DAPAR and *ProStaR* user manual

***Samuel Wieczorek*[*], *Florence Combes*[*], and *Thomas Burger*[*]**

[*]firstname.lastname@cea.fr

April 10, 2021

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).

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 pipeline (Liquid Chromatography and Tandey Mass spectrometry). The experiment package *DAPARdata* is the companion package for *ProStaR* and *DAPAR*. It contains many datasets that can be used as example.

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 contains 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 quantitative analysis: (i) filtering and data cleaning; (ii) cross-replicate normalization; (iii) missing value imputation; (iv) aggregation of peptide intensities into protein intensities; (v) statistical tests and false discovery rate computation; (vi) Gene Ontology (GO) analysis (grouping and category enrichment).

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 quantitative analysis process.

In *DAPAR*, a serie of functions may be called from two types of variables: a dataframe that contains quantitative data and an object of class MSnSet. The first case has been developped to make *DAPAR* easier to use (in command line) for users who do not want to work with MSnSet files and to be compliant in the future with the Proline Suite (http://proline.profiproteomics.fr/).

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 workstation, as instructed in Section 2.2;
- 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 is necessary to install *DAPAR* first, as instructed in Section 2.2, and *ProStaR* then, as instructed in Section 2.1.1;
- 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 shiny server installed 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 2.2), while on the other hand, the installation of *ProStaR* is slightly different on a server (Section 2.1.2).

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 (≥ 3.4) is needed.

2.1 ProStaR (with DAPAR)

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.

2.1.1 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:

```
> install.packages("BiocManager")
> BiocManager::install("Prostar")
```

This step will automatically install the following packages: *shiny*, *rhandsontable*, *DAPARdata*, *DAPAR*, *data.table*, *DT*, *shinyjs*, *openxlsx*, *shinyAce*, *highcharter*, *rhandsontable*, *htmlwidgets*, *webshot*.

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

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

A new window of the default web browser opens.

2.1.2 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 2.1.1 in order to have the dependencies installed.

Then, execute the following line in order to get the install directory of Prostar:

```
> installed.packages()["Prostar","LibPath"]
```

Change the owner of the shiny-server directory and log as shiny

```
# sudo chown shiny /srv/shiny-server
# su shiny
```

Create a directory named *ProStaR* in the Shiny Server directory with the user shiny as owner and then copy the Prostar files.

```
# mkdir /srv/shiny-server/Prostar
# chown -R shiny /srv/shiny-server/Prostar
# cp -R prostarDir/ProstarApp/* /srv/shiny-server/ProStaR/.
```

Then, complete the installation by copying the 'www' directory of *ProStaR* and giving the following permissions:

```
# cp -R inst/extdata/www /srv/shiny-server/Prostar/.
# chmod 755 /srv/shiny-server/Prostar/www
```

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.

2.2 DAPAR (alone)

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

```
> install.packages("BiocManager")
> BiocManager::install("DAPAR")
```

This step will automatically install the following packages:

- From CRAN: *RColorBrewer*, *Cairo*, *png*, *lattice*, *reshape2*, *tmvtnorm*, *norm*, *ggplot2*, *imputeLCMD*, *gplots*, *openxlsx*, *knitr*, *cp4p*, *doParallel*, *foreach*, *scales*, *stats*, *grDevices*, *vioplot*, *sm*, *graphics*, *utils*, *parallel*, *openxlsx*, *Matrix*, *highcharter*, *dplyr*, *tidyr*, *imp4p*, *readxl*, *lme4*,*graph*
- From Bioconductor: *MSnbase*, *DAPARdata*, *preprocessCore*, *impute*, *limma*, *pcaMethods*, *clusterProfiler*, *AnnotationDbi*, *siggenes*

2.3 DAPARdata

The installation of the package *DAPARdata* follows the classical way for Bioconductor packages. In a R console, enter:

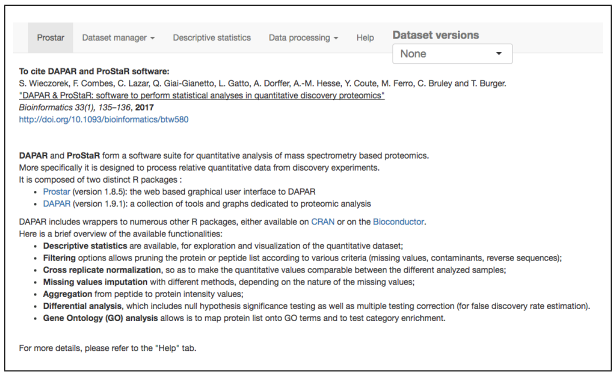
```
> install.packages("BiocManager")
> BiocManager::install("DAPARdata")
```

3 Navigating through the ProStaR interface

3.1 Overview of the interface

As illustrated on Fig. 1, *ProStaR* proposes a classical Graphical User Interface (GUI) to visualize and interact with the data. On the top, a navbar menu helps navigating through the various *DAPAR* functionalities and running them. It is divided into five submenus:

- **ProStar**: The welcome page, depicted on Fig. 1.
- **Dataset manager**: It contains the tools to import and export datasets;



To cite DAPAR and ProStaR software:
S. Wieczorek, F. Combes, C. Lazar, Q. Giai-Gianetto, L. Gatto, A. Dorffer, A.-M. Hesse, Y. Coute, M. Ferro, C. Bruley and T. Burger.
"DAPAR & ProStaR: software to perform statistical analyses in quantitative discovery proteomics"
Bioinformatics 33(1), 135–136, **2017**
http://doi.org/10.1093/bioinformatics/btw580

DAPAR and **ProStaR** form a software suite for quantitative analysis of mass spectrometry based proteomics.
More specifically it is designed to process relative quantitative data from discovery experiments.
It is composed of two distinct R packages :

- Prostar (version 1.8.5): the web based graphical user interface to DAPAR
- DAPAR (version 1.9.1): a collection of tools and graphs dedicated to proteomic analysis

DAPAR includes wrappers to numerous other R packages, either available on CRAN or on the Bioconductor.
Here is a brief overview of the available functionalities:

- **Descriptive statistics** are available, for exploration and visualization of the quantitative dataset;
- **Filtering** options allows pruning the protein or peptide list according to various criteria (missing values, contaminants, reverse sequences);
- **Cross replicate normalization**, so as to make the quantitative values comparable between the different analyzed samples;
- **Missing values imputation** with different methods, depending on the nature of the missing values;
- **Aggregation** from peptide to protein intensity values;
- **Differential analysis**, which includes null hypothesis significance testing as well as multiple testing correction (for false discovery rate estimation).
- **Gene Ontology (GO) analysis** allows is to map protein list onto GO terms and to test category enrichment.

For more details, please refer to the "Help" tab.

Figure 1: Default screen of ProStaR

- **Descriptive statistics**: It provides different visualization tools that are helpful to understand the dataset, and to picture the influence of the various processing;
- **Data processing**: This is the heart of ProStaR, where all the *DAPAR* functionalities can be accessed to;
- **Help**: A serie of informations about the software, associated references, etc.

On the right hand side of the navbar menu, a dropdown menu referred to as "Dataset versions" makes it possible to navigate back through the history of the processing. Its use is detailed in Section 3.6

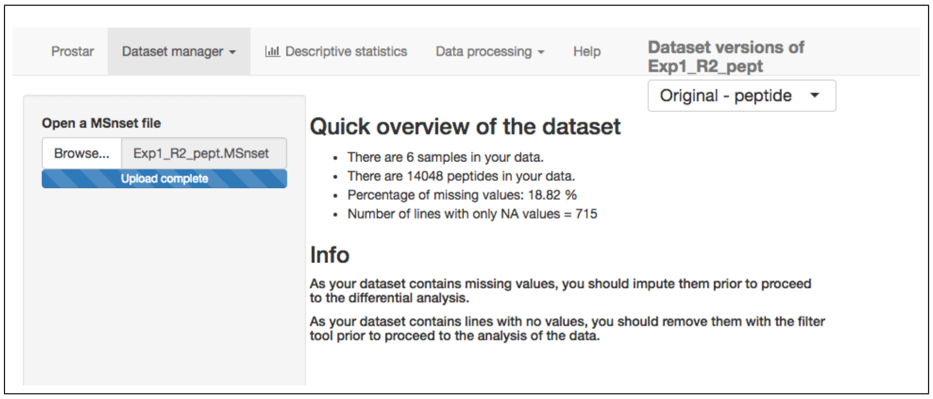
3.2 Dataset manager

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*: It is either possible to load existing MSnSet files (see Section 3.2.1) or to import text (-tabulated) and Excel files (see Section 3.2.2). The "Demo mode" allows to load the datasets of the package *DAPARdata* (see Section 3.2.5).

3.2.1 Open MSnSet

The user can upload a dataset that is already formated as an MSnSet file, by clicking on "Open MSnSet File" (see Fig. 2). 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, the menu of "Dataset versions" is updated to "Original - peptide"



Quick overview of the dataset

- There are 6 samples in your data.
- There are 14048 peptides in your data.
- Percentage of missing values: 18.82 %
- Number of lines with only NA values = 715

Info

As your dataset contains missing values, you should impute them prior to proceed to the differential analysis.

As your dataset contains lines with no values, you should remove them with the filter tool prior to proceed to the analysis of the data.

Figure 2: Open a MSnSet file

or "Original - protein" whether the file contains quantitative information about peptides or proteins (see Section 3.6). All the plots in the "Descriptive statistics" submenu (see Section 3.3) 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 examples of MSnSet file in the 'extdata' directory of *DAPARdata*. The user can find this directory by entering:

```
> installed.packages()["DAPARdata","LibPath"]
```

3.2.2 Import

Alternatively, the user can create a quantitative dataset in the MSnSet format, on the basis of TSV (Tab Separated Values) or Excel files (in format xls or xlsx) that contain 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 procedure to create the MSnSet object. Let us describe the import format first, and the import procedure then.

Import data
Data are imported through a text file (.txt) formatted as a column separated file, with tabulations (*i.e.* "\t" character) as column separator. The first line of the text file contains the column names. A minimum of 4 columns with quantitative values are necessary: As *ProStaR* is made for label-free discovery

proteomics, it is required to have *a minima* 2 conditions (or groups, or labels) to compare; moreover, 2 replicates per conditions are necessary, so that it is possible to compute a condition-wise variance. Regarding the quantitative values, the decimal separator is ".", and the intensities may be either log-transformed or not. It is also advised to have an additional columns that contains IDs (that is that gives a unique name to each line of the dataset). If such a unique ID does not appear in the dataset, one will be automatically generated.

Of course, it is possible to have more columns with quantitative values; as well as additional columns with other information. The latter ones will be considered as metadata when imported.

It is recommended to avoid special characters such as "♯", "@", "$", "%", etc. that are automatically removed. Similarly spaces in column names are replaced by ".".

If the dataset describes proteins, each line should correspond to one and only one protein. If the dataset describes peptides, each line should correspond to one and only one peptide. In addition, it is necessary to have a column that lists (separated by a ";") all the parent proteins of each peptide, as this piece of information is mandatory for the aggregation. It is also necessary that these parent proteins are describe by a unique ID (so as to avoid confusion between the proteins).

As an example, please find at the following address a dataset abstract that can be inspired from: http://www.bioconductor.org/packages/release/bioc/readmes/Prostar/README.

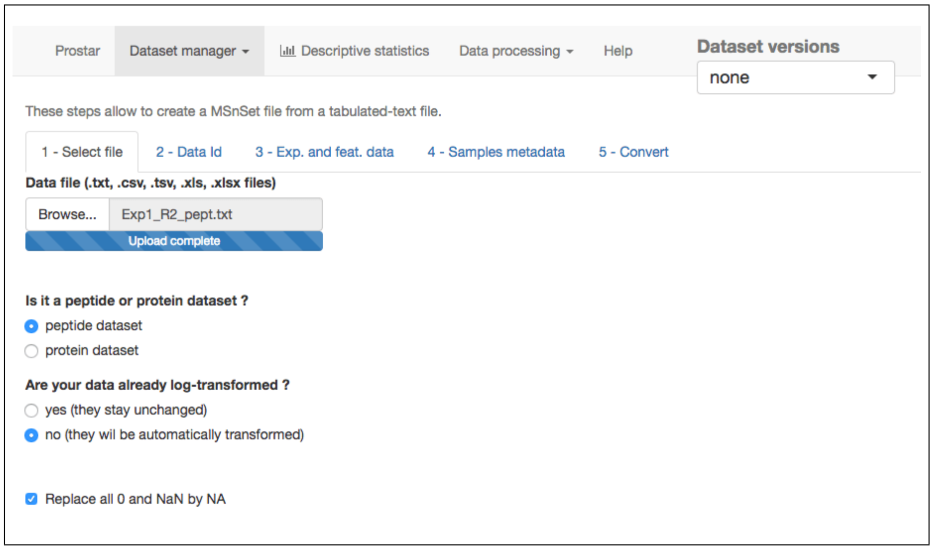
Import procedure
Several panels guide the user through the different steps of the procedure:

Select file: Select the CSV/txt or Excel (*.xls, *.xlsx) file to import (see Fig. 3). The file must contain a table where each line corresponds to a peptide or protein, except the first one which must contain the names of the colums. If the user chooses an Excel file, a dropdown menu appears and ask the user to select the spreadsheet containing the data.

As it appears in Fig. 3, some options allows for specifying if data are related to peptides or proteins, if the abundance values are already log-transformed or not [1], and also if 0 and NaN values should be replaced by **NA**.

Data ID: This step is to set the column that corresponds to the unique ID of peptides or proteins. The user has two options: let *ProStaR* creates such an ID by itself or choose among the columns available in the data file. If choosing the second option, then a drop-down menu appears and provides the list of the

[1]In *DAPAR*, the analysis is always conducted on log-transformed data. They may have previously been transformed, but if not, then *DAPAR* automatically performs the transformation. The user should not try applying any *DAPAR* processing on data that are not log-transformed, for the result would be dubious.



These steps allow to create a MSnSet file from a tabulated-text file.

1 - Select file | 2 - Data Id | 3 - Exp. and feat. data | 4 - Samples metadata | 5 - Convert

Data file (.txt, .csv, .tsv, .xls, .xlsx files)

Browse... Exp1_R2_pept.txt

Upload complete

Is it a peptide or protein dataset ?

[✓ peptide dataset]
protein dataset

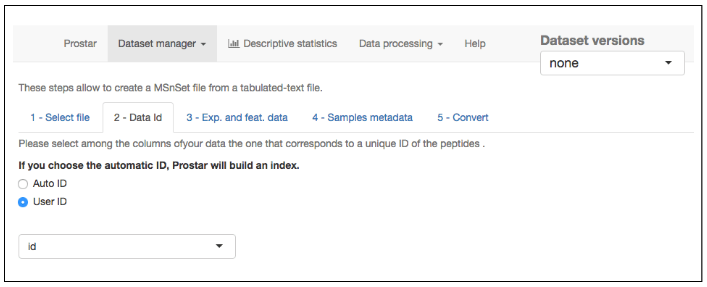
Are your data already log-transformed ?

yes (they stay unchanged)
[✓ no (they wil be automatically transformed)]

[✓ Replace all 0 and NaN by NA]

Figure 3: Importing an Excel file, tab 1

column names. A column corresponding to the unique ID of the peptides or proteins should be selected (see Fig. 4). If the column contains non unique IDs, a warning alerts the user and suggests him to choose another column.



These steps allow to create a MSnSet file from a tabulated-text file.

1 - Select file | 2 - Data Id | 3 - Exp. and feat. data | 4 - Samples metadata | 5 - Convert

Please select among the columns ofyour data the one that corresponds to a unique ID of the peptides .

If you choose the automatic ID, Prostar will build an index.

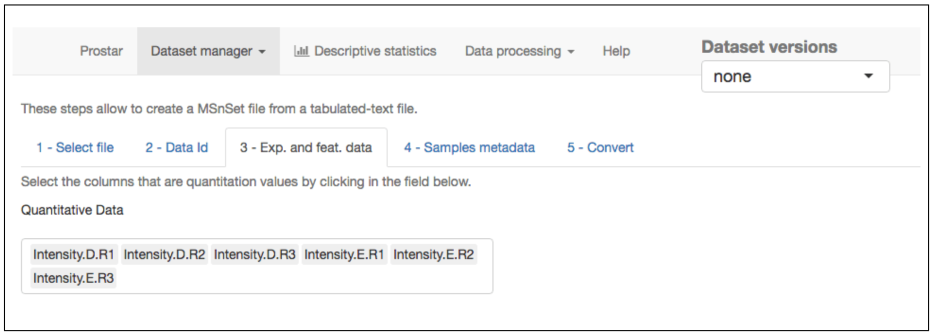
Auto ID
[✓ User ID]

id

Figure 4: Importing a CSV file, tab 2

Exp. and Feat. data: In the "Quantitative data" list, select (one after the other) 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. 5). If an item is selected by mistake, it can be removed by pressing on the SUPPR key.
Please note that the decimal separator should be ".".

Sample metadata: In this tab, the user fills the informations related to the samples, *i.e.* the experimental design. The colum named *Experiment* is filled by default with the name of the different samples. The user fills the other columns: *Label* corresponds to the conditions of the experiment that will be compared during the differential analysis; *Bio.Rep*, *Tech.rep* and *Analyt.Rep.* correspond respectively to the biological, technical and analytical replicates (Fig. 6). The



These steps allow to create a MSnSet file from a tabulated-text file.

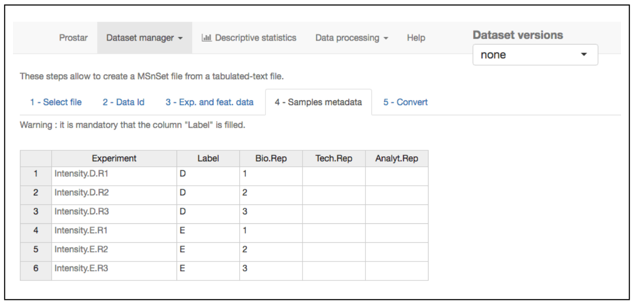
Select the columns that are quantitation values by clicking in the field below.

Quantitative Data

Intensity.D.R1 Intensity.D.R2 Intensity.D.R3 Intensity.E.R1 Intensity.E.R2 Intensity.E.R3

Figure 5: Importing a CSV file, tab 3

column Label is mandatory (for the subsequent differential analysis), the other ones are optional.
For the case of a peptide dataset, and in order to be able to agregate the peptides intensities on proteins ones : there should be a column indicating, for each peptide, the protein/s the peptide belongs to. If a peptide belongs to more than one protein, proteins names should be separated by ";".



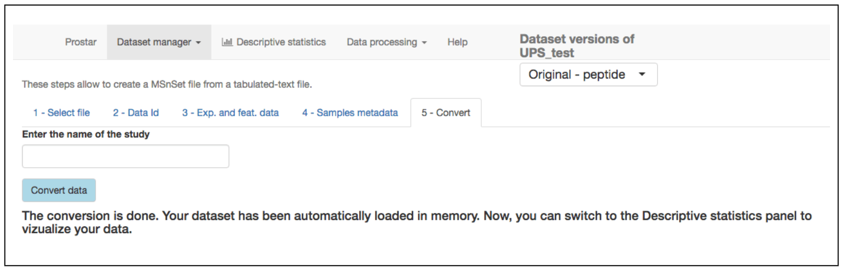
These steps allow to create a MSnSet file from a tabulated-text file.

Warning : it is mandatory that the column "Label" is filled.

	Experiment	Label	Bio.Rep	Tech.Rep	Analyt.Rep
1	Intensity.D.R1	D	1		
2	Intensity.D.R2	D	2		
3	Intensity.D.R3	D	3		
4	Intensity.E.R1	E	1		
5	Intensity.E.R2	E	2		
6	Intensity.E.R3	E	3		

Figure 6: Importing a CSV file, tab 4

Convert: Finally, enter the name of the MSnSet to be created (Fig. 7) 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.



These steps allow to create a MSnSet file from a tabulated-text file.

Enter the name of the study

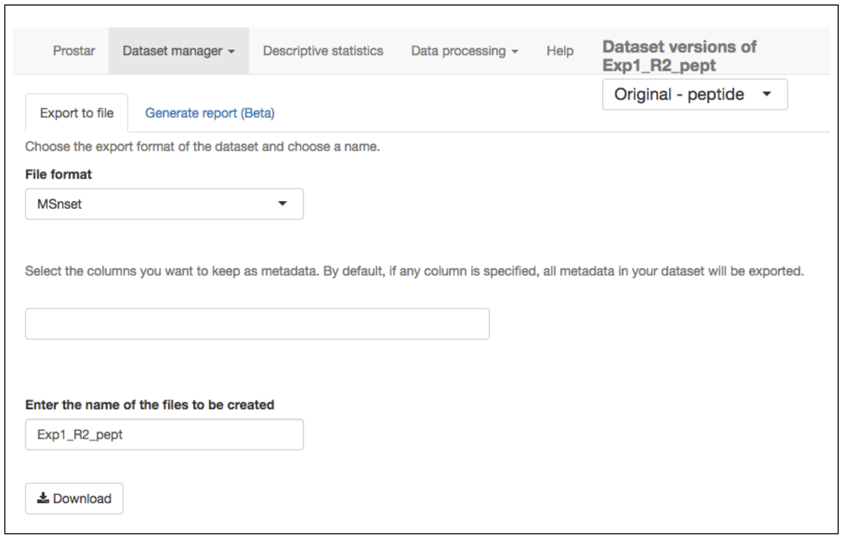
The conversion is done. Your dataset has been automatically loaded in memory. Now, you can switch to the Descriptive statistics panel to vizualize your data.

Figure 7: Importing a CSV file, tab 5

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

3.2.3 Export dataset to a file

Once an MSnSet has been created, it is possible to save it as a MSnSet binary object (so that next time, it is not necessary to create it, and a simple uploads makes it, as described in Section 3.2.1). It is also possible to export it as an Excel spreadsheet (in xlsx format). To do so, one simply goes on the corresponding tab and select the appropriate option (Fig. 8).



Choose the export format of the dataset and choose a name.

File format: MSnset

Select the columns you want to keep as metadata. By default, if any column is specified, all metadata in your dataset will be exported.

Enter the name of the files to be created: Exp1_R2_pept

Figure 8: Exporting to an Excel file

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

The user can download the plots showed in *Prostar* by right-clicking on the plot. A contextual menu appears and let the user choose either "Save image as" or "Copy image". In the latter case, he/she has to paste the image in appropriate software.

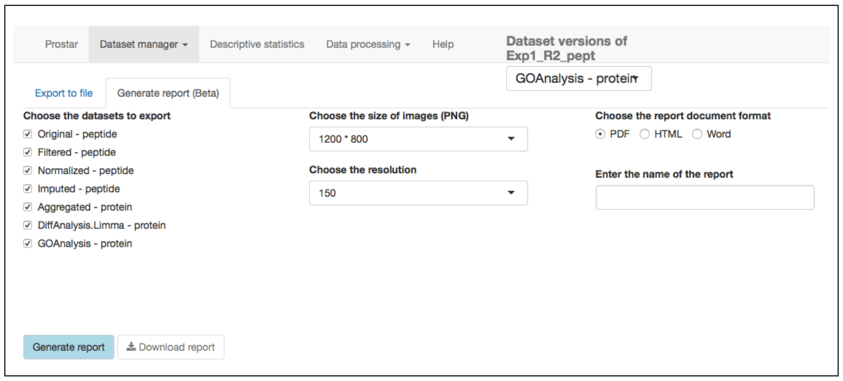
3.2.4 Export plots as report (Beta)

In addition to the exported datasets (see previous section), the user can generate and download a report, dynamically created on demand (e.g. at the end of an analysis), through the interface presented on Fig. 9.

The left hand side of the interface shows a list of checkboxes (one for each dataset created during the analysis). By default, all the datasets are selected. The user chooses which results to include in the report. Even if no analysis has been performed (e.g. right after the upload of a dataset), the plots of the Descriptive Statistics panel are available.

The right hand side of the interface allows the user to choose the size and resolution of the images and the format of the report: PDF, HTML or DOC file.

After selecting the desired options, one clicks on the button "Generate a report". Then, ProStaR rebuilds all the required images and includes them in the report file. During this step, the button "Download" remains disabled. Only when the report is ready, the "Download" buttons is enabled.



Dataset versions of Exp1_R2_pept: GOAnalysis - protein

Choose the datasets to export
- [✓] Original - peptide
- [✓] Filtered - peptide
- [✓] Normalized - peptide
- [✓] Imputed - peptide
- [✓] Aggregated - protein
- [✓] DiffAnalysis.Limma - protein
- [✓] GOAnalysis - protein

Choose the size of images (PNG): 1200 * 800

Choose the resolution: 150

Choose the report document format: [✓ PDF] HTML Word

Enter the name of the report:

Figure 9: Exporting analysis as a report

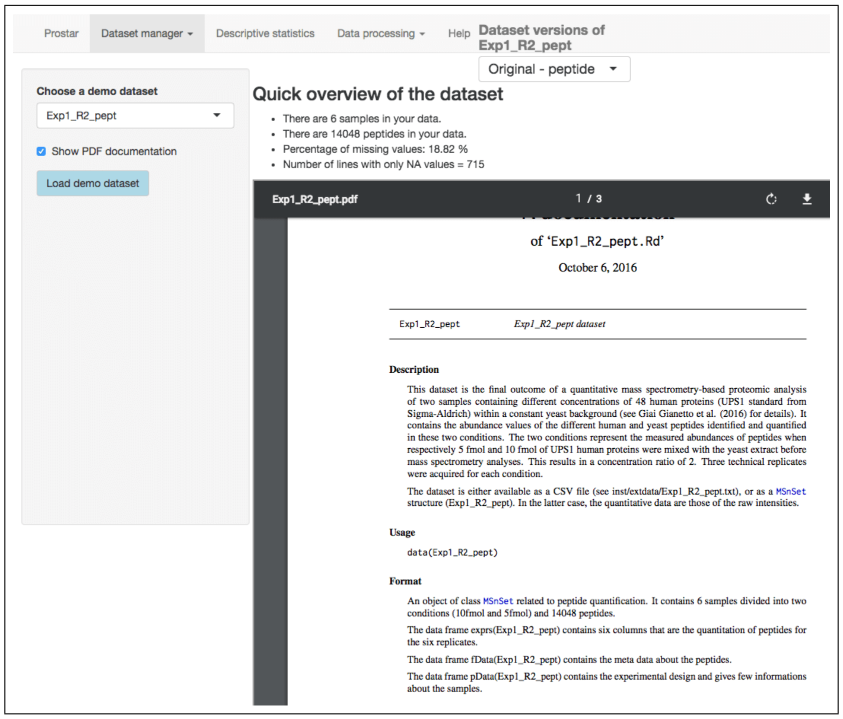
Please note this report generation functionality is still in Beta version, so that some bugs may remain. In the future, and depending on the users' expectations, the report will be completed:

- So far, the R commands run for the report generation does not appear in the log console (see 3.2.6);
- The texts accompanying the figures is minimal;
- The report cannot be customized.

In the future, we hope to improve it, as well as to generate additional ready-to-use files for publication (Material & Methods sketch, R script for reproducibility, etc.)

3.2.5 Demo mode

In order to facilitate first steps with Prostar, the "demo mode" menu allows the user to access the datasets contained in the package *DAPARdata*. When the user chooses one of those datasets, it is automatically loaded in *Prostar*. It can be used to easily test the various functionalities of *Prostar* (Fig. 10).



Quick overview of the dataset

- There are 6 samples in your data.
- There are 14048 peptides in your data.
- Percentage of missing values: 18.82 %
- Number of lines with only NA values = 715

of 'Exp1_R2_pept.Rd'

October 6, 2016

Exp1_R2_pept *Exp1_R2_pept dataset*

Description

This dataset is the final outcome of a quantitative mass spectrometry-based proteomic analysis of two samples containing different concentrations of 48 human proteins (UPS1 standard from Sigma-Aldrich) within a constant yeast background (see Giai Gianetto et al. (2016) for details). It contains the abundance values of the different human and yeast peptides identified and quantified in these two conditions. The two conditions represent the measured abundances of peptides when respectively 5 fmol and 10 fmol of UPS1 human proteins were mixed with the yeast extract before mass spectrometry analyses. This results in a concentration ratio of 2. Three technical replicates were acquired for each condition.

The dataset is either available as a CSV file (see inst/extdata/Exp1_R2_pept.txt), or as a MSnSet structure (Exp1_R2_pept). In the latter case, the quantitative data are those of the raw intensities.

Usage

```
data(Exp1_R2_pept)
```

Format

An object of class MSnSet related to peptide quantification. It contains 6 samples divided into two conditions (10fmol and 5fmol) and 14048 peptides.

The data frame exprs(Exp1_R2_pept) contains six columns that are the quantitation of peptides for the six replicates.

The data frame fData(Exp1_R2_pept) contains the meta data about the peptides.

The data frame pData(Exp1_R2_pept) contains the experimental design and gives few informations about the samples.

Figure 10: Loadind a demo dataset

A checkbox allows the user to show the PDF file in the interface; by default, it is unchecked.

3.2.6 Session log

Each time the user validates a processing step (by clicking on the "Save <*the_step*>" button, see Section 3.4), 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 11). 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, please refer to Section 3.3.2).

Moreover, in the "R source code" tab, the user has access to the R commands (from DAPAR) that have been used to process its dataset (see Figure 12). This code may be copy and paste in a R script to automate the analysis of a dataset.

3.3 Descriptive statistics

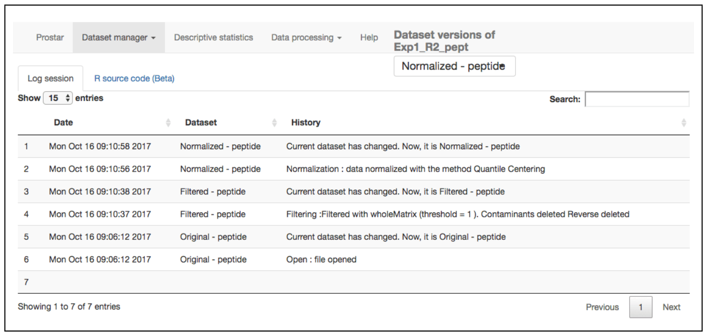
Several plots (one plot per tab) are proposed to help the user to have a quick and as complete as possible overview of his/her dataset. This menu is an essentiel element for the user to check that each processing step indeed gave the expected result.

3.3.1 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 histogram in the middle displays the distribution of missing values; the red bin counts the peptide or protein lines that only contains missing values (Fig. 13). The barplot on the right shows the distribution of missing values per condition.

Command line: In *DAPAR*, the functions for these three plots are:

- for the dataframe parameter: `mvPerLinesHisto_HC()`, `mvHisto_HC()` and `mvPerLinesHistoPerCondition_HC()`,
- for an object of class MSnSet: `wrapper.mvPerLinesHisto_HC()`, `wrapper.mvHisto_HC()` and `wrapper.mvPerLinesHistoPerCondition_HC()`.



	Date	Dataset	History
1	Mon Oct 16 09:10:58 2017	Normalized - peptide	Current dataset has changed. Now, it is Normalized - peptide
2	Mon Oct 16 09:10:56 2017	Normalized - peptide	Normalization : data normalized with the method Quantile Centering
3	Mon Oct 16 09:10:38 2017	Filtered - peptide	Current dataset has changed. Now, it is Filtered - peptide
4	Mon Oct 16 09:10:37 2017	Filtered - peptide	Filtering :Filtered with wholeMatrix (threshold = 1). Contaminants deleted Reverse deleted
5	Mon Oct 16 09:06:12 2017	Original - peptide	Current dataset has changed. Now, it is Original - peptide
6	Mon Oct 16 09:06:12 2017	Original - peptide	Open : file opened
7			

Showing 1 to 7 of 7 entries

Figure 11: Example of the log of a session in ProStaR

3.3.2 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. 14). The missing values are represented by empty cells.

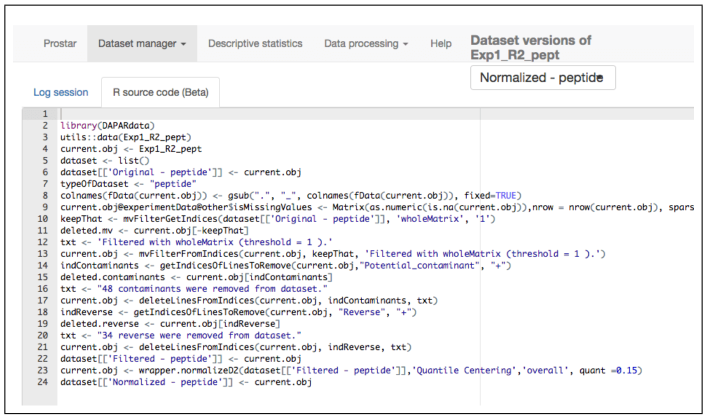
The second tab is named "Analyte metadata". It contains the metadata of the proteins (see Fig. 15).

The thrid tab is named "Replicate metadata". The information displayed here is the one entered by the user during the import step (see Fig. 16).

The last tab, named "Dataset history" contains the log of the previous processing. Contrarily to the "Session log" panel (see Section 3.2.6), the information here does not relate to the session, and is saved from a session to the next one.

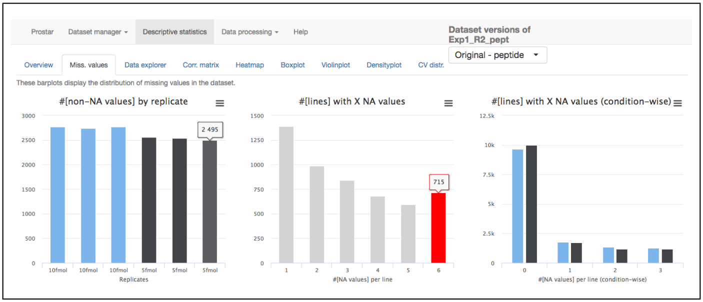
Command line: The *DAPAR* functions to get the three first tables are in fact those from the *MSnbase* package: `exprs()` (Quantitative data), `fData()` (Analyte metadata) and `pData()` (Replicate metadata). Similarly, the "Dataset history" information is also accessible. In fact, it is stored in the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

```
> getProcessingInfo(obj)
```



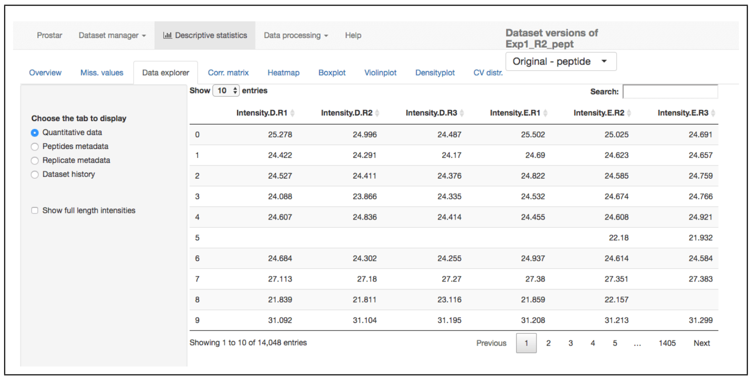
```

library(DAPARdata)
utils::data(Exp1_R2_pept)
current.obj <- Exp1_R2_pept
dataset <- list()
dataset[['Original - peptide']] <- current.obj
typeOfDataset <- "peptide"
colnames(fData(current.obj)) <- gsub(".", "_", colnames(fData(current.obj)), fixed=TRUE)
current.obj@experimentData@other$isMissingValues <- Matrix(as.numeric(is.na(current.obj)),nrow = nrow(current.obj), spars
keepThat <- mvFilterGetIndices(dataset[['Original - peptide']], 'wholeMatrix', '1')
deleted.mv <- current.obj[-keepThat]
txt <- 'Filtered with wholeMatrix (threshold = 1 ).'
current.obj <- mvFilterFromIndices(current.obj, keepThat, 'Filtered with wholeMatrix (threshold = 1 ).')
indContaminants <- getIndicesOfLinesToRemove(current.obj,"Potential_contaminant", "+")
deleted.contaminants <- current.obj[indContaminants]
txt <- "48 contaminants were removed from dataset."
current.obj <- deleteLinesFromIndices(current.obj, indContaminants, txt)
indReverse <- getIndicesOfLinesToRemove(current.obj, "Reverse", "+")
deleted.reverse <- current.obj[indReverse]
txt <- "34 reverse were removed from dataset."
current.obj <- deleteLinesFromIndices(current.obj, indReverse, txt)
dataset[['Filtered - peptide']] <- current.obj
current.obj <- wrapper.normalizeD2(dataset[['Filtered - peptide']],'Quantile Centering','overall', quant =0.15)
dataset[['Normalized - peptide']] <- current.obj
```

Figure 12: Example of the R code generated during a session in ProStaR



These barplots display the distribution of missing values in the dataset.

Figure 13: Histrograms for the overview of the missing values



Choose the tab to display
● Quantitative data
○ Peptides metadata
○ Replicate metadata
○ Dataset history

☐ Show full length intensities

Show 10 entries

	Intensity.D.R1	Intensity.D.R2	Intensity.D.R3	Intensity.E.R1	Intensity.E.R2	Intensity.E.R3
0	25.278	24.996	24.487	25.502	25.025	24.691
1	24.422	24.291	24.17	24.69	24.623	24.657
2	24.527	24.411	24.376	24.822	24.585	24.759
3	24.088	23.866	24.335	24.532	24.674	24.766
4	24.607	24.836	24.414	24.455	24.608	24.921
5					22.18	21.932
6	24.684	24.302	24.255	24.937	24.614	24.584
7	27.113	27.18	27.27	27.38	27.351	27.383
8	21.839	21.811	23.116	21.859	22.157	
9	31.092	31.104	31.195	31.208	31.213	31.299

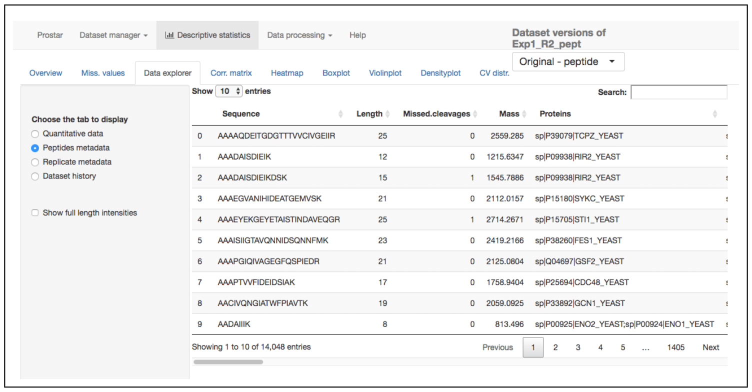
Showing 1 to 10 of 14,048 entries

Figure 14: View of quantitative data in the MSnSet dataset

3.3.3 Correlation matrix

In this tab, it is possible to visualize the extent to which the replicates correlate or not (see Fig. 17). The contrast in the matrix may be changed by modifying the color gradient.

Command line: In *DAPAR*, the corresponding function are:



Choose the tab to display
○ Quantitative data
● Peptides metadata
○ Replicate metadata
○ Dataset history

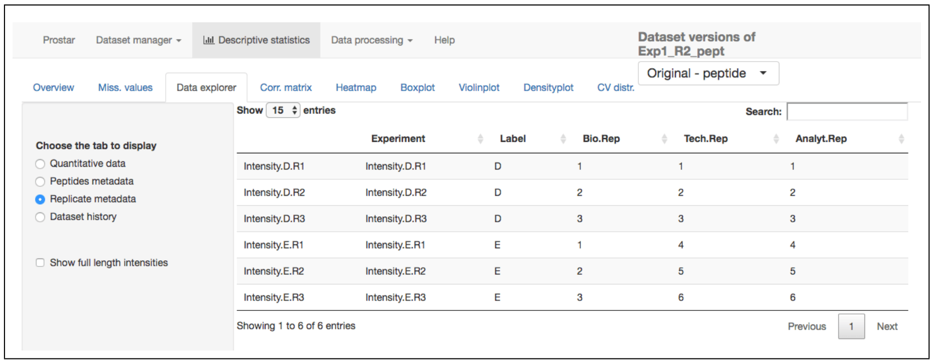
☐ Show full length intensities

Show 10 entries

	Sequence	Length	Missed.cleavages	Mass	Proteins
0	AAAAQDEITGDGTTTVVCIVGEIIR	25	0	2559.285	sp|P39079|TCPZ_YEAST
1	AAADAISDIEIK	12	0	1215.6347	sp|P09938|RIR2_YEAST
2	AAADAISDIEIKDSK	15	1	1545.7886	sp|P09938|RIR2_YEAST
3	AAAEGVANIHIDEATGEMVSK	21	0	2112.0157	sp|P15180|SYKC_YEAST
4	AAAEYEKGEYETAISTINDAVEQGR	25	1	2714.2671	sp|P15705|STI1_YEAST
5	AAAISIIGTAVQNNIDSQNNFMK	23	0	2419.2166	sp|P38260|FES1_YEAST
6	AAAPGIQIVAGEGFQSPIEDR	21	0	2125.0804	sp|Q04697|GSF2_YEAST
7	AAAPTVVFIDEIDSIAK	17	0	1758.9404	sp|P25694|CDC48_YEAST
8	AACIVQNGIATWFPIAVTK	19	0	2059.0925	sp|P33892|GCN1_YEAST
9	AADAIIIK	8	0	813.496	sp|P00925|ENO2_YEAST;sp|P00924|ENO1_YEAST

Showing 1 to 10 of 14,048 entries

Figure 15: View of feature meta-data in the MSnSet dataset



Choose the tab to display
- Quantitative data
- Peptides metadata
- [✓ Replicate metadata]
- Dataset history

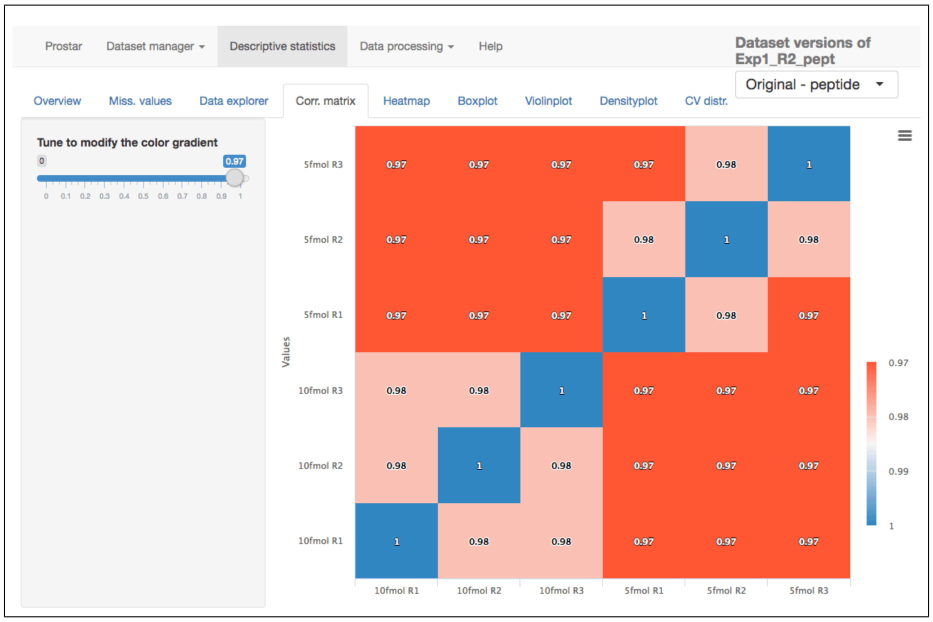
Show full length intensities

Show 15 entries

	Experiment	Label	Bio.Rep	Tech.Rep	Analyt.Rep
Intensity.D.R1	Intensity.D.R1	D	1	1	1
Intensity.D.R2	Intensity.D.R2	D	2	2	2
Intensity.D.R3	Intensity.D.R3	D	3	3	3
Intensity.E.R1	Intensity.E.R1	E	1	4	4
Intensity.E.R2	Intensity.E.R2	E	2	5	5
Intensity.E.R3	Intensity.E.R3	E	3	6	6

Showing 1 to 6 of 6 entries

Figure 16: View of samples meta-data in the MSnSet dataset



	10fmol R1	10fmol R2	10fmol R3	5fmol R1	5fmol R2	5fmol R3
5fmol R3	0.97	0.97	0.97	0.97	0.98	1
5fmol R2	0.97	0.97	0.97	0.98	1	0.98
5fmol R1	0.97	0.97	0.97	1	0.98	0.97
10fmol R3	0.98	0.98	1	0.97	0.97	0.97
10fmol R2	0.98	1	0.98	0.97	0.97	0.97
10fmol R1	1	0.98	0.98	0.97	0.97	0.97

Figure 17: Correlation matrix for the quantitative data

- for the dataframe parameter: `corrMatrixD_HC()`,
- for an object of class MSnSet: `wrapper.corrMatrixD_HC()`.

3.3.4 Heatmap

A heatmap is drawn with the associated dendrogram (see Fig. 18). 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**: the parameter *method* of the function `stats::dist`. The default value is *'euclidean'*
- **Linkage**: the parameter *method* of the function `stats::hclust`. The default value is *'complete'*

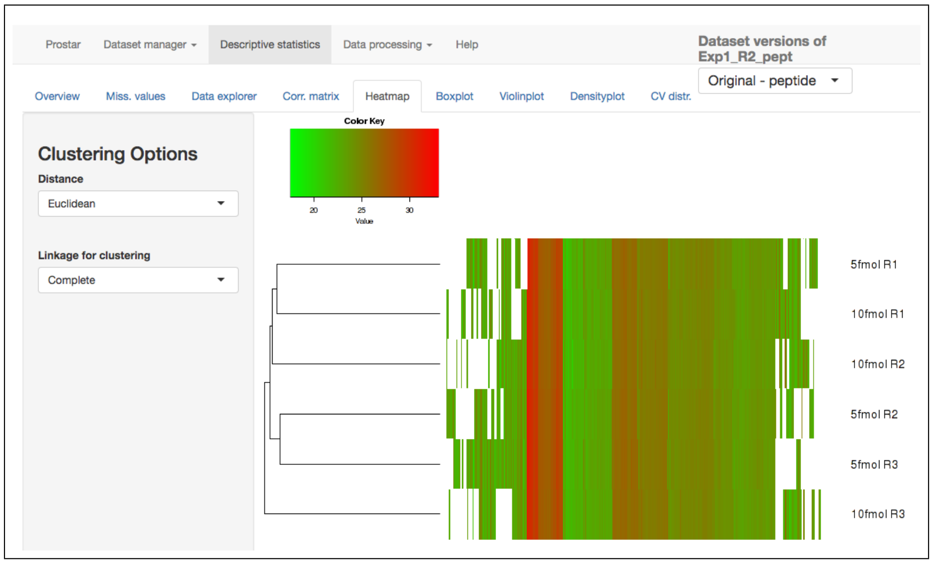


Figure 18: Heatmap and dendrogram for the quantitative data

Command line: In *DAPAR*, the corresponding function are:

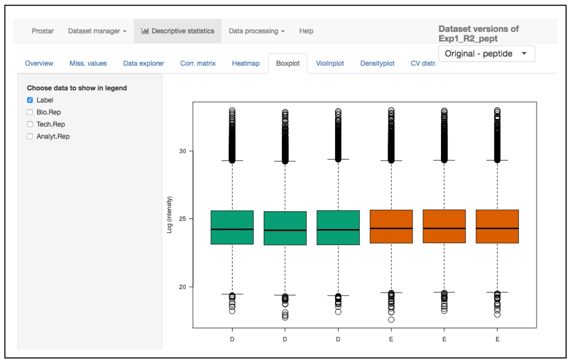
- for the dataframe parameter: `heatmapD()`,
- for an object of class MSnSet: `wrapper.heatmapD()`.

3.3.5 Boxplot

The protein distribution by replicates is summarized with boxplots (see Fig. 19). 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 functions are:

- for the dataframe parameter: `boxPlotD()`,
- for an object of class MSnSet: `wrapper.boxPlotD()`.



Dataset versions of Exp1_R2_pept

Original - peptide

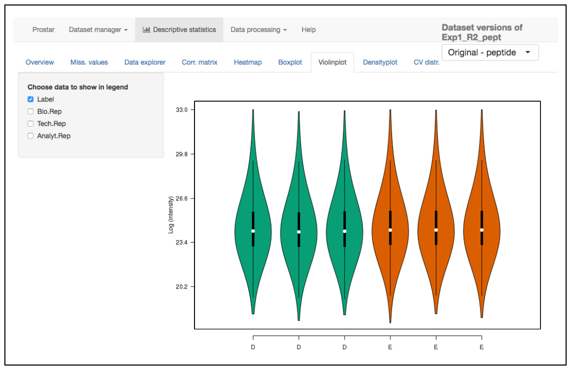
Choose data to show in legend

- [✓] Label
- [ ] Bio.Rep
- [ ] Tech.Rep
- [ ] Analyt.Rep

Figure 19: Boxplot for the quantitative data

3.3.6 Violin plot

The protein distribution by replicates is summarized with violin plots (see Fig. 20). 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*).



Dataset versions of Exp1_R2_pept

Original - peptide

Choose data to show in legend

- [✓] Label
- [ ] Bio.Rep
- [ ] Tech.Rep
- [ ] Analyt.Rep

Figure 20: Violin plot for the quantitative data

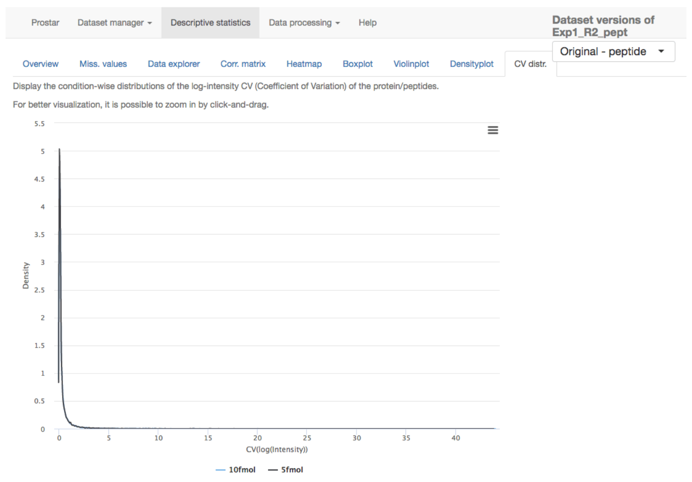
Command line: In *DAPAR*, the corresponding functions are:

- for the dataframe parameter: `violinPlotD()`,
- for an object of class MSnSet: `wrapper.violinPlotD()`.

3.3.7 CV distribution

This plot shows the distribution of the CV of the log-intensity of proteins for each condition (see Fig. 21). For better visualization, the user can zoom in by click-and-drag (see Fig. 22).

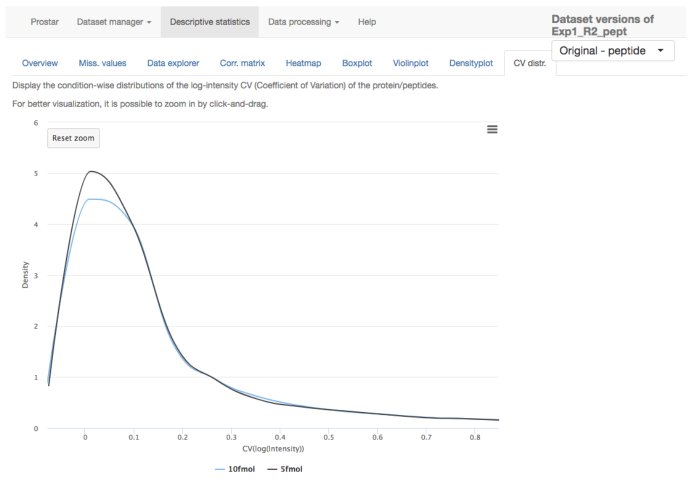
Command line: In *DAPAR*, the corresponding function is



Display the condition-wise distributions of the log-intensity CV (Coefficient of Variation) of the protein/peptides.

For better visualization, it is possible to zoom in by click-and-drag.

Figure 21: CV distribution for the quantitative data



Display the condition-wise distributions of the log-intensity CV (Coefficient of Variation) of the protein/peptides.

For better visualization, it is possible to zoom in by click-and-drag.

Figure 22: Zoom in of the CV distribution for the quantitative data

- for the dataframe parameter: `CVDistD_HC()`,
- for an object of class MSnSet: `wrapper.CVDistD_HC()`.

3.3.8 Density plot

This plots shows the distribution of the log-intensity of proteins for each condition (see Fig. 23).

Two options are available to custom the plot:

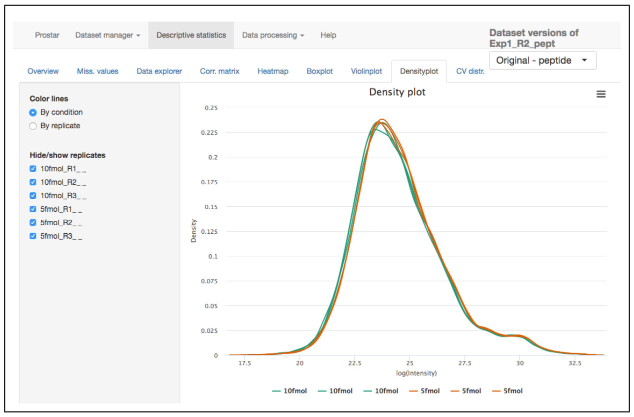


Figure 23: Densityplot the quantitative data

- **Plot to show** which defines how to color the replicates: one color for each condition (value "By condition") or one color per replicate (value "By replicate"). By default, the data are colored by condition.
- **Select data to show** Select the replicates to display. By default, all the replicates are showed.

Command line: In *DAPAR*, the corresponding functions are:

- for the dataframe parameter: `densityPlotD_HC()`,
- for an object of class MSnSet: `wrapper.densityPlotD_HC()`.

3.4 Data processing

The "Data processing" menu contains the 5 predefined steps of a quantitative analysis. They are designed to be used in a specific order:

1. Filtering
2. Normalization
3. Missing values imputation
4. Aggregation
5. Differential analysis
6. Gene Ontology analysis

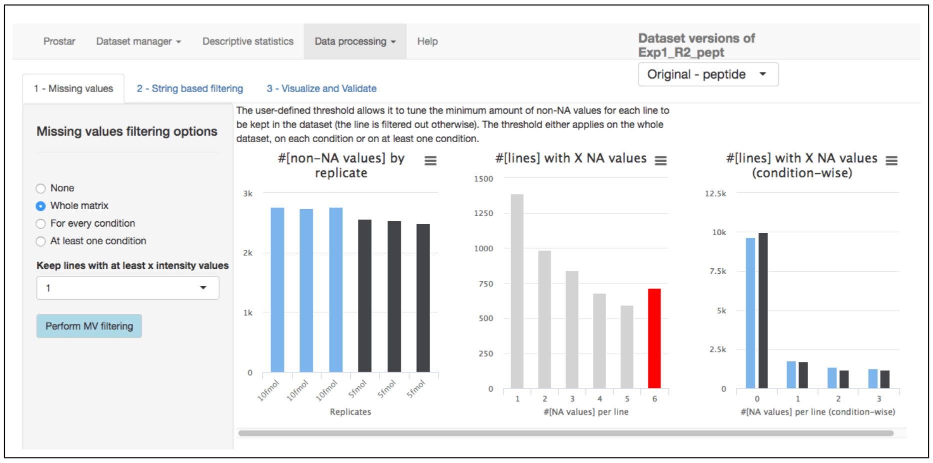
For each step, several algorithms or parameters are available, and they are toroughly detailled in the sequel of this section.

During each of these steps, it is possible to test several options, and to observe the influence of the processing in the descriptive statistics menu (see Section 3.3), which is dynamically updated.

Finally, once the ultimate tuning is chosen for a given step, it is advised to save the processing. By doing so, another dataset appears in the "Dataset versions" list (see Section 3.6). Thus, it is possible to go back to any previous step of the analysis if necessary, without starting back the analysis from scratch.

3.4.1 Filtering

In this step, the user may decide to delete several peptides or proteins according to several criteria: If the amount of missing values is too important to expect confident processing (Tab 1); or if they are identified as reverse sequences (for target-decoy approaches) or contaminants (Tab 2).


Prostar
Dataset manager
Descriptive statistics
Data processing
Help
Dataset versions of Exp1_R2_pept
Original - peptide
1 - Missing values
2 - String based filtering
3 - Visualize and Validate
Missing values filtering options
Perform MV filtering
#[non-NA values] by replicate
3k
2k
1k
0
10fmol
10fmol
10fmol
5fmol
5fmol
5fmol
Replicates
#[lines] with X NA values
1500
1250
1000
750
500
250
0
1
2
3
4
5
6
#[NA values] per line
#[lines] with X NA values (condition-wise)
12.5k
10k
7.5k
5k
2.5k
0
0
1
2
3
#[NA values] per line (condition-wise)

The user-defined threshold allows it to tune the minimum amount of non-NA values for each line to be kept in the dataset (the line is filtered out otherwise). The threshold either applies on the whole dataset, on each condition or on at least one condition.

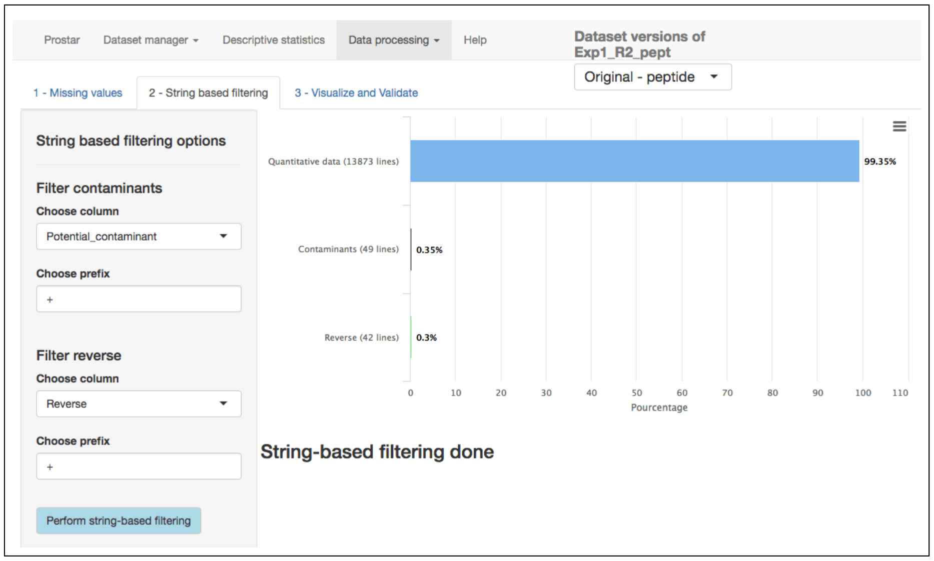
None [✓ Whole matrix] For every condition At least one condition

Keep lines with at least x intensity values: 1

Figure 24: Interface of the filtering tool - 1

To filter the missing values (first tab called "Missing values"), the choice of the lines to be deleted is made by different options (see Fig. 24):

- **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;
- **For every condition**: 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;



Dataset versions of Exp1_R2_pept

Original - peptide

String based filtering options

Filter contaminants

Choose column: Potential_contaminant

Choose prefix: +

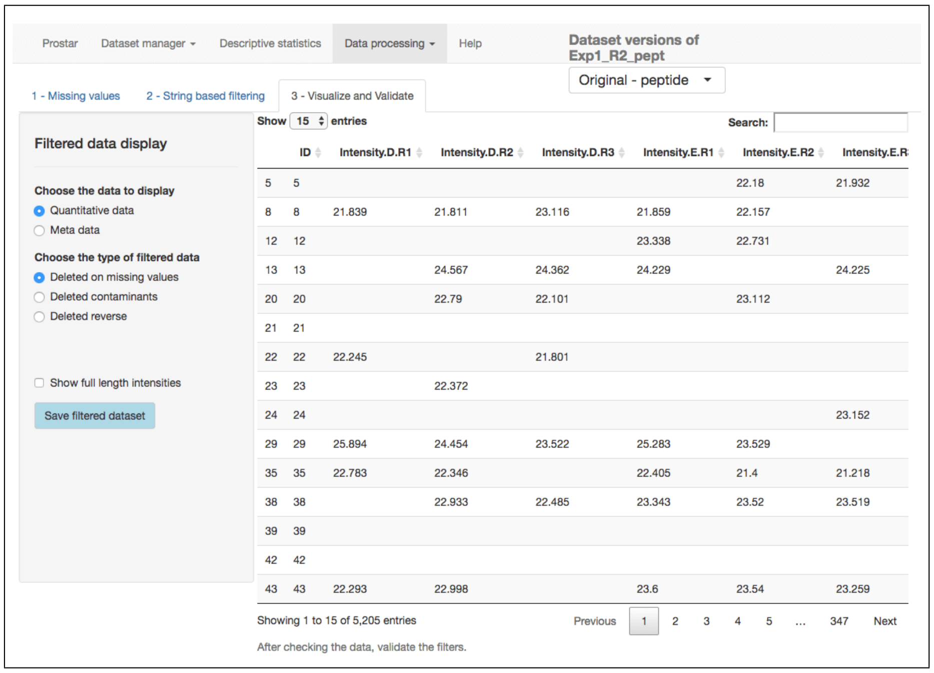
Filter reverse

Choose column: Reverse

Choose prefix: +

String-based filtering done

Figure 25: Interface of the filtering tool - 2



Dataset versions of Exp1_R2_pept

Original - peptide

Filtered data display

Choose the data to display
- [✓ Quantitative data]
- Meta data

Choose the type of filtered data
- [✓ Deleted on missing values]
- Deleted contaminants
- Deleted reverse

Show full length intensities

Show 15 entries

Search:

	ID	Intensity.D.R1	Intensity.D.R2	Intensity.D.R3	Intensity.E.R1	Intensity.E.R2	Intensity.E.R
5	5					22.18	21.932
8	8	21.839	21.811	23.116	21.859	22.157	
12	12				23.338	22.731	
13	13		24.567	24.362	24.229		24.225
20	20		22.79	22.101		23.112	
21	21						
22	22	22.245		21.801			
23	23		22.372				
24	24						23.152
29	29	25.894	24.454	23.522	25.283	23.529	
35	35	22.783	22.346		22.405	21.4	21.218
38	38		22.933	22.485	23.343	23.52	23.519
39	39						
42	42						
43	43	22.293	22.998		23.6	23.54	23.259

Showing 1 to 15 of 5,205 entries

After checking the data, validate the filters.

Figure 26: Interface of the filtering tool - 3

The user can visualize the effect of filter options without changing the current dataset by clicking on "Perform filtering". If the filtering 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 again on "Perform filtering". The plots are automatically updated. This action does not modify the dataset but offers a preview of the filtered data. The user can visualize as many times he/she wants several filtering options.

Afterwards, the user can choose to remove contaminants and reverses in the second tab called "String based filtering". To do so, he/she selects the appropriate columns of the metadata listed in the dropdown menus. Then, he/she specifies in each of these columns the prefix chain of characters that identifies the analytes to filter. Note : the button "Perform string-based filtering" is disabled until all the fields are complete.

Remark: If he/she has no idea of the prefixes, he/she can switch to the Data Explorer in the Descriptive Statistics menu, so as to visualize the corresponding metadata.

Once the choices are made, the user click on "Perform string-based filtering" to remove corresponding lines. Then, the barplot below shows the proportion of quantitative data, contaminants and reverses.

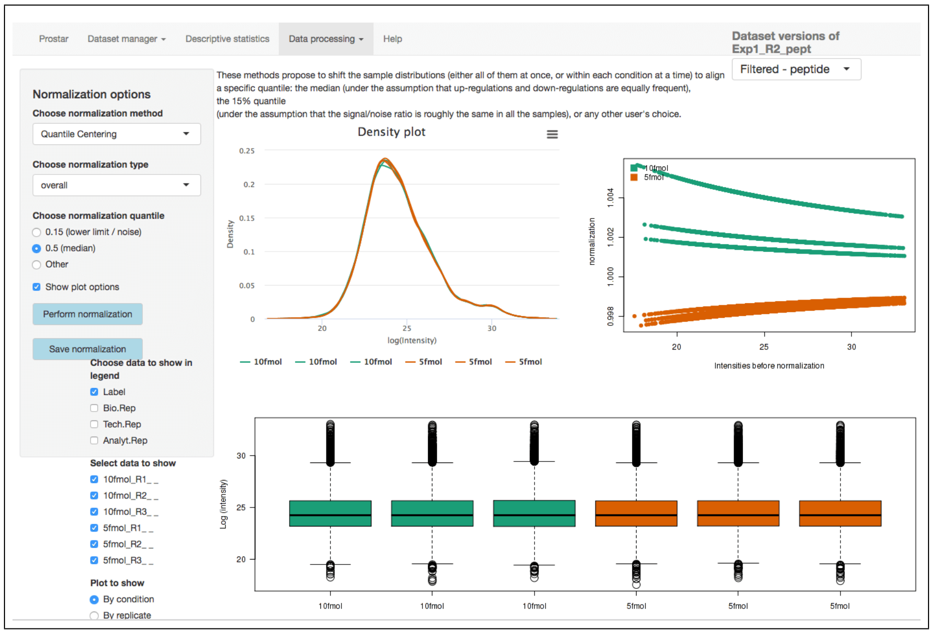
Once the filtering is appropriately tuned, the user goes to the last tab (called "visualize and Validate") (see Fig. 26), to visualize the set of analytes that have been previously filtered. On the left panel, one chooses among the lines filtered on missing values, contaminants or reverse; Then, the corresponding data table is diplayed on the right panel. Finally, one clicks on "Apply filter" so as to validate the user's choice and to apply it to the dataset. The information related to the type of filtering as well as to the chosen options appears in the Session log tab (see Section 3.2.6). A new dataset is created; it becomes the new current dataset and its name appears in the **Dataset versions** menu at the top of the screen. All plots and tables available in *ProStaR* are automatically updated.

Command line: In *DAPAR*, the function to filter missing values is `mvFilter()`. The other types of filters corresponds to classical data structure manipulation with R.

3.4.2 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 visualize the data after normalization, three plots are displayed: a boxplot, a plot that displays the differences between data before and after the normalization and a densityplot (see Fig. 27). The first and the third plots are the same as the one showed in **Descriptive Statistics**, thus they have the same options (see Sections 3.3.5 and 3.3.8).



These methods propose to shift the sample distributions (either all of them at once, or within each condition at a time) to align a specific quantile: the median (under the assumption that up-regulations and down-regulations are equally frequent), the 15% quantile (under the assumption that the signal/noise ratio is roughly the same in all the samples), or any other user's choice.

Normalization options

Choose normalization method: Quantile Centering

Choose normalization type: overall

Choose normalization quantile: 0.15 (lower limit / noise) [✓ 0.5 (median)] Other

[✓ Show plot options]

Choose data to show in legend: [✓ Label] Bio.Rep Tech.Rep Analyt.Rep

Select data to show: [✓ 10fmol_R1_ _] [✓ 10fmol_R2_ _] [✓ 10fmol_R3_ _] [✓ 5fmol_R1_ _] [✓ 5fmol_R2_ _] [✓ 5fmol_R3_ _]

Plot to show: [✓ By condition] By replicate

Figure 27: Interface of the normalization tool

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

Global alignment These methods propose normalizations of important magnitude that should be cautiously used:

- **sum by columns** operates on the original scale (not the log2 one) and propose to normalize each abundance by the total abundance of the sample (so as to focus on the analyte proportions among each sample).
- **quantile alignment** proposes to align the quantiles of all the replicates as described in [ref1]; practically it amounts to replace abundances by order statistics.

Quantile centering These methods propose to shift the sample distributions (either all of them at once, or within each condition at a time) to align a specific quantile: the median (under the assumption that up-regulations and down-regulations are equally frequent), the 15 quantile (under the assumption that the signal/noise ratio is roughly the same in all the samples), or any other user's choice. Two parameters are available:

- **Normalization type:** the centering can operate over the entire dataset (value "overall") or over each condition (value "within conditions"),
- **Value of quantile (in %):** 0.15 (lower limit/noise), 0.5 (median) or Other (In that case, the user can choose its own quantile value).

Mean centering These methods propose to shift the sample distributions (either all of them at once, or within each condition at a time) to align their means. It is also possible to force unit variance (or not). Two parameters are available:

- **Normalization type:** the centering can operate over the entire dataset (value "overall") or over each condition (value "within conditions"),
- **Include variance reduction:** Let the user choose if he/she wants to rescale the dataset to have unitary variance.

LOcally weighted regrESSion (LOESS) (See ref1 and ref2)

This method proposes to apply a cyclic LOESS normalization to the data (either all of them at once, or on each condition independently). It relates to a combination of multiple regression models. The user can tune the regression span between 0 and 1. An higher span smooths the fit more, while a lower span captures more trends.

Variance Stabilization Normalization (vsn) (See ref)

This method proposes to apply the Variance Stabilization Normalization to the data (either all of them at once, or on each condition independently). There are no specific parameters required.

Each time the user selects a method, the explanation above is displayed. The user can visualize 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 visualize 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 3.2.6). It becomes the new current dataset and the name "Normalized" appears in "Dataset versions". All plots and tables in other menus are automatically updated.

Command line: In *DAPAR*, the functions for the plot "before-after normalization" are:

- for the dataframe parameter: `compareNormalizationD()` ,
- for an object of class MSnSet: `wrapper.compareNormalizationD()` .

The corresponding functions for the normalization are:

- for an object of class MSnSet: `wrapper.normalizeD()`.

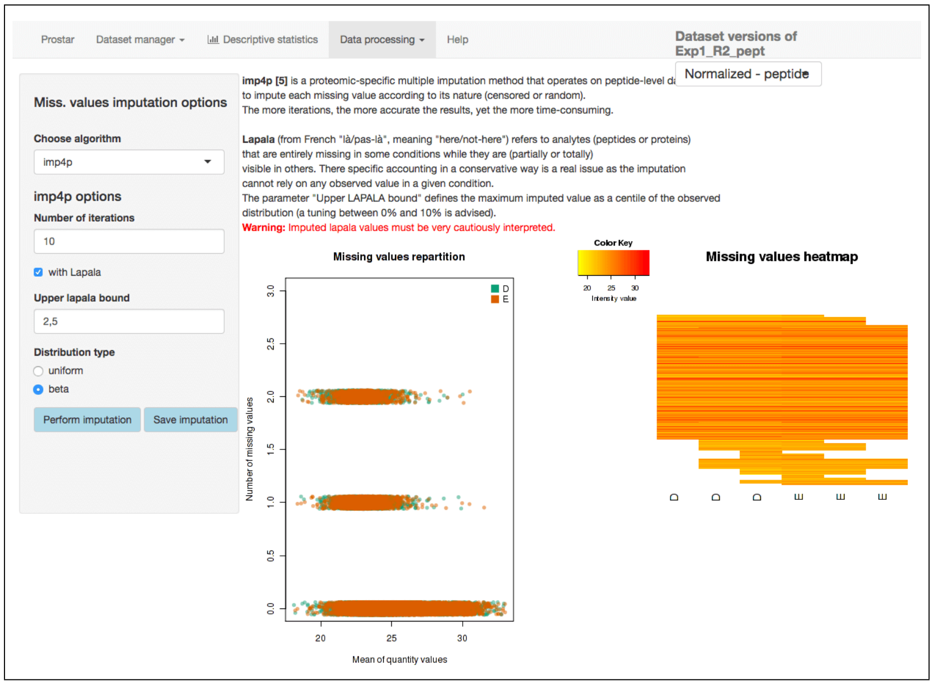
3.4.3 Imputation

Two plots are available in order to help the user choose the right imputation method for his dataset (see Fig. 28).

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 plot 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 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).



Miss. values imputation options

Choose algorithm: imp4p

imp4p options

Number of iterations: 10

[✓ with Lapala]

Upper lapala bound: 2,5

Distribution type: uniform [✓ beta]

Perform imputation | Save imputation

imp4p [5] is a proteomic-specific multiple imputation method that operates on peptide-level da
to impute each missing value according to its nature (censored or random).
The more iterations, the more accurate the results, yet the more time-consuming.

Lapala (from French "là/pas-là", meaning "here/not-here") refers to analytes (peptides or proteins)
that are entirely missing in some conditions while they are (partially or totally)
visible in others. There specific accounting in a conservative way is a real issue as the imputation
cannot rely on any observed value in a given condition.
The parameter "Upper LAPALA bound" defines the maximum imputed value as a centile of the observed
distribution (a tuning between 0% and 10% is advised).
Warning: Imputed lapala values must be very cautiously interpreted.

Figure 28: Interface of the imputation of missing values tool

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

- If the missing values are due to a mixture of censorship and of randomness mechanisms, it is advised to use the functions of the package *imp4p*. This package is a collection of proteomic-specific multiple imputation methods that operate on peptide-level datasets and which propose to impute each missing value according to its nature (censored or random). Two parameters are available: the number of iterations (the more iterations, the more accurate the results, yet the more time-consuming) and a checkbox to let the user choose if he/she wants to impute the LAPALA. This term coined from French "là/pas-là"(meaning "here/not-here") refers to analytes (peptides or proteins) that are entirely missing in some conditions while they are (partially or totally) visible in others. Their specific accounting in a conservative way is a real issue as the imputation cannot rely on any observed value in a given condition. The parameter "Upper LAPALA bound" defines the maximum imputed value as a centile of the observed distribution (a tuning between 0% and 10% is advised).
- As an alternative, it is possible to rely on several basics methods from the state of the art to perform imputation: KNN (K Nearest Neighbors) from *impute* package, MLE (Maximum Likelihood Estimation) from *norm* package, or deterministic quantile imputation. In the latter, the user tunes

a specific quantile, which is computed replicate-wise, so as to provide an imputation value for each replicate. Possibly, it is possible to multiply these values by a user-tuned factor (for instance, one may use $q(5\%) \times 0.9$ instead of $q(1\%)$).

To date, we advise to use IMP4P for peptide-level imputation and other methods for protein-level imputation.

The user can visualize 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 visualize 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 3.2.6). This new dataset becomes the new current dataset and the name "Imputed" appears in "Dataset versions". 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 respectively:

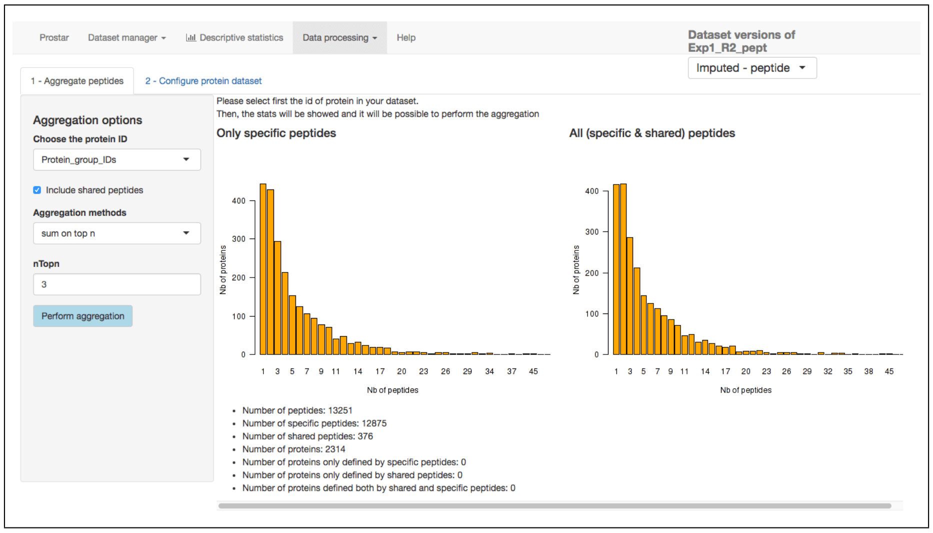
- for the dataframe parameter: `mvTypePlot()` and `mvImage()`,
- for an object of class MSnSet: `wrapper.mvTypePlot()` and `wrapper.mvImage()`.

3.4.4 Aggregation

When working on a protein dataset, this step should be bypassed. On the other hand, when working on peptide datasets, one may want to conduct the differential analysis at protein level, for proteins are the biological units of interest. To do so, it is necessary to estimate the abundance of the proteins on the basis of those of the peptides. This is what the Aggregation step is made for.

First, the user chooses the "protein id" of the dataset, i.e. the column in the metadata, that contains the IDs of all the parent proteins for each peptide. Two barplots show up (Fig. 29). They provide the distribution of proteins according to their number of peptides (either all of them, or only those which are specific to a single protein). These statistics are helpful to visualize the adjacency matrix of the peptide-protein graph, that is sometime rather big.

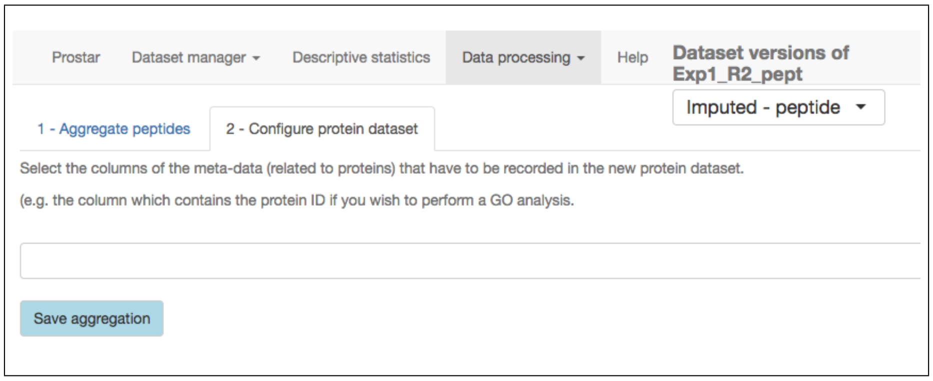
Second, a checkbox is used to indicate whether the user wants the shared peptides to be accounted for during the aggregation process.



Please select first the id of protein in your dataset.
Then, the stats will be showed and it will be possible to perform the aggregation

- Number of peptides: 13251
- Number of specific peptides: 12875
- Number of shared peptides: 376
- Number of proteins: 2314
- Number of proteins only defined by specific peptides: 0
- Number of proteins only defined by shared peptides: 0
- Number of proteins defined both by shared and specific peptides: 0

Figure 29: **Interface of the agregation tool - 1**



Select the columns of the meta-data (related to proteins) that have to be recorded in the new protein dataset.

(e.g. the column which contains the protein ID if you wish to perform a GO analysis.

Figure 30: **Interface of the agregation tool - 2**

Third, the aggregation method itself ust be chosen:

- Sum: that is the sum of the peptide intensities,
- Mean: the mean of the peptide intensities,
- Sum on top n: that is the sum over the N peptides with the highest median intensities - in this case, the additional parameter N must be tuned.

On the next tab, the user selects the columns of the peptide dataset that are of interest to be kept in the metadata of the protein dataset (e.g. the sequence of the peptides). The effect of this action is to compile, for a given parent-protein, the information of all of its child-peptides, and to store them in a dedicated column. Once done, one validates the user's choice by clicking on "Save aggregation". Then, a new "aggregated" dataset is created and loaded in memory. The aggregation method that was finally used is recorded in the Session log tab (see section 3.2.6). This new dataset becomes the new current dataset and the name "Aggregated" appears in "Dataset versions". All plots and tables in other menus are automatically updated. As the new dataset is a protein one, the "Aggregation" menu has been disabled. Thus, the interface automatically switches to the "Descriptive Statistics" menu in order to let the user check the results of the aggreation step.

The aggregation being more computationaly demanding than other processing steps, the current version of *ProStaR* does not provide the same flexibility regarding the parameter tuning. Here, it is necessary to save the aggregation result first, then, check the results in the "descriptive statistics", and possibly to go back to the imputed dataset with the "Dataset versions" dropdown menu to test another aggregation tuning. Contrarily to other processing steps, it is not possible to visualize on-the-fly the consequences of the parameter tuning, and to save it afterwards. We are currently working on improving this issue for the next versions of *ProStaR*.

Naturally, the output of this step is not a peptide dataset anymore, but a protein dataset. As a result, all the plots available in *ProStaR* are deeply modified. For instance, the barplots summarising the peptide-protein graphs disapear because they have become meaningless.

Command line: In *DAPAR*, the function used to compute the adjacency matrix peptides-proteins is `BuildAdjacencyMatrix()` and the one used to agregate the peptides into proteins is `AggregatePeptides()`. The aforementioned plot is obtained with the functions `GraphPepProt()`.

3.4.5 Differential analysis

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

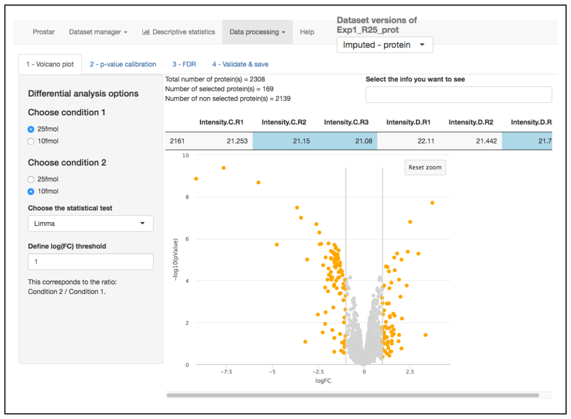
The differential analysis is divided into four steps, each impersonated by a different tab:

- Volcano plot,
- p-value calibration,
- FDR,
- Validate & save.

Volcano plot (see Fig. 31): It is a scatter plot where each analyte is represented by 2 coordinates, namely a p-value on the Y-axis (more precisely -log10(p-value)) and a fold change (FC) on the X-axis. Regarding the computation of the p-values, two tests are available in *DAPAR*, depending on the user's choice: the Welch t-test (from package *stats*) and the moderated t-test (from package *limma*). 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. Finally, the user can tune a threshold on the FC. It allows discriminating some analytes for which the difference of expression between the condition is not important enough to be biologically relevant.

This is an interactive plot which reacts to mouse's events :

- When the user puts the pointer of his mouse over a point of the plot, a tooltip window appears and shows some informations about that point. He can select the items to show in the Select widget where the different choices correspond to the columns of the feature meta-data table. The tooltip window is automatically updated,
- When the user clicks on a point, a table is displayed above the volcanoplot. It shows the values of intensities for all the samples related to the selected point. The cells colored in blue indicate that the corresponding value was a missing value in the original dataset and has been imputed,
- The user can click and draw a rectangle on the plot to zoom in. By clicking on the button named "Reset zoom", the user can return to the entire plot.



Total number of protein(s) = 2308
Number of selected protein(s) = 169
Number of non selected protein(s) = 2139

Select the info you want to see

Differential analysis options

Choose condition 1

[✓ 25fmol] 10fmol

Choose condition 2

25fmol [✓ 10fmol]

Choose the statistical test: Limma

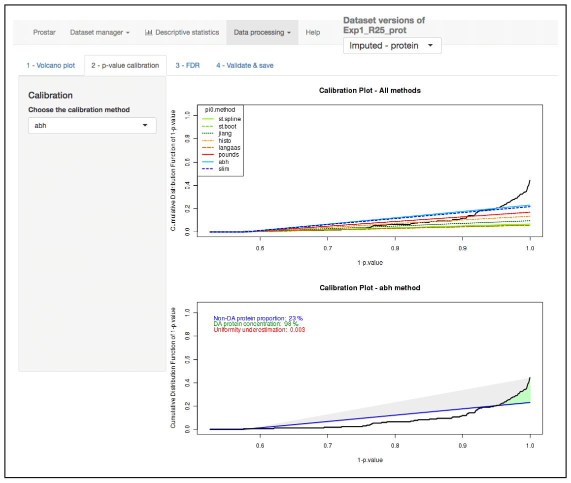
Define log(FC) threshold: 1

This corresponds to the ratio: Condition 2 / Condition 1.

	Intensity.C.R1	Intensity.C.R2	Intensity.C.R3	Intensity.D.R1	Intensity.D.R2	Intensity.D.R
2161	21.253	21.15	21.08	22.11	21.442	21.7

Figure 31: Volcanoplot of the differential analysis tool - 1

p-value calibration (see Fig. 32): In this tab, the fonctionalities of *CP4P* have been wrapped. Future versions of *ProStaR* will propose a more refined integration. To date, we redirect the reader to the *CP4P* tutorial: https://sites.google.com/site/thomasburgerswebpage/download/tutorial-CP4P-4.pdf?attredirects=0.

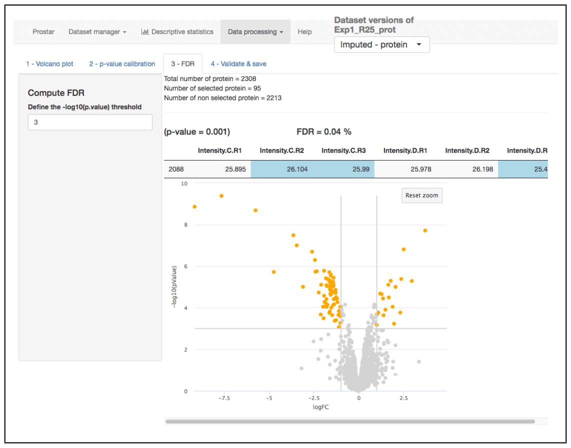


Calibration

Choose the calibration method: abh

Non-DA protein proportion: 23 %
DA protein concentration: 98 %
Uniformity underestimation: 0.003

Figure 32: Calibration plot of the differential analysis tool - 2



Compute FDR

Define the -log10(p.value) threshold: 3

Total number of protein = 2308
Number of selected protein = 95
Number of non selected protein = 2213

(p-value = 0.001) FDR = 0.04 %

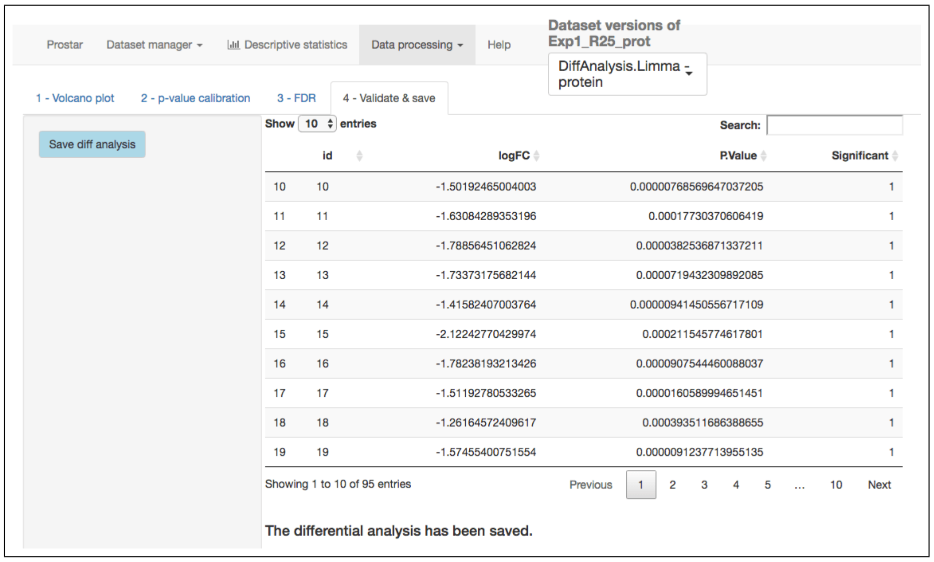
	Intensity.C.R1	Intensity.C.R2	Intensity.C.R3	Intensity.D.R1	Intensity.D.R2	Intensity.D.R
2088	25.895	26.104	25.99	25.978	26.198	25.4

Figure 33: p-Value threshold of the differential analysis tool - 3

FDR (see Fig. 33): This tab also displays the volcano plot. A threshold along the p-value axis can be tuned by the user, so as to discriminate the differentially abondant proteins (which are highlighted). A horizontal straight line is drawn to visualize the threshold. The corresponding FDR is computed. The user can adjust the thresholds in order to select the maximum of proteins by minimizing the FDR.

Command line: In *DAPAR*, the function used to compute the FDR is `diffAnaComputeFDR()`.

Validate & save (see Fig. 34): A table shows the results of the statistical test (see Fig. 34): the value of -log10(p-value) and the Fold Change (*i.e.* the log2 of the ratio of the mean values per condition). Finally, it is advised to save the results by clicking on "Save diff analysis". Then, a new "DiffAnalysis" dataset is created and loaded in memory. This dataset is the same as the previous one, except that three 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 3.2.6) of this new dataset. It becomes the new current dataset and its name, "DiffAnalysis.<test>" (where <test> indicates the test performed), appears in "Dataset versions". All plots and tables in other menus are automatically updated. Note that it is possible to keep stored in memory different "DiffAnalysis" datasets: one for each type of <test>.



	id	logFC	P.Value	Significant
10	10	-1.50192465004003	0.00000768569647037205	1
11	11	-1.63084289353196	0.00017730370606419	1
12	12	-1.78856451062824	0.0000382536871337211	1
13	13	-1.73373175682144	0.0000719432309892085	1
14	14	-1.41582407003764	0.00000941450556717109	1
15	15	-2.12242770429974	0.000211545774617801	1
16	16	-1.78238193213426	0.0000907544460088037	1
17	17	-1.51192780533265	0.0000160589994651451	1
18	18	-1.26164572409617	0.000393511686388655	1
19	19	-1.57455400751554	0.0000091237713955135	1

Showing 1 to 10 of 95 entries

Previous 1 2 3 4 5 … 10 Next

The differential analysis has been saved.

Figure 34: Table of the results of statistical test in the differential analysis tool

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".

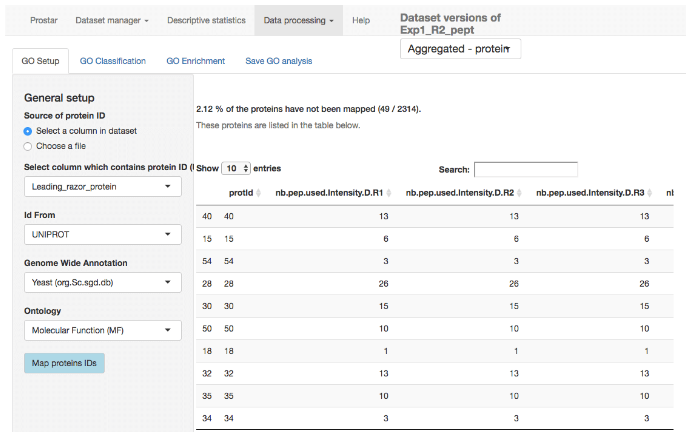
3.4.6 Gene Ontology analysis (Beta)

The Gene Ontology (GO, www.geneontology.org) is a controlled vocabulary for annotating three biological aspects of gene products. This ontology is made of three parts : Molecular Function (MF), Biological Process (BP) and Cellular Component (CC).

GO analysis is the last step proposed in the "Data processing" menu. It aims to provide the user with a "global view" of what is impacted (in a biological point of view) in his/her experiment, by showing which GO terms are represented (GO classification tab), or over-represented compared to a reference (GO Enrichment tab).

Prostar relies on the package *clusterProfiler* to perform both GO Classification and GO Enrichment. We propose a GO analysis interface with four separated tabs (see Fig. 35):

- GO Setup
- GO classification
- Go Enrichement
- Save GO analysis



Dataset versions of Exp1_R2_pept: Aggregated - protein

General setup

Source of protein ID
(● Select a column in dataset) ○ Choose a file

Select column which contains protein ID: Leading_razor_protein

Id From: UNIPROT

Genome Wide Annotation: Yeast (org.Sc.sgd.db)

Ontology: Molecular Function (MF)

2.12 % of the proteins have not been mapped (49 / 2314).

These proteins are listed in the table below.

Show 10 entries   Search:

	protId	nb.pep.used.Intensity.D.R1	nb.pep.used.Intensity.D.R2	nb.pep.used.Intensity.D.R3
40	40	13	13	13
15	15	6	6	6
54	54	3	3	3
28	28	26	26	26
30	30	15	15	15
50	50	10	10	10
18	18	1	1	1
32	32	13	13	13
35	35	10	10	10
34	34	3	3	3

Figure 35: **Input parameters for GO analysis (GO Setup tab)**

The left-hand side of the **GO Setup** tab allows it to set the input parameters, namely:

- Source of protein ID: user indicates either a column in the current dataset or chooses a file (1 ID per line).
- Id From: the type of ID supplied (UNIPROT by default).
- Genome Wide Annotation: the organism to consider for the analysis.
- Ontology: the level of the ontology to work with.

Once these parameters filled, clicking on "Map proteins IDs" launches the mapping of the IDs onto the GO categories of the annotation package. Then, on the right-hand side of the panel, the proportion of proteins that cannot be mapped onto the annotation package is indicated (this informative ouput does not interrupt the process, unless no protein maps).

Command line: The R function to map protein IDs on the annotation package is `bitr()` (function of the Bioconductor package *clusterProfiler*). In *DAPAR*, the call to the `bitr` is integrated inside the `group_GO()` and `enrich_GO()` functions. The mapping is done implicitly and the user does not have to perform the mapping.

Next step is to perform either GO Classification or GO Enrichment (or both).

In the **GO Classification** tab (see Fig. 36), one has to indicate which level(s) of the ontology to consider. Then clicking on the "Perform GO grouping" button launches the analysis (function `groupGO()` of the *clusterProfiler* package). The graphics shows the most represented GO categories for a user-defined ontology at (a) user-defined level(s).

Command line: The *DAPAR()* function to perform the GO classification is `group_GO()`. It returns a 'groupGOResult' instance. The plot as seen in the GO-Classification tab of ProStaR (see Fig. 36) can be generated with the `barplot()` function.

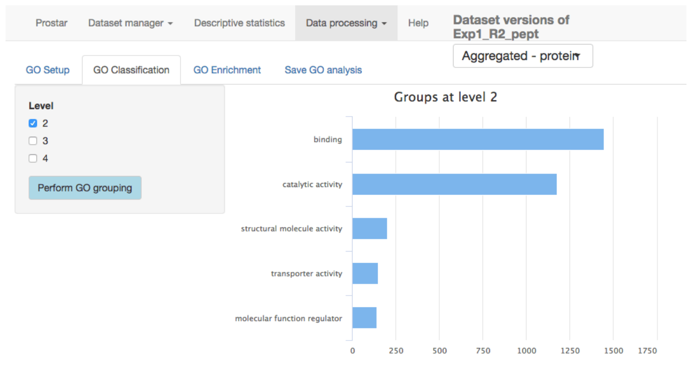
```
> ggo <- group_GO(data, idFrom="UNIPROT", orgdb=org.Hs.eg.db, ont="MF", level=2)
> barplotGroupGO_HC(ggo)
```

The **GO Enrichment** tab (see Fig. 37) allows it to know which GO categories are significantly enriched in the users list, compared to a chosen reference ('background' or 'universe'). This background can either be :

(i) the entire organism (in this case, the totality of the proteins identified with an "ENTREZGENE" ID in the annotation package specified in the GO Setup tab constitutes the background), or

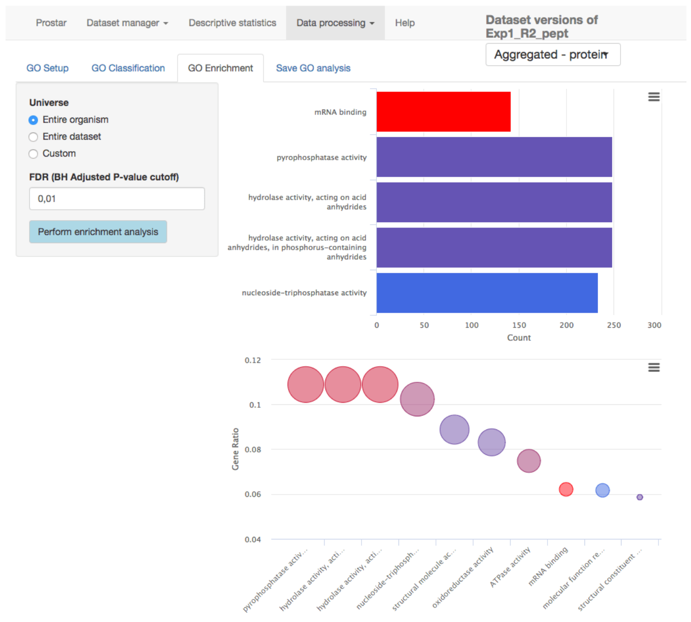
(ii) the entire dataset loaded in ProStaR (e.g. all the proteins IDs of the 'Leading_razor_protein' column of the dataset, as illustrated on Fig. 35), or

(iii) a custom IDs list provided in a separate file by the user (one ID per line).



Level
[✓ 2]
3
4

Figure 36: Tab to perform the GO classification



Universe
[✓ Entire organism]
Entire dataset
Custom

FDR (BH Adjusted P-value cutoff)
0,01

Figure 37: GO enrichment tab

The enrichment tab calls the `groupGO()` function of the *clusterProfiler* package. This function performs a significance test for each category, followed by a multiple test correction at a user-defined level. Concretely, this level is tuned thanks to the "FDR (BH Adjusted P-value cutoff)" field. Analysis is launched by clicking the "Perform enrichment analysis" button.

Once the analysis has been performed, the result is displayed via two graphics on the right-hand side of the panel (see Fig. 37). The first one (top) is a barplot showing the five most significant categories. The length of each bar represents the number of proteins within the corresponding category. The second one (bottom) is a dotplot ranked by decreasing *GeneRatio*, which reads:

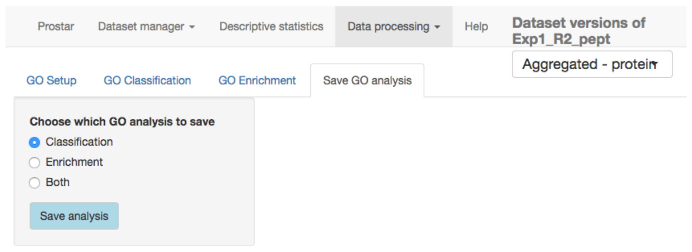
$$GeneRatio = \frac{\#(\text{Genes of the input list in this category})}{\#(\text{Total number of Genes in the category})}.$$

Command line: The *DAPAR()* function to perform the GO Enrichment is `enrich_GO()`. It returns an 'enrichResult' instance. The plots as seen in the GO Enrichment tab of ProStaR (see Fig. 37) can be generated with the `barplot()` and `dotplot()` functions. The universe argument allows to indicate the list of IDs considered as the reference (or background) to which the IDs input list of the user will be compared. To consider the whole organism IDs as a reference, one can extract it from the annotation package with the *DAPAR* function `univ_AnnotDbPkg()`. (see code below)

```
> univ.Hs<-univ_AnnotDbPkg(org.Hs.eg.db)
> ego<-group_GO(data, idFrom="UNIPROT", orgdb=org.Hs.eg.db, ont="MF",
+                 universe=univ.Hs, pval=0.05, pAdjustMethod="BH")
> barplotEnrichGO_HC(ego)
> scatterplotEnrichGO_HC(ego)
```

The last tab is the **Save GO analysis** one. It allows saving the results: GO classification, GO enrichment, or both (see Fig. 38). Then, a new GOAnalysis dataset is created and loaded in memory.

As usual in ProStaR, it is possible to export this new dataset via the Dataset manager menu , either in MSnSet or in Excel format.



Choose which GO analysis to save
● Classification
○ Enrichment
○ Both

Figure 38: Saving the GO analysis

Command line: In *DAPAR*, the function to save the results of the GO analysis in the MSnSet dataset is `GOAnalysisSave()`.

Nota Bene: The GO Analysis functionalities are in beta version and improvements will be considered in the future, such as for instance:

- authorizing a larger range of input format ID;

- handling external mapping file between ID and GO ontology (or even a custom ontology), so as to be able to work on other organisms than those for which a Bioconductor annotation package exists.

Any additional suggestion is welcome.

3.5 Help

The Help screen offers various information:

- **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 Versions of dataset

This major element of the Dataset manager is not in the corresponding menu, but on the contrary is detached on the right hand side of the navbar. The reason is, it is convenient to have a constant view on it. It is a drop-down menu that lists the different versions of dataset of interest, i.e. the restauration points that were progressively saved along the quantitative analysis.

Basically, each time the modifications of the current dataset are saved, the new dataset does not overwrite the previous one. 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 (peptide)" or "Original (protein)" depending on the data being related to peptides or proteins. This information is registered in the MSnSet file (the slot "typeOfData" of `experimentData(object)`). After the filtering step, if the user saves his/her results, another dataset becomes available, named "Filtered (peptide)" or "Filtered (protein)". Similarly, after the saving of the normalization, of the imputation of missing values, of the aggregation into proteins and of 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 "Dataset versions" drop-down menu.

At any time, the name of the current dataset and the type of data are 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. 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.

Remarks:

- 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 "Dataset versions" drop-down menu, and to observe the variations in the "Descriptive statistics" menu.

4 Bugs

Both packages *DAPAR* and *Prostar* are under active development. Despite our attention bugs may remain. To signal any, as well as typos, suggestions, etc. or even to ask a question, please contact us by email. Please join to the message as much information as possible, a reproducible example and the output of sessionInfo().

Here follow some error messages that the user may encounter and the tip to work around (please note that this section will be enriched with your feedbacks):

5 Session information

- R version 4.0.5 (2021-03-31), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 18.04.5 LTS`
- Matrix products: default

- BLAS: `/home/biocbuild/bbs-3.12-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.12-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Loaded via a namespace (and not attached): BiocManager 1.30.12, BiocStyle 2.18.1, compiler 4.0.5, digest 0.6.27, evaluate 0.14, htmltools 0.5.1.1, knitr 1.31, rlang 0.4.10, rmarkdown 2.7, tools 4.0.5, xfun 0.22, yaml 2.2.1