Mass spectrometry-based proteomics

Laurent Gatto

2019-03-23

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.

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.

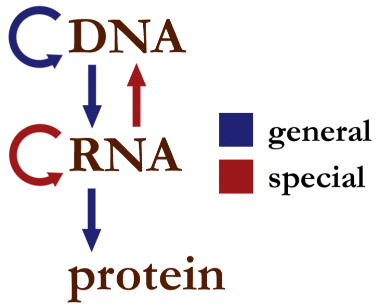

Figure 1: Information flow in biological systems (Source *Central dogma of biology* on Wikipedia).

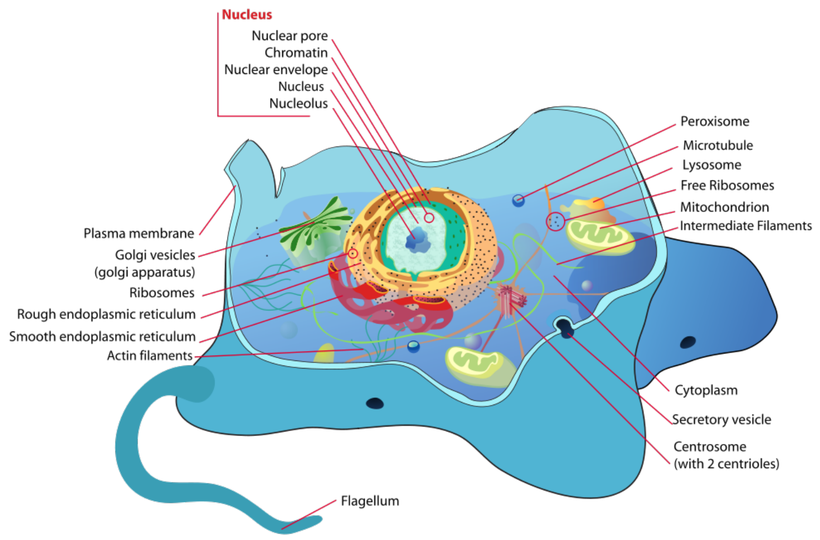

Figure 2: Sub-cellular structure of an animal cell (Source *Cell biology* on Wikipedia).

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

Figure 4: The 'MSnbase' package.

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:

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

Now, install `MSnbase` and its dependencies with

```r
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*.

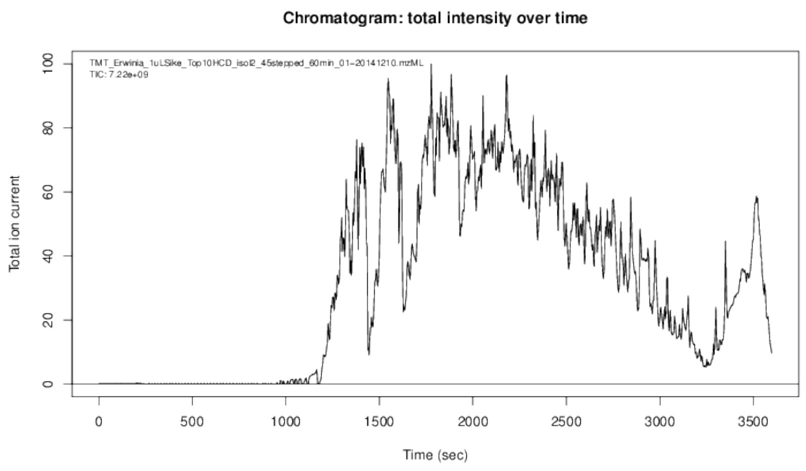

Figure 5: A chromatogram, illustrating the total amount of analytes over the retention time.

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).

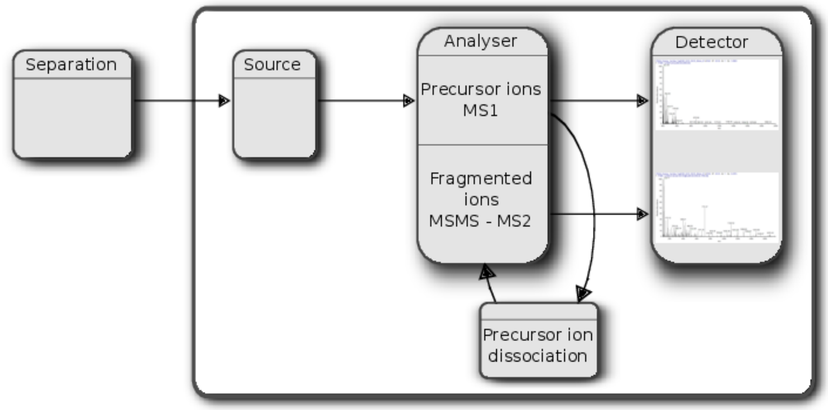

Figure 6: Schematics of a mass spectrometer and two rounds of MS.

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.

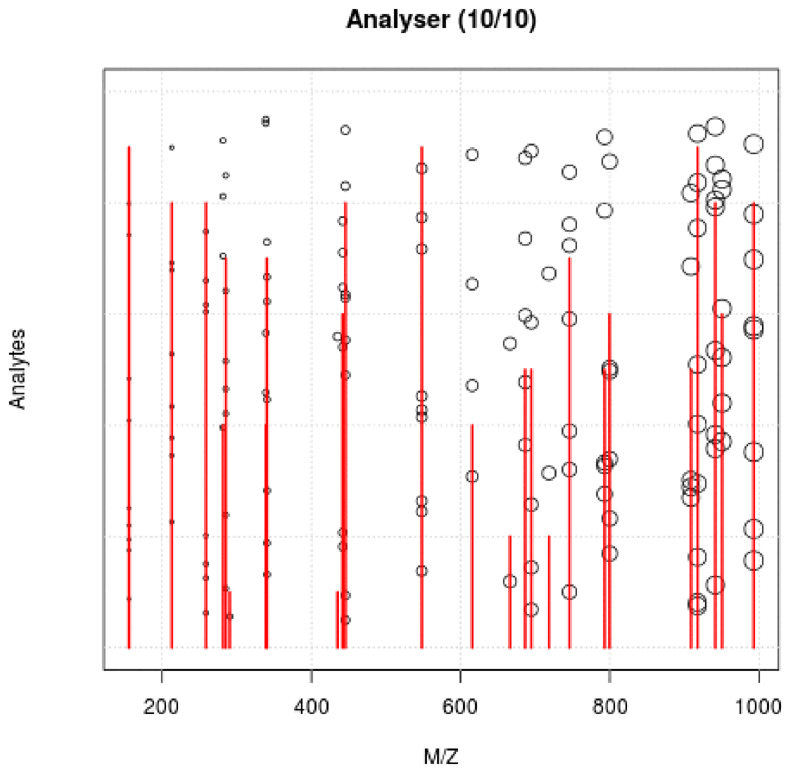

Figure 7: Separation and detection of ions in a mass spectrometer.

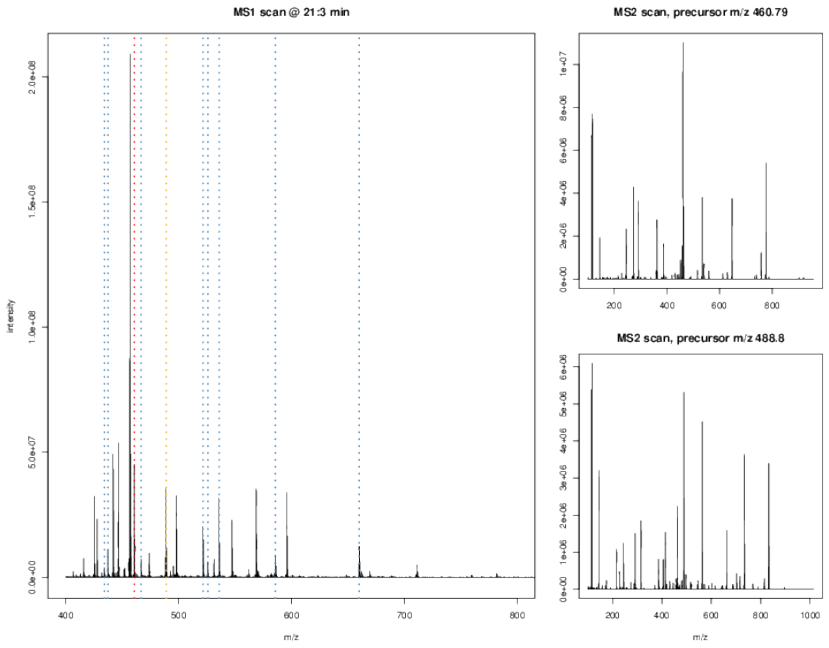

Figure 8: Parent ions in the MS1 spectrum (left) and two sected fragment ions MS2 spectra (right).

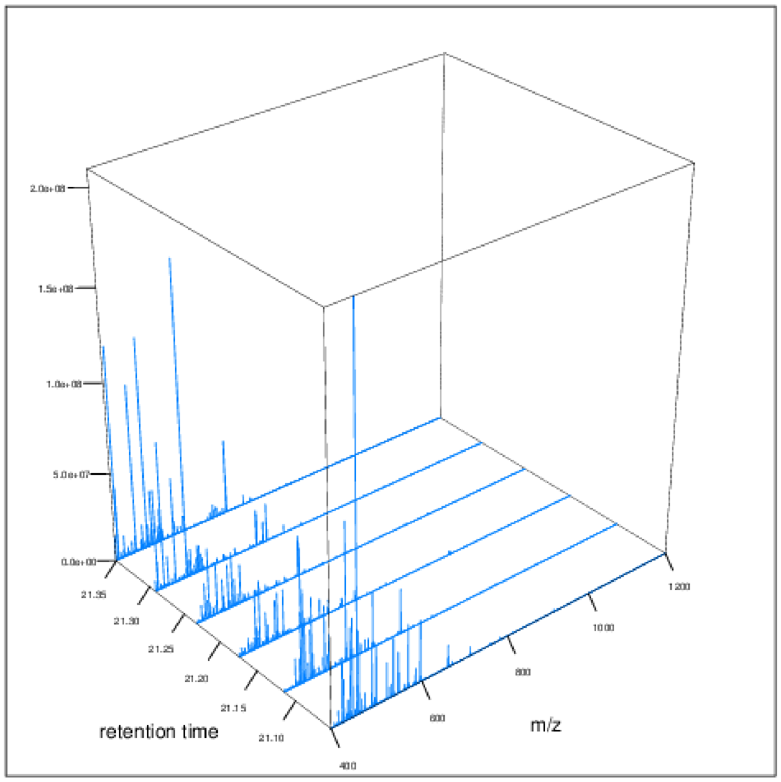

Figure 9: MS1 spectra over retention time.

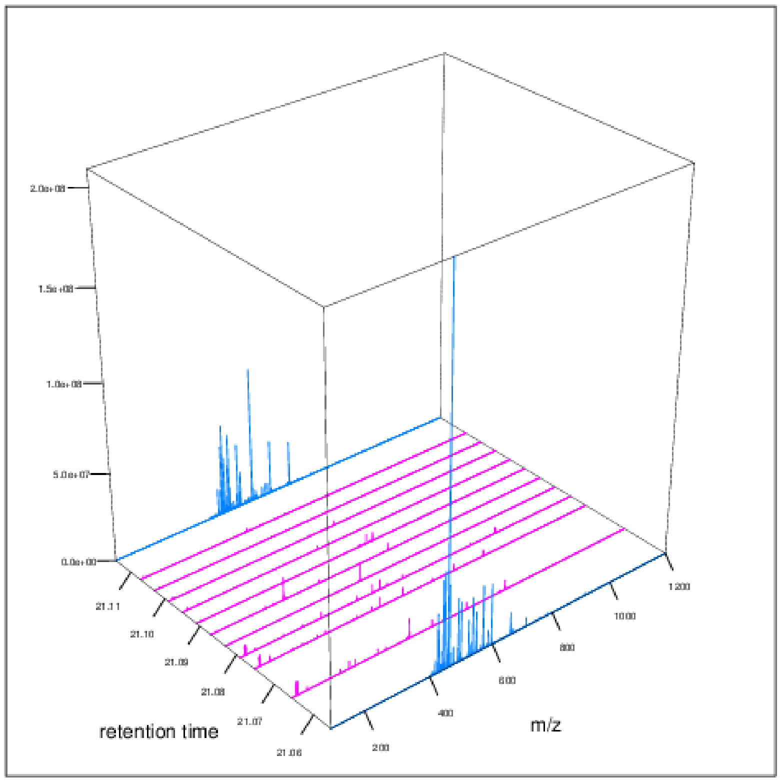

Figure 10: MS2 spectra interleaved between two MS1 spectra.

Practical: 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: Sat Mar 23 16:05:54 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]])
```

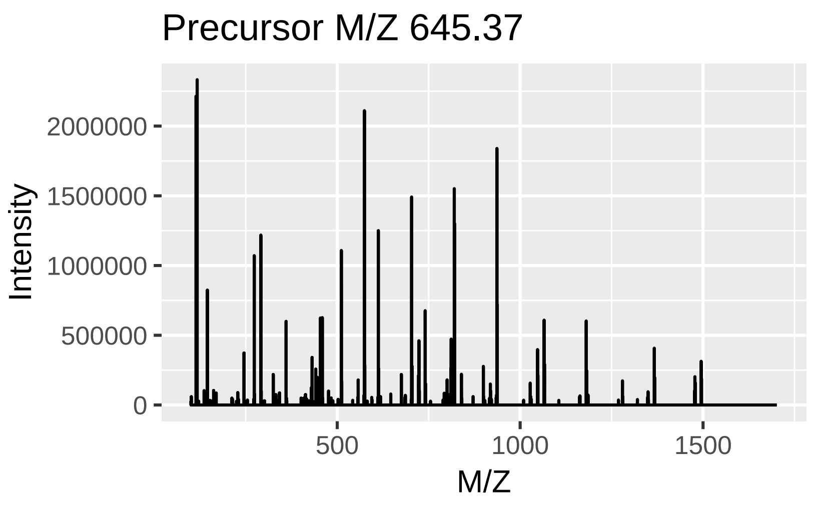

Figure 11: Visualisation of the 3rd MS spectrum in our small test data set.

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"
```

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.

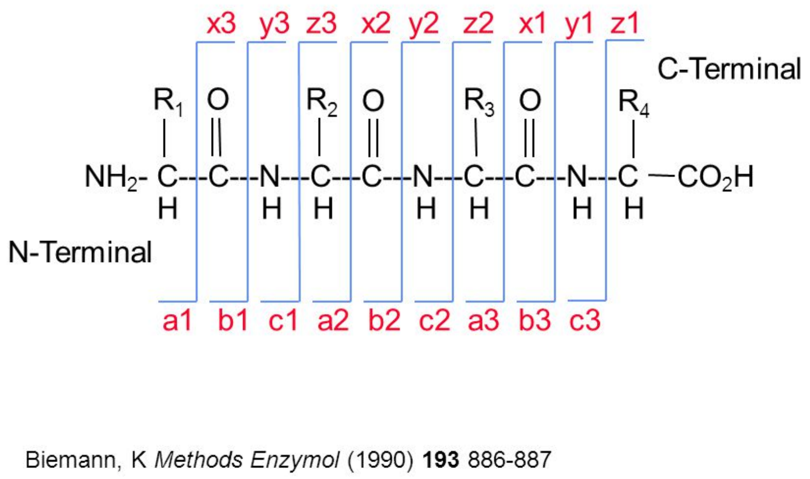
x3 y3 z3 x2 y2 z2 x1 y1 z1
C-Terminal
R_1
O
R_2
O
R_3
O
R_4
NH_2- C---C---N---C---C---N---C---C---N---C —CO_2H
H H H H H H H
N-Terminal
a1 b1 c1 a2 b2 c2 a3 b3 c3
Biemann, K *Methods Enzymol* (1990) **193** 886-887

Figure 12: Peptide fragmentation.

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
## 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)
```

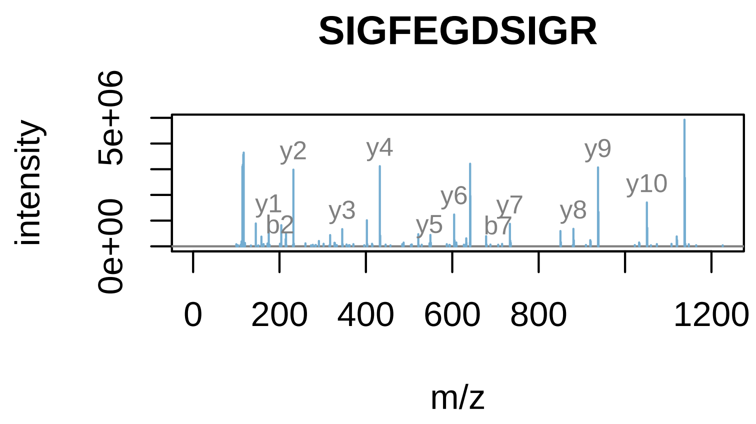

Figure 13: Matching observed and expected peaks.

In a full experiment, all possible peptides from the known (or relevant) proteome of interest are compared to the millions of observed spectra.

Exercise

Quantitation

Quantitative data processing

Applications in statistical learning: hypthesis testing, classification, clustering

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
## 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] MSnbase_2.9.3       ProtGenerics_1.14.0
## [3] S4Vectors_0.20.1    mzR_2.16.2
## [5] Rcpp_1.0.1          Biobase_2.42.0
## [7] BiocGenerics_0.28.0
##
```

```
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5
##  [2] xfun_0.5
##  [3] purrr_0.3.2
##  [4] lattice_0.20-38
##  [5] rhdf5_2.26.2
##  [6] colorspace_1.4-1
##  [7] htmltools_0.3.6
##  [8] yaml_2.2.0
##  [9] vsn_3.50.0
## [10] XML_3.98-1.19
## [11] rlang_0.3.2
## [12] pillar_1.3.1
## [13] glue_1.3.1
## [14] BiocParallel_1.16.6
## [15] affy_1.60.0
## [16] foreach_1.4.4
## [17] affyio_1.52.0
## [18] plyr_1.8.4
## [19] mzID_1.20.1
## [20] stringr_1.4.0
## [21] zlibbioc_1.28.0
## [22] munsell_0.5.0
## [23] pcaMethods_1.74.0
## [24] gtable_0.2.0
## [25] codetools_0.2-16
## [26] evaluate_0.13
## [27] labeling_0.3
## [28] knitr_1.22
## [29] IRanges_2.16.0
## [30] doParallel_1.0.14
## [31] preprocessCore_1.44.0
## [32] tufte_0.4
## [33] scales_1.0.0
## [34] formatR_1.6
## [35] BiocManager_1.30.4
## [36] limma_3.38.3
## [37] impute_1.56.0
## [38] ggplot2_3.1.0
## [39] digest_0.6.18
## [40] stringi_1.4.3
## [41] dplyr_0.8.0.1
## [42] ncdf4_1.16.1
## [43] grid_3.5.3
## [44] tools_3.5.3
## [45] magrittr_1.5
## [46] lazyeval_0.2.2
## [47] tibble_2.1.1
## [48] crayon_1.3.4
## [49] pkgconfig_2.0.2
## [50] MASS_7.3-51.1
## [51] iterators_1.0.10
## [52] assertthat_0.2.1
## [53] rmarkdown_1.12
## [54] Rhdf5lib_1.4.2
## [55] R6_2.4.0
## [56] MALDIquant_1.19.2
## [57] compiler_3.5.3
```

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

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.