MSI project report and manual

GOLI HAVEESH

haveeshgoli4@gmail.com

Mar – Nov 2020

1. Introduction

Mass spectrometry imaging (MSI) is applied to measure the spatial distribution of hundreds of biomolecules in a sample. A mass spectrometer scans over the entire sample and collects a mass spectrum every 5-200 µm. This results in thousands of spots in which a mass spectrum is acquired. Each mass spectrum consists of hundreds of analytes that are measured by their mass-to-charge (m/z) ratio. For each analyte the peak intensity in the mass spectra of every pixel is known and can be set together to map the spatial distribution of the analyte in the sample.

The technique has a broad range of applications as it is able to measure many different kinds of analytes such as peptides, proteins, metabolites or chemical compounds in a large variety of samples such as cells, tissues and liquid biopsies [1]. Application areas include pharmacokinetic studies, biomarker discovery, molecular pathology, forensic studies, plant research and material sciences. The strength of MSI is the simultaneous analysis of hundreds of analytes in an unbiassed, untargeted, label free, fast and affordable measurement while maintaining morphological information.

Depending on the analyte of interest and the application, different mass spectrometers are used. A mass spectrometer measures the analytes by ionizing, evaporating and sorting them by their mass-to-charge (m/z) ratio. Put simply, a mass spectrometer needs basically three parts: an ionization source, a mass analyzer and a detector. Most common ionization sources for MSI are MALDI (Matrix Assisted Laser Desorption/Ionization), DESI (Desorption Electrospray Ionization)[2].

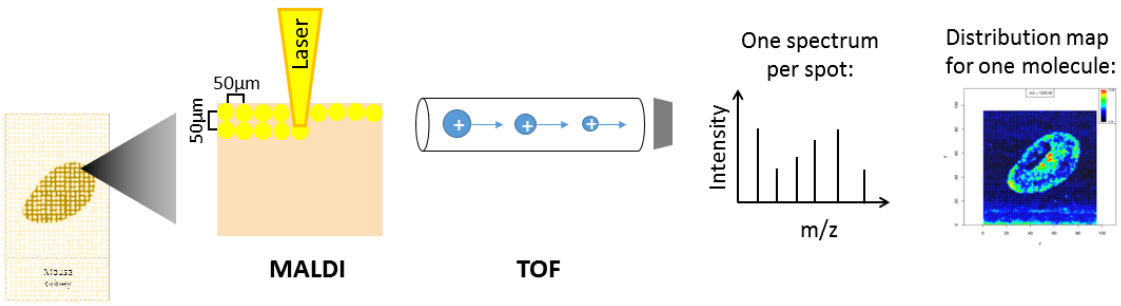


Figure 1: MALDI TOF imaging of a human colorectal sample

2. Objectives

The objectives of the project are as follows:

- To extract and pre process the 3D MALDI raw data of human colorectal samples
- To develop Mass Spectroscopy imaging analysis pipeline to identify proteins and metabolites of tumor tissues in human colorectal cancer samples

3. Data pre-processing

During MALDI ionization a laser shoots onto the sample that was covered with a special matrix which absorbs the laser energy and transfers it to the analytes. This process evaporizes and ionizes the analytes that due to their charge can then be accelerated in an electrical field towards the TOF tube. The time of flight through the tube to the detector is measured and as this correlates with the mass over charge (m/z) of the analyte, the flight time allows the calculation of m/z . During measurement complete mass spectra with hundreds of m/z - intensity pairs are acquired in thousands of sample plots leading to big and complex data. Each mass spectrum is annotated with coordinates (x,y) that define its location in the sample. This allows to visualize the intensity distribution of each m/z feature in the sample as a heatmap. Depending on the analyte of interest, the sample type and the mass spectrometer the sample preparation steps as well as the properties of the acquired data differ.

The imzML file format was introduced to ease the exchange of MSI data between different instruments and data analysis software [3]. More and more vendors provide an imzML export option and there are software solutions to convert proprietary files into imzML files. Before starting any analysis, it is recommended to visualize all features of the data in different ways to better understand the data and to obtain an idea about it's quality and usefulness.

In case no direct imzML export is provided by the mass spectrometer software, proprietary files can be converted to imzML files with the tools provided on the following website: ms-imaging.org. The imzML file consists of two files: The first file contains the metadata in an XML file and has the extension `. imzML`. The second file contains the mass spectra data and is saved as binary file and its extension is `.ibd`. To be valid both files must have the same filename before the extension. More information about the imzML file structure can be found here: ms-imaging.org. Galaxy provides the `composite` upload for files consisting of several components.

The data for this tutorial comes from MALDI-TOF imaging of peptides in a mouse kidney. To make computation times more suitable for this training, only the m/z range from 1220 - 1625 and a part of the pixels (purple rectangle in figure 1) containing about half of the kidney and one digestion control spot were kept.

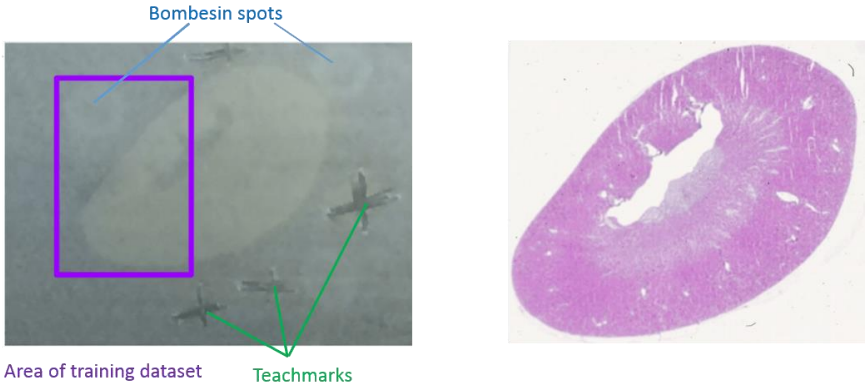


Figure 2: Mouse kidney sample before and after mass spectrometry measurement and H&E staining

4. Data Analysis

Data analysis pipeline of 3D MALDE images is explained in this section.

4.1 Version info

- R version: R version 4.0.2 (2020-06-22)
- Bioconductor version: 3.12
- Package version: 1.13.1

4.2 Setup

The following packages were used throughout the pipeline. R version `3.3.1` or higher is required to install all the packages using `BiocManager::install`.

```
library("mzR")
library("mzID")
library("MSnID")
library("MSnbase")
library("rpx")
library("MLInterfaces")
library("pRoloc")
library("pRolocdata")
library("MSGFplus")
library("rols")
library("hpar")
```

The most convenient way to install all the tutorials requirement (and more related content), is to install *RforProteomics* with all its dependencies.

```
library("BiocManager")
```

```
BiocManager::install("RforProteomics", dependencies = TRUE)
```

This workflow illustrates R / Bioconductor infrastructure for proteomics. Topics covered focus on support for open community-driven formats for raw data and identification results, packages for peptide-spectrum matching, data processing and analysis:

- Exploring available infrastructure
- Mass spectrometry data
- Handling raw MS data
- Handling identification data
- MS/MS database search
- Analysing search results
- High-level data interface
- Data processing and analysis
- Statistical analysis

In Bioconductor version 3.12, there are respectively 138 proteomics, 94 mass spectrometry software packages and 23 mass spectrometry experiment packages. These respective packages can be extracted with the proteomicsPackages(), massSpectrometryPackages() and massSpectrometryDataPackages() and explored interactively.

```
library("RforProteomics")
pp <- proteomicsPackages()
display(pp
```

4.3 Mass spectrometry data

The following are the image formats observed in our data

Type	Format	Package
raw	mzML, mzXML, netCDF, mzData	`mzR` (read)
identification	mzIdentML	`mzR` (read) and `mzID` (read)
quantitation	mzQuantML	
peak lists	mgf	`MSnbase` (read/write)
other	mzTab	`MSnbase` (read)

4.4 Handling raw MS data

Below, we access the raw data file and open a file handle that will allow us to extract data and metadata of interest.

```
library("mzR")
ms <- openMSfile(mzf)
ms
## Mass Spectrometry file handle.
## Filename:  TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01-20141210.mzML
## Number of scans:  7534
```

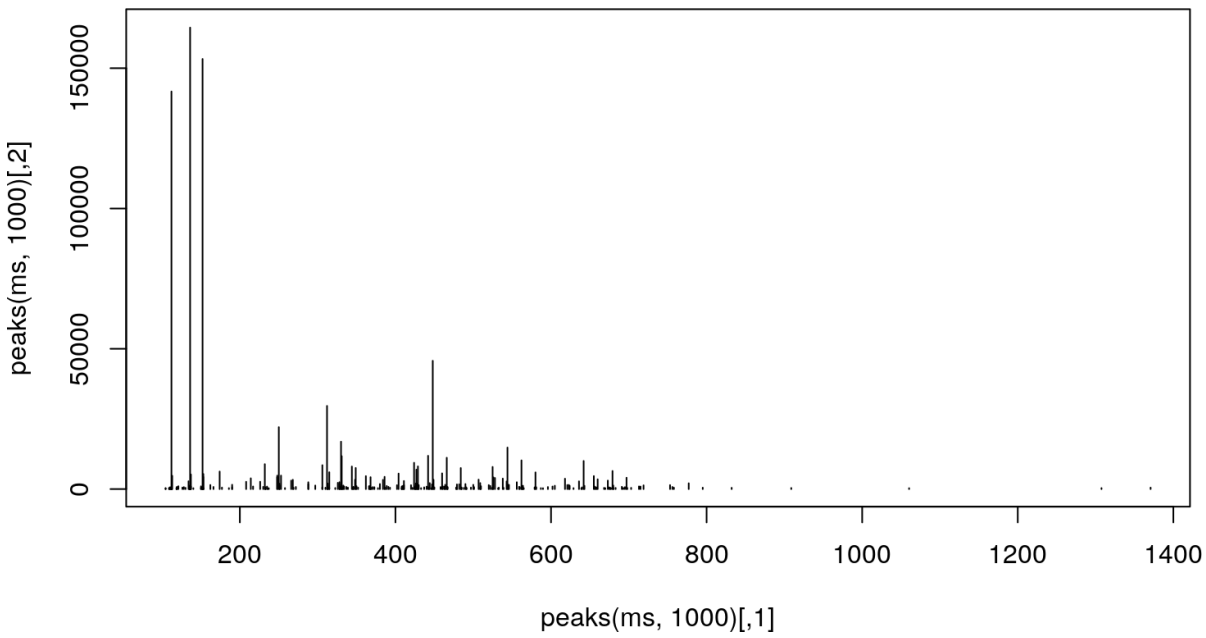
The `header` function returns the metadata of all available peaks:

```
hd <- header(ms)
dim(hd)
## [1] 7534   31
names(hd)
##  [1] "seqNum"                    "acquisitionNum"
##  [3] "msLevel"                   "polarity"
##  [5] "peaksCount"                "totIonCurrent"
##  [7] "retentionTime"             "basePeakMZ"
##  [9] "basePeakIntensity"         "collisionEnergy"
## [11] "ionisationEnergy"          "lowMZ"
## [13] "highMZ"                    "precursorScanNum"
## [15] "precursorMZ"               "precursorCharge"
## [17] "precursorIntensity"        "mergedScan"
## [19] "mergedResultScanNum"       "mergedResultStartScanNum"
## [21] "mergedResultEndScanNum"    "injectionTime"
## [23] "filterString"              "spectrumId"
## [25] "centroided"                "ionMobilityDriftTime"
## [27] "isolationWindowTargetMZ"   "isolationWindowLowerOffset"
## [29] "isolationWindowUpperOffset" "scanWindowLowerLimit"
## [31] "scanWindowUpperLimit"
```

We can extract metadata and scan data for scan 1000 as follows:

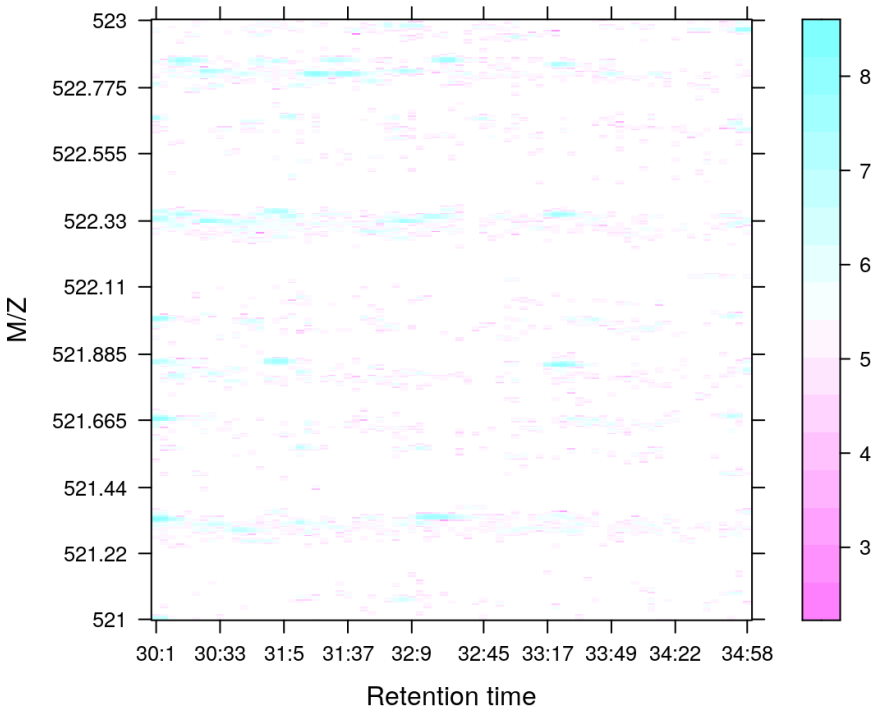
```
hd[1000, ]
##      seqNum acquisitionNum msLevel polarity peaksCount totIonCurrent
## 1000   1000           1000       2        1        274       1048554
##      retentionTime basePeakMZ basePeakIntensity collisionEnergy
## 1000      1106.916    136.061            164464              45
##      ionisationEnergy    lowMZ   highMZ precursorScanNum precursorMZ
## 1000                0 104.5467 1370.758              992    683.0817
##      precursorCharge precursorIntensity mergedScan mergedResultScanNum
## 1000               2           689443.7         NA                  NA
##      mergedResultStartScanNum mergedResultEndScanNum injectionTime
## 1000                       NA                     NA      55.21463
##                                                 filterString
## 1000 FTMS + p NSI d Full ms2 683.08@hcd45.00 [100.00-1380.00]
##                                    spectrumId centroided
## 1000 controllerType=0 controllerNumber=1 scan=1000       TRUE
##      ionMobilityDriftTime isolationWindowTargetMZ isolationWindowLowerOffset
## 1000                   NA                  683.08                          1
##      isolationWindowUpperOffset scanWindowLowerLimit scanWindowUpperLimit
## 1000                          1                  100                 1380
head(peaks(ms, 1000))
##          [,1]     [,2]
## [1,] 104.5467 308.9326
## [2,] 104.5684 308.6961
## [3,] 108.8340 346.7183
```

```
## [4,] 109.3928 365.1236
## [5,] 110.0345 616.7905
## [6,] 110.0703 429.1975
plot(peaks(ms, 1000), type = "h")
```

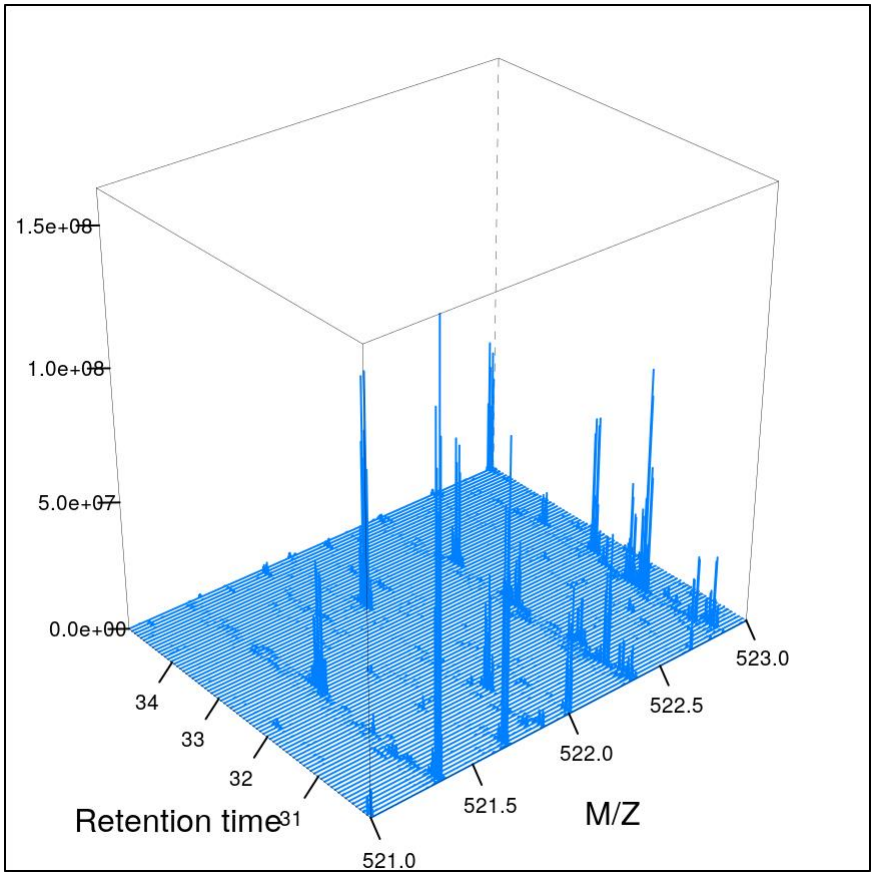


Below we reproduce the example from the `MSmap` function from the `MSnbase` package to plot a specific slice of the raw data using the `mzR` functions we have just described.

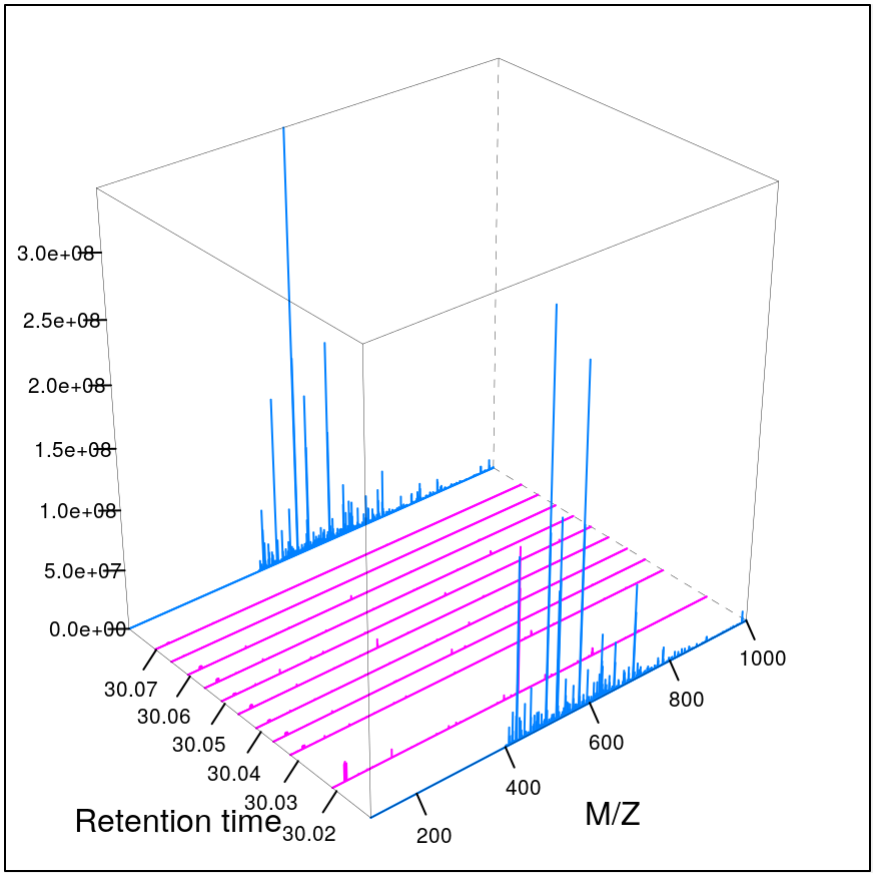
```
## a set of spectra of interest: MS1 spectra eluted
## between 30 and 35 minutes retention time
ms1 <- which(hd$msLevel == 1)
rtsel <- hd$retentionTime[ms1] / 60 > 30 &
    hd$retentionTime[ms1] / 60 < 35

## the map
M <- MSmap(ms, ms1[rtsel], 521, 523, .005, hd)

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




```
plot3D(M)
```




```
## with some MS2 spectra
i <- ms1[which(rtsel)][1]
j <- ms1[which(rtsel)][2]
M2 <- MSmap(ms, i:j, 100, 1000, 1, hd)
plot3D(M2)
```



4.5 Handling identification data

```
library("mzID")
f <- dir(system.file("extdata", package = "RforProteomics"),
         pattern = "mzid", full.names=TRUE)
basename(f)
## [1] "TMT_Erwinia.mzid.gz"
id <- mzID(f)
## reading TMT_Erwinia.mzid.gz... DONE!
id
## An mzID object
##
## Software used:   MS-GF+ (version: Beta (v10072))
##
## Rawfile:         /home/lgatto/dev/00_github/RforProteomics/sandbox/TMT_Erwinia_1uLSike_Top10HCD_isol2_45stepped_60min_01.mzXML
##
## Database:        /home/lgatto/dev/00_github/RforProteomics/sandbox/erwinia_carotovora.fasta
##
## Number of scans: 5287
## Number of PSM's: 5563
```

Various data can be extracted from the `mzID` object, using one the accessor functions such as `database`, `scans`, `peptides`, … The object can also be converted into a `data.frame` using the `flatten` function.

The `mzR` package also provides support fasta parsing `mzIdentML` files with the `openIDfile` function. As for raw data, the underlying C/C++ code comes from the proteowizard.

```
library("mzR")
f <- dir(system.file("extdata", package = "RforProteomics"),
         pattern = "mzid", full.names=TRUE)

id1 <- openIDfile(f)
fid1 <- mzR::psms(id1)

head(fid1)
##   spectrumID chargeState rank passThreshold experimentalMassToCharge
## 1  scan=5782           3    1          TRUE                1080.2325
## 2  scan=6037           3    1          TRUE                1002.2089
## 3  scan=5235           3    1          TRUE                1189.2836
## 4  scan=5397           3    1          TRUE                 960.5365
## 5  scan=6075           3    1          TRUE                1264.3409
## 6  scan=5761           2    1          TRUE                1268.6429
##   calculatedMassToCharge                           sequence peptideRef modNum
## 1              1080.2321 PVQIQAGEDSNVIGALGGAVLGGFLGNTIGGGSGR       Pep1      0
## 2              1002.2115        TQVLDGLINANDIEVPVALIDGEIDVLR       Pep2      0
## 3              1189.2800   TKGLNVMQNLLTAHPDVQAVFAQNDEMALGALR       Pep3      0
## 4               960.5365         SQILQQAGTSVLSQANQVPQTVLSLLR       Pep4      0
## 5              1264.3419 PIIGDNPFVVVLPDVVLDESTADQTQENLALLISR       Pep5      0
## 6              1268.6501             WTSQSSLDLGEPLSLITESVFAR       Pep6      0
##   isDecoy post pre start end DatabaseAccess DBseqLength DatabaseSeq
## 1   FALSE    S   R    50  84        ECA1932         155
## 2   FALSE    R   K   288 315        ECA1147         434
## 3   FALSE    A   R   192 224        ECA0013         295
## 4   FALSE    -   R   264 290        ECA1731         290
## 5   FALSE    F   R   119 153        ECA1443         298
## 6   FALSE    Y   K   264 286        ECA1444         468
##                                     DatabaseDescription scan.number.s.
## 1                        ECA1932 outer membrane lipoprotein           5782
## 2                                   ECA1147 trigger factor           6037
## 3                  ECA0013 ribose-binding periplasmic protein           5235
## 4                                     ECA1731 flagellin           5397
## 5      ECA1443 UTP--glucose-1-phosphate uridylyltransferase           6075
## 6 ECA1444 6-phosphogluconate dehydrogenase, decarboxylating           5761
##   acquisitionNum
## 1           5782
## 2           6037
```

```
## 3           5235
## 4           5397
## 5           6075
## 6           5761
```

4.6 Analysis of metabolites

Cleaning irregular cleavages at the termini of the metabolites and missing cleavage site within the metabolite sequences. The following two function call create the nuMinClevages and numIrr Cleavages columns in the MSnID object

```
msnid <- assess_termini(msnid, validCleavagePattern="[KR]\\.[^P]")
msnid <- assess_missed_cleavages(msnid, missedCleavagePattern="[KR](?=[^P$])")
```

4.7 Trimming the data

Now, we can use the `apply_filter` function to effectively apply filters. The strings passed to the function represent expressions that will be evaluated, this keeping only PSMs that have 0 irregular cleavages and 2 or less missed cleavages.

```
msnid <- apply_filter(msnid, "numIrregCleavages == 0")
msnid <- apply_filter(msnid, "numMissCleavages <= 2")
show(msnid)
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 7838 at 17 % FDR
## #peptides: 5598 at 23 % FDR
## #accessions: 3759 at 53 % FDR
```

4.8 Filtering criteria

Filtering of the identification data will rely on

- -log10 transformed MS-GF+ Spectrum E-value, reflecting the goodness of match experimental and theoretical fragmentation patterns

```
msnid$msmsScore <- -log10(msnid$`MS-GF:SpecEValue`)
```

- the absolute mass measurement error (in ppm units) of the parent ion

```
msnid$absParentMassErrorPPM <- abs(mass_measurement_error(msnid))
```

MS2 fiters are handled by a special `MSnIDFilter` class objects, where individual filters are set by name (that is present in `names(msnid)`) and comparison operator (>, <, = , …) defining if we should retain hits with higher or lower given the threshold and finally the threshold value itself.

```
filtObj <- MSnIDFilter(msnid)
filtObj$absParentMassErrorPPM <- list(comparison="<", threshold=10.0)
filtObj$msmsScore <- list(comparison=">", threshold=10.0)
show(filtObj)
## MSnIDFilter object
```

```
## (absParentMassErrorPPM < 10) & (msmsScore > 10)
```

We can then evaluate the filter on the identification data object, which return the false discovery rate and number of retained identifications for the filtering criteria at hand.

```
evaluate_filter(msnid, filtObj)
##           fdr    n
## PSM         0 3807
## peptide     0 2455
## accession   0 1009
```

4.9 Filter optimization

Rather than setting filtering values by hand, as shown above, these can be set automatically to meet a specific false discovery rate.

```
filtObj.grid <- optimize_filter(filtObj, msnid, fdr.max=0.01,
                                method="Grid", level="peptide",
                                n.iter=500)
show(filtObj.grid)
## MSnIDFilter object
## (absParentMassErrorPPM < 3) & (msmsScore > 7.4)
evaluate_filter(msnid, filtObj.grid)
##                   fdr    n
## PSM       0.004097561 5146
## peptide   0.006447651 3278
## accession 0.021996616 1208
```

Filters can eventually be applied (rather than just evaluated) using the `apply_filter` function.

```
msnid <- apply_filter(msnid, filtObj.grid)
show(msnid)
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 5146 at 0.41 % FDR
## #peptides: 3278 at 0.64 % FDR
## #accessions: 1208 at 2.2 % FDR
```

And finally, identifications that matched decoy and contaminant protein sequences are removed

```
msnid <- apply_filter(msnid, "isDecoy == FALSE")
msnid <- apply_filter(msnid, "!grepl('Contaminant',accession)")
show(msnid)
## MSnID object
## Working directory: "."
## #Spectrum Files:  1
## #PSMs: 5117 at 0 % FDR
## #peptides: 3251 at 0 % FDR
## #accessions: 1179 at 0 % FDR
```

The resulting filtered identification data can be exported to a `data.frame` or to a dedicated `MSnSet` data structure for quantitative MS data, described below, and further processed and analyses using appropriate statistical tests.

5 References

[1] W. M. Abdelmoula *et al.*, "Data-driven identification of prognostic tumor subpopulations using spatially mapped t-SNE of Mass spectrometry imaging data," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 113, no. 43, pp. 12244–12249, 2016.

[2] P. Inglese *et al.*, "Deep learning and 3D-DESI imaging reveal the hidden metabolic heterogeneity of cancer," *Chem. Sci.*, vol. 8, no. 5, pp. 3500–3511, 2017.

[3] T. Schramm *et al.*, "imzML-A common data format for the flexible exchange and processing of mass spectrometry imaging data ☆," 2012.