PLAQUE PICKER Example

Load needed libraries

```
suppressPackageStartupMessages({
  library(PlaquePicker)
  library(MALDIquant)         # general MS functions
  library(MALDIquantForeign) # for import of imzML data
  library(tidyverse)          # general data science tools

  # for the plotting of Venn-Diagrams we will be using the package Vennerable
  # which is not in the CRAN repository and has some dependencies to the
  # BioConductor repository. To install Vennerable and its dependencies
  # use the following commands:
  #
  # if (!requireNamespace("BiocManager", quietly = TRUE))
  #     install.packages("BiocManager")
  # BiocManager::install(version = "3.11")
  # BiocManager::install("RBGL")
  # BiocManager::install("graph")
  # devtools::install_github("js229/Vennerable")
  library(Vennerable)})
```

This package comes with a preprared set of ion images that can directly be used by the plaquePicker-function. Nevertheless, here we show how we prepared this example data:

```
# unzip spectra
unzip("data-raw/NLGF67w_mouse1_rep1.zip",
      files = "data-raw/unzipped/")

# read spectra
spec <- importImzMl("data-raw/unzipped/NLGF_Prot_NLGF1.imzML",
                    verbose = FALSE)

# tidy up
unlink("data-raw/unzipped/",
       force = TRUE,
       recursive = TRUE)
```

Preprocess spectra using MALDIquant functions: Normalize, smooth, remove baseline. Compute average spectrum and plot it to get an overview of the dataset.

```
spec <- calibrateIntensity(spec,
                           method = "TIC")

# small halfWindowSize needed as number of points
# per spectra reduced for smaller example datasets
spec <- smoothIntensity(spec,
                        method = "SavitzkyGolay",
                        halfWindowSize = 2)
```

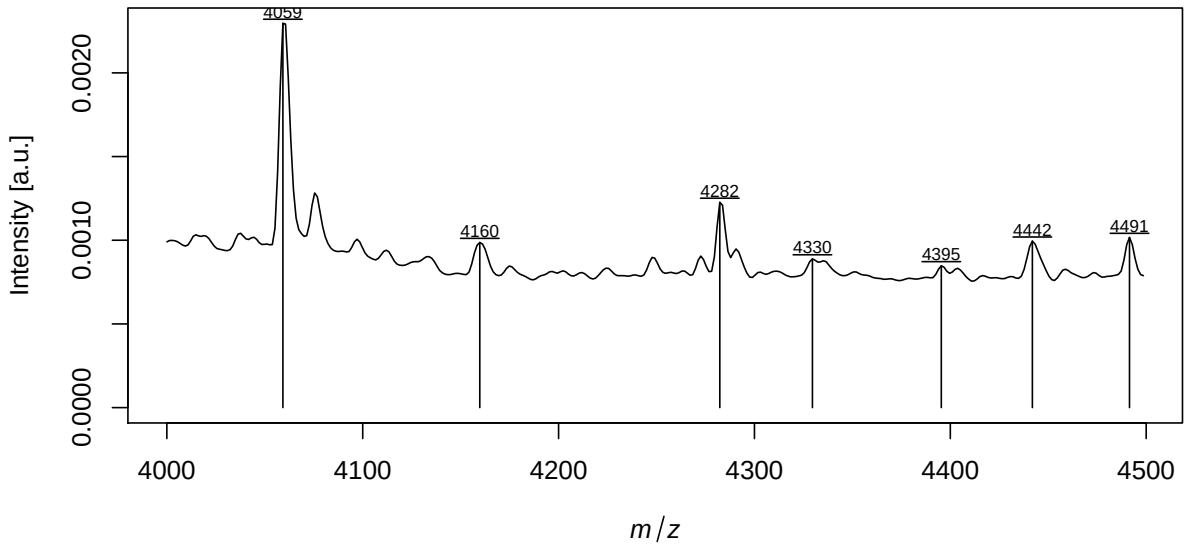
```
spec <- removeBaseline(spec,
                       method = "TopHat")
```

```
# load example data as prepared above
spec <- NLGF67w_mouse1_rep1_spec
coord <- NLGF67w_mouse1_rep1_coord

avgSpec <- averageMassSpectra(spec)

# shorten filepath (for plotting reasons only)
avgSpec@metaData$file <- basename(avgSpec@metaData$file)

plot(avgSpec, ylab = "Intensity [a.u.]")
lines(detectPeaks(avgSpec))
labelPeaks(detectPeaks(avgSpec),
           digits = 0)
```



We observe a strong peak at m/z 4059 (Ab1-38Arc), and two smaller peaks at m/z 4159 (Ab1-39Arc) and 4442 (Ab1-42Arc). Note that as we computed the average spectrum for all spectra and the Abeta signals are only found in a small subset of spectra,this leads to underrepesentation of the signals in the average spectrum. Also, because of the lower resolution of the example data set, the peak maxima shifted slightly in comparision to those reported in the publication.

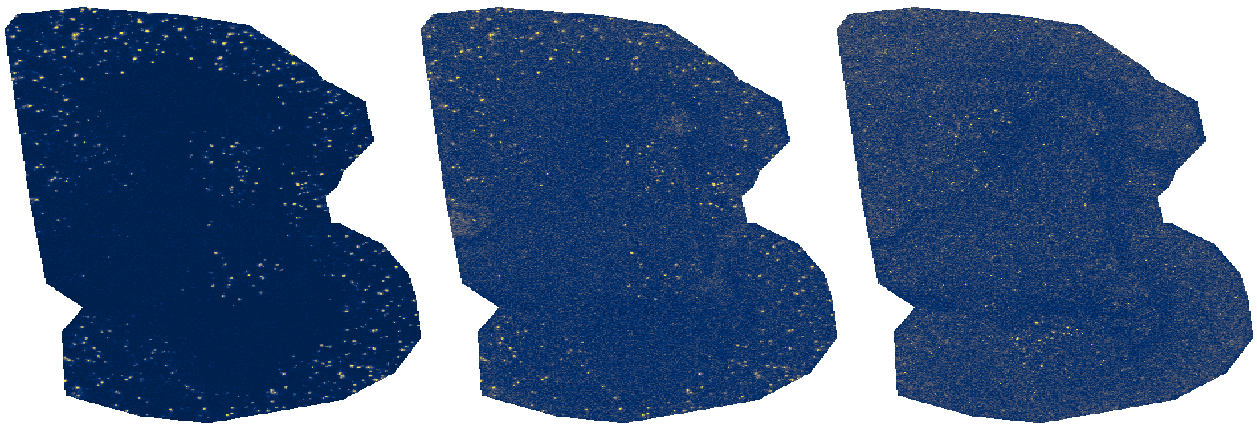
Next we extract ion images of interest.

```
ionImages <- msiSlices(spec,
                       center = c(4059,
                                  4160,
                                  4442),
                       tolerance = 5)
```

When we take a look at these ion images we can observe distinct accumulations of Abeta as plaques. Ab1-

38Arc and Ab1-39Arc are highly co-localized wheras Ab1-42Arc seems to also accumulate at other locations then the other two masses. Note that we applied quantile correction to the 99.95%-quantile to remove hotspots for better visualization. As mentioned above, the signals of interest are only found in a small subset of spectra (-> sparse-signals).

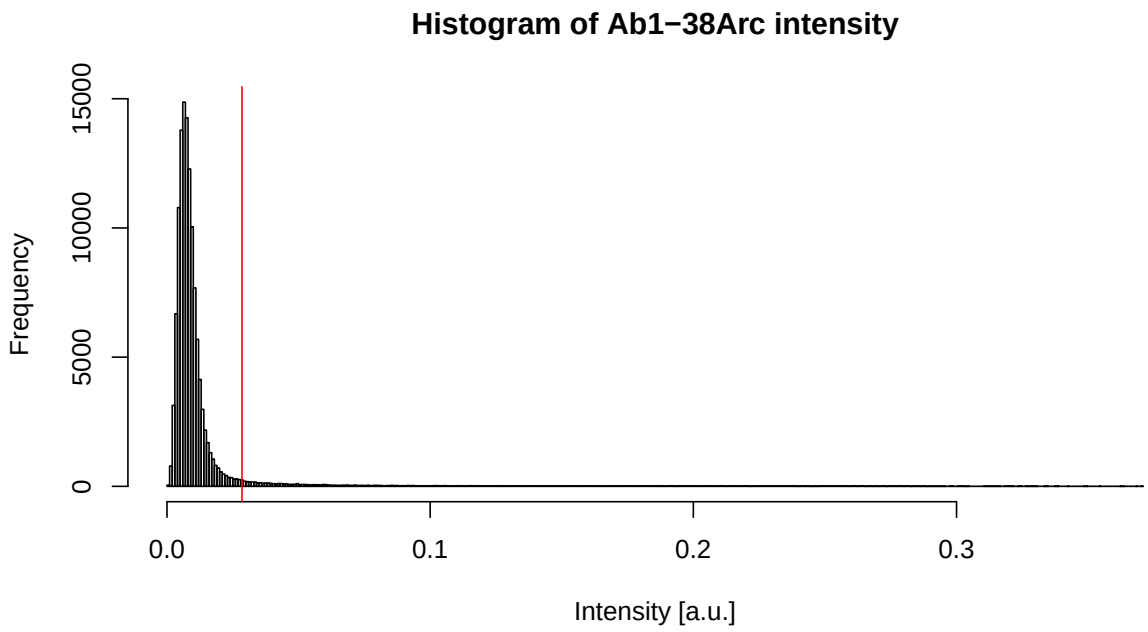
```
par(mfrow = c(1, 3), mar =c(0,0,0,0))
image(qcor(ionImages[,,1], 0.9995),
      col = viridis::cividis(30),
      asp = 1,
      axes = FALSE)
title("Ab1-38Arc", line = -2)
image(qcor(ionImages[,,2], 0.9995),
      col = viridis::cividis(30),
      asp = 1,
      axes = FALSE)
title("Ab1-39Arc", line = -2)
image(qcor(ionImages[,,3], 0.9995),
      col = viridis::cividis(30),
      asp = 1,
      axes = FALSE)
title("Ab1-42Arc", line = -2)
```



Next we take a look at the histogram of intensities for Ab1-38Arc. We observe a unimodal distribution. Most of the pixels have a intensity at the level of noise and there are only a few above that. We use the tpoint method to find a threshold.

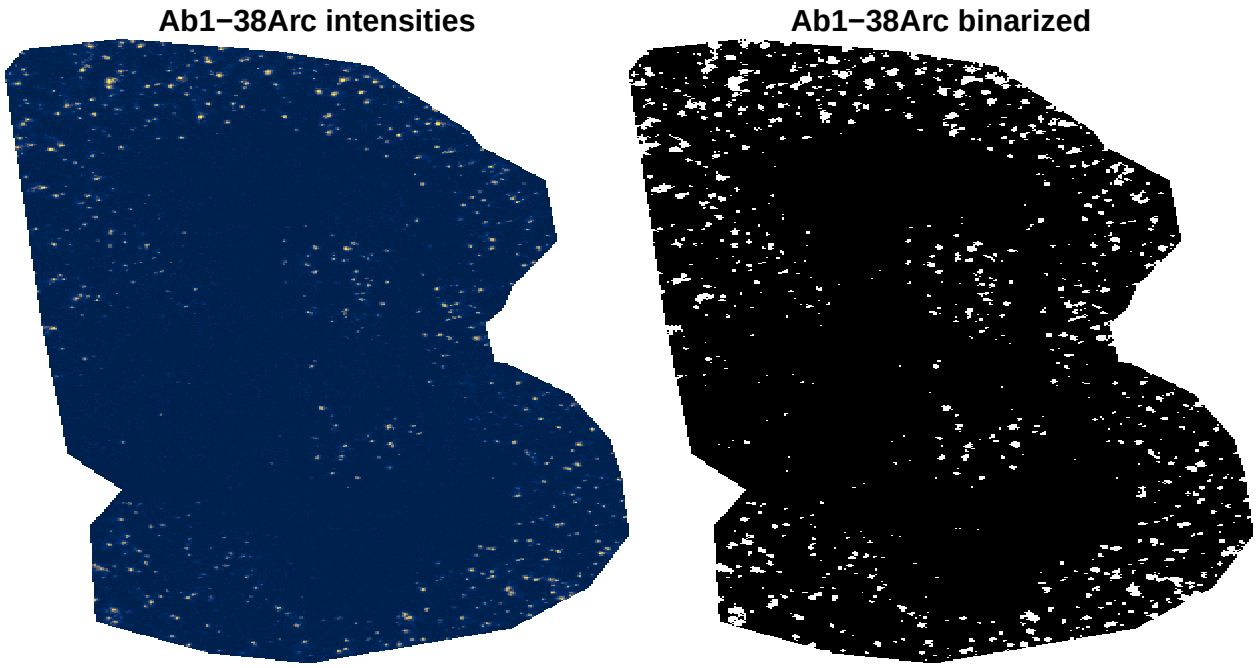
```
hist(ionImages[,,1],
     breaks = 300,
     xlab = "Intensity [a.u.]",
     main = "Histogram of Ab1-38Arc intensity")
thresh_Ab38 <- tpoint(ionImages[,,1])
abline(v=thresh_Ab38,
```

```
       col = "red")
```



If we apply this threshold to the corresponding ion image we get a binized image.

```
bin <- ifelse(test = thresh_Ab38 < as.vector(ionImages[,,1]),
              yes = 1,
              no = ifelse(is.na(as.vector(ionImages[,,1])),
                          yes = NA,
                          no = 0))
# rebuild the matrix
binMat <- matrix(bin,
                 nrow = dim(ionImages[,,1])[1],
                 ncol = dim(ionImages[,,1])[2])
par(mfrow = c(1, 2), mar = c(0,0,0,0))
image(qcor(ionImages[,,1], 0.9999),
      col = viridis::cividis(30),
      asp = 1,
      axes = FALSE)
title("Ab1-38Arc intensities", line = -1)
image(binMat,
      col = c("black", "white"),
      asp = 1,
      axes = FALSE)
title("Ab1-38Arc binarized", line = -1)
```



Using this image we could apply raster::clump to perform connected component labeling and assign each individual plaque an ID. Or we could first compute the binary pictures of all ion images of interest, combine them and then apply connected component labeling. Following the same principle as shown above we now apply the plaquePicker-function to the ion images. In additon to the ion images this function also needs the coordinates of the dataset so we have to also extract them.

```
coord <- coordinates(spec)
pp <- plaquePicker(ionImages = ionImages,
                   coord = coord)
```

```
##
##  threshold of mz 4059 based on t-point thresholding =  0.0285
```

```
## Loading required namespace: igraph
```

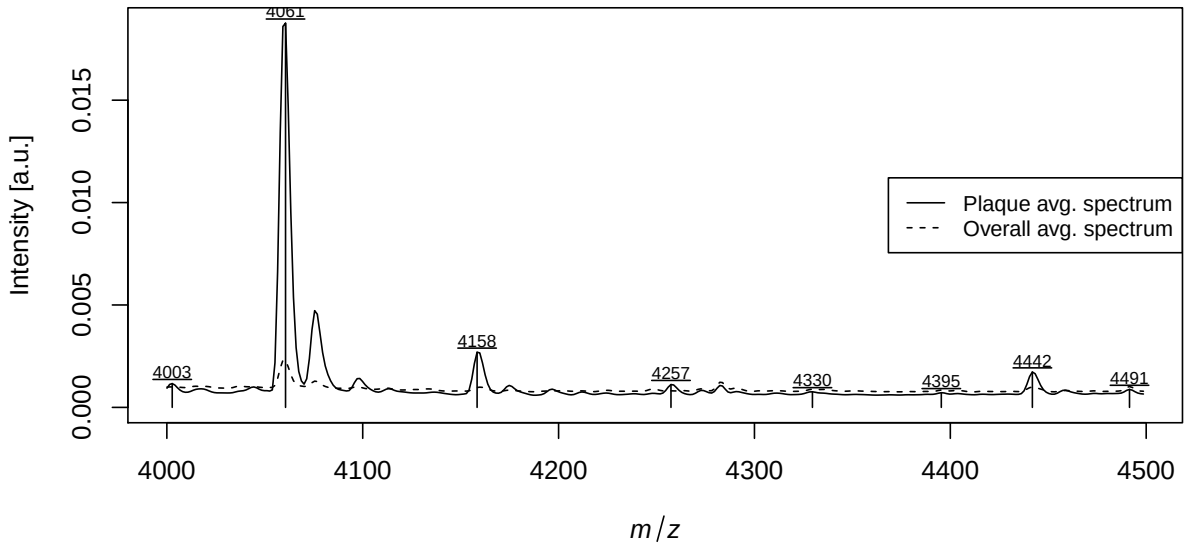
```
## extracting clump information...
##
##  threshold of mz 4160 based on t-point thresholding =  0.02185
## extracting clump information...
##
##  threshold of mz 4442 based on t-point thresholding =  0.0217
## extracting clump information...
## extracting unified clump information...
```

Now that we have the dataset structed in a way suitable for single-object analysis we can perform some basic tasks. For example, lets take all spectra assosiated with Abeta signals and caluclate an average spectrum of plaques and compare it to the average spectrum of the whole dataset we computed above.

```
plaqueAvg <- averageMassSpectra(spec[unlist(pp$unified$spectraIdx)])
# shorten filepath (for plotting reasons only)
plaqueAvg@metaData$file <- basename(plaqueAvg@metaData$file)

plot(plaqueAvg,
     ylab = "Intensity [a.u.]")
```

```
lines(avgSpec,
      lty=2)
labelPeaks(detectPeaks(plaqueAvg),
           digits = 0)
lines(detectPeaks(plaqueAvg))
legend("right",
       legend=c("Plaque avg. spectrum","Overall avg. spectrum"),
       lty=1:2,
       cex=0.8)
```



NLGF_Prot_NLGF1.ibd

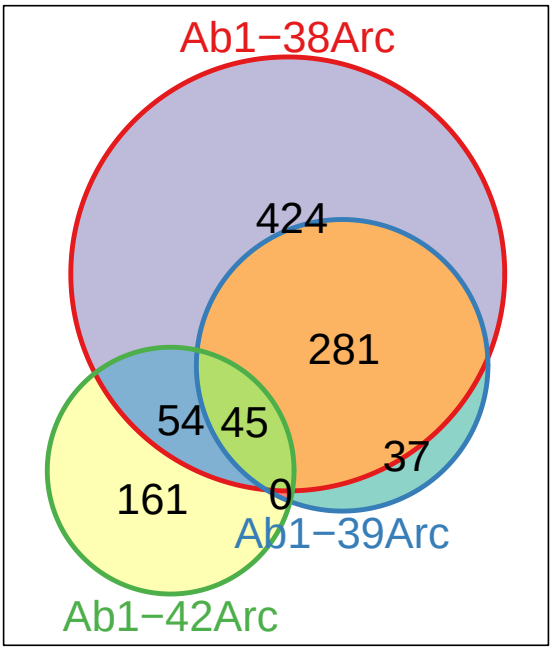
In addition to the unified results, the pp object has an entry containing the spectra indices of each individual object. Using the unified connected component labeled matrix (uniComp) in conjunction with the binary images for each ion image of interest we extract the plaques that co-localize with each other. This enables as to assess the plaques regarding their composition.

```
# extract matrix of unified plaque IDs
unified <- pp$unified$uniComp
# get mzValues of IonImages
mzVal <- attr(ionImages, "center")
PlaqueIDs_venn <- vector("list",
                         length = length(mzVal))
for(i in 1:length(mzVal)) {
  PlaqueIDs_venn[[i]] <- pp[[i]]$binMat * unified
  PlaqueIDs_venn[[i]] <- PlaqueIDs_venn[[i]] %>%
    as.vector() %>%
    unique() %>%
    na.omit() %>%
    sort() %>%
    .[-1]
}
names(PlaqueIDs_venn) <- c("Ab1-38Arc",
                           "Ab1-39Arc",
```

```
                                "Ab1-42Arc")
```

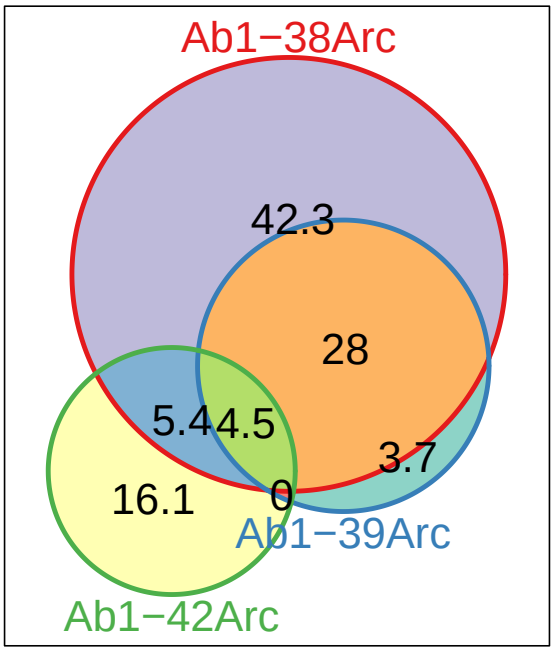
We can use this data to analysis the general qualitative composition of plaques using a venn diagramm. Instead of using the number of plaques as labels we could also use relative values:

```
v <- Venn(PlaqueIDs_venn)
plot(v)
```




```
# calculate relative values
v_rel <- v
v_rel@IndicatorWeight[,4] <-
  round(v_rel@IndicatorWeight[,4]/sum(v_rel@IndicatorWeight[,4]),3) * 100

plot(v_rel)
```



Now we can quantify the visual impression we had when we took a look at the ion images above: Ab1-38Arc and Ab1-39Arc are highly co-localized but there is also a large number of plaques composed only of Ab1-38Arc. Also for Ab1-42 we find a large population that is only composed of Ab1-42Arc.

```
sessionInfo()
```

```
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] Vennerable_3.1.0.9000  forcats_0.5.0          stringr_1.4.0
##  [4] dplyr_0.8.5            purrr_0.3.3            readr_1.3.1
##  [7] tidyr_1.0.2            tibble_2.1.3           ggplot2_3.3.0
## [10] tidyverse_1.3.0        MALDIquantForeign_0.12 MALDIquant_1.99.1
## [13] PlaquePicker_0.1.0
##
## loaded via a namespace (and not attached):
##  [1] viridis_0.5.1            httr_1.4.1               jsonlite_1.6.1
##  [4] viridisLite_0.3.0        modelr_0.1.6             assertthat_0.2.1
##  [7] sp_1.4-1                 stats4_3.6.3             RBGL_1.62.1
## [10] cellranger_1.1.0         yaml_2.2.1               pillar_1.4.3
## [13] backports_1.1.5          lattice_0.20-40          glue_1.3.2
```

```
## [16] digest_0.6.25           RColorBrewer_1.1-2      rvest_0.3.5
## [19] colorspace_1.4-1        htmltools_0.4.0         plyr_1.8.6
## [22] XML_3.99-0.3            pkgconfig_2.0.3         broom_0.5.5
## [25] raster_3.0-12           haven_2.2.0             scales_1.1.0
## [28] generics_0.0.2          withr_2.1.2             BiocGenerics_0.32.0
## [31] cli_2.0.2               magrittr_1.5            crayon_1.3.4
## [34] readxl_1.3.1            evaluate_0.14           fs_1.3.2
## [37] fansi_0.4.1             nlme_3.1-145            xml2_1.2.5
## [40] graph_1.64.0            tools_3.6.3             hms_0.5.3
## [43] lifecycle_0.2.0         munsell_0.5.0           reprex_0.3.0
## [46] compiler_3.6.3          rlang_0.4.5             grid_3.6.3
## [49] rstudioapi_0.11         igraph_1.2.4.2          base64enc_0.1-3
## [52] rmarkdown_2.1           codetools_0.2-16        gtable_0.3.0
## [55] DBI_1.1.0               reshape2_1.4.3          R6_2.4.1
## [58] gridExtra_2.3           lubridate_1.7.4         knitr_1.28
## [61] stringi_1.4.6           readMzXmlData_2.8.1     parallel_3.6.3
## [64] Rcpp_1.0.3              vctrs_0.2.4             readBrukerFlexData_1.8.5
## [67] dbplyr_1.4.2            tidyselect_1.0.0        xfun_0.12
```