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

Laurent Gatto
lg390@cam.ac.uk
Cambridge Center for Proteomics
Kathryn S. Lilley Group
University of Cambridge

August 27, 2012

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 (R Development Core Team, 2011) by providing specific data classes and methods and (2) enabling the utilisation of throughput-high data analysis pipelines provided by the Bioconductor (Gentleman et al., 2004) project.

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

Contents

Foreword

MSnbase is under active developed; 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).

You are welcome to contact me for questions, bugs, typos or suggestions about MSnbase. If you wish to reach a broader audience for general questions about proteomics analysis using `R`, you may want to use the Bioconductor mailing list[1].

1 Introduction

MSnbase (Gatto and Lilley, 2012) aims are 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 `Expression`

[1] `https://stat.ethz.ch/mailman/listinfo/bioconductor`

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 5.2 on page 12 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` format[2]

Raw data The `XML`-based formats, `mzXML` (Pedrioli et al., 2004), `mzData` (Orchard et al., 2007) and `mzML` (Martens et al., 2010) can be imported with the `readMSData` function, as illstrated 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)
```

Either MS1 or MS2 spectra 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.

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

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).

See also section 6.2 to import quantitative data stored in spreadsheets into `R` for further processing using MSnbase.

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"
 Object size in memory: 1.86 Mb
- - - Spectra data - - -
 MS level(s): 2
 Number of MS1 acquisitions: 1
 Number of MSn scans: 55
 Number of precursor ions: 55
 55 unique MZs
 Precursor MZ's: 401.74 - 1236.1
 MSn M/Z range: 100 2069.27
 MSn retention times: 19:9 - 50:18 minutes
- - - Processing information - - -
Data loaded: Wed May 11 18:54:39 2011
 MSnbase version: 1.1.22
- - - Meta data  - - -
phenoData: none
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
X1         1              BSA                      bovine serum albumin
X10       10          ECA1422 glucose-1-phosphate cytidylyltransferase
X11       11          ECA4030          50S ribosomal subunit protein L4
X12       12          ECA3882                    chaperone protein DnaK
X13       13          ECA1364      succinyl-CoA synthetase alpha chain
X14       14          ECA0871              NADP-dependent malic enzyme
    PeptideSequence
X1           NYQEAK
X10 VTLVDTGEHSMTGGR
X11           SPIWR
X12        TAIDDALK
X13          SILINK
X14    DFEVVNNESDPR
```

As illustrated above, showing the experiment textually displays it's content:

- Information about the raw data, i.e. the spectra.
- Specific information about the experiment processing[3] 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 general information about the experiment as well as MIAPE (Minimum Information About a Proteomics Experiment) information (Taylor et al., 2007, 2008). 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 a `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
```

[3]this part will be automatically updated when the object is modified with it's *ad hoc* methods, as illustrated later

```
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 with 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 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
 - 114.1 +/- 0.05 (red)
 - 115.1 +/- 0.05 (green)
 - 116.1 +/- 0.05 (blue)
 - 117.1 +/- 0.05 (yellow)
```

3 Plotting raw data

3.1 Default plots

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 1).

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

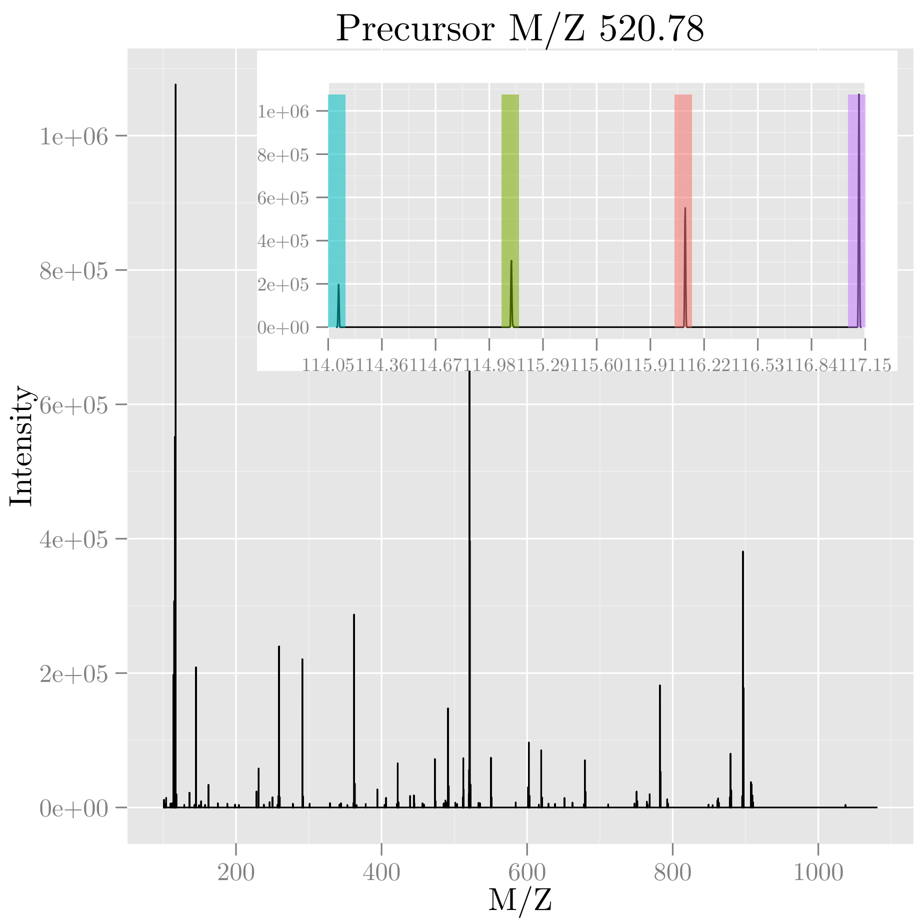

Figure 1: Raw MS2 spectrum with details about reporter ions.

It is also possible to plot all spectra of an experiment (figure 2). Lets start by subsetting the `itraqdata` experiment using the protein accession numbers included in the feature metadata, and keep the 3 from the *BSA* protein.

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

```
Object of class "MSnExp"
 Object size in memory: 0.1 Mb
- - - Spectra data - - -
 MS level(s): 2
 Number of MS1 acquisitions: 1
 Number of MSn scans: 3
 Number of precursor ions: 3
 3 unique MZs
 Precursor MZ's: 434.95 - 651.92
 MSn M/Z range: 100 1351.77
 MSn retention times: 19:9 - 36:17 minutes
- - - Processing information - - -
Data loaded: Wed May 11 18:54:39 2011
Data [logically] subsetted 3 spectra: Mon Aug 27 21:53:41 2012
 MSnbase version: 1.1.22
- - - Meta data  - - -
phenoData: none
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 2.

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

```
> plot(bsa, reporters = iTRAQ4, full = FALSE)
```

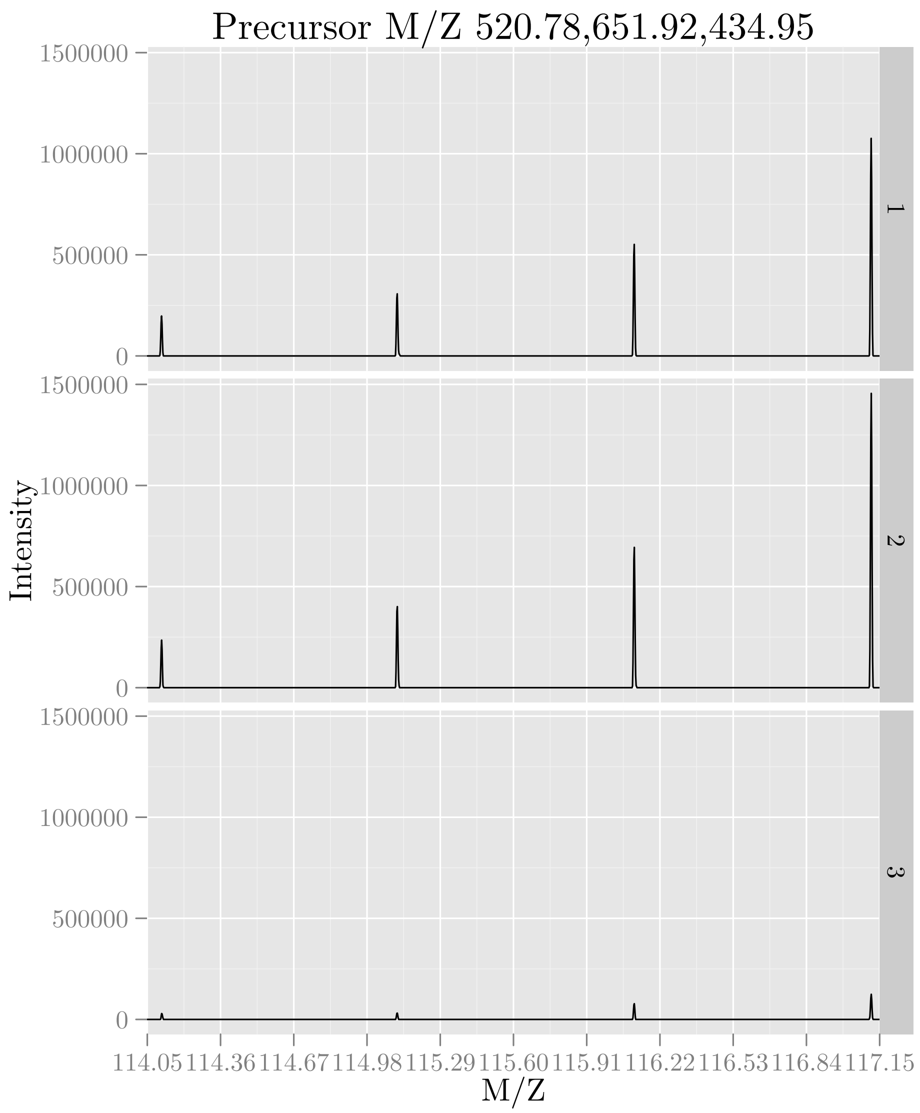

Figure 2: Experiment-wide raw MS2 spectra. The y-axes of the individual spectra are automatically rescaled to the same range. See section 6.4 to rescale peaks identically.

MSnbase uses the ggplot2 package to generate figures, which can subsequently easily be customised. More details about ggplot2 can be found in Wickham (2009) (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 about the object with `summary(p)`. To view the data frame used to generate the plot, use `p@data`.

4 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 firs 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 current (`z="tic"`), number of peaks in the MS2 spectra `z="peaks.count"`) or, when multiple data files were loaded, file `z="file"`), as illustrated on figure 3. The lower right panel is produced for only a subset of proteins. See the method documentation for more details.

The `plotDensity` method illustrates the distribution of several parameters of interest (see figure 4). Similarly to `plot2d`, the first argument is an `MSnExp` instance. The second is one of `precursor.mz`, `peaks.count` or `tic`, whose density will be plotted. An optional third argument specifies whether the x axes should be logged.

The `plotMzDelta` method[4] implements the M/Z delta plot from Foster et al. (2011) 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 shown outstanding bars at amino acid residu masses. More details and parameters are described in the method documentation (`?plotMzDelta`). Figure 5 has been generated using the PRIDE experiment 12011, as in Foster et al. (2011).

In section 7 on page 25, we illustrate how to assess incomplete reporter ion dissociation.

[4]The code to generate the histograms has been contributed by Guangchuang Yu from Jinan University, China.

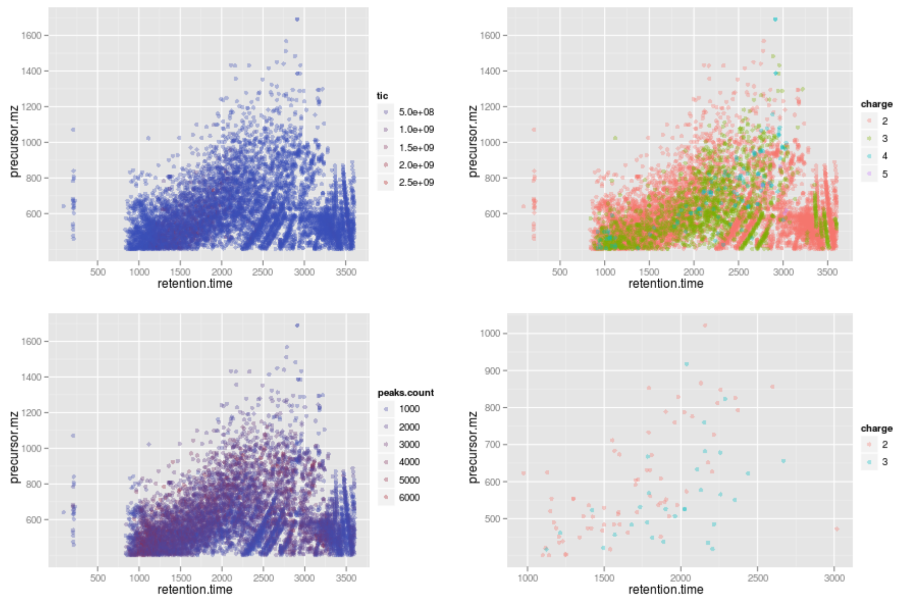

Figure 3: Illustration of the `plot2d` output.

5 Data processing

5.1 Cleaning spectra

There are several methods implemented to perform basic data 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 `tic` method before (object `itraqdata`) and after (object `experiment`) for spectrum `X55`.

```
> experiment <- removePeaks(itraqdata, t = 400, verbose = FALSE)
> ## total ion current
> tic(itraqdata[["X55"]])

[1] 555408.8

> tic(experiment[["X55"]])

[1] 499769.6
```

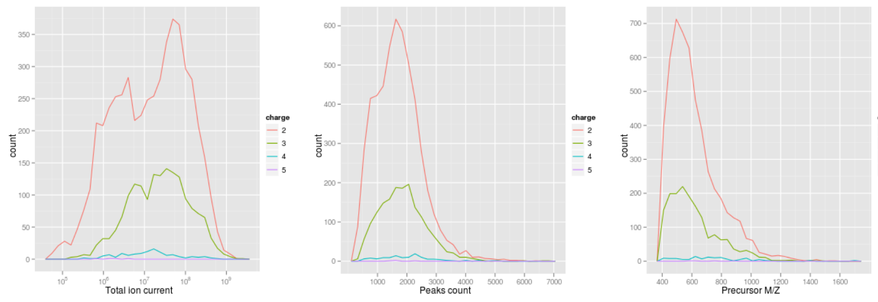

Figure 4: Illustration of the `plotDensity` output.

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 7 on page 14.

```
> ## number of peaks
> peaksCount(itraqdata[["X55"]])

[1] 1726

> peaksCount(experiment[["X55"]])

[1] 1726

> experiment <- clean(experiment, verbose = FALSE)
> peaksCount(experiment[["X55"]])

[1] 442
```

5.2 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 then `min(mzlim)` or greater then `max(mzmax)` 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 1.86 Mb.

```
> range(mz(itraqdata[["X55"]]))

[1] 100.0002 977.6636
```

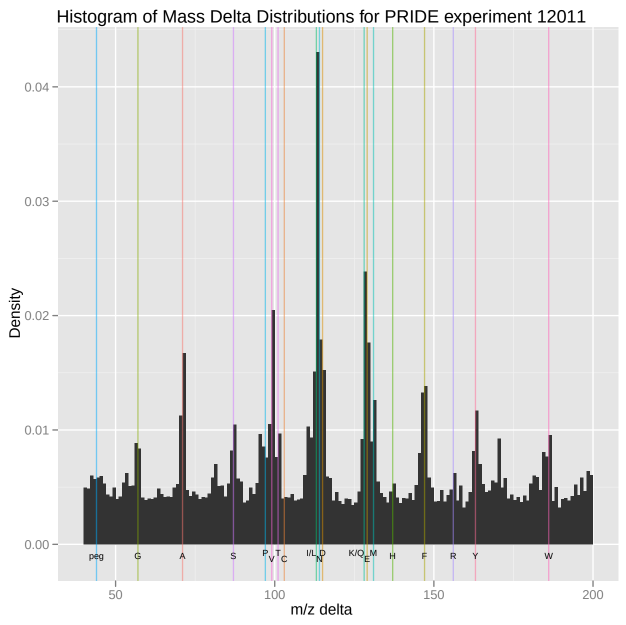

Figure 5: Illustration of the `plotMzDelta` output for the PRIDE experiment 12011, as in figure 4A from Foster et al. (2011).

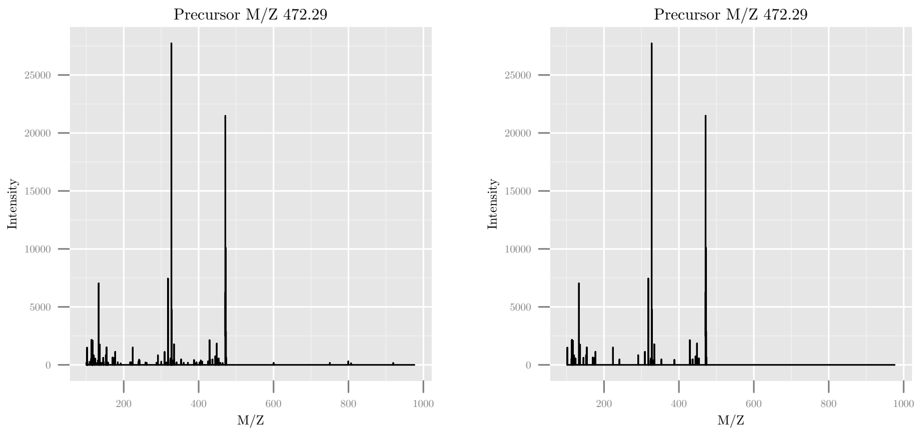

Figure 6: Same spectrum before (left) and after setting peaks <= 400 to 0.

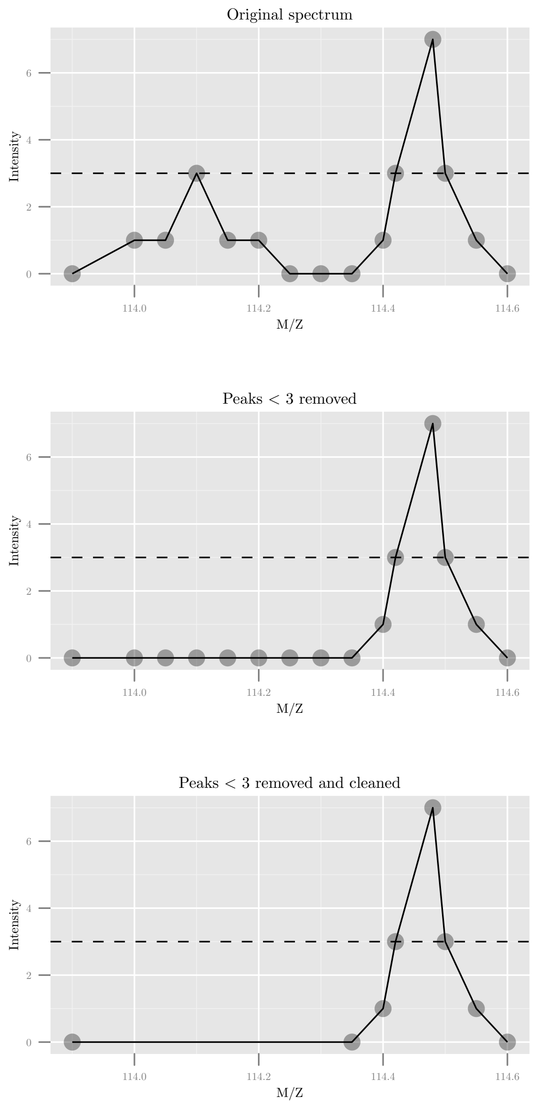

Figure 7: This figure illustrated the effect or the `removePeaks` and `clean` methods. The left-most spectrum displays two peaks, of max height 3 and 7 respectively. The middle spectrum shows the result of calling `removePeaks` with argument `t=3`, which sets all data points of the first peak, whose 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.

```
> experiment <- trimMz(experiment, mzlim = c(112,120))
> range(mz(experiment[["X55"]]))

[1] 113.0532 117.1219

> experiment

Object of class "MSnExp"
 Object size in memory: 0.3 Mb
- - - Spectra data - - -
 MS level(s): 2
 Number of MS1 acquisitions: 1
 Number of MSn scans: 55
 Number of precursor ions: 55
 55 unique MZs
 Precursor MZ's: 401.74 - 1236.1
 MSn M/Z range: 112.04 119.87
 MSn retention times: 19:9 - 50:18 minutes
- - - Processing information - - -
Data loaded: Wed May 11 18:54:39 2011
Curves <= 400 set to '0': Mon Aug 27 21:53:51 2012
Spectra cleaned: Mon Aug 27 21:53:58 2012
MZ trimmed [112..120]
 MSnbase version: 1.1.22
- - - Meta data  - - -
phenoData: none
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 From spectra to quantitative expression data

6.1 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 it's expected `mz` value and is searched for within `mz` $\pm$ `width`. If no data is found, `NA` is returned.

```
> mz(iTRAQ4)

[1] 114.1 115.1 116.1 117.1

> 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 8: `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 m 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 <- quantify(experiment,
+                  method = "trap",
+                  reporters = iTRAQ4,
+                  strict = FALSE,
+                  verbose = FALSE) > 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
```

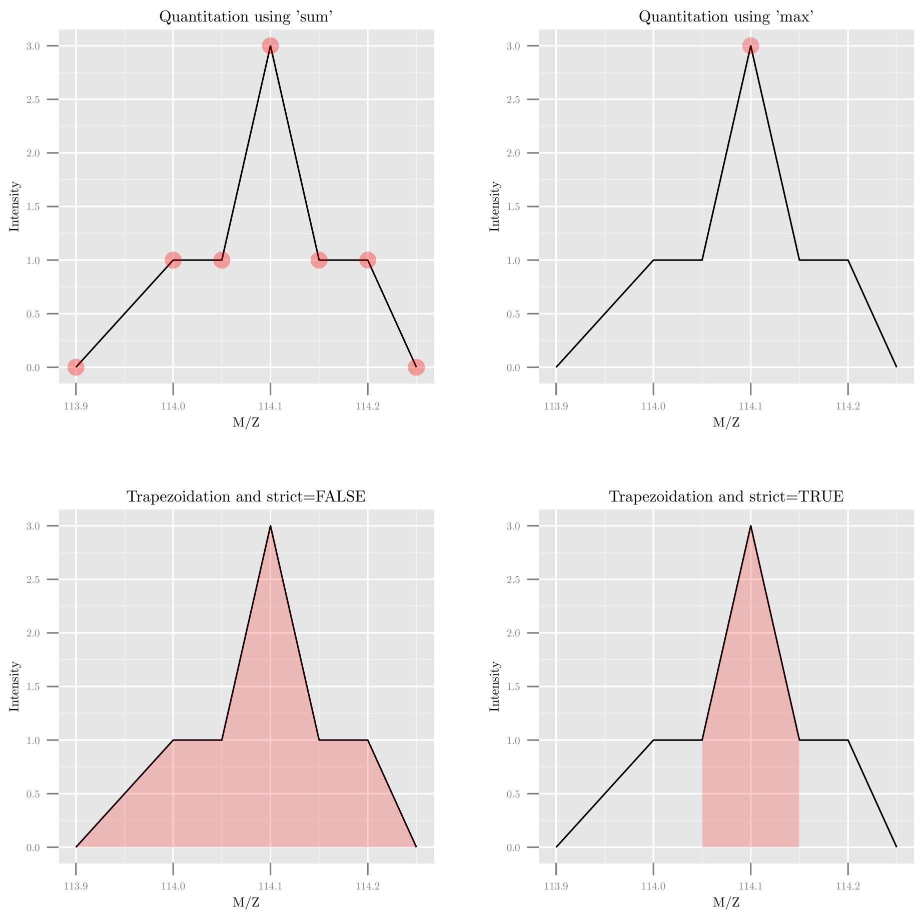

Figure 8: The different quantitation methods are illustrated above. Quantitation using `sum` sums all the data points in the peaks to produce, for this example, 7, whereas method `max` only uses the peak's maximum intensity, 3. `Trapezoidation` calculates the area under the peak taking the full with into account (using `strict=FALSE` gives 0.375) or only the width as defined by the reporter (using `strict=TRUE` gives 0.2).

```
  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
Curves <= 400 set to '0': Mon Aug 27 21:53:51 2012
Spectra cleaned: Mon Aug 27 21:53:58 2012
MZ trimmed [112..120]
iTRAQ4 quantification by trapezoidation: Mon Aug 27 21:54:12 2012
 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
```

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, as we do not expect any missing peaks.

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

```
FALSE  TRUE
  219     1
```

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

```
[1] 0
```

The `filterNA` method allows to filter features based on the proportion of missing data their carry, specified using the `pNA` argument. The same result as above could have been achieved with `qnt <- filterNA(qnt, pNA = 0)`, specifying that we allow 0 percent of missing data. See also the `plotNA` method to get an overview of the completeness of a data set.

6.2 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.

6.3 Peak adjustments

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

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 substract 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   1402.9442  2214.0346  3762.2549  8114.4429
X10   779.4666   793.0792   678.8083   985.2003
X11 29034.3781 33271.0470 31484.7131 27279.1383
X12 33618.9092 33046.3075 37031.6133 38492.1376
X13 27508.0038 29440.9296 28390.4561 28814.2463
X14  6809.7600  6090.7894  6799.5030  6636.1450
```

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

6.4 Normalisation

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

- `quantiles` Applies quantile normalisation (Bolstad et al., 2003) as implemented in the `normalize.quantiles` function of the preprocessCore package.
- `quantiles.robust` Applies robust quantile normalisation (Bolstad et al., 2003) as implemented in the `normalize.quantiles.robust` function of the preprocessCore package.
- `vsn` Applies variance stabilisation normalization (Huber et al., 2002) 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.

```
> 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 9 and figure 10 reproduces figure 3 of Karp et al. (2010) 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 this first trim the spectra using method `trimMz` as illustrated in section 5, then normalise the `MSnExp` with `normalise` using `method="max"` as illustrated above and plot the data using `plot` (figure 11).

6.5 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 6.1 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)
```

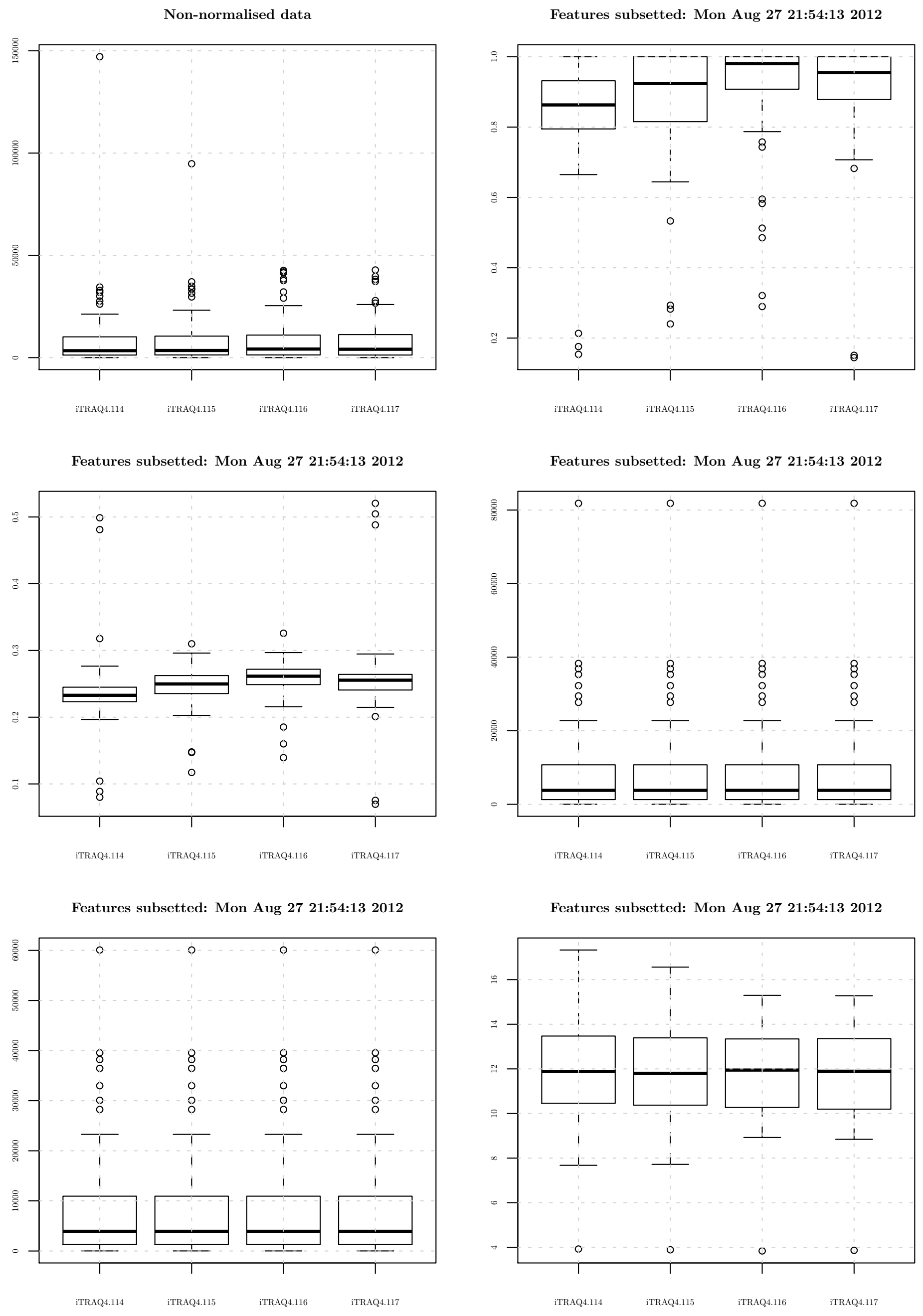

Figure 9: Comparison of the normalisation `MSnSet` methods. Note that vsn also glog-transforms the intensities.

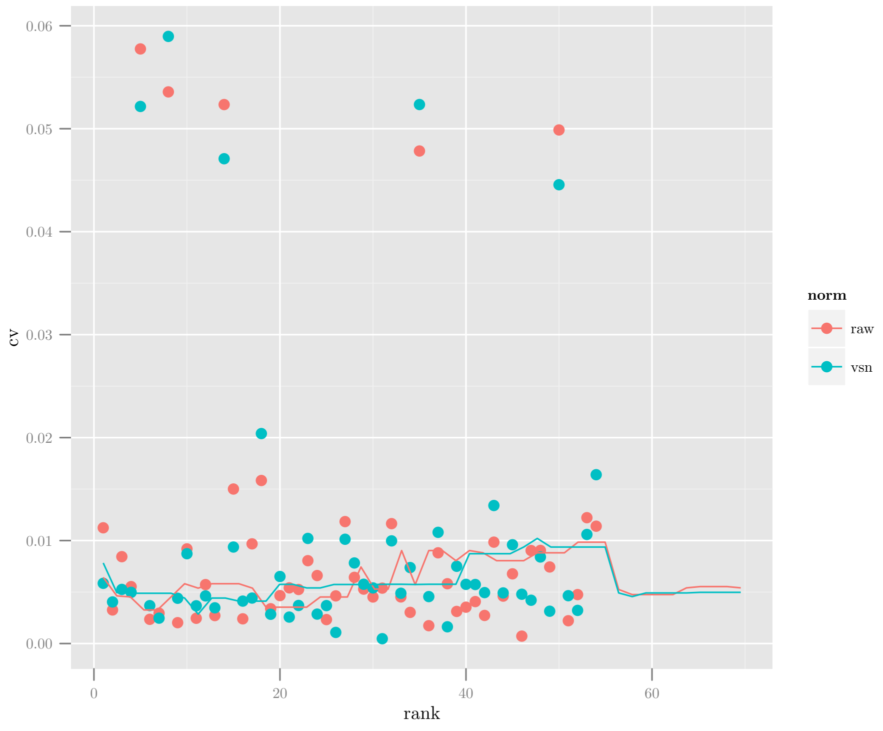

Figure 10: CV versus signal intensity comparison for log2 and vsn transformed data. Lines indicate running CV medians.

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

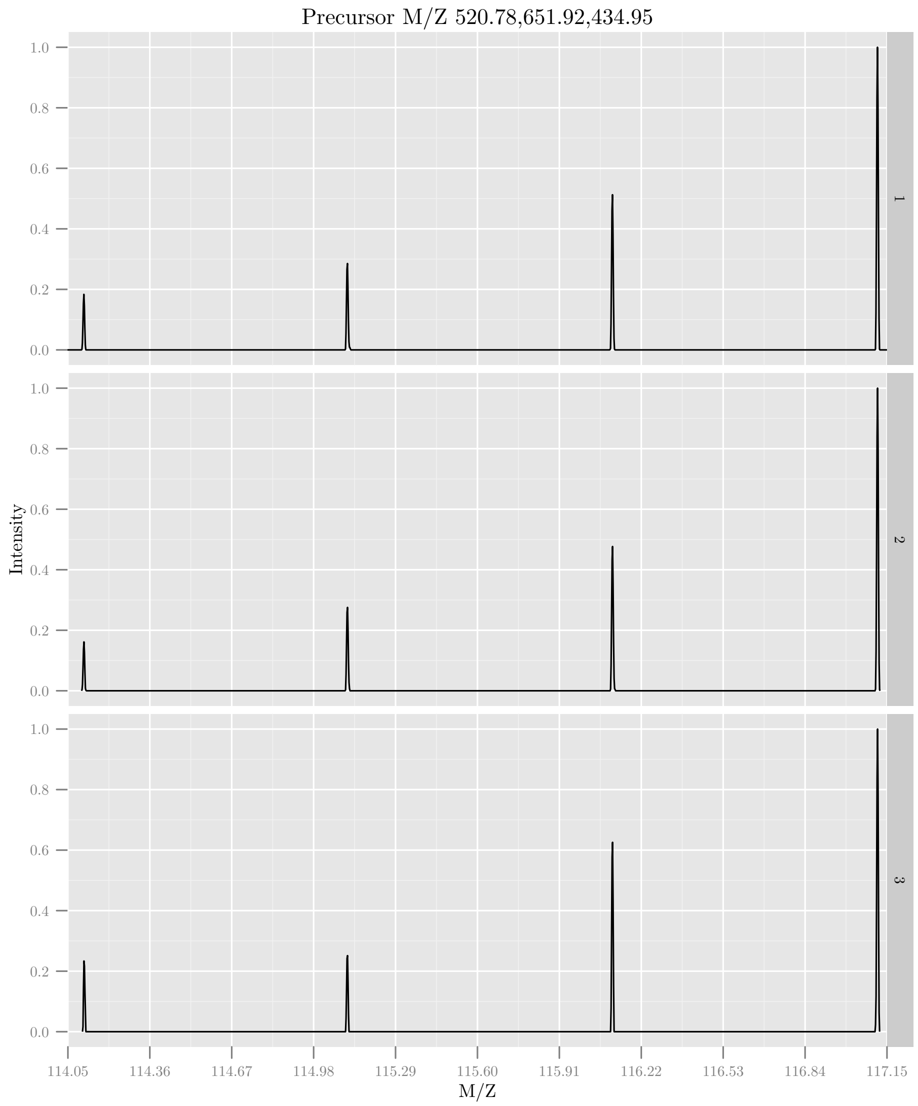

Figure 11: Experiment-wide normalised MS2 spectra. The y-axes of the individual spectra is now rescaled between 0 and 1 (highest peak), as opposed to figure 2.

compute respectively the mean, media, sum, weighted mean or median polish of the features to be aggregated). Alternatively, is 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
  sampleNames: iTRAQ4.114 iTRAQ4.115 iTRAQ4.116 iTRAQ4.117
  varLabels: mz reporters
  varMetadata: labelDescription
featureData
  featureNames: X1 X41 ... X27 (40 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
Curves <= 400 set to '0': Mon Aug 27 21:53:51 2012
Spectra cleaned: Mon Aug 27 21:53:58 2012
MZ trimmed [112..120]
iTRAQ4 quantification by trapezoidation: Mon Aug 27 21:54:12 2012
Features subsetted: Mon Aug 27 21:54:13 2012
Combined 54 features into 40 using median: Mon Aug 27 21:54:36 2012
 MSnbase version: 1.1.22
```

6.6 Label-free MS2 quantitation

Note that if samples are not multiplexed, label-free MS2 quantitation is possible using MSnbase. Once individual spectra have been assigned to peptides and proteins, it becomes straightforward to estimate protein quantities using the spectral counting method, as illustrated in section 6.5, when the `groupBy` argument is defined.

7 Quantitative assessment of incomplete dissociation

Quantitation using isobaric reporter tags assumes complete dissociation between the reporter group (red on figure 12), 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.

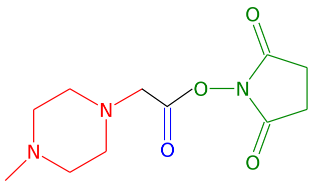

Figure 12: iTRAQ 4-plex isobaric tags reagent consist of three parts: (1) a charged reporter group (MZ of 114, 115, 116 and 117) that is unique to each of the four reagents (red), (2) an uncharged mass balance group (28-31 Da) (blue)and (3) a peptide reactive group (NHS ester) that binds to the peptide. 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 6.1 to estimate incomplete dissociation for each spectrum.

```
> iTRAQ5

Object of class "ReporterIons"
iTRAQ4: '4-plex iTRAQ with isobaric tag' with 5 reporter ions
 - 114.1 +/- 0.05 (red)
 - 115.1 +/- 0.05 (green)
 - 116.1 +/- 0.05 (blue)
 - 117.1 +/- 0.05 (yellow)
 - 145.1 +/- 0.05 (grey)

> incompdiss <- quantify(itraqdata,
+                         method = "trap",
+                         reporters = iTRAQ5,
+                         strict = FALSE,
+                         verbose = FALSE) > incompdiss <- quantify(itraqdata,
+                         method = "trap",
+                         reporters = iTRAQ5,
+                         strict = FALSE,
+                         verbose = FALSE) > 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 13 compares these intensities for the whole experiment.

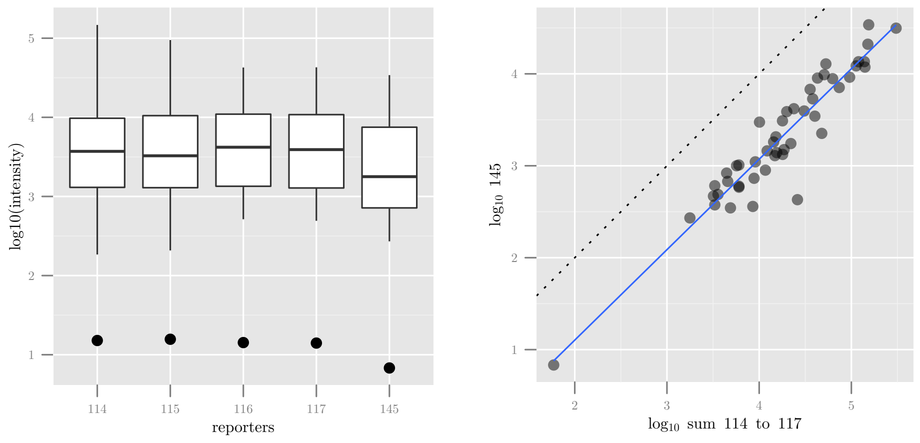

Figure 13: Boxplot and scatterplot comparing intensities of the 4 reporter ions (or their sum, on the right) and the incomplete dissociation specific peak.

8 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 combine 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 8.1, 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 8.2, 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.

8.1 Combining identical samples

To simulate this situation, let us use quantiation 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)
> exp1 <- quantify(itraqdata, reporters = iTRAQ4, verbose = FALSE)
> exp1 <- quantify(itraqdata, reporters = iTRAQ4, verbose = FALSE)

> sampleNames(exp1)

[1] "iTRAQ4.114" "iTRAQ4.115" "iTRAQ4.116" "iTRAQ4.117"

> exp2 <- quantify(rawdata, reporters = iTRAQ4, verbose = FALSE) >
exp2 <- quantify(rawdata, reporters = iTRAQ4, verbose = FALSE) > exp2
<- quantify(rawdata, reporters = iTRAQ4, verbose = FALSE)
> sampleNames(exp2)

[1] "iTRAQ4.114" "iTRAQ4.115" "iTRAQ4.116" "iTRAQ4.117"
```

It 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 experiment 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" "X2" "X3" "X4" "X5"
```

```
> exp2 <- updateFeatureNames(exp2)
> head(featureNames(exp2))
```

```
[1] "X1.exp2" "X2.exp2" "X3.exp2" "X4.exp2" "X5.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)
> dim(exp1)
```

```
[1] 55  4
```

```
> dim(exp2)
```

```
[1] 5 4
```

```
> dim(exp12)
```

```
[1] 60  4
```

8.2 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 6.5), we combine the specta 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)
> exp1 <- quantify(itraqdata[i], reporters = iTRAQ4, verbose = FALSE)
> exp1 <- quantify(itraqdata[i], reporters = iTRAQ4, verbose = FALSE)

> exp2 <- quantify(itraqdata[j], reporters = iTRAQ4, verbose = FALSE)
> exp2 <- quantify(itraqdata[j], reporters = iTRAQ4, verbose = FALSE)
> exp2 <- quantify(itraqdata[j], reporters = iTRAQ4, verbose = FALSE)

> 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] "X1"  "X29" "X39" "X24" "X42" "X36"

> featureNames(exp1) <- fData(exp1)$ProteinAccession
> featureNames(exp2) <- fData(exp2)$ProteinAccession
> head(featureNames(exp1))

[1] "BSA"     "ECA0435" "ECA0469" "ECA0621" "ECA0631" "ECA0978"
```

Setting the feature names 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 experiements variable name and maintain all the metadata.

```
> exp1 <- updateFvarLabels(exp1)
> exp2 <- updateFvarLabels(exp2)
> head(fvarLabels(exp1))
```

```
[1] "spectrum.exp1"           "ProteinAccession.exp1"
[3] "ProteinDescription.exp1" "PeptideSequence.exp1"
[5] "index.exp1"              "file.exp1"
```

```
> head(fvarLabels(exp2))
```

```
[1] "spectrum.exp2"           "ProteinAccession.exp2"
[3] "ProteinDescription.exp2" "PeptideSequence.exp2"
[5] "index.exp2"              "file.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)
```

```
         mz reporters
Ctrl1 114.1    iTRAQ4
Cond1 115.1    iTRAQ4
Ctrl2 116.1    iTRAQ4
Cond2 117.1    iTRAQ4
Ctrl3 114.1    iTRAQ4
Cond3 115.1    iTRAQ4
Ctrl4 116.1    iTRAQ4
Cond4 117.1    iTRAQ4
```

```
> exprs(exp12)[25:28, ]

           Ctrl1    Cond1    Ctrl2     Cond2    Ctrl3    Cond3    Ctrl4
ECA4513 10154.95 10486.94 11018.19 11289.552       NA       NA       NA
ECA4514 20396.49 20832.98 23280.82 23693.574 15965.52 16206.91 18455.76
ENO     50826.03 31978.10       NA  7528.967 39965.73 24967.40       NA
ECA0172       NA       NA       NA        NA 17593.55 18545.62 19361.84
            Cond4
ECA4513        NA
ECA4514 18704.058
ENO      5925.663
ECA0172 18328.237

> exp12

MSnSet (storageMode: lockedEnvironment)
assayData: 35 features, 8 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Ctrl1 Cond1 ... Cond4 (8 total)
  varLabels: mz reporters
  varMetadata: labelDescription
featureData
  featureNames: BSA ECA0435 ... ECA4512 (35 total)
  fvarLabels: spectrum.exp1 ProteinAccession.exp1 ...
    collision.energy.exp2 (30 total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: No annotation
- - - Processing information - - -
Combined [35,8] and [27,4] MSnSets Mon Aug 27 21:55:06 2012
 MSnbase version: 1.1.22
```

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 combines, one needs to (1) differentiate the feature provenance with `updateFeatureNames` prior to use `combine`.

9 Session information

- R version 2.15.1 (2012-06-22), `x86_64-unknown-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=C`, `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, stats, tools, utils
- Other packages: Biobase 2.16.0, BiocGenerics 0.2.0, MSnbase 1.4.1, Rcpp 0.9.13, RcppClassic 0.9.2, Rdisop 1.16.0, cacheSweave 0.6-1, codetools 0.2-8, doMC 1.2.5, filehash 2.2-1, foreach 1.4.0, formatR 0.6, ggplot2 0.9.1, highlight 0.3.2, iterators 1.0.6, multicore 0.1-7, mzR 1.2.2, parser 0.0-16, pgfSweave 1.3.0, plyr 1.7.1, reshape 0.8.4, stashR 0.3-5, zoo 1.7-7
- Loaded via a namespace (and not attached): BiocInstaller 1.4.7, IRanges 1.14.4, MASS 7.3-20, RColorBrewer 1.0-5, affy 1.34.0, affyio 1.24.0, colorspace 1.1-1, compiler 2.15.1, dichromat 1.2-4, digest 0.5.2, labeling 0.1, lattice 0.20-10, limma 3.12.1, memoise 0.1, munsell 0.3, parallel 2.15.1, preprocessCore 1.18.0, proto 0.3-9.2, reshape2 1.2.1, scales 0.2.1, stats4 2.15.1, stringr 0.6.1, tikzDevice 0.6.2, vsn 3.24.0, zlibbioc 1.2.0

References

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.

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.

L Gatto and K 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.

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. doi: 10.1186/gb-2004-5-10-r80. URL `http://dx.doi.org/10.1186/gb-2004-5-10-r80`.

Wolfgang Huber, Anja von Heydebreck, Holger Sueltmann, 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.

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.

Lennart Martens, Matthew Chambers, Marc Sturm, Darren Kes sner, Fredrik Levander, Jim Shofstahl, Wilfred H Tang, Andreas Ro mpp, Steffen Neumann, Angel D Pizarro, Lu isa Montecchi-Palazzi, Natalie Tasman, Mike Coleman, Florian Reisinger, Pune et 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.

Sandra Orchard, Luisa Montechi-Palazzi, Eric W Deutsch, Pierre-Alain Binz, Andrew R Jones, Norman Paton, Angel Pizarro, David M Creasy, JÃl'rÃt'me 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 supÃl'rieure (ens), lyon, france. *Proteomics*, 7(19):3436–40, 2007. doi: 10.1002/pmic.200700658.

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.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

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, Ioannnis Xenarios, John R. Yates, and Henning Hermjakob. The minimum information about a proteomics experiment (miape). *Nat Biotechnol*, 25(8):887–893, Aug 2007. doi: 10.1038/nbt1329. URL http://dx.doi.org/10.1038/nbt1329.

Chris F Taylor, Pierre-Alain Binz, Ruedi Aebersold, Michel Affolter, Robert Barkovich, Eric W Deutsch, David M Horn, Andreas HÃijhmer, Martin Kussmann, Kathryn Lilley, Marcus Macht, Matthias Mann, Dieter MÃijller, 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.

Hadley Wickham. *ggplot2: elegant graphics for data analysis.* Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book.