DAPAR and *ProStaR* user manual

Samuel Wieczorek*, Florence Combes, Alexia Dorffer, Thomas Burger

October 14, 2015

Abstract

DAPAR (Differential Analysis of Protein Abundance with R) and *ProStaR* (Proteomics and Statistics with R) are two Bioconductor packages that contain the necessary functions to analyze proteomics data (*DAPAR*), as well as the corresponding graphical user interfaces (*ProStaR*). This document guides the practitioner through the use of *DAPAR* (R command lines) and *ProStaR* (click-button interface, so that no programming skill is required).

*samuel.wieczorek@cea.fr

Contents

1 Introduction

DAPAR and *ProStaR* are a series of software dedicated to the processing of proteomics data. More precisely, they are devoted to the analysis of quantitative datasets produced in bottom-up discovery proteomics with a LC-MS/MS pipe-line (Liquid Chromatography and Tandey Mass spectrometry).

DAPAR (Differential Analysis of Protein Abundance with R) is an R package that contains all the necessary functions to:

- Import/export a quantitative dataset. Here, a quantitative dataset denotes a table where each protein is represented by a line and and each replicate is represented by a column; each cell of the table contain the abundance of a given protein in a given sample; the replicates are clustered into different conditions (or groups), and the purpose of the analysis is to isolate the few proteins the abundance of which significantly differ between the conditions (or groups).
- Compute and display meaningful statistics regarding the quantitative dataset.
- Perform the various processing steps of a complete data analysis: (i) filtering and data cleaning; (ii) cross-replicate normalization; (iii) missing value imputation; (iv) statistical tests and false discovery rate computation.

This package can be used on its own; or as a complement to the numerous Bioconductor packages (https://www.bioconductor.org/) it is compliant with; or through the *ProStaR* interface. *ProStaR* (Proteomics and Statistics with R) is a web-interface based on Shiny (http://shiny.rstudio.com/) that provides Graphical User Interfaces (GUI) to all the *DAPAR* functionalities, so as to guide any practitioner that is not comfortable with R programming through the complete data analysis process.

2 Installation

There are 3 ways to use *DAPAR*:

- The first one is to use *DAPAR* alone, through command lines or scripts. To do so, the user simply has to install *DAPAR* on his/her own work station, as instructed in Section 2.1;
- The second one is to use *DAPAR* along with its graphical interface *ProStaR*, and to have them running on the user's station (referred to as stand-alone install). In such case, it necessary to install *DAPAR* first, as instructed in Section 2.1, and *ProStaR* then, as instructed in Section 2.2.1;
- In the case where several *ProStaR* users who are not confortable with R (programming or installing), it is best to have a single version of *DAPAR* and *ProStaR* running on a Unix/Linux server. The users will use *ProStaR* through a web browser, exactly if it were locally installed, yet, a single install has to be administrated. In that case, *DAPAR* has to be classically installed (Section 2.1), while on the other hand, the install of *ProStaR* is slightly different on a server (Section 2.2.2).

For a stand-alone use, both *DAPAR* and *ProStaR* can run on any operating system (Unix/Linux, Mac OS X and Windows) as long as R is installed. In any case (stand-alone or server), a recent version of R ($\geq$ 3.2) is needed.

2.1 DAPAR

To install the package *DAPAR* from the source file with administrator rights, start R and enter:

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("DAPAR")
```

This step will automatically install the following packages:

- From CRAN: *RColorBrewer*, *Cairo*, *png*, *lattice*, *reshape2*, *tmvtnorm*, *norm*, *ggplot2*, *imputeLCMD*, *gplots*, *XLConnect*, *knitr*
- From Bioconductor: *MSnbase*, *preprocessCore*, *impute*, *limma*, *pcaMethods*

2.2 DAPAR with ProStaR

ProStaR can be run in two differents ways: standalone or server. The pre-requested packages described above have to be installed on the server if the user run a shiny-server to distribute *ProStaR* or on a local machine if *ProStaR* is run locally.

2.2.1 Stand-alone version

To run the stand-alone version, it is necessary to install the package in a directory where the user have read/write permissions. If the user have administrator privileges, then in a R console, enter:

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("Prostar")
```

This step will automatically install the following packages: *shinyIncubator*, *shinySky*, *rhandsontable*, *shinyTree*.

Once the package is installed, to launch *ProStaR*, then enter:

```
> library(Prostar)
> Prostar()
```

A new window of the default web browser opens.

2.2.2 Server version

This version uses a Shiny Server (https://github.com/rstudio/shiny-server). It is a server program that makes Shiny applications available over the web. Please follow installation instructions if you do not have a server yet.

The first step is to install *Prostar* as described in section 2.2.1 in order to have the dependencies installed.

In the sequel, we suppose the directory of shiny-server is `/srv/shiny-server` (this is the default configuration of a Shiny Server). In the home directory, unzip the package tarball and move to the R directory.

```
# unzip Prostar_0.99.1.tar.gz
# cd Prostar/R
```

Create a directory named Prostar in the Shiny Server directory and then copy the 3 files: ui.R, global.R and server.R.

```
# sudo mkdir /srv/shiny-server/Prostar
# sudo cp R/ui.R /srv/shiny-server/Prostar/.
# sudo cp R/global.R /srv/shiny-server/Prostar/.
# sudo cp R/server.R /srv/shiny-server/Prostar/.
```

Then, complete the installation by copying the 'www' directory of *ProStaR*:

```
# sudo cp -R inst/extdata/www /srv/shiny-server/Prostar/.
```

Check if the configuration file of shiny-server is correct.
For more details, please visit http://rstudio.github.io/shiny-server/latest/.

Now, the application should be available via a web browser at http://*servername*:*port*/Prostar.

3 Navigating through the ProStaR interface

3.1 Overview of the interface

As illustrated on Fig. 1, the interface has a classic Shiny layout with two panels:

- **Left panel**: The menu to navigate throught the DAPAR functionalities and to run them;
- **Right panel**: The place where the inputs and outputs of the various processing are dispayed.

More precisely (see Fig. 2), the main menu on the upper part of left panel takes the form of a tree, which is divided into four submenus[1]:

- **Dataset manager**: contains the tools to import and export datasets;
- **Descriptive statistics**: provides different plots that are helpful to understand the dataset, and to picture the influence of the various processing;
- **Data processing**: This is the heart of ProStaR, as it contains the interfaces to DAPAR functions;
- **Help**: A serie of informations about the sofware, associated communications, etc.

[1]Please note that to expand or hide the content of the sub-menus ("Dataset manager" or "Data processing"), it is necessary to click on the tiny triangles on the left hand side, which depict the nodes of the tree.

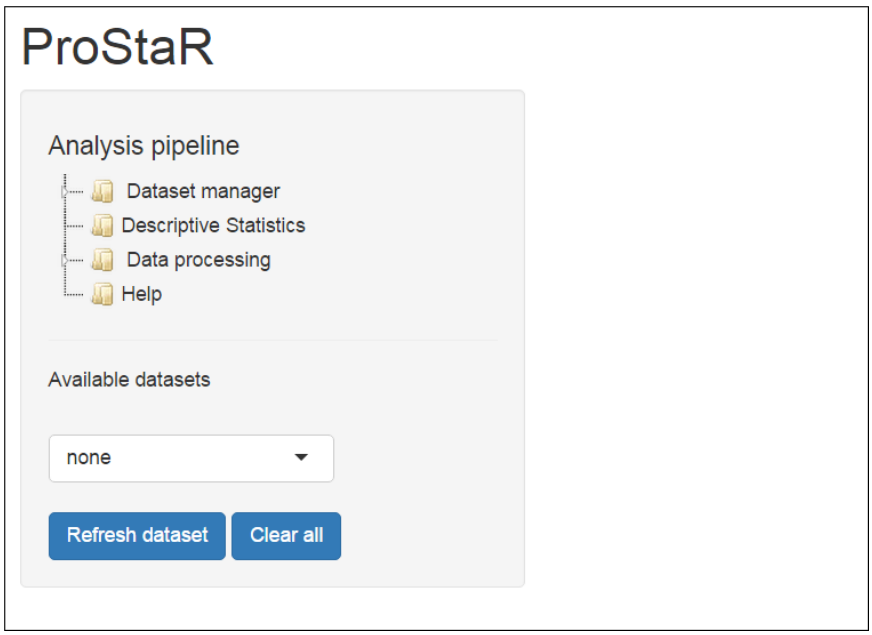


Figure 1: Default screen of ProStaR

Below the main menu, a drop-down menu referred to as "Available datasets" makes it possible to navigate back through the history of the processing. Its use is detailed in Section 3.6.

3.2 Data processing

To expand the "Data processing" menu, click on the small triangle on the left. It contains the 4 predefined steps of a data analysis. They are designed to be used in a specific order:

1. Filtering
2. Normalization
3. Missing values imputation
4. Differential analysis

For each step, several algorithms or parameters are available. They are detailed in Section 4. During each of these 4 steps, it is possible to test several options, and to observe the influence of the processing in the descriptive statistics menu (see Section 3.4), which is dynamically updated. Finally, once the ultimate tuning is chosen, the processing is applied to the dataset. In order to finalize the step, it is possible to save the result, so that another dataset appears in the "Available datasets" list (see Section 3.6).

3.3 Dataset manager

The "Dataset manager" allows the user to open, import or export quantitative datasets. *ProStaR* and *DAPAR* use the MSnSet format which is part of the package *MSnbase*. The user can load previously

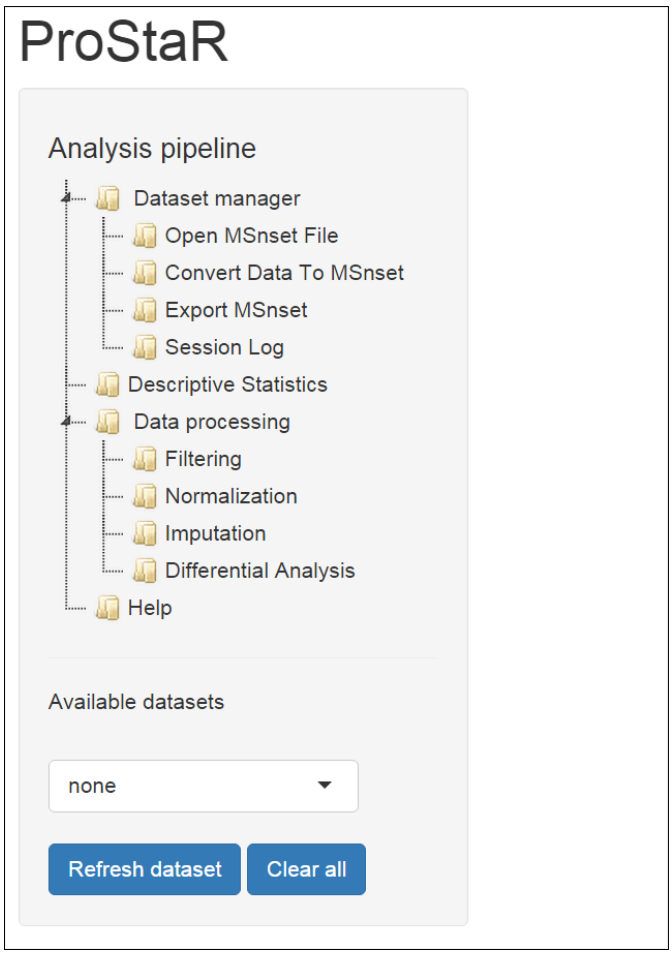


Figure 2: Detailed menu of ProStaR

existing MSnSet files (see Section 3.3.1) or text (-tabulated) files (see Section 3.3.2).

3.3.1 Open MSnset

The user can upload a dataset that is already formated as an MSnset file, by clicking on "OpenMSnset File" (see Fig. 3). This action opens a pop-up window, so as to let the user choose the appropriate file. Once the file is uploaded, a short summary of the dataset is shown, which includes the number of samples, the number of proteins in the dataset, the percentage of missing values and the number of lines which only contain missing values.

Once done, all the plots in the "Descriptive statistics" submenu (see Section 3.4) become accessible and all the widgets to interact with *ProStaR* are preloaded.

Command line: It is possible to open an MSnset dataset directly in command line (*i.e.* without *ProStaR* interface), using function `readRDS()`.

The user can find an example of MSnset file in the installation directory of *DAPAR* in:
`inst/extdata/UPSprotx2.MSnset`

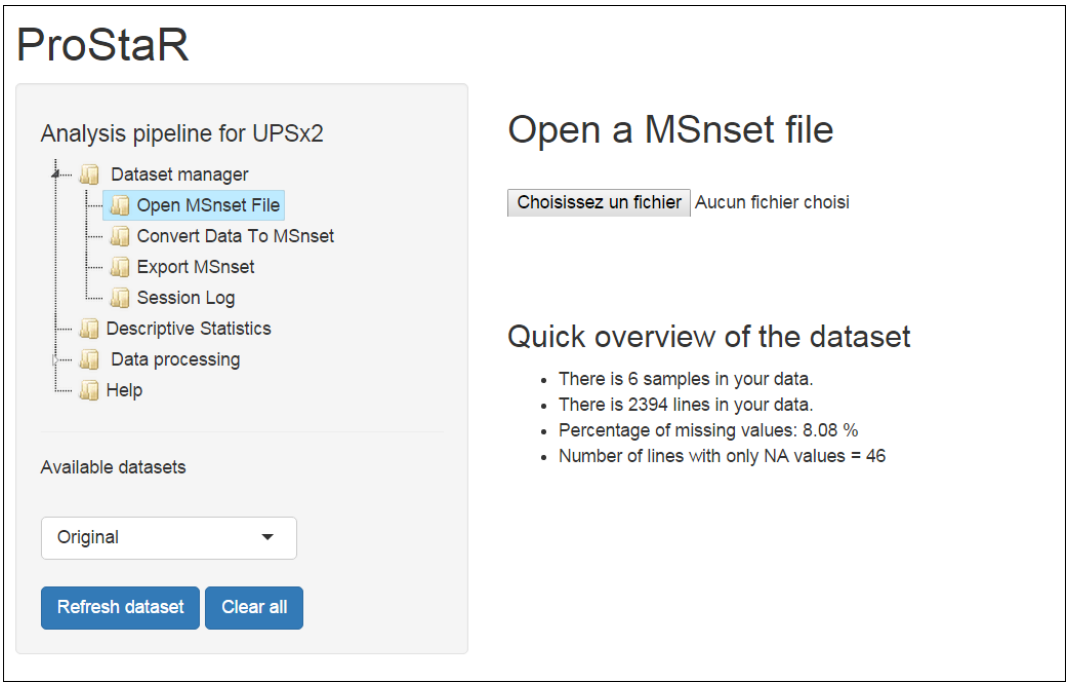


Figure 3: Open a MSnSet file

3.3.2 Import data

Alternatively, the user can create a quantitative dataset in the MSnset format, on the basis of CSV (Comma Separated Values) files that contains the results of a proteomics analysis. To do so, one has to click on "Convert data to MSnset". Then, the right panel splits into 5 tabs that guide the user through the various creation of the MSnset object:

Select file: Select the CSV file to import (see Fig. 4). This file must contain a table where each line corresponds to a protein, except the first one which must contain the names of the colums. Among the colums, one must contain an ID that uniquely defines the proteins, as well as a series of columns containing the abundance values (either log-transformed or not). As it appears in Fig. 4, some options allows for the log2 transformation the abundance values, as well as for automatically replacing the 0 and NaN values by **NA**.

Data ID: A drop-down menu provides the list of the column names. Select the column corresponding to the unique ID of the proteins (see Fig. 5).

Exp. and Feat. data: In the "Quantitative data" list, select the columns that correspond to the quantitative data. Each time the user selects an item in the list, it is moved up to the field above (see Fig. 6). If an item is selected by mistake, it can be removed by pressing on the SUPPR key.

Sample metadata: In this tab, the user fills the informations related to the samples. The colum named *Experiment* is filled by default with the name of the different samples. The user fills the other columns: *Label* correspond to the conditions of the experiment that will be compared during the differential analysis; *Bio.Rep*,*Tech.rep* et *Analyt.Rep.* correspond respectively to the biological, technical and analytical replicates (Fig. 7). The column Label is mandatory (for the subsequent differential analysis),

Figure 4: Importing a CSV file, tab 1.

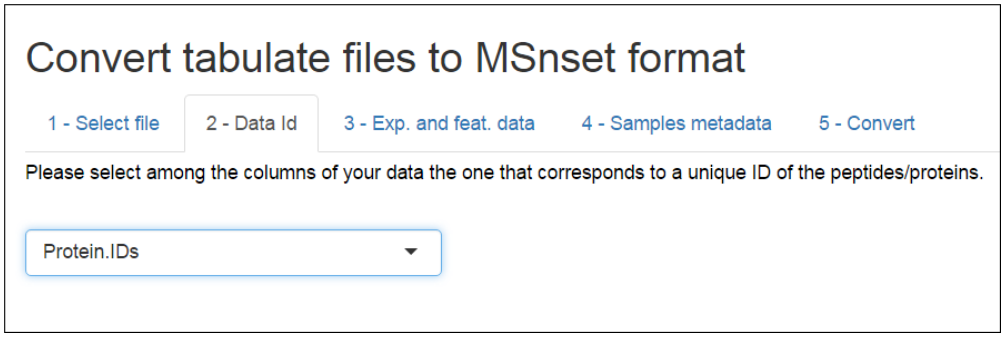


Figure 5: Importing a CSV file, tab 2.

the other ones are optional.

Convert: Finally, enter the name of the MSnset to be created (Fig. 8) and click on "Convert data". The data are converted and automatically loaded in *ProStaR*. The name of the file appears on the top of the left panel, above the menu.

Command line: In *DAPAR*, the function to create an MSnset from a CSV file is `createMSnset()`.

3.3.3 Export

Once an MSnset has been created, it is possible to save it as an MSnset binary object (so that next time, it is not nessary to create it, and a simple uploads makes it, as described in Section 3.3.1). It is also possible to export it as an Excel spreadsheet. To do so, one simply goes on the corresponding tab and select the appropriate option.

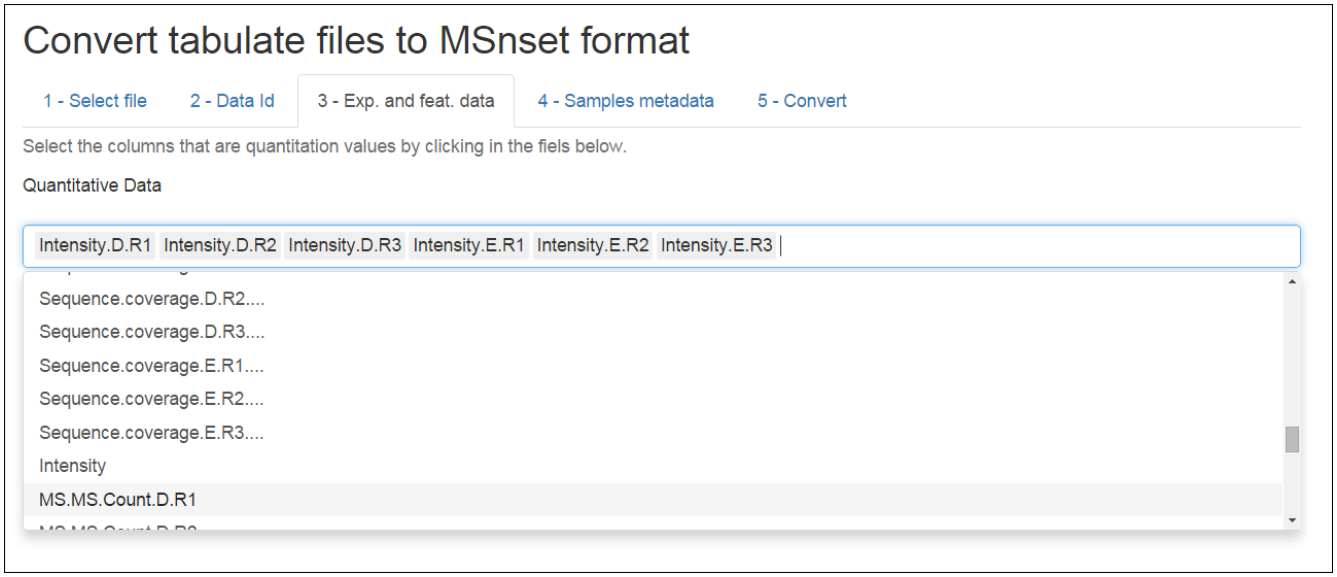


Figure 6: Importing a CSV file, tab 3.

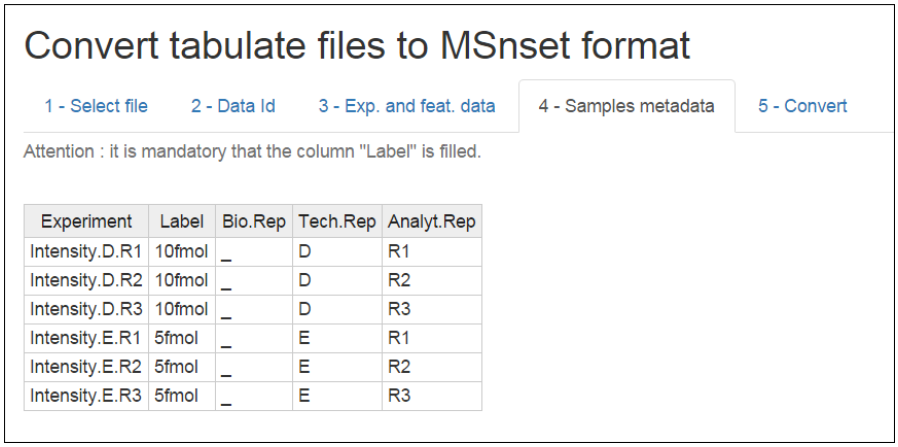


Figure 7: Importing a CSV file, tab 4.

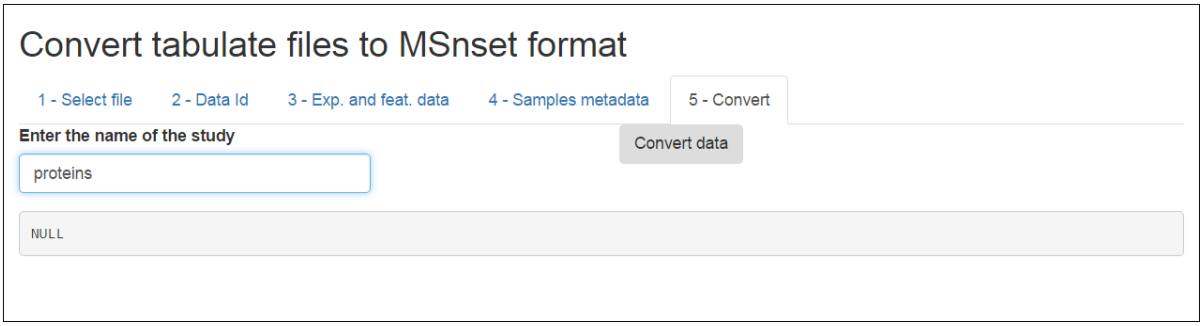


Figure 8: Importing a CSV file, tab 5.

Command line: When working exclusively with *DAPAR*, the functions are `writeMSnsetToExcel()` (to export in Excel format) and `saveRDS()` (to export in MSnset format).

3.3.4 Session log

Each time the user validates a processing step (by clicking on the "Save <*the_step*>" button, see Section 4), the entire related information (such as the method name and its parameters) is added to the table shown in the "Session log" tab (see Figure 9). Hence, this table is a history of how the data were processed during the session. Let us note that, if a dataset is processed, then saved and reloaded in a new session, the session log is naturally empty. To have a complete view on the previous processing applied to a given dataset, pelase refer to Section 3.4.2).

3.4 Descriptive statistics

Several plots (one plot per tab) are proposed to help the user to have a quick and as complete as possible overview of his/her dataset. This menu is an essentiel element for the user to check that each processing step indeed gave the expected result.

3.4.1 Missing value summary

The barplot on the left represents the number of missing values in each sample. The different colors correspond to the different conditions (or label). The histogramm on the right displays the distribution of missing values. The red bin counts the protein lines that only contains missing values (Fig. 10).

Command line: In *DAPAR*, the functions for these two plots are `mvPerLinesHisto()` and `mvHisto()`.

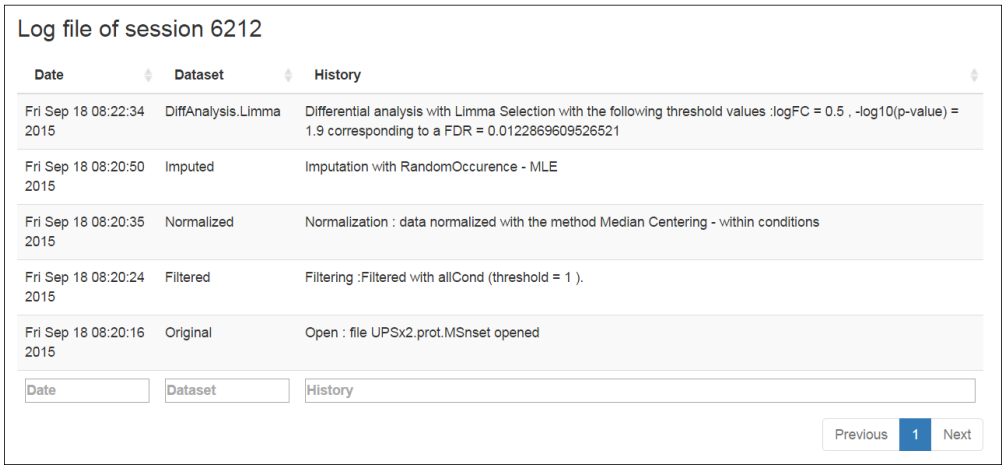


Figure 9: Example of the log of a session in ProStaR

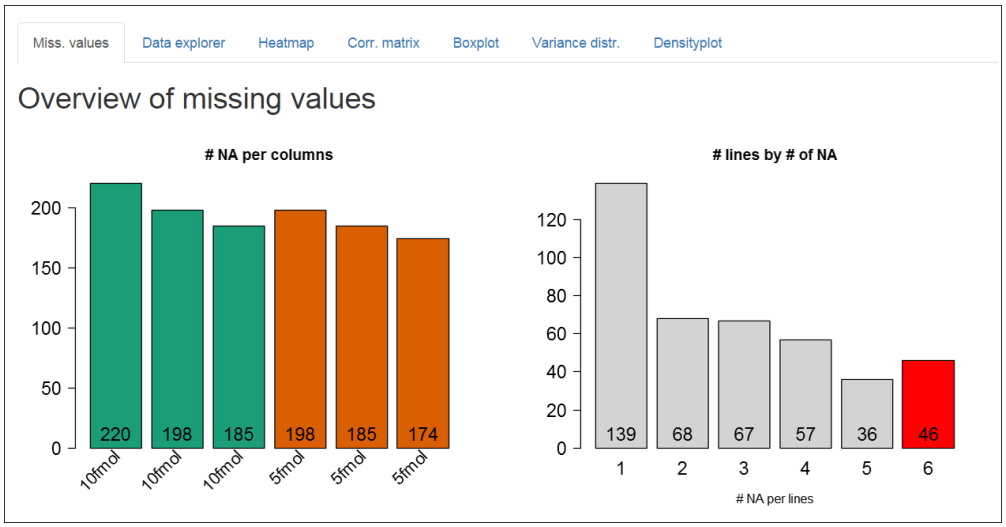
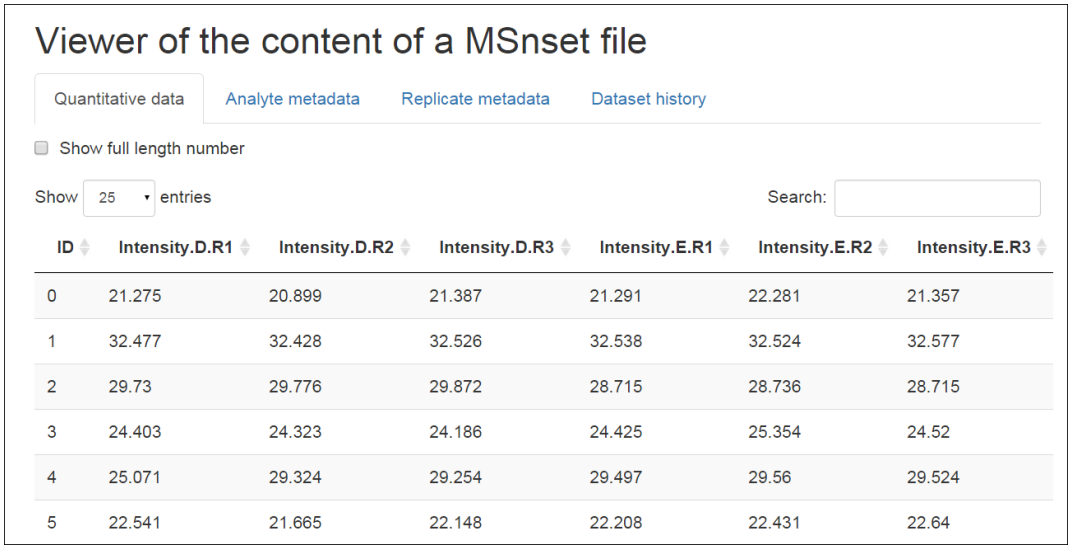


Figure 10: Histrograms for the overview of the missing values



ID	Intensity.D.R1	Intensity.D.R2	Intensity.D.R3	Intensity.E.R1	Intensity.E.R2	Intensity.E.R3
0	21.275	20.899	21.387	21.291	22.281	21.357
1	32.477	32.428	32.526	32.538	32.524	32.577
2	29.73	29.776	29.872	28.715	28.736	28.715
3	24.403	24.323	24.186	24.425	25.354	24.52
4	25.071	29.324	29.254	29.497	29.56	29.524
5	22.541	21.665	22.148	22.208	22.431	22.64

Figure 11: View of quantitative data in the MSnSet dataset

3.4.2 Data explorer

This panel allows viewing the content of the msnset structure. It is made of four tables, that are represented in a tab each. The first one, named "Quantitative data" contains quantitative values (see Fig. 11). The missing values are represented by empty cells.

The second tab is named "Analyte metadata". It contains the metadata of the proteins (see Fig. 12).

The thrid tab is named "Replicate metadata". The information displayed here is the one entered by the user during the import step (see Fig. 13).

The last tab, named "Dataset history" contains the log of the previous processing. Contrarily to the "Session log" panel (see Section 3.3.4), the information here does not relate to the session, and is saved

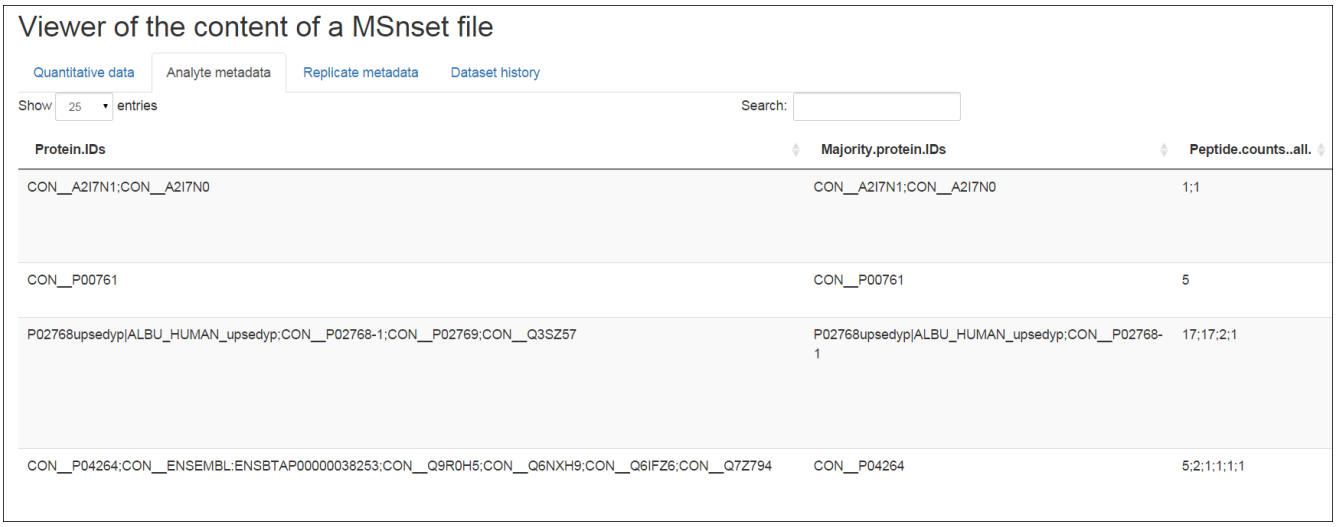


Figure 12: View of feature meta-data in the MSnSet dataset

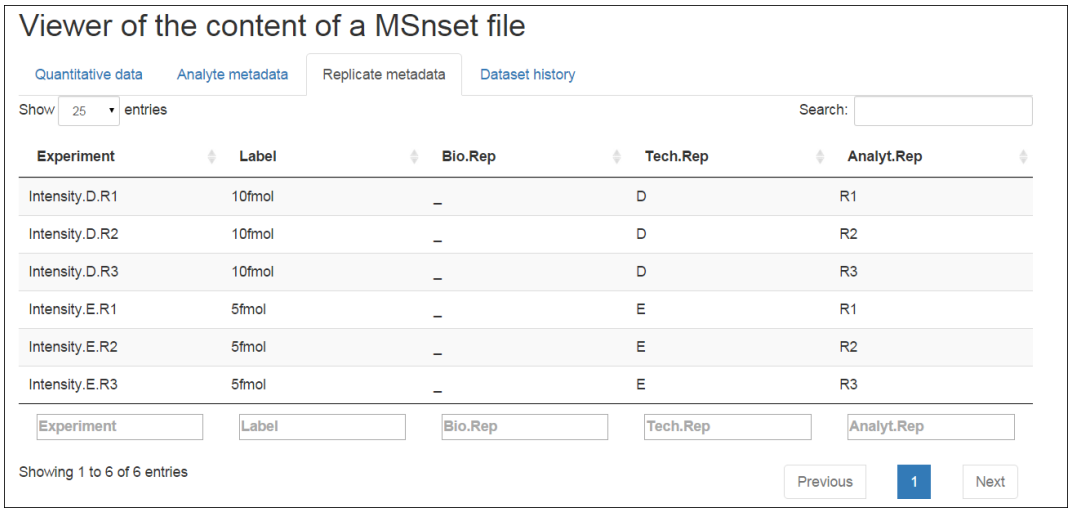


Figure 13: View of samples meta-data in the MSnSet dataset

from a session to the next one.

Command line: The *DAPAR* functions to get the three first tables are in fact those from the *MSnbase* package: `exprs()` (Quantitative data), `fData()` (Analyte metadata) and `pData()` (Replicate meta-data). Similarly, the "Dataset history" information is also accessible. In fact, it is stored in a specific slot (`processingData@processing`) of the current MSnSet object. In a R console, if `obj` is the current dataset, it can be accessed by entering:

```
> obj@processingData@processing
```

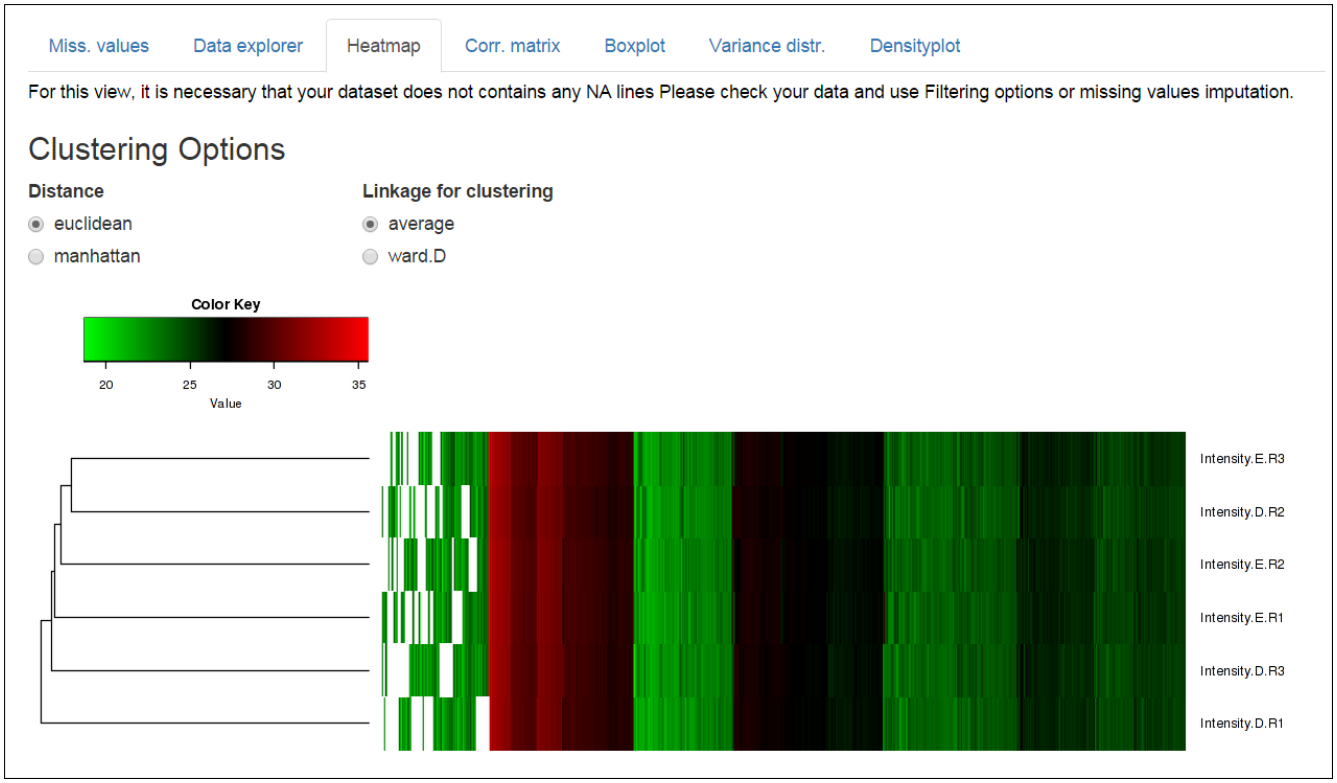


Figure 14: Heatmap and dendrogram for the quantitative data.

3.4.3 Heatmap

A heatmap is drawn with the associated dendrogram (see Fig. 14). The colors represent the intensities: red for high intensities and green for low intensities. White color is reserved for missing values. The dendrogram shows the hierarchical classification of the samples. This classification can be tuned by two parameters:

- **Distance**: Euclidean or Manhattan
- **Linkage**: Ward.D or mean

Command line: In *DAPAR*, the corresponding function is `heatmapD()`.

3.4.4 Correlation matrix

In this tab, it is possible to visualize the extent to which the replicates correlate or not (see Fig. 15).

Command line: In *DAPAR*, the corresponding function is `corrMatrixD()`.

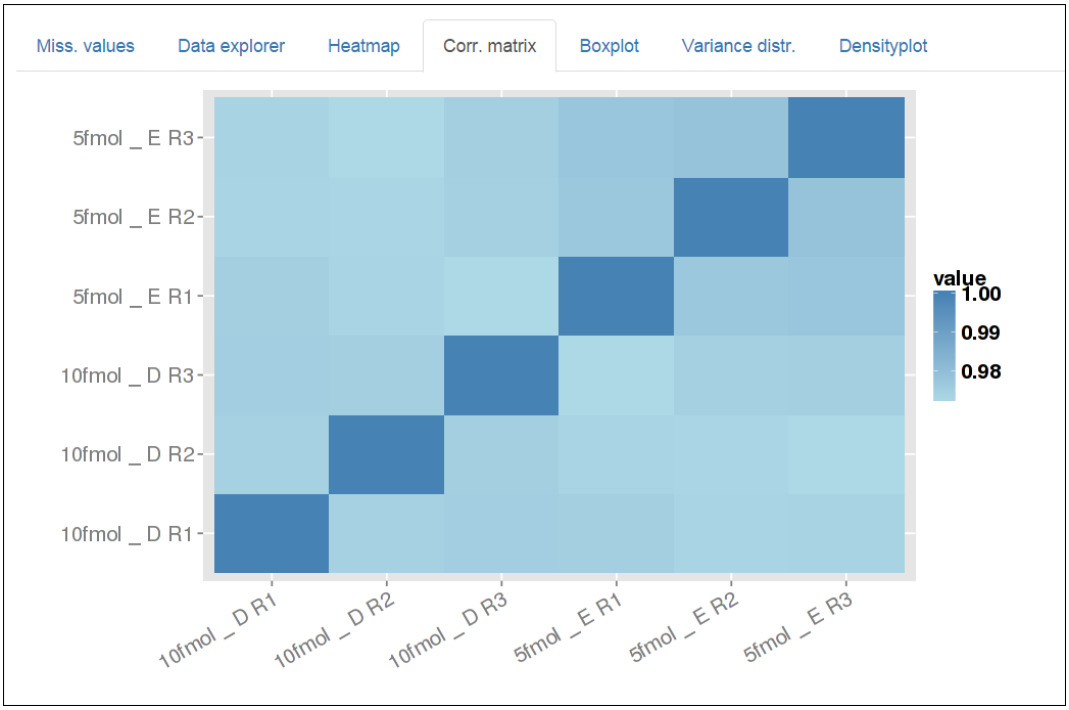


Figure 15: Correlation matrix for the quantitative data.

3.4.5 Boxplot

The protein distribution by replicates is summarized with boxplots (see Fig. 16). The user can change the legend of the samples (X-axis) by checking items in the checkboxes group. The colors of the boxes correspond to the different conditions (column **Label** in the table of *Samples Meta Data*).

Command line: In *DAPAR*, the corresponding function is `boxPlotD()`.

3.4.6 Variance distribution

This plot shows the distribution of the variance of the log-intensity of proteins for each condition (see Fig. 17).

Command line: In *DAPAR*, the corresponding function is `varianceDistD()`.

3.4.7 Density plot

This plots shows the distribution of the log-intensity of proteins for each condition (see Fig. 18).

Two options are available to custom the plot. Theye useful when to many lines superimpose:

- **Highlight a condition** among the others. By default, no condition is highlighted,
- **Show conditions** Select the conditions to display. By default, all the conditions are showed.

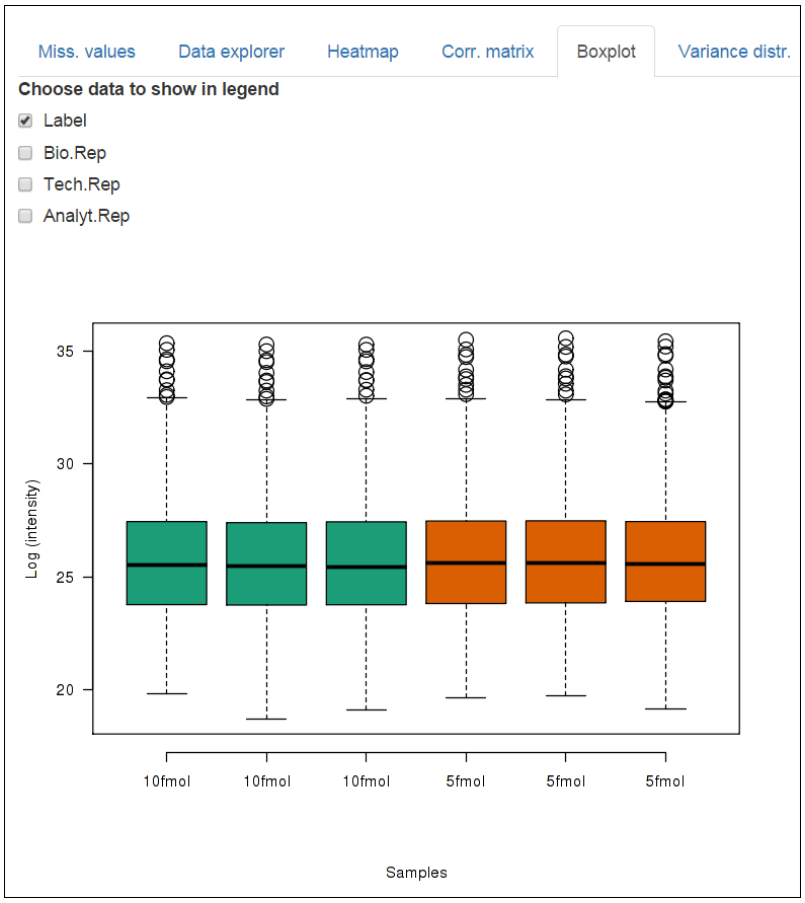


Figure 16: Boxplot for the quantitative data.

Command line: In *DAPAR*, the corresponding function is `densityPlotD()`.

3.5 Help

The Help screen offers various information through three panels:

- **About**. This gives the version of the two packages *DAPAR* and *ProStaR* and a link to this document,
- **The MSnSet format**. On this screen, there is a link to an article about the MSnSet format in order to explain its architecture to the user,
- **Refs**. The references associated and/or related to the packages *DAPAR* and *ProStaR*.

3.6 Available datasets

A major element of the "Dataset manager" is detached from the menus, for it is convenient to have a continuous view on it. This the drop-down menu entitled "Available datasets" that lists between 1 and 6 different datasets, depending on the progress of the data analysis. Basically, each time the

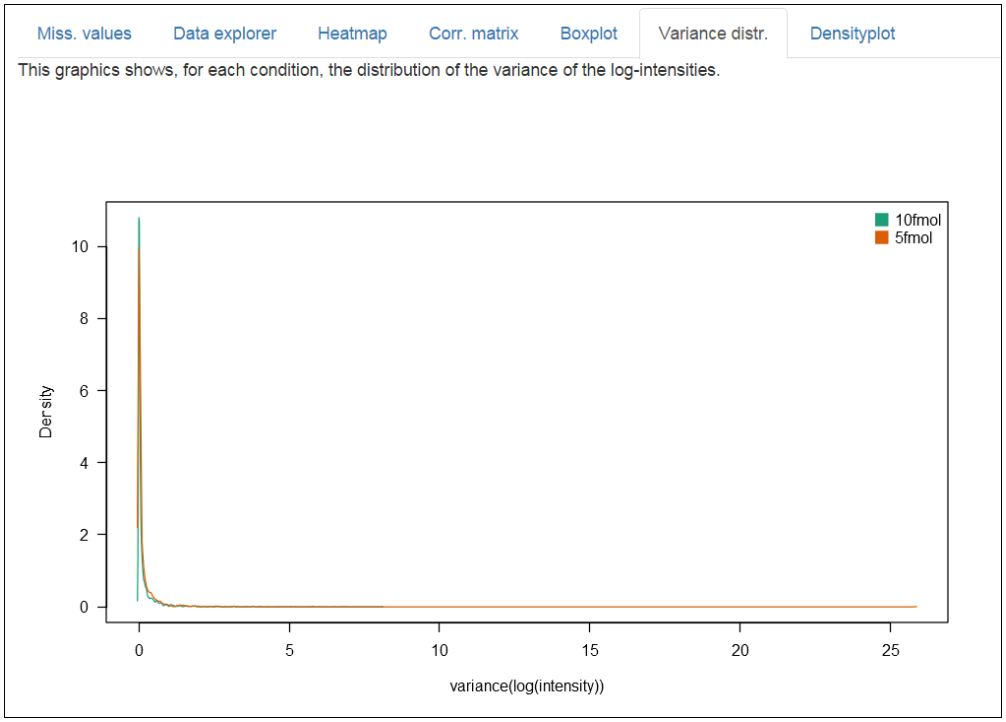


Figure 17: Variance distribution for the quantitative data.

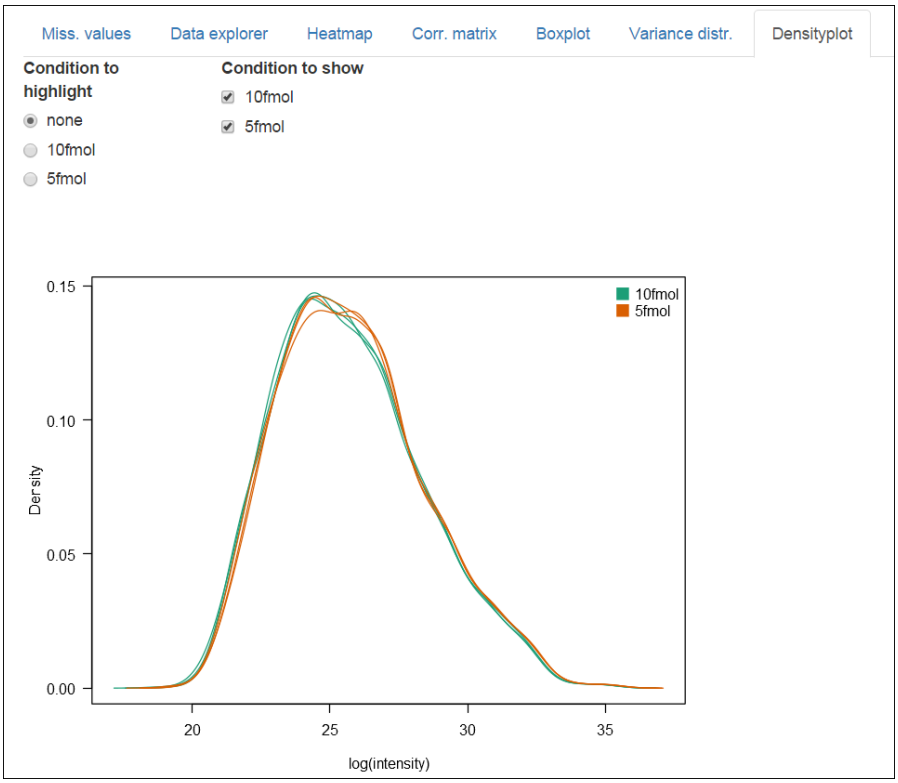


Figure 18: Densityplot the quantitative data.

modifications of the current dataset are saved, the new dataset does not overwrite the previous dataset. On the contrary, the different versions are stored in memory. Thus, *ProStaR* keeps a history of all processing performed on a dataset. Concretely, right after creating or uploading a dataset, only a single dataset is available: it is named "Original". After the filtering step, if the user saves his/her results, another dataset becomes available, named "Filtered". Similarly, after the saving of the normalization, of the imputation of missing values and the differential analysis, a new dataset is created and stored. Each time a new dataset is created, it is by default the one on which the processing goes on. However, the previous one is accessible through the "Available datasets" drop-down menu.

At any time, the name of the current dataset is displayed. If the user needs to return to a previous dataset (for example, the current dataset is "Imputed" and the user wants to return to "Filtered"), he/she chooses it in the select field, then click on "Refresh dataset". The dataset is then automatically loaded in memory and becomes the current one; the new dataset becomes the new current one. Naturally, all the plots that are displayed throughout the various panels of *ProStaR* are dynamically updated without any action from the user.

Remark:

- If the user chooses a dataset within those available, the dataset is not directly reloaded as the working one. To do so, it is mandatory to click on "Refresh dataset".
- Moreover, let us note that if the user saves the current step (let us say the imputation step), then goes back to a previous step (say the normalization step) and start working on this older dataset (for instance, by performing another imputation) and then saves it, the new version of the processing overwrites the previous version (the older imputation is lost and only the newest one is stored in memory): In fact, only a single version of the dataset can be saved for a given processing step.
- For a refined analysis regarding the influence of a processing step, it also possible to switch from an older to a newer dataset (that has been saved before) with the "Available dataset" drop-down menu, and to observe the variations in the "Descriptive statistics" menu.

The "Clear all" button deletes all the Available datasets.

4 Processing a dataset

In this section, the four steps of a quantitative data analysis are detailed. At the end of each step, it is advised to save the processing, so that another dataset shows up in the "Available datasets" menu, and so that it is possible to go back to a previous step of the analysis if necessary, without starting back the analysis from scratch.

After applying any processing, it is advised to check the "Descriptive Statistics" menu and to observe the influence of the processing (all the graphics are dynamically updated).

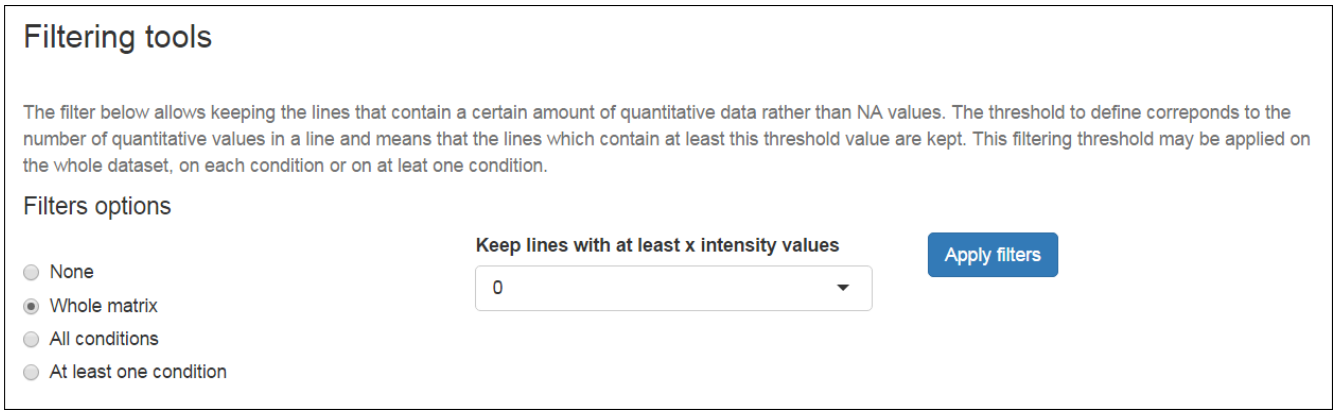


Figure 19: Interface of the filtering tool.

4.1 Filtering

In this step, the user may decide to delete proteins where the amount of missing values is too important to expect confident processing.

The choice of the lines to be deleted is made by different options (see Fig. 19):

- **None**: No filtering, the quantitative data is left unchanged. This is the default option;
- **Whole Matrix**: The lines (across all conditions) in the quantitative dataset which contain less non-missing value than a user-defined threshold are deleted;
- **All conditions**: The lines for which each condition contain less non-missing value than a user-defined threshold are deleted;
- **At least one condition**: The lines for which at least one condition contain less non-missing value than a user-defined threshold are deleted;

Once the the filtering is appropriately tuned, the user clicks on "Apply filter" so as to validate his/her choice and to apply it to the dataset. Then, a new dataset is created. The informations related to the type of filtering and the options choosen appear in the Session log tab (see section 3.3.4). This latter becomes the new current dataset and its name appears in the menu **Datasets available** at the bottom left of the screen. All plots and tables are automatically updated in *ProStaR*.

Command line: In *DAPAR*, the corresponding function is `mvFilter()`.

4.2 Normalization

The next step is to normalize the replicates so as to have more accurate comparisons. *ProStaR* offers a number of different normalization routines that are described below.

In order to vizualize the data after normalization, two plots are displayed: a boxplot and a densityplot (see Fig. 20). Those plots are the same as the one showed in **Descriptive Statistics**, thus they have the same options (see Sections 3.4.5 and 3.4.7).

If no normalization is necessary, it is possible to skip this step. If the user wants to compare the influence

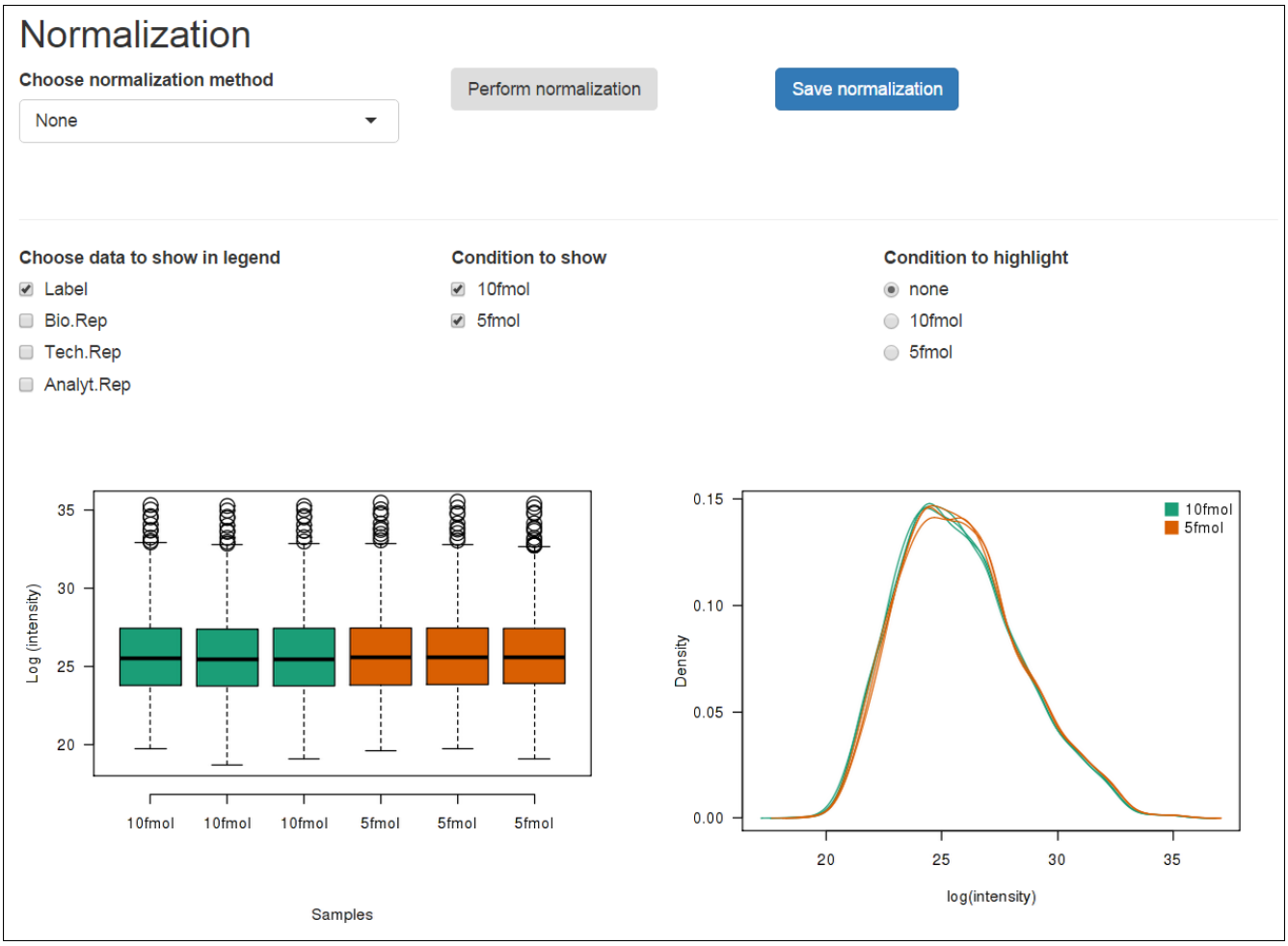


Figure 20: Interface of the normalization tool.

of several normalization methods, it is possible to select them in a row, and to alternate between this menu and the "Descriptive statistics" one. It is possible to go back to the original dataset by selecting "None". Several methods are implemented:

Sum by column The abundance of each protein is divided by the total abundance of all the proteins in the same replicates. This normalization is interesting to compare the proportions of a given protein in different samples that do not necessarily contain the same amount of biological material. Contrarily to the others, this normalization is not performed on the log2 scale, for it would not have any interpretation (the data are thus exponentiated and re-log2-transformed as pre-and post-processing).

Quantiles The protein abundance are roughly replaced by the order statics on their abundance (from package *preprocessCore*). This is the strongest normalization method available, and it should be use carefully, for it erazes most of the difference between the samples.

Mean / median centering The central tendancies of the samples are aligned. To do so, one computes first the central tendancy (either the mean of the median, depending on the user choice) for each replicates. Then, to each abundance value, one subtracts the corresponding central tendancy. Finally, one adds to this abundance value, an offset in order to find roughly back the original range

of values. Depending on the user's choice, this offset can be the mean of all the central tendancies, whatever the conditions (then, any global difference between the conditions will disappear); or it can be the mean of all the central tendancies within each conditions (then, any global difference between the conditions is preserved). Note that all these computations are performed on values that were originaly log2-transformed.

Mean centering and scaling The spirit of this normalization is the same as the previous one, yet, it is stronger, and it only applies to log2-tranformed abundance values that distributes roughly normaly for each sample. Basically, a mean centering as described above is applied. Then, the variance of the distribution is re-scaled to 1. Let us note that median centering is not really adapted to a rescaling the variance; this is why such combination of parameters is not available. Once again, the centering can operate over the entire dataset, or over each condition.

The user can vizualize the effect of a normalization method without changing the current dataset. If the normalization does not produce the expected effect, the user can test another one. To do so, one simply has to choose another method in the list and click on "Perform normalization". The plots are automatically updated. This action does not modify the dataset but offers a preview of the normalized quantitative data. The user can vizualize as many times he/she wants several normalization methods. Once he finds the correct one, he/she validates his/her choice by clicking on "Save normalization". Then, a new "normalized" dataset is created and loaded in memory. The method of normalization that has been used is added to the Session log tab (see section 3.3.4). It becomes the new current dataset and the name "Normalized" appears in "Available datasets". All plots and tables in other menus are automatically updated.

Command line: In *DAPAR*, the corresponding function is `normalizeD()`.

4.3 Imputation

Two plots are available in order to help the user to choose the right imputation method for his dataset (see Fig. 21).

The scatter plot on the left hand side displays the proteins in a space spanned by the mean abundance (x axis) and the number of missing values (y axis). Note that for each protein, as many points (of different colors) as conditions are displayed, for each condition is processed independently of the others. As a result, the maximum value on the y axis is given by the number of replicates in a condition (depending on the filtering step). Let us note that the points have been slightly jittered on the y axis to enhance a better visualization.

This plots indicates how the missing values are distributed over the range of intensity: if there are lots of missing values in the low intensity region (indicating a censoring mechanism produced the missing values) or if they are uniformly distributed.

The heatmap on the right hand side clusters the proteins according to their distribution of missing values across the conditions. Each line of the map depicts a protein. On the contrary, the columns do not depicts the replicates anymore, as the abundance values have been reordered so as to cluster the missing values together. Similarly, the proteins have been reordered, so as to cluster the proteins that have a similar amount of missing values distributed in the same way over the conditions. Each

Figure 21: Interface of the imputation of missing values tool.

line is colored so as to depicts the mean abundance value within each condition. This heatmap is also helpful to decide what is the main origin of missing values (random missingness or censoring of the low intensities).

The user can choose one of the several available imputation methods, depending on the type of missing values:

- If the missing values are mainly due to a censoring process of the low intensity proteins, it is advised to use the QRILC (Quantile Regression for the Imputation of Left Censored data) imputation method (function `impute.QRILC()` of from package *imputeLCMD*).
- Alternatively, if the missing values are roughly uniformly distributed, it is advised to use BPCA (Bayesian Principal Component Analysis) from package *pcaMethods*, KNN (K Nearest Neighbors) from package *impute* or MLE (Maximum Likelihood Estimation) from package *norm*.

The user can vizualize the effect of an imputation method without changing the current dataset. If the imputation does not produce the expected effect, the user can test another one. To do so, one simply has to choose another method in the list and click on "Perform imputation". The plots are automatically updated. This action does not modify the dataset but offers a preview of the imputed quantitative data. The user can vizualize as many times he/she wants several imputation methods. Once he finds the correct one, he/she validates his/her choice by clicking on "Save imputation". Then, a new "imputed" dataset is created and loaded in memory. The method of imputation used is added to the Session log tab (see section 3.3.4). This new dataset becomes the new current dataset and the name "Imputed" appears in "Available datasets". All plots and tables in other menus are automatically updated.

Command line: In *DAPAR*, the function used to impute the missing values is `mvImputation()`. The two aforementioned plots are obtained with the functions `mvTypePlot()` and `mvImage()`, respectively.

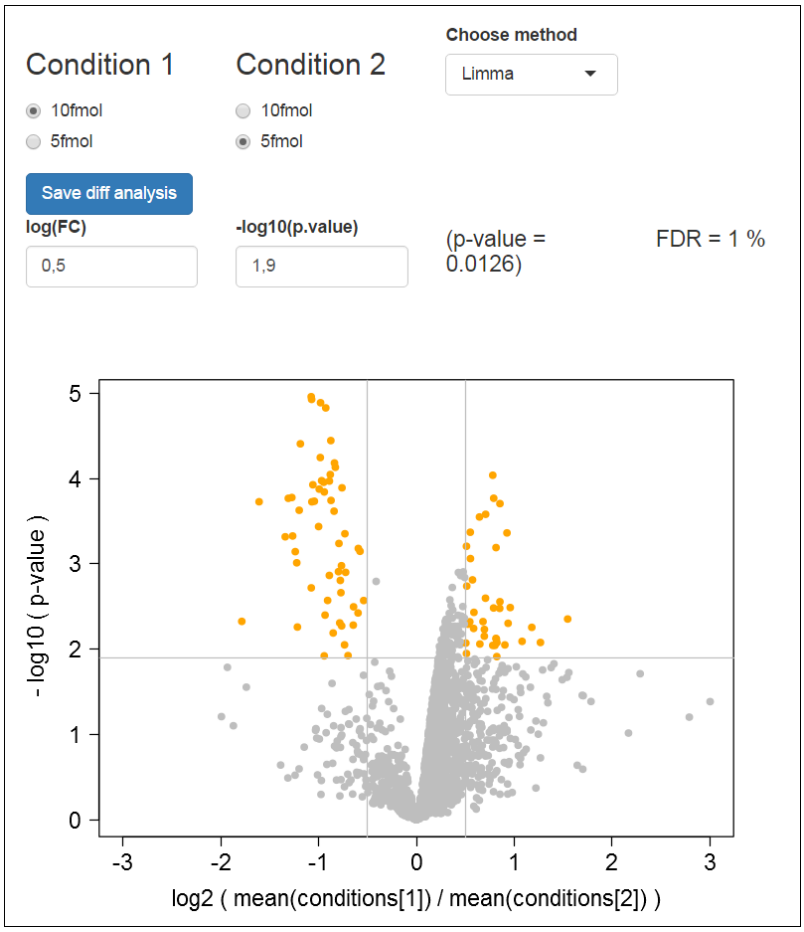


Figure 22: Volcanoplot of the differential analysis tool.

4.4 Differential analysis

This step cannot be conducted if the dataset still contains some missing values: They must be imputed before.

Two statistical tests are available in *DAPAR*: the Welch t-test (from package *stats*) and the moderated t-test (from package *limma*).

First, the user chooses the test (see Fig. 22). As an option it is possible to redefined the sets of conditions that are tested one against the other. Then, the p-values are computed and a volcanoplot is displayed. It shows on the x axis the Fold Change (FC) between the two conditions, and on the y axis, -log10(p-value).

Two thresholds 'one on each axis) can be tuned by the used, so as to discriminate the differentially abondant proteins (which are colored in orange). Two straight lines (resp. horizontal and vertical) are drawn to vizualize these thresholds.

The False Discovery Rate (FDR) is computed on the basis of selected proteins in the volcanoplot. The user can adjust the thresholds in order to select the maximum of proteins by minimizing the FDR.

id	P.Value	logFC
2	1.185783e-05	-1.0708460
7	4.349170e-04	0.9257492
10	1.334963e-04	-0.9932521
11	9.744355e-04	-1.2229159
12	6.618736e-04	-0.5937294
13	1.369285e-03	-0.8898587
14	1.490641e-05	-0.9265229
16	4.829030e-04	-1.3405611
17	1.709669e-04	-1.3090715

Figure 23: Table of the results of statistical test in the differential analysis tool.

Below the volcanoplot, a table shows the results of the statistical test (see Fig. 23): the value of -log10(p-value) and the Fold Change (*i.e.* the log2 of the ratio of the mean values per condition).

When the user has selected the proteins of interest, he/she can save them by clicking on "Save diff analysis". Then, a new "AnaDiff" dataset is created and loaded in memory. This dataset is the same as the previsous one except that 3 columns have been added in the "Quantitative data" table: "-log10(p-value)", "Fold Change" and "Significant". The two first contain the coordinates of the proteins on the volcano plot, and the third one contains a boolean value indicating whether each protein is differentially abondant or not. As with the other processing steps, the information related to the user's choices is added to the "Session log" tab (see section 3.3.4) of this new dataset. It becomes the new current dataset and its name, "DiffAnalysis.<test>" (where <test> indicates the test performed), appears in "Available datasets". All plots and tables in other menus are automatically updated.

Command line: The *DAPAR* functions for the Welch t-test and moderated t-test are `diffAnaWelch()` and `diffAnaLimma()`, respectively. These functions return a `data.frame` which contains 2 columns: the p-values and the Fold Change of the test. These columns can be added to the current MSnSet object `imputed_dataset` (as explained earlier) with the function `diffAnaSave()`:

```
> res <- diffAnaLimma(imputed_dataset, condition1, condition2)
> obj <- diffAnaSave(imputed_dataset, res, "limma", condition1, condition2)
```

Moreover, `diffAnaSave()` adds the aforementioned third column named "Significant" to the MSnset object. Two optional arguments allows the user defining the thresholds on the p-values and on the Fold Change, so has to be more or less stringent on the number of proteins called "Significant".

5 Session information

- R version 3.2.2 (2015-08-14), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`

- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Loaded via a namespace (and not attached): BiocStyle 1.8.0, knitr 1.11, tools 3.2.2