A web application for the quantitative analysis of proteomic data with MSstats

Cristina Elisabetta Pasi
Máster universitario en Bioinformática y bioestadística UOC-UB
TFM-Estadística y Bioinformática

Consultor: Eduard Sabidó Aguadé
Responsable de la asignatura: María Jesús Marco Galindo

6th June 2017

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Una aplicación web para el análisis cuantitativo de datos proteómicos con MSstats
Nombre del autor:	Cristina Elisabetta Pasi
Nombre del consultor/a:	Eduard Sabidó Aguadé
Nombre del PRA:	María Jesús Marco Galindo
Fecha de entrega (mm/aaaa):	05/2017
Titulación::	Máster universitario en Bioinformática y bioestadística UOC-UB
Área del Trabajo Final:	Estadística y Bioinformática
Idioma del trabajo:	Inglés
Palabras clave	Shiny, Proteómica, Espectrometría de Masas
Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*	

El desarrollo de técnicas de alto rendimiento ha generado grandes conjuntos de datos que requieren poderosas herramientas estadísticas para extraer significado biológico de enormes cantidades de información. La espectrometría de masas es una técnica analítica que permite la caracterización del proteoma de diferentes muestras biológicas. Los investigadores han utilizado ampliamente el software estadístico R para el análisis de datos biológicos, y varios paquetes están disponibles para el análisis de la proteómica basada en espectrometría. Uno de estos paquetes es MSstats, una colección de código abierto de funciones estadísticas para cuantificar proteínas basadas en modelos lineales mixtos. Para habilitar el uso de MSstats por investigadores con formación no estadística, MSstats ha sido incluido en la herramienta de interfaz gráfica Skyline, una aplicación cliente de Windows para el análisis de datos de espectrometría de masas. Con este proyecto he desarrollado una interfaz gráfica de usuario basada en web para MSstats usando Shiny, un framework de aplicación web para R. Esto ampliará el uso de esta aplicación a usuarios que no son de Windows sin el conocimiento de R y lo facilitará desde un punto de vista computacional .

La aplicación web Shiny-MSstats es capaz de aceptar datos brutos del análisis de espectrometría de masas, realiza un control de calidad personalizado y análisis estadístico, lo que resulta en la cuantificación de proteínas o la medida de la diferencia en la expresión. El análisis se integra luego con herramientas para estudios funcionales

Abstract (in English, 250 words or less):

The development of high throughput techniques has generated large sets of data that require powerful statistical tools for extracting biological significance from huge amounts of information. Mass spectrometry is an analytical technique that allows for the characterisation of the proteome of different biological samples. Researchers have widely used the statistical software R for the analysis of biological data, and several packages are available for the analysis of spectrometry based proteomics. One of these packages is MSstats, an open source collection of statistical functions to quantify proteins based on linear mixed models. To enable the use of MSstats by researchers with non-statistical backgrounds, MSstats has been included in the graphical user interface tool Skyline, a Windows client application for the analysis of mass spectrometry data. With this project I have developed a web based graphical user interface for MSstats using Shiny, a web application framework to R. This will extend the use of this application to non-Windows users without R knowledge and facilitate it use from a computational point of view.
The Shiny-MSstats web application is able to accept raw data from mass spectrometry analysis, performs customised quality control and statistical analysis, resulting in quantification of proteins or measure of difference in expression. The analysis is then integrated with tools for functional studies.

Index

List of figures

1. Introduction

1.1 Background and rationale

Quantitative proteomics has emerged in the last 15 years as an extremely powerful tool for measuring biological variations in terms of abundance, turnover, post-translational modification, localisation and bimolecular interactions of an organism's proteins, or proteome.
Mass spectrometry-based assays are currently widely used for quantitative proteomics as they are highly reproducible, precise, and they allow for the large-scale analysis of hundreds of peptide transitions per assay. Clearly, these assays generate vast datasets which need to be addressed computationally to process, manage, analyse and interpret the data. Ultimately, the analysis of mass spectrometry-generated data is aimed at defining a model for a biological process through different steps of data handling, statistical modelling and interpretation. However, the road from mass spectrometry to data interpretation is complex and therefore the algorithms and tools of computational proteomics are very sophisticated and often not within everybody's expertise [1].

MSstats is an open source package of the statistical programming language R which was developed in the laboratory of Olga Vitek in the University of Purdue and the Northeastern University only a few years ago; it comprises a series of algorithms and functions which can be used for the quantification of proteins and peptides in mass spectrometry assays [2].
Because of its completeness and versatility in accepting inputs from different types of experiments, it is now an established tool for proteomic quantification and it represents an invaluable substrate for the development of user-accessible interfaces for mass spectrometry analysis.
To this end, researchers have developed the application Skyline as interface for the use of the R package MSstats, however the use of

this application is restricted to the Windows operating system and needs to be installed locally [3].
The aim of this project is to develop a web-based application for the use of MSstats, where the users can upload their data and perform statistical analysis with no prior knowledge of the R programming language. This application will render this tool globally available for proteomic analysis, filling the gap between biological researchers and statisticians. The application has been developed using the R framework Shiny.

1.2 Objectives

This project has the following objectives:

- Become familiar with the R package MSstats, the technical and statistical requirements of proteomic analysis and the type of questions that may be answered with this technique

- Explore the R framework Shiny, its potential and the main commands and structure of a Shiny web application

- Implement a web interface that uses MSstats for proteomic analysis. The web interface will be able to:
    - accept proteomic data (in a form that is suitable for the R package MSstats) and transform it accordingly
    - perform quality control analysis with parameters set by the user and output a transformed dataset and quality control plots
    - perform group comparisons based on the user's needs (according to the experimental design) and output significant variations in the expression of proteins
    - calculate the relationship between sample size and power for the particular experiment in order to help with designing future experiments
    - complement the MSstats package with functional analysis of the proteins that have been isolated by the group comparisons

- Optimise the user interface to ensure full functionality and an intuitive user experience

- Provide the user with a user manual to understand the basics of the statistical analysis as well as helping to choose the best parameters for the experimental design and requirements

- Write a report on the work and prepare a short presentation to explain the project

1.3 Focus and methods

To render the quantitative analysis of proteomic data available to everyone and easy to use for researchers without a statistical background, a new web application was developed implementing the MSstats R package. The application will be available online, without the need to install any software locally. This will facilitate its access universally and free the user from particular needs in terms of hardware and software.

The statistical algorithms for the analysis are those of the open source R package MSstats.
MSstats is a very useful R package for the quantitative analysis of proteomic data. It has been widely used in the literature to interpret mass spectrometry data and in different settings and to answer different biological question, from isolating proteins involved in cancer processes [4,5], to resolving protein networks in living cells [6,7], to many more applications in different organisms and different biological processes and diseases.
The simplicity of its functions, together with the possibility of changing all parameters of the analysis, as well as the fact that it is applicable to a virtually all scenarios of sample preparation and experimental design as well as types of mass spectrometry acquisition, make MSstats a great

substrate for the implementation of a graphical user interface which can be accessible by any user, even with no prior experience with its use [8]. By using this package as basis for the application, I was able to focus on the usability and graphical interface rather than the statistical process itself.

The R statistical algorithms are implemented onto an interactive web application with the framework Shiny. The R framework Shiny for building interactive web applications was developed by the R studio team in order to allow R programmers to implement R based softwares as web applications without any prior knowledge of html and css [9]. It is used for biological sciences as well as other fields where R is applicable to statistical problems. Its structure and input-output flow of data is perfect to implement the analysis of data through a graphical interface rather than a console, with no need for knowledge of the R language. Shiny is open source and extensively documented online, where tutorials, articles and examples of existing applications are easily available.

The web application will be deployed using a server provided by Shiny, shinyapps.io, designed to host Shiny apps. This server on the cloud is easy to use and scalable, to accommodate a large number of users.

1.4 Project plan

The web application is coded in R using R studio and the packages `shiny`, `shinyBS`, `shinyjs`, `MSstats`, and the packages for functional annotation `biomaRt` and `Biobase`.

The tasks of the project are the following (milestones in bold):

1. Define the project

a) Examine the project definition, its guidelines and deadlines: the documentation provided by the University describes the tasks, the evaluation methods and the phases and deadlines of the project
b) Collect and examine the necessary documentation: web search of the current tools available for proteomic analysis

2. Write project plan (PEC1)
   a) Write the index of the final thesis
   b) Define objectives and sub-objectives
   c) Define a timeline including the PECs: the timeline takes into account the workload of the objectives and personal time availability including holidays
   d) Write a draft of the plan: index, introduction, rationale, objectives and sub-objectives, timeline, possible problems and solutions, bibliography
   e) **Submit Project Plan (15/3/17)**

3. Project elaboration I (PEC2)
   a) Explore and understand the Shiny tool for web applications in R: follow Shiny tutorial to get accustomed to the framework
   b) Explore the package MSstats and its functions: view Bioconductor documentation and examples of its application
   c) Explore other useful R packages to integrate the application for annotation and functional analysis
   d) Define the pipeline of the analysis: quality control, data visualisation, statistical analysis, future experimental design, functional analysis (22/3/17)
   e) Define the parameters which can be regulated by the user: to be implemented in Shiny application
   f) Code necessary functions in R and write guidelines and documentation
   g) Implement drafting of summary through R markdown
   h) Submit PEC2 and updated project plan (date: 12/04/17)

4. Project elaboration II (PEC3)
   a) Implement the application in Shiny: create locally the Shiny application

b) Implement parameters to be regulated by the user: user interface
c) Choose server (and setup): must be able to accommodate large amounts of data
d) Verify the functionality of the application: try with available datasets and review and refactor code for responsiveness and usability
e) Submit PEC3 (date: 10/05/17)
f) deploy

5. Thesis and presentation
   a) Define future work
   b) Unify chapters: as per index
   c) Proofread
   d) Create presentation with Powerpoint (23/5/17)
   e) Submit Thesis (24/5/17)

The following report shows the status of the tasks, their schedule and progress (fig 1-3):

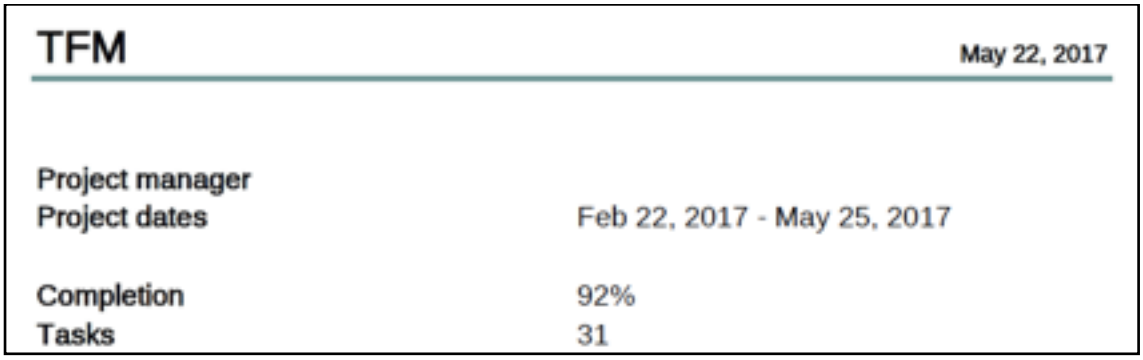

Fig.1 Overall progress of the project

Gantt Chart

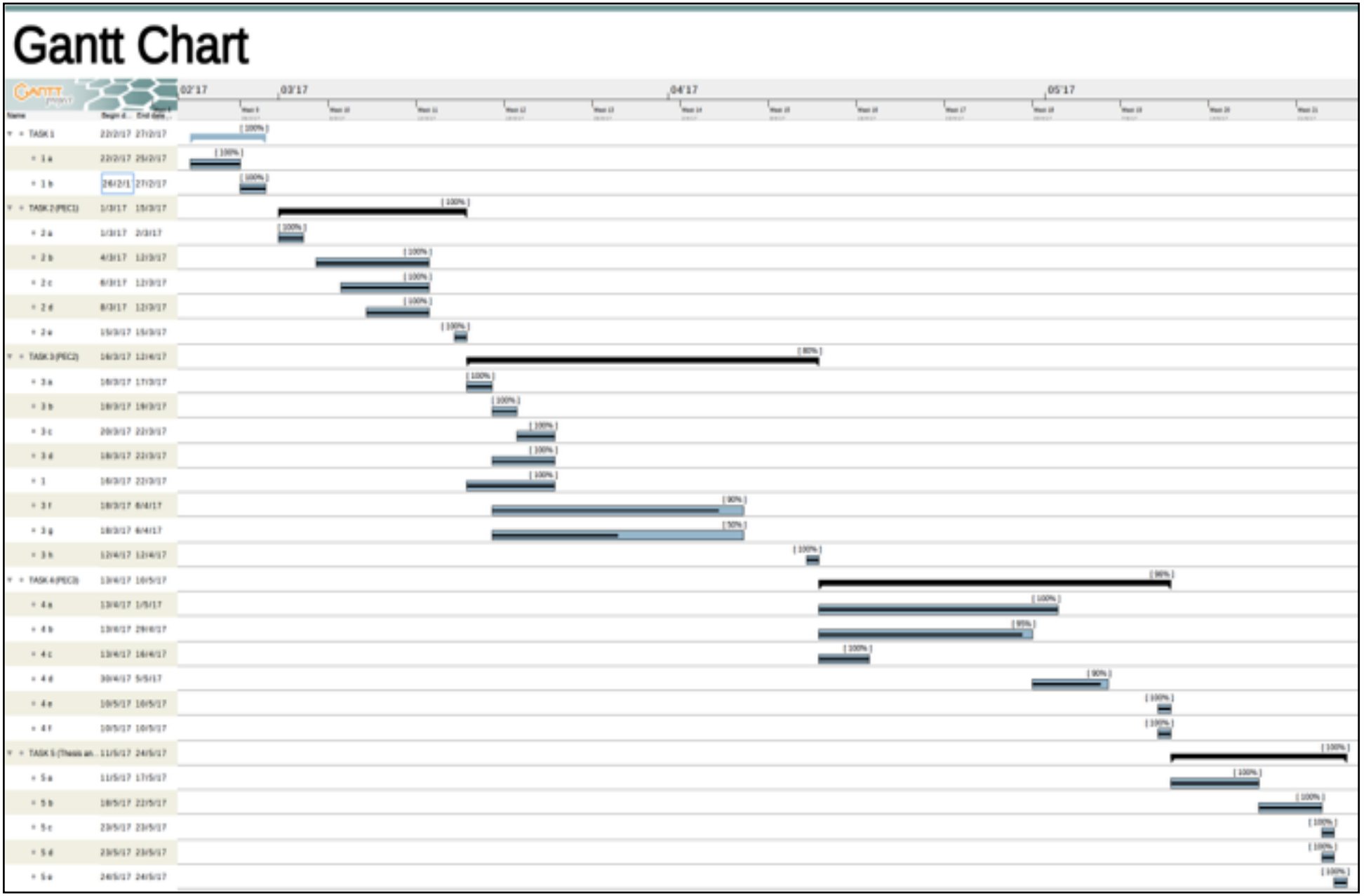
02'17
03'17
04'17
05'17
Name
Begin d...
End date
TASK 1
22/2/17 27/2/17
1 a
22/2/17 25/2/17
1 b
26/2/1
27/2/17
TASK 2 (PEC1)
1/3/17 15/3/17
2 a
1/3/17 2/3/17
2 b
4/3/17 12/3/17
2 c
6/3/17 12/3/17
2 d
8/3/17 12/3/17
2 e
15/3/17 15/3/17
TASK 3 (PEC2)
16/3/17 12/4/17
3 a
16/3/17 17/3/17
3 b
18/3/17 19/3/17
3 c
20/3/17 22/3/17
3 d
18/3/17 22/3/17
3 f
18/3/17 6/4/17
3 g
18/3/17 6/4/17
3 h
12/4/17 12/4/17
TASK 4 (PEC3)
13/4/17 16/5/17
4 a
13/4/17 1/5/17
4 b
13/4/17 29/4/17
4 c
13/4/17 18/4/17
4 d
30/4/17 5/5/17
4 e
10/5/17 10/5/17
4 f
10/5/17 10/5/17
TASK 5 (Thesis an...
11/5/17 24/5/17
5 a
11/5/17 17/5/17
5 b
18/5/17 22/5/17
5 c
23/5/17 23/5/17
5 d
23/5/17 23/5/17
5 e
24/5/17 24/5/17

Fig.2 Percentage of completion of the tasks

TFM May 22, 2017

Tasks 2

Name	Begin date	End date
TASK 1	22/2/17	27/2/17
1 a	22/2/17	25/2/17
1 b	26/2/17	27/2/17
TASK 2 (PEC1)	1/3/17	15/3/17
2 a	1/3/17	2/3/17
2 b	4/3/17	12/3/17
2 c	6/3/17	12/3/17
2 d	8/3/17	12/3/17
2 e	15/3/17	15/3/17
TASK 3 (PEC2)	16/3/17	12/4/17
3 a	16/3/17	17/3/17
3 b	18/3/17	19/3/17
3 c	20/3/17	22/3/17
3 d	18/3/17	22/3/17
1	16/3/17	22/3/17
3 f	18/3/17	6/4/17
3 g	18/3/17	6/4/17
3 h	12/4/17	12/4/17
TASK 4 (PEC3)	13/4/17	10/5/17
4 a	13/4/17	1/5/17
4 b	13/4/17	29/4/17
4 c	13/4/17	16/4/17
4 d	30/4/17	5/5/17
4 e	10/5/17	10/5/17
4 f	10/5/17	10/5/17
TASK 5 (Thesis and presentation)	11/5/17	24/5/17
5 a	11/5/17	17/5/17
5 b	18/5/17	22/5/17
5 c	23/5/17	23/5/17
5 d	23/5/17	23/5/17
5 e	24/5/17	24/5/17

Fig.3 List of tasks and schedule

1.5 Summary of obtained products

The result of this work is a web application developed in the Shiny framework which is a graphical interface for the use of the R package MSstats. The application is interactive and is able to take as input a data frame in any format accepted by MSstats. Virtually all parameters in the statistical analysis can be modified by the user with the exception of the graphical parameters of the plots.

The application is structured in tabs as follows:

- Load data: where the .csv file will be uploaded with the possibility to edit different accepted formats. A first high-level summary of the data will be provided at this stage.
- Quality Control: where the data is pre-processed and visualised as the parameters are being chosen (transformation, normalisation, imputation for missing values, and summarisation).
- Statistical model: where a model is built to compare protein abundance and the results are visualised.
- Functional analysis: where protein with different expression levels are chosen and examined through query to Ensembl
- Experiment design: where the requirements for future experiments to achieve a certain power, fold change and false discovery rate is calculated

Although the application already has a good functionality, some work still needs to be done to improve the user experience in the application and to render it more usable and intuitive.

Along with the application there is a user manual (see Appendix), where the user can find a brief description of the statistics behind the application and find references to the bibliography or documentation to the R packages in use, and a presentation where the application is described.

1.6 Short description of the other chapters

The remaining chapters of this report will include:

- A brief description of bottom-up proteomic techniques to introduce the types of experiment and data that the application will receive; a short introduction on different acquisition methods and spectra interpretation softwares will be provided (chapter 2.1)

- A brief description of the statistical background of the MSstats package (chapter 2.2)

- An introduction to Ensembl and Biomart, for functional analysis (chapter 2.3)

- The structure of the application and its workflow, focussing on the features implemented in the Shiny framework (chapter 2.4)

- The server for deploy (chapter 2.5)

- The tools and softwares used for the work (chapter 2.6)

2. MSstats web app

2.1 Mass spectrometry analysis

Virtually all biological processes are governed and catalysed by proteins. The study of the proteome is key to understanding the phenotype of a cell and its perturbations. Because the proteome is a highly complex and dynamic entity and and proteins cannot be amplified like oligonucleotides, proteomic analyses rely on sensitivity and scalable technologies. Mass spectrometry is the golden standard in proteomic analysis, as it is versatile and allows for large scale analyses [1].
Depending on the focus of the investigation - be it characterisation of known proteins, quantification of protein variation or identification of unknown compounds - the sample preparation and mass spectrometry approach will be different.

The main issues that quantitative proteomics assays face are: biological variation between samples, technical variation due to the processing of the sample and data acquisition, and the interpretation of the data.
The researcher is required to provide adequate biological replicates to limit the effect of biological variation. To overcome technical variation, the preparation of the sample may be performed by labelling the samples according to their experimental condition. There are several techniques used to label the samples metabolically, such as Stable Isotope Labelling by Aminoacids in Culture (SILAC), or chemically such as labelling of peptides with stable isotopes, incorporation of radio labeled aminoacids, or the addition of specific tags to the peptides such as isobaric Tags for Relative and Absolute Quantification (iTRAQ) [10].

MSstats is able to accept data from label-free samples but also from samples which use labeled reference proteins and peptides with stable isotopes. It is not yet able to accept data where multiple

samples with different labelling are read in the same run (as with SILAC or iTRAQ).

Bottom-up proteomics is a technique whereby the proteins are enzymatically digested and then separated through liquid chromatography (LC). This is then coupled with mass spectrometry analysis where the masses of the proteolytic peptides are acquired and then compared to their predicted masses in a sequence database for protein identification.
Three main approaches are used in mass spectrometry proteomics: Data Dependent Acquisition (DDA), which has been often used for the study of the complete proteome, Data Independent Acquisition (DIA), which was developed to overcome the lack of reproducibility of DDA, and targeted Selected Reaction Monitoring (SRM) where the acquisition is performed only on a subset of known targets. The Shiny MSstats web application is able to perform analysis on all three types of acquisitions (fig.4).

DDA has been widely used in the past for the identification of novel compounds. In DDA, peptides that enter the mass spectrometer are quickly scanned and chosen based on their abundance; the most abundant peptides are fragmented and identified through the Tandem Mass Spectrometry (MS/MS) data. Because this method performs a first scan and then a random selection of peptides based on abundance, this approach does not ensure reproducibility and misses poorly represented peptides. Also, due to the broadness of the analysis, the absolute quantification of the peptide is impossible with DDA, although relative quantification between controlled samples is still possible [1,11].

In contrast, DIA allows for a more reliable quantification of the peptide species in a sample by repeatedly analysing randomly taken samples within a specified mass range. Because of its advantages compared to DDA based methods, DIA are being used widely in label-free quantitative assays.
Although the interpretation of DIA analysis may be more complex due to the large number of identified peptides, the improvement in computational tools has enabled the correct handling of DIA generated data [12].

One particular type of DIA acquisition that overcomes these difficulties in data interpretation is the SWATH Acquisition, in which the instrument performs several cycles of acquisition focussing on a narrow window of mass. This allows the data generated by mass spectrometry to be comprehensive over the entire sample, but reduces the complexity of the interpretation by analysis one mass range at the time [13].

In SRM a particular protein population is isolated and then fragmented and the resulting fragment's abundances are used to infer the abundance of the chosen protein. This technique has been used for the absolute quantification of proteins or relative to pathological conditions, and also to quantify peptides which would otherwise fall under the detection limit of other acquisition techniques [14].

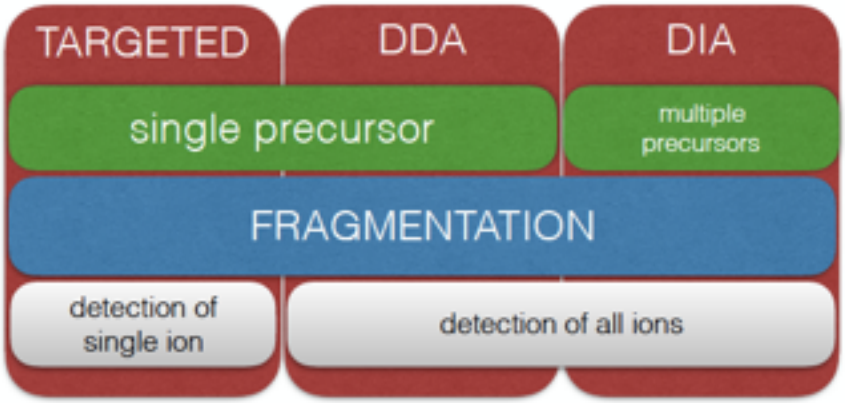

Fig.4 Schematic representation of mass spec acquisition techniques

Subsequent to the data acquisition is the interpretation of the spectra collected by the mass spectrometer and finally the statistical analysis of the data. Different softwares exist for the interpretation of mass spectrometry generated spectra, and they are specific to the type of experiment and acquisition.

The statistical R package MSstats is able to perform statistical analysis on data that has been elaborated by the following softwares:

- Skyline: an open source Windows client application developed in the University of Washington for the interpretation of targeted acquisition (SRM), DIA or DDA [15]. The MSstats package is already available as external tool for this software

- MaxQuant: an open source software written in C# at the Max Plank Institute of Biochemistry to analyse labelled or label free data acquisitions [16]

- ProgenesisQI: a software developed by Nonlinear Dynamics for the quantification and identification of proteins from DDA and DIA [17]

- Proteome Discoverer: developed by Thermo Fisher to analyse data from different workflows [18]

- OpenSWATH: developed at the ETH in Zurich for the data extraction in DIA experiments [19]

- Spectronaut: a software developed by Biognosys for the analysis of DIA data [20]

Previous to the analysis, the data coming from the softwares mentioned above is transformed to a data frame which can be read by the MSstats package.

2.2 MSstats

The package MSstats takes as input the elaborated data resulting from identification and quantification of the peaks in the mass spectrometer generated spectra. It then allows for data processing and visualisation, statistical modelling and experimental design [2]. The Shiny MSstats application follows in its structure the pipeline of

the MSstats package with the addition of a functional annotation tool.

The first part of the analysis allows to transform and normalise the peak intensities and to visualise the data through workflow-specific summaries and plots for quality control. In this section the missing values are also handled.
The quality control step is aimed at identifying possible systematic errors between runs of the mass spectrometer and to visualise the effect of the normalisation process.

In the second part the experimental design is automatically interpreted by the algorithms and a linear-mixed model is fitted to reflect the design, the acquisition method and the focus of the contrast. This is done with he R functions lm() and lmer(). Based on the model, differentially abundant proteins can be identified between samples of the absolute abundance of a single protein can be summarised per replicate or condition. These significant differentially abundant proteins can be visualised with volcano plots, heat maps or comparison plots.
The fitted linear model relies on two assumptions: that the measurement errors have a Normal distribution, and that their variance is constant. These assumptions can be verified by visualising the QQPlots and the Residual Plots for the model.

The third part uses the dataset in study as a pilot for future experiments and calculates the parameters for the best experimental design in order to achieve a particular statistical power or predicts the achieved statistical power with a given number of replicates.

For this work I have used the latest version of MSstats (v 3.8.2)

2.3 Ensembl and BioMart

For the functional analysis section of this application I have used the R package biomaRt, the R interface of the BioMart data mining

tool for Ensembl. Ensembl is an enormous repository which stores a great amount of data on gene annotations, sequences, prediction of function and disease data [21]. It hosts information for a large number of species and is therefore one of the most comprehensive gene-protein repositories available. BioMart is an interface to easily query Ensembl and retrieve the information on any aspect of a particular genome [22].

2.4 Shiny architecture

Because of the stepwise nature of the analysis, the Shiny application is structured as a NavbarPage, with eight tabs corresponding to different steps of the analysis, a homepage and a help page (fig.5):

- Homepage
- Load data
- Quality control
- Statistical model
- Functional analysis
- Future experiments
- Help

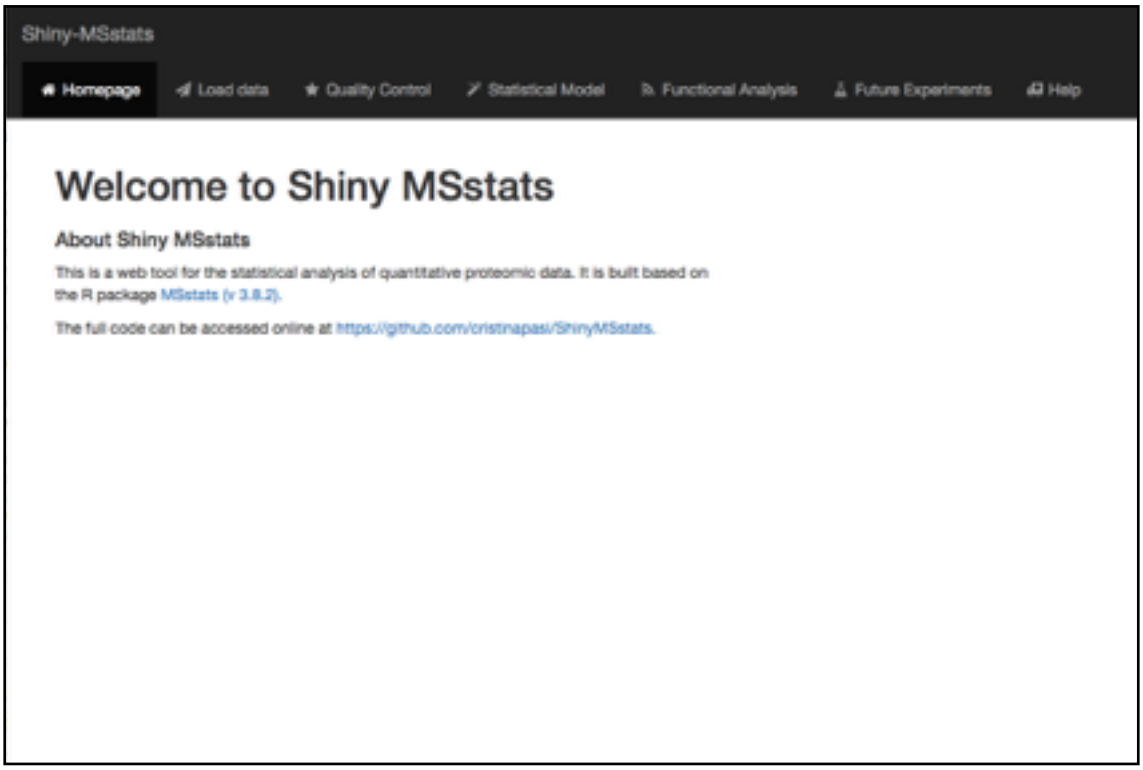

Fig.5 Homepage

The analysis is progressive, meaning that subsequent tabs require the previous tabs to be completed, as they take as input the output of the previous tabs.

As this will be an open source software and the code is quite consistent, I have tried to keep the R scripts as clear as possible; the simplest way has been to divide the code into two main scripts (ui.R and server.R) which then source an external script for each tab (also subdivided into ui and server). For example, the script ui.R sources the file xxx-ui.R for tab xxx and the script server.R sources the file xxx-server.R for the server side.

```r

library(shiny)


source("panels/home-ui.R", local = T)
source("panels/loadpage-ui.R", local = T)
source("panels/qc-ui.R", local = T)
source("panels/statmodel-ui.R", local = T)
source("panels/expdes-ui.R", local = T)
source("panels/analysis-ui.R", local = T)
source("panels/help-ui.R", local = T)

########################################################################

ui <- navbarPage(
  title = "Shiny-MSstats",
  tabPanel("Homepage", icon = icon("home"), home),
  tabPanel("Load data", icon = icon("send"), loadpage),
  tabPanel("Quality Control", icon = icon("star"), qc),
  tabPanel("Statistical Model", icon = icon("magic"), statmodel),
  tabPanel("Functional Analysis", icon = icon("feed"), analysis),
  tabPanel("Future Experiments", icon = icon("flask"), expdes),
  tabPanel("Help", icon = icon("ambulance"), help),
  inverse = T,
  collapsible = T,
  windowTitle = "Shiny-MSstats"
)


shinyUI(ui)
```

Fig.6 ui.R layout

```r

library(shiny)
library(MSstats)
library(shinyBS)
library(uuid)
library(shinyjs)
library(biomaRt)
library(Biobase)


shinyServer(function(input, output, session) {
  # load data
  source("panels/loadpage-server.R", local = T)
  # quality control
  source("panels/qc-server.R", local = T)
  # statistical model
  source("panels/statmodel-server.R", local = T)
  # functional analysis
  source("panels/analysis-server.R", local = T)
  # future experiment
  source("panels/expdes-server.R", local = T)

})
```

Fig.7 server.R layout

• Load data tab

The load data tab is organised in a sidebar panel where on the left side the user can upload the data as .csv file, specify if the file has a header and what kind of separator it uses, the type of acquisition and whether the file is the classic 10-column file or if it comes from previous analysis softwares which are supported by MSstats (as described in chapter 2.1); here the user can also choose to explore the application using a sample dataset, taken from the article "Reproducible and Consistent Quantification of the Saccharomyces cerevisiae Proteome by SWATH-mass spectrometry" [23].
Some input formats require accompanying annotation files that can be uploaded at the bottom of the sidebar (fig.8).

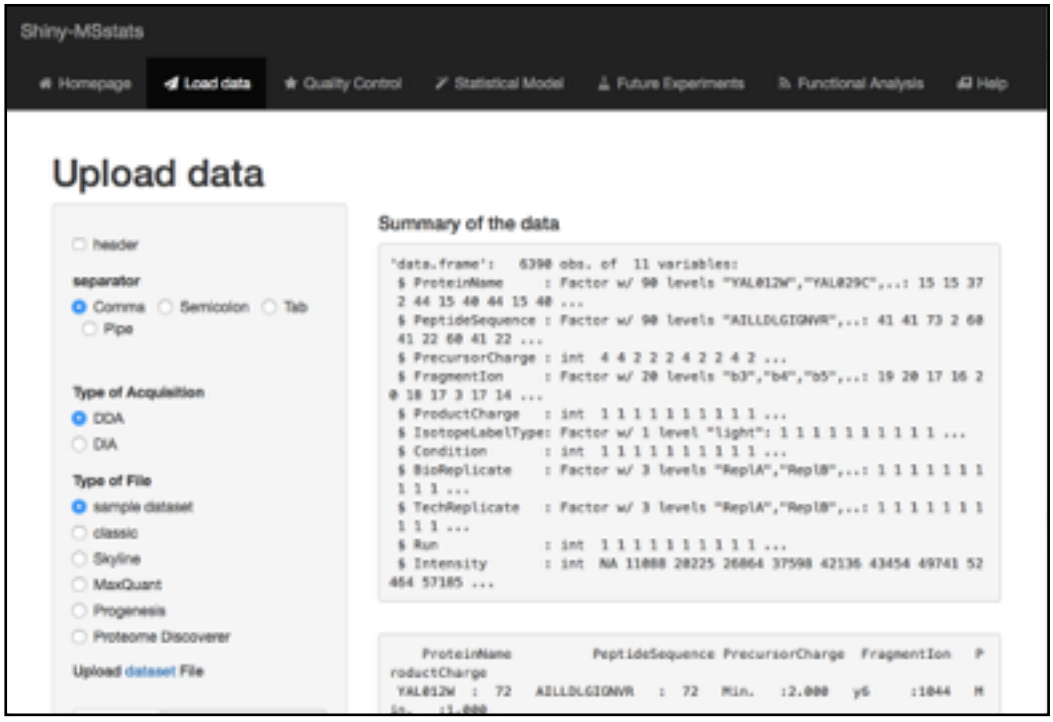

Fig.8 Upload data

On the right side the application outputs a brief high level summary of the uploaded data with the R functions str() and summary(). The

server side of this tab reads the uploaded .csv file, it transforms it based on the type of data frame and acquisition method, and then performs statistical summarisation.

- Quality control tab

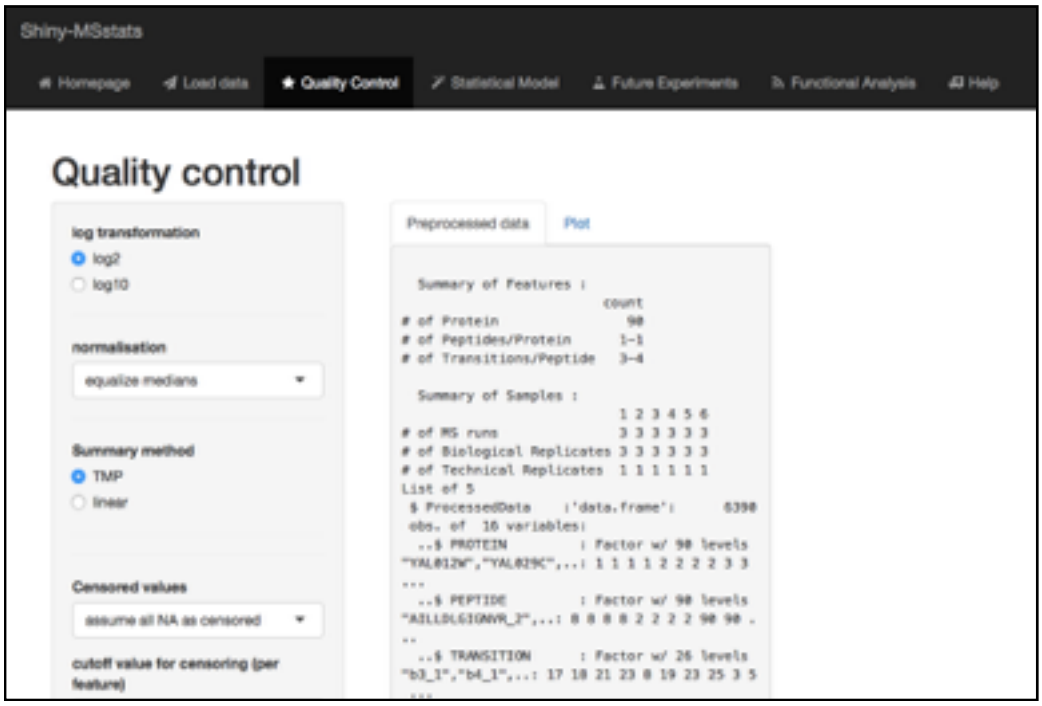

Fig.9 Quality control, tab 1

This tab is also organised with a sidebar where the user is able to choose the parameters for preprocessing the data (fig.9). The options at this stage are numerous, so the layout may need some optimisation.
In the sidebar the user can choose the log value for transformation, the normalisation method, how to handle censored values, and which features to use.
To avoid overcrowding of the side panel some features only appear when a particular input is selected. For example, upon choosing “global standards” as normalisation methods, a new select menu appears to introduce the name of the protein that will be used as

standard (taken automatically from the data frame). Other inputs follow the same behaviour.

The main panel is divided into two tabs, one to show the effects of the preprocessing on the data, the other to visualise the quality control plots. The analysis is triggered by an action button on the first tab of the main panel and the output is a summary of the preprocessed table. As the calculation often takes a few seconds, a progress bar indicates that the application is working. Once the calculation has taken place the user has the opportunity here to download the .csv of the preprocessed or summarised data.

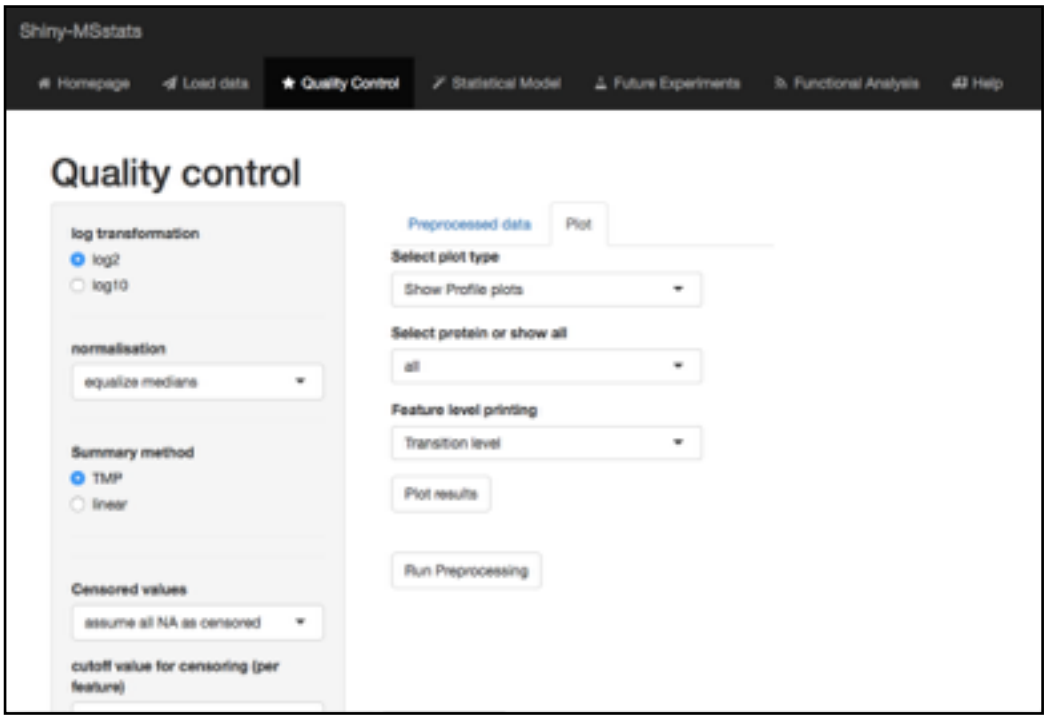

Fig.10 Quality Control, tab 2

The second tab of the main panel allows to view and download the plots relative to the preprocessing step: Profile Plots, Condition Plots and QC Plots (fig.10). In this case too, depending on the chosen type of plot, the user is able to customise different parameters. The plot files in output to the function DataProcessPlots are saved locally on a temporary folder with a

unique filename which is then accessed when the download link is clicked. The unique filename is generated through a uuid generator in the R package uuid (fig.11,12).

```
# plots

plotresult <- function() {
  progress <- Progress$new(session, min=1, max=15)
  on.exit(progress$close())

  progress$set(message = 'Calculation in progress',
               detail = 'This may take a while...')

  for (i in 1:3) {
    progress$set(value = i)
    Sys.sleep(0.5)}

  id <- as.character(UUIDgenerate(FALSE))
  id_address <- paste("tmp/",id, sep = "")
  path = paste("www/", id_address, sep = "")

  dataProcessPlots(data = preprocess_data(),
                   type=input$type,
                   featureName = input$fname,
                   ylimUp = F,
                   ylimDown = F,
                   scale = input$cond_scale,
                   interval = input$interval,
                   which.Protein = input$which,
                   originalPlot = TRUE,
                   summaryPlot = TRUE,
                   save_condition_plot_result = FALSE,
                   address = path
    )

  return(id_address)
}
```

Fig.11 Plot function in the qc-server.R script

```
# download plots

output$showplot <- renderUI({
      tags$br()
      tags$br()
      if (input$type == "ProfilePlot") {
        tagList(
          a("Open Plot", href=paste(plotresult(), "ProfilePlot.pdf", sep = ""), target="_blank"),
          tags$br(),
          a("Open Plot with summarization", href=paste(plotresult(),"ProfilePlot_wSummarization.pdf", sep = ""),
        )
      }
      else if (input$type == "ConditionPlot") {
        tagList(
          a("Open Plot", href=paste(plotresult(),"ConditionPlot.pdf", sep = ""), target="_blank")
        )
      }
      else if (input$type == "QCPlot") {
        tagList(
          a("Open Plot", href=paste(plotresult(),"QCPlot.pdf", sep = ""), target="_blank")
        )
      }
    })
```

Fig.12 Generating download links in qc-server.R script

In this section the modification of the preprocessing parameters does not reflect in the immediate visualisation of the quality control plots, as the MSstats function itself automatically generates a pdf. However at every trigger of the preprocessing button and plot results button, a new report is generated and accessed through the links.

- Statistical model tab

In this tab the user is able to perform statistical analysis of the data based on the experimental design and to then select the proteins that are found to be differentially expressed in the compared conditions.
The panel is divided into three tabs (fig.13).

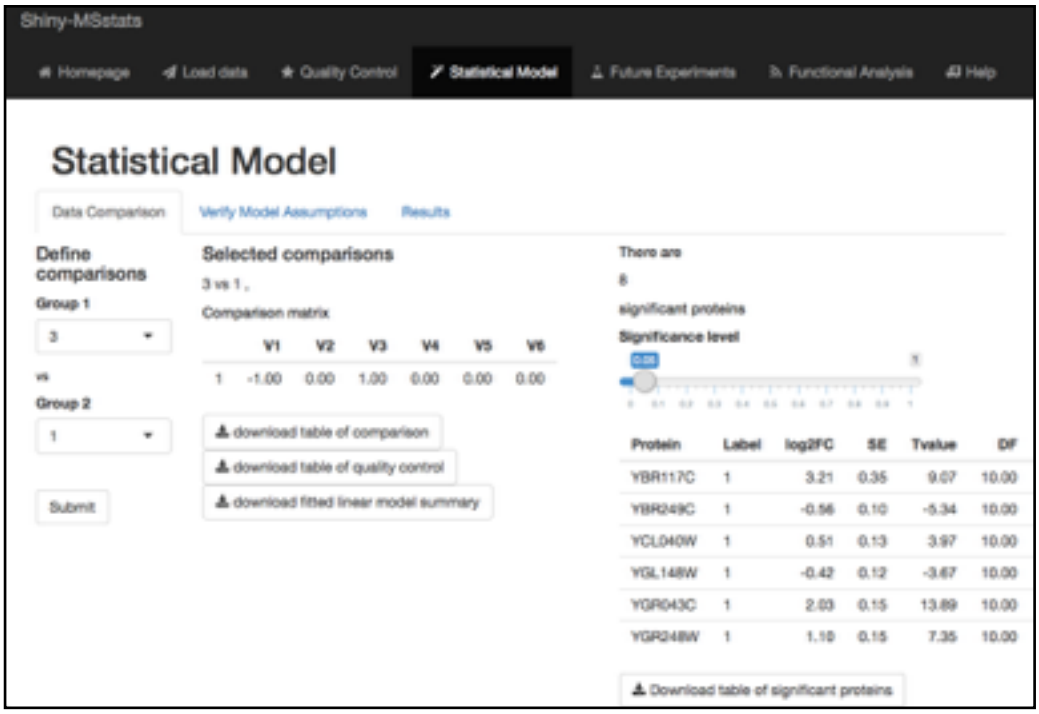

Fig.13 Statistical Model, contrast matrix

In the first tab the user will define a contrast matrix to perform comparisons between groups (or time points etc, depending on the

experimental design). The comparisons are selected with two automatically generated menus that list the experimental groups present in the dataset. The user can choose one or more comparisons to build the matrix and identical choices are discarded. The names of the comparisons (i.e. the row names of the comparison matrix) are automatically defined as incremental numbers. The submit button for the creation of the contrast matrix triggers the comparison analysis and the calculation of the number of significant proteins (depending on the significance threshold introduced by the user). The application outputs the number of significant proteins identified with the current parameters and shows the first rows of the table of significant protiens. The full table of significant proteins is downloadable at this stage, and the user is also able to download the table of comparison results, the table of quality control of the model and a summary of the linear model statistics.

The second tab allows to verify the model assumptions by plotting QQ plots and residual plots for all proteins. This time too the plots are saved locally on a temporary folder with a random unique id and accessed through links (fig.14).

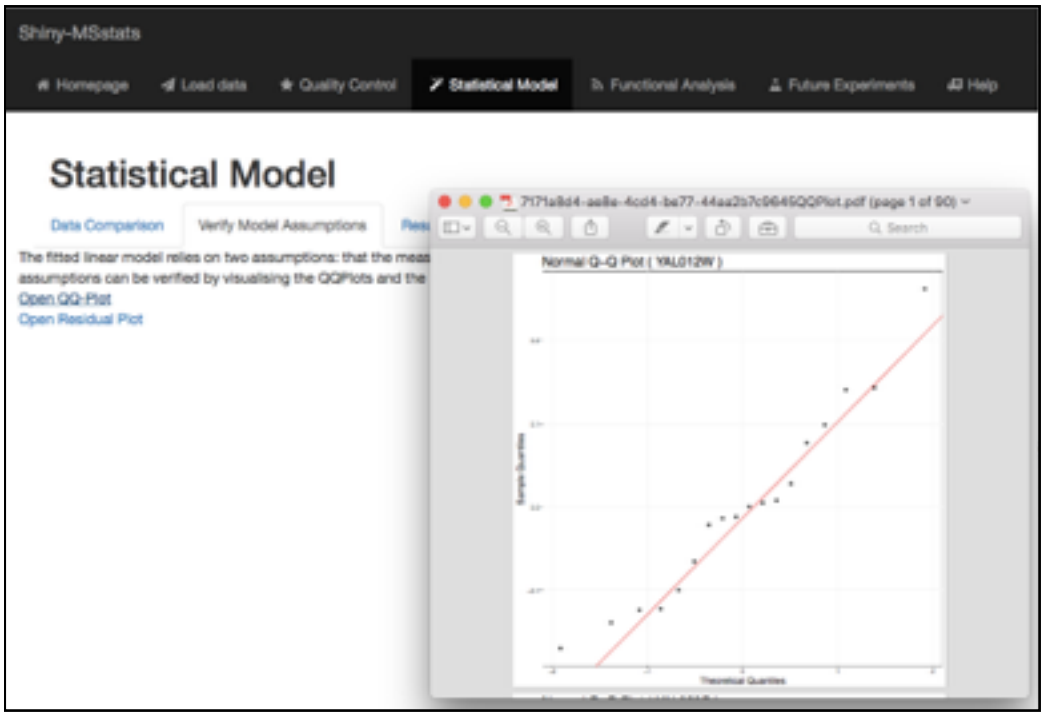

Fig.14 Statistical model, verify assumptions

In the third and last panel the user can set some parameters to plot the results of the comparisons. These plots can be Vulcano Plots, Heatmaps or Comparison Plots. Here too the user can set different parameters depending on the type of plot that he wishes to view. Here too, the plots are accessible via a link to the temporary folder where they have been saved (fig.15).

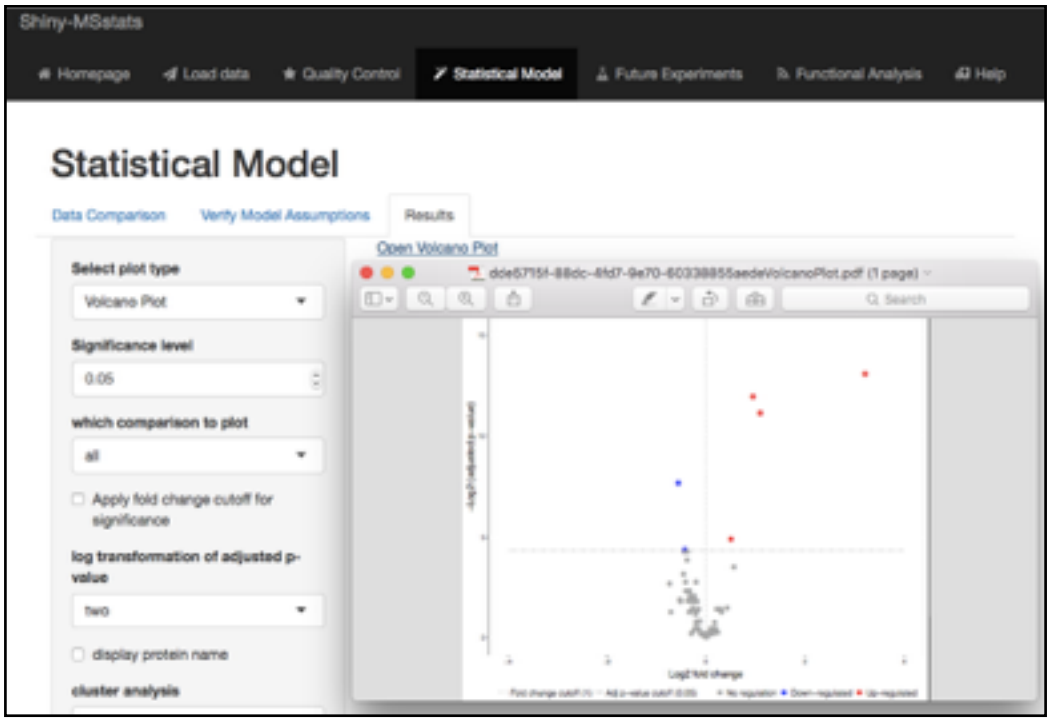

Fig.15 Statistical Model, plots

- Functional analysis tab

This tab is the only added functionality to the MSstats package. As the identification of significantly different abundances per se is not fully informative, it is often important to be able to translate these proteins into a biological function or biological process. To this end I have implemented the possibility of interrogating the database Ensembl to retrieve information on the identified proteins (fig.16). This is performed through the R package biomaRt, which queries the BioMart database (fig.17).

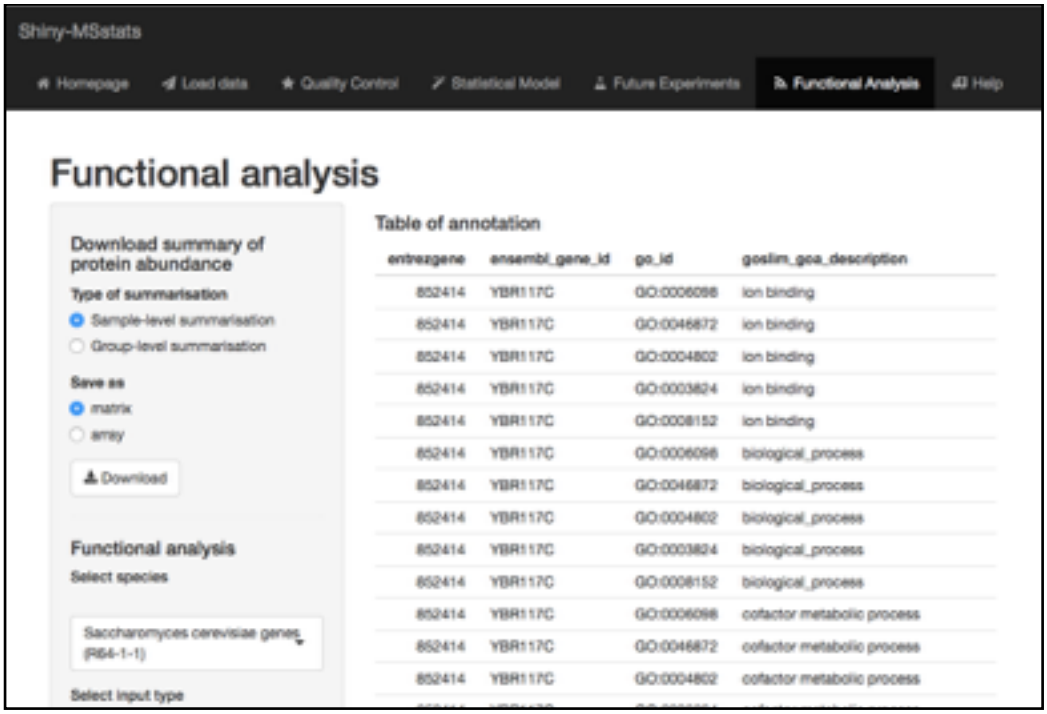

Fig.16 Functional analysis

```r
# annotation

ensembl <- useMart("ensembl")
dbs <- listDatasets(ensembl)
output$species <- renderUI({
  selectizeInput("species", "", dbs[2])
})
dataset.input <- reactive({
  as.character(dbs$dataset[dbs$description == input$species])
})
ensembl1 <- reactive(useDataset(dataset.input(), mart = ensembl))
filters <- reactive(listFilters(ensembl1()))
output$filter <- renderUI({
  selectizeInput("filter", "", filters()[2])
})
filter.input <- reactive({
  as.character(filters()$name[filters()$description == input$filter])
})
attributes <- reactive({listAttributes(ensembl1())
})
output$attributes <- renderUI ({
  selectizeInput("attribute_input", "", attributes(), multiple=TRUE, options = list(placeholder="select attributes"))
})
attribute.input <- reactive({
  as.vector(input$attribute_input)
})
id <- reactive(SignificantProteins()[1])

results <- reactive({
  getBM(attributes = attribute.input(),
                filters = filter.input(),
                values = id(),
                mart = ensembl1())
})
```

Fig.17 Annotation in analysis-server.R

Other possible annotation packages are available in R from Bioconductor, to mention a few:

- hpar: this is the interface to the Human Protein Atlas to query the database through the Ensembl id of the protein, however this is only confined to human samples and not ideal for a broad web application [24]

- GOstats: which comprises a set of tools to interact with the Gene Ontology (GO) annotations and for the visualisation of the data [25]. This package too is specific for GO annotation, therefore limiting slightly the functional analysis

- RDAVIDWebService: which amplifies the possibilities of the GOstats package by introducing multiple protein query and visualisation through the package ggplot2 [26]

However, compared to the above packages, the biomaRt package serves as interface for different data repositories such as Ensembl, COSMIC, Uniprot, HGNC, Gramene, Wormbase, making it a very powerful tool, even considering possible future expansions of this area of the application. Nevertheless, I will consider implementing other R packages for the functional analysis in the future to improve the graphical outcome of the analysis, which at the moment is lacking.

In the functional analysis tab the user can download the sample-level or group-level summarisation of the data as matrix or array.

In a second section the user can perform functional analysis with the significant proteins selected with the comparison. Currently the application is set to query the database Ensembl by default, but in the future I may be able to implement the query of other databases too.
The user must select a species and the input type (the name that indicates the proteins in the data frame) and then the attributes that the query to Ensembl with biomaRt will perform. The attributes

input allows for multiple selection and performs search while typing through the selectize function. The user is then able to download the functional annotation table.

- Future experiments tab

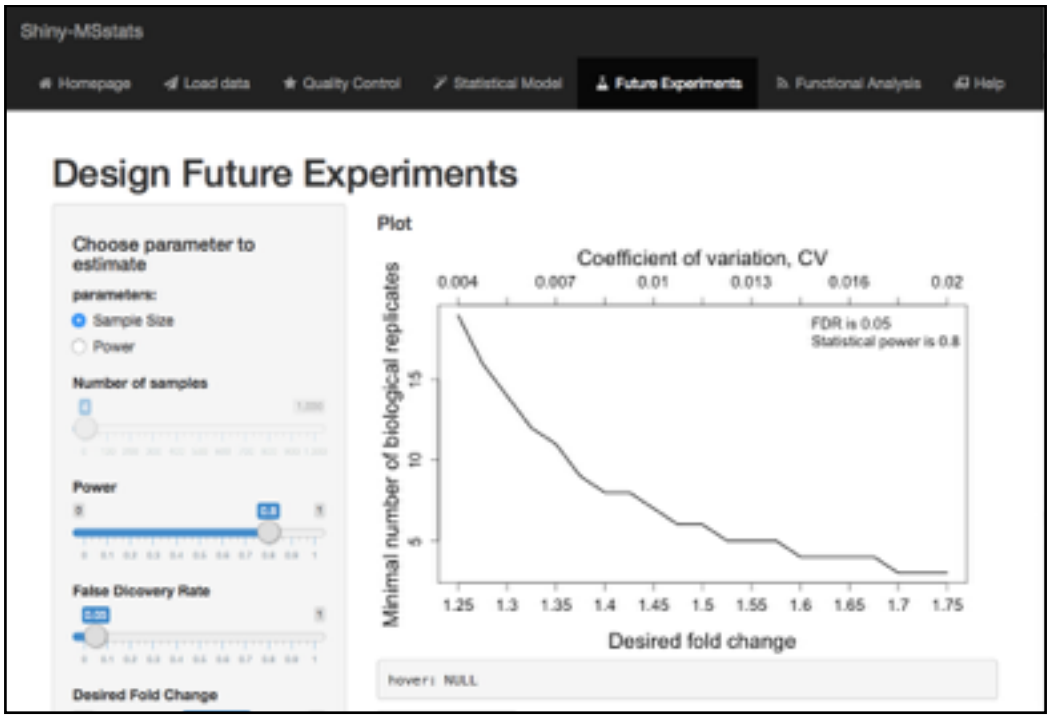

Fig.18 Design Future Experiments

The future experiment tab uses the dataset in exam as a pilot to visualise the dependency between power, number of samples and false discovery rate, given a desired fold change, in a future experiment performed under identical experimental conditions. The user can estimate one parameter between sample size and power by setting the others. A plot better indicates the dependency and by hovering on the plot, the user can retrieve the plot values. The user is able to download the plot as .png file (fig.18).
In this page I use the library shinyjs to toggle between inputs, as one between sample size and power needs to be estimated and the other introduced by the user (fig.19).

```
# toggle input elements and plot

observe({
  if (input$param == "sample") {
    shinyjs::disable("nsample")
    sample_x = TRUE
    }
  else {
    sample_x <- input$nsample
    shinyjs::enable("nsample")
  }

  if (input$param == "npower") {
    shinyjs::disable("power")
    power_x = TRUE
    }
  else {
    power_x <- input$power
    shinyjs::enable("power")
  }
  FDR_x <- input$FDR
  FCR_x <- input$desirFC
  future_exp <- function(){
    exp <- designSampleSize(data=data_comparison()$fittedmodel,
                            desiredFC = input$desirFC,
                            FDR = FDR_x,
                            numSample = sample_x,
                            power= power_x)
  }

```

Fig.19 Toggle inputs with shinyjs in expdes-server.R

This page is the most dynamic of the application, as it really reflects the reactivity of a Shiny application in which the plot in output reacts to the change in input and it is immediately appreciated.

- Help tab

In the help tab is the documentation relative to MSstats with the description of all inputs and a brief statistical explanation of the analysis (fig.20).
See the Appendix for the full documentation.

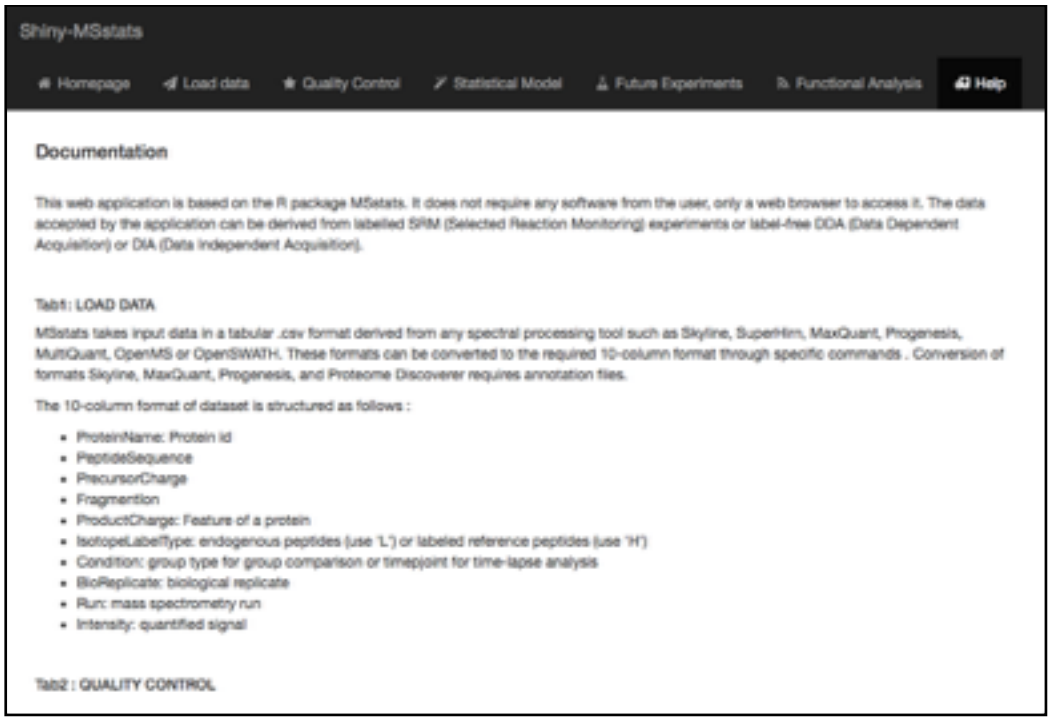

Fig.20 Help page

2.5 Deploy

The publication online of the Shiny MSstats application will be through the server shinyapps.io [27]. Because it is open source, it was designed to support Shiny applications and it is easily configured and accessed via R, it seems the best choice at this moment. With shinyapps.io the application and its usage, metrics, and settings are easily accessible through an online dashboard.
The deploy process is still pending because the server shinyapps.io first needs to replicate the working environment exactly like the one on the machine where the application was generated. This means installing all packages that the application will run. Unfortunately the package MSstats is not currently present on shinyapps.io and I have requested the technical support to add it.

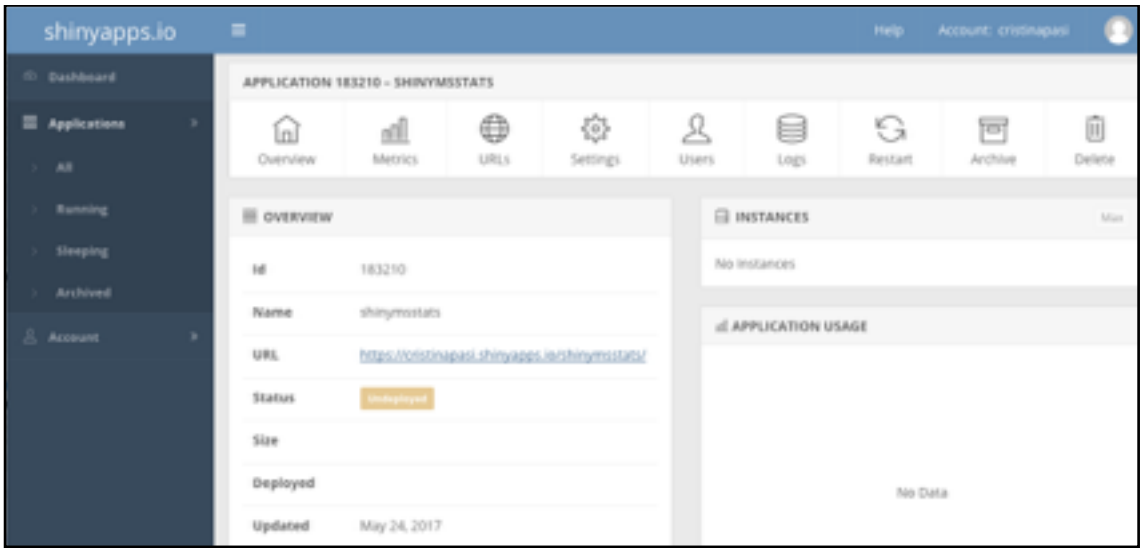

Fig.21 shinyapps.io dashboard

The limits of this server (for the free access) are set on the number of applications that one user can have deployed simultaneously (5), the number of active hours per month of the applications (25) and to the fact that there is no password protection available. In the case of Shiny-MSstats the only issue may become in the future that of the number of active hours per month. In this scenario I will re-evaluate the case and choose other alternative options such as heroku [28] or the payed version of shinyapps.io.

2.6 Tools and software

- All the coding has been carried out with R Studio (version 1.0.143) in the following environment:

R version 3.4.0 (2017-04-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X El Capitan 10.11.6

attached base packages:
parallel
stats

graphics
grDevices utils
datasets
methods
base

other attached packages:
uuid (version 0.1-2): to generate a unique user id
Biobase (version 2.34.0)
BiocGenerics (version 0.20.0)
biomaRt (version 2.30.0)
shines (version 0.9): to implement javaScript widgets in the application
shinyBS (version 0.61): to insert in the application some Bootstrap styling options
MSstats (version 3.8.2)
shiny (version 1.0.1)

- All the documentation necessary to build the application and write this report has been accessed online through web searches with the web browser Google Chrome Version 58.0.3029.110 (64-bit).

- Other software used for this work include:
    - Pages for Mac
    - Keynote
    - GANTT Project
    - Adobe Acrobat Reader

3. Conclusions

The product of this project is the online application Shiny-MSstats which can be accessed by anybody who has data from a mass-spectrometry analysis and needs to analyse it statistically for the identification of differentially abundant proteins. The sample dataset already present in the application allows also to explore what is a normal statistical analysis performed on mass spec data. In the future the addition of a couple of other sample datasets with different features (for example different experimental designs or types of acquisition) will allow the users to get familiar with the workflow before analysing their own data. Overall the application has a good functionality, with some features still needing to be implemented and some elements of the infrastructure to be optimised. In general, the user experience can still be optimised.

The Shiny framework for R applications is a very powerful and useful tool for the implementation of statistical analysis softwares with an appealing and easy to use Graphical User Interface (GUI). The documentation available online for Shiny is extensive with tutorials [29] and examples or existing applications with open source code on GitHub.
The structure of a Shiny application is easy to understand and the commands in R are easy to implement, however Shiny introduces the concept of reactivity which is a completely new concept in the pure R programming. This is very interesting and adds great versatility to R programming, as it gives the possibility of creating an R-based web application without the need to know html and css.

3.1 Completion of the objectives

The objectives that were set at the beginning of this work have been completed for just over 90% (see fig. 1-3). The incomplete

tasks have suffered principally from lack of time or from the difficulty of finding the right tools to implement them. To achieve the minimum functionality I have prioritised tasks in order to make sure to produce a working application which then can be improved regularly, as is the case for all softwares.

In general the incomplete tasks are:

- Code necessary functions in R and write guidelines and documentation: this task is completed to 90% as the functional analysis tab is still lacking a graphical output for a better representation of the data. As mentioned above, this will be done by exploring other R package to accompany biomaRt, such as GOstats or RDAVIDWebService.

- Implement drafting of summary through R markdown: Shiny gives the possibility of generating and downloading a report on the interaction with the application through knitr and markdown. To do this all parameters must be collected and passed to a markdown file, these parameters are then updated to the user's choice once the report is knitted. This task has been fairly difficult to implement as the application is quite complex with extensive code. This feature will be added in the future, trying to take advantage of the automatically generated log file from MSstats, however this poses some problems of file naming with multiple concomitant users, as the file is automatically generated by the R package.

- Implement parameters to be regulated by the user: user interface: Currently almost all parameters that may be modified through MSstats are editable by the users with the exception of the graphical parameters for the plots. For the moment these parameters have not been added to lighten up a little the application, which already is quite complex and has a great number of variables at play. Once the graphical interface is more fluent and usability more intuitive, I will consider adding the possibility of managing the graphical parameters of plots.

- Verify the functionality of the application: try with available datasets and review and refactor code for responsiveness and usability. Extensive user testing to check all functionalities has not been performed yet. This will be done during testing, during verification and through usage of the application.

3.2 Problems encountered and their solution

During the completion of this work, it has not always been easy to comply with the project plan. Conceptually, the proposed methodology was correct and has been followed like planned. However the timeframe of the project has needed some adaptation because some task have taken longer to be completed because of some complication or the difficulty in finding the correct documentation. The only conceptual deviation from the project plan has been that I have not implemented the output of the plots as a carousel in a popup window. This is due to the fact that the functions in MSstats automatically generate a pdf file. Currently these files are accessible through a link to the temporary folder where they are stored. I will explore the possibility of redirecting the output to a browser page by setting the parameter “address” to FALSE.

The workload was divided into: project planning, R coding and Shiny implementation. This approach worked out to be a good approach, as during the R coding I was able to focus on the statistical side of the application, whereas during the Shiny implementation I could concentrate on the interface and functionality of the application rather than the statistical functions connected with it. The development of the Shiny interface has taken longer than expected due to the fact that it was the first time for me to approach Shiny altogether. In this case, in fact, I have had to document myself and learn the basic concepts in Shiny prior to being able to apply them concretely.

3.2 Future work

Together with the completion of the tasks that have not been carried out fully (see above), several aspects of the application still need to be improved and changed. Mostly, these improvements are set to render the application more usable, to give the user more information and guidance, and to make the application in general accessible to even the most unaccustomed users.
Like it is for all software, these improvements will be prioritised based on importance and carried out in the future as the application is already running and available to the public. Clearly, these faults in the application do not impair its functionality, therefore the first basic version of the application can be considered complete.

The improvements I will add to the application in the future are listed below.

- General improvements:

    - Improve the workflow, as the stepwise nature of the analysis is not entirely clear at the moment for users who are not accustomed to the MSstats package. A diagram of the steps and links to the different tabs may be a possible solution, or to render the subsequent tabs visible only once the preceding step has been completed. The Homepage, Load Data and Help tabs will be always present.

    - Add more written guidelines for the user across the application, to explain what each option represents and to help decide how to insert the different parameters. Ideally every tab will have at the bottom the description of the page explaining the inputs and the outputs of the application and the flow of the analysis.

    - Introduce the possibility of downloading the MSstat log file which is automatically generated by MSstats and saved in the local folder of the application. I will have to make sure the log file is named uniquely and therefore uniquely

accessible by every user. Alternatively, I will use the ability of Shiny to generate downloadable reports through R Markdown.

- Add the possibility of using different versions of MSstats when new versions are released.

• Upload data section

- Keep only one upload data button to add the data frame and eventually any accompanying annotation file depending on the nature of the data frame itself. The type of uploaded file will serve to determine if it is the annotation file or the data frame.

- Add more documentation on the sample dataset in such a way that the user knows what kind of an experiment he is analysing when exploring the application. As mentioned, I would like to introduce a few different trial datasets to try different types of analysis.

• Quality control section

- Suggest preprocessing settings depending on input: depending on the type of experiment and in some cases whether the initial uploaded file is of a different type compared to the classic 10-column dataset, some settings are more indicated than others (especially in handling missing values). I will implement controls to suggest the best settings to the user if the dataset was different from the classic one

- Add download of Between Run Interference Score if the user requires to, which at the moment is lacking

- Statistical model section

    - Improve the generation of the contrast matrix by introducing the possibility of renaming the comparisons, as these names are then used for plotting subsequently.

- Functional analysis section

    - Visualise the significant protein table (produced by the contrast function) together with the table of attributes that the user chooses to extract from Ensembl. One way could be to default the extraction of the type of id that the proteins are coded in in the significant protein dataset and then use this common column to merge the two datasets in one.

- Design future experiment section

    - Limit the desired fold range extension to avoid errors by MSstats

4. Glossary

LC: Liquid Chromatography
DDA: Data Dependent Acquisition
MS/MS: Tandem Mass Spectrometry
DIA: Data Independent Acquisition
SRM: Selected Reaction Monitoring
SILAC: Stable Isotope Labelling by Aminoacids in Culture
iTRAQ: isobaric Tags for Relative and Absolute Quantification
GO: Gene Ontology

5. Bibliography

1. Aebersold, R., & Mann, M. (2016). Mass-spectrometric exploration of proteome structure and function. Nature, 537(7620), 347–355. http://doi.org/10.1038/nature19949
2. Choi, M., Chang, C.-Y., Clough, T., Broudy, D., Killeen, T., MacLean, B., & Vitek, O. (2014). MSstats: an R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments. Bioinformatics (Oxford, England), 30(17), 2524–6. http://doi.org/10.1093/bioinformatics/btu305
3. MacLean, B., Tomazela, D. M., Shulman, N., Chambers, M., Finney, G. L., Frewen, B., … MacCoss, M. J. (2010). Skyline: An open source document editor for creating and analyzing targeted proteomics experiments. Bioinformatics, 26(7), 966–968. http://doi.org/10.1093/bioinformatics/btq054
4. Moniz, L. S., Surinova, S., Ghazaly, E., Velasco, L. G., Haider, S., Rodríguez-Prados, J. C., … Vanhaesebroeck, B. (2017). Phosphoproteomic comparison of Pik3ca and Pten signalling identifies the nucleotidase NT5C as a novel AKT substrate. Scientific Reports, 7, 39985. http://doi.org/10.1038/srep39985
5. Hartwig, T., Montinaro, A., von Karstedt, S., Sevko, A., Surinova, S., Chakravarthy, A., … Walczak, H. (2017). The TRAIL-Induced Cancer Secretome Promotes a Tumor-Supportive Immune Microenvironment via CCR2. Molecular Cell, 65(4), 730–742.e5. http://doi.org/10.1016/j.molcel.2017.01.021
6. Lobingier, B. T., Hü, R., Eichel, K., Miller, K. B., Ting, A. Y., Von Zastrow, M., & Krogan, N. J. (2017). An Approach to Spatiotemporally Resolve Protein Interaction Networks in Living Cells Resource An Approach to Spatiotemporally Resolve Protein Interaction Networks in Living Cells. Cell, 169(6), 350–360. http://doi.org/10.1016/j.cell.2017.03.022
7. Caron, E., Roncagalli, R., Hase, T., Wolski, W. E., Choi, M., Menoita, M. G., … Gstaiger, M. (2017). Precise Temporal Profiling of Signaling Complexes in Primary Cells Using SWATH Mass Spectrometry. Cell Reports, 18(13), 3219–3226. http://doi.org/10.1016/j.celrep.2017.03.019

8. http://msstats.org/wp-content/uploads/2017/01/MSstats_v3.7.3_manual.pdf , March 2017
9. http://shiny.rstudio.com/tutorial/ , March 2017
10. Ong, S.-E. (2002). Stable Isotope Labeling by Amino Acids in Cell Culture, SILAC, as a Simple and Accurate Approach to Expression Proteomics. Molecular & Cellular Proteomics, 1(5), 376–386. https://doi.org/10.1074/mcp.M200025-MCP200
11. Hu, A., Noble, W. S., & Wolf-Yadlin, A. (2016). Technical advances in proteomics: new developments in data-independent acquisition. F1000Research, 5, 419. http://doi.org/10.12688/f1000research.7042.1
12. Bruderer, R., Bernhardt, O. M., Gandhi, T., Xuan, Y., Sondermann, J., Schmidt, M., … Reiter, L. (2017). Heralds of parallel MS: Data-independent acquisition surpassing sequential identification of data dependent acquisition in proteomics. Molecular & Cellular Proteomics, mcp.M116.065730. http://doi.org/10.1074/mcp.M116.065730
13. Gillet, L. C., Navarro, P., Tate, S., Rost, H., Selevsek, N., Reiter, L., … Aebersold, R. (2012). Targeted Data Extraction of the MS/MS Spectra Generated by Data-independent Acquisition: A New Concept for Consistent and Accurate Proteome Analysis. Molecular & Cellular Proteomics, 11(6), O111.016717-O111.016717. http://doi.org/10.1074/mcp.O111.016717
14. Picotti, P., & Aebersold, R. (2012). Selected reaction monitoring–based proteomics: workflows, potential, pitfalls and future directions. Nature Methods, 9(6), 555–566. https://doi.org/10.1038/nmeth.2015
15. MacLean, B., Tomazela, D. M., Shulman, N., Chambers, M., Finney, G. L., Frewen, B., … MacCoss, M. J. (2010). Skyline: An open source document editor for creating and analyzing targeted proteomics experiments. Bioinformatics, 26(7), 966–968. https://doi.org/10.1093/bioinformatics/btq054
16. Cox, J., & Mann, M. (2008). MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification. Nature Biotechnology, 26(12), 1367–72. https://doi.org/10.1038/nbt.1511
17. http://www.nonlinear.com/progenesis/qi/ , March 2017
18. https://www.thermofisher.com/order/catalog/product/IQLAAEGABSFAKJMAUH , March 2017

19. Röst, H. L., Rosenberger, G., Navarro, P., Gillet, L., Miladinović, S. M., Schubert, O. T., … Aebersold, R. (2014). OpenSWATH enables automated, targeted analysis of data-independent acquisition MS data. Nature Biotechnology, 32(3), 219–223. https://doi.org/10.1038/nbt.2841
20. https://biognosys.com/shop/spectronaut , March 2017
21. http://www.ensembl.org/index.html , March 2017
22. http://www.ensembl.org/info/data/biomart/index.html#biomartapi , March 2017
23. Selevsek, N., Chang, C.-Y., Gillet, L. C., Navarro, P., Bernhardt, O. M., Reiter, L., … Aebersold, R. (2015). Reproducible and Consistent Quantification of the Saccharomyces cerevisiae Proteome by SWATH-mass spectrometry. Molecular & Cellular Proteomics, 14(3), 739–749. https://doi.org/10.1074/mcp.M113.035550
24. https://bioconductor.org/packages/release/bioc/html/hpar.html ,March 2017
25. https://www.bioconductor.org/packages/release/bioc/html/GOstats.html , March 2017
26. https://www.bioconductor.org/packages/release/bioc/html/RDAVIDWebService.html , March 2017
27. http://www.shinyapps.io/, March 2017
28. https://www.heroku.com/ , March 2017
29. https://shiny.rstudio.com/tutorial/ , March 2017

6. Appendix

6.1 Documentation

This web application is based on the R package MSstats. It does not require any software from the user, only a web browser to access it.

The data accepted by the application can be derived from labelled SRM (Selected Reaction Monitoring) experiments or label-free DDA (Data Dependent Acquisition) or DIA (Data Independent Acquisition).

- Tab1: LOAD DATA

MSstats takes input data in a tabular .csv format derived from any spectral processing tool such as Skyline, SuperHirn, MaxQuant, Progenesis, MultiQuant, OpenMS or OpenSWATH. These formats can be converted to the required 10-column format through specific commands .
Conversion of formats Skyline, MaxQuant, Progenesis, and Proteome Discoverer requires annotation files.

The 10-column format of dataset is structured as follows (https://github.com/MeenaChoi/MSstats_document/blob/master/MSstats_v3/MSstats.Rmd#2-allowable-data-formats):

- *ProteinName*: Protein id
- *PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge:* Feature of a protein
- *IsotopeLabelType*: endogenous peptides (use "L") or labeled reference peptides (use “H").
- *Condition*: group type for group comparison or timepjoint for time-lapse analysis.
- *BioReplicate*: biological replicate
- *Run*: mass spectrometry run
- *Intensity*: quantified signal

- Tab2 : QUALITY CONTROL

The data is transformed, normalised and summarised for statistical modelling.

- Transformation: the Intensity column is transformed in either log2 or log10
- Normalisation: the available methods are Equalisation of medians (indicated when the majority of proteins don't change between runs, less indicated for label-free DDA), use of Global standards (defined by the user), 'quantile' (for label-free, all intensity distributions will become equal across runs; for label-based, all intensity distributions for references will become equal across runs and endogenous intensities shifted according to references), or no normalisation.
- Feature subset: use all feature or top 3 most significant (highest log2 transformed intensity) or top n (custom) most significant, or 'high quality' for most informative features (to eliminate unexplainable variation in features - interference)
- Handling missing values (random missing measurements are independent of the abundance of the peptide; censored missing measurements are due to very low abundance, which falls below the level of detection of the instrument):
  - Assume all NAs as censored
  - Assume all intensities between 0 and 1 as censored and NAs as random missing (value of 1 can be changed)
  - Assume that all missing values are random missing

With the assumption of the presence of censored values, a cutoff for the AFT model is determined:

- Minimum value for each feature across runs
- Minimum value for each run across features
- Smallest value between minimum value of corresponding feature and minimum value of corresponding run (in this case runs with substantial missing runs will be biased so it may be a good option to remove all runs with more than 50% missing values)

- Summary method: TMP (Turkey's median polish) or linear (linear mixed model). TMP can be with model-based imputation of missing values (censored missing values will be imputed by

Accelerated Failure Time model and cutoff is defined by the user - see above) or without imputation (in this case censored values will be substituted with a cutoff value specified by the user - see above). With linear summary method no imputation is performed; in the presence of censored values parameters are estimated using AFT model, otherwise if all missing values are assumed to be random, the parameters are estimated through linear modelling. With linear summary the variance in intensity from features can be considered equal or heterogeneous.

The preprocessed data is visualised through plots (for all proteins or individual proteins):

- QC plot: to visualise systematic biases between runs and to view the effects of normalisation
- Profile plot: to identify potential sources of variation (individual measurements per protein and summarised data) and show missing data
- Condition plot: to visualise potential differences in protein intensities between conditions (shown with arbitrary confidence interval or standard deviation bars).

• Tab3 : STATISTICAL MODEL

Based on the design of the experiment in analysis, a number of contrasts to be tested are specified in the application. The contrasts will be between different conditions, different time points or different subjects of conditions for each subject. The application then performs comparisons between the specified groups and returns a table with fold change and significance. Positive values of fold change indicate up regulation, whereas negative values indicate down regulation.

The statistical model is based on two assumptions:

- normal distribution of the measurement errors (QQ plot)
- constant variance of the measurement error (residual plot)

The results of the group comparisons may be visualised with:

- Vulcano plot: to show the comparison between groups and the significance of the difference (adjusted p-value on the y axis): in

red are the unregulated proteins, in blue the down regulated ones. An horizontal line signals the false discovery rate cutoff (proteins above the line are statistically significant). Each comparison has its own plot.
- Heat map: to view the patterns of regulation of proteins (rows) in the comparisons (columns). Max 180 proteins will be shown for heat map.
- Comparison plot: to show log-fold changes in the different comparisons for each protein.

• Tab4 : FUTURE EXPERIMENTS

In light of the statistical analysis performed in the application, the dataset can be viewed as a pilot study (assuming same experimental design and same expression of 99% of proteins) for future experiments, in order to relate the following statistical parameters:
- Sample size: number of biological replicates per sample, number of peptides per protein, number of transition per peptide
- Power: average across all proteins
- Fold change: minimal fold change to detect or range
- False discovery rate

All parameters must be specified but one.

The results of the estimations are visualised with plots.

• Tab5 : FUNCTIONAL ANALYSIS

The functional analysis is performed with the R package biomaRt, which accesses the Ensembl database for retrieval of GO terms and KEGG terms.
The analysis requires the selection of the following parameters:
- Organism: in which the analysis is performed (es. Homo sapiens, Mus musculus, etc)
- ID type: format of ID for the proteins in the dataset (es. entrez gene, ensemble gene id, etc)
- Attributes: features to retrieve from the query to the Ensembl database (es. GO id, GO description, KEGG id, etc)

6.2 Full R code

The full code is also available at https://github.com/cristinapasi/ShinyMSstats .

ui.R

```
library(shiny)


source("panels/home-ui.R", local = T)
source("panels/loadpage-ui.R", local = T)
source("panels/qc-ui.R", local = T)
source("panels/statmodel-ui.R", local = T)
source("panels/expdes-ui.R", local = T)
source("panels/analysis-ui.R", local = T)
source("panels/help-ui.R", local = T)

##################################################################
#########

ui <- navbarPage(
  title = "Shiny-MSstats",
  tabPanel("Homepage", icon = icon("home"), home),
  tabPanel("Load data", icon = icon("send"), loadpage),
  tabPanel("Quality Control", icon = icon("star"), qc),
  tabPanel("Statistical Model", icon = icon("magic"),
statmodel),
  tabPanel("Functional Analysis", icon = icon("feed"),
analysis),
  tabPanel("Future Experiments", icon = icon("flask"), expdes),
  tabPanel("Help", icon = icon("ambulance"), help),
  inverse = T,
  collapsible = T,
  windowTitle = "Shiny-MSstats"
)


shinyUI(ui)
```

server.R

```
library(shiny)
```

```
library(MSstats)
library(shinyBS)
library(uuid)
library(shinyjs)
library(biomaRt)
library(Biobase)


shinyServer(function(input, output, session) {
  # load data
  source("panels/loadpage-server.R", local = T)
  # quality control
  source("panels/qc-server.R", local = T)
  # statistical model
  source("panels/statmodel-server.R", local = T)
  # functional analysis
  source("panels/analysis-server.R", local = T)
  # future experiment
  source("panels/expdes-server.R", local = T)

})
```

home-ui.R

```
home = fluidPage(
  headerPanel("Welcome to Shiny MSstats"),
  tags$br(),
  mainPanel(
    div(tagList(
      h4("About Shiny MSstats"),
      p("This is a web tool for the statistical analysis of
quantitative proteomic data.  It is built based on the R package
MSstats (v 3.3.1).  The full code can be accessed online at "),
      a("https://github.com/cristinapasi/ShinyMSstats.",
href="https://github.com/cristinapasi/ShinyMSstats"),
      br()
      )
      )
    )
  )
```

loadpage-ui.R

```
sbp_load = sidebarPanel(

  #  uiOutput('MSdataset'),

  checkboxInput("header", "header"),
  radioButtons("sep", "separator", c(Comma=",",Semicolon=";",
Tab="\t",Pipe="|"), inline = T),
```

```
  tags$hr(),

  # selection for DIA or DDA

  radioButtons(
    'DDA_DIA', "Type of Acquisition",
    c(DDA = "DDA", DIA = "DIA")),


  # DDA

  conditionalPanel(
    condition = "input.DDA_DIA == 'DDA'",


  # selection for file type

  radioButtons(
    'filetype', "Type of File",
    c("sample dataset" = "sample", "classic" = "10col", "Sky-
line" = "sky", "MaxQuant" = "maxq", "Progenesis" = "prog", "Pro-
teome Discoverer" = "PD")),

  # upload

  h5("Upload",
     a("dataset", href="https://bioconductor.org/packages/devel/
bioc/vignettes/MSstats/inst/doc/MSstats.html"),
     "File"),
  fileInput('data', "", multiple = F, accept = c("text/csv",
"text/comma-separated-values,text/plain", ".csv")),

  h6("Upload annotation File - only for Skyline, MaxQuant, Prog-
enesis, Proteome Discoverer"),
  fileInput('annot', "", multiple = F, accept = c("text/csv",
"text/comma-separated-values,text/plain", ".csv")),

  h6("Upload evidence File - only for MaxQuant"),
  fileInput('evidence', "", multiple = F, accept = c("text/csv",
"text/comma-separated-values,text/plain", ".csv"))

  ),


  # DIA

  conditionalPanel(
    condition = "input.DDA_DIA == 'DIA'",

  radioButtons(
    'filetype', "Type of File",
```

```
    c(classic = "10col", Skyline = "sky", Spectronaut = "spec",
OpenSWATH = "open")),

  # upload

  h5("Upload",
     a("dataset", href="https://bioconductor.org/packages/devel/
bioc/vignettes/MSstats/inst/doc/MSstats.html"),
     "File"),
  fileInput('data', "", multiple = F, accept = c("text/csv",
"text/comma-separated-values,text/plain", ".csv")),

  h6("Upload annotation File - only OpenSWATH"),
  fileInput('annot', "", multiple = F, accept = c("text/csv",
"text/comma-separated-values,text/plain", ".csv"))
  )
)
```

```
loadpage = fluidPage(
  headerPanel("Upload data"),
  sbp_load,
  column(width = 8,
         h4("Summary of the data"),
         verbatimTextOutput('summary'),
         tags$br(),
         verbatimTextOutput('summary1')
  )
 )
```

loadpage-server.R

```
### functions ###

get_annot = reactive({
  annot <- input$annot
  if(is.null(annot))
    return(NULL)
  read.csv(annot$datapath)
})

get_evidence = reactive({
  evidence <- input$evidence
  if(is.null(evidence))
    return(NULL)
  read.csv(evidence$datapath)
})
```

```
get_data = reactive({
```

```
  if(input$filetype == 'sample') {
    mydata <- read.csv("dataset.csv", header = T, sep = ";")
  }

  else {

  infile <- input$data
  if(is.null(infile))
    {return(NULL)}

  if(input$filetype == '10col') {
    mydata <- read.csv(infile$datapath, header = input$header, sep =
input$sep)
  }
  else if(input$filetype == 'sky') {
      mydata <- SkylinetoMSstatsFormat(data, annotation = get_annot())
    }
  else if(input$filetype == 'maxq') {
      mydata <- MaxQtoMSstatsFormat(data, annotation = get_annot(),
evidence = get_evidence())
    }
  else if(input$filetype == 'prog') {
      mydata <- ProgenesistoMSstatsFormat(data, annotation = get_an-
not())
    }
  else if(input$filetype == 'PD') {
      mydata <- PDtoMSstatsFormat(data, annotation = get_annot())
    }
  else if(input$filetype == 'spec') {
      mydata <- SpectronauttoMSstatsFormat(data)
    }
  else if(input$filetype == 'open') {
      raw <- sample_annotation(data=data,
                               sample.annotation=get_annot(),
                               data.type='OpenSWATH')
      data.filtered <- filter_mscore(raw, 0.01)
      data.transition <- disaggregate(data.filtered)
      mydata <- convert4MSstats(data.transition)
  }}
  mydata <- unique(data.frame(mydata))
  return(mydata)
})



### outputs ###


output$summary <- renderPrint(
  {
    req(get_data())
    str(get_data())
  }
)

output$summary1 <-  renderPrint(
```

```
  {
    req(get_data())
    summary(get_data())
  }
)
```

qc-ui.R

```
### sidebar ###

  sbp_params = sidebarPanel(

  # transformation
  radioButtons("log", "log transformation", c(log2 = "2", log10
= "10")),
  tags$hr(),
  selectInput('norm', "normalisation", c("none" = "FALSE",
"equalize medians" = "equalizeMedians", "quantile" = "quantile",
"global standards" = "globalStandards"), selected = "equalizeMe-
dians"),
  conditionalPanel(condition = "input.norm ==
'globalStandards'",
                   uiOutput("Names")


  ),
  tags$hr(),

  # summary method
  radioButtons("method", "Summary method", c("TMP" = "TMP",
"linear" = "linear")),
  tags$hr(),

  # equal var
  conditionalPanel(condition = "input.method == 'linear'",
                   checkboxInput("equal", "do not assume equal
variance among intensities")),

  # remove runs with more than 50%missing
  conditionalPanel(condition = "input.method == 'tmp'",
                   checkboxInput("remove50", "remove runs with
over 50% missing values")),
  tags$hr(),

  # censored
  selectInput('censInt', "Censored values", c("assume all NA as
censored" = "NA", "assume all between 0 and 1 as censored" =
"0", "all missing values are random" = "null"), selected =
"NA"),

  # cutoff for censored
```

```
  conditionalPanel(condition = "input.censInt == 'NA' || input.-
censInt == '0'",
                   selectInput("cutoff", "cutoff value for cen-
soring (per feature)", c("min value"="minFeature", "smallest be-
tween min value and min run"="minFeatureNRun", "min run"="min-
Run"))),

  # max quantile for censored
  numericInput("maxQC", "max quantile for censored", 0.99, 0.00,
1.00, 0.01),

  # MBi
  conditionalPanel(condition = "input.method == 'TMP' && (in-
put.censInt == 'NA' || input.censInt == '0')",
                   checkboxInput("MBi", "impute with cutoff val-
ues", value = TRUE)),
  tags$hr(),


  # features
  radioButtons("feat", "Used features", c("all"="all",
"top3"="top 3", "topn"="top n")),
  conditionalPanel(condition = "input.feat == 'top n'",
                   numericInput("n_feat", "number of features",
3, 3, 100, 1)),

  # interference score
#  checkboxInput("score", "create interference score", value =
FALSE),
#  conditionalPanel(condition = "input.score == true",
#                   actionButton("download1", "Download .csv")),

  # fill incomplete rows
  checkboxInput("fill", "fill incomplete rows", value = TRUE),

  # remove proteins with interference
  checkboxInput("interf", "remove proteins with interference",
value = FALSE)



  )


### main panel ###

main = mainPanel(


  tabsetPanel(
    tabPanel("Preprocessed data",
             verbatimTextOutput('effect'),
             tags$div(id='placeholder')),
```

```
    tabPanel("Plot",
             selectInput("type","Select plot type", c("Show Pro-
file plots"="ProfilePlot","Show Condition
plot"="ConditionPlot","Show QC plots"="QCPlot")),
             uiOutput("Which"),
             conditionalPanel(condition = "input.type == 'Pro-
filePlot'",
                              selectInput("fname", "Feature lev-
el printing", c("Transition level"="Transition", "Peptide
level"="Peptide", "No feature legend"="NA"))
                              ),
             conditionalPanel(condition = "input.type == 'Condi-
tionPlot'",
                              checkboxInput("cond_scale", "Scale
conditional level at x-axis", value = FALSE),
                              radioButtons("interval", "width of
error bar", c("use Confidence Interval"="CI", "use Standard De-
viation"="SD"))
             ),
             actionButton("goplot", "Plot results"),
             conditionalPanel(condition = "input.goplot > 0",
                              uiOutput("showplot")
    )


             )

  ),
  tags$br(),
  tags$br(),
  actionButton("run", "Run Preprocessing")
)


##########################################################################
########################

qc = fluidPage(
  headerPanel("Quality control"),
  sbp_params,
  column(width = 8,
  main
 )
  #  fluidRow(column(width = 12,
  #                  includeMarkdown(""))

  #  )
)
```

qc-server.R

```
# standards name

output$Names <- renderUI({
  selectInput("names", "choose standard", unique(get_data()[1]))
})

# preprocess data

preprocess_data = eventReactive(input$run, {
  preprocessed <- dataProcess(raw=get_data(),
                              logTrans=input$log,
                              normalization=input$norm,
                              nameStandards=input$names,
#                               betweenRunInterferenceScore=in-
put$interf,
                              fillIncompleteRows=input$fill,
                              featureSubset=input$feat,

remove_proteins_with_interference=input$interf,
                              n_top_feature=input$n_feat,
                              summaryMethod=input$method,
                              equalFeatureVar=input$equal,
                              censoredInt=input$censInt,
                              cutoffCensored=input$cutoff,
                              MBimpute=input$MBi,
                              maxQuantileforCensored=input$max-
QC,
                              remove50missing=input$remove50
 #                              skylineReport=input$report
)
  return(preprocessed)
})


# output preprocessed data

observeEvent(input$run, {
  output$effect <- renderPrint(
  {
    progress <- Progress$new(session, min=1, max=15)
    on.exit(progress$close())

    progress$set(message = 'Calculation in progress',
                 detail = 'This may take a while...')

    for (i in 1:3) {
      progress$set(value = i)
      Sys.sleep(0.5)
    }
    str(preprocess_data())
```

```
  }
)

  insertUI(selector = "#placeholder",
           where = "afterEnd",
            ui= tags$div(tags$br(),

downloadButton("prepr_csv","Download .csv of preprocessed
data"),

downloadButton("summ_csv","Download .csv of summarised data")
                           )
  )
  })

# plots

plotresult <- function() {
  progress <- Progress$new(session, min=1, max=15)
  on.exit(progress$close())

  progress$set(message = 'Calculation in progress',
               detail = 'This may take a while...')

  for (i in 1:3) {
    progress$set(value = i)
    Sys.sleep(0.5)}

  id <- as.character(UUIDgenerate(FALSE))
  id_address <- paste("tmp/",id, sep = "")
  path = paste("www/", id_address, sep = "")

  dataProcessPlots(data = preprocess_data(),
                   type=input$type,
                   featureName = input$fname,
                   ylimUp = F,
                   ylimDown = F,
                   scale = input$cond_scale,
                   interval = input$interval,
                   which.Protein = input$which,
                   originalPlot = TRUE,
                   summaryPlot = TRUE,
                   save_condition_plot_result = FALSE,
                   address = path
    )

  return(id_address)
}
```

```
# download preprocessed data

output$prepr_csv <- downloadHandler(
  filename = function() {
    paste("data-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(preprocess_data()$ProcessedData, file)
  }
)

output$summ_csv <- downloadHandler(
  filename = function() {
    paste("data-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(preprocess_data()$RunlevelData, file)
  }
)

# which protein to plot

output$Which <- renderUI({
  selectInput("which", "Select protein or show all", c("all",
unique(get_data()[1])))
  })

# download plots

output$showplot <- renderUI({
      tags$br()
      tags$br()
      if (input$type == "ProfilePlot") {
        tagList(
          a("Open Plot", href=paste(plotresult(), "Profile-
Plot.pdf", sep = ""), target="_blank"),
          tags$br(),
          a("Open Plot with summarization", href=paste(plotre-
sult(),"ProfilePlot_wSummarization.pdf", sep = ""),
target="_blank")
        )
      }
      else if (input$type == "ConditionPlot") {
        tagList(
          a("Open Plot", href=paste(plotresult(),"Condition-
Plot.pdf", sep = ""), target="_blank")
        )
      }
      else if (input$type == "QCPlot") {
        tagList(
          a("Open Plot", href=paste(plotresult(),"QCPlot.pdf",
sep = ""), target="_blank")
        )
```

```
      }
    })
```

statmodel-ui.R

```
statmodel = fluidPage(
  headerPanel("Statistical Model"),
  tabsetPanel(
    tabPanel("Data Comparison",
             fluidRow(
               column(2,
                      h4("Define comparisons"),
                      uiOutput('choice1'),
                      h6("vs"),
                      uiOutput("choice2"),
                      tags$br(),
                      actionButton("submit", "Submit")
                      ),
               column(3,
                      h4("Selected comparisons"),
                      textOutput("comparisons"),
                      h5("Comparison matrix"),
                      tableOutput("matrix"),
                      conditionalPanel(condition = "input.submit > 0",
                                       downloadButton("compar", "down-
load table of comparison"),
                                       downloadButton("model_QC",
"download table of quality control"),

downloadButton("fitted_v","download fitted linear model summary")
                      )),
               column(4,
                      h5("There are"),
                      textOutput("number"),
                      h5("significant proteins"),
                      sliderInput("signif", "Significance level", 0,
1, 0.05),
                      tableOutput("significant"),
                      downloadButton("download_signif", "Download ta-
ble of significant proteins"),
                      offset = 2
                      )
             )
             ),
    tabPanel("Verify Model Assumptions",
             fluidRow("The fitted linear model relies on two assump-
tions: that the measurement errors have a Normal distribution, and
that their variance is constant.  These assumptions can be verified by
visualising the QQPlots and the Residual Plots for the model.
"),
             fluidRow(
               uiOutput("verify")
             )
             ),
    tabPanel("Results",
```

```
            sidebarPanel(
              selectInput("typeplot", "Select plot type", c("Volcano
Plot" = "VolcanoPlot", "Heatmap"="Heatmap", "Comparison Plot"="Compar-
isonPlot")),
              numericInput("sig", "Significance level", 0.05, 0, 1 ,
0.01),
              uiOutput("WhichComp"),
              conditionalPanel(condition = "input.typeplot == 'Vol-
canoPlot' || input.typeplot == 'Heatmap'",
                               checkboxInput("FC1", "Apply fold
change cutoff for significance"),
                               conditionalPanel(condition = "in-
put.FC1 == true",
                                                numericInput("FC",
"cutoff", 1, 0, 100, 0.1)),
                               selectInput("logp", "log transforma-
tion of adjusted p-value", c("two" = "2", "ten" = "10"), selected =
"ten")),
              conditionalPanel(condition = "input.typeplot == 'Vol-
canoPlot'",
                               checkboxInput("pname", "display protein
name")),
              conditionalPanel(condition = "input.typeplot ==
'Heatmap'",
                               numericInput("nump", "Number of pro-
teins in heatmap", 100, 1, 180, 1)),
              selectInput("cluster", "cluster analysis", c("protein
dendogram" = "protein", "comparison dendogram" = "comparison", "pro-
tein and comparison dendograms" = "both"))
            ),
            mainPanel(
              uiOutput("comparison_plots")
            )

  )
)
)
```

statmodel-server.R

```
# choices of groups for contrast matrix

choices <- reactive({levels(preprocess_data()$ProcessedData$GROUP_O-
RIGINAL)})
row <- reactive({rep(0,length(choices()))})
index1 <- reactive({which(choices() == input$group1)})
index2 <- reactive({which(choices() == input$group2)})

contrast <- reactiveValues()

comp_list <- reactiveValues()

# build matrix

matrix_build <- eventReactive(input$submit, {
  progress <- Progress$new(session, min=1, max=15)
```

```
  on.exit(progress$close())

  progress$set(message = 'Calculation in progress',
               detail = 'This may take a while...')

  for (i in 1:8) {
    progress$set(value = i)
    Sys.sleep(0.5)}
  comp_list$dList <- c(isolate(comp_list$dList), c(input$group1, "vs",
input$group2, ", "))
  contrast$row <- matrix(row(), nrow=1)
  contrast$row[index1()] = 1
  contrast$row[index2()] = -1
  if (is.null(contrast$matrix)) {
    contrast$matrix <- contrast$row
  } else {
    contrast$matrix <- rbind(contrast$matrix, contrast$row)
    contrast$matrix <- contrast$matrix[!duplicated(contrast$matrix),]
  }
  row.names(contrast$matrix) <- seq(1, nrow(contrast$matrix), 1)
  return(contrast$matrix)

})

# compare data

data_comparison <- reactive({groupComparison(contrast.matrix = ma-
trix_build(), data = preprocess_data())})

# download comparison data

output$compar <- downloadHandler(
  filename = function() {
  paste("comparison-", Sys.Date(), ".csv", sep="")
},
content = function(file) {
  write.csv(data_comparison()$ComparisonResult, file)
})

output$model_QC <- downloadHandler(
  filename = function() {
    paste("ModelQC-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(data_comparison()$ModelQC, file)
  })

output$fitted_v <- downloadHandler(
  filename = function() {
    paste("model_summary-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(capture.output(data_comparison()$fittedmodel), file)
  })

SignificantProteins <- reactive({with(data_comparison(),
                            ComparisonResult[ComparisonResult$adj.p-
value < input$signif, ])
```

```
})


# comparison plots

group_comparison <- function(){
  progress <- Progress$new(session, min=1, max=15)
  on.exit(progress$close())

  progress$set(message = 'Calculation in progress',
               detail = 'This may take a while...')

  for (i in 1:3) {
    progress$set(value = i)
    Sys.sleep(0.5)}

  id1 <- as.character(UUIDgenerate(FALSE))
  id_address1 <- paste("tmp/",id1, sep = "")
  path1 = paste("www/", id_address1, sep = "")

  groupComparisonPlots(data=data_comparison()$ComparisonResult,
                       type=input$typeplot,
                       sig=input$sig,
                       FCcutoff=input$FC,
                       logBase.pvalue=input$logp,
#                        ylimUp=input_ylup,
#                        ylimDown=input_yldown,
#                        xlimUp=input_xlimUp,
#                        x.axis.size=input_xax,
#                        y.axis.size=input_yax,
#                        dot.size=input_dot,
#                        text.size=input_text,
#                        legend.size=input_legend,
                       ProteinName=input$pname,
                       numProtein=input$nump,
                       clustering=input$cluster,
#                        height=input_h,
#                        width=input_w,
                       which.Comparison=input$whichComp,
                       address=path1
)
  return(id_address1)
}

# model assumptions plots

assumptions1 <- function() {
  progress <- Progress$new(session, min=1, max=15)
  on.exit(progress$close())

  progress$set(message = 'Calculation in progress',
               detail = 'This may take a while...')

  for (i in 1:3) {
    progress$set(value = i)
    Sys.sleep(0.5)}
  # normal quantile-quantile plots
  id2 <- as.character(UUIDgenerate(FALSE))
```

```
  id_address2 <- paste("tmp/",id2, sep = "")
  path2 = paste("www/", id_address2, sep = "")
  QQ <- modelBasedQCPlots(data=data_comparison(), type="QQPlots", ad-
dress = path2)
  return(id_address2)
}
assumptions2 <- function() {
  progress <- Progress$new(session, min=1, max=15)
  on.exit(progress$close())

  progress$set(message = 'Calculation in progress',
               detail = 'This may take a while...')

  for (i in 1:3) {
    progress$set(value = i)
    Sys.sleep(0.5)}
  # residual plots
  id3 <- as.character(UUIDgenerate(FALSE))
  id_address3 <- paste("tmp/",id3, sep = "")
  path3 = paste("www/", id_address3, sep = "")
  RES <- modelBasedQCPlots(data=data_comparison(), type="Residual-
Plots", address = path3)
  return(id_address3)

}

# outputs

output$choice1 <- renderUI({
  selectInput("group1", "Group 1", choices())

})

output$choice2 <- renderUI({
  selectInput("group2", "Group 2", choices())

})

output$comparisons <- renderText({
  comp_list$dList
})

output$matrix <- renderTable({
  matrix_build()
}, rownames = T)

output$significant <- renderTable({
  head(SignificantProteins())
  })

output$number <- renderText({
  nrow(SignificantProteins())
})

# downloads

output$verify <- renderUI({
  assumptions1()
```

```
  assumptions2()
  tagList(
    a("Open QQ-Plot", href=paste(assumptions1(),"QQPlot.pdf", sep =
"")),
    tags$br(),
    a("Open Residual Plot", href=paste(assumptions2(),"Residual-
Plot.pdf", sep = ""))
  )
}
)


output$download_compar <- downloadHandler(
  filename = function() {
    paste("data-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(data_comparison(), file)
  }
)

output$download_signif <- downloadHandler(
  filename = function() {
    paste("data-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(SignificantProteins(), file)
  }
)

Rownames <- reactive({
  rownames(matrix_build())
})

output$WhichComp <- renderUI ({
  selectInput("whichComp", "which comparison to plot", c("all", Row-
names()))
})

output$comparison_plots <- renderUI ({
  if (input$typeplot == "VolcanoPlot") {
    a("Open Volcano Plot", href=paste(group_comparison(),"Volcano-
Plot.pdf", sep = ""))
  }
  else if (input$typeplot == "Heatmap") {
    a("Open Heatmap", href=paste(group_comparison(),"Heatmap.pdf", sep
= ""))
  }
  else if (input$typeplot == "ComparisonPlot") {
    a("Open Comparison Plot", href=paste(group_comparison(),"Compar-
isonPlot.pdf", sep = ""))
  }
})
```

analysis-ui.R

```
side = sidebarPanel(
  h4("Download summary of protein abundance"),
  radioButtons("typequant", "Type of summarisation", c("Sample-
level summarisation" = "Sample", "Group-level summarisation" =
"Group")),
  radioButtons("format", "Save as", c("matrix" = "matrix", "ar-
ray" = "long")),
  downloadButton("download_summary", "Download"),
  tags$br(),
  tags$hr(),
  h4("Functional analysis"),
  h5("Select species"),
  uiOutput("Species"),
  h5("Select input type"),
  uiOutput("Filter"),
  h5("Select attributes to retreive (es. go_id, goslim_goa_de-
scription etc"),
  uiOutput("Attributes"),
  downloadButton("table_annot")


)

main_p = mainPanel(
  h4("Table of annotation"),
  tableOutput("annotation")

)


#################################################################

analysis = fluidPage(
  headerPanel("Functional analysis"),
  side,
  main_p
)
```

analysis-server.R

```
### functions ###

# quantification

abundance <- reactive({
  quantification(preprocess_data(),
                 type = input$typequant,
                 format = input$format)
})

# downloads
```

```
output$download_summary <- downloadHandler(
  filename = function() {
    paste("data-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(abundance(), file)
  }
)

# annotation

ensembl <- useMart("ensembl")
dbs <- listDatasets(ensembl)


output$Species <- renderUI({
  selectizeInput("species", "", dbs[2])
})

dataset.input <- reactive({
  as.character(dbs$dataset[dbs$description == input$species])
})

ensembl1 <- reactive(useDataset(dataset.input(), mart =
ensembl))

filters <- reactive(listFilters(ensembl1()))

output$Filter <- renderUI({
  selectizeInput("filter", "", filters()[2])
})

filter.input <- reactive({
  as.character(filters()$name[filters()$description == input
$filter])
})

attributes <- reactive({listAttributes(ensembl1())
})

output$Attributes <- renderUI ({
  selectizeInput("attribute_input", "", attributes(),
multiple=TRUE, options = list(placeholder="select attributes"))
})

attribute.input <- reactive({
  as.vector(input$attribute_input)
})


id <- reactive(SignificantProteins()[1])
```

```
results <- reactive({
  getBM(attributes = attribute.input(),
                 filters = filter.input(),
                 values = id(),
                 mart = ensembl1())
})

# table output

output$annotation <- renderTable({
#  merge(SignificantProteins()[,c(1,2,3,8)], results(),
by.x="Protein", by.y="ensembl_gene_id")
results()
})

# download

output$table_annot <-downloadHandler(
  filename = function() {
    paste("annotation-", Sys.Date(), ".csv", sep="")
  },
  content = function(file) {
    write.csv(results(), file)
  }
)
```

expdes-ui.R

```
expdes = fluidPage(
  shinyjs::useShinyjs(),
  headerPanel("Design Future Experiments"),
  sidebarPanel(
    h4("Choose parameter to estimate"),
    radioButtons("param", "parameters:", c("Sample Size" = "sam-
ple", "Power" = "npower")),
    sliderInput("nsample", "Number of samples", 0,1000,4,1),
    sliderInput("power", "Power", 0,1,0.8,0.1),
    sliderInput("FDR", "False Dicovery Rate", 0,1,0.05, 0.01),
    sliderInput("desirFC", "Desired Fold Change", -10, 10,
c(1.25, 1.75), 0.01)
  ),
  mainPanel(
    h4("Plot"),
    plotOutput("result_plot", hover = "plot_hover"),
    verbatimTextOutput("info"),
    downloadButton("download_future", "Download plot")
  )
)
```

expdes-server.R

```
# toggle input elements and plot

observe({
  if (input$param == "sample") {
    shinyjs::disable("nsample")
    sample_x = TRUE
    }
  else {
    sample_x <- input$nsample
    shinyjs::enable("nsample")
  }

  if (input$param == "npower") {
    shinyjs::disable("power")
    power_x = TRUE
    }
  else {
    power_x <- input$power
    shinyjs::enable("power")
  }
  FDR_x <- input$FDR
  FCR_x <- input$desirFC
  future_exp <- function(){
    exp <- designSampleSize(data=data_comparison()$fittedmodel,
                            desiredFC = input$desirFC,
                            FDR = FDR_x,
                            numSample = sample_x,
                            power= power_x)
  }

#  plot output

  output$result_plot <- renderPlot({
    designSampleSizePlots(future_exp())
  })

  #download

  output$download_future <- downloadHandler(
    filename = "future_exp.png",
    content = function(file) {
      png(file)
      designSampleSizePlots(future_exp())
      dev.off()
    })

# hover

  output$info <- renderText({
```

```
    xy_str <- function(e) {
      if(is.null(e)) return("NULL\n")
      paste0("x=", round(e$x, 1), " y=", round(e$y, 1), "\n")
    }
    paste0(
      "hover: ", xy_str(input$plot_hover)
    )

  })
})
```

help-ui.R

```
help = fluidPage(
  h4('Documentation'),
  tags$br(),
  p("This web application is based on the R package MSstats.  It
does not require any software from the user, only a web browser
to access it.  The data accepted by the application can be de-
rived from labelled SRM (Selected Reaction Monitoring) experi-
ments or label-free DDA (Data Dependent Acquisition) or DIA
(Data Independent Acquisition). " ),
  tags$br(),
  h5("Tab1: LOAD DATA"),
  p("MSstats takes input data in a tabular .csv format derived
from any spectral processing tool such as Skyline, SuperHirn,
MaxQuant, Progenesis, MultiQuant, OpenMS or OpenSWATH.  These
formats can be converted to the required 10-column format
through specific commands.  Conversion of formats Skyline, Max-
Quant, Progenesis, and Proteome Discoverer requires annotation
files."),
  p("The 10-column format of dataset is structured as
follows :"),
  tags$ul(
    tags$li("ProteinName: Protein id"),
    tags$li("PeptideSequence"),
    tags$li("PrecursorCharge"),
    tags$li("FragmentIon"),
    tags$li("ProductCharge: Feature of a protein"),
    tags$li("IsotopeLabelType: endogenous peptides (use 'L') or
labeled reference peptides (use 'H')"),
    tags$li("Condition: group type for group comparison or
timepjoint for time-lapse analysis"),
    tags$li("BioReplicate: biological replicate"),
    tags$li("Run: mass spectrometry run"),
    tags$li("Intensity: quantified signal")
  ),
  tags$br(),
  h5("Tab2 : QUALITY CONTROL"),
  p("The data is transformed, normalised and summarised for sta-
tistical modelling."),
```

```
  tags$ul(
    tags$li("Transformation: the Intensity column is transformed
in either log2 or log10"),
    tags$li("Normalisation: the available methods are Equalisa-
tion of medians (indicated when the majority of proteins don't
change between runs, less indicated for label-free DDA), use of
Global standards (defined by the user), 'quantile' (for label-
free, all intensity distributions will become equal across runs;
for label-based, all intensity distributions for references will
become equal across runs and endogenous intensities shifted ac-
cording to references), or no normalisation."),
    tags$li("Feature subset: use all feature or top 3 most sig-
nificant (highest log2 transformed intensity) or top n (custom)
most significant, or 'high quality' for most informative fea-
tures (to eliminate unexplainable variation in features - inter-
ference)"),
    tags$li("Handling missing values (random missing measure-
ments are independent of the abundance of the peptide; censored
missing measurements are due to very low abundance, which falls
below the level of detection of the instrument):"),
    tags$ul(
      tags$li("Assume all NAs as censored"),
      tags$li("Assume all intensities between 0 and 1 as cen-
sored and NAs as random missing (value of 1 can be changed)"),
      tags$li("Assume that all missing values are random missing
")
    ),
    p("With the assumption of the presence of censored values, a
cutoff for the AFT model is determined:"),
    tags$ul(
      tags$li("Minimum value for each feature across runs"),
      tags$li("Minimum value for each run across features"),
      tags$li("Smallest value between minimum value of corre-
sponding feature and minimum value of corresponding run (in this
case runs with substantial missing runs will be biased so it may
be a good option to remove all runs with more than 50% missing
values)")
    ),
    tags$li("Summary method: TMP (Turkey's median polish) or
linear (linear mixed model).  TMP can be with model-based impu-
tation of missing values (censored missing values will be imput-
ed by Accelerated Failure Time model and cutoff is defined by
the user- see above) or without imputation (in this case cen-
sored values will be substituted with a cutoff value specified
by the user - see above). With linear summary method no imputa-
tion is performed;  in the presence of censored values parame-
ters are estimated using AFT model, otherwise if all missing
values are assumed to be random, the parameters are estimated
through linear modelling.   With linear summary the variance in
intensity from features can be considered equal or heteroge-
neous. ")
  ),
  tags$br(),
```

```
  p("The preprocessed data is visualised through plots (for all
proteins or individual proteins):"),
  tags$ul(
    tags$li("QC plot: to visualise systematic biases between
runs and to view the effects of normalisation"),
    tags$li("Profile plot: to identify potential sources of
variation (individual measurements per protein and summarised
data) and show missing data"),
    tags$li("Condition plot: to visualise potential differences
in protein intensities between conditions (shown with arbitrary
confidence interval or standard deviation bars).")
  ),
  tags$br(),
  h5("Tab3 : STATISTICAL MODEL"),
  p("Based on the design of the experiment in analysis, a number
of contrasts to be tested are specified in the application.  The
contrasts will be between different conditions, different time
points or different subjects of conditions for each subject.
The application then performs comparisons between the specified
groups and returns a table with fold change and significance.
Positive values of fold change indicate up regulation, whereas
negative values indicate down regulation."),
  tags$br(),
  p("The statistical model is based on two assumptions:"),
  tags$ul(
    tags$li("normal distribution of the measurement errors (QQ
plot)"),
    tags$li("constant variance of the measurement error  (resid-
ual plot)")
  ),
  tags$br(),
  p("The results of the group comparisons may be visualised
with:"),
  tags$ul(
    tags$li("Vulcano plot: to show the comparison between groups
and the significance of the difference (adjusted p-value on the
y axis): in red are the unregulated proteins, in blue the down
regulated ones.  An horizontal line signals the false discovery
rate cutoff (proteins above the line are statistically signifi-
cant).  Each comparison has its own plot."),
    tags$li("Heat map: to view the patterns of regulation of
proteins (rows) in the comparisons (columns).  Max 180 proteins
will be shown for heat map.  "),
    tags$li("Comparison plot: to show log-fold changes in the
different comparisons for each protein.")
  ),
  tags$br(),
  h5("Tab4 : FUTURE EXPERIMENTS"),
  p("In light of the statistical analysis performed in the ap-
plication, the dataset can be viewed as a pilot study (assuming
same experimental design and same expression of 99% of proteins)
for future experiments, in order to relate the following statis-
tical parameters:"),
```

```
  tags$ul(
    tags$li("Sample size: number of biological replicates per
sample, number of peptides per protein, number of transition per
peptide"),
    tags$li("Power: average across all proteins"),
    tags$li("Fold change: minimal fold change to detect or
range"),
    tags$li("False discovery rate")
  ),
  p("All parameters must be specified but one."),
  tags$br(),
  p("The results of the estimations are visualised with
plots."),
  tags$br(),
  h5("Tab5 : FUNCTIONAL ANALYSIS"),
  p("The functional analysis is performed with the R package
biomaRt, which accesses the Ensembl database for retrieval of GO
terms and KEGG terms."),
  p("The analysis requires the selection of the following para-
meters:"),
  tags$ul(
    tags$li("Organism: in which the analysis is performed (es.
Homo sapiens, Mus musculus, etc)"),
    tags$li("ID type: format of ID for the proteins in the
dataset (es. entrez gene, ensemble gene id, etc)"),
    tags$li("Attributes: features to retrieve from the query to
the Ensembl database (es. GO id, GO description, KEGG id, etc)")
  )



)
```