PLAQUE PICKER Example

Introduction

This document gives an overview about the core functionality of the plaquePicker-package. The general idea of plaquePicker is to analyze sparsely-distributed signals in an MSI-data set as single objects. In our example we did that for mouse models of Alzheimer's disease were we analyzed the data on the single Amyloid (Abeta) plaque level. We defined a plaque as containing at least one Abeta species in at least one pixel. For that a thresholding on the single ion images level was sufficient. Of course also other - more complex - methods of feature selection would be possible. Any feature selection method that leads to a binary image would also work.

Data preprocessing

Load needed libraries

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

  # 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 prepared 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")

# read spectra
spec <- importImzMl("NLGF_Prot_NLGF1.imzML",
                    verbose = FALSE)
```

```
# tidy up
file.remove("NLGF_Prot_NLGF1.imzML")
```

```
## [1] TRUE
```

```
file.remove("NLGF_Prot_NLGF1.ibd")
```

```
## [1] TRUE
```

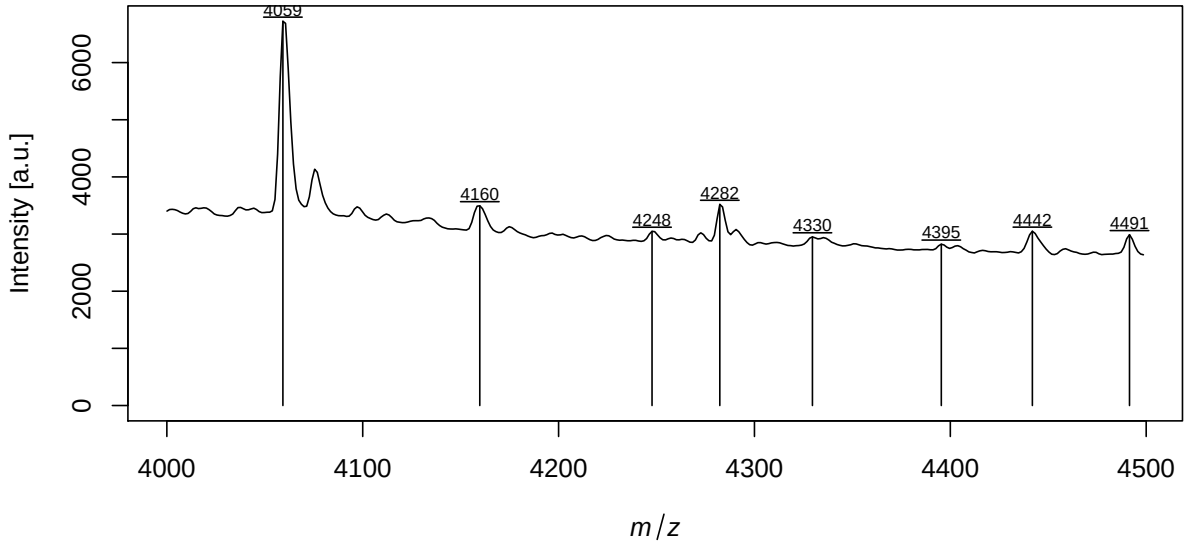
Preprocess spectra using MALDIquant functions: Normalize, smooth, remove baseline.

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

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

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

General overview of the data set

Compute average spectrum and plot it to get an overview of the dataset.

```
avgSpec <- averageMassSpectra(spec)

# shorten file path (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.

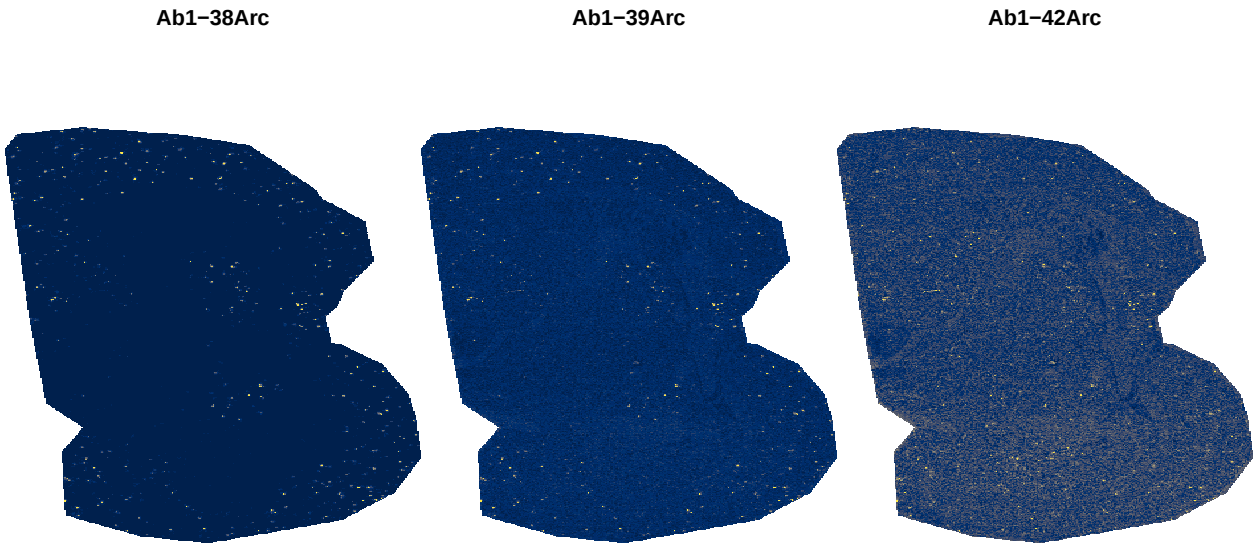
Next we extract ion images of interest.

```
ionImages <- msiSlices(spec,
                       center = c(4059.9,
                                  4159.1,
                                  4442.6),
                       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)
```

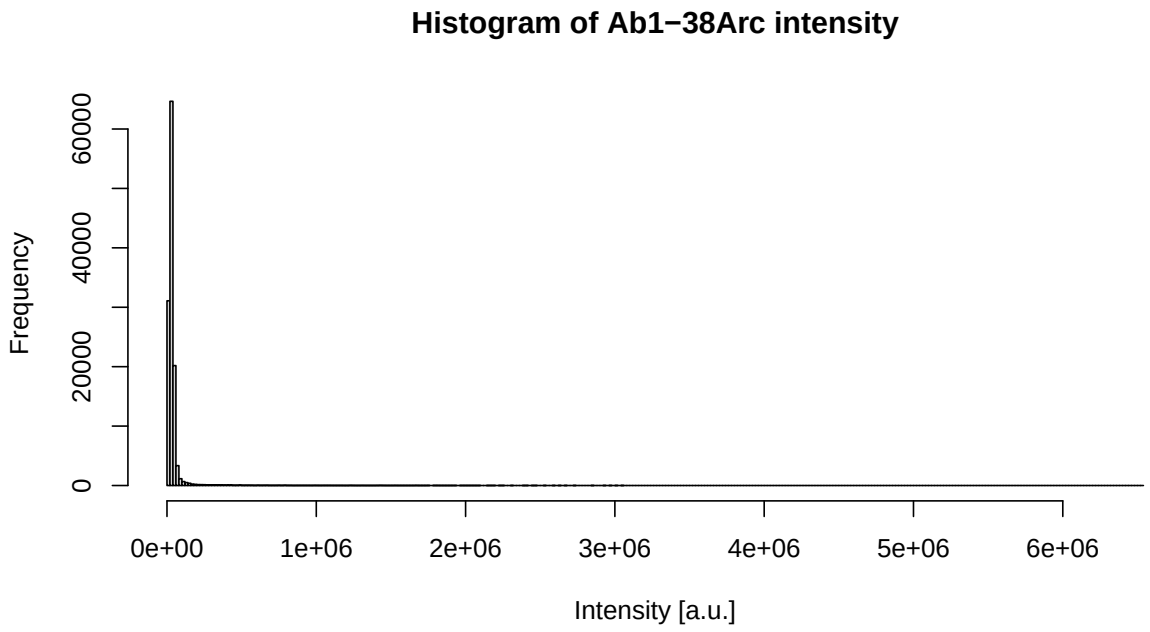
```
image(qcor(ionImages[,,3], 0.9995),
      col = viridis::cividis(30),
      asp = 1,
      axes = FALSE)
title("Ab1-42Arc", line = -2)
```



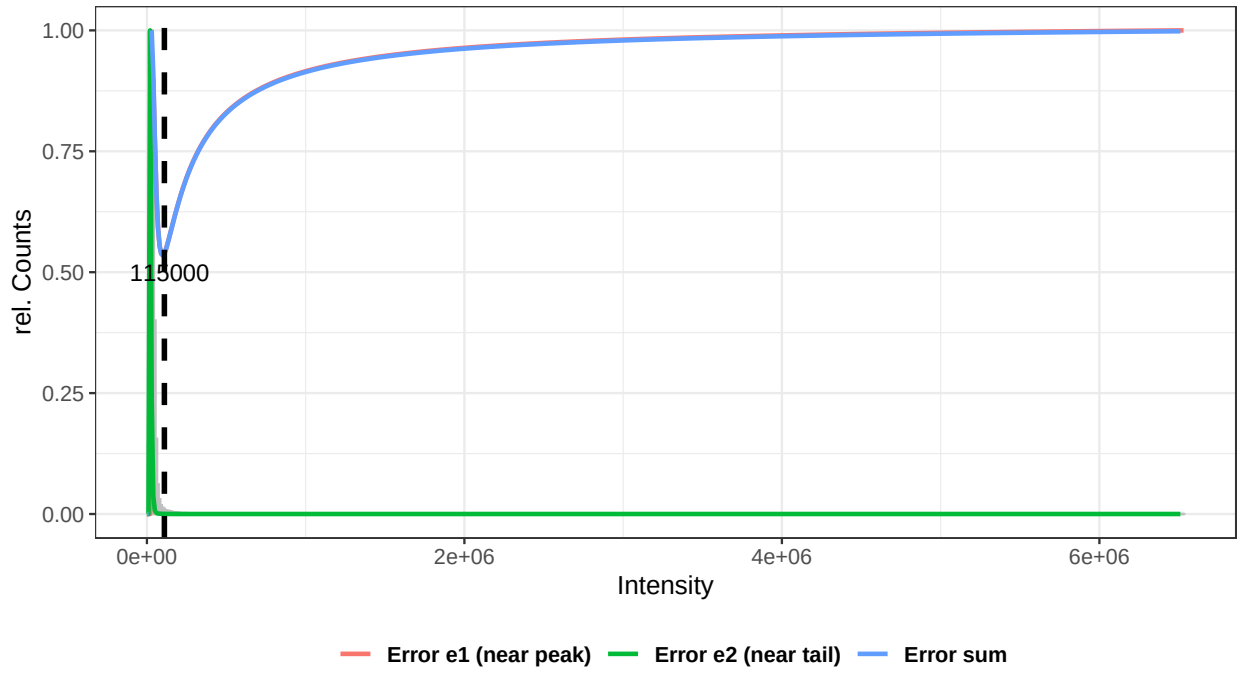
Thresholding and general principle

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. This method fits to lines to the histogram, one for the ascending and one for the descending part. Both will of course have error. The intensity value where the sum of this two error is the lowest will be defined as threshold (see second plot). For more information check Coudray, Nicolas; Buessler, Urban (2010). "Robust threshold estimation for images with unimodal histograms". Pattern Recognition Letters. 31 (9): 1010–1019. doi:10.1016/j.patrec.2009

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



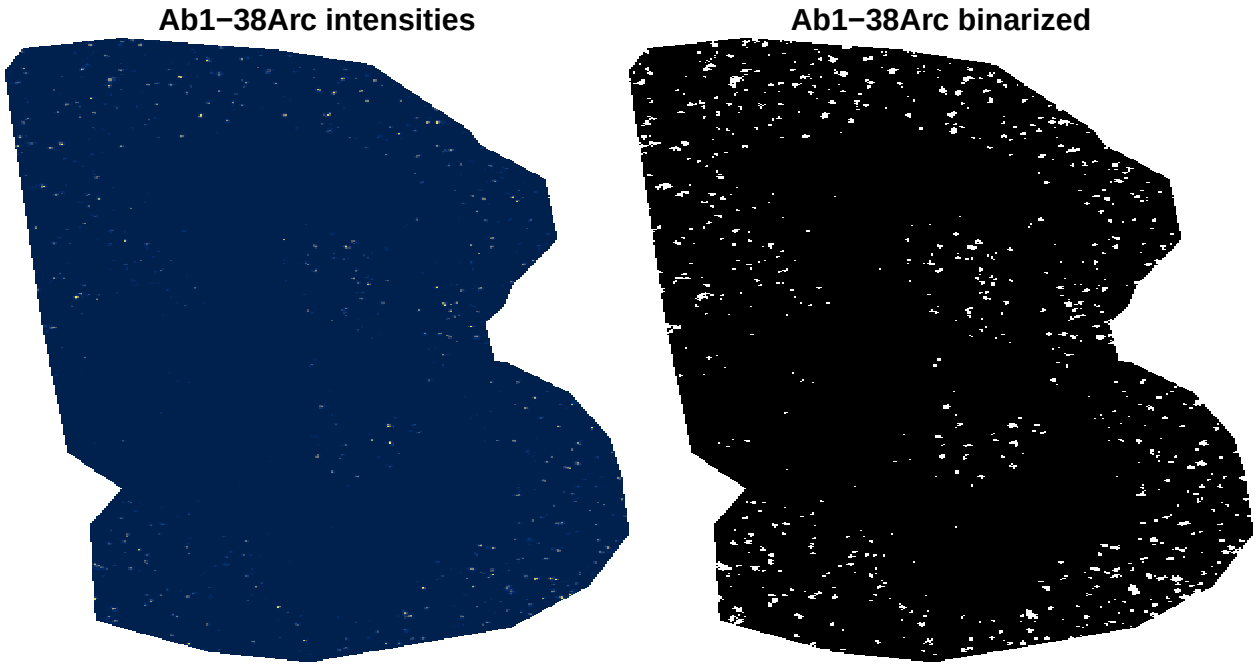

```
thresh_Ab38 <- tpoint(ionImages[,,1], plot = TRUE)
```



If we apply this threshold to the corresponding ion image we get a binarized 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 addition to the ion images this function also needs the coordinates of the data set so we have to also extract them. Note that there are two other thresholding methods implemented. If you do not wish to apply any thresholding but rather use an ion image of data that was already peak picked set method to "peak". Now each pixel with a intensity > 0 ("signal-bearing-pixel") will be considered for the following processing.

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

```
## processing 4059.9 1 / 3
##
##  threshold of mz 4059.9 based on t-point thresholding =  115000

## Loading required namespace: igraph

## extracting clump information...
## processing 4159.1 2 / 3
##
##  threshold of mz 4159.1 based on t-point thresholding =  89000
## extracting clump information...
## processing 4442.6 3 / 3
##
##  threshold of mz 4442.6 based on t-point thresholding =  81000
## extracting clump information...
## extracting unified clump information...
```

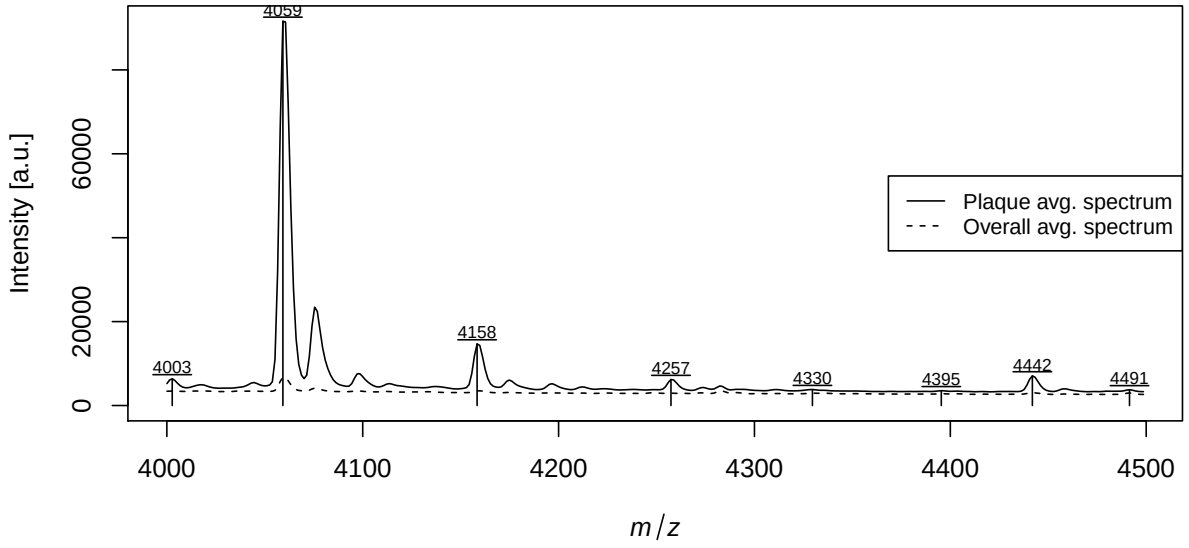
Single object analysis

Now that we have the data set structured in a way suitable for single-object analysis we can perform some basic tasks. For example, lets take all spectra associated with Abeta signals and calculate an average spectrum of plaques and compare it to the average spectrum of the whole data set we computed above. We see that as discussed above, the Abeta signals were underrepresented in the original average spectrum as they only appear in a small number of the total pixels.

```
# first extract the MALDIquant indicies associated with plaque
idx <- get_IdxFromID(pp, ID = NA) # setting ID to NA will extract all IDs instead of specific IDs
```

```
## ID set to NA. Returning all indicies associated with any clump.
```

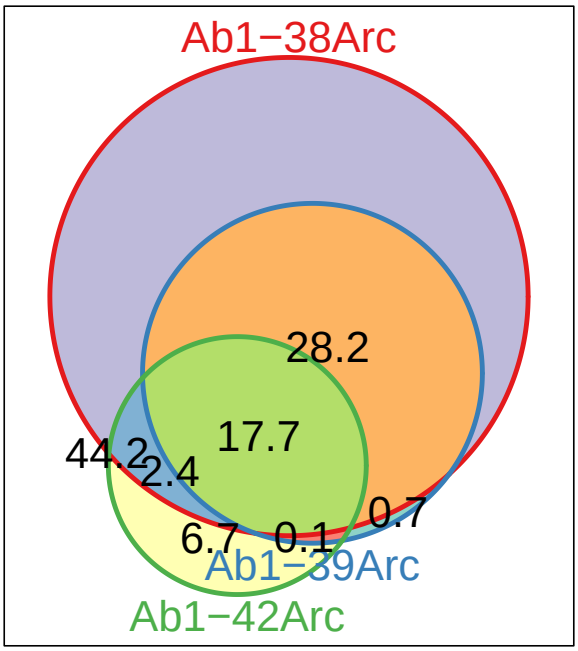
```
# compute average spectrum of plaque associated pixels
plaqueAvg <- averageMassSpectra(spec[idx])

# shorten file path (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)
```



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 (binMat) of interest we extract the plaques that co-localize with each other. This enables us to assess the plaques regarding their composition.

We can use this data to analysis the general qualitative composition of plaques using a Venn diagram. 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. Instead of using the number of plaques as labels we used relative values (%-total number of plaques).

```
plot_venn(pp, mzNames = c("Ab1-38Arc",
                          "Ab1-39Arc",
                          "Ab1-42Arc"),
          relative = TRUE)
```



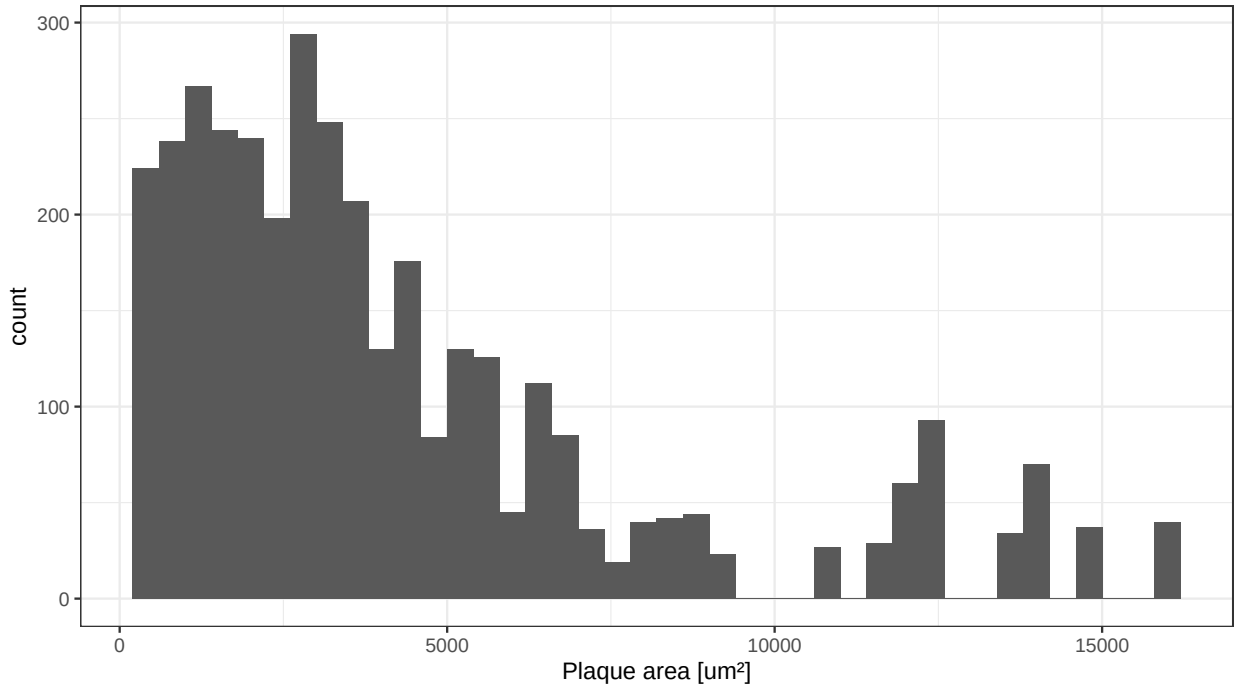
As a next step lets look at the size distribution of the plaques. We recorded the MSI data set with a pixel-size of 20x20 micrometer (um) which results in a pixel-area of 400 um². The list "unified" in the pp object contains an entry "intensities" here we find a list for each plaque ID that contains all the intensities for the different mz values. The length of these vectors is equal to the number of pixels per plaque. First we transform this list structure to tibble for better usability. We will count the number of pixels per ID and then multiply it by 400 um² (pixel-area).

```
df <- bind_rows(pp$unified$intensities, .id = "ID") %>%
        mutate(ID = as.numeric(ID)) %>%
        group_by(ID) %>%
        mutate(size = n() * 400)
head(df)
```

```
## # A tibble: 6 x 5
## # Groups:   ID [2]
##      ID `4059.9` `4159.1` `4442.6`  size
##   <dbl>    <dbl>    <dbl>    <dbl> <dbl>
## 1     1   252950    33675    28550   800
## 2     1   326100    45900    38375   800
## 3     2   172025    40650    35025  6800
## 4     2   123025    33250    27225  6800
## 5     2   116575    50650    39300  6800
## 6     2   121225    32500    26500  6800
```

Using this tibble we first plot the histogram for the size distribution. We find that most plaques are small but there are also some plaques that are really big.

```
ggplot(df, aes(x = size)) +
  geom_histogram(bins = 40) +
  theme_bw() +
  labs(x = "Plaque area [um²]")
```



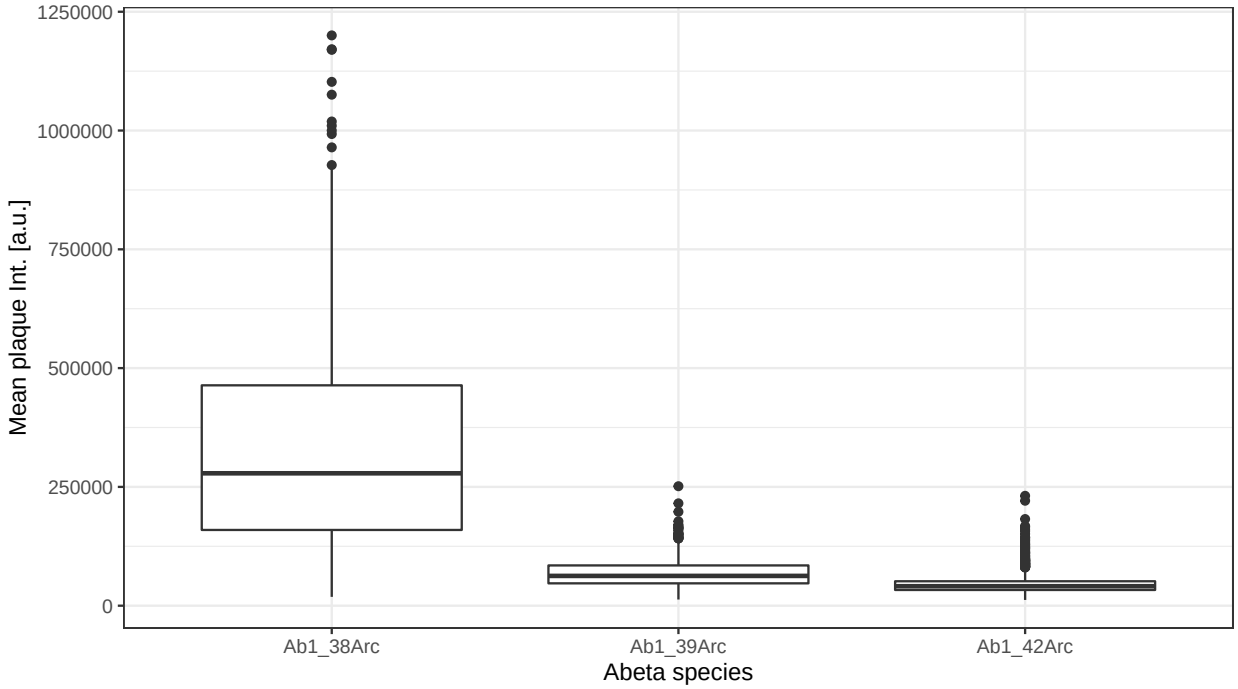
Using the Venn-diagram we already got an impression about the mean composition of the plaque but we did not make use about the intensity information we have. Note: MALDI and especially MALDI imaging are not easy to use quantitatively and in this studies we did not take any matters (like spraying and standard) to facilitate quantification. The ionization in MALDI imaging is a complex process and especially in case of peptides the relative ionization of different large peptides is well documented. As the peptides become larger they require more energy to ionize and also hit the instrument detector with a lower energy which leads them to create a lower signal. Still, although we may not be able to make absolute statements we can still learn about relative distributions although a ratio of 1:1 for two different signals will not really mean equal amounts and rather then looking at one signal alone or the ratio of two we should focus more on looking how ratios change. With that in mind lets take a look at the data: First we need to calculate the mean intensities per plaque.

```
df_mean <- df %>%
  ungroup() %>%
  rename("Ab1_38Arc" = "4059.9",
         "Ab1_39Arc" = "4159.1",
         "Ab1_42Arc" = "4442.6") %>%
  group_by(ID) %>%
  gather(mz, int, -size, -ID) %>%
  group_by(ID, mz) %>%
  summarise(meanInt = mean(int),
            size = first(size),
            .groups = "drop_last")
df_mean %>%
  spread(mz, meanInt) %>%
  head()
```

```
## # A tibble: 6 x 5
## # Groups:   ID [6]
##      ID  size Ab1_38Arc Ab1_39Arc Ab1_42Arc
##   <dbl> <dbl>     <dbl>     <dbl>     <dbl>
## 1     1   800   289525     39788.    33462.
```

```
## 2     2  6800   344522.    61004.    33000
## 3     3   800   204475     39112.    21988.
## 4     4 10800   473161.    85405.    39796.
## 5     5   800   157625     41850     29500
## 6     6  2000   314170     73500     60005
```

```
ggplot(df_mean, aes(x = mz, y = meanInt)) +
  geom_boxplot() +
  theme_bw() +
  labs(x = "Abeta species",
       y = "Mean plaque Int. [a.u.]")
```

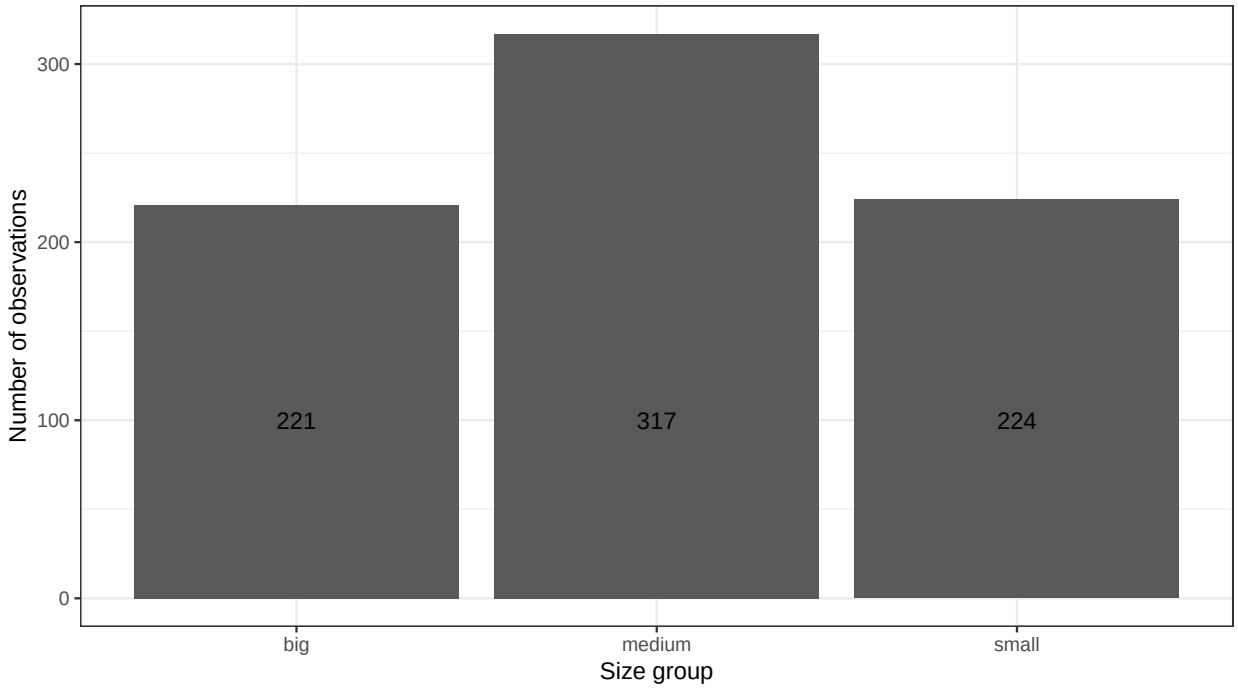


From the figure above we see that Ab1-38Arc has a much higher overall intensity then the other two species. As already pointed out the raw intensity values will not tell us that much and we should rather use ratios of intensities.

Next we define some size-groups and see if there is anything we can learn regarding molecular composition of different plaque sizes. As we see from the histogram most plaques are well below 10000 um^2 and as we want to have comparable group sizes we have to set the group boundaries according to that. We will define everything <= 400 um^2 (which means 1 pixel only) as "small", <= 2000 um^2 as "medium" and > 2000 um^2 as "big".

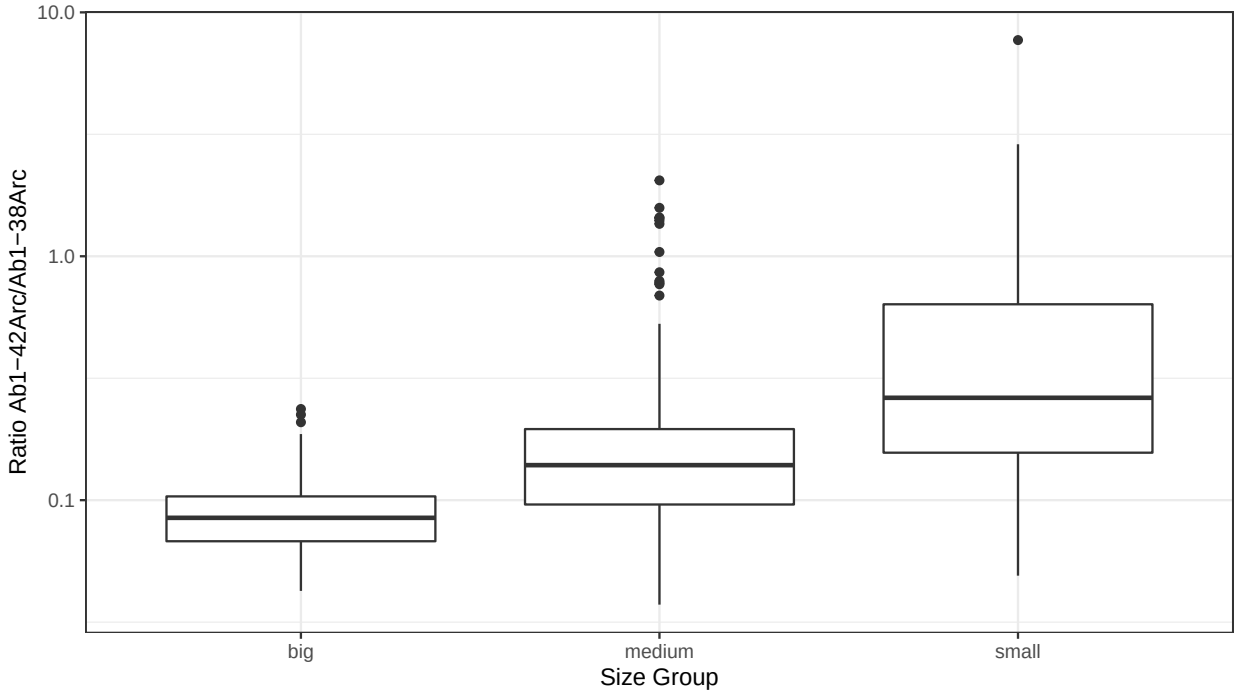
```
df_size <- df_mean %>%
  spread(mz, meanInt) %>%
  group_by(ID) %>%
  mutate(sizeGroup = ifelse(size <= 400,
                            "small",
                            ifelse(size <= 2000,
                                   "medium",
                                   "big")))

df_size %>%
```

```
  group_by(sizeGroup) %>%
  summarise(n = length(unique(ID)), .groups = "drop_last") %>%
  ggplot(aes(x = sizeGroup,
             y = n,
             label = n)) +
  geom_col() +
  geom_text(y = 100) +
  theme_bw() +
  labs(x = "Size group",
       y = "Number of observations")
```



Next we plot the Ab1-42Arc vs Ab1-38Arc ratio against the group size. As we see the smaller plaques have a much higher median ratio (meaning more Ab1-42Arc in relation to Ab1-38Arc) then the big ones.

```
df_size %>%
  mutate(ratio = Ab1_42Arc/Ab1_38Arc) %>%
  ggplot(aes(x = sizeGroup, y = ratio)) +
  geom_boxplot() +
  scale_y_log10() +
  theme_bw() +
  labs(x = "Size Group",
       y = "Ratio Ab1-42Arc/Ab1-38Arc")
```



Session info

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## 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  viridis_0.5.1          viridisLite_0.3.0
##  [4] forcats_0.5.0          stringr_1.4.0          dplyr_1.0.0
##  [7] purrr_0.3.4            readr_1.3.1            tidyr_1.1.0
## [10] tibble_3.0.3           ggplot2_3.3.2          tidyverse_1.3.0
## [13] MALDIquantForeign_0.12 MALDIquant_1.19.3      PlaquePicker_0.2.0
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.2              jsonlite_1.7.0          modelr_0.1.8
##  [4] assertthat_0.2.1        sp_1.4-2                stats4_4.0.2
```

```
##  [7] RBGL_1.64.0             blob_1.2.1              cellranger_1.1.0
## [10] yaml_2.2.1              pillar_1.4.6            backports_1.1.8
## [13] lattice_0.20-41         glue_1.4.1              digest_0.6.25
## [16] RColorBrewer_1.1-2      rvest_0.3.6             colorspace_1.4-1
## [19] htmltools_0.5.0         plyr_1.8.6              XML_3.99-0.5
## [22] pkgconfig_2.0.3         broom_0.7.0             raster_3.3-13
## [25] haven_2.3.1             scales_1.1.1            generics_0.0.2
## [28] farver_2.0.3            ellipsis_0.3.1          withr_2.2.0
## [31] BiocGenerics_0.34.0     cli_2.0.2               magrittr_1.5
## [34] crayon_1.3.4            readxl_1.3.1            evaluate_0.14
## [37] fs_1.4.2                fansi_0.4.1             xml2_1.3.2
## [40] graph_1.66.0            tools_4.0.2             hms_0.5.3
## [43] lifecycle_0.2.0         munsell_0.5.0           reprex_0.3.0
## [46] compiler_4.0.2          rlang_0.4.7             grid_4.0.2
## [49] rstudioapi_0.11         igraph_1.2.5            base64enc_0.1-3
## [52] labeling_0.3            rmarkdown_2.3           codetools_0.2-16
## [55] gtable_0.3.0            DBI_1.1.0               reshape2_1.4.4
## [58] R6_2.4.1                gridExtra_2.3           lubridate_1.7.9
## [61] knitr_1.29              utf8_1.1.4              stringi_1.4.6
## [64] readMzXmlData_2.8.1     parallel_4.0.2          Rcpp_1.0.5
## [67] vctrs_0.3.2             readBrukerFlexData_1.8.5 dbplyr_1.4.4
## [70] tidyselect_1.1.0        xfun_0.16
```