Mass spectrometry and proteomics data analysis

Laurent Gatto

12 June 2015

Contents

1 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("MSnbase")
library("rpx")
library("MLInterfaces")
library("pRoloc")
library("pRolocdata")
library("MSGFplus")
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)
```

Other packages of interest, such as rTANDEM or MSGFgui will be described later in the document but are not required to execute the code in this workflow.

2 Introduction

This workflow 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 quantitation 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.

3 Exploring available infrastructure

In Bioconductor version 3.2, there are respectively 71 proteomics, 51 mass spectrometry software packages and 11 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)
```

4 Mass spectrometry data

Most community-driven formats are supported in R, as detailed in the table below.

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

5 Getting data from proteomics repositories

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 access to PX data.

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

```
## 15 new ProteomeXchange annoucements

##     Data.Set    Publication.Data             Message
## 1  PXD002046 2015-06-12 14:38:10                 New
## 2  PXD002044 2015-06-12 05:50:10                 New
## 3  PXD002357 2015-06-11 20:10:15                 New
## 4  PXD002356 2015-06-11 19:53:53                 New
## 5  PXD002355 2015-06-11 19:40:52                 New
## 6  PXD000394 2015-06-11 11:47:53 Updated information
## 7  PXD002209 2015-06-11 10:11:00                 New
## 8  PXD002002 2015-06-10 10:45:43                 New
## 9  PXD001858 2015-06-09 11:30:49 Updated information
## 10 PXD001564 2015-06-09 11:29:51                 New
## 11 PXD001450 2015-06-09 11:15:55                 New
## 12 PXD002113 2015-06-08 14:40:27                 New
## 13 PXD002142 2015-06-08 13:29:25                 New
## 14 PXD002067 2015-06-08 09:11:36                 New
## 15 PXD002000 2015-06-05 14:40:25                 New
```

Using the unique PXD000001 identifier, we can retrieve the relevant metadata that will be stored in a PXDataset object. The names of the files available in this data can be retrieved with the pxfiles accessor function.

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

```
## Object of class "PXDataset"
##  Id: PXD000001 with 10 files
##  [1] 'F063721.dat' ... [10] '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_01-20141210.mzML"
##  [7] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzXML"
##  [8] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML"
##  [9] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.raw"
## [10] "erwinia_carotovora.fasta"
```

Other metadata for the px data set:

```
pxtax(px)
```

```
## [1] "Erwinia carotovora"
```

```
pxurl(px)
```

```
## [1] "ftp://ftp.pride.ebi.ac.uk/pride/data/archive/2012/03/PXD000001"
```

```
pxref(px)
```

```
## [1] "Gatto L, Christoforou A. Using R and Bioconductor for proteomics data analysis. Biochim Biophys Ac
```

Data files can then be downloaded with the pxget function. Below, we retrieve the sixth file, TMT_Erwinia_1uLSike_Top10HCD_isol2_4 20141210.mzML. The file is downloaded in the working directory and the name of the file is return by the function and stored in the `mzf` variable for later use.

```
mzf <- pxget(px, pxfiles(px)[6])
```

```
## Downloading 1 file
```

```
mzf
```

```
## [1] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzML"
```

6 Handling raw MS data

The `mzR` package provides an interface to the proteowizard C/C++ code base to access various raw data files, such as `mzML`, `mzXML`, `netCDF`, and `mzData`. The data is accessed on-disk, i.e it is not loaded entirely in memory by default but only when explicitly requested. 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.

Below, we access the raw data file downloaded in the previous section and open a file handle that will allow us to extract data and metadata of interest.

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

```
## Mass Spectrometry file handle.
## Filename:  TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzML
## Number of scans:  7534
```

The header function returns the metadata of all available peaks:

```
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"
```

We can extract metadata and scan data for scan 1000 as follows:

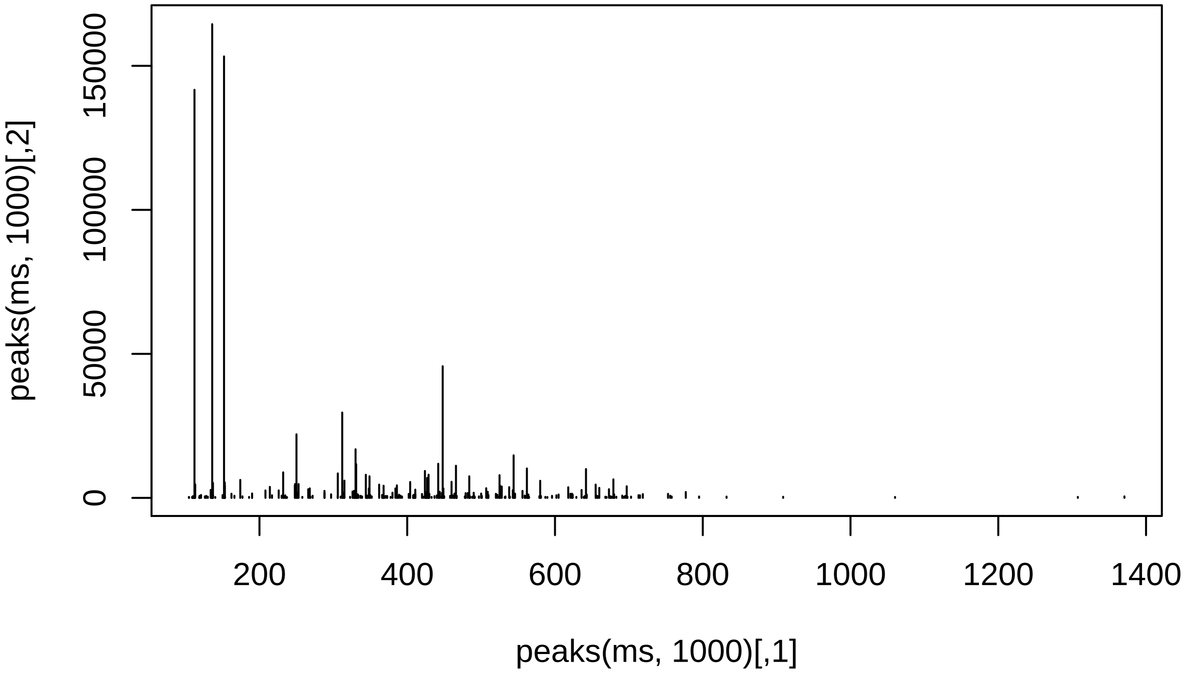
```
hd[1000, ]
```

```
##      seqNum acquisitionNum msLevel polarity peaksCount totIonCurrent
## 1000   1000           1000       2       -1        274       1048554
##      retentionTime basePeakMZ basePeakIntensity collisionEnergy
## 1000      1106.916    136.061            164464              45
##      ionisationEnergy    lowMZ   highMZ precursorScanNum precursorMZ
## 1000                0 104.5467 1370.758              992    683.0817
##      precursorCharge precursorIntensity mergedScan mergedResultScanNum
## 1000               2           689443.7          0                   0
##      mergedResultStartScanNum mergedResultEndScanNum
## 1000                        0                      0
```

```
head(peaks(ms, 1000))
```

```
##          [,1]     [,2]
## [1,] 104.5467 308.9326
## [2,] 104.5684 308.6961
## [3,] 108.8340 346.7183
## [4,] 109.3928 365.1236
## [5,] 110.0345 616.7905
## [6,] 110.0703 429.1975
```

```
plot(peaks(ms, 1000), type = "h")
```

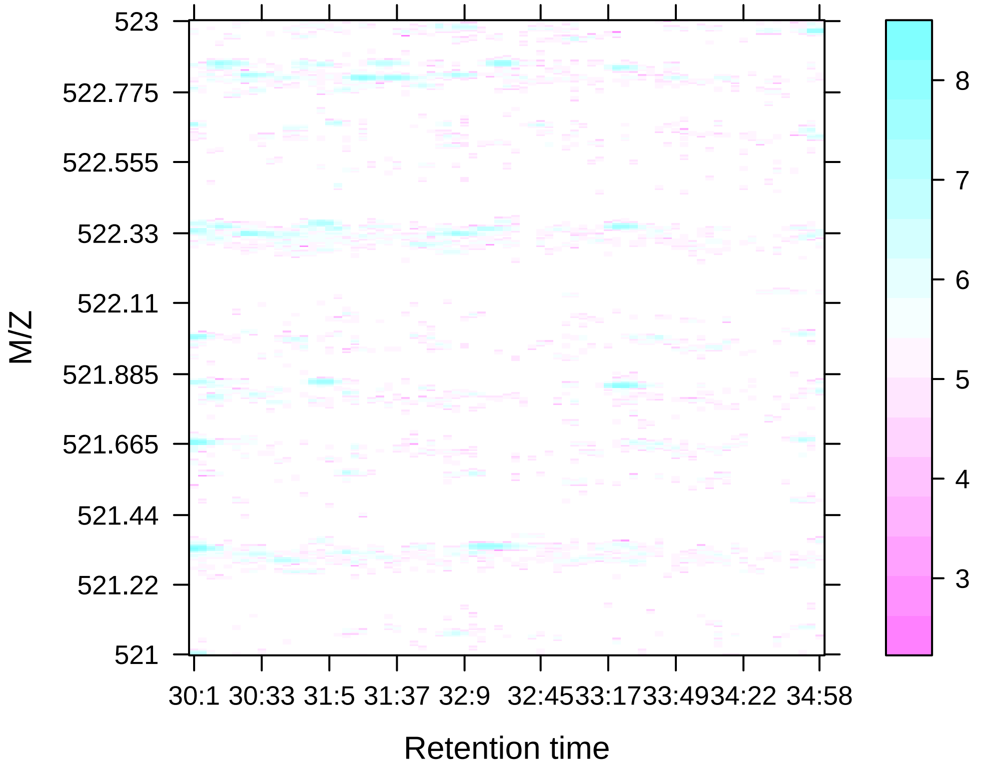


Below we reproduce the example from the `MSmap` function from the `MSnbase` package to plot a specific slice of the raw data using the `mzR` functions we have just described.

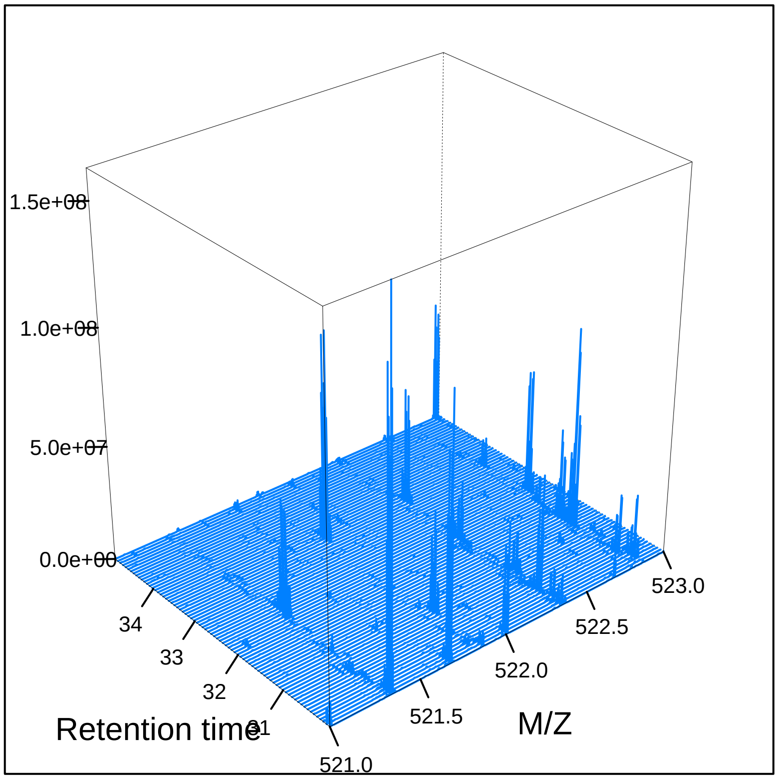
```
## a set of spectra of interest: MS1 spectra eluted
## between 30 and 35 minutes retention time
ms1 <- which(hd$msLevel == 1)
rtsel <- hd$retentionTime[ms1] / 60 > 30 &
    hd$retentionTime[ms1] / 60 < 35

## the map
M <- MSmap(ms, ms1[rtsel], 521, 523, .005, hd)
```

```
## 1
```

```
plot(M, aspect = 1, allTicks = FALSE)
```



```
plot3D(M)
```

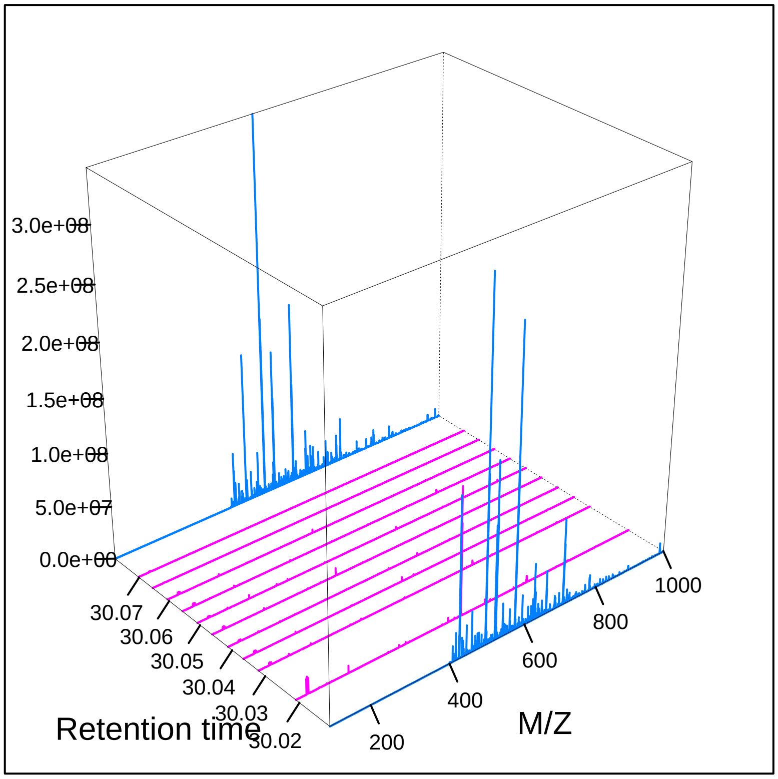


```
## With some MS2 spectra
i <- ms1[which(rtsel)][1]
j <- ms1[which(rtsel)][2]
```

```
M2 <- MSmap(ms, i:j, 100, 1000, 1, hd)
```

```
## 1
```

```
plot3D(M2)
```



7 Handling identification data

The `RforProteomics` package distributes a small identification result file (see `?TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_`
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_4
##
```

```
## 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.

The mzR package also provides support fast parsing mzIdentML files with the openIDfile function. As for raw data, the underlying C/C++ code comes from the proteowizard.

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

id1 <- openIDfile(f)
fid1 <- mzR::psms(id1)

head(fid1)
```

```
##   spectrumID chargeState rank passThreshold experimentalMassToCharge
## 1  scan=5782           3    1          TRUE                1080.2325
## 2  scan=6037           3    1          TRUE                1002.2089
## 3  scan=5235           3    1          TRUE                1189.2836
## 4  scan=5397           3    1          TRUE                 960.5365
## 5  scan=6075           3    1          TRUE                1264.3409
##   calculatedMassToCharge                            sequence modNum
## 1              1080.2321 PVQIQAGEDSNVIGALGGAVLGGFLGNTIGGGSGR      0
## 2              1002.2115        TQVLDGLINANDIEVPVALIDGEIDVLR      0
## 3              1189.2800   TKGLNVMQNLLTAHPDVQAVFAQNDEMALGALR      0
## 4               960.5365         SQILQQAGTSVLSQANQVPQTVLSLLR      0
## 5              1264.3419 PIIGDNPFVVVLPDVVLDESTADQTQENLALLISR      0
##   isDecoy post pre start end DatabaseAccess DBseqLength DatabaseSeq
## 1   FALSE    S   R    50  84        ECA1932         155
## 2   FALSE    R   K   288 315        ECA1147         434
## 3   FALSE    A   R   192 224        ECA0013         295
## 4   FALSE    -   R   264 290        ECA1731         290
## 5   FALSE    F   R   119 153        ECA1443         298
##                                          DatabaseDescription acquisitionNum
## 1                         ECA1932 outer membrane lipoprotein           5782
## 2                                     ECA1147 trigger factor           6037
## 3                 ECA0013 ribose-binding periplasmic protein           5235
## 4                                          ECA1731 flagellin           5397
## 5     ECA1443 UTP--glucose-1-phosphate uridylyltransferase           6075
##  [ reached getOption("max.print") -- omitted 1 row ]
```

8 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 an experimental 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.

We search the TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzML file against the fasta file from `PXD000001` using `MSGFplus`.

We first download the fasta files:

```
fas <- pxget(px, pxfiles(px)[10])
```

```
## Downloading 1 file
```

```
basename(fas)
```

```
## [1] "erwinia_carotovora.fasta"
```

```
library("MSGFplus")
msgfpar <- msgfPar(database = fas,
                   instrument = 'HighRes',
                   tda = TRUE,
                   enzyme = 'Trypsin',
                   protocol = 'iTRAQ')
idres <- runMSGF(msgfpar, mzf, memory=1000)
```

```
## java -Xmx1000M -jar '/home/lg390/R/x86_64-unknown-linux-gnu-library/3.2/MSGFplus/MSGFPlus/MSGFPlus.jar'
##
## reading TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzid... DONE!
```

```
idres
```

```
## An mzID object
##
## Software used:   MS-GF+ (version: Beta (v10072))
##
## Rawfile:         /home/lg390/Documents/Teaching/CSAMA_Brixen/Bressanone2015/CSAMA2015/materials/labs/5_
##
## Database:        /home/lg390/Documents/Teaching/CSAMA_Brixen/Bressanone2015/CSAMA2015/materials/labs/5_
##
## Number of scans: 5343
## Number of PSM's: 5656
```

```
## identification file (needed below)
basename(mzID::files(idres)$id)
```

```
## [1] "TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzid"
```

(Note that in the `runMSGF` call above, I explicitly reduce the memory allocated to the java virtual machine to 3.5GB. In general, there is no need to specify this argument, unless you experience an error regarding the *maximum heap size*).

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

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

9 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 mzIdenML 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
```

```
msnid <- read_mzIDs(msnid,
                    basename(mzID::files(idres)$id))
```

```
## Loaded cached data
```

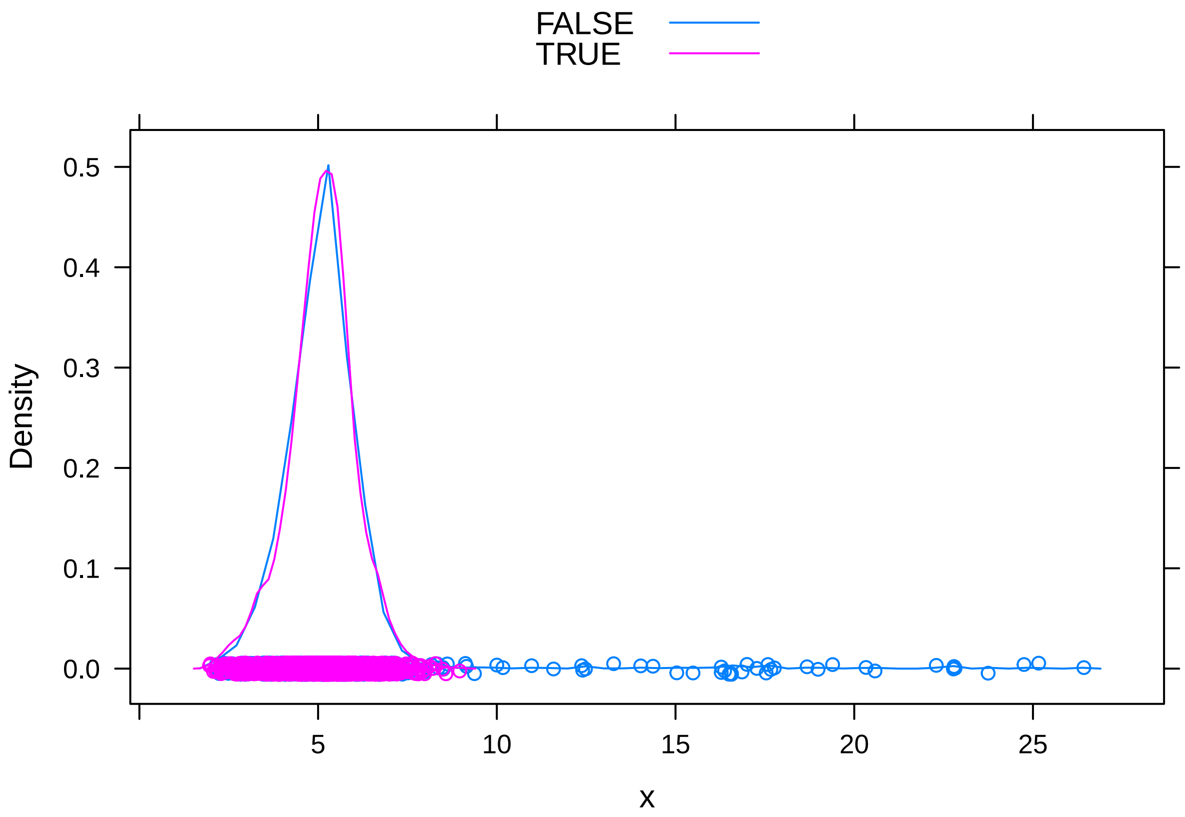
```
show(msnid)
```

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 5759 at 100 % FDR
## #peptides: 4942 at 99 % FDR
## #accessions: 3148 at 100 % 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. Below, we start by apply a correction of monoisotopic peaks (see ?correct_peak_selection for details) and define two variables to be used for identification filtering.

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

As shown below, this particular spiked-in data set displays few high scoring non-decoy hits



We define a filter object, assigning arbitrary threshold and evaluate it on the `msnid` data

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

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

```
evaluate_filter(msnid, filtObj)
```

```
##                  fdr  n
## PSM       0.04545455 23
## peptide   0.06250000 17
## accession 0.05882353 18
```

We can also optimise the filtering with a target protein FDR value of 0.01

```
filtObj.grid <- optimize_filter(filtObj, msnid, fdr.max=0.01,
                                method="Grid", level="PSM",
                                n.iter=50000)
filtObj.grid
```

```
## MSnIDFilter object
## (absParentMassErrorPPM < 12) & (msmsScore > 8.1)
```

```
evaluate_filter(msnid, filtObj.grid)
```

```
##           fdr  n
## PSM         0 39
## peptide     0 31
## accession   0 26
```

We can now apply the filter to the data

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

```
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 39 at 0 % FDR
## #peptides: 31 at 0 % FDR
## #accessions: 26 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.

10 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, see below), 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.

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 object of class MSnExp.

(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: Fri Jun 12 19:47:43 2015
##  MSnbase version: 1.17.5
## - - - 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 as a subset of the original `MSnExp` data with the [operator. Individual spectra can be accessed with `[[`.

```
length(msexp)
```

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

```
msexp[1:2]
```

```
## Object of class "MSnExp"
##  Object size in memory: 0.09 Mb
## - - - Spectra data - - -
##  MS level(s): 2
##  Number of MS1 acquisitions: 1
##  Number of MSn scans: 2
##  Number of precursor ions: 2
##  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: Fri Jun 12 19:47:43 2015
## Data [numerically] subsetted 2 spectra: Fri Jun 12 19:47:43 2015
##  MSnbase version: 1.17.5
## - - - Meta data  - - -
## phenoData
##   rowNames: 1
##   varLabels: sampleNames
##   varMetadata: labelDescription
## Loaded from:
##   dummyiTRAQ.mzXML
## protocolData: none
## featureData
##   featureNames: X1.1 X2.1
##   fvarLabels: spectrum
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
```

```
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
##      experimentalmasstocharge chargestate ms-gf:denovoscore ms-gf:evalue
## X1.1                 645.3741           3                77     79.36958
## X2.1                 546.9586           3                39     13.46615
## X3.1                       NA          NA                NA           NA
##      ms-gf:rawscore ms-gf:specevalue assumeddissociationmethod
## X1.1            -39     5.527468e-05                       CID
## X2.1            -30     9.399048e-06                       CID
## X3.1             NA               NA                      <NA>
##      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
##                                                                       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>
##                 pepseq modified modification          idFile
## X1.1 VESITARHGEVLQLRPK    FALSE           NA dummyiTRAQ.mzid
## X2.1     IDGQWVTHQWLKK    FALSE           NA dummyiTRAQ.mzid
## X3.1              <NA>       NA           NA            <NA>
##                  databaseFile nprot npep.prot npsm.prot npsm.pep
## X1.1 erwinia_carotovora.fasta     2         1         1        1
## X2.1 erwinia_carotovora.fasta     1         1         1        1
## X3.1                     <NA>    NA        NA        NA       NA
##  [ reached getOption("max.print") -- omitted 2 rows ]
```

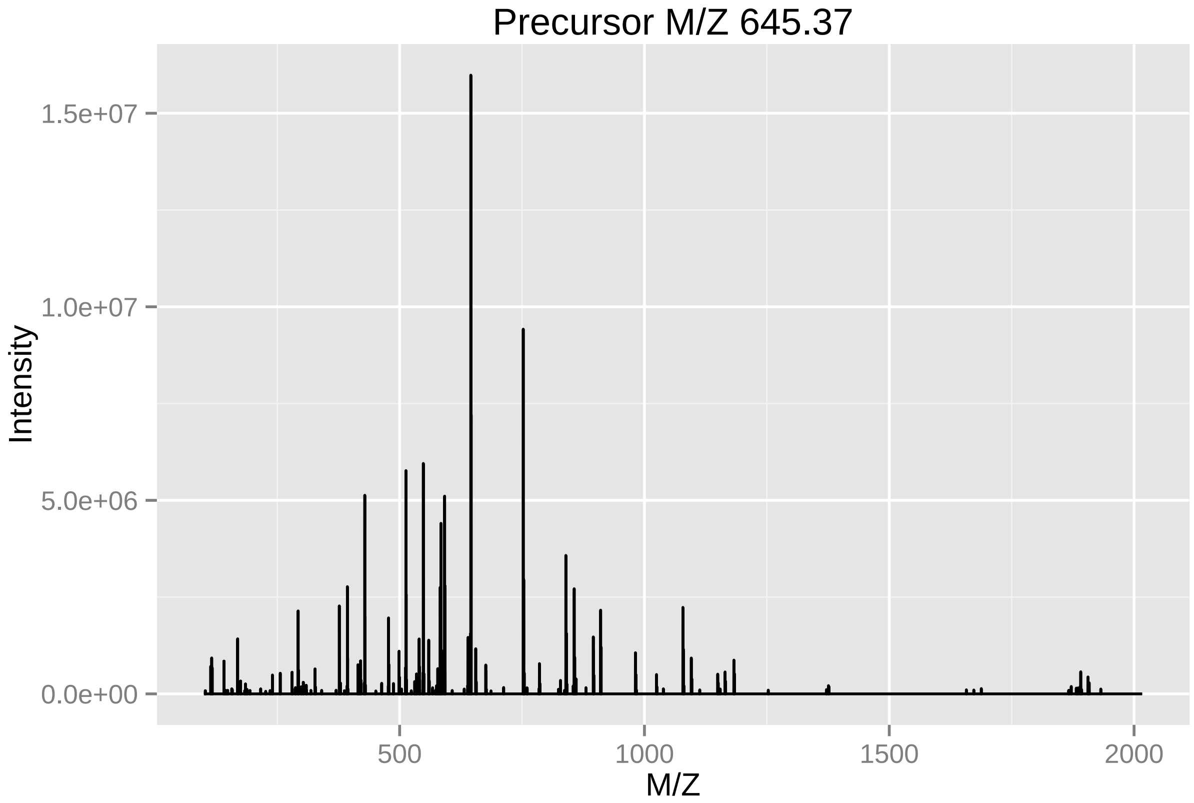
The readMSData and addIdentificationData make use of mzR and mzID packages to access the raw and identification data.

Spectra and (parts of) experiments can be extracted 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
```

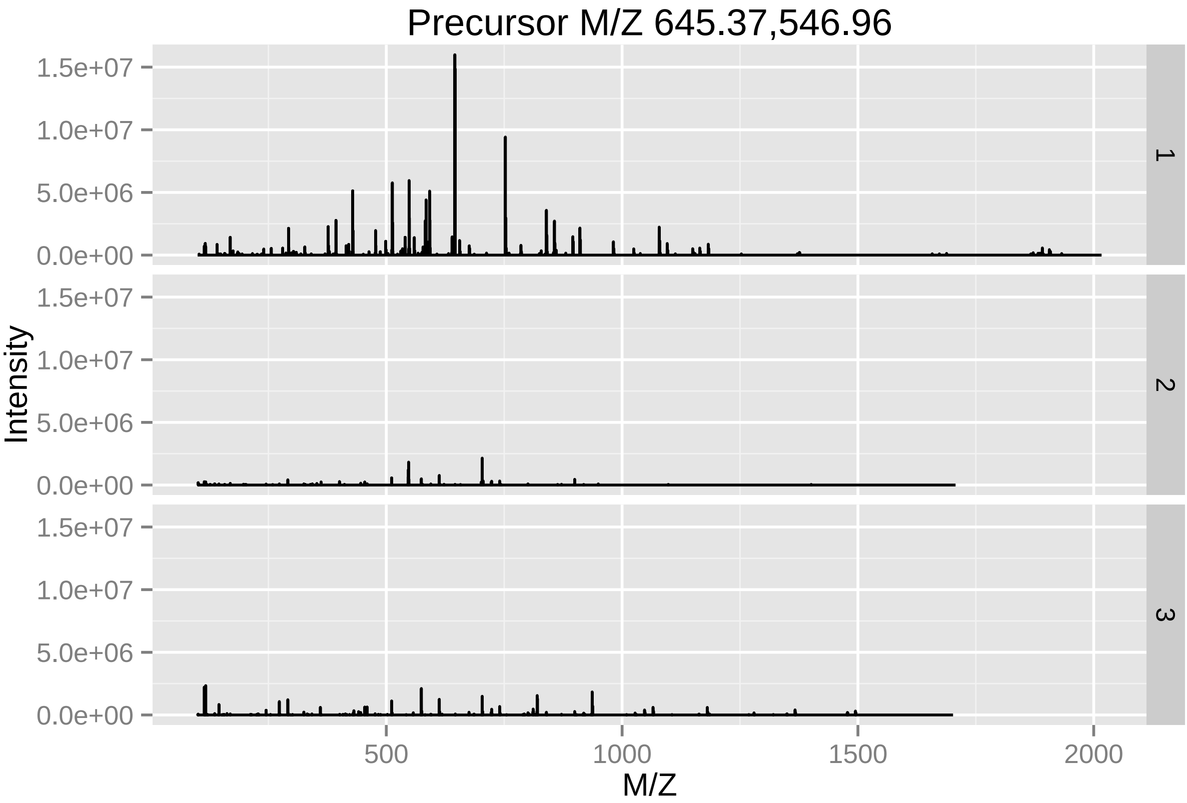
```
plot(msexp[[1]], full=TRUE)
```



```
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: Fri Jun 12 19:47:43 2015
## Data [numerically] subsetted 3 spectra: Fri Jun 12 19:47:44 2015
##  MSnbase version: 1.17.5
## - - - Meta data  - - -
## phenoData
##   rowNames: 1
##   varLabels: sampleNames
##   varMetadata: labelDescription
## Loaded from:
##   dummyiTRAQ.mzXML
## 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)
```



11 Quantitative proteomics

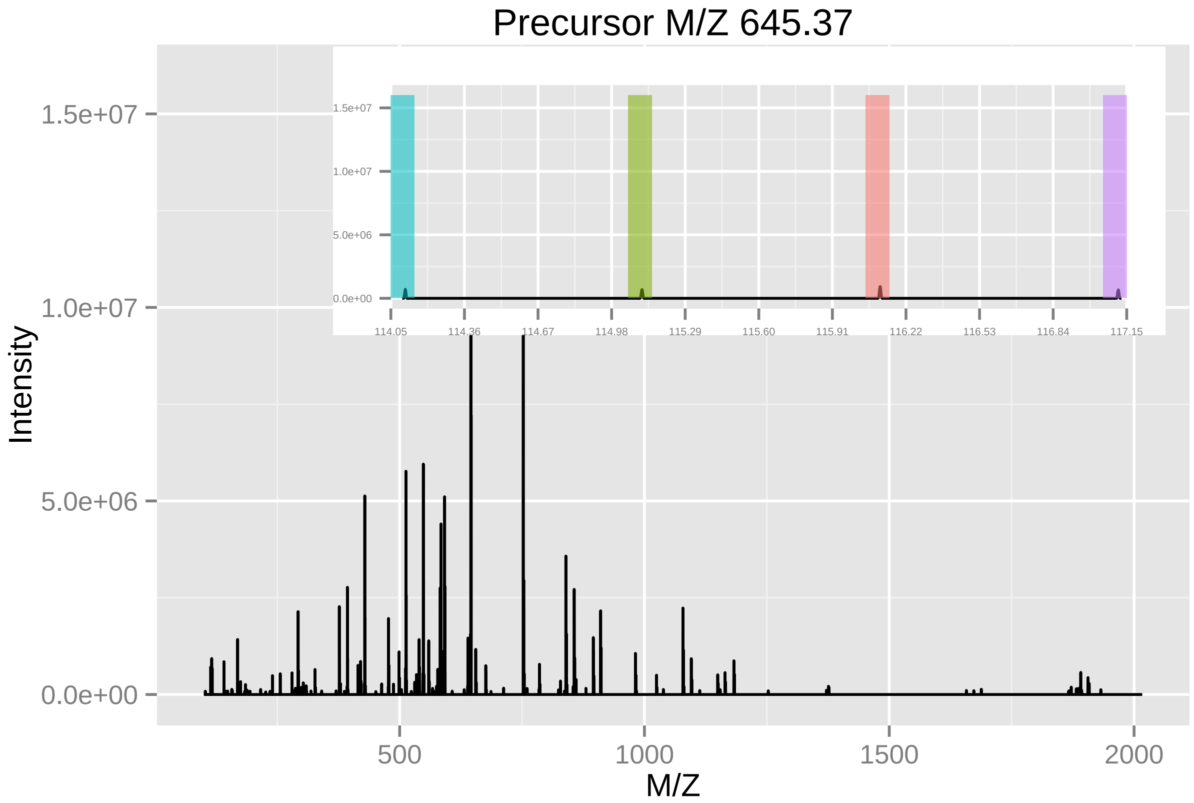
There are a wide range of proteomics quantitation techniques that can broadly be classified as labelled vs. label-free, depending whether the features are labelled prior the MS acquisition and the MS level at which quantitation is inferred, namely MS1 or MS2.

	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) and the trapezoidation method to calculate the area under the isobaric reporter peaks.

```
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: Fri Jun 12 19:47:43 2015
## iTRAQ4 quantification by trapezoidation: Fri Jun 12 19:47:48 2015
##  MSnbase version: 1.17.5
```

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.

12 Importing third-party quantitation 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])
```

```
## Downloading 1 file
```

```
(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: Fri Jun 12 19:17:28 2015
##  MSnbase version: 1.17.5
```

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. The latter can be queried with the getEcols function.

```
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
```

13 Data processing and analysis

13.1 Raw data processing

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

The MALDIquant and xcms packages also features a wide range of raw data processing methods on their own ad hoc data instance types.

13.2 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: Fri Jun 12 19:47:54 2015
## Purity corrected: Fri Jun 12 19:47:54 2015
##  MSnbase version: 1.1.22
```

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")
```

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).

```
## arbitraty grouping
g <- factor(c(rep(1, 25), rep(2, 15), rep(3, 15)))
g
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## Levels: 1 2 3
```

```
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: Fri Jun 12 19:47:54 2015
## Purity corrected: Fri Jun 12 19:47:54 2015
## Normalised (quantiles): Fri Jun 12 19:47:54 2015
## Combined 55 features into 3 using sum: Fri Jun 12 19:47:54 2015
##  MSnbase version: 1.1.22
```

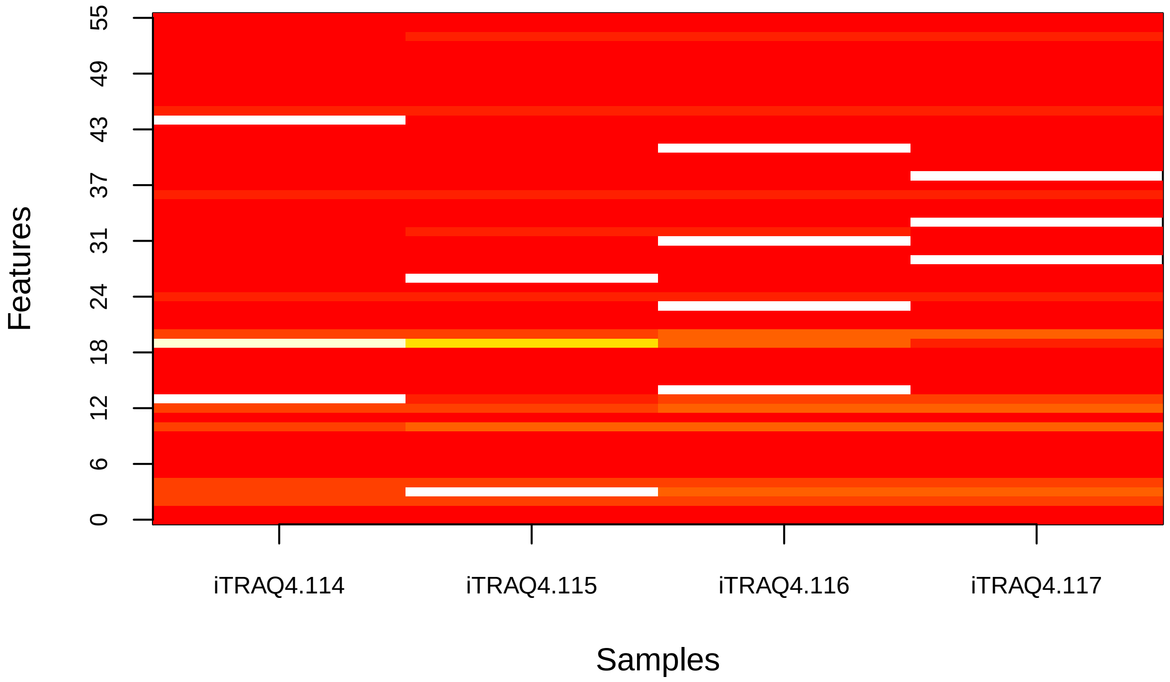
Finally, proteomics data analysis is generally hampered by missing values. Missing data imputation is a sensitive operation whose success will be guided by many factors, such as degree and (non-)random nature of the missingness.

Below, missing values are randomly assigned to our test data and visualised on a heatmap.

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

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

```
image(qnt0)
```



Missing value in `MSnSet` instances can be filtered out and imputed using the `filterNA` and `impute` functions.

```
## remove features with missing values
qnt00 <- filterNA(qnt0)
dim(qnt00)
```

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

```
any(is.na(qnt00))
```

```
## [1] FALSE
```

```
## impute missing values using knn imputation
qnt.imp <- impute(qnt0, method = "knn")
dim(qnt.imp)
```

```
## [1] 55  4
```

```
any(is.na(qnt.imp))
```

```
## [1] FALSE
```

There are various methods to perform data imputation, as described in ?impute.

14 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. Data stored in data.frame or MSnSet objects can be used as input.
- 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. All can be readily applied on MSnSet instances produced, for example by MSnID.
- isobar also provides dedicated infrastructure for the statistical analysis of isobaric data.

15 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.

15.1 Classification

The example below uses knn with the 5 closest neighbours as an illustration to classify proteins of unknown sub-cellular localisation to one of 9 possible organelles.

```
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
```

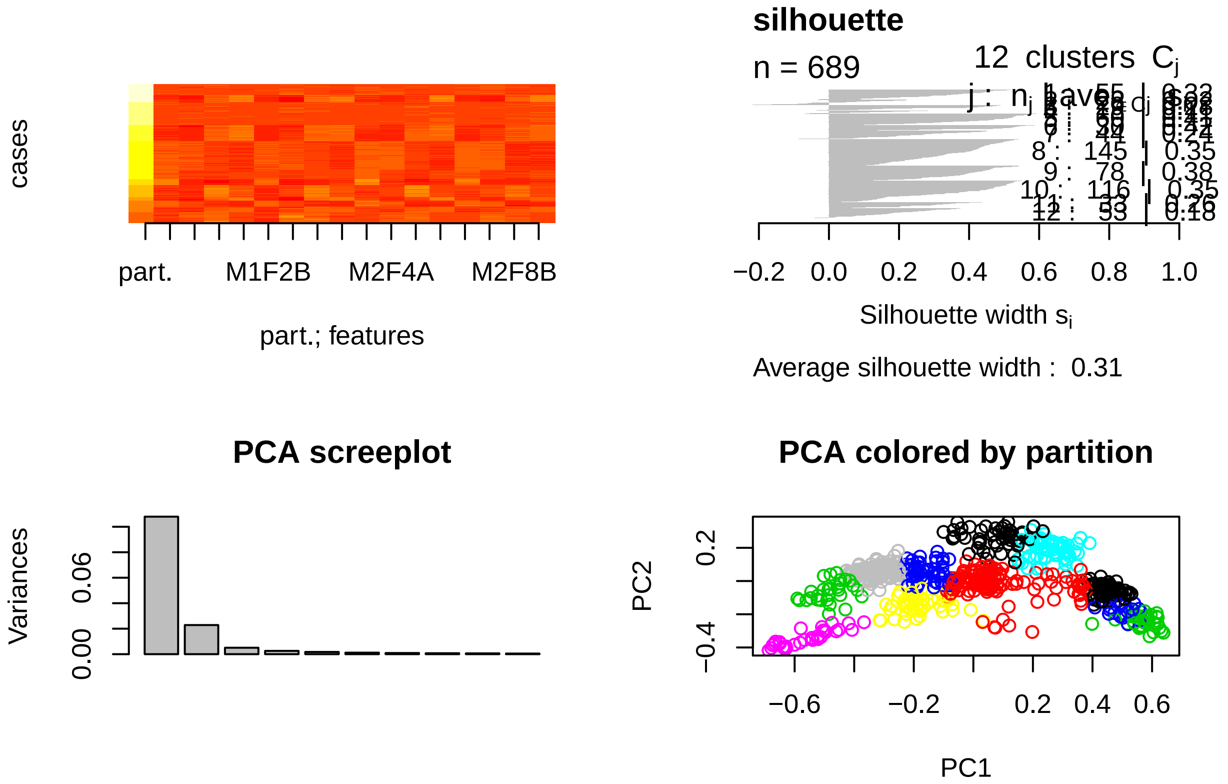
15.2 Clustering

15.2.1 kmeans

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

```
## clusteringOutput: partition table
##
##   1   2   3   4   5   6   7   8   9  10  11  12
##  55  28  29  18  60  30  44 145  78 116  33  53
## The call that created this object was:
## MLearn(formula = ~., data = dunkley2006, .method = kmeansI, centers = 12)
```

```
plot(kcl, exprs(dunkley2006))
```



A wide range of classification and clustering 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.

16 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.

17 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.

18 Session information

```
## R version 3.2.0 Patched (2015-04-22 r68234)
## Platform: x86_64-unknown-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.2 LTS
##
## attached base packages:
## [1] stats4    parallel  methods   stats     graphics  grDevices utils
## [8] datasets  base
##
## other attached packages:
##  [1] lattice_0.20-31       hpar_1.11.0           rols_1.11.0
##  [4] MSGFplus_1.3.0        pRolocdata_1.7.1      pRoloc_1.9.3
##  [7] MLInterfaces_1.49.0   cluster_2.0.1         annotate_1.47.0
## [10] XML_3.98-1.2          AnnotationDbi_1.31.16 IRanges_2.3.11
## [13] S4Vectors_0.7.4       rpx_1.5.0             MSnID_1.3.1
## [16] mzID_1.7.0            nloptr_1.0.4          RforProteomics_1.7.1
## [19] MSnbase_1.17.5        ProtGenerics_1.1.0    BiocParallel_1.3.25
## [22] mzR_2.3.1             Rcpp_0.11.6           Biobase_2.29.1
## [25] BiocGenerics_0.15.2   BiocInstaller_1.19.6  knitr_1.10.5
## [28] BiocStyle_1.7.3
##
## loaded via a namespace (and not attached):
##  [1] minqa_1.2.4                  colorspace_1.2-6
##  [3] class_7.3-12                 mclust_5.0.1
##  [5] futile.logger_1.4.1          pls_2.4-3
##  [7] proxy_0.4-14                 affyio_1.37.0
##  [9] interactiveDisplayBase_1.7.0 mvtnorm_1.0-2
## [11] codetools_0.2-11             splines_3.2.0
## [13] R.methodsS3_1.7.0            doParallel_1.0.8
## [15] impute_1.43.0                brglm_0.5-9
## [17] BradleyTerry2_1.0-6          caret_6.0-47
## [19] pbkrtest_0.4-2               rda_1.0.2-2
## [21] kernlab_0.9-20              vsn_3.37.1
## [23] R.oo_1.19.0                 sfsmisc_1.0-27
## [25] graph_1.47.2                interactiveDisplay_1.7.1
## [27] shiny_0.12.0.9002           sampling_2.6
## [29] Matrix_1.2-1                limma_3.25.9
## [31] formatR_1.2                 htmltools_0.2.6
## [33] quantreg_5.11               tools_3.2.0
## [35] gtable_0.1.2                affy_1.47.1
## [37] Category_2.35.1             reshape2_1.4.1
## [39] MALDIquant_1.12             RJSONIO_1.3-0
## [41] gdata_2.16.1                preprocessCore_1.31.0
## [43] nlme_3.1-120                iterators_1.0.7
## [45] stringr_1.0.0               proto_0.3-10
## [47] lme4_1.1-7                  mime_0.3
## [49] lpSolve_5.6.11              gtools_3.5.0
## [51] zlibbioc_1.15.0             MASS_7.3-40
## [53] scales_0.2.4                pcaMethods_1.59.0
## [55] RBGL_1.45.1                 SparseM_1.6
## [57] lambda.r_1.1.7              RColorBrewer_1.1-2
## [59] yaml_2.1.13                 ggplot2_1.0.1
## [61] biomaRt_2.25.1              rpart_4.1-9
## [63] stringi_0.4-1               RSQLite_1.0.0
## [65] highr_0.5                   genefilter_1.51.0
## [67] gridSVG_1.4-3               foreach_1.4.2
## [69] randomForest_4.6-10         e1071_1.6-4
## [71] chron_2.3-45                bitops_1.0-6
## [73] evaluate_0.7                labeling_0.3
## [75] GSEABase_1.31.3             plyr_1.8.2
## [77] magrittr_1.5                R6_2.0.1
## [79] RUnit_0.4.28                DBI_0.3.1
## [81] mgcv_1.8-6                  biocViews_1.37.7
## [83] survival_2.38-1             RCurl_1.95-4.6
## [85] nnet_7.3-9                  car_2.0-25
## [87] futile.options_1.0.0        rmarkdown_0.6.1
## [89] SSOAP_0.8-0                 grid_3.2.0
## [91] XMLSchema_0.7-2             data.table_1.9.4
## [93] FNN_1.1                     digest_0.6.8
## [95] xtable_1.7-4                R.cache_0.10.0
## [97] httpuv_1.3.2                R.utils_2.1.0
## [99] munsell_0.4.2
```