Mass spectrometry-based proteomics

Laurent Gatto

2019-03-24

Biological information flow

There are three main biological entities that respectively store information, act as data intermediates, and the functional units, and the figure below show how information flows between these three levels.

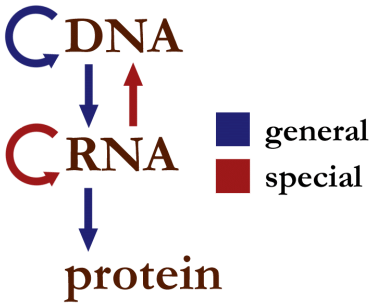


Figure 1: Information flow in biological systems (Source *Central dogma of biology* on Wikipedia).

DNA, that lives in the nucleus of cells, is the central information storage mechanism, and encodes the blueprint of the functional units as genes. DNA is **transcribed** into **messenger RNA** (mRNA), that relocalises outside the nucleus and is further processed into its mature *exon*-only form after removal of the non-coding *introns* sequences. Finally, the mRNA is translated by the ribosomial machinery into **proteins** directly into the endoplasmic reticulum (ER) where they are then redirected to their final destination.

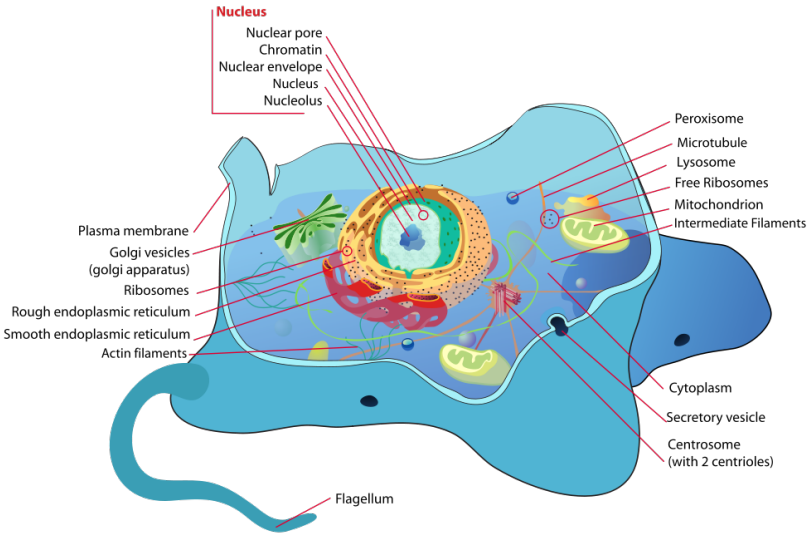


Figure 2: Sub-cellular structure of an animal cell (Source *Cell biology* on Wikipedia).

In addition to the standard information worflow where DNA is transcribed into RNA that itself is translated into proteins, information flow, there is also reverse transcription, that generates (complementary) DNA from and RNA molecule, as well as replication of

DNA (during cell division) and RNA molecules.

Why studying proteins

Proteins as the functional units in all living organisms, and they are highly dynamic. The caterpillar and the resulting butterfly have the same genome. The complement of all the expressed proteins, termed the proteome is however very different.

Figure 3: The metamorphosis from a caterpilar to a monarch butterfly. (Image from Phys.prg)

There are different modalities of the proteome that are of interest. In addition to the presence or amount of protein in a biological samples, it is also important to study the interactions between proteins forming protein-protein complexes, the presence of post-transcriptional modification (such as, for example, phosphorylations), the rate at which proteins are produced and degrated, or where the proteins reside inside a cell.

The technique of choice to study proteins in a high throughput way is *mass spectrometry*.

Setup

We are going to use the Bioconductor (Huber et al. 2015) `MSnbase` package (L. Gatto and Lilley 2012), which can be install with the `BiocManager` package, available from CRAN. If `BiocManager` isn't available on your computer, install it with:

Figure 4: The 'MSnbase' package.

```
install.packages("BiocManager")
```

Now, install `MSnbase` and its dependencies with

```
BiocManager::install("MSnbase")
```

For additional information on how to analyse mass spectrometry-based proteomics data, refer to (Gatto and Christoforou 2014) and (L. Gatto 2019), or explore the the proteomics- and mass spectrometry-related packages on the Bioconductor page

How does mass spectrometry work?

Mass spectrometry (MS) is a technology that *separates* charged molecules (ions) based on their mass to charge ratio (M/Z). It is often coupled to chromatography (liquid LC, but can also be gas-based GC). The time an analytes takes to elute from the chromatography column is the *retention time*.

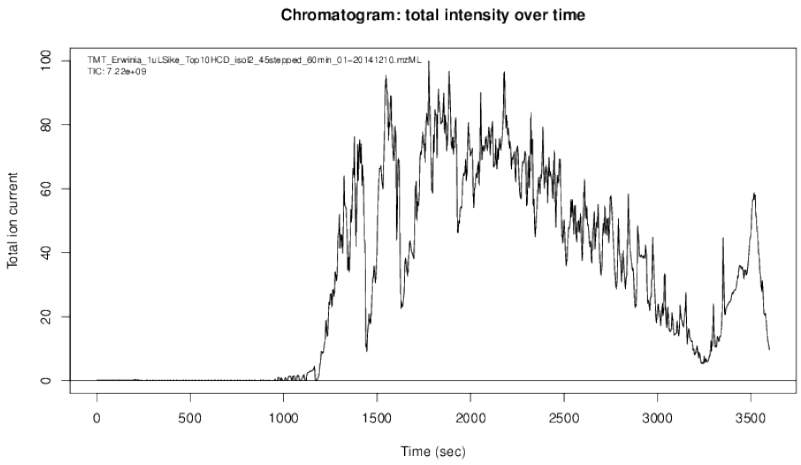


Figure 5: A chromatogram, illustrating the total amount of analytes over the retention time.

An mass spectrometer is composed of three components:

1. The *source*, that ionises the molecules: examples are Matrix-assisted laser desorption/ionisation (MALDI) or electrospray ionisation. (ESI)
2. The *analyser*, that separates the ions: Time of flight (TOF) or Orbitrap.
3. The *detector* that quantifies the ions.

When using mass spectrometry for proteomics, the proteins are first digested with a protease such as trypsin. In mass shotgun proteomics, the analytes assayed in the mass spectrometer are peptides.

Often, ions are subjected to more than a single MS round. After a first round of separation, the peaks in the spectra, called MS1 spectra, represent peptides. At this stage, the only information we possess about these peptides are their retention time and their mass-to-charge (we can also infer their charge be inspecting their isotopic envelope, i.e the peaks of the individual isotopes, see below), which is not enough to infer their identify (i.e. their sequence).

In MSMS (or MS2), the settings of the mass spectrometer are set automatically to select a certain number of MS1 peaks (for example 20). Once a narrow M/Z range has been selected (corresponding to

one high-intensity peak, a peptide, and some background noise), it is fragmented (using for example collision-induced dissociation (CID), higher energy collisional dissociation (HCD) or electron-transfer dissociation (ETD)). The fragment ions are then themselves separated in the analyser to produce a MS2 spectrum. The unique fragment ion pattern can then be used to infer the peptide sequence using de novo sequencing (when the spectrum is of high enough quality) of using a search engine such as, for example Mascot, MSGF+, . . . , that will match the observed, experimental spectrum to theoratical spectra (see details below).

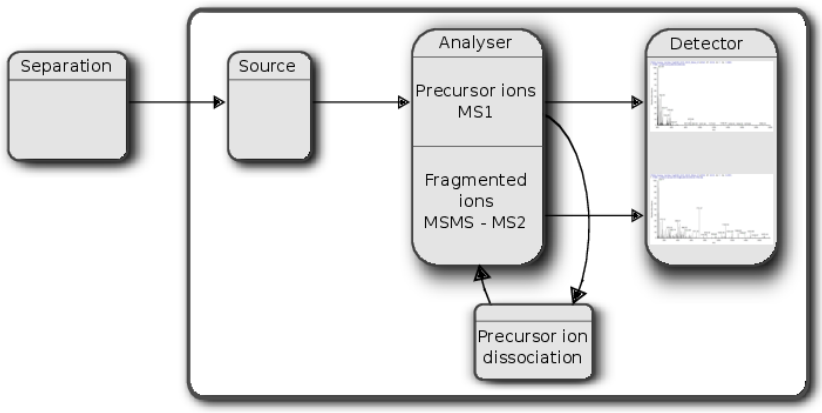


Figure 6: Schematics of a mass spectrometer and two rounds of MS.

The animation below show how 25 ions different ions (i.e. having different M/Z values) are separated throughout the MS analysis and are eventually detected (i.e. quantified). The final frame shows the hypothetical spectrum.

The figures below illustrate the two rounds of MS. The spectrum on the left is an MS1 spectrum acquired after 21 minutes and 3 seconds of elution. 10 peaks, highlited by dotted vertical lines, were selected for MS2 analysis. The peak at M/Z 460.79 (488.8) is highlighted by a red (orange) vertical line on the MS1 spectrum and the fragment spectra are shown on the MS2 spectrum on the top (bottom) right figure.

The figures below represent the 3 dimensions of MS data: a set of spectra (M/Z and intensity) of retention time, as well as the interleaved nature of MS1 and MS2 (and there could be more levels) data.

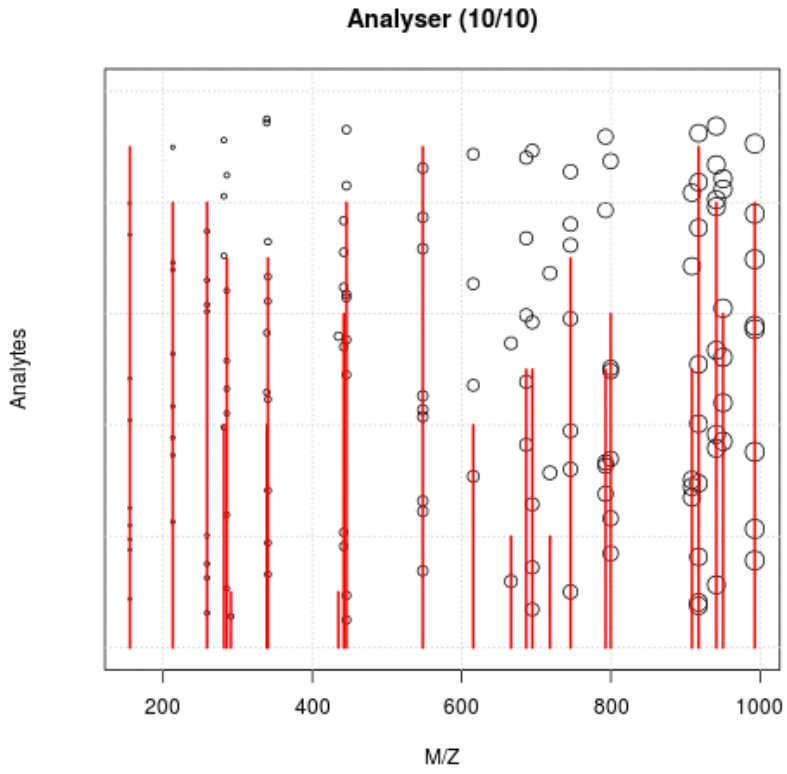


Figure 7: Separation and detection of ions in a mass spectrometer.

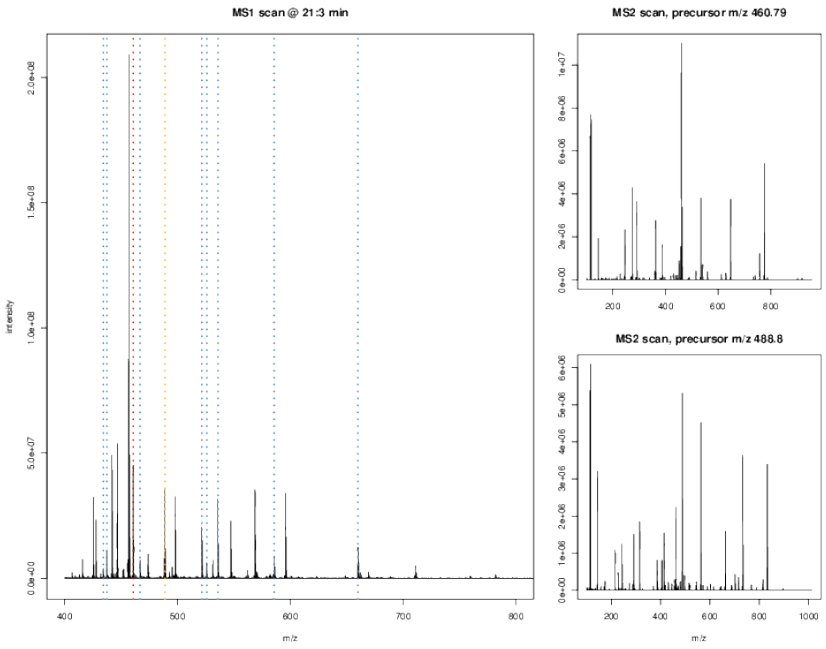


Figure 8: Parent ions in the MS1 spectrum (left) and two sected fragment ions MS2 spectra (right).

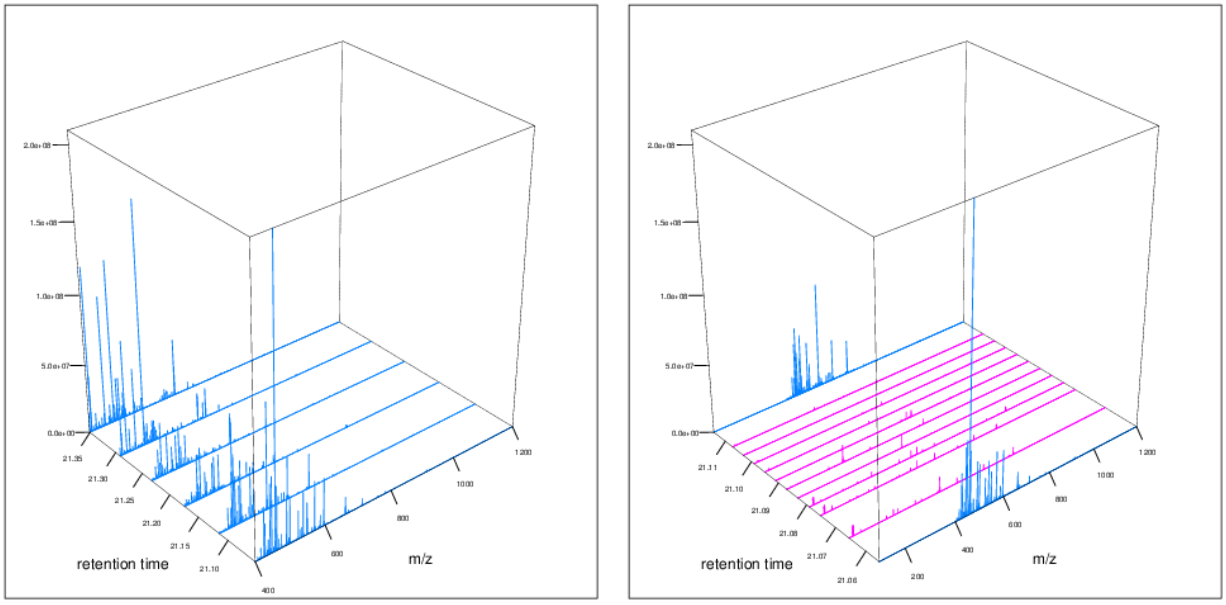


Figure 9: MS1 spectra (blue) over retention time (left). MS2 spectra (pink) interleaved between two MS1 spectra (right),

Reading and accessing MS data

Let's read a very small raw MS data file into R using the `readMSData` from the `MSnbase` package. The file that we are going to load is also available in the package.

1. Load the `MSnbase` package

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

2. Get the path to the `dummyiTRAQ.mzXML` file

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

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

3. Read it in using the `readMSData` function.

```
x <- readMSData(rawf)
x
```

```
## MSn experiment data ("MSnExp")
## Object size in memory: 0.18 Mb
## - - - Spectra data - - -
##  MS level(s): 2
```

```
##  Number of spectra: 5
##  MSn retention times: 25:1 - 25:2 minutes
## - - - Processing information - - -
## Data loaded: Sun Mar 24 22:44:45 2019
##  MSnbase version: 2.9.3
## - - - Meta data  - - -
## phenoData
##   rowNames: dummyiTRAQ.mzXML
##   varLabels: sampleNames
##   varMetadata: labelDescription
## Loaded from:
##   dummyiTRAQ.mzXML
## protocolData: none
## featureData
##   featureNames: F1.S1 F1.S2 ... F1.S5
##     (5 total)
##   fvarLabels: spectrum
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
```

The object that is returned by `readMSData` is of class `MSnExp`, that can store, access and manipulate raw MS data. Note that here we are focusing on MS-based proteomics data, but this also applied to MS-based metabolomics data.

```
class(x)
```

```
## [1] "MSnExp"
## attr(,"package")
## [1] "MSnbase"
```

4. We can find out how many spectra are available in that data using the function `length`. Full MS acquisitions would contain hundreds of thousands spectra.

```
length(x)
```

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

5. We can use various accessor function to get the MS level of these spectra, their retention time, or the M/Z and intensity of the precursor peaks of the ion corresponding to the MS2 spectra.

```
msLevel(x)
```

```
## F1.S1 F1.S2 F1.S3 F1.S4 F1.S5
##     2     2     2     2     2
```

```
rtime(x)
```

```
##   F1.S1   F1.S2   F1.S3   F1.S4   F1.S5
## 1501.35 1501.59 1501.85 1502.07 1502.31
```

```
precursorMz(x)
```

```
##    F1.S1    F1.S2    F1.S3    F1.S4    F1.S5
## 645.3741 546.9586 645.3741 716.3405 437.8040
```

```
precursorIntensity(x)
```

```
##    F1.S1    F1.S2    F1.S3    F1.S4    F1.S5
## 47659400 26356100 23432400 24854800  7052960
```

6. We can also extract individual spectra using `[[` and plot them.

```
x[[3]]
```

```
## Object of class "Spectrum2"
##  Precursor: 645.3741
##  Retention time: 25:2
##  Charge: 2
##  MSn level: 2
##  Peaks count: 2125
##  Total ion count: 150838188
```

```
plot(x[[3]])
```

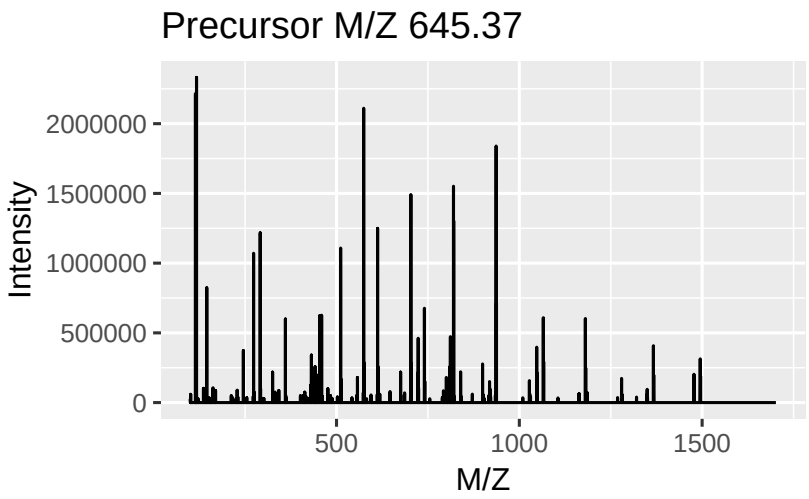


Figure 10: Visualisation of the 3rd MS spectrum in our small test data set.

Exercise

For the rest of this tutorial, we will be using a slightly larger dataset (still tiny compared to full acquisitions) that is distributed with the `MSnbase` package. Load it as shown below and compute the number of spectra available in that dataset, their MS level, and the retention time range over which these spectra have been acquired.

```
data(itraqdata)
```

```
length(itraqdata)
```

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

```
unique(msLevel(itraqdata))
```

```
## [1] 2
```

```
formatRt(range(rtime(itraqdata)))
```

```
## [1] "19:9"  "50:18"
```

This object also contains additional metadata for each spectrum, that can be accessed, as a `data.frame`, with `fData`.

Identification

The raw data is still a long way of obtaining biologically relevant proteomics data. The first step to obtain proteomics data is to identify the peptides that have been acquired in the MS. Peptide identification work by comparing expected and observed spectra. As shown below, when a precursor peptide ion is fragmented in a CID cell, it breaks at specific bonds, producing sets of peaks (a, b, c and x, y, z) that can be predicted.

It is thus possible to calculate the expected set of fagment peaks for a given peptide, such as *SIGFEGDSIGR* below.

```
calculateFragments("SIGFEGDSIGR")
```

```
##            mz  ion type pos z        seq
## 1    88.03931   b1    b   1 1          S
## 2   201.12337   b2    b   2 1         SI
## 3   258.14483   b3    b   3 1        SIG
## 4   405.21324   b4    b   4 1       SIGF
## 5   534.25583   b5    b   5 1      SIGFE
## 6   591.27729   b6    b   6 1     SIGFEG
## 7   706.30423   b7    b   7 1    SIGFEGD
## 8   793.33626   b8    b   8 1   SIGFEGDS
```

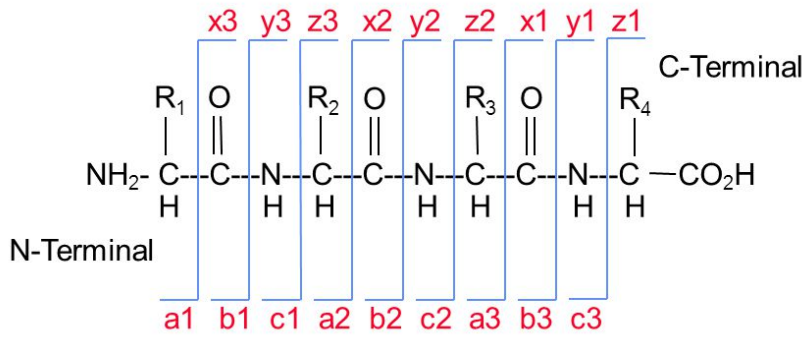


Figure 11: Peptide fragmentation.

Biemann, K *Methods Enzymol* (1990) **193** 886-887

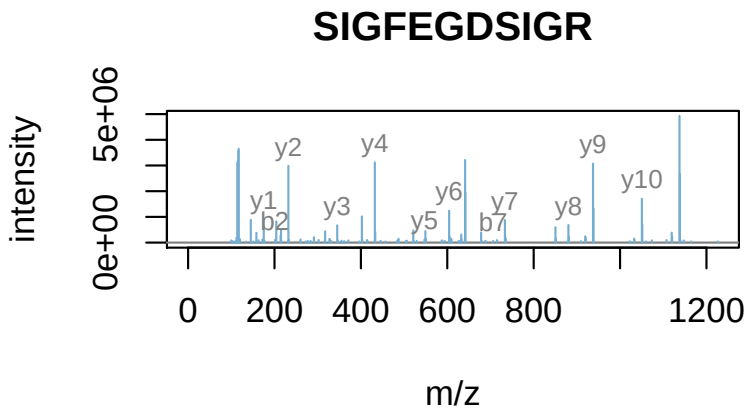
```
## 9   906.42032   b9    b   9 1  SIGFEGDSI
## 10  963.44178  b10    b  10 1 SIGFEGDSIG
## 11  175.11895   y1    y   1 1          R
## 12  232.14041   y2    y   2 1         GR
## 13  345.22447   y3    y   3 1        IGR
## 14  432.25650   y4    y   4 1       SIGR
## 15  547.28344   y5    y   5 1      DSIGR
## 16  604.30490   y6    y   6 1     GDSIGR
## 17  733.34749   y7    y   7 1    EGDSIGR
## 18  880.41590   y8    y   8 1   FEGDSIGR
## 19  937.43736   y9    y   9 1  GFEGDSIGR
## 20 1050.52142  y10    y  10 1 IGFEGDSIGR
## 21  873.42266  b9_   b_   9 1  SIGFEGDSI
## 22  930.44412 b10_   b_  10 1 SIGFEGDSIG
## 23  514.28579  y5_   y_   5 1      DSIGR
## 24  571.30725  y6_   y_   6 1     GDSIGR
## 25  700.34984  y7_   y_   7 1    EGDSIGR
## 26  847.41825  y8_   y_   8 1   FEGDSIGR
## 27  904.43971  y9_   y_   9 1  GFEGDSIGR
## 28 1017.52377 y10_   y_  10 1 IGFEGDSIGR
## 29  142.12130  y1_   y_   1 1          R
## 30  199.14276  y2_   y_   2 1         GR
## 31  312.22682  y3_   y_   3 1        IGR
## 32  399.25885  y4_   y_   4 1       SIGR
```

The last step is to compare obseved and expected peaks. If there is a good match, the MS2 spectrum is assigned the peptide sequence.

```
itraqdata2 <- pickPeaks(itraqdata, verbose = FALSE)
```

```
s <- "SIGFEGDSIGR"
plot(itraqdata2[[14]], s, main = s)
```

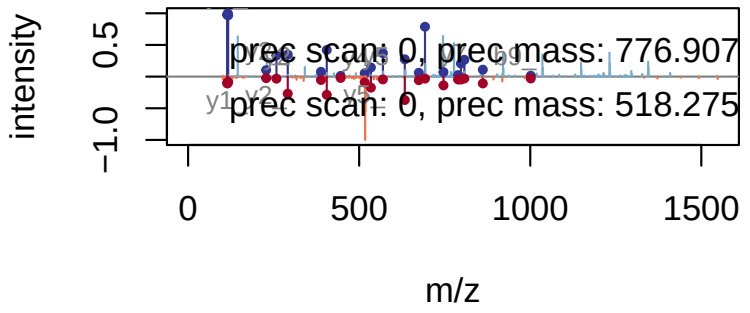
Figure 12: Matching observed and expected peaks.



It is also possible to plot 2 spectra to compare them directly.

```
plot(itraqdata2[[25]], itraqdata2[[28]], sequences = rep("IMIDLDGTENK",
    2))
```

Figure 13: Direct comparison of 2 MS2 spectra.



In a full experiment, all possible peptides from the known (or relevant) proteome of interest (such as databases that can be downloaded from the UniProt site[1]) are compared to the millions of observed

[1] The Universal Protein Resource (UniProt) is a freely and accessible comprehensive resource for protein sequence and annotation data.

spectra.

From the list of identified petides, it is then necessary to infer the most propable proteins that were present in the biological sample.

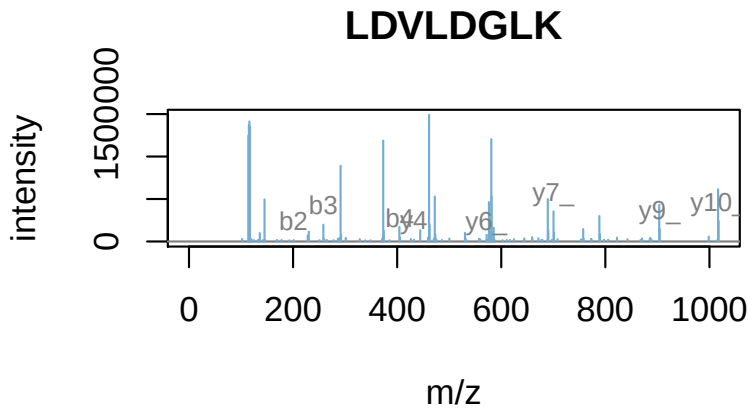
Exercise

Plot the 44th spectrum of the `itraqdata2` experiment. The sequence can be accessed in the feature metadata with

```
fData(itraqdata2)$PeptideSequence[[44]]
```

```
## [1] LDVLDGLK
## 47 Levels: AADALLLK ... VWVVEGSK
```

```
plot(itraqdata2[[44]], s, main = fData(itraqdata2)$PeptideSequence[[44]])
```



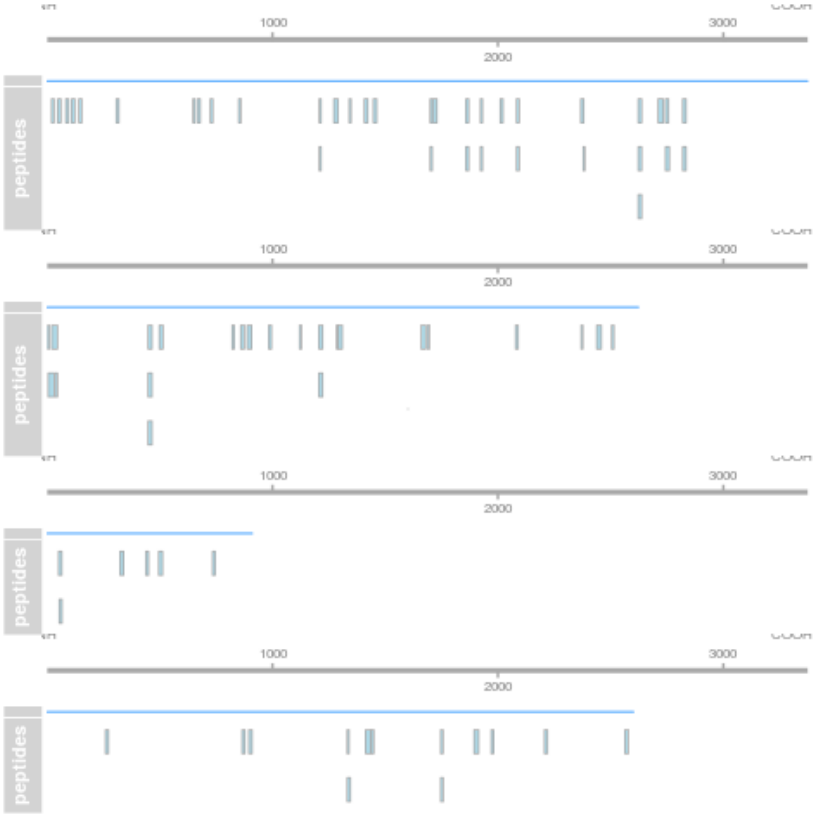
Quantitation

The last step of MS data processing is to quantify peptide abundances in the biological samples. The table below summarises the different possibilites depending whether the proteins or peptides are labelled, and whether the quantitation is performed in MS1 or MS2.

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

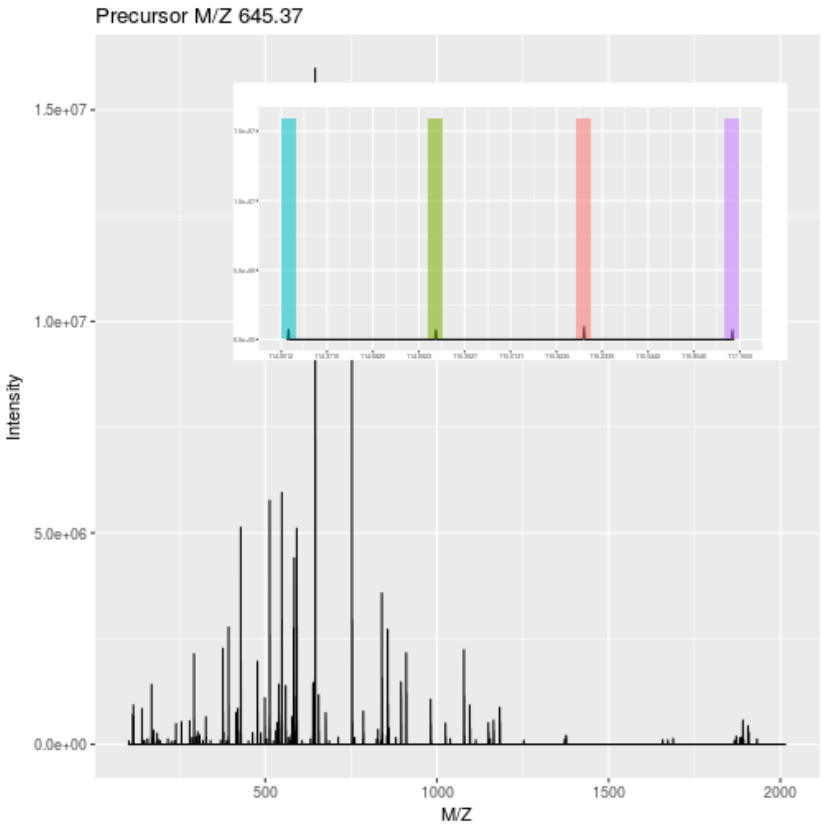
Label-free MS2: Spectral counting

In spectral counting, on simply counts the number of quantified peptides that are assigned to a protein.



Labelled MS2: Isobaric tagging

Isobaric tagging refers to the labelling using isobaric tags, i.e. chemical tags that have the same mass and hence can't be distinguish by the spectrometer. The peptides of different samples (4, 6, 10 or 11) are labelled with different tags and combined prior to mass spectrometry acquisition. Given that they are isobaric, all identical peptides, irrespective of the tag and this the sample of origin, are co-analysed, up to fragmentation prior to MS2 analysis. During fragmentation, the isobaric tags fall of, fragment themselves, and result in a set of sample specific peaks. These specific peaks can be used to infer sample-specific quantitation, while the rest of the MS2 spectrum is used for identification.



Label-free MS1: extracted ion chromatograms

In label-free quantitation, the precursor peaks that match an identified peptide are integrated of retention time and the area under that *extracted ion chromatogram* is used to quantify that peptide in that sample.

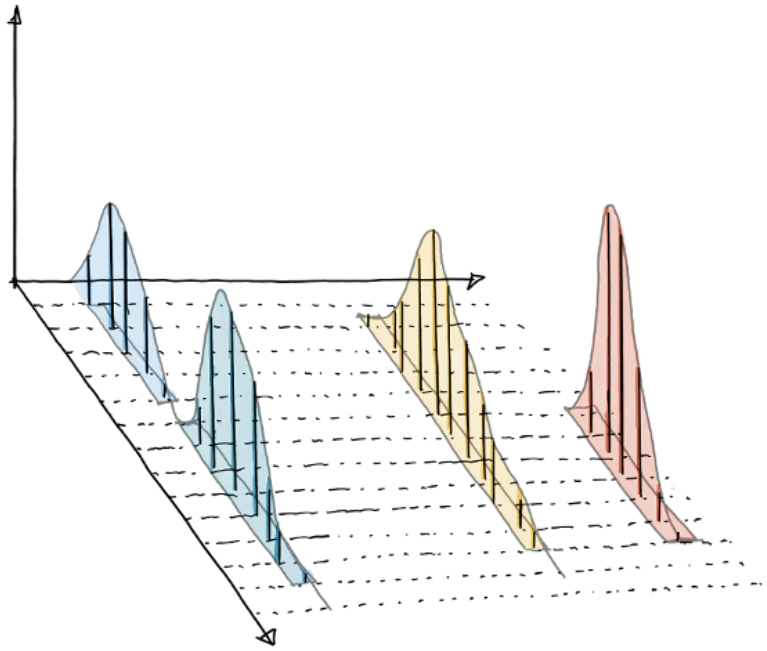


Figure: credit Johannes Rainer.

Labelled MS1: SILAC

In SILAc quantitation, sample are grown in a medium that contains heavy amino acids (typically arginine and lysine). All proteins gown in this *heavy* growth medium contain the heavy form of these amino acids. Two samples, one grown in heavy medium, and one grown in normal (light) medium are then combined and analysed together. The heavy peptides precursor peaks are systematically shifted compared to the light ones, and the ratio between the height of a heavy and light peaks can be used to calculate peptide and protein fold-changes.

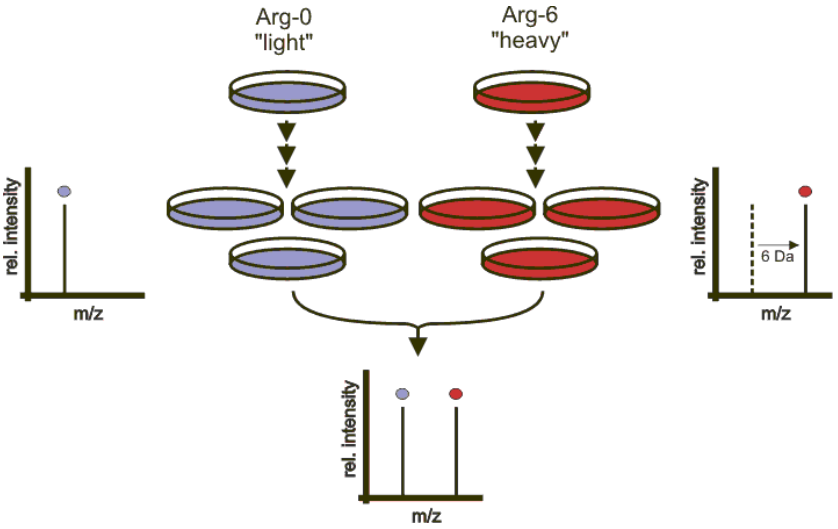


Figure: credit Wikimedia Commons.

Exercise

As its name implies, the `itraqdata` is an iTRAQ-based isobar quantitation experiment. We can visualise the reporter peaks as follows:

```
plot(itraqdata[[14]], reporters = iTRAQ4, full = TRUE)
```

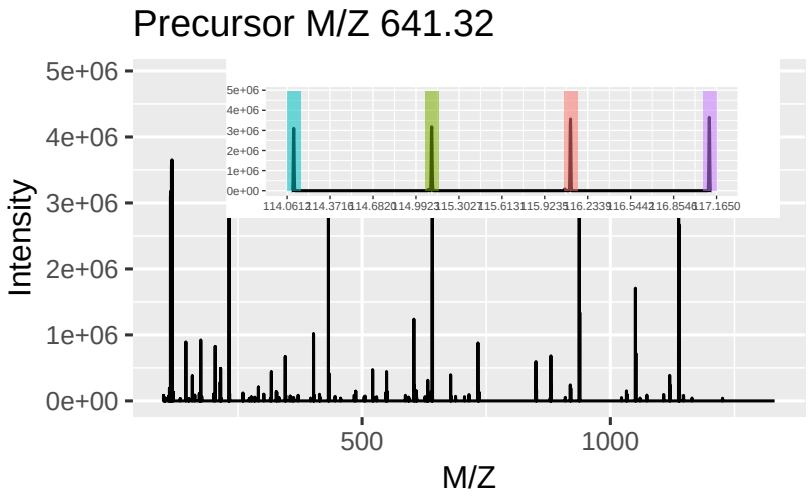


Figure 14: Visualisation of the iTRAQ reporter peaks.

We can quantify these four peaks with the `quantify` method, to produce and object of class `MSnSet` containing quantitation data. The quantitation values can be accessed with `exprs`. This data also contains feature metadata that can be accessed with the `fData` function.

```
msnset <- quantify(itraqdata, method = "trap",
    reporters = iTRAQ4)
msnset
```

```
## 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]
## iTRAQ4 quantification by trapezoidation: Sun Mar 24 22:44:47 2019
##  MSnbase version: 1.1.22
```

```
head(exprs(msnset))
```

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

```
head(fData(msnset))
```

```
##     spectrum ProteinAccession
## X1         1              BSA
## X10       10          ECA1422
## X11       11          ECA4030
## X12       12          ECA3882
## X13       13          ECA1364
## X14       14          ECA0871
##                           ProteinDescription
## X1                      bovine serum albumin
## X10 glucose-1-phosphate cytidylyltransferase
## X11         50S ribosomal subunit protein L4
## X12                   chaperone protein DnaK
## X13      succinyl-CoA synthetase alpha chain
## X14              NADP-dependent malic enzyme
##     PeptideSequence fileIdx retention.time
## X1           NYQEAK       1        1149.31
## X10 VTLVDTGEHSMTGGR       1        1503.03
## X11           SPIWR       1        1663.61
## X12        TAIDDALK       1        1663.86
## X13          SILINK       1        1664.08
## X14    DFEVVNNESDPR       1        1664.32
##     precursor.mz precursor.intensity charge
## X1      520.7833             3449020      2
## X10     573.9539             7849420      3
## X11     401.7392            41253600      2
## X12     567.8339            23549500      2
## X13     488.3269            13025200      2
## X14     782.8715            18405000      2
##     peaks.count       tic  ionCount ms.level
## X1         1922  26413754  26413754        2
## X10        1376  24482281  24482281        2
## X11        1571 231075934 231075934        2
## X12        2397 247323187 247323187        2
## X13        2574 207247502 207247502        2
## X14        1829 115317275 115317275        2
##     acquisition.number collision.energy
## X1                   2               40
## X10                 11               40
## X11                 12               40
## X12                 13               40
## X13                 14               40
## X14                 15               40
```

Quantitative data processing

In our examples, we not have processing data for the 55 peptides and 4 samples. In this data, there is only 1 missing value, corresponding to an absent reporter peak. We are going to simply drop that feature.

```
table(is.na(exprs(msnset)))
```

```
##
## FALSE  TRUE
##   219     1
```

```
msnset <- filterNA(msnset)
```

In MS1 label-free experiments, given that each sample is acquired independently, the proportion of missing values can be as high several tens of percent. In such situations, removing rows with missing values isn't possible at all. Imputation is possible, albeit tricky, as different mechanisms can be responsible for missing value that appear either at random or not at random (Lazar et al. 2016).

Next, we aggregate the spectrum-level quantitation values into protein-level data using the median and the `combineFeatures` function:

```
prots <- combineFeatures(msnset, fcol = "ProteinAccession",
    method = "median")
head(exprs(prots))
```

```
##         iTRAQ4.114 iTRAQ4.115 iTRAQ4.116
## BSA       1347.616   2247.310   3927.693
## ECA0172  17593.548  18545.620  19361.837
## ECA0435   4923.628   5557.818   5775.203
## ECA0452   1524.148   1399.897   1547.218
## ECA0469   1069.945   1035.689   1029.420
## ECA0621   1101.062   1124.167   1140.093
##         iTRAQ4.117
## BSA      7661.1463
## ECA0172 18328.2365
## ECA0435  5079.2952
## ECA0452  1563.2299
## ECA0469   999.6957
## ECA0621  1191.8055
```

Following on from here, many data processing such as normalisation, non-specific filtering, and hypothesis testing is very similar to other omics data.

Applications in statistical learning

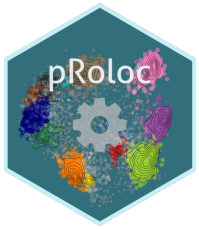


Figure 15: The 'pRoloc' package.

In this section, I illustrate a use case of mass spectrometry-based proteomics to infer sub-cellular protein localisation and the application of statistical learning. This section requires the `pRoloc` and `pRolocdata` packages (Gatto et al. 2014; Breckels et al. 2016). Both packages can be installed with the `BiocManager::install` function, as illustrated above.

```
library("pRoloc")
library("pRolocdata")
```

The `hyperLOPIT2015` data contains localisation data for 5032 proteins in mouse embryonic stem cells (Christoforou et al. 2016). The protein information is collected along a 20 fraction density gradient - our data matrix has dimensions 5032 rows and 20 columns. We use PCA to easily visualise it in 2 dimensions.

```
data(hyperLOPIT2015)
## set colours
setStockcol(paste0(getStockcol(), 80))
## produce PCA plot
plot2D(hyperLOPIT2015)
```

Each dot on the PCA plot represents a single protein. The protein is either of unknown localisation, and represented by a grey circle, or is of known localisation (and called an organelle marker) and is coloured according to its expected sub-cellular localisation.

In the next figure, we have trained an support vector machine (SVM) model and classified proteins of unknown localisation using an SVM model. The size of the dots are scaled according to the classifier score and only assignments corresponding to a false discovery rate of 5% have been considered.

```
sz <- exp(fData(hyperLOPIT2015)$svm.score) - 1
plot2D(hyperLOPIT2015, fcol = "final.assignment",
    cex = sz)
```

Session information

```
## R version 3.5.3 Patched (2019-03-11 r76221)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.2 LTS
##
## Matrix products: default
```

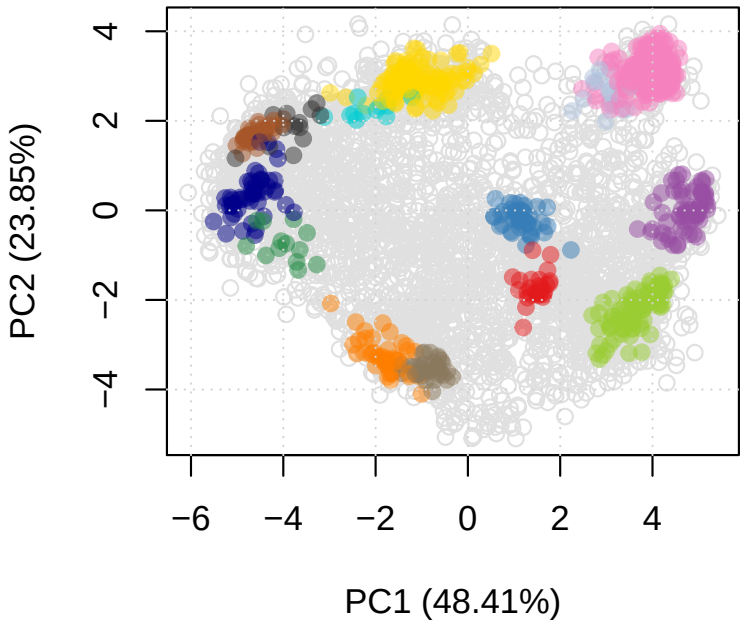


Figure 16: Mouse stem cell spatial proteomics data from Christoforou et al.

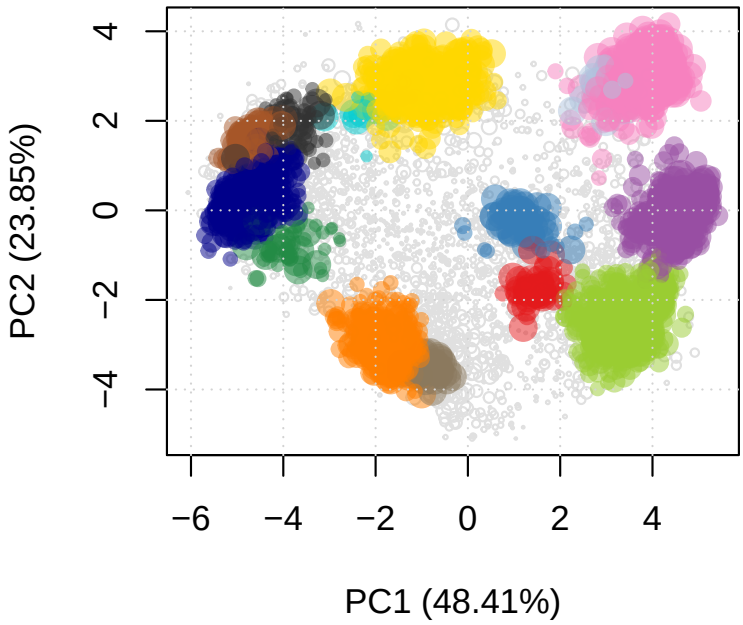


Figure 17: New sub-cellular assignment after using support vector machine classifier.

```
## BLAS: /usr/lib/x86_64-linux-gnu/libf77blas.so.3.10.3
## LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8
##  [2] LC_NUMERIC=C
##  [3] LC_TIME=fr_FR.UTF-8
##  [4] LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=fr_FR.UTF-8
##  [6] LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=fr_FR.UTF-8
##  [8] LC_NAME=C
##  [9] LC_ADDRESS=C
## [10] LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fr_FR.UTF-8
## [12] LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics
## [5] grDevices utils     datasets  methods
## [9] base
##
## other attached packages:
##  [1] pRolocdata_1.20.0
##  [2] pRoloc_1.22.2
##  [3] BiocParallel_1.16.6
##  [4] MLInterfaces_1.62.0
##  [5] cluster_2.0.7-1
##  [6] annotate_1.60.1
##  [7] XML_3.98-1.19
##  [8] AnnotationDbi_1.44.0
##  [9] IRanges_2.16.0
## [10] MSnbase_2.9.3
## [11] ProtGenerics_1.14.0
## [12] S4Vectors_0.20.1
## [13] mzR_2.16.2
## [14] Rcpp_1.0.1
## [15] Biobase_2.42.0
## [16] BiocGenerics_0.28.0
##
## loaded via a namespace (and not attached):
##   [1] plyr_1.8.4
##   [2] igraph_1.2.4
##   [3] lazyeval_0.2.2
```

```
##   [4] splines_3.5.3
##   [5] ggvis_0.4.4
##   [6] crosstalk_1.0.0
##   [7] ggplot2_3.1.0
##   [8] digest_0.6.18
##   [9] foreach_1.4.4
##  [10] htmltools_0.3.6
##  [11] viridis_0.5.1
##  [12] gdata_2.18.0
##  [13] magrittr_1.5
##  [14] memoise_1.1.0
##  [15] doParallel_1.0.14
##  [16] mixtools_1.1.0
##  [17] sfsmisc_1.1-3
##  [18] limma_3.38.3
##  [19] recipes_0.1.5
##  [20] gower_0.2.0
##  [21] rda_1.0.2-2.1
##  [22] lpSolve_5.6.13
##  [23] prettyunits_1.0.2
##  [24] colorspace_1.4-1
##  [25] blob_1.1.1
##  [26] xfun_0.5
##  [27] dplyr_0.8.0.1
##  [28] crayon_1.3.4
##  [29] RCurl_1.95-4.12
##  [30] hexbin_1.27.2
##  [31] genefilter_1.64.0
##  [32] impute_1.56.0
##  [33] survival_2.43-3
##  [34] iterators_1.0.10
##  [35] glue_1.3.1
##  [36] gtable_0.2.0
##  [37] ipred_0.9-8
##  [38] zlibbioc_1.28.0
##  [39] kernlab_0.9-27
##  [40] Rhdf5lib_1.4.2
##  [41] prabclus_2.2-7
##  [42] DEoptimR_1.0-8
##  [43] scales_1.0.0
##  [44] vsn_3.50.0
##  [45] mvtnorm_1.0-10
##  [46] DBI_1.0.0
##  [47] viridisLite_0.3.0
```

```
##  [48] xtable_1.8-3
##  [49] progress_1.2.0
##  [50] bit_1.1-14
##  [51] proxy_0.4-23
##  [52] mclust_5.4.3
##  [53] preprocessCore_1.44.0
##  [54] lava_1.6.5
##  [55] prodlim_2018.04.18
##  [56] sampling_2.8
##  [57] htmlwidgets_1.3
##  [58] httr_1.4.0
##  [59] threejs_0.3.1
##  [60] FNN_1.1.3
##  [61] RColorBrewer_1.1-2
##  [62] fpc_2.1-11.1
##  [63] modeltools_0.2-22
##  [64] pkgconfig_2.0.2
##  [65] flexmix_2.3-15
##  [66] nnet_7.3-12
##  [67] caret_6.0-81
##  [68] tidyselect_0.2.5
##  [69] labeling_0.3
##  [70] rlang_0.3.2
##  [71] reshape2_1.4.3
##  [72] later_0.8.0
##  [73] munsell_0.5.0
##  [74] mlbench_2.1-1
##  [75] tools_3.5.3
##  [76] LaplacesDemon_16.1.1
##  [77] generics_0.0.2
##  [78] RSQLite_2.1.1
##  [79] pls_2.7-0
##  [80] evaluate_0.13
##  [81] stringr_1.4.0
##  [82] mzID_1.20.1
##  [83] yaml_2.2.0
##  [84] tufte_0.4
##  [85] ModelMetrics_1.2.2
##  [86] knitr_1.22
##  [87] bit64_0.9-7
##  [88] robustbase_0.93-4
##  [89] randomForest_4.6-14
##  [90] purrr_0.3.2
##  [91] dendextend_1.10.0
```

```
##  [92] ncdf4_1.16.1
##  [93] nlme_3.1-137
##  [94] whisker_0.3-2
##  [95] mime_0.6
##  [96] formatR_1.6
##  [97] biomaRt_2.38.0
##  [98] compiler_3.5.3
##  [99] e1071_1.7-1
## [100] affyio_1.52.0
## [101] tibble_2.1.1
## [102] stringi_1.4.3
## [103] lattice_0.20-38
## [104] trimcluster_0.1-2.1
## [105] Matrix_1.2-15
## [106] gbm_2.1.5
## [107] pillar_1.3.1
## [108] BiocManager_1.30.4
## [109] MALDIquant_1.19.2
## [110] data.table_1.12.0
## [111] bitops_1.0-6
## [112] httpuv_1.5.0
## [113] R6_2.4.0
## [114] pcaMethods_1.74.0
## [115] affy_1.60.0
## [116] hwriter_1.3.2
## [117] promises_1.0.1
## [118] gridExtra_2.3
## [119] codetools_0.2-16
## [120] MASS_7.3-51.1
## [121] gtools_3.8.1
## [122] assertthat_0.2.1
## [123] rhdf5_2.26.2
## [124] withr_2.1.2
## [125] diptest_0.75-7
## [126] hms_0.4.2
## [127] timeDate_3043.102
## [128] grid_3.5.3
## [129] rpart_4.1-13
## [130] coda_0.19-2
## [131] class_7.3-15
## [132] rmarkdown_1.12
## [133] segmented_0.5-3.0
## [134] lubridate_1.7.4
## [135] shiny_1.2.0
```

```
## [136] base64enc_0.1-3
```

Breckels, L M, C M Mulvey, K S Lilley, and L Gatto. 2016. "A Bioconductor Workflow for Processing and Analysing Spatial Proteomics Data." *F1000Res* 5: 2926. doi:10.12688/f1000research.10411.2.

Christoforou, A, C M Mulvey, L M Breckels, A Geladaki, T Hurrell, P C Hayward, T Naake, et al. 2016. "A Draft Map of the Mouse Pluripotent Stem Cell Spatial Proteome." *Nat Commun* 7 (January): 8992. doi:10.1038/ncomms9992.

Gatto, L, and A Christoforou. 2014. "Using R and Bioconductor for Proteomics Data Analysis." *Biochim. Biophys. Acta* 1844 (1 Pt A): 42–51.

Gatto, L, L M Breckels, S Wieczorek, T Burger, and K S Lilley. 2014. "Mass-Spectrometry-Based Spatial Proteomics Data Analysis Using pRoloc and pRolocdata." *Bioinformatics* 30 (9): 1322–4. doi:10.1093/bioinformatics/btu013.

Gatto, Laurent. 2019. *Bioconductor Tools for Mass Spectrometry and Proteomics*. `https://rawgit.com/lgatto/bioc-ms-prot/master/lab.html`.

Gatto, Laurent, and Kathryn S Lilley. 2012. "MSnbase-an R/Bioconductor Package for Isobaric Tagged Mass Spectrometry Data Visualization, Processing and Quantitation." *Bioinformatics* 28 (2): 288–89.

Huber, W, V J Carey, R Gentleman, S Anders, M Carlson, B S Carvalho, H C Bravo, et al. 2015. "Orchestrating High-Throughput Genomic Analysis with Bioconductor." *Nat. Methods* 12 (2): 115–21.

Lazar, C, L Gatto, M Ferro, C Bruley, and T Burger. 2016. "Accounting for the Multiple Natures of Missing Values in Label-Free Quantitative Proteomics Data Sets to Compare Imputation Strategies." *J Proteome Res* 15 (4): 1116–25. doi:10.1021/acs.jproteome.5b00981.