Journal of Statistical Software

November 2018, Volume 87, Code Snippet 2. doi: 10.18637/jss.v087.c02

Rqc: A Bioconductor Package for Quality Control of High-Throughput Sequencing Data

Wélliton de Souza University of Campinas

Benilton de Sá Carvalho University of Campinas

Iscia Lopes-Cendes University of Campinas

Abstract

As sequencing costs drop with the constant improvements in the field, next-generation sequencing becomes one of the most used technologies in biological research. Sequencing technology allows the detailed characterization of events at the molecular level, including gene expression, genomic sequence and structural variants. Such experiments result in billions of sequenced nucleotides and each one of them is associated to a quality score. Several software tools allow the quality assessment of whole experiments. However, users need to switch between software environments to perform all steps of data analysis, adding an extra layer of complexity to the data analysis workflow.

We developed **Rqc**, a **Bioconductor** package designed to assist the analyst during assessment of high-throughput sequencing data quality. The package uses parallel computing strategies to optimize large data sets processing, regardless of the sequencing platform. We created new data quality visualization strategies by using established analytical procedures. That improves the ability of identifying patterns that may affect downstream procedures, including undesired sources technical variability. The software provides a framework for writing customized reports that integrates seamlessly to the R/**Bioconductor** environment, including publication-ready images. The package also offers an interactive tool to generate quality reports dynamically.

Rqc is implemented in R and it is freely available through the **Bioconductor** project (`https://bioconductor.org/packages/Rqc/`) for Windows, Linux and Mac OS X operating systems.

Keywords: next-generation sequencing, quality assessment, high-performance computing, R.

1. Introduction

Next-generation sequencing (NGS) has become the standard tool to investigate the association between molecular data and phenotypes of interest. This is the result of improvements on the

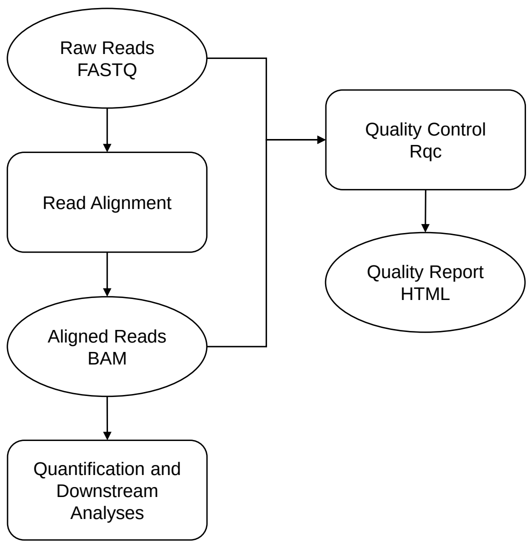

Figure 1: Workflow for data analysis using the **Rqc** package. The standard analytical pipeline processes the FASTQ files to map the reads to a reference. The resulting BAM files are later used for quantification and downstream analyses. Our software supports both FASTQ and BAM files as input and does not affect existing pipelines as it provides information that complements the workflow. The standard **Rqc** output is a self-contained HTML report.

technology and constant drops in costs. High-performance equipment sequences millions of short DNA fragments, also known as reads, yielding billions of nucleotides. During the base-calling process, the sequencer assigns a quality score to every base that comprises the read. This score indicates the degree of certainty that the equipment has correctly identified the nucleotide. Compared to previous approaches, NGS technologies produce bigger yields faster and cheaper. However, higher performance comes at a price: Larger amounts of data make it harder to identify failures that might have occurred during the sequencing run. Common problems are contamination with adapter sequences, drops of quality scores at specific cycles and redundant reads. These issues should be accounted for as early as possible, as they may affect downstream analyses negatively (Bravo and Irizarry 2010). Also, studies may use different sequencing platforms to generate data, which may add extra complexity to the quality assessment step (The 1000 Genomes Project Consortium 2012), requiring tools to be efficient and capable of processing large amounts of data regardless of the technology used for sequencing.

The analysis of high-throughput sequencing data is a procedure that requires careful quality assessment (QA), which can take place at different moments, as Figure 1 shows. The FASTQ file, obtained after base-calling, is comprised of sequenced fragments along with base-specific quality scores. This is the first opportunity for quality control, where researchers search for systematic deviations of quality using PHRED-scaled scores. The second occasion where additional quality control can take place is after mapping the reads to a reference. At this point, the FASTQ files are no longer the objects of interest and, instead, the researcher performs QA on BAM files, which result from the mapping strategy of choice. On both situations, the analyst gathers information to decide what procedures (e.g., removal of reads, trimming, clipping) should be applied to ensure that the data comply with the quality requirements of the study. Acting at these moments allows the researcher to obtain higher success rates in downstream procedures, as they may be affected by low quality reads or contaminants.

Another source of complexity during the analysis of NGS data is the integration of different tools used during the process. One common workflow to identify candidates for differential expression using RNA-Seq data includes: (A) QA using **FastQC** (Andrews 2016), which is implemented in Java and executed either from the command line or using a graphical user interface. (B) Manipulation of the reads via **Trimmomatic** (Bolger, Lohse, and Usadel 2014), implemented in Java and called from the command line. (C) Mapping through **TopHat** (Trapnell, Pachter, and Salzberg 2009), executed from the command line. (D) Feature counting via **HTSeq** (Anders, Pyl, and Huber 2015), implemented in Python and called from the command line. (E) Statistical modeling through **DESeq2** (Love, Huber, and Anders 2014), implemented in R (R Core Team 2018) and called from within R. This mix of environments (Java, Python, Perl, bash, R, etc.) is one of the origins of difficulties when analyzing high-throughput data sets and deserves special attention from developers.

We developed **Rqc** (Souza and Carvalho 2018) to address QA of NGS data using high-performance strategies to handle file formats that are agnostic to sequencing platforms (i.e., FASTQ and BAM files). Additionally, our solution uses the **Bioconductor** environment, which is known for delivering high-quality software that implements cutting-edge methodologies for analysis of biological data. Therefore, we provide the users with a quality control tool for NGS data that can be easily integrated to existing pipelines and also used to establish a **Bioconductor**-based workflow for the analysis of high-throughput data.

This paper describes the **Rqc** software. Section 2 describes how the **Rqc** package was developed. In Section 3, we show the usefulness of **Rqc** package in a QA of public NGS data. Section 4 describes how the **Rqc** package can help analyzing NGS data.

2. Implementation

The **Rqc** package is developed in R, using the **Bioconductor** framework to efficiently process FASTQ and BAM files and to simplify the delivery of a unified pipeline for NGS data analysis. Our software makes constant use of the **Bioconductor** three main concepts (Gentleman *et al.* 2004): transparency, through the development of a free and open-source software; efficiency, by using the infrastructure defined by core-maintained packages integrated into a single development ecosystem; and reproducibility, by ensuring that our software runs on any operating system supported by the R software and generates the same result.

Rqc uses the **ShortRead** (Morgan, Anders, Lawrence, Aboyoun, Pages, and Gentleman 2009) and **Rsamtools** (Morgan, Pagés, Obenchain, and Hayden 2016b) packages to extract information from raw and aligned data files respectively, allowing the use of different file formats. Our package supports both FASTQ and BAM file formats. By default, the software processes a random sample of records from each input file to improve execution time. The user can adjust the size of the chunk: larger sizes increase the statistical power, generating more reliable results; smaller values allow for better control of computational resources. The user can also choose to process the whole file, which **Rqc** does without compromising the available computing resources.

We use the **BiocParallel** (Morgan, Obenchain, Lang, and Thompson 2016a) package, the main **Bioconductor**'s backend for high-performance computing, to process multiple files in parallel.

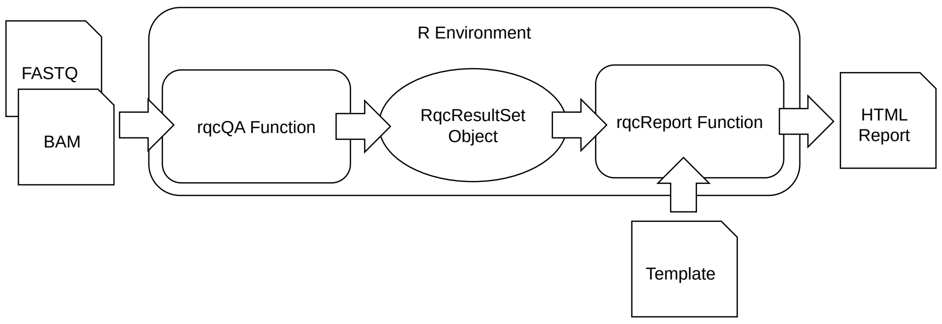

Figure 2: **Rqc** uses the R statistical environment to create reports for quality assessment. The `rqcQA` method processes the input data and returns a list of '`RqcResultSet`' objects, which contains quality-related statistics used to create the final report. The `rqcReport` method uses the result list combined with a template file, which can be customized by the user, to produce the HTML report.

The parallel processing takes place on single thread or multiple cores, depending on the user's setup. Using this feature, execution time can be significantly reduced. **Rqc** performs parallel processing of files automatically, setting the number of processing units according to the available computational resources. We made the choice for automatic configuration to deploy a software that is both efficient and simple to use. However, the user has full control of the parallel processing settings, which can be customized when needed.

Given the input files, as shown by Figure 2, **Rqc** depends on the execution of the `rqcQA` function to produce the table containing the summary statistics required to create the images for the final report. These statistics are stored as tidy data (Wickham 2014) using a list-like data structure called '`RqcResultSet`'. The list containing the result data is processed through the `rqcReport` method, which combines high-quality images produced via the **ggplot2** (Wickham 2009) package with R Markdown template files to generate an HTML report using the **knitr** (Xie 2018) package.

3. Results

We developed **Rqc**, an optimized **Bioconductor** package to assist the analyst during quality control steps for high-throughput sequencing data. It uses parallel computing strategies to process multiple files efficiently. The package handles both FASTQ and BAM files, allowing the user to assess the quality of the data at different stages of the analysis. Sequencing data are summarized into frequency tables that are used later to generate charts, tables and other statistics.

To demonstrate the operation of the **Rqc** package we used a set of public data of a study on RNA sequencing (RNA-seq), which is part of the 1000 Genomes Project (The 1000 Genomes Project Consortium 2012). We used 5 paired-end samples of each population group available in the public data set resulting in 50 FASTQ files. These files are used as input for the `rqcQA` function, which was configured to process 4 files in parallel. To reproduce this demonstration, R version equal or greater than 3.4.2, **Bioconductor** 3.6 and **Rqc** 1.12.0 are required. The following code shows how to install the **Rqc** package.

```
R> source("https://bioconductor.org/biocLite.R")
R> biocLite("Rqc")
```

Run `source("http://bioconductor.org/biocLite.R")` (with `http://` instead of `https://`) if HTTPS URLs are not supported. After installation, we load the package and perform data analysis.

```
R> library("Rqc")
R> data <- read.delim2("data.txt", stringsAsFactors = FALSE)
R> files <- file.path("data", data$filename)
R> pair <- rep(1:25, each = 2)
R> group <- factor(data$group)
R> checkpoint("qa", path = ".", {
+    qa <- rqcQA(files, pair = pair, group = group, workers = 4)
+  }, keep = "qa")
R> qa[1]

$ERR188040_1.fastq.gz
class: RqcResultSet(3)
QA elements (access with qa[["elt"]]):
  perFile: list(2)
    information: data.frame(1 7)
    topReads: data.frame(10 2)
  perCycle: list(2)
    quality: data.frame(3150 4)
    baseCall: data.frame(375 3)
  perRead: list(3)
    width: data.frame(1 2)
    averageQuality: data.frame(2507 2)
    frequency: data.frame(94 2)
```

3.1. ‘RqcResultSet’ data structure

The ‘RqcResultSet’ class stores summarized data from one file that was previously processed. As **Rqc** package may process multiple files at the same time, a list of ‘RqcResultSet’ objects is returned. This result list is used as input by many methods with different tasks such as computation of statistics used in plots, chart and report generation. Table 1 presents the available methods for accessing a list of ‘RqcResultSet’ objects. These methods are also used by other **Rqc** functions to extract data and produce additional statistics required by different visualization strategies. Table 2 shows general information about processed files.

```
R> library("xtable")
R> fileInformation <- perFileInformation(qa)
R> fileInformation$path <- NULL
R> colnames(fileInformation) <- c("File name", "Pair", "Format",
+    "Group", "Reads", "Total reads")
```

Method	Description
`perFileInformation`	File name, base directory, number of sampled reads, total reads.
`perFileTopReads`	Most represented sequencing reads and their counts.
`perReadWidth`	Frequency distribution of read width.
`perReadFrequency`	Number of unique reads, duplicated reads, etc.
`perReadQuality`	Frequency distribution of mean quality of reads.
`perCycleQuality`	Frequency distribution of cycle-specific quality.
`perCycleBasecall`	Frequency distribution of cycle-specific base call.

Table 1: Accessor methods for the '`RqcResultSet`' class.

File name	Pair	Format	Group	Reads	Total reads
`ERR188040_1.fastq.gz`	1	FASTQ	GBR	1000000	27256165
`ERR188040_2.fastq.gz`	1	FASTQ	GBR	1000000	27256165
`ERR188190_1.fastq.gz`	6	FASTQ	FIN	1000000	14760621
`ERR188190_2.fastq.gz`	6	FASTQ	FIN	1000000	14760621
`ERR188325_1.fastq.gz`	11	FASTQ	CEU	1000000	22136382
`ERR188325_2.fastq.gz`	11	FASTQ	CEU	1000000	22136382
`ERR188214_1.fastq.gz`	16	FASTQ	YRI	1000000	47079579
`ERR188214_2.fastq.gz`	16	FASTQ	YRI	1000000	47079579
`ERR188380_1.fastq.gz`	21	FASTQ	TSI	1000000	23146161
`ERR188380_2.fastq.gz`	21	FASTQ	TSI	1000000	23146161

Table 2: Information of FASTQ files.

```
R> index <- c(1, 2, 11, 12, 21, 22, 31, 32, 41, 42)
R> tab2 <- xtable(fileInformation[index, ], label = "tabinfo",
+    "Information of FASTQ files.", table.placement ="")
R> print(tab2, include.rownames = FALSE)
```

3.2. Available plots

The **Rqc** package provides a number of plots to the user. The overall objective of these graphical summaries is to allow the analyst to make an informed decision regarding the quality of the data under inspection. All **Rqc** methods that generate graphs return an object of class '`ggplot`'. These objects can be combined with graphical elements provided by other **ggplot2** functions, such as themes and color palettes.

`rqcReadQualityBoxPlot` generates a graphic chart of per file average read quality distribution that provides an overview of the files (Figure 3). From this chart we can check whether all files have similar qualities. In situations where the DNA samples are sequenced by different research groups, or over several days, there may be quality changes of the fragments through this graph.

Working with many files may complicate the visualization of some graphic charts created by **Rqc**. The package provides methods for subsetting a list of '`RqcResultSet`' objects to address this issue. For example, we could select the result data only from CEU samples or we can also subset by pair of files.

```
R> rqcReadQualityBoxPlot(qa[index]) + scale_fill_brewer(palette = "Set1")
```

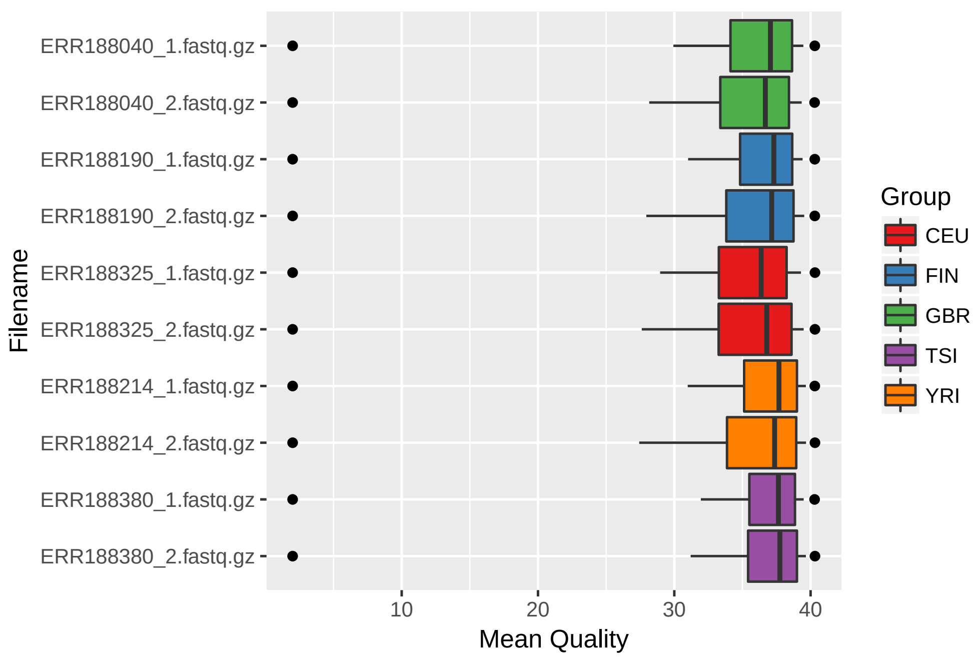

Figure 3: Per file average read average quality plot. Dots mean minimum and maximum average quality. Box plots are colored by sample group. For this data set, all FASTQ files have more than 75% of the reads with average quality above 30 in PHRED scale.

```
R> qa.ceu <- subsetByGroup(qa, "CEU")
R> qa.pair1 <- subsetByPair(qa, 1)
```

One new visualization scheme that we implemented in **Rqc** allows the user to easily determine what percentage of the reads exceed a given threshold of average quality. We can use this information as an indicator of success of a sequencing experiment. For example, a sample that presents 10% of the reads exceeding the Q20 threshold (i.e., 10% of the reads have PHRED-scaled average quality of at least 20) will likely be removed from the analysis or even re-sequenced, if possible. A graphic chart with this information can be obtained using the `rqcReadQualityPlot` method. Figure 4 presents the average quality pattern by showing on the x-axis quality thresholds and on the y-axis the percentage of reads that exceed that quality level; we use this chart as an indicator of the need for re-sequencing samples according to the minimum average quality.

The biplot is generated by performing a principal component analysis (PCA) on a matrix containing the sample-specific average quality scores per cycle. Using this approach, we can easily investigate differences of patterns of quality at different sequencing cycles. The user can generate such plot using the method `rqcCycleAverageQualityPcaPlot` (Figure 5).

The analyst can investigate the average quality per cycle by using box plots or line plots. This strategy allows for the user to pinpoint cycles that show specific behaviors, like sudden drop in quality. `rqcCycleAverageQualityPlot` generates a graphic chart of cycle-specific average quality (Figure 6).

```
R> rqcReadQualityPlot(qa.pair1) + scale_color_brewer(palette = "Set1")
```

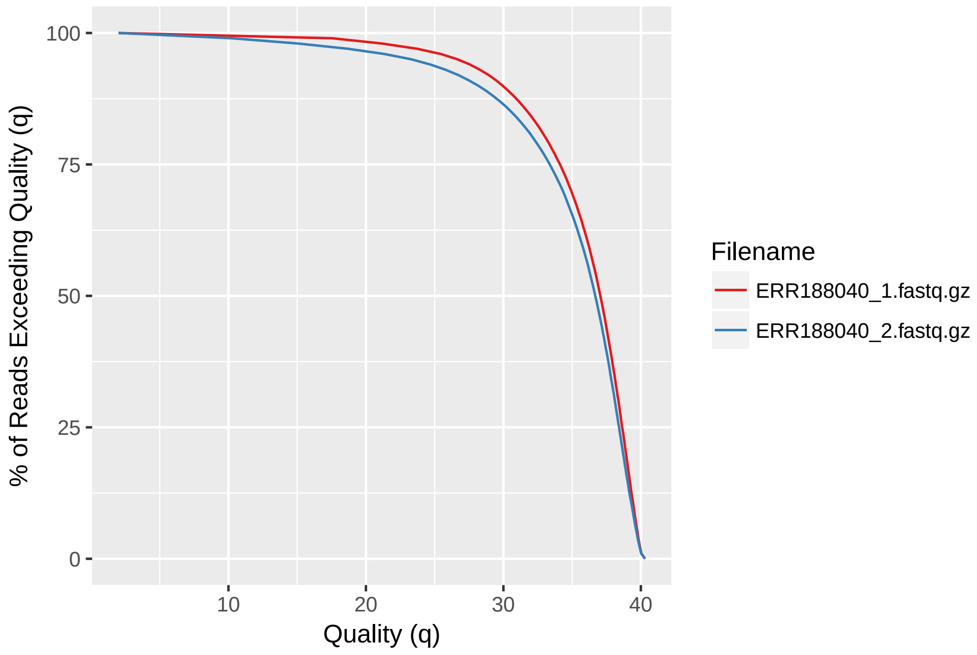

Figure 4: Survival curve for reads that exceed different quality thresholds.

```
R> rqcCycleAverageQualityPcaPlot(qa.pair1) + theme_bw()
```

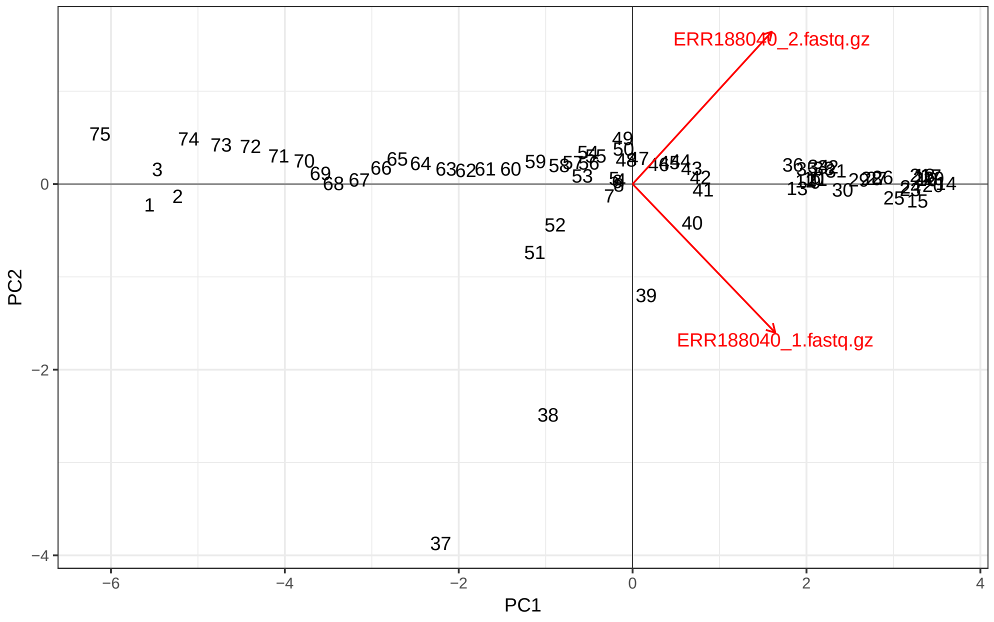

Figure 5: Biplot from PCA of cycle-specific read average quality. There are some discrepancies in the quality of the files of the same sample.

```
R> rqcCycleAverageQualityPlot(qa.pair1) + scale_color_brewer(palette = "Set1")
```

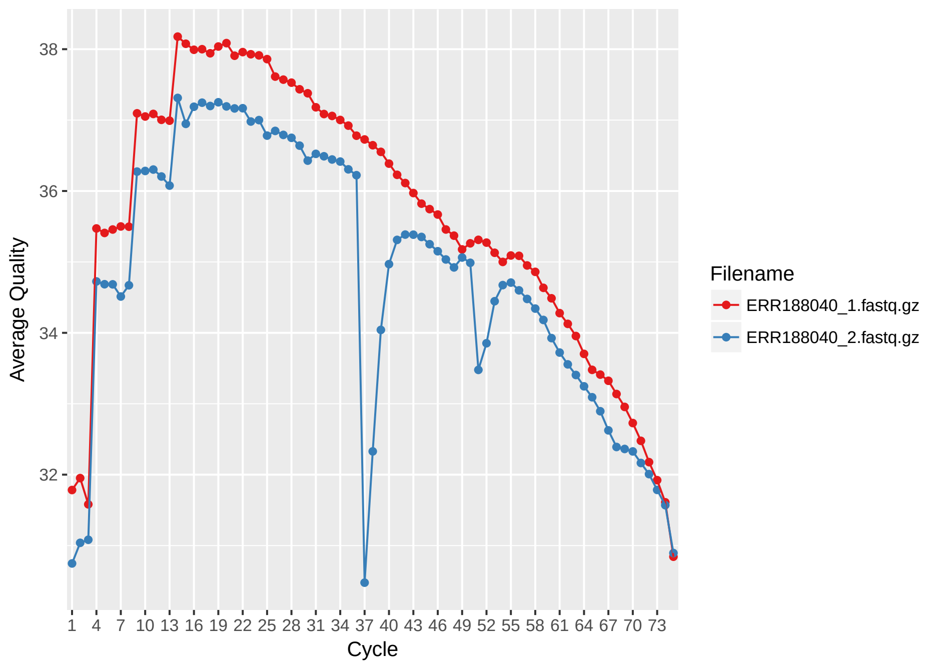

Figure 6: Average quality score per sequencing cycle. It is possible to see differences between these two files of the same sample, which proves the results shown by Figure 5.

```
R> rqcCycleQualityPlot(qa.pair1)
```

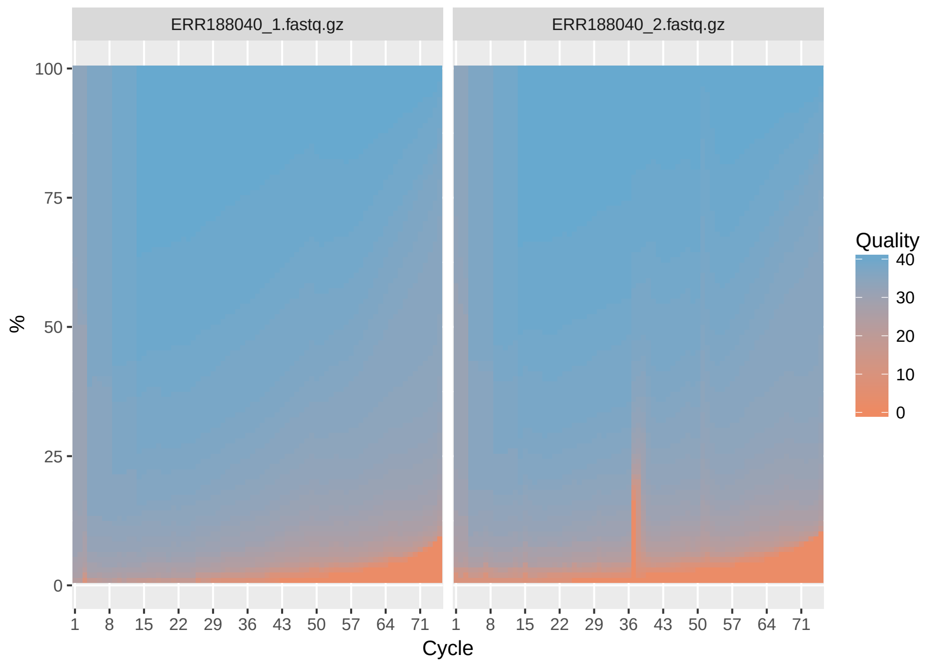

Figure 7: Cycle-specific quality distribution plot.

```
R> rqcFileHeatmap(qa[[1]])
```

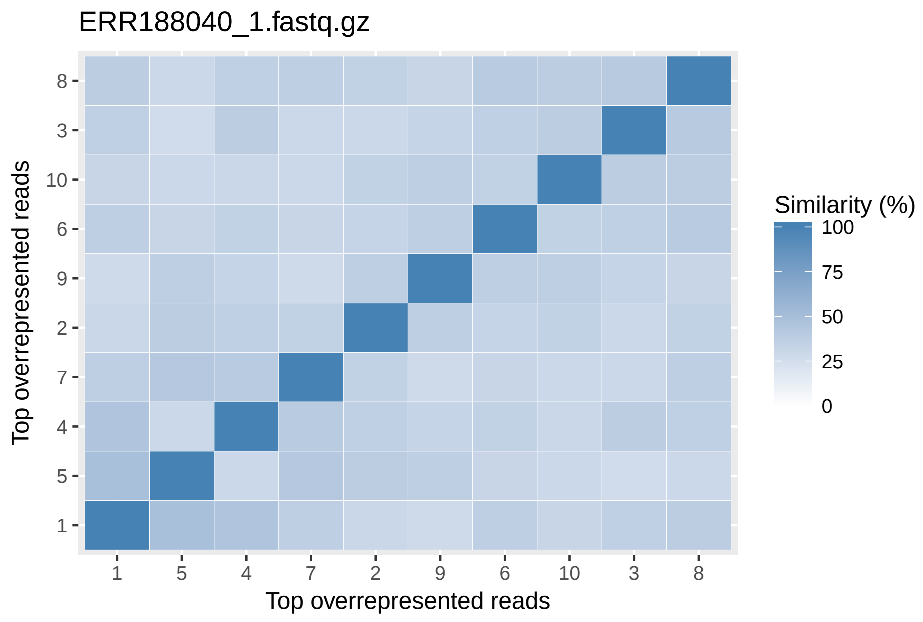

Figure 8: The heatmap shows the similarity between the most frequent reads observed for the sample in question; this information can be used, for example, to investigate the presence of contaminants on the data.

`rqcCycleQualityPlot` shows the proportion of quality calls per cycle. Colors are presented in a gradient red-blue, where red identifies lower quality. This visualization provides a fast overview of qualities of the files (Figure 7).

Another addition by the **Rqc** package is the ability of generating a heatmap of the similarity between the most common reads observed at a given sample. In our experience, this is of great importance when assessing whether or not different reads may actually be originated from the same fragment, but one of them has some contamination by adapters. This chart can be generated with the `rqcFileHeatmap` method, which allows for setting the number of most frequent reads to be used. On Figure 8, we present the similarity heatmap, which can assist on the identification of reads that are likely to be measuring the same target.

The analysis of nucleotide composition per cycle is essential to identify biases like those caused by Illumina primers on transcriptome sequencing (Hansen, Brenner, and Dudoit 2010). `rqcCycleBaseCallsPlot` generates a stacked bar plot that describes the proportion of each nucleotide called for every cycle of sequencing (Figure 9). `rqcCycleBaseCallsLinePlot` shows the same result using lines instead of stacked bars.

3.3. Customized and interactive reports

The `rqcQA` and `rqcReport` functions are closely related, as the former generates the input required by the latter. We chose to have these two different functions to allow the user to use the **Rqc** results in other situations, like using a different engine to create graphics. The `rqcQA` function requires the input files (FASTQ or BAM) and returns a list of '`RqcResultSet`'

```
R> rqcCycleBaseCallsLinePlot(qa[[1]])
```

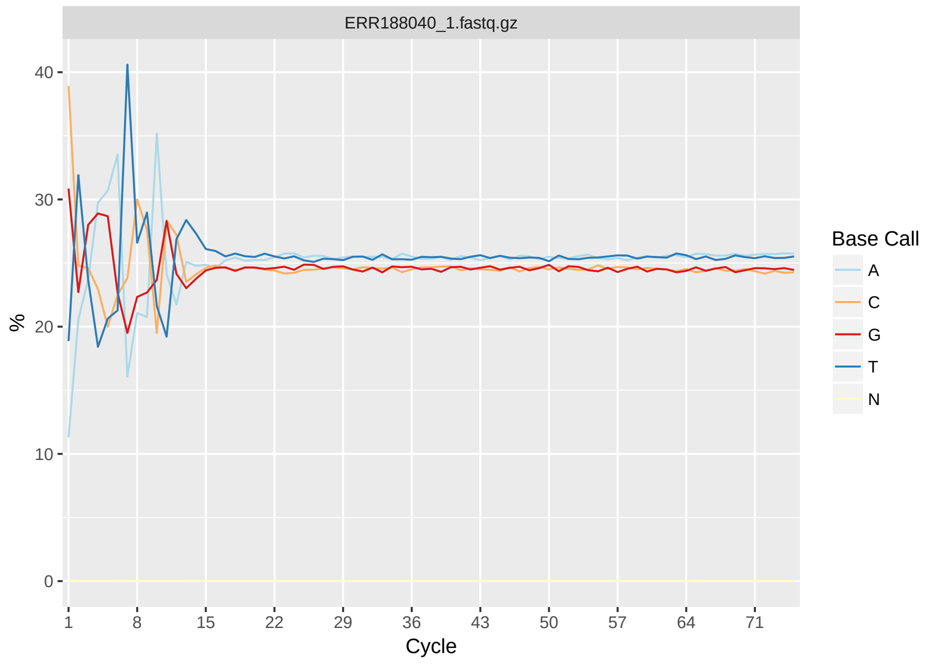

Figure 9: Cycle-specific base call proportion.

objects, which is a table containing the summary statistics needed for the quality report. This table and a template file are the input for the `rqcReport`, which creates the HTML quality report.

```
R> rqcReport(qa, outdir = ".", file = "qa_report")

[1] "/home/welliton/git/rqcpaper/Code/qa_report.html"
```

Users can compose R Markdown files, which contain texts and chunks of R code. These files should be passed as arguments to the `rqcReport` method. Chunks of R code are executed and their results are captured and merged with pre-existing text, generating a self-contained HTML report that contains text, tables and figures.

The user can also choose to use interactive tools to generate quality reports. We provide the `rqcShinyReport` method, which uses the **shiny** package (Chang, Cheng, Allaire, Xie, and McPherson 2018) to deliver a graphical interface where the user can select samples, groups of samples and plots of interest. This service is deployed as a web server and all computations happen in real-time.

```
R> rqcShinyReport(qa)
```

4. Conclusion

We developed the **Rqc Bioconductor** package to provide a simple and effective strategy for reporting information about the quality high-throughput sequencing data sets. In addition, it can provide a deeper view of data quality by reading entire files without degrading the system performance and allowing the data analyst to identify biases that would be undetectable if only a subset of the data was analyzed. **Rqc** builds a self-contained report with high-resolution graphics that can be used directly in publications or shared with others. It also implements an interactive web application that simplifies the analysis of data sets, as it does not require coding. **Rqc** is completely integrated with and deployed through **Bioconductor**, simplifying its incorporation in **Bioconductor**-based pipelines. All of the software dependencies are transparently resolved by the R/**Bioconductor** package management system. The **Rqc** package is technology-independent and cross-platform, providing the user with the same experience on the three major operating systems (MS Windows, Mac OS X and Linux).

Computational details

- R version 3.4.2 (2017-09-28), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 14.04.5 LTS`
- Matrix products: default
- BLAS: `/usr/lib/libblas/libblas.so.3.0`
- LAPACK: `/usr/lib/lapack/liblapack.so.3.0`
- Base packages: **base**, **datasets**, **graphics**, **grDevices**, **methods**, **parallel**, **stats**, **stats4**, **utils**
- Other packages: **Biobase** 2.38.0, **BiocGenerics** 0.24.0, **BiocParallel** 1.12.0, **BiocStyle** 2.6.0, **Biostrings** 2.46.0, **DelayedArray** 0.4.0, **GenomeInfoDb** 1.14.0, **GenomicAlignments** 1.14.0, **GenomicRanges** 1.30.0, **ggplot2** 2.2.1, **IRanges** 2.12.0, **matrixStats** 0.52.2, **Rqc** 1.12.0, **Rsamtools** 1.30.0, **S4Vectors** 0.16.0, **ShortRead** 1.36.0, **SummarizedExperiment** 1.8.0, **xtable** 1.8-2, **XVector** 0.18.0
- Loaded via a namespace (and not attached): **acepack** 1.4.1, **AnnotationDbi** 1.40.0, **AnnotationFilter** 1.2.0, **AnnotationHub** 2.10.0, **assertthat** 0.2.0, **backports** 1.1.1, **base64enc** 0.1-3, **BiocInstaller** 1.28.0, **biomaRt** 2.34.0, **biovizBase** 1.26.0, **bit** 1.1-12, **bit64** 0.9-7, **bitops** 1.0-6, **blob** 1.1.0, **BSgenome** 1.46.0, **checkmate** 1.8.5, **cluster** 2.0.6, **colorspace** 1.3-2, **compiler** 3.4.2, **curl** 3.0, **data.table** 1.10.4-3, **DBI** 0.7, **dichromat** 2.0-0, **digest** 0.6.12, **ensembldb** 2.2.0, **evaluate** 0.10.1, **foreign** 0.8-69, **Formula** 1.2-2, **GenomeInfoDbData** 0.99.1, **GenomicFeatures** 1.30.0, **GenomicFiles** 1.14.0, **grid** 3.4.2, **gridExtra** 2.3, **gtable** 0.2.0, **highr** 0.6, **Hmisc** 4.0-3, **htmlTable** 1.9, **htmltools** 0.3.6, **htmlwidgets** 0.9, **httpuv** 1.3.5, **httr** 1.3.1, **hwriter** 1.3.2, **interactiveDisplayBase** 1.16.0, **knitr** 1.17, **labeling** 0.3, **lattice** 0.20-35, **latticeExtra** 0.6-28, **lazyeval** 0.2.1, **magrittr** 1.5, **markdown** 0.8, **Matrix** 1.2-11,

memoise 1.1.0, **mime** 0.5, **munsell** 0.4.3, **nnet** 7.3-12, **plyr** 1.8.4, **prettyunits** 1.0.2, **progress** 1.1.2, **ProtGenerics** 1.10.0, **R6** 2.2.2, **RColorBrewer** 1.1-2, **Rcpp** 0.12.13, **RCurl** 1.95-4.8, **reshape2** 1.4.2, **rlang** 0.1.2, **rmarkdown** 1.6, **RMySQL** 0.10.13, **rpart** 4.1-11, **rprojroot** 1.2, **RSQLite** 2.0, **rtracklayer** 1.38.0, **scales** 0.5.0, **shiny** 1.0.5, **splines** 3.4.2, **stringi** 1.1.5, **stringr** 1.2.0, **survival** 2.41-3, **tibble** 1.3.4, **tools** 3.4.2, **VariantAnnotation** 1.24.0, **XML** 3.98-1.9, **yaml** 2.1.14, **zlibbioc** 1.24.0

Acknowledgments

This work was supported by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). W.S. and B.S.C. are recipients of FAPESP fellowships. W.S. received a masters degree fellowship (Proc. 2013/24801-2). I.L-C. is also supported by the Conselho Nacional de Pesquisa (CNPq), Brazil.

References

Anders S, Pyl PT, Huber W (2015). "**HTSeq** – A Python Framework to Work with High-Throughput Sequencing Data." *Bioinformatics*, **31**(2), 166–169. doi:10.1101/002824.

Andrews S (2016). ***FastQC**: A Quality Control Tool for High Throughput Sequence Data*. URL http://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

Bolger AM, Lohse M, Usadel B (2014). "**Trimmomatic**: A Flexible Trimmer for Illumina Sequence Data." *Bioinformatics*, **30**(15), 2114–2120. doi:10.1093/bioinformatics/btu170.

Bravo HC, Irizarry RA (2010). "Model-Based Quality Assessment and Base-Calling for Second-Generation Sequencing Data." *Biometrics*, **66**(3), 665–674. doi:10.1111/j.1541-0420.2009.01353.x.

Chang W, Cheng J, Allaire JJ, Xie Y, McPherson J (2018). ***shiny**: Web Application Framework for R*. R package version 1.1.0, URL https://CRAN.R-project.org/package=shiny.

Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry RA, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JYH, Zhang J (2004). "**Bioconductor**: Open Software Development for Computational Biology and Bioinformatics." *Genome Biology*, **5**(10), R80. doi:10.1186/gb-2004-5-10-r80.

Hansen KD, Brenner SE, Dudoit S (2010). "Biases in Illumina Transcriptome Sequencing Caused by Random Hexamer Priming." *Nucleic Acids Research*, **38**(12), 1–7. doi:10.1093/nar/gkq224.

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. doi:10.1186/s13059-014-0550-8.

Morgan M, Anders S, Lawrence M, Aboyoun P, Pages H, Gentleman R (2009). "**ShortRead**: A **Bioconductor** Package for Input, Quality Assessment and Exploration of High-Throughput Sequence Data." *Bioinformatics*, **25**(19), 2607–2608. doi:10.1093/bioinformatics/btp450.

Morgan M, Obenchain V, Lang M, Thompson R (2016a). ***BiocParallel**: **Bioconductor** Facilities for Parallel Evaluation.* URL https://bioconductor.org/packages/BiocParallel/.

Morgan M, Pagés H, Obenchain V, Hayden N (2016b). ***Rsamtools**: Binary Alignment (BAM), FASTA, Variant Call (BCF), and Tabix File Import.* URL https://bioconductor.org/packages/Rsamtools/.

R Core Team (2018). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Souza W, Carvalho B (2018). ***Rqc**: Quality Control Tool for High-Throughput Sequencing Data.* R package version 1.16.1, URL https://bioconductor.org/packages/Rqc/.

The 1000 Genomes Project Consortium (2012). "An Integrated Map of Genetic Variation from 1,092 Human Genomes." *Nature*, **491**(7422), 56–65. doi:10.1038/nature11632.

Trapnell C, Pachter L, Salzberg SL (2009). "**TopHat**: Discovering Splice Junctions with RNA-Seq." *Bioinformatics*, **25**(9), 1105–1111. doi:10.1093/bioinformatics/btp120.

Wickham H (2009). ***ggplot2**: Elegant Graphics for Data Analysis.* Springer-Verlag.

Wickham H (2014). "Tidy Data." *Journal of Statistical Software*, **59**(10), 1–23. doi:10.18637/jss.v059.i10.

Xie Y (2018). ***knitr**: A General-Purpose Package for Dynamic Report Generation in R.* R package version 1.20, URL https://CRAN.R-project.org/package=knitr.

Affiliation:

Iscia Lopes-Cendes
Department of Medical Genetics
School of Medical Sciences and the Brazilian Institute of Neuroscience and Neurotechnology (BRAINN)
University of Campinas
E-mail: icendes@unicamp.br
URL: http://bcblab.org/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
November 2018, Volume 87, Code Snippet 2 *Submitted:* 2016-07-07
doi:10.18637/jss.v087.c02 *Accepted:* 2017-11-30