with

Reproducible Research

User Manual*

Francesco Russo and Claudia Angelini
CNR-IAC, Naples
July 23, 2014

RNASeqGUI_0.99.2

---

*This work was supported by the Italian Flagship InterOmics Project (PB.P05) and by BMBS COST Action BM1006.
to Luisa
Contents

1 Introduction 5
   1.1 Overview of RNASeqGUI R package 5
   1.2 Other GUIs for RNASeq data analysis 5
   1.3 Scope and availability 6

2 RGTK2 installation guide 8
   2.1 For Linux users 8
   2.2 For MacOS users 8
   2.3 For Windows users 9

3 Installation of R and the required R-packages 10

4 Quick start 15

5 What’s new 16

6 Structure of RNASeqGUI main interface 18

7 How to create a new project or select an existing one 22

8 BAM EXPLORATION SECTION 24
   8.1 Bam Exploration Interface 24

9 COUNT SECTION 28
   9.1 Read Count Interface 28

10 PRE-ANALYSIS SECTION 31
   10.1 Data Exploration Interface 31
   10.2 Normalization Interface 34

11 DATA ANALYSIS SECTION 36
   11.1 Data Analysis Interface 36
   11.2 EdgeR 36
   11.3 EdgeR Multi Factor / Complex Design 39
   11.4 DESeq 36
   11.5 DESeq Multi Factor / Complex Design 41
   11.6 DESeq2 40
   11.7 DESeq2 Multi Factor / Complex Design 43
   11.8 NoiSeq 46
   11.9 BaySeq 47
12 POST ANALYSIS SECTION 50
12.1 Result Inspection Interface ............................... 50
12.2 Result Comparison Interface ............................. 52

13 REPORT AND UTILITY SECTION 53
13.1 Reproducible Research: the Log Files ................. 53
13.2 Utility Interface ............................................. 55

14 Usage Example 56
14.1 Data Preparation ............................................. 56
14.2 Usage of RNASeqGUI ...................................... 57

15 How to customize RNASeqGUI 71
15.1 Adding a new button in just three steps ............... 71

16 Technical Details 73

17 Errors/Warning/Bugs 74
17.1 Read Count Interface Errors ......................... 74
17.1.1 Error in data.frame(...) ......................... 74
17.1.2 Warning messages: In .deduceExonRankings(exs... 74

Acknowledgement 75
1 Introduction

1.1 Overview of RNASEqGUI R package

This manual describes RNASEqGUI R package that is a graphical user interface for the identification of differentially expressed genes from RNA-Seq experiments. R (http://cran.r-project.org/) is an open source object oriented language for statistical computing and graphics. RNASEqGUI package includes several well known RNA-Seq tools, available as command line in www.bioconductor.org. RNASEqGUI main interface is divided into six sections. Each section is dedicated to a particular step of the data analysis process. The first section covers the exploration of the bam files. The second concerns the counting process of the mapped reads against a gene annotation file (GTF). The third focuses on the exploration of count-data and on data preprocessing, including the normalization procedures. The fourth is about the identification of the differentially expressed genes that can be performed by several methods, such as: EdgeR, EdgeRComplexDesign, DESeq, DESeqEdgeRComplexDesign, DESeq2, DESeq2EdgeRComplexDesign, NoiSeq, BaySeq. The fifth section regards the inspection of the results produced by these methods and the quantitative comparison among them. Finally, in the spirit of Reproducible Research in the sixth section we find the Log File button that the user can click to generate the report in html format of all steps performed during the analysis of a specific project. The report is produced in R markdown format via knitr library and they include the documentation of the methods used and the R code that has been executed during the RNASEqGUI usage. Moreover, results can be viewed and explored on a web browser thanks to ReportingTools [Huntley et al., 2013] library that allows the user to navigate through them.

1.2 Other GUIs for RNASeq data analysis

This package was implemented following and expanding the idea presented in [Villa-Vialaneix et al., 2013] and in http://tuxette.nathalievilla.org/?p=866&lang=en. The idea of RNASEqGUI is similar to that one presented in [Wettenhall et al., 2004].
Sanges et al., 2007, Lohse et al., 2012, Pramana et al., 2013, Wettenhall et al., 2006, Angelini et al., 2008] with specific attention on RNA-Seq data analysis. Moreover, RNASeqGUI is designed to facilitate RNA-seq work-flow analysis (via its organization in several different sections and interfaces and via the inclusions of numerous concise and clear vignettes) and also to facilitate the extensibility of the GUI (via its software development organization that facilitate the task of expanding and redesign its interfaces). In fact, it is extremely easy to add new buttons that calls new functionalities. Therefore, a user can customize RNASeqGUI interfaces for his own purposes and benefits by adding the methods he needs mostly (for more details see Section 15 How to customize RNASeqGUI: Adding a new button in just three steps). Hence, we think that RNASeqGUI represents a useful and valid alternative to other existing GUIs.

1.3 Scope and availability

RNASeqGUI is an R package designed for the identification of differentially expressed genes across multiple biological conditions. This software is not just a collection of some known methods and functions, but it is designed to guide the user during the entire analysis process. Moreover, the GUI is also helpful for those who are expert R-users since it speeds up the usage of the included RNA-Seq methods drastically. Current implementation allows to handle the simple experimental design where the interest is on the experimental condition, future work will cover complex designs.

RNASeqGUI is freely available at (see Figure 1):

http://bioinfo.na.iac.cnr.it/RNASeqGUI/Download
Figure 1: The [http://bioinfo.na.iac.cnr.it/RNASeqGUI](http://bioinfo.na.iac.cnr.it/RNASeqGUI) web page
2 RGTK2 installation guide

RNASeqGUI package requires the RGTK2 graphical library [Lawrence et al., 2010] to run. The installation process consists in two steps. The first depends on the operating system (devoted to installation the GTK+ 2.0, an open-source GUI tool written in C). The second regards the required R packages.

2.1 For Linux users

We tested RNASeqGUI on Ubuntu 12.04 (precise) 64-bit, Kernel Linux 3.2.0-37-generic, GNOME 3.4.2.

1 - Open a terminal and type:

```sh
sudo apt-get update
dsudo apt-get install libgtk2.0-dev
```

2 - Type:

```sh
sudo apt-get install libcurl4-gnutls-dev
```

3 - Type:

```sh
sudo apt-get install libxml2-dev
```

4 - Then, go to Section 3.

2.2 For MacOS users

1 - Install Xcode developer tools (at least version 5.0.1) from Apple Store (it is free).

2 - Install XQuartz-2.7.5.dmg from http://xquartz.macosforge.org/landing/

3 - Install GTK\textsubscript{2}.24.17-X11.pkg from http://r.research.att.com

**WARNING:** Please, install the binary version GTK\textsubscript{2}.24.17-X11.pkg for Mac OS 10.6 Snow Leopard even though you have Mac OS 10.9 Mavericks.

4 - Then, go to Section 3.
2.3 For Windows users


2 - This is a bundle containing the GTK+ stack and its dependencies for Windows. To use it, create some empty folder like C:\opt\gtk.

3 - Unzip this bundle.

4 - Now, you have to add the bin folder to your PATH variable. Make sure you have no other versions of GTK+ in PATH variable. To do this, execute the following instructions: Open Control Panel, click on System and Security, click on System, click on Advanced System Settings, click on Environment Variables. In the Environment Variables window you will notice two columns User variables for a user name and System variables. Change the PATH variable in the System variables to be C:\opt\gtk\bin.

5 - Then, go to Section 3.
3 Installation of R and the required R-packages

1 - For Linux and MacOS, install R version 3.1.0 (2014-04-10) "Spring Dance" from http://cran.r-project.org/ according to your operating system.

For Windows, install R version 3.0.3 from http://cran.r-project.org/ since Rsubread package does not work on Windows.

2 - Download RNASeqGUI package from http://bioinfo.na.iac.cnr.it/RNASeqGUI/Download

For Windows operating system, download the zip binary file. For MacOS and Linux download the tar.gz file.

- For Windows users: select “Install packages(s) from local zip files”, under the “Packages” pull-down menu, as in the Figure 2
Figure 3: Under “Package and Data” pull-down menu, select “Package Installer”. From http://outmodedbonsai.sourceforge.net/InstallingLocalRPackages.html

Figure 4: In the “Package Installer”, pull down the top-left menu, select “Local Source Package” and navigate to where you have downloaded the source package.
• For MacOS users: under “Package and Data” pull-down menu, select “Package Installer”, see Figure 4.
In the “Package Installer”, pull down the top-left menu, select “Local Source Package” and navigate to where you have downloaded the source package, see Figure 4.

• For Linux users: open a shell and go to the directory containing the package tree and type the command

```
sudo R CMD INSTALL -l /path/to/library RNASEqGUI
```

3 - Finally, if the libraries required by RNASEqGUI are not automatically downloaded and installed, we suggest the user to install all the packages that are needed to run RNASEqGUI package before loading it. Open R and type (the order of the list below is important):

**For MacOS**: go to [http://cran.r-project.org/web/packages/RGtk2/index.html](http://cran.r-project.org/web/packages/RGtk2/index.html) and choose the binary version for OS X Snow Leopard binaries: r-release: RGtk2_2.20.29.tgz. Then, in the “Package Installer”, pull down the top-left menu and select “Local Binary Package”.

```
install.packages("e1071")
install.packages("ineq")
install.packages("RGtk2")
install.packages("RCurl")
install.packages("digest")
install.packages("ggplot2")
install.packages("RCOLORBrewer")
install.packages("VennDiagram")
install.packages("XML")
install.packages("tcltk")
install.packages("knitr")
```

3 - Type (the order of the list below is important):

```
source("http://bioconductor.org/biocLite.R")
biocLite("biomaRt")
biocLite("DEXSeq")
biocLite("pasilla")
```
Once the installation is complete, please, check that all the packages listed above have been installed correctly. To see this, copy and paste the following list into R to see whether there are errors coming out.

```r
library(e1071)
library(ineq)
library(RGtk2)
library(RCurl)
library(digest)
library(ggplot2)
library(RColorBrewer)
library(VennDiagram)
library(XML)
library(tcltk)
library(knitr)
library(biomaRt)
library(DEXSeq)
library(pasilla)
library(GenomicRanges)
library(GenomicFeatures)
library(Rsamtools)
library(edgeR)
```
library(baySeq)
library(NOISEq)
library(DESeq)
library(DESeq2)
library(gplots)
library(EDASeq)
library(leeBamViews)
library(preprocessCore)
library(scatterplot3d)
library(BiocParallel)
library(digest)
library(Rsubread)
library(biomaRt)
library(ReportingTools)

In case an error message is displayed, repeat step 3 for the missing packages, otherwise go to Section 4.
4 Quick start

If you have successfully gone through the installation you are ready to use RNASEqGUI, as follows.

1 - Open R.

2 - Type

   library(RNASEqGUI)

   in the R environment. Wait for the package to be loaded.

3 - Finally, type

   RNASEqGUI()

After that, a dialog window, as that one shown in Figure 5 will appear and you can start interacting with the program.
5 What’s new

• July 16, 2014 RNASeqGUI_0.99.2 was released

In the version RNASeqGUI_0.99.2, we present some new features, such as:

1 - Reactive Data Exploration via a web browser thanks to Reporting-Tools package (Show Results button for all the methods),

2 - Reproducible Research thanks to knitr package (Log file button),

3 - Complex Design Analysis for EdgeR, DESeq and DESeq2,

4 - Utility Interface,

5 - FeatureCounts (a new alternative method included in the Read Count Interface),

6 - Venn Diagrams DE 4 sets in the Result Inspection Interface,

7 - bplapply function of BiocParallel package was introduced again to speed up the Count Section.

• May 15, 2014 RNASeqGUI_0.99.1 was released

In the version RNASeqGUI_0.99.1

1 - We fix a bug present in DESeq and in DESeq2, since up and down regulated genes where swapped

2 - Minor point. In this version, we replaced "bplapply" function of BiocParallel with "lapply" function since with BiocParallel_0.4.1 RNASeqGUI worked fine, but with the latest version (BiocParallel_0.7.0)
we found some problems. We are now trying to find out why things have changed.

- March 26, 2014 RNASeqGUI 0.99.0 was released

First release of RNASeqGUI
6 Structure of RNASeqGUI main interface

The RNASeqGUI main interface is divided into six Sections, as shown in Figure 5. Each section corresponds to a particular step of the RNA-Seq data analysis work-flow. Each section contains one or more Graphical Interfaces that can be called by clicking the corresponding button.

Inside each interface, there is a How to use this interface button that displays a vignette to help the user to use the interface (see Figure 11) and there are several available functionalities (also called functions or methods in the rest of the manual). Each function takes specific inputs that can be numeric ones, strings or both and generate an output that can be a plot, a text file or both.

The sections of RNASeqGUI will be described one by one in the next sections of this manual.
Figure 5: Sections of RNASeqGUI main interface
Figure 6: Creation of a new project
Figure 7: Selection of an existing project

Figure 8: Structure of the MyProject directory
7 How to create a new project or select an existing one

To start using RNASeqGUI, you must either create a new project by choosing a name for it (suppose you choose as name MyProject) and then clicking on the Create a New Project button (see Figure 6) or select an existing project by typing the name and then clicking on the Select this Project! button (see Figure 7). The two cases are explained below.

1. In the first case, if you are using RNASeqGUI for the first time a directory called RNASeqGUI_Projects is created in your current working directory (type getwd() in the R environment to know where you are). Inside RNASeqGUI_Projects directory, a project folder is created with the name chosen by you (in this case with the name MyProject).

At any moment, you can see or change your working directory with the following R commands, respectively.

getwd()

setwd("path/you/want/to/set")

The creation of RNASeqGUI_Projects directory will only occur the first time you start using RNASeqGUI. Subsequently, when you click the Create a New Project button, RNASeqGUI checks whether the RNASeqGUI_Projects folder already exists in your working directory. If this folder, was already created then RNASeqGUI does not create a copy of it and all the projects you will create will be stored in it.

Now, inside RNASeqGUI_Projects, you find MyProjects directory. Inside this directory, three folders are automatically created (see Figure 8), such as: Logs, Results, Plots. In the Logs folder, a report.Rmd file is created to report all the actions you perform and which parameters you use by performing those actions. A session information that summaries all the versions of the used packages is automatically written in the report.Rmd file (see Figure 36) at the creation of the project and each time you start this project.
Figure 9: An example of the file report.Rmd automatically created in Logs directory at the creation of MyProject project. Note that the session information is included.

2. In the second case, an existing project is selected, see Figure 7. RNASe-qGUI checks whether the selected name already exists in the RNASe-qGUI Projects folder. If no project with the chosen name is found, a message warns the user that the selected project does not exist. When an existing project is restarted, RNASeqGUI continues to write in the same report.Rmd file created previously.
8 BAM EXPLORATION SECTION

8.1 Bam Exploration Interface

In the first section of the GUI, we find the *Bam Exploration Interface* (see Figure 10) that can be easily called by clicking the corresponding button. In this interface we find five different methods to explore the bam files: *Read Counts, Mean Quality of the Reads, Per Base Quality of Reads, Reads Per Chromosome, Nucleotide Frequencies*. Each of these functions takes a folder name as input. This input folder must contain all the bam files that the user wants to explore. To select the entire bam folder, select just one bam file inside the bam folder you want to use. The entire folder will be loaded. To use this interface you can also click on *How to use this Interface* button and a vignette window will appear on the screen describing the interface usage briefly, as shown in Figure 11.

- The **Read Counts** makes use of `barplot` function of the `graphics` package. This function returns an histogram (as the one shown in Figure 11) showing the number of mapped reads in each bam file (stored in the input folder) and a txt (tab-delimited) file summarizing the counts.

- The **Mean Quality of the Reads** makes use of `plotQuality` function of the `EDASeq` package [Risso et al., 2011]. This function returns a plot showing the quality of each base of the reads averaged across all bam files.

- The **Per Base Quality of Reads** makes use of `plotQuality` function of the `EDASeq` package [Risso et al., 2011]. This function returns as many box-plots as the number of bam files stored in the provided input folder. Each box-plot shows the quality of the reads per each base. This function makes use of `bplapply` function of the `BiocParallel` package [Morgan et al., 2014] to parallelize the code in order to reduce the execution time.

- The **Reads Per Chromosome** makes use of `barplot` function of the `graphics` package. This function returns as many histograms as the number of bam files stored in the provided input folder. Each histogram shows the number of reads are present in each chromosome. This function makes use of `bplapply` function of the `BiocParallel` package [Morgan et al., 2014] to parallelize the code in order to reduce the execution time.
Figure 10: By clicking the **Bam Exploration Interface** button (in the red cycle), the interface to explore bam files will be displayed.
Figure 11: By clicking **How to use this Interface** button, a vignette window will appear on the screen.
• The Nucleotide Frequencies makes use of `plotNtFrequency` function of EDASeq package [Risso et al., 2011]. This function returns a plot showing the percentage of each nucleotide at each position of the reads.

Figures will be stored in folder **Plots**, tables in folder **Results**.
9 COUNT SECTION

9.1 Read Count Interface

In the second section of the GUI, you find two functions for counting reads: 

SummarizeOverlaps [Lawrence et al., 2013] and FeatureCounts [Liao et al., 2013].

- **SummarizeOverlaps** takes four inputs (see Figure 12). The first input must be the name of the folder containing the bam files we want to process. The second input must be an annotation file in GTF format (General Transfer Format). The third input specifies the count mode that can be one of the following: **Union**, **IntersectionStrict** and **IntersectionNotEmpty**. The fourth input is **Ignore Strand?** check-box that allows to perform a strand specific counting task or not.

The **SummarizeOverlaps** button calls **summarizeOverlaps** function of the **GenomicRanges** package [Lawrence et al., 2013] to obtain gene counts and returns a data-frame, as the one shown in Figure 13. The first column of this data-frame represents the **Gene Id**, while the other columns correspond to the names of the loaded bam files. The other entries report the number of reads that have hit a particular gene for each sample (see [www.bioconductor.org/packages/release/bioc/vignettes/GenomicRanges/inst/doc/summarizeOverlaps.pdf](www.bioconductor.org/packages/release/bioc/vignettes/GenomicRanges/inst/doc/summarizeOverlaps.pdf) for more information about the counting modes).

- The second one is **FeatureCounts** of the **Rsubread** package [Liao et al., 2013]. This method takes four inputs (see Figure 12). The first input must be the name of the folder containing the bam files we want to process. The second input must be an annotation file in GTF format (General Transfer Format). The third input is the **Strand Number** field that can be one of the following: 0 (unstranded), 1 (stranded), 2 (reversely stranded). The fourth input is **Number of threads** field that specifies the number of the threads to use for the counting process. The fifth input is **Paired End?** check-box that allows the counting mode either for paired-end reads or for single-end ones.

The **FeatureCounts** button calls **FeatureCounts** function of the Rsubread package to obtain gene counts and returns a data-frame, as the one shown in Figure 13. The first column of this data-frame represents
Figure 12: Read Count Interface
Figure 13: An example of a count file with 20062 genes. The row names are given by the Gene Id in the annotation file (gtf), the column names are given by the alignment file names (the bam files)

<table>
<thead>
<tr>
<th>Gene Id</th>
<th>control_1</th>
<th>control_2</th>
<th>treated_1</th>
<th>treated_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG000000000003</td>
<td>455</td>
<td>463</td>
<td>583</td>
<td>598</td>
</tr>
<tr>
<td>ENSG000000000005</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ENSG000000000419</td>
<td>1174</td>
<td>1210</td>
<td>1545</td>
<td>1533</td>
</tr>
<tr>
<td>ENSG000000000457</td>
<td>260</td>
<td>256</td>
<td>305</td>
<td>349</td>
</tr>
<tr>
<td>ENSG000000000460</td>
<td>550</td>
<td>607</td>
<td>709</td>
<td>741</td>
</tr>
</tbody>
</table>

Read counting process can be a very computational demanding task, especially for large experiments with several samples and big alignment files. The R environment is not optimized from this point of view. Therefore, the counting task can be problematic on standard PC with limited clock speed and memory space. In this case, it could be beneficial either to process samples independently or to import count tables (in the format specified in Figure 13) in RNASeqGUI obtained from other tools, such as HTSeq-count (www-huber.embl.de/users/anders/HTSeq/). Therefore, this function makes use of `bplapply` function of the BiocParallel package [Morgan et al., 2014] to parallelize the code in order to reduce the execution time.
10 PRE-ANALYSIS SECTION

The third section of the GUI contains two interfaces: Data Exploration Interface (see Figure 14) and Normalization Interface (see Figure 15). Both interfaces take an input count file that must be tab-delimited and must have the structure shown in Figure 13. The rows represent genes ids and the columns represent the samples.

10.1 Data Exploration Interface

In Data Exploration Interface there are twelve methods: Plot Pairs of Counts, Plot all Counts, Count Distr, Density, MDPlot, Mean-VarPlot, Heatmap, PCA, PCA3D, Component Histogram, Qplot Histogram, Qplot Density.

- The Plot Pairs of Counts makes use of plot function of the graphics.
This function takes a count file as input (in **txt** or **cvs** format) where the rows correspond to the gene ids and the columns correspond to the samples. This function also takes two integers, one specifying **Column1** and the other specifying **Column2** of the count file (see Figure [14]) and plots the counts of sample in **Column1** against the counts of sample in **Column2**. Moreover, for this function it is possible to plot either the raw counts or the log of the counts (we add 1 to each number in the count file to avoid the problem of \( \log(0) \)).

- The **Plot all Counts** makes use of **plot** function of the **graphics** package. This function takes a count file as input and produces all possible plots that can be generated by each column in the file against all the other columns. If the input text file has \( n \) columns then \( n(n-1) \) plots will be produced. An example of this plot is shown in Figure [48]. For this function, the **log** check box does not change anything.

- The **Count Distr** makes use of **boxplot** function of the **graphics** package. This function takes a count file as input and generates a box plot showing the distribution of the counts for each column in the file. An example of this plot is shown in Figure [46]. Moreover, for this function it is possible to generate the box plot either of the raw counts or the log of the counts (we add 1 to each number in the count file to avoid the problem of \( \log(0) \)).

- The **Density** makes use of **density** function of the **stats** package. This function takes a count file, and a sample specified by an integer in **Column1** as input and produces a curve representing the density function of the counts for the selected sample. The method is available in two modes. By default the log of the counts (we add 1 to each number in the count file to avoid the problem of \( \log(0) \)) will be used to generate the density function. It is possible to uncheck this mode by clicking in the **log?** check-box (see Figure [13]).

- The **MDPlot** makes use of **MDplot** function of the **EDASeq** package [Risso et al., 2011]. This function takes a count file and two integers **Column1** and **Column2** and returns a plot showing the mean of the two selected columns against their difference gene by gene. For this function, the **log** check box does not change anything.

- The **MeanVarPlot** makes use of **meanVarPlot** function of the **EDASeq** package [Risso et al., 2011]. This function takes a count file and returns a plot showing the mean of all columns found in the file against the
variance gene by gene. For this function, the log check box does not change anything.

- The **Heatmap** makes use of `heatmap` function of the `stats` package. This function takes a count file and an integer $N$ in the **How many genes in the Heatmap?** field. The function returns an heat-map of the $N^{th}$ most expressed genes (on average). The columns of the heatmap are the samples, while the rows in the heat-map represent the gene ids of the most expressed ones. An example of heat-map is shown in Figure [40]. Moreover, for this function it is possible to generate the heatmap either of the raw counts or the log of the counts (we add 1 to each number in the count file to avoid the problem of $\log(0)$).

- The **PCA** makes use of `prcomp` function of the `stats` package. This function takes a count file, a comma separated sequence of strings (e.g.: a,b,c,d) indicating what are the labels for the legend, to be specified in the field **Factors** (see Figure [41] and **Legend position in PCA** that can be: topright, bottomright, topleft, bottomleft. The PCA function returns the principal component analysis plot between the first two components. An example of PCA plot is shown in Figure [49]. For this function, the log check box does not change anything.

- The **PCA3D** makes use of `scatterplot3d` function of the `scatterplot3d` package. This function takes the same inputs of the PCA function and returns the 3D PCA plot between the first, the second and the third principal component. For this function, the log check box does not change anything.

- The **Component Histogram** makes use of `screeplot` function of the `stats` package. This function takes a count file and returns an histogram showing the variance level of each component. For this function, the log check box does not change anything.

- The **Qplot Histogram** makes use of `qplot` function of the `ggplot2` package. This function takes a count file and and returns a plot showing the density function of each column in the count file. Moreover, for this function it is possible to generate the histogram either of the raw counts or the log of the counts (we add 1 to each number in the count file to avoid the problem of $\log(0)$).

- The **Qplot Density** makes use of `qplot` function of the `ggplot2` package. This function takes a count file and and returns a plot showing the density function of each column in the count file. Moreover, for this
function it is possible to generate the density either of the raw counts or the log of the counts (we add 1 to each number in the count file to avoid the problem of $\log(0)$).

### 10.2 Normalization Interface

The **Normalization Interface** (see Figure 15) includes four normalization procedures: **RPKM, Upper Quartile, TMM, Full Quantile**.

- **RPKM** makes use of `rpkm` function of the NOISeq package [Tarazona *et al.*., 2011]. This function takes a count file as specified in Figure 13 and returns a count file with normalized numbers. This function performs the RPKM [Mortazavi *et al.*, 2008] normalization.

- **Upper Quartile** makes use of `uqua` function of the NOISeq package [Tarazona *et al.*, 2011]. This function takes a count file as specified in Figure 13 and returns a count file with normalized numbers. This function performs the Upper Quartile [Bullard *et al.*, 2010] normalization.

- **TMM** makes use of `tmm` function of the NOISeq package [Tarazona *et al.*, 2011]. This function takes a count file as specified in Figure 13 and returns a count file with normalized numbers. This function performs the TMM [Robinson *et al.*, 2010] normalization.
• **Full Quantile** makes use of `normalize.quantiles` function of the `preprocessCore` package. This function takes a count file as specified in Figure 13 and returns a count file with normalized numbers. This function performs the Full Quantile [Bolstad et al., 2003, Smyth et al., 2005] normalization.
DATA ANALYSIS SECTION

11.1 Data Analysis Interface

This section contains the Data Analysis Interface shown in Figure 16 and represents the core of RNASeqGUI. This interface includes eight different statistical methods to detect differentially gene expression, such as: EdgeR, EdgeRComplexDesign, DESeq, DESeqEdgeRComplexDesign, DESeq2, DESeq2EdgeRComplexDesign, NoiSeq, BaySeq.

Results of all methods can be viewed and explored on a web browser thanks to ReportingTools [Huntley et al., 2013] library that allows the user to navigate through them (see figure Figure 51).

11.2 EdgeR

- The EdgeR method [Robinson et al., 2007, Robinson et al., 2008, Robinson et al., 2010] [McCarthy et al., 2012] (see Figure 17) takes an input count file (as the one shown in Figure 13) via the Open button. In the Factors? field the user can specify each condition of the count file loaded.
  - In the FDR? field the user can specify the False Discovery Rate cor-
Figure 17: EdgeR interface

<table>
<thead>
<tr>
<th>id</th>
<th>logFC</th>
<th>logCPM</th>
<th>PValue</th>
<th>FDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG..003</td>
<td>0.023</td>
<td>9.181</td>
<td>0.736</td>
<td>1</td>
</tr>
<tr>
<td>ENSG..005</td>
<td>2.357</td>
<td>1.058</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ENSG..419</td>
<td>0.072</td>
<td>10.003</td>
<td>0.178</td>
<td>0.571</td>
</tr>
<tr>
<td>ENSG..457</td>
<td>-0.043</td>
<td>8.418</td>
<td>0.612</td>
<td>0.966</td>
</tr>
<tr>
<td>ENSG..460</td>
<td>-0.0006</td>
<td>9.164</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ENSG..938</td>
<td>2.5e-15</td>
<td>0.888</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ENSG..971</td>
<td>0.078</td>
<td>1.472</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>..........</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
<tr>
<td>..........</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
<td>.....</td>
</tr>
</tbody>
</table>

Figure 18: The first text file produced by the EdgeR method. The first column reports the gene ids, logFC reports the log of the fold-changes, logCPM reports the log of the counts per million, PValue reports the p-values and FDR reports the false discovery rates calculated by the Benjamini and Hochberg’s algorithm.
<table>
<thead>
<tr>
<th>id</th>
<th>logFC</th>
<th>logCPM</th>
<th>PValue</th>
<th>FDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG..3756</td>
<td>-0.151</td>
<td>10.652</td>
<td>0.001</td>
<td>0.035</td>
</tr>
<tr>
<td>ENSG..4777</td>
<td>-0.523</td>
<td>8.455</td>
<td>2.6e-10</td>
<td>4.3e-08</td>
</tr>
<tr>
<td>ENSG..5961</td>
<td>-0.506</td>
<td>6.340</td>
<td>0.002</td>
<td>0.049</td>
</tr>
<tr>
<td>ENSG..6025</td>
<td>-0.577</td>
<td>8.699</td>
<td>2.8e-14</td>
<td>7.1e-12</td>
</tr>
<tr>
<td>ENSG..6047</td>
<td>-0.627</td>
<td>6.027</td>
<td>0.001</td>
<td>0.027</td>
</tr>
<tr>
<td>ENSG..6118</td>
<td>-0.152</td>
<td>10.456</td>
<td>0.001</td>
<td>0.039</td>
</tr>
<tr>
<td>ENSG..6282</td>
<td>-0.418</td>
<td>9.966</td>
<td>1.0e-14</td>
<td>3.3e-12</td>
</tr>
</tbody>
</table>

Figure 19: The EdgeR second text file showing the differentially expressed genes only. Columns are the same as in Figure 18.

rected by the Benjamini and Hochberg’s algorithm to infer which are the differentially expressed genes.

Finally, click on the **Run EdgeR** button.

**Run EdgeR** returns two text files and two plots.

The first text file shows the overall result obtained by edgeR (see Figure 18), while the second text file extracts the subset of differentially expressed genes only (see Figure 19).

The output count file is saved with the name specified by the user in the **Name?** field (see Figure 17).

If no name is specified by the user, then the first output count file is named with the name of the input file plus "*_results_EdgeR.txt" suffix. The second file is named with the name of the input file plus "*_fdr=0.05_DE_genes_EdgeR.txt" suffix, where 0.05 is the chosen FDR. Both text files are saved in the **Results** folder.

The first plot shows the Biological Coefficient of Variation for a given CPM (Count Per Million) and is named with the name of the input file plus "*_Dispersion_EdgeR.pdf" suffix. The second plot shows the relative similarities of the samples and is named with the name of the input file plus "*_MDS_EdgeR.pdf" suffix. Both plots are saved in the **Plots** folder.
11.3 EdgeR Multi Factor / Complex Design

If you want to perform a multiple test or you have a more complex design you can use the *EdgeR Multi Factor / Complex Design* interface (see Figure 20).

Suppose you have two treatments (T1, T2) and one control (U). For instance, \textit{Factors}: U, U, T1, T1, T2, T2.

In the \textit{LibTypes} field the user can specify an extra feature regarding the factors.

Suppose that \textit{LibTypes} specifies the type of reads used in your experiment for each factor.

For instance, \textit{LibTypes}: single-end, paired-end, single-end, paired-end, paired-end, single-end.

Finally, you need to specify the \textit{Coefficient} field.

Set \textit{Coefficient}: 2, to compare T1 vs U
Set \textit{Coefficient}: 3, to compare T2 vs U
\textit{Coefficient}: 1, should not be used.

Finally, click on the \textbf{Run EdgeRComplexDesign} button.

For further information, see \url{www.bioconductor.org/packages/release/bioc/vignettes/edgeR/inst/doc/edgeR.pdf}.

\textbf{Run EdgeRComplexDesign} returns two text files and two plots.

The first text file shows the overall result obtained by \textbf{Run EdgeRComplexDesign}, while the second text file extracts the subset of differentially expressed genes only.

The output count file is saved with the name specified by the user in the \textit{Name} field (see Figure 20).

If no name is specified by the user, then the first output count file is named with the name of the input file plus \texttt{“results\_EdgeRComplexDesign.txt”} suffix. The second file is named with the name of the input file plus \texttt{“fdr=0.05\_DE\_genes\_EdgeRComplexDesign.txt”} suffix, where 0.05 is the chosen FDR. Both text files are saved in the \textbf{Results} folder.

The first plot shows the Biological Coefficient of Variation for a given CPM (Count Per Million) and is named with the name of the input file plus \texttt{“Dispersion\_EdgeRComplexDesign.pdf”} suffix. The second plot shows the relative similarities of the samples and is named with the name of the input file plus \texttt{“MDS\_EdgeRComplexDesign.pdf”} suffix. Both plots are saved in the \textbf{Plots} folder.
11.4 DESeq

The DESeq method [Anders et al., 2010] (see Figure 21) takes an input count file (as the one shown in Figure 13) via the Open button. In the Factors? field the user can specify each condition of the count file loaded.

In the Padj? field the user can specify the P-value adjusted corrected by the Benjamini and Hochberg’s algorithm to infer which are the differentially expressed genes.

In the LibTypes? field the user can specify an extra feature regarding the factors.

For the count example in the Figure 13 LibTypes? is set to be: paired-end, paired-end, paired-end, single-end.

In the Treated field the user can specify which factor is the treated one.

In the Control field the user can specify which factor is the control one.

Finally, click on the Run DESeq button.

Run DESeq returns two text files and two plots.

The first text file shows the results of this method (see Figure 22), while
the second text file shows the differentially expressed genes only. The output count file is saved with the name specified by the user in the Name? field (see Figure 21).

If no name is specified by the user, then the first output count file is named with the name of the input file plus “results_DESeq.txt” suffix. The second file is named with the name of the input file plus “padj=0.05_DE_genes_DESeq.txt” suffix, where 0.05 is the chosen p-value adjusted.

Both text files are saved in the Results folder. The generated plot shows the dispersion value for a given mean of normalized counts. This plot is named with the name of the input file plus “Dispersion_DESeq.pdf” suffix and it is saved in the Plots folder.

11.5 DESeq Multi Factor / Complex Design

If you want to perform a multiple test or you have a more complex design you can use the DESeq Multi Factor / Complex Design interface.
Suppose you have two treatments (T1, T2) and one control (U). For instance, \textbf{Factors?}: U, U, T1, T1, T2, T2.

In the \textbf{LibTypes?} field the user can specify an extra feature regarding the factors. Suppose that \textbf{LibTypes} specifies the type of reads used in your experiment for each factor. For instance, \textbf{LibTypes?): single-end, paired-end, single-end, paired-end, paired-end, single-end. Finally, click on the \textbf{Run DESeqComplexDesign} button.

A file with For further information, see \url{www.bioconductor.org/packages/release/bioc/vignettes/DESeq/inst/doc/DESeq.pdf}.

\textbf{Run DESeqComplexDesign} returns two text files and two plots. The first text file shows the results of this method, while the second text file shows the differentially expressed genes only.

The output count file is saved with the name specified by the user in the \textbf{Name?} field.
<table>
<thead>
<tr>
<th>id</th>
<th>baseMean</th>
<th>baseMeanA</th>
<th>baseMeanB</th>
<th>foldChange</th>
<th>log2FoldChange</th>
<th>pval</th>
<th>padj</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG...0003</td>
<td>625.025</td>
<td>630.902</td>
<td>619.147</td>
<td>0.981</td>
<td>-0.027</td>
<td>0.774</td>
<td>1</td>
</tr>
<tr>
<td>ENSG...0005</td>
<td>0.264</td>
<td>0.528</td>
<td>0</td>
<td>0</td>
<td>-Inf</td>
<td>0.985</td>
<td>1</td>
</tr>
<tr>
<td>ENSG...0419</td>
<td>1106.882</td>
<td>1136.118</td>
<td>1077.646</td>
<td>0.948</td>
<td>-0.076</td>
<td>0.297</td>
<td>0.935</td>
</tr>
<tr>
<td>ENSG...0457</td>
<td>367.367</td>
<td>362.361</td>
<td>372.374</td>
<td>1.027</td>
<td>0.039</td>
<td>0.744</td>
<td>1</td>
</tr>
<tr>
<td>ENSG...0460</td>
<td>617.493</td>
<td>618.055</td>
<td>616.931</td>
<td>0.998</td>
<td>-0.002</td>
<td>0.982</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 23: DESeq output. The first column reports the gene ids, `baseMean` reports the mean normalised counts, averaged over all samples from both conditions, `baseMeanA` reports the mean normalised counts from condition A, `baseMeanB` mean normalised counts from condition B, `foldChange` reports the fold changes from condition A to B, `log2FoldChange` reports the logarithm (to basis 2) of the fold changes, `pval` reports the p values for the statistical significance and `padj` reports the p values adjusted for multiple testing calculated by the Benjamini-Hochberg algorithm.

If no name is specified by the user, then the first output count file is named with the name of the input file plus “.DESeqComplexDesign.txt” suffix.

The second file is named with the name of the input file plus “.padj=0.05.DE_genes.DESeqDESeqComplexDesign.txt” suffix, where 0.05 is the chosen p-value adjusted.

Both text files are saved in the Results folder. The generated plot shows the dispersion value for a given mean of normalized counts.

This plot is named with the name of the input file plus “Dispersion_DESeqComplexDesign.pdf” suffix and it is saved in the Plots folder.

### 11.6 DESeq2

- The DESeq2 method [Anders et al., 2010](#) (see Figure 24) takes an input count file (as the one shown in Figure 13) via the Open button and returns two text files and three plots.

The first text file shows the results of this method (see Figure 23), while the second text file shows the differentially expressed genes only.
The output count file is saved with the name specified by the user in the Name? field (see Figure 24).

If no name is specified by the user, then the first file is named with the name of the input file plus “results_DESeq2.txt” suffix. Both text files are saved in the Results folder.

The second file is named with the name of the input file plus “padj=0.05_DE_genes_DESeq2.txt” suffix, where 0.05 is the chosen adjusted p-value for rejection.

The first plot shows the dispersion value for a given mean of normalized counts and it is named with the name of the input file plus the “Dispersion_DESeq2.pdf” suffix.

The second plot shows the dispersion mean value for a given mean of normalized counts and it is named with the name of the input file plus the “Dispersion_Mean_DESeq2.pdf” suffix.

The third plot shows the dispersion local value for a given mean of normalized counts and it is named with the name of the input file plus the Dispersion_Local_DESeq2.pdf suffix.
If you want to perform a multiple test or you have a more complex design you can use the DESeq2 Multi Factor / Complex Design interface (see Figure 25).

Suppose you have two treatments (T1, T2) and one control (U). For instance, Factors?: U, U, T1, T1, T2, T2.

In the LibTypes? field the user can specify an extra feature regarding the factors.

Suppose that LibTypes specifies the type of reads used in your experiment for each factor.

For instance, LibTypes?: single-end,paired-end,single-end,paired-end, paired-end,single-end.

Finally, click on the Run DESeq2ComplexDesign button.

A file with For further information, see www.bioconductor.org/packages/release/bioc/vignettes/DESeq/inst/doc/DESeq2.pdf.

Run DESeq2ComplexDesign returns two text files and two plots. The first text file shows the results of this method, while the second text file shows the differentially expressed genes only.
<table>
<thead>
<tr>
<th>id</th>
<th>baseMean</th>
<th>log2FoldChange</th>
<th>lfcSE</th>
<th>stat</th>
<th>pvalue</th>
<th>padj</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG000000000003</td>
<td>625.025</td>
<td>-0.025</td>
<td>0.079</td>
<td>-0.318</td>
<td>0.750</td>
<td>0.954</td>
</tr>
<tr>
<td>ENSG000000000005</td>
<td>0.264</td>
<td>-0.014</td>
<td>0.020</td>
<td>-0.675</td>
<td>0.499</td>
<td>0.911</td>
</tr>
<tr>
<td>ENSG000000000419</td>
<td>1106.882</td>
<td>-0.072</td>
<td>0.062</td>
<td>-1.174</td>
<td>0.240</td>
<td>0.768</td>
</tr>
<tr>
<td>ENSG00000000457</td>
<td>367.367</td>
<td>0.035</td>
<td>0.095</td>
<td>0.365</td>
<td>0.714</td>
<td>0.937</td>
</tr>
<tr>
<td>ENSG00000000460</td>
<td>617.493</td>
<td>-0.002</td>
<td>0.079</td>
<td>-0.033</td>
<td>0.973</td>
<td>0.994</td>
</tr>
</tbody>
</table>

Figure 26: DESeq2 output. The first column reports the gene ids, baseMean reports the base mean over all rows, log2FoldChange reports the logarithm (to basis 2) of the fold changes, lfcSE reports the standard errors, stat reports the Wald statistic, pval reports the p values for the statistical significance and padj reports the p values adjusted for multiple testing calculated by the Benjamini-Hochberg algorithm.

The output count file is saved with the name specified by the user in the Name? field.

If no name is specified by the user, then the first output count file is named with the name of the input file plus “.results_DESeq2.txt” suffix.

The second file is named with the name of the input file plus “.padj=0.05_DE_genes_DESeq2.txt” suffix, where 0.05 is the chosen p-value adjusted.

Both text files are saved in the Results folder. The generated plot shows the dispersion value for a given mean of normalized counts.

This plot is named with the name of the input file plus “_Dispersion_DESeq2.pdf” suffix and it is saved in the Plots folder.

### 11.8 NoiSeq

- The NoiSeq [Tarazona et al., 2011] method (see Figure 27) takes an input count file (as the one shown in Figure 13) via the Open button and returns two text files.

The first text file shows the results of this method (see Figure 28), where M is the log2 ratio of the two conditions. The second text file shows the differentially expressed genes only.

46
The first file is named with the name of the input file plus “_results_Noiseq.txt” suffix. The output count file is saved with the name specified by the user in the Name? field (see Figure 27).

If no name is specified by the user, then the second file is named with the name of the input file plus “_prob=0.8_DE_genes_Noiseq.txt” suffix, where 0.8 is the chosen posterior probability for rejection. Both text files are saved in the Results folder. Both plots are saved in the Plots folder.

11.9 BaySeq

- The BaySeq [Hardcastle et al., 2010] method (see Figure 29) takes an input count file (as the one shown in Figure 13) via the Open button, a list of factors (e.g. treated,treated, control,control) in the Factors? field, a NDE list (e.g. 1,1,1,1), a DE list (e.g. 1,1,2,2), an Estimation Type? (e.g. quantile), the SampleSize (e.g. 1000), an FDR level, SampleA (e.g. treated) and SampleB (e.g. control).

The BaySeq function returns two text files and two plots.
<table>
<thead>
<tr>
<th>id</th>
<th>control_mean</th>
<th>treated_mean</th>
<th>M</th>
<th>D</th>
<th>prob</th>
<th>ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000000003</td>
<td>575.05</td>
<td>582.71</td>
<td>-0.019</td>
<td>7.659</td>
<td>0.104</td>
<td>-7.659</td>
</tr>
<tr>
<td>ENSG00000000005</td>
<td>0.22</td>
<td>0.47</td>
<td>-1.083</td>
<td>0.251</td>
<td>0.037</td>
<td>-1.112</td>
</tr>
<tr>
<td>ENSG000000000419</td>
<td>1000.84</td>
<td>1049.17</td>
<td>-0.068</td>
<td>48.333</td>
<td>0.405</td>
<td>-48.333</td>
</tr>
<tr>
<td>ENSG000000000457</td>
<td>345.75</td>
<td>334.47</td>
<td>0.047</td>
<td>11.275</td>
<td>0.164</td>
<td>11.275</td>
</tr>
<tr>
<td>ENSG000000000460</td>
<td>572.81</td>
<td>570.80</td>
<td>0.005</td>
<td>2.004</td>
<td>0.028</td>
<td>2.004</td>
</tr>
</tbody>
</table>

Figure 28: NoiSeq result file. The first column reports the gene ids, control_mean is the mean across the control samples, treated_mean is the mean across the treated samples, M is the log2-ratio of the means of the two conditions) and D is the difference between the two conditions means, prob is the probability of differential expression, the ranking is a summary statistic of M and D values (equal to $-\text{sign}(M) \times \sqrt{M^2 + D^2}$).

Figure 29: BaySeq Interface
<table>
<thead>
<tr>
<th>id</th>
<th>rowID</th>
<th>control_1</th>
<th>control_2</th>
<th>treated_1</th>
<th>treated_2</th>
<th>Likelihood</th>
<th>FDR.DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG..971</td>
<td>row_7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.261</td>
<td>0.738</td>
</tr>
<tr>
<td>ENSG..419</td>
<td>row_3</td>
<td>1132</td>
<td>1070</td>
<td>1088</td>
<td>1138</td>
<td>0.217</td>
<td>0.760</td>
</tr>
<tr>
<td>ENSG..457</td>
<td>row_4</td>
<td>354</td>
<td>348</td>
<td>392</td>
<td>377</td>
<td>0.111</td>
<td>0.803</td>
</tr>
<tr>
<td>ENSG..003</td>
<td>row_1</td>
<td>633</td>
<td>590</td>
<td>618</td>
<td>661</td>
<td>0.074</td>
<td>0.833</td>
</tr>
<tr>
<td>ENSG..460</td>
<td>row_5</td>
<td>618</td>
<td>580</td>
<td>653</td>
<td>621</td>
<td>0.067</td>
<td>0.853</td>
</tr>
<tr>
<td>ENSG..005</td>
<td>row_2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.051</td>
<td>0.869</td>
</tr>
</tbody>
</table>

Figure 30: BaySeq result file. Bayseq reports the input counts and the number of the row (rowID) in the first columns and the Likelihood and the false discovery rate (FDR.DE) in the remaining columns.

The first text file shows the results of this method (see Figure 30), while the second text file shows the differentially expressed genes only.

The output count file is saved with the name specified by the user in the Name? field (see Figure 29).

If no name is specified by the user, then the first file is named with the name of the input file plus "results_BaySeq.txt" suffix. Both text files are saved in the Results folder.

The second file is named with the name of the input file plus "fdr=0.05_DE_genes_BaySeq.txt" suffix, where 0.05 is the chosen FDR for rejection.

The first plot shows the log ratios of the counts against the mean average of the counts and it is named with the name of the input file plus the _PlotMA_BaySeqNB.pdf suffix.

The second plot shows the posterior likelihood. This plot is named with the name of the input file plus the _Posteriors_BaySeqNB.pdf suffix.

This method is very time consuming.
12 POST ANALYSIS SECTION

In the fifth section of the GUI, called Post Analysis Interface, there are two interfaces: Result Inspection Interface (see Figure 31) and Result Comparison Interface (see Figure 34). The first interface includes the possibility to generate several plots for each method. The second allows to compare the outcomes obtained from several methods.

12.1 Result Inspection Interface

To explore the results of a specific method, we have to click on the used method in Data Analysis Section (say EdgeR) and the interface in Figure 31 will display the functions available for the selected method (for EdgeR Plot FC, FDR Hist, P-value Hist functions are available). If we click all buttons in Figure 31 the interface will grow and we get the interface shown in Figure 32.

Therefore, for each method, we have Plot FC, FDR Hist (or P-value Hist) and Volcano Plot functions, except for the BaySeq method since this method already provides an MAplot and a PosteriorPlot during the analysis process that can be run in the BaySeq Analysis Interface.

For each function (e.g.: FDR Hist, P-value Hist, Likelihood Hist) of each method, we just need to provide a “full result” file placed in the Results
Figure 32: Result Inspection Interface after clicking all the eight buttons at the top.
folder. For Volcano Plot and Plot FC functions, we must provide a path to a “full result” file (as the one shown in Figure 18) and a FDR, P-value or Prob value (it depends on the chosen method) to point out the differentially expressed genes (shown in red). In this case, it is also possible to provide a gene id, provided into the Gene Id field, to point out that particular gene in the Volcano or FC plot (that gene will be displayed in green).

All generated plots are saved in pdf format in the Plots folder.

12.2 Result Comparison Interface

The second interface includes the possibility to generate Venn diagrams of two, three or four result text files (See Figure 34).

The user must provide two, three or four text files reporting the results of the used methods and the corresponding labels to recognize these files in the generated diagrams.

A Venn diagram is generated and saved in the Plots folder. Moreover, a text file (showing the gene ids belonging to the intersection of the selected methods) is created and saved in the Results folder.
13 REPORT AND UTILITY SECTION

13.1 Reproducible Research: the Log Files

In the spirit of Reproducible Research, RNASeqGUI is able to automatically generate a report, in html format, of all steps performed during the analysis of a specific project. Reports are produced in R markdown format via knitr library and they include the documentation of the methods used and the R code that has been executed during the RNASeqGUI usage.

Hence, all the functionalities used by the user are automatically saved in a report file (as the one shown in Figure 36) inside the Logs directory of the user project. This report reports the session information that describes all used package versions by RNASeqGUI at the time of the project creation, along side with the name of the project, time, date and the parameters (fdr, padj, etc.) the user selected during the usage of the GUI.
In the *Bam Exploration Interface*, you clicked the **Read Counts** button at 2014-07-15 17:03:19 and the *plot1 Reads CountHist.pdf* file has been saved in the *MyProject/Plots* folder and the *Bam_ReadCount.txt* file has been saved in the *MyProject/Results* folder.

This R code has been run:

```r
the.file <- '/exedir/88a4404-f3c2-4f8c-9e4-85e05c80f05d/88a/dem'  
Project <- 'MyProject'  
files_list.files(path=the.file, pattern = 'bam$', full.names = TRUE)  
bfs <- BamFileList(files)  
colors <- c('red', 'blue', 'green', 'brown', 'purple', 'darkgreen', 'pink', 'orange', 'gold', 'darkblue', 'cyan', 'darkred')  
PPR.countbam(bfs)  
for (i in 1:length(rownames(bam))){  
    rownames(bam)[i] = substring(rownames(bam)[i],1,nchar(rownames(bam)[i])-4)  
}  
barplot(bam$counts, names.arg=rownames(bam), las=2, col=colors, main='Read Count Histogram')
```

**Figure 35:** An example of the html report file generated by the **html** button from the log file report.Rmd.
13.2 Utility Interface

Select a count folder by clicking on the corresponding Open button. To select the entire folder, select just one file inside the folder you want to use. The entire folder will be loaded. Please, be sure that the folder only contains the files you want to bind. Finally, click on Bind Count Files button.
14 Usage Example

We can start using RNASeqGUI by downloading the example data at [http://bioinfo.na.iac.cnr.it/RNASeqGUI/Example](http://bioinfo.na.iac.cnr.it/RNASeqGUI/Example), as shown in Figure 37.

We download the folder called example_RNASeqGUI.tar.gz, we extract this bundle and open it. Inside this, we find a folder called demo, a gtf file called 2L_Drosophila_melanogaster.BDGP5.70.gtf and a text file called README.txt file.

14.1 Data Preparation

In this usage example, we start the analysis of the RNA-Seq data from bam files and we compare the results of EdgeR, DESeq and NOISeq against each other.

We downloaded the dataset published by [Brooks et al., 2011](http://bioinfo.na.iac.cnr.it/RNASeqGUI/Example). This dataset has already been used in [Anders et al., 2013](http://bioinfo.na.iac.cnr.it/RNASeqGUI/Example) as a real data working example.


The dataset consists of seven samples. Three samples represent the response to a treatment and four samples are controls. Each sample is a cell culture of *Drosophila melanogaster* (For more details about this experiment see...
We downloaded and aligned the fastq files by running tophat2 [Kim et al., 2013] as described in [Anders et al., 2013] at page 1774. Once the bam files were obtained (we called them CG8144_RNA-1, CG8144_RNA-3, CG8144_RNA-4, Untreated-1, Untreated-3, Untreated-4, Untreated-6 as in [Anders et al., 2013]), it is possible to perform the analysis with RNASeqGUI.

For illustrative purpose and for keeping the computational cost of the demonstrative example under control, we limit our attention to chromosome 2L. Alignment data (bam files) are contained in the folder called demo inside the Bam folder, with the following names: 2L_1.bam, 2L_3.bam, 2L_4.bam, 2L_U1.bam, 2L_U3.bam, 2L_U4.bam, 2L_U6.bam (see Figure 38).

14.2 Usage of RNASeqGUI

We open R, then we type

```r
library(RNASeqGUI)
```

and we type

```r
RNASeqGUI()
```

Once the main RNASeqGUI interface (see Figure 5) has appeared on the screen, we create a new project (for instance, we can call it demoProject) and then we click on Bam Exploration Interface button. We select the demo folder with the Open button. After that, we start the analysis by using the Read Counts button in the Bam Exploration Interface. This action creates the plot shown in Figure 41. The bam files in the demo folder are
Figure 39: **Mean Quality of Reads** of the bam files stored in the folder demo without the 2L_1.bam file.

loaded in alphabetically order and their name are displayed at x axis in Figure alphabetically. This plot is automatically saved in pdf format in the Plots folder of the project you selected.

A text file is also generated and saved in the Results folder with the demo_Read Count.txt name, as shown in Figure 42. This file shows the number of reads for each bam file.

**Critical:** We cannot use the **Mean Quality of Reads** or **Per Base Quality of Reads** function for this dataset, since the 2L_1.bam file was generated by pulling fastq files containing reads of different length (This file correspond to 028144_861-1 at page 1774 of [Anders et al., 2013]). To use these functions, we need bam files containing reads of the same length. Otherwise, we get the following error:

Error in as.vector(x, 'character'): cannot coerce type 'environment' to vector of type 'character'.

If the user wants to use these functions, in this case the 2L_1.bam file must be temporary removed from the demo folder before using them. In this case, if we use those functions without the 2L_1.bam file, we get the plots in Figure 40 and in Figure 41 respectively.

Subsequently, we click on **Read Count Interface** and select the bam folder demo and the 2L_Drosophila_melanogaster.BDGP5.70.gtf annotation file. We select Union as **Counting Mode** and check the **Ignore Strand** box, as shown in Figure 43. Hence, we click on **Count Reads** button. As result of
Figure 40: **Per Base Quality of Reads** of the bam files stored in the folder `demo` without the `2L_1.bam` file.

Figure 41: Read Count Histogram of the bam files stored in the folder `demo`.

59
<table>
<thead>
<tr>
<th>fileName</th>
<th>NumberOfReads</th>
</tr>
</thead>
<tbody>
<tr>
<td>../Data/Bam/demo/2L_1</td>
<td>12320205</td>
</tr>
<tr>
<td>../Data/Bam/demo/2L_3</td>
<td>6477978</td>
</tr>
<tr>
<td>../Data/Bam/demo/2L_4</td>
<td>7741241</td>
</tr>
<tr>
<td>../Data/Bam/demo/2L_U1</td>
<td>9473462</td>
</tr>
<tr>
<td>../Data/Bam/demo/2L_U3</td>
<td>6586330</td>
</tr>
<tr>
<td>../Data/Bam/demo/2L_U4</td>
<td>6071744</td>
</tr>
<tr>
<td>../Data/Bam/demo/2L_U6</td>
<td>5883666</td>
</tr>
</tbody>
</table>

Figure 42: The `demo_ReadCount.txt` file saved in the **Results** folder.

Figure 43: Read Count Interface.
this action, a text file named 2L_counts.csv (see Figure 44) is generated and saved in the Results folder. A file named counts.txt is also generated in case the user forgets to use the Save Results? check-box at the bottom of the interface. The column names in Figure 43 follow the alphabetical order of the bam files placed in the demo folder.

Now, we can explore the obtained count file, shown in Figure 44. We click on Data Exploration Interface button. Once this interface has appeared on the screen (see Figure 45), we select the 2L_counts.csv file.

First, we use the Count Distr and the Plot All Counts functions by clicking the corresponding buttons (see Figure 45). The generated plots are shown in Figure 46 and Figure 48 respectively. From Figure 46, we can see that all the count means (the black lines in the box plot) and all the count distributions are almost aligned. Therefore, we decide not to normalize the counts since a normalization procedure does not seem to be necessary.

To better understand whether a normalization procedure is needed, we can also use the MDPlot by plotting each sample counts (by selecting Column1 and Column2 fields) against all the other sample counts.

Anyway, if we use the full quantile normalization procedure by clicking the Full Quantile button in the Normalization Interface, we get the plot show in Figure 47 and a text file of normalized counts saved in Results folder.

Subsequently, we use the PCA function by typing the 1,3,4,U1,U3,U4,U6...
sequence in the **PCA Factors?** field (see Figure 45) to specify the labels that will be displayed in the legend at the top-right of the plot generated by this function (shown in Figure 49).

Finally, we can use the **HeatMap** function to see what are the first (say thirty) most expressed genes. Therefore, we typed the number 30 in the **How many genes in the Heatmap?** field (see Figure 50). From the heatmap, we can notice that the the most expressed gene is the one called FBgn0000559 (look at the bottom of the Figure 50).

Now, we can start with the analysis. We decide to use EdgeR, DESeq and NOISeq and compare the results among them.

We click on **Data Analysis Interface** button.

We start the EdgeR analysis by clicking on the **EdgeR** button. In the **EdgeR Analysis Interface**, we select the **2L_counts.csv** count file.

We type the **T,T,T,U,U,U,U** sequence in the **Factors?** field to specify which are the treated samples (called **T**) and which are the untreated ones (called **U**) as reported in Figure 38. We choose a 0.05 value as the **FDR**. Finally, we click on **Run EdgeR** button. The EdgeR analysis is performed and two result text files are created and saved in the **Results** folder.
Figure 46: Box plot generated by the Count Distr function.
Figure 47: Boxplot of the counts shown in Figure 46 after the full quantile normalization.

Figure 48: Count plots generated by the Plot All Counts function.
Figure 49: PCA plot generated by the PCA function.
Figure 50: Heatmap
We click on DESeq button. In the DESeq Analysis Interface, we select the 2L_counts.csv count file. We type the T,T,T,U,U,U,U sequence in the Factors? field to specify the treated and untreated samples as in EdgeR analysis. We type single-end,paired-end,paired-end,single-end,paired-end,paired-end,single-end in the LibTypes field to specify the library layout as reported in Figure 38. We choose a 0.05 value as the Padj. Finally, we click on Run DESeq button. The DESeq analysis is performed and two result text files are created and saved in the Results folder. We can look at results by clicking on the Show Result Figure 51.

We click on NOISeq button. In the NOISeq Analysis Interface, we select the 2L_counts.csv count file. We type T1,T3,T4,U1,U3,U4,U6 in the TissueRun field to specify the library layout as specified in Figure 38. We choose a 0.6 value as the prob. Finally, we click on Run NOISeq button. The NOISeq analysis is performed and two result text files are created and saved in the Results folder.

Once all the results have been obtained, we can start inspecting them by...
clicking on Result Inspection Interface. We click on EdgeR, DESEQ and NOISeq buttons at the same time. At each click we can see the Result Inspection Interface growing (see the top-right of the Figure 52).

For each method, we select the corresponding result file (by giving the all path to the file in the Select File field) and we click on Plot FC on FDR Hist and on Volcano Plot of each method. We also provide a gene id to display a specific gene (in this case we type FBgn0000559 in the Gene Id field, as shown in Figure 52 that is the most expressed gene found in the heatmap in Figure 50).

Finally, we compare the results by clicking on Result Comparison Interface.

We fill all the fields as shown in Figure 53. We click on VennDiagrams3setsDE button. This action creates two files. The first file is the pdf shown in Figure 54 and saved in Plots folder. The second file is a text file, called NOISEQ_DESEQ_EDGER_genes_in_intersection.txt and saved in the Results folder. This text file reports the 86 gene-ids that fall in the intersection of all the three methods (see in Figure 54).

All the functionalities we have used are automatically saved in a report file inside the Logs directory.
Figure 52: Fold Change Plot generated by using the function PlotFC of EdgeR.
Figure 53: Result Comparison Interface

Figure 54: Venn Diagram
15 How to customize RNASeqGUI

It is extremely easy to add new buttons that calls new functions. Hence, a user can customize RNASeqGUI interfaces for his purposes and benefits by adding the methods he needs mostly.

15.1 Adding a new button in just three steps

For the sake of example, suppose you have written a function that generates a heat-map as the one written below.

```r
MyHeatmap <- function(x, geneNum) {
  require(RColorBrewer)
  n <- as.numeric(geneNum)
  x <- as.matrix(x)
  means = rowMeans(x)
  select = order(means, decreasing = TRUE)[1:n] # show first n genes
  hmcol = colorRampPalette(brewer.pal(7, "Greens"))(100)
  heatmap(x[select,], col = hmcol, margins = c(5, 8), main = "MyHeatMap")
}
```

If you want to add `MyHeatmap` function to RNASeqGUI, follow these three simple steps.

1 - Place `MyHeatmap` function in a file (for instance, called `MyHeatmap.R`) in the `R` folder inside the `RNASeqGUI` directory.

2 - Open `calculateGUI1.R` file (This is the file that generates the Data Exploration Interface) and copy the following 3 lines and paste them at the bottom of this file before “}” parenthesis.

```r
MYOWNBUTTON <- gtkButtonNewWithMnemonic("MY OWN FUNCTION", show = TRUE)
gSignalConnect(MYOWNBUTTON, "clicked", MyHeatmapConn)
the.buttons$packStart(MYOWNBUTTON, fill = FALSE)
```

3 - Finally, Copy the following code

```r
MyHeatmapConn <- function(button, user.data) {
  res = NULL
  # Get the information about data and the file
  the.file <- filename$getText()
  the.sep <- sepEntry$getText()
  the.headers <- headersEntry$active
  the.geneNum <- geneNum$getText()
  d <- read.table(the.file, sep = the.sep, header = the.headers, row.names = 1)
  # Select numerical variables
  numVar <- sapply(1:ncol(d), function(x) {is.numeric(d[, x])})
  if (sum(numVar) == 0) {
    error = "ERROR: No numerical variables in the data!"
  } else {
    res = MyHeatmap(d, the.geneNum) # HERE YOU CALL THE FUNCTION YOU DEFINED!
  }
}
```

71
Figure 55: A new button called **MY OWN FUNCTION** is created and paste it before the two following lines below that are written inside the `calculateGUI1.R` file.

```r
# Create window
tab <- gtkWindow()
```

At this point, **MY OWN FUNCTION** button is created and the result is the one shown in Figure 55. By clicking this button, we call `MyHeatmapConn` function that calls `MyHeatmap` function defined before.
To see the versions of the used methods, we type

```
sessionInfo()
```

and we get the list shown in Figure 56.
17 Errors/Warning/Bugs

17.1 Read Count Interface Errors

17.1.1 Error in data.frame(...

> Error in data.frame(..., check.names = FALSE) :
> arguments imply differing number of rows:

This Error has been corrected in the latest version of RNASeqGUI. Therefore, download the new version of RNASeqGUI (i.e. RNASeqGUI_0.99.2.tar.gz).

Anyway, This error is caused since the Results folder inside your project folder is NOT empty at the time you click the Read Count button. Please, be sure that the Results folder is empty before using the Count Section and it should run properly.

17.1.2 Warning messages: In .deduceExonRankings(exs... 

> Warning messages:
> In .deduceExonRankings(exs, format = "gtf") :
> Infering Exon Rankings. If this is not what you expected, then
> please be sure that you have provided a valid attribute for
> exonRankAttributeName

This happens when in the provided GTF file there is no exon ranking information. Therefore, the only way to get exon rank information is by deducing it based on the provided coordinate positions. This inference task can be performed by the parser, but it takes time to be completed. Moreover, the parser makes assumptions on your data. Hence, it is better to avoid it when possible. That’s why the deduceExonRankings function is throwing a warning about the exon ranking inference process.
Acknowledgement

We want to thank M. Franzese, V. Costa and R. Esposito for suggestions and discussions, D. Granata for technical support.

Thanks to D. Righelli for reporting bugs and version control.

This work was supported by the Italian Flagship InterOmics Project (PB.P05) and by BMBS COST Action BM1006.
References


