

MSnbase: labelled and label-free MS2 data pre-processing, visualisation and quantification.

Laurent Gatto* and Sebastian Gibb

January 4, 2017

Abstract

This vignette describes the functionality implemented in the *MSnbase* package. *MSnbase* aims at (1) facilitating the import, processing, visualisation and quantification of mass spectrometry data into the *R* environment [1] by providing specific data classes and methods and (2) enabling the utilisation of throughput-high data analysis pipelines provided by the Bioconductor [2] project.

Keywords: Mass Spectrometry (MS), proteomics, infrastructure, quantitative.

*lg390@cam.ac.uk

Contents

1	Introduction	5
2	Data structure and content	5
2.1	Importing experiments	5
2.2	MS experiments	6
2.3	Spectra objects	8
2.4	Reporter ions	9
3	Plotting raw data	10
3.1	MS data space	10
3.2	MS Spectra	11
4	Tandem MS identification data	16
4.1	Adding identification data	16
4.2	Filtering identification data	17
4.3	Calculate Fragments	18
5	Quality control	21
6	Raw data processing	24
6.1	Cleaning spectra	24
6.2	Focusing on specific MZ values	25
6.3	Spectrum processing	26
7	MS² isobaric tagging quantitation	28
7.1	Reporter ions quantitation	28
7.2	Importing quantitation data	30
7.3	Peak adjustments	30
8	Processing quantitative data	34
8.1	Data imputation	34
8.2	Normalisation	37
9	Feature aggregation	41
10	Label-free MS² quantitation	43
10.1	Peptide counting	43
10.2	Spectral counting and intensity methods	43
11	Spectra comparison	45
11.1	Plotting two spectra	45
11.2	Comparison metrics	45
12	Quantitative assessment of incomplete dissociation	48

<i>MSnbase</i> introduction	3
13 Combining MSnSet instances	50
13.1 Combining identical samples	50
13.2 Combine different samples	51
13.3 Splitting and unsplitting <i>MSnSet</i> instances	54
13.4 Averaging <i>MSnSet</i> instances	55
14 MS^E data processing	58
15 Session information	58

Foreword

MSnbase is under active development; current functionality is evolving and new features will be added. This software is free and open-source software. If you use it, please support the project by citing it in publications:

Laurent Gatto and Kathryn S. Lilley. *MSnbase - an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation*. *Bioinformatics* 28, 288-289 (2011).

Questions and bugs

You are welcome to contact me directly about *MSnbase*. For bugs, typos, suggestions or other questions, please file an issue in our tracking system (<https://github.com/lgatto/MSnbase/issues>) providing as much information as possible, a reproducible example and the output of `sessionInfo()`.

If you wish to reach a broader audience for general questions about proteomics analysis using R, you may want to use the Bioconductor support site: <https://support.bioconductor.org/>.

1 Introduction

MSnbase [3] aims at providing a reproducible research framework to proteomics data analysis. It should allow researcher to easily mine mass spectrometry data, explore the data and its statistical properties and visually display these.

MSnbase also aims at being compatible with the infrastructure implemented in Bioconductor, in particular *Biobase*. As such, classes developed specifically for proteomics mass spectrometry data are based on the *eSet* and *ExpressionSet* classes. The main goal is to assure seamless compatibility with existing meta data structure, accessor methods and normalisation techniques.

This vignette illustrates *MSnbase* utility using a dummy data sets provided with the package without describing the underlying data structures. More details can be found in the package, classes, method and function documentations. A description of the classes is provided in the *MSnbase-development* vignette.

Speed and memory requirements Raw mass spectrometry file are generally several hundreds of MB large and most of this is used for binary raw spectrum data. As such, data containers can easily grow very large and thus require large amounts of RAM. This requirement is being tackled by avoiding to load the raw data into memory and using on-disk random access to the content of *mzXML*/*mzML* data files on demand. When focusing on reporter ion quantitation, a direct solution for this is to trim the spectra using the `trimMz` method to select the area of interest and thus substantially reduce the size of the *Spectrum* objects. This is illustrated in section 6.2 on page 25 of the *MSnbase-demo* vignette.

The independent handling of spectra is ideally suited for parallel processing. The `quantify` method now performs reporter peaks quantitation in parallel. More functions are being updated.

2 Data structure and content

2.1 Importing experiments

MSnbase is able to import raw MS data stored in one of the XML-based formats as well as peak lists in the *mfg* format¹

Raw data The XML-based formats, *mzXML* [4], *mzData* [5] and *mzML* [6] can be imported with the `readMSData` function, as illustrated below (see `?readMSData` for more details).

```
file <- dir(system.file(package = "MSnbase", dir = "extdata"),
           full.names = TRUE, pattern = "mzXML$")
rawdata <- readMSData(file, msLevel = 2, verbose = FALSE)
```

¹Mascot Generic Format – http://www.matrixscience.com/help/data_file_help.html#GEN

Only spectra of a give MS level can be loaded at a time by setting the `msLevel` parameter accordingly. In this document, we will use the `itraqdata` data set, provided with *MSnbase*. It includes feature metadata, accessible with the `fData` accessor. The metadata includes identification data for the 55 MS2 spectra.

MSnbase 2.0 The new major version of *MSnbase* uses a new *on-disk* data storage model (see the *benchmarking* vignette for more details). The new data backend is compatible with the original *in-memory* model. To make use of the new infrastructure, read your raw data using `readMSData2`, rather than `readMSData`. All existing operations work irrespective of the backend.

Peak lists Peak lists can often be exported after spectrum processing from vendor-specific software and are also used as input to search engines. Peak lists in `mgf` format can be imported with the function `readMgfData` (see `?readMgfData` for details) to create experiment objects. Experiments or individual spectra can be exported to an `mgf` file with the `writeMgfData` methods (see `?writeMgfData` for details and examples).

Experiments with multiple runs Although it is possible to load and process multiple files serially and later merge the resulting quantitation data as show in section 13 (page 50), it is also feasible to load several raw data files at once. Here, we report the analysis of an LC-MSMS experiment were 14 liquid chromatography (LC) fractions were loaded using `readMSData` on a 32-cores servers with 128 Gb of RAM. It took about 90 minutes to read the 14 uncentroided `mzXML` raw files (4.9 Gb on disk in total) and create a 3.3 Gb raw data object (an *MSnExp* instance, see next section). Quantitation of 9 reporter ions (*iTRAQ9* object, see 2.4) for 88690 features was performed in parallel on 16 processors and took 76 minutes. The resulting quantitation data was only 22.1 Mb and could easily be further processed and analysed on a standard laptop computer.

Since verions 1.13.5, parallel support is provided by the *BiocParallel* and various backends including multicore (forking), simple network network of workstations (SNOW) using sockets, forking or MPI among others.

See also section 7.2 to import quantitative data stored in spreadsheets into *R* for further processing using *MSnbase*. The `MSnbase-io` vignette gives a general overview of *MSnbase*'s input/output capabilities.

2.2 MS experiments

Raw data is contained in *MSnExp* objects, that stores all the spectra of an experiment, as defined by one or multiple raw data files.

```
library("MSnbase")
itraqdata
```

```
## Object of class "MSnExp" (in memory)
## Object size in memory: 1.88 Mb
## - - - Spectra data - - -
## MS level(s): 2
## Number of spectra: 55
## MSn retention times: 19:9 - 50:18 minutes
## - - - Processing information - - -
## Data loaded: Wed May 11 18:54:39 2011
## Updated from version 0.3.0 to 0.3.1 [Fri Jul 8 20:23:25 2016]
## MSnbase version: 1.1.22
## - - - Meta data - - -
## phenoData
##   rowNames: 1
##   varLabels: sampleNames sampleNumbers
##   varMetadata: labelDescription
## Loaded from:
##   dummyiTRAQ.mzXML
## protocolData: none
## featureData
##   featureNames: X1 X10 ... X9 (55 total)
##   fvarLabels: spectrum ProteinAccession ProteinDescription PeptideSequence
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
```

```
head(fData(itraqdata))
```

##	spectrum	ProteinAccession	ProteinDescription	PeptideSequence
## X1	1	BSA	bovine serum albumin	NYQEAK
## X10	10	ECA1422	glucose-1-phosphate cytidylyltransferase	VTLVDTGEHSMTGGR
## X11	11	ECA4030	50S ribosomal subunit protein L4	SPIWR
## X12	12	ECA3882	chaperone protein DnaK	TAIDDALK
## X13	13	ECA1364	succinyl-CoA synthetase alpha chain	SILINK
## X14	14	ECA0871	NADP-dependent malic enzyme	DFEVVNESDPR

As illustrated above, showing the experiment textually displays its content:

- Information about the raw data, i.e. the spectra.
- Specific information about the experiment processing² and package version. This slot can be accessed with the `processingData` method.
- Other meta data, including experimental phenotype, file name(s) used to import the data, protocol data, information about features (individual spectra here) and experiment data. Most of these are implemented as in the `eSet` class and are described in more details in their respective manual pages. See `?MSnExp` and references therein for additional background information.
The experiment meta data associated with an *MSnExp* experiment is of class *MIAPE*. It stores

²this part will be automatically updated when the object is modified with its *ad hoc* methods, as illustrated later

general information about the experiment as well as MIAPE (Minimum Information About a Proteomics Experiment) information [7, 8]. This meta-data can be accessed with the `experimentData` method. When available, a summary of MIAPE-MS data can be printed with the `msInfo` method. See `?MIAPE` for more details.

2.3 Spectra objects

The raw data is composed of the 55 MS spectra. The spectra are named individually (X1, X10, X11, X12, X13, X14, ...) and stored in an environment. They can be accessed individually with `itraqdata[["X1"]]` or `itraqdata[[1]]`, or as a list with `spectra(itraqdata)`. As we have loaded our experiment specifying `msLevel=2`, the spectra will all be of level 2 (or higher, if available).

```
sp <- itraqdata[["X1"]]
sp
## Object of class "Spectrum2"
## Precursor: 520.7833
## Retention time: 19:9
## Charge: 2
## MSn level: 2
## Peaks count: 1922
## Total ion count: 26413754
```

Attributes of individual spectra or of all spectra of an experiment can be accessed with their respective methods: `precursorCharge` for the precursor charge, `rtime` for the retention time, `mz` for the MZ values, `intensity` for the intensities, ... see the *Spectrum*, *Spectrum1* and *Spectrum2* manuals for more details.

```
peaksCount(sp)
## [1] 1922

head(peaksCount(itraqdata))
##   X1  X10  X11  X12  X13  X14
## 1922 1376 1571 2397 2574 1829

rtime(sp)
## [1] 1149.31

head(rtime(itraqdata))
##      X1      X10      X11      X12      X13      X14
## 1149.31 1503.03 1663.61 1663.86 1664.08 1664.32
```


2.4 Reporter ions

Reporter ions are defined with the *ReporterIons* class. Specific peaks of interest are defined by a MZ value, a width around the expected MZ and a name (and optionally a colour for plotting, see section 3). *ReporterIons* instances are required to quantify reporter peaks in *MSnExp* experiments. Instances for the most commonly used isobaric tags like iTRAQ 4-plex and 8-plex and TMT 6- and 10-plex tags are already defined in *MSnbase*. See ?ReporterIons for details about how to generate new *ReporterIons* objects.

```
iTRAQ4

## Object of class "ReporterIons"
## iTRAQ4: '4-plex iTRAQ' with 4 reporter ions
## - 114.1112 +/- 0.05 (red)
## - 115.1083 +/- 0.05 (green)
## - 116.1116 +/- 0.05 (blue)
## - 117.115 +/- 0.05 (yellow)

TMT10

## Object of class "ReporterIons"
## TMT10HCD: '10-plex TMT HCD' with 10 reporter ions
## - 126.1277 +/- 0.002 (#8DD3C7)
## - 127.1248 +/- 0.002 (#####B3)
## - 127.1311 +/- 0.002 (#####ADA)
## - 128.1281 +/- 0.002 (#####072)
## - 128.1344 +/- 0.002 (#####1D3)
## - 129.1315 +/- 0.002 (#####462)
## - 129.1378 +/- 0.002 (#####3DE69)
## - 130.1348 +/- 0.002 (#####CCDE5)
## - 130.1411 +/- 0.002 (#####D9D9D9)
## - 131.1382 +/- 0.002 (#####C80BD)
```

3 Plotting raw data

3.1 MS data space

The *MSmap* class can be used to isolate specific slices of interest from a complete MS acquisition by specifying m/z and retention time ranges. One needs a raw data file in a format supported by *mzR*'s `openMSfile` (`mzML`, `mzXML`, ...). Below we first download a raw data file from the PRIDE repository and create³ an *MSmap* containing all the MS¹ spectra between acquired between 30 and 35 minutes and peaks between 521 and 523 m/z . See `?MSmap` for details.

```
## downloads the data
library("rpx")
px1 <- PXDataset("PXD000001")
mzf <- pxget(px1, 6)

## reads the data
ms <- openMSfile(mzf)
hd <- header(ms)

## 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, zeroIsNA = TRUE)
```

```
M
## Object of class "MSmap"
## Map [75, 401]
## [1] Retention time: 30:1 - 34:58
## [2] M/Z: 521 - 523 (res 0.005)
```

The M map object can be rendered as a heatmap with `plot`, as shown on figure 1.

One can also render the data in 3 dimension with the `plot3D` function, as show on figure 2.

To produce figure 3, we create a second *MSmap* object containing the first two MS¹ spectra of the first map (object M above) and all intermediate MS² spectra and display m/z values between 100 and 1000.

```
i <- ms1[which(rtsel)][1]
j <- ms1[which(rtsel)][2]
```

³This code chunk is not evaluated to avoid repeated downloaded of the raw data file. The M map is provided with the package and loaded to evaluate subsequent code chunks.

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

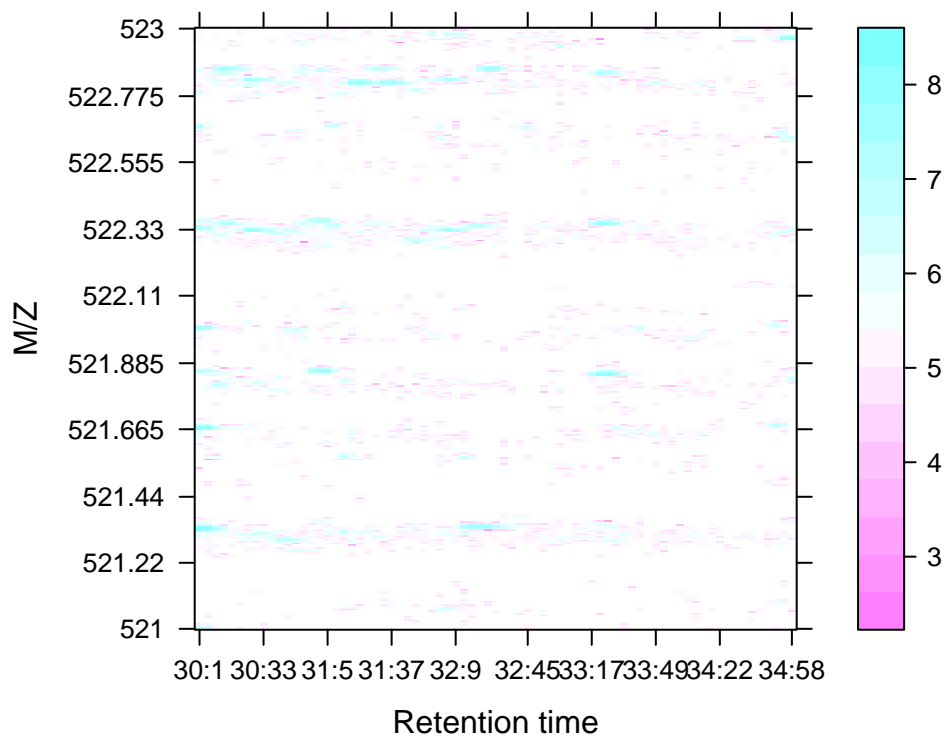


Figure 1: Heat map of a chunk of the MS data.

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

```
M2
## Object of class "MSmap"
## Map [12, 901]
## [1] Retention time: 30:1 - 30:5
## [2] M/Z: 100 - 1000 (res 1)
```

3.2 MS Spectra

Spectra can be plotted individually or as part of (subset) experiments with the `plot` method. Full spectra can be plotted (using `full=TRUE`), specific reporter ions of interest (by specifying with `reporters` with `reporters=iTRAQ4` for instance) or both (see figure 4).

```
plot3D(M)
```

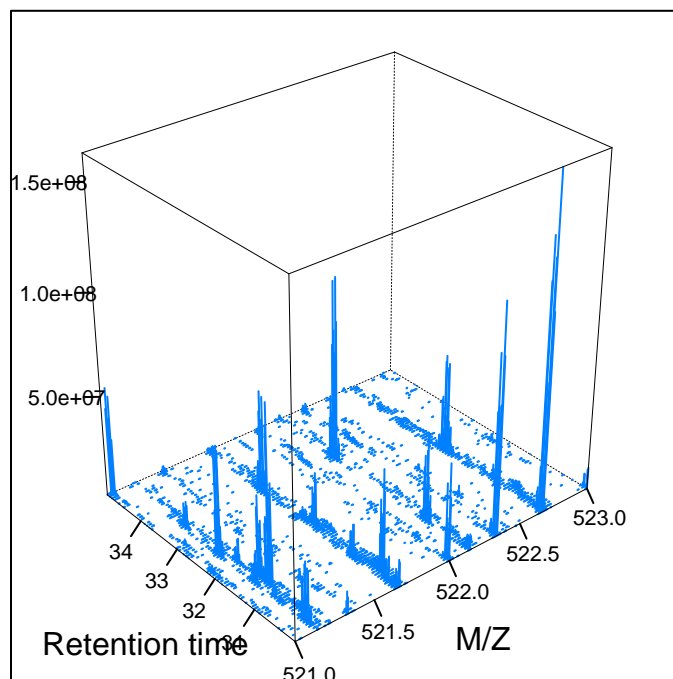


Figure 2: 3 dimensional representation of MS map data.

It is also possible to plot all spectra of an experiment (figure 5). Lets start by subsetting the `itraqdata` experiment using the protein accession numbers included in the feature metadata, and keep the 6 from the *BSA* protein.

```
sel <- fData(itraqdata)$ProteinAccession == "BSA"
bsa <- itraqdata[sel]
bsa

## Object of class "MSnExp" (in memory)
## Object size in memory: 0.1 Mb
## - - - Spectra data - - -
## MS level(s): 2
## Number of spectra: 3
## MSn retention times: 19:9 - 36:17 minutes
## - - - Processing information - - -
## Data loaded: Wed May 11 18:54:39 2011
## Updated from version 0.3.0 to 0.3.1 [Fri Jul 8 20:23:25 2016]
## Data [logically] subsetting 3 spectra: Wed Jan 4 18:52:39 2017
## MSnbase version: 1.1.22
## - - - Meta data - - -
## phenoData
```

```
plot3D(M2)
```

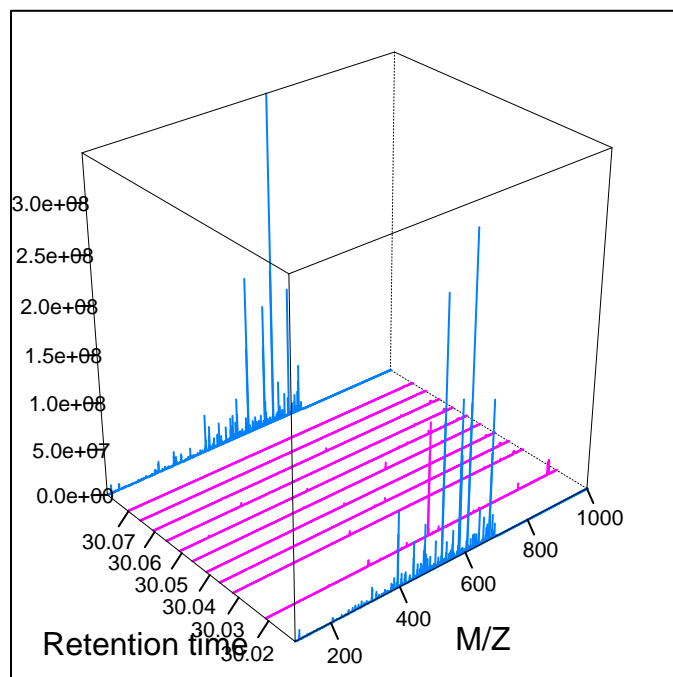


Figure 3: 3 dimensional representation of MS map data. MS¹ and MS² spectra are coloured in blue and magenta respectively.

```
## rowNames: 1
## varLabels: sampleNames sampleNumbers
## varMetadata: labelDescription
## Loaded from:
## dummyiTRAQ.mzXML
## protocolData: none
## featureData
## featureNames: X1 X52 X53
## fvarLabels: spectrum ProteinAccession ProteinDescription PeptideSequence
## fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
as.character(fData(bsa)$ProteinAccession)
## [1] "BSA" "BSA" "BSA"
```

These can then be visualised together by plotting the *MSnExp* object, as illustrated on figure 5.

Customising your plots The *MSnbase* plot methods have a logical `plot` parameter (default is `TRUE`), that specifies if the plot should be printed to the current device. A plot object is also (invisibly)

```
plot(sp, reporters = iTRAQ4, full = TRUE)
```

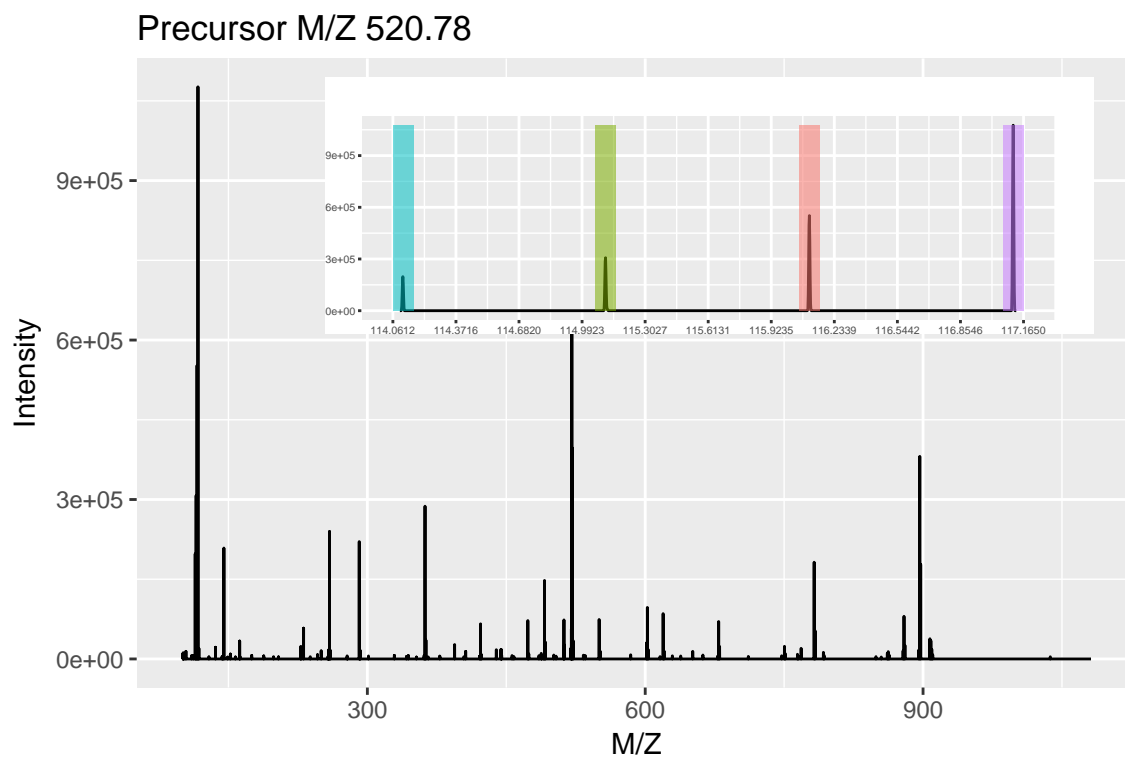


Figure 4: Raw MS2 spectrum with details about reporter ions.

returned, so that it can be saved as a variable for later use or for customisation.

MSnbase uses the *ggplot2* package to generate plots, which can subsequently easily be customised. More details about *ggplot2* can be found in [9] (especially chapter 8) and on <http://had.co.nz/ggplot2/>. Finally, if a plot object has been saved in a variable *p*, it is possible to obtain a summary of the object with `summary(p)`. To view the data frame used to generate the plot, use `p@data`.

```
plot(bsa, reporters = iTRAQ4, full = FALSE) + theme_gray(8)
```

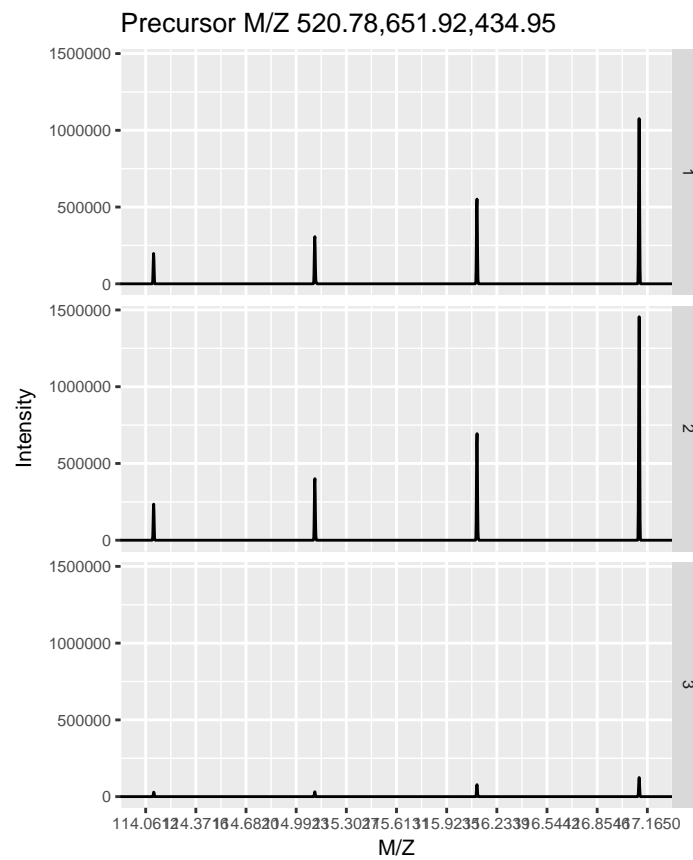


Figure 5: Experiment-wide raw MS2 spectra. The y-axes of the individual spectra are automatically rescaled to the same range. See section 8.2 to rescale peaks identically.

4 Tandem MS identification data

4.1 Adding identification data

MSnbase is able to integrate identification data from *mzIdentML* [10] files.

We first load two example files shipped with the *MSnbase* containing raw data (as above) and the corresponding identification results respectively. The raw data is read with the `readMSData`, as demonstrated above. As can be seen, the default feature data only contain spectra numbers⁴.

```
## find path to a mzXML file
quantFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "mzXML$")
## find path to a mzIdentML file
identFile <- dir(system.file(package = "MSnbase", dir = "extdata"),
                 full.name = TRUE, pattern = "dummyiTRAQ.mzid")
## create basic MSnExp
msexp <- readMSData(quantFile, verbose = FALSE)
head(fData(msexp), n = 2)

##      spectrum
## X1.1      1
## X2.1      2
```

The `addIdentificationData` method takes an *MSnExp* instance (or an *MSnSet* instance storing quantitation data, see section 7.1) as first argument and one or multiple *mzIdentML* file names (as a character vector) as second one and updates the *MSnExp* feature data using the identification data read from the *mzIdentML* file(s).

```
## add identification information
msexp <- addIdentificationData(msexp, id = identFile,
                              verbose = FALSE)
head(fData(msexp), n = 2)

##      spectrum scan number(s) passthreshold rank calculatedmasstocharge
## X1.1      1      1      TRUE      1      645.0375
## X2.1      2      2      TRUE      1      546.9633
##      experimentalmasstocharge chargestate ms-gf:denovoscore ms-gf:evaluate ms-gf:rawscore
## X1.1      645.3741      3      77      79.36958      -39
## X2.1      546.9586      3      39      13.46615      -30
##      ms-gf:specevalue assumeddissociationmethod isotopeerror isdecoy post pre end start
## X1.1      5.527468e-05      CID      1      FALSE      A      R      186      170
## X2.1      9.399048e-06      CID      0      FALSE      A      K      62      50
##      accession length
## X1.1 ECA0984;ECA3829      231
```

⁴More data about the spectra is of course available in an *MSnExp* object, as illustrated in the previous sections. See also `?pSet` and `?MSnExp` for more details.


```
## X2.1          ECA1028      275
##
##                                     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
##                                     pepseq modified modification      idFile      databaseFile
## X1.1 VESITARHGEVLQLRPK      FALSE      NA dummyiTRAQ.mzid erwinia_carotovora.fasta
## X2.1      IDGQWVTHQWLKK      FALSE      NA dummyiTRAQ.mzid erwinia_carotovora.fasta
##      nprot npep.prot npsm.prot npsm.pep
## X1.1      2          1          1          1
## X2.1      1          1          1          1
```

Finally we can use `idSummary` to summarise the percentage of identified features per quantitation/identification pairs.

```
idSummary(msexp)
##      spectrumFile      idFile coverage
## 1 dummyiTRAQ.mzXML dummyiTRAQ.mzid      0.6
```

When identification data is present, and hence peptide sequences, one can annotation fragment peaks on the MS2 figure by passing the peptide sequence to the `plot` method.

```
itraqdata2 <- pickPeaks(itraqdata, verbose=FALSE)
i <- 14
s <- as.character(fData(itraqdata2)[i, "PeptideSequence"])
```

The fragment ions are calculated with the `calculateFragments`, described in section 4.3 on page ??.

4.2 Filtering identification data

One can remove the features that have not been identified using `removeNoId`. This function uses by default the `pepseq` feature variable to search the presence of missing data (NA values) and then filter these non-identified spectra.

```
fData(msexp)$pepseq
## [1] "VESITARHGEVLQLRPK" "IDGQWVTHQWLKK"      NA      NA
## [5] "LVILLFR"

msexp <- removeNoId(msexp)
fData(msexp)$pepseq
## [1] "VESITARHGEVLQLRPK" "IDGQWVTHQWLKK"      "LVILLFR"

idSummary(msexp)
##      spectrumFile      idFile coverage
## 1 dummyiTRAQ.mzXML dummyiTRAQ.mzid      1
```

```
plot(itraqdata2[[i]], s, main = s)
```

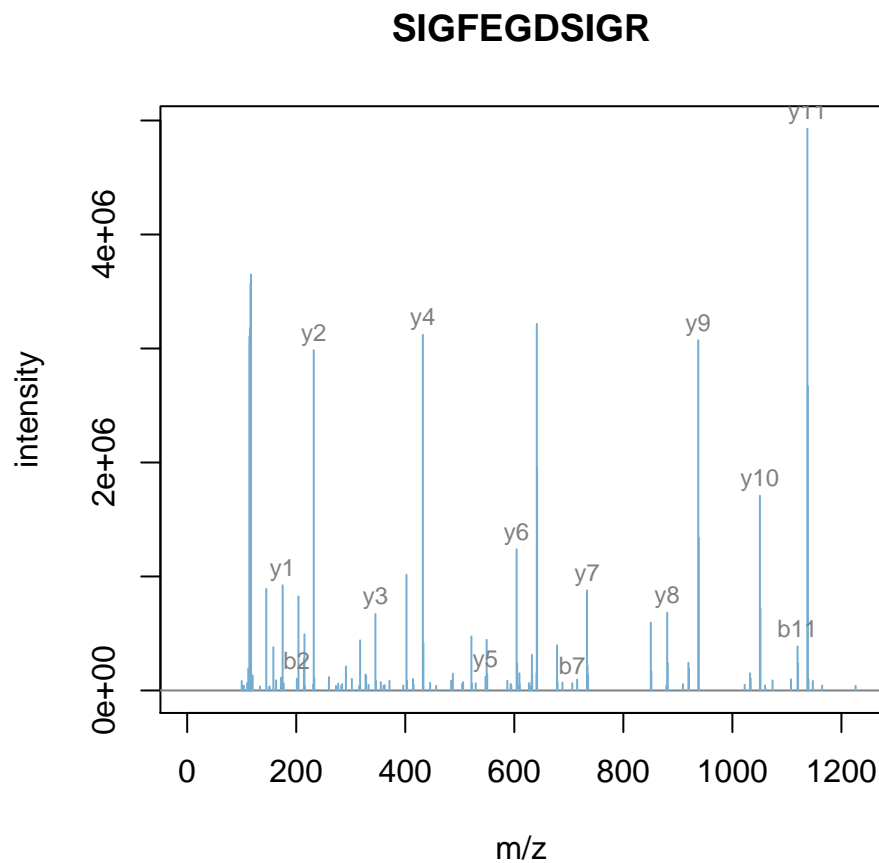


Figure 6: Annotated MS2 spectrum.

Similarly, the `removeMultipleAssignment` method can be used to filter out non-unique features, i.e. that have been assigned to protein groups with more than one member. This function uses by default the `nprot` feature variable.

Note that `removeNoId` and `removeMultipleAssignment` methods can also be called on *MSnExp* instances.

4.3 Calculate Fragments

MSnbase is able to calculate theoretical peptide fragments via `calculateFragments`.

```
calculateFragments("ACEK",
                  type = c("a", "b", "c", "x", "y", "z"))
##           mz ion type pos z seq
```

```
## 1 44.04947 a1 a 1 1 A
## 2 204.08012 a2 a 2 1 AC
## 3 333.12271 a3 a 3 1 ACE
## 4 461.21767 a4 a 4 1 ACEK
## 5 72.04439 b1 b 1 1 A
## 6 232.07504 b2 b 2 1 AC
## 7 361.11763 b3 b 3 1 ACE
## 8 489.21259 b4 b 4 1 ACEK
## 9 89.07094 c1 c 1 1 A
## 10 249.10159 c2 c 2 1 AC
## 11 378.14417 c3 c 3 1 ACE
## 12 506.23913 c4 c 4 1 ACEK
## 13 173.09207 x1 x 1 1 K
## 14 302.13466 x2 x 2 1 EK
## 15 462.16531 x3 x 3 1 CEK
## 16 533.20242 x4 x 4 1 ACEK
## 17 147.11280 y1 y 1 1 K
## 18 276.15539 y2 y 2 1 EK
## 19 436.18604 y3 y 3 1 CEK
## 20 507.22315 y4 y 4 1 ACEK
## 21 130.08625 z1 z 1 1 K
## 22 259.12884 z2 z 2 1 EK
## 23 419.15949 z3 z 3 1 CEK
## 24 490.19660 z4 z 4 1 ACEK
## 25 269.13700 x2_ x_ 2 1 EK
## 26 243.15774 y2_ y_ 2 1 EK
## 27 226.13119 z2_ z_ 2 1 EK
## 28 140.09441 x1_ x_ 1 1 K
## 29 429.16765 x3_ x_ 3 1 CEK
## 30 500.20476 x4_ x_ 4 1 ACEK
## 31 114.11515 y1_ y_ 1 1 K
## 32 403.18839 y3_ y_ 3 1 CEK
## 33 474.22550 y4_ y_ 4 1 ACEK
## 34 97.08860 z1_ z_ 1 1 K
## 35 386.16184 z3_ z_ 3 1 CEK
## 36 457.19895 z4_ z_ 4 1 ACEK
```

It is also possible to match these fragments against an *Spectrum2* object.

```
pepseq <- fData(msexp)$pepseq[1]
calculateFragments(pepseq, msexp[[1]], type=c("b", "y"))
```

##	mz	intensity	ion	type	pos	z	seq	error
## 1	100.0005	0.00	b1	b	1	1	V	0.07522824
## 2	114.1109	706555.69	y1_	y_	1	1	K	0.00425275
## 3	429.2563	1972344.00	b4	b	4	1	VESI	-0.02189010

## 4	513.3047	2574137.00	y4	y	4	1	LRPK	0.04598246
## 5	754.4504	537234.81	y6	y	6	1	LQLRPK	0.04293155
## 6	836.6139	82364.42	y7*	y*	7	1	VLQLRPK	-0.07865960
## 7	982.5354	500159.06	y8	y	8	1	EVLQLRPK	0.06897061
## 8	1080.5867	209363.69	b10	b	10	1	VESITARHGE	-0.04344392
## 9	1656.9252	0.00	b15_	b_	15	1	VESITARHGEVLQLR	0.01662010
## 10	1672.8380	76075.02	b15*	b*	15	1	VESITARHGEVLQLR	0.07488430
## 11	1688.0375	136748.83	y15*	y*	15	1	SITARHGEVLQLRPK	-0.07729359
## 12	1882.0074	149649.14	b17_	b_	17	1	VESITARHGEVLQLRPK	0.08206471

5 Quality control

The current section is not executed dynamically for package size and processing time constrains. The figures and tables have been generated with the respective methods and included statically in the vignette for illustration purposes.

MSnbase allows easy and flexible access to the data, which allows to visualise data features to assess it's quality. Some methods are readily available, although many QC approaches will be experiment specific and users are encourage to explore their data.

The `plot2d` method takes one *MSnExp* instance as first argument to produce retention time vs. precursor MZ scatter plots. Points represent individual MS2 spectra and can be coloured based on precursor charge (with second argument `z="charge"`), total ion count (`z="ionCount"`), number of peaks in the MS2 spectra (`z="peaks.count"`) or, when multiple data files were loaded, file (`z="file"`), as illustrated on figure 7. The lower right panel is produced for only a subset of proteins. See the method documentation for more details.

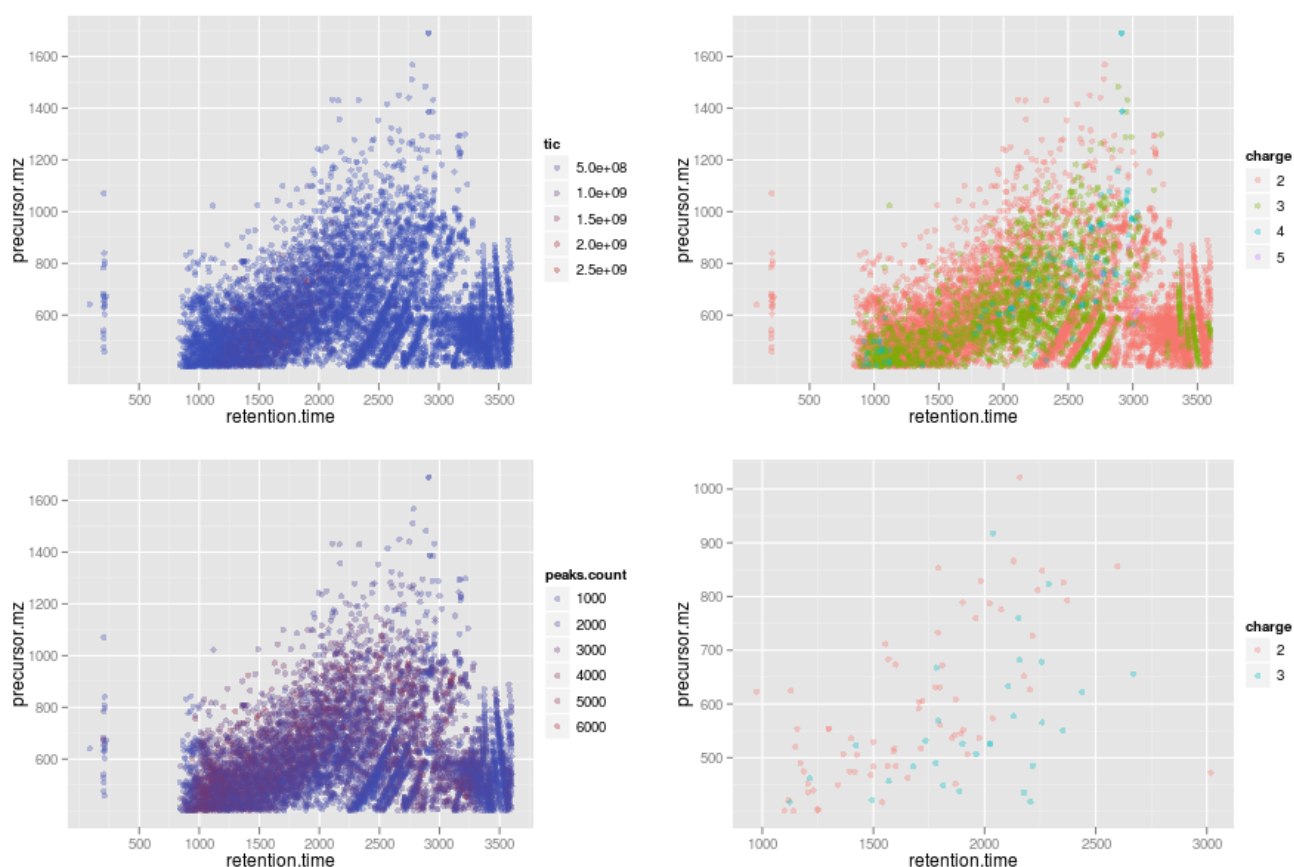


Figure 7: Illustration of the `plot2d` output.

The `plotDensity` method illustrates the distribution of several parameters of interest (see figure 8). Similarly to `plot2d`, the first argument is an *MSnExp* instance. The second is one of `precursor.mz`,

peaks.count or ionCount, whose density will be plotted. An optional third argument specifies whether the x axes should be logged.

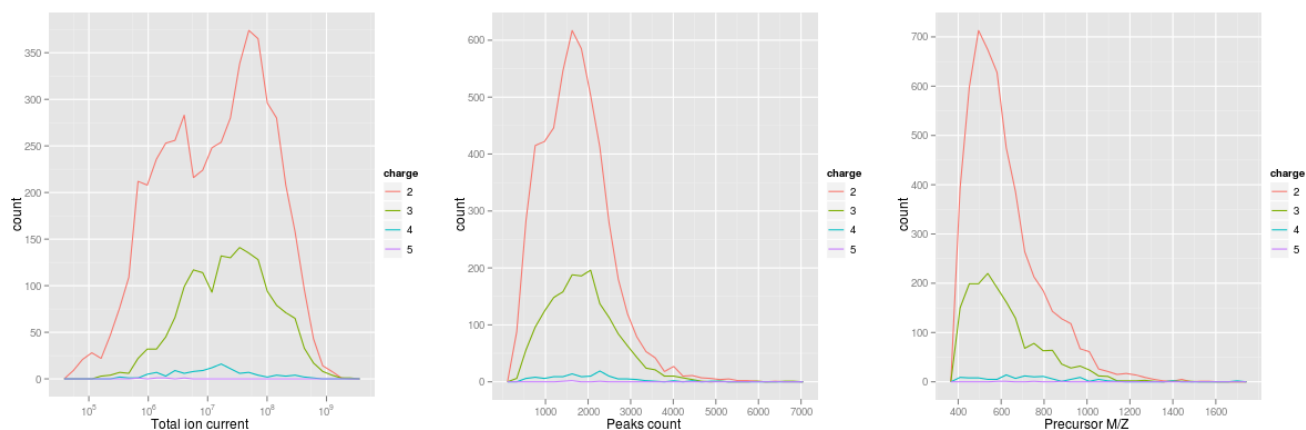


Figure 8: Illustration of the plotDensity output.

The plotMzDelta method⁵ implements the m/z delta plot from [11]. The m/z delta plot illustrates the suitability of MS2 spectra for identification by plotting the m/z differences of the most intense peaks. The resulting histogram should optimally show outstanding bars at amino acid residue masses. More details and parameters are described in the method documentation (?plotMzDelta). Figure 9 has been generated using the PRIDE experiment 12011, as in [11].

In section 12 on page 48, we illustrate how to assess incomplete reporter ion dissociation.

⁵The code to generate the histograms has been contributed by Guangchuan Yu from Jinan University, China.

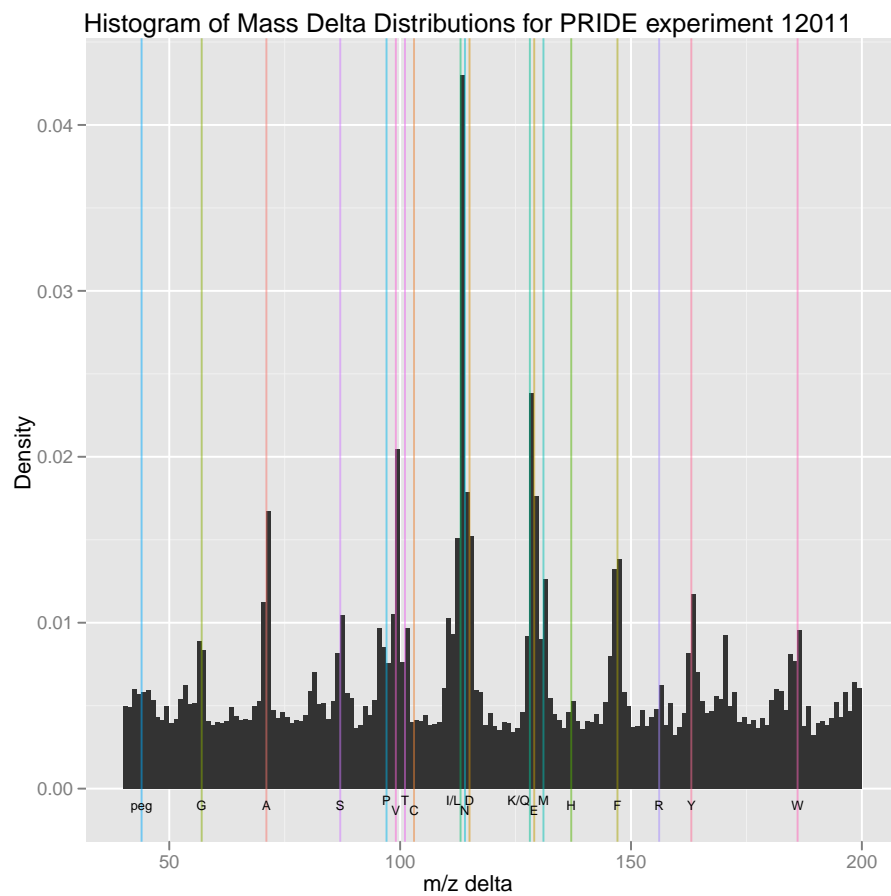


Figure 9: Illustration of the `plotMzDelta` output for the PRIDE experiment 12011, as in figure 4A from [11].

6 Raw data processing

6.1 Cleaning spectra

There are several methods implemented to perform basic raw data processing and manipulation. Low intensity peaks can be set to 0 with the `removePeaks` method from spectra or whole experiments. The intensity threshold below which peaks are removed is defined by the `t` parameter. `t` can be specified directly as a numeric. The default value is the character "min", that will remove all peaks equal to the lowest non null intensity in any spectrum. We observe the effect of the `removePeaks` method by comparing total ion count (i.e. the total intensity in a spectrum) with the `ionCount` method before (object `itraqdata`) and after (object `experiment`) for spectrum X55. The respective spectra are shown on figure 10 (page 24).

```
experiment <- removePeaks(itraqdata, t = 400, verbose = FALSE)
## total ion current
ionCount(itraqdata[["X55"]])
## [1] 555408.8
ionCount(experiment[["X55"]])
## [1] 499769.6
```

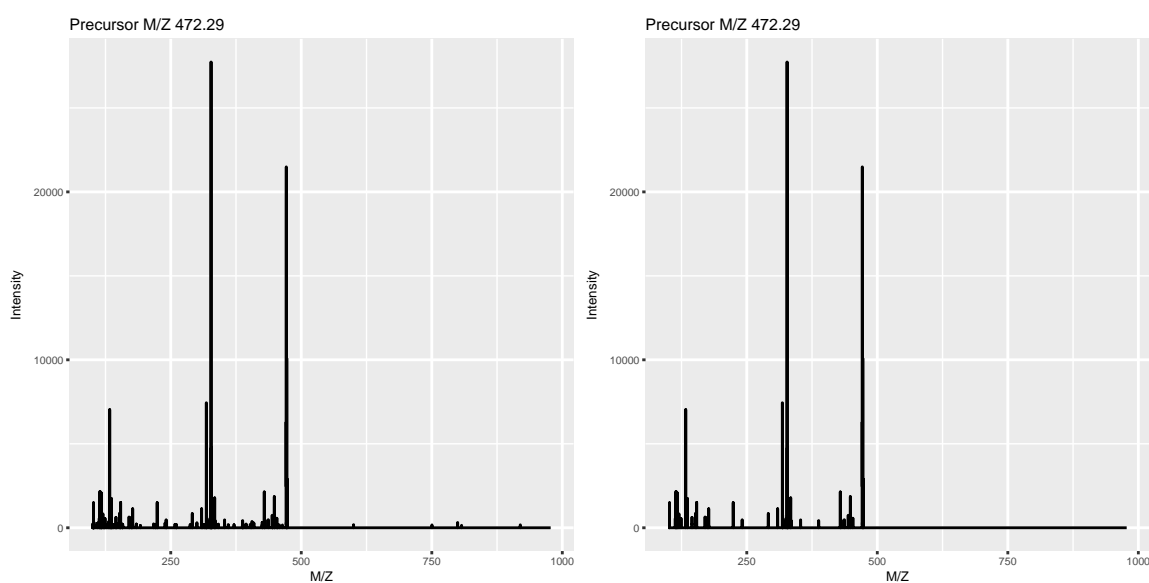


Figure 10: Same spectrum before (left) and after setting peaks $i \leq 400$ to 0.

Unlike the name might suggest, the `removePeaks` method does not actually remove peaks from the spectrum; they are set to 0. This can be checked using the `peaksCount` method, that returns the number of peaks (including 0 intensity peaks) in a spectrum. To effectively remove 0 intensity peaks from spectra, and reduce the size of the data set, one can use the `clean` method. The effect of the `removePeaks` and `clean` methods are illustrated on figure 11 on page 27.


```
## number of peaks
peaksCount(itraqdata[["X55"]])
## [1] 1726
peaksCount(experiment[["X55"]])
## [1] 1726
experiment <- clean(experiment, verbose = FALSE)
peaksCount(experiment[["X55"]])
## [1] 442
```

6.2 Focusing on specific MZ values

Another useful manipulation method is `trimMz`, that takes as parameters and *MSnExp* (or a *Spectrum*) and a numeric `mzlim`. MZ values smaller than `min(mzlim)` or greater than `max(mzlim)` are discarded. This method is particularly useful when one wants to concentrate on a specific MZ range, as for reporter ions quantification, and generally results in substantial reduction of data size. Compare the size of the full trimmed experiment to the original 1.88 Mb.

```
range(mz(itraqdata[["X55"]]))
## [1] 100.0002 977.6636
experiment <- filterMz(experiment, mzlim = c(112,120))
range(mz(experiment[["X55"]]))
## [1] 100.0002 977.6636
experiment
## Object of class "MSnExp" (in memory)
## Object size in memory: 1.17 Mb
## - - - Spectra data - - -
## MS level(s): 2
## Number of spectra: 55
## MSn retention times: 19:9 - 50:18 minutes
## - - - Processing information - - -
## Data loaded: Wed May 11 18:54:39 2011
## Updated from version 0.3.0 to 0.3.1 [Fri Jul 8 20:23:25 2016]
## Curves <= 400 set to '0': Wed Jan 4 18:52:44 2017
## Spectra cleaned: Wed Jan 4 18:52:46 2017
## MSnbase version: 1.1.22
## - - - Meta data - - -
## phenoData
## rowNames: 1
## varLabels: sampleNames sampleNumbers
```

```
## varMetadata: labelDescription
## Loaded from:
## dummyiTRAQ.mzXML
## protocolData: none
## featureData
## featureNames: X1 X10 ... X9 (55 total)
## fvarLabels: spectrum ProteinAccession ProteinDescription PeptideSequence
## fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
```

As can be seen above, all processing performed on the experiment is recorded and displayed as integral part of the experiment object.

6.3 Spectrum processing

MSnExp and *Spectrum2* instances also support standard MS data processing such as smoothing and peak picking, as described in the `smooth` and `pickPeak` manual pages. The methods that either single spectra of experiments, process the spectrum/spectra, and return a updated, processed, object. The implementations originate from the *MALDIquant* package [12].

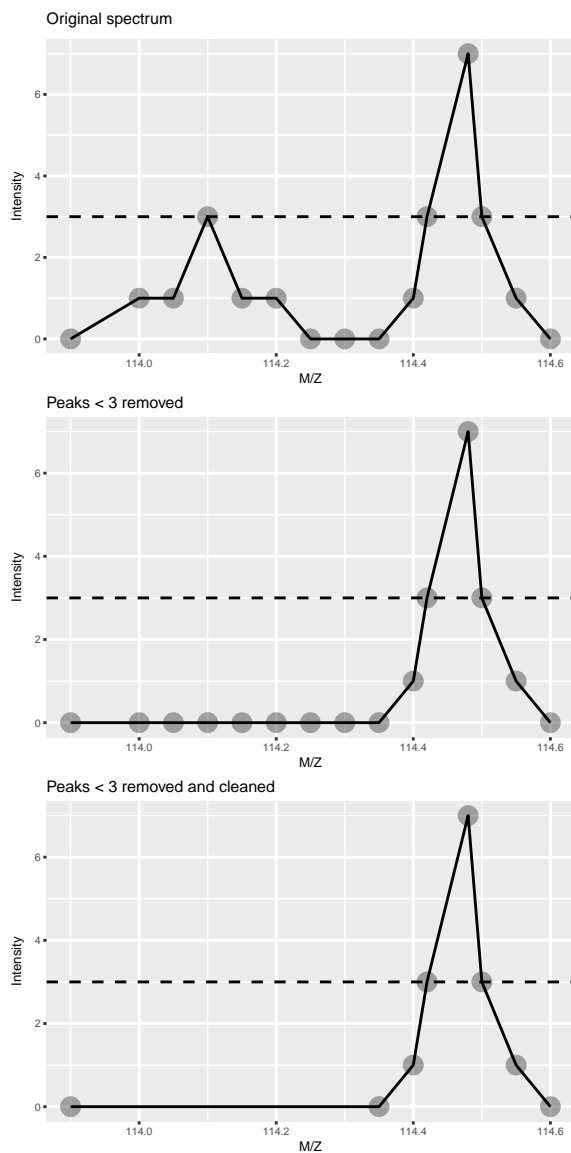


Figure 11: This figure illustrated the effect of the `removePeaks` and `clean` methods. The left-most spectrum displays two peaks, of max height 3 and 7 respectively. The middle spectrum shows the result of calling `removePeaks` with argument `t=3`, which sets all data points of the first peak, whose maximum height is smaller or equal to `t` to 0. The second peak is unaffected. Calling `clean` after `removePeaks` effectively deletes successive 0 intensities from the spectrum, as shown on the right plot.

7 MS² isobaric tagging quantitation

7.1 Reporter ions quantitation

Quantitation is performed on fixed peaks in the spectra, that are specified with an *ReporterIons* object. A specific peak is defined by its expected m/z value and is searched for within $mz \pm width$. If no data is found, NA is returned.

```
mz(iTRAQ4)
## [1] 114.1112 115.1083 116.1116 117.1150

width(iTRAQ4)
## [1] 0.05
```

The `quantify` method takes the following parameters: an *MSnExp* experiment, a character describing the quantification method, the reporters to be quantified and a `strict` logical defining whether data points ranging outside of $mz \pm width$ should be considered for quantitation. Additionally, a progress bar can be displaying when setting the `verbose` parameter to `TRUE`. Three quantification methods are implemented, as illustrated on figure 12: `trapezoidation` returns the area under the peak of interest, `max` returns the apex of the peak and `sum` returns the sum of all intensities of the peak. See `?quantify` for more details.

The `quantify` method returns *MSnSet* objects, that extend the well-known *eSet* class defined in the *Biobase* package. *MSnSet* instances are very similar to *ExpressionSet* objects, except for the experiment meta-data that captures MIAPE specific information. The assay data is a matrix of dimensions $n \times m$, where m is the number of features/spectra originally in the *MSnExp* used as parameter in `quantify` and n is the number of reporter ions, that can be accessed with the `exprs` method. The meta data is directly inherited from the *MSnExp* instance.

```
qnt <- quantify(experiment,
                method = "trap",
                reporters = iTRAQ4,
                strict = FALSE,
                verbose = FALSE)

qnt
## MSnSet (storageMode: lockedEnvironment)
## assayData: 55 features, 4 samples
## element names: exprs
## protocolData: none
## phenoData
## sampleNames: iTRAQ4.114 iTRAQ4.115 iTRAQ4.116 iTRAQ4.117
## varLabels: mz reporters
## varMetadata: labelDescription
## featureData
```

```
## featureNames: X1 X10 ... X9 (55 total)
## fvarLabels: spectrum ProteinAccession ... collision.energy (15 total)
## fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: No annotation
## - - - Processing information - - -
## Data loaded: Wed May 11 18:54:39 2011
## Updated from version 0.3.0 to 0.3.1 [Fri Jul 8 20:23:25 2016]
## Curves <= 400 set to '0': Wed Jan 4 18:52:44 2017
## Spectra cleaned: Wed Jan 4 18:52:46 2017
## iTRAQ4 quantification by trapezoidation: Wed Jan 4 18:52:53 2017
## MSnbase version: 1.1.22

head(exprs(qnt))

## iTRAQ4.114 iTRAQ4.115 iTRAQ4.116 iTRAQ4.117
## X1 1347.6158 2247.3097 3927.6931 7661.1463
## X10 739.9861 799.3501 712.5983 940.6793
## X11 27638.3582 33394.0252 32104.2879 26628.7278
## X12 31892.8928 33634.6980 37674.7272 37227.7119
## X13 26143.7542 29677.4781 29089.0593 27902.5608
## X14 6448.0829 6234.1957 6902.8903 6437.2303
```

Figure 13 illustrated the quantitation of the TMT 10-plex isobaric tags using the `quantify` method and the TMT10 reporter instance. The data on the x axis has been quantified using `method = "max"` and centroided data (as generated using ProteoWizard's `msconvert` with vendor libraries' peak picking); on the y axis, the quantitation method was trapezoidation and `strict = TRUE` (that's important for TMT 10-plex) and the profile data. We observe a very good correlation.

If no peak is detected for a reporter ion peak, the respective quantitation value is set to NA. In our case, there is 1 such case in row 41. We will remove the offending line using the `filterNA` method. The `pNA` argument defines the percentage of accepted missing values per feature. As we do not expect any missing peaks, we set it to be 0 (which is also the default value).

```
table(is.na(qnt))

##
## FALSE TRUE
## 219 1

qnt <- filterNA(qnt, pNA = 0)
sum(is.na(qnt))

## [1] 0
```

The filtering criteria for `filterNA` can also be defined as a pattern of columns that can have missing values and columns that must not exhibit any. See `?filterNA` for details and examples.

The infrastructure around the `MSnSet` class allows flexible filtering using the `[` sub-setting operator.

Below, we mimic the behaviour of `filterNA(, pNA = 0)` by calculating the row indices that should be removed, i.e. those that have at least one NA value and explicitly remove these rows. This method allows one to devise and easily apply any filtering strategy.

```
whichRow <- which(is.na((qnt))) %% nrow(qnt)
qnt <- qnt[-whichRow, ]
```

See also the `plotNA` method to obtain a graphical overview of the completeness of a data set.

7.2 Importing quantitation data

If quantitation data is already available as a spreadsheet, it can be imported, along with additional optional feature and sample (pheno) meta data, with the `readMSnSet` function. This function takes the respective text-based spreadsheet (comma- or tab-separated) file names as argument to create a valid *MSnSet* instance.

Note that the quantitation data of *MSnSet* objects can also be exported to a text-based spreadsheet file using the `write.exps` method.

MSnbase also supports the `mzTab` format⁶, a light-weight, tab-delimited file format for proteomics data. `mzTab` files can be read into *R* with `readMzTabData` to create an *MSnSet* instance.

See the `MSnbase-io` vignette for a general overview of *MSnbase*'s input/output capabilities.

7.3 Peak adjustments

Single peak adjustment In certain cases, peak intensities need to be adjusted as a result of peak interference. For example, the +1 peak of the phenylalanine (F, Phe) immonium ion (with m/z 120.03) interferes with the 121.1 TMT reporter ion. Below, we calculate the relative intensity of the +1 peaks compared to the main peak using the *Rdisop* package.

```
library(Rdisop)
## Phenylalanine immonium ion
Fim <- getMolecule("C8H10N")
getMass(Fim)

## [1] 120.0813

isotopes <- getIsotope(Fim)
F1 <- isotopes[2, 2]
F1

## [1] 0.08573496
```

⁶<https://github.com/HUPO-PSI/mzTab>

If desired, one can thus specifically quantify the F immonium ion in the MS2 spectrum, estimate the intensity of the +1 ion (0.0857% of the F peak) and subtract this calculated value from the 121.1 TMT reporter intensity.

The above principle can also be generalised for a set of overlapping peaks, as described below.

Reporter ions purity correction Impurities in the reporter reagents can also bias the results and can be corrected when manufacturers provide correction coefficients. These generally come as percentages of each reporter ion that have masses differing by -2, -1, +1 and +2 Da from the nominal reporter ion mass due to isotopic variants. The `purityCorrect` method applies such correction to `MSnSet` instances. It also requires a square matrix as second argument, `impurities`, that defines the relative percentage of reporter in the quantified each peak. See `?purityCorrect` for more details.

```
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)
qnt.crct <- purityCorrect(qnt, impurities)
head(exprs(qnt))

##      iTRAQ4.114 iTRAQ4.115 iTRAQ4.116 iTRAQ4.117
## X1    1347.6158  2247.3097  3927.6931  7661.1463
## X10    739.9861   799.3501   712.5983   940.6793
## X11  27638.3582 33394.0252 32104.2879 26628.7278
## X12  31892.8928 33634.6980 37674.7272 37227.7119
## X13  26143.7542 29677.4781 29089.0593 27902.5608
## X14   6448.0829  6234.1957  6902.8903  6437.2303

head(exprs(qnt.crct))

##      iTRAQ4.114 iTRAQ4.115 iTRAQ4.116 iTRAQ4.117
## X1    1304.7675  2168.1082  3784.2244  8133.9211
## X10    743.8159   806.5647   696.9024   988.0787
## X11  27547.6515 33592.3997 32319.1803 27413.1833
## X12  32127.1898 33408.8353 37806.0787 38658.7865
## X13  26187.3141 29788.6254 29105.2485 28936.6871
## X14   6533.1862  6184.1103  6945.2074  6666.5633
```

The `makeImpuritiesMatrix` can be used to create impurity matrices. It opens a rudimentary spreadsheet that can be directly edited.

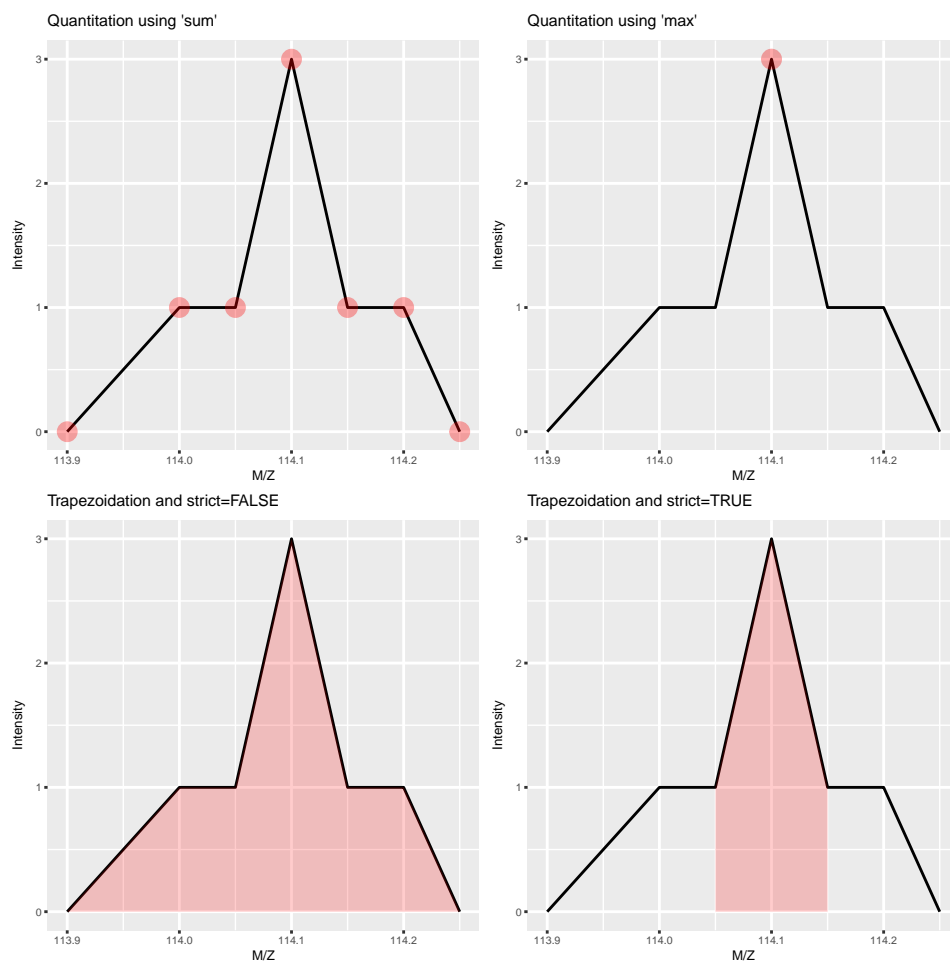


Figure 12: The different quantitation methods are illustrated above. Quantitation using `sum` sums all the data points in the peaks to produce, for this example, 7, whereas method `max` only uses the peak's maximum intensity, 3. Trapezoidation calculates the area under the peak taking the full width into account (using `strict=FALSE` gives 0.375) or only the width as defined by the reporter (using `strict=TRUE` gives 0.1).

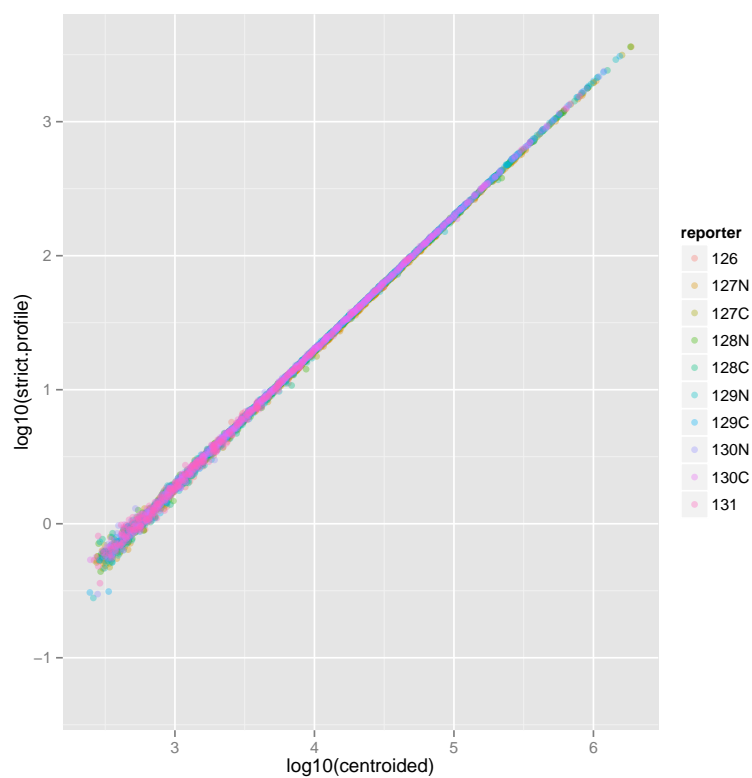


Figure 13: TMT 10-plex quantitation.

8 Processing quantitative data

8.1 Data imputation

A set of imputation methods are available in the `impute` method: it takes an *MSnSet* instance as input, the name of the imputation method to be applied (one of `bpca`, `knn`, `QRILC`, `MLE`, `MinDet`, `MinProb`, `min`, `zero`, `mixed`, `nbavg`), possible additional parameters and returns an updated for *MSnSet* without any missing values. Below, we apply a deterministic minimum value imputation on the `naset` example data:

```
## an example MSnSet containing missing values
data(naset)
table(is.na(naset))

##
## FALSE TRUE
## 10254 770

## number of NAs per protein
table(fData(naset)$nNA)

##
## 0 1 2 3 4 8 9 10
## 301 247 91 13 2 23 10 2

x <- impute(naset, "min")
processingData(x)

## - - - Processing information - - -
## Data imputation using min Wed Jan 4 18:52:53 2017
## MSnbase version: 1.15.6

table(is.na(x))

##
## FALSE
## 11024
```

There are two types of mechanisms resulting in missing values in LC/MSMS experiments.

- Missing values resulting from absence of detection of a feature, despite ions being present at detectable concentrations. For example in the case of ion suppression or as a result from the stochastic, data-dependent nature of the MS acquisition method. These missing values are expected to be randomly distributed in the data and are defined as *missing at random* (MAR) or *missing completely at random* (MCAR).
- Biologically relevant missing values, resulting from the *absence* of the low abundance of ions (below the limit of detection of the instrument). These missing values are not expected to be randomly distributed in the data and are defined as *missing not at random* (MNAR).

MAR and MCAR values can be reasonably well tackled by many imputation methods. MNAR data,

however, requires some knowledge about the underlying mechanism that generates the missing data, to be able to attempt data imputation. MNAR features should ideally be imputed with a *left-censor* (for example using a deterministic or probabilistic minimum value) method. Conversely, it is recommended to use *hot deck* methods (for example nearest neighbour, maximum likelihood, etc) when data are missing at random.

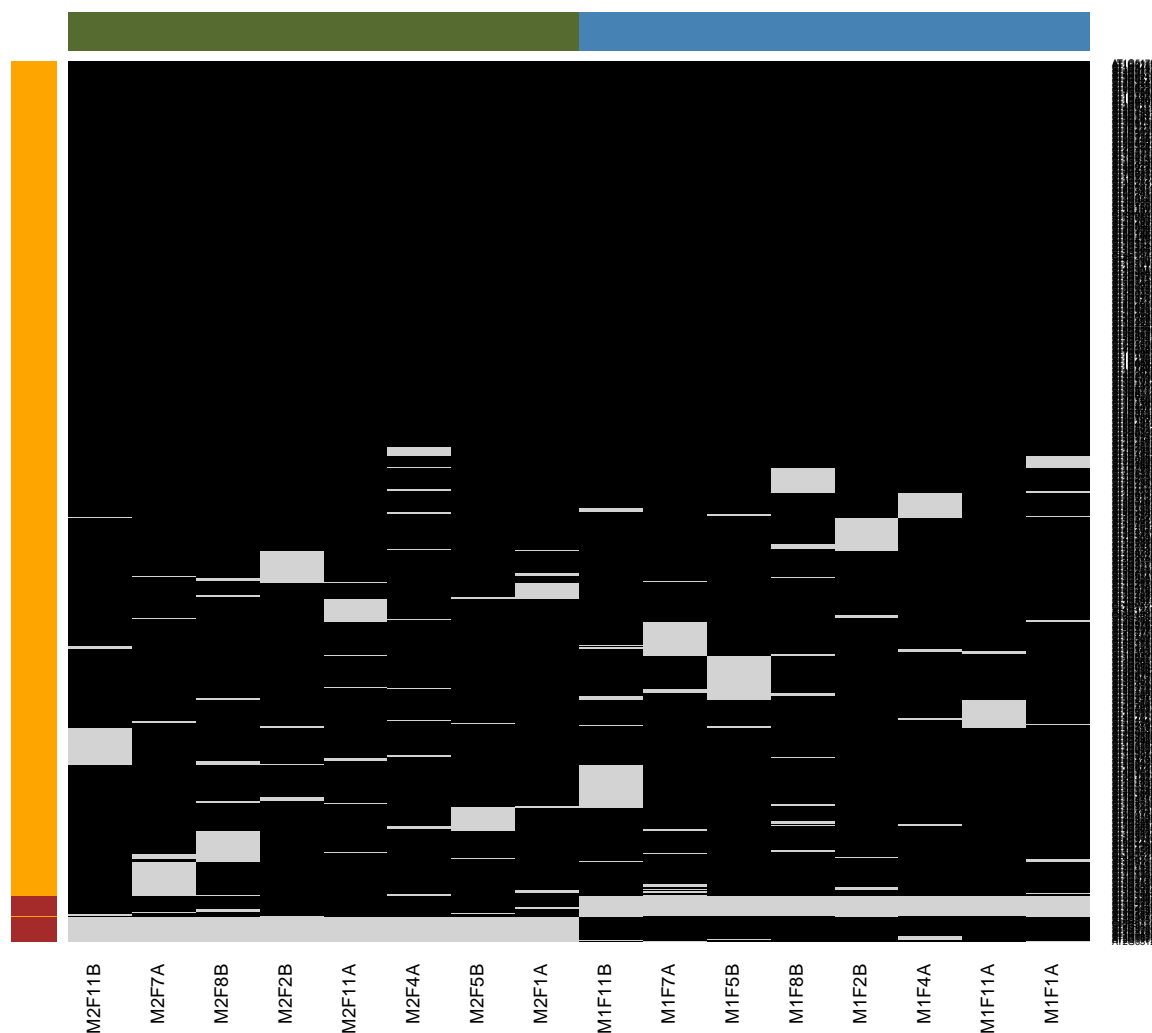


Figure 14: Mixed imputation method. Black cells represent presence of quantitation values and light grey corresponds to missing data. The two groups of interest are depicted in green and blue along the heatmap columns. Two classes of proteins are annotated on the left: yellow are proteins with randomly occurring missing values (if any) while proteins in brown are candidates for non-random missing value imputation.

It is anticipated that the identification of both classes of missing values will depend on various factors, such as feature intensities and experimental design. Below, we use perform mixed imputation, applying

nearest neighbour imputation on the 654 features that are assumed to contain randomly distributed missing values (if any) (yellow on figure 14) and a deterministic minimum value imputation on the 35 proteins that display a non-random pattern of missing values (brown on figure 14).

```
x <- impute(naset, method = "mixed",
           randna = fData(naset)$randna,
           mar = "knn", mmar = "min")
x

## MSnSet (storageMode: lockedEnvironment)
## assayData: 689 features, 16 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: M1F1A M1F4A ... M2F11B (16 total)
##   varLabels: nNA
##   varMetadata: labelDescription
## featureData
##   featureNames: AT1G09210 AT1G21750 ... AT4G39080 (689 total)
##   fvarLabels: nNA randna
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
## Data imputation using mixed Wed Jan  4 18:52:54 2017
##   Using default parameters
## MSnbase version: 1.15.6
```

Please read `?impute` for a description of the different methods.

8.2 Normalisation

A *MSnSet* object is meant to be compatible with further downstream packages for data normalisation and statistical analysis. There is also a `normalise` (also available as `normalize`) method for expression sets. The method takes an instance of class *MSnSet* as first argument, and a character to describe the method to be used:

`quantiles` Applies quantile normalisation [13] as implemented in the `normalize.quantiles` function of the *preprocessCore* package.

`quantiles.robust` Applies robust quantile normalisation [13] as implemented in the `normalize.quantiles.robust` function of the *preprocessCore* package.

`vsn` Applies variance stabilisation normalization [14] as implemented in the `vsn2` function of the *vsn* package.

`max` Each feature's reporter intensity is divided by the maximum of the reporter ions intensities.

`sum` Each feature's reporter intensity is divided by the sum of the reporter ions intensities.

See `?normalise` for more methods. A `scale` method for *MSnSet* instances, that relies on the `base::scale` function.

```
qnt.max <- normalise(qnt, "max")
qnt.sum <- normalise(qnt, "sum")
qnt.quant <- normalise(qnt, "quantiles")
qnt.qrob <- normalise(qnt, "quantiles.robust")
qnt.vsn <- normalise(qnt, "vsn")
```

The effect of these are illustrated on figure 15 and figure 16 reproduces figure 3 of [15] that described the application of `vsn` on iTRAQ reporter data.

Note that it is also possible to normalise individual spectra or whole *MSnExp* experiments with the `normalise` method using the `max` method. This will rescale all peaks between 0 and 1. To visualise the relative reporter peaks, one should first trim the spectra using method `trimMz` as illustrated in section 6, then normalise the *MSnExp* with `normalise` using `method="max"` as illustrated above and plot the data using `plot` (figure 17).

Additional dedicated normalisation methods are available for MS² label-free quantitation, as described in section 10 and in the `quantify` documentation.

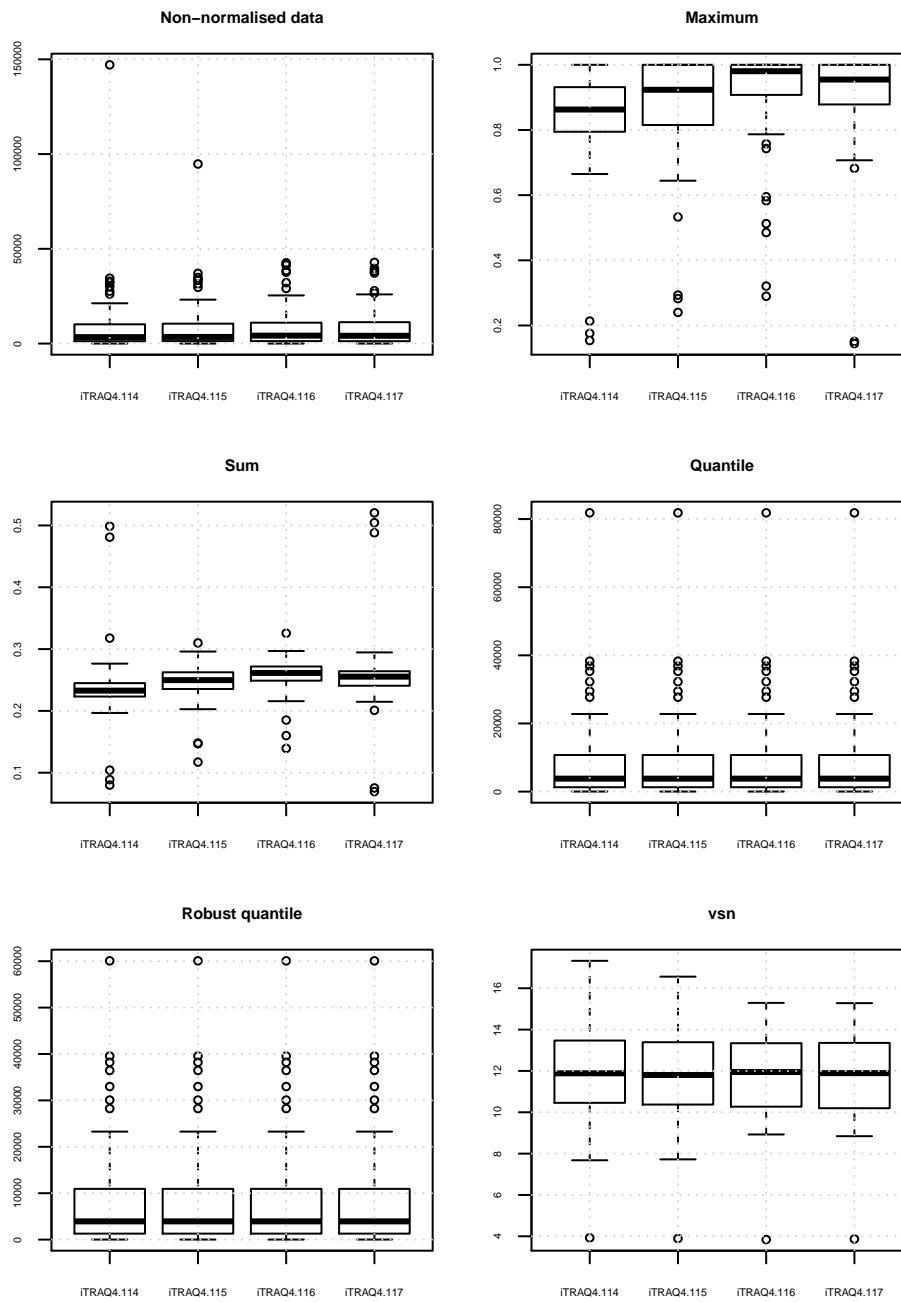


Figure 15: Comparison of the normalisation *MSnSet* methods. Note that vsn also glog-transforms the intensities.

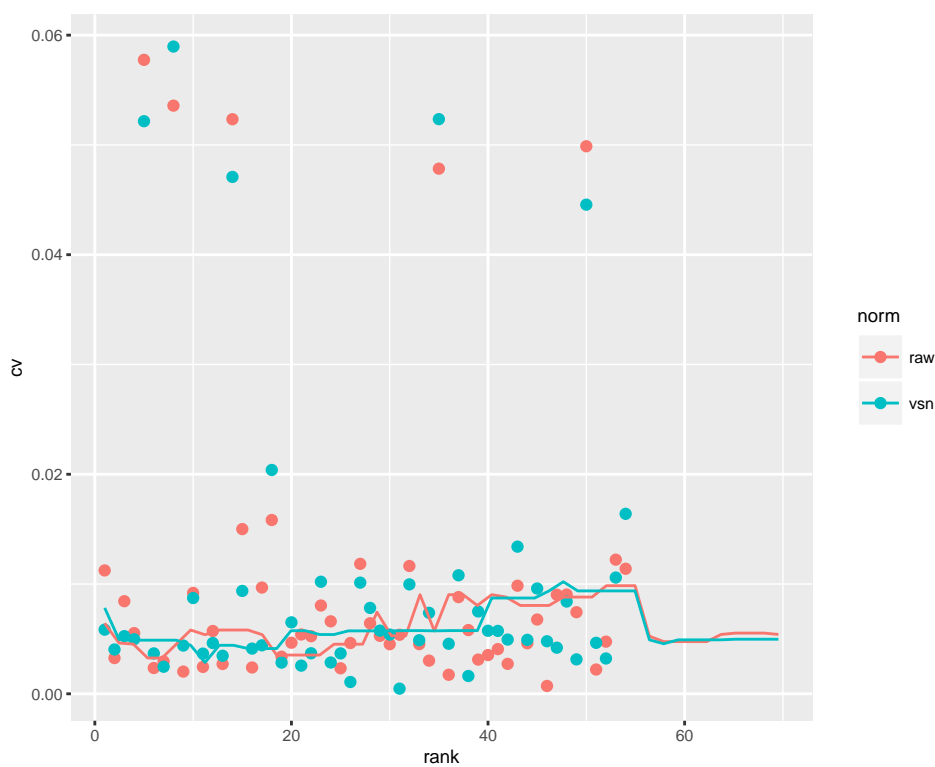


Figure 16: CV versus signal intensity comparison for log2 and vsn transformed data. Lines indicate running CV medians.

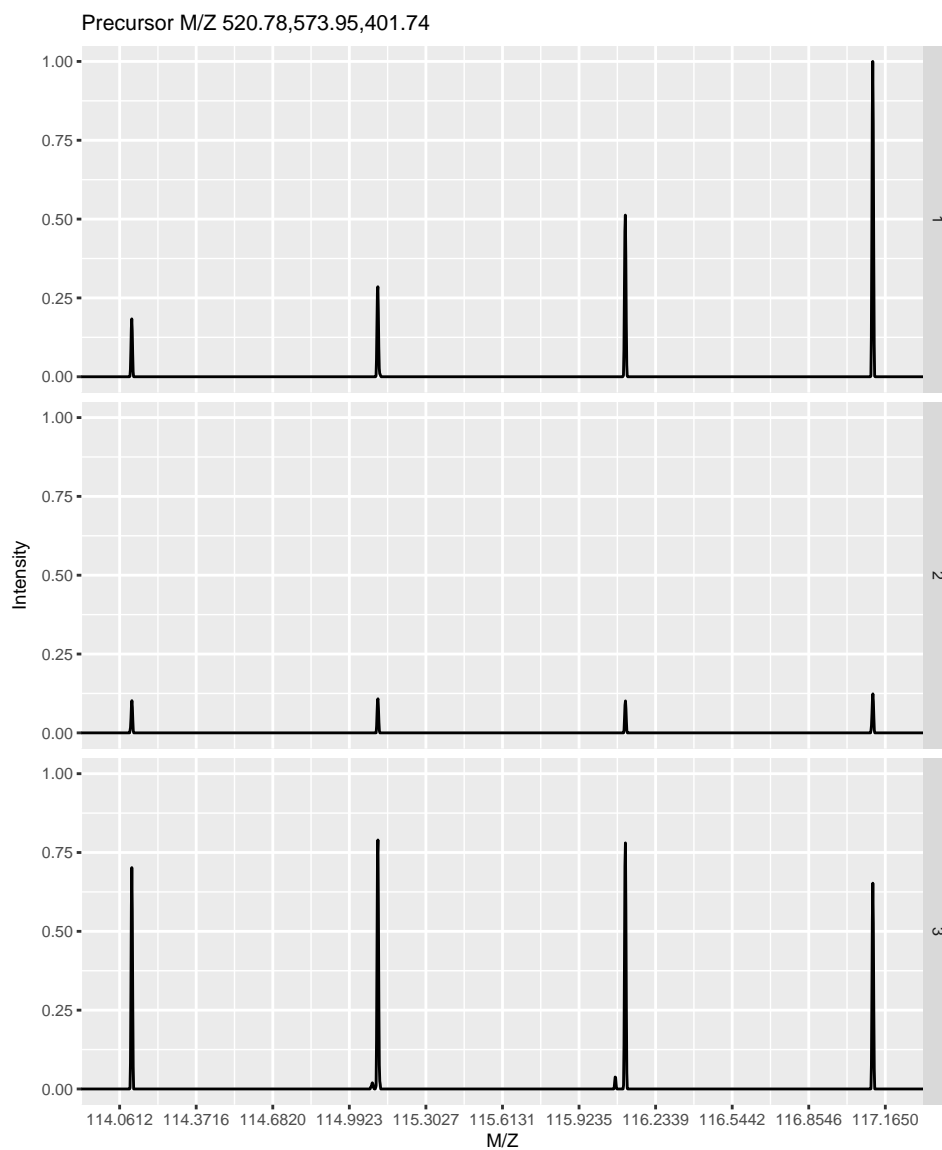


Figure 17: Experiment-wide normalised MS2 spectra. The y-axes of the individual spectra is now rescaled between 0 and 1 (highest peak), as opposed to figure 5.

9 Feature aggregation

The above quantitation and normalisation has been performed on quantitative data obtained from individual spectra. However, the biological unit of interest is not the spectrum but the peptide or the protein. As such, it is important to be able to summarise features that belong to a same group, i.e. spectra from one peptide, peptides that originate from one protein, or directly combine all spectra that have been uniquely associated to one protein.

MSnbase provides one function, `combineFeatures`, that allows to aggregate features stored in an *MSnSet* using build-in or user defined summary function and return a new *MSnSet* instance. The three main arguments are described below. Additional details can be found in the method documentation.

`combineFeatures`'s first argument, `object`, is an instance of class *MSnSet*, as has been created in the section 7.1 for instance. The second argument, `groupBy`, is a factor than has as many elements as there are features in the *MSnSet* object argument. The features corresponding to the `groupBy` levels will be aggregated so that the resulting *MSnSet* output will have `length(levels(groupBy))` features. Here, we will combine individual MS2 spectra based on the protein they originate from. As shown below, this will result in 40 new aggregated features.

```
gb <- fData(qnt)$ProteinAccession
table(gb)

## gb
##      BSA ECA0172 ECA0435 ECA0452 ECA0469 ECA0621 ECA0631 ECA0691 ECA0871 ECA0978 ECA1032
##      3      1      2      1      2      1      1      1      1      1      1
## ECA1093 ECA1104 ECA1294 ECA1362 ECA1363 ECA1364 ECA1422 ECA1443 ECA2186 ECA2391 ECA2421
##      1      1      1      1      1      1      1      1      1      1      1
## ECA2831 ECA3082 ECA3175 ECA3349 ECA3356 ECA3377 ECA3566 ECA3882 ECA3929 ECA3969 ECA4013
##      1      1      1      2      1      1      2      1      1      1      1
## ECA4026 ECA4030 ECA4037 ECA4512 ECA4513 ECA4514      ENO
##      2      1      1      1      1      6      3

length(unique(gb))

## [1] 40
```

The third argument, `fun`, defined how to combine the features. Predefined functions are readily available and can be specified as strings (`fun="mean"`, `fun="median"`, `fun="sum"`, `fun="weighted.mean"` or `fun="medianpolish"` to compute respectively the mean, media, sum, weighted mean or median polish of the features to be aggregated). Alternatively, it is possible to supply user defined functions with `fun=function(x) { ... }`. We will use the median here.

```
qnt2 <- combineFeatures(qnt, groupBy = gb, fun = "median")
qnt2

## MSnSet (storageMode: lockedEnvironment)
## assayData: 40 features, 4 samples
## element names: exprs
```

```
## protocolData: none
## phenoData: none
## featureData
##   featureNames: BSA ECA0172 ... ENO (40 total)
##   fvarLabels: spectrum ProteinAccession ... CV.iTRAQ4.117 (19 total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
## Data loaded: Wed May 11 18:54:39 2011
## Updated from version 0.3.0 to 0.3.1 [Fri Jul 8 20:23:25 2016]
## Curves <= 400 set to '0': Wed Jan 4 18:52:44 2017
## Spectra cleaned: Wed Jan 4 18:52:46 2017
## iTRAQ4 quantification by trapezoidation: Wed Jan 4 18:52:53 2017
## Subset [55,4][54,4] Wed Jan 4 18:52:53 2017
## Removed features with more than 0 NAs: Wed Jan 4 18:52:53 2017
## Dropped featureData's levels Wed Jan 4 18:52:53 2017
## Combined 54 features into 54 using median: Wed Jan 4 18:52:57 2017
## MSnbase version: 2.0.2
```

10 Label-free MS² quantitation

10.1 Peptide counting

Note that if samples are not multiplexed, label-free MS² quantitation by spectral counting is possible using *MSnbase*. Once individual spectra have been assigned to peptides and proteins (see section 4), it becomes straightforward to estimate protein quantities using the simple peptide counting method, as illustrated in section 9.

```
sc <- quantify(msexp, method = "count")
## lets modify out data for demonstration purposes
fData(sc)$accession[1] <- fData(sc)$accession[2]
fData(sc)$accession

## [1] "ECA1028" "ECA1028" "ECA0510"

sc <- combineFeatures(sc, groupBy = fData(sc)$accession,
                     fun = "sum")

exprs(sc)

##           dummyiTRAQ.mzXML
## ECA0510                   1
## ECA1028                   2
```

Such count data could then be further analysed using dedicated count methods (originally developed for high-throughput sequencing) and directly available for *MSnSet* instances in the *msmsTests* Bioconductor package.

10.2 Spectral counting and intensity methods

The spectral abundance factor (SAF) and the normalised form (NSAF) [16] as well as the spectral index (SI) and other normalised variations (SI_{GI} and SI_N) [17] are also available. Below, we illustrate how to apply the normalised SI_N to the experiment containing identification data produced in section 4.

The spectra that did not match any peptide have already been removed with the `removeNoId` method. As can be seen in the following code chunk, the first spectrum could not be matched to any single protein. Non-identified spectra and those matching multiple proteins are removed automatically prior to any label-free quantitation. One can also remove peptide that do not match uniquely to proteins (as defined by the `nprot` feature variable column) with the `removeMultipleAssignment` method.

```
fData(msexp)[, c("accession", "nprot")]

##           accession nprot
## X1.1 ECA0984;ECA3829    2
## X2.1           ECA1028    1
## X5.1           ECA0510    1
```

Note that the label-free methods implicitly apply feature aggregation (section 9) and normalise (section 8.2) the quantitation values based on the total sample intensity and or the protein lengths (see [16] and [17] for details).

Let's now proceed with the quantitation using the `quantify`, as in section 7.1, this time however specifying the method of interest, `SIn` (the `reporters` argument can of course be ignored here). The required peptide-protein mapping and protein lengths are extracted automatically from the feature meta-data using the default `accession` and `length` feature variables.

```
siquant <- quantify(msexp, method = "SIn")
processingData(siquant)

## - - - Processing information - - -
## Quantitation by total ion current [Wed Jan  4 18:52:57 2017]
## Combined 2 features into 2 using sum: Wed Jan  4 18:52:58 2017
## Quantification by SIn [Wed Jan  4 18:52:58 2017]
## MSnbase version: 2.0.2

exprs(siquant)
##           dummyiTRAQ.mzXML
## ECA0510      0.003588641
## ECA1028      0.001470129
```

Other label-free methods can be applied by specifying the appropriate `method` argument. See `?quantify` for more details.

11 Spectra comparison

11.1 Plotting two spectra

MSnbase provides functionality to compare spectra against each other. The first notable function is `plot`. If two *Spectrum2* objects are provided `plot` will draw two plots: the upper and lower panel contain respectively the first and second spectrum. Common peaks are drawn in a slightly darker colour.

11.2 Comparison metrics

Currently *MSnbase* supports three different metrics to compare spectra against each other: `common` to calculate the number of common peaks, `cor` to calculate the Pearson correlation and `dotproduct` to calculate the dot product. See `?compareSpectra` to apply other arbitrary metrics.

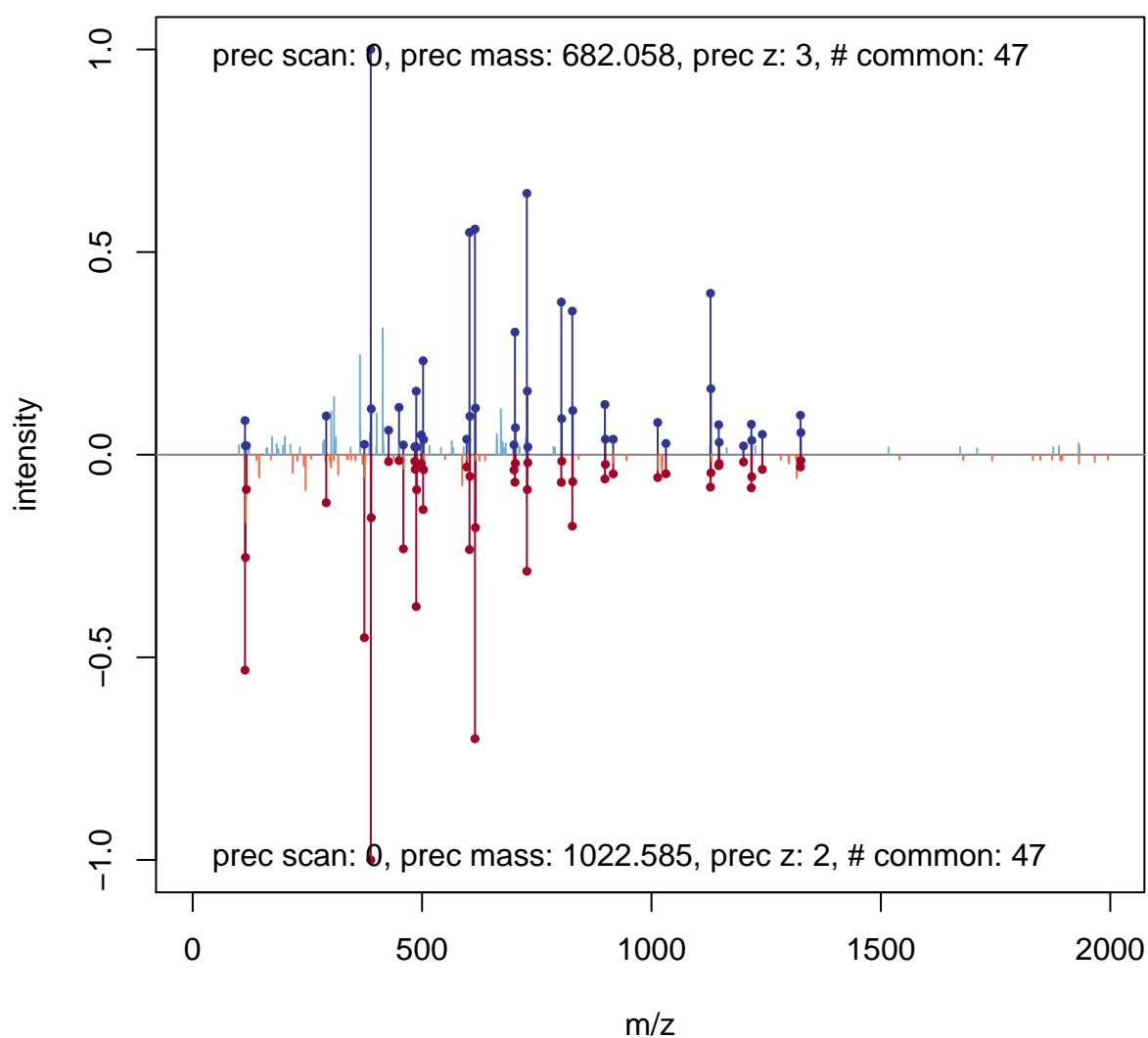
```
compareSpectra(centroided[[2]], centroided[[3]],
               fun = "common")
## [1] 8
compareSpectra(centroided[[2]], centroided[[3]],
               fun = "cor")
## [1] 0.1105021
compareSpectra(centroided[[2]], centroided[[3]],
               fun = "dotproduct")
## [1] 0.1185025
```

`compareSpectra` supports *MSnExp* objects as well.

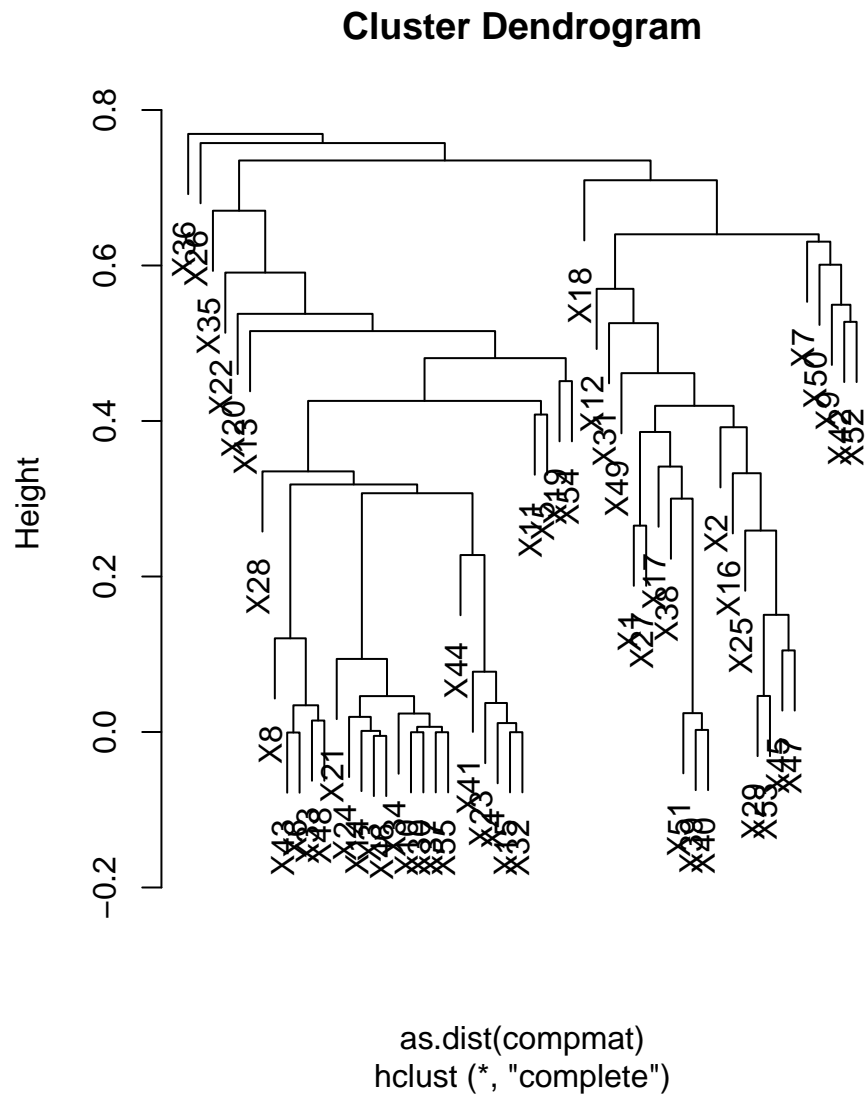
```
compmat <- compareSpectra(centroided, fun="cor")
compmat[1:10, 1:5]
##           X1          X10          X11          X12          X13
## X1          NA 0.07672973 0.38024702 0.51579989 0.46647324
## X10 0.07672973          NA 0.11050214 0.11162512 0.08611906
## X11 0.38024702 0.11050214          NA 0.47184437 0.47905818
## X12 0.51579989 0.11162512 0.47184437          NA 0.57909089
## X13 0.46647324 0.08611906 0.47905818 0.57909089          NA
## X14 0.09999703 0.01558385 0.12165400 0.12057251 0.11853321
## X15 0.03314059 0.00416184 0.01733228 0.04796236 0.03196115
## X16 0.39140514 0.06634870 0.42259036 0.45624614 0.45469020
## X17 0.37945538 0.07188420 0.52292845 0.44791250 0.43679447
## X18 0.55367861 0.10286983 0.56621755 0.66884285 0.64262061
```

Below, we illustrate how to compare a set of spectra using a hierarchical clustering.

```
centroided <- pickPeaks(itraqdata, verbose = FALSE)
(k <- which(fData(centroided)[, "PeptideSequence"] == "TAGIQIVADDLTVTNPK"))
## [1] 41 42
mzk <- precursorMz(centroided)[k]
zk <- precursorCharge(centroided)[k]
mzk * zk
##      X46      X47
## 2046.175 2045.169
plot(centroided[[k[1]]], centroided[[k[2]]])
```

Figure 18: Comparing two MS² spectra.

```
plot(hclust(as.dist(compmat)))
```



12 Quantitative assessment of incomplete dissociation

Quantitation using isobaric reporter tags assumes complete dissociation between the reporter group (red on figure 19), balance group (blue) and peptide (the peptide reactive group is drawn in green). However, incomplete dissociation does occur and results in an isobaric tag (i.e reporter and balance groups) specific peaks.

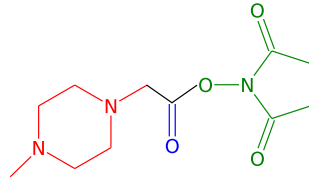


Figure 19: iTRAQ 4-plex isobaric tags reagent consist of three parts: (1) a charged reporter group (MZ of 114, 115, 116 and 117) that is unique to each of the four reagents (red), (2) an uncharged mass balance group (28-31 Da) (blue) and (3) a peptide reactive group (NHS ester) that binds to the peptide. In case of incomplete dissociation, the reporter and balance groups produce a specific peaks at MZ 145.

MSnbase provides, among others, a *ReporterIons* object for iTRAQ 4-plex that includes the 145 peaks, called *iTRAQ5*. This can then be used to quantify the experiment as show in section 7.1 to estimate incomplete dissociation for each spectrum.

```
iTRAQ5

## Object of class "ReporterIons"
## iTRAQ5: '4-plex iTRAQ and reporter + balance group' with 5 reporter ions
## - 114.1112 +/- 0.05 (red)
## - 115.1083 +/- 0.05 (green)
## - 116.1116 +/- 0.05 (blue)
## - 117.115 +/- 0.05 (yellow)
## - 145.1 +/- 0.05 (grey)

incompdiss <- quantify(itraqdata,
                       method = "trap",
                       reporters = iTRAQ5,
                       strict = FALSE,
                       verbose = FALSE)

head(exprs(incompdiss))

##      iTRAQ5.114 iTRAQ5.115 iTRAQ5.116 iTRAQ5.117 iTRAQ5.145
## X1      1347.6158  2247.3097  3927.6931  7661.1463  2063.8947
## X10     739.9861   799.3501   712.5983   940.6793   467.3615
## X11    27638.3582 33394.0252 32104.2879 26628.7278 13543.4565
## X12    31892.8928 33634.6980 37674.7272 37227.7119 11839.2558
## X13    26143.7542 29677.4781 29089.0593 27902.5608 12206.5508
## X14     6448.0829  6234.1957  6902.8903  6437.2303   427.6654
```


Figure 20 compares these intensities for the whole experiment.

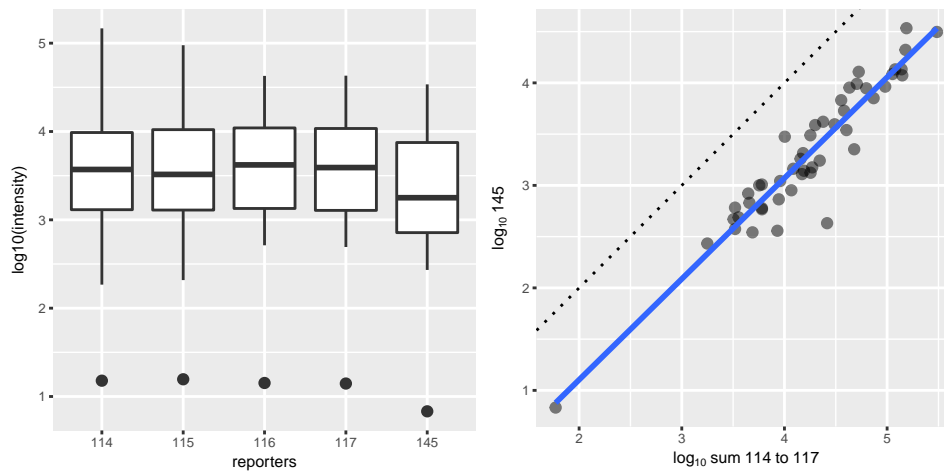


Figure 20: Boxplot and scatterplot comparing intensities of the 4 reporter ions (or their sum, on the right) and the incomplete dissociation specific peak.

13 Combining MSnSet instances

Combining mass spectrometry runs can be done in two different ways depending on the nature of these runs. If the runs represent repeated measures of identical samples, for instance multiple fractions, the data has to be combined along the row of the quantitation matrix: all the features (along the rows) represent measurements of the same set of samples (along the columns). In this situation, described in section 13.1, two experiments of dimensions n_1 (rows) by m (columns) and n_2 by m will produce a new experiment of dimensions $n_1 + n_2$ by m .

When however, different sets of samples have been analysed in different mass spectrometry runs, the data has to be combined along the columns of the quantitation matrix: some features will be shared across experiments and should thus be aligned on a same row in the new data set, whereas unique features to one experiment should be set as missing in the other one. In this situation, described in section 13.2, two experiments of dimensions n_1 by m_1 and n_2 by m_2 will produce a new experiment of dimensions $unique_{n_1} + unique_{n_2} + shared_{n_1, n_2}$ by $m_1 + m_2$. The two first terms of the first dimension will be complemented by NA values.

Default *MSnSet* feature names (X1, X2, ...) and sample names (iTRAQ4.114, iTRAQ4.115, iTRAQ4.116, ...) are not informative. The features and samples of these anonymous quantitative data-sets should be updated before being combined, to guide how to meaningfully merge them.

13.1 Combining identical samples

To simulate this situation, let us use quantitation data from the `itraqdata` object that is provided with the package as experiment 1 and the data from the `rawdata` *MSnExp* instance created at the very beginning of this document. Both experiments share the *same* default iTRAQ 4-plex reporter names as default sample names, and will thus automatically be combined along rows.

```
exp1 <- quantify(itraqdata, reporters = iTRAQ4,
                 verbose = FALSE)
sampleNames(exp1)
## [1] "iTRAQ4.114" "iTRAQ4.115" "iTRAQ4.116" "iTRAQ4.117"

centroided(rawdata) <- FALSE
exp2 <- quantify(rawdata, reporters = iTRAQ4,
                 verbose = FALSE)
sampleNames(exp2)
## [1] "iTRAQ4.114" "iTRAQ4.115" "iTRAQ4.116" "iTRAQ4.117"
```

It is important to note that the features of these independent experiments share the same default anonymous names: X1, X2, X3, ..., that however represent quantitation of distinct physical analytes. If the experiments were to be combined as is, it would result in an error because data points for the same *feature* name (say X1) and the same *sample name* (say iTRAQ4.114) have different values. We thus first update the feature names to explicitate that they originate from different experiments and represent

quantitation from different spectra using the convenience function `updateFeatureNames`. Note that updating the names of one experiment would suffice here.

```
head(featureNames(exp1))
## [1] "X1" "X10" "X11" "X12" "X13" "X14"

exp1 <- updateFeatureNames(exp1)
head(featureNames(exp1))
## [1] "X1.exp1" "X10.exp1" "X11.exp1" "X12.exp1" "X13.exp1" "X14.exp1"

head(featureNames(exp2))
## [1] "X1.1" "X2.1" "X3.1" "X4.1" "X5.1"

exp2 <- updateFeatureNames(exp2)
head(featureNames(exp2))
## [1] "X1.1.exp2" "X2.1.exp2" "X3.1.exp2" "X4.1.exp2" "X5.1.exp2"
```

The two experiments now share the same sample names and have different feature names and will be combined along the row. Note that all meta-data is correctly combined along the quantitation values.

```
exp12 <- combine(exp1, exp2)

## Warning in combine(experimentData(x), experimentData(y)):
## unknown or conflicting information in MIAPE field 'email'; using information from
## first object 'x'

dim(exp1)
## [1] 55 4

dim(exp2)
## [1] 5 4

dim(exp12)
## [1] 60 4
```

13.2 Combine different samples

Lets now create two *MSnSets* from the same raw data to simulate two different independent experiments that share some features. As done previously (see section 9), we combine the spectra based on the proteins they have been identified to belong to. Features can thus naturally be named using protein accession numbers. Alternatively, if peptide sequences would have been used as grouping factor in `combineFeatures`, then these would be good feature name candidates.

```
set.seed(1)
i <- sample(length(itraqdata), 35)
```

```

j <- sample(length(itraqdata), 35)
exp1 <- quantify(itraqdata[i], reporters = iTRAQ4,
                 verbose = FALSE)
exp2 <- quantify(itraqdata[j], reporters = iTRAQ4,
                 verbose = FALSE)
exp1 <- droplevels(exp1)
exp2 <- droplevels(exp2)
table(featureNames(exp1) %in% featureNames(exp2))

##
## FALSE  TRUE
##    12    23

exp1 <- combineFeatures(exp1,
                       groupBy = fData(exp1)$ProteinAccession)
exp2 <- combineFeatures(exp2,
                       groupBy = fData(exp2)$ProteinAccession)
head(featureNames(exp1))
## [1] "BSA"      "ECA0435" "ECA0469" "ECA0621" "ECA0631" "ECA0978"
head(featureNames(exp2))
## [1] "BSA"      "ECA0172" "ECA0435" "ECA0452" "ECA0469" "ECA0621"

```

The `droplevels` drops the unused `featureData` levels. This is required to avoid passing absent levels as `groupBy` in `combineFeatures`. Alternatively, one could also use `factor(fData(exp1)$ProteinAccession)` as `groupBy` argument.

The feature names are updated automatically by `combineFeatures`, using the `groupBy` argument. Proper feature names, reflecting the nature of the features (spectra, peptides or proteins) is critical when multiple experiments are to be combined, as this is done using common features as defined by their names (see below).

Sample names should also be updated to replace anonymous reporter names with relevant identifiers; the individual reporter data is stored in the `phenoData` and is not lost. A convenience function `updateSampleNames` is provided to append the *MSnSet*'s variable name to the already defined names, although in general, biologically relevant identifiers are preferred.

```

sampleNames(exp1)
## [1] "iTRAQ4.114" "iTRAQ4.115" "iTRAQ4.116" "iTRAQ4.117"

exp1 <- updateSampleNames(exp1)
sampleNames(exp1)
## [1] "iTRAQ4.114.exp1" "iTRAQ4.115.exp1" "iTRAQ4.116.exp1" "iTRAQ4.117.exp1"

sampleNames(exp1) <- c("Ctrl1", "Cond1", "Ctrl2", "Cond2")
sampleNames(exp2) <- c("Ctrl3", "Cond3", "Ctrl4", "Cond4")

```

At this stage, it is not yet possible to combine the two experiments, because their feature data is not compatible yet; they share the same feature variable labels, i.e. the feature data column names (spectrum, ProteinAccession, ProteinDescription, ...), but the part of the content is different because the original data was (in particular all the spectrum centric data: identical peptides in different runs will have different retention times, precursor intensities, ...). Feature data with identical labels (columns in the data frame) and names (row in the data frame) are expected to have the same data and produce an error if not conform.

```
stopifnot(all(fvarLabels(exp1) == fvarLabels(exp2)))
fData(exp1)["BSA", 1:4]
##      spectrum ProteinAccession  ProteinDescription PeptideSequence
## BSA          1             BSA bovine serum albumin          NYQEAK
fData(exp2)["BSA", 1:4]
##      spectrum ProteinAccession  ProteinDescription PeptideSequence
## BSA          52             BSA bovine serum albumin          QTALVELLK
```

Instead of removing these identical feature data columns, one can use a second convenience function, `updateFvarLabels`, to update feature labels based on the experiments variable name and maintain all the metadata.

```
exp1 <- updateFvarLabels(exp1)
exp2 <- updateFvarLabels(exp2)
head(fvarLabels(exp1))
## [1] "spectrum.exp1"          "ProteinAccession.exp1"  "ProteinDescription.exp1"
## [4] "PeptideSequence.exp1"    "file.exp1"              "retention.time.exp1"
head(fvarLabels(exp2))
## [1] "spectrum.exp2"          "ProteinAccession.exp2"  "ProteinDescription.exp2"
## [4] "PeptideSequence.exp2"    "file.exp2"              "retention.time.exp2"
```

It is now possible to combine `exp1` and `exp2`, including all the meta-data, with the `combine` method. The new experiment will contain the union of the feature names of the individual experiments with missing values inserted appropriately.

```
exp12 <- combine(exp1, exp2)
dim(exp12)
## [1] 35  8
pData(exp12)
## data frame with 0 columns and 8 rows
exprs(exp12)[25:28, ]
##           Ctrl11  Cond1  Ctrl12  Cond2  Ctrl13  Cond3  Ctrl14  Cond4
## ECA4513 10154.95 10486.94 11018.19 11289.552      NA      NA      NA      NA
```

```
## ECA4514 20396.49 20832.98 23280.82 23693.574 15965.52 16206.91 18455.76 18704.058
## ENO      50826.03 31978.10      NA  7528.967 39965.73 24967.40      NA  5925.663
## ECA0172      NA      NA      NA      NA 17593.55 18545.62 19361.84 18328.237

exp12

## MSnSet (storageMode: lockedEnvironment)
## assayData: 35 features, 8 samples
##   element names: exprs
## protocolData: none
## phenoData: none
## featureData
##   featureNames: BSA ECA0435 ... ECA4512 (35 total)
##   fvarLabels: spectrum.exp1 ProteinAccession.exp1 ... CV.iTRAQ4.117.exp2 (38
##     total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
## Combined [35,8] and [27,4] MSnSets Wed Jan  4 18:53:27 2017
## MSnbase version: 2.0.2
```

In summary, when experiments with different samples need to be combined (along the columns), one needs to (1) clarify the sample names using `updateSampleNames` or better manually, for biological relevance and (2) update the feature data variable labels with `updateFvarLabels`. The individual experiments (there can be more than 2) can then easily be combined with the `combine` method while retaining the meta-data.

If runs for the same sample (different fractions for example) need to be combined, one needs to (1) differentiate the feature provenance with `updateFeatureNames` prior to use `combine`.

13.3 Splitting and unsplitting MSnSet instances

A single *MSnSet* can also be split along the features/rows or samples/columns using the `split` method and a factor defining the splitting groups, resulting in an instance of class *MSnSetList*:

```
data(dunkley2006)
head(pData(dunkley2006))

##      membrane.prep fraction replicate
## M1F1A             1           1       A
## M1F4A             1           4       A
## M1F7A             1           7       A
## M1F11A            1          11       A
## M1F2B             1           2       B
## M1F5B             1           5       B
```

```
split(dunkley2006, dunkley2006$replicate)
## Instance of class 'MSnSetList' containig 2 objects.
## or, defining the appropriate annotation variable name
dun <- split(dunkley2006, "replicate")
```

Above, we split along the columns/samples, but the function would equally work with a factor of length equal to the number of rows of the *MSnSet* (or a feature variable name) to split along the rows/features.

Finally, the effect of `split` can be reverted by `unsplit`.

```
dun2 <- unsplit(dun, pData(dunkley2006)$replicate)
compareMSnSets(dunkley2006, dun2)
## [1] TRUE
```

See `?MSnSetList` for more details about the class, `split` and `unsplit` and comments about storing multiple assays pertaining the same experiment.

13.4 Averaging MSnSet instances

It is sometimes useful to average a set of replicated experiments to facilitate their visualisation. This can be easily achieved with the `averageMSnSet` function, which takes a list of valid *MSnSet* instances as input and creates a new object whose expression values are an average of the original values. A value of dispersion (`disp`) and a count of missing values (`nNA`) is recorded in the feature metadata slot. The average and dispersion are computed by default as the median and (non-parametric) coefficient of variation (see `?npcv` for details), although this can easily be parametrised, as described in `?averageMSnSet`.

The next code chunk illustrates the averaging function using three replicated experiments from [18] available in the *pRolocdata* package.

```
library("pRolocdata")
data(tan2009r1)
data(tan2009r2)
data(tan2009r3)
msnl <- MSnSetList(list(tan2009r1, tan2009r2, tan2009r3))
avgtan <- averageMSnSet(msnl)
head(exprs(avgtan))

##           X114      X115      X116      X117
## P20353 0.3605000 0.3035000 0.2095000 0.1265000
## P53501 0.4299090 0.1779700 0.2068280 0.1852625
## Q7KU78 0.1704443 0.1234443 0.1772223 0.5290000
## P04412 0.2567500 0.2210000 0.3015000 0.2205000
## Q7KJ73 0.2160000 0.1830000 0.3420000 0.2590000
## Q7JZNO 0.0965000 0.2509443 0.4771667 0.1750557
```

```
head(fData(avgtan)$disp)
##           X114      X115      X116      X117
## P20353 0.076083495 0.1099127 0.109691169 0.14650198
## P53501 0.034172542 0.2640556 0.005139653 0.17104568
## Q7KU78 0.023198743 0.4483795 0.027883087 0.04764499
## P04412 0.053414021 0.2146751 0.090972139 0.27903810
## Q7KJ73 0.000000000 0.0000000 0.000000000 0.00000000
## Q7JZN0 0.007681865 0.1959534 0.097873350 0.06210542

head(fData(avgtan)$nNA)
##           X114 X115 X116 X117
## P20353      1   1   1   1
## P53501      1   1   1   1
## Q7KU78      0   0   0   0
## P04412      1   1   1   1
## Q7KJ73      2   2   2   2
## Q7JZN0      0   0   0   0
```

We are going to visualise the average data on a principle component (PCA) plot using the `plot2D` function from the *pRoloc* package [19]. In addition, we are going to use the measure of dispersion to highlight averages with high variability by taking, for each protein, the maximum observed dispersion in the 4 samples. Note that in the default implementation, dispersions estimated from a single measurement (i.e. that had 2 missing values in our example) are set to 0; we will set these to the overall maximum observed dispersion.


```
disp <- rowMax(fData(avgtan)$disp)
disp[disp == 0] <- max(disp)
range(disp)
## [1] 0.01152877 1.20888923
library("pRoloc")
plot2D(avgtan, cex = 3 * disp)
```

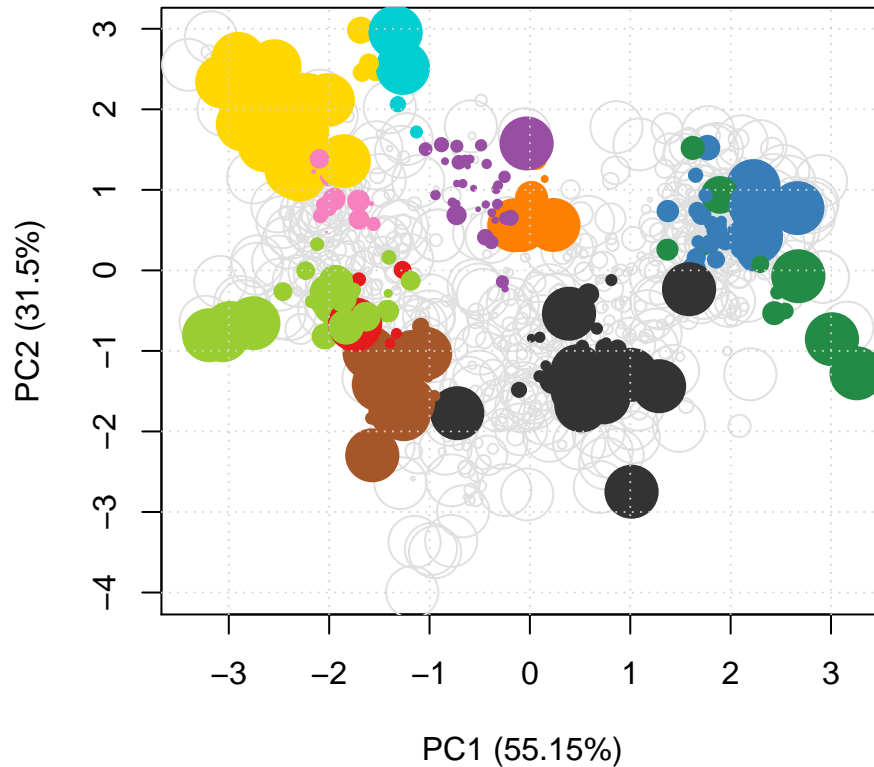


Figure 21: PCA plot of the averaged *MSnSet*. The point sizes are proportional to the dispersion of the protein quantitation across the averaged data.

14 MS^E data processing

MSnbase can also be used for MS^E data independent acquisition from Waters instrument. The MS^E pipeline depends on the Bioconductor *synapter* package [20] that produces *MSnSet* instances for individual acquisitions. The *MSnbase* infrastructure can subsequently be used to further combine experiments, as shown in section 13.2 and apply *top3* quantitation using the *topN* method.

15 Session information

- R version 3.3.2 (2016-10-31), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.36.0, Biobase 2.34.0, BiocGenerics 0.20.0, BiocParallel 1.8.1, BiocStyle 2.2.1, IRanges 2.8.1, MLInterfaces 1.54.0, MSnbase 2.0.2, ProtGenerics 1.6.0, Rcpp 0.12.8, RcppClassic 0.9.6, Rdisop 1.34.0, S4Vectors 0.12.1, XML 3.98-1.5, annotate 1.52.1, cluster 2.0.5, ggplot2 2.2.1, gplots 3.0.1, microbenchmark 1.4-2.1, msdata 0.14.0, mzR 2.8.0, pRoloc 1.14.5, pRolocdata 1.12.0, pryr 0.1.2, reshape2 1.4.2, zoo 1.7-14
- Loaded via a namespace (and not attached): BiocInstaller 1.24.0, DBI 0.5-1, DEoptimR 1.0-8, FNN 1.1, KernSmooth 2.23-15, MALDIquant 1.16, MASS 7.3-45, Matrix 1.2-7.1, MatrixModels 0.4-1, ModelMetrics 1.1.0, R6 2.2.0, RColorBrewer 1.1-2, RCurl 1.95-4.8, RSQLite 1.1-1, SparseM 1.74, TH.data 1.0-7, affy 1.52.0, affyio 1.44.0, assertthat 0.1, backports 1.0.4, base64enc 0.1-3, biomaRt 2.30.0, bitops 1.0-6, caTools 1.17.1, car 2.1-4, caret 6.0-73, class 7.3-14, codetools 0.2-15, colorspace 1.3-2, dendextend 1.3.0, digest 0.6.11, diptest 0.75-7, doParallel 1.0.10, dplyr 0.5.0, e1071 1.6-7, evaluate 0.10, flexmix 2.3-13, foreach 1.4.3, fpc 2.1-10, gbm 2.1.1, gdata 2.17.0, genefilter 1.56.0, ggvis 0.4.3, gridExtra 2.2.1, gtable 0.2.0, gtools 3.5.0, highr 0.6, htmltools 0.3.5, htmlwidgets 0.8, httpuv 1.3.3, hwriter 1.3.2, impute 1.48.0, iterators 1.0.8, jsonlite 1.2, kernlab 0.9-25, knitr 1.15.1, labeling 0.3, lattice 0.20-34, lazyeval 0.2.0, limma 3.30.7, lme4 1.1-12, lpSolve 5.6.13, magrittr 1.5, mclust 5.2.1, memoise 1.0.0, mgcv 1.8-16, mime 0.5, minqa 1.2.4, mlbench 2.1-1, modeltools 0.2-21, multcomp 1.4-6, munsell 0.4.3, mvtnorm 1.0-5, mzID 1.12.0, nlme 3.1-128, nloptr 1.0.4, nnet 7.3-12, pbkrtest 0.4-6, pcaMethods 1.66.0, pls 2.6-0, plyr 1.8.4, prabclus 2.2-6, preprocessCore 1.36.0, proxy 0.4-16, quantreg 5.29, randomForest 4.6-12, rda 1.0.2-2, rmarkdown 1.3, robustbase 0.92-7, rpart 4.1-10, rprojroot 1.1, sampling 2.8, sandwich 2.3-4, scales 0.4.1, sfsmisc 1.1-0, shiny 0.14.2, splines 3.3.2, stringi 1.1.2, stringr 1.1.0, survival 2.40-1, threejs 0.2.2, tibble 1.2, tools 3.3.2, trimcluster 0.1-2, vsn 3.42.3, whisker 0.3-2, xtable 1.8-2, yaml 2.1.14, zlibbioc 1.20.0

References

- [1] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0. URL: <http://www.R-project.org/>.
- [2] Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. H. Yang, and Jianhua Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, 5(10):–80, 2004. URL: <http://dx.doi.org/10.1186/gb-2004-5-10-r80>, doi:10.1186/gb-2004-5-10-r80.
- [3] Laurent Gatto and Kathryn S Lilley. MSnbase – an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation. *Bioinformatics*, 28(2):288–9, Jan 2012. doi:10.1093/bioinformatics/btr645.
- [4] Patrick G A Pedrioli, Jimmy K Eng, Robert Hubley, Mathijs Vogelzang, Eric W Deutsch, Brian Raught, Brian Pratt, Erik Nilsson, Ruth H Angeletti, Rolf Apweiler, Kei Cheung, Catherine E Costello, Henning Hermjakob, Sequin Huang, Randall K Julian, Eugene Kapp, Mark E McComb, Stephen G Oliver, Gilbert Omenn, Norman W Paton, Richard Simpson, Richard Smith, Chris F Taylor, Weimin Zhu, and Ruedi Aebersold. A common open representation of mass spectrometry data and its application to proteomics research. *Nat. Biotechnol.*, 22(11):1459–66, 2004. doi:10.1038/nbt1031.
- [5] Sandra Orchard, Luisa Montecchi-Palazzi, Eric W Deutsch, Pierre-Alain Binz, Andrew R Jones, Norman Paton, Angel Pizarro, David M Creasy, Jérôme Wojcik, and Henning Hermjakob. Five years of progress in the standardization of proteomics data 4th annual spring workshop of the hupo-proteomics standards initiative april 23-25, 2007 ecole nationale supérieure (ens), lyon, france. *Proteomics*, 7(19):3436–40, 2007. doi:10.1002/pmic.200700658.
- [6] Lennart Martens, Matthew Chambers, Marc Sturm, Darren Kesner, Fredrik Levander, Jim Shofstahl, Wilfred H Tang, Andreas Römpp, Steffen Neumann, Angel D Pizarro, Luisa Montecchi-Palazzi, Natalie Tasman, Mike Coleman, Florian Reisinger, Puneet Souda, Henning Hermjakob, Pierre-Alain Binz, and Eric W Deutsch. mzml - a community standard for mass spectrometry data. *Molecular & Cellular Proteomics : MCP*, 2010. doi:10.1074/mcp.R110.000133.
- [7] Chris F. Taylor, Norman W. Paton, Kathryn S. Lilley, Pierre-Alain Binz, Randall K. Julian, Andrew R. Jones, Weimin Zhu, Rolf Apweiler, Ruedi Aebersold, Eric W. Deutsch, Michael J. Dunn, Albert J. R. Heck, Alexander Leitner, Marcus Macht, Matthias Mann, Lennart Martens, Thomas A. Neubert, Scott D. Patterson, Peipei Ping, Sean L. Seymour, Puneet Souda, Akira Tsugita, Joel Vandekerckhove, Thomas M. Vondriska, Julian P. Whitelegge, Marc R. Wilkins, Ioannis Xenarios, John R. Yates, and Henning Hermjakob. The minimum information about a proteomics experiment (miape). *Nat Biotechnol*, 25(8):887–893, Aug 2007. URL: <http://dx.doi.org/10.1038/nbt1329>, doi:10.1038/nbt1329.

- [8] Chris F Taylor, Pierre-Alain Binz, Ruedi Aebersold, Michel Affolter, Robert Barkovich, Eric W Deutsch, David M Horn, Andreas Hhmer, Martin Kussmann, Kathryn Lilley, Marcus Macht, Matthias Mann, Dieter Mller, Thomas A Neubert, Janice Nickson, Scott D Patterson, Roberto Raso, Kathryn Resing, Sean L Seymour, Akira Tsugita, Ioannis Xenarios, Rong Zeng, and Randall K Julian. Guidelines for reporting the use of mass spectrometry in proteomics. *Nat. Biotechnol.*, 26(8):860–1, 2008. doi:10.1038/nbt0808-860.
- [9] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. URL: <http://had.co.nz/ggplot2/book>.
- [10] Andrew R. Jones, Martin Eisenacher, Gerhard Mayer, Oliver Kohlbacher, Jennifer Siepen, Simon J. Hubbard, Julian N. Selley, Brian C. Searle, James Shofstahl, Sean L. Seymour, Randall Julian, Pierre-Alain Binz, Eric W. Deutsch, Henning Hermjakob, Florian Reisinger, Johannes Griss, Juan Antonio Vizcano, Matthew Chambers, Angel Pizarro, and David Creasy. The mzIdentML data standard for mass spectrometry-based proteomics results. *Molecular & Cellular Proteomics*, 11(7), 2012. URL: <http://www.mcponline.org/content/11/7/M111.014381.abstract>, [arXiv:http://www.mcponline.org/content/11/7/M111.014381.full.pdf+html](http://www.mcponline.org/content/11/7/M111.014381.full.pdf+html), doi:10.1074/mcp.M111.014381.
- [11] Joseph M Foster, Sven Degroeve, Laurent Gatto, Matthieu Visser, Rui Wang, Johannes Griss, Rolf Apweiler, and Lennart Martens. A posteriori quality control for the curation and reuse of public proteomics data. *Proteomics*, 11(11):2182–94, 2011. doi:10.1002/pmic.201000602.
- [12] S Gibb and K Strimmer. MALDIquant: a versatile R package for the analysis of mass spectrometry data. *Bioinformatics*, 28(17):2270–1, Sep 2012. doi:10.1093/bioinformatics/bts447.
- [13] B M Bolstad, R A Irizarry, M Astrand, and T P Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–93, 2003.
- [14] Wolfgang Huber, Anja von Heydebreck, Holger Suetmann, Annemarie Poustka, and Martin Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18 Suppl. 1:S96–S104, 2002.
- [15] Natasha A Karp, Wolfgang Huber, Pawel G Sadowski, Philip D Charles, Svenja V Hester, and Kathryn S Lilley. Addressing accuracy and precision issues in itraq quantitation. *Mol. Cell Proteomics*, 9(9):1885–97, 2010. doi:10.1074/mcp.M900628-MCP200.
- [16] A C Paoletti, T J Parmely, C Tomomori-Sato, S Sato, D Zhu, R C Conaway, J W Conaway, L Florens, and M P Washburn. Quantitative proteomic analysis of distinct mammalian mediator complexes using normalized spectral abundance factors. *Proc Natl Acad Sci U S A*, 103(50):18928–33, Dec 2006. doi:10.1073/pnas.0606379103.
- [17] N M Griffin, J Yu, F Long, P Oh, S Shore, Y Li, J A Koziol, and J E Schnitzer. Label-free, normalized quantification of complex mass spectrometry data for proteomic analysis. *Nat Biotechnol*, 28(1):83–9, Jan 2010. doi:10.1038/nbt.1592.
- [18] Denise J Tan, Heidi Dvinge, Andy Christoforou, Paul Bertone, Alfonso A Martinez, and Kathryn S Lilley. Mapping organelle proteins and protein complexes in drosophila melanogaster. *J Proteome Res*, 8(6):2667–78, Jun 2009. doi:10.1021/pr800866n.

- [19] L Gatto, L M Breckels, S Wieczorek, T Burger, and K S Lilley. Mass-spectrometry-based spatial proteomics data analysis using pRoloc and pRolocdata. *Bioinformatics*, 30(9):1322–4, May 2014. doi:[10.1093/bioinformatics/btu013](https://doi.org/10.1093/bioinformatics/btu013).
- [20] N J Bond, P V Shliha, K S Lilley, and L Gatto. Improving qualitative and quantitative performance for MS^e-based label-free proteomics. *J Proteome Res*, 12(6):2340–53, Jun 2013. URL: <http://pubs.acs.org/doi/abs/10.1021/pr300776t>, doi:[10.1021/pr300776t](https://doi.org/10.1021/pr300776t).