Using *R* and *Bioconductor* for Proteomics Data Analysis

Laurent Gatto*, Vlad Petyuk, Thomas Lin Pedersen, Sebastian Gibb

December 5, 2015

Abstract

This vignette shows and executes the code presented in the manuscript *Using R for proteomics data analysis*. It also aims at being a general overview for users who wish to explore the *R* environment and programming language for the analysis of proteomics data.

Keywords: proteomics, mass spectrometry, tutorial.

Contents

*lg390@cam.ac.uk, http://cpu.sysbiol.cam.ac.uk

1 Introduction

This document illustrates some existing *R* infrastructure for the analysis of proteomics data. It presents the code for the use cases taken from [1, 2]. A pre-print of [1] available on arXiv[1] and [2] is open access[2].

There are however numerous additional *R* resources distributed by the *Bioconductor*[3] and CRAN[4] repositories, as well as packages hosted on personal websites. Section 9 on page 64 tries to provide a wider picture of available packages, without going into details.

1.1 General R resources

The reader is expected to have basic *R* knowledge to find the document helpful. There are numerous *R* introductions freely available, some of which are listed below.

From the *R* project web-page:

- **An Introduction to R** is based on the former *Notes on R*, gives an introduction to the language and how to use R for doing statistical analysis and graphics. [browse HTML — download PDF]
- Several introductory tutorials in the contributed documentation section.
- The `TeachingMaterial` repository[5] contains several sets of slides and vignettes about *R* programming.

Relevant background on the *R* software and its application to computational biology in general and proteomics in particular can also be found in [1]. For details about the *Bioconductor* project, the reader is referred to [3].

1.2 Bioconductor resources

The Bioconductor offers many educational resources on its help page http://bioconductor.org/help/, in addition the package's vignettes (vignettes are a requirement for Bioconductor packages). We want to draw the attention to the Bioconductor work flows that offer a cross-package overview about a specific topic. In particular, there is now a *Mass spectrometry and proteomics data analysis*[6] work flow.

1.3 Getting help

All *R* packages come with ample documentation. Every command (function, class or method) a user is susceptible to use is documented. The documentation can be accessed by preceding the command by a `?` in the *R* console. For example, to obtain help about the `library` function, that will be used in the next section, one would type `?library`. In addition, all *Bioconductor* packages come with at least one vignette (this document is the vignette that comes with the *RforProteomics* package), a document that combines text and *R* code that is executed before the pdf is assembled. To look up all vignettes that come with a package, say *RforProteomics* and then open the vignette of interest, one uses the `vignette` function as illustrated below. More details can be found in `?vignette`.

```
## list all the vignettes in the RforProteomics package
vignette(package = "RforProteomics")
## Open the vignette called RforProteomics
vignette("RforProteomics", package = "RforProteomics")
## or just
vignette("RforProteomics")
```

[1] http://arxiv.org/abs/1305.6559
[2] http://onlinelibrary.wiley.com/doi/10.1002/pmic.201400392/full
[3] http://www.bioconductor.org
[4] http://cran.r-project.org/web/packages/
[5] https://github.com/lgatto/TeachingMaterial
[6] http://bioconductor.org/help/workflows/proteomics/

R has several mailing lists[7]. The most relevant here being the main `R-help` list, *for discussion about problem and solutions using R*, ideal for general *R* content and is not suitable for bioinformatics or proteomics questions. *Bioconductor* also offers several resources dedicated to bioinformatics matters and *Bioconductor* packages, in particular the main *Bioconductor* support forum[8] for Bioconductor-related queries. A dedicated *RforProteomics* Google group[9] also welcomes questions/comments/annoucements related to *R* and mass-spectrometry/proteomics, although the Bioconductor forum is the preferred channel.

It is advised to read and comply to the posting guides (here and here) to maximise the chances to obtain good responses. It is important to specify the software versions using the `sessionInfo()` functions (see an example output at the end of this document, on page 68). It the question involves some code, make sure to isolate the relevant portion and report it with your question, trying to make your code/example reproducible[10].

1.4 Installation

The package should be installed using as described below:

```
## only first time you install Bioconductor packages
source("http://www.bioconductor.org/biocLite.R")
## else
library("BiocInstaller")
biocLite("RforProteomics")
```

To install all dependencies and reproduce the code in the vignette, replace the last line in the code chunk above with:)

```
biocLite("RforProteomics", dependencies = TRUE)
```

Finally, the package can be loaded with

```
library("RforProteomics")
```

```
##
## This is the 'RforProteomics' version 1.9.2.
##
##  To get started, visit
##    http://lgatto.github.com/RforProteomics/
##
##  or, in R, open package vignettes by typing
##    RforProteomics() # R/Bioc for proteomics overview
##    RProtVis()       # R/Bioc for proteomics visualisation
##
##  For a full list of available documents:
##    vignette(package='RforProteomics')
```

See also the *RforProteomics* web page[11] for more information on installation.

1.5 External dependencies

Some packages used in the document depend on external libraries that need to be installed prior to the *R* packages:

mzR depends on the Common Data Format[12] (CDF) to CDF-based raw mass-spectrometry data. On Linux, the `libcdf` library is required. On Debian-based systems, for instance, one needs to install the `libnetcdf-dev` package.

[7] http://www.r-project.org/mail.html
[8] https://support.bioconductor.org/
[9] https://groups.google.com/forum/#!forum/rbioc-sig-proteomics
[10] https://github.com/hadley/devtools/wiki/Reproducibility
[11] http://lgatto.github.io/RforProteomics/
[12] http://cdf.gsfc.nasa.gov/

IPPD (and others) depend on the *XML* package which requires the `libxml2` infrastructure on Linux. On Debian-based systems, one needs to install `libxml2-dev`.

biomaRt performs on-line requests using the `curl`[13] infrastructure. On Debian-based systems, you one needs to install `libcurl-dev` or `libcurl4-openssl-dev`.

MSGFplus Is based on the MS-GF+ java program and thus requires Java 1.7[14] in order to work.

1.6 Obtaining the code

The code in this document describes all the examples presented in [1] and can be copy, pasted and executed. It is however more convenient to have it in a separate text file for better interaction with *R* (using ESS[15] for Emacs or RStudio[16] for instance) to easily modify and explore it. This can be achieved with the `Stangle` function. One needs the Sweave source of this document (a document combining the narration and the *R* code) and the `Stangle` then specifically extracts the code chunks and produces a clean *R* source file. If the package is installed, the following code chunk will create a `RforProteomics.R` file in your working directory containing all the annotated source code contained in this document.

```
## gets the vignette source
rnwfile <- system.file("doc/vigsrc/RforProteomics.Rnw",
                       package = "RforProteomics")
## produces the R file in the working directory
library("knitr")
purl(rnwfile, quiet = TRUE)
```

```
## [1] ".R"
```

Alternatively, you can obtain the `Rnw` file on the github page https://github.com/lgatto/RforProteomics/blob/master/inst/doc/vigsrc/RforProteomics.Rnw.

1.7 Prepare the working environment

The packages that we will depend on to execute the examples will be loaded in the respective sections. Here, we pre-load packages that provide general functionality used throughout the document.

```
library("RColorBrewer") ## Color palettes
library("ggplot2")  ## Convenient and nice plotting
library("reshape2") ## Flexibly reshape data
```

2 Data standards and input/output

2.1 The mzR package

2.1.1 Raw MS data

The *mzR* package [4] provides a unified interface to various mass spectrometry open formats. This code chunk, taken from the `openMSfile` documentation, illustrated how to open a connection to an raw data file. The example `mzML` data is taken from the *msdata* data package. The code below would also be applicable to an `mzXML`, `mzData` or `netCDF` file.

```
## load the required packages
library("mzR") ## the software package
```

[13] http://curl.haxx.se/
[14] https://java.com
[15] http://ess.r-project.org/
[16] http://rstudio.org/

```
library("msdata") ## the data package
## below, we extract the releavant example file
## from the local 'msdata' installation
filepath <- system.file("microtofq", package = "msdata")
file <- list.files(filepath, pattern="MM14.mzML",
                   full.names=TRUE, recursive = TRUE)
## creates a commection to the mzML file
mz <- openMSfile(file)
## demonstraction of data access
basename(fileName(mz))
```

```
## [1] "MM14.mzML"
```

```
isInitialized(mz)
```

```
## [1] TRUE
```

```
runInfo(mz)
```

```
## $scanCount
## [1] 112
##
## $lowMz
## [1] 0
##
## $highMz
## [1] 0
##
## $dStartTime
## [1] 270.334
##
## $dEndTime
## [1] 307.678
##
## $msLevels
## [1] 1
```

```
instrumentInfo(mz)
```

```
## $manufacturer
## [1] "Unknown"
##
## $model
## [1] "instrument model"
##
## $ionisation
## [1] "electrospray ionization"
##
## $analyzer
## [1] "mass analyzer type"
##
## $detector
## [1] "detector type"
```

```
## once finished, it is good to explicitely
## close the connection
close(mz)
```

mzR is used by other packages, like *MSnbase* [5], *TargetSearch* [6] and *xcms* [7, 8, 9], that provide a higher level abstraction to the data.

2.1.2 Identification data

The *mzR* package also provides very fast access to `mzIdentML` data by leveraging proteowizard's `C++` parser.

```
file <- system.file("mzid", "Tandem.mzid.gz", package="msdata")
mzid <- openIDfile(file)
mzid
```

```
## Identification file handle.
## Filename:  Tandem.mzid.gz
## Number of psms:  171
```

Once and `mzRident` identification file handle has been established, various data and metadata can be extracted, as illustrated below.

```
softwareInfo(mzid)
```

```
## [1] "xtandem x! tandem CYCLONE (2010.06.01.5) "
## [2] "ProteoWizard MzIdentML 3.0.6239 ProteoWizard"
```

```
enzymes(mzid)
```

```
##      name nTermGain cTermGain minDistance missedCleavages
## 1 Trypsin         H        OH           0               1
```

```
names(psms(mzid))
```

```
##  [1] "spectrumID"               "chargeState"
##  [3] "rank"                     "passThreshold"
##  [5] "experimentalMassToCharge" "calculatedMassToCharge"
##  [7] "sequence"                 "modNum"
##  [9] "isDecoy"                  "post"
## [11] "pre"                      "start"
## [13] "end"                      "DatabaseAccess"
## [15] "DBseqLength"              "DatabaseSeq"
## [17] "DatabaseDescription"      "acquisitionNum"
```

```
head(psms(mzid))[, 1:13]
```

```
##   spectrumID chargeState rank passThreshold
## 1   index=12           3    1         FALSE
## 2  index=285           3    1         FALSE
## 3   index=83           3    1         FALSE
## 4   index=21           3    1         FALSE
## 5  index=198           3    1         FALSE
## 6   index=13           2    1         FALSE
##   experimentalMassToCharge calculatedMassToCharge
## 1                903.7209               903.4032
## 2                792.3792               792.3899
## 3                792.5295               792.3899
## 4                850.0782               849.7635
## 5                527.2592               527.2849
## 6                724.8816               724.3771
##                    sequence modNum isDecoy post pre start
## 1  LCYIALDFDEEMKAAEDSSDIEK      2   FALSE    S   K   217
```

```
## 2   KDLYGNVVLSGGTTMYEGIGER      1   FALSE    L   R   292
## 3   KDLYGNVVLSGGTTMYEGIGER      1   FALSE    L   R   292
## 4 VIDENFGLVEGLMTTVHAATGTQK      1   FALSE    V   K   842
## 5          GVGGAIVLVLYDEMK      1   FALSE    R   R   297
## 6            HAVGGRYSSLLCK      1    TRUE    D   K   392
##   end
## 1 239
## 2 313
## 3 313
## 4 865
## 5 311
## 6 404
```

2.2 Handling MS^2 identification data with mzID

The *mzID* package allows to load and manipulate MS^2 data in the `mzIdentML` format. The main `mzID` function reads such a file and constructs an instance of class `mzID`.

```
library("mzID")
id <- mzID("http://psi-pi.googlecode.com/svn/trunk/examples/1_1examples/55merge_tandem.mzid")
```

```
## reading 55merge_tandem.mzid... DONE!
```

```
id
```

```
## An mzID object
##
## Software used:   X\!Tandem (version: x! tandem CYCLONE (2010.06.01.5))
##
## Rawfile:         D:/TestSpace/NeoTestMarch2011/55merge.mgf
##
## Database:        D:/Software/Databases/Neospora_3rndTryp/Neo_rndTryp_3times.fasta.pro
##
## Number of scans: 169
## Number of PSM's: 170
```

Multiple files can be parsed in one go, possibly in parallel if the environment supports it. When this is done an mzIDCollection object is returned:

```
ids <- mzID(c(
    "http://psi-pi.googlecode.com/svn/trunk/examples/1_1examples/55merge_tandem.mzid",
    "http://psi-pi.googlecode.com/svn/trunk/examples/1_1examples/55merge_omssa.mzid"))
ids
```

```
## An mzIDCollection object containing 2 samples
```

Peptides, scans, parameters, ... can be extracted with the respective `peptides`, `scans`, `parameters`, ... functions. The `mzID` object can also be converted into a `data.frame` using the `flatten` function.

```
fid <- flatten(id)
names(fid)
```

```
##  [1] "spectrumid"               "spectrum title"
##  [3] "acquisitionnum"           "passthreshold"
##  [5] "rank"                     "calculatedmasstocharge"
##  [7] "experimentalmasstocharge" "chargestate"
##  [9] "x\\!tandem:expect"        "x\\!tandem:hyperscore"
```

```
## [11] "isdecoy"                  "post"
## [13] "pre"                      "end"
## [15] "start"                    "accession"
## [17] "length"                   "sequence"
## [19] "pepseq"                   "modified"
## [21] "modification"             "idFile"
## [23] "spectrumFile"             "databaseFile"
```

```
dim(fid)
```

```
## [1] 171  24
```

3 Raw data abstraction with `MSnExp` objects

MSnbase [5] provides base functions and classes for MS-based proteomics that allow facile data and meta-data processing, manipulation and plotting (see for instance figure 1 on page 11).

```
library("MSnbase")
## uses a simple dummy test included in the package
mzXML <- dir(system.file(package="MSnbase",dir="extdata"),
             full.name=TRUE,
             pattern="mzXML$")
basename(mzXML)
```

```
## [1] "dummyiTRAQ.mzXML"
```

```
## reads the raw data into and MSnExp instance
raw <- readMSData(mzXML, verbose = FALSE)
raw
```

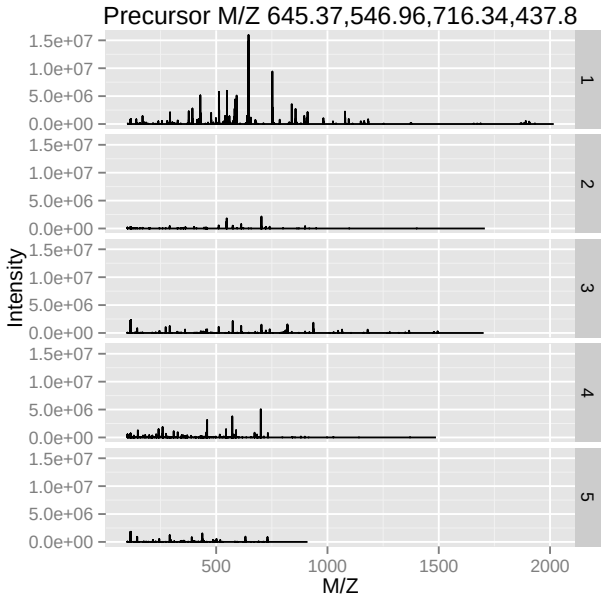
```
## Object of class "MSnExp"
##  Object size in memory: 0.2 Mb
## - - - Spectra data - - -
##  MS level(s): 2
##  Number of MS1 acquisitions: 1
##  Number of MSn scans: 5
##  Number of precursor ions: 5
##  4 unique MZs
##  Precursor MZ's: 437.8 - 716.34
##  MSn M/Z range: 100 2016.66
##  MSn retention times: 25:1 - 25:2 minutes
## - - - Processing information - - -
## Data loaded: Sat Dec  5 02:06:06 2015
##  MSnbase version: 1.19.4
## - - - Meta data  - - -
## phenoData
##   rowNames: 1
##   varLabels: sampleNames
##   varMetadata: labelDescription
## Loaded from:
##   dummyiTRAQ.mzXML
## protocolData: none
## featureData
##   featureNames: X1.1 X2.1 ... X5.1 (5 total)
```

```
##   fvarLabels: spectrum
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
```

```
## Extract a single spectrum
raw[[3]]
```

```
## Object of class "Spectrum2"
##  Precursor: 645.3741
##  Retention time: 25:2
##  Charge: 2
##  MSn level: 2
##  Peaks count: 2125
##  Total ion count: 150838188
```

```
plot(raw, full=TRUE)
```




```
plot(raw[[3]], full=TRUE, reporters=iTRAQ4)
```

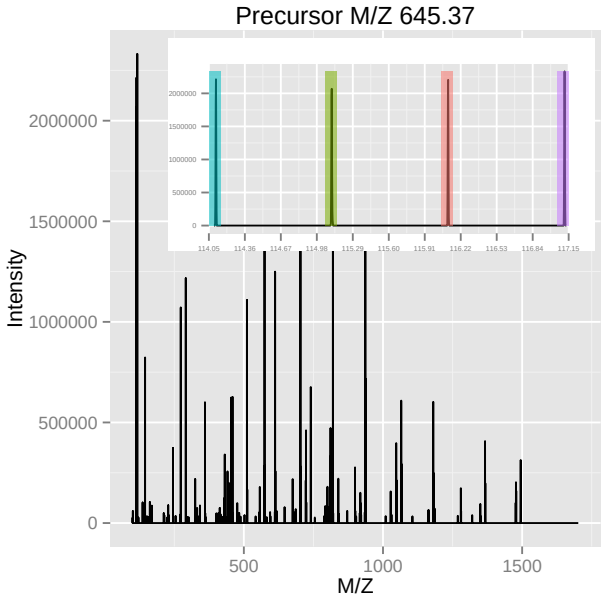


Figure 1: The `plot` method can be used on experiments, i.e. spectrum collections (top), or individual spectra (bottom).

3.1 `mgf` read/write support

Read and write support for data in the `mgf`[17] and `mzTab`[18] formats are available via the `readMgfData`/`writeMgfData` and `readMzTabData`/`writeMzTabData` functions, respectively. An example for the latter is shown in the next section.

4 Quantitative proteomics

As an running example throughout this document, we will use a TMT 6-plex data set, `PXD000001` to illustrate quantitative data processing. The code chunk below first downloads this data file from the ProteomeXchange server using the *rpx* package.

4.1 The `mzTab` format

The first code chunk downloads the `mzTab` data from the ProteomeXchange repository [10].

```
## Experiment information
library("rpx")
px1 <- PXDataset("PXD000001")
px1
```

```
## Object of class "PXDataset"
##  Id: PXD000001 with 10 files
##  [1] 'F063721.dat' ... [10] 'erwinia_carotovora.fasta'
##  Use 'pxfiles(.)' to see all files.
```

```
pxfiles(px1)
```

```
##  [1] "F063721.dat"
##  [2] "F063721.dat-mztab.txt"
##  [3] "PRIDE_Exp_Complete_Ac_22134.xml.gz"
##  [4] "PRIDE_Exp_mzData_Ac_22134.xml.gz"
##  [5] "PXD000001_mztab.txt"
##  [6] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzML"
##  [7] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzXML"
##  [8] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML"
##  [9] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.raw"
## [10] "erwinia_carotovora.fasta"
```

```
## Downloading the mzTab data
mztab <- pxget(px1, "PXD000001_mztab.txt")
```

```
## Downloading 1 file
## PXD000001_mztab.txt already present.
```

```
mztab
```

```
## [1] "PXD000001_mztab.txt"
```

The code below loads the `mzTab` file into *R* and generates an `MSnSet` instance[19], removes missing values and calculates protein intensities by summing the peptide quantitation data. Figure 2 illustrates the intensities for 5 proteins.

[17] http://www.matrixscience.com/help/data_file_help.html#GEN
[18] https://code.google.com/p/mztab/
[19] Here, we specify `mzTab` format version 0.9. Recent files have been generated according to the latest specifications, version 1.0, and the `version` does not need to be specified explicitly.

```
## Load mzTab peptide data
qnt <- readMzTabData(mztab, what = "PEP", version = "0.9")
```

```
## Version 0.9 is deprecated.  Please see '?readMzTabData' and '?MzTab' for details.
## Detected a metadata section
## Detected a peptide section
```

```
## Warning in `mode<-`(`*tmp*`, value = "numeric"):  NAs introduced by coercion
```

```
sampleNames(qnt) <- reporterNames(TMT6)
head(exprs(qnt))
```

```
##   TMT6.126 TMT6.127 TMT6.128 TMT6.129 TMT6.130 TMT6.131
## 1       NA       NA       NA       NA       NA       NA
## 2 10630132 11238708 12424917 10997763  9928972 10398534
## 3       NA       NA       NA       NA       NA       NA
## 4       NA       NA       NA       NA       NA       NA
## 5 11105690 12403253 13160903 12229367 11061660 10131218
## 6  1183431  1322371  1599088  1243715  1306602  1159064
```

```
## remove missing values
qnt <- filterNA(qnt)
processingData(qnt)
```

```
## - - - Processing information - - -
## mzTab read: Sat Dec  5 02:06:13 2015
## Subset [2351,6][1504,6] Sat Dec  5 02:06:14 2015
## Removed features with more than 0 NAs: Sat Dec  5 02:06:14 2015
## Dropped featureData's levels Sat Dec  5 02:06:14 2015
##  MSnbase version: 1.19.4
```

```
## combine into proteins
## - using the 'accession' feature meta data
## - sum the peptide intensities
protqnt <- combineFeatures(qnt,
                           groupBy = fData(qnt)$accession,
                           fun = sum)
```

```
## Combined 1504 features into 399 using user-defined function
```

```
qntS <- normalise(qnt, "sum")
qntV <- normalise(qntS, "vsn")
qntV2 <- normalise(qnt, "vsn")

acc <- c("P00489", "P00924",
         "P02769", "P62894",
         "ECA")

idx <- sapply(acc, grep, fData(qnt)$accession)
idx2 <- sapply(idx, head, 3)
small <- qntS[unlist(idx2), ]

idx3 <- sapply(idx, head, 10)
medium <- qntV[unlist(idx3), ]

m <- exprs(medium)
colnames(m) <- c("126", "127", "128",
                 "129", "130", "131")
```

```
cls <- brewer.pal(5, "Set1")
matplot(t(tail(exprs(protqnt), n = 5)), type = "b",
        lty = 1, col = cls,
        ylab = "Protein intensity (summed peptides)",
        xlab = "TMT reporters")
legend("topright", tail(featureNames(protqnt), n=5),
       lty = 1, bty = "n", cex = .8, col = cls)
```

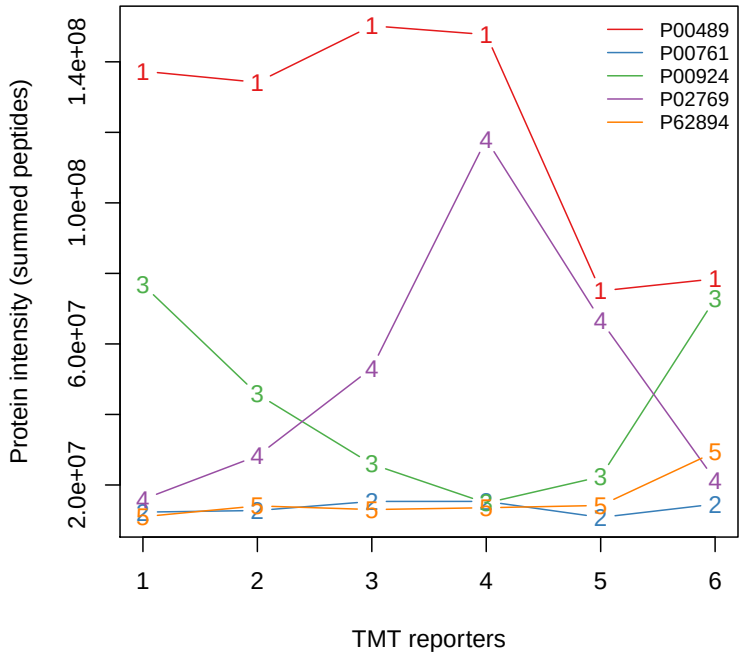


Figure 2: Protein quantitation data.

```
rownames(m) <- fData(medium)$accession
rownames(m)[grep("CYC", rownames(m))] <- "CYT"
rownames(m)[grep("ENO", rownames(m))] <- "ENO"
rownames(m)[grep("ALB", rownames(m))] <- "BSA"
rownames(m)[grep("PYGM", rownames(m))] <- "PHO"
rownames(m)[grep("ECA", rownames(m))] <- "Background"

cls <- c(brewer.pal(length(unique(rownames(m)))-1, "Set1"),
         "grey")
names(cls) <- unique(rownames(m))
wbcol <- colorRampPalette(c("white", "darkblue"))(256)
```

4.2 Third-party data

It is possible to import any arbitrary text-based spreadsheet as *MSnSet* object using either `readMSnSet` or `readMSnSet2`. The former takes three spreadsheets as input (for the expression data and the feature and sample meta-data). The latter

```
heatmap(m, col = wbcol, RowSideColors=cls[rownames(m)])
```

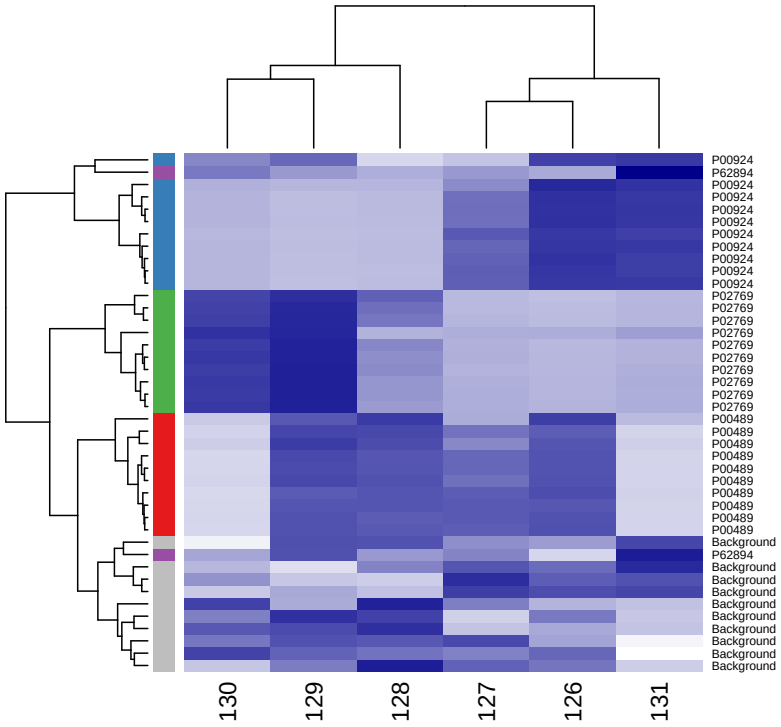


Figure 3: A heatmap.

uses a single spreadsheet and a vector of expression columns to populate the assay data and the feature meta-data. Detailed examples are provided in the `MSnbase-io` vignette, that can be consulted from *R* with `vignette("MSnbase-io")` or online[20].

[20] http://bioconductor.org/packages/release/bioc/vignettes/MSnbase/inst/doc/MSnbase-io.pdf

```
dfr <- data.frame(exprs(small),
                  Protein = as.character(fData(small)$accession),
                  Feature = featureNames(small),
                  stringsAsFactors = FALSE)
colnames(dfr) <- c("126", "127", "128", "129", "130", "131",
                   "Protein", "Feature")
dfr$Protein[dfr$Protein == "sp|P00924|ENO1_YEAST"] <- "ENO"
dfr$Protein[dfr$Protein == "sp|P62894|CYC_BOVIN"]  <- "CYT"
dfr$Protein[dfr$Protein == "sp|P02769|ALBU_BOVIN"] <- "BSA"
dfr$Protein[dfr$Protein == "sp|P00489|PYGM_RABIT"] <- "PHO"
dfr$Protein[grep("ECA", dfr$Protein)] <- "Background"
dfr2 <- melt(dfr)
## Using Protein, Feature as id variables
ggplot(aes(x = variable, y = value, colour = Protein),
       data = dfr2) +
  geom_point() +
  geom_line(aes(group=as.factor(Feature)), alpha = 0.5) +
  facet_grid(. ~ Protein) + theme(legend.position="none") +
  labs(x = "Reporters", y = "Normalised intensity")
```

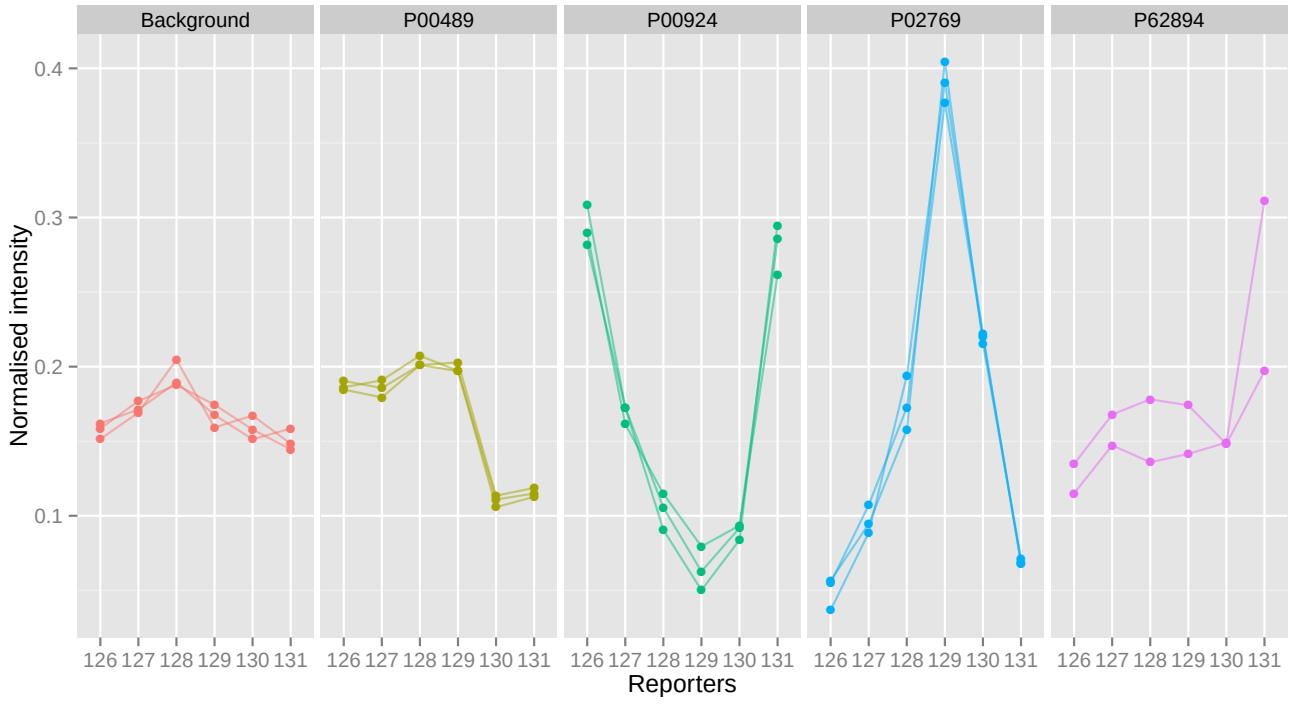


Figure 4: Spikes plot using *ggplot2*.

4.3 Working with raw data

We reuse our dedicated px1 ProteomeXchange data object to download the raw data (in mzXML format) and load it with the readMSData from the *MSnbase* package that produces a raw data experiment object of class MSnExp. The raw data is then quantified using the quantify method specifying the TMT 6-plex isobaric tags and a 7^{th} peak of interest corresponding to the un-dissociated reporter tag peaks (see the MSnbase-demo vignette in *MSnbase* for details).

```
mzxml <- pxget(px1, "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML")
```

```
## Downloading 1 file
## TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML already present.
```

```
rawms <- readMSData(mzxml, centroided = TRUE, verbose = FALSE)
qntms <- quantify(rawms, reporters = TMT7, method = "max")
```

```
## Using default parallel backend:  MulticoreParam
## Original MSnExp and new MSnSet have different number of samples in phenoData.  Dropping original.
## Creating 'MSnSet' object
```

```
qntms
```

```
## MSnSet (storageMode: lockedEnvironment)
## assayData: 6103 features, 7 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: TMT7.126 TMT7.127 ... TMT7.230 (7
##     total)
##   varLabels: mz reporters
##   varMetadata: labelDescription
## featureData
##   featureNames: X1000.1 X100.1 ... X999.1 (6103
##     total)
##   fvarLabels: spectrum file ... collision.energy (12
##     total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: No annotation
## - - - Processing information - - -
## Data loaded: Fri Dec  4 22:56:56 2015
## TMT7 quantification by max: Fri Dec  4 23:01:22 2015
##  MSnbase version: 1.19.4
```

Identification data in the mzIdentML format can be added to MSnExp or MSnSet instances with the addIdentificationData function. See the function documentation for examples.

```
d <- data.frame(Signal = rowSums(exprs(qntms)[, 1:6]),
                Incomplete = exprs(qntms)[, 7])
d <- log(d)
cls <- rep("#00000050", nrow(qnt))
pch <- rep(1, nrow(qnt))
cls[grep("P02769", fData(qnt)$accession)] <- "gold4" ## BSA
cls[grep("P00924", fData(qnt)$accession)] <- "dodgerblue" ## ENO
cls[grep("P62894", fData(qnt)$accession)] <- "springgreen4" ## CYT
cls[grep("P00489", fData(qnt)$accession)] <- "darkorchid2" ## PHO
pch[grep("P02769", fData(qnt)$accession)] <- 19
pch[grep("P00924", fData(qnt)$accession)] <- 19
pch[grep("P62894", fData(qnt)$accession)] <- 19
```

```
pch[grep("P00489", fData(qnt)$accession)] <- 19
```

```
mzp <- plotMzDelta(rawms, reporters = TMT6, verbose = FALSE) + ggtitle("")
```

```
mzp
```

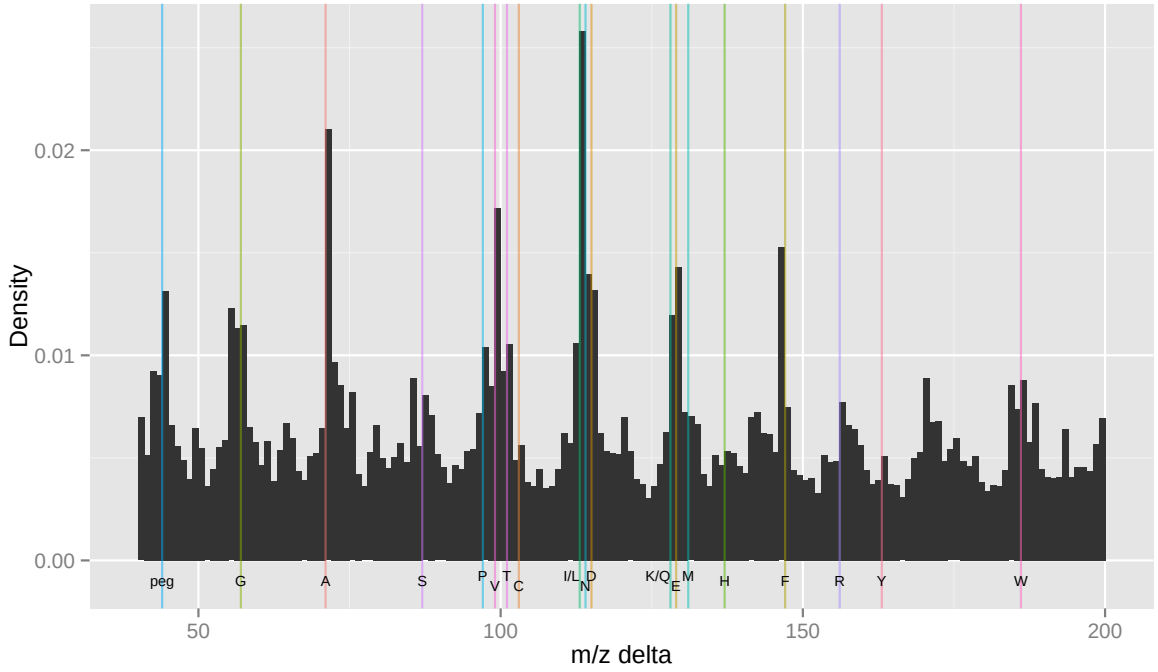


Figure 5: A m/z delta plot.

```
plot(Signal ~ Incomplete, data = d,
     xlab = expression(Incomplete~dissociation),
     ylab = expression(Sum~of~reporters~intensities),
     pch = 19,
     col = "#4582B380")
grid()
abline(0, 1, lty = "dotted")
abline(lm(Signal ~ Incomplete, data = d), col = "darkblue")
```

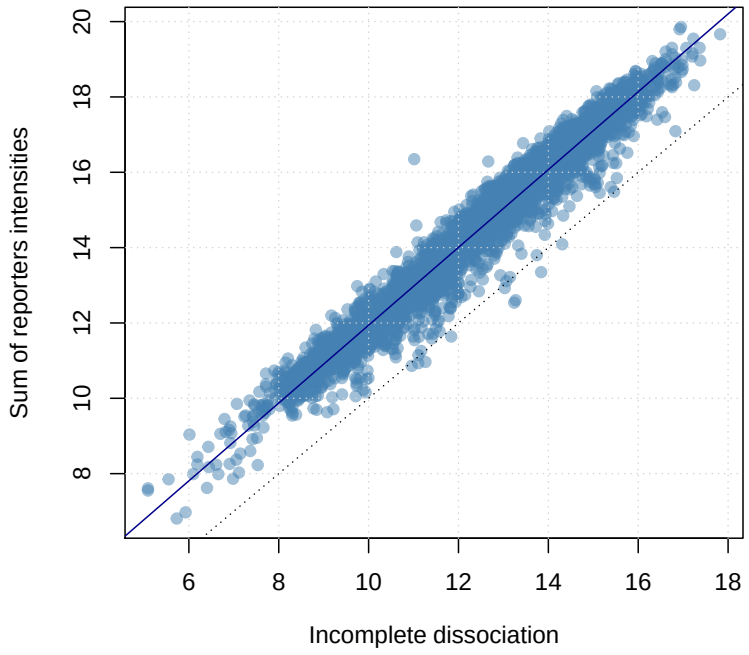


Figure 6: Incomplete dissociation.

```
MAplot(qnt[, c(4, 2)], cex = .9, col = cls, pch = pch, show.statistics = FALSE)
```

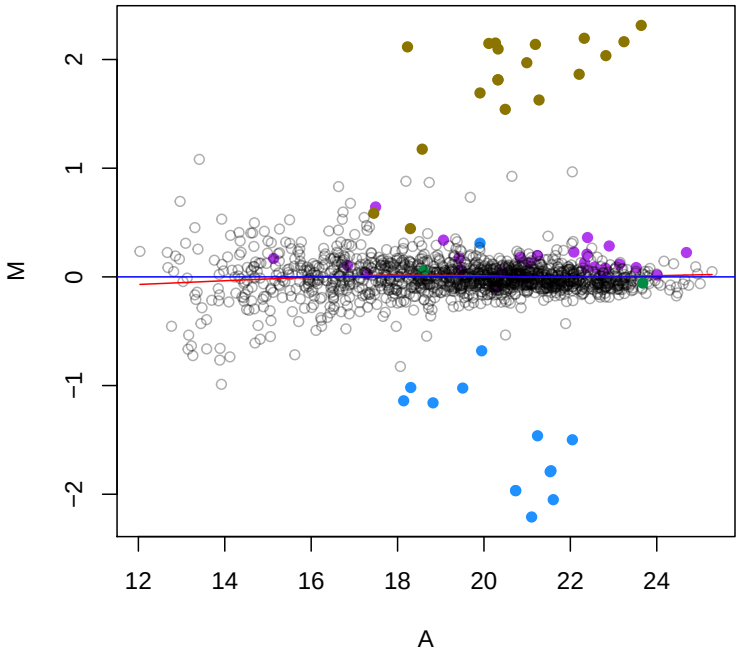


Figure 7: MAplot on an `MSnSet` instance.

4.4 The MALDIquant package

This section illustrates some of *MALDIquant*'s data processing capabilities [11]. The code is taken from the `processing-peaks.R` script downloaded from the package homepage[21].

Loading the data

```
## load packages
library("MALDIquant")
library("MALDIquantForeign")
## getting test data
datapath <-
  file.path(system.file("Examples",
                        package = "readBrukerFlexData"),
            "2010_05_19_Gibb_C8_A1")
dir(datapath)
```

```
## [1] "0_A1" "0_A2"
```

```
sA1 <- importBrukerFlex(datapath, verbose=FALSE)
# in the following we use only the first spectrum
s <- sA1[[1]]

summary(mass(s))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   999.9  2373.0  4331.0  4721.0  6874.0 10000.0
```

```
summary(intensity(s))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       4     180    1562    2841    4656   32590
```

```
head(as.matrix(s))
```

```
##           mass intensity
## [1,]  999.9388     11278
## [2,] 1000.1316     11350
## [3,] 1000.3244     10879
## [4,] 1000.5173     10684
## [5,] 1000.7101     10740
## [6,] 1000.9030     10947
```

Preprocessing

```
## sqrt transform (for variance stabilization)
s2 <- transformIntensity(s, method="sqrt")
s2
```

```
## S4 class type            : MassSpectrum
## Number of m/z values     : 22431
## Range of m/z values      : 999.939 - 10001.925
## Range of intensity values: 2e+00 - 1.805e+02
## Memory usage             : 359.875 KiB
## Name                     : 2010_05_19_Gibb_C8_A1.A1
```

[21] http://strimmerlab.org/software/maldiquant/

```
plot(s)
```

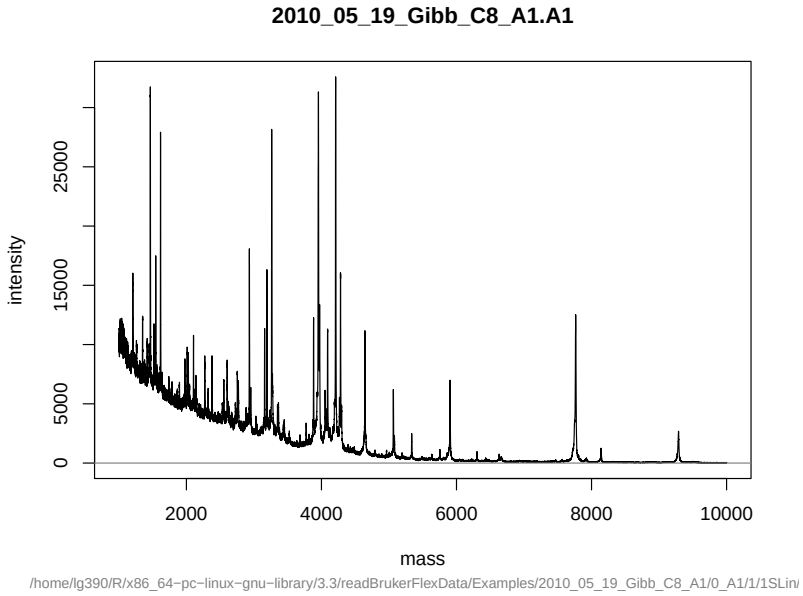


Figure 8: Spectrum plotting in *MALDIquant*.

```
## File                    : /home/lg390/R/x86_64-pc-linux-gnu-library/3.3/readBrukerFlexData/Examples/20
```

```
## smoothing - 5 point moving average
s3 <- smoothIntensity(s2, method="MovingAverage", halfWindowSize=2)
s3
```

```
## S4 class type           : MassSpectrum
## Number of m/z values    : 22431
## Range of m/z values     : 999.939 - 10001.925
## Range of intensity values: 3.606e+00 - 1.792e+02
## Memory usage            : 359.875 KiB
## Name                    : 2010_05_19_Gibb_C8_A1.A1
## File                    : /home/lg390/R/x86_64-pc-linux-gnu-library/3.3/readBrukerFlexData/Examples/20
```

```
## baseline subtraction
s4 <- removeBaseline(s3, method="SNIP")
s4
```

```
## S4 class type           : MassSpectrum
## Number of m/z values    : 22431
## Range of m/z values     : 999.939 - 10001.925
## Range of intensity values: 0e+00 - 1.404e+02
## Memory usage            : 359.875 KiB
## Name                    : 2010_05_19_Gibb_C8_A1.A1
## File                    : /home/lg390/R/x86_64-pc-linux-gnu-library/3.3/readBrukerFlexData/Examples/20
```

Peak picking

```
## peak picking
p <- detectPeaks(s4)
```

```
length(p) # 181
```

```
## [1] 186
```

```
peak.data <- as.matrix(p) # extract peak information
```

```
par(mfrow=c(2,3))
xl <- range(mass(s))
# use same xlim on all plots for better comparison
plot(s, sub="", main="1: raw", xlim=xl)
plot(s2, sub="", main="2: variance stabilisation", xlim=xl)
plot(s3, sub="", main="3: smoothing", xlim=xl)
plot(s4, sub="", main="4: base line correction", xlim=xl)
plot(s4, sub="", main="5: peak detection", xlim=xl)
points(p)
top20 <- intensity(p) %in% sort(intensity(p), decreasing=TRUE)[1:20]
labelPeaks(p, index=top20, underline=TRUE)
plot(p, sub="", main="6: peak plot", xlim=xl)
labelPeaks(p, index=top20, underline=TRUE)
```

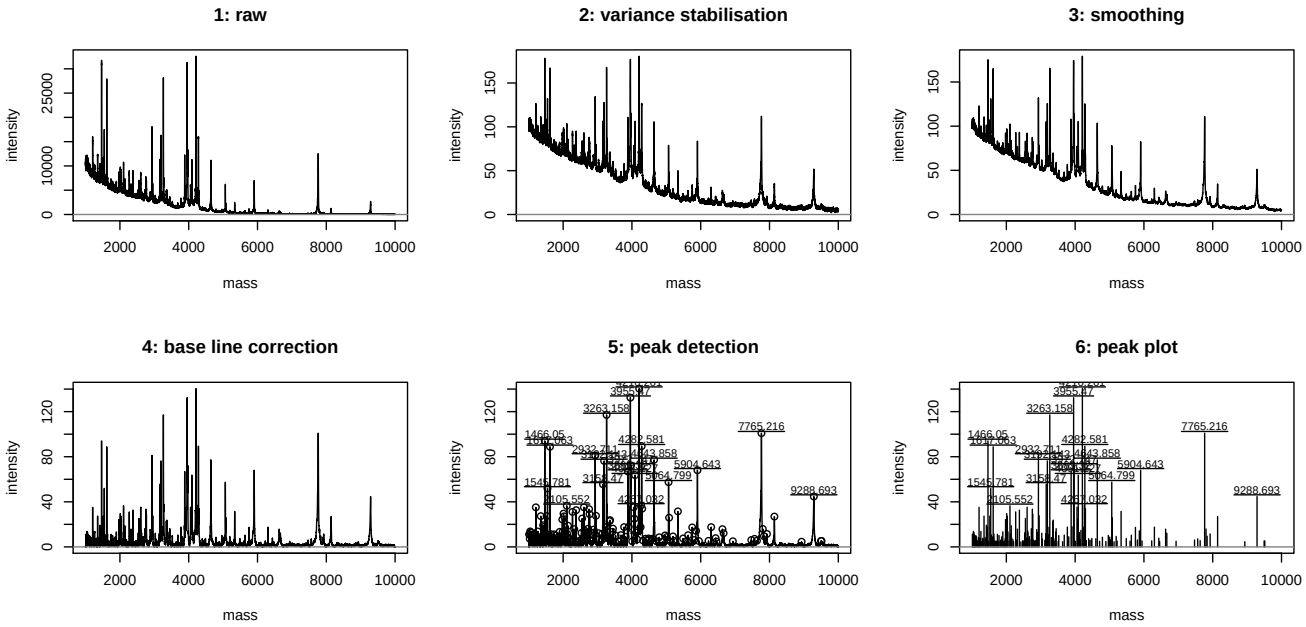


Figure 9: Spectrum plotting in *MALDIquant*.

4.5 Working with peptide sequences

```
library(IPPD)
library(BRAIN)
atoms <- getAtomsFromSeq("SIVPSGASTGVHEALEMR")
unlist(atoms)
```

```
##   C   H   N   O   S
##  77 129  23  27   1
```

```
library(Rdisop)
pepmol <- getMolecule(paste0(names(atoms),
                             unlist(atoms),
                             collapse = ""))
pepmol
```

```
## $formula
## [1] "C77H129N23O27S"
##
## $score
## [1] 1
##
## $exactmass
## [1] 1839.915
##
## $charge
## [1] 0
##
## $parity
## [1] "e"
##
## $valid
## [1] "Valid"
##
## $DBE
## [1] 25
##
## $isotopes
## $isotopes[[1]]
##              [,1]         [,2]         [,3]         [,4]
## [1,] 1839.9148973 1840.9177412 1841.9196777 1.842921e+03
## [2,]    0.3427348    0.3353456    0.1960976 8.474135e-02
##              [,5]         [,6]         [,7]         [,8]
## [1,] 1.843923e+03 1.844925e+03 1.845927e+03 1.846928e+03
## [2,] 2.952833e-02 8.691735e-03 2.226358e-03 5.066488e-04
##              [,9]        [,10]
## [1,] 1.847930e+03 1.848932e+03
## [2,] 1.040196e-04 1.949686e-05
```

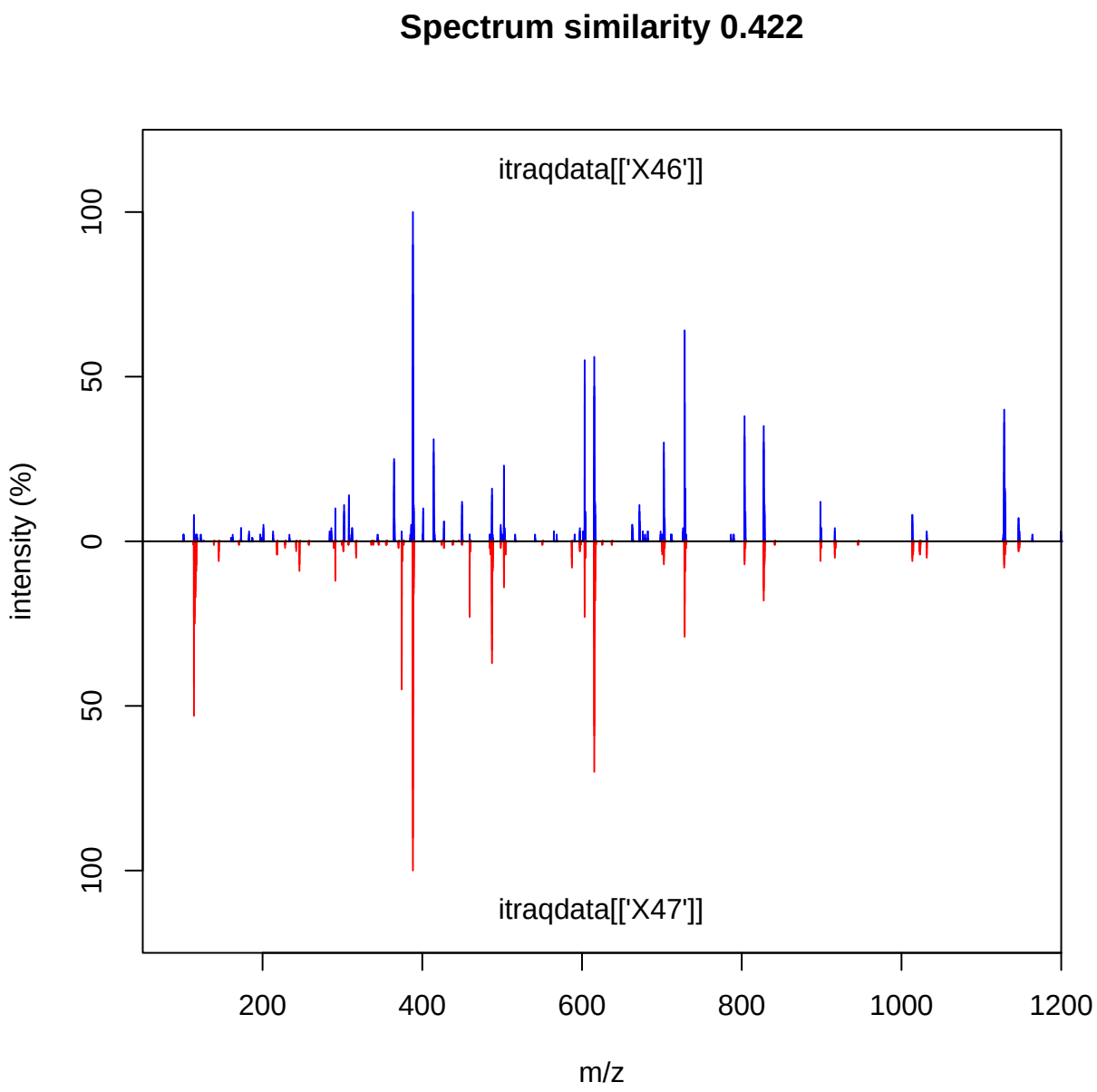
```
##
library(OrgMassSpecR)
data(itraqdata)

simplottest <-
  itraqdata[featureNames(itraqdata) %in% paste0("X", 46:47)]
sim <- SpectrumSimilarity(as(simplottest[[1]], "data.frame"),
```

```
                        as(simplottest[[2]], "data.frame"),
                        top.lab = "itraqdata[['X46']]",
                        bottom.lab = "itraqdata[['X47']]",
                        b = 25)
```

```
##           mz intensity.top intensity.bottom
## 1   114.1091             0               44
## 2   114.1109             0               53
## 3   114.1127             0               43
## 4   115.1085             0               25
## 5   364.7215            25                0
## 6   374.2082             0               39
## 7   374.2191             0               45
## 8   374.2301             0               35
## 9   388.2442             0               35
## 10  388.2558             0               75
## 11  388.2673             0              100
## 12  388.2789             0               90
## 13  388.2904            35               53
## 14  388.2904           100               53
## 15  388.2904            90               53
## 16  388.2904            53               53
## 17  388.2904            75               53
## 18  414.2582            31                0
## 19  414.2709            27                0
## 20  487.2887             0               33
## 21  487.3050             0               37
## 22  487.3213             0               28
## 23  603.3339            42                0
## 24  603.3563            55                0
## 25  603.3787            48                0
## 26  603.4011            27                0
## 27  615.3124             0               28
## 28  615.3354             0               56
## 29  615.3585             0               70
## 30  615.3816             0               59
## 31  615.4047            26               32
## 32  615.4047            44               32
## 33  615.4047            56               32
##  [ reached getOption("max.print") -- omitted 17 rows ]
```

```
title(main = paste("Spectrum similarity", round(sim, 3)))
```




```
MonoisotopicMass(formula = list(C = 2, O = 1, H=6))
```

```
## [1] 46.04186
```

```
molecule <- getMolecule("C2H5OH")
molecule$exactmass
```

```
## [1] 46.04186
```

```
## x11()
## plot(t(.pepmolfisotopes[[1]]), type = "h")

## x <- IsotopicDistribution(formula = list(C = 2, O = 1, H=6))
## t(moleculefisotopes[[1]])
```

```
## par(mfrow = c(2,1))
## plot(t(moleculeIsotopes[[1]]), type = "h")
## plot(x[, c(1,3)], type = "h")

## data(myo500)
## masses <- c(147.053, 148.056)
## intensities <- c(93, 5.8)
## molecules <- decomposeIsotopes(masses, intensities)

## experimental eno peptides
exppep <-
  as.character(fData(qnt[grep("ENO", fData(qnt)[, 2]), ])[, 1]) ## 13
minlength <- min(nchar(exppep))


if (!file.exists("P00924.fasta"))
    eno <- download.file("http://www.uniprot.org/uniprot/P00924.fasta",
                         destfile = "P00924.fasta")
eno <- paste(readLines("P00924.fasta")[-1], collapse = "")
enopep <- Digest(eno, missed = 1)
nrow(enopep) ## 103
```

```
## [1] 103
```

```
sum(nchar(enopep$peptide) >= minlength) ## 68
```

```
## [1] 0
```

```
pepcnt <- enopep[enopep[, 1] %in% exppep, ]
nrow(pepcnt) ## 13
```

```
## [1] 0
```

The following code chunks demonstrate how to use the *cleaver* package for in-silico cleavage of polypeptides, e.g. cleaving of *Gastric juice peptide 1 (P01358)* using *Trypsin*:

```
library(cleaver)
```

```
## Warning:  replacing previous import by 'IRanges::as.list.Vector' when loading 'cleaver'
```

```
cleave("LAAGKVEDSD", enzym = "trypsin")
```

```
## $LAAGKVEDSD
## [1] "LAAGK" "VEDSD"
```

Sometimes cleavage is not perfect and the enzym miss some cleavage positions:

```
## miss one cleavage position
cleave("LAAGKVEDSD", enzym = "trypsin", missedCleavages = 1)
```

```
## $LAAGKVEDSD
## [1] "LAAGKVEDSD"
```

```
## miss zero or one cleavage positions
cleave("LAAGKVEDSD", enzym = "trypsin", missedCleavages = 0:1)
```

```
## $LAAGKVEDSD
## [1] "LAAGK"      "VEDSD"      "LAAGKVEDSD"
```

Example code to generate an Texshade image to be included directly in a Latex document or *R* vignette is presented below. The *R* code generates a Texshade environment and the annotated sequence display code that is written to a

TEX file that can itself be included into a LATEX of Sweave document.

```
seq1file <- "seq1.tex"
cat("\\begin{texshade}{Figures/P00924.fasta}
     \\setsize{numbering}{footnotesize}
     \\setsize{residues}{footnotesize}
     \\residuesperline*{70}
     \\shadingmode{functional}
     \\hideconsensus
     \\vsepspace{1mm}
     \\hidenames
     \\noblockskip\n", file = seq1file)
tmp <- sapply(1:nrow(pepcnt), function(i) {
  col <- ifelse((i %% 2) == 0, "Blue", "RoyalBlue")
  cat("\\shaderegion{1}{", pepcnt$start[i], "..", pepcnt$stop[i], "}{White}{", col, "}\n",
      file = seq1file, append = TRUE)
})
cat("\\end{texshade}
    \\caption{Visualising observed peptides for the Yeast enolase protein. Peptides are shaded in blue and black.
              The last peptide is a mis-cleavage and overlaps with \\texttt{IEEELGDNAVFAGENFHHGDK}.}
    \\label{fig:seq}
  \\end{center}
\\end{figure}\n\n",
    file = seq1file, append = TRUE)
```

^{15}N incorporation

```
## 15N incorporation rates from 0, 0.1, ..., 0.9, 0.95, 1
increate <- c(seq(0, 0.9, 0.1), 0.95, 1)
inc <- lapply(increate, function(inc)
             IsotopicDistributionN("YEVQGEVFTKPQLWP", inc))
par(mfrow = c(4,3))
for (i in 1:length(inc))
  plot(inc[[i]][, c(1, 3)], xlim = c(1823, 1848), type = "h",
       main = paste0("15N incorporation at ", increate[i]*100, "%"))
```

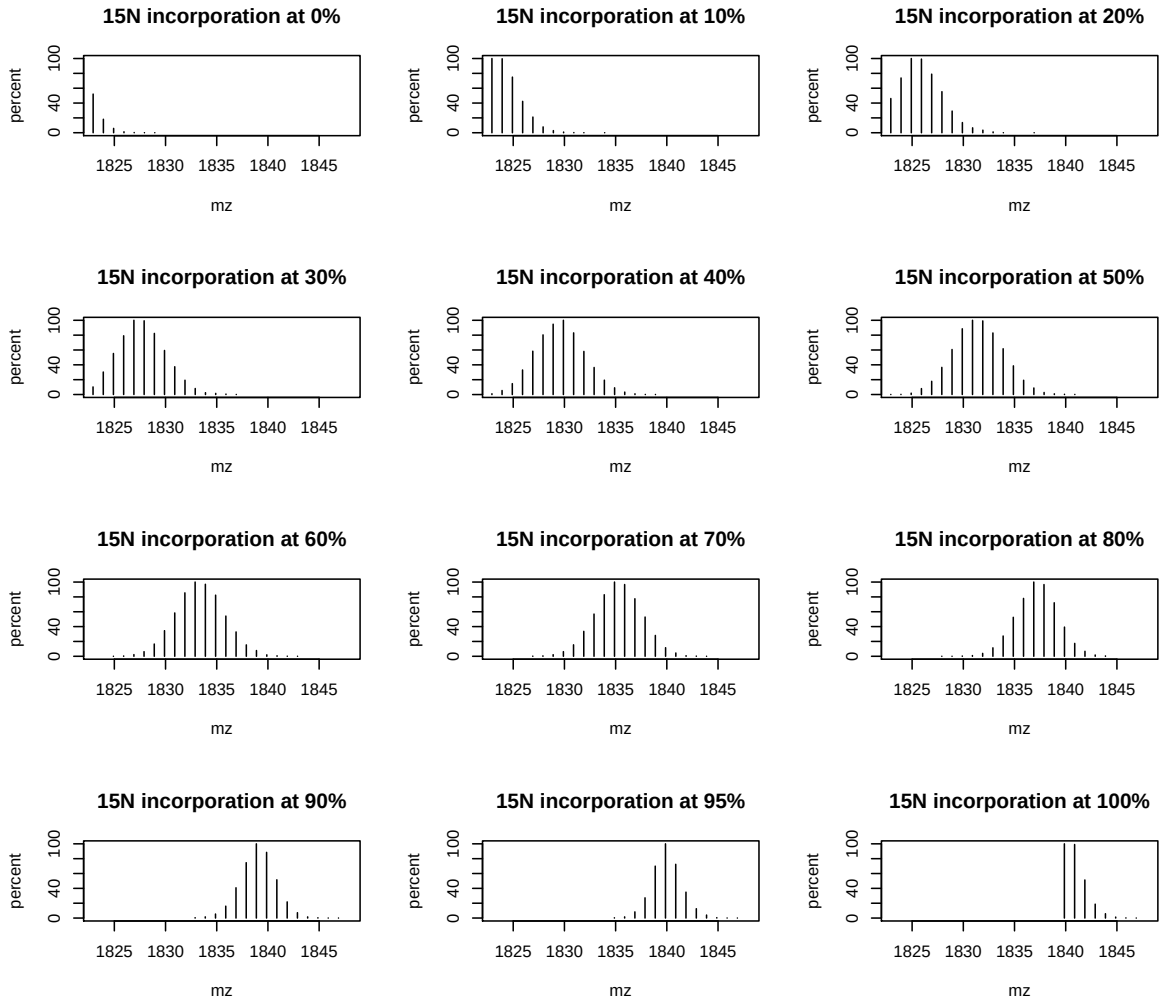


Figure 10: Isotopic envelope for the YEVQGEVFTKPQLWP peptide at different ^{15}N incorporation rates.

4.6 The isobar package

The *isobar* package [12] provides methods for the statistical analysis of isobarically tagged MS^2 experiments.

```
library(isobar)

## Prepare the PXD000001 data for isobar analysis
.ions <- exprs(qnt)
.mass <- matrix(mz(TMT6), nrow(qnt), byrow=TRUE, ncol = 6)
colnames(.ions) <- colnames(.mass) <-
  reporterTagNames(new("TMT6plexSpectra"))
rownames(.ions) <- rownames(.mass) <-
  paste(fData(qnt)$accession, fData(qnt)$sequence, sep = ".")
pgtbl <- data.frame(spectrum = rownames(.ions),
                    peptide = fData(qnt)$sequence,
                    modif = ":",
                    start.pos = 1,
                    protein = fData(qnt)$accession,
                    accession = fData(qnt)$accession)
x <- new("TMT6plexSpectra", pgtbl, .ions, .mass)
```

```
##  data.frame columns OK
## Creating ProteinGroup ...  done
```

```
featureData(x)$proteins <- as.character(fData(qnt)$accession)

x <- correctIsotopeImpurities(x) ## using identity matrix here
```

```
## LOG: isotopeImpurities.corrected:  TRUE
```

```
x <- normalize(x, per.file = FALSE)
```

```
## LOG: is.normalized:  TRUE
## LOG: normalization.multiplicative.factor channel 126:  0.8846
## LOG: normalization.multiplicative.factor channel 127:  0.9244
## LOG: normalization.multiplicative.factor channel 128:  1
## LOG: normalization.multiplicative.factor channel 129:  0.9421
## LOG: normalization.multiplicative.factor channel 130:  0.8593
## LOG: normalization.multiplicative.factor channel 131:  0.889
```

```
## spikes
spks <- c(protein.g(proteinGroup(x), "P00489"),
          protein.g(proteinGroup(x), "P00924"),
          protein.g(proteinGroup(x), "P02769"),
          protein.g(proteinGroup(x), "P62894"))

cls2 <- rep("#00000040", nrow(x))
pch2 <- rep(1, nrow(x))
cls2[grep("P02769", featureNames(x))] <- "gold4" ## BSA
cls2[grep("P00924", featureNames(x))] <- "dodgerblue" ## ENO
cls2[grep("P62894", featureNames(x))] <- "springgreen4" ## CYT
cls2[grep("P00489", featureNames(x))] <- "darkorchid2" ## PHO
pch2[grep("P02769", featureNames(x))] <- 19
pch2[grep("P00924", featureNames(x))] <- 19
pch2[grep("P62894", featureNames(x))] <- 19
pch2[grep("P00489", featureNames(x))] <- 19
```

```
nm <- NoiseModel(x)

## [1] 7.306091e-02 1.140614e+04 3.489853e+00

ib.background <- subsetIBSpectra(x, protein=spks,
                                 direction = "exclude")

## Creating ProteinGroup ...  done

nm.background <- NoiseModel(ib.background)

## [1] 0.01425222 3.49812516 0.89685036

ib.spks <- subsetIBSpectra(x, protein = spks,
                           direction="include",
                           specificity="reporter-specific")

## Creating ProteinGroup ...  done

nm.spks <- NoiseModel(ib.spks, one.to.one=FALSE, pool=TRUE)

## 4 proteins with more than 10 spectra, taking top 50.
## [1] 0.0000000001 6.1927071539 0.6721054619

ratios <- 10^estimateRatio(x, nm,
                           channel1="127", channel2="129",
                           protein = spks,
                           combine = FALSE)[, "lratio"]


res <- estimateRatio(x, nm,
                     channel1="127", channel2="129",
                     protein = unique(fData(x)$proteins),
                     combine = FALSE,
                     sign.level = 0.01)[, c(1, 2, 6, 8)]
res <- as.data.frame(res)
res$lratio <- -(res$lratio)

cls3 <- rep("#00000050", nrow(res))
pch3 <- rep(1, nrow(res))
cls3[grep("P02769", rownames(res))] <- "gold4" ## BSA
cls3[grep("P00924", rownames(res))] <- "dodgerblue" ## ENO
cls3[grep("P62894", rownames(res))] <- "springgreen4" ## CYT
cls3[grep("P00489", rownames(res))] <- "darkorchid2" ## PHO
pch3[grep("P02769", rownames(res))] <- 19
pch3[grep("P00924", rownames(res))] <- 19
pch3[grep("P62894", rownames(res))] <- 19
pch3[grep("P00489", rownames(res))] <- 19

rat.exp <- c(PHO = 2/2,
             ENO = 5/1,
             BSA = 2.5/10,
             CYT = 1/1)
```

```
maplot(x,
       noise.model = c(nm.background, nm.spks, nm),
       channel1="127", channel2="129",
       pch = 19, col = cls2,
       main = "Spectra MA plot")
abline(h = 1, lty = "dashed", col = "grey")
legend("topright",
       c("BSA", "ENO", "CYT", "PHO"),
       pch = 19, col = c("gold4", "dodgerblue",
                  "springgreen4", "darkorchid2"),
       bty = "n", cex = .7)
```

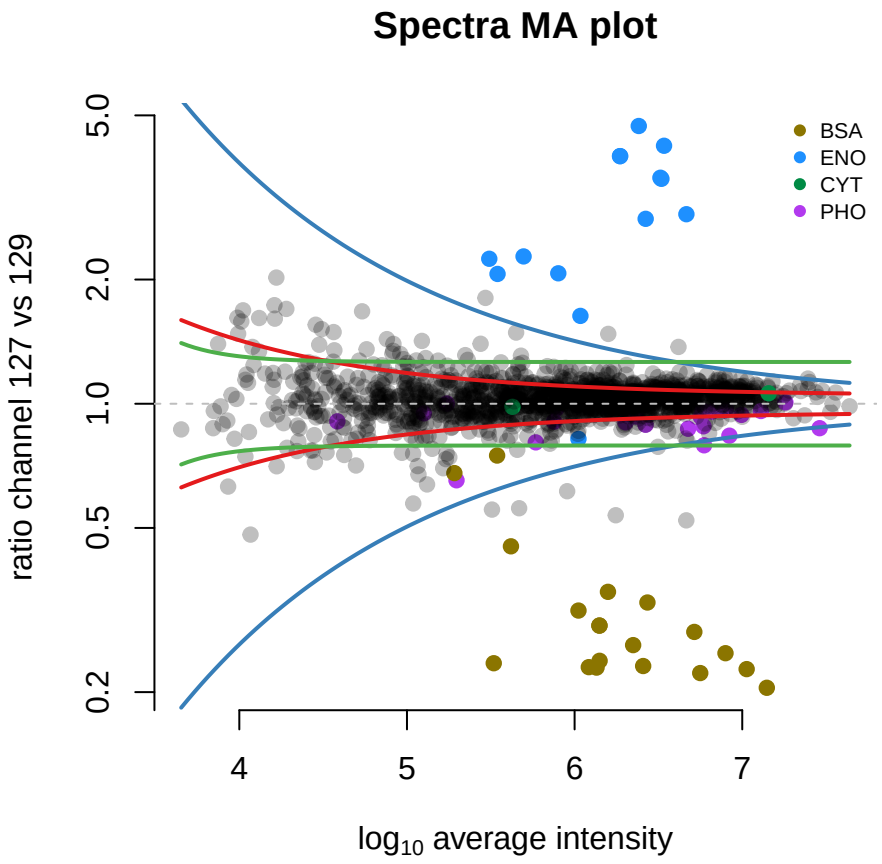


Figure 11: Result from the *isobar* pipeline.

4.7 The synapter package

The *synapter* [13] package comes with a detailed vignette that describes how to prepare the MS^E data and then process it in *R*. Several interfaces are available provided the user with maximum control, easy batch processing capabilities or a graphical user interface. The conversion into `MSnSet` instances and filter and combination thereof as well as statistical analysis are also described.

```
## open the synapter vignette
library("synapter")
synapterGuide()
```

5 MS^2 spectra identification

At the moment two packages allow the user to run peptide identifications from within R. Each of the packages interface to an external peptide database search tool and have more or less the same workflow, though their syntax differs:

1. Prepare the input data.
2. Run the search.
3. Import the search results and extract the peptides and proteins

5.1 X! Tandem

Following Bioconductor 2.12 the *rTANDEM* package provides the means to run the popular X! Tandem software [14].

Using example code/data from the *rTANDEM* vignette/package, the following is an example of a typical workflow

5.1.1 Preparation of the input data

```
library(rTANDEM)
taxonomy <- rTTaxo(taxon = "yeast",
                   format = "peptide",
                   URL = system.file(
                     "extdata/fasta/scd.fasta.pro",
                     package="rTANDEM"))
param <- rTParam()
param <- setParamValue(param,
                       'protein', 'taxon',
                       value="yeast")
param <- setParamValue(param, 'list path',
                       'taxonomy information', taxonomy)
param <- setParamValue(param,
                       'list path', 'default parameters',
                       value = system.file(
                         "extdata/default_input.xml",
                         package="rTANDEM"))
param <- setParamValue(param, 'spectrum', 'path',
                       value = system.file(
                         "extdata/test_spectra.mgf",
                         package="rTANDEM"))
param <- setParamValue(param, 'output', 'xsl path',
                       value = system.file(
                         "extdata/tandem-input-style.xsl",
```

```
                          package="rTANDEM"))
param <- setParamValue(param, 'output', 'path',
                       value = paste(getwd(),
                         "output.xml", sep="/"))
```

5.1.2 Performing the search

The analysis is run using the `tandem` function (see also the `rtandem` function), which returns the results data file path (only the file name is displayed below).

```
resultPath <- tandem(param)
```

```
## Loading spectra
##  (mgf). loaded.
## Spectra matching criteria = 242
## Starting threads . started.
## Computing models:
##  testin
##  sequences modelled = 5 ks
## Model refinement:
##  partial cleavage ..... done.
##  unanticipated cleavage ..... done.
##  modified N-terminus ..... done.
##  finishing refinement ... done.
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 40
## Unique models = 41
## Estimated false positives = 1 +/- 1
```

```
basename(resultPath)
```

```
## [1] "output.2015_12_05_02_06_20.t.xml"
```

5.1.3 Import and analyse results

```
res <- GetResultsFromXML(resultPath)
## the inferred proteins
proteins <- GetProteins(res,
                        log.expect = -1.3,
                        min.peptides = 2)
proteins[, -(4:5), with = FALSE]
```

```
##      uid expect.value   label description num.peptides
##  1:  576        -27.2 YCR012W     YCR012W            5
##  2: 1811        -14.5 YFR053C     YFR053C            3
##  3: 2301        -12.8 YGR254W     YGR254W            3
##  4:    4        -12.0 YAL005C     YAL005C            3
```

```
##  5: 3517        -12.0 YLL024C     YLL024C            3
##  6: 3328        -10.3 YKL152C     YKL152C            2
##  7: 3386        -10.1 YKL216W     YKL216W            2
##  8: 2281         -7.9 YGR234W     YGR234W            2
##  9: 2568         -7.5 YHR174W     YHR174W            2
## 10: 2044         -7.1 YGL253W     YGL253W            2
```

```
## the identified peptides for YFR053C
peptides <- GetPeptides(protein.uid = 1811,
                        results = res,
                        expect = 0.05)
peptides[, c(1:4, 9, 10:16), with = FALSE]
```

```
##     pep.id prot.uid spectrum.id spectrum.mh expect.value
## 1: 102.1.1     1811         102    942.5147      0.00660
## 2: 250.1.1     1811         250   1212.5610      0.00043
## 3:  60.1.1     1811          60    863.4933      0.00870
##    tandem.score        mh   delta peak.count
## 1:         31.9  942.5370 -0.0220         NA
## 2:         35.0 1212.5531  0.0079         NA
## 3:         21.7  863.4985 -0.0052         NA
##    missed.cleavages start.position end.position
## 1:                0            166          173
## 2:                0            437          447
## 3:                0            309          315
```

More details are provided in the vignette available with (vignette("rTANDEM")), for instance the extraction of degenerated peptides, i.e. peptides found in multiple proteins.

The *shinyTANDEM* package offers a web-based graphical interface to *rTANDEM*.

5.2 MS-GF+

With the release of Bioconductor 3.0 the *MSGFplus* package has provided an interface to MS-GF+ [15, 16]. The package vignette describe in detail the different ways an MS-GF+ analysis can be initiated and only a simple example will be given here:

5.2.1 Preparation of the input data

```
library("MSGFplus")
## Create a parameter object with a set of parameters
param <- msgfPar(database = system.file('extdata',
                                        'milk-proteins.fasta',
                                        package='MSGFplus'),
                 tolerance = '10 ppm',
                 enzyme = 'Trypsin')

## Add parameters after creation
instrument(param) <- 'QExactive'
tda(param) <- TRUE
ntt(param) <- 2

## Add expected modifications
```

```
mods(param)[[1]] <- msgfParModification('Carbamidomethyl',
                                        composition = 'C2H3N1O1',
                                        residues = 'C',
                                        type = 'fix',
                                        position = 'any')

mods(param)[[2]] <- msgfParModification(name = 'Oxidation',
                                        mass = 15.994915,
                                        residues = 'M',
                                        type = 'opt',
                                        position = 'any')

nMod(param) <- 2    # Number of allowed modifications per peptide

## Get a summary of your parameters
show(param)
```

```
## An msgfPar object
##
## Database:                   /home/lg390/R/x86_64-pc-linux-gnu-library/3.3/MSGFplus/extdata/milk-proteins
## Tolerance:                  10 ppm
## TDA:                        TRUE
## Instrument:                 3: QExactive
## Enzyme:                     1: Trypsin
## No. tolerable termini:      2
##
## Modifications:
##
## Number of modifications per peptide: 2
##
## Carbamidomethyl: C2H3N1O1, C, fix, any
## Oxidation: 15.99492, M, opt, any
```

5.2.2 Performing the search

Initiating the search is done using the `runMSGF` method. As a minimum it takes a parameter object and a list of raw data files and performs the search for each data file in sequence. More specialised operations are also possible such as running it asynchronously, but interested readers should refer to the *MSGFplus* vignette for additional information.

The first time a search is initialised the MS-GF+ code is downloaded, so be sure to have an active internet connection (only applies to the first time a search is run).

```
result <- runMSGF(param, 'path/to/a/rawfile.mzML')
```

5.2.3 Import and analyse results

By default MSGFplus imports the results automatically using *mzID*. If only one file was analysed, the return value is an mzID object; if multiple files are analysed at once the return value is an mzIDCollection object.

If `import=FALSE` the results are not imported and can be accessed at a later time using the *mzID* package (see section 2.2 on page 8).

5.2.4 Running MS-GF+ through a GUI

MSGFplus comes with a sister package, *MSGFgui*, which provide a graphic interface to setting up and running MS-GF+ through *R*. Besides facilitating MS-GF+ analyses, which is arguably just as easy from the command line, it provides an intuitive way to investigate and evaluate the resulting identification data.

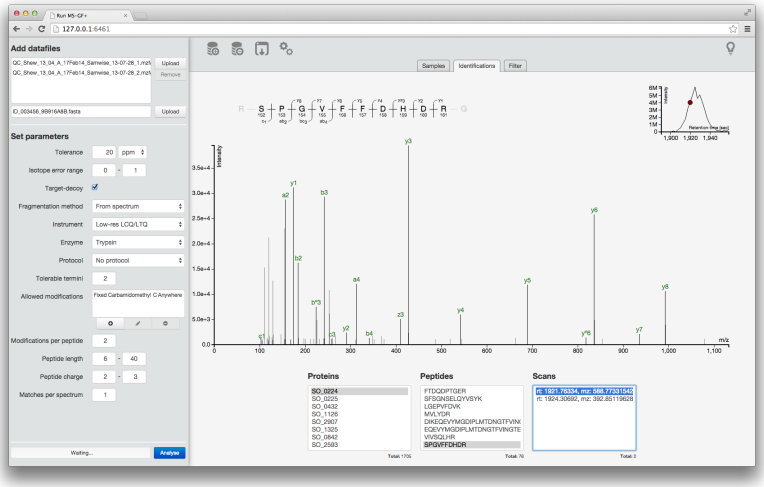


Figure 12: A screenshot of MSGFgui

Figure 12 shows an example of using *MSGFgui*. It is possible to gradually drill down in the results starting from the protein level and ending at the raw spectrum level. mzIdentML files already created with MS-GF+ (using *MSGFplus* or in other ways) can easily be imported into the gui to take advantage of the visualisation features, and results can be exported as either rds (for R), xlsx (for excel) or txt (for everything else) files.

5.3 Post-search Filtering of MS/MS IDs Using MSnID

The main purpose of MSnID package is to make sure that the peptide and protein identifications resulting from MS/MS searches are sufficiently confident for a given application. MS/MS peptide and protein identification is a process that prone to uncertainties. A typical and currently most reliable way to quantify uncertainty in the list of identify spectra, peptides or proteins relies on so-called decoy database. For bottom-up (i.e. involving protein digestion) approaches a common way to construct a decoy database is simple inversion of protein amino-acid sequences. If the spectrum matches to normal protein sequence it can be true or false match. Matches to decoy part of the database are false only (excluding the palindromes). Therefore the false discovery rate (FDR) of identifications can be estimated as ratio of hits to decoy over normal parts of the protein sequence database. There are multiple levels of identification that FDR can be estimated for. First, is at the level of peptide/protein- to-spectrum matches. Second is at the level of unique peptide sequences. Note, true peptides tend to be identified by more then one spectrum. False peptide tend to be sporadic. Therefore, after collapsing the redundant peptide identifications from multiple spectra to the level of unique peptide sequence, the FDR typically increases. The extend of FDR increase depends on the type and complexity of the sample. The same trend is true for estimating the identification FDR at the protein level. True proteins tend to be identified with multiple peptides, while false protein identifications are commonly covered only by one peptide. Therefore FDR estimate tend to be even higher for protein level compare to peptide level. The estimation of the FDR is also affected by the number of LC-MS (runs) datasets in the experiment. Again, true identifications tend to be more consistent from run to run, while false are sporadic. After collapsing the redundancy across the runs, the number of true identification reduces much stronger compare to false identifications. Therefore, the peptide and protein FDR estimates need to be re-evaluated. The main objective of the MSnID package is to provide convenience tools for handling tasks on estimation of FDR, defining and optimizing the filtering criteria and ensuring confidence in MS/MS identification data. The user can specify the criteria for filtering the data (e.g. goodness or p-value of matching of experimental and theoretical fragmentation mass spectrum,

deviation of theoretical from experimentally measured mass, presence of missed cleavages in the peptide sequence, etc), evaluate the performance of the filter judging by FDRs at spectrum, peptide and protein levels, and finally optimize the filter to achieve the maximum number of identifications while not exceeding maximally allowed FDR upper threshold.

5.3.1 Starting Project & Importing Data

To start a project one have to specify a directory. Currently the only use of the directory is for storing cached results.

```
library("MSnID")
```

```
## Warning:  replacing previous import by 'data.table::melt' when loading 'MSnID'
```

```
## Warning:  replacing previous import by 'data.table::dcast' when loading 'MSnID'
```

```
##
## Attaching package:  'MSnID'
##
## The following object is masked from 'package:isobar':
##
##    peptides
##
## The following object is masked from 'package:ProtGenerics':
##
##    peptides
```

```
msnid <- MSnID(".")
```

```
## Note, the anticipated/suggested columns in the
## peptide-to-spectrum matching results are:
## ---------------------------------------------
## accession
## calculatedMassToCharge
## chargeState
## experimentalMassToCharge
## isDecoy
## peptide
## spectrumFile
## spectrumID
```

Data can imported as `data.frame` or read from mzIdentML file.

```
PSMresults <- read.delim(system.file("extdata", "human_brain.txt",
                                     package="MSnID"),
                         stringsAsFactors=FALSE)
psms(msnid) <- PSMresults
show(msnid)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 997 at 37 % FDR
## #peptides: 687 at 57 % FDR
## #accessions: 665 at 65 % FDR
```

```
mzids <- system.file("extdata", "c_elegans.mzid.gz", package="MSnID")
msnid <- read_mzIDs(msnid, mzids)
```

```
## Reading from mzIdentMLs ...
```

```
## reading c_elegans.mzid.gz... DONE!
```

```
show(msnid)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 19055 at 29 % FDR
## #peptides: 9489 at 44 % FDR
## #accessions: 7414 at 76 % FDR
```

5.3.2 Analysis of Peptide Sequences

A particular properties of peptide sequences we are interested in are

1. irregular cleavages at the termini of the peptides and
2. missing cleavage site within the peptide sequences.

A particular properties of peptide sequences we are interested in are (1) irregular cleavages at the termini of the peptides and (2) missing cleavage site within the peptide sequences:

- Counting the number of irregular cleavage termimi (0, 1 or 2) in peptides sequence creates a new column `numIrregCleavages`.
- Counting the number of missed cleavages in peptides sequence correspondingly creates a `numMissCleavages` column.

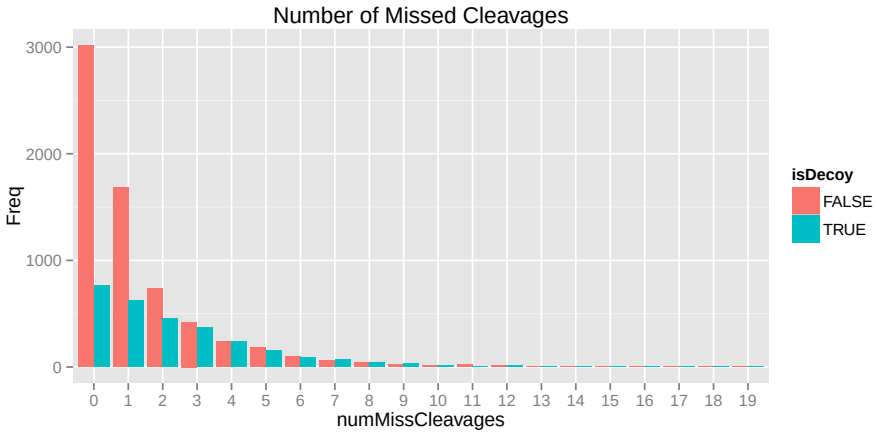
The default regular expressions for the `validCleavagePattern` and `missedCleavagePattern` correspond to trypsin specificity.

```
msnid <- assess_termini(msnid, validCleavagePattern="[KR]\\.[^P]")
msnid <- assess_missed_cleavages(msnid, missedCleavagePattern="[KR](?=[^P$])")
prop.table(table(msnid$numIrregCleavages))
```

```
##
##           0           1           2
## 0.801574390 0.189294149 0.009131462
```

Now the object has two more columns, `numIrregCleavages` and `numMissCleavages`, evidently corresponding to the number of termini with irregular cleavages and number of missed cleavages within the peptide sequence. The figure below shows that peptides with 2 or more missed cleavages are likely to be false identifications.

```
pepCleav <- unique(psms(msnid)[,c("numMissCleavages", "isDecoy", "peptide")])
pepCleav <- as.data.frame(table(pepCleav[,c("numMissCleavages", "isDecoy")]))
library("ggplot2")
ggplot(pepCleav, aes(x=numMissCleavages, y=Freq, fill=isDecoy)) +
    geom_bar(stat='identity', position='dodge') +
    ggtitle("Number of Missed Cleavages")
```



5.3.3 Defining the Filter

The criteria that will be used for filtering the MS/MS data has to be present in the `MSnID` object. We will use -log10 transformed MS-GF+ Spectrum E-value, reflecting the goodness of match experimental and theoretical fragmentation patterns as one the filtering criteria. Let's store it under the "msmsScore" name. The score density distribution shows that it is a good discriminant between non-decoy (red) and decoy hits (green).

For alternative MS/MS search engines refer to the engine-specific manual for the names of parameters reflecting the quality of MS/MS spectra matching. Examples of such parameters are `E-Value` for X!Tandem and `XCorr` and Δ`Cn2` for SEQUEST.

As a second criterion we will be using the absolute mass measurement error (in ppm units) of the parent ion. The mass measurement errors tend to be small for non-decoy (enriched with real identificaiton) hits (red line) and is effectively uniformly distributed for decoy hits.

```
msnid$msmsScore <- -log10(msnid$`MS-GF:SpecEValue`)
msnid$absParentMassErrorPPM <- abs(mass_measurement_error(msnid))
```

MS/MS fiters are handled by a special *MSnIDFilter* class objects. Individual filtering criteria can be set by name (that is present in `names(msnid)`), comparison operator (¿, ¡, =, ...) defining if we should retain hits with higher or lower given the threshold and finally the threshold value itself. The filter below set in such a way that retains only those matches that has less then 5 ppm of parent ion mass measurement error and more the 10^7 MS-GF:SpecEValue.

```
filtObj <- MSnIDFilter(msnid)
filtObj$absParentMassErrorPPM <- list(comparison="<", threshold=5.0)
filtObj$msmsScore <- list(comparison=">", threshold=8.0)
show(filtObj)
```

```
## MSnIDFilter object
## (absParentMassErrorPPM < 5) & (msmsScore > 8)
```

The stringency of the filter can be evaluated at different levels.

```
evaluate_filter(msnid, filtObj, level="PSM")
```

```
##             fdr    n
## PSM 0.002307439 9122
```

```
evaluate_filter(msnid, filtObj, level="peptide")
```

```
##                fdr    n
## peptide 0.00424371 3313
```

```
evaluate_filter(msnid, filtObj, level="accession")
```

```
##                   fdr    n
## accession 0.01770658 1207
```

5.3.4 Optimizing the Filter

The threshold values in the example above are not necessarily optimal and set just be in the range of probable values. Filters can be optimized to ensure maximum number of identifications (peptide-to-spectrum matches, unique peptide sequences or proteins) within a given FDR upper limit.

First, the filter can be optimized simply by stepping through individual parameters and their combinations. The idea has been described in [17]. The resulting `MSnIDFilter` object can be used for final data filtering or can be used as a good starting parameters for follow-up refining optimizations with more advanced algorithms.

```
filtObj.grid <- optimize_filter(filtObj, msnid, fdr.max=0.01,
                                method="Grid", level="peptide",
                                n.iter=500)
show(filtObj.grid)
```

```
## MSnIDFilter object
## (absParentMassErrorPPM < 10) & (msmsScore > 7.8)
```

The resulting `filtObj.grid` can be further fine tuned with such optimization routines as simulated annealing or Nelder-Mead optimization.

```
filtObj.nm <- optimize_filter(filtObj.grid, msnid, fdr.max=0.01,
                                method="Nelder-Mead", level="peptide",
                                n.iter=500)
show(filtObj.nm)
```

```
## MSnIDFilter object
## (absParentMassErrorPPM < 10) & (msmsScore > 7.8)
```

Evaluate non-optimized and optimized filters.

```
evaluate_filter(msnid, filtObj, level="peptide")
```

```
##                fdr    n
## peptide 0.00424371 3313
```

```
evaluate_filter(msnid, filtObj.grid, level="peptide")
```

```
##                  fdr    n
## peptide 0.009220702 3393
```

```
evaluate_filter(msnid, filtObj.nm, level="peptide")
```

```
##                  fdr    n
## peptide 0.009777778 3408
```

Finally applying filter to remove predominantly false identifications.

```
msnid <- apply_filter(msnid, filtObj.nm)
show(msnid)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 9480 at 0.49 % FDR
## #peptides: 3408 at 0.98 % FDR
```

```
## #accessions: 1253 at 3.8 % FDR
```

Removing hits to decoy and contaminant sequences using the same `apply_filter` method.

```
msnid <- apply_filter(msnid, "isDecoy == FALSE")
show(msnid)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 9434 at 0 % FDR
## #peptides: 3375 at 0 % FDR
## #accessions: 1207 at 0 % FDR
```

```
msnid <- apply_filter(msnid, "!grepl('Contaminant',accession)")
show(msnid)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 9425 at 0 % FDR
## #peptides: 3368 at 0 % FDR
## #accessions: 1205 at 0 % FDR
```

5.3.5 Interface with Other Bioconductor Packages

One can extract the entire PSMs tables as `data.frame` or `data.table`

```
psm.df <- psms(msnid)
psm.dt <- as(msnid, "data.table")
```

If only interested in the non-redundant list of confidently identified peptides or proteins

```
peps <- peptides(msnid)
head(peps)
```

```
## [1] "K.AISQIQEYVDYYGGSGVQHIALNTSDIITAIEALR.A"
## [2] "K.SAGSGYLVGDSLTFVDLLVAQHTADLLAANAALLDEFPQFK.A"
## [3] "K.NSIFTNVAETANGEYFWEGLEDEIADKNVDITTWLGEK.W"
## [4] "R.VFCLLGDGESAEGSVWEAAAFASIYKLDNLVAIVDVNR.L"
## [5] "R.TTDSDGNNTGLDLYTVDQVEHSNYVEQNFLDFIFVFR.K"
## [6] "R.KFDADGSGKLEFDEFCALVYTVANTVDKETLEKELR.E"
```

```
prots <- accessions(msnid)
head(prots)
```

```
## [1] "CE02347" "CE07055" "CE12728" "CE36358" "CE36359"
## [6] "CE36360"
```

```
prots <- proteins(msnid) # may be more intuitive then accessions
head(prots)
```

```
## [1] "CE02347" "CE07055" "CE12728" "CE36358" "CE36359"
## [6] "CE36360"
```

The *MSnID* package is aimed at providing convenience functionality to handle MS/MS identifications. Quantification *per se* is outside of the scope of the package. The only type of quantitation that can be seamlessly tied with MS/MS identification analysis is so-called *spectral counting* approach. In such an approach a peptide abundance is considered to be directly proportional to the number of matched MS/MS spectra. In its turn protein abunance is proportional to

the sum of the number of spectra of the matching peptides. The *MSnID* object can be converted to an *MSnSet* object defined in *MSnbase* that extends generic Bioconductor *eSet* class to quantitative proteomics data. The spectral count data can be analyzed with *msmsEDA*, *msmsTests* or *DESeq* packages.

```
msnset <- as(msnid, "MSnSet")
library("MSnbase")
head(fData(msnset))
```

```
##                                                 peptide
## A.AGLKPTQAMVTK.A                          A.AGLKPTQAMVTK.A
## A.AVLEYLAAEVLELAGNAAR.D               A.AVLEYLAAEVLELAGNAAR.D
## A.DCLHCICMR.E                                A.DCLHCICMR.E
## A.DLFTSIADMQNLLETER.N                  A.DLFTSIADMQNLLETER.N
## A.EKKRKAAETSLMEK.D                        A.EKKRKAAETSLMEK.D
## A.EQLPEKFYGTFDLDHSENFDEYLTAK.G A.EQLPEKFYGTFDLDHSENFDEYLTAK.G
##                                                   accession
## A.AGLKPTQAMVTK.A                                CE01236, CE30652
## A.AVLEYLAAEVLELAGNAAR.D                         CE04501, CE05477
## A.DCLHCICMR.E                  CE04442, CE17549, CE24850, CE34002
## A.DLFTSIADMQNLLETER.N                             CE20261
## A.EKKRKAAETSLMEK.D                                CE27133
## A.EQLPEKFYGTFDLDHSENFDEYLTAK.G                    CE04532
```

```
head(exprs(msnset))
```

```
##                               c_elegans_A_3_1_21Apr10_Draco_10-03-04_dta.txt
## A.AGLKPTQAMVTK.A                                                           1
## A.AVLEYLAAEVLELAGNAAR.D                                                    1
## A.DCLHCICMR.E                                                              1
## A.DLFTSIADMQNLLETER.N                                                      1
## A.EKKRKAAETSLMEK.D                                                         1
## A.EQLPEKFYGTFDLDHSENFDEYLTAK.G                                             1
```

Note, the convertion from `MSnID` to `MSnSet` uses peptides as features. The number of redundant peptide observations represent so-called spectral count that can be used for rough quantitative analysis. Summing of all of the peptide counts to a proteins level can be done with `combineFeatures` function from *MSnbase* package.

```
msnset <- combineFeatures(msnset,
                          fData(msnset)$accession,
                          redundancy.handler="unique",
                          fun="sum",
                          cv=FALSE)
```

```
## Combined 2082 features into 670 using sum
```

```
head(fData(msnset))
```

```
##                                     peptide accession
## CE00078                          K.RLPVAPR.G   CE00078
## CE00103 K.LPNDDIGVQVSYLGEPHTFTPEQVLAALLTK.L   CE00103
## CE00134                        I.PAEVAEHLK.A   CE00134
## CE00209             K.ALEGPGPGEDAAHSENNPPR.N   CE00209
## CE00302                   K.LTYFDIHGLAEPIR.L   CE00302
## CE00318         K.ALNALCAQLMTELADALEVLDTDK.S   CE00318
```

```
head(exprs(msnset))
```

```
##         c_elegans_A_3_1_21Apr10_Draco_10-03-04_dta.txt
## CE00078                                              4
```

```
## CE00103                                                    3
## CE00134                                                    4
## CE00209                                                    8
## CE00302                                                    2
## CE00318                                                   11
```

6 A comprehensive example

This subsection demonstrates a real-world example of using *R* for proteomics data processing, starting from download of raw data and FASTA files with the *rpx* package, peptide and protein identification with X!Tandem and *rTANDEM*, filtering MS/MS data with *MSnID* and statistical analysis of spectral counts with *msmsTests*.

6.1 Getting the data

```
library("rpx")
id <- "PXD002161"
px <- PXDataset(id)
try(setInternet2(FALSE),silent=TRUE)
library("jsonlite")
addr <- "http://www.ebi.ac.uk:80/pride/ws/archive/%s/list/project/%s"
files <- fromJSON(sprintf(addr, "file", id))$list
assays <- fromJSON(sprintf(addr, "assay", id))$list

files <- subset(files, fileType == 'PEAK',
                select = c("assayAccession","fileName"))
assays <- assays[,c("assayAccession",
                    "experimentalFactor",
                    "proteinCount",
                    "peptideCount",
                    "uniquePeptideCount",
                    "identifiedSpectrumCount",
                    "totalSpectrumCount")]

pttrn <- "Age: young, Diet; fully fed"
assays <- subset(assays, grepl(pttrn, experimentalFactor, fixed=T))
# encode phenotype and sample names
group <- sub(".*Name: Y-(.+?)-FF\\.(\\d)", "\\1", assays$experimentalFactor)
splnm <- sub(".*Name: Y-(.+?)-FF\\.(\\d)", "\\1_\\2", assays$experimentalFactor)

assays <- with(assays, {data.frame(assayAccession,
                                   phenotype=sub(".*Name: Y-(.+?)-FF\\.(\\d)",
                                                 "\\1", experimentalFactor),
                                   sampleName=sub(".*Name: Y-(.+?)-FF\\.(\\d)",
                                                  "\\1_\\2", experimentalFactor),
                                   stringsAsFactors=F)})
files <- subset(files, assayAccession %in% assays$assayAccession)

## do we already have all files?
allfiles <- length(dir(pattern = "c_elegans_.*mzML")) == 10

if (!allfiles)
```

```
    pxget(px, files$fileName) # fetch them from PX

files$datasetName <- sub('.mzML.gz','', files$fileName, fixed=TRUE)
meta <- merge(files[,c("assayAccession","datasetName")], assays)
rownames(meta) <- meta$datasetName
meta <- meta[order(meta$sampleName),]
rownames(meta) <- NULL

if (!allfiles) {
    library("R.utils")
    sapply(list.files(pattern = "mzML.gz"), gunzip)
}
```

Now we have `*.mzML` files for MS/MS search. Let's download FASTA file from the current release of the Wormbase. We will leverage Bioconductor's *Biostrings* package to reverse the protein sequence and append them to the original FASTA file. The reverse sequences will be use for estimate of the false discovery rate of peptide-to-spectrum matches, peptide and protein identifications.

```
library("Biostrings")
fasta_location <-  "ftp://ftp.wormbase.org/pub/wormbase/releases/WS250/species/c_elegans/PRJNA13758/c_eleg
fwd.seqs <- readAAStringSet(fasta_location, format="fasta",
                            nrec=-1L, skip=0L, use.names=TRUE)
rev.seqs <- reverse(fwd.seqs)
names(rev.seqs) <- paste("XXX", names(rev.seqs), sep='_')
fwd.rev.seqs <- append( fwd.seqs, rev.seqs)
writeXStringSet(x=fwd.rev.seqs, filepath="c_elegans_fwd_rev.fasta", format="fasta")
```

6.2 Peptide identification

Setting up X!Tandem parameter files.

```
library("rTANDEM")
param <- setParamOrbitrap()
taxonomy <- rTTaxo(taxon="celegans",
                   format="peptide",
                   URL= "c_elegans_fwd_rev.fasta")
param <- setParamValue(param, 'list path', 'taxonomy information', taxonomy)
param <- setParamValue(param, 'protein', 'taxon', value='celegans')

def.input.path <- system.file("extdata/default_input.xml", package="rTANDEM")
param <- setParamValue(param, 'list path', 'default parameters',
                       value=def.input.path)

param <- setParamValue(param, "output", "message", "r-for-proteomics ")
param <- setParamValue(param, "refine", value="no")
```

Note, we will be executing X!Tandem on all available cores on the computer. To determine the number of cores we leverage `detectCores()` function from the *parallel* package.

```
library("parallel")
param <- setParamValue(param, "spectrum", "threads", detectCores())
```

Actuall excetion of the X!Tandem searches. As the output we will capture the names of corresponging XML files with the MS/MS search results.

```
output.files <- lapply(sub("\\.gz","",files$fileName),
                       function(x){
                           param <- setParamValue(param, 'spectrum', 'path', value=x)
                           output.file <- tandem(param)})

## Loading spectra
##  (mzML)....... loaded.
## Spectra matching criteria = 12325
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2...3...4...
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3692
## Unique models = 2773
## Estimated false positives = 30 +/- 5
##
##
## Loading spectra
##  (mzML)...... loaded.
## Spectra matching criteria = 11861
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2..3..4..
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3617
## Unique models = 2649
## Estimated false positives = 29 +/- 5
##
```

```
##
## Loading spectra
##  (mzML)...... loaded.
## Spectra matching criteria = 11821
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2..3..4..
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3936
## Unique models = 2966
## Estimated false positives = 31 +/- 6
##
##
## Loading spectra
##  (mzML)....... loaded.
## Spectra matching criteria = 12413
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2...3...4...
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3423
## Unique models = 2424
## Estimated false positives = 30 +/- 5
##
##
## Loading spectra
##  (mzML)....... loaded.
## Spectra matching criteria = 12067
```

```
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2...3...4...
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3122
## Unique models = 2111
## Estimated false positives = 29 +/- 5
##
##
## Loading spectra
##  (mzML)...... loaded.
## Spectra matching criteria = 11616
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2..3..4..
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3566
## Unique models = 2445
## Estimated false positives = 29 +/- 5
##
##
## Loading spectra
##  (mzML)....... loaded.
## Spectra matching criteria = 12896
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
```

```
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2...3...4...
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3639
## Unique models = 2465
## Estimated false positives = 34 +/- 6
##
##
## Loading spectra
##  (mzML)...... loaded.
## Spectra matching criteria = 11628
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2..3..4..
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3729
## Unique models = 2590
## Estimated false positives = 28 +/- 5
##
##
## Loading spectra
##  (mzML)....... loaded.
## Spectra matching criteria = 12839
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2...3...4...
```

```
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3516
## Unique models = 2679
## Estimated false positives = 31 +/- 6
##
##
## Loading spectra
##  (mzML)....... loaded.
## Spectra matching criteria = 12714
## Starting threads .... started.
## Computing models:
##  r-for-proteomics r-for-proteomics r-for-proteomic | 50 ks
##  s r-fo
##  sequences modelled = 56 ks
## Model refinement:
## Merging results:
##  from 2...3...4...
##
## Creating report:
##  initial calculations  ..... done.
##  sorting  ..... done.
##  finding repeats ..... done.
##  evaluating results ..... done.
##  calculating expectations ..... done.
##  writing results ..... done.
##
## Valid models = 3467
## Unique models = 2179
## Estimated false positives = 30 +/- 5
```

Since X!Tandem reports the results in its own XML format, parsing then is a little bit involved process. As we mentioned above, fortunately, the most recent version of X!Tandem[22] PILEDRIVER (2015.04.01) is capable of producing results in `mzIdentML` format. Then extracting the results will be trivial with the help of *mzID* package. The change will be probably caught up in the upcoming versions of the *rTANDEM* package. However, at this point we will use a custom function that produces a `data.frame` with all the necessary columns. The columns that are required for the *MSnIDP* package to create an MS/MS identification object are: accession, calculatedMassToCharge, chargeState, experimentalMassToCharge, isDecoy, peptide, spectrumFile and spectrumID.

```
library("dplyr")
.P <- MSnID:::.PROTON_MASS # 1.007276

## parse to comply with MSnID
xtandem_to_df <- function(output.file){
    res <- GetResultsFromXML(output.file)
    proteins <- GetProteins(res)
```

[22] http://www.thegpm.org/tandem/

```
    peptides <- GetPeptides(protein.uid = proteins$uid, results = res)
    peptides <- select(as.data.frame(res@peptides),
                       prot.uid, spectrum.mh, mh, expect.value,
                       tandem.score, start.position, end.position,
                       spectrumID = spectrum.id,
                       chargeState = spectrum.z,
                       pepSeq = sequence)
    peptides <- mutate(peptides,
                       calculatedMassToCharge = (mh + .P*(chargeState - 1))/chargeState,
                       experimentalMassToCharge = (spectrum.mh + .P*(chargeState - 1))/chargeState,
                       spectrum.mh = NULL,
                       mh = NULL)
    proteins <- select(as.data.frame(res@proteins),
                       prot.uid=uid, sequence, description)
    proteins <- mutate(proteins,
                       accession = sub("(\\S+)\t.*", "\\1", description),
                       isDecoy = grepl("XXX_", accession))
    x <- inner_join(peptides, proteins)
    pre <- with(x, substr(sequence, start.position-1, start.position-1))
    pre <- ifelse(pre == "", "-", pre)
    post <- with(x, substr(sequence, end.position+1, end.position+1))
    post <- ifelse(post == "", "-", post)
    x <- mutate(x,
                peptide = paste(pre,x$pepSeq, post, sep='.'),
                sequence = NULL)
    return(x)
}

out <- lapply(output.files, xtandem_to_df)
for(i in seq_along(out))
    out[[i]]$spectrumFile <- sub("\\.mzML\\.gz","",files$fileName)[i]
out <- Reduce(rbind, out)
```

6.3 Filtering identification data

The *MSnID* class object from the library of the same name allows flexible manipulation and filterting of MS/MS identification results data.

```
library(MSnID)
m <- MSnID()

## Note, the anticipated/suggested columns in the
## peptide-to-spectrum matching results are:
## -----------------------------------------------
## accession
## calculatedMassToCharge
## chargeState
## experimentalMassToCharge
## isDecoy
## peptide
## spectrumFile
## spectrumID

psms(m) <- out
```

```
show(m)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  10
## #PSMs: 73999 at 1.2 % FDR
## #peptides: 6314 at 9 % FDR
## #accessions: 3512 at 28 % FDR
```

Let's take a look and mass measurement error and a large scale. Some peptide-to-spectrum matches have the error as large as +1 and +2 Dalton. This is due to the fact that the instrument sometime picks non-monoisotopic peaks for fragmentation. This is more common for the peptides with large mass.

```
library("ggplot2")
dM <- with(psms(m),
           round((experimentalMassToCharge-calculatedMassToCharge)*chargeState))
x <- as.data.frame(table(data.frame(dM, isDecoy=m$isDecoy)))
x <- as.data.frame(table(dM, isDecoy=m$isDecoy))
ggplot(x, aes(x=dM, fill=isDecoy, y=Freq)) +
      geom_bar(stat="identity", width=0.25)
```

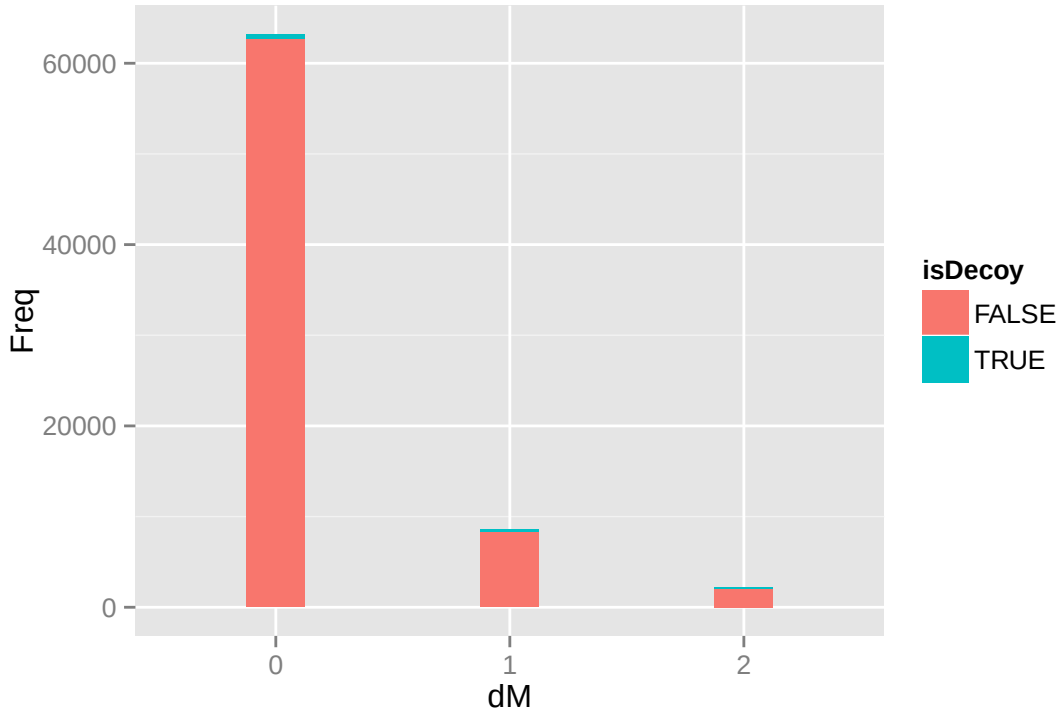


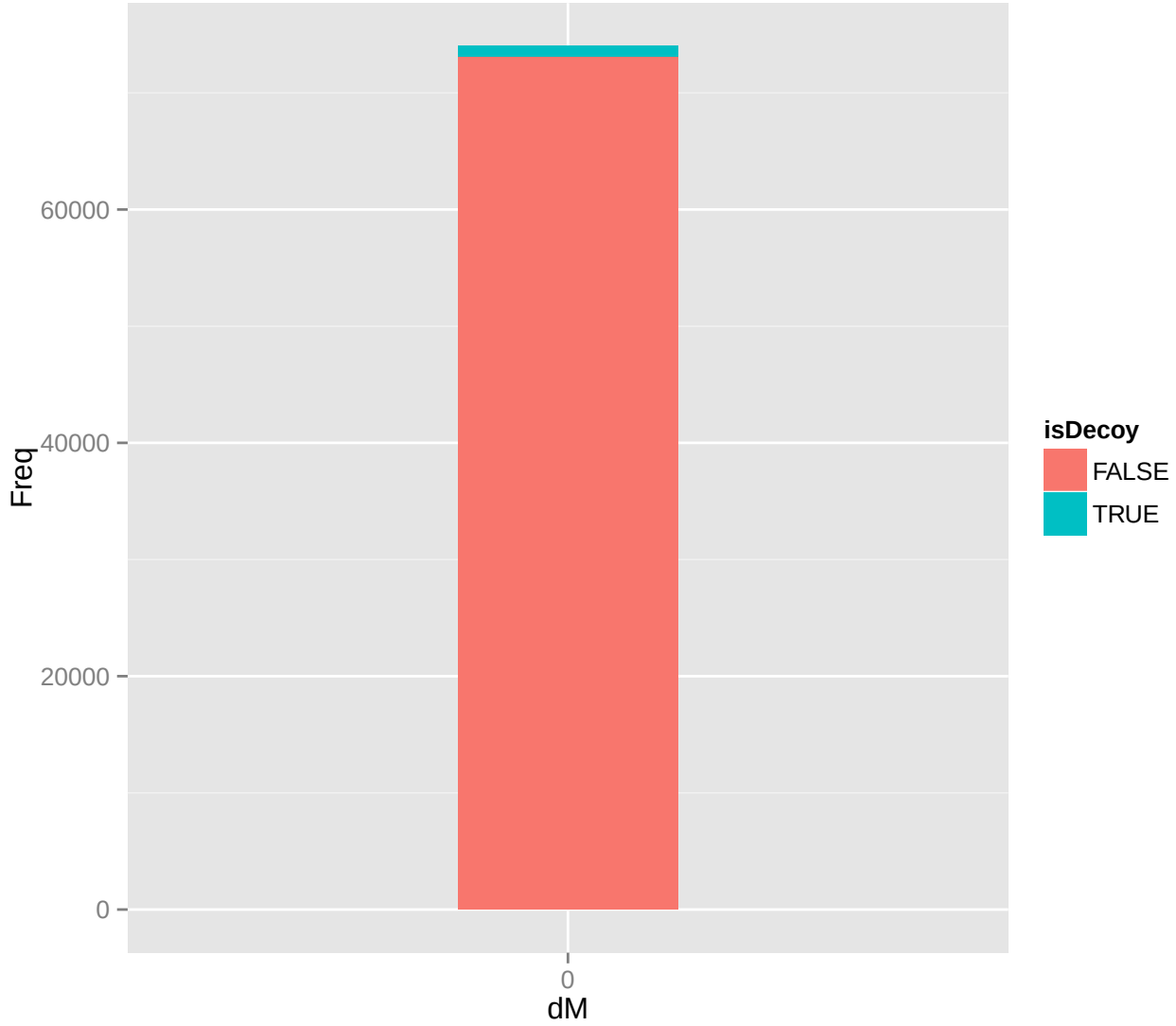
Figure 13: Isotope selection error before and after correction.

For correction of the parent ion mass at the isotope-scale level there is a method with the corresponding name `correct_peak_selection()`.

The mass measurement error at a finer grain, typicaly related to the mis-calibrated instument, can be accessed with `mass_measurement_error` method.

Finding the right set of filtering criteria for MS/MS data is not a trivial task. For the purpose of flexible filter optimization the *MSnID* package has the *MSnIDFilter* class. We will start with defining the criteria we want to filter on.

```
m <- correct_peak_selection(m)
dM <- with(psms(m),
           round((experimentalMassToCharge-calculatedMassToCharge)*chargeState))
x <- as.data.frame(table(data.frame(dM, isDecoy=m$isDecoy)))
x <- as.data.frame(table(dM, isDecoy=m$isDecoy))
ggplot(x, aes(x=dM, fill=isDecoy, y=Freq)) +
      geom_bar(stat="identity", width=0.25)
```




```
m$score <- -log10(m$expect.value) # higher the better
m$mme <- abs(mass_measurement_error(m)) # lower the better
fltr <- MSnIDFilter(m)
```

Let define some starting values. What is important is to define what should be retained higher or lower the given threshold. Threshold itself can be set to arbitrary value if "Grid" optimization method will be used as a first pass. "Grid" optimization loops through a preset number (`n.iter` argument) of combinations of thresholds ignoring the initial values. In the filtering criteria optimization routines, the objective function is the number of peptide-to-spectrum matches, peptide or protein identifications defined by the `level` argument. However, if the FDR (estimated using reverse

```
mme <- data.frame(ppm=mass_measurement_error(m), isDecoy=m$isDecoy)
ggplot(mme, aes(x=ppm, fill=isDecoy)) +
      geom_histogram(binwidth=1) +
      xlim(-20,+20) +
      facet_wrap( ~ isDecoy) +
        scale_y_sqrt()
```

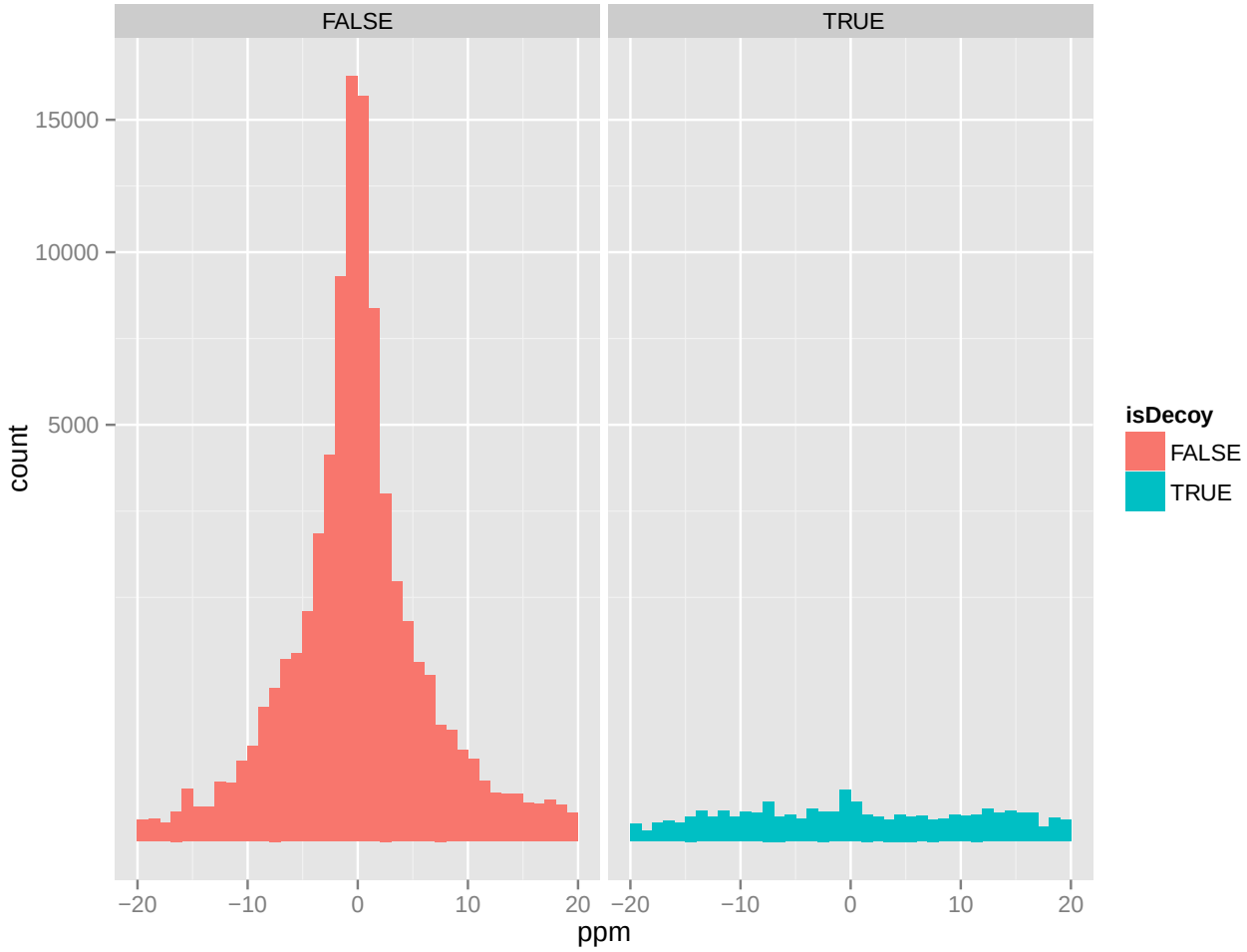


Figure 14: Parent ion mass measurement error histograms for non-decoy and decoy peptide-to-spectrum matches

sequences) exceeds the pre-defined maxium value, the objective function sets to zero.

```
fltr$score <- list(comparison=">", threshold=0)
fltr$mme <- list(comparison="<", threshold=0)
fltr.grid <- optimize_filter(fltr, m, fdr.max=0.01,
method="Grid", level="peptide",
n.iter=1000)
show(fltr.grid)
```

```
## MSnIDFilter object
## (score > 1.7) & (mme < 8.8)
```

```
as.numeric(fltr.grid)
```

```
##    score      mme
```

```
## 1.698970 8.847089
```

```
evaluate_filter(m, fltr.grid)
```

```
##                    fdr     n
## PSM       0.001225841 62891
## peptide   0.009682172  4797
## accession 0.031218014  2015
```

Further fine-tuning of the filtering criteria can be approached with Nelder-Mead or simulated annealing optimization techniques specified by the `method` arugment.

```
fltr.sann <- optimize_filter(fltr.grid, m, fdr.max=0.01,
method="SANN", level="peptide",
n.iter=1000)
show(fltr.sann)
```

```
## MSnIDFilter object
## (score > 1.6) & (mme < 8.4)
```

```
as.numeric(fltr.sann)
```

```
##    score      mme
## 1.649328 8.423798
```

```
evaluate_filter(m, fltr.sann)
```

```
##                    fdr     n
## PSM       0.001217276 63333
## peptide   0.009993754  4851
## accession 0.030503305  2027
```

Applying the filter.

```
m <- apply_filter(m, fltr.sann)
```

Visualizing the number of peptides covering the proteins.

Filtering out single-hit-wonders (that is proteins covered with only one peptide). Note the massive drop in FDR (down to 0%). Therefore, if this particular path of filtering MS/MS identifications seems stringent the steps can be easily flipped around in the script or re-arranged in any other way that produce the results suitable for a particular application.

```
nms <- names(which(table(pp$accession) > 1))
m <- apply_filter(m, "accession %in% nms")
show(m)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  10
## #PSMs: 60535 at 0 % FDR
## #peptides: 4291 at 0 % FDR
## #accessions: 1117 at 0 % FDR
```

```
pp <- unique(psms(m)[,c("peptide","accession")])
pep_per_prot <- as.data.frame(table(table(pp$accession)))
pep_per_prot$peptides <- with(pep_per_prot,
                              ifelse(as.numeric(Var1) > 5, "6+", as.numeric(Var1)))
ggplot(pep_per_prot,
         aes(x=factor(1), fill=peptides, y=Freq)) +
       geom_bar(width=1, stat = "identity", position = "stack") +
       coord_polar(theta = "y") +
       ylab('') + xlab('') +
       scale_fill_discrete(name="number of\npeptides per\nprotein")
```

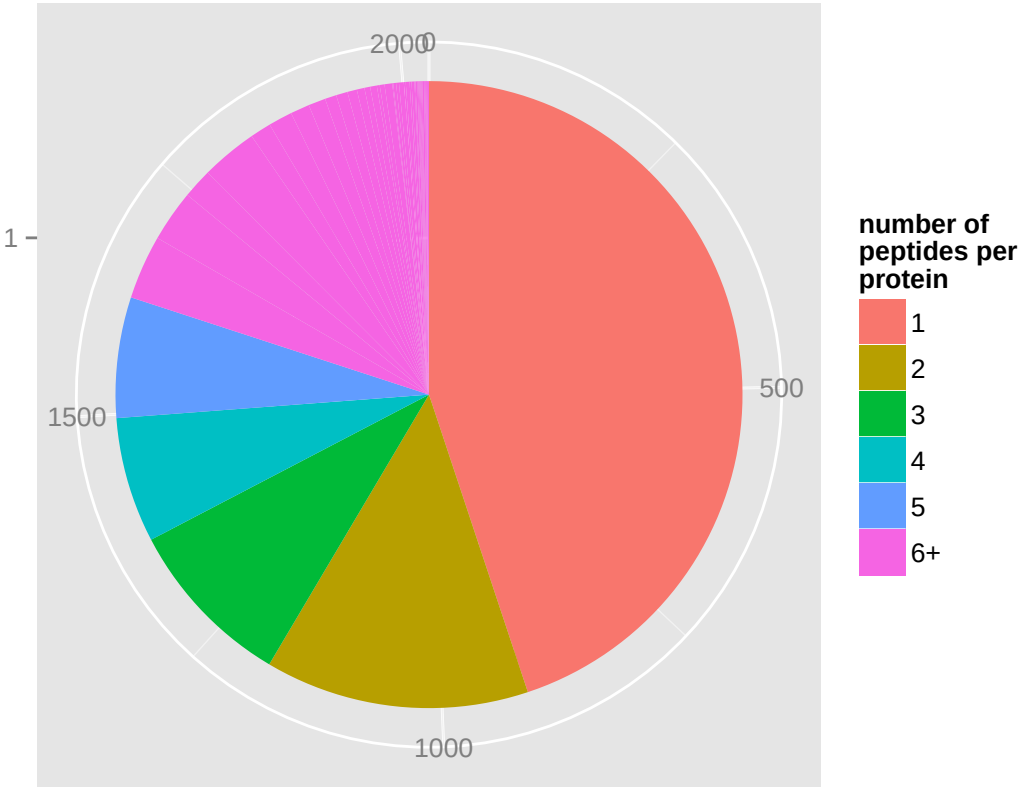


Figure 15: Number of identified peptides per protein. Almost half of the proteins are 'single-hit wonders', i.e. they have been identified with one peptide only.

6.4 Exploratory data analysis

We continue using the `MSnID` object results from X!Tandem searches of 10 *C. elegans* files. First, we will convert it to an `MSnSet` object that is more taylored to quantitative analysis and explore the data using the *msmsEDA* package.

```
mset <- as(m, "MSnSet")
show(mset)
```

```
## MSnSet (storageMode: lockedEnvironment)
## assayData: 4291 features, 10 samples
##   element names: exprs
## protocolData: none
## phenoData: none
## featureData
##   featureNames: A.PAAKPQVK.K A.PAAKPQVKK.N ...
##     R.YYYDHSKK.H (4291 total)
##   fvarLabels: peptide accession
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
##  MSnbase version: 1.19.4
```

Summing up peptide counts to protein (or assession in general terms) level. The redundancy of peptide to protein mapping is controlled by the `redundancy.handler` argument. In this case we will use only uniquely mapping peptides for protein quantification.

```
library("MSnbase")
mset <- combineFeatures(mset,
fData(mset)$accession,
redundancy.handler="unique",
fun="sum",
cv=FALSE)
```

```
## Combined 2525 features into 524 using sum
```

```
show(mset)
```

```
## MSnSet (storageMode: lockedEnvironment)
## assayData: 524 features, 10 samples
##   element names: exprs
## protocolData: none
## phenoData: none
## featureData
##   featureNames: B0041.4 B0250.1 ... ZK973.6 (524
##     total)
##   fvarLabels: peptide accession
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
## Subset [4291,10][2525,10] Sat Dec  5 02:13:11 2015
## Combined 2525 features into 524 using sum: Sat Dec  5 02:13:11 2015
##  MSnbase version: 1.19.4
```

Subsetting to the ones that were detected in at least half of the samples

```
mset <- mset[rowSums(exprs(mset) > 0) >= 6,]
```

Updating `phenoData` using the meta-information on experimental designed fetched from ProteomeXchange project page[23].

```
pData(mset) <- meta[match(sampleNames(mset), meta$datasetName),]
mset$phenotype <- as.factor(mset$phenotype)
sampleNames(mset) <- mset$sampleName
pData(mset)
```

```
##        assayAccession
## ctrl_1          52450
## daf2_1          52454
## ctrl_2          52456
## daf2_2          52448
## ctrl_3          52445
## daf2_3          52425
## ctrl_4          52443
## daf2_4          52441
## ctrl_5          52437
## daf2_5          52439
##                                    datasetName phenotype
## ctrl_1 c_elegans_A_3_1_21Apr10_Draco_10-03-04      ctrl
## daf2_1 c_elegans_A_3_3_21Apr10_Draco_10-03-04      daf2
## ctrl_2 c_elegans_B_2_1_21Apr10_Draco_10-03-05      ctrl
## daf2_2 c_elegans_B_2_3_21Apr10_Draco_10-03-05      daf2
## ctrl_3 c_elegans_C_1_1_21Apr10_Draco_10-03-06      ctrl
## daf2_3 c_elegans_C_1_3_21Apr10_Draco_10-03-06      daf2
## ctrl_4 c_elegans_D_1_1_21Apr10_Draco_10-03-07      ctrl
## daf2_4 c_elegans_D_1_3_21Apr10_Draco_10-03-07      daf2
## ctrl_5 c_elegans_E_3_1_21Apr10_Draco_10-03-04      ctrl
## daf2_5 c_elegans_E_3_3_21Apr10_Draco_10-03-04      daf2
##        sampleName
## ctrl_1     ctrl_1
## daf2_1     daf2_1
## ctrl_2     ctrl_2
## daf2_2     daf2_2
## ctrl_3     ctrl_3
## daf2_3     daf2_3
## ctrl_4     ctrl_4
## daf2_4     daf2_4
## ctrl_5     ctrl_5
## daf2_5     daf2_5
```

The PCA plot below shows clear separation of control and *daf-2* mutant *C. elegans* strains.

6.5 Statistical analysis

Null hypothesis significance testing using quasi-likelihood Poisson approach from the *msmsTests* package. The p-value histogram shows a substantial increase of p-value frequency in low-value bins (tailing-up), indicating a significant proteome difference between the strains.

A volcano plot is an alternative way to visualize the estimate fold and significance of change of change.

[23] http://www.ebi.ac.uk/pride/archive/projects/PXD002161

```
library("msmsEDA")
counts.pca(mset,
           facs = pData(mset)[,'phenotype',drop=FALSE],
           snms = sampleNames(mset))
```

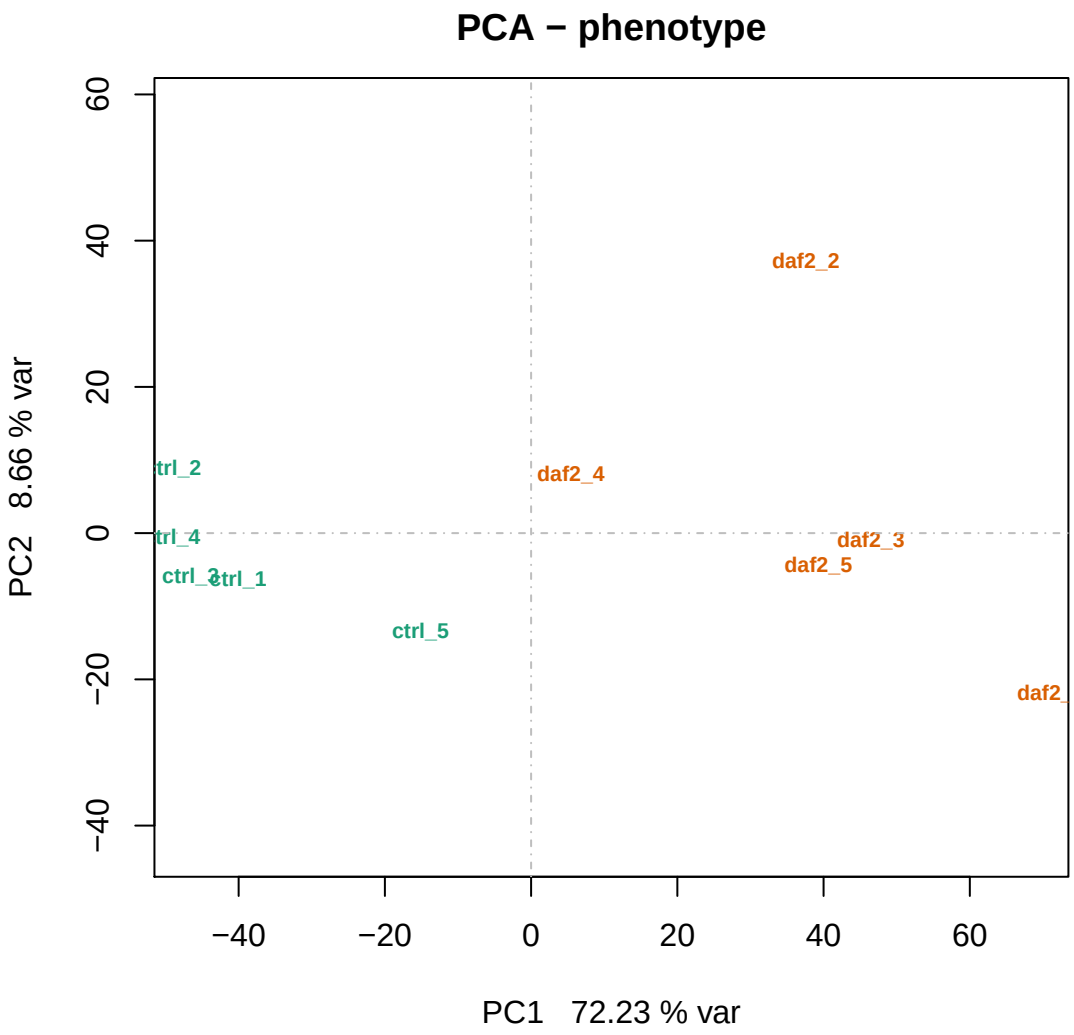


Figure 16: Clear separation of control and daf-2 strains on PCA plot indicates on substantial difference in proteome composition.

Finally, we use a heatmap to visualize relative protein abundances across the study samples. In this example we will subset the proteins only to the ones that have a fold change (up or down) more then 2-fold and pass the 0.05 threshold for the adjusted p-value.

```
library("msmsTests")
  alt.f <- "y ~ phenotype"
    null.f <- "y ~ 1"
      div <- colSums(exprs(mset)) # normalization factor
        res <- msms.glm.qlll(mset, "y ~ phenotype", "y ~ 1", div=div)
        hist(res$p.value, 100)
```

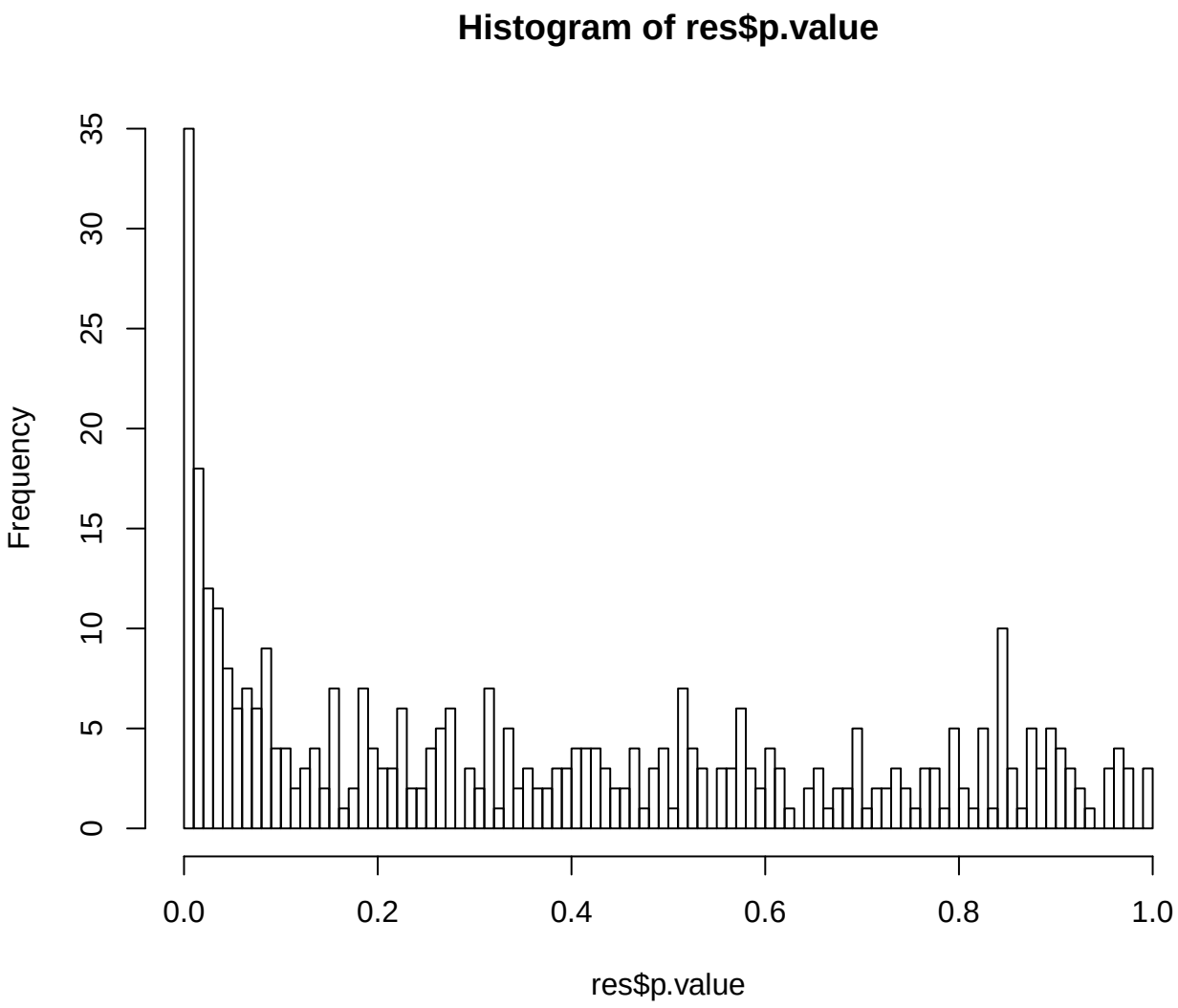


Figure 17: Histogram of p-values.

```
lst <- test.results(res, mset, pData(mset)$phenotype,
                    "ctrl","daf2", div, alpha=0.05,
                    minSpC=0, minLFC=1, method="BH")
res.volcanoplot(lst$tres, min.LFC=1, max.pval=0.05, ylbls=NULL, maxy=4)
```

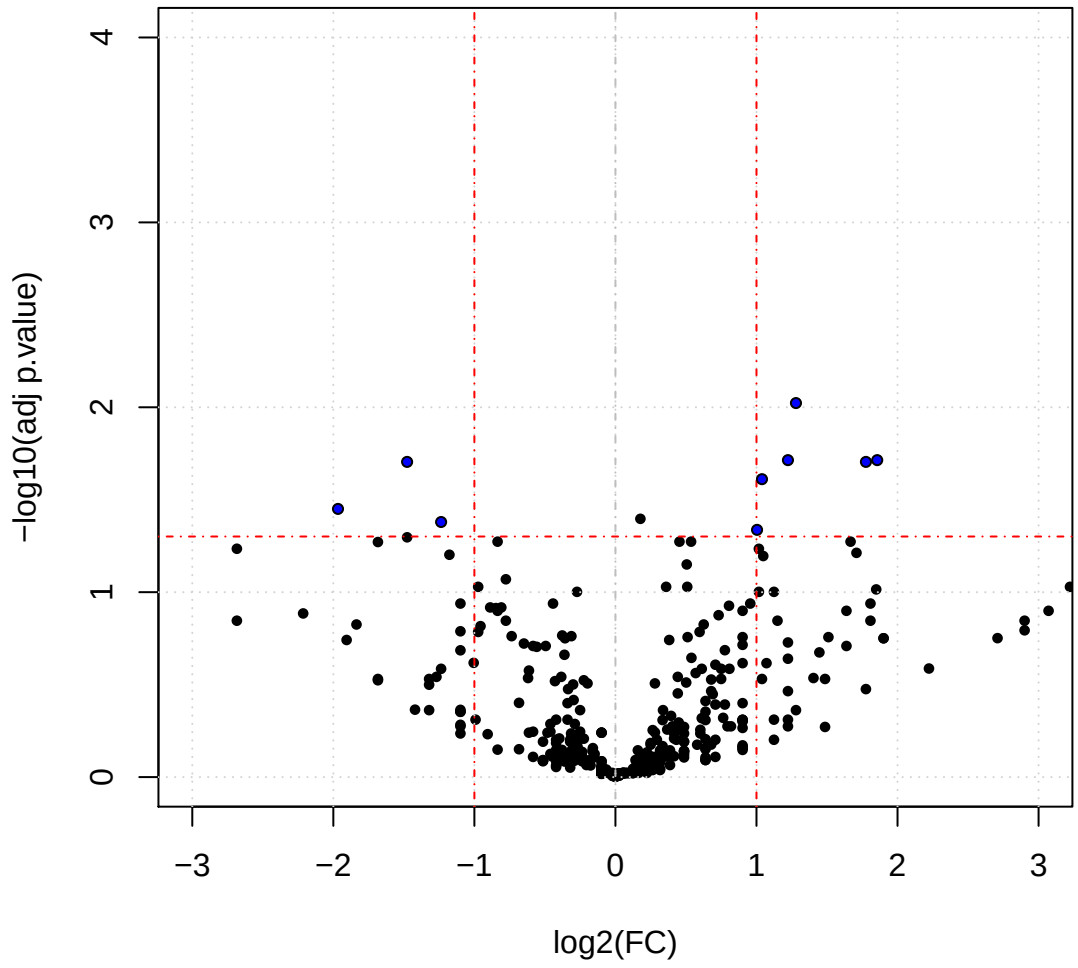


Figure 18: Volcano plot

```
library("Heatplus")
regulated <- subset(lst$tres, adjp < 0.05 & abs(LogFC) > 1)
## order MSnSet object the daf-16 status
mset <- mset[,order(pData(mset)$phenotype)]
## matrix with regulated proteins
selected.data <- exprs(mset[rownames(regulated),])
## scaling counts from 0 to 1
selected.data <- sweep(selected.data, 1, apply(selected.data, 1, min), '-')
selected.data <- sweep(selected.data, 1, apply(selected.data, 1, max), '/')
heatmap_plus(selected.data,
               scale='none',
               col=colorRampPalette(c("snow","steelblue"))(10))
```

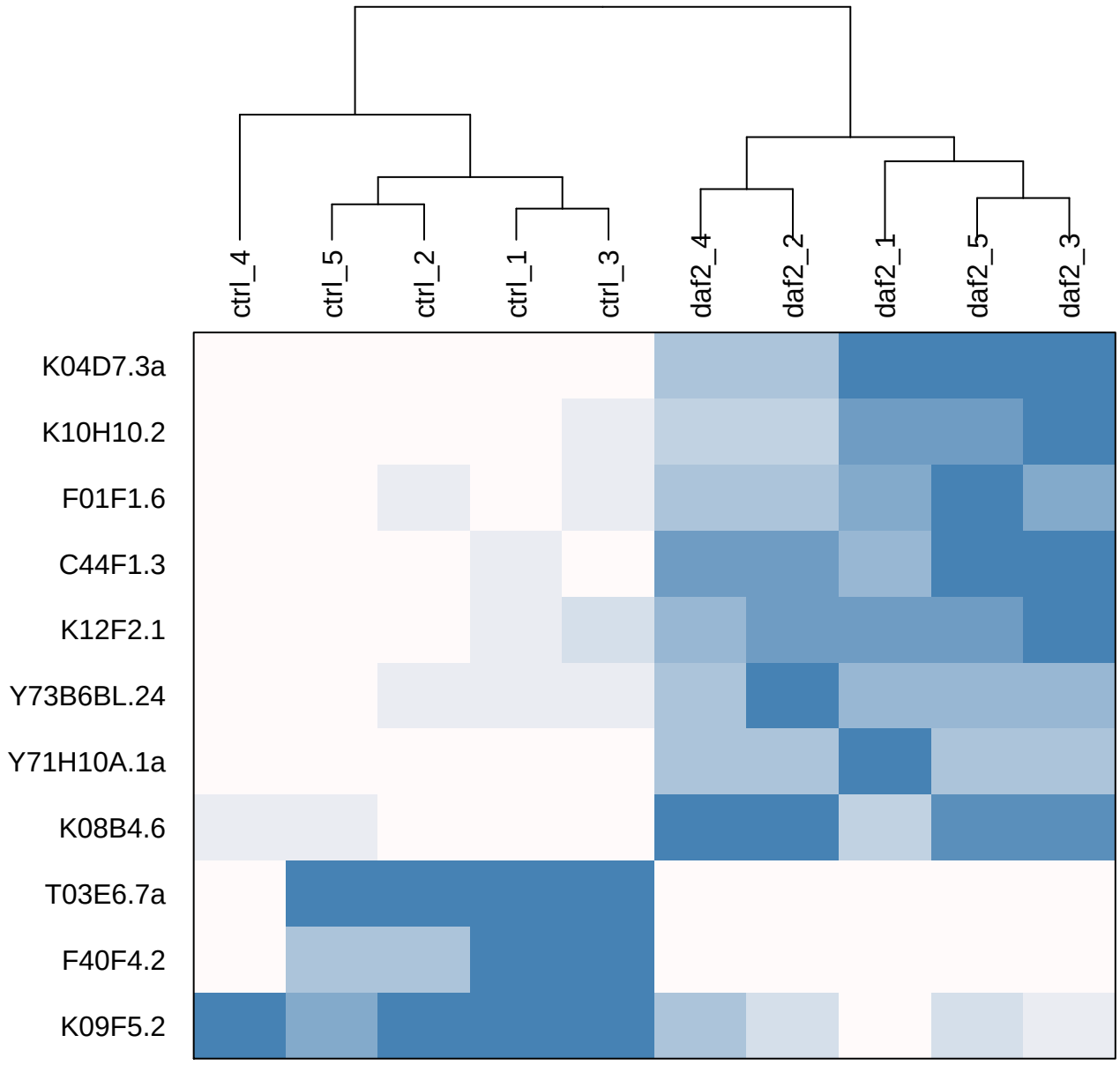


Figure 19: Heatmap of top proteins.

7 Quality control

Quality control (QC) is an essential part of any high throughput data driven approach. Bioconductor has a rich history of QC for various genomics data and currently two packages support proteomics QC.

proteoQC provides a dedicated a dedicated pipeline that will produce a dynamic and extensive html report. It uses the *rTANDEM* package to automate the generation of identification data and uses information about the experimental/replication design.

The *qcmetrics* package is a general framework to define QC metrics and bundle them together to generate html or pdf reports. It provides some ready made metrics for MS data and ^{15}N labelled data.

8 Annotation

In this section, we briefly present some *Bioconductor* annotation infrastructure.

We start with the *hpar* package, an interface to the *Human Protein Atlas* [18, 19], to retrieve subcellular localisation information for the ENSG00000002746 ensemble gene.

```
id <- "ENSG00000105323"
library("hpar")
getHpa(id, "SubcellularLoc")
```

```
##                 Gene            Main.location
## 1830 ENSG00000105323 Nucleus but not nucleoli
##      Other.location Expression.type Reliability
## 1830                           APE  Supportive
```

Below, we make use of the human annotation package *org.Hs.eg.db* and the Gene Ontology annotation package *GO.db* to retrieve compatible information with above.

```
library("org.Hs.eg.db")
library("GO.db")
ans <- AnnotationDbi::select(org.Hs.eg.db,
keys = id,
columns = c("ENSEMBL", "GO", "ONTOLOGY"),
keytype = "ENSEMBL")
```

```
## 'select()' returned 1:many mapping between keys and
## columns
```

```
ans <- ans[ans$ONTOLOGY == "CC", ]
ans
```

```
##            ENSEMBL         GO EVIDENCE ONTOLOGY
## 4  ENSG00000105323 GO:0005634      IDA       CC
## 5  ENSG00000105323 GO:0005654      IDA       CC
## 6  ENSG00000105323 GO:0005654      TAS       CC
## 14 ENSG00000105323 GO:0030529      IEA       CC
```

```
sapply(as.list(GOTERM[ans$GO]), slot, "Term")
```

```
##                  GO:0005634                  GO:0005654
##                   "nucleus"               "nucleoplasm"
##                  GO:0005654                  GO:0030529
##               "nucleoplasm" "ribonucleoprotein complex"
```

Finally, this information can also be retrieved from on-line databases using the *biomaRt* package [20].

```
library("biomaRt")
ensembl <- useMart("ensembl",dataset="hsapiens_gene_ensembl")
efilter <- "ensembl_gene_id"
eattr <- c("go_id", "name_1006", "namespace_1003")
bmres <- getBM(attributes=eattr, filters = efilter, values = id, mart = ensembl)
bmres[bmres$namespace_1003 == "cellular_component", "name_1006"]
```

```
## [1] "nucleoplasm"               "ribonucleoprotein complex"
## [3] "nucleus"                   "viral nucleocapsid"
```

9 Other packages

9.1 Bioconductor packages

This section provides a complete list of packages available in the relevant *Bioconductor* version 3.3 (as of December 5, 2015) *biocView*[24] categories. Tables 1, 2 and 3 represent the packages for the `Proteomics` (79 packages), `MassSpectrometry` (55 packages) and `MassSpectrometryData` (12 experiment packages) categories.

Package	Title	Version
ASEB	Predict Acetylated Lysine Sites	1.15.0
bioassayR	R library for Bioactivity analysis	1.9.0
biobroom	Turn Bioconductor objects into tidy data frames	1.3.2
BRAIN	Baffling Recursive Algorithm for Isotope distributioN calculations	1.17.0
Cardinal	A mass spectrometry imaging toolbox for statistical analysis	1.3.0
CellNOptR	Training of boolean logic models of signalling networks using prior knowledge networks and perturbation data.	1.17.0
ChemmineOB	R interface to a subset of OpenBabel functionalities	1.9.0
ChemmineR	Cheminformatics Toolkit for R	2.23.0
cisPath	Visualization and management of the protein-protein interaction networks.	1.11.0
cleaver	Cleavage of Polypeptide Sequences	1.9.0
clippda	A package for the clinical proteomic profiling data analysis	1.21.0
CNORdt	Add-on to CellNOptR: Discretized time treatments	1.13.0
CNORfeeder	Integration of CellNOptR to add missing links	1.11.0
CNORode	ODE add-on to CellNOptR	1.13.0
customProDB	Generate customized protein database from NGS data, with a focus on RNA-Seq data, for proteomics search.	1.11.0
DAPAR	Tools for the Differential Analysis of Proteins Abundance with R	1.1.0
deltaGseg	deltaGseg	1.11.0
eiR	Accelerated similarity searching of small molecules	1.11.0
fCI	f-divergence Cutoff Index	1.1.0
fmcsR	Mismatch Tolerant Maximum Common Substructure Searching	1.13.0
GraphPAC	Identification of Mutational Clusters in Proteins via a Graph Theoretical Approach.	1.13.0
hpar	Human Protein Atlas in R	1.13.0
iPAC	Identification of Protein Amino acid Clustering	1.15.0
IPPD	Isotopic peak pattern deconvolution for Protein Mass Spectrometry by template matching	1.19.0
isobar	Analysis and quantitation of isobarically tagged MSMS proteomics data	1.17.0
LPEadj	A correction of the local pooled error (LPE) method to replace the asymptotic variance adjustment with an unbiased adjustment based on sample size.	1.31.0
MassSpecWavelet	Mass spectrum processing by wavelet-based algorithms	1.37.0
MSGFgui	A shiny GUI for MSGFplus	1.5.0
MSGFplus	An interface between R and MS-GF+	1.5.0
msmsEDA	Exploratory Data Analysis of LC-MS/MS data by spectral counts	1.9.0
msmsTests	LC-MS/MS Differential Expression Tests	1.9.0
MSnbase	Base Functions and Classes for MS-based Proteomics	1.19.4
MSnID	Utilities for Exploration and Assessment of Confidence of LC-MSn Proteomics Identifications.	1.5.0
mzID	An mzIdentML parser for R	1.9.0
mzR	parser for netCDF, mzXML, mzData and mzML and mzIdentML files (mass spectrometry data)	2.5.2
PAA	PAA (Protein Array Analyzer)	1.5.1
PAnnBuilder	Protein annotation data package builder	1.35.0
Path2PPI	Prediction of pathway-related protein-protein interaction networks	1.1.1
pathview	a tool set for pathway based data integration and visualization	1.11.2
Pbase	Manipulating and exploring protein and proteomics data	0.11.0
PCpheno	Phenotypes and cellular organizational units	1.33.0
PECA	Probe-level Expression Change Averaging	1.7.0
pepXMLTab	Parsing pepXML files and filter based on peptide FDR.	1.5.0
PGA	An package for identification of novel peptides by customized database derived from RNA-Seq	1.1.0

[24] http://www.bioconductor.org/packages/devel/BiocViews.html

plgem	Detect differential expression in microarray and proteomics datasets with the Power Law Global Error Model (PLGEM)	1.43.0
PLPE	Local Pooled Error Test for Differential Expression with Paired High-throughput Data	1.31.0
ppiStats	Protein-Protein Interaction Statistical Package	1.37.0
proBAMr	Generating SAM file for PSMs in shotgun proteomics data	1.5.0
PROcess	Ciphergen SELDI-TOF Processing	1.47.0
procoil	Prediction of Oligomerization of Coiled Coil Proteins	1.21.0
ProCoNA	Protein co-expression network analysis (ProCoNA).	1.9.0
pRoloc	A unifying bioinformatics framework for spatial proteomics	1.11.0
pRolocGUI	Interactive visualisation of spatial proteomics data	1.5.0
Prostar	Provides a GUI for DAPAR	1.1.1
prot2D	Statistical Tools for volume data from 2D Gel Electrophoresis	1.9.0
ProteomicsAnnotationHubData	Transform public proteomics data resources into Bioconductor Data Structures	1.1.0
proteoQC	An R package for proteomics data quality control	1.7.0
ProtGenerics	S4 generic functions for Bioconductor proteomics infrastructure	1.3.3
Pviz	Peptide Annotation and Data Visualization using Gviz	1.5.0
qcmetrics	A Framework for Quality Control	1.9.1
QuartPAC	Identification of mutational clusters in protein quaternary structures.	1.3.0
rain	Rhythmicity Analysis Incorporating Non-parametric Methods	1.5.0
RCASPAR	A package for survival time prediction based on a piecewise baseline hazard Cox regression model.	1.17.0
Rchemcpp	Similarity measures for chemical compounds	2.9.0
Rcpi	Toolkit for Compound-Protein Interaction in Drug Discovery	1.7.0
ropls	PCA, PLS(-DA) and OPLS(-DA) for multivariate analysis and feature selection of omics data	1.3.4
RpsiXML	R interface to PSI-MI 2.5 files	2.13.0
rpx	R Interface to the ProteomeXchange Repository	1.7.0
rTANDEM	Interfaces the tandem protein identification algorithm in R	1.11.0
sapFinder	A package for variant peptides detection and visualization in shotgun proteomics.	1.9.0
ScISI	In Silico Interactome	1.43.0
shinyTANDEM	Provides a GUI for rTANDEM	1.9.0
SLGI	Synthetic Lethal Genetic Interaction	1.31.0
SpacePAC	Identification of Mutational Clusters in 3D Protein Space via Simulation.	1.9.0
specL	specL - Prepare Peptide Spectrum Matches for Use in Targeted Proteomics	1.5.0
spliceSites	Manages align gap positions from RNA-seq data	1.9.0
SWATH2stats	Transform and Filter SWATH Data for Statistical Packages	1.1.5
synapter	Label-free data analysis pipeline for optimal identification and quantitation	1.13.0
TPP	Analyze thermal proteome profiling (TPP) experiments	2.1.2

Table 1: Packages available under the `Proteomics` *biocViews* category.

Package	Title	Version
apComplex	Estimate protein complex membership using AP-MS protein data	2.37.0
BRAIN	Baffling Recursive Algorithm for Isotope distributioN calculations	1.17.0
CAMERA	Collection of annotation related methods for mass spectrometry data	1.27.0
Cardinal	A mass spectrometry imaging toolbox for statistical analysis	1.3.0
cosmiq	cosmiq - COmbining Single Masses Into Quantities	1.5.0
customProDB	Generate customized protein database from NGS data, with a focus on RNA-Seq data, for proteomics search.	1.11.0
cytofkit	cytofkit: an integrated analysis pipeline for mass cytometry data	1.3.0
DAPAR	Tools for the Differential Analysis of Proteins Abundance with R	1.1.0
flagme	Analysis of Metabolomics GC/MS Data	1.27.1
gaga	GaGa hierarchical model for high-throughput data analysis	2.17.0
iontree	Data management and analysis of ion trees from ion-trap mass spectrometry	1.17.0
isobar	Analysis and quantitation of isobarically tagged MSMS proteomics data	1.17.0
MAIT	Statistical Analysis of Metabolomic Data	1.5.0
MassArray	Analytical Tools for MassArray Data	1.23.0
MassSpecWavelet	Mass spectrum processing by wavelet-based algorithms	1.37.0
Metab	Metab: An R Package for a High-Throughput Analysis of Metabolomics Data Generated by GC-MS.	1.5.0
metabomxtr	A package to run mixture models for truncated metabolomics data with normal or lognormal distributions	1.5.0
metaMS	MS-based metabolomics annotation pipeline	1.7.0
metaX	An R package for metabolomic data analysis	1.3.5
MSGFgui	A shiny GUI for MSGFplus	1.5.0
MSGFplus	An interface between R and MS-GF+	1.5.0
msmsEDA	Exploratory Data Analysis of LC-MS/MS data by spectral counts	1.9.0
msmsTests	LC-MS/MS Differential Expression Tests	1.9.0
MSnbase	Base Functions and Classes for MS-based Proteomics	1.19.4
MSnID	Utilities for Exploration and Assessment of Confidence of LC-MSn Proteomics Identifications.	1.5.0
mzID	An mzIdentML parser for R	1.9.0
mzR	parser for netCDF, mzXML, mzData and mzML and mzIdentML files (mass spectrometry data)	2.5.2
PAPi	Predict metabolic pathway activity based on metabolomics data	1.11.0
Pbase	Manipulating and exploring protein and proteomics data	0.11.0
pepXMLTab	Parsing pepXML files and filter based on peptide FDR.	1.5.0
PGA	An package for identification of novel peptides by customized database derived from RNA-Seq	1.1.0
plgem	Detect differential expression in microarray and proteomics datasets with the Power Law Global Error Model (PLGEM)	1.43.0
proBAMr	Generating SAM file for PSMs in shotgun proteomics data	1.5.0

PROcess	Ciphergen SELDI-TOF Processing	1.47.0
pRoloc	A unifying bioinformatics framework for spatial proteomics	1.11.0
Prostar	Provides a GUI for DAPAR	1.1.1
proteoQC	An R package for proteomics data quality control	1.7.0
ProtGenerics	S4 generic functions for Bioconductor proteomics infrastructure	1.3.3
qcmetrics	A Framework for Quality Control	1.9.1
Rdisop	Decomposition of Isotopic Patterns	1.31.0
Risa	Converting experimental metadata from ISA-tab into Bioconductor data structures	1.13.0
RMassBank	Workflow to process tandem MS files and build MassBank records	1.13.0
rols	An R interface to the Ontology Lookup Service	1.13.0
ropls	PCA, PLS(-DA) and OPLS(-DA) for multivariate analysis and feature selection of omics data	1.3.4
rpx	R Interface to the ProteomeXchange Repository	1.7.0
rTANDEM	Interfaces the tandem protein identification algorithm in R	1.11.0
sapFinder	A package for variant peptides detection and visualization in shotgun proteomics.	1.9.0
shinyTANDEM	Provides a GUI for rTANDEM	1.9.0
SIMAT	GC-SIM-MS data processing and alaysis tool	1.3.0
specL	specL - Prepare Peptide Spectrum Matches for Use in Targeted Proteomics	1.5.0
SWATH2stats	Transform and Filter SWATH Data for Statistical Packages	1.1.5
synapter	Label-free data analysis pipeline for optimal identification and quantitation	1.13.0
TargetSearch	A package for the analysis of GC-MS metabolite profiling data	1.27.0
TPP	Analyze thermal proteome profiling (TPP) experiments	2.1.2
xcms	LC/MS and GC/MS Data Analysis	1.47.0

Table 2: Packages available under the `MassSpectrometry` *biocViews* category.

Package	Title	Version
CardinalWorkflows	Datasets and workflows for the Cardinal mass spectrometry imaging package	1.3.0
faahKO	Saghatelian et al. (2004) FAAH knockout LC/MS data	1.11.0
gcspikelite	Spike-in data for GC/MS data and methods within flagme	1.9.0
iontreeData	Data provided to show the usage of functions in iontree package	1.7.0
metaMSdata	Example CDF data for the metaMS package	1.7.0
msdata	Various Mass Spectrometry raw data example files	0.9.0
mtbls2	MetaboLights MTBLS2: Comparative LC/MS-based profiling of silver nitrate-treated Arabidopsis thaliana leaves of wild-type and cyp79B2 cyp79B3 double knockout plants. Bttcher et al. (2004)	1.1.0
ProData	SELDI-TOF data of Breast cancer samples	1.9.0
pRolocdata	Data accompanying the pRoloc package	1.9.1
RforProteomics	Companion package to the 'Using R and Bioconductor for proteomics data analysis' publication	1.9.0
RMassBankData	Test dataset for RMassBank	1.9.0
synapterdata	Data accompanying the synapter package	1.9.0

Table 3: Experimental Packages available under the `MassSpectrometryData` *biocViews* category.

The tables can easily be generated with the `proteomicsPackages`, `massSpectrometryPackages` and `massSpectrometryDataPackag` functions. The respective package tables can then be interactively explored using the `display` function.

```
pp <- proteomicsPackages()
display(pp)
```

9.2 Other CRAN packages

The CRAN task view on Chemometrics and Computational p Physics[25] is another useful ressource listing 81 packages, including a set of packages for mass spectrometry and proteomics, some of which are illustrated in this document.

MALDIquant provides tools for quantitative analysis of MALDI-TOF mass spectrometry data, with support for baseline

[25] http://cran.r-project.org/web/views/ChemPhys.html

correction, peak detection and plotting of mass spectra (http://cran.r-project.org/web/packages/MALDIquant/index.html).

OrgMassSpecR is for organic/biological mass spectrometry, with a focus on graphical display, quantification using stable isotope dilution, and protein hydrogen/deuterium exchange experiments (http://cran.r-project.org/web/packages/OrgMassSpecR/index.html).

FTICRMS provides functions for Analyzing Fourier Transform-Ion Cyclotron Resonance Mass Spectrometry Data (http://cran.r-project.org/web/packages/FTICRMS/index.html).

titan provides a GUI to analyze mass spectrometric data on the relative abundance of two substances from a titration series (http://cran.r-project.org/web/packages/titan/index.html).

digeR provides a GUI interface for analysing 2D DIGE data. It allows to perform correlation analysis, score plot, classification, feature selection and power analysis for 2D DIGE experiment data. (http://cran.r-project.org/web/packages/digeR/index.html)

protViz helps with quality checks, visualizations and analysis of mass spectrometry data, coming from proteomics experiments. The package is developed, tested and used at the Functional Genomics Center Zurich. (http://cran.r-project.org/web/packages/protViz/index.html)

Suggestions for additional *R* packages are welcome and will be added to the vignette. Please send suggestions and possibly a short description and/or a example utilisation with code to lg390@cam.ac.uk. The only requirement is that the package must be available on an official package channel (CRAN, *Bioconductor*, R-forge, Omegahat), i.e. not only available through a personal web page.

10 Session information

All software and respective versions used in this document, as returned by `sessionInfo()` are detailed below.

- R Under development (unstable) (2015-10-23 r69563), `x86_64-pc-linux-gnu`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.33.1, Biobase 2.31.0, BiocGenerics 0.17.1, BiocInstaller 1.21.2, BiocParallel 1.5.0, Biostrings 2.39.2, bitops 1.0-6, BRAIN 1.17.0, cleaver 1.9.0, data.table 1.9.6, DBI 0.3.1, digest 0.6.8, dplyr 0.4.3, ggplot2 1.0.1, GO.db 3.2.2, Heatplus 2.17.0, hpar 1.13.0, IPPD 1.19.0, IRanges 2.5.8, isobar 1.17.0, jsonlite 0.9.19, knitr 1.11, lattice 0.20-33, MALDIquant 1.14, MALDIquantForeign 0.10, MASS 7.3-45, Matrix 1.2-3, msdata 0.9.0, MSGFgui 1.5.0, MSGFplus 1.5.0, msmsEDA 1.9.0, msmsTests 1.9.0, MSnbase 1.19.4, MSnID 1.5.0, mzID 1.9.0, mzR 2.5.2, org.Hs.eg.db 3.2.3, OrgMassSpecR 0.4-4, PolynomF 0.94, ProtGenerics 1.3.3, RColorBrewer 1.1-2, Rcpp 0.12.2, RcppClassic 0.9.6, Rdisop 1.31.0, reshape2 1.4.1, RforProteomics 1.9.2, rJava 0.9-7, rols 1.13.0, rpx 1.7.1, RSQLite 1.0.0, rTANDEM 1.11.0, S4Vectors 0.9.10, xlsx 0.5.7, xlsxjars 0.6.1, XML 3.98-1.3, xtable 1.8-0, XVector 0.11.1
- Loaded via a namespace (and not attached): affy 1.49.0, affyio 1.41.0, annotate 1.49.0, assertthat 0.1, base64enc 0.1-3, BiocStyle 1.9.2, biocViews 1.39.2, Category 2.37.0, caTools 1.17.1, chron 2.3-47, codetools 0.2-14, colorspace 1.2-6, compiler 3.3.0, curl 0.9.4, distr 2.5.3, doParallel 1.0.10, edgeR 3.13.0, evaluate 0.8, foreach 1.4.3, formatR 1.2.1, futile.logger 1.4.1, futile.options 1.0.0, gdata 2.17.0, genefilter 1.53.0, gplots 2.17.0, graph 1.49.1, grid 3.3.0, gridSVG 1.5-0, GSEABase 1.33.0, gtable 0.1.2, gtools 3.5.0, highr 0.5.1, htmltools 0.2.6, httpuv 1.3.3, impute 1.45.0, interactiveDisplay 1.9.0, interactiveDisplayBase 1.9.0, iterators 1.0.8, KernSmooth 2.23-15, labeling 0.3, lambda.r 1.1.7, lazyeval 0.1.10, limma 3.27.5, magrittr 1.5, mime 0.4, munsell 0.4.2, pcaMethods 1.61.0, plyr 1.8.3, preprocessCore 1.33.0, proto 0.3-10, qvalue 2.3.0, R6 2.1.1, RBGL 1.47.0, R.cache 0.12.0, RCurl 1.95-4.7, readBrukerFlexData 1.8.2, readMzXmlData 2.8.1, RJSONIO 1.3-0, R.methodsS3 1.7.0, R.oo 1.19.0, RUnit 0.4.31, R.utils 2.1.0, scales 0.3.0, sfsmisc 1.0-28, shiny 0.12.2, shinyFiles 0.6.0, splines 3.3.0, SSOAP 0.8-0, startupmsg 0.9, stringi 1.0-1, stringr 1.0.0, survival 2.38-3, SweaveListingUtils 0.6.2, tools 3.3.0, vsn 3.39.0, XMLSchema 0.7-2, zlibbioc 1.17.0

References

[1] Laurent Gatto and Andy Christoforou. Using R and bioconductor for proteomics data analysis. *BBA - Proteins and Proteomics*, 2013. doi:10.1016/j.bbapap.2013.04.032.

[2] L Gatto, L M Breckels, T Naake, and S Gibb. Visualisation of proteomics data using R and Bioconductor. *Proteomics*, Feb 2015. doi:10.1002/pmic.201400392.

[3] Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. H. Yang, and Jianhua Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, 5(10):–80, 2004. URL: http://dx.doi.org/10.1186/gb-2004-5-10-r80, doi:10.1186/gb-2004-5-10-r80.

[4] M C Chambers, B Maclean, R Burke, D Amodei, D L Ruderman, S Neumann, L Gatto, B Fischer, B Pratt, J Egertson, K Hoff, D Kessner, N Tasman, N Shulman, B Frewen, T A Baker, M Y Brusniak, C Paulse, D Creasy, L Flashner, K Kani, C Moulding, S L Seymour, L M Nuwaysir, B Lefebvre, F Kuhlmann, J Roark, P Rainer, S Detlev, T Hemenway, A Huhmer, J Langridge, B Connolly, T Chadick, K Holly, J Eckels, E W Deutsch, R L Moritz, J E Katz, D B Agus, M MacCoss, D L Tabb, and P Mallick. A cross-platform toolkit for mass spectrometry and proteomics. *Nat Biotechnol*, 30(10):918–20, Oct 2012. doi:10.1038/nbt.2377.

[5] L Gatto and K S Lilley. MSnbase – an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation. *Bioinformatics*, 28(2):288–9, Jan 2012. doi:10.1093/bioinformatics/btr645.

[6] Alvaro Cuadros-Inostroza, Camila Caldana, Henning Redestig, Jan Lisec, Hugo Pena-Cortes, Lothar Willmitzer, and Matthew A Hannah. TargetSearch - a Bioconductor package for the efficient pre-processing of GC-MS metabolite profiling data. *BMC Bioinformatics*, 10:428, 2009. doi:10.1186/1471-2105-10-428.

[7] C A Smith, E J Want, G O'Maille, R Abagyan, and G Siuzdak. XCMS: processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification. *Anal Chem*, 78(3):779–87, Feb 2006. doi:10.1021/ac051437y.

[8] H P Benton, D M Wong, S A Trauger, and G Siuzdak. XCMS2: processing tandem mass spectrometry data for metabolite identification and structural characterization. *Anal Chem*, 80(16):6382–9, Aug 2008. doi:10.1021/ac800795f.

[9] R Tautenhahn, C Böttcher, and S Neumann. Highly sensitive feature detection for high resolution LC/MS. *BMC Bioinformatics*, 9:504, 2008. doi:10.1186/1471-2105-9-504.

[10] Juan A Vizcaino, Eric W Deutsch, Rui Wang, Attila Csordas, Florian Reisinger, Daniel Rios, Jose A Dianes, Zhi Sun, Terry Farrah, Nuno Bandeira, Pierre-Alain Binz, Ioannis Xenarios, Martin Eisenacher, Gerhard Mayer, Laurent Gatto, Alex Campos, Robert J Chalkley, Hans-Joachim Kraus, Juan Pablo Albar, Salvador Martinez-Bartolome, Rolf Apweiler, Gilbert S Omenn, Lennart Martens, Andrew R Jones, and Henning Hermjakob. ProteomeXchange provides globally coordinated proteomics data submission and dissemination. *Nat Biotech*, 32:223–226, 2014. doi:10.1038/nbt.2839.

[11] S Gibb and K Strimmer. MALDIquant: a versatile R package for the analysis of mass spectrometry data. *Bioinformatics*, 28(17):2270–1, Sep 2012. doi:10.1093/bioinformatics/bts447.

[12] F P Breitwieser, A Müller, L Dayon, T Köcher, A Hainard, P Pichler, U Schmidt-Erfurth, G Superti-Furga, J C Sanchez, K Mechtler, K L Bennett, and J Colinge. General statistical modeling of data from protein relative expression isobaric tags. *J Proteome Res*, 10(6):2758–66, Jun 2011. doi:10.1021/pr1012784.

[13] Nicholas J. Bond, Pavel V. Shliaha, Kathryn S. Lilley, and Laurent Gatto. Improving qualitative and quantitative performance for label free proteomics. *J. Proteome Res.*, 2013. doi:10.1021/pr300776t.

[14] R Craig and R C Beavis. Tandem: matching proteins with tandem mass spectra. *Bioinformatics*, 20(9):1466–7, Jun 2004. doi:10.1093/bioinformatics/bth092.

[15] Sangtae Kim, Nitin Gupta, and Pavel A Pevzner. Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *Journal of Proteome Research*, 7(8):3354–3363, August 2008.

[16] Sangtae Kim, Nikolai Mischerikow, Nuno Bandeira, J Daniel Navarro, Louis Wich, Shabaz Mohammed, Albert J R Heck, and Pavel A Pevzner. The generating function of CID, ETD, and CID/ETD pairs of tandem mass spectra: applications to database search. *Molecular & Cellular Proteomics: MCP*, 9(12):2840–2852, December 2010.

[17] Paul D. Piehowski, Vladislav A. Petyuk, John D. Sandoval, Kristin E. Burnum, Gary R. Kiebel, Matthew E. Monroe, Gordon A. Anderson, David G Camp, 2nd, and Richard D. Smith. Steps: a grid search methodology for optimized peptide identification filtering of ms/ms database search results. *Proteomics*, 13(5):766–770, Mar 2013. URL: http://dx.doi.org/10.1002/pmic.201200096, doi:10.1002/pmic.201200096.

[18] Mathias Uhlén, Erik Björling, Charlotta Agaton, Cristina Al-Khalili A. Szigyarto, Bahram Amini, Elisabet Andersen, Ann-Catrin C. Andersson, Pia Angelidou, Anna Asplund, Caroline Asplund, Lisa Berglund, Kristina Bergström, Harry Brumer, Dijana Cerjan, Marica Ekström, Adila Elobeid, Cecilia Eriksson, Linn Fagerberg, Ronny Falk, Jenny Fall, Mattias Forsberg, Marcus Gry G. Björklund, Kristoffer Gumbel, Asif Halimi, Inga Hallin, Carl Hamsten, Marianne Hansson, My Hedhammar, Görel Hercules, Caroline Kampf, Karin Larsson, Mats Lindskog, Wald Lodewyckx, Jan Lund, Joakim Lundeberg, Kristina Magnusson, Erik Malm, Peter Nilsson, Jenny Odling, Per Oksvold, Ingmarie Olsson, Emma Oster, Jenny Ottosson, Linda Paavilainen, Anja Persson, Rebecca Rimini, Johan Rockberg, Marcus Runeson, Asa Sivertsson, Anna Sköllermo, Johanna Steen, Maria Stenvall, Fredrik Sterky, Sara Strömberg, Mårten Sundberg, Hanna Tegel, Samuel Tourle, Eva Wahlund, Annelie Waldén, Jinghong Wan, Henrik Wernérus, Joakim Westberg, Kenneth Wester, Ulla Wrethagen, Lan Lan L. Xu, Sophia Hober, and Fredrik Pontén. A human protein atlas for normal and cancer tissues based on antibody proteomics. *Molecular & cellular proteomics : MCP*, 4(12):1920–1932, December 2005. URL: http://dx.doi.org/10.1074/mcp.M500279-MCP200, doi:10.1074/mcp.M500279-MCP200.

[19] Mathias Uhlen, Per Oksvold, Linn Fagerberg, Emma Lundberg, Kalle Jonasson, Mattias Forsberg, Martin Zwahlen, Caroline Kampf, Kenneth Wester, Sophia Hober, Henrik Wernerus, Lisa Björling, and Fredrik Ponten. Towards a knowledge-based Human Protein Atlas. *Nature biotechnology*, 28(12):1248–1250, December 2010. URL: http://dx.doi.org/10.1038/nbt1210-1248, doi:10.1038/nbt1210-1248.

[20] S Durinck, Y Moreau, A Kasprzyk, S Davis, B De Moor, A Brazma, and W Huber. Biomart and bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, 21(16):3439–40, Aug 2005. doi:10.1093/bioinformatics/bti525.