Mass spectrometry-based proteomics

Laurent Gatto

2025-03-10

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.

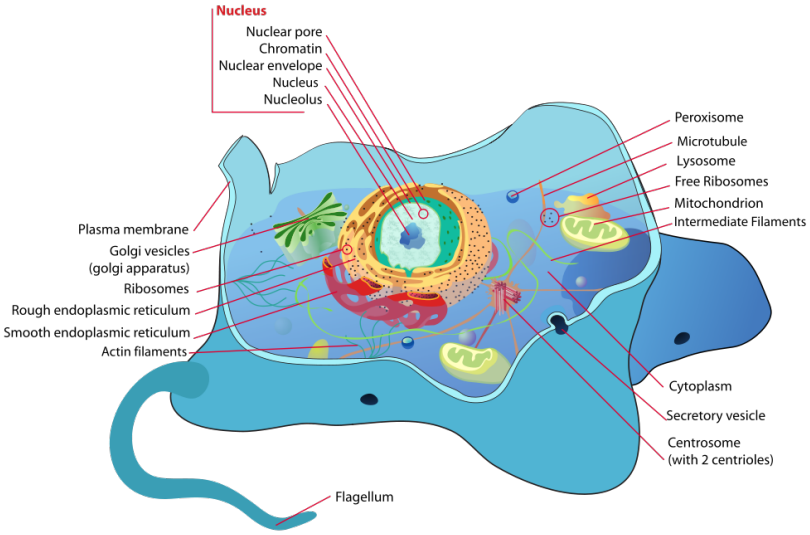


Figure 2: Sub-cellular structure of an animal cell (Source *Cell biology* on Wikipedia).

In addition to the standard information worflow where DNA is transcribed into RNA that itself is translated into proteins, there is also reverse transcription, that generates (complementary) DNA

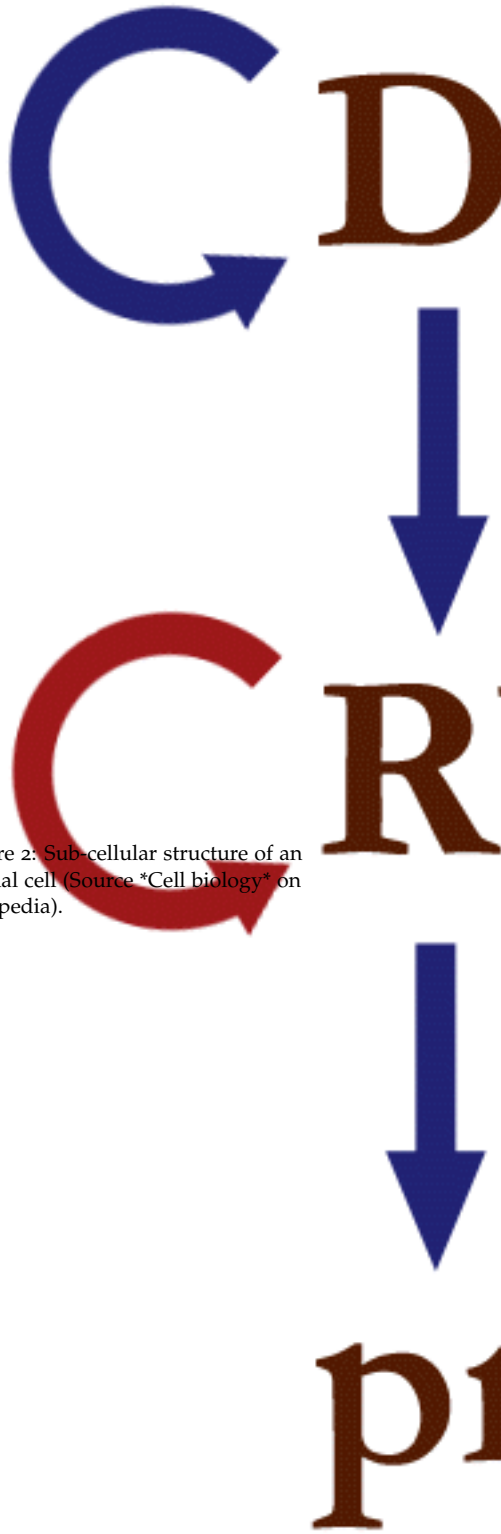


Figure 1: Information flow in biological systems (Source *Central dogma of biology* on Wikipedia).

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.org)

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 degraded, or where the proteins reside inside a cell.

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

A beginner's guide to mass spectrometry–based proteomics (Sinha and Mann 2020) is an approachable introduction to sample preparation, mass spectrometry and data analysis.

Setup

We are going to use the Bioconductor (Huber et al. 2015) `MSnbase` package (Laurent Gatto and Lilley 2012; Laurent Gatto, Gibb, and Rainer 2020), 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 (L. Gatto and Christoforou 2014) and (Laurent Gatto 2019), or explore the the proteomics- and mass spectrometry-related packages on the Bioconductor page

The recent R for Mass Spectrometry initiative, and the documetation book provide further details and state-of-the-art infrastructure for MS and proteomics data analysis.

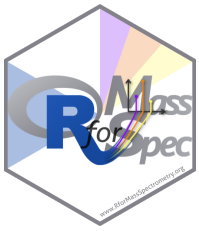


Figure 5: The *R for Mass Spectrometry* intiative.

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*.

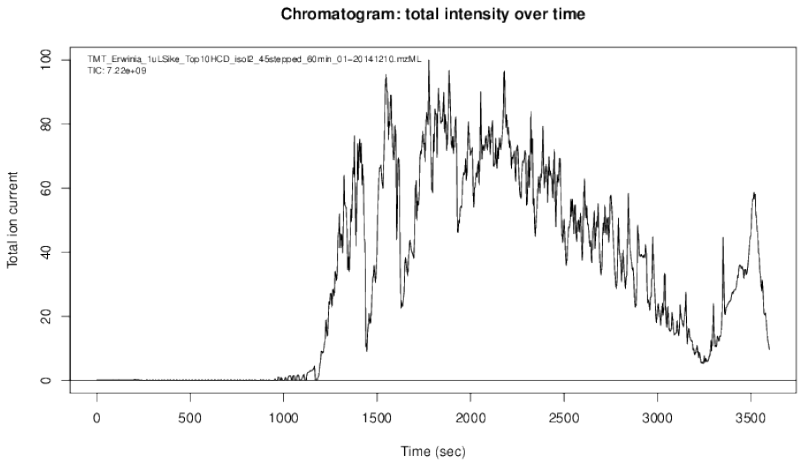


Figure 6: A chromatogram, illustrating the total amount of analytes over the retention time.

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).

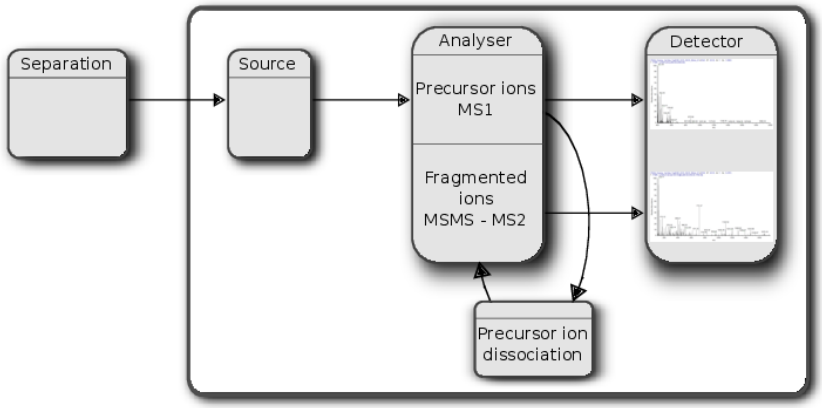


Figure 7: Schematics of a mass spectrometer and two rounds of MS.

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 (bot-

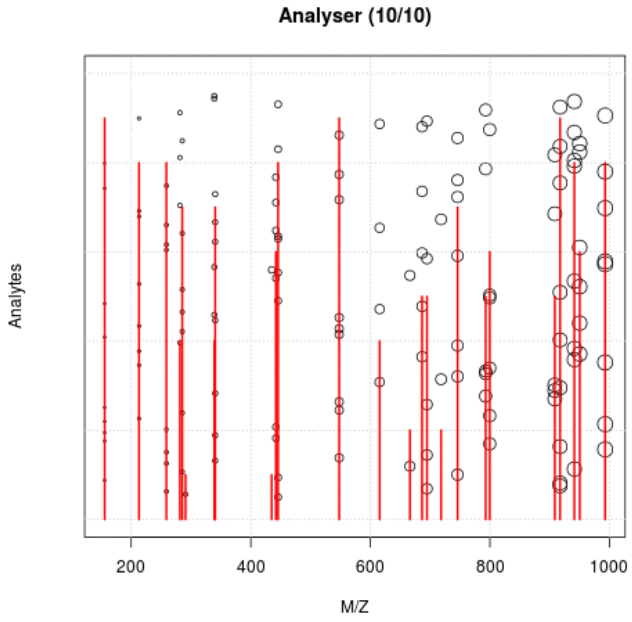


Figure 8: Separation and detection of ions in a mass spectrometer.

tom) 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.

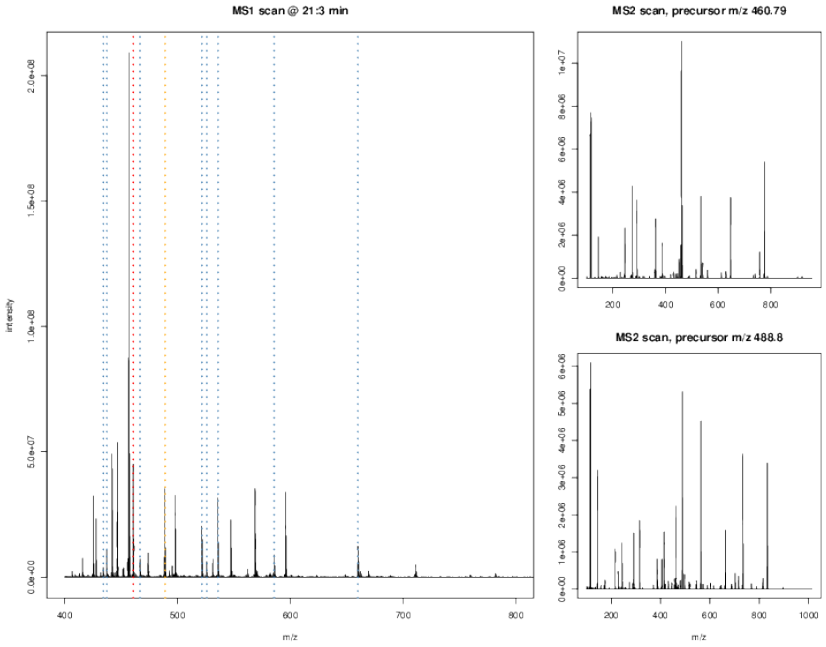


Figure 9: Parent ions in the MS1 spectrum (left) and two sected fragment ions MS2 spectra (right).

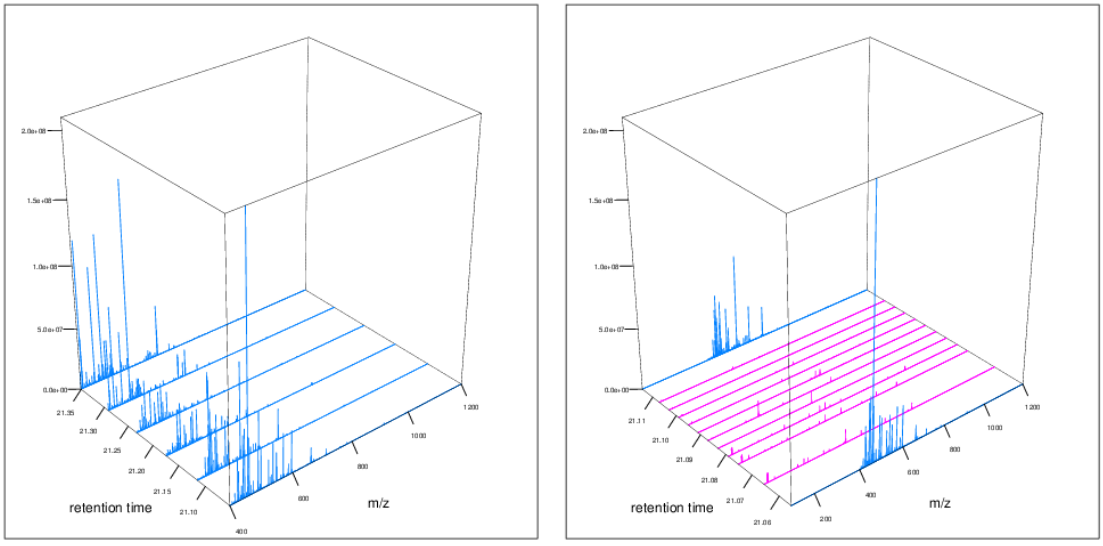


Figure 10: MS1 spectra (blue) over retention time (left). MS2 spectra (pink) interleaved between two MS1 spectra (right),

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")
```

```
## Warning: multiple methods tables found for 'intersect'
## Warning: multiple methods tables found for 'intersect'
```

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: 25:01 - 25:02 minutes
## - - - Processing information - - -
## Data loaded: Mon Mar 10 11:47:33 2025
##  MSnbase version: 2.32.0
## - - - 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: 25:02
##  Charge: 2
##  MSn level: 2
##  Peaks count: 2125
##  Total ion count: 150838188
```

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

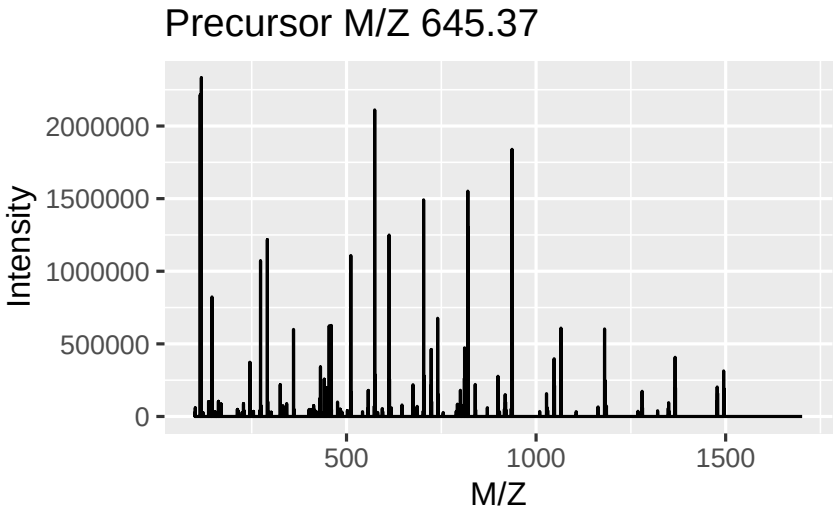


Figure 11: Visualisation of the 3rd MS spectrum in our small test data set.

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] "19:09" "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
## 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
##  [ reached 'max' / getOption("max.print") -- omitted 16 rows ]
```

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

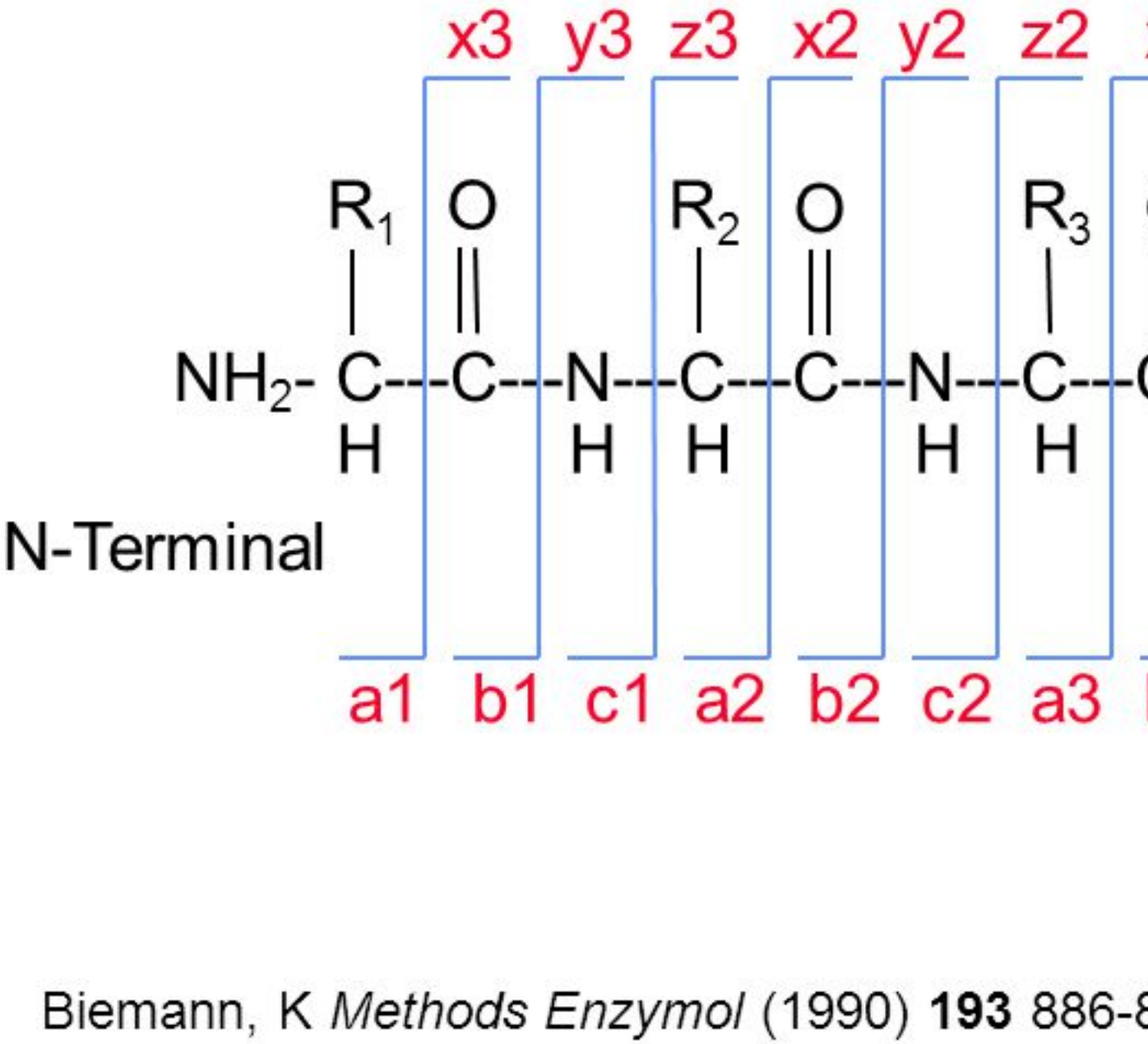
```
itraqdata2 <- pickPeaks(itraqdata, verbose = FALSE)
s <- "SIGFEGDSIGR"
plot(itraqdata2[[14]], s, main = s)
```

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

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

In a full experiment, all possible peptides from the known (or relevant) proteome of interest (such as databases that can be downloaded

Figure 12: Peptide fragmentation.



Biemann, K *Methods Enzymol* (1990) **193** 886-8

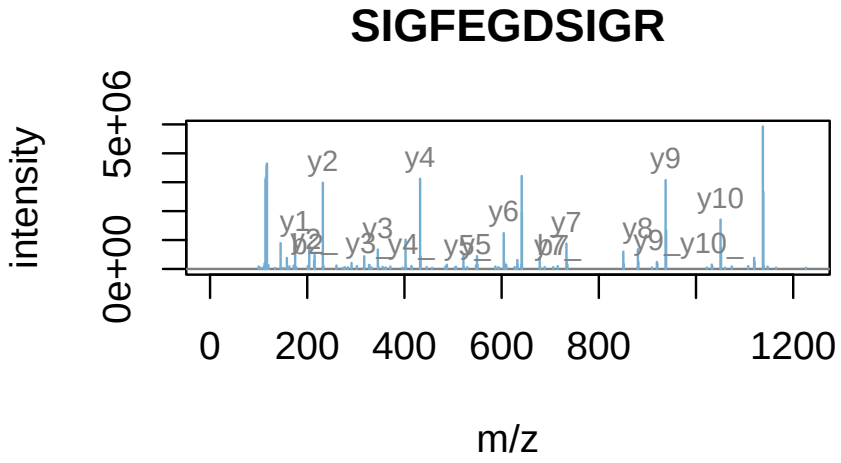


Figure 13: Matching observed and expected peaks.

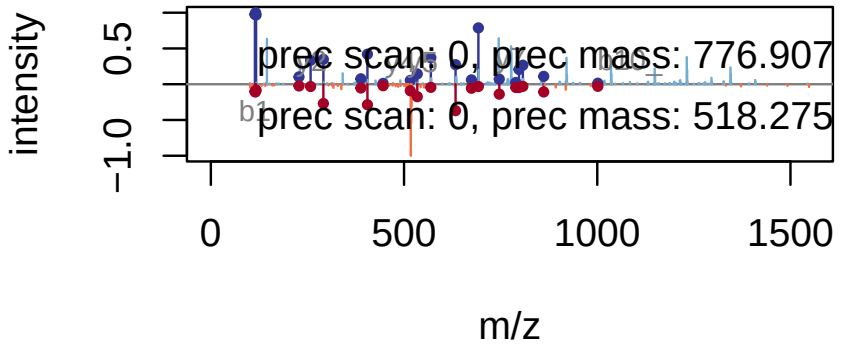


Figure 14: Direct comparison of 2 MS2 spectra.

from the UniProt site[1]) are compared to the millions of observed spectra.

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

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

Statistical challenge The matching between millions of observed and possible spectra causes real challenges due to the large search space and the risk of false positives. See [@Kall:2008] for further reading.

Statistical challenge Protein inference is a difficult task, as peptides often match multiple proteins (either different isoforms or proteins stemming from different gene but with identical domains), which leads to the definition of protein groups, i.e. sets of proteins that can't be distinguished with the set of identified peptides at hand. See [@Nesvizhskii:2005] for further reading.

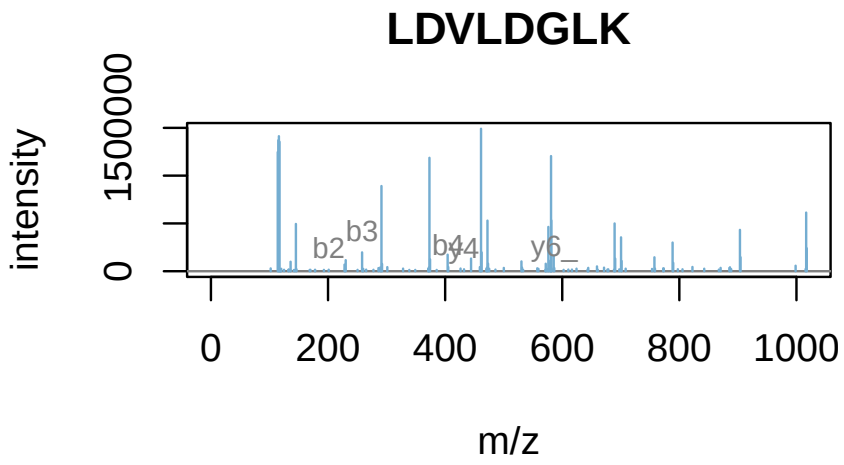
Exercise

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 AAGHDGK AAKPEAPAASPAPALGAR AALESAVK ... VWVVEGSK
```

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



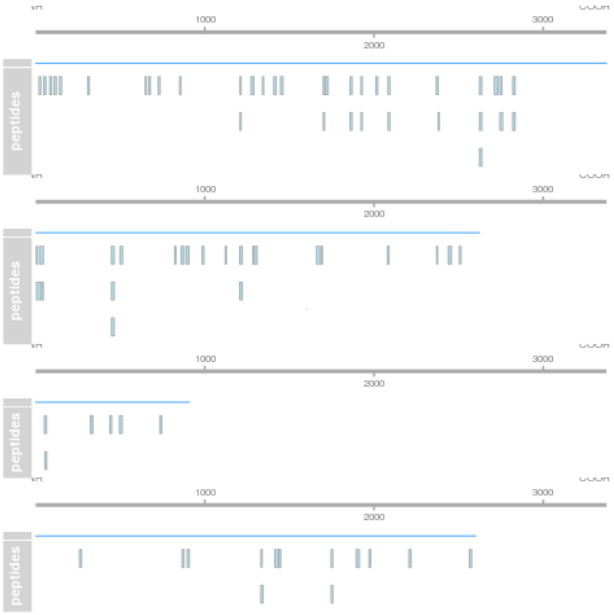
Quantitation

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

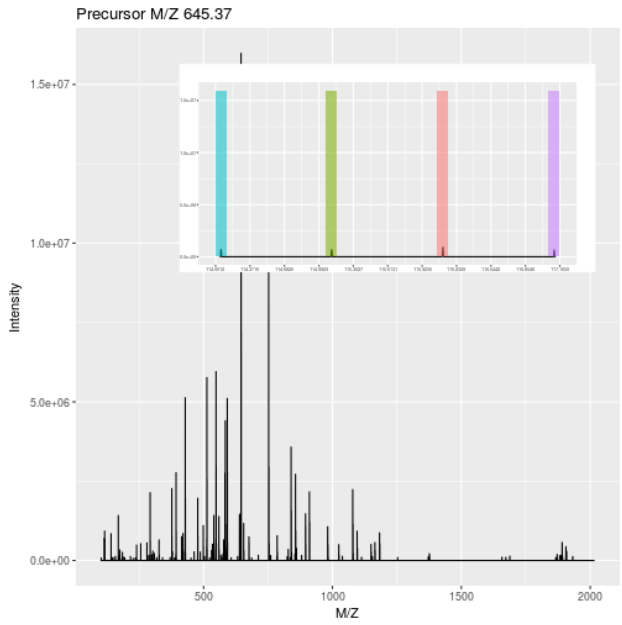
Label-free MS2: Spectral counting

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.

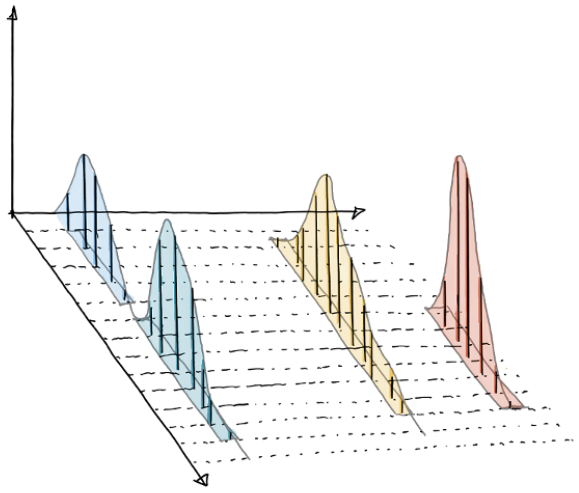


Figure: credit Johannes Rainer.

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.

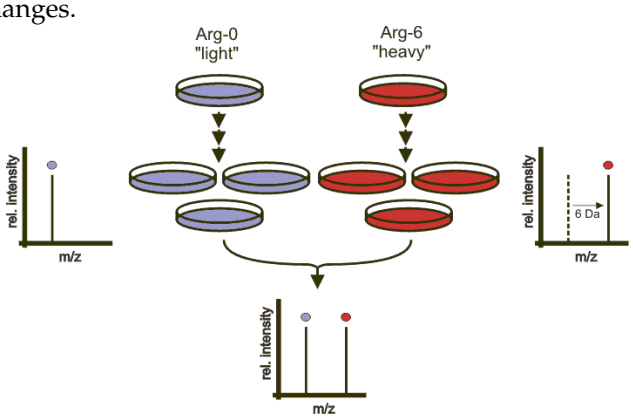


Figure: credit Wikimedia Commons.

Statistical challenge These different quantitation techniques come with their respective and distinct challenges, such as large quantities of raw data processing, data transformation and normalisation, and different underlying statistical models for the quantitative data (count data for spectral counting, continuous data for the others).

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)
```

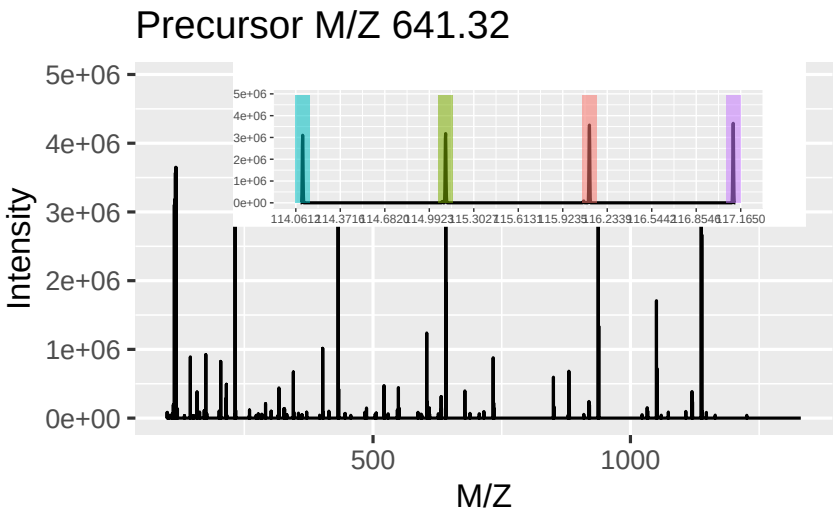


Figure 15: Visualisation of the iTRAQ reporter peaks.

Interpret the figure above.

From the closed-up panel, we see that the 4 reporter ions indicate that there was slightly less of that peptide in the two first samples, i.e. those labelled with the 114 and 115 reporters.

The rest of the spectrum (i.e $m/z > 120$) correspond to the dissociated fragment ions, used for identification.

Quantification

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: Mon Mar 10 11:47:39 2025
##  MSnbase version: 1.1.22
```

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

```
##     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(fData(msnset))
```

```
##     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 fileIdx retention.time precursor.mz precursor.intensity
## X1           NYQEAK       1        1149.31     520.7833             3449020
## X10 VTLVDTGEHSMTGGR       1        1503.03     573.9539             7849420
## X11           SPIWR       1        1663.61     401.7392            41253600
## X12        TAIDDALK       1        1663.86     567.8339            23549500
## X13          SILINK       1        1664.08     488.3269            13025200
## X14    DFEVVNNESDPR       1        1664.32     782.8715            18405000
##     charge peaks.count       tic  ionCount ms.level acquisition.number
## X1       2        1922  26413754  26413754        2                 2
## X10      3        1376  24482281  24482281        2                11
## X11      2        1571 231075934 231075934        2                12
## X12      2        2397 247323187 247323187        2                13
## X13      2        2574 207247502 207247502        2                14
## X14      2        1829 115317275 115317275        2                15
##     collision.energy
## X1                40
## X10               40
## X11               40
## X12               40
## X13               40
## X14               40
```

Quantitative data processing

Statistical challenge Quantitative data processing come with numerous statistical challenges such as missing data handling, aggregation of quantitation data, data normalisation, ... and the effect of these on the downstream statistical tests.

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

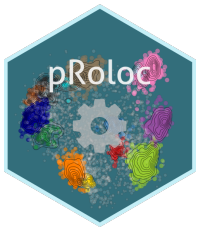


Figure 16: The 'pRoloc' package.

Applications in statistical learning

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 (L. Gatto et al. 2014; Breckels et al. 2016). Both packages can be installed with the `BiocManager::install` function, as illustrated above.

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

```
## Warning: multiple methods tables found for 'intersect'
```

```
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)
```

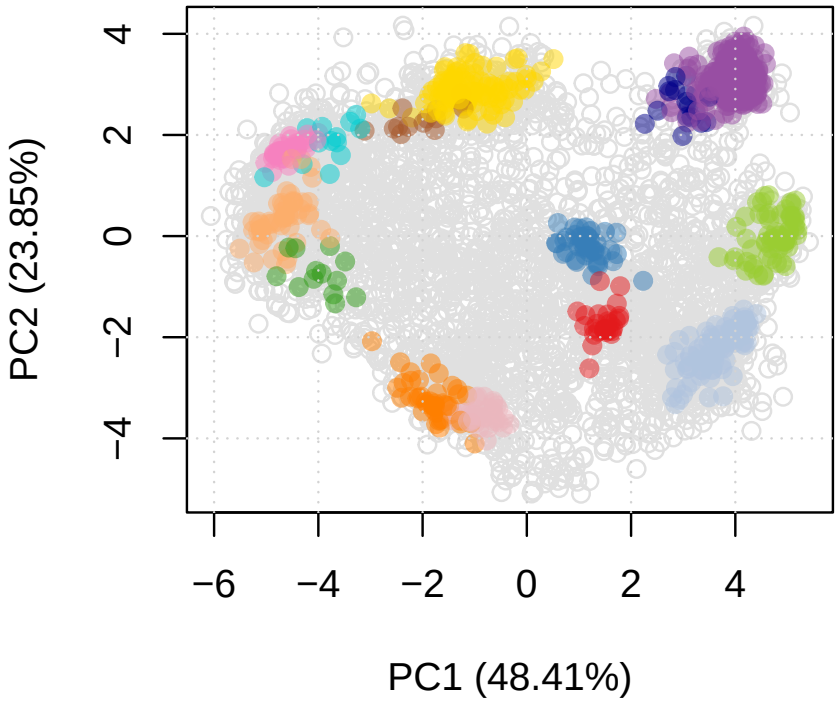


Figure 17: Mouse stem cell spatial proteomics data from Christoforou et al.

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 4.4.1 (2024-06-14)
```

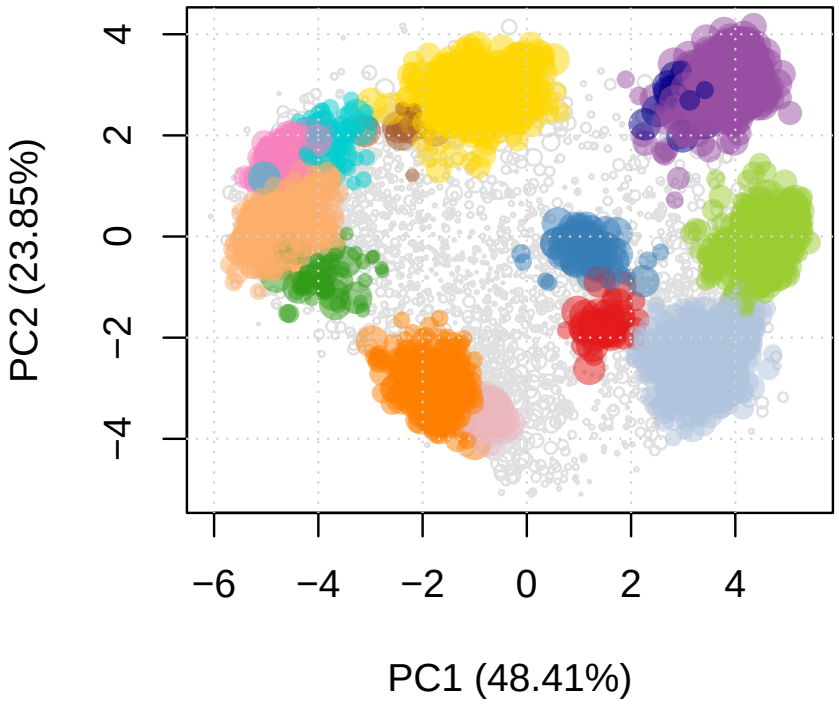


Figure 18: New sub-cellular assignment after using support vector machine classifier.

```
## Platform: x86_64-pc-linux-gnu
## Running under: Ubuntu 24.04.2 LTS
##
## Matrix products: default
## BLAS:   /opt/Rpatched/lib/R/lib/libRblas.so
## LAPACK: /opt/Rpatched/lib/R/lib/libRlapack.so;  LAPACK version 3.12.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Europe/Brussels
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
```

```
## other attached packages:
##  [1] pRolocdata_1.44.1    pRoloc_1.46.1        BiocParallel_1.40.0
##  [4] MLInterfaces_1.86.0  cluster_2.1.6        annotate_1.84.0
##  [7] XML_3.99-0.18        AnnotationDbi_1.68.0 IRanges_2.40.1
## [10] MSnbase_2.32.0       ProtGenerics_1.38.0  S4Vectors_0.44.0
## [13] mzR_2.40.0           Rcpp_1.0.14          Biobase_2.66.0
## [16] BiocGenerics_0.52.0
##
## loaded via a namespace (and not attached):
##   [1] splines_4.4.1               filelock_1.0.3
##   [3] tibble_3.2.1                hardhat_1.4.1
##   [5] preprocessCore_1.68.0       pROC_1.18.5
##   [7] rpart_4.1.23                lifecycle_1.0.4
##   [9] httr2_1.1.0                 doParallel_1.0.17
##  [11] globals_0.16.3              lattice_0.22-6
##  [13] MASS_7.3-60.2               MultiAssayExperiment_1.32.0
##  [15] dendextend_1.19.0           magrittr_2.0.3
##  [17] limma_3.62.2                plotly_4.10.4
##  [19] rmarkdown_2.29              yaml_2.3.10
##  [21] MsCoreUtils_1.18.0          DBI_1.2.3
##  [23] RColorBrewer_1.1-3          lubridate_1.9.4
##  [25] abind_1.4-8                 zlibbioc_1.52.0
##  [27] GenomicRanges_1.58.0        purrr_1.0.4
##  [29] mixtools_2.0.0              AnnotationFilter_1.30.0
##  [31] nnet_7.3-20                 rappdirs_0.3.3
##  [33] ipred_0.9-15                lava_1.8.1
##  [35] GenomeInfoDbData_1.2.13     listenv_0.9.1
##  [37] parallelly_1.42.0           ncdf4_1.23
##  [39] codetools_0.2-20            DelayedArray_0.32.0
##  [41] xml2_1.3.7                  tidyselect_1.2.1
##  [43] UCSC.utils_1.2.0            farver_2.1.2
##  [45] viridis_0.6.5               matrixStats_1.5.0
##  [47] BiocFileCache_2.14.0        jsonlite_1.9.1
##  [49] caret_7.0-1                 e1071_1.7-16
##  [51] survival_3.6-4              iterators_1.0.14
##  [53] foreach_1.5.2               segmented_2.1-4
##  [55] tools_4.4.1                 progress_1.2.3
##  [57] glue_1.8.0                  prodlim_2024.06.25
##  [59] gridExtra_2.3               SparseArray_1.6.2
##  [61] xfun_0.51                   tufte_0.13
##  [63] MatrixGenerics_1.18.1       GenomeInfoDb_1.42.3
##  [65] dplyr_1.1.4                 withr_3.0.2
##  [67] BiocManager_1.30.25         fastmap_1.2.0
##  [69] digest_0.6.37               timechange_0.3.0
```

```
##  [71] R6_2.6.1                    colorspace_2.1-1
##  [73] gtools_3.9.5                lpSolve_5.6.23
##  [75] biomaRt_2.62.1              RSQLite_2.3.9
##  [77] tidyr_1.3.1                 generics_0.1.3
##  [79] hexbin_1.28.5               data.table_1.17.0
##  [81] recipes_1.1.1               FNN_1.1.4.1
##  [83] class_7.3-22                prettyunits_1.2.0
##  [85] PSMatch_1.10.0              httr_1.4.7
##  [87] htmlwidgets_1.6.4           S4Arrays_1.6.0
##  [89] ModelMetrics_1.2.2.2        pkgconfig_2.0.3
##  [91] gtable_0.3.6                timeDate_4041.110
##  [93] blob_1.2.4                  impute_1.80.0
##  [95] XVector_0.46.0              htmltools_0.5.8.1
##  [97] MALDIquant_1.22.3           clue_0.3-66
##  [99] scales_1.3.0                png_0.1-8
##  [ reached getOption("max.print") -- omitted 52 entries ]
```

References

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. https://doi.org/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. https://doi.org/10.1038/ncomms9992.

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

Gatto, Laurent, Sebastian Gibb, and Johannes Rainer. 2020. "MSnbase, Efficient and Elegant R-Based Processing and Visualisation of Raw Mass Spectrometry Data." *bioRxiv*. https://doi.org/10.1101/2020.04.29.067868.

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.

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): 1322–24. https://doi.org/10.1093/bioinformatics/btu013.

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

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. https://doi.org/10.1021/acs.jproteome.5b00981.

Sinha, Ankit, and Matthias Mann. 2020. "A beginner's guide to mass spectrometry–based proteomics." *The Biochemist*, September. https://doi.org/10.1042/BIO20200057.