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

\textit{Keywords:} bioinformatics, proteomics, mass spectrometry, tutorial.
## Contents

1 Introduction .................................................. 3
   1.1 General R resources .................................. 3
   1.2 Getting help .......................................... 3
   1.3 Installation .......................................... 4
   1.4 External dependencies ................................. 5
   1.5 Obtaining the code .................................. 6
   1.6 Prepare the working environment ...................... 6

2 Data standards and input/output ......................... 7
   2.1 The **mzR** package .................................. 7

3 Raw data abstraction with **MSnExp** objects .......... 9
   3.1 **mgf** read/write support ............................ 12

4 Quantitative proteomics .................................. 12
   4.1 The **mzTab** format .................................. 12
   4.2 Working with raw data ............................... 17
   4.3 The **MALDIquant** package ......................... 21
   4.4 Working with peptide sequences ...................... 25
   4.5 The **isobar** package ................................ 32
   4.6 The **synapter** package .............................. 36

5 MS\(^2\) spectra identification ......................... 36
   5.1 Preparation of the input data ....................... 36
   5.2 Performing the search ................................ 37
   5.3 Import and analyse results ......................... 38

6 Annotation .................................................. 39

7 Other packages ............................................ 41
   7.1 Bioconductor packages ............................... 41
   7.2 The Chemometrics and Computational Physics CRAN Task View .... 42
   7.3 Other CRAN packages ................................ 43

8 Session information ...................................... 44
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 [8]. A pre-print of the manuscript is available on arXiv\(^1\).

There are however numerous additional R resources distributed by the Bioconductor\(^2\) and CRAN\(^3\) repositories, as well as packages hosted on personal websites. Section 7 on page 41 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\(^4\) 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 [8]. For details about the Bioconductor project, the reader is referred to [10].

1.2 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

\(^1\)http://arxiv.org/abs/1305.6559
\(^2\)http://www.bioconductor.org
\(^3\)http://cran.r-project.org/web/packages/
\(^4\)https://github.com/lgatto/TeachingMaterial
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.

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

R has several mailing lists\(^5\). 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\(^6\) dedicated to bioinformatics matters and Bioconductor packages. The main Bioconductor list is the most relevant one. It is possible to post\(^7\) questions without subscribing to the list. Finally, the dedicated RforProteomics google group\(^8\) welcomes questions/comments/announcements related to R and mass-spectrometry/proteomics.

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 document, on page 44). 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\(^9\).

All lists have browsable archives.

### 1.3 Installation

The package should be installed using as described below:

\(^5\)http://www.r-project.org/mail.html
\(^6\)http://bioconductor.org/help/mailing-list/
\(^7\)http://bioconductor.org/help/mailing-list/mailform/
\(^8\)https://groups.google.com/forum/#!forum/rbioc-sig-proteomics
\(^9\)https://github.com/hadley/devtools/wiki/Reproducibility
## only first time you install Bioconductor packages

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

## else

```r
library("BiocInstaller")
biocLite("RforProteomics")
```

To install all dependencies (78 packages) and reproduce the code in the vignette, replace the last line in the code chunk above with:

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

Finally, the package can be loaded with

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

### This is the ‘RforProteomics’ version 1.0.12.  
### Run ‘RforProteomics()’ in R or visit  
### ‘http://lgatto.github.com/RforProteomics/’ to get started.

See also the ‘RforProteomics’ web page\(^\text{10}\) for more information on installation.

### 1.4 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\(^\text{11}\) (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.

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

- **biomaRt** performs on-line requests using the **curl**\(^\text{12}\) infrastructure. On debian-based systems, you one needs to install **libcurl-dev** or **libcurl4-openssl-dev**.

\(^{10}\text{http://lgatto.github.io/RforProteomics/}\)  
\(^{11}\text{http://cdf.gsfc.nasa.gov/}\)  
\(^{12}\text{http://curl.haxx.se/}\)  

5
1.5 Obtaining the code

The code in this document describes all the examples presented in [8] 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\textsuperscript{13} for emacs or RStudio\textsuperscript{14} 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.

```r
## 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] "RforProteomics.R"
```


1.6 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.

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

\textsuperscript{13}\url{http://ess.r-project.org/}
\textsuperscript{14}\url{http://rstudio.org/}
# Data standards and input/output

## The mzR package

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

```{r}
## load the required packages
library("mzR")  ## the software package
library("msdata")  ## the data package

## below, we extract the relevant 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 connection to the mzML file
mz <- openMSfile(file)

## demonstration of data access
basename(fileName(mz))

## [1] "MM14.mzML"

isInitialized(mz)

## [1] TRUE

runInfo(mz)

## $scanCount
## [1] 112

## $lowMz
## [1] 0

## $highMz
```

---
mzR is used by other packages, like **MSnbase** [9], **TargetSearch** [6] and **xcms** [12, 1, 13], that provide a higher level abstraction to the data.
3 Raw data abstraction with MSnExp objects

**MSnbase** [9] 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).

```r
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 2017
## MSn retention times: 25:1 - 25:2 minutes
## - - - Processing information - - -
## Data loaded: Thu Sep 19 23:07:14 2013
## MSnbase version: 1.9.7
## - - - 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 single 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
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

Read and write support for data in the mgf\textsuperscript{15} and mzTab\textsuperscript{16} 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 getPXD000001mzTab function from the RforProteomics package.

4.1 The mzTab format

The first code chunk downloads the data, reads it into R and generates an MSnSet instance and then calculates protein intensities by summing the peptide quantitation data. Figure \ref{fig:protein_intensities} illustrates the intensities for 5 proteins.

```r
## Downloads the experiment
mztab <- getPXD000001mzTab()
mztab ## the mzTab file name
## [1] "/F063721.dat-mztab.txt"

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

## Detected a metadata section
## Detected a peptide section

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

## 1 10630132 11238708 12424917 10997763 9928972 10398534
```

\textsuperscript{15}http://www.matrixscience.com/help/data_file_help.html#GEN

\textsuperscript{16}https://code.google.com/p/mztab/
## `combine into proteins`
## - using the `accession` feature meta data
## - sum the peptide intensities

```r
protqnt <- combineFeatures(qnt,
                           groupBy = fData(qnt)$accession,
                           fun = sum)
```

## Combined 1528 features into 404 using user-defined function

```r
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)
```

```r
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")
```
Figure 2: Protein quantitation data.
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, parallel = FALSE)
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) +
ggttitle(""")

## Scale for 'x' is already present. Adding another scale for 'x', which
will replace the existing scale.
**Figure 5:** A m/z delta plot.
plot(Signal ~ Incomplete, data = d,
    xlab = expression(Incomplete^{dissociation}),
    ylab = expression(Sum\text{of\textasciitilde}reporters\text{intensities}),
    pch = 19,
    col = "#4582B380")
grid()
abline(0, 1, lty = "dotted")
abline(lm(Signal ~ Incomplete, data = d), col = "darkblue")

\begin{figure}
\centering
\includegraphics[width=\textwidth]{plot.png}
\caption{Incomplete dissociation.}
\end{figure}
MAplot(qnt[, c(4, 2)], cex = 0.9, col = cls, pch = pch, 
   show.statistics = FALSE)

**Figure 7:** MAplot on an MSnSet instance.
4.3 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.\(^{17}\)

Loading the data

```r
## 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.
## 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))
```

\(^{17}\)http://strimmerlab.org/software/maldiquant/
## 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

```
plot(s)
```

![Spectrum plotting in MALDIquant.](image)

**Preprocessing**

```r
## 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 : 360.039 KiB
## Name : 2010_05_19_Gibb_C8_A1.A1
## File : /home/lgatto/R/x86_64-unknown-linux-gnu-library/3.0/readBrukerFlexData/Examples/2010_05_19_Gibb_C8_A1/0_A1/1/1SLin/fid

### smoothing - 5 point moving average
```r
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 : 360.039 KiB
## Name : 2010_05_19_Gibb_C8_A1.A1
## File : /home/lgatto/R/x86_64-unknown-linux-gnu-library/3.0/readBrukerFlexData/Examples/2010_05_19_Gibb_C8_A1/0_A1/1/1SLin/fid
```

### baseline subtraction
```r
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 : 360.039 KiB
## Name : 2010_05_19_Gibb_C8_A1.A1
## File : /home/lgatto/R/x86_64-unknown-linux-gnu-library/3.0/readBrukerFlexData/Examples/2010_05_19_Gibb_C8_A1/0_A1/1/1SLin/fid
```

### Peak picking

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

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

---

23
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

```r
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] 1840
##
## $charge
## [1] 0
##
## $parity
## [1] "e"
##
## $valid
## [1] "Valid"
##
## $DBE
## [1] 25
```
```r
## $isotopes
## $isotopes[[1]]
## [1,] 1839.9149 1840.9177 1841.9197 1.843e+03 1.844e+03
## [2,] 0.3427 0.3353 0.1961 8.474e-02 2.953e-02
## [1,] 1.845e+03 1.846e+03 1.847e+03 1.848e+03 1.849e+03
## [2,] 8.692e-03 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)

##   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
```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>388.3</td>
<td>90</td>
<td>53</td>
</tr>
<tr>
<td>16</td>
<td>388.3</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>17</td>
<td>388.3</td>
<td>75</td>
<td>53</td>
</tr>
<tr>
<td>18</td>
<td>414.3</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>414.3</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>487.3</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>21</td>
<td>487.3</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>22</td>
<td>487.3</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>23</td>
<td>603.3</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>603.4</td>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>603.4</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>603.4</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>615.3</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>28</td>
<td>615.3</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>29</td>
<td>615.4</td>
<td>0</td>
<td>70</td>
</tr>
<tr>
<td>30</td>
<td>615.4</td>
<td>0</td>
<td>59</td>
</tr>
<tr>
<td>31</td>
<td>615.4</td>
<td>26</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>615.4</td>
<td>44</td>
<td>32</td>
</tr>
<tr>
<td>33</td>
<td>615.4</td>
<td>56</td>
<td>32</td>
</tr>
<tr>
<td>34</td>
<td>615.4</td>
<td>47</td>
<td>32</td>
</tr>
<tr>
<td>35</td>
<td>702.4</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>702.4</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>728.4</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>38</td>
<td>728.5</td>
<td>64</td>
<td>29</td>
</tr>
<tr>
<td>39</td>
<td>728.5</td>
<td>64</td>
<td>29</td>
</tr>
<tr>
<td>40</td>
<td>728.5</td>
<td>42</td>
<td>29</td>
</tr>
<tr>
<td>41</td>
<td>728.5</td>
<td>42</td>
<td>29</td>
</tr>
<tr>
<td>42</td>
<td>803.4</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>43</td>
<td>803.5</td>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>803.5</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>827.5</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>827.5</td>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>47</td>
<td>827.5</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>1128.6</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>1128.6</td>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>
### 50 1128.7 29 0

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

```
Spectrum similarity 0.422
```

![Spectrum similarity plot](attachment:image.png)

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

```r
## [1] 46.04
```

```r
molecule <- getMolecule("C2H5OH")
molecule$exactmass
```
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)
cleave("LAAGKVEDSD", enzym = "trypsin")

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

Sometimes cleavage is not perfect and the enzyme misses 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 a 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.

```r
text <- "seq1.tex"
cat("\begin{texshade}{Figures/P00924.fasta}
   \setsize{numbering}{footnotesize}
   \setsize{residues}{footnotesize}
   \residuesperline*{70}
   \shadingmode{functional}
   \hideconsensus
   \vsepspace{1mm}
   \hidenames
   \noblockskip
", file = text, append = TRUE)
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 = text, 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{document}
```

30
$^{15}$N incorporation

```r
# 15N incorporation rates from 0, 0.1, ..., 0.9, 0.95, 1
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]*100, "\%"))
```

![Figure 10: Isotopic envelope for the YEVQGEVFTKPQLWP peptide at different $^{15}$N incorporation rates.](image)
4.5 The isobar package

The isobar package [3] provides methods for the statistical analysis of isobarically tagged MS² experiments.

```r
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)$accession)
x <- new("TMT6plexSpectra", pgtbl, .ions, .mass)
## data.frame columns OK
## done creating protein group
featureData(x)$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:  0.8905
## LOG: normalization.multiplicative.factor channel 127:  0.9288
## LOG: normalization.multiplicative.factor channel 128:  1
## LOG: normalization.multiplicative.factor channel 129:  0.949
## LOG: normalization.multiplicative.factor channel 130:  0.8677
## LOG: normalization.multiplicative.factor channel 131:  0.8965
```
## spikes

```r
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("P00489", featureNames(x))] <- 19
pch2[grep("P00924", featureNames(x))] <- 19
pch2[grep("P62894", featureNames(x))] <- 19
pch2[grep("P02769", featureNames(x))] <- 19
nm <- NoiseModel(x)
## [1] 0.07346 941.45023 2.82447

ib.background <- subsetIBSpectra(x, protein=spks,
                                  direction = "exclude")
nm.background <- NoiseModel(ib.background)
## [1] 0.01346 2.85121 0.84631

ib.spks <- subsetIBSpectra(x, protein = spks,
                           direction="include",
                           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 = 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)
Figure 11: Result from the isobar pipeline.
4.6 The synapter package

The synapter [2] package comes with a detailed vignette that describes how to prepare the MS² 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.

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

5 MS² spectra identification

A recent addition to Bioconductor 2.12 is the rTANDEM package, that provides a direct interface to the X!Tandem software [5]. A typical rTANDEM pipeline comprises

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

Using example code/data from the rTANDEM vignette/package, these steps are executed as described below.

5.1 Preparation of the input data

```r
library(rTANDEM)
taxonomy <- rTTaxo(taxon = "yeast",
                   format = "peptide",
                   URL = system.file("extdata/fasta/scd.fasta.pro",
                                      package="rTANDEM"))
param <- rTParam()
param <- setParamValue(param,
```
5.2 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:
```
Valid models = 30
Unique models = 30
Estimated false positives = 1 +/- 1

5.3 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]

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 102 942.5 0.0048
## 2: 250.1.1 1811 250 1212.6 0.0009
## tandem.score mh delta peak.count missed.cleavages
## 1: 31.9 942.5 -0.0220 NA 0
## 2: 35.0 1212.6 0.0079 NA 0
## start.position end.position
## 1: 166 173
## 2: 437 447

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.

6 Annotation

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

We start with the `hpar` package, an interface to the *Human Protein Atlas* [14, 15], to retrieve subcellular localisation information for the ENSG00000002746 ensemble gene.

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

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

Below, we make use of the human annotation package `org.Hs.eg.db` and the Gene Ontology annotation package `GO.db` to retrieve the same information as above.
library(org.Hs.eg.db)
library(GO.db)
ans <- select(org.Hs.eg.db, keys = id, columns = c("ENSEMBL",
        "GO", "ONTROLOGY"), keytype = "ENSEMBL")
ans <- ans[ans$ONTROLOGY == "CC", ]
ans
## ENSEMBL GO EVIDENCE ONTROLOGY
## 2 ENSG00000002746 GO:0005634 IDA CC
## 3 ENSG00000002746 GO:0005737 IDA CC
sapply(as.list(GOTERM[ans$GO]), slot, "Term")
## GO:0005634 GO:0005737
## "nucleus" "cytoplasm"

Finally, this information can also be retrieved from on-line databases using the biomaRt package [7].

library("biomaRt")
ensmbl <- 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] "cytoplasm" "nucleus"
7 Other packages

7.1 Bioconductor packages

This section provides a complete list of packages available in the relevant Bioconductor version 2.13 (as of September 19, 2013) biocView\(^\text{18}\) categories. Tables 1, 2 and 3 represent the packages for the Proteomics (40 packages), MassSpectrometry (22 packages) and MassSpectrometryData (6 experiment packages) categories.

<table>
<thead>
<tr>
<th>Package</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASEB</td>
<td>Predict Acetylated Lysine Sites</td>
</tr>
<tr>
<td>BRAIN</td>
<td>Baffling Recursive Algorithm for Isotope distribution calculations</td>
</tr>
<tr>
<td>CellNOptR</td>
<td>Training of boolean logic models of signalling networks using prior knowledge networks and perturbation data.</td>
</tr>
<tr>
<td>ChemmineR</td>
<td>Cheminformatics of Drug-like Small Molecule Data</td>
</tr>
<tr>
<td>cisPath</td>
<td>Visualization and manage of the protein-protein interaction networks.</td>
</tr>
<tr>
<td>cleaver</td>
<td>Cleavage of polypeptide sequences</td>
</tr>
<tr>
<td>clippda</td>
<td>A package for the clinical proteomic profiling data analysis</td>
</tr>
<tr>
<td>CNORdt</td>
<td>Add-on to CellNOptR: Discretized time treatments</td>
</tr>
<tr>
<td>CNORfeeder</td>
<td>Integration of CellNOptR to add missing links</td>
</tr>
<tr>
<td>CNORode</td>
<td>ODE add-on to CellNOptR</td>
</tr>
<tr>
<td>deltaGseg</td>
<td>deltaGseg</td>
</tr>
<tr>
<td>eiR</td>
<td>Accelerated similarity searching of small molecules</td>
</tr>
<tr>
<td>fmsR</td>
<td>Mismatch Tolerant Maximum Common Substructure Searching</td>
</tr>
<tr>
<td>GraphPAC</td>
<td>Identification of Mutational Clusters in Proteins via a Graph Theoretical Approach.</td>
</tr>
<tr>
<td>hpar</td>
<td>Human Protein Atlas in R</td>
</tr>
<tr>
<td>iPAC</td>
<td>Identification of Protein Amino acid Clustering</td>
</tr>
<tr>
<td>IPPD</td>
<td>Isotopic peak pattern deconvolution for Protein Mass Spectrometry by template matching</td>
</tr>
<tr>
<td>isobar</td>
<td>Analysis and quantitation of isobarically tagged MSMS proteomics data</td>
</tr>
<tr>
<td>LPEadl</td>
<td>A correction of the local pooled error (LPE) method to replace the asymptotic variance adjustment with an unbiased adjustment based on sample size.</td>
</tr>
<tr>
<td>MassSpecWavelet</td>
<td>Mass spectrum processing by wavelet-based algorithms</td>
</tr>
<tr>
<td>MSnbase</td>
<td>MSnbase: Base Functions and Classes for MS-based Proteomics</td>
</tr>
<tr>
<td>mzID</td>
<td>An mZidentML parser for R</td>
</tr>
<tr>
<td>mzR</td>
<td>parser for netCDF, mzXML, mzData and mzML files (mass spectrometry data)</td>
</tr>
<tr>
<td>PAntBuilder</td>
<td>Protein annotation data package builder</td>
</tr>
<tr>
<td>pathview</td>
<td>a tool set for pathway based data integration and visualization</td>
</tr>
<tr>
<td>PCpheno</td>
<td>Phenotypes and cellular organizational units</td>
</tr>
<tr>
<td>plgsem</td>
<td>Detect differential expression in microarray and proteomics datasets with the Power Law Global Error Model (PLGEM)</td>
</tr>
<tr>
<td>PLPE</td>
<td>Local Pooled Error Test for Differential Expression with Paired High-throughput Data</td>
</tr>
<tr>
<td>ppiStats</td>
<td>Protein-Protein Interaction Statistical Package</td>
</tr>
<tr>
<td>PRCess</td>
<td>Ciphergen SELDI-TOF Processing</td>
</tr>
<tr>
<td>procoil</td>
<td>Prediction of Oligomerization of Coiled Coil Proteins</td>
</tr>
<tr>
<td>prot2D</td>
<td>Statistical Tools for volume data from 2D Gel Electrophoresis</td>
</tr>
<tr>
<td>RCASPAR</td>
<td>A package for survival time prediction based on a piecewise baseline hazard Cox regression model.</td>
</tr>
<tr>
<td>Rchemcpp</td>
<td>Similarity measures for chemical compounds</td>
</tr>
<tr>
<td>RpsiXML</td>
<td>R interface to PSI-MI 2.5 files</td>
</tr>
<tr>
<td>rTANDEM</td>
<td>Encapsulate XTandem in R</td>
</tr>
<tr>
<td>ScISI</td>
<td>In Silico Interactome</td>
</tr>
<tr>
<td>SLGI</td>
<td>Synthetic Lethal Genetic Interaction</td>
</tr>
<tr>
<td>SpacePAC</td>
<td>Identification of Mutational Clusters in 3D Protein Space via Simulation.</td>
</tr>
<tr>
<td>synapter</td>
<td>Label-free data analysis pipeline for optimal identification and quantitation</td>
</tr>
</tbody>
</table>

Table 1: Packages available under the Proteomics biocViews category.

\(^{18}\)http://www.bioconductor.org/packages/devel/BiocViews.html
Package Title

apComplex Estimate protein complex membership using AP-MS protein data
BRAIN Baffling Recursive Algorithm for Isotope distribution calculation
CAMERA Collection of annotation related methods for mass spectrometry data
flagme Analysis of Metabolomics GC/MS Data
gaga GaGa hierarchical model for high-throughput data analysis
iontree Data management and analysis of ion trees from ion-trap mass spectrometry
isobar Analysis and quantitation of isobarically tagged MSMS proteomics data
MassArray Analytical Tools for MassArray Data
MassSpecWavelet Mass spectrum processing by wavelet-based algorithms
MSSubbase MSubbase: Base Functions and Classes for MS-based Proteomics
mzID An mzIdentML parser for R
mzR parser for netCDF, mzXML, mzData and mzML files (mass spectrometry data)
PAPi Predict metabolic pathway activity based on metabolomics data
PROcess Ciphergen SELDI-TOF Processing
Rdisop Decomposition of Isotopic Patterns
Risa Converting experimental metadata from ISA-tab into Bioconductor data structures
RMassBank Workflow to process tandem MS files and build MassBank records
rosl An R interface to the Ontology Lookup Service
rTANDEM Encapsulate XTandem in R.
synapter Label-free data analysis pipeline for optimal identification and quantitation
TargetSearch A package for the analysis of GC-MS metabolite profiling data.
xcms LC/MS and GC/MS Data Analysis

Table 2: Packages available under the MassSpectrometry biocViews category.

<table>
<thead>
<tr>
<th>Package</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>faahKO</td>
<td>Saghatelian et al. (2004) FAAH knockout LC/MS data</td>
</tr>
<tr>
<td>gcspike-lite</td>
<td>Spike-in data for GC/MS data and methods within flagme</td>
</tr>
<tr>
<td>msdata</td>
<td>Various Mass Spectrometry raw data example files</td>
</tr>
<tr>
<td>RforProteomics</td>
<td>Companion package to the 'Using R and Bioconductor for proteomics data analysis' publication</td>
</tr>
<tr>
<td>RMassBankData</td>
<td>Test dataset for RMassBank</td>
</tr>
<tr>
<td>synapterdata</td>
<td>Data accompanying the synapter package</td>
</tr>
</tbody>
</table>

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

7.2 The Chemometrics and Computational Physics CRAN Task View

The CRAN task view on Chemometrics and Computational Physics lists 71 packages, including a set of packages for mass spectrometry and proteomics, some of which are illustrated in this document. The most relevant (non Bioconductor) packages are summarised below.

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

http://cran.r-project.org/web/learn/ChemPhys.html
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).

7.3 Other CRAN packages

Finally, digeR20, which is available on CRAN but not listed in the Chemometrics and Computational Physics Task View, 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.

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.

20http://cran.r-project.org/web/packages/digeR/index.html
8 Session information

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

- R version 3.0.2 beta (2013-09-15 r63933), x86_64-unknown-linux-gnu
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: AnnotationDbi 1.23.23, Biobase 2.21.7, BiocGenerics 0.7.5, biocViews 1.29.0, biomartR 2.17.2, Biostrings 2.29.19, bitops 1.0-6, BRAIN 1.7.0, cleaver 0.99.5, data.table 1.8.10, DBI 0.2-7, digest 0.6.3, ggplot2 0.9.3.1, GO.db 2.9.0, hpar 1.3.1, IPPD 1.9.0, IRanges 1.19.37, isobar 1.7.6, knitr 1.4.1, lattice 0.20-23, MALDIquant 1.8, MALDIquantForeign 0.5, MASS 7.3-29, Matrix 1.0-14, mdata 0.1.13, MSbase 1.9.7, mzR 1.7.3, org.Hs.eg.db 2.9.0, OrgMassSpecR 0.3-12, plyr 1.8, PolynomF 0.94, RColorBrewer 1.0-5, Rcpp 0.10.4, RcppClassic 0.9.4, Rdisop 1.21.0, reshape2 1.2.2, RforProteomics 1.0.12,_rods 1.3.2, RSQLite 0.11.4, rTANDEM 1.1.3, XML 3.98-1.1, xtable 1.7-1, XVector 0.1.4
- Loaded via a namespace (and not attached): affy 1.39.2, affyio 1.29.0, base64enc 0.1-1, BiocInstaller 1.11.4, codetools 0.2-8, colorspace 1.2-3, dichromat 2.0-0, distr 2.5.2, downloader 0.3, evaluate 0.4.7, formatR 0.9, graph 1.39.3, grid 3.0.2, gtable 0.1.2, highr 0.2.1, impute 1.35.0, labeling 0.2, limma 3.17.23, munsell 0.4.2, preprocessCore 1.23.0, proto 0.3-10, RBGL 1.37.2, RCurl 1.95-4.1, readBrukerFlexData 1.7, readMzXmlData 2.7, R.methodsS3 1.5.1, R.oo 1.15.1, RUnit 0.4.26, R.utils 1.27.1, scales 0.2.3, sfsmisc 1.0-24, SSOAP 0.8-0, startupmsg 0.9, stats4 3.0.2, stringr 0.6.2, SweaveListingUtils 0.6.1, tools 3.0.2, vsn 3.29.1, XMLSchema 0.7-2, zlibbioc 1.7.0

References


