Prostar user manual

Samuel Wieczorek [1,*] ***and Thomas Burger*** [1,2,*]

[1]Univ. Grenoble Alpes, CEA, INSERM, BIG-BGE, 38000 Grenoble, France
[2]CNRS, BIG-BGE, F-38000 Grenoble, France
[*]firstname.lastname@cea.fr

November 22, 2018

Abstract

Prostar (Proteomics statistical analysis with R) is a Bioconductor distributed R package which provides all the necessary functions to analyze quantitative data from label-free proteomics experiments. Contrarily to most other similar R packages, it is endowed with rich and user-friendly graphical interfaces, so that no programming skill is required. This document covers the functionalities available in Prostar 1.14.3.

Package

Prostar 1.14.3; DAPAR 1.14.2; DAPARdata 1.12.1

Contents

1 Introduction . . . 4

2 Installation . . . 4

2.1 Zero-install . . . 5

2.2 Stand-alone Bioconductor install . . . 5

2.3 Server install . . . 6

2.4 *DAPAR* and *DAPARdata* (alone) . . . 7

3 Navigating through *Prostar* interface . . . 8

3.1 First contact with *Prostar* . . . 8

3.2 Synoptic view of *Prostar* interface . . . 9

3.3 Successive dataset versions . . . 11

4 Data type and format . . . 12

5 Data manager . . . 13

5.1 Open MSnSet file . . . 13

5.2 Convert data . . . 13

5.3 Demo mode . . . 17

5.4 Export . . . 18

5.4.1 Export a dataset . . . 18

5.4.2 Build a report (Beta) . . . 18

6 Data processing . . . 19

6.1 Filtering . . . 19

6.2 Normalization . . . 21

6.3 Imputation . . . 23

6.3.1 Focus on: Missing value nomenclature . . . 23

6.3.2 Protein-level imputation . . . 25

6.3.3 Peptide-level imputation . . . 28

6.4 Aggregation . . . 31

6.5 Hypothesis testing . . . 33

6.5.1 Peptide-level . . . 33

6.5.2 Protein-level . . . 33

7 Data mining . . . 34

7.1 Descriptive statistics . . . 34
7.1.1 Overview . . . 35
7.1.2 Missing values . . . 35
7.1.3 Data explorer . . . 36
7.1.4 Correlation matrix . . . 37
7.1.5 Heatmap . . . 38
7.1.6 PCA . . . 38
7.1.7 Intensity distributions . . . 38
7.1.8 CV distribution . . . 39

7.2 Differential analysis . . . 40
7.2.1 General methodology . . . 41
7.2.2 Focus on: Push p-values . . . 43
7.2.3 Focus on: Calibration plots . . . 44

7.3 Gene Ontology analysis . . . 45

8 Session information . . . 50

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 by bottom-up discovery proteomics experiments with a LC-MS/MS pipeline (Liquid Chromatography and Tandem Mass spectrometry). *DAPAR* (Differential Analysis of Protein Abundance with R) is an R package that contains all the necessary functions to process the data in command lines. It 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 statistical analysis with R) is a web interface based on Shiny technology (http://shiny.rstudio.com/) that provides GUI (Graphical User Interfaces) to all the *DAPAR* functionalities, so as to guide any practitioner that is not comfortable with R programming through the complete quantitative analysis process. The experiment package *DAPARdata* contains many datasets that can be used as examples. *Prostar* functionalities make it possible to easily:

- **Manage quantitative datasets**[1]: This includes import/export, conversion and report generation functionalities.
- **Perform a complete data processing chain**: (*i*) filtering and data cleaning; (*ii*) cross-replicate normalization; (*iii*) missing value imputation; (*iv*) aggregation of peptide intensities into protein intensities (optional); (*v*) null hypothesis significance testing.
- **Mine the dataset** at any step of the processing chain with various tools such as: (*i*) descriptive statistics; (*ii*) differential analysis; (*iii*) Gene Ontology (GO) analysis.

[1] Here, a quantitative dataset denotes a table where each analyte (either a peptide or a protein) is represented by a line and each replicate (or sample) is represented by a column; each cell of the table contains the abundance of a given analyte in a given replicate; the replicates are clustered into different conditions (or groups).

2 Installation

There are many ways to install *Prostar*, as well as its dependencies[2]:

1. **Zero-install**: The easiest way, so far only available on Microsoft Windows desktop machines.
2. **Stand-alone Bioconductor install**: The standard method to install Bioconductor distributed software. This method works for any operating systems (Unix/Linux, Mac OS X and Windows) as long as R is installed.
3. **Server install**: When one wants *Prostar* to run on a Unix server, on which remote users connect.

[2] i.e. Packages that are necessary for *Prostar* to correctly run.

In addition, it is also possible to only install *DAPAR* or *DAPARdata*, so as to work in command lines (for expert users only).

For all the desktop installs, we advise to use a machine with a minimum of 8GB of RAM (although there are no strict constraints).

2.1 Zero-install

This method works for any desktop machine with Microsoft Windows. It is not necessary to have R already installed. Go to http://www.prostar-proteomics.org and click on Zero-install download (Prostar menu). Download the zip file and unzip it. The unzipped folder contains an executable file which directly launches *Prostar*. Notice that at the first launch, an internet connection is necessary, so as to finish the install.

2.2 Stand-alone Bioconductor install

For this type of install, the operating system of the desktop machine can be of any type (Unix/Linux, Mac OS X or Windows). However, it is necessary to have the latest version of R (see Section 8) installed in a directory where the user has read/write permissions. Optionally, an IDE (integrated development environment) such as R Studio (https://www.rstudio.com/) may be useful to conveniently deal with the various R package installs.

To install *Prostar*, enter in the R console the following instructions:

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

Then, the following packages should be successfully installed: *MSnbase*, *RColorBrewer*, stats, *preprocessCore*, *Cairo*, *png*, *lattice*, *reshape2*, *gplots*, *pcaMethods*, *ggplot2*, *limma*, *knitr*, *tmvtnorm*, *norm*, *impute*, *doParallel*, *stringr*, parallel, *foreach*, *grDevices*, graphics, *openxlsx*, *utils*, *cp4p*, *scales*, *Matrix*, *vioplot*, *imp4p*, *imputeLCMD*, *highcharter*, *DAPARdata*, *siggenes*, *graph*, *lme4*, *readxl*, *clusterProfiler*, *dplyr*, *tidyr*, *tidyverse*, *AnnotationDbi*, *DAPAR*, *rhandsontable*, *data.table*, *shinyjs*, *DT*, *shiny*, *shinyBS*, *shinyAce*, *htmlwidgets*, *webshot*, *R.utils*, *shinythemes*, *XML*, *later*.

For a better experience, it is advised (but not mandatory) to install the development version of the following packages: DT and highcharter. To do so, install the devtools package and execute the following commands:

```
> devtools::install_github('rstudio/DT')
> devtools::install_github('jbkunst/highcharter')
```

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

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

A new window of the default web browser opens.

2.3 Server install

In the case of several *Prostar* users that are not confortable with R (programming or installing), it is best to have a single version of *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 (see Section 2.4), while on the other hand, the installation of *Prostar* is slightly different.

Shiny Server (https://github.com/rstudio/shiny-server) is a server program that makes Shiny applications available over the web. If Shiny Server is not installed yet, follow Shiny installation instructions.

Once a Shiny server is available, first install *Prostar* as described in Section 2.2, so as to have the dependencies installed.

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

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

The result of this command is now referred as */path_to_Prostar*.

Change the owner of the Shiny Server directory and log as shiny

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

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

Create the directory for the shiny application

```
# mkdir /srv/shiny-server/Prostar
```

Copy the ProstarApp directory within the shiny-server directory

```
# sudo cp -r /path_to_Prostar/ProstarApp/ /srv/shiny-server/Prostar
```

Change the owner of the Shiny Server directory

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

Give the following permissions to the www directory

```
# 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.

2.4 *DAPAR* and *DAPARdata* (alone)

This type of install should only be performed by advanced R users/programmers, or by the admin of a server version of *Prostar*.

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

DAPARdata is automatically installed with *Prostar* or *DAPAR*. However, it is possible to install it alone. Then, it follows the classical way for Bioconductor packages. In a R console, enter:

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

3 Navigating through *Prostar* interface

3.1 First contact with *Prostar*

Right after *Prostar* being launched, the web page depicted in Figure 1 shows up in the default browser (stand-alone install) or in a portable one (zero install). So far, the navbar only contains 3 menus: Prostar, Data manager and Help. However, as soon as data are loaded in the software, new menus contextualy appear (see Section 3.2).

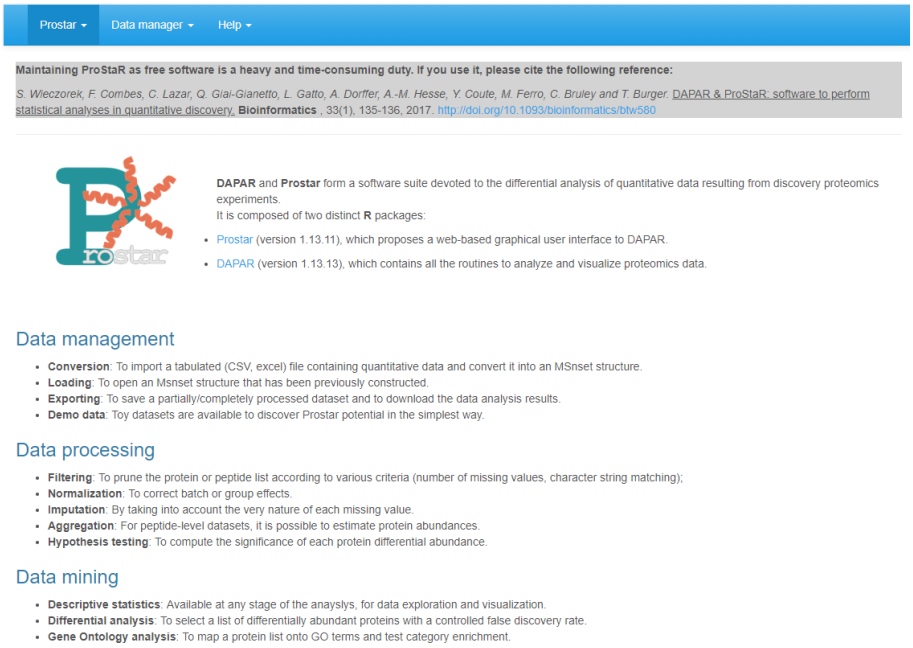


Figure 1: *Prostar* home screen

The **Data manager** is detailed in a dedicated section (Section 5). In the **Prostar** menu, one has access to:

- The **Home** screen[3], as illustrated on Figure 1.
- A **Global settings** menu, where it is possible to tune few parameters (such as colors, number of digits to display, etc.) that will affect the entire session.
- A page gathering **Release notes**, i.e. some information regarding the new features/changes/corrections with respect to the previous version.
- A **Check for updates** page, where it is possible to verify that the current version of *Prostar* is not outdated.

[3] Going back to the home screen does not re-initialize the session.

In the **Help** menu, one has access to:

- A list of **Useful links**: This page gather links towards numerous documents, such as scientific publications related to *Prostar* as well as user manuals, reference manuals and tutorials.
- A series of frequently asked questions (**FAQ**) is also available. This page should be regularly visisted!
- A form screen to **Report a bug** (see below for details).

Prostar is under active development, so that despite the developers' attention, bugs may remain. To signal any, as well as typos, suggestions, etc. or even to ask a question, please contact the developers.

3.2 Synoptic view of *Prostar* interface

When data are loaded for analysis, mor options are available in the navbar, as illustrated on Figure 2. In addition to the navbar, the screen is composed of a large working panel (below the navbar) and a drop-down menu on the upper right corner (which purpose is detailed in Section 3.3).

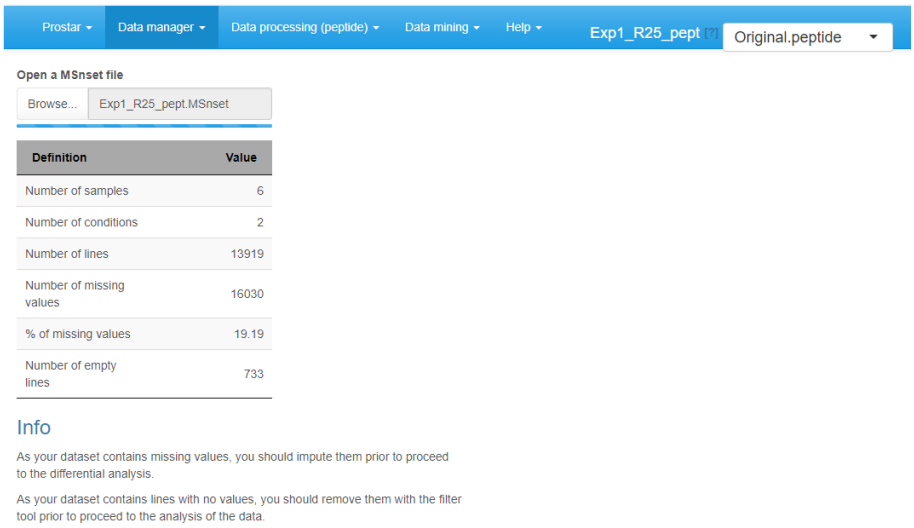


Figure 2: Screenshot illustrating the fully displayed navbar

Table 1 summarizes the content of the navbar. Let us note that depending on the dataset content (either proteins or peptides), the menu can slightly change: Notably, if protein-level dataset is loaded, then **Aggregation** (which purpose is to roll up from peptides to proteins) is not proposed in the **Data processing** menu.

Prostar (3.1)	Data manager	Data processing
Home	Open MSnset file (5.1)	Filter data (6.1)
Global settings	Convert data (5.2)	Normalization (6.2)
Release notes	Demo data (5.3)	Imputation (6.3)
Check for updates	Export (5.4)	Aggregation (6.4)
		Hypothesis testing (6.5)

Data mining	Help (3.1)
Descriptive statistics (7.1)	Useful links
Differential analysis (7.2)	FAQ
GO analysis (7.3)	Bug report

Table 1: Synoptic view of the ProStaR menus with the corresponding reference sections

The **Data manager** menu gathers all the functionalities that relates to data import, export or conversion:

- **Open MSnset file** (see Section 5.1) should be used to reload a dataset that has previously been converted and saved as an MSnset object.
- **Convert data** (see Section 5.2) should be used to load a new dataset from a tabular file containing quantitative proteomics data, such as a Maxquant or Proline output.
- **Demo data** (see Section 5.3) makes it possible for any new user to discover *Prostar* capabilities on toy datasets.
- **Export** (see Section 5.4) gathers all the functionalities to save a dataset in various formats, as well as to export the main results in a single document.

As soon as one of the three first options of the menu has been used to load a dataset, the **Data processing** and **Data mining** menus appear in the navbar.

Conducting a rigorous differential analysis requires a well-defined pipeline made of several tightly connected steps. In *Prostar*, this pipeline has been designed to be as general as possible. Thus, the **Data processing** menu contains numerous steps which, for a given dataset may not all be necessary, so that some can be occasionally skiped. However, the pipeline has been assembled and organized so as to propose coherent processing, so that the respective order of the different steps should be respected. These steps are the following:

1. **Filtering** (see Section 6.1): to remove some analytes (because of too many missing values, or because they are irrelevent)
2. **Normalization** (see Section 6.2): to correct batch effect and sample/condition biases

3. **Imputation** (see Section 6.3): to impute the remaning missing values
4. **Aggregation** (see Section 6.4): if the dataset is a peptide-level one, it is possible to aggregate them, so as to form a protein-level dataset
5. **Hypothesis testing** (see Section 6.5) of the differential abundance

It should be carefully noted that each of these steps modifies the dataset: In other words, the **Data processing** menu offers a succession of data transformations which should be performed by the user in an educated way.

As opposed to the **Data processing** menu, the **Data mining** menu offers a series of tools to analyze and visualize the dataset without transforming it. Thus, there is far less restriction on how and when applying them:

- The **Descriptive statistics** (see Section 7.1) panel is composed of several tabs which provide various "static" views on the dataset. It is advised to look at them between each processing step, so as to check how the dataset has been transformed.
- Once p-values have been computed (either at peptide or protein level) (see **Hypothesis testing** in the **Data processing** menu), it is possible to conduct a **Differential analysis** (see Section 7.2). This consists in a series of statistical computations which purpose is to isolate a subset of proteins deemed differentially abundant between the compared conditions, according to a specific false discovery rate threshold.
- As long as the dataset is a protein-level one, it is possible to conduct a Gene Ontology (GO) enrichment or analysis on it (see Section 7.3). However, if the dataset is a peptide-level one, this option does not appear.

3.3 Successive dataset versions

As explained above, each functionality in the **Data processing** menu transforms the current dataset. To authorize some flexibility and to avoid unwanted data corruption, it is possible to save in *Prostar* the original dataset, as well as each intermediate dataset along the processing chain. Concretely, it is possible to store one "original dataset", one "filtered dataset", one "normalized dataset", one "imputed dataset" and so on. Moreover, at any moment, it is possible to go back to a previous state of the dataset, either to restart a step that went wrong, or just to compare with the **Data mining** tools how the dataset was changed (rapid switching makes it easier to visualize it).

To create a new item in the dataset history, one simply has to click on the save button at the end of each processing step. Each time a new dataset is created, it is by default the one on which the processing goes on.

To navigate through the dataset history, one simply uses the drop-down menu of the upper right corner. Notice that if the user saves the current step (e.g. imputation), then goes back to a previous step (e.g. normalization) and start working on this older dataset (to perform 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. As a side effect, if any processing further than imputation was already done (e.g. aggregation), then, the aggregated dataset is not coherent anymore with the imputed one (as the new imputation cannot be automatically transmitted to update the previously tuned aggregation).

Finally, let us note that the name of each dataset version (normalized, imputed, etc.) also indicates if the dataset is a protein-level or a peptide-level one (as for instance the aggregation step transforms a peptide-level dataset into a protein-level one).

4 Data type and format

The quantitative data should fit into a matrix-like representation where each line corresponds to an analyte[4] and each column to a sample. Within the (i-th, j-th) cell of the matrix, one reads the abundance of analyte i in sample j.

Although strictly speaking, there is no lower or upper bound to the number of lines, it should be recalled that the statistical tools implemented in *Prostar* have been chosen and tuned to fit a discovery experiment dataset with large amount of analytes, so that the result may lack of reliability on too small datasets. Conversely, very large datasets are not inherently a problem, as R algorithms are well scalable, but one should keep in mind the hardware limitations of the machine on which *Prostar* runs to avoid overloading.

As for the number of samples (the columns of the dataset), it is necessary to have at least 2 conditions (or groups of samples) as it is not possible to perform relative comparison otherwise. Moreover, it is necessary to have at least 2 samples per condition[5], as otherwise, it is not possible to compute an intra-condition variance, which is a prerequisite to numerous processing.

[4] The analytes can be either proteins or peptides. However, in a given dataset, all the analytes should be of the same nature.

[5] With only two replicates per conditions, the computations are tractable. It does not mean that statistical validity is guaranteed. Classically, 3 replicates per conditions are considered a minimum in case of a controlled experiment with a small variability, mainly issuing from technical or analytical repetitions. Analysis of complex proteomes between conditions with a large biological variability requires more replicates per conditions (5 to 10).

The data table should be formatted in a tabulated file where the first line of the text file contains the column names. It is recommended to avoid special characters such as "]", "@", "$", "%", etc. that are automatically removed. Similarly, spaces in column names are replaced by dots ("."). Dot must be used as decimal separator for quantitative values. In addition to the columns containing quantitative values, the file may contain additional columns for metadata. Alternatively, if the data have already been processed by *Prostar* and saved as an MSnset file (see *MSnbase*), it is possible to directly reload them (see Section 5.1).

5 Data manager

The **Dataset manager** allows it to open, import or export quantitative datasets. *Prostar* relies on the MSnSet format which is part of the package *MSnbase*: It is either possible to load existing MSnSet files (see Section 5.1), or to import text (-tabulated) and Excel files (see Section 5.2). The **Demo data** menu allows it to load the datasets of the package *DAPARdata* as examples to discover *Prostar* functionalities (see Section 5.3).

5.1 Open MSnSet file

To reload a dataset that was already formated into an MSnSet file, click on **Open MSnset File**. This 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 (see Figure 2): It includes the number of samples, the number of proteins (or peptides) in the dataset, the percentage of missing values and the number of lines which only contain missing values. Once done, the menu displaying the version of the dataset appears and display "Original - peptide" or "Original - protein", depending on whether the file contains peptide-level or protein-level quantitative information (see Section 3.3). Similarly, the Data processing and Data mining menus become available.

5.2 Convert data

To upload data from tabular file (i.e. stored in a file with one of the following extensions: .txt, .csv, .tsv, .xls, or .xlsx) click on the upper menu **Data manager** then chose **Convert data**.

First, go to the **Select File** tab (see Figure 3): Click on the **Browse**... button and select the tabular file of interest[6] (If an Excel file is chosen, a drop-down menu appears to select the spreadsheet containing the data). Once the upload is complete, indicate whether it is a protein level dataset (i.e., each line of the data table should correspond to a single protein) or a peptide-level one. Indicate if the data are already log-transformed or not. If not they will be automatically log-transformed[7]. If the quantification software uses "0" in places of missing values, tick the last option "Replace all 0 and NaN by NA" (as in *Prostar*, 0 is considered a value, not a missing value).

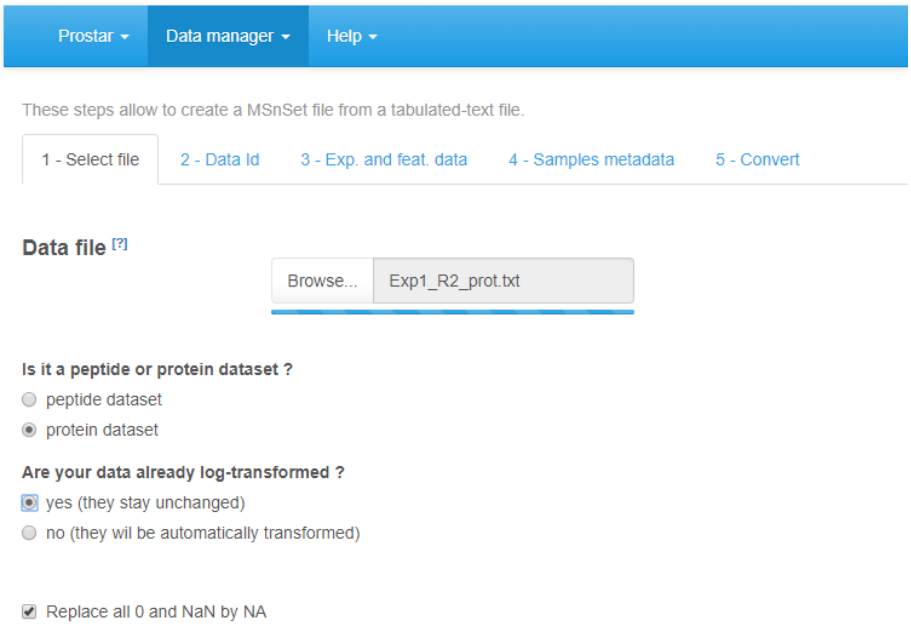


Figure 3: File conversion, step 1

Second, move on to the **Data Id** tab (see Figure 4): If the dataset already contains an ID column (a column where each cell has a unique content, which can serve as an ID for the peptides/proteins), select its name in the drop-down menu. Otherwise, it is possible to use the first option of the drop-down menu, that is the **Automatic ID definition**, which creates an artificial index. Finally, if the dataset is a peptide-level one, it is in addition important to indicate the column containing the IDs of the parent proteins, so as to prepare for future peptide to protein aggregation.

On the third tab (see Figure 5), referred to as **Exp. and feat. data**, select the columns which contain the protein abundances (one column for each sample of each condition). To select several column names in a row, click-on on the

[6] The *DAPARdata* package also contains tabular versions (in txt format) of the datasets available in the Demo data menu. Thus, it is also possible to test the import/export/save/load functions of *Prostar* with these toy datasets. Concretely, one simply has to import them from the folder where the R packages are installed, in the following sub-folder: `.../R/R-3.4.0/library/DAPARdata/extdata`. Note that each dataset is also available in the MSnset format, but these datasets should not be considered to test conversion functions from/to tabular formats.

[7] *Prostar* cannot process non-log-transformed data. Thus, do not cheat the software by indicating data on their original scale are log-transformed.

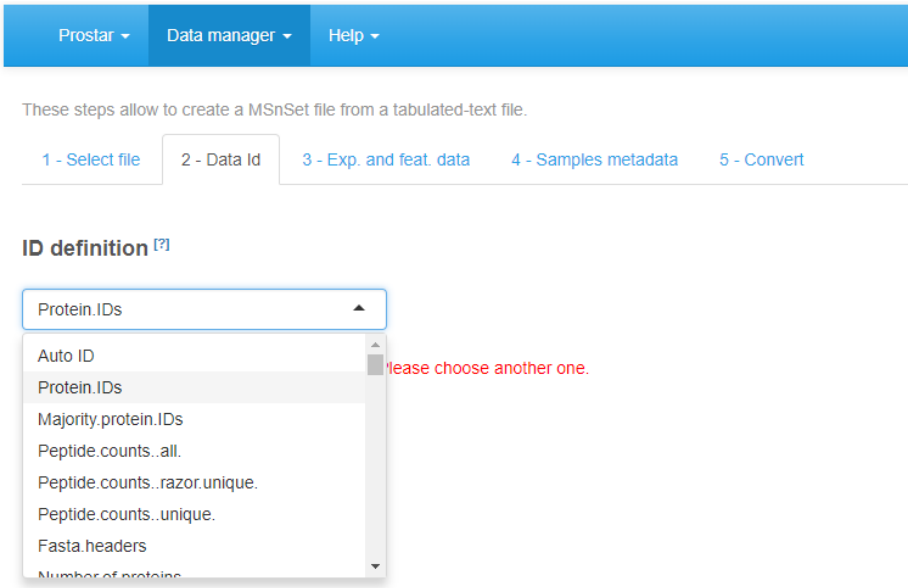


Figure 4: File conversion, step 2

first one, and click-off on the last one. Alternatively, to select several names which are not continuously displayed, use the **Ctrl** key to maintain the selection. If, for each sample, a column of the dataset provides information on the identification method (e.g. by direct MS/MS evidence, or by mapping) check the corresponding tick box. Then, for each sample, select the corresponding column. If none of these pieces of information is given, or, on the contrary, if all of them are specified with a different column name, a green logo appears, indicating it is possible to proceed (however, the content of the specified columns are not checked, so that it is the user's responsibility to select the correct ones). Otherwise (i.e. the identification method is given only for a subset of samples, or a same identification method is referenced for two different samples), then a red mark appears, indicating some corrections are mandatory.

Move on to the fourth tab, **Sample metadata** (see Figure 6). This tab guides the user through the definition of the experimental design. Fill the empty columns with as different names as biological conditions to compare (minimum 2 conditions and 2 samples per condition) and click on **Check conditions**. If necessary, correct until the conditions are valid[8]. When achieved, a green logo appears and the sample are reordered according to the conditions. Choose the number of levels in the experimental design (either 1, 2 or 3), and fill the additional column(s) of the table[9]. Once the design is valid (a green check logo appears), move on to the last tab.

[8] As an help, it is possible to refer to FAQ # 2

[9] In case of difficulty, either to choose the adapted design hierarchy or to fill the table design, it is possible to click on the interrogation mark beside the sentence "Choose the type of experimental design and complete it accordingly". Except for flat design, which are automatically defined, it displays an example of the corresponding design. It is possible to rely on this example to precisely fill the design table.

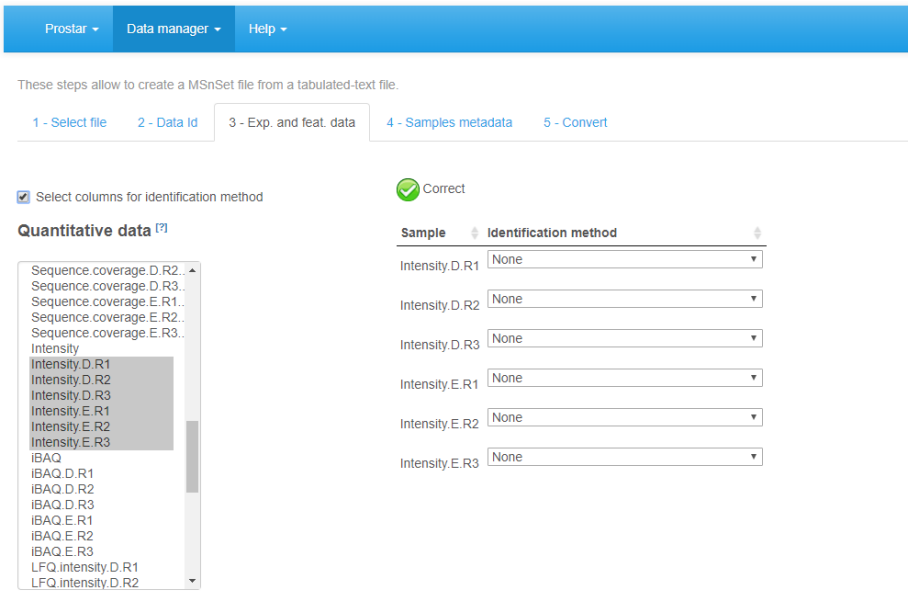


Figure 5: File conversion, step 3

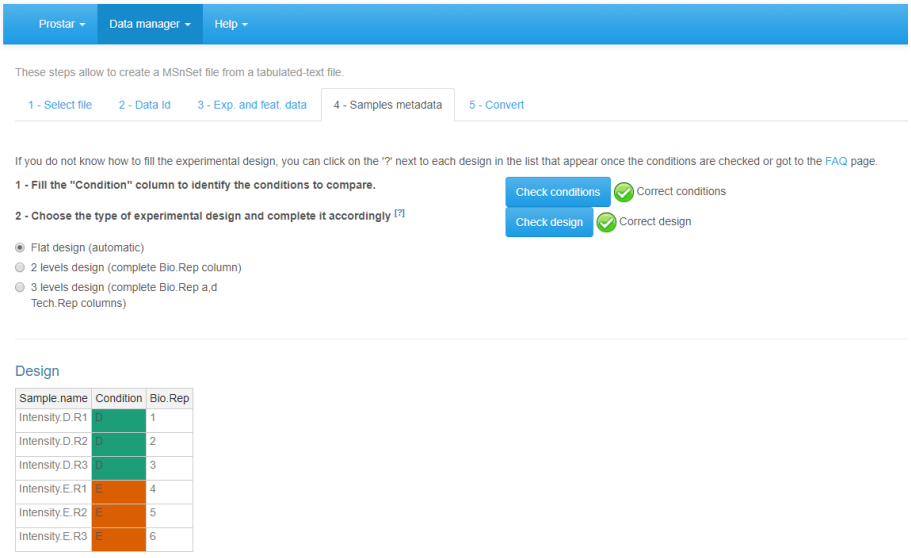


Figure 6: File conversion, step 4

Finally, move on to the **Convert** tab (see Figure 7). Provide a name to the dataset to be created and click on the **Convert** button. As a result, a new MSnset structure is created and automatically loaded. This can be checked with the name of the file appearing in the upper right hand side of the screen, as a title to a new drop-down menu. So far, it only contains "Original - protein" or "Original - peptide", but other versions of the dataset will be added along the

course of the processing. Pay attention to any red message appears below the **Convert data** button, which indicates a mistake or an incomplete parameter tuning that must be sorted out before converting the data.

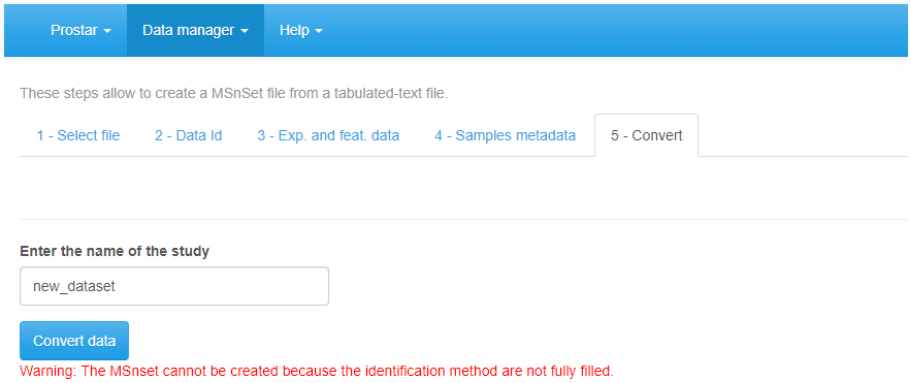


Figure 7: **Importing a CSV file, tab 5**

5.3 Demo mode

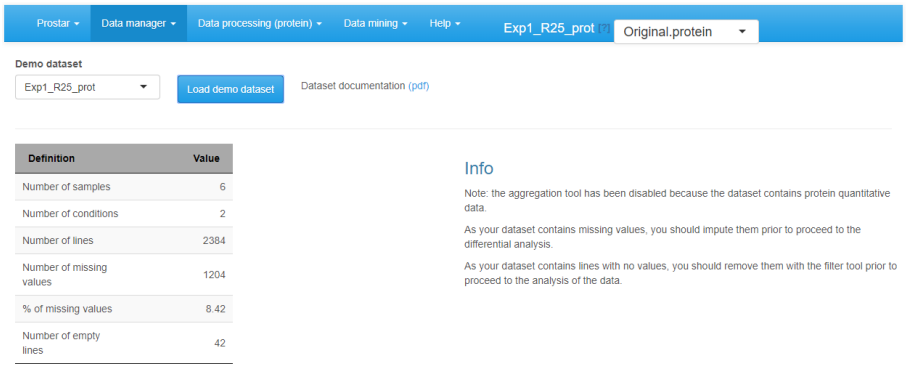


Figure 8: **Loading a demo dataset**

To ease *Prostar* discovery, a "demo mode" is proposed. In this mode, the datasets contained in the package *DAPARdata* can be directly uploaded to test *Prostar* functionalities. To do so, simply click on **Demo data** in the **Data manager** (Figure 8). Note that it possible to display the PDF vignette of the dataset directly from *Prostar* screen ("Dataset documentation (pdf)" link).

5.4 Export

The **Export** menu from the **Data manager** gathers all the functionality to save a dataset in various formats, or to compile the results in a scientific report.

5.4.1 Export a dataset

As importing a new dataset from a tabular file is a tedious procedure, we advise to save the dataset as an MSnset binary file right after the conversion (The, it becomes easy to reload it, as detailed in Section 5.1). This makes it possible to restart the statistical analysis from scratch if a problem occurs without having to convert the data another time. Moreover, it is also possible to export the dataset as an Excel spreadsheet (in xlsx format) or as a series of tabulated files grouped in a zipped folder. Any any case, the procedure is similar: First, choose the version of the dataset to be saved. Then, choose the desired file format and provide a file name[10]. Then, click on **Download** (Figure 9). Once the downloading is over, store the file in the appropriate directory.

[10] Optionally, it is possible to select a subset of the column metadata to make the file smaller.

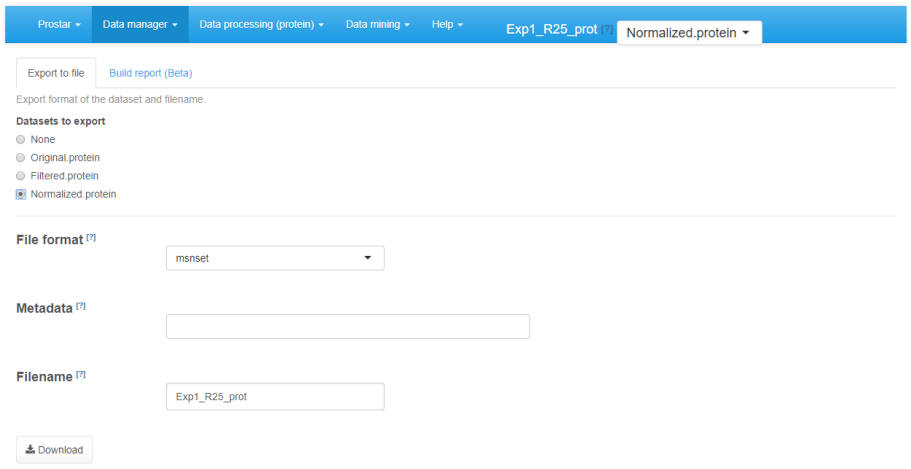


Figure 9: **Exporting to an Excel file**

5.4.2 Build a report (Beta)

The automatic reporting functionalities are under active development. However, they are still in Beta version and amenable to numerous modifications. This vignette will be completed with an exhaustive description of *Prostar* reporting functionality in a near future.

TODO: missing functionality

6 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. Imputation
4. Aggregation (only for peptide-level data)
5. Hypothesis testing

For each step, several algorithms or parameters are available, all of them being thoroughly detailed 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 7.1), 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 3.3). Thus, it is possible to go back to any previous step of the analysis if necessary, without starting back the analysis from scratch.

6.1 Filtering

In this step, the user may decide to delete several peptides or proteins according to two criteria: First is the amount of missing values (if it is too important to expect confident processing, see tab 1); Second is string-based filtering (some analyte can be deleted after having been tagged, such as for instance reverse sequences in target-decoy approaches, or known contaminants, see tab 2).

To filter the missing values (first tab called **Missing values**), the choice of the lines to be deleted is made by different options (see Figure 10):

- **None**: No filtering, the quantitative data is left unchanged. This is the default option;
- **Empty lines**: Only the lines with 100% of missing values (analytes that have been identified but not quantified) are filtered out.
- **Whole Matrix**: The lines (across all conditions) in the quantitative dataset which contain less non-missing value than a user-defined threshold are deleted;

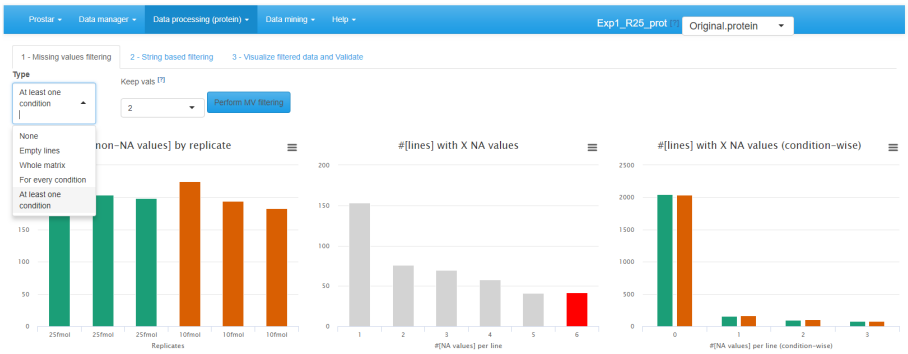


Figure 10: Interface of the filtering tool - 1

- **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;

To visualize the effect of the filtering options (without saving the changes or impacting the current dataset), just click on **Perform filtering**. If the filtering does not produce the expected effect, it is possible to 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.

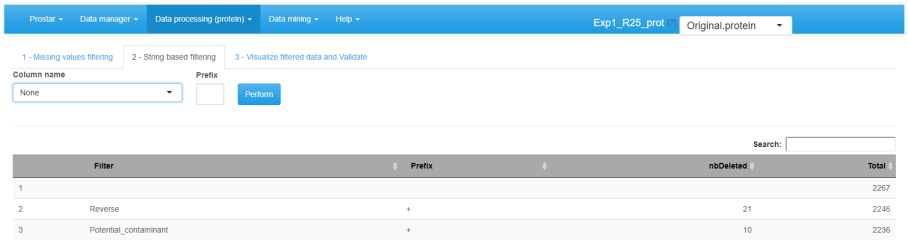


Figure 11: Interface of the filtering tool - 2

Afterward, proceed to the **String based filtering**, where it is possible to filter out proteins according to information stored in the metadata. To do so: Among the columns constituting the protein metadata listed in the drop-down menu, select the one containing the information[11] of interest (for instance, "Contaminant" or "Reverse")[12]. Then, specify in each case the prefix chain of characters that identifies the proteins to filter[13]. Click on **Perform** to remove the corresponding proteins. A new line appears in the table listing all the filters that have been applied. If other string-based filters must be applied, iterate the same process as many times as necessary.

[11] To work properly, the selected column must contain information encoded as a string of characters. For each protein, the beginning of the corresponding string is compared to a given prefix. If the prefix matches, the protein is filtered out. Otherwise, it is conserved in the protein list. Note that the filter only operates a prefix search (at the beginning of the string), not a general tag match search (anywhere in the string). Similarly, filters based on regular expressions are not implemented.

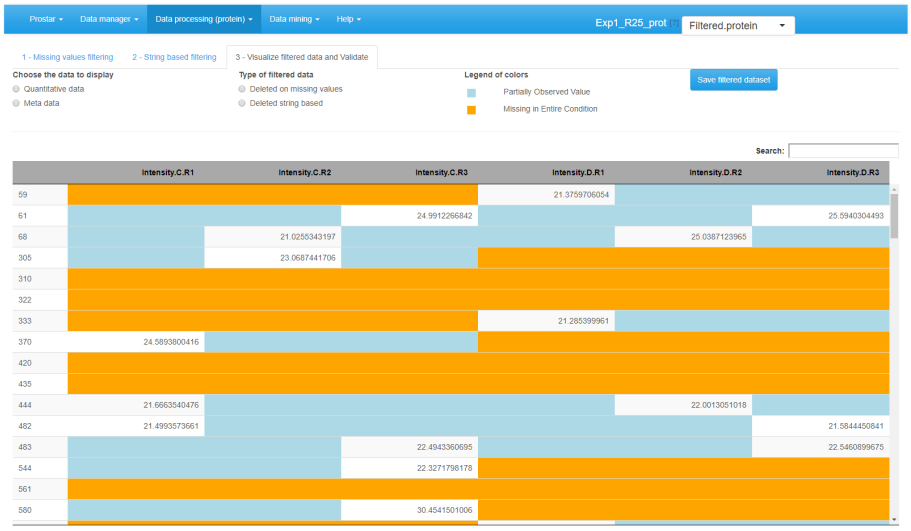


Figure 12: **Interface of the filtering tool - 3**

Once the filtering is appropriately tuned,go to the last tab (called **Visualize filtered data and validate**) (see Figure 12), to visualize the set of analytes that have been filtered. Finally, click on **Save filtered dataset**. A new dataset is created; it becomes the new current dataset and its name appears in the dropdown menu upper right corner of the screen. All plots and tables available in *Prostar* are automatically updated.

6.2 Normalization

The next processing step proposed by Prostar is data normalization. Its objective is to reduce the biases introduced at any preliminary stage (such as for instance batch effects). *Prostar* offers a number of different normalization routines that are described below.

To visualize the influence of the normalization, three plots are displayed (see Figure 13): The first two plots are those of the **Intensity distribution** tab of the **Descriptive statistics** (see Section 7.1.7). The last one depicts the distortion induced by the chosen normalization method on each sample.

Choose the normalization method among the following ones:

1. **None**: No normalization is applied. This can be useful in case of tightly calibrated proteomics experiments (such as spiked-in or benchmark samples), where normalization may be more harmful than useful. Moreover, when testing several normalizations, clicking on "None" is useful to recover the original distribution, which serves as reference to compare the methods.

[12] In datasets resulting from a MaxQuant, metadata indicates under a binary form which proteins are reversed sequences (resulting from a target-decoy approach) and which are potential contaminants. Both of them are indicated by a "+" in the corresponding column (the other proteins having an NA instead). It is thus possible to filter both reversed and contaminants out by indicating "+" as the prefix filter. However, if adequately encoded, filtering on other type of information is possible.

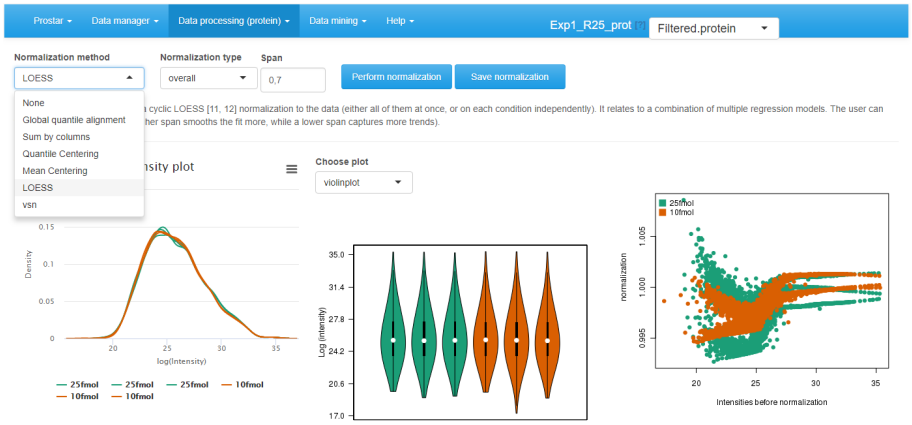


Figure 13: **Interface of the normalization tool**

2. **Global quantile alignment**: The Quantile of the intensity distributions of all the samples are equated, as proposed by the `normalize.quantiles()` of *preprocessCore*. This method proposes a normalization of important magnitude that should be cautiously used. It practically amounts to replace abundances by order statistics.

3. **Column sums**: The total intensity values of all the samples are equated. The rationale behind is to normalize according to the total amount of biological material within each sample. Thus, 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)

4. **Quantile Centering**: A given quantile of the intensity distribution is used as reference[14]. It proposes 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.

5. **Mean Centering**: sample intensity distributions are aligned on their mean intensity values (and optionally, the variance distributions are equated to one).

6. **VSN**: (or Variance Stabilizing Normalization) from the *vsn* package. Its objective is to have the analyte variance independent of the intensity. Note that this normalization is computed on the original intensity values, not the log2 transformed ones. The result from the normalization is not in

[13] If one has no idea of the prefixes, it is possible to switch to the **Data Explorer** in the **Descriptive Statistics** menu (see Section 7.1), so as to visualize the corresponding metadata.

[14] This normalization method should not be confused with Global quantile alignment.

the log2 scale, nor in the original one, but in-betwee: the scale is roughly linear for low-intensity values and roughly logarithmic for high-intensity values (so as to avoid the logarithm singularity).

7. **LOESS normalization**: The intensity values are normalized by means of a local regression model of the difference of intensities as function of the mean intensity value, as implemented in the `normalizeCyclicLoess()` from *limma* package.

Then, for each normalization method, the interface is automatically updated to display the method parameters that must be tuned. Notably, for most of the methods, it is necessary to indicate whether the method should apply to the entire dataset at once (the **overall** tuning), or whether each condition should be normalized independently of the others (the **within conditions** tuning).

Other parameters are method specific:

- **Quantile centering**: It is necessary to define the quantile of the intensity distribution that is used as reference. The most classically used is the median (quantile 0.5), but other quantiles can aslo be useful, depending on the dataset. Notably, as the lower limit of quantitation is roughly constant over the samples, relying on a lower quantile such as 0.15 can be efficient to alilgn the signal/noise ratios.
- **Mean centering**: It is necessary to indicate whether the distribution variances are scaled to 1.
- **LOESS**: The span corresponds to the proportion of the other analytes that are considered to perform the regression.

Once the method is correctly parametrized, click on **Perform normalization**. Observe the influence of the normalization method on the graphs. If the result of the normalization does not correspond to the expectations, change the normalization method or change its tuning. Once the normalization is effective, click on **Save normalization**. Check that a new version appears in the dataset version drop-down menu, referred to as **Normalized - Protein** or **Normalized - Peptide**.

6.3 Imputation

6.3.1 Focus on: Missing value nomenclature

Classically, missing values are categorized according to their underlying missingness mechanism:

- **Missing Completely At Random (MCAR)**: This category of missing values gather all those which result from the combination and propagation of multiple minor errors or stochastic fluctuations. As a result, each missing value cannot be directly explained by the nature of the analyte, nor by its measured intensity. MCAR values are assumed to distribute uniformely in the dataset as well as along the intensity range.
- **Missing Not At Random (MNAR)**: On the contrary, MNAR have a more precise cause. Notably, in mass spectrometry-based analysis, chemical species whose abundance are too close to the lower limit of detection of the instrument record a higher rate of missing values. This is why, MNAR-devoted imputation methods used in proteomics focus on left-censored data (that is, the distribution of which with respect to the abundance is truncated on the left side, i.e. on the region depicting the lower abundances).

However, it also makes sense to classify the missing value according to the analyte they impact, regardless of the underlying mechanism. This is why, in *Prostar*, it has been decided to separate:

- **POV** (standing for **Partially Observed Value**).
- **MEC** (standing for **Missing in the Entire Condition**).

All the missing values for a given protein in a given condition are considered POVs **if and only if there is at least one observed value for this protein in this condition**. Alternatively, **if all the intensity values are missing for this protein in this condition**, the missing values are considered MECs. As a result, each missing values is either POV or MEC. Moreover, for a given protein across several conditions, the missing values can split into POVs and MECs, even though within a same condition they are all of the same type.

With the default color setting, POVs are depicted in light blue and MECs in light orange.

In *Prostar*, the following assumptions are made:

- **In peptide-level datasets**, it is either possible to consider all the missing values, either POVs or MECS, as of the same type and to blindly apply one of the **basic methods** proposed. As a more refined alternative, we propose to rely on *imp4p*, which considers that:
  - MECs should always assumed to be MNAR. Note that imputing MEC values is a real issue as, in the condition of interest, there is no observed value to rely the imputation on.

- No assumption should be made on POVs: they can be a mix of MCAR and MNAR values, and the missingness mechanism of each is diagnosed before the imputation, thanks to an *imp4p* routine.

- **In protein-level datasets**:
  - The number of POVs is generally not large enough to perform an accurate diagnosis of the POV missingness mechanism, as proposed with *imp4p* at peptide-level. Thus, they are all assumed to be MCAR.
  - MECs are always assumed to be MNAR and imputed accordingly.

As a whole, it is advised to work at peptide-level rather than protein-level, and to use the refined imputation mechanism of *imp4p*.

Remarks:

- Sometimes, an analyte (and not a missing value) can be referred to as **Lapala**, when it has both observed values and MEC (which means it is observed in some conditions and not observed in others). This term was coined from French "là/pas-là"(meaning "here/not-here").
- Contrarily to the normalization step, where it is possible to choose between **within conditions** and **overall** application of the algorithm, the imputation step cannot be applied overall-wise, but only condition-wise. The rationale behind is that in case of very different conditions, it does not make sense to rely on the observed values of one condition to impute those of the others. With this regards, *Prostar* strongly differs from most of the other proteomics software tools, where the imputation can borrow information from multiple independent conditions.

6.3.2 Protein-level imputation

As a consequence of the previous paragraph, in protein-level datasets, *Prostar* proposes to use an MCAR-devoted imputation algorithm for POVs, and an MNAR-devoted one for MECs.

On the first tab (see Figure 14), select the algorithm to impute POV values, among the following ones, and tune its parameter accordingly:

- **None**: No imputation is performed for POVs.
- **SLSA** (Structured Least Square Adaptative): It is a regression based imputation method which account for a possible hierachical design. It does not require any parameter tuning.

Figure 14: **POV imputation**

- **DetQuantile**: It is advised to use DetQuantile to impute MECs rather than POVs, however, in some case, it may be interesting to rely on a single imputation method for all the missing values. This is why, this method is proposed also proposed for POV imputation. DetQuantile description and parameter tuning is described below.
- **KNN** (K-nearest neighbors): KNN imputation proposes to estimate each missing value by the mean of the observed values of other proteins with a similar intensity pattern (called neighbors). One only has to tune K, the number of neighbors to account for. The method simply wraps the one of the *impute* package, however, contrarily to the classical implementation and use, the function processes one condition at a time, rather than all the conditions at once.

According to our expertise, we advise to select the **SLSA** algorithm from *imp4p* but the other methods can also be of interest in specific situations.

The first distribution plot depicts the mean intensity of each condition conditionally to the number of missing values it contains. It is useful to check that more values are missing in the lower intensity range (due to left censorship).

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 (MCAR or MNAR).

Click on **Perform Imputation**. A short text shows up to summarize the result of the imputation, but the graphics are not updated. However, the next tab is enabled, on which the plots are updated with the imputation results.

After POVs, it is possible to deal with MECs. As a matter of fact, it is always dangerous to impute them, as in absence of any value to rely on, the imputation is arbitrary and risks to spoil the dataset with maladjusted values. As an alternative, it is possible to (1) keep the MEC as is in the dataset, yet, it may possibly impede further processing, (2) discard them at the filter step (see Section 6.1) so as to process them separately. However, this will make it impossible to include these proteins (and their processing) in the final statistics, such as for instance FDR.

For MEC imputation, several methods are available (see Figure 15):

- **None**: If MEC are not going to be imputed, this is what to select.
- **DetQuantile**: It proposes to impute each missing value within a given sample by a deterministic value (usually a low value). The rationale behind is that MEC values corresponds to proteins that are below the quantification limit in one condition, so that they should not be imputed according to observed values in the other conditions. Although the use of a deterministic value slightly disturb the intensity distribution, it makes the MEC values easy to spot, as they all correspond to a known numerical value. Concretely, the imputation value is chosen as a (small) **quantile** -the first parameter) of each sample distribution. Depending on the dataset, a quantile between 1% and 5% should be used, depending on the stringency you want to apply on proteins quantified only in one condition of a pairwise comparison. In case of a dataset with too few proteins, the lower quantile may lead to instable values. In such a case, we advise to use a larger quantile value (for instance 10% or greater) but to use a smaller multiplying **factor** (the second parameter) so as to keep the imputation value reasonably small with respect to the detection limit (for instance, consider a factor of 0.2 or smaller). In any case, when using detQuantile, the list of imputation values for each sample appears above the graphics.
- **Fixed value**: As an alternative, it is possible for the user to tune the imputation with a specific value that will be used in all samples of all conditions. Although appealing for its simplicity, this method may lead to important data distortion, so it is advised to use **detQuantile** algorithm whenever possible.

Based on our experience, we advise to use **detQuantile** algorithm.

Click on **Perform Imputation**. A short text shows up to summarize the result of the imputation, but the graphics are not updated. As the imputation is finished, the updated plots would not be informative, so they are not displayed on the final tab (referred to as **Validate & save**). Note that when displaying or

exporting the data, the color code used with missing values is still used for the imputed values, so that at any moment, it is possible to trace which proteins were imputed.

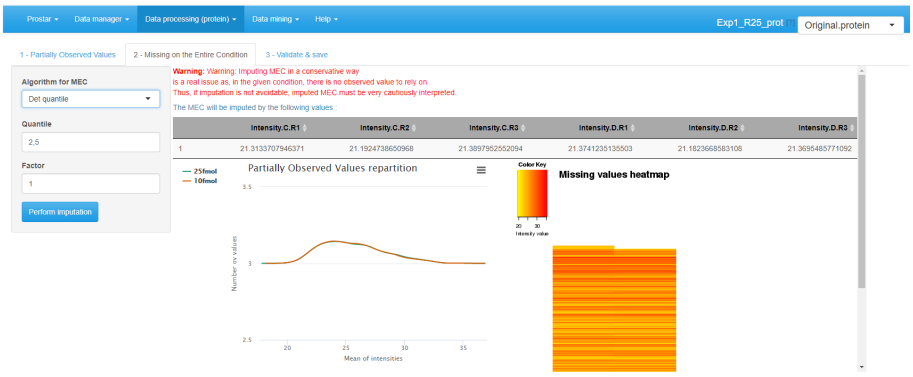


Figure 15: **POV imputation**

6.3.3 Peptide-level imputation

Notice that at peptide level, many lines of the dataset may corresponds to identified peptides with no quantification values. In order to avoid spoiling the dataset with their meaningless imputed values, it is demanded to filter them before proceeding (see Section 6.1).

Two plots are available in order to facilitate the understanding of the missing value distribution. These plots are the same as for protein-level imputation:

The distribution plot on the left depicts the mean intensity of each condition conditionally to the number of missing values it contains. It is useful to check that more values are missing in the lower intensity range (due to left censorship).

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 (MCAR or MNAR).

Figure 16: Interface of the imputation of missing values tool

To impute the missing peptide intensity values, it is either possible to rely on classical methods of the state of the art, or to use *imp4p*. The former ones are available by choosing **Basic methods** in the **Algorithm** drop-down menu (see Figure 16). Another drop-down menu appears proposing one of the following methods:

- **None**: No imputation is performed. This option is convenient to recover the original missing value distribution when comparing several imputation methods.
- **DetQuantile**: It proposes to impute each missing value within a given sample by a deterministic value (usually a low value). The rationale behind is that MEC values corresponds to peptides that are below the quantification limit in one condition, so that they should not be imputed according to observed values in the other conditions. Although the use of a deterministic value slightly disturb the intensity distribution, it makes the MEC values easy to spot, as they all correspond to a known numerical value. Concretely, the imputation value is chosen as a (small) **quantile** -the first parameter) of each sample distribution. Depending on the dataset, a quantile between 1% and 5% should be used, depending on the stringency you want to apply on proteins quantified only in one condition of a pairwise comparison. In case of a dataset with too few analytes (which is more likely to be the case at protein-level than at peptide-level), the lower

quantile may lead to instable values. In such a case, we advise to use a larger quantile value (for instance 10% or greater) but to use a smaller multiplying **factor** (the second parameter) so as to keep the imputation value reasonably small with respect to the detection limit (for instance, consider a factor of 0.2 or smaller). In any case, when using detQuantile, the list of imputation values for each sample appears above the graphics.

- **KNN** (K-nearest neighbors): KNN imputation proposes to estimate each missing value by the mean of the observed values of other peptides with a similar intensity pattern (called neighbors). One only has to tune K, the number of neighbors to account for. The method simply wraps the one of the *impute* package, however, contrarily to the classical implementation and use, the function processes one condition at a time, rather than all the conditions at once.
- **MLE** (Maximum Likelihood Estimation): This method, from the *norm* package, propose to estimate the expected abundance value of each peptide in each condition (with the maximum likelihood estimate), and to use this expected value as replacement of missing values. There is no parameter to tune with MLE imputation.

All these methods are applied on all missing values, without diagnosing their nature, as explained in Section 6.3.1. Alternatively, it is possible to rely on *imp4p* (see Figure 17). It works as follows:

1. For POVs, a diagnosis of the nature of each missing value is perform. As an output each POV is endowed with a probability of being either MNAR or MCAR.
2. Several iterations (as many as tuned on the interface, default is 10 - the more iteration, the more accurate the imputation) of the following algorithm are performed:
   - Randomly assign each POV to either MCAR or MNAR, according to its diagnosis probability.
   - If it is MCAR, SLSA algorithm is used for imputation.
   - If it is MNAR, IGCDA algorithm is used for imputation.
3. All the POV imputed values computed along the iterations are average to provide the imputation final results.
4. Optionally, it is possible to impute the MECs. To do so, one relies on a slightly modified version of IGCDA, which requires tuning 2 parameters: First is the upper bound of the MNAR imputed values, expressed as a quantile of the distribution of observed values in each sample (default is centile 2.5). Second is the probability distribution defined on the trange

of MNAR values (either uniform or beta): Beta is more conservative, as it provides more weight on values that are close to the detection limit, which reduces the risk of creating artificial differentially abundant peptides (and thus proteins).

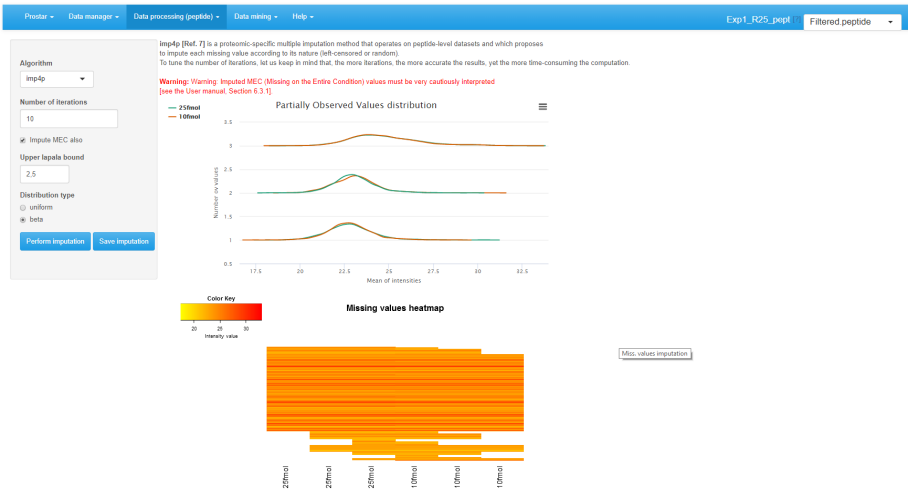


Figure 17: **Interface of the imputation of missing values tool**

It is possible to directly visualize the effect of an imputation method on the updated plots. If the imputation does not produce the expected effect, it is possible to test another one. To do so, one chooses another method in the list and click on **Perform imputation**. This action does not modify the dataset but offers a preview of the imputed quantitative data. It is possible to visualize as many imputation methods/tuning as desired. Once the correct one is found, one validates the choice by clicking on **Save imputation**. Then, a new "imputed" dataset is created and loaded in memory. This new dataset becomes the new current dataset and the "Imputed" version appears in the upper right drop-down menu. All plots and tables in other menus are automatically updated.

6.4 Aggregation

This steps only hold for peptide-level dataset. Its purpose is to build a protein-level dataset on the basis of the peptide intensities that have been processed in *Prostar*.

If this step has not been fulfilled at the initial data conversion (see Section 5.2), it is first necessary to specify which column metadata of the peptide-level dataset contains the parent protein IDs (which will be used to index the protein-level dataset that is going to be created).

Once the parent protein IDs are specified, two barplots are displayed (see Figure 18). 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 understand the distribution of shared peptides, as well as the peptide/protein relationships.

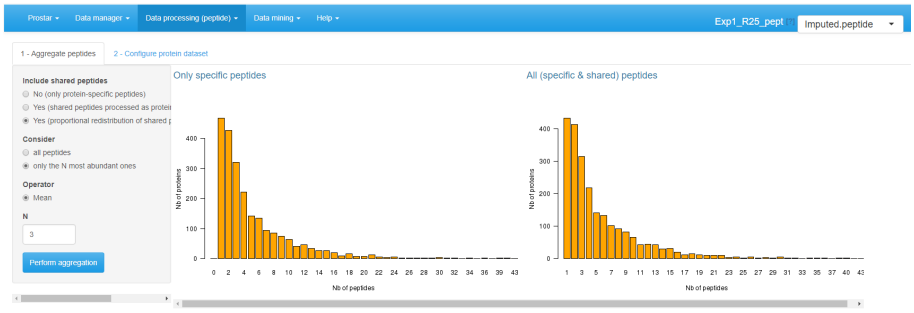


Figure 18: **Aggregation screen**

Aggregation requires to tune the following parameters:

- **Include shared peptides**: Here, one defines if protein intensity should be defined only by aggregating protein-specific peptides, or if (and thus how) shared peptides should be included. If shared peptides are not included, one faces the risk of losing some proteins (those which do not have a single specific peptide). Three options are available:
  - **No (only protein-specific peptides)**.
  - **Yes (shared peptides processed as protein specific)**.
  - **Yes (proportional redistribution of shared peptides)**. This last option is the most accurate one, yet, it is slower, for proportional redistribution is iterated as many time as necessary to reach convergence.
- **Consider**: The user specifies which peptides should be considered for aggregation.
  - **All peptides**, regardless of their intensities.
  - **Only the N most abundant ones**. In such a case, one has to specify a value for N.
- Finally, one defines the aggregation **Operator**: Sum or Mean. However, in case of proportional redistribution of the shared peptide intensities, only the mean operator is available.

Once the aggregation is appropriately tuned, click on **Perform aggregation**.

On the second tab, one selects the columns of the peptide dataset that should be kept in the metadata of the protein dataset (e.g. the sequence of the peptides). For any given parent-protein, the corresponding information of all of its child-peptides will be grouped and stored in the protein dataset metadata. Once done, one clicks on **Save aggregation**. This creates a new "aggregated" dataset that is directly loaded in memory.

As the new dataset is a protein one, the **Aggregation** menu has been disabled. Thus, the interface automatically switches to the homepage. However, the aggregated dataset is accessible and can be analyzed (in the **Data mining** menu) and processed (in the **Data processing** menu).

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 afterward.

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. Notably, all those which are meaningless at a protein-level are not displayed anymore.

6.5 Hypothesis testing

6.5.1 Peptide-level

TODO: missing functionality

6.5.2 Protein-level

For datasets that do not contain any missing values, or for those where these missing values have been imputed, it is possible to test whether each protein is significantly differentially abundant between the conditions. To do so, click on **Hypothesis testing** in the **Data processing** menu (see Figure **??**).

First, choose the test contrasts. In case of 2 conditions to compare, there is only one possible contrast. However, in case of $N \geq 3$ conditions, several pairwise contrasts are possible. Notably, it is possible to perform N tests of the **1vsAll**

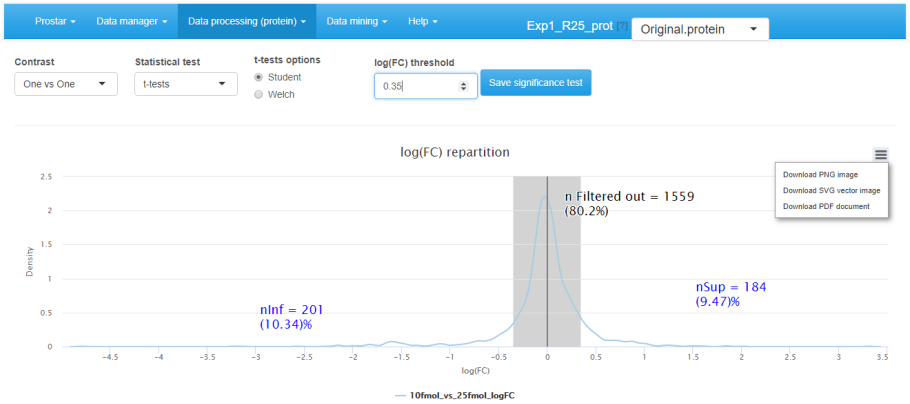


Figure 19: Hypothesis testing step

type, or $N(N-1)/2$ tests of the **1vs1** type. Then, choose the type of statistical test, between limma (see the *limma* package) or t-test (either Welch or Student, up to the user's choice). This makes appear a density plot representing the log fold-change (logFC) (as many density curves on the plot as contrasts). Thanks to the FC density plot, tune the **log(FC) threshold**. We advise to tune the logFC threshold conservatively by avoiding discarding to many proteins with it. Moreover, it is important tune the logFC to a small enough value, so as to avoid discarding too many proteins. In fact, it is important to keep enough remaining proteins for the next coming FDR computation step (see Section 7.2.2), as *(i)* FDR estimation is more reliable with many proteins, *(ii)* FDR, which relates to a percentage, does not make sense on too few proteins. Finally, run the tests and save the dataset to preserve the results (i.e. all the computed p-values). Then, a new dataset containing the p-values and logFC cut-off for the desired contrasts, can be explored in the **Differential analysis** tab available in the **Data mining** menu.

7 Data mining

7.1 Descriptive statistics

Several plots are proposed to help the user have a quick and as complete as possible overview of the dataset. This menu is an essential element for to check that each processing step gives the expected result.

7.1.1 Overview

This panel simply displays a table summarizing various quantitative elements on the datasets (see Figure 20). It roughly amounts to the data summary that is displayed along with each demo dataset (see Section 5.3).

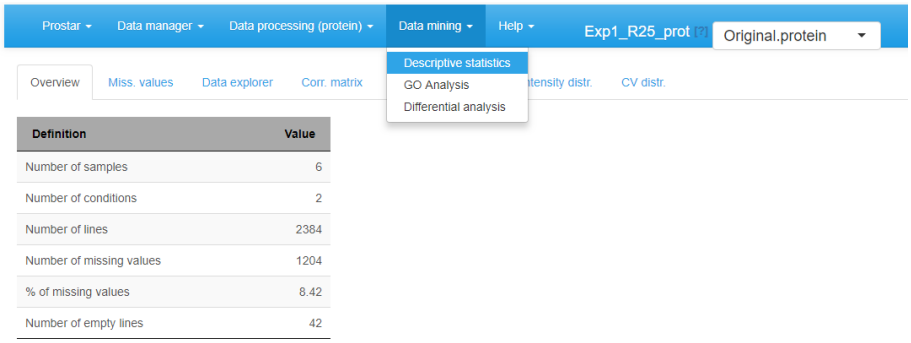


Figure 20: Home screen of the Descriptive statistics menu

7.1.2 Missing values

On the second tab, barplots depicts the distribution of missing values: the left hand side barplot represents the number of missing values in each sample. The different colors correspond to the different conditions (or groups, or labels). The second barplot (in the middle) displays the distribution of missing values; the red bar represents the empty protein count (i.e. the number of lines in the quantitative data that are only made of missing values). The last barplot represents the same information as the previous one, yet, condition-wise. Let us note that protein with no missing values are represented in the last barplot while not on the second one (to avoid a too large Y-scale).

Figure 21: Histrograms for the overview of the missing values

7.1.3 Data explorer

The third tab makes it possible to view the content of the MSnset structure. It is made of three tables, which can be displayed one at a time thanks to the radio button menu. The first one, named **Quantitative data** contains the intensity values (see Figure 22).

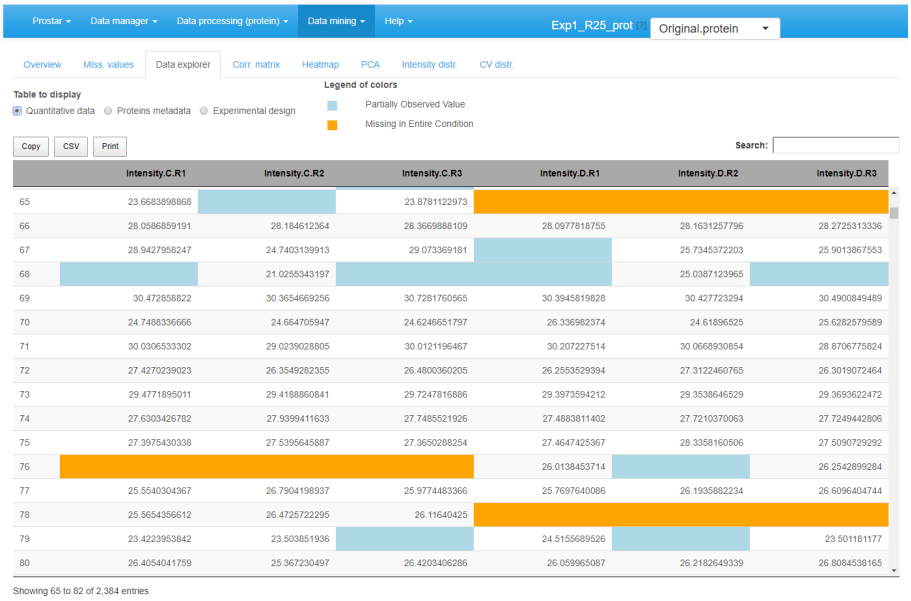


Figure 22: View of quantitative data in the MSnSet dataset

The missing values are represented by colored empty cells. The second one is referred to **Protein metadata**. It contains all the column dataset that are not the quantitative data (see Figure 23).

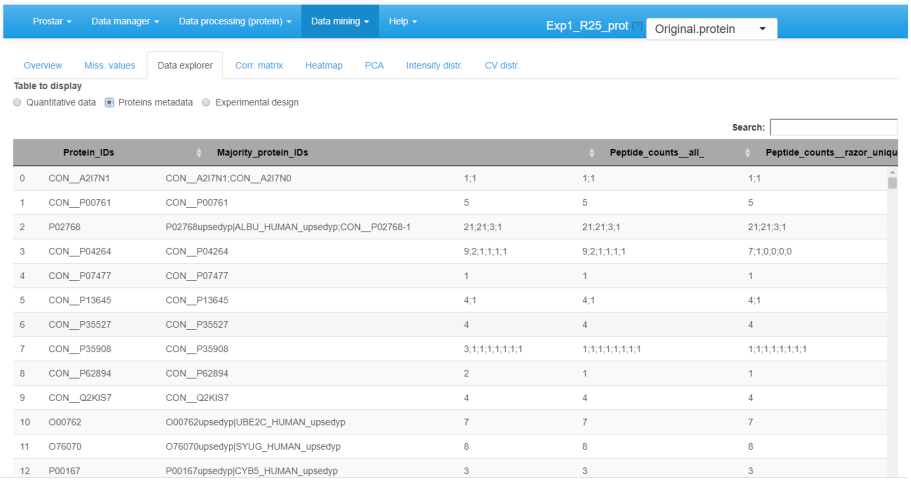


Figure 23: View of feature meta-data in the MSnSet dataset

The third tab, **Replicate metadata**, summarizes the experimental design, as defined when converting the data (see Section 5.2).

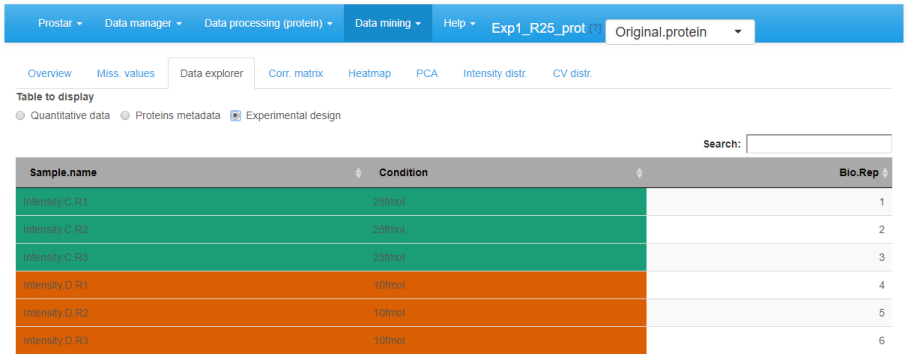


Figure 24: View of samples meta-data in the MSnSet dataset

7.1.4 Correlation matrix

In this tab, it is possible to visualize the extent to which the replicates correlate or not (see Figure 25). The contrast of the correlation matrix can be tuned thanks to the color scale on the left hand side menu.

Figure 25: Correlation matrix for the quantitative data

7.1.5 Heatmap

A heatmap as well as the associated dendrogram is depicted on the fifth tab (see Figure 26). The colors represent the intensities: red for high intensities and green for low intensities. White color corresponds to missing values. The dendrogram shows a hierarchical classification of the samples, so as to check that samples are related according to the experimental design. It is possible to tune the clustering algorithm[15] that produces the dendrogram by adjusting the two parameters of the `hclust` *R* function:

- **Distance**: the parameter method of `stats::dist` function (default is *'euclidean'*).
- **Linkage**: the parameter method of `stats::hclust` function (default is *'complete'*).

[15] Computing the heatmap and the dendrogram may be very computationally demanding depending on the dataset.

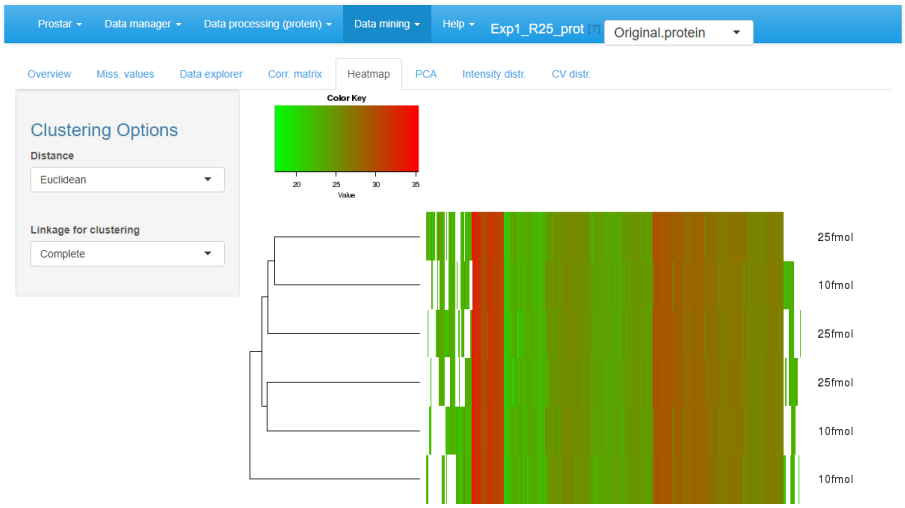


Figure 26: Heatmap and dendrogram for the quantitative data

7.1.6 PCA

A Principal Component Analysis visualization is provided by wrapping the *FactoMineR* package (see Figure 27). To better interpret the PCA displays, the reader is referred to the *FactoMineR* documentation.

7.1.7 Intensity distributions

These plots shows the distribution of the log-intensity of proteins for each condition (see Figure 28).

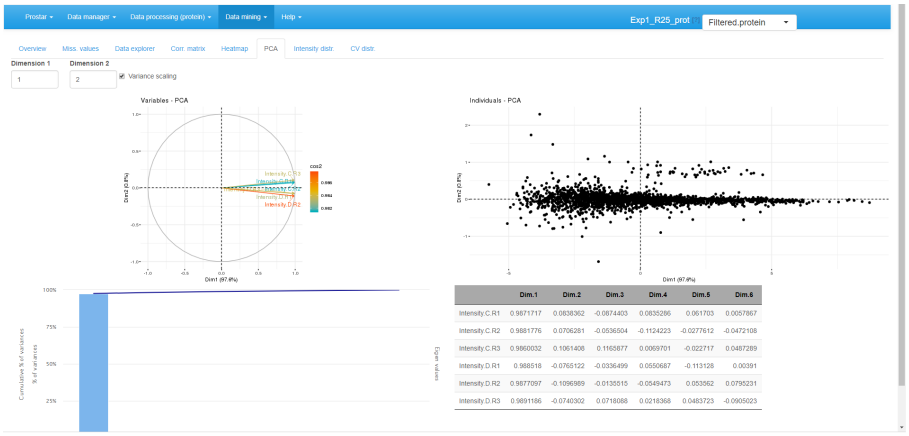


Figure 27: **Principal compenent analysis tab**

The left hand plot represents a smoothed histogram of each sample intensity. The right hand side plot display the same information under the form of a boxplot or a violin plot, depending on the user's choice. In both cases, it is possible to ajust the plot legends, as well as to specify the color code (one color per condition or per sample).

Figure 28: **Densityplot the quantitative data**

7.1.8 CV distribution

Finally, the last tabs display a density plot of the variance (within each condition) conditionally to the log-intensities (see Figure 29). As is, the plot is often difficult to read, as the high variances are concentrated on the lower intensity values. However, it is possible to interactively zoom in on any part of the plot by clicking and dragging (see Figure 30).

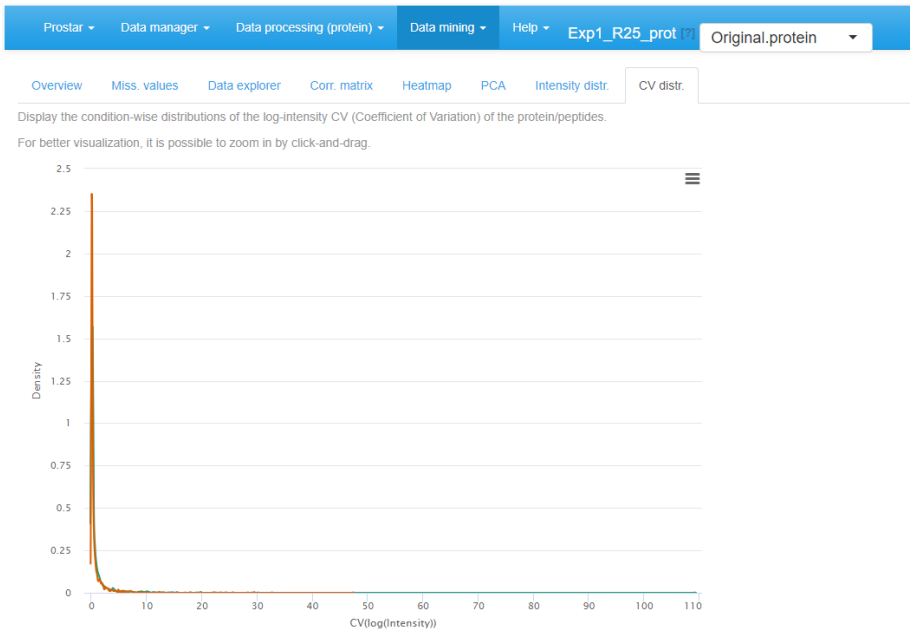


Figure 29: CV distribution for the quantitative data

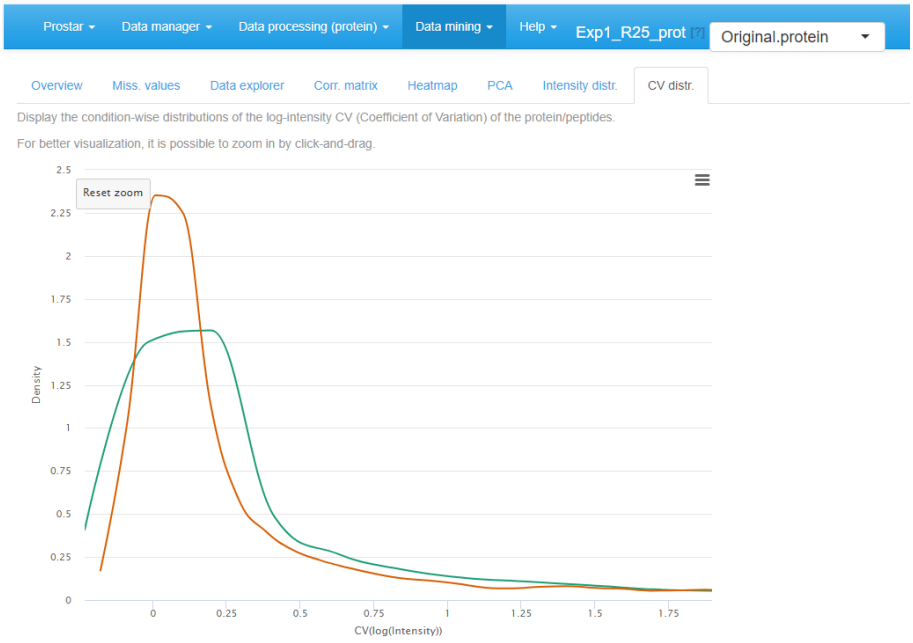


Figure 30: Zoom in of the CV distribution for the quantitative data

7.2 Differential analysis

If one clicks on **Differential analysis** in the **Data mining** menu, it is possible to analyze the protein-level outputs of all statistical tests (see Section 6.5). Such analysis follows 5 steps, each corresponding to a separate tab in the **Differential analysis** menu.

7.2.1 General methodology

On the first tab, select a pairwise comparison of interest[16]. The corresponding volcano plot is displayed. By clicking on a protein in the volcano plot, one or several tables appears below the plot. In any case, a table containing the intensity values of the protein across the samples is displayed, with the same color code as in the Data explorer (for missing values for instance). In addition, if peptide-level information has been uploaded in the *Prostar* session, then, the intensity values of the protein can be linked to the original peptides of the dataset. Thus, the peptides intensities (for both protein-specific and shared peptides) are displayed in the 2 other tables.

[16] In case of a dataset with only 2 different conditions, only one comparison is possible.

Possibly, swap the FC axis with the corresponding tick-box, depending on layout preferences. Possibly, push some p-values to 1, as detailed in Section 7.2.2 (see Figure 31).

Figure 31: **Differential analysis - Tab 1**

Then move on to the next tab, referred to as **p-value calibration** (see Figure 32). Possibly[17], tune the calibration method, as as detailed in Section 7.2.3.

[17] The default tuning of the calibration being the most conservative one, it is possible to skip this step without any risk to spoil the data.

Move on to the next tab and adjust the FDR threshold (see Figure 33), which corresponds to the horizontal dashed line on the volcano plot. The checkbox "Show p-value table" displays a table below the volcano plot, where each line represents a protein along with its p-value and logFC. Moreover, it is indicated by a binary variable if, according to the FDR threshold, the protein is deemed differentially abundant or not. For better visualization, this binary variable also

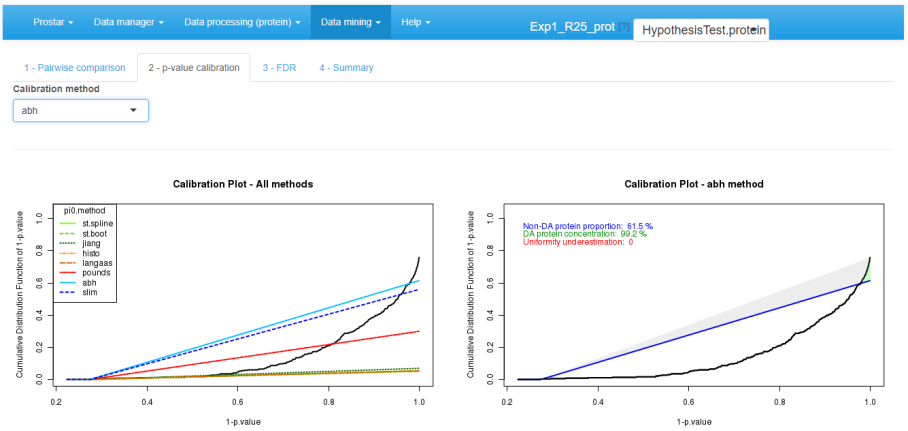


Figure 32: Differential analysis - Tab 2

encodes the color code applied to each line. The "Tooltip" parameter amounts to the list of the avialable meta-data. It is possible to select several of them. The selected one will (1) appear beside the mouse pointer when flying over each protein point in the volcano plot; (2) add the corresponding column in the table below the volcano plot. This table can be easily exported: The user only has to choose whether all the proteins should appear in the exported table, or only the differentially abundant ones. The volcano plot can be saved thanks to the menu available in the upper right corner of the plot.

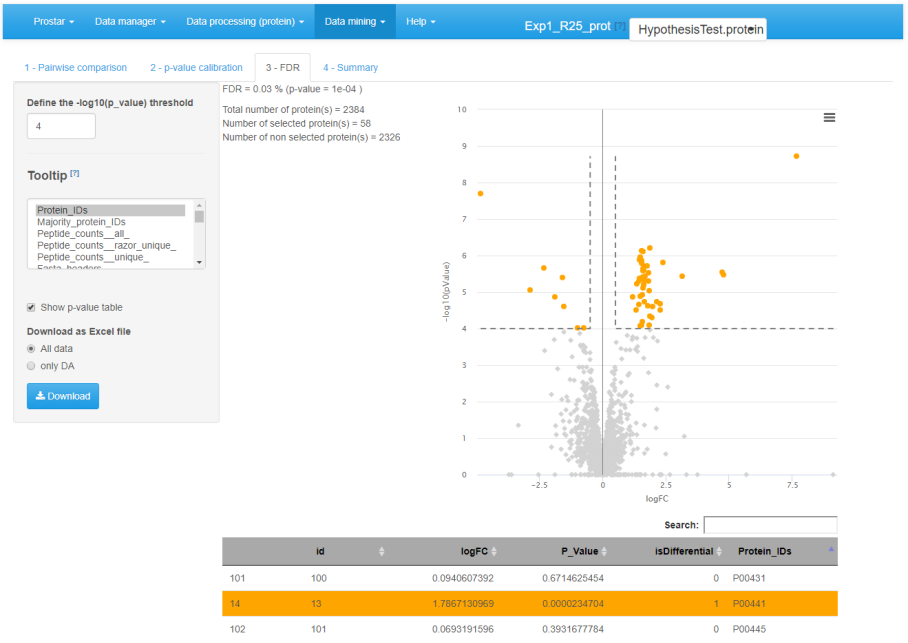
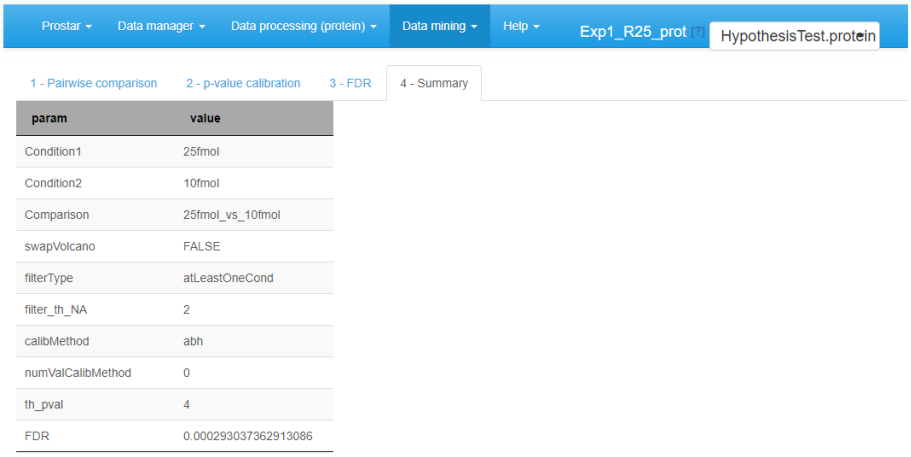


Figure 33: Differential analysis - Tab 3

Move on to the last tab (referred to as **Summary**) to have a comprehensive overview of the differential analysis parameters (see Figure 34). If necessary copy/paste this table for later use.



param	value
Condition1	25fmol
Condition2	10fmol
Comparison	25fmol_vs_10fmol
swapVolcano	FALSE
filterType	atLeastOneCond
filter_th_NA	2
calibMethod	abh
numValCalibMethod	0
th_pval	4
FDR	0.00029303736291308 6

Figure 34: Differential analysis - Tab 4

Possibly, go back to the first tab, so as to select another pairwise comparison and process it. Alternatively, it is possible to continue with the current protein list so as to explore the functional profiles of the proteins found statistically differentially abundant between the compared conditions (see Section 7.3).

7.2.2 Focus on: Push p-values

When working on more than two conditions to compare, the missing value filtering options may be difficult to tune, as a given protein can be well-observed in a group of conditions (with no or few missing values) and largely missing in the remaining conditions. To avoid missing relevant proteins with such a behavior, it is advised to be rather loose at the filtering step (for instance by filtering only proteins that are never seen in the majority of the samples of each condition).

However, with loose filters, proteins with many missing values are kept in the dataset and are then imputed, so as to present a complete intensity distribution for the differential analysis. Therefore, when focusing on a specific comparison among the multiple possible ones, it is thus possible to face proteins which have largely been imputed in the two conditions, so that in practice, one analyses the differential abundance of a protein that is probabibly missing in the two conditions. From an analytical viewpoint, such protein must not appears as differentially abundant, even if its p-value is exceptional, for it is too largely based on imputed values that are not trustworthy.

This is why, the push p-value option is proposed: it makes it possible to force the p-values of these proteins to 1, and prevent them to become false discoveries. Concretely, one simply has to tune parameters of the menu, which are similar to those of the missing value filtering (Section 6.1). However, instead of producing discarding some proteins, it keeps them with a p-value forced to 1 (i.e. a log(p-value) to 0).

As an illustration, Figure 31 displays a case were all the proteins with less than 2 observed (i.e. not imputed) values within each condition have their p-values pushed to 1. These proteins appear as points on the horizontal axis of the graph. Notably, the snapshot was made when the mouse pointer had clicked on one of these proteins which p-value was pushed (the one with an intensity value between 5.5 and 6). Its intensity values across the samples appears in the table below, where one observes that 5 out of 6 values are colored as POV or MEC (see Section 6.3). Clearly, such a protein as very poor mass spectrometry evidence, so that even if the imputed values are of different magnitude (leading to a theoretically appealing p-value), it makes sense to discard it.

7.2.3 Focus on: Calibration plots

The primary goal of a calibration plot is to check that the p-values have a distribution that is compliant with FDR computation theory. Given a supposedly known proportion of non-diffentially abundant proteins (generally noted π_0), a good calibration plot is supposed to depict a curve that follows a straight line of slope π_0 except for a sharp increase on the right hand side, as depicted on Figure 35. The sharp increase, which is the sign of a good discrimination between proteins which are differentially abundant and those which are not, is depicted in green. On the contrary, any bulb of the curve above the π_0 line (in blue), anywhere in the middle of the graph is depicted in red, for it is the sign of an insufficient calibration. On Figure 35, the calibration deviation is almost immaterial and the sharpness of the right hand side increase is real: both can be noticed thanks to colored caption ("Uniformity underestimation" relates to possible red bulps and "DA protein concentration" relates to the sharpness of the green region; the last caption, in blue, being π_0).

However, it is possible to face situations where the right hand side is not sharp enough (see Figure 36), or where the entire distribution is ill-calibrated (see Figure 37). In such a cases, FDR computation would lead to spurious values and to biologically irrelevant conclusions. However, if the calibration is not too bad, it is possible to compensate it, by overestimating π_0. This what the first calibration is made for (see Figure 38): several different estimates are depicted by lines of various slope, so as to help the practitioner to choose the most adapted one to the p-value distribution at hand. In the worst case, it is always

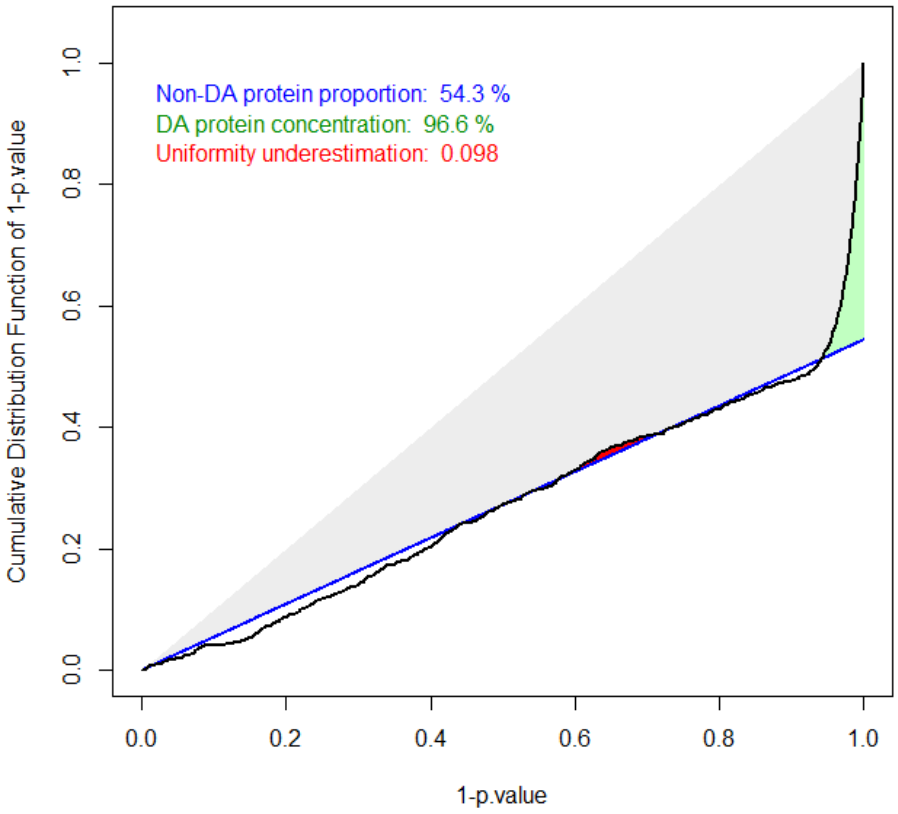


Figure 35: The ideal case of well calibrated p-values

possible to chose the Benjamini-Hochberg options (corresponding to the safer case of $\pi_0 = 1$), which is not represented on the graph for it always corresponds to a diagonal line.

For more details, the interested reader is referred to the *CP4P* package and the companion publication available in the "Useful links" page of *Prostar* (as well as our tutorial on FDR published by JPR in 2017, also referenced on the same page).

7.3 Gene Ontology analysis

The Gene Ontology (GO, http://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).

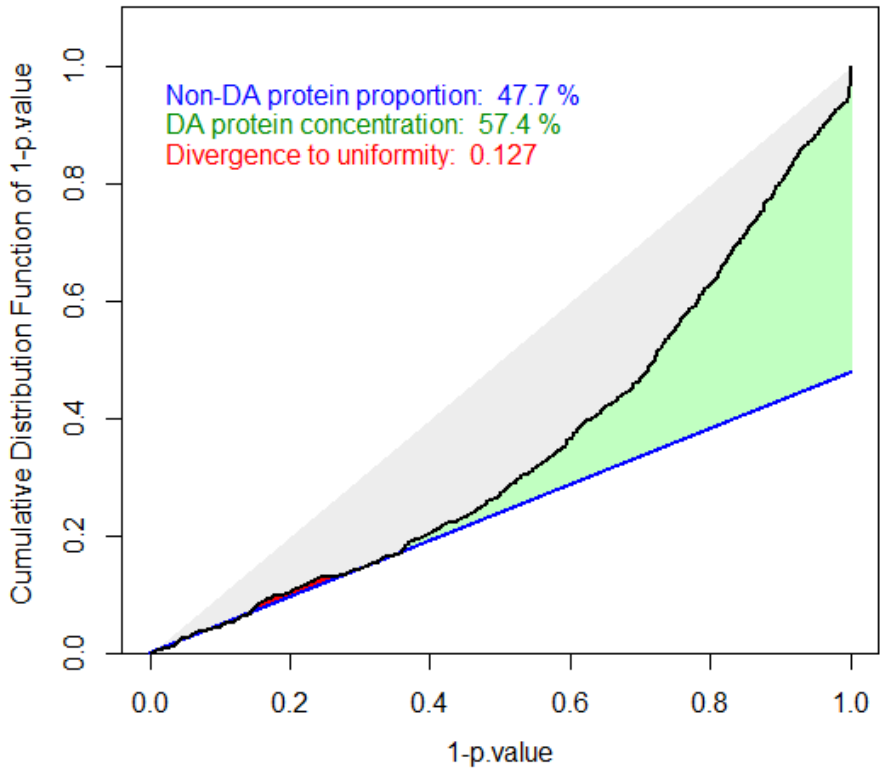


Figure 36: Badly calibrated p-values, case 1

GO analysis is proposed in the **Data mining** menu. It aims to provide the user with a global view of what is impacted (in a biological point of view) in the 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 Figure 39):

- GO Setup
- GO classification
- Go Enrichment
- Parameter summary

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).

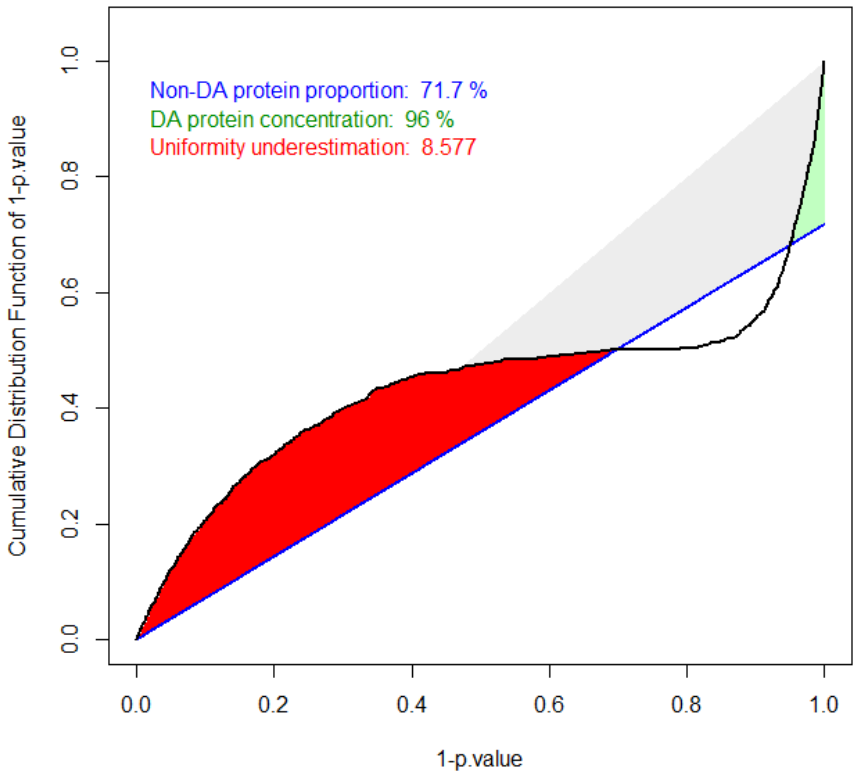


Figure 37: Badly calibrated p-values, case 2

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

In the **GO Classification** tab (see Figure 40), 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).

The **GO Enrichment** tab (see Figure 41) 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 :

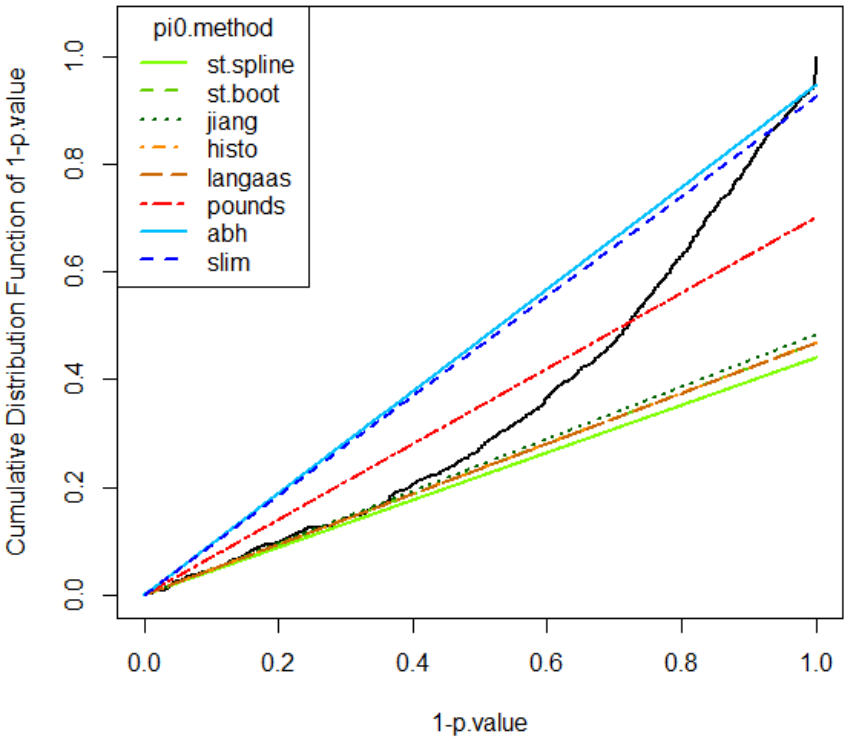


Figure 38: Tuning the π_0 estimate

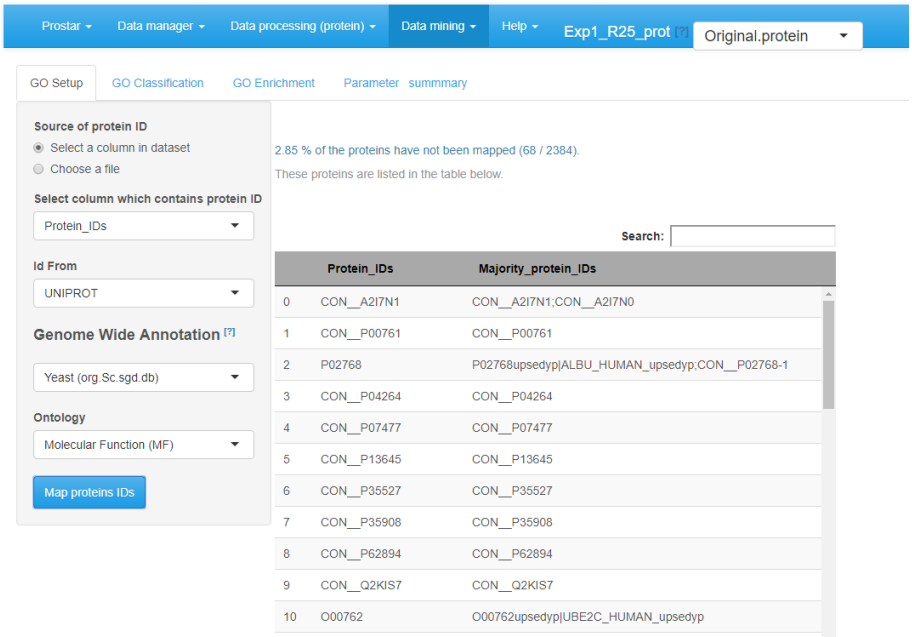


Figure 39: GO Setup tab

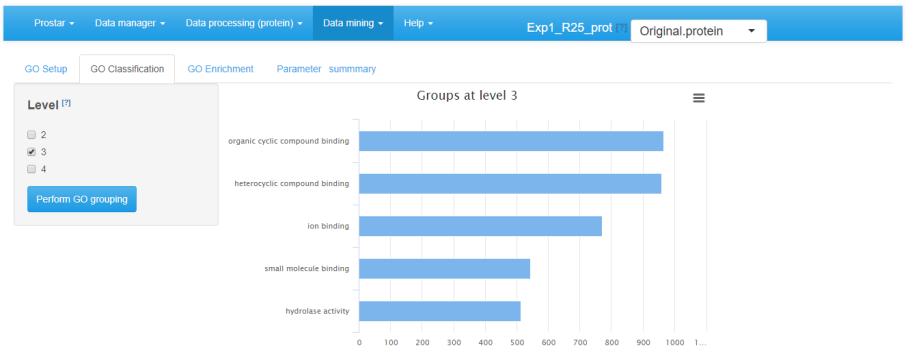


Figure 40: GO classification tab

1. 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
2. the entire dataset loaded in Prostar (e.g. all the proteins IDs of the 'Leading_razor_protein' column of the dataset, as illustrated on Figure 39), or
3. a custom IDs list provided in a separate file by the user (one ID per line).

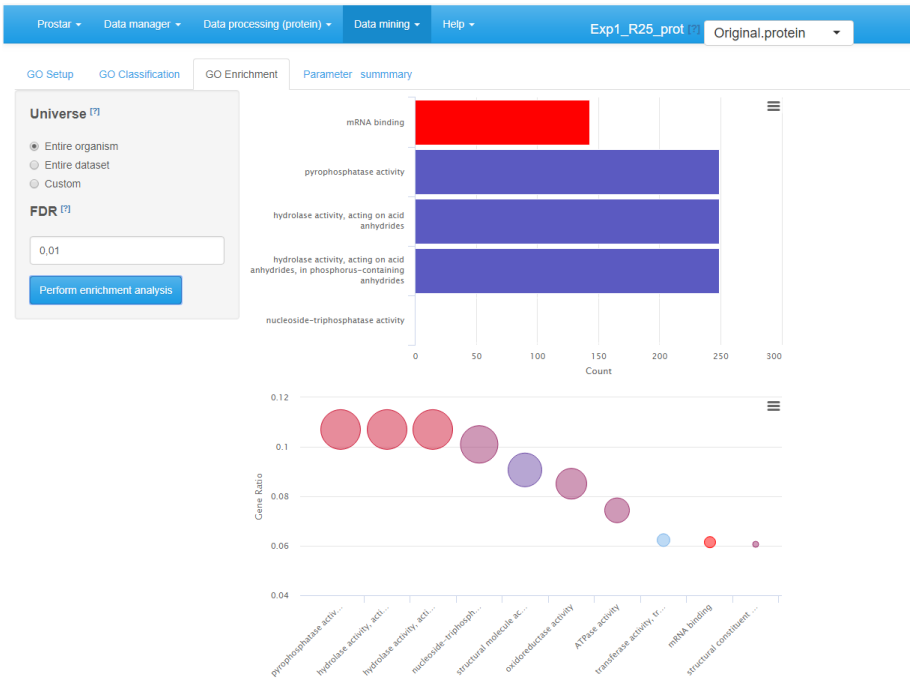


Figure 41: GO enrichment tab

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 Figure 41). 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:

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

The last tab is the **Save GO analysis** one. It allows saving the results: GO classification, GO enrichment, or both (see Figure 42). 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 **Data manager** menu, either in MSnSet or in Excel format.

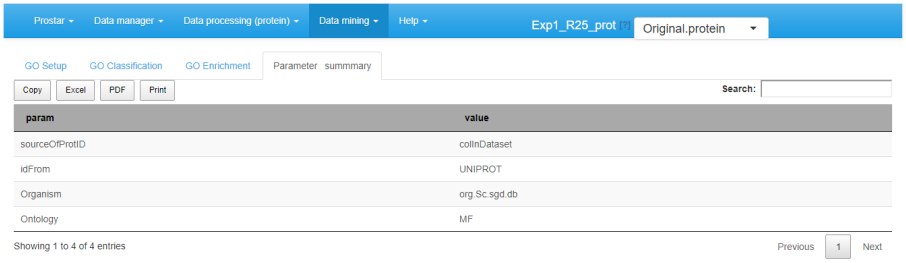


Figure 42: GO analysis summary

8 Session information

- R version 3.5.1 (2018-07-02), `x86_64-w64-mingw32`
- Locale: `LC_COLLATE=French_France.1252`, `LC_CTYPE=French_France.1252`, `LC_MONETARY=French_France.1252`, `LC_NUMERIC=C`, `LC_TIME=French_France.1252`
- Running under: `Windows >= 8 x64 (build 9200)`
- Matrix products: default

- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: Biobase 2.40.0, BiocGenerics 0.26.0, BiocParallel 1.14.2, MSnbase 2.6.4, mzR 2.14.0, Prostar 1.15.0, ProtGenerics 1.12.0, Rcpp 0.12.19
- Loaded via a namespace (and not attached): affy 1.58.0, affyio 1.50.0, AnnotationDbi 1.42.1, assertthat 0.2.0, backports 1.1.2, bindr 0.1.1, bindrcpp 0.2.2, BiocInstaller 1.32.1, BiocManager 1.30.4, bit 1.1-14, bit64 0.9-7, bitops 1.0-6, blob 1.1.1, broom 0.5.0, Cairo 1.5-9, caTools 1.17.1.1, cellranger 1.1.0, cluster 2.0.7-1, clusterProfiler 3.8.1, codetools 0.2-15, colorspace 1.3-2, colourpicker 1.0, compiler 3.5.1, cowplot 0.9.3, cp4p 0.3.5, crayon 1.3.4, curl 3.2, DAPAR 1.15.0, DAPARdata 1.11.2, data.table 1.11.8, DBI 1.0.0, digest 0.6.18, DO.db 2.9, doParallel 1.0.14, DOSE 3.6.1, dplyr 0.7.8, DT 0.5, enrichplot 1.0.2, factoextra 1.0.5, FactoMineR 1.41, farver 1.0, fastmatch 1.1-0, fgsea 1.6.0, flashClust 1.01-2, forcats 0.3.0, foreach 1.4.4, future 1.10.0, gdata 2.18.0, geeM 0.10.1, geepack 1.2-1, ggforce 0.1.3, ggplot2 3.1.0, ggraph 1.0.2, ggrepel 0.8.0, ggridges 0.5.1, globals 0.12.4, glue 1.3.0, gmm 1.6-2, GO.db 3.6.0, GOSemSim 2.6.2, gplots 3.0.1, graph 1.58.2, grid 3.5.1, gridExtra 2.3, gtable 0.2.0, gtools 3.8.1, haven 1.1.2, highcharter 0.6.0, hms 0.4.2, htmltools 0.3.6, htmlwidgets 1.3, httpuv 1.4.5, httr 1.3.1, igraph 1.2.2, imp4p 0.7, impute 1.54.0, IRanges 2.14.12, Iso 0.0-17, iterators 1.0.10, jsonlite 1.5, KernSmooth 2.23-15, knitr 1.20, later 0.7.5, lattice 0.20-38, lazyeval 0.2.1, leaps 3.0, limma 3.36.5, listenv 0.7.0, lubridate 1.7.4, magrittr 1.5, MALDIquant 1.18, MASS 7.3-51.1, Matrix 1.2-15, memoise 1.1.0, MESS 0.5.2, mime 0.6, miniUI 0.1.1.1, modelr 0.1.2, multtest 2.36.0, munsell 0.5.0, mvtnorm 1.0-8, mzID 1.18.0, nlme 3.1-137, norm 1.0-9.5, openxlsx 4.1.0, pcaMethods 1.72.0, pillar 1.3.0, pkgconfig 2.0.2, plyr 1.8.4, png 0.1-7, preprocessCore 1.42.0, promises 1.0.1, purrr 0.2.5, quantmod 0.4-13, qvalue 2.12.0, R.methodsS3 1.7.1, R.oo 1.22.0, R.utils 2.7.0, R6 2.3.0, rclipboard 0.1, RColorBrewer 1.1-2, readr 1.1.1, readxl 1.1.0, reshape2 1.4.3, rhandsontable 0.3.6, rlang 0.3.0.1, rlist 0.4.6.1, RSQLite 2.1.1, rstudioapi 0.8, rvcheck 0.1.1, rvest 0.3.2, S4Vectors 0.18.3, sandwich 2.5-0, scales 1.0.0, scatterplot3d 0.3-41, shiny 1.2.0, shinyAce 0.3.2, shinyBS 0.61, shinycssloaders 0.2.0, shinyjqui 0.3.2, shinyjs 1.0, shinythemes 1.1.2, shinyTree 0.2.6, shinyWidgets 0.4.4, siggenes 1.54.0, splines 3.5.1, stats4 3.5.1, stringi 1.2.4, stringr 1.3.1, survival 2.43-1, tibble 1.4.2, tidyr 0.8.2, tidyselect 0.2.5, tidyverse 1.2.1, tmvtnorm 1.4-10, tools 3.5.1, truncnorm 1.0-8, TTR 0.23-4, tweenr 1.0.0, units 0.6-1, UpSetR 1.3.3,

vioplot 0.2, viridis 0.5.1, viridisLite 0.3.0, vsn 3.48.1, webshot 0.5.1, whisker 0.3-2, XML 3.98-1.16, xml2 1.2.0, xtable 1.8-3, xts 0.11-2, yaml 2.2.0, zip 1.0.0, zlibbioc 1.26.0, zoo 1.8-4