

Using R and Bioconductor for proteomics data analysis

Laurent Gatto [Computational Proteomics Unit](#)

[Atelier PROSPECTOM](#) 19 Nov 2014, Grenoble, France

Version of this document: e8c452c [2014-11-19 12:55:42 +0000]

Setup

The follow packages will be used throughout this documents. R version 3.1.1 or higher is required to install all the packages using `BiocInstaller::biocLite`.

```
library("mzR")
library("mzID")
library("MSnID")
library("MSGFplus")
library("MSnbase")
library("rpx")
library("MLInterfaces")
library("pRoloc")
library("pRolocdata")
library("rTANDEM")
library("MSGFplus")
library("MSGFgui")
library("rols")
library("hpar")
```

The most convenient way to install all the tutorials requirement (and more related content), is to install [RforProteomics](#) with all its dependencies.

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

Introduction

This tutorial illustrates R / Bioconductor infrastructure for proteomics. Topics covered focus on support for open community-driven formats for raw data and identification results, packages for peptide-spectrum matching, data processing and analysis:

- Exploring available infrastructure
- Mass spectrometry data
- Getting data from proteomics repositories
- Handling raw MS data
- Handling identification data
- MS/MS database search
- Analysing search results
- High-level data interface
- Quantitative proteomics
- Importing third-party quantitative data

- Data processing and analysis
- Statistical analysis
- Machine learning
- Annotation
- Other relevant packages/pipelines

Links to other packages and references are also documented. In particular, the vignettes included in the [RforProteomics](#) package also contains relevant material.

Exploring available infrastructure

In Bioconductor version 3.0, there are respectively 65 [proteomics](#), 44 [mass spectrometry software packages](#) and 7 [mass spectrometry experiment packages](#). These respective packages can be extracted with the `proteomicsPackages()`, `massSpectrometryPackages()` and `massSpectrometryDataPackages()` and explored interactively.

```
library("RforProteomics")
pp <- proteomicsPackages()
display(pp)
```

Mass spectrometry data

| Type | Format | Package |
|----------------|-----------------------------|---|
| raw | mzML, mzXML, netCDF, mzData | mzR (read) |
| identification | mzIdentML | mzR and mzID (read) |
| quantitation | mzQuantML | |
| peak lists | mgf | MSnbase (read/write) |
| other | mzTab | MSnbase (read/write) |

Getting data from proteomics repositories

Contemporary MS-based proteomics data is disseminated through the [ProteomeXchange](#) infrastructure, which centrally coordinates submission, storage and dissemination through multiple data repositories, such as the [PRIDE](#) data base at the EBI for MS/MS experiments, [PASSEL](#) at the ISB for SRM data and the [MassIVE](#) resource. The [rpx](#) is an interface to ProteomeXchange and provides a basic and unified access to PX data.

```
library("rpx")
pxannounced()
```

```
## 15 new ProteomeXchange announcements
```

```
##      Data.Set      Publication.Data      Message
## 1 PXD001283 2014-11-19 08:44:40      New
## 2 PXD001301 2014-11-19 08:42:01      New
## 3 PXD000837 2014-11-19 08:30:45 Updated information
## 4 PXD000715 2014-11-18 16:34:39 Updated information
```

```
## 5 PXD000837 2014-11-18 16:30:02 Updated information
## 6 PXD001354 2014-11-18 16:29:15 Updated information
## 7 PXD000627 2014-11-18 16:28:07 Updated information
## 8 PXD001125 2014-11-18 16:27:02 Updated information
## 9 PXD001045 2014-11-18 16:26:04 Updated information
## 10 PXD001260 2014-11-18 16:24:55 Updated information
## 11 PXD001414 2014-11-18 16:22:44 Updated information
## 12 PXD000715 2014-11-18 09:35:10 New
## 13 PXD000837 2014-11-18 09:27:36 New
## 14 PXD001260 2014-11-18 09:13:08 Updated information
## 15 PXD001045 2014-11-18 09:12:15 Updated information
```

```
px <- PXDataset("PXD000001")
px
```

```
## Object of class "PXDataset"
## Id: PXD000001 with 8 files
## [1] 'F063721.dat' ... [8] 'erwinia_carotovora.fasta'
## Use 'pxfiles(.)' to see all files.
```

```
pxfiles(px)
```

```
## [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.mzXML"
## [7] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.raw"
## [8] "erwinia_carotovora.fasta"
```

Other metadata for the px dataset:

```
pntax(px)
pxurl(px)
pxref(px)
```

Data files can then be downloaded with the `pxget` function as illustrated below.

```
## mzf <- pxget(px, pxfiles(px)[6])
mzf <- "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML"
mzf
```

```
## [1] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML"
```

Exercise

Explore what data files have been deposited by Pandey's recent [draft map of the human proteome](#).

Solution

Handling raw MS data

The `mzR` package provides an interface to the [proteowizard](#) code base, the legacy RAMP is a non-sequential parser and other C/C++ code to access various raw data files, such as `mzML`, `mzXML`, `netCDF`, and `mzData`. The data is accessed on-disk, i.e it does not get loaded entirely in memory by default. The three main functions are `openMSfile` to create a file handle to a raw data file, `header` to extract metadata about the spectra contained in the file and `peaks` to extract one or multiple spectra of interest. Other functions such as `instrumentInfo`, or `runInfo` can be used to gather general information about a run.

```
library("mzR")
ms <- openMSfile(mzf)
ms
```

```
## Mass Spectrometry file handle.
## Filename: TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
## Number of scans: 7534
```

```
hd <- header(ms)
dim(hd)
```

```
## [1] 7534 21
```

```
names(hd)
```

```
## [1] "seqNum"          "acquisitionNum"
## [3] "msLevel"         "polarity"
## [5] "peaksCount"      "totIonCurrent"
## [7] "retentionTime"   "basePeakMZ"
## [9] "basePeakIntensity" "collisionEnergy"
## [11] "ionisationEnergy" "lowMZ"
## [13] "highMZ"          "precursorScanNum"
## [15] "precursorMZ"     "precursorCharge"
## [17] "precursorIntensity" "mergedScan"
## [19] "mergedResultScanNum" "mergedResultStartScanNum"
## [21] "mergedResultEndScanNum"
```

Exercise

Extract the index of the MS2 spectrum with the highest base peak intensity and plot its spectrum.
Is the data centroided or in profile mode?

```
hd2 <- hd[hd$msLevel == 2, ]
i <- which.max(hd2$basePeakIntensity)
hd2[i, ]
```

Solution

```
##      seqNum acquisitionNum msLevel polarity peaksCount totIonCurrent
## 5404      5404           5404      2      1          275    2283283712
##      retentionTime basePeakMZ basePeakIntensity collisionEnergy
## 5404      2751.31    859.5032      354288224          0
##      ionisationEnergy lowMZ highMZ precursorScanNum precursorMZ
## 5404          0 100.5031 1995.63          5403    859.1722
##      precursorCharge precursorIntensity mergedScan mergedResultScanNum
## 5404          3      627820480          0          0
##      mergedResultStartScanNum mergedResultEndScanNum
## 5404          0          0
```

```
pi <- peaks(ms, hd2[i, 1])
mz <- hd2[i, "basePeakMZ"]

par(mfrow = c(2, 2))
plot(pi, type = "h", main = paste("Acquisition", i))
plot(pi, type = "h", xlim = c(mz-0.5, mz+0.5))

pj <- peaks(ms, 100)
plot(pj, type = "l", main = paste("Acquisition", 100))
plot(pj, type = "l", xlim = c(536, 540))
```

Read the `MSnbase::MSmap` manual and look at the example to learn how the `mzR` raw data support can be exploited to generate maps of slides of raw MS data. (Note that the `hd` variable containing the raw data header was missing in `MSnbase` version < 1.14.1.)

Handling identification data

The `RforProteomics` package distributes a small identification result file (see `?TMT_Erwinia_1uLSike_Top10HCD_isol2_45ste`) that we load and parse using infrastructure from the `mzID` package.

```
library("mzID")
f <- dir(system.file("extdata", package = "RforProteomics"),
        pattern = "mzid", full.names=TRUE)
basename(f)
```

```
## [1] "TMT_Erwinia.mzid.gz"
```

```
id <- mzID(f)
```

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

```
id
```

```
## An mzID object
```

```
##
```

```
## Software used: MS-GF+ (version: Beta (v10072))
```

```
##
```

```
## Rawfile: /home/lgatto/dev/00_github/RforProteomics/sandbox/TMT_Erwinia_1uLSike_Top10HCD_isol2_45ste.ms
```

```
##
```

```
## Database: /home/lgatto/dev/00_github/RforProteomics/sandbox/erwinia_carotovora.fasta
```

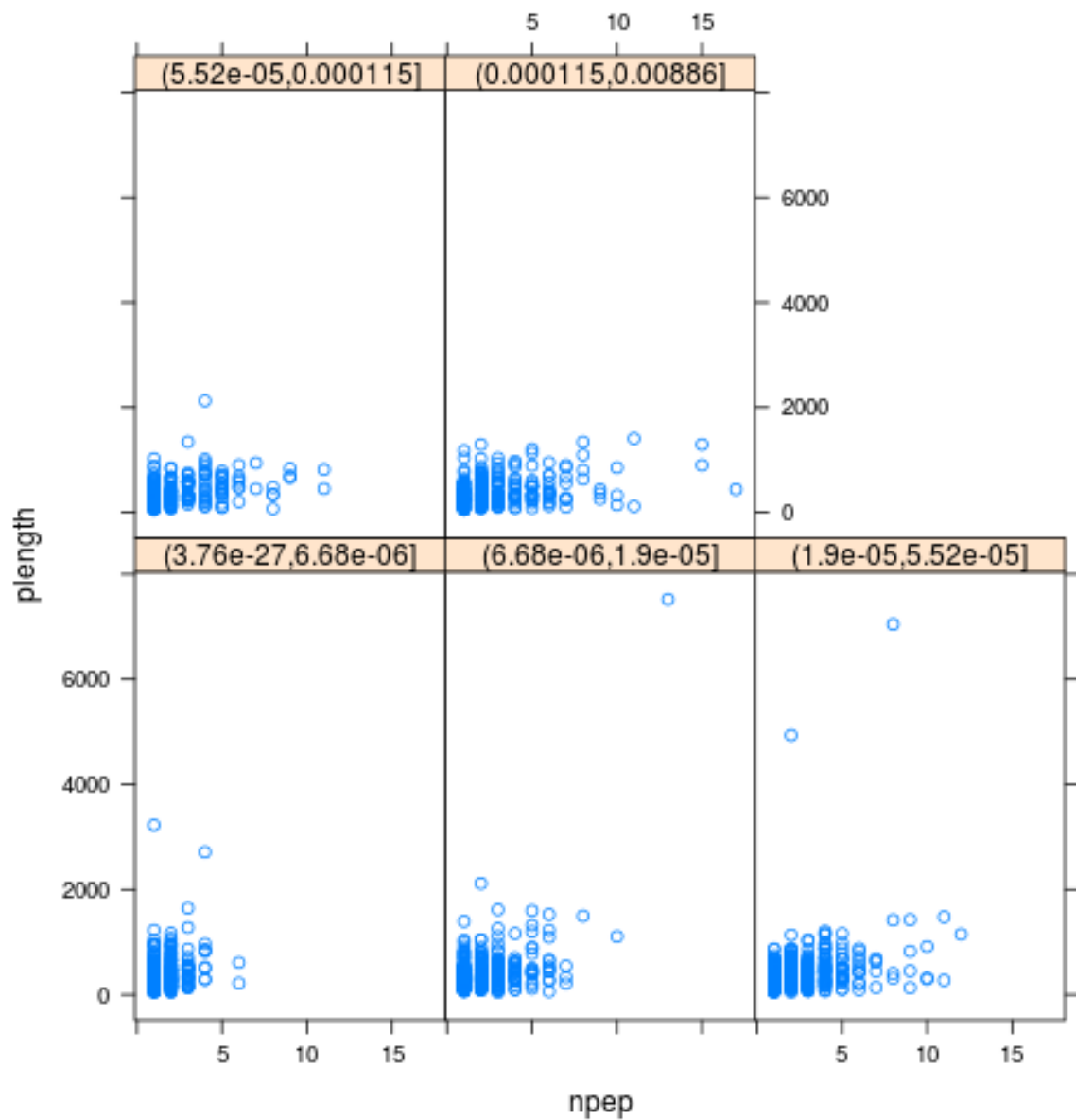
```
##  
## Number of scans: 5287  
## Number of PSM's: 5563
```

Various data can be extracted from the `mzID` object, using one the accessor functions such as `database`, `scans`, `peptides`, ... The object can also be converted into a `data.frame` using the `flatten` function.

Exercise

Is there a relation between the length of a protein and the number of identified peptides, conditioned by the (average) e-value of the identifications?

```
fid <- flatten(id)  
x <- by(fid, fid$accession, function(x)  
  c(unique(x$length),  
    length(unique(x$pepseq)),  
    mean(x$'ms-gf:specvalue')))  
x <- data.frame(do.call(rbind, x))  
colnames(x) <- c("plength", "npep", "eval")  
x$bins <- cut(x$eval, summary(x$eval))  
library("lattice")  
xyplot(plength ~ npep | bins, data = x)
```



Solution

The `mzR` package also support fast parsing of `mzIdentML` files with the `openIDfile` function. Compare it, it terms of output and speed with `mzID`.

```
library("mzR")
library("mzID")
f <- dir(system.file("extdata", package = "RforProteomics"),
         pattern = "mzid", full.names=TRUE)

system.time({
  id0 <- mzID(f)
  fid0 <- flatten(id0)
})
```

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

```
## user system elapsed
## 27.540 0.100 27.679
```

```
head(fid0)
```

```
## spectrumid scan number(s) acquisitionnum passthreshold rank
## 1 scan=5782 5782 5782 TRUE 1
## 2 scan=6037 6037 6037 TRUE 1
## 3 scan=5235 5235 5235 TRUE 1
## 4 scan=5397 5397 5397 TRUE 1
## 5 scan=6075 6075 6075 TRUE 1
## 6 scan=5761 5761 5761 TRUE 1
## calculatedmasstocharge experimentalmasstocharge chargestate
## 1 1080.2321 1080.2325 3
## 2 1002.2115 1002.2089 3
## 3 1189.2800 1189.2836 3
## 4 960.5365 960.5365 3
## 5 1264.3419 1264.3409 3
## 6 1268.6501 1268.6429 2
## ms-gf:denovoscore ms-gf:evaluate ms-gf:rawscore ms-gf:specevalue
## 1 174 5.430080e-21 147 3.764831e-27
## 2 245 9.943751e-20 214 6.902626e-26
## 3 264 2.564787e-19 211 1.778789e-25
## 4 178 2.581753e-18 154 1.792541e-24
## 5 252 2.178423e-17 188 1.510364e-23
## 6 138 2.329453e-17 123 1.618941e-23
## assumedissociationmethod isotopeerror isdecoy post pre end start
## 1 HCD 0 FALSE S R 84 50
## 2 HCD 0 FALSE R K 315 288
## 3 HCD 0 FALSE A R 224 192
## 4 HCD 0 FALSE - R 290 264
## 5 HCD 0 FALSE F R 153 119
## 6 HCD 0 FALSE Y K 286 264
## accession length description
## 1 ECA1932 155 outer membrane lipoprotein
## 2 ECA1147 434 trigger factor
## 3 ECA0013 295 ribose-binding periplasmic protein
## 4 ECA1731 290 flagellin
## 5 ECA1443 298 UTP--glucose-1-phosphate uridylyltransferase
## 6 ECA1444 468 6-phosphogluconate dehydrogenase, decarboxylating
## pepseq modified modification
## 1 PVQIQAGEDSNVIGALGGAVLGGFLGNTIGGSGR FALSE <NA>
## 2 TQVLDGLINANDIEVPVALIDGEIDVLR FALSE <NA>
## 3 TKGLNVMQNLLTAHPDVQAVFAQNDEMAGALR FALSE <NA>
## 4 SQILQQAGTSVLSQANQVPQTVLSLLR FALSE <NA>
## 5 PIIGDNPFFVVLPDVVLDESTADQTQENLALLISR FALSE <NA>
## 6 WTSQSSLDLGEPLSLITESVFAR FALSE <NA>
## spectrumFile
## 1 TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
## 2 TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
## 3 TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
## 4 TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
## 5 TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
## 6 TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
```



```
##           databaseFile
## 1 erwinia_carotovora.fasta
## 2 erwinia_carotovora.fasta
## 3 erwinia_carotovora.fasta
## 4 erwinia_carotovora.fasta
## 5 erwinia_carotovora.fasta
## 6 erwinia_carotovora.fasta
```

```
system.time({
  id1 <- openIDfile(f)
  fid1 <- psms(id1)
})
```

```
## Error in (function (classes, fdef, mtable) : unable to find an inherited method for function 'psms' :
```

```
## Timing stopped at: 0.237 0.001 0.237
```

```
head(fid1)
```

```
## Error in head(fid1): error in evaluating the argument 'x' in selecting a method for function 'head':
```

MS/MS database search

While searches are generally performed using third-party software independently of R or can be started from R using a `system` call, the [rTANDEM](#) package allows one to execute such searches using the X!Tandem engine. The [shinyTANDEM](#) provides a interactive interface to explore the search results.

```
library("rTANDEM")
?rtandem
library("shinyTANDEM")
?shinyTANDEM
```

Similarly, the [MSGFplus](#) package enables to perform a search using the MSGF+ engine, as illustrated below:

```
library("MSGFplus")
parameters <- msgfPar(database = 'proteins.fasta',
                      tolerance='20 ppm',
                      instrument='TOF',
                      enzyme='Lys-C')
runMSGF(parameters, c('file1.mzML', 'file2.mzML'))
```

A graphical interface to perform the search the data and explore the results is also available:

```
library("MSGFgui")
MSGFgui()
```

Exercise

Search TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML against the fasta file from PXD000001 using, for example, MSGFplus/MSGFgui.

Solution

1. Get the fasta database:

```
## fas <- pxget(px, pxfiles(px)[8])
fas <- "erwinia_carotovora.fasta"
```

2. One could run MSGF+ from the command-line directly from R:

```
msgf <- system.file(package = "MSGFplus", "MSGFplus", "MSGFplus.jar")
system(paste0("java -jar ", msgf))
cmd <- paste("java -jar", msgf, "-protocol 2 -inst 1 -s", mzf, "-d", fas)
cmd
```

```
## [1] "java -jar /home/lg390/R/x86_64-unknown-linux-gnu-library/3.1/MSGFplus/MSGFplus/MSGFplus.jar -pr
```

```
system(cmd)
```

or, use MSGFplus:

```
library("MSGFplus")
msgfpar <- msgfPar(database = fas,
                   instrument = 'HighRes',
                   enzyme = 'Trypsin'm
                   protocol = 'iTRAQ')
runMSGF(msgfpar, mzf)
```

or, through the graphical user interface:

```
library("MSGFgui")
MSGFgui()
```

Analysing search results

The [MSnID](#) package can be used for post-search filtering of MS/MS identifications. One starts with the construction of an MSnID object that is populated with identification results that can be imported from a `data.frame` or from `mzIdentML` files.

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

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

```
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
```

The package then enables to define, optimise and apply filtering based for example on missed cleavages, identification scores, precursor mass errors, etc. and assess PSM, peptide and protein FDR levels.

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

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)
```

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

```
## MSnIDFilter object
## (absParentMassErrorPPM < 2.3) & (msmsScore > 7.8)
```

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

```
## MSnID object
## Working directory: "."
## #Spectrum Files: 1
## #PSMs: 346 at 0 % FDR
## #peptides: 160 at 0 % FDR
## #accessions: 132 at 0 % FDR
```

The resulting data can be exported to a `data.frame` or to a dedicated `MSnSet` data structure for quantitative MS data, described below, and further processed and analyses using appropriate statistical tests.

High-level data interface

The above sections introduced low-level interfaces to raw and identification results. The **MSnbase** package provides abstractions for raw data through the **MSnExp** class and containers for quantification data via the **MSnSet** class. Both store

1. the actual assay data (spectra or quantitation matrix), accessed with **spectra** (or the **[, [** operators) or **exprs**;
2. sample metadata, accessed as a **data.frame** with **pData**;
3. feature metadata, accessed as a **data.frame** with **fData**.

The figure below give a schematics of an **MSnSet** instance and the relation between the assay data and the respective feature and sample metadata.

Another useful slot is **processingData**, accessed with **processingData(.)**, that records all the processing that objects have undergone since their creation (see examples below).

The **readMSData** will parse the raw data, extract the MS2 spectra (by default) and construct an MS experiment file.

(Note that while **readMSData** supports MS1 data, this is currently not convenient as all the data is read into memory.)

```
library("MSnbase")
rawFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
               full.name = TRUE, pattern = "mzXML$")
basename(rawFile)
```

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

```
msexp <- readMSData(rawFile, verbose = FALSE)
msexp
```

```
## 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: Wed Nov 19 14:44:21 2014
## MSnbase version: 1.14.0
## - - - 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)'
```

MS2 spectra can be extracted as a list of `Spectrum2` objects with the `spectra` accessor or with the `[]` operator. Individual can be accessed with `[[`.

```
length(msexp)
```

```
## [1] 5
```

```
msexp[[2]]
```

```
## Object of class "Spectrum2"
## Precursor: 546.9586
## Retention time: 25:2
## Charge: 3
## MSn level: 2
## Peaks count: 1012
## Total ion count: 56758067
```

The identification results stemming from the same raw data file can then be used to add PSM matches.

```
fData(msexp)
```

```
##      spectrum
## X1.1      1
## X2.1      2
## X3.1      3
## X4.1      4
## X5.1      5
```

```
## find path to a mzIdentML file
identFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "dummyiTRAQ.mzid")
basename(identFile)
```

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

```
msexp <- addIdentificationData(msexp, identFile)
```

```
## reading dummyiTRAQ.mzid... DONE!
```

```
fData(msexp)
```

```

##      spectrum scan number(s) passthreshold rank calculatedmasstocharge
## X1.1      1      1      TRUE      1      645.0375
## X2.1      2      2      TRUE      1      546.9633
## X3.1      3      NA      NA      NA      NA
## X4.1      4      NA      NA      NA      NA
## X5.1      5      5      TRUE      1      437.2997
##      experimentalmasstocharge chargestate ms-gf:denovoscore ms-gf:evaluate
## X1.1      645.3741      3      77      79.36958
## X2.1      546.9586      3      39      13.46615
## X3.1      NA      NA      NA      NA
## X4.1      NA      NA      NA      NA
## X5.1      437.8040      2      5      366.38422
##      ms-gf:rawscore ms-gf:specvalue assumedissociationmethod
## X1.1      -39      5.527468e-05      CID
## X2.1      -30      9.399048e-06      CID
## X3.1      NA      NA      <NA>
## X4.1      NA      NA      <NA>
## X5.1      -42      2.577830e-04      CID
##      isotopeerror isdecoy post pre end start accession length
## X1.1      1 FALSE A R 186 170 ECA0984;ECA3829 231
## X2.1      0 FALSE A K 62 50 ECA1028 275
## X3.1      <NA> NA <NA> <NA> NA NA <NA> NA
## X4.1      <NA> NA <NA> <NA> NA NA <NA> NA
## X5.1      1 FALSE L K 28 22 ECA0510 166
##
##      description
## X1.1 DNA mismatch repair protein;acetolactate synthase isozyme III large subunit
## X2.1 2,3,4,5-tetrahydropyridine-2,6-dicarboxylate N-succinyltransferase
## X3.1 <NA>
## X4.1 <NA>
## X5.1 putative capsular polysaccharide biosynthesis transferase
##      pepseq modified modification databaseFile
## X1.1 VESITARHGEVLQLRPK FALSE NA erwinia_carotovora.fasta
## X2.1 IDGQWVTHQWLKK FALSE NA erwinia_carotovora.fasta
## X3.1 <NA> NA NA <NA>
## X4.1 <NA> NA NA <NA>
## X5.1 LVILLFR FALSE NA erwinia_carotovora.fasta
##      identFile nprot npep.prot npsm.prot npsm.pep
## X1.1      2      2      1      1      1
## X2.1      2      1      1      1      1
## X3.1      NA      NA      NA      NA      NA
## X4.1      NA      NA      NA      NA      NA
## X5.1      2      1      1      1      1

```

Spectra and (parts of) experiments can be extraced and plotted.

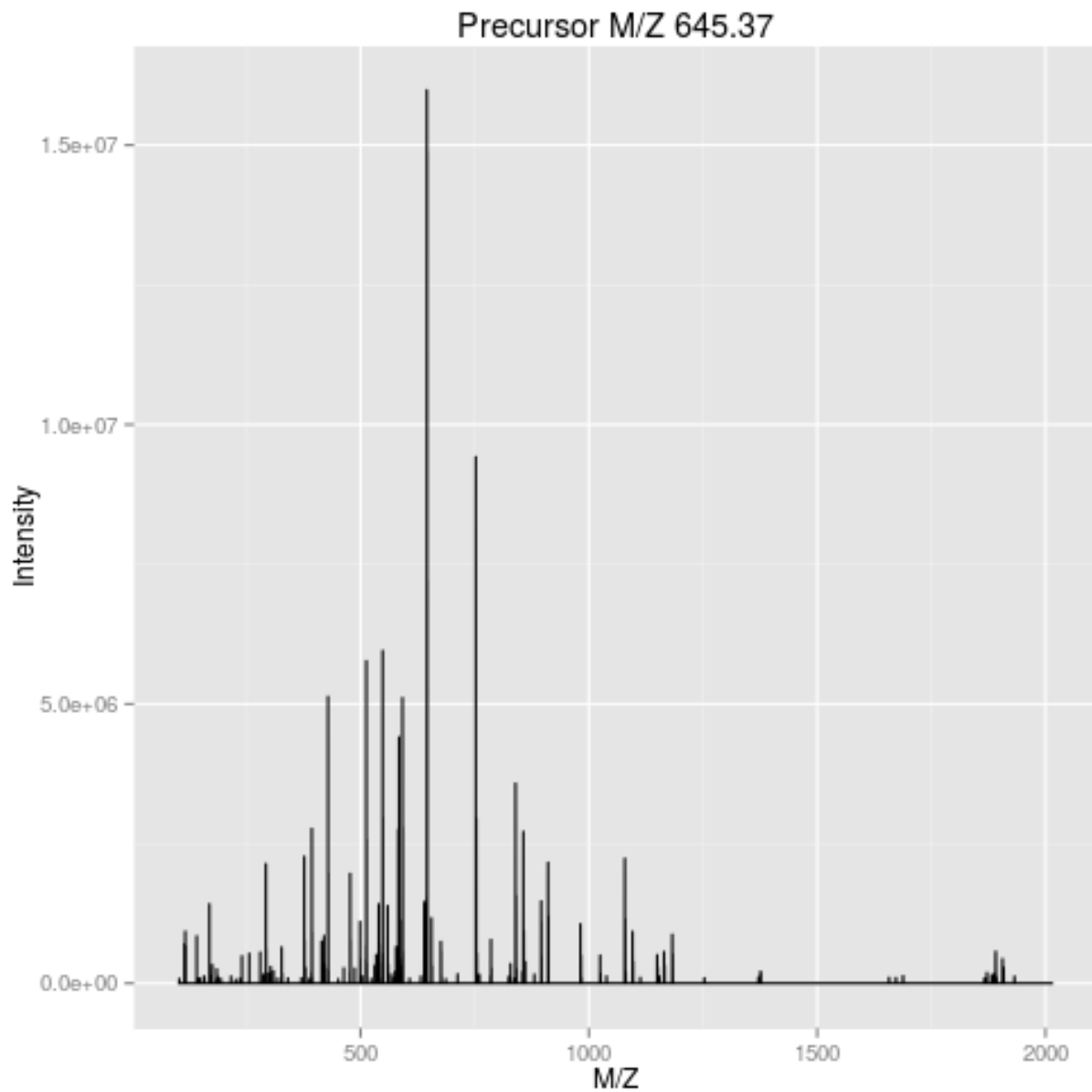
```
msexp[[1]]
```

```

## Object of class "Spectrum2"
## Precursor: 645.3741
## Retention time: 25:1
## Charge: 3
## MSn level: 2
## Peaks count: 2921
## Total ion count: 668170086

```

```
plot(msexp[[1]], full=TRUE)
```



```
msexp[1:3]
```

```
## Object of class "MSnExp"
## Object size in memory: 0.13 Mb
## - - - Spectra data - - -
## MS level(s): 2
## Number of MS1 acquisitions: 1
## Number of MSn scans: 3
## Number of precursor ions: 3
## 2 unique MZs
## Precursor MZ's: 546.96 - 645.37
```

```

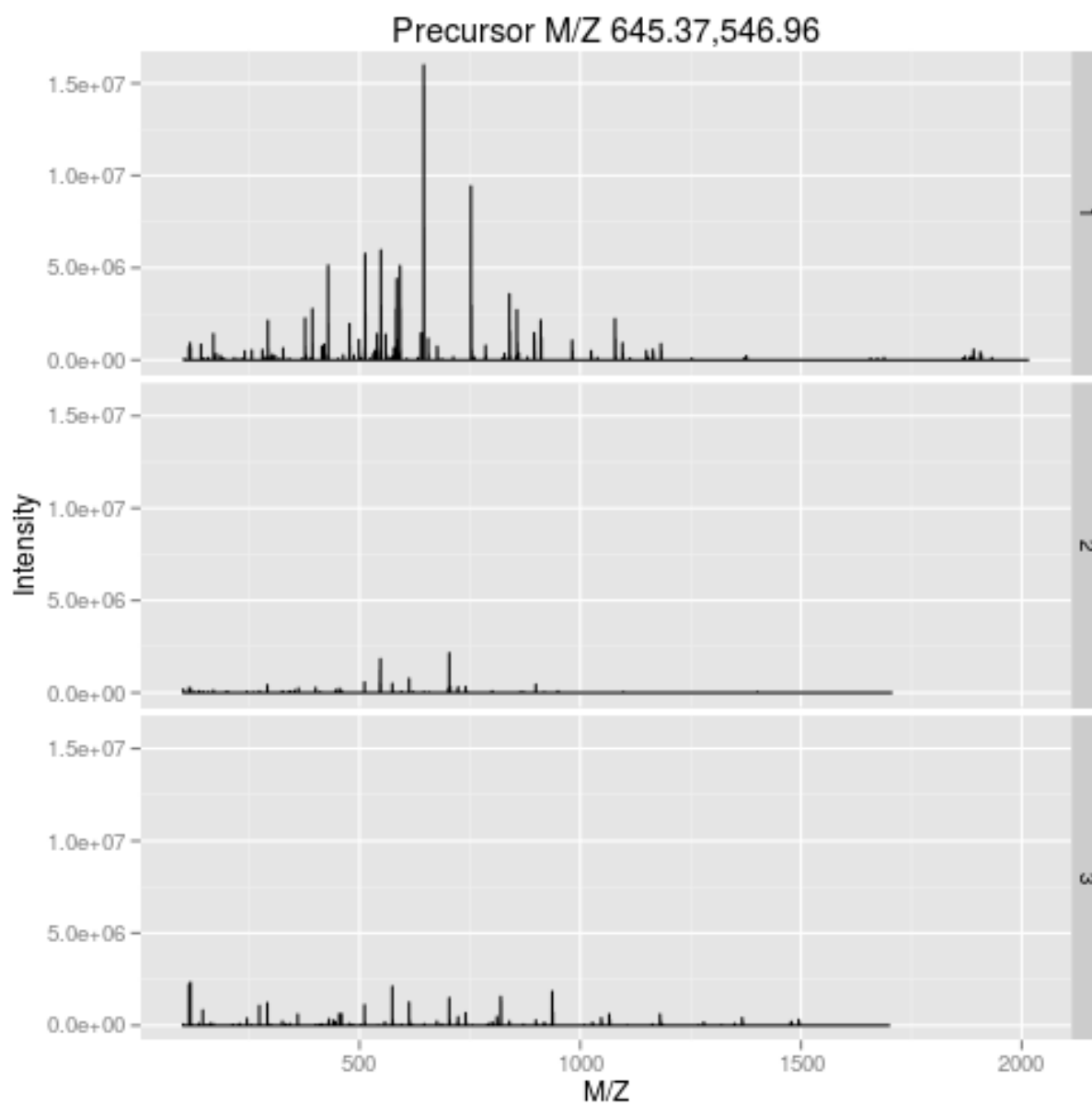
## MSn M/Z range: 100 2016.66
## MSn retention times: 25:1 - 25:2 minutes
## - - - Processing information - - -
## Data loaded: Wed Nov 19 14:44:21 2014
## Data [numerically] subsetted 3 spectra: Wed Nov 19 14:44:21 2014
## MSnbase version: 1.14.0
## - - - Meta data - - -
## phenoData
##   rowNames: 1
##   varLabels: sampleNames
##   varMetadata: labelDescription
## Loaded from:
##   dummyiTRAQ.mzXML,   dummyiTRAQ.mzid
## protocolData: none
## featureData
##   featureNames: X1.1 X2.1 X3.1
##   fvarLabels: spectrum scan number(s) ... npsm.pep (30 total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'

```

```

plot(msexp[1:3], full=TRUE)

```

Coercion to a `data.frame` is straightforward.

```
as(msexp[[1]], "data.frame")[100:105, ]
```

```
##      mz      i
## 100 141.0990 588594.812
## 101 141.1015 845401.250
## 102 141.1041 791352.125
## 103 141.1066 477623.000
## 104 141.1091 155376.312
## 105 141.1117  4752.541
```

Quantitative proteomics

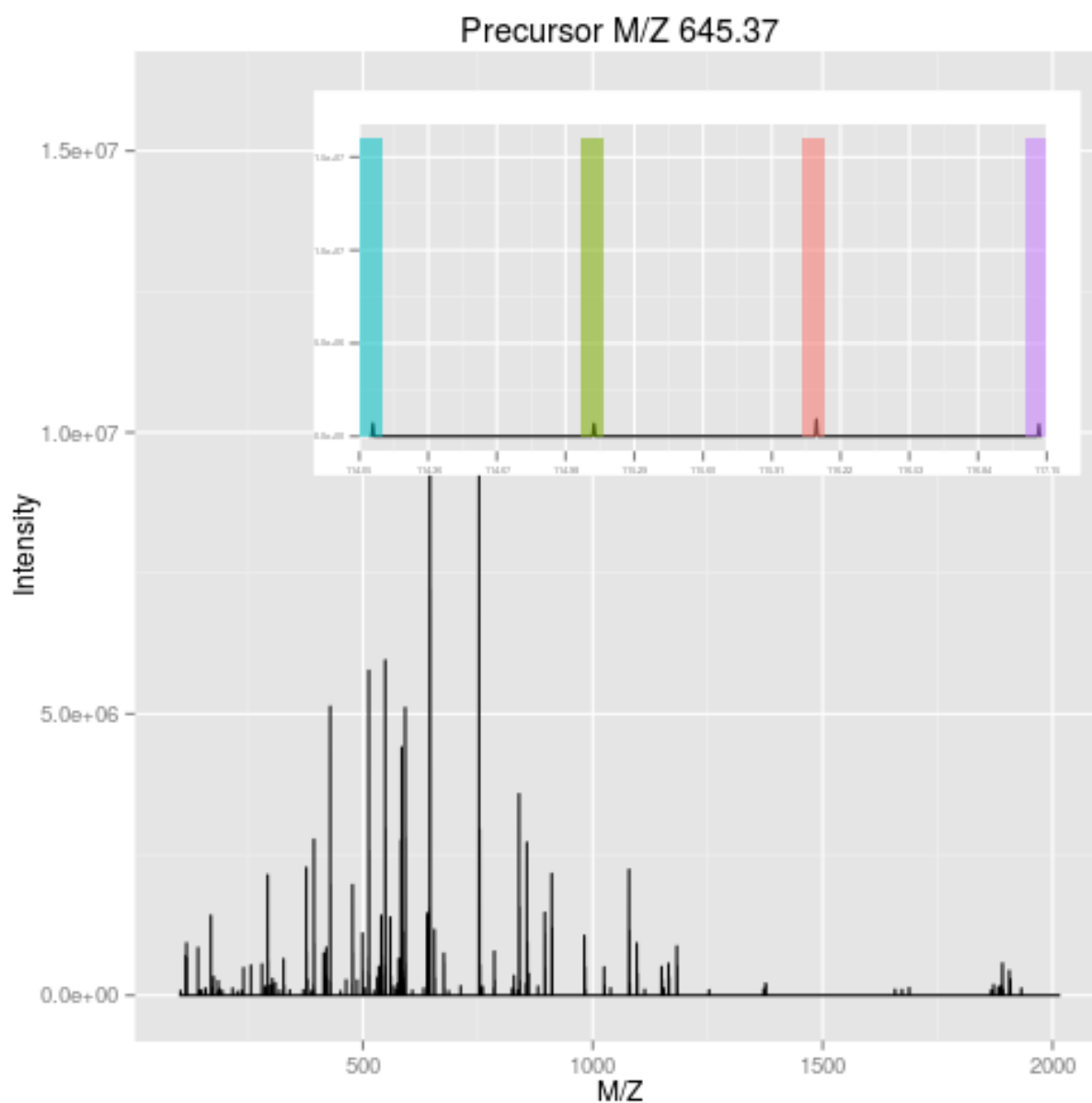
There are a wide range of proteomics quantitation techniques that can broadly be classified as labelled vs. label-free, depending whether the features are labelled prior the MS acquisition and the MS level at which quantitation is inferred, namely MS1 or MS2.

| | Label-free | Labelled |
|-----|------------|------------|
| MS1 | XIC | SILAC, 15N |
| MS2 | Counting | iTRAQ, TMT |

In terms of raw data quantitation, most efforts have been devoted to MS2-level quantitation. Label-free XIC quantitation has however been addressed in the frame of metabolomics data processing by the [xcms](#) infrastructure.

An `MSnExp` is converted to an `MSnSet` by the `quantitation` method. Below, we use the iTRAQ 4-plex isobaric tagging strategy (defined by the `iTRAQ4` parameter; other tags are available).

```
plot(msexp[[1]], full=TRUE, reporters = iTRAQ4)
```



```
msset <- quantify(msexp, method = "trap", reporters = iTRAQ4, verbose=FALSE)
exprs(msset)
```

```
##      iTRAQ4.114 iTRAQ4.115 iTRAQ4.116 iTRAQ4.117
## X1.1   4483.320  4873.996  6743.441  4601.378
## X2.1   1918.082  1418.040  1117.601  1581.954
## X3.1  15210.979 15296.256 15592.760 16550.502
## X4.1   4133.103  5069.983  4724.845  4694.801
## X5.1  11947.881 13061.875 12809.491 12911.479
```

```
processingData(msset)
```

```
## - - - Processing information - - -
```

```
## Data loaded: Wed Nov 19 14:44:21 2014
## iTRAQ4 quantification by trapezoidation: Wed Nov 19 14:44:23 2014
## MSnbase version: 1.14.0
```

Other MS2 quantitation methods available in `quantify` include the (normalised) spectral index **SI** and (normalised) spectral abundance factor **SAF** or simply a simple count method.

```
exprs(si <- quantify(msexp, method = "SIn"))
```

```
##              1
## ECA0510 0.003588641
## ECA1028 0.001470129
```

```
exprs(saf <- quantify(msexp, method = "NSAF"))
```

```
##              1
## ECA0510 0.6235828
## ECA1028 0.3764172
```

Note that spectra that have not been assigned any peptide (**NA**) or that match non-unique peptides (`npsm > 1`) are discarded in the counting process.

See also The `isobar` package supports quantitation from centroided `mgf` peak lists or its own tab-separated files that can be generated from Mascot and Phenyx vendor files.

Have a look at the `?quantify` documentation file and review the above by walking through the example.

Importing third-party quantitative data

The PSI `mzTab` file format is aimed at providing a simpler (than XML formats) and more accessible file format to the wider community. It is composed of a key-value metadata section and peptide/protein/small molecule tabular sections.

```
## mztf <- pxget(px, pxfiles(px)[2])
mztf <- "F063721.dat-mztab.txt"
(mzt <- readMzTabData(mztf, what = "PEP"))
```

```
## Warning in readMzTabData(mztf, what = "PEP"): Support for mzTab version
## 0.9 only. Support will be added soon.
```

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

```
## MSnSet (storageMode: lockedEnvironment)
## assayData: 1528 features, 6 samples
##   element names: exprs
## protocolData: none
## phenoData
##   rowNames: sub[1] sub[2] ... sub[6] (6 total)
```

```
## varLabels: abundance
## varMetadata: labelDescription
## featureData
## featureNames: 1 2 ... 1528 (1528 total)
## fvarLabels: sequence accession ... uri (14 total)
## fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
## mzTab read: Wed Nov 19 14:44:23 2014
## MSnbase version: 1.14.0
```

It is also possible to import arbitrary spreadsheets as `MSnSet` objects into R with the `readMSnSet2` function. The main 2 arguments of the function are (1) a text-based spreadsheet and (2) column names of indices that identify the quantitation data.

```
csv <- dir(system.file("extdata", package = "pRolocdata"),
            full.names = TRUE, pattern = "pr800866n_si_004-rep1.csv")
getEcols(csv, split = ",")
```

```
## [1] "\"Protein ID\""          "\"FBgn\""
## [3] "\"Flybase Symbol\""      "\"No. peptide IDs\""
## [5] "\"Mascot score\""        "\"No. peptides quantified\""
## [7] "\"area 114\""           "\"area 115\""
## [9] "\"area 116\""           "\"area 117\""
## [11] "\"PLS-DA classification\"" "\"Peptide sequence\""
## [13] "\"Precursor ion mass\""  "\"Precursor ion charge\""
## [15] "\"pd.2013\""            "\"pd.markers\""
```

```
ecols <- 7:10
res <- readMSnSet2(csv, ecols)
head(exprs(res))
```

```
## area.114 area.115 area.116 area.117
## 1 0.379000 0.281000 0.225000 0.114000
## 2 0.420000 0.209667 0.206111 0.163889
## 3 0.187333 0.167333 0.169667 0.476000
## 4 0.247500 0.253000 0.320000 0.179000
## 5 0.216000 0.183000 0.342000 0.259000
## 6 0.072000 0.212333 0.573000 0.142667
```

```
head(fData(res))
```

```
## Protein.ID      FBgn Flybase.Symbol No..peptide.IDs Mascot.score
## 1   CG10060 FBgn0001104      G-ialpha65A           3         179.86
## 2   CG10067 FBgn0000044           Act57B           5         222.40
## 3   CG10077 FBgn0035720           CG10077           5         219.65
## 4   CG10079 FBgn0003731           Egfr            2          86.39
## 5   CG10106 FBgn0029506           Tsp42Ee          1          52.10
## 6   CG10130 FBgn0010638           Sec61beta        2          79.90
## No..peptides.quantified PLS.DA.classification Peptide.sequence
## 1                        1                        PM
```

```

## 2          9          PM
## 3          3
## 4          2          PM
## 5          1          GGVFDTIQK
## 6          3          ER/Golgi
## Precursor.ion.mass Precursor.ion.charge    pd.2013 pd.markers
## 1                                PM    unknown
## 2                                PM    unknown
## 3                                unknown unknown
## 4                                PM    unknown
## 5          626.887          2 Phenotype 1    unknown
## 6                                ER/Golgi    ER

```

Data processing and analysis

Raw data processing

For raw data processing look at MSnbases's `clean`, `smooth`, `pickPeaks`, `removePeaks` and `trimMz` for MSnExp and spectra processing methods.

The [MALDIquant](#) and [xcms](#) packages also feautres a wide range of raw data processing methods on their own ad hoc data instance types.

Processing and normalisation

Each different types of quantitative data will require their own pre-processing and normalisation steps. Both isobar and MSnbase allow to correct for isobaric tag impurities normalise the quantitative data.

```

data(itraqdata)
qnt <- quantify(itraqdata, method = "trap",
               reporters = iTRAQ4, verbose = FALSE)
impurities <- matrix(c(0.929,0.059,0.002,0.000,
                      0.020,0.923,0.056,0.001,
                      0.000,0.030,0.924,0.045,
                      0.000,0.001,0.040,0.923),
                    nrow=4, byrow = TRUE)
## or, using makeImpuritiesMatrix()
## impurities <- makeImpuritiesMatrix(4)
qnt.crct <- purityCorrect(qnt, impurities)
processingData(qnt.crct)

```

```

## - - - Processing information - - -
## Data loaded: Wed May 11 18:54:39 2011
## iTRAQ4 quantification by trapezoidation: Wed Nov 19 14:44:25 2014
## Purity corrected: Wed Nov 19 14:44:25 2014
## MSnbase version: 1.1.22

```

```

plot0 <- function(x, y, main = "") {
  old.par <- par(no.readonly = TRUE)
  on.exit(par(old.par))
  par(mar = c(4, 4, 1, 1))
  par(mfrow = c(2, 2))

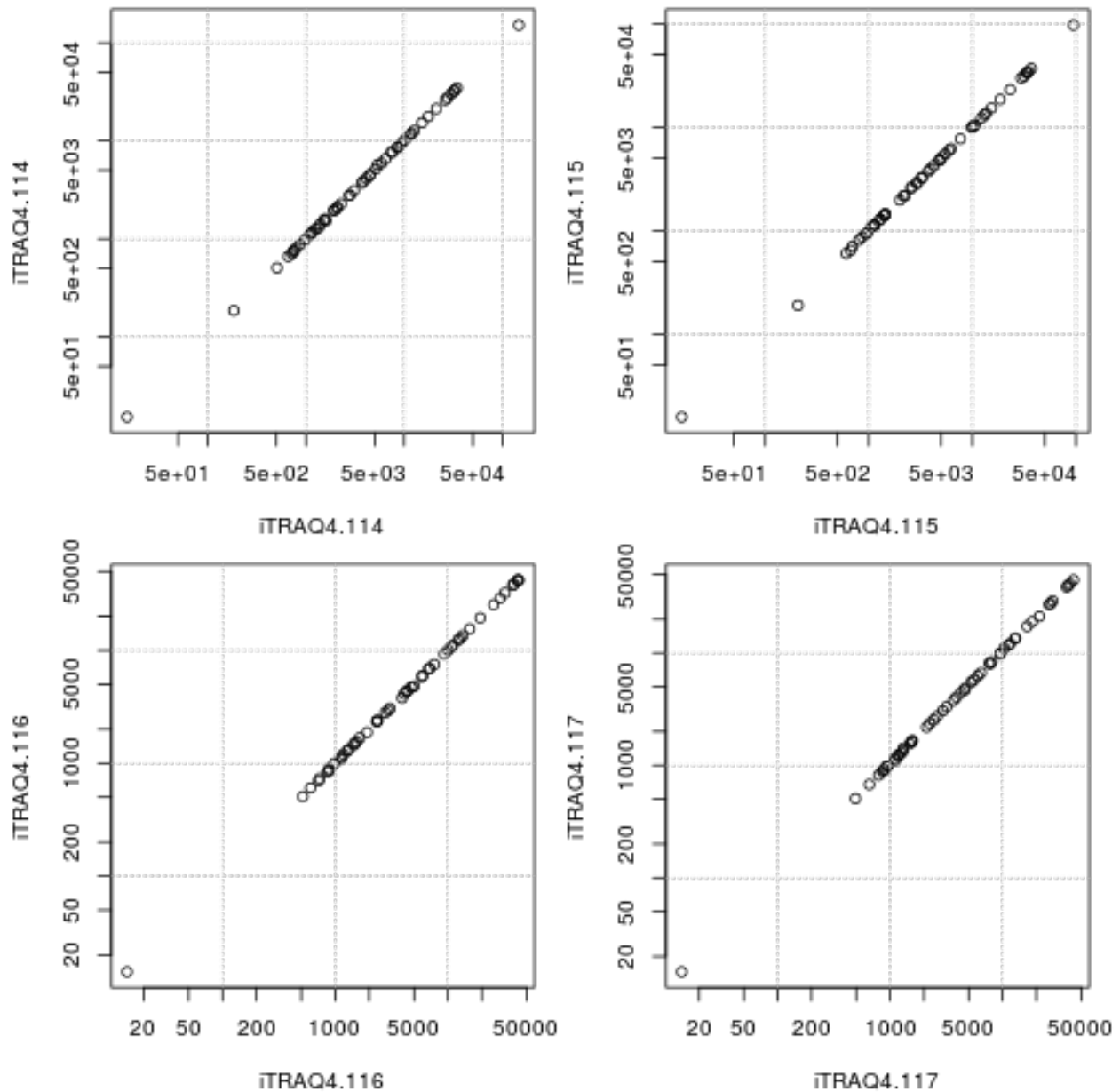
```

```

sx <- sampleNames(x)
sy <- sampleNames(y)
for (i in seq_len(ncol(x))) {
  plot(exprs(x)[, i], exprs(y)[, i], log = "xy",
       xlab = sx[i], ylab = sy[i])
  grid()
}
}

plot0(qnt, qnt.crct)

```



Various normalisation methods can be applied the MSnSet instances using the `normalise` method: variance stabilisation (`vsn`), quantile (`quantiles`), median or mean centring (`center.media` or `center.mean`), ...

```
qnt.crct.nrm <- normalise(qnt.crct, "quantiles")
plot0(qnt, qnt.crct.nrm)
```

The `combineFeatures` method combines spectra/peptides quantitation values into protein data. The grouping is defined by the `groupBy` parameter, which is generally taken from the feature metadata (protein accessions, for example).

```
## arbitrary grouping
g <- factor(c(rep(1, 25), rep(2, 15), rep(3, 15)))
prt <- combineFeatures(qnt.crct.nrm, groupBy = g, fun = "sum")
```

```
## Combined 55 features into 3 using sum
```

```
processingData(prt)
```

```
## - - - Processing information - - -
## Data loaded: Wed May 11 18:54:39 2011
## iTRAQ4 quantification by trapezoidation: Wed Nov 19 14:44:25 2014
## Purity corrected: Wed Nov 19 14:44:25 2014
## Normalised (quantiles): Wed Nov 19 14:44:26 2014
## Combined 55 features into 3 using sum: Wed Nov 19 14:44:26 2014
## MSnbase version: 1.1.22
```

Finally, proteomics data analysis is generally hampered by missing values. Missing data imputation is a sensitive operation whose success will be guided by many factors, such as degree and (non-)random nature of the missingness. Missing value in `MSnSet` instances can be filtered out and imputed using the `filterNA` and `impute` functions.

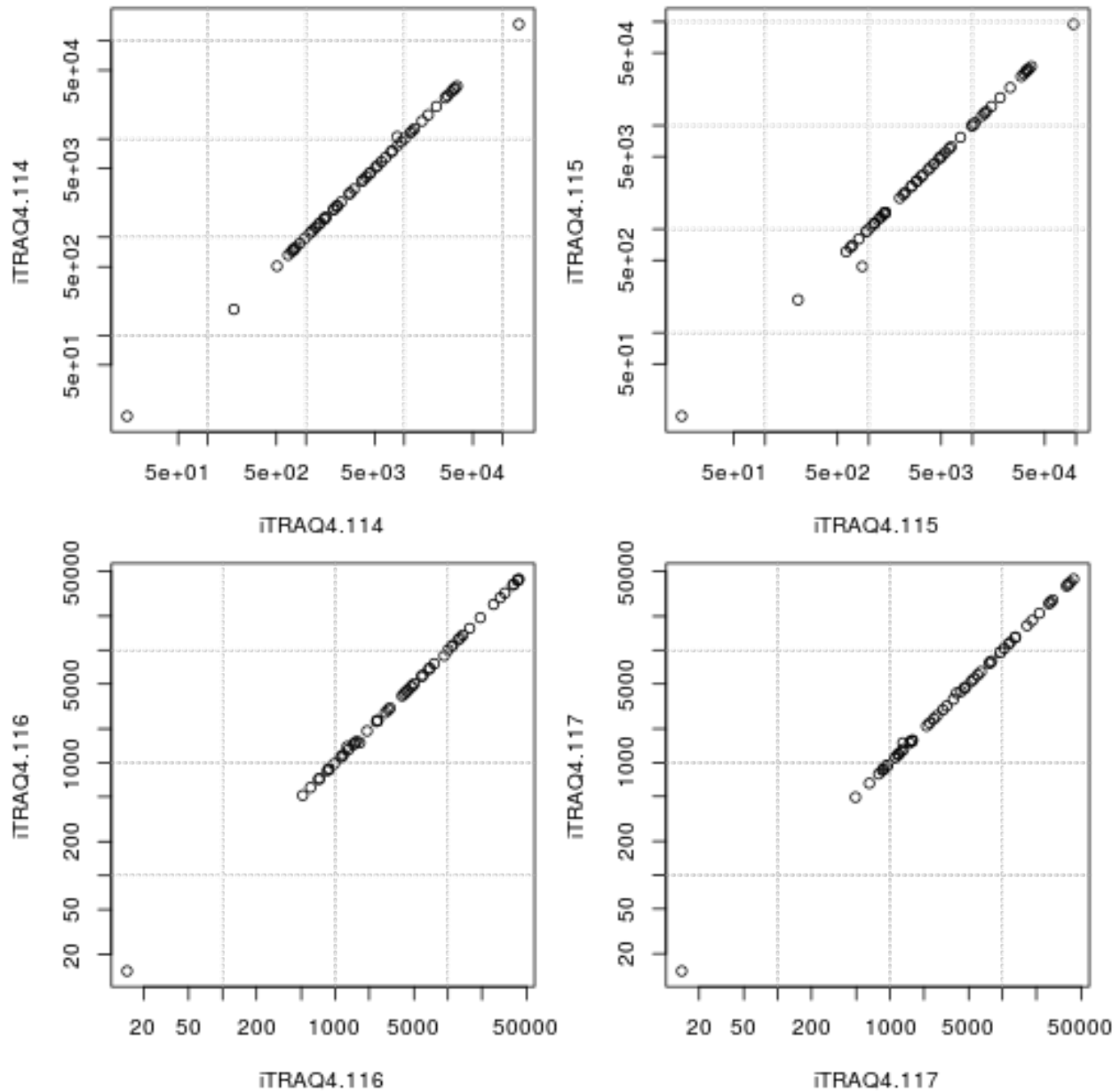
```
set.seed(1)
qnt0 <- qnt
exprs(qnt0)[sample(prod(dim(qnt0)), 10)] <- NA
table(is.na(qnt0))
```

```
##
## FALSE TRUE
##    209    11
```

```
qnt00 <- filterNA(qnt0)
dim(qnt00)
```

```
## [1] 44  4
```

```
qnt.imp <- impute(qnt0)
plot0(qnt, qnt.imp)
```

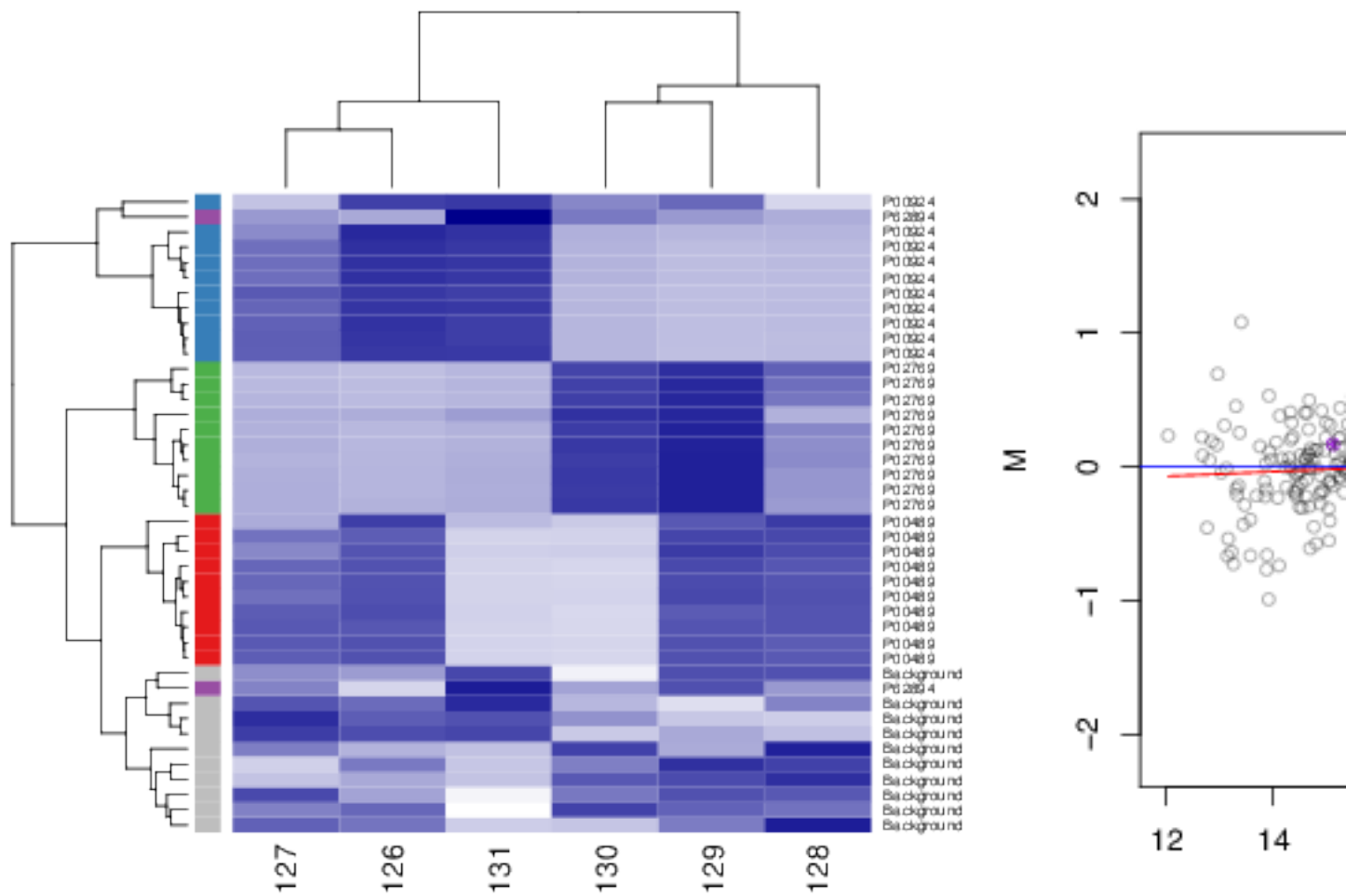
Exercise

The `mzt` instance created from the `mzTab` file has the following is a TMT 6-plex with the following design:

In this TMT 6-plex experiment, four exogenous proteins were spiked into an equimolar *Erwinia carotovora* lysate with varying proportions in each channel of quantitation; yeast enolase (ENO) at 10:5:2.5:1:2.5:10, bovine serum albumin (BSA) at 1:2.5:5:10:5:1, rabbit glycogen phosphorylase (PHO) at 2:2:2:2:1:1 and bovin cytochrome C (CYT) at 1:1:1:1:1:2. Proteins were then digested, differentially labelled with TMT reagents, fractionated by reverse phase nanoflow UPLC (nanoACQUITY, Waters), and analysed on an LTQ Orbitrap Velos mass spectrometer (Thermo Scientific).

Explore the `mzt` data using some of the illustrated functions. The heatmap and MAplot (see

MAplot function), taken from the [RforProteomics](#) vignette, have been produced using the same data.



Statistical analysis

R in general and Bioconductor in particular are well suited for the statistical analysis of data. Several packages provide dedicated resources for proteomics data:

- **MSstats**: A set of tools for statistical relative protein significance analysis in DDA, SRM and DIA experiments.
- **msmsTest**: Statistical tests for label-free LC-MS/MS data by spectral counts, to discover differentially expressed proteins between two biological conditions. Three tests are available: Poisson GLM regression, quasi-likelihood GLM regression, and the negative binomial of the **edgeR** package.

```
library(msmsTests)
```

```
## Error in library(msmsTests): there is no package called 'msmsTests'
```

```

data(msms.dataset)

## Warning in data(msms.dataset): data set 'msms.dataset' not found

msms.dataset

## Error in eval(expr, envir, enclos): object 'msms.dataset' not found

e <- pp.msms.data(msms.dataset)

## Error in eval(expr, envir, enclos): could not find function "pp.msms.data"

e

## Error in eval(expr, envir, enclos): object 'e' not found

null.f <- "y~batch"
alt.f <- "y~treat+batch"
div <- apply(exprs(e),2,sum)

## Error in exprs(e): error in evaluating the argument 'object' in selecting a method for function 'exprs'

res <- msms.edgeR(e,alt.f,null.f,div=div,fnm="treat")

## Error in eval(expr, envir, enclos): could not find function "msms.edgeR"

head(res)

## MSnSet (storageMode: lockedEnvironment)
## assayData: 1 features, 4 samples
##   element names: exprs
## protocolData: none
## phenoData: none
## featureData
##   featureNames: 1
##   fvarLabels: Protein.ID FBgn ... pd.markers (12 total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
## Subset [888,4][1,4] Wed Nov 19 14:44:26 2014
## MSnbase version: 1.14.0

```

- [isobar](#) also provides dedicated infrastructure for the statistical analysis of isobaric data.

Machine learning

The [MLInterfaces](#) package provides a unified interface to a wide range of machine learning algorithms. Initially developed for microarray and `ExpressionSet` instances, the [pRoloc](#) package enables application of these algorithms to `MSnSet` data.

```

library("MLInterfaces")
library("pRoloc")
library("pRolocdata")
data(dunkley2006)
traininds <- which(fData(dunkley2006)$markers != "unknown")
ans <- MLearn(markers ~ ., data = t(dunkley2006), knnI(k = 5), traininds)
ans

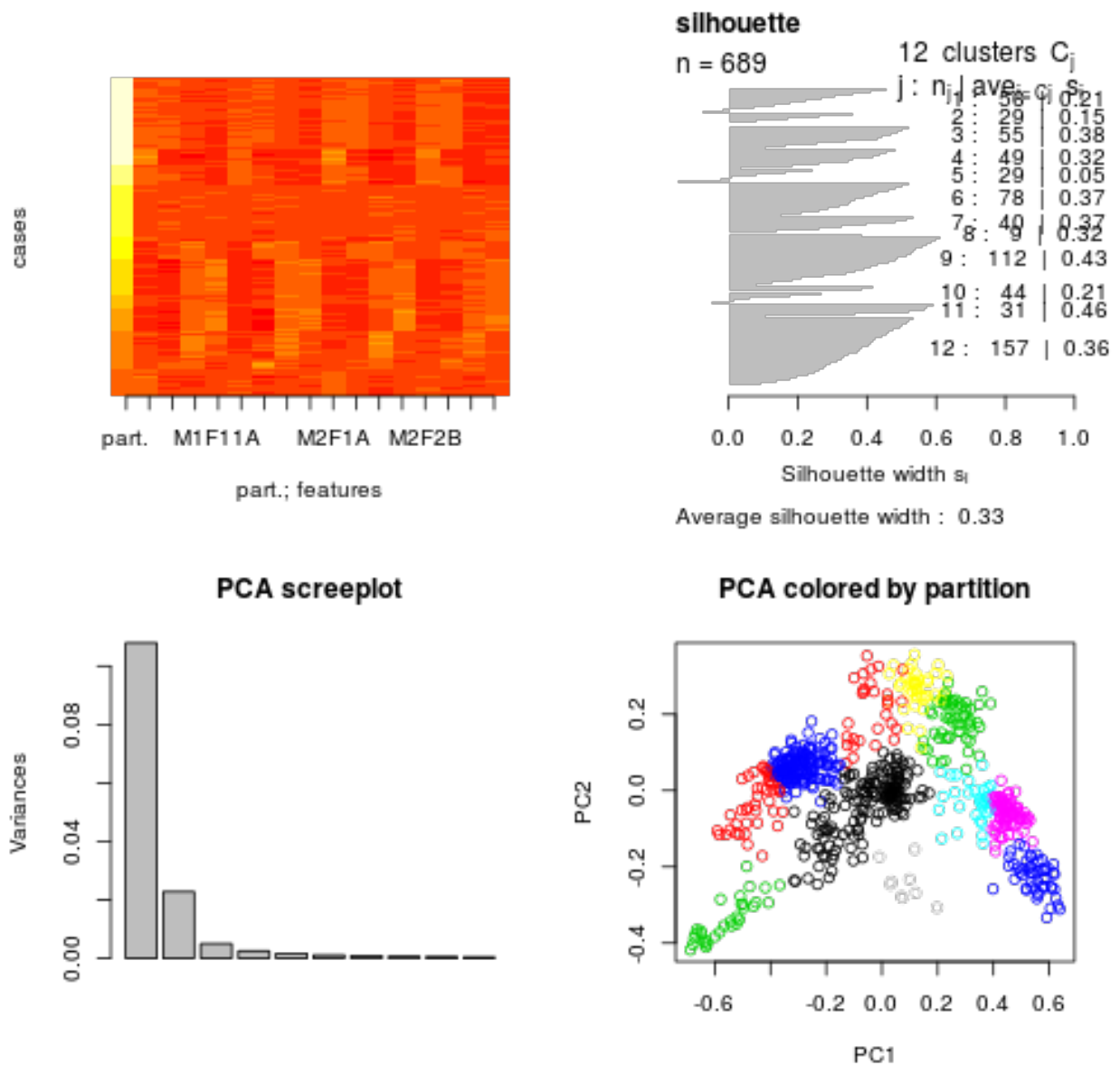
## MLInterfaces classification output container
## The call was:
## MLearn(formula = markers ~ ., data = t(dunkley2006), .method = knnI(k = 5),
##       trainInd = traininds)
## Predicted outcome distribution for test set:
##
##      ER lumen   ER membrane   Golgi Mitochondrion   Plastid
##           5         140         67         51         29
##      PM      Ribosome      TGN      vacuole
##      89         31         6         10
## Summary of scores on test set (use testScores() method for details):
##   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.4000 1.0000  1.0000 0.9332 1.0000 1.0000

kcl <- MLearn( ~ ., data = dunkley2006, kmeansI, centers = 12)
kcl

## clusteringOutput: partition table
##
##   1  2  3  4  5  6  7  8  9 10 11 12
## 56 29 55 49 29 78 40  9 112 44 31 157
## The call that created this object was:
## MLearn(formula = ~., data = dunkley2006, .method = kmeansI, centers = 12)

plot(kcl, exprs(dunkley2006))

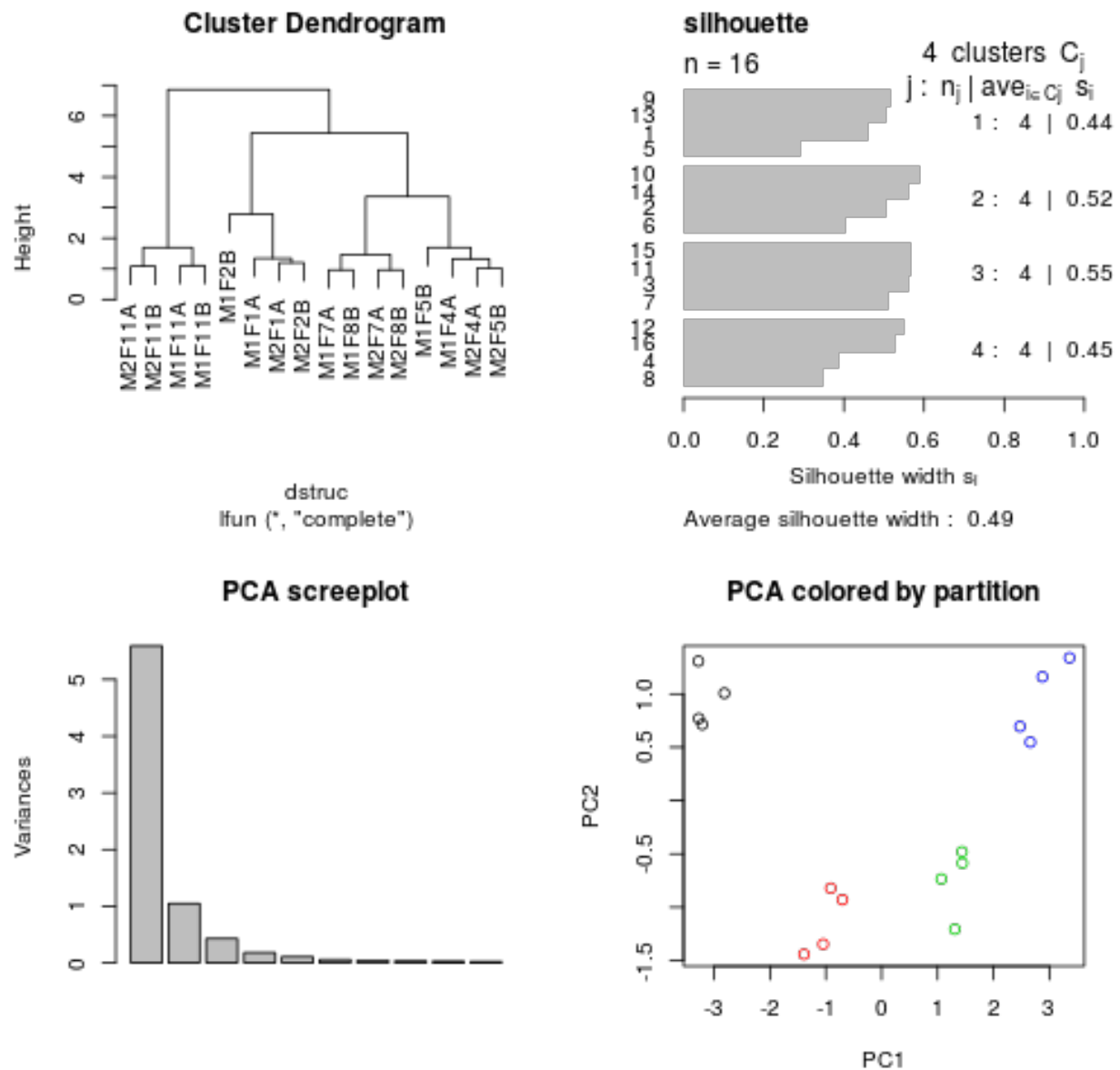
```



```
hcl <- MLearn( ~ ., data = t(dunkley2006), hclustI(distFun = dist, cutParm = list(k = 4)))
hcl
```

```
## clusteringOutput: partition table
##
## 1 2 3 4
## 4 4 4 4
## The call that created this object was:
## MLearn(formula = ~., data = t(dunkley2006), .method = hclustI(distFun = dist,
##   cutParm = list(k = 4)))
```

```
plot(hcl, exprs(t(dunkley2006)))
```



A wide range of classification algorithms are also available, as described in the [?Mlearn](#) documentation page. The `pRoloc` package also uses `MSnSet` instances as input and, while being conceived with the analysis of spatial/organelle proteomics data in mind, is applicable many use cases.

Annotation

All the [Bioconductor annotation infrastructure](#), such as [biomaRt](#), [GO.db](#), organism specific annotations, .. are directly relevant to the analysis of proteomics data. A total of 93 ontologies, including some proteomics-centred annotations such as the PSI Mass Spectrometry Ontology, Molecular Interaction (PSI MI 2.5) or Protein Modifications are available through the [rols](#).

```
library("rols")
olsQuery("ESI", "MS")
```

```
## MS:1000073 MS:1000162
##      "ESI" "HiRes ESI"
```

Data from the [Human Protein Atlas](#) is available via the [hpar](#) package.

Other relevant packages/pipelines

- Analysis of post translational modification with [isobar](#).
- Analysis of label-free data from a Synapt G2 (including ion mobility) with [synapter](#).
- Analysis of spatial proteomics data with [pRoloc](#).
- Analysis of MALDI data with the [MALDIquant](#) package.
- Access to the Proteomics Standard Initiative Common QUery InterfaCe with the [PSICQUIC](#) package.

Additional relevant packages are described in the [RforProteomics](#) vignettes.

Session information

```
## R version 3.1.1 Patched (2014-09-02 r66514)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## attached base packages:
## [1] stats4      parallel  stats      graphics  grDevices  utils      datasets
## [8] methods     base
##
## other attached packages:
## [1] lattice_0.20-29      hpar_1.8.0           rols_1.8.0
## [4] MSGFgui_1.0.1         rTANDEM_1.6.0        data.table_1.9.4
## [7] pRolocdata_1.5.2     pRoloc_1.7.1         MLInterfaces_1.46.0
## [10] cluster_1.15.3       annotate_1.44.0       XML_3.98-1.1
## [13] AnnotationDbi_1.28.1 GenomeInfoDb_1.2.3   IRanges_2.0.0
## [16] S4Vectors_0.4.0      rpx_1.2.0            MSGFplus_1.0.3
## [19] MSnID_1.0.0           mzID_1.4.1           RforProteomics_1.5.2
## [22] MSnbase_1.14.0       BiocParallel_1.0.0   mzR_2.0.0
## [25] Rcpp_0.11.3          Biobase_2.26.0       BiocGenerics_0.12.1
## [28] BiocInstaller_1.16.1 knitr_1.8
##
## loaded via a namespace (and not attached):
## [1] affy_1.44.0           affyio_1.34.0
## [3] base64enc_0.1-2       BatchJobs_1.5
## [5] BBmisc_1.8            biocViews_1.34.1
## [7] BradleyTerry2_1.0-5   brew_1.0-6
## [9] brglm_0.5-9           car_2.0-21
## [11] caret_6.0-37          Category_2.32.0
## [13] checkmate_1.5.0       chron_2.3-45
## [15] class_7.3-11          codetools_0.2-9
## [17] colorspace_1.2-4      DBI_0.3.1
## [19] digest_0.6.4          doParallel_1.0.8
## [21] e1071_1.6-4           evaluate_0.5.5
## [23] fail_1.2              FNN_1.1
## [25] foreach_1.4.2         formatR_1.0
## [27] gdata_2.13.3          genefilter_1.48.1
## [29] ggplot2_1.0.0         graph_1.44.0
```

| | |
|----------------------------------|------------------------------|
| ## [31] grid_3.1.1 | gridSVG_1.4-0 |
| ## [33] GSEABase_1.28.0 | gtable_0.1.2 |
| ## [35] gtools_3.4.1 | htmltools_0.2.6 |
| ## [37] httpuv_1.3.2 | impute_1.40.0 |
| ## [39] interactiveDisplay_1.4.0 | interactiveDisplayBase_1.4.0 |
| ## [41] iterators_1.0.7 | kernlab_0.9-19 |
| ## [43] labeling_0.3 | limma_3.22.1 |
| ## [45] lme4_1.1-7 | lpSolve_5.6.10 |
| ## [47] MALDIquant_1.11 | MASS_7.3-35 |
| ## [49] Matrix_1.1-4 | mclust_4.4 |
| ## [51] mime_0.2 | minqa_1.2.4 |
| ## [53] munsell_0.4.2 | mvtnorm_1.0-0 |
| ## [55] nlme_3.1-118 | nloptr_1.0.4 |
| ## [57] nnet_7.3-8 | pcaMethods_1.56.0 |
| ## [59] pls_2.4-3 | plyr_1.8.1 |
| ## [61] preprocessCore_1.28.0 | proto_0.3-10 |
| ## [63] proxy_0.4-13 | R6_2.0.1 |
| ## [65] randomForest_4.6-10 | RBGL_1.42.0 |
| ## [67] R.cache_0.10.0 | RColorBrewer_1.0-5 |
| ## [69] RCurl_1.95-4.3 | rda_1.0.2-2 |
| ## [71] reshape2_1.4 | rJava_0.9-6 |
| ## [73] RJSONIO_1.3-0 | R.methodsS3_1.6.1 |
| ## [75] R.oo_1.18.0 | rpart_4.1-8 |
| ## [77] RSQLite_1.0.0 | RUnit_0.4.27 |
| ## [79] R.utils_1.34.0 | sampling_2.6 |
| ## [81] scales_0.2.4 | sendmailR_1.2-1 |
| ## [83] sfsmisc_1.0-26 | shiny_0.10.2.1 |
| ## [85] shinyFiles_0.4.0 | splines_3.1.1 |
| ## [87] SSOAP_0.8-0 | stringr_0.6.2 |
| ## [89] survival_2.37-7 | tools_3.1.1 |
| ## [91] vsn_3.34.0 | xlsx_0.5.7 |
| ## [93] xlsxjars_0.6.1 | XMLSchema_0.7-2 |
| ## [95] xtable_1.7-4 | zlibbioc_1.12.0 |