Using R and Bioconductor for proteomics data analysis.

Laurent Gatto
lg390@cam.ac.uk
Cambridge Center for Proteomics
University of Cambridge

September 28, 2012

Abstract

This vignette shows and executes the code presented in the manuscript *Using R for proteomics data analysis*.

Keywords: bioinformatics, proteomics, mass spectrometry, tutorial

1 Introduction

1.1 General R ressources

The reader is expected to have basic R knowledge to find to 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 introductionary tutorials in the contributed documentation section.

1.2 Getting help

R has several mailing lists[1]. The most relevant here being the main `R-help` list, *for discussion about problem and solutions using R* . This one is for general R content and is not suitable for bioinformatics or proteomics questions.

Bioconductor also offers several mailing lists[2] dedicated to bioinformatics matters and Bioconductor packages. The main `bioconductor` list is the most relevant one. It is possible to post[3] questions without subscribing to the list.

It is important 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 docuement, on page 30). It the question involves some code, make sure to isolate the relevant portion and filed it with your question and try to make your code/example reproducible[4].

All lists have browsable archives.

1.3 Installation

Since the package is not yet in Bioconductor, it is not yet possible to use its automatice installation and update framework. An ad hoc script is however available to facilitate installation of RforProteomics and all dependencies. The `source_url` function from the devtools package is used to source the R code over `https`. Simply open R and type

```
> ## only if 'devtools' is not installed
> install.packages("devtools")
> library("devtools")
> source_url("https://raw.github.com/lgatto/RforProteomics/master/inst/scripts/insta
```

The script installs missing dependencies and then RforProteomics, which can then be loaded with

```
> library("RforProteomics")

This is the 'RforProteomics' version 0.1.0.
Run 'RforProtemics()' in R or visit
'http://lgatto.github.com/RforProteomics/' to get started.
```

1.4 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

[1] http://www.r-project.org/mail.html
[2] http://bioconductor.org/help/mailing-list/
[3] http://bioconductor.org/help/mailing-list/mailform/
[4] https://github.com/hadley/devtools/wiki/Reproducibility

text file for better interaction with R (using ESS[5] for emacs or RStudio[6]) and to easily modify and explore it. This can be achieved with the `Stangle` function. It only need the Sweave source of this document, 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 <- dir(system.file(package = "RforProteomics", dir = "doc"), full.name = T
+     pattern = "Rnw$")
> ## produces an R file in the working directory
> Stangle(rnwfile)

Writing to file RforProteomics.R

> dir(pattern = "RforProteomics.R")

[1] "RforProteomics.R"   "RforProteomics.Rnw"
```

1.5 Prepare the working environment

The packages that we will depend on to execture 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/ouput

2.1 The mzR package

This code chunk, taken mainly 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 be applicatble to an `mzXML` of `mzData` file.

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

[5] http://ess.r-project.org/
[6] 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
> fileName(mz)

[1] "/home/lgatto/R/x86_64-unknown-linux-gnu-library/2.15/msdata/microtofq/MM14.mzMl

> isInitialized(mz)

[1] TRUE

> runInfo(mz)

$scanCount
[1] 112

$lowMz
[1] 0

$highMz
[1] 0

$dStartTime
[1] 270.3

$dEndTime
[1] 307.7

$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)
```

3 Raw data abstraction with MSnExp objects

```
> mzXML <- dir(system.file(package = "MSnbase", dir = "extdata"), full.name = TRUE,
+     pattern = "mzXML$")
> 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 2017
 MSn retention times: 25:1 - 25:2 minutes
- - - Processing information - - -
Data loaded: Fri Sep 28 01:24:04 2012
 MSnbase version: 1.5.25
- - - Meta data  - - -
phenoData
  rowNames: 1
  varLabels: sampleNames fileNumbers
  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 sigle spectrum
> raw[[3]]

Object of class "Spectrum2"
 Precursor: 645.4
 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)
```

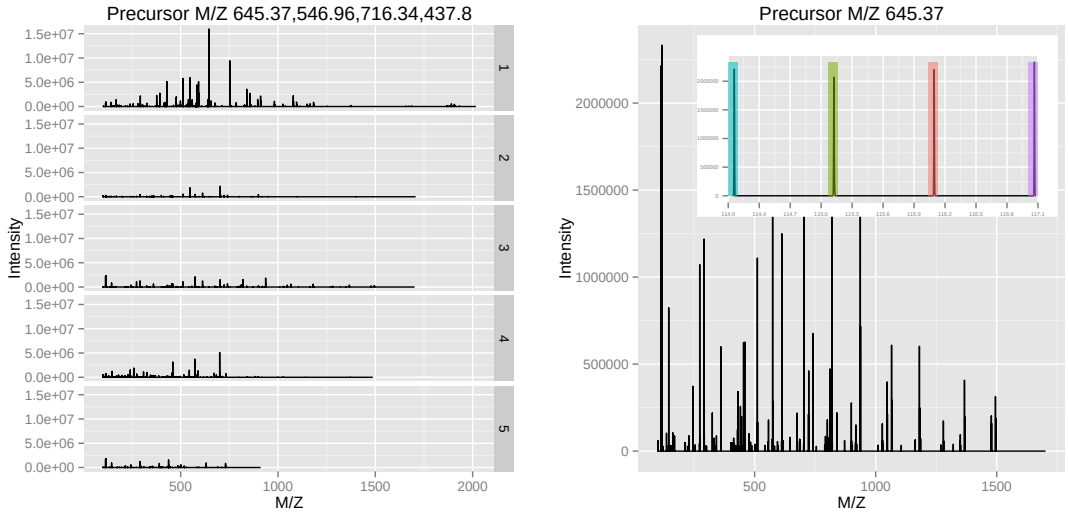


Figure 1: The `plot` method can be used on experiments, i.e. spectrum collections (left), or individual spectra (right).

3.1 `mgf` read/write support

See `readMgfData` and `writeMgfData` in MSnbase.

4 Quantitative proteomics

As an running example throughout this document, we will use the PXD000001 data set. The code chunk below first downloads this data file from the ProteomeXchange server using the getPXD000001mzXML function from the RforProteomics package.

4.1 The mzTab format

```
> mztab <- getPXD000001mzTab()
> mztab  ## the mzTab file name

[1] "./F063721.dat-mztab.txt"

> ## Load mzTabs's peptide data
> qnt <- readMzTabData(mztab, what = "PEP")

Detected a metadata section
Detected a peptide section

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

  TMT6.126 TMT6.127 TMT6.128 TMT6.129 TMT6.130 TMT6.131
1 10630132 11238708 12424917 10997763  9928972 10398534
2 11105690 12403253 13160903 12229367 11061660 10131218
3  1183431  1322371  1599088  1243715  1306602  1159064
4  5384958  5508454  6883086  6136023  5626680  5213771
5 18033537 17926487 21052620 19810368 17381162 17268329
6  9873585 10299931 11142071 10258214  9664315  9518271

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

Combined 1528 features into 404 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)
```

```
> 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 = 0.8, col = cls)
```

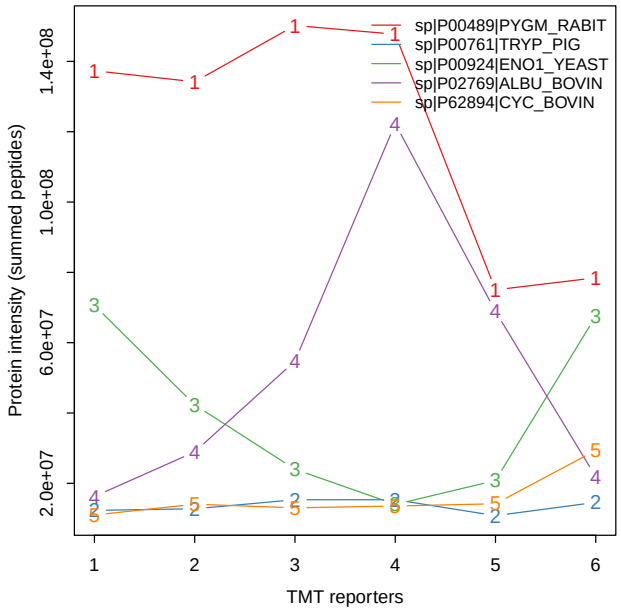


Figure 2: Protein quantitation data.

```
> 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")
> 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)
```

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

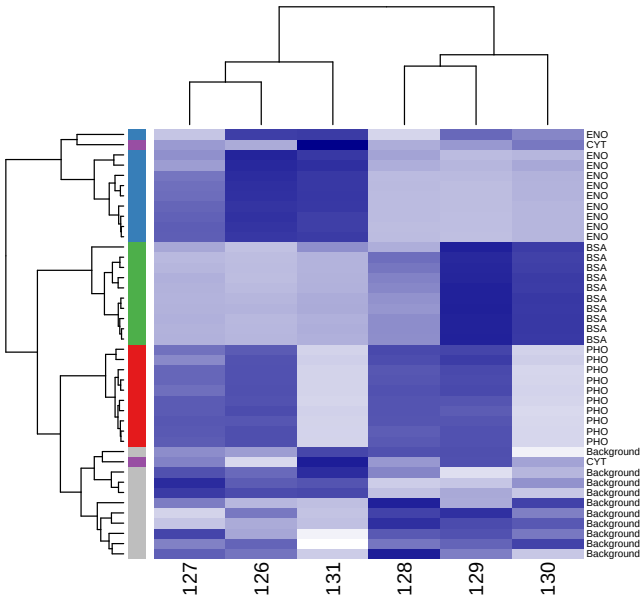


Figure 3: A heatmap.

```
> 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")
```

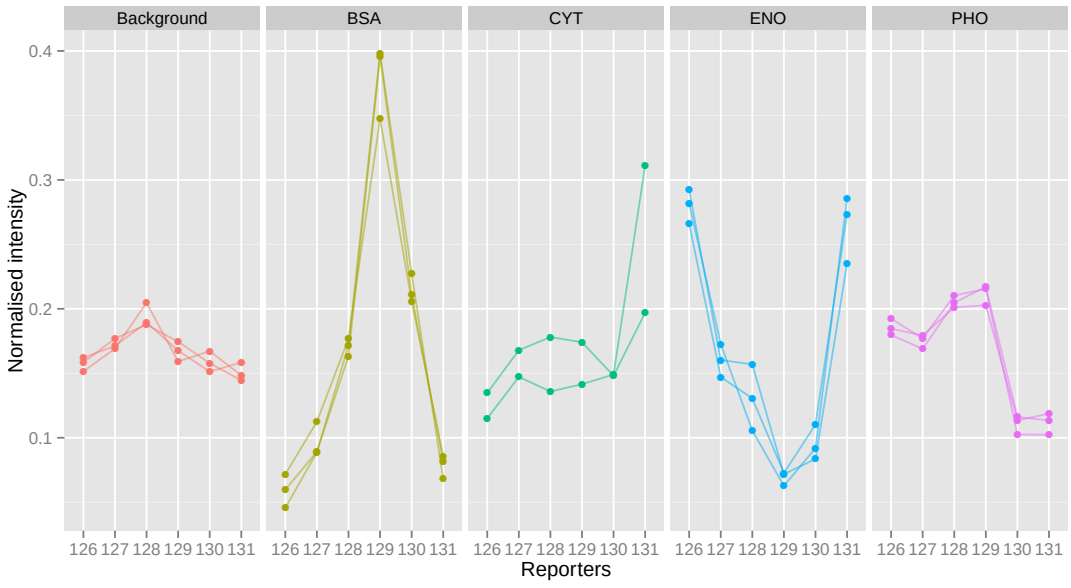


Figure 4: Spikes plot using ggplot2.

4.2 Working with raw data

```
> mzxml <- getPXD000001mzXML()
> rawms <- readMSData(mzxml, centroided = TRUE, verbose = FALSE)
> qntms <- quantify(rawms, reporters = TMT7, method = "max", verbose = FALSE)

Loading required package:  parallel

Attaching package:  'parallel'
The following object(s) are masked from 'package:multicore':

mclapply, mcparallel, pvec

>
> 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("")

Scale for 'x' is already present.  Adding another scale for 'x', which
will replace the existing scale.
Warning:  Removed 2 rows containing missing values (geom_text).
```

```
> mzp

Warning:  Removed 2 rows containing missing values (geom_text).
```

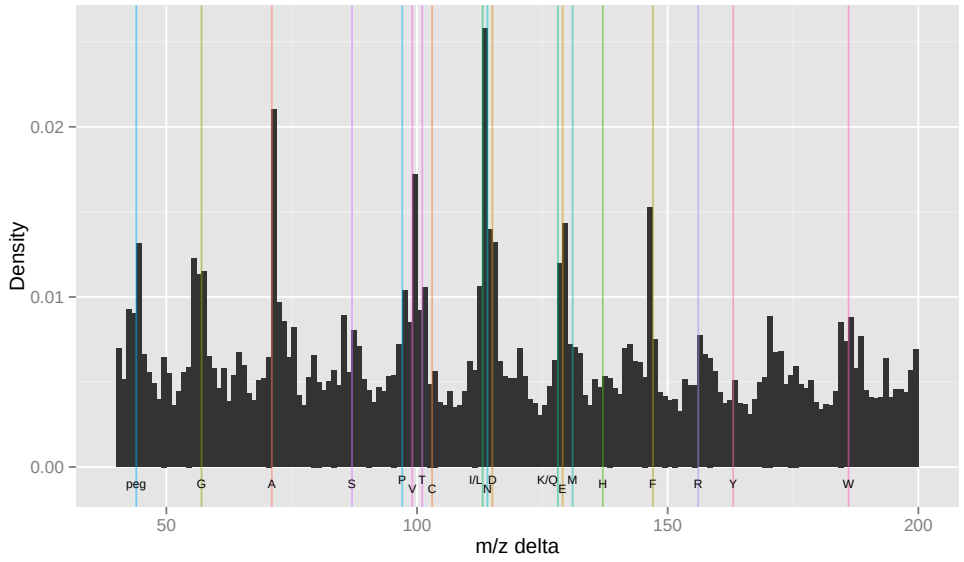


Figure 5: A m/z delta plot.

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

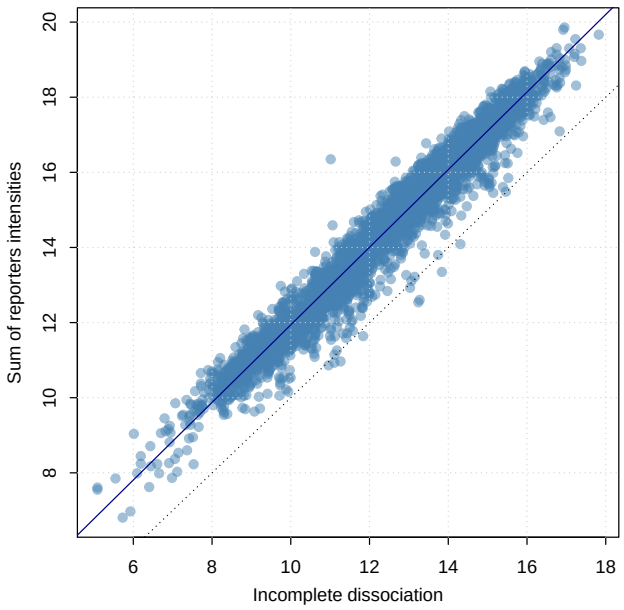


Figure 6: Incomplete dissociation.

```
> MAplot(qnt[, c(4, 2)], cex = 0.9, col = cls, pch = pch, show.statistics = FALSE)
> abline(lm(Signal ~ Incomplete, data = d), col = "darkblue")
```

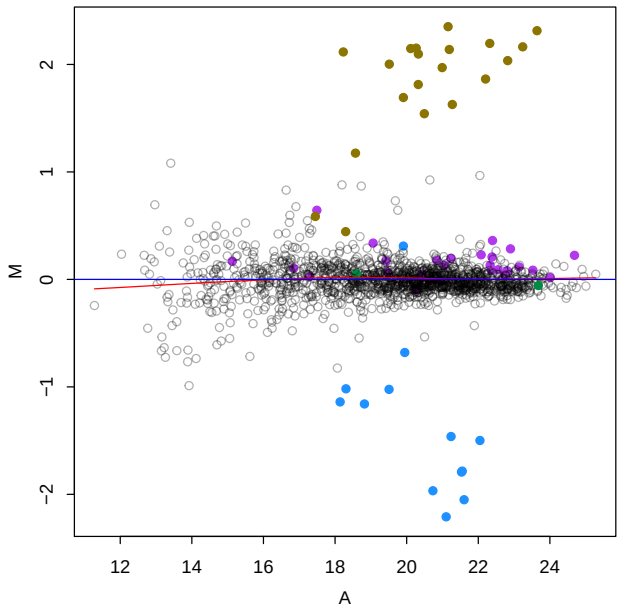


Figure 7: MAplot on an `MSnSet` instance.

4.3 The MALDIquant package

This section illustrates some of MALDIquant's data processing capabilities. The code is taken from the `processing-peaks.R` script downloaded from the package homepage[7].

Loading the data

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

[1] "0_A1" "0_A2"

> sA1 <- mqReadBrukerFlex(datapath)
> # in the following we use only the first spectrum
> s <- sA1[[1]]
>
> summary(mass(s))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1000    2370    4330    4720    6870   10000

> summary(intensity(s))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      4     180    1560    2840    4660   32600

> head(as.matrix(s))

       mass intensity
[1,]  999.9     11278
[2,] 1000.1     11350
[3,] 1000.3     10879
[4,] 1000.5     10684
[5,] 1000.7     10740
[6,] 1000.9     10947
```

Preprocessing

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

```
> plot(s)
```

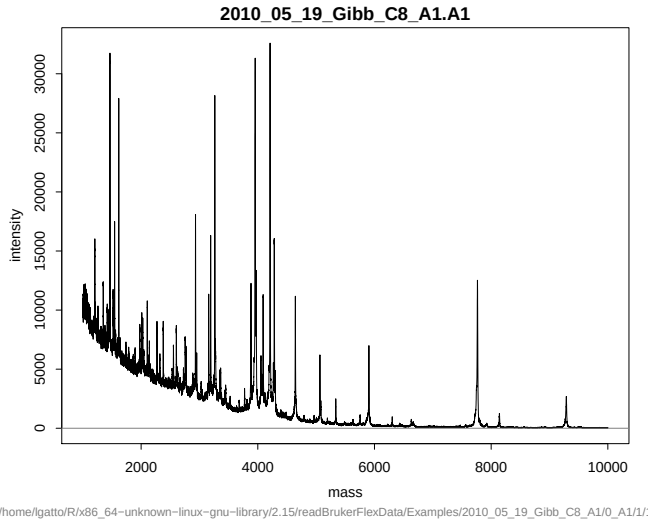


Figure 8: Spectrum plotting in MALDIquant.

```
> ## sqrt transform (for variance stabilization)
> s2 <- transformIntensity(s, fun = 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
Name                     : 2010_05_19_Gibb_C8_A1.A1
File                     : /home/lgatto/R/x86_64-unknown-linux-gnu-library/2.15/read

>
> ## smoothing
> simpleSmooth <- function(y) {
+     return(filter(y, rep(1, 5)/5, sides = 2))  # 5 point moving average
+ }
>
> s3 <- transformIntensity(s2, simpleSmooth)
> s3

S4 class type            : MassSpectrum
Number of m/z values     : 22427
Range of m/z values      : 1000.324 - 10000.705
Range of intensity values: 3.606e+00 - 1.792e+02
Name                     : 2010_05_19_Gibb_C8_A1.A1
File                     : /home/lgatto/R/x86_64-unknown-linux-gnu-library/2.15/read
```

```
> length(s2)  # 22431

[1] 22431

> length(s3)  # 22427 - at both ends data points have been removed

[1] 22427

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

S4 class type            : MassSpectrum
Number of m/z values     : 22427
Range of m/z values      : 1000.324 - 10000.705
Range of intensity values: 0e+00 - 1.414e+02
Name                     : 2010_05_19_Gibb_C8_A1.A1
File                     : /home/lgatto/R/x86_64-unknown-linux-gnu-library/2.15/reac
```

Peak picking

```
> ## peak picking
> p <- detectPeaks(s4)
> length(p)  # 181

[1] 181

> 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)
```

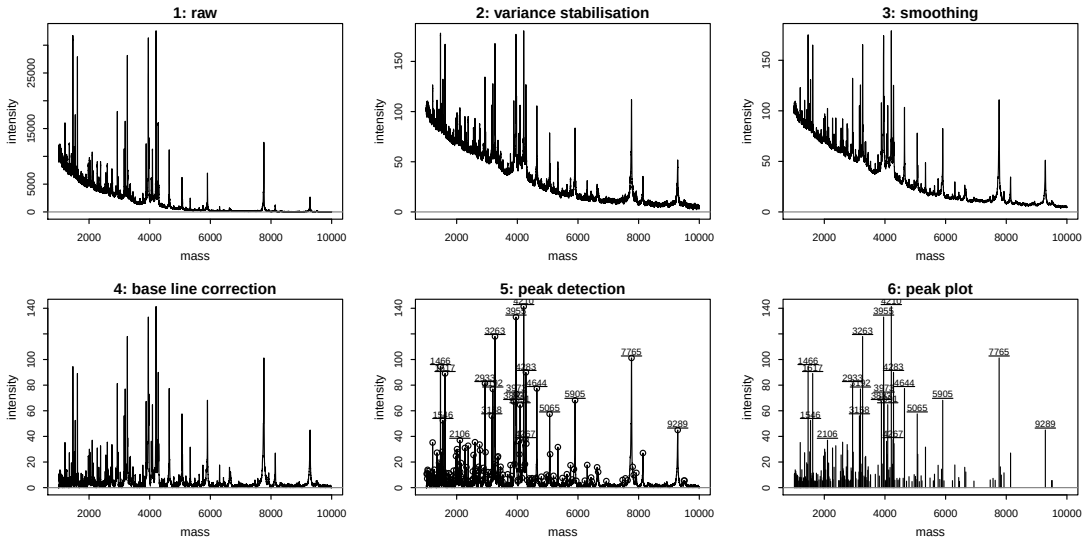


Figure 9: Spectrum plotting in MALDIquant.

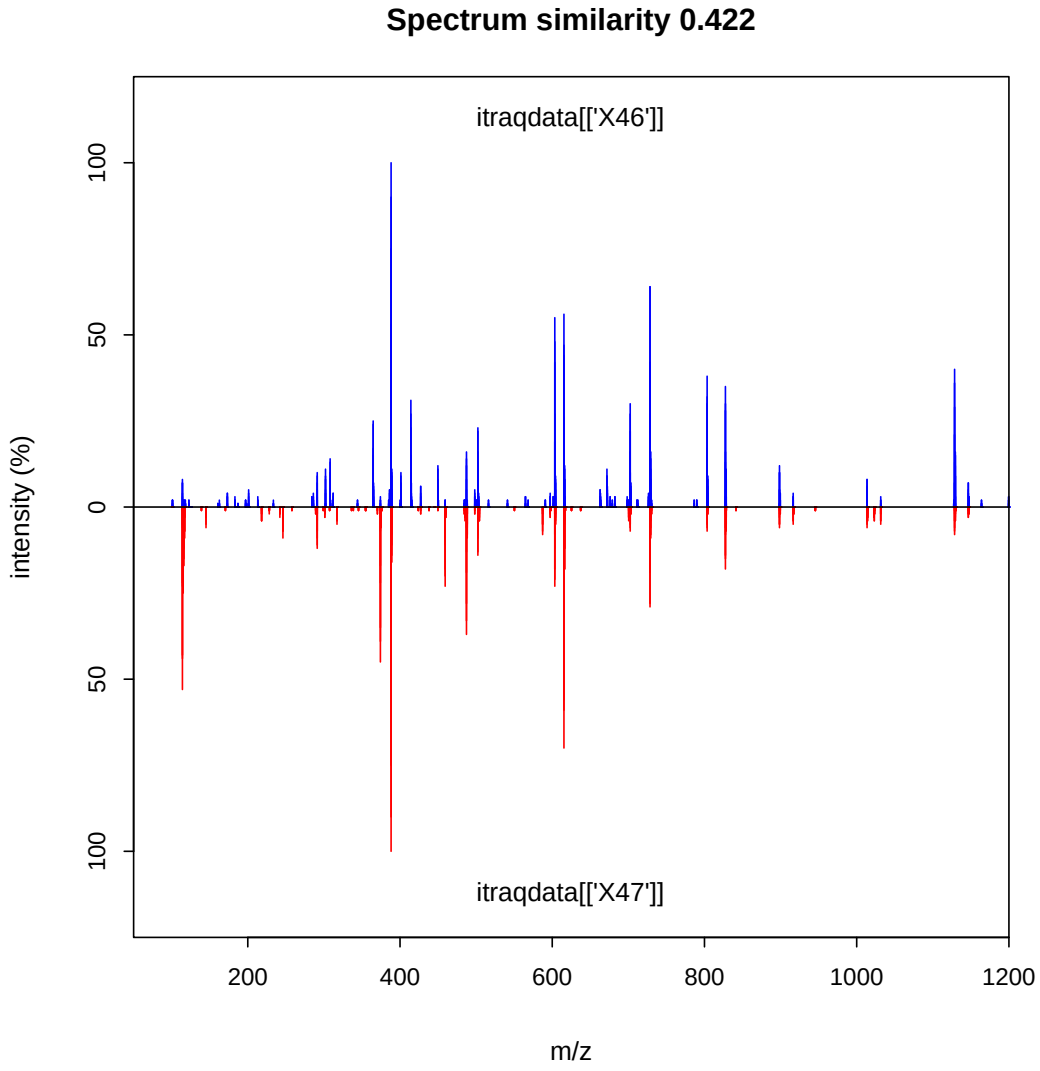
4.4 Working with peptide sequences

```
> library(IPPD)
> ## x <- myo500[,'mz'] y <- myo500[,'intensities'] sel <- x <= 2500 y
> ## <- y[sel] x <- x[sel] see vignette
>
> ## add example
>
> library(BRAIN)

Loading required package:  PolynomF
Loading required package:  Biostrings

> atoms <- getAtomsFromSeq("SIVPSGASTGVHEALEMR")
> unlist(atoms)

  C   H   N   O   S
 77 129  23  27   1

>
> library(Rdisop)

Loading required package:  RcppClassic

> pepmol <- getMolecule(paste0(names(atoms), unlist(atoms), collapse = ""))
> pepmol

$formula
[1] "C77H129N23O27S"

$score
[1] 1

$exactmass
[1] 1840

$charge
[1] 0

$parity
[1] "e"

$valid
[1] "Valid"

$DBE
```

```
[1] 25

$isotopes
$isotopes[[1]]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 1839.9149 1840.9177 1841.9197 1.843e+03 1.844e+03 1.845e+03
[2,]    0.3427    0.3353    0.1961 8.474e-02 2.953e-02 8.692e-03
          [,7]      [,8]      [,9]     [,10]
[1,] 1.846e+03 1.847e+03 1.848e+03 1.849e+03
[2,] 2.226e-03 5.066e-04 1.040e-04 1.950e-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)

Warning:  the m/z tolerance is set too high

       mz intensity.top intensity.bottom
1   114.1             0               44
2   114.1             0               53
3   114.1             0               43
4   115.1             0               25
5   364.7            25                0
6   374.2             0               39
7   374.2             0               45
8   374.2             0               35
9   388.2             0               35
10  388.3             0               75
11  388.3             0              100
12  388.3             0               90
13  388.3            35               53
14  388.3           100               53
15  388.3            90               53
16  388.3            53               53
17  388.3            75               53
18  414.3            31                0
19  414.3            27                0
```

```
20  487.3             0                33
21  487.3             0                37
22  487.3             0                28
23  603.3            42                 0
24  603.4            55                 0
25  603.4            48                 0
26  603.4            27                 0
27  615.3             0                28
28  615.3             0                56
29  615.4             0                70
30  615.4             0                59
31  615.4            26                32
32  615.4            44                32
33  615.4            56                32
34  615.4            47                32
35  702.4            27                 0
36  702.4            30                 0
37  728.4             0                28
38  728.5            64                29
39  728.5            64                29
40  728.5            42                29
41  728.5            42                29
42  803.4            30                 0
43  803.5            38                 0
44  803.5            32                 0
45  827.5            28                 0
46  827.5            35                 0
47  827.5            30                 0
48 1128.6            36                 0
49 1128.6            40                 0
50 1128.7            29                 0

> title(main = paste("Spectrum similarity", round(sim, 3)))
>
> MonoisotopicMass(formula = list(C = 2, O = 1, H = 6))

[1] 46.04

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

[1] 46.04

> ## x11() plot(t(.pepmol$isotopes[[1]]), type = 'h')
>
```

```
> ## x <- IsotopicDistribution(formula = list(C = 2, O = 1, H=6))
> ## t(molecule$isotopes[[1]]) par(mfrow = c(2,1))
> ## plot(t(molecule$isotopes[[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))
>
> eno <- download.file("http://www.uniprot.org/uniprot/P00924.fasta", destfile = "P(
> eno <- paste(readLines("P00924.fasta")[-1], collapse = "")
> enopep <- Digest(eno, missed = 1)
> nrow(enopep)  ## 103

[1] 103

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

[1] 68

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

[1] 13
```




```
> ## example code to generate an Texshade image to be included directly
> ## in a Latex document or R vignette
>
> ## 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 example
> incrate <- c(seq(0, 0.9, 0.1), 0.95, 1)
> inc <- lapply(incrate, 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 ", incrate[i], "%"))
```

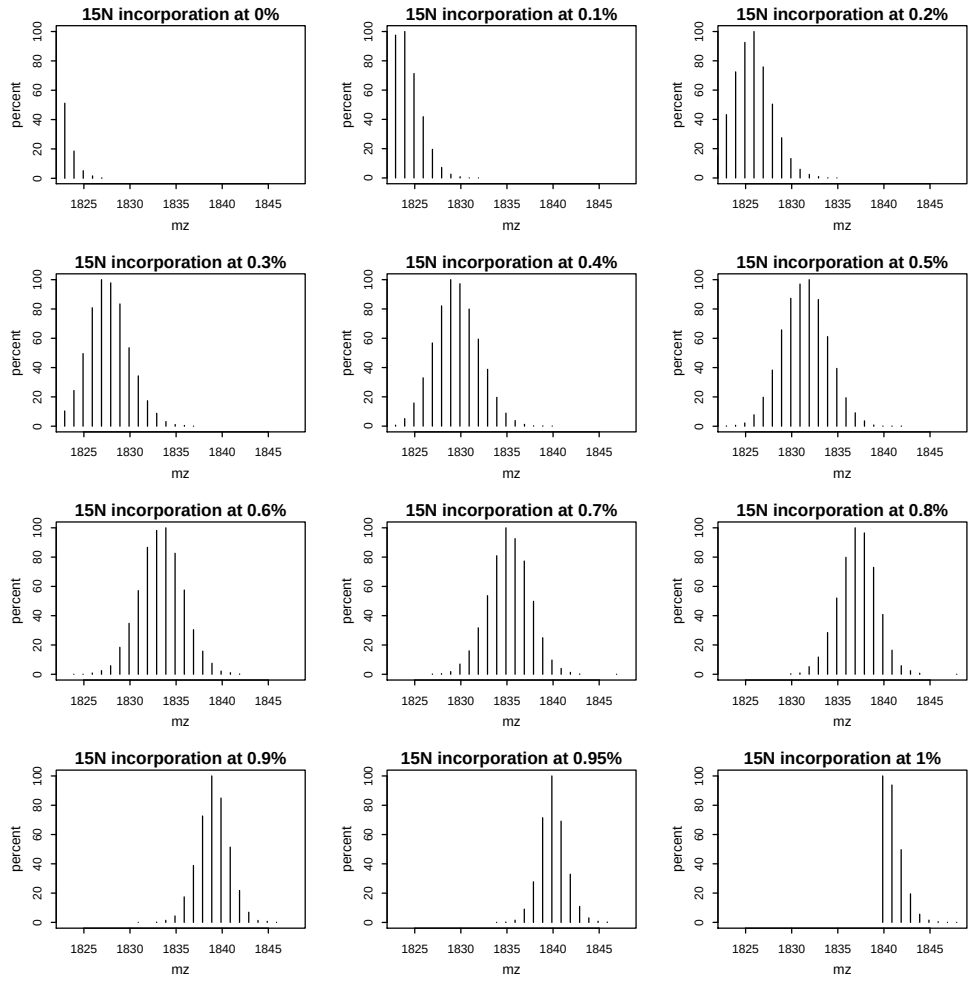


Figure 10: Isotopic envelope for the `YEVQGEVFTKPQLWP` peptide at different ^{15}N incorporation rates.

4.5 The isobar package

```
> library(isobar)
> .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)$accession, fData(qnt)$seque
+     sep = ".")
> pgtbl <- data.frame(spectrum = rownames(.ions), peptide = fData(qnt)$sequence,
+     modif = ":", start.pos = 1, protein = fData(qnt)$accession, accession = fData(
> x <- new("TMT6plexSpectra", pgtbl, .ions, .mass)

merging identifications

> featureData(x)$proteins <- as.character(fData(qnt)$accession)
>
> x <- correctIsotopeImpurities(x)  ## useless, identity matrix

LOG: isotopeImpurities.corrected:  TRUE

> x <- normalize(x, per.file = FALSE)  ## little effect

LOG: is.normalized:  TRUE
LOG: normalization.multiplicative.factor channel 126:  1.1229
LOG: normalization.multiplicative.factor channel 127:  1.0766
LOG: normalization.multiplicative.factor channel 128:  1
LOG: normalization.multiplicative.factor channel 129:  1.0537
LOG: normalization.multiplicative.factor channel 130:  1.1524
LOG: normalization.multiplicative.factor channel 131:  1.1154

> 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]   0.07345 941.48624   2.82448

> ib.background <- subsetIBSpectra(x, protein = spks, "exclude")
> nm.background <- NoiseModel(ib.background)

[1] 0.01346 2.85121 0.84631

> ib.spks <- subsetIBSpectra(x, protein = spks, direction = "exclude others",
+     specificity = "reporter-specific")
> nm.spks <- NoiseModel(ib.spks, one.to.one = FALSE, pool = TRUE)

4 proteins with more than 10 spectra, taking top 50.
[1] 1.000e-10 5.829e+00 6.610e-01

>
> ratios <- 10^estimateRatio(x, nm, channel1 = "127", channel2 = "129", protein = sp
+     combine = FALSE)[, "lratio"]
>
> res <- estimateRatio(x, nm, channel1 = "127", channel2 = "129", protein = unique(
+     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
> ## cls3[grep('P00761', rownames(res))] <- 'red' ## Trypsin
> pch3[grep("P02769", rownames(res))] <- 19
> pch3[grep("P00924", rownames(res))] <- 19
> pch3[grep("P62894", rownames(res))] <- 19
> pch3[grep("P00489", rownames(res))] <- 19
> ## pch3[grep('P00761', rownames(res))] <- 19
>
> rat.exp <- c(PHO = 2/2, ENO = 5/1, BSA = 2.5/10, CYT = 1/1)
```

```
> par(mfrow = c(1, 2))
> 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 = 0.7)
> plot(res$lratio, -log10(res$p.value.rat), col = cls3, pch = pch3, xlab = expressio
+     fold - change), ylab = expression(-log[10] ~ p - value), main = "Protein volca
+     xlim = c(-0.7, 0.7))
> grid()
> abline(h = -log10(0.01), lty = "dotted")
> abline(v = log10(c(2, 0.5)), lty = "dotted")
> abline(v = -0.003, col = "springgreen4", lty = "dashed", lwd = 2)
> abline(v = 0.003, col = "darkorchid2", lty = "dashed", lwd = 2)
> abline(v = log10(5), col = "dodgerblue", lty = "dashed", lwd = 2)
> abline(v = log10(0.25), col = "gold4", lty = "dashed", lwd = 2)
> points(res[spks, "lratio"], -log10(res[spks, "p.value.rat"]), col = c("darkorchid2
+     "dodgerblue", "gold4", "springgreen4"), pch = 19)
```

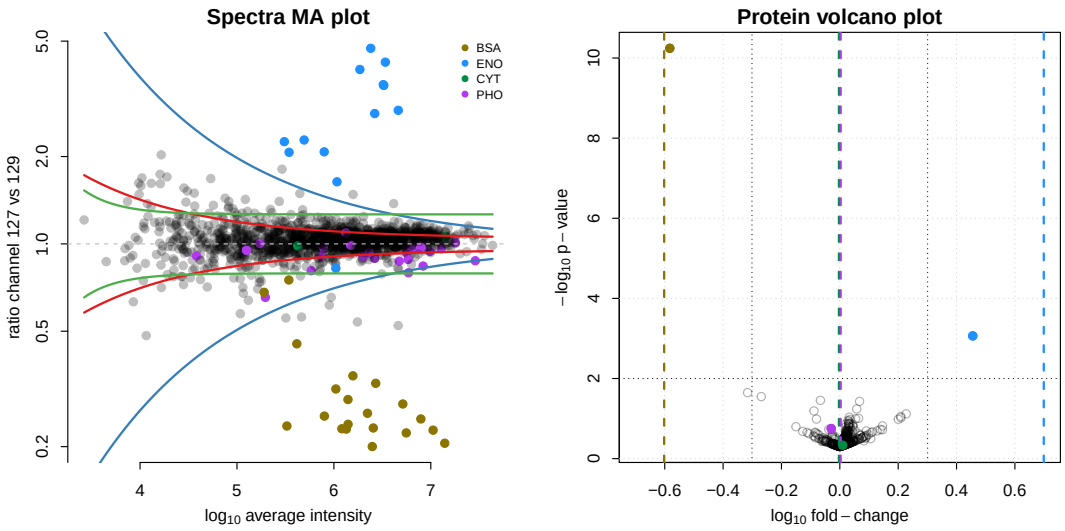


Figure 11: Result from the isobar pipeline.

4.6 The synapter package

The synapter 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 capabilites 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 Annotation

See rols and hpar vignettes.

```
> id <- "ENSG00000002746"
> library("hpar")
> getHpa(id, "SubcellularLoc")

              Gene                       Main.location Other.location
24 ENSG00000002746 Nucleus but not nucleoli;Cytoplasm
   Expression.type Reliability
24             APE        High

>
> library(org.Hs.eg.db)
> library(GO.db)
> ans <- select(org.Hs.eg.db, keys = id, cols = c("ENSEMBL", "GO", "ONTOLOGY"),
+     keytype = "ENSEMBL")

Warning:  'select' resulted in 1:many mapping between keys and return
rows

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

          ENSEMBL         GO EVIDENCE ONTOLOGY
2 ENSG00000002746 GO:0005634      IDA       CC
3 ENSG00000002746 GO:0005737      IDA       CC

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

 GO:0005634  GO:0005737
  "nucleus" "cytoplasm"

>
> 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] "nucleus"       "cytoplasm"     "intracellular"

>
>
> library("rols")
> ## see vignette
```

Session information

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

- R version 2.15.1 Patched (2012-09-19 r60762), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_GB.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB.UTF-8`, `LC_COLLATE=en_GB.UTF-8`, `LC_MONETARY=en_GB.UTF-8`, `LC_MESSAGES=en_GB.UTF-8`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_GB.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: AnnotationDbi 1.19.39, Biobase 2.17.8, BiocGenerics 0.3.2, biomaRt 2.13.2, Biostrings 2.25.12, bitops 1.0-4.1, BRAIN 1.1.1, codetools 0.2-8, DBI 0.2-5, digest 0.5.2, doMC 1.2.5, foreach 1.4.0, ggplot2 0.9.2.1, GO.db 2.8.0, hpar 0.99.1, IPPD 1.5.1, IRanges 1.15.44, isobar 1.3.2, iterators 1.0.6, knitr 0.8, lattice 0.20-10, MALDIquant 1.3, MASS 7.3-21, Matrix 1.0-9, msdata 0.1.10, MSnbase 1.5.25, multicore 0.1-7, mzR 1.3.9, org.Hs.eg.db 2.8.0, OrgMassSpecR 0.3-12, plyr 1.7.1, PolynomF 0.94, RColorBrewer 1.0-5, Rcpp 0.9.13, RcppClassic 0.9.2, Rdisop 1.17.1, readBrukerFlexData 1.4, reshape2 1.2.1, RforProteomics 0.1.0, rols 0.99.12, RSQLite 0.11.2, XML 3.9-4
- Loaded via a namespace (and not attached): affy 1.35.1, affyio 1.25.0, BiocInstaller 1.5.12, colorspace 1.1-1, dichromat 1.2-4, distr 2.3.3, evaluate 0.4.2, formatR 0.6, grid 2.15.1, gtable 0.1.1, labeling 0.1, limma 3.13.20, memoise 0.1, munsell 0.4, preprocessCore 1.19.0, proto 0.3-9.2, RCurl 1.91-1, R.methodsS3 1.4.2, R.oo 1.9.9, R.utils 1.16.2, scales 0.2.2, sfsmisc 1.0-20, SSOAP 0.8-0, startupmsg 0.7.2, stats4 2.15.1, stringr 0.6.1, tools 2.15.1, vsn 3.25.0, XMLSchema 0.7-2, zlibbioc 1.3.0

References

[1] L. Gatto and A. Christoforou. Using R for proteomics data analysis. *BBA - Proteins and Proteomics*, 2012.