

Supplementary file to "MetCirc: Navigating mass spectral similarity in high-resolution MS/MS metabolomics data"

Naake, Thomas¹ and Gaquerel, Emmanuel¹

¹ Centre for Organismal Studies, University of Heidelberg, Im Neuenheimer Feld 360, Heidelberg 69120, Germany.

1 Availability of the vignettes

The Bioconductor page for `MetCirc` hosts the vignettes for the `release` and the `devel` branch.

2 Installation of MetCirc

The `MetCirc` package is installed via the Bioconductor project by entering:

```
source("https://bioconductor.org/biocLite.R")
biocLite("MetCirc")
```

3 'A typical pipeline'

This section will give more detailed information on how to run the `MetCirc` pipeline than described in the main text. The section includes a subsection on how to reproduce the figure in the main text. We will use the `tissue` data set that accompanies the package to navigate through the analysis pipeline.

Load the `MetCirc` package and the `tissue` data set

```
library(MetCirc)
data("idMSMStissueproject", package = "MetCirc")
```

3.1 Annotation of known MS/MS features

In the following, we will use prior information to annotate MS/MS features. Attenoside, Lyciumoside II, Lyciumoside IV, Nicotianoside I, Nicotianoside II, Nicotianoside VI, Nicotianoside VII and Nicotianaoside XI belong to the the metabolic class of 17-hydroxygeranylinalool diterpene glycosides (HGL-DTG). We will create a column "names" and "classes" to store meta-data of MS/MS data within the `MSMS` object.

```

## create column names and assign to MSMS
MSMS <- cbind(tissue, names = rep("Unknown", 37863))
MSMS[, "names"] <- as.character(MSMS[, "names"])
MSMS <- cbind(MSMS, classes = rep("Unknown", 37863))
MSMS[, "classes"] <- as.character(MSMS[, "classes"])

att <- which(MSMS[, "id"] == "961.4618056_990.172976_50")
MSMS[att, "names"] <- "Attenoside"
MSMS[att, "classes"] <- "HGL-DTG"

lycII <- which(MSMS[, "id"] == "815.4041363_998.770992_265")
MSMS[lycII, "names"] <- "Lyciumoside II"
MSMS[lycII, "classes"] <- "HGL-DTG"

lycIV <- which(MSMS[, "id"] == "799.4097129_1086.734472_57")
MSMS[lycIV, "names"] <- "Lyciumoside IV"
MSMS[lycIV, "classes"] <- "HGL-DTG"

nicI <- which(MSMS[, "id"] == "885.4097679_1125.859976_74")
MSMS[nicI, "names"] <- "Nicotianoside I"
MSMS[nicI, "classes"] <- "HGL-DTG"

nicII <- which(MSMS[, "id"] == "971.4105174_1178.289448_79")
MSMS[nicII, "names"] <- "Nicotianoside II"
MSMS[nicII, "classes"] <- "HGL-DTG"

nicVI <- which(MSMS[, "id"] == "1047.45788_1025.362472_103")
MSMS[nicVI, "names"] <- "Nicotianoside VI"
MSMS[nicVI, "classes"] <- "HGL-DTG"

nicVII <- which(MSMS[, "id"] == "1133.464104_1049.943472_53")
MSMS[nicVII, "names"] <- "Nicotianoside VII"
MSMS[nicVII, "classes"] <- "HGL-DTG"

nicXI <- which(MSMS[, "id"] == "901.3833707_1124.966976_74")
MSMS[nicXI, "names"] <- "Nicotianoside XI"
MSMS[nicXI, "classes"] <- "HGL-DTG"

```

3.2 Creation of tissue-specific MSP objects

The binary matrix `compartmentTissue` stores information in which compartment MS/MS features were detected. The matrix is used to create MSP objects that contain tissue-specific

MS/MS features. We will therefore first create character vectors that contain features that are present in the groups sepal (SPL), limb (LIM), anther (ANT) and style (STY). Second, we will truncate MSMS and create MSMS_GROUP objects for four different tissue (MSMS_GROUP is a subset of MSMS for each tissue). Third, we will create tissue-specific MSP objects and combine these to a MSP object comprising all MS/MS features:

```
MSMS_features_SPL <- compartmentTissue[compartmentTissue[, "SPL"] == TRUE, 1]
MSMS_features_LIM <- compartmentTissue[compartmentTissue[, "LIM"] == TRUE, 1]
MSMS_features_ANT <- compartmentTissue[compartmentTissue[, "ANT"] == TRUE, 1]
MSMS_features_STY <- compartmentTissue[compartmentTissue[, "STY"] == TRUE, 1]

MSMS_SPL <- MSMS[MSMS[, "id"] %in% MSMS_features_SPL,]
MSMS_LIM <- MSMS[MSMS[, "id"] %in% MSMS_features_LIM,]
MSMS_ANT <- MSMS[MSMS[, "id"] %in% MSMS_features_ANT,]
MSMS_STY <- MSMS[MSMS[, "id"] %in% MSMS_features_STY,]

finalMSP <- convert2MSP(MSMS_SPL, rt = TRUE,
                         splitIndRT = 2, names = TRUE, classes = TRUE)
finalMSP <- combine(finalMSP, convert2MSP(MSMS_LIM, rt = TRUE,
                                             splitIndRT = 2, names = TRUE, classes = TRUE))
finalMSP <- combine(finalMSP, convert2MSP(MSMS_ANT, rt = TRUE,
                                             splitIndRT = 2, names = TRUE, classes = TRUE))
finalMSP <- combine(finalMSP, convert2MSP(MSMS_STY, rt = TRUE,
                                             splitIndRT = 2, names = TRUE, classes = TRUE))
```

For the binning function used below we will create a compartment vector that contains the group affiliation:

```
compSPL <- rep("SPL", length(convert2MSP(MSMS_SPL)))
compLIM <- rep("LIM", length(convert2MSP(MSMS_LIM)))
compANT <- rep("ANT", length(convert2MSP(MSMS_ANT)))
compSTY <- rep("STY", length(convert2MSP(MSMS_STY)))
compartment <- c(compSPL, compLIM, compANT, compSTY)
```

3.3 Binning, creation of similarity matrix and visualization

Bin the m/z values of fragments using a tolerance value of 0.01:

```
binnedMSP <- binning(msp = finalMSP, tol = 0.01, group = compartment)
```

Create the similarity matrix by entering:

```
similarityMat <- createSimilarityMatrix(binnedMSP)
```

At this step, interactive visualization using `shinyCircos` can be started by entering:

```
shinyCircos(similarityMat, finalMSP)
```

3.4 Visualization of highly-related features using clustering analysis

To display highly-similar MS/MS features we can apply clustering analysis to define modules that contain MS/MS features with similar spectra. We will create here three modules:

```
hClust <- hcluster(similarityMat, method = "spearman")
cutTree <- cutree(hClust, k = 3)
module1 <- names(cutTree)[as.vector(cutTree) == "1"]
module2 <- names(cutTree)[as.vector(cutTree) == "2"]
module3 <- names(cutTree)[as.vector(cutTree) == "3"]
```

Here, we will continue with module 3. We create first a vector `groupname` that is mapped against the names of the features in the module 3. We then truncate the MSP object and the `compartment` vector such that they only contain features found in module 3.

```
groupname <- paste(compartment, "_", getPrecursorMZ(finalMSP),
                     "/", getRT(finalMSP), sep = "")
module_ind <- match(module3, groupname)

finalMSP_module <- finalMSP[module_ind]
compartment_module <- compartment[module_ind]
```

Next, we bin the m/z values by using a tolerance parameter of 0.01:

```
binnedMSP_module <- binning(msp = finalMSP_module, tol = 0.01,
                               group = compartment_module, method = "median")
```

Create the similarity matrix:

```
similarityMat_module <- createSimilarityMatrix(binnedMSP_module)
```

Start the interactive visualization of MS/MS features of module 3 by entering

```
shinyCircos(similarityMat_module, finalMSP_module)
```

3.5 Script to reproduce figure 1 in the main text

To create the circular plots that are displayed in the main text run the following section. The respective pdf files will be created in the current working directory.

Create ordered similarity matrix:

```
simMat_order <- createOrderedSimMat(similarityMat_module, order = "clustering")
```

Create the link matrix with the thresholds 0.7-1:

```
link0Matrix <- createLink0Matrix(simMat_order)
LinkMatrix_cut <- cutLinkMatrix(link0Matrix, "all")
LinkMatrix_threshold <- thresholdLinkMatrix(LinkMatrix_cut, 0.70, 1)
```

Define general parameters using the `circlize` package:

```
circos.par(gap.degree = 0, cell.padding = c(0, 0, 0, 0),
           track.margin = c(0.0, 0))
```

Create circular plot where Nicotianoside I is selected:

```
pdf(file = "./nicotianosideI.pdf")
circos.initialize(factor(colnames(simMat_order)), xlim = matrix(rep(c(0,1),
                                                               dim(simMat_order)[1])), ncol = 2, byrow = TRUE)
circos.trackPlotRegion(colnames(simMat_order), ylim=c(0,1))
plotCircos(colnames(simMat_order), LinkMatrix_threshold,
           initialize=FALSE, featureNames = TRUE,
           groupSector = TRUE, groupName = FALSE,
           links = TRUE, highlight = TRUE, cexFeatureNames = 0.3)
## select and highlight Nicotianoside I
highlight(colnames(simMat_order),
          ind = which(colnames(simMat_order) == "STY_0090_885.4097679/1125.859976"),
          LinkMatrix = LinkMatrix_threshold, links = FALSE)
dev.off()
```

Create the circular plot where Nicotianoisde VI is selected:

```
pdf(file = "./nicotianosideVI.pdf")
circos.initialize(factor(colnames(simMat_order)), xlim = matrix(rep(c(0,1),
                                                               dim(simMat_order)[1])), ncol = 2, byrow = TRUE)
circos.trackPlotRegion(colnames(simMat_order), ylim=c(0,1))
plotCircos(colnames(simMat_order), LinkMatrix_threshold,
           initialize=FALSE, featureNames = TRUE,
           groupSector = TRUE, groupName = FALSE,
```

```

    links = TRUE, highlight = TRUE, cexFeatureNames = 0.3)
## select Nicotianoside VI
highlight(colnames(simMat_order),
  ind = which(colnames(simMat_order) == "ANT_0065_1047.45788/1025.362472"),
  LinkMatrix = LinkMatrix_threshold, links = FALSE)
dev.off()

```

To build the legend enter:

```

## create legend
pdf(file = "./legend.pdf")
circosLegend(c("anther", "limb", "style"), highlight = TRUE)
dev.off()

```

4 Session information

Software information, installed packages and respective versions that were used when building the figure 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=de_DE.UTF-8          LC_COLLATE=en_US.UTF-8
## [5] LC_MONETARY=de_DE.UTF-8       LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=de_DE.UTF-8          LC_NAME=C
## [9] LC_ADDRESS=C                  LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8   LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics   grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] MetCirc_1.1.4   shiny_1.0.0    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_0.12.9      digest_0.6.11    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
```