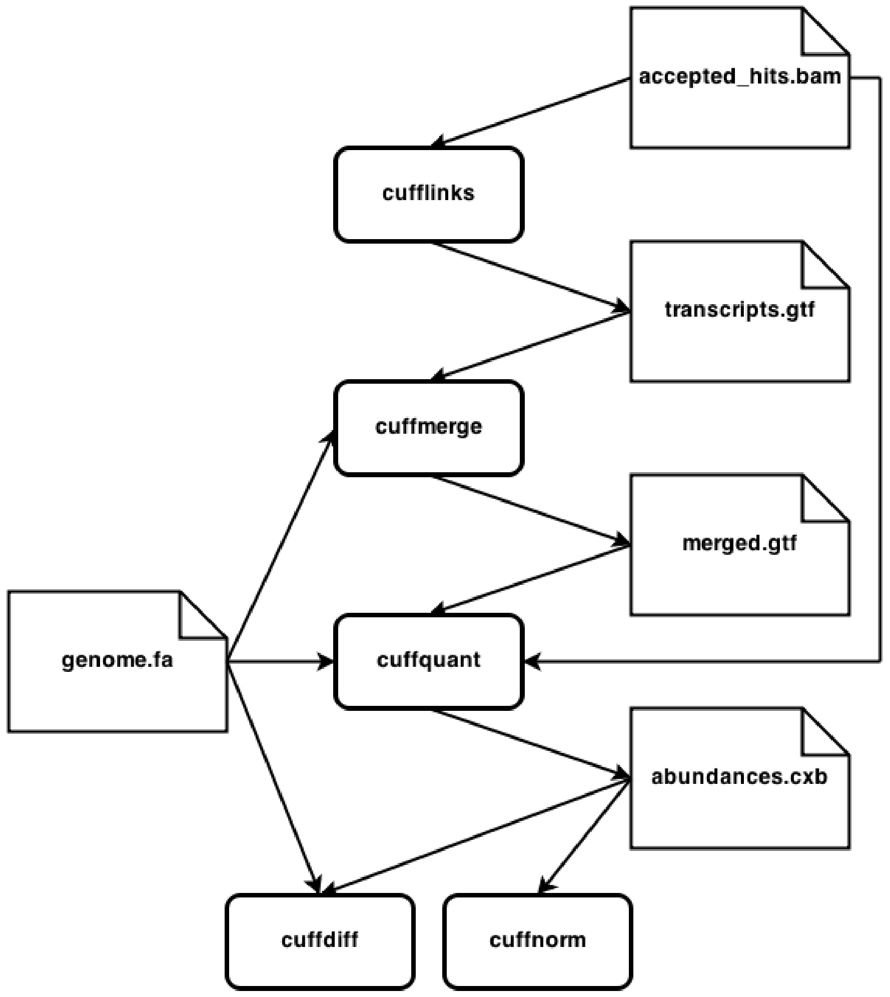


Figure 3: Typical Cufflinks2 pipeline

4.3 iRAP

The software iRAP [33] is being developed with the purpose of implementing an integrated RNA-seq pipeline. It covers the main pipeline (see Figure 3) described in this thesis with additional tools and functions. iRAP implements multiple mapping, quantification and differential expression tools by using a single configuration file that needs to be created by the user with information about the input files and necessary methods. This pipeline is created for command line usage on Linux operating systems. Properly working iRAP would have several advantages over individual tools, for example:

1. it is a software for automatic RNA-seq data analysis;
2. it gives a good example of how a RNA-seq data analysis pipeline should work;
3. it can be used by (early-stage) bioinformaticians in their research without the need for a thorough knowledge of different RNA-seq data analysis tools.

4.3.1 iRAP versus customized pipeline

While the idea of iRAP grew out from a necessity for bioinformatic analysis of RNA-seq data, it holds many challenges. First of all, the implementation needs to be user friendly. Software like that needs to be well documented and self-evident because the end user is subjected to various errors. If a mistake breaks the pipeline, the user needs to begin troubleshooting which is hard without a good knowledge of how the software works. On the other hand, if the mistakes do not break the pipeline, the resulting data might be useless causing valuable resources, time and money, go to waste.

Secondly, the software needs to be reliable. Bugs breaking the pipeline should be rare and they should result only in very specific situations not directly caused by the end user. If a bug cannot be avoided by the user, another way of performing the pipeline needs to be found which typically is a customized pipeline where each tool is used manually.

Thirdly, the software should use resources efficiently. Mixing single-threaded programs with multithreaded programs in a computational cluster results in idle resources that cannot be used by other users. Some clusters charge the user by reserved resources. This causes the user to overpay simply because the software reserves resources but does not use them.

All of the challenges discussed above are the main deficiency of iRAP and are yet to be resolved. This is why a customized pipeline is better than iRAP in its current state because it can settle these challenges. Each tool in the pipeline is sufficiently documented and updated while depending on the end user instead of the integrated pipeline. When a tool does not perform as wished, another tool can be used instead. Also, the user can specify resources separately for each tool so there is no need to overpay.

4.3.2 Potential iRAP improvements

Until iRAP is developed to a state where the bugs are minimized, an automated testing module should be implemented. Currently, iRAP only has a dry run function which checks for basics to start the pipeline but does not actually test the pipeline itself. The module should take fragments of the input data and perform a full but quick run to see if anything breaks. It would be useful to developers by reducing the number of faulty updates and it would help end users start troubleshooting immediately in case something does not work correctly.

Additionally, a proper documentation should be created that covers all parts of the software. The documentation should provide solutions to common problems and cover different example pipelines.

5 Implementation of the pipeline on example datasets

The more simple RNA-seq analysis pipelines can be basically run by modern computers that are used in a home environment because modern multicore processors provide the sufficient computational power. For testing our RNA-seq pipeline, we chose datasets from human and dog RNA-seq experiments. Since they were more compute-heavy, we used 60 cores from Intel Xeon E7-2860 processors running at 2.27 GHz clock speed. The experiments used as examples for the RNA-seq data analysis pipeline are the following:

1. *Homo sapiens* (human) - GSM1124072 [34] samples.

2. *Canis familiaris* (dog) - SRP016141 [35] samples.

The machine used for these experiments was Alligaator (`http://www.hpc.ut.ee/alligaator_usage`, High Performance Computing Center, University of Tartu).

5.1 Human dataset experiment

The raw reads from human samples were roughly 13 GB in total size and consisted of 7 files in SRA format. After the SRA to FASTA format conversion, the result was 14 paired-end read files with a total size of roughly 74 GB. The data was gathered from human embryonic stem cell-derived hepatoblasts (liver cells) and sequenced with Illumina Hiseq 2000 machine. The reason for choosing this data for experimentation purposes is the thorough paper describing the original experiment and supplementary file describing FPKM values. The genome was downloaded from UCSC [16] and annotation files were downloaded from Ensembl [7].

Firstly, the genome chromosome names were converted from "chrN" to "N". After that, the reads were submitted into iRAP's pipeline starting with processing the reads with FastQC and FastX. The reads were then mapped to the genome with TopHat2 and after that the iRAP's pipeline had to be stopped due to bugs related to early stages of development. The resulting mapping data could not be processed by Cufflinks2 either but judging by the alignment summary, the reads were correctly mapped. We then mapped unprocessed reads to transcriptome only with TopHat2. The resources used by mapping reads from human samples to the genome and transcriptome are provided in Table 3 and 4. Results of mapping can be seen in Table 5. The mapping rates provided are not different because the data mapped to transcriptome was not processed but because mapping to genome generally has more coverage.

The results from mapping to transcriptome with TopHat2 were then submitted into Cufflinks2 pipeline starting with *cufflinks.* After *cufflinks* finished, *cuffmerge* was ran with the original annotation file and transcript annotation files produced

Resource type	**Resources used**
CPU time	1891 hours
Walltime	66 hours
Memory	34.5 GB
Virtual Memory	38.4 GB

Table 3: Resources used by mapping to genome with TopHat2

Resource type	**Resources used**
CPU time	1768 hours
Walltime	38 hours
Memory	44.4 GB
Virtual Memory	47.5 GB

Table 4: Resources used by mapping to transcriptome with TopHat2

by *cufflinks*. Because the reads were mapped to transcriptome only, *cuffquant* could not use the merged transcript file and the original annotation file had to be used with the reads. The produced abundance files were then ran with *cuffnorm* which produced multiple files. The most interesting file (in this experimental pipeline) is "genes.fpkm_table" which, describing the normalized gene expression levels, can be used by bioinformaticians for further analysis.

5.2 Dog dataset experiment

The dog experiment reads consisted of 14 files in SRA format, roughly 71 GB in total size. SRA to FASTA format conversion resulted in 28 paired-end read files with a total size of roughly 346 GB. This data was used in the first experiment for this thesis' and therefore served a benchmarking role as the mapping data could not be use. The reason for that was wrong conversion from the SRA format (paired-end data was converted into single-end data). The benchmarking results are provided in Table 6. From the results we can see that each sample from the dog dataset took approximately 7.5 hours to map (in comparison, each human sample took approximately 4.5 hours to map). The main reason for slower mapping is the size of raw read files (the dog's raw read files were more than twice the size of the human's files).

Experiment	**Genome**	**Transcriptome**
SRR828796	85.1%	73.0%
SRR828797	84.9%	71.7%
SRR828798	85.2%	72.1%
SRR828799	86.2%	74.7%
SRR828800	86.5%	75.1%
SRR828801	84.9%	74.0%
SRR828802	85.2%	74.3%

Table 5: Overall mapping rates of experiments to genome and transcriptome with TopHat2

Resource type	**Resources used**
CPU time	roughly 6200 hours
Walltime	206 hours
Memory	N/A
Virtual Memory	N/A

Table 6: Resources used by TopHat in the dog experiment

6 Conclusion

The purpose of this thesis was to provide good knowledge on creating and running a typical RNA-seq data analyis pipeline. The author described general RNA-seq data analysis pipeline using specific tools; completed an example pipeline by using data from a public database and provided the basic results together with statistics; gave an analysis of integrated pipeline software and compared it to a customized pipeline setup; and explained the different files and computational methods used in a pipeline. The pipeline provided together with information about computational tools and input files covers the general workflow of RNA-seq data analysis and the information has great practical value. The experimental pipeline testing resulted in acceptable data considering the fact that the author had no control over the biological parts of RNA-seq. The conclusion from the analysis of an integrated pipeline setup iRAP is that until the implemented software is not fully released, it should not be used for heavy pipelines due to numerous downsides. It can be used for more simple pipelines because it creates feedback for the developers of the software and simple pipelines can be run in a customized pipeline in cases were the integrated pipeline does not suffice.

The information this thesis holds can be used in the future in the following ways.

1. Complete various RNA-seq data analysis pipelines.
2. Improve the provided pipeline even more due to active evolving of RNA-seq.
3. Develop a customized pipeline software that resolves the described challenges.
4. Develop testing tools for the data and tools in the pipeline.
5. Improve and upgrade the integrated pipeline software iRAP.

7 Acknowledgements

This thesis could not have been completed without the help of the author's supervisors. I would like to thank Hedi Peterson for helping me design the structure of this thesis and giving me feedback and Kersti Jääger for spending countless hours helping me improve this thesis and motivating me.

References

[1] Casey A Gifford, Michael J Ziller, Hongcang Gu, Cole Trapnell, Julie Donaghey, Alexander Tsankov, Alex K Shalek, David R Kelley, Alexander A Shishkin, Robbyn Issner, et al. Transcriptional and epigenetic dynamics during specification of human embryonic stem cells. *Cell*, 153(5):1149–1163, 2013.

[2] Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, 2009.

[3] Rodger B. Voelker, Clay Small, and William Cresko. Experimental details: Experimental design, May 2014. `http://rnaseq.uoregon.edu/index.html`.

[4] Simon Andrews. A quality control tool for high throughput sequence data, May 2014. `http://www.bioinformatics.babraham.ac.uk/projects/fastqc/`.

[5] Assaf Gordon. Fastq/a short-reads pre-processing tools, May 2014. `http://hannonlab.cshl.edu/fastx_toolkit/`.

[6] Sequence Read Archive. Sra toolkit, 2014 May. `http://eutils.ncbi.nih.gov/Traces/sra/sra.cgi?view=software`.

[7] Ensembl genome browser, 2014 May. `http://www.ensembl.org/index.html`.

[8] National Center for Biotechnology Information, 2014 May. `http://www.ncbi.nlm.nih.gov/`.

[9] Lincoln Stein. Generic feature format version 3 (gff3), Feb 2013. `http://www.sequenceontology.org/gff3.shtml`.

[10] Gtf2.2: A gene annotation format, May 2014. `http://mblab.wustl.edu/GTF22.html`.

[11] Richard Durbin and David Haussler. Gff: an exchange format for feature description, Sep 2000. `http://www.sanger.ac.uk/resources/software/gff/`.

[12] What is fasta format?, May 2014. `http://zhanglab.ccmb.med.umich.edu/FASTA/`.

[13] Tao Tao. Single letter codes for nucleotides, Mar 2012. `http://www.ncbi.nlm.nih.gov/staff/tao/tools/tool_lettercode.html`.

[14] Peter JA Cock, Christopher J Fields, Naohisa Goto, Michael L Heuer, and Peter M Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic acids research*, 38(6):1767–1771, 2010.

[15] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[16] UCSC Genome Informatics Group. Ucsc genome browser, 2014 May. `http://genome.ucsc.edu/`.

[17] Inc. DNAnexus. Sequence read archive +, May 2014. `http://sra.dnanexus.com/`.

[18] NCBI. The sequence read archive, May 2014. `http://www.ncbi.nlm.nih.gov/sra`.

[19] Ben Langmead, Cole Trapnell, Mihai Pop, Steven L Salzberg, et al. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[20] Cole Trapnell, Lior Pachter, and Steven L Salzberg. Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[21] Cole Trapnell, Adam Roberts, Loyal Goff, Geo Pertea, Daehwan Kim, David R Kelley, Harold Pimentel, Steven L Salzberg, John L Rinn, and Lior Pachter. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols*, 7(3):562–578, 2012.

[22] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE, 2000.

[23] Michael Burrows and David J Wheeler. A block-sorting lossless data compression algorithm. 1994.

[24] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[25] Ben Langmead. Bowtie 2: Fast and sensitive read alignment, May 2014. `http://bowtie-bio.sourceforge.net/bowtie2/`.

[26] Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. Tophat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol*, 14(4):R36, 2013.

[27] Cole Trapnell, Daehwan Kim, Geo Pertea, Harold Pimentel, Ryan Kelley, Lior Pachter, and Steven Salzberg. Tophat: A spliced read mapper for rna-seq, May 2014. `http://tophat.cbcb.umd.edu/`.

[28] Heng Li and R Durbin. Maq: Mapping and assembly with qualities. *Version 0.6*, 3, 2008.

[29] Cole Trapnell, Adam Roberts, Geo Pertea, David Hendrickson, Loyal Goff, Martin Sauvageau, Brian Williams, Ali Mortazavi, Gordon Kwan, Jeltje Baren, John Rinn, Steven Salzberg, Barbara Wold, and Lior Pachter. Cufflinks: Transcript assembly, differential expression, and differential regulation for rna-seq, May 2014. `http://cufflinks.cbcb.umd.edu/`.

[30] Loyal A. Goff, Cole Trapnell, and David Kelley. Cummerbund: Exploration, analysis and visualization of cufflinks high-throughput rna-seq data, May 2014. `http://compbio.mit.edu/cummeRbund/`.

[31] Robert Gentleman and Ross Ihaka. The r project for statistical computing, May 2014. `http://www.r-project.org/`.

[32] MathWorks. Matlab: The language of technical computing, May 2014. `http://www.mathworks.se/products/matlab/`.

[33] Nuno Fonseca. irap, 2014 May. `https://code.google.com/p/irap/`.

[34] Michael Johannes Ziller. Hepatoblast derived from hues64 rna-seq reps1and2, May 2013. `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM1124072`.

[35] Van Andel Institute. Canis lupus familiaris transcriptome or gene expression, Oct 2012. `http://sra.dnanexus.com/studies/SRP016141/experiments`.