# MetCirc: Navigating mass spectral similarity in high-resolution MS/MS metabolomics data

Thomas Naake[1] and Emmanuel Gaquerel[2]

[1] Max Planck Institute of Molecular Plant Physiology
14476 Potsdam-Golm, Germany
[2] Plant Defense Metabolism
Centre for Organismal Studies
69120 Heidelberg, Germany

April 27, 2020

## 1 Introduction

The MetCirc package comprises functionalities to display, (interactively) explore similarities and annotate features of MS/MS metabolomics data. It is especially designed to improve the interactive exploration of metabolomics data obtained from cross-species/cross-tissues comparative experiments. MetCirc relies on the MSnbase infrastructure to handle MS/MS spectra. Specifically, MetCirc uses the `Spectra` class from which similarities between `Spectrum2` objects are calculated and which stores information on spectra in the columns of the slot `elementMetadata`. Notably, MetCirc provides functions to calculate the similarity between individual MS/MS spectra based on a normalised dot product (NDP, see **Li2015** for further details) calculation taking into account shared fragments or main neutral losses.

Visualisation of molecular networks was pioneered by the Dorrestein lab (**Watrous2012**) to efficiently organize MS/MS spectra in such a way that clusters of MS/MS spectra sharing multiple common fragments are rapidly inferred.
In contrary to the analysis there, the MetCirc framework offers two ways for the computation of mass spectrum similarity: one way deploys common (shared) fragments and the other shared neutral losses that are deduced from the spectra. Especially the latter approach allows to extract clusters of spectra corresponding to compound families facilitating the rapid dereplication of hitherto unknown metabolites. Small molecules, as produced by plants, return during fragmentation few fragments. Sometimes only a few fragments among them are shared and diagnostic across the compound family. Compared to the MS/MS similarity scoring based on shared fragments, the second similarity measure incorporates, in a non-supervised manner, neutral losses, which definitely helps obtaining biochemically-meaningful MS/MS groupings. Furthermore, any other function to calculate similarities can be provided or a similarity matrix

providing similarity (in the range from 0 to 1) for features that are present in the `Spectra` object.

The interpretation drawn from network reconstruction highly depends from topological parameters applied during the network construction steps. MetCirc circumvents this confinement. Furthermore, it uses instead smaller MS/MS datasets obtained from experiments involving *a priori* defined biological groups (organisms, tissues, etc.) to visualise within and between MS/MS feature similarities on a circular layout - inspired by the Circos framework (**Krzywinkski2009**). The visualisation is adjustable (MS/MS ordering and similarity thresholds) via the shiny interface and does not uniquely emphasize on large clusters of MS/MS features (a frequent caveat of network visualisation). Furthermore, the implemented functionality for annotation may improve dereplication of known and unknown molecules.

This vignette uses as a case study indiscriminant MS/MS (idMS/MS) data from **Li2015**, unpublished idMS/MS data collected from different organs of tobacco flowers and data from the Global Natural Product Social Molecular Networking (GNPS) library to navigate through the analysis pipeline. The pipeline includes the creation of `Spectra` objects, the calculation of a similarity measure (NDP), assignment to a similarity matrix and visualisation of similarity based on interactive and non-interactive graphical tools using the circlize framework (**Gu2014**).

MetCirc is currently under active development. If you discover any bugs, typos or develop ideas of improving MetCirc feel free to raise an issue via GitHub or send a mail to the developers.

## 2 Prepare the environment

To install MetCirc enter the following to the `R` console

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
BiocManager::install("MetCirc")
```

Before starting, load the MetCirc package. This will also load the required packages MSnbase, circlize, amap, scales and shiny:

```
library(MetCirc)

## Loading required package:  amap
## Loading required package:  circlize
## ========================================
## circlize version 0.4.8
## CRAN page:  https://cran.r-project.org/package=circlize
## Github page:  https://github.com/jokergoo/circlize
## Documentation:  http://jokergoo.github.io/circlize_book/book/
##
```

```
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##  in R. Bioinformatics 2014.
## ========================================
## Loading required package:  scales
## Loading required package:  shiny
## Loading required package:  MSnbase
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, parApply, parCapply, parLapply,
##    parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##    IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##    Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##    as.data.frame, basename, cbind, colnames, dirname, do.call,
##    duplicated, eval, evalq, get, grep, grepl, intersect, is.unsorted,
##    lapply, mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##    pmin.int, rank, rbind, rownames, sapply, setdiff, sort, table,
##    tapply, union, unique, unsplit, which, which.max, which.min
## Loading required package:  Biobase
## Welcome to Bioconductor
##
##    Vignettes contain introductory material; view with
##    'browseVignettes()'.  To cite Bioconductor, see
##    'citation("Biobase")', and for packages 'citation("pkgname")'.
## Loading required package:  mzR
## Loading required package:  Rcpp
## Loading required package:  S4Vectors
## Loading required package:  stats4
##
## Attaching package:  'S4Vectors'
## The following object is masked from 'package:base':
##
##    expand.grid
```

```
## Loading required package:  ProtGenerics
##
## Attaching package:  'ProtGenerics'
## The following object is masked from 'package:stats':
##
##    smooth
##
## This is MSnbase version 2.14.0
##  Visit https://lgatto.github.io/MSnbase/ to get started.
##
## Attaching package:  'MSnbase'
## The following object is masked from 'package:base':
##
##    trimws
## No methods found in package 'MSnbase' for request:  'Spectra' when loading 'MetCirc'
## No methods found in package 'MSnbase' for request:  'Spectrum2' when loading 'MetCirc'
```

The central infrastructure class of the MetCirc package is the `Spectra` object. The `Spectra` objects stores information on precursor m/z, retention time, m/z and intensity values of fragments and possible additional information, e.g. to which class the MS/MS feature belong to, the identity of the MS/MS feature, information on the adduct of the precursor, additional information. Furthermore the `Spectra` object can host information under which condition (in which tissue, etc.) the MS/MS feature was detected.

We provide here three examplary workflows to create `Spectra` from data frames that are loaded into R: (1) by loading a data frame with a minimum requirement of a column "id" (comprising unique identifiers for MS/MS features) and of the columns "mz" and "intensity" comprising the fragment ions and their intensities, respectively, or (2) by loading a data frame resembling a .MSP file object.

**Load example data sets from Li2015, tissue idMS/MS data and GNPS data.** sd02_deconvoluted comprises 360 idMS/MS deconvoluted spectra with fragment ions (m/z, chromatographic retention time, relative intensity in %) and a column with unique identifiers for MS/MS features (here, the corresponding principal component group with the precursor ion). The data set is derived from the study of **Li2015**.

```
## load data from Li et al., 2015
data("sd02_deconvoluted", package = "MetCirc")

## Warning in data("binnedMSP", package = "MetCirc"):  data set 'binnedMSP' not found
```

The second data set comes from the data-independent MS/MS collection of different floral organs from tobacco plants. Using our pipeline, this data set will be used to visualise shared

metabolites between tissues as well as structural relationships among within- and between-organ MS/MS spectra. MS/MS data are merged across floral organs in one unique data file `idMSMStissueproject.Rdata` as `tissue`. Information on the organ-localisation of each MS/MS spectrum is stored in `compartmentTissue`.

```
## load idMSMStissueproject
data("idMSMStissueproject", package = "MetCirc")
```

The third data set comes from the GNPS data base (downloaded at February 11, 2017 from: http://prime.psc.riken.jp/Metabolomics_Software/MS-DIAL/MSMS-GNPS-Curated-Neg.msp). It contains 22 MS/MS features (a truncated file of the original one) and is formatted in the .MSP file format, a typical format for MS/MS libraries. The MS/MS spectra are (normally) separated by blank lines and give information on the metabolite (its name, precursor m/z value, retention time, class, adduct ion) and contain additional information.

```
## load data from GNPS
data("convertMsp2Spectra", package = "MetCirc")
```

# 3 Prepare data for mass spectral similarity calculations

## 3.1 Preparing the sd02_deconvoluted data set for analysis

Here, we convert the examplatory data from **Li2015** into a `Spectra` object that is later used as the input for mass spectral similarity calculations. Data in the formof the data frame sd02_deconvoluted require columns that store the m/z values of fragments, the intensity values for the respective fragments and a column containing an "id". The latter column contains information about the precursor ion and should be a unique descriptor for the MS/MS features. For instance, this column may be the derived from the output of the xcms/CAMERA processing creating unique identifiers for each metabolite decoded in the column "pcgroup". A pcgroup is defined as a peak correlation group obtained by this workflow. It corresponds to a MS or MS/MS spectrum deconvoluted by CAMERA and contains information about the precursor identity and its isotope cluster. The column "id" may additionally contain information about the precursor retention time.

Create `Spectra` object from the data of (**Li2015**):

```
## get all MS/MS spectra
id_uniq <- unique(sd02_deconvoluted[, "id"])

## obtain precursor m/z from id_uniq
prec_mz_l <- lapply(strsplit(as.character(id_uniq), split=" _ "), "[", 2)
prec_mz_l <- lapply(prec_mz_l, as.numeric)
```

```
## obtain m/z from fragments per precursor m/z
mz_l <- lapply(id_uniq, function(x) sd02_deconvoluted[sd02_deconvoluted[, "id"] == x, "mz"])
## obtain corresponding intensity values
int_l <- lapply(id_uniq, function(x) sd02_deconvoluted[sd02_deconvoluted[, "id"] == x, "inte

## obtain retention time by averaging all retention time values
rt_l <- lapply(id_uniq, function(x) sd02_deconvoluted[sd02_deconvoluted[, "id"] == x, "rt"])
rt_l <- lapply(rt_l, mean)

## create list of spectrum2 objects
spN_l <- lapply(1:length(mz_l), function(x) new("Spectrum2", rt=rt_l[[x]], precursorMz=prec_

## combine list of spectrum2 objects to Spectra object
spectra_li <- MSnbase::Spectra(spN_l, elementMetadata=S4Vectors::DataFrame(show=rep(TRUE, le
```

Furthermore, further columns in the `DataFrame` passed to `elementMetadata` can be specified to host additional information: metabolite name (column "names") and classes (column "classes"), adduct ion names (column "adduct"), further information (column "information").

### 3.2 Preparing the floral organ data set for analysis

We would like to restrict the proof-of-function analysis to four tissues (sepal, SPL; limb, LIM; anther, ANT; style, STY). We will truncate `tissue` in order that it contains only these instances belonging to these types of tissue. In a next step, we will create a Spectra-object, `spl` with information concerning if the MS/MS features are found in the tissues SPL, LIM, ANT and STY.

```
## get all MS/MS spectra
tissue <- tissue[tissue[, "id"] %in% compartmentTissue[, "mz_rt_pcgroup"], ]
id_uniq <- unique(tissue[, "id"])

## obtain precursor m/z from id_uniq
prec_mz_l <- lapply(strsplit(as.character(id_uniq), split="_"), "[", 1)
prec_mz_l <- lapply(prec_mz_l, as.numeric)

## obtain m/z from fragments per precursor m/z
mz_l <- lapply(id_uniq, function(x) tissue[tissue[, "id"] == x, "mz"])
## obtain corresponding intensity values

## obtain retention time by averaging all retention time values
int_l <- lapply(id_uniq, function(x) tissue[tissue[, "id"] == x, "intensity"])
rt_l <- lapply(id_uniq, function(x) tissue[tissue[, "id"] == x, "rt"])
```

```
rt_l <- lapply(rt_l, mean)

## create list of spectrum2 objects
spN_l <- lapply(1:length(mz_l), function(x) new("Spectrum2", rt=rt_l[[x]], precursorMz=prec_

## combine list of spectrum2 objects to Spectra object,
## use SPL, LIM, ANT, STY for further analysis
spectra_tissue <- MSnbase::Spectra(spN_l,
    elementMetadata=S4Vectors::DataFrame(compartmentTissue[, c("SPL", "LIM", "ANT", "STY")])
```

### 3.3 Preparing the GNPS data set for analysis

Alternatively as mentioned above, an `Spectra`-object can also created from .MSP objects, that are typical data formats for storing MS/MS libraries. Required properties of such a data frame are the name of the metabolite (row entry "NAME:"), the m/z value of the precursor ion ("PRECURSORMZ" or "EXACTMASS:"), retention time ("RETENTIONTIME:"), information on the number of peaks ("Num Peaks:") and information on fragments and peak areas (for further information see convertMsp2Spectra and retrieve `data("convertMsp2Spectra", package = "MetCirc")` for a typical msp data frame).

Create `Spectra`-object from the GNPS .MSP file:

```
spectra_msp <- convertMsp2Spectra(msp2spectra)
```

## 4 Binning and calculation of similarity matrix

### 4.1 Workflow for `tissue` data set using fragment ions

**Calculation of the similarity matrix.** The similarity matrix can be obtained by by the MSnbase package function compare_Spectra which takes a `Spectra` object and a function for the similarity calculation as input.

The MetCirc package comes with two functions to calculate pair-wise similarities between MS/MS features. One of them, `normalizeddotproduct` calculates the similarity coefficient between two MS/MS features using the normalised dot product (NDP). For a considered MS/MS pair, peak intensities of shared m/z values for precursor/fragment ions are employed as weights $W_{S1,i}$ and $W_{S1,i}$ within the following formula:

$$NDP = \frac{\sum(W_{S1,i} \cdot W_{S2,i})^2}{\sum(W_{S1,i}^2) * \sum(W_{S2,i}^2)},$$

with S1 and S2 the spectra 1 and 2, respectively, of the $ith$ of j common peaks differing by the tolerance parameter specified in `compare_Spectra`'s argument`binSize`.

Weights are calculated according to $W = [peak\ intensity]^m \cdot [m/z]^n$, with m = 0.5 and n = 2 as default values as suggested by MassBank. For further information see **Li2015**.

Similarly, the function `neutralloss` takes into account neutral losses between precursor m/z values and all fragments.
Alternatively, the user can specify a custom-defined function to calculate similarities between MS/MS features or use an implemented function in MSnbase. The function `compare_Spectra` returns a symmetrical squared similarity matrix with pair-wsie similarity coefficients between MS/MS features.

```
## similarity Matrix
similarityMat <- compare_Spectra(spectra_tissue[1:100,], fun=normalizeddotproduct, binSize=0
```

`createSimilarityMatrix` returns a square matrix with the names of the `Spectra` MS/MS features as column and row names. The entries of the returned matrix have to be similarites cores ranging between 0 and 1 that are the pair-wise similarities between the MS/MS features.

**Clustering/visualisation.** At this stage, we would like to visualise the pair-wise similarities of MS/MS features after clustering them. Many functions are available to cluster features such as `hclust` from stats, `Mclust` from mclust or `hcluster` from amap. We would like to use the latter to cluster similar features. To cluster features and visualise the result (see figure 1) we enter:

```
## load package amap
hClustMSP <- hcluster(similarityMat, method="spearman")
## visualise clusters
plot(hClustMSP, labels = FALSE, xlab="", sub="")
```

To promote readibility we will not show labels. These can be printed to the `R` console by `colnames(similarityMat)[hClustMSP$order]`.

**Extraction of highly-related features using clustering.** Within the `R` session the similarity matrix can be truncated by using the above-mentioned functions to retrieve a similarity matrix that contains highly-related MS/MS features. For instance, this might be needed when we want to analyse modules of high similarity, representing e.g. a certain metabolite class. By way of example, we extract in the following all features from module 1 when defining three modules in total and define a new similarity matrix of highly-related features. This kind of matrix can be used in later analysis steps, e.g. in the analysis with `shinyCircos`.

```
## define three clusters
cutTreeMSP <- cutree(hClustMSP, k=3)
## extract feature identifiers that belong to module 1
module1 <- names(cutTreeMSP)[as.vector(cutTreeMSP) == "1"]
```

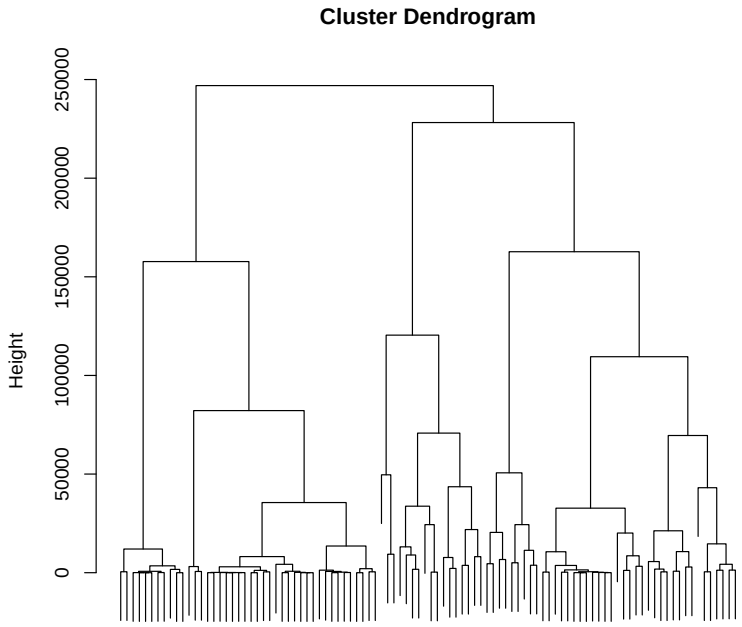


Figure 1: Cluster dendrogramm for similarity matrix based on fragment ion NDP calculation

```
## create a new similarity matrix that contains only highly related features
similarityMat_module1 <- similarityMat[module1, module1]
```

## 5 Visualisation using the shiny/circlize framework

MetCirc's main functionality is to visualise metabolomics data, exploring it interactively and annotate unknown features based on similarity to other (known) features. One of the key features of the implemented interactive framework is, that groups can be compared. A group specifies the affiliation of the sample: it can be any biological identifier relevant to the comparitive experiment conducted, e.g. it can be a specific tissue, different times, different species, etc. The visualisation tools implemented in MetCirc allow then to display similarity between precursor ions between and/or within groups.

`shinyCircos` uses the function `createLinkDf` which selects these precursor ions that have a similarity score (e.g. a normalised dot product) within the range defined by `threshold_low` and `threshold_high` to a precursor ion. Internally, in `shinyCircos`, `createLinkDf` will be called to produce a data.frame with link information based on the given thresholds.

```
linkDf <- createLinkDf(similarityMatrix=similarityMat, spectra=spectra_tissue,
    condition=c("SPL", "LIM", "ANT", "STY"), lower=0.5, upper=1)
```

As we have calculated similarity coefficients between precursors, we would like to visualise these connections interactively and explore the data. The MetCirc package implements `shinyCircos` that allows for such kind of exploration. It is based on the shiny and on the circlize (**Gu2014**) framework. Inside of `shinyCircos` information of precursor ions are displayed by (single-) clicking over precursors. Precursors can also be permanently selected by double-clicking on them. The similarity coefficients can be thresholded by changing the slider input. Also, on the sidebar panel, the type of link to be displayed can be selected: should only links between groups be displayed, should only links within groups be displayed or should all links be displayed? Ordering inside of groups can be changed by selecting the respective option in the sidebar panel. Momentarily, there are options to reorder features based on clustering, the m/z or the retention time of the precursor ion. In the "Appearance" tab, the size of the plot can be changed, as well as the precision of the displayed values for the m/z and retention time values. Please note, that the precision will only be changed in the text output (not in the graphical output). Also, the user can decide if the legend is displayed or not.
On exiting the shiny application via the exit button in the sidebar panel, selected precursors will be returned which are allocated here to `selectedFeatures`. `selectedFeatures` is a vector of the precursor names.

By way of example, we would like to analyse the tissue data set focusing on the tissues "SPL", "LIM", "ANT" and "STY".
93.067 95.085 .... 93.068 95.084 ....
To start the shiny app, run

```
selectedFeatures <- shinyCircos(similarityMat, spectra=spectra_tissue, condition=c("LIM", "A
```

to the console. This will load information stored in the `Spectra` object `spl` to the `shinyCircos` interface. Information (metabolite names and class, adduct ion name and further information) will be printed to the interface when (single-)clicking on MS/MS features. Since this `Spectra` object did not contain any information about metabolite names, class, adduct ion name and further information, initally all fields for these instances are set to "Unkwown". Note, that on the sidebar on the right side text fields will appear, that allow for changing the annotation of the selected feature. Click "update annotation" to save the changes to the `Spectra`-object. On exiting `shinyCircos` via the exit button, selected precursors and the (updated) `Spectra`-object will be returned to the console. In the example above, enter `selectedFeatures$spectra` to retrieve the `Spectra` object with updated annotation.

MetCirc allows also to create such figures outside of an interactive context, which might be helpful to create figures and export them e.g. in .pdf or .jpeg format. `shinyCircos` does currently not support to export figures as they can be easily rebuilt outside of `shinyCircos`; building figures outside of the interactive context also promotes reproducibility of such figures.

To rebuild the figure in a non-interactive environment, run

```r
## order similarity matrix according to retention time
condition <- c("SPL", "LIM", "ANT", "STY")
simM <- orderSimilarityMatrix(similarityMatrix=similarityMat,
    spectra=spectra_tissue, type="retentionTime", group=FALSE)
groupname <- rownames(simM)
inds <- MetCirc:::spectraCond(spectra_tissue,
    condition=condition)
inds_match <- lapply(inds, function(x) {inds_match <- match(groupname, x)
    inds_match <- inds_match[!is.na(inds_match)]; x[inds_match]})
inds_cond <- lapply(seq_along(inds_match),
    function(x) {
        if (length(inds_match[[x]]) > 0) {
            paste(condition[x], inds_match[[x]], sep="_")
        } else character()
})
inds_cond <- unique(unlist(inds_cond))

## create link matrix
linkDf <- createLinkDf(similarityMatrix=simM, spectra=spectra_tissue,
    condition=c("SPL", "LIM", "ANT", "STY"), lower=0.9, upper=1)
## cut link matrix (here: only display links between groups)
linkDf_cut <- cutLinkDf(linkDf=linkDf, type="inter")

## set circlize paramters
circos.par(gap.degree=0, cell.padding=c(0, 0, 0, 0),
            track.margin=c(0, 0))

## here set indSelected arbitrarily
indSelected <- 14
selectedFeatures <- inds_cond[indSelected]

## actual plotting
plotCircos(inds_cond, linkDf_cut, initialize=TRUE, featureNames=TRUE,
    cexFeatureNames=0.2, groupSector=TRUE, groupName=FALSE,
    links=FALSE, highlight=TRUE)

highlight(groupname=inds_cond, ind=indSelected, linkDf=linkDf_cut)
## plot without highlighting
plotCircos(inds_cond, linkDf_cut, initialize=TRUE, featureNames=TRUE,
    cexFeatureNames=0.2, groupSector=TRUE, groupName=FALSE, links=TRUE,
    highlight=FALSE)
```

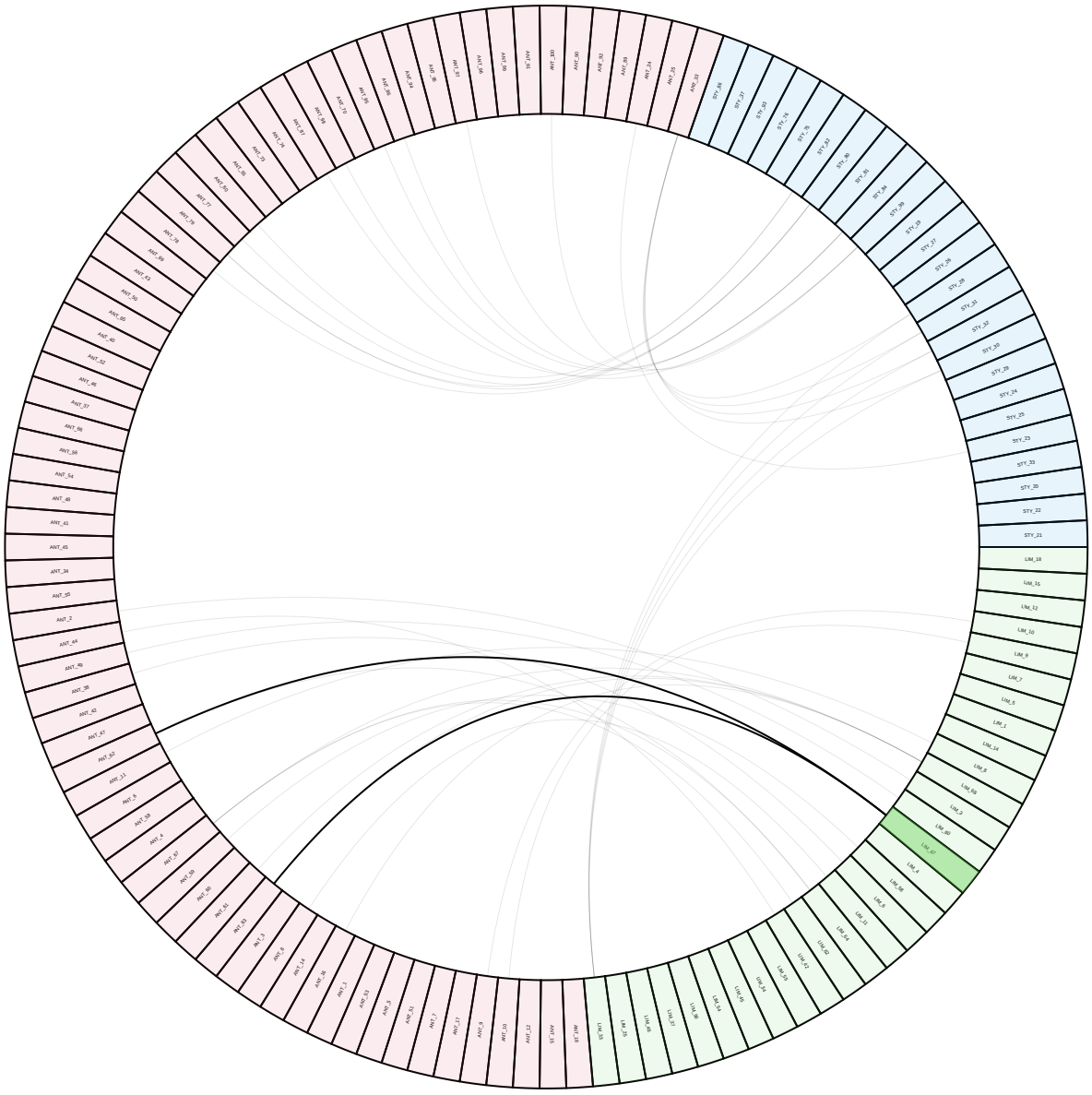

Figure 2: Examplary circos plot highlighting an arbitrary feature. Upon highlighting, all links will be plotted in grey (expect links to and from highlighted features). The intensity of the background colour of features will be reduced as well. Features belonging to a group (species, individual, organ, different time) will be indicated by the same background colour.

# Appendix

## Session information

All software and respective versions to build this vignette are listed here:

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.4 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.11-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.11-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [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
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] MetCirc_1.18.0      MSnbase_2.14.0      ProtGenerics_1.20.0
##  [4] S4Vectors_0.26.0    mzR_2.22.0          Rcpp_1.0.4.6
##  [7] Biobase_2.48.0      BiocGenerics_0.34.0 shiny_1.4.0.2
## [10] scales_1.1.0        circlize_0.4.8      amap_0.8-18
## [13] knitr_1.28
##
## loaded via a namespace (and not attached):
##  [1] lattice_0.20-41       foreach_1.5.0         assertthat_0.2.1
##  [4] digest_0.6.25         mime_0.9              R6_2.4.1
##  [7] plyr_1.8.6            mzID_1.26.0           evaluate_0.14
## [10] ggplot2_3.3.0         highr_0.8             pillar_1.4.3
## [13] GlobalOptions_0.1.1   zlibbioc_1.34.0       rlang_0.4.5
## [16] preprocessCore_1.50.0 BiocParallel_1.22.0   stringr_1.4.0
## [19] munsell_0.5.0         compiler_4.0.0        httpuv_1.5.2
## [22] xfun_0.13             pkgconfig_2.0.3       shape_1.4.4
## [25] pcaMethods_1.80.0     htmltools_0.4.0       tidyselect_1.0.0
## [28] tibble_3.0.1          IRanges_2.22.0        codetools_0.2-16
## [31] XML_3.99-0.3          crayon_1.3.4          dplyr_0.8.5
## [34] later_1.0.0           MASS_7.3-51.6         grid_4.0.0
## [37] xtable_1.8-4          gtable_0.3.0          lifecycle_0.2.0
## [40] affy_1.66.0           magrittr_1.5          ncdf4_1.17
```

```
## [43] stringi_1.4.6          impute_1.62.0          farver_2.0.3
## [46] promises_1.1.0         affyio_1.58.0          doParallel_1.0.15
## [49] limma_3.44.0           ellipsis_0.3.0         vctrs_0.2.4
## [52] iterators_1.0.12       tools_4.0.0            glue_1.4.0
## [55] purrr_0.3.4            fastmap_1.0.1          colorspace_1.4-1
## [58] BiocManager_1.30.10    vsn_3.56.0             MALDIquant_1.19.3
```

## Neutral losses

Table 1: The table gives examplatory fractionation of precursors into neutral losses (given their m/z and the corresponding atoms):

| | |
|---|---|
| $CH_2$ | 14.0157 |
| $CH_4$ | 16.0313 |
| $NH_3$ | 17.0265 |
| $H_2O$ | 18.0106 |
| $K^+$ to $NH_4{}^+$" | 20.9293 |
| $Na^+$ to $H^+$ | 21.9819 |
| $C_2H_2$ | 26.0157 |
| CO | 27.9949 |
| $C_2H_4$ | 28.0313 |
| $CH_3N$ | 29.0266 |
| $CH_2O$ | 30.0106 |
| $CH_5N$ | 31.0422 |
| S | 31.9721 |
| $H_2S$ | 33.9877 |
| $K^+$to $H^+$ | 37.9559 |
| $C_2H_2O$ | 42.0106 |
| $C_3H_6$ | 42.0470 |
| CHNO | 43.0058 |
| $CO_2$ | 43.9898 |
| $CH_2$ $O_2$ | 46.0055 |
| $C_4H_8$ | 56.0626 |
| $C_3H_9N$ | 59.0735 |
| $C_2H_4$ $O_2$ | 60.0211 |
| $CH_4N_2O$ | 60.0324 |
| $SO_2$ | 63.9619 |
| $C_5H_8$ | 68.0626 |
| $C_3H_6$ $O_2$ | 74.0368 |
| $C_6H_6$ | 78.0470 |
| $SO_3$ | 79.9568 |
| $C_3H_2O_3$ | 86.0004 |
| $C_4H_8O_2$ | 88.0517 |
| $C_4H_{12}N_2$ | 88.1000 |
| $H_2(SO)_4$ | 97.9674 |
| $H_3(PO)_4$ | 97.9769 |
| $C_5H_{10}O_2$ | 102.0618 |
| $C_3H_4O_4$ | 104.0110 |
| $C_6H_{12}O_2$ | 116.0861 |
| $C_2H_5O_4P$ | 123.9926 |

| | |
|---|---|
| $C_5H_8O_4$ | 132.0423 |
| $C_7H_{19}N_3$ | 145.1579 |
| $C_6H_{10}O_4$ | 146.0579 |
| $C_6H_{10}O_5$ | 162.0528 |
| $C_6H_{12}O_5$ | 164.0685 |
| $C_6H_8O_6$ | 176.0321 |
| $C_6H_{12}O_6$ | 180.0634 |
| $C_6H_{10}O_7$ | 194.0427 |
| $C_8H_{12}O_6$ | 204.0655 |
| $C_{11}H_{10}O_4$ | 206.0579 |
| $C_{10}H_{15}N_3\ O_6\ S$ | 305.0682 |
| $C_{10}H_{17}N_3\ O_6\ S$ | 307.0838 |
| $C_{12}H_{20}O_{10}$ | 324.1057 |
| $C_{12}H_{22}O_{11}$ | 342.1162 |