Metabolomics data pre-processing using xcms

Johannes Rainer
Eurac Research, Bolzano, Italy
johannes.rainer@eurac.edu - github/twitter: jotsetung

24 June 2018

Hands-on?

- https://github.com/jotsetung/metabolomics2018
- Open *xcms-preprocessing.Rmd* in e.g. RStudio.

Content

This presentation focuses on updates of `xcms`:

- re-use data structures from Bioconductor's `MSnbase` package
- simplified raw data access

Content:

- Basic MS data handling (`MSnbase`)
- Simple MS data centroiding (`MSnbase`)
- LC-MS data pre-processing (`xcms`):
    - chromatographic peak detection
    - alignment
    - correspondence

Basic MS data handling

Data import and representation

- **Data set**:
  - subset from 2 files with pooled human serum samples
  - UHPLC (Agilent 1290) coupled with Q-TOF MS (TripleTOF 5600+ AB Sciex)
  - HILIC-based chromatographic separation
- Define file names and sample descriptions.

```
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)

## Define a data.frame with additional information on the files.
pd <- data.frame(file = basename(fls), injection_idx = c(1, 19),
                 sample = c("POOL_1", "POOL_2"), group = "POOL")
```

Data import and representation

- Read data from mzML/mzXML/CDF files with `readMSData` function.

```
## Read the data
data <- readMSData(fls, pdata = new("NAnnotatedDataFrame", pd),
                   mode = "onDisk")
```

- `mode = "onDisk"`: reads only spectrum header from files, but no data.
- on-disk mode enables analysis of very large experiments.

Basic data access

- Access sample/phenotype information using `pData` or `$`:

```
## Access phenotype information
pData(data)
```

```
##                                 file injection_idx sample group
## 1 20171016_POOL_POS_1_105-134.mzML             1 POOL_1  POOL
## 2 20171016_POOL_POS_3_105-134.mzML            19 POOL_2  POOL
```

```
## Or individual columns directly using the $ operator
data$injection_idx
```

```
## [1]  1 19
```

Basic data access

- Access general spectrum information: `msLevel`, `centroided`, `rtime`, `polarity`.
- Access MS data: `spectra`, `mz`, `intensity`: reads data from files.
- In most cases we work with subsets: use filter functions to subset the data:
  - `filterFile` subset to individual files/samples.
  - `filterRtime` restrict to specific retention time window.
  - `filterMz` restrict to m/z range.
  - `filterMsLevel` subset to certain MS level(s).

- Example: extract all spectra measured between 180 and 181 seconds. Using the %>% (pipe) operator to avoid nested function calls.

```
## Get all spectra measured between 180 and 181 seconds
## Use %>% for better readability
sps <- data %>%
    filterRt(rt = c(180, 181)) %>%
    spectra
```

```
## How many spectra?
length(sps)
```

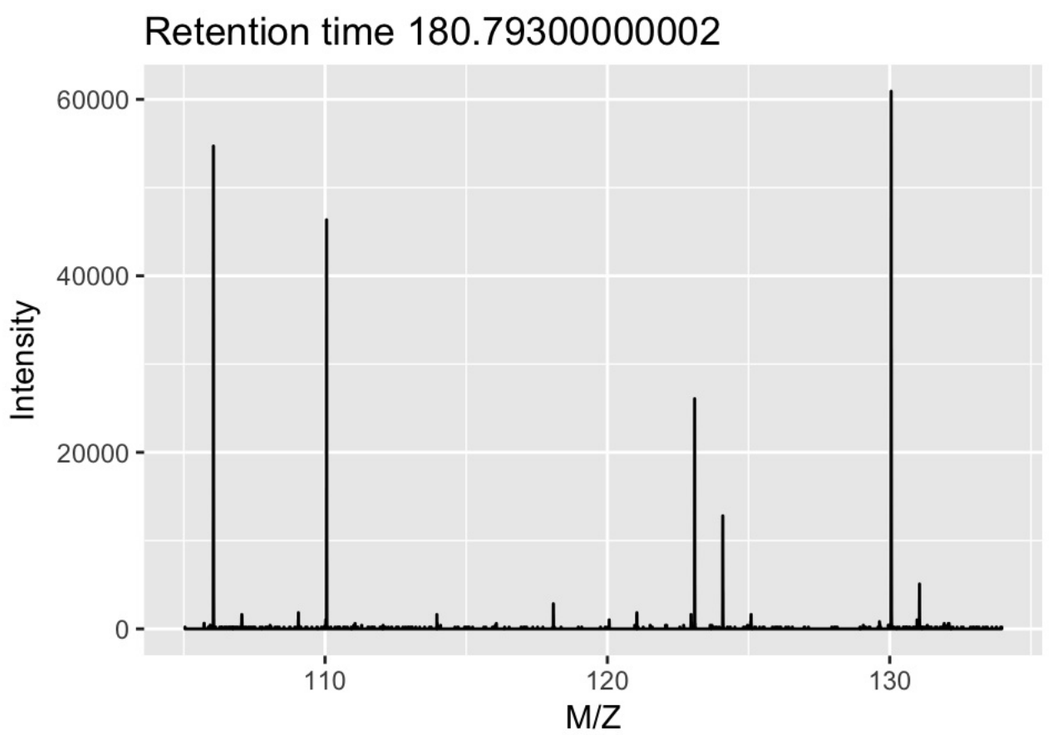
```
## [1] 6
```

```
## From which file?
sapply(sps, fromFile)
```

```
## F1.S646 F1.S647 F1.S648 F2.S646 F2.S647 F2.S648
##       1       1       1       2       2       2
```

- Example: plot the data from the last spectrum

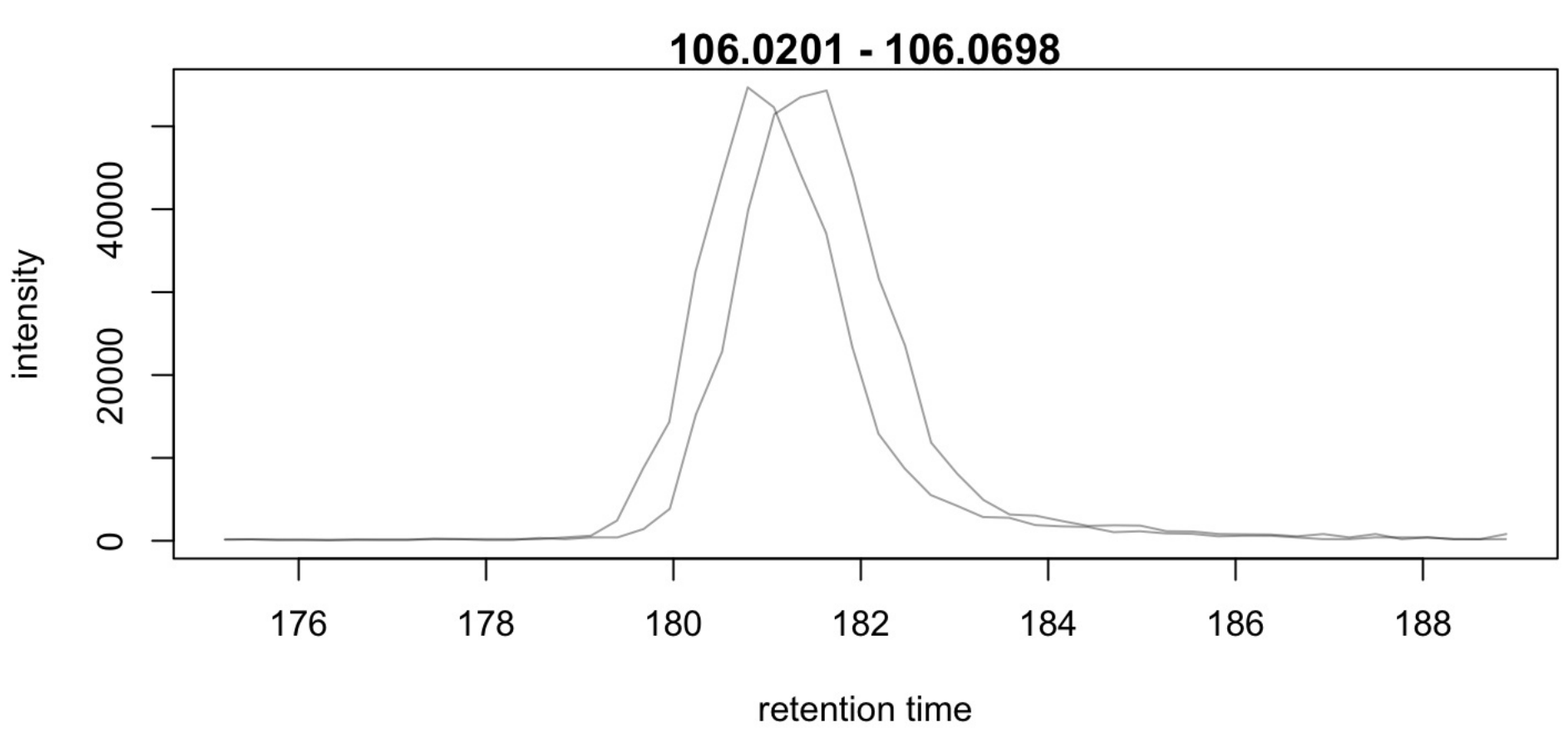
```
plot(sps[[6]])
```



- But how to get chromatographic data?

- `chromatogram`: extract chromatographic data.
- Example: XIC for Serine (m/z of [M+H]+ adduct 106.0455).

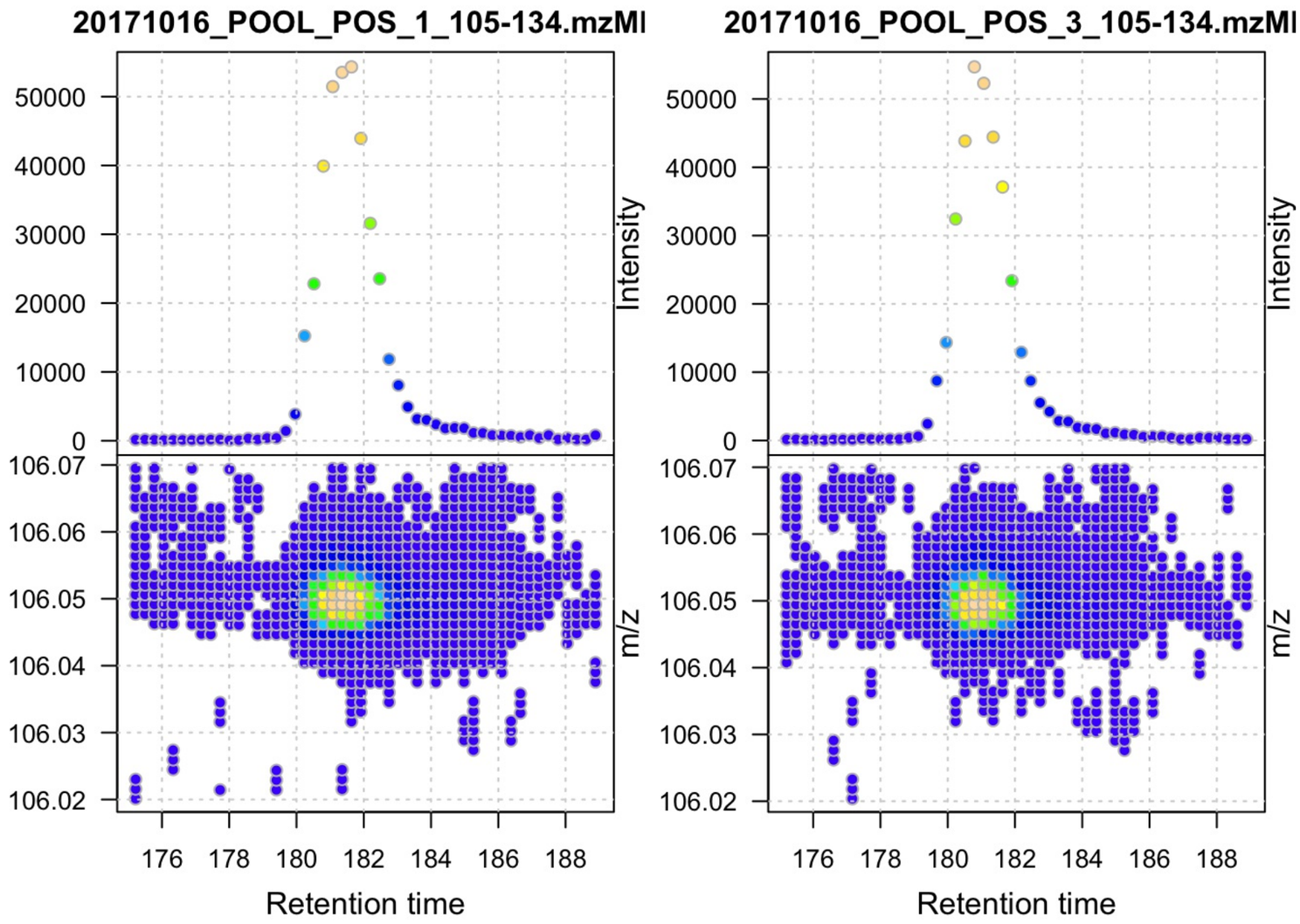
```
data %>%
    filterRt(rt = c(175, 189)) %>%
    filterMz(mz = c(106.02, 106.07)) %>%
    chromatogram(aggregationFun = "max") %>%
    plot()
```



Centroiding of profile MS data

- *centroiding* is the process in which mass peaks are reduced to a single, representative signal, their centroids.
- `xcms`, specifically *centWave* was designed for centroided data.
- `MSnbase` provides basic tools to perform MS data smoothing and centroiding: `smooth` and `pickPeaks`.
- Example: show the combined m/z, rt and intensity data for Serine.

```
data %>%
    filterRt(rt = c(175, 189)) %>%
    filterMz(mz = c(106.02, 106.07)) %>%
    plot(type = "XIC")
```

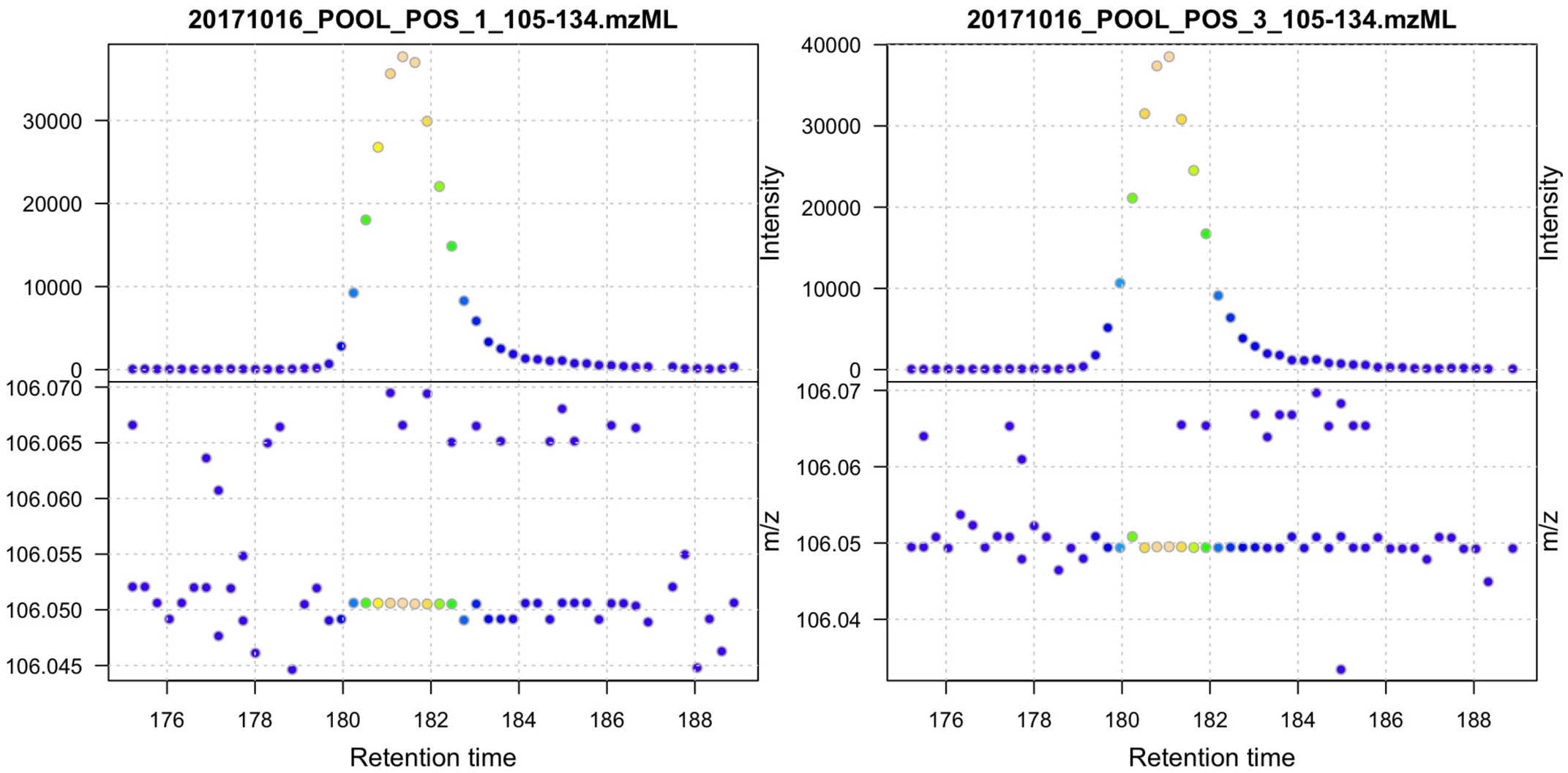


- plot `type = "XIC"` creates a combined chromatographic and *map* visualization of the data.

- Example: smooth data with Savitzky-Golay filter followed by a centroiding that simply reports the maximum signal for each mass peak in each spectrum. See `?pickPeaks` for more advanced options.

```
## Smooth the signal, then do a simple peak picking.
data_cent <- data %>%
    smooth(method = "SavitzkyGolay", halfWindowSize = 6) %>%
    pickPeaks()
```

```
## Plot the centroided data for Serine
data_cent %>%
    filterRt(rt = c(175, 189)) %>%
    filterMz(mz = c(106.02, 106.07)) %>%
    plot(type = "XIC")
```


20171016_POOL_POS_1_105-134.mzML
20171016_POOL_POS_3_105-134.mzML
Intensity
m/z
Retention time
30000
20000
10000
0
106.070
106.065
106.060
106.055
106.050
106.045
40000
106.07
106.06
106.05
106.04
176
178
180
182
184
186
188

- Note: since data is not available in memory, data smoothing and centroiding is applied *on-the-fly* each time m/z or intensity values are accessed.
- To make changes persistent: export and re-read the data.

```
## Write the centroided data to files with the same names in the current
## directory
fls_new <- basename(fileNames(data))
writeMSData(data_cent, file = fls_new)

## Read the centroided data.
data_cent <- readMSData(fls_new, pdata = new("NAnnotatedDataFrame", pd),
                        mode = "onDisk")
```

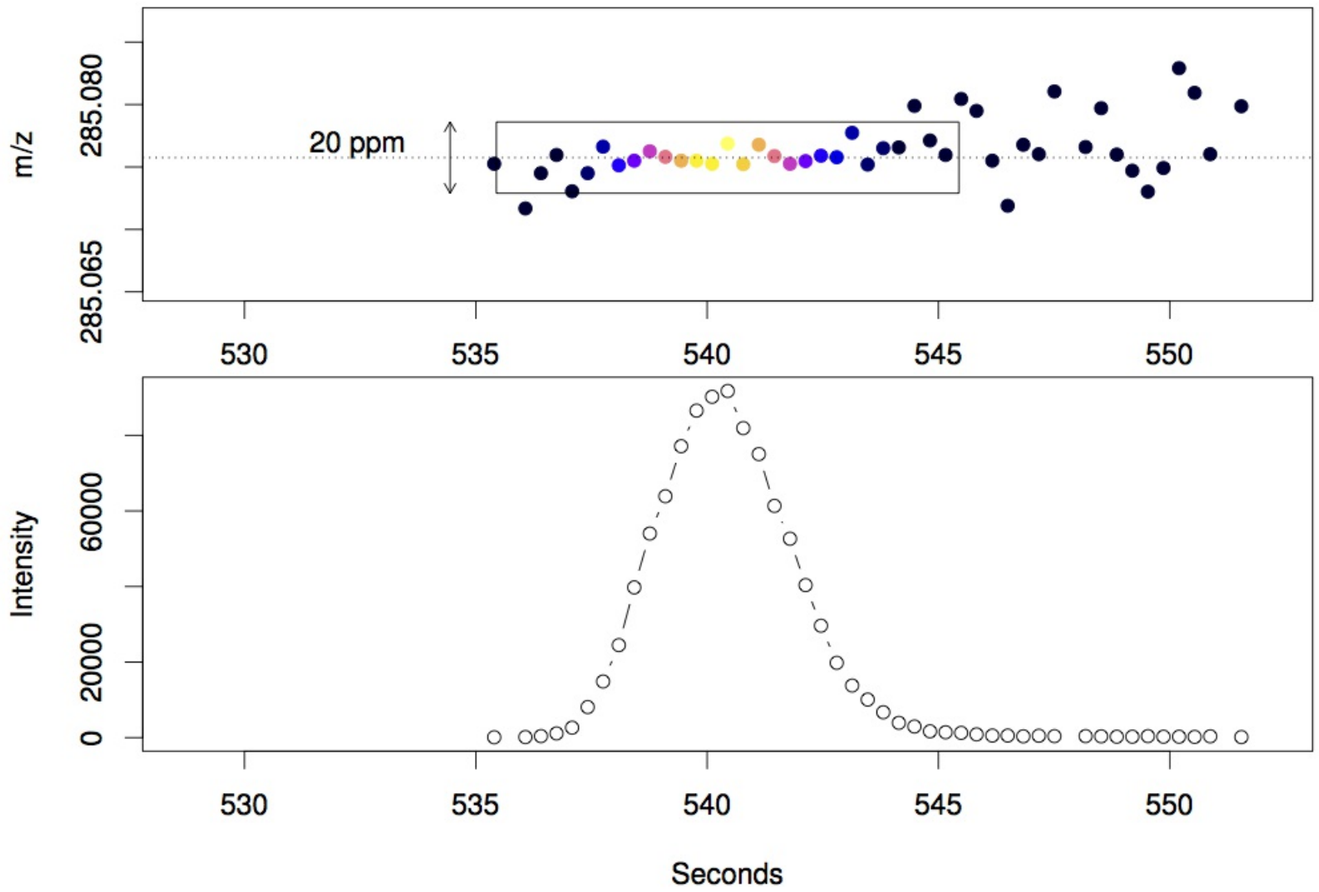
LC-MS data preprocessing

Chromatographic peak detection

- Aim: identify chromatographic peaks in the data.
- Function: `findChromPeaks`.
- Available methods:
  - *matchedFilter* (`MatchedFilterParam`) [Smith *Anal. chem.* 2006].
  - *centWave* (`CentWaveParam`) [Tautenhahn *BMC Bioinformatics* 2008].
  - *massifquant* (`MassifquantParam` [Conley *Bioinformatics* 2014].

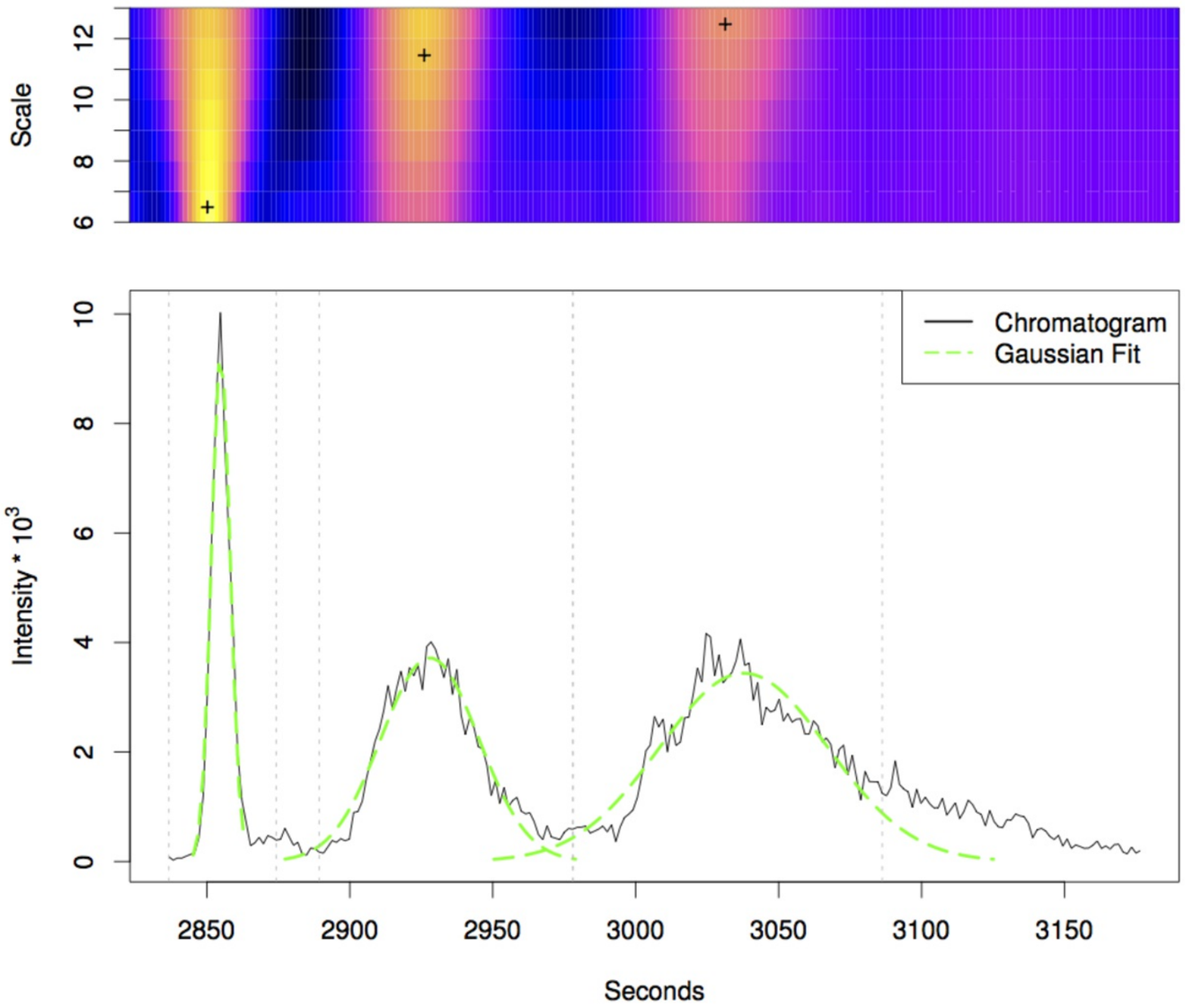
centWave

- First step: identify regions of interest.



[Tautenhahn *BMC Bionformatics* 2008]

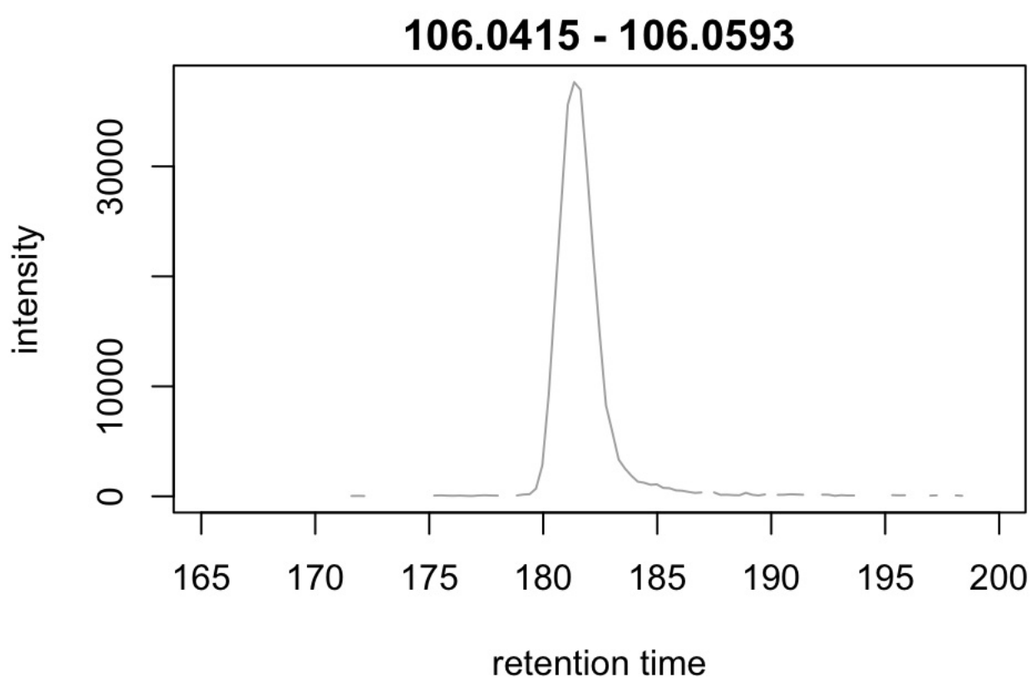
- 2nd: peak detection in these regions using continuous wavelet transform.



[Tautenhahn *BMC Bionformatics* 2008]

- Crucial centWave parameters: `peakwidth`, `ppm`; list all with `?CentWaveParam`.
- `peakwidth`: minimal and maximal expected peak width.
- Example: extract chromatographic data for Serine.

```
srn_chr <- chromatogram(data_cent, rt = c(165, 200),
                        mz = c(106.03, 106.06),
                        aggregationFun = "max")[1, 1]
plot(srn_chr)
```



- **New**: peak detection on `Chromatogram` objects.
- Perform peak detection using default centWave parameters in that data.

```
cwp <- CentWaveParam()
findChromPeaks(srn_chr, param = cwp)
```

```
## Warning in peaksWithCentWave(int = c(F1.S592 = NA, F1.S593 = NA, F1.S594 =
## NA, : No peaks found!
```

- **What went wrong?** What's the default for `peakwidth`?

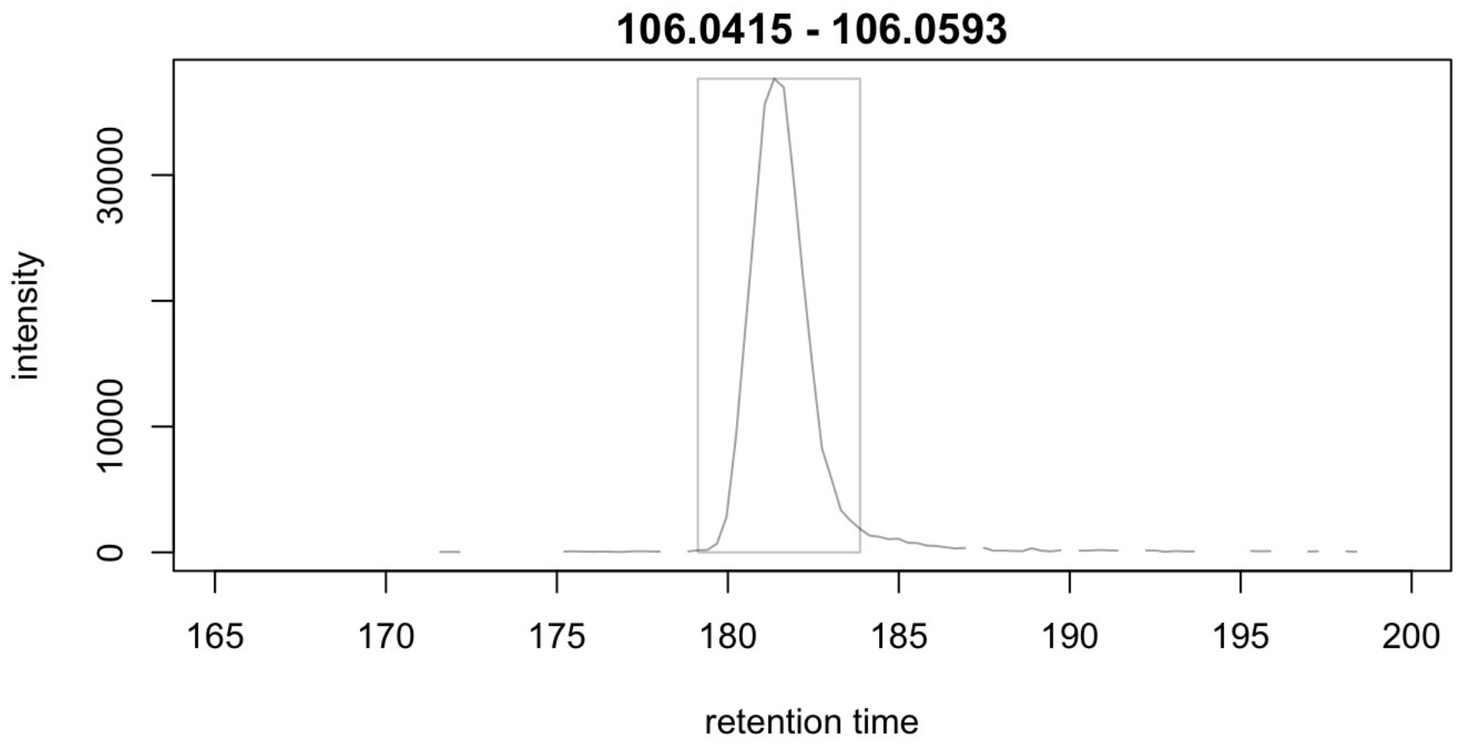
```
peakwidth(cwp)
```

```
## [1] 20 50
```

- Default for `peakwidth` does not match the current data.

- Reduce `peakwidth` and run peak detection again.

```
peakwidth(cwp) <- c(2, 10)
pks <- findChromPeaks(srn_chr, param = cwp)
## Plot the data and higlight identified peak area
plot(srn_chr)
rect(pks[, "rtmin"], 0, pks[, "rtmax"], pks[, "maxo"], border = "#00000040")
```

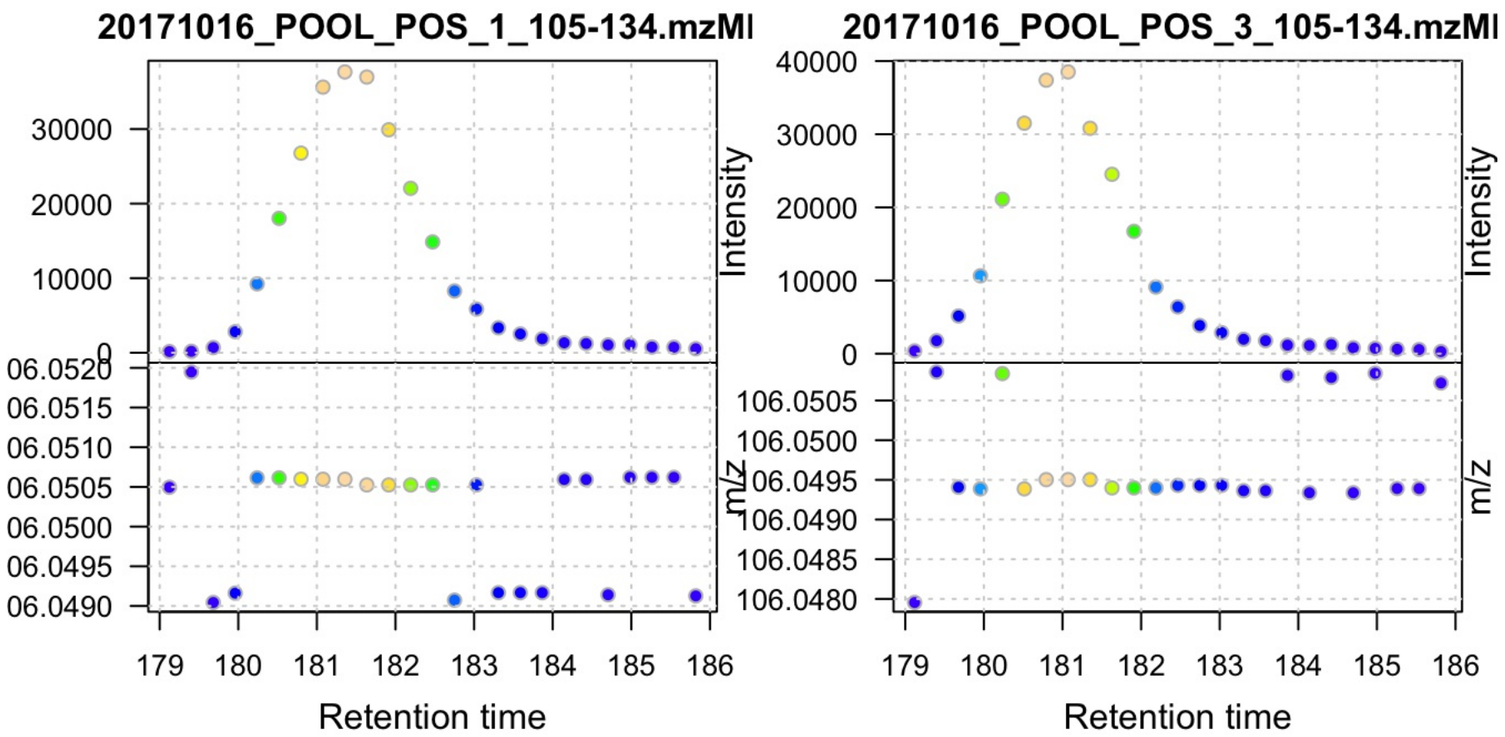


- Ideally check settings on more known compounds.

- `ppm`: maximal allowed scattering of m/z values for one ion.
- Example: evaluate the m/z scattering of the signal for Serine.

```
## Restrict the data to signal from Serine
srn <- data_cent %>%
    filterRt(rt = c(179, 186)) %>%
    filterMz(mz = c(106.04, 106.06))

## Plot the data
plot(srn, type = "XIC")
```



- Example: calculate the difference of m/z values between consecutive scans.

```
## Extract mz values for Serine from first file
srn_mz <- unlist(mz(filterFile(srn, 1)))
## The difference between m/z values from consecutive scans in ppm
diff(srn_mz) * 1e6 / mean(srn_mz)
```

```
##       F1.S643       F1.S644       F1.S645       F1.S646       F1.S647
##  13.695973646 -27.391665930   1.112565444  13.695804399   0.000000000
##       F1.S648       F1.S649       F1.S650       F1.S651       F1.S652
##  -0.158840806   0.000000000   0.000000000  -0.682098923   0.000000000
##       F1.S653       F1.S654       F1.S655       F1.S656       F1.S657
##   0.000000000   0.007189239 -13.695795336  13.695795336 -12.807200180
##       F1.S658       F1.S659       F1.S660       F1.S661       F1.S662
##   0.000000000   0.000000000  13.443799681   0.000000000 -13.695795190
##       F1.S663       F1.S664       F1.S665       F1.S666
##  13.957010392   0.000000000   0.000000000 -14.085629933
```

- This should be performed ideally on more compounds.
- `ppm`: large enough to capture the full chromatographic peak.

- Perform chromatographic peak detection with our data set-specific settings.

```
## Perform peak detection
ppm(cwp) <- 30
data_cent <- findChromPeaks(data_cent, param = cwp)
```

- Result: `XCMSnExp` object extends the `OnDiskMSnExp`, contains preprocessing results **and** enables data access as described above.

- Use `chromPeaks` to access the peak detection results.

```
head(chromPeaks(data_cent), n = 5)
```

```
##             mz    mzmin    mzmax     rt  rtmin  rtmax     into     intb
## [1,] 111.0443 111.0431 111.0476 25.670 24.554 27.065 574.3303 562.2062
## [2,] 129.0541 129.0522 129.0553 25.391 24.275 27.065 806.1325 770.1417
## [3,] 114.0727 114.0715 114.0731 26.786 25.670 28.460 767.0744 764.5634
## [4,] 111.0057 111.0044 111.0073 28.739 27.623 29.297 397.1809 395.7859
## [5,] 127.0387 127.0378 127.0410 28.739 27.902 29.576 227.0865 225.6915
##          maxo  sn sample is_filled
## [1,] 578.7692  30      1         0
## [2,] 733.1538  21      1         0
## [3,] 498.2657 497      1         0
## [4,] 400.4895 399      1         0
## [5,] 231.5594 231      1         0
```

Alignment - in short

- Aim: adjust shifts in retention times between samples.
- Function: `adjustRtime`.
- Available methods:
    - *obiwarp* (`ObiwarpParam`) [Prince *Anal. chem.* 2006]: warps the (full) data to a reference sample.
- *peakGroups* (`PeakGroupsParam`) [Smith *Anal. chem.* 2006]:
    - align spectra from different samples based on *hook* peaks.
    - Need to define the hook peaks first: peaks present in most/all samples.

- Example: perform a peak grouping to define potential hook peaks and align the samples based on these.
- *Note:* details on initial peak grouping provided in the next section.

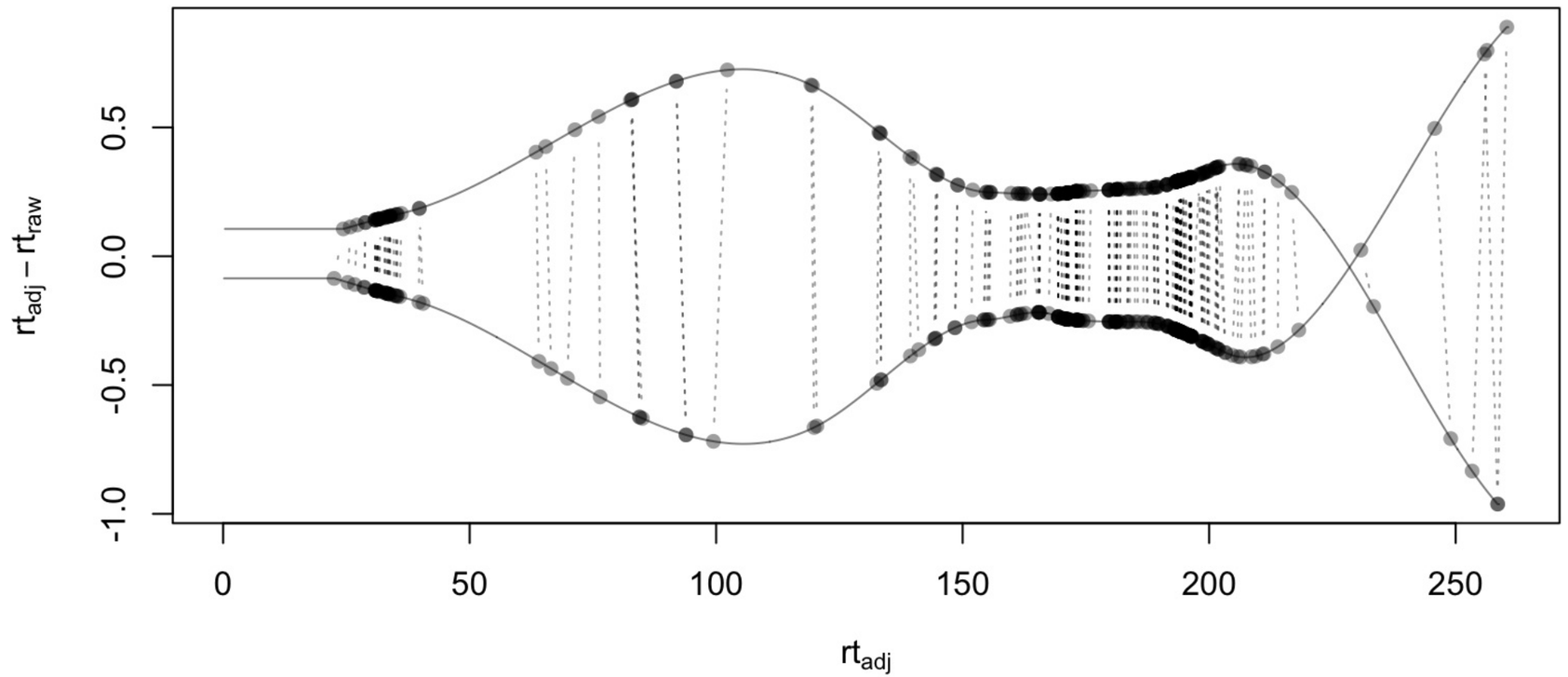
```
## Define the settings for the initial peak grouping
pdp <- PeakDensityParam(sampleGroups = data_cent$group, bw = 1.8,
                        minFraction = 1, binSize = 0.02)
data_cent <- groupChromPeaks(data_cent, pdp)
```

- Align the samples.

```
## Define settings for the alignment
pgp <- PeakGroupsParam(minFraction = 1, span = 0.6)
data_cent <- adjustRtime(data_cent, param = pgp)
```

- Inspect difference between raw and adjusted retention times.

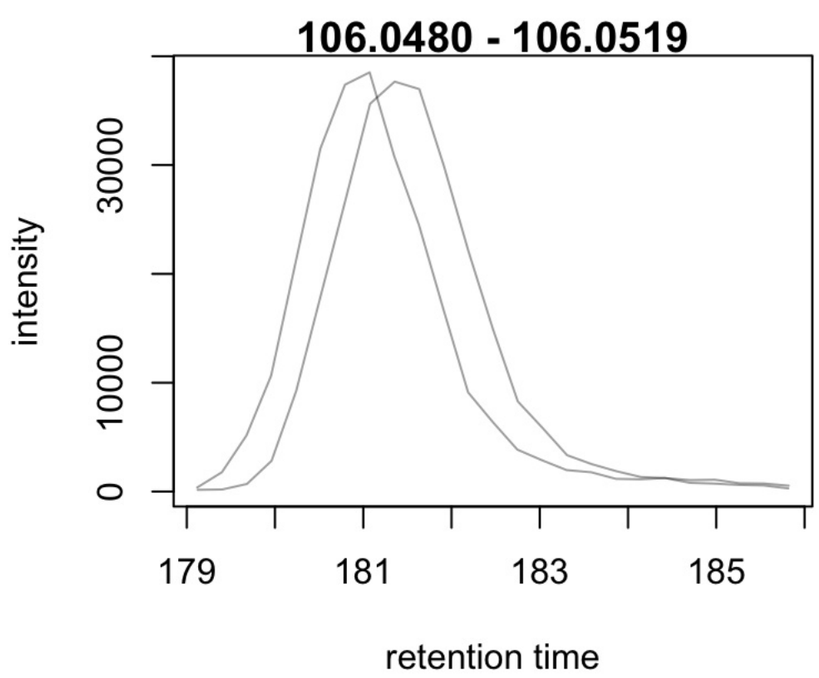
```
plotAdjustedRtime(data_cent)
```

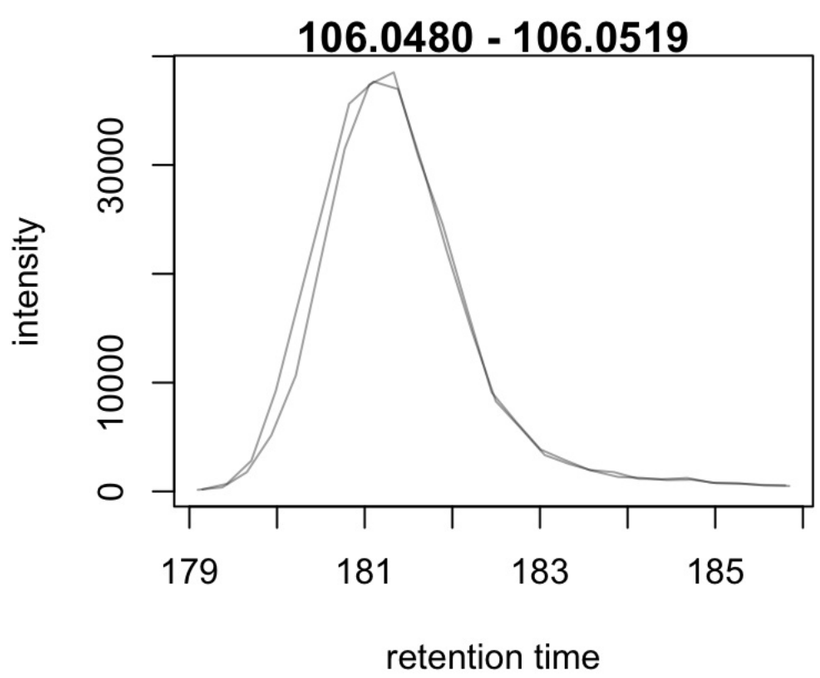


- Difference between raw and adjusted retention times resonable.
- Hook peaks along the full retention time range.

- Plot BPC before and after alignment.
- Plot XIC of known compounds before and after alignment.

```
## Use adjustedRtime parameter to access raw/adjusted retention times
par(mfrow = c(1, 2), mar = c(4, 4.5, 0.9, 0.5))
plot(chromatogram(data_cent, mz = c(106.04, 106.06),
                  rt = c(179, 186), adjustedRtime = FALSE))
plot(chromatogram(data_cent, mz = c(106.04, 106.06),
                  rt = c(179, 186)))
```





Correspondence

- Aim: group signal (peaks) from the same ion across samples.
- Function: `groupChromPeaks`.
- Methods available:
  - *peak density* (`PeakDensityParam`) [Smith *Anal. chem.* 2006].
  - *nearest* (`NearestPeaksParam`) [Katajamaa *Bioinformatics* 2006].

peak density

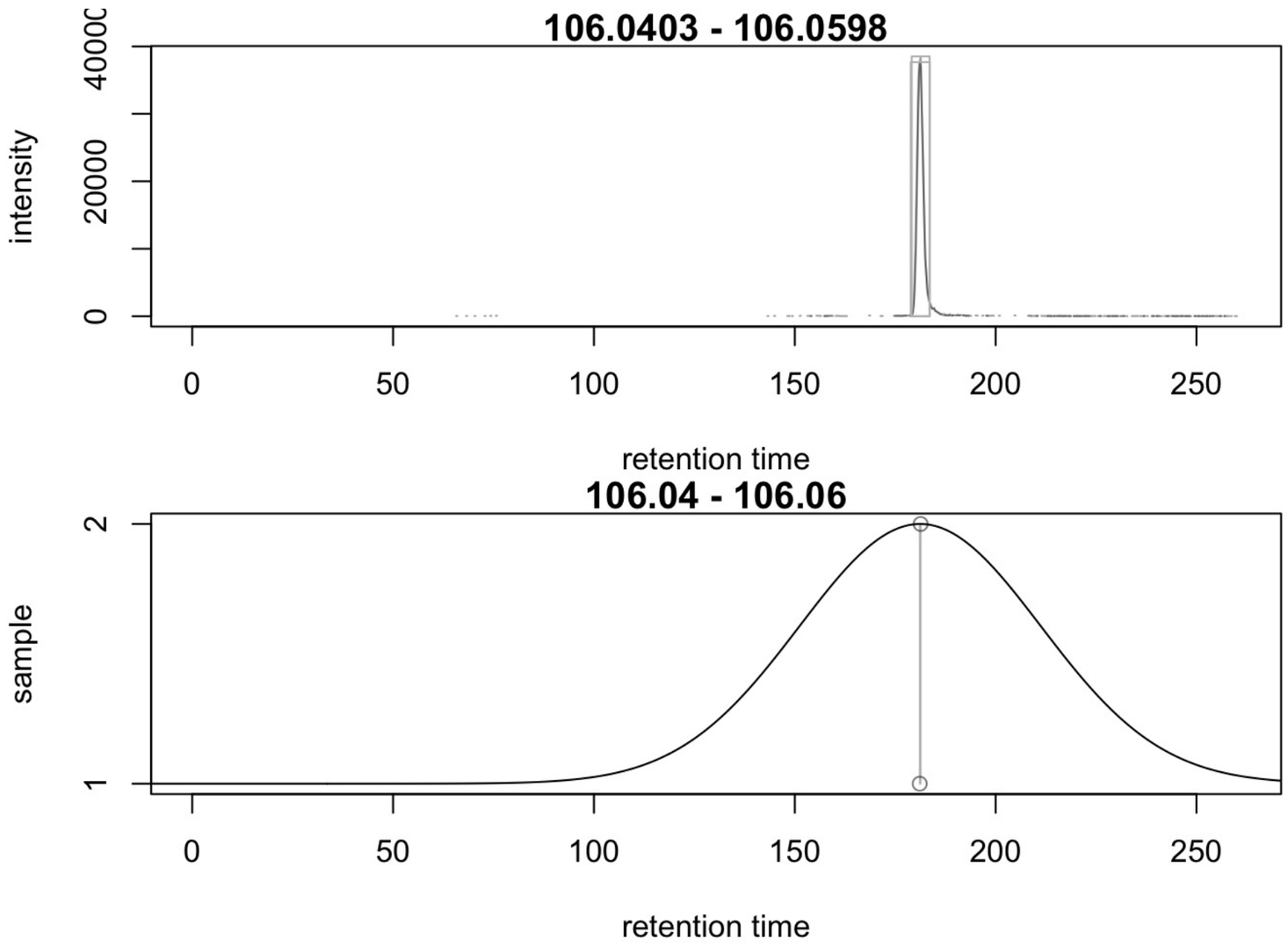
- Iterate through MS data slices along m/z
- Group chromatographic in each slice if peaks (from same or different samples) are close in retention time.
- Distribution of peaks along retention time axis is used to define which peaks to group.
- `plotChromPeakDensity`: plot distribution of identified peaks along rt for a given m/z slice.

- Example:
  - Plot data for the m/z slice containing the Serine peak.
  - Use `plotChromPeakDensity` to simulate a correspondence analysis in the same slice.

```
## Plot the BPC for the m/z slice containing serine
par(mfrow = c(2, 1), mar = c(4, 4.3, 1, 0.5))
plot(chromatogram(data_cent, mz = c(106.04, 106.06), aggregationFun = "max"))
highlightChromPeaks(data_cent, mz = c(106.04, 106.06),
                    whichPeaks = "apex_within")

## Get default parameters for the grouping
pdp <- PeakDensityParam(sampleGroups = data_cent$group)

## Dry-run correspondence and show the results.
plotChromPeakDensity(data_cent, mz = c(106.04, 106.06),
                     type = "apex_within", param = pdp)
```



- Points are peaks per sample;
- black line: peak density distribution;
- grey rectangles: grouped peaks (features).

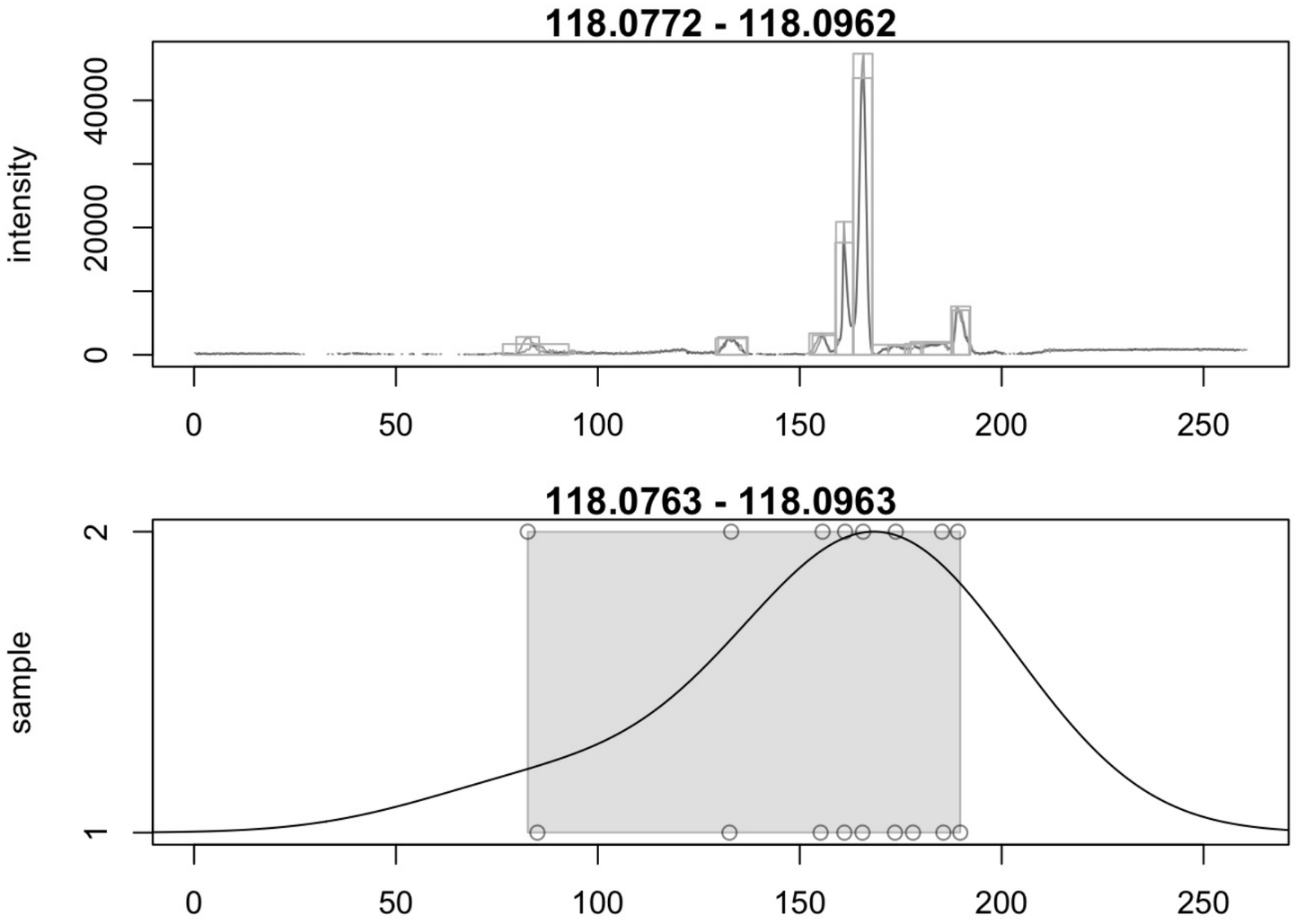
- Parameters:
  - `binSize`: m/z width of the data slice in which peaks are grouped.
  - `bw` defines the smoothness of the density function.
  - `maxFeatures`: maximum number of features to be defined in one bin.
  - `minFraction`: minimum proportion of samples (of one group!) for which a peak has to be present.
  - `minSamples`: minimum number of samples a peak has to be present.

- Parameters `minFraction` and `minSamples` depend on experimental layout!
- `binSize` should be small enough to avoid peaks from different ions measured at similar retention times to be grouped together.
- `bw` is the most important parameter.

- Test default settings for a slice containing ions with similar m/z and rt: isomers Betaine and Valine ([M+H]+ m/z 118.08625).

```
par(mfrow = c(2, 1), mar = c(3, 4.3, 1, 1))
## Plot the chromatogram for an m/z slice containing Betaine and Valine
mzr <- 118.08625 + c(-0.01, 0.01)
plot(chromatogram(data_cent, mz = mzr, aggregationFun = "max"))
highlightChromPeaks(data_cent, mz = mzr, whichPeaks = "apex_within")

## Correspondence in that slice using default settings
pdp <- PeakDensityParam(sampleGroups = data_cent$group)
plotChromPeakDensity(data_cent, mz = mzr, param = pdp, type = "apex_within")
```

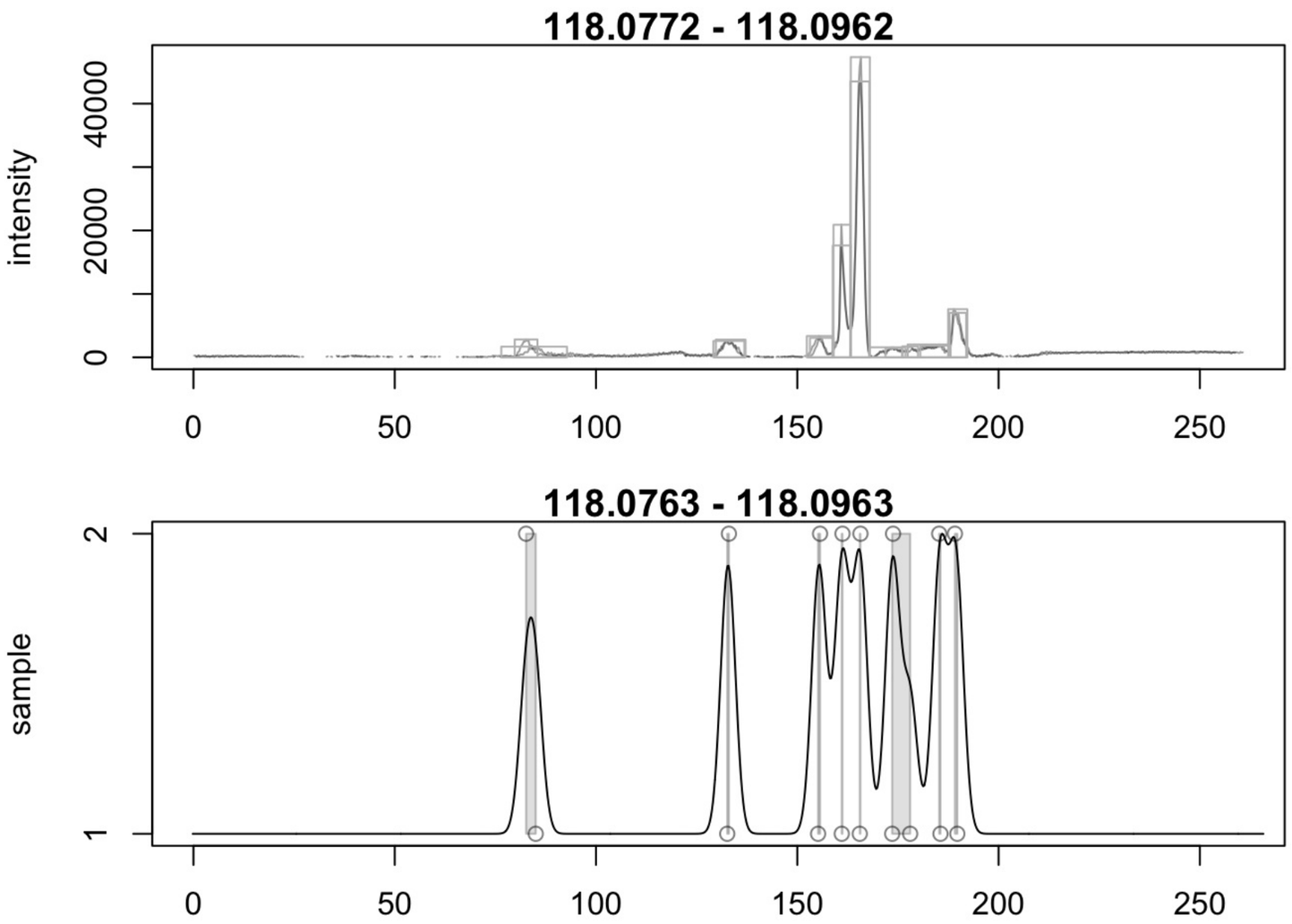


- **Correspondence failed**: all peaks grouped into one feature!
- Default for `bw` (`30`) too large for present data set.

- `plotChromPeakDensity` allows to evaluate and tune settings on data subsets.
- Test smaller `bw` (`1.8`) on the same slice.

```
par(mfrow = c(2, 1), mar = c(3, 4.3, 1, 1))
## Plot the chromatogram for an m/z slice containing Betaine and Valine
mzr <- 118.08625 + c(-0.01, 0.01)
plot(chromatogram(data_cent, mz = mzr, aggregationFun = "max"))
highlightChromPeaks(data_cent, mz = mzr, whichPeaks = "apex_within")

## Reducing the bandwidth
pdp <- PeakDensityParam(sampleGroups = data_cent$group, bw = 1.8)
plotChromPeakDensity(data_cent, mz = mzr, param = pdp, type = "apex_within")
```



- Reducing the bw enabled grouping of isomers into different features.

- Perform the correspondence analysis with tuned settings.

```
pdp <- PeakDensityParam(sampleGroups = data_cent$group, bw = 1.8,
                        minFraction = 0.4, binSize = 0.02)

## Perform the correspondence analysis
data_cent <- groupChromPeaks(data_cent, param = pdp)
```

- Evaluate results after correspondence: `plotChromPeakDensity` with `simulate = FALSE` shows the actual results from the correspondence.

- Feature definitions are stored within the `XCMSnExp` object, can be accessed with `featureDefinitions`.

- Use `featureValues` to access the features' abundance estimates.

```
## feature intensity matrix
fmat <- featureValues(data_cent, value = "into", method = "maxint")
head(fmat)
```

```
##       20171016_POOL_POS_1_105-134.mzML 20171016_POOL_POS_3_105-134.mzML
## FT001                        3159.7569                         3093.752
## FT002                        4762.3987                               NA
## FT003                         744.8752                         1033.232
## FT004                       20211.2634                        15839.550
## FT005                       10220.8762                        10837.710
## FT006                       19653.1073                        31816.844
```

- `featureValues` parameters:
  - `value`: name of the column in `chromPeaks` that should be returned.
  - `method`: for features with multiple peaks in one sample: from which peak should the value be returned?

Missing values

- Peak detection may have failed in one sample.
- Ion is not present in a sample.
- `fillChromPeaks` allows to *fill-in* signal for missing peaks from the feature area (defined by the median rt and mz of all peaks assigned to the feature).
- `fillChromPeaks` Parameters:
  - `expandMz`, `expandRt`: expands the region from which signal is integrated in m/z or rt dimension. A value of 0 means no expansion, 1 means the region is grown by half of the feature's m/z width on both sides.
  - `ppm`: expand the m/z width by a m/z dependent value.

- Example: evaluate number of missing peaks and use `fillChromPeaks` to retrieve a signal for them from the raw files.

```
## Number of missing values
sum(is.na(fmat))
```

```
## [1] 137
```

```
## Define the settings for the fill-in of missing peaks
fpp <- FillChromPeaksParam(expandMz = 0.5, expandRt = 0.5, ppm = 20)
data_cent <- fillChromPeaks(data_cent, param = fpp)

## How many missing values after
sum(is.na(featureValues(data_cent)))
```

```
## [1] 4
```

Summary

- The new data objects and functions aim to:
  - simplify data access and inspection of results
  - facilitate data set-dependent definition of algorithm parameters.
- More work to come for the analysis of chromatographic data (SRM/MRM) and eventually for data normalization.
- **Don’t blindly use default parameters!**

Acknowledgments

- Jan Stanstrup (University of Copenhagen, Denmark)
- Laurent Gatto (University of Cambridge, UK); `MSnbase`, `mzR`.
- Steffen Neumann (IPB Halle, Germany); `xcms`, `mzR`
- **YOU for your attention!**

https://github.com/jotsetung/metabolomics2018