MetCirc: A R workflow for circular visualisation of mass spectral similarity in metabolomics data

Thomas Naake and Emmanuel Gaquerel

Plant Defense Metabolism
Centre for Organismal Studies

December 21, 2016

1 Introduction

The MetCirc package comprises functionalities to display and (interactively) explore metabolomics data. It is especially designed to improve the interactive exploration of metabolomics data obtained from cross-species/cross-tissues comparative experiments. Notably, MetCirc includes functions to calculate the similarity between individual MS/MS spectra based on a normalised dot product (NDP, see Li et al., (2015) for further details) calculation taking into account shared fragments or main neutral losses. This vignette uses as a case study indiscriminant MS/MS (idMS/MS) data from Li et al., (2015) and unpublished idMS/MS data collected from different organs of tobacco flowers to navigate through the analysis pipeline. The pipeline includes creation of an MSP-object, binning of fragment ions, 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 (Gu et al., 2014).

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

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

```
library(MetCirc)

## Loading required package:  amap
## Loading required package:  circlize
## Loading required package:  scales
## Loading required package:  shiny
```

Load example data sets from Li et al., (2015). sd01_outputXCMS is the output of the XCMS and CAMERA processing and statistical analysis and XCMS and CAMERA scripts (see Li et al., (2015) for further information). sd02_deconvoluted comprises 360 idMS/MS deconvoluted spectra with fragment ions (m/z, retention time, relative intensity in %) and the corresponding principal component group with the precursor ion.

```
## load data
data("sd01_outputXCMS", package = "MetCirc")
data("sd02_deconvoluted", package = "MetCirc")
```

3 Prepare data for mass spectral similarity calculations

Here, we convert examplatory data into `MSP-objects` that are used later as input for mass spectral similarity calculations. The `MSP-class` mimicks the .msp format is ASCII text format used by mass spectral libraries. The function `convert2MSP` creates an entry for each precursor and identifies the m/z and retention time of fragment ions. Each entry of the `MSP-object` has the following entries: `NAME`, `RETENTIONTIME`, `PRECURSORMZ`, `METABOLITENAME`, `METABOLITECLASS`, `ADDUCTIONNAME`, `Num Peaks` (Number of peaks) and all fragment ions together with their relative intensity in %. The retention time calculated by the function `convert2MSP` is the average value of the retention time values of all fragment ions belonging to the specific precursor. The actual `MSP data frame` can be accessed by `getMSP(MSP-object)`.

3.1 Preparing the sd02_deconvoluted data set for analysis

Here, we convert `sd02_deconvoluted` into a `MSP-object`.

```
## identify precursor mz
finalMSPLi2015 <- convert2MSP(sd02_deconvoluted, split = " _ ",
                              splitIndMZ = 2, splitIndRT = 3)
##
## ## optional:
## ## write finalMSPLi2015 to idMSMStoMSPLi2015.RData
## save(finalMSPLi2015, file = "idMSMStoMSPLi2015.RData")
```

For the `binning` function used later, we need to pass a vector containing the groups (compartments, times, species, etc.) of the metabolites. Here, we will create a vector, `compartment`, with randomised assignment of compartments to show functionality. This can be replaced with a vector comprising actual compartments, with dummy variables or with any other affilitation to a group relevant to the comparitive experiment conducted.

```
compartment <- sample(c("yl", "ol", "s","r"), size = length(finalMSPLi2015),
    replace=TRUE)
```

3.2 Preparing the tissue data set for analysis

Convert `idMSMStissueproject` into MSP-object. The data set used in this section 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`.

Often, the data is not in the right format to initiate the analysis - please make sure that the fourth column contains information about the precursor ion and that the data contains the column names `mz`, `intensity` and `rt`. The order of these columns (except the fourth column) is not important. `convert2MSP` will check internally if the data complies these requirements.

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

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 MSP-object, `finalMSP`, comprising the features found in the tissues SPL, LIM, ANT and STY.

```
## create vectors with precursor names present in tissue
tissueSPL <- compartmentTissue[compartmentTissue[,"SPL"] == TRUE, 1]
tissueLIM <- compartmentTissue[compartmentTissue[,"LIM"] == TRUE, 1]
tissueANT <- compartmentTissue[compartmentTissue[,"ANT"] == TRUE, 1]
tissueSTY <- compartmentTissue[compartmentTissue[,"STY"] == TRUE, 1]

## truncate tissue
tissueSPL <- tissue[tissue[,4] %in% tissueSPL,]
tissueLIM <- tissue[tissue[,4] %in% tissueLIM,]
tissueANT <- tissue[tissue[,4] %in% tissueANT,]
tissueSTY <- tissue[tissue[,4] %in% tissueSTY,]

## create msp and combine msp objects of different tissues
finalMSP <- convert2MSP(tissueSPL)
finalMSP <- combine(finalMSP, convert2MSP(tissueLIM))
finalMSP <- combine(finalMSP, convert2MSP(tissueANT))
finalMSP <- combine(finalMSP, convert2MSP(tissueSTY))

## ## optional:
## ## write finalMSP to idMSMStoMSP.RData
## save(finalMSP, file = "idMSMStoMSP.RData")
```

For the binning function, we will derive a vector, compartment, which gives the compartment for each entry of finalMSP. compartment refers here to floral organs, but could be species names, experimental conditions, etc., too; i.e. the object can be any biological identifier relevant to the comparative experiment conducted.

```
## create vector with compartments
compSPL <- rep("SPL", length(convert2MSP(tissueSPL)))
compLIM <- rep("LIM", length(convert2MSP(tissueLIM)))
compANT <- rep("ANT", length(convert2MSP(tissueANT)))
compSTY <- rep("STY", length(convert2MSP(tissueSTY)))

compartment <- c(compSPL, compLIM, compANT, compSTY)
```

4 Binning and calculation of similarity matrix

4.1 Workflow for tissue data set using fragment ions

Binning. Due to slight differences in m/z values over measurements fragments might have m/z values which differ from other fragments even though they are in theory identical. binning will bin together fragment ions which are similar (set by the parameter tol for tolerance). In the following this will allow for comparison between m/z values. The functions binning bins fragments together based on minimal distance to bins which were calculated either by the mean or the median of fragments which were put in intervals according to the tol parameter.

binning expects a vector (group) which comprises membership of the entries in the msp object, to a compartment, species, individual, etc. If group is not specified binning will create an internal dummy variable group (”a” with the length of the msp object). We use here the tissue data set.

```
## create data frame with binned fragments
binnedMSP <- binning(msp = finalMSP, tol = 0.01,
                     group = compartment, method = "median")
```

Calculation of the similarity matrix. The normalised dot product (NDP) is the similarity coefficient to calculate the similarity between two precursor ions and their fragments, respectively. The NDP uses the m/z value of fragment/precursor ions and their peak intensityof two metabolites, respectively. The NDP is calculated according to:

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

with $W = [peak\ intensity]^m \cdot [m/z]^n$ For further information see (Li et al., 2015).

createSimilarityMatrix calculates the NDP between all precursor ions and their respective fragment ions. createSimilarityMatrix needs a matrix as an argument which has the fragment ions as rows (m/z / retention time) and all fragment ions as columns. Entries of such a matrix are intensities for a specific fragment ion (intensity will be zero if fragment ion does not occur for the respective precursor). The function binning will return a matrix with such properties.

```r
## similarity Matrix
similarityMat <- createSimilarityMatrix(binnedMSP)
```

createSimilarityMatrix returns a matrix with precursor m/z / retention time as column and row names. The entries of the returned matrix are NDP scores ranging between 0 and 1 which indicate the similarity between the features.

Clustering/Visualisation. At this stage, we would like to visualise the similarity 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 (a combination of hclust and dist from stats) to cluster similar features. To cluster features and visualise the clustering we enter:

```r
## 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].

4.2 Workflow for tissue data set using neutral losses

Another way to compare the similarity of metabolites is based on neutral losses (cf. table 1 in the appendix for a selection of common neutral losses): common neutral losses are shared among structurally-related metabolites. MetCirc contains functionality to calculate neutral losses from MSP-objects. To convert a MSP-object with fragments into a MSP-object with neutral losses enter:

```r
nlMSP <- msp2FunctionalLossesMSP(finalMSP)
```

Binning and calculation of the similarity matrix. Analogously to the MSP-object with fragments we can bin the nlMSP MSP-object with neutral losses and create a similarity matrix:

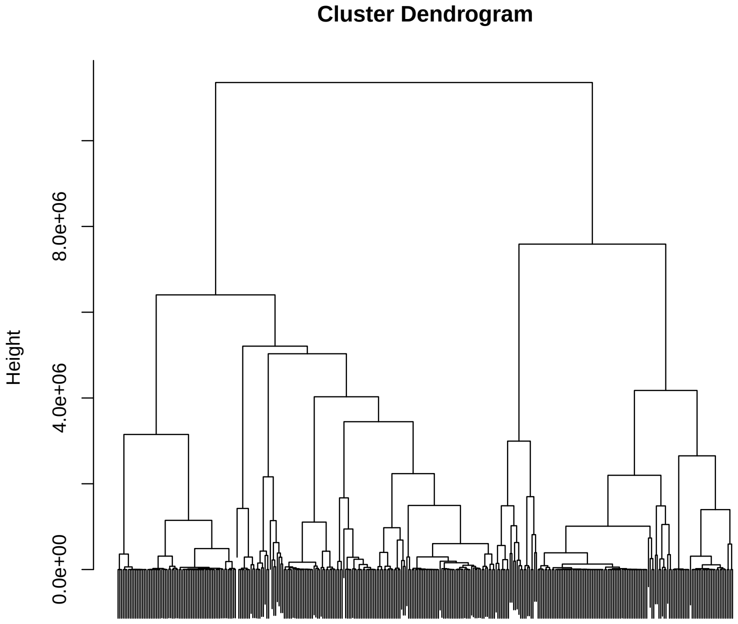

Figure 1: Cluster dendrogramm for similarity matrix based on fragment ion NDP calculation

```r
## bin msp file with functional losses, create table with same fragments
## (binning)
nlBinnedMSP <- binning(nlMSP, tol = 0.01, group = compartment, method = "median")
## similarity Matrix
nlSimilarityMat <- createSimilarityMatrix(nlBinnedMSP)
```

Clustering/Visualisation. Analogously to the `MSP-object` with fragment ions, we are able to cluster the similarity matrix based on neutral losses.

```r
## Clustering
nlHClustMSP <- hcluster(nlSimilarityMat, method = "spearman")
```

To visualise the clustering enter the following line of code (labels will not be displayed due to readibility):

```r
plot(nlHClustMSP, labels = FALSE)

## labels can be reproduced by entering in the console
colnames(nlSimilarityMat)[nlHClustMSP$order]
```

5 Visualisation using the shiny/circlize framework

MetCirc comprises functionality to visualise metabolomics data and exploring it interactively. 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 inside of groups.

`shinyCircos` uses the function `createLinkMatrix` which comprises per row these precursor ions which are linked by a normalised dot product ≥ `threshold` to the precursor ion in the same row. Internally, in `shinyCircos`, `createLinkMatrix` will be called to produce link matrices based on the given threshold.

```
linkMat <- createLinkMatrix(similarityMatrix = similarityMat, threshold=0.5)
```

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 (Gu et al., 2014) framework. Inside of `shinyCircos` information of precursor ions are displayed by hovering over precursors. Precursors can also be permanently selected by 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. 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.

To start the shiny app, run

```
selectedFeatures <- shinyCircos(similarityMat)
```

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

```
## order similarity matrix according to retention time
simM <- createOrderedSimMat(similarityMat, order = "retentionTime")
groupname <- rownames(simM)
```

```r
## create link matrix
linkMat <- createLinkMatrix(similarityMatrix = simM, threshold=0.99)
## cut link matrix (here: only display links between groups)
linkMat_cut <- cutLinkMatrix(linkMat, 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 <- 1
selectedFeatures <- groupname[1]

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

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

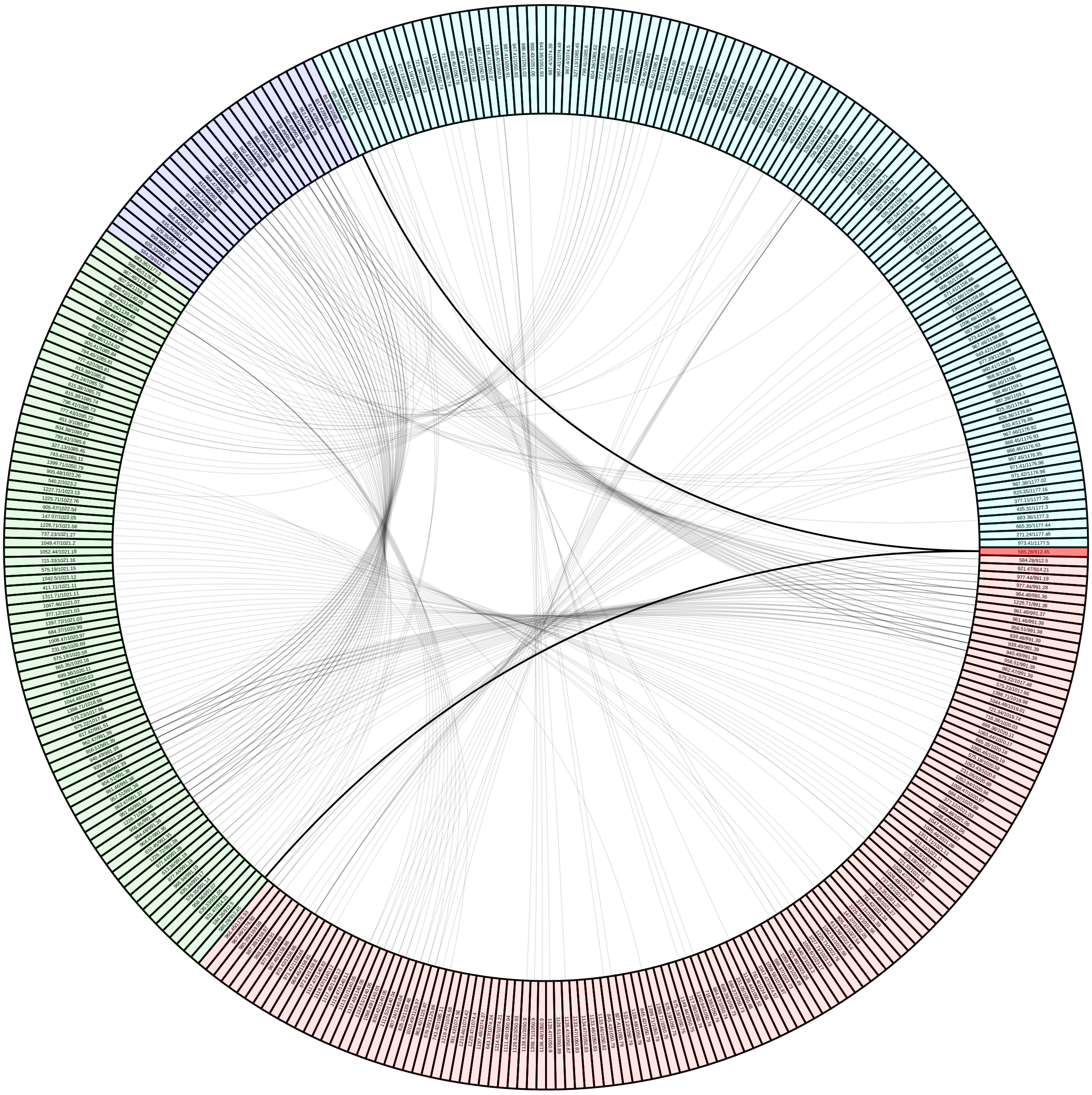

Figure 2: Examplary plot where arbitrary features are highlighted. Upon highlighting all links will be plotted in grey (expect links to and from highlighted features). The intensity of the background colour of features will be reduced as well. Features belonging to a group (species, individual, organ,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

References

Gu, Z. et al. (2014). "circlize implements and enhances circular visualization in R". In: *Bioinformatics* 30, pp. 2811–2812. DOI: 10.1093/bioinformatics/btu393.

Li, D., I.T. Baldwin, and E. Gaquerel (2015). "Navigating natural variation in herbivory-induced secondary metabolism in coyote tobacco populations using MS/MS structural analysis". In: *PNAS* 112, E4147–E4155. DOI: 10.1073/pnas.1503106112.

Appendix

Session information

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

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## 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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] MetCirc_1.0.1  shiny_0.14.2   scales_0.4.1   circlize_0.3.9
## [5] amap_0.8-14    knitr_1.15.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.8          digest_0.6.10        mime_0.5
##  [4] R6_2.2.0             grid_3.3.2           plyr_1.8.4
##  [7] xtable_1.8-2         magrittr_1.5         evaluate_0.10
## [10] highr_0.6            stringi_1.1.2        GlobalOptions_0.0.10
## [13] tools_3.3.2          stringr_1.1.0        munsell_0.4.3
## [16] httpuv_1.3.3         colorspace_1.3-2     shape_1.4.2
## [19] htmltools_0.3.5
```

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