ANM_Lipid_WCNA

Asger Wretlind

2023-10-04

this script is based of a tutorial on WGCNA from Lindseynicer https://github.com/Lindseynicer/WGCNA_tutorial

```
#Load libraries
library(tidyverse)
```

```
## Warning: pakke 'ggplot2' blev bygget under R version 4.3.1

## Warning: pakke 'purrr' blev bygget under R version 4.3.1

## Warning: pakke 'dplyr' blev bygget under R version 4.3.1

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(here)
```

```
## here() starts at H:/Desktop/ANM_PRS/ANM_Data_Analysis
```

```
library(vroom)
```

```
## Warning: pakke 'vroom' blev bygget under R version 4.3.1

##
## Vedhæfter pakke: 'vroom'
##
## De følgende objekter er maskerede fra 'package:readr':
##
##     as.col_spec, col_character, col_date, col_datetime, col_double,
##     col_factor, col_guess, col_integer, col_logical, col_number,
##     col_skip, col_time, cols, cols_condense, cols_only, date_names,
##     date_names_lang, date_names_langs, default_locale, fwf_cols,
##     fwf_empty, fwf_positions, fwf_widths, locale, output_column,
##     problems, spec
```

```r
library(WGCNA)
```

```
## Warning: pakke 'WGCNA' blev bygget under R version 4.3.1
```

```
## Indlæser krævet pakke: dynamicTreeCut
## Indlæser krævet pakke: fastcluster
##
## Vedhæfter pakke: 'fastcluster'
##
## Det følgende objekt er maskeret fra 'package:stats':
##
##     hclust
##
##
##
## Vedhæfter pakke: 'WGCNA'
##
## Det følgende objekt er maskeret fra 'package:stats':
##
##     cor
```

```r
library(ggsci)

#Set Seed
set.seed(123)

#Set color palette
color_palette <- c("#11A1B7", "#FF660C", "#0CA61E", "#FE3C1A",
                   "#9966CC", "#4DDF2C", "#FE5387", "#85D0AB",
                   "#18548A", "#FCBB0B", "#FD908F", "#DF56BD", "#F0E4AD")

color_palette2 <- c("#935116", "#b7950b", "#ec407a", "#48c9b0","#ec7063",
                    "#2874a6", "#7d3c98", "#f8c471", "#1e8449","#C71B42")
```

```r
#Use residuals
use_residuals <- TRUE

#Network type
network_type <- "signed" #"signed" or "unsigned"

#Correlation function
correlation_function <- "bicor" #"cor" or "bicor"

#Clustering method
clustering_method <- "average" #"average" or "ward.D"

#Minimum cluster size
min_cluster_size <- 13

#Cutoff for dendrogram merging
dendrogram_cutoff <- 0.15
```

```r
#Load data
data <- vroom(here("data/ANM_Lipid_Preprocessed_v4.csv"))
```

```
## Rows: 841 Columns: 293
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr   (8): ID, Site, Date, Status, Sex, DOB, Accommodation, Marital_Status
## dbl (285): Visit, Order, Label, Age, Fulltime_Education_Years, apoe, e4_p, e...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#Subset lipid data
data_lipid <- tibble(data) %>%
    select(`Cer(d42:0)`:length(.)) %>%
    as.matrix()

#Set rownames of data_lipid to IDs
rownames(data_lipid) <- data$ID
```

```r
#Run this chunk if residuals are needed
if(use_residuals){

#Function to clear special characters that may create confusion
Clean_names <- function(lipid_names){
    tmp_lipid_names <- gsub("\\(", "", lipid_names)
    tmp_lipid_names <- gsub("\\:", "", tmp_lipid_names)
    tmp_lipid_names <- gsub("\\.", "", tmp_lipid_names)
    tmp_lipid_names <- gsub("\\)", "", tmp_lipid_names)
    tmp_lipid_names <- gsub("\\-", "", tmp_lipid_names)
    tmp_lipid_names <- gsub("\\/", "", tmp_lipid_names)

    return(tmp_lipid_names)
}

#temporary data set
tmp_data <- data %>%
    filter(!is.na(Age))

colnames(tmp_data) <- Clean_names(colnames(tmp_data))

#Loop iterating over each lipid and overwrite with residuals
for (i in colnames(tmp_data)[(which(colnames(tmp_data) %in% "Cerd420")):length(tmp_data)]){

    tmp_formula <- as.formula(paste0(i, "~ Age + Site"))

    tmp_model <- glm(tmp_formula, data = tmp_data)

    tmp_data[, i] <- residuals(tmp_model)

}

#Set names back to original
```

```
colnames(tmp_data) <- colnames(data)

#Use residual data
data <- tmp_data

#Subset lipid data
data_lipid <- tibble(data) %>%
    select(`Cer(d42:0)`:length(.)) %>%
    as.matrix()

#Set rownames of data_lipid to IDs
rownames(data_lipid) <- data$ID

rm(tmp_data, tmp_model, tmp_formula, i, Clean_names)

}

rm(use_residuals)
```

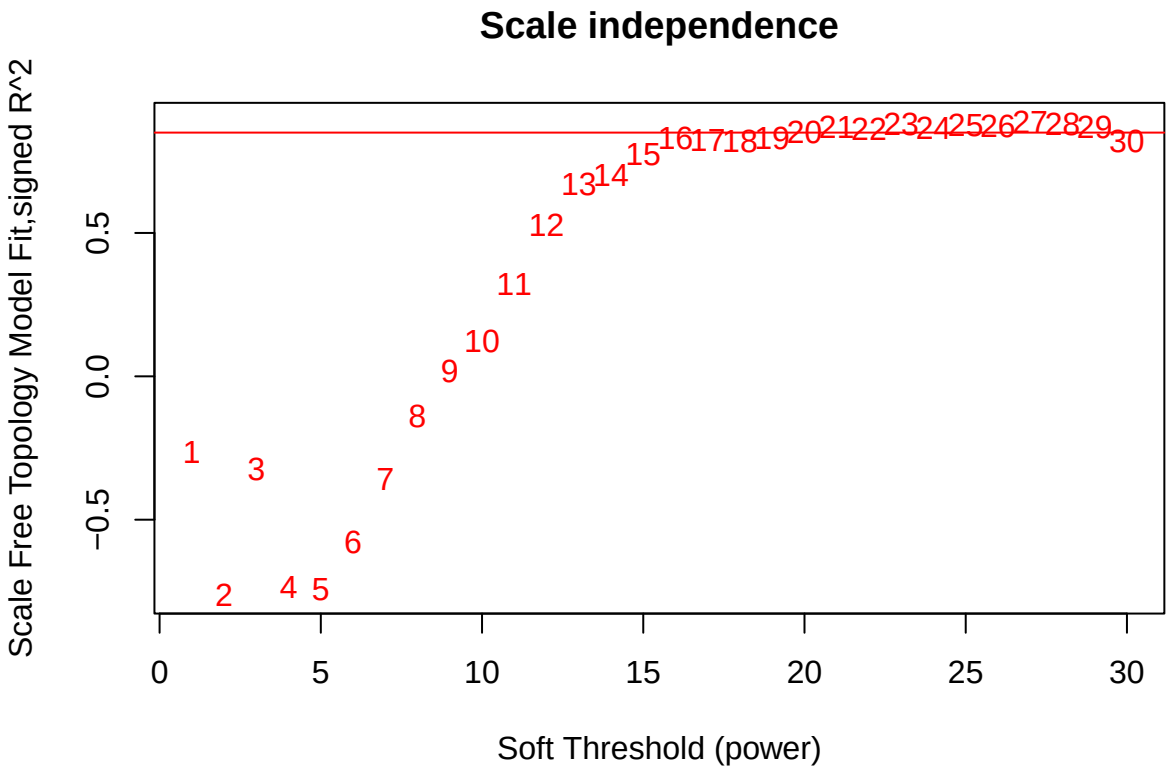
```
#Set soft threshold parameters
tmp_soft_threshold <- pickSoftThreshold(data_lipid,
                        powerVector = c(1:30),
                        networkType = network_type,
                        corFnc = correlation_function)
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
##    Power SFT.R.sq   slope truncated.R.sq mean.k. median.k. max.k.
## 1      1   0.2650 19.1000         0.0882  172.00   173.000 192.00
## 2      2   0.7610  4.8200         0.6970  115.00   115.000 141.00
## 3      3   0.3220  7.6700         0.1280   79.10    79.500 104.00
## 4      4   0.7350  2.2000         0.6590   56.00    56.200  78.30
## 5      5   0.7440  1.4600         0.6990   40.80    40.400  62.50
## 6      6   0.5790  0.8950         0.6750   30.50    29.900  51.90
## 7      7   0.3570  0.5520         0.7360   23.40    22.400  44.20
## 8      8   0.1380  0.2840         0.8650   18.30    17.000  38.30
## 9      9   0.0177 -0.0923         0.8590   14.70    13.200  33.80
## 10    10   0.1240 -0.2790         0.7050   12.00    10.600  30.20
## 11    11   0.3220 -0.4510         0.6940    9.94     8.360  27.30
## 12    12   0.5270 -0.6680         0.6220    8.38     6.750  24.90
## 13    13   0.6690 -0.7860         0.6670    7.17     5.570  22.90
## 14    14   0.7040 -0.8980         0.6460    6.21     4.630  21.20
## 15    15   0.7730 -0.9200         0.7260    5.44     3.870  19.70
## 16    16   0.8330 -0.9450         0.7850    4.82     3.280  18.30
## 17    17   0.8270 -0.9940         0.7840    4.30     2.790  17.20
## 18    18   0.8190 -1.0000         0.7810    3.87     2.430  16.10
## 19    19   0.8320 -1.0000         0.8120    3.50     2.080  15.20
## 20    20   0.8520 -1.0200         0.8280    3.19     1.810  14.40
## 21    21   0.8690 -1.0200         0.8640    2.92     1.600  13.70
## 22    22   0.8650 -1.0500         0.8700    2.69     1.400  13.00
## 23    23   0.8780 -1.0500         0.8900    2.49     1.220  12.50
## 24    24   0.8650 -1.0500         0.8750    2.31     1.060  11.90
## 25    25   0.8750 -1.0600         0.8970    2.15     0.965  11.40
```

```
## 26    26   0.8730 -1.0800          0.8910    2.01      0.889  10.90
## 27    27   0.8880 -1.0700          0.9120    1.88      0.824  10.50
## 28    28   0.8800 -1.0900          0.9120    1.77      0.751  10.10
## 29    29   0.8690 -1.0900          0.9050    1.67      0.692   9.67
## 30    30   0.8190 -1.1000          0.8220    1.57      0.645   9.30
```

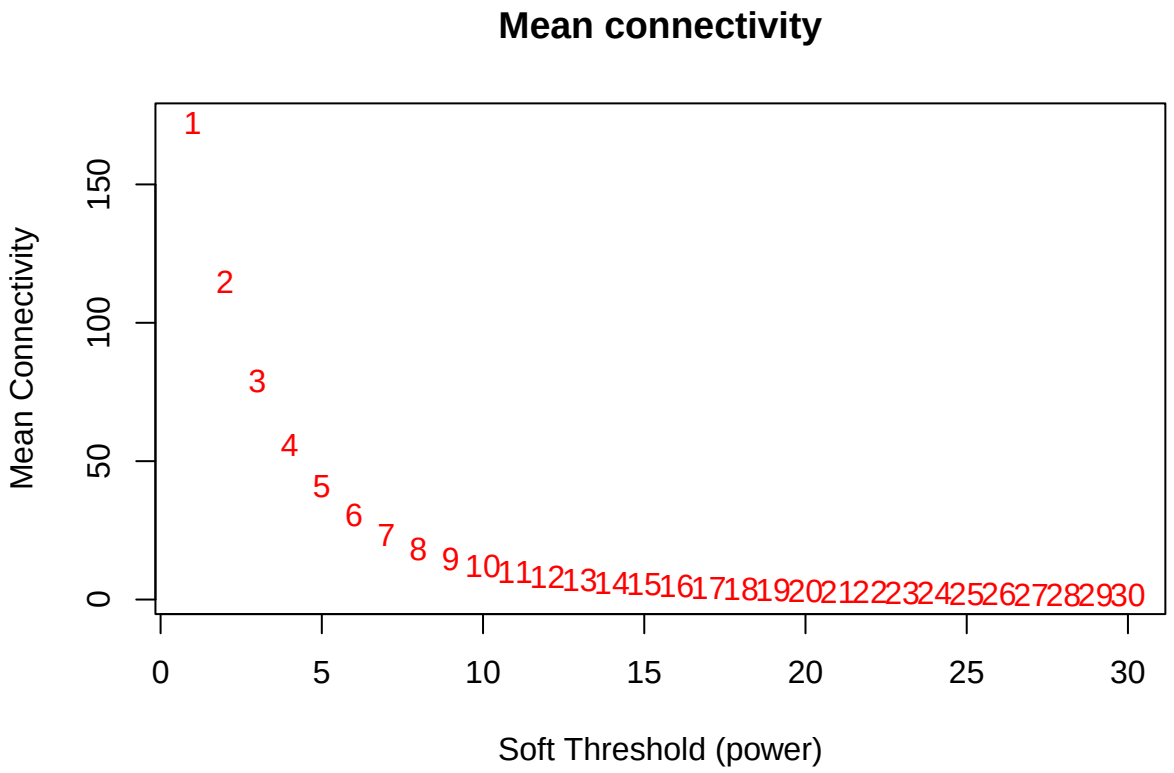
```
#Scale-free topology fit index as a function of the soft-thresholding power
plot(tmp_soft_threshold$fitIndices[,1],
     -sign(tmp_soft_threshold$fitIndices[,3])*
         tmp_soft_threshold$fitIndices[,2],
     xlab="Soft Threshold (power)",
     ylab="Scale Free Topology Model Fit,signed R^2", type="n",
     main = paste("Scale independence"))+
    text(tmp_soft_threshold$fitIndices[,1],
    -sign(tmp_soft_threshold$fitIndices[,3])*
        tmp_soft_threshold$fitIndices[,2],
    labels=c(1:30), col="red")+
    abline(h=0.85,col="red")
```




```
## integer(0)
```

```
#Mean connectivity as a function of the soft-thresholding power
plot(tmp_soft_threshold$fitIndices[,1], tmp_soft_threshold$fitIndices[,5],
    xlab="Soft Threshold (power)",
    ylab="Mean Connectivity", type="n",
```

```
    main = paste("Mean connectivity"))+
    text(tmp_soft_threshold$fitIndices[,1],
         tmp_soft_threshold$fitIndices[,5], labels=c(1:30),col="red")
```




```
## integer(0)
```

```
#Estimate power visually as the lower power for which the scale-free topology fit the index curve flatt
print(paste("Estimated soft power threshold:", tmp_soft_threshold$powerEstimate))
```

```
## [1] "Estimated soft power threshold: 20"
```

```
#Choose the power based on the graph reading
soft_power <- 15 #signed 15, unsigned 9

print(paste("Used soft power threshold:", soft_power))
```

```
## [1] "Used soft power threshold: 15"
```

```
rm(tmp_soft_threshold)

#Calculate network adjacency
adjacency <- adjacency(data_lipid,
                       power = soft_power,
```

```
                      type = network_type,
                      corFnc = correlation_function)

rm(soft_power)
```

```
#Temporary adjacency matrix with numbers for column names
tmp_adjacency <- adjacency
colnames(tmp_adjacency) <- as.character(seq(1:ncol(adjacency)))

#Set lipid edges
lipid_edges <- data.frame(tmp_adjacency) %>%
    mutate(Source = colnames(.)) %>%
    tibble() %>%
    pivot_longer(cols = where(is.double),
                 names_to = "Target", values_to = "Edge_value") %>%
    mutate(Source = as.integer(gsub("X", "", Source))) %>%
    mutate(Target = as.integer(gsub("X", "", Target))) %>%
    filter(Edge_value > 0.1 & Edge_value < .9999) #Important cutoff

#Set lipid nodes
lipid_nodes <- data.frame("Id" = seq(1:ncol(adjacency)),
                          "Name" = colnames(adjacency))
#Add lipid family
lipid_nodes <- tibble(lipid_nodes) %>%
    mutate("Family" = Name) %>%
    separate(Family, sep = "\\(", into = c("Family", "Left_over")) %>%
    select(-"Left_over")
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 34 rows [69, 70, 71, 72, 73,
## 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, ...].
```

```
#Add degree of connectedness
tmp_connect <- tibble(v = c(1:ncol(adjacency),
                            lipid_edges$Source, lipid_edges$Target))
lipid_nodes <- lipid_nodes %>%
    mutate("Degree" = count(tmp_connect, v)$n - 1)

#Number of connected lipids
print(paste0("Number of connected lipids: ", nrow(lipid_nodes[lipid_nodes$Degree > 0,]),
             " out of ", ncol(data_lipid)))
```

```
## [1] "Number of connected lipids: 245 out of 268"
```

```
#Create graph from data frames
library(igraph)
```

```
## Warning: pakke 'igraph' blev bygget under R version 4.3.1
```

```
##
## Vedhæfter pakke: 'igraph'
```

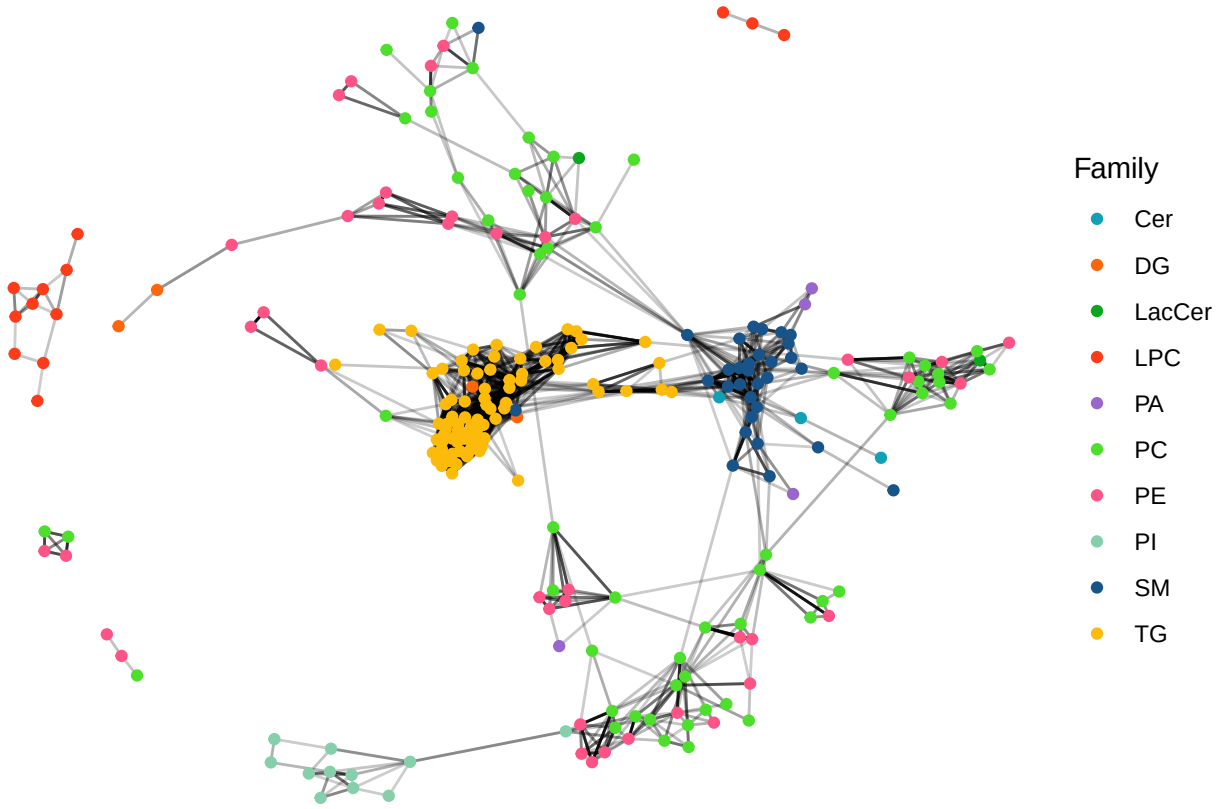
```
## De følgende objekter er maskerede fra 'package:lubridate':
##
##     %--%, union

## De følgende objekter er maskerede fra 'package:dplyr':
##
##     as_data_frame, groups, union

## De følgende objekter er maskerede fra 'package:purrr':
##
##     compose, simplify

## Det følgende objekt er maskeret fra 'package:tidyr':
##
##     crossing

## Det følgende objekt er maskeret fra 'package:tibble':
##
##     as_data_frame

## De følgende objekter er maskerede fra 'package:stats':
##
##     decompose, spectrum

## Det følgende objekt er maskeret fra 'package:base':
##
##     union
```

```
network_graph <- graph_from_data_frame(lipid_edges,
                           directed = FALSE,
                           lipid_nodes[lipid_nodes$Degree > 0,])

#Create network layout
library(ggraph)
```

```
## Warning: pakke 'ggraph' blev bygget under R version 4.3.1
```

```
network_layout = create_layout(network_graph, layout = "fr")


# #Save figure
# pdf(here(paste0("figures/Network_", network_type, "_", correlation_function,  "_v1.0.pdf")), width =

#Plot with ggraph
ggraph(network_layout) +
    geom_edge_link(aes(alpha = Edge_value),
                   show.legend = FALSE) +
    geom_node_point(aes(color = Family))+
    scale_color_manual(values = color_palette)+
    #geom_node_text(aes(label = Name), repel=TRUE)+
    theme_void()
```

```
## Warning: Using the ‘size‘ aesthetic in this geom was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ in the ‘default_aes‘ field and elsewhere instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



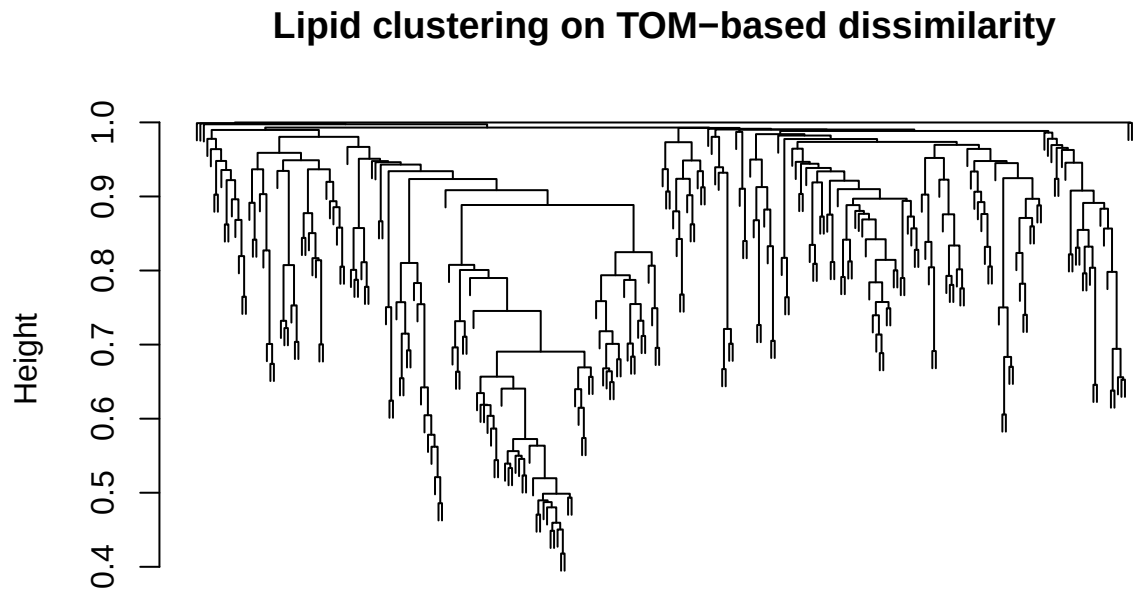

```
# dev.off()

rm(tmp_adjacency, tmp_connect, network_graph, network_layout)
```

```
#To minimize effects of noise and spurious associations, we transform the adjacency into TOM, and calcu

#Turn adjacency into topological overlap matrix (TOM)
TOM_dissimilarity <- 1-TOMsimilarity(adjacency, TOMType = network_type)
```

```
## ..connectivity..
## ..matrix multiplication (system BLAS)..
## ..normalization..
## ..done.
```

```
#Create dendrogram
dendrogram = hclust(as.dist(TOM_dissimilarity),
                    method = clustering_method)
```

```r
#Plot dendrogram
plot(dendrogram, xlab="", sub="",
     main = "Lipid clustering on TOM-based dissimilarity",
     labels = FALSE, hang = 0.04)
```




```r
#Use dynamic tree cut to identify modules
#and assign each lipid to a module
lipid_module_nr = cutreeDynamic(dendro = dendrogram,
                            method = "tree",
                            deepSplit = TRUE,
                            minClusterSize = min_cluster_size)

#Avoid module 0 by adding 1 to all module labels
lipid_module_nr <- lipid_module_nr+1

#Number and size of identified modules
print(paste0("Number of identified modules: ",
             length(unique(lipid_module_nr))))
```

```
## [1] "Number of identified modules: 12"
```

```r
table(lipid_module_nr)
```

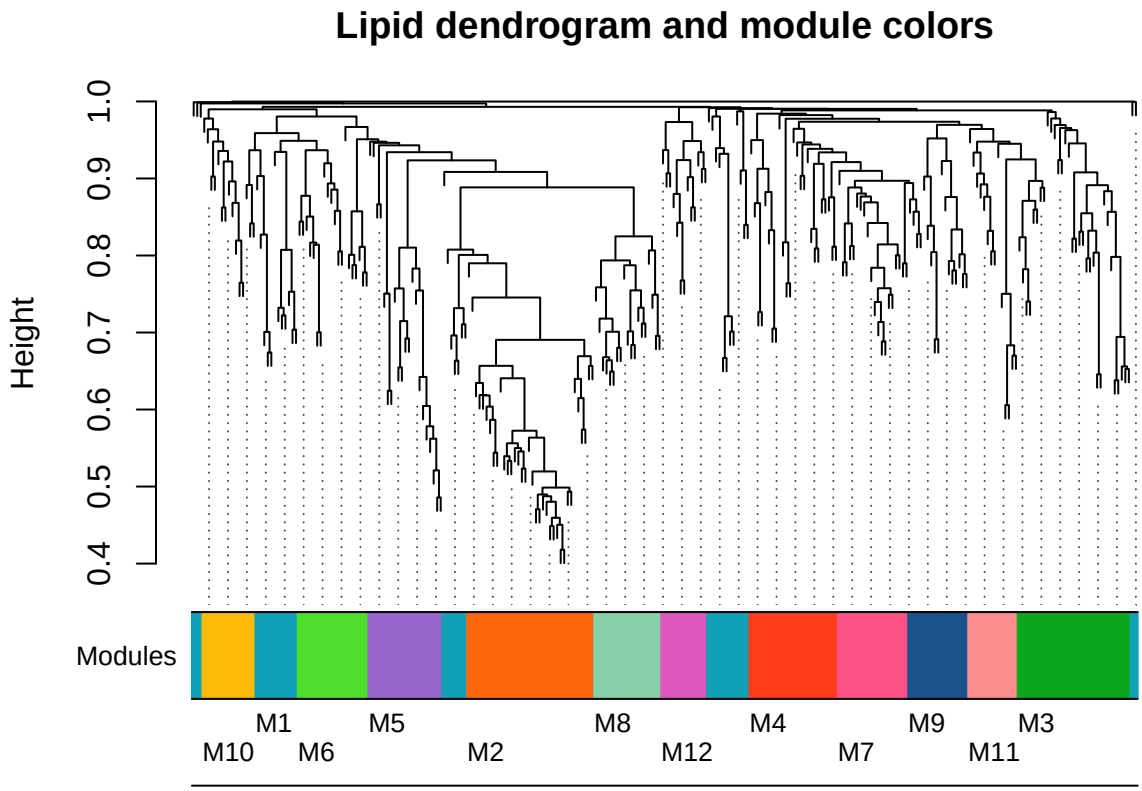
```
## lipid_module_nr
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12
## 36 36 32 25 21 20 20 19 17 15 14 13
```

```
#Convert numeric labels into colors
lipid_module_color <- labels2colors(lipid_module_nr,
                                    colorSeq = color_palette)
table(lipid_module_color)
```

```
## lipid_module_color
## #0CA61E #11A1B7 #18548A #4DDF2C #85D0AB #9966CC #DF56BD #FCBB0B #FD908F #FE3C1A
##      32      36      17      20      19      21      13      15      14      25
## #FE5387 #FF660C
##      20      36
```

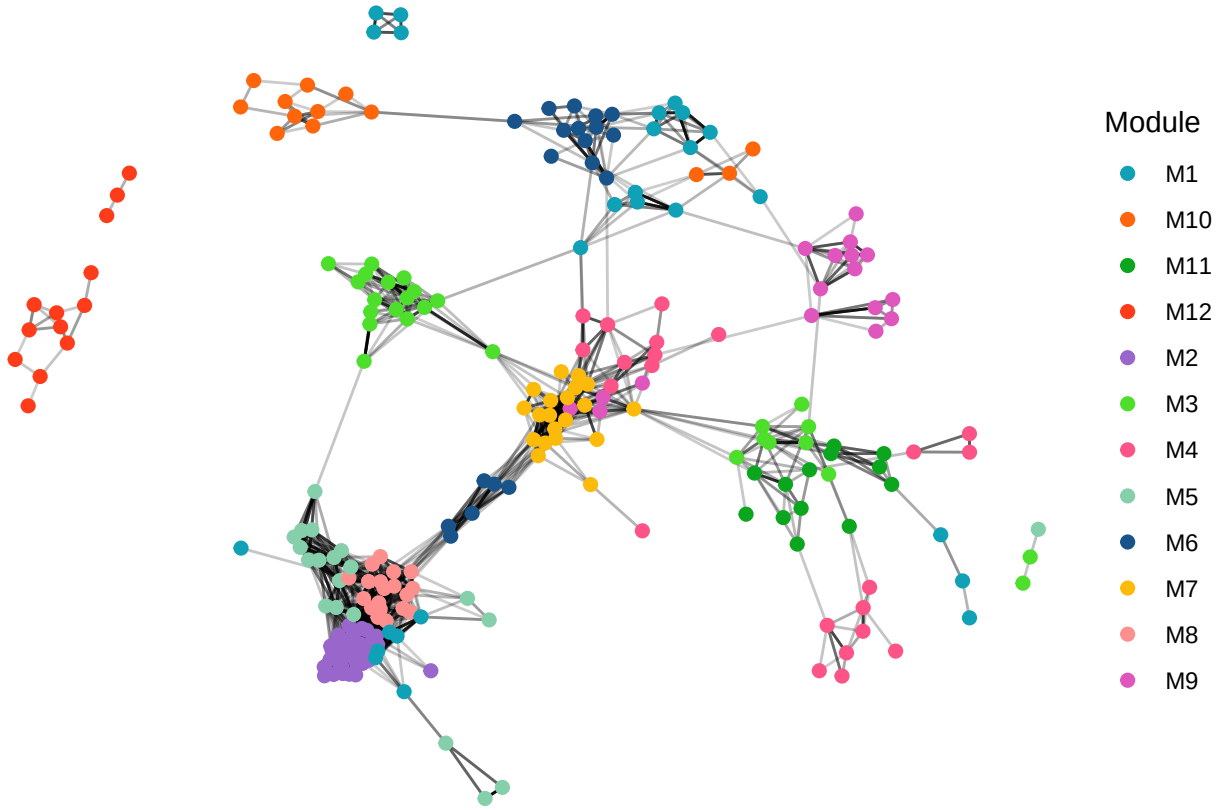
```
# #Save figure
# pdf(here(paste0("figures/Dendrogram_",
#                 gsub("\\.D", "", clustering_method),
#                 "_MinClusterSize_", min_cluster_size,  "_v1.0.pdf")),
#     width = 6, height = 4)

#Plot the dendrogram with colors under it
plotDendroAndColors(dendrogram, lipid_module_color, "Modules",
                    dendroLabels = FALSE,
                    hang = 0.03,
                    addGuide = TRUE,
                    guideHang = 0.05,
                    main = "Lipid dendrogram and module colors",
                    rowText = paste0("M", lipid_module_nr))
```




```
# dev.off()

rm(TOM_dissimilarity)
```

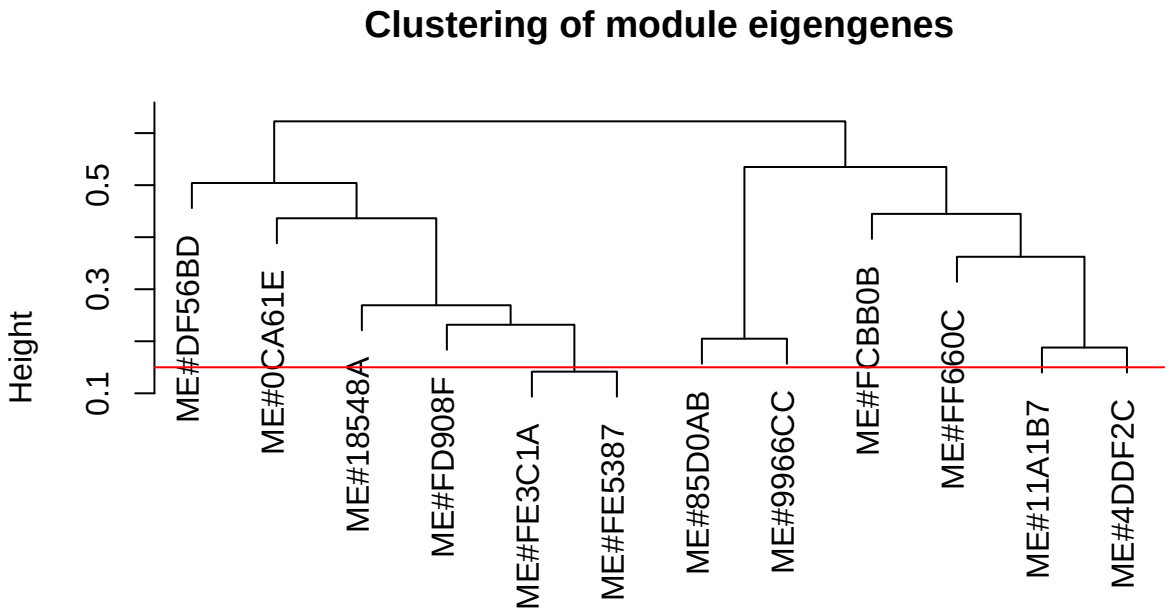
```
#Add module info to lipid nodes
lipid_nodes <- lipid_nodes %>%
    mutate("Module" = paste0("M", lipid_module_nr))

#Create graph from data frames
network_graph <- graph_from_data_frame(lipid_edges,
                           directed = FALSE,
                           lipid_nodes[lipid_nodes$Degree > 0,])

#Create network layout
network_layout = create_layout(network_graph, layout = "fr")

# #Save figure
# pdf(here(paste0("figures/Network_",
#                 network_type, "_Modules_by_",
#                 gsub("\\.D", "", clustering_method),
#                 "_v1.0.pdf")), width = 6, height = 5)

#Plot with ggraph
ggraph(network_layout) +
    geom_edge_link(aes(alpha = Edge_value),
                   show.legend = FALSE) +
```

```
    geom_node_point(aes(color = Module), size = 2)+
    scale_color_manual(values = color_palette)+
    #geom_node_text(aes(label = Name), repel=TRUE)+
    theme_void()
```




```
# dev.off()

rm(network_graph, network_layout)
```

```
#Merging of modules whose co-expression profiles are very similar

#To quantify co-expression similarity of entire modules, calculate their eigengenes and cluster them on

#Calculate eigengenes (first principal component) for each observation
tmp_eigengenes <- moduleEigengenes(data_lipid, colors = lipid_module_color)$eigengenes
eigengenes <- 1 - cor(tmp_eigengenes)

#Cluster new modules based original module eigengenes
dendrogram_eigen <- hclust(as.dist(eigengenes), method = clustering_method)

#Plot dendrogram
plot(dendrogram_eigen, main = "Clustering of module eigengenes", xlab = "", sub = "")+
    abline(h = dendrogram_cutoff, col = "red")
```




```
## integer(0)
```

```
#Merge modules into a new set of modules
tmp_merged_module_color <- mergeCloseModules(data_lipid,
                                     lipid_module_color,
                                     cutHeight = dendrogram_cutoff,
                                     verbose = 3)
```

```
##  mergeCloseModules: Merging modules whose distance is less than 0.15
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 12 module eigengenes in given set.
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 11 module eigengenes in given set.
##    Calculating new MEs...
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 11 module eigengenes in given set.
```

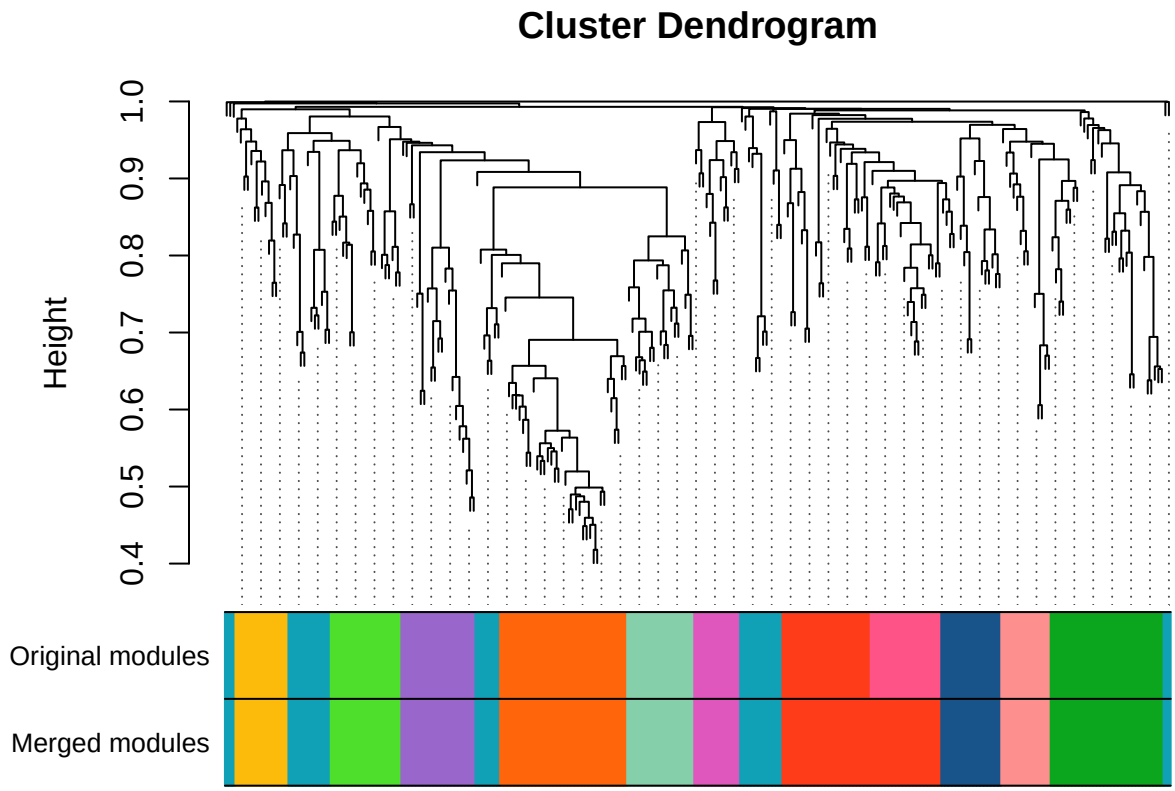
```
#Repeat to keep consistent module numbers
tmp_merged_module_nr <- mergeCloseModules(data_lipid, lipid_module_nr,
                           cutHeight = dendrogram_cutoff,
                           verbose = 3)
```

```
##  mergeCloseModules: Merging modules whose distance is less than 0.15
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 12 module eigengenes in given set.
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 11 module eigengenes in given set.
##    Calculating new MEs...
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 11 module eigengenes in given set.
```

```
#Resulting number of modules after merging
print(paste0("Merging with a cutoff at: ", dendrogram_cutoff,
             " resulted in ", length(unique(tmp_merged_module_color$colors)), " modules"))
```

```
## [1] "Merging with a cutoff at: 0.15 resulted in 11 modules"
```

```
##Change Module numbers to be in sequence
tmp_seq <- 1:max(unique(tmp_merged_module_nr$colors))
tmp_removed_modules <- tmp_seq[!tmp_seq %in% unique(tmp_merged_module_nr$colors)]

#Loop through each module number removed and substitute them instead of the largest modules number
for (i in 1:length(tmp_removed_modules)){

    #Correct in Module colors
    tmp_merged_module_nr$colors[tmp_merged_module_nr$colors == (max(unique(tmp_merged_module_nr$colors)))

    #Correct in Module numbers
    tmp_merged_module_nr$dendro[["labels"]][as.numeric(gsub("ME", "", tmp_merged_module_nr$dendro[["labe

}

#Bind colors and numbers together in a key
lipid_merged_modules <- data.frame(
    "Name" = paste0("M", tmp_merged_module_nr$colors),
    "Color" = tmp_merged_module_color$colors,
    "Lipid" = colnames(data_lipid))

#Plot original dendrogram with original module and merged modules
plotDendroAndColors(dendrogram, cbind(lipid_module_color, lipid_merged_modules$Color),
                    c("Original modules", "Merged modules"),
                    dendroLabels = FALSE,
                    hang = 0.03,
                    addGuide = TRUE,
                    guideHang = 0.05)
```




```
#Visualize merged module dendrogram for figure 1
library(dendextend)
```

```
## Warning: pakke 'dendextend' blev bygget under R version 4.3.2
```

```
##
## ---------------------
## Welcome to dendextend version 1.17.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Vedhæfter pakke: 'dendextend'
```

```
## Det følgende objekt er maskeret fra 'package:stats':
##
##     cutree
```

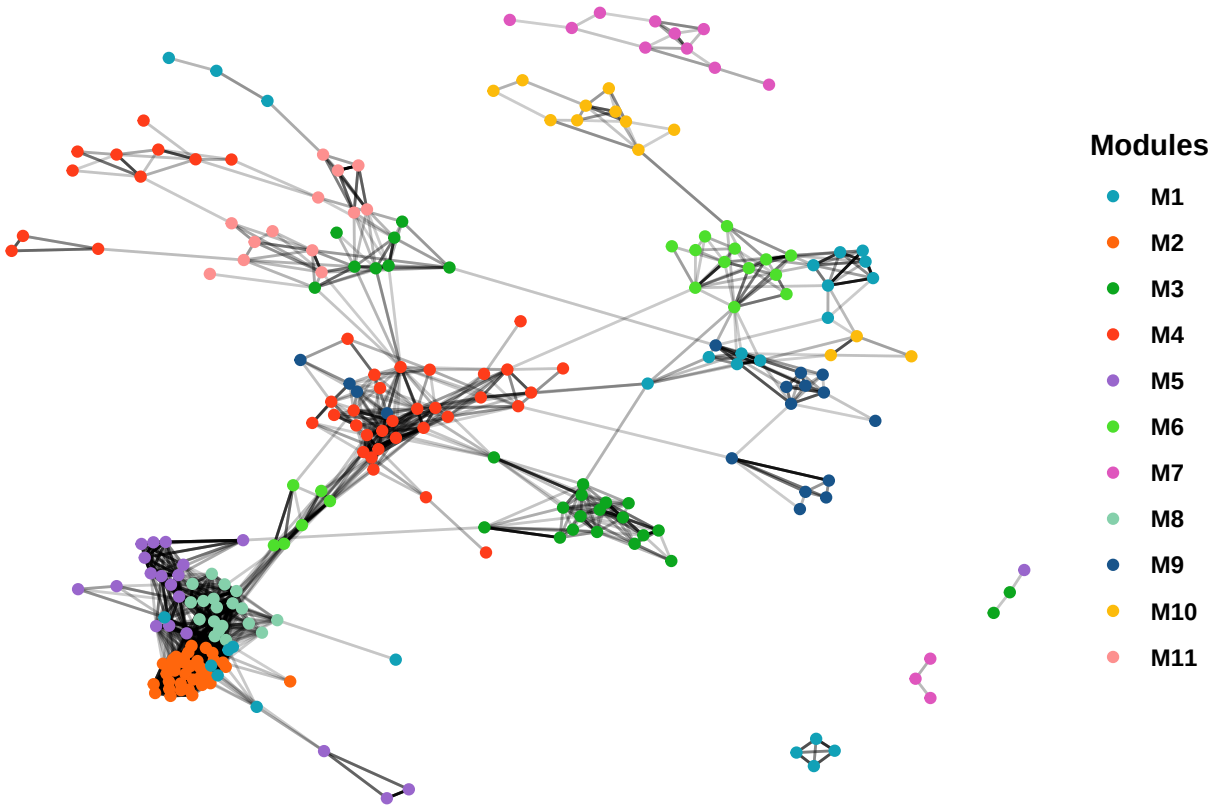
```r
#color order
tmp_color <- gsub("ME", "", tmp_merged_module_color$dendro[["labels"]])

#Save dendrogram for Figure 1 (Chunk 15)
fig1_dendro <- tmp_merged_module_nr$dendro %>%
    as.dendrogram %>%
    set("leaves_pch", 19) %>%
    set("leaves_col", tmp_color) %>%
    set("labels", "") %>%
    ggplot()
```

```
## Warning in `labels<-.dendrogram`(dend, value = value, ...): The lengths of the
## new labels is shorter than the number of leaves in the dendrogram - labels are
## recycled.
```

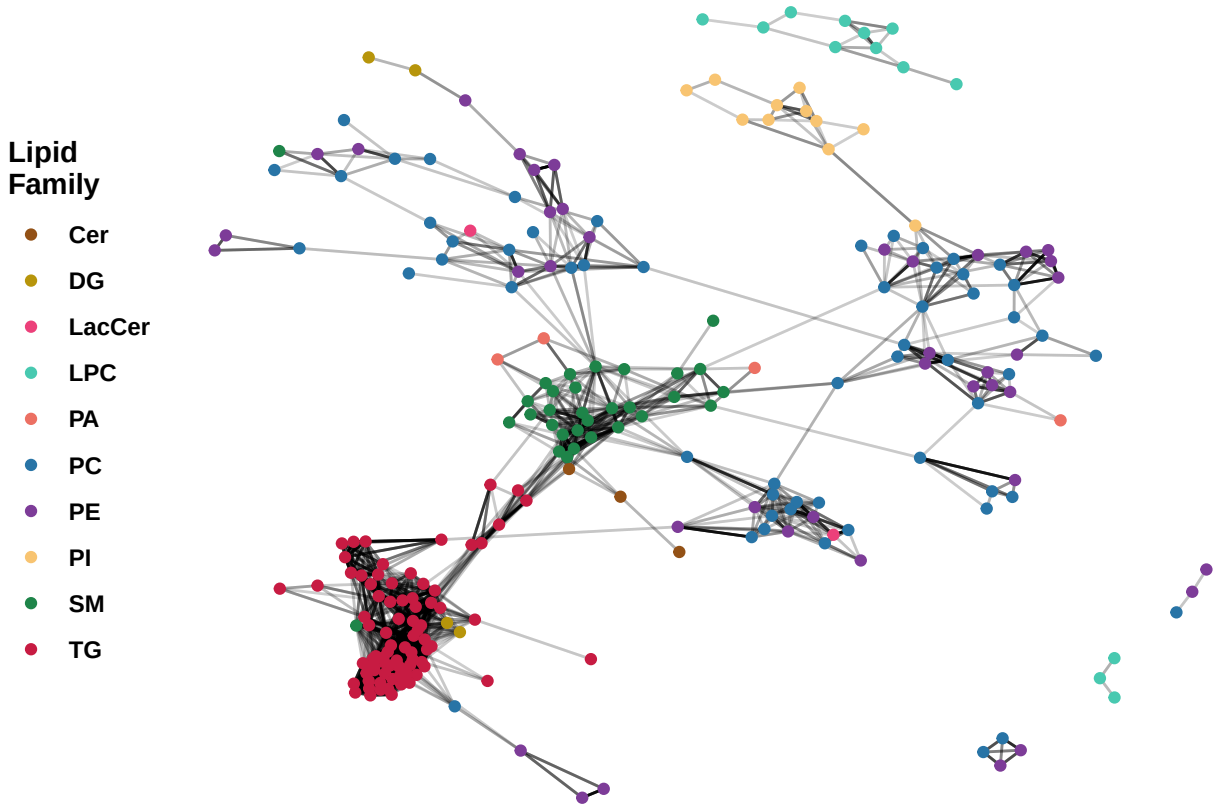
```r
#Module order dendrogram
module_order_dendrogram <- gsub("E", "", tmp_merged_module_nr$dendro[["labels"]])

#Modules in order
module_order <- unique(lipid_merged_modules$Name)[order(as.numeric(gsub("M", "", unique(lipid_merged_mo

#Cleaning
rm(dendrogram, dendrogram_eigen, eigengenes, tmp_eigengenes, tmp_merged_module_color, dendrogram_cutoff
```

```r
#Add merged module info to lipid nodes
lipid_nodes <- lipid_nodes %>%
    mutate("Merged_module" = lipid_merged_modules$Name)

#Create graph from data frames
network_graph <- graph_from_data_frame(lipid_edges,
                           directed = FALSE,
                           lipid_nodes[lipid_nodes$Degree > 0,])

#Create network layout
network_layout <- create_layout(network_graph, layout = "fr")

#Set color sequence for the merged modules
tmp_color <- unique(lipid_merged_modules$Color)[order(as.numeric(gsub("M", "", unique(lipid_merged_modul

#Plot with ggraph - Modules
fig1_network_module <- ggraph(network_layout) +
    geom_edge_link(aes(alpha = Edge_value),
                   show.legend = FALSE) +
    geom_node_point(aes(color = factor(Merged_module,
                                       levels = module_order)))+
    scale_color_manual(values = tmp_color)+
    #geom_node_text(aes(label = Name), repel=TRUE)+
    guides(color=guide_legend(title="Modules"))+
    theme_void()+
    theme(legend.title = element_text(face = "bold"),
          legend.text = element_text(face = "bold"),
          legend.position = "right" )
```

```
# #Save figure
# pdf(here(paste0("figures/Network_",
#                 network_type, "_Merged_Modules_by_",
#                 gsub("\\.D", "", clustering_method),
#                 "_v1.0.pdf")), width = 6, height = 5)

fig1_network_module
```




```
# dev.off()

#Plot with ggraph - lipid family
fig1_network_lipids <- ggraph(network_layout) +
    geom_edge_link(aes(alpha = Edge_value),
                   show.legend = FALSE) +
    geom_node_point(aes(color = Family))+
    scale_color_manual(values = color_palette2)+
    #geom_node_text(aes(label = Name), repel=TRUE)+
    guides(color=guide_legend(title="Lipid\nFamily"))+
    theme_void()+
    theme(legend.title = element_text(face = "bold"),
          legend.text = element_text(face = "bold"),
          legend.position = "left")

fig1_network_lipids
```



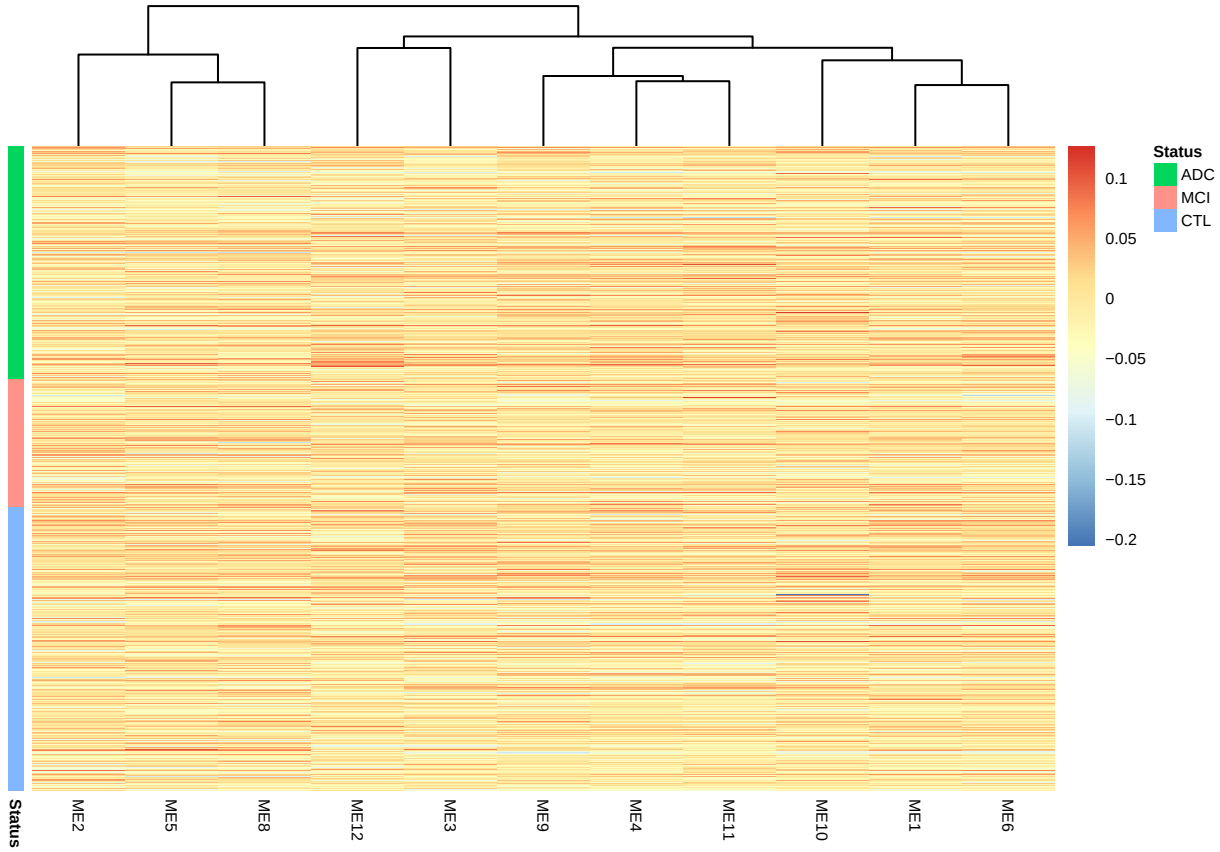

```
rm(network_graph, network_layout,
   clustering_method, correlation_function, min_cluster_size, network_type,
   adjacency, lipid_nodes, lipid_edges, tmp_color)
```

```
library("pheatmap")
```

```
## Warning: pakke 'pheatmap' blev bygget under R version 4.3.1
```

```
#Create key for AD status
AD_annotation <- tibble(data) %>%
    select(ID, Status) %>%
    mutate(Status = factor(Status, levels = c("ADC", "MCI", "CTL"))) %>%
    arrange(Status) %>%
    data.frame()

#Set rownames to ID to match with "merge"
rownames(AD_annotation) <- AD_annotation$ID
AD_annotation$ID <- NULL

#AD color key
AD_color <- list("AD_annotation" = c("ADC" = "red",
                                     "MCI" = "yellow",
                                     "CTL" = "green"))

#temporary data frame order by AD annotation
```

```
tmp_data <- data.frame(tmp_merged_module_nr$newMEs)
tmp_data <- tmp_data[order(match(rownames(tmp_data), rownames(AD_annotation))),]

#Heatmap of new module eigen-genes and sample trait (e.g. AD)
pheatmap(tmp_data,
         cluster_col=T,
         cluster_row=F,
         show_rownames=F,
         show_colnames=T,
         fontsize=6,
         annotation_row = AD_annotation, annotation_colors = AD_color)
```




```
rm(AD_annotation, AD_color, tmp_data, tmp_merged_module_nr)
```

```
#Recalculate module eigengenes with number labels
tmp_eigengenes <- moduleEigengenes(data_lipid, lipid_merged_modules$Name)$eigengenes

#Define lipid families
lipid_fam <- tibble(data) %>%
    select(`Cer(d42:0)`:length(.)) %>%
    pivot_longer(everything()) %>%
    separate(name, sep = "\\(", c("Family", "Chain1")) %>%
    pull(Family) %>%
    unique()
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 28118 rows [69, 70, 71, 72,
## 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, ...].
```

```
#Find lipid family mean
lipid_fam_df <- data.frame("ID" = data$ID)
for (i in lipid_fam){
    lipid_fam_df[, i] <- tibble(data) %>%
    select(starts_with(i)) %>%
    rowMeans()
}

#Move ID to row name
rownames(lipid_fam_df) <- lipid_fam_df$ID
lipid_fam_df$ID <- NULL

#Make sure rownames matches between lipid family and modules
lipid_fam_df <- lipid_fam_df[match(rownames(tmp_eigengenes), rownames(lipid_fam_df)),]
table(rownames(tmp_eigengenes) == rownames(lipid_fam_df))
```

```
##
## TRUE
##  827
```

```
#Calculate pearson correlation coefficients between module eigen-genes and traits
moduleTraitCor = cor(tmp_eigengenes, lipid_fam_df, use = "p")
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nrow(data_lipid))

#Will display correlations and their p-values
textMatrix =  paste(signif(moduleTraitCor, 2), "\n(",
                    signif(moduleTraitPvalue, 1), ")", sep = "")

#Organize module order
moduleTraitCor <- moduleTraitCor[order(as.numeric(gsub("MEM", "", rownames(moduleTraitCor)))),]

# #Save figure
# pdf(here("figures/Heatmap_lipid_family_v1.2.pdf"), width = 9, height = 6)

# #Plot heatmap
# labeledHeatmap(Matrix = moduleTraitCor,
#                xLabels = colnames(lipid_fam_df),
#                yLabels = gsub("ME", "", rownames(moduleTraitCor)),
#                ySymbols = gsub("ME", "", rownames(moduleTraitCor)),
#                colorLabels = FALSE,
#                colors = blueWhiteRed(50),
#                textMatrix = signif(moduleTraitCor, 2), #textMatrix,
#                #setStdMargins = FALSE,
#                #cex.text = 0.5,
#                zlim = c(-1,1),
#                main = paste("Module correlation to lipid family"))

# #Save fig end
# dev.off()
```

```
#### List lipids that correlated more than 0.65 with each module

#Organize correlation matrix of correlation between lipids and modules
lipid_family_correlation_matrix <- t(moduleTraitCor > 0.65)
colnames(lipid_family_correlation_matrix) <- gsub("E", "", colnames(lipid_family_correlation_matrix))

#Find the highest threshold were each module is correlated with at least one lipid family
colSums(lipid_family_correlation_matrix > 0.65)
```

```
##  MM1  MM2  MM3  MM4  MM5  MM6  MM7  MM8  MM9 MM10 MM11
##    4    1    4    5    2    3    1    2    5    1    4
```

```
#Empty dataframe to populate
module_correlated_lipids <- data.frame(row.names = colnames(lipid_family_correlation_matrix))

#Go through each module and populate with correlated lipids, output as a single string per module
for(i in colnames(lipid_family_correlation_matrix)){
    module_correlated_lipids[i, "Lipid"] <- paste(rownames(lipid_family_correlation_matrix)
                                [lipid_family_correlation_matrix[, i]],collapse = ", ")
    }

#Correlated lipids found in each module
module_correlated_lipids
```

```
##                     Lipid
## MM1      Cer, PC, SM, PE
## MM2                   TG
## MM3   LacCer, PC, SM, PE
## MM4  Cer, PC, SM, PA, PE
## MM5               DG, TG
## MM6           PC, SM, PE
## MM7                  LPC
## MM8               DG, TG
## MM9  Cer, PC, SM, PA, PE
## MM10                  PI
## MM11     Cer, PC, SM, PE
```

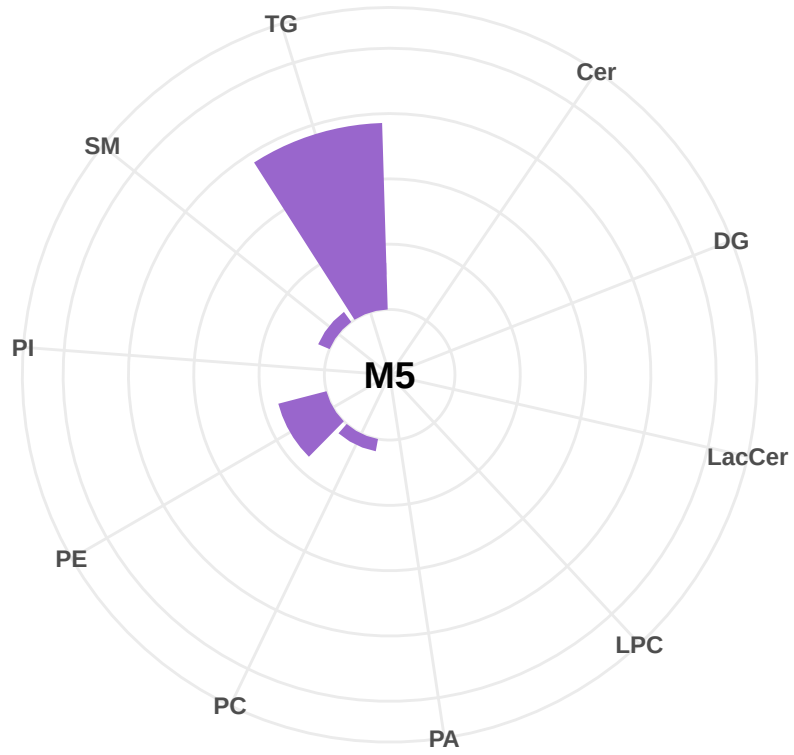
```
rm(i, moduleTraitCor, moduleTraitPvalue, textMatrix, lipid_fam_df, lipid_family_correlation_matrix,
   module_correlated_lipids, tmp_eigengenes)
```

```
### Plot a single module
#Module to investigate
module_x <- "M5"

#What lipids are included in X module
module_composistion <-
    colnames(data_lipid)[lipid_merged_modules$Name == module_x]

module_composistion
```

```
##  [1] "PC(33:1)_B" "SM(d44:1)"  "TG(50:5)_B" "TG(50:6)_B" "TG(52:6)_A"
```

```
##  [6] "TG(52:7)_A" "TG(54:6)_B" "TG(54:7)_A" "TG(54:8)_A" "TG(56:3)_B"
## [11] "TG(56:6)"   "TG(56:7)"   "TG(56:8)"   "TG(56:9)"   "TG(58:10)"
## [16] "TG(58:11)"  "TG(58:4)"   "PE(36:2)_B" "PE(36:3)_B" "PE(36:3)_D"
## [21] "PE(36:4)"
```

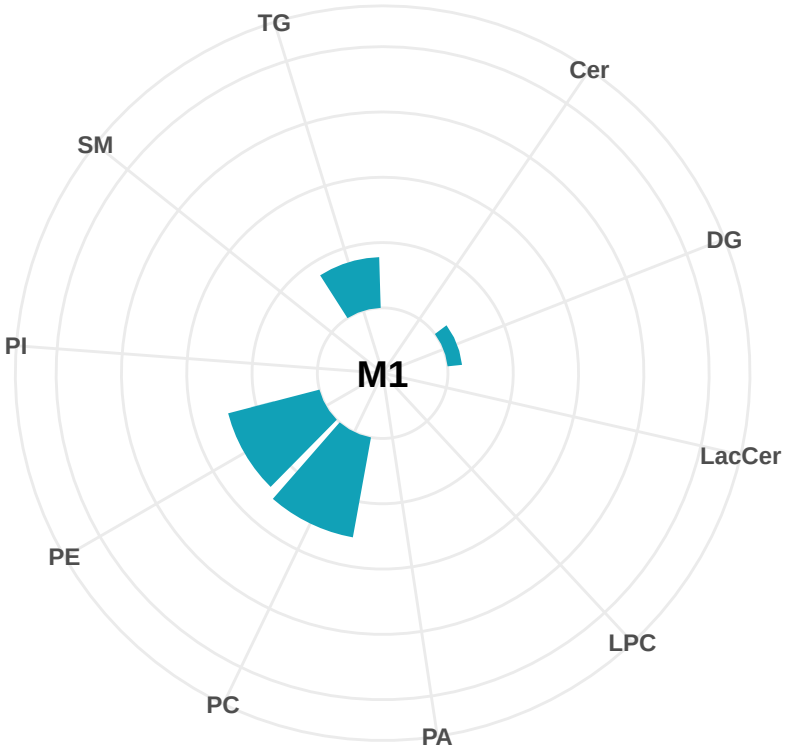
```r
#Number of lipids from each lipid family
tmp_lipids_included <- tibble(module_composistion) %>%
    separate(module_composistion,
             sep = "\\(",
             into = c("Family", "Left_over")) %>%
    select(-"Left_over") %>%
    count(Family)

#Lipid families not found in this module
tmp_lipids_excluded <- data.frame("Family" = lipid_fam[!lipid_fam %in% tmp_lipids_included$Family],
                                  "n" = rep(0, length(lipid_fam[!lipid_fam %in% tmp_lipids_included$Fami

#Module color
module_color <-
    lipid_merged_modules$Color[lipid_merged_modules$Name == module_x][1]

#Combine and plot
tmp_lipids_included %>%
    rbind(., tmp_lipids_excluded) %>%
    mutate("Ratio" = n / sum(n)) %>%
    ggplot(aes(x = Family, y = Ratio)) +
    geom_bar(stat = "identity", fill = c(module_color)) +
    ylim(-0.25, 1) +
    theme_minimal() +
    theme(
        axis.title = element_blank(),
        axis.text.y = element_blank(),
        #panel.grid = element_blank(),
        axis.text.x = element_text(face = "bold"),
        legend.position = c(0.5, 0.5)) +
    coord_polar(start = 0) +
    annotate("text", x = 0, y = -0.25, label = module_x, size = 5, fontface = "bold")
```



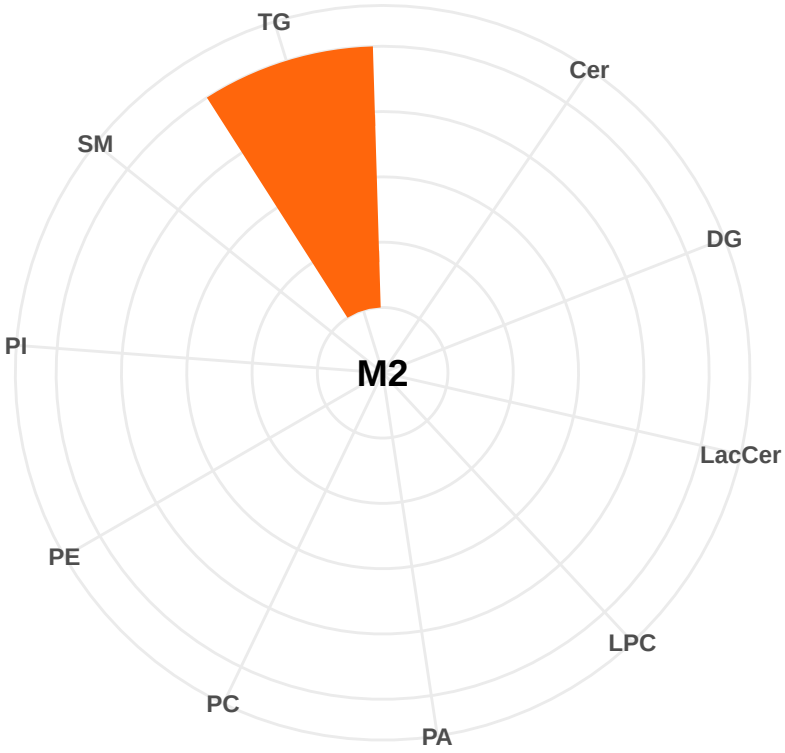

```
### plot all modules

circle_barplots <- list()
#Loop through each module
for(i in module_order){

    #What lipids are included in X module
    module_composistion <- colnames(data_lipid)[lipid_merged_modules$Name == i]

    #Number of lipids from each lipid family
    tmp_lipids_included <- tibble(module_composistion) %>%
        separate(module_composistion,
                 sep = "\\(", into = c("Family", "Left_over")) %>%
        select(-"Left_over") %>%
        count(Family)

    #Lipid families not found in this module
    tmp_lipids_excluded <- data.frame(
        "Family" = lipid_fam[!lipid_fam %in% tmp_lipids_included$Family],
        "n" = rep(0, length(lipid_fam[!lipid_fam %in% tmp_lipids_included$Family])))

    #Module color
    module_color <- lipid_merged_modules$Color[lipid_merged_modules$Name == i][1]

    #Combine and plot
    tmp_plot <- tmp_lipids_included %>%
```

```
        rbind(., tmp_lipids_excluded) %>%
        mutate("Ratio" = n/sum(n)) %>%
        ggplot(aes(x = Family, y = Ratio))+
            geom_bar(stat="identity", fill = c(module_color))+
            ylim(-0.25, 1)+
            theme_minimal()+
            theme(axis.title = element_blank(),
                  axis.text.y = element_blank(),
                  #panel.grid = element_blank(),
                  axis.text.x = element_text(face = "bold"),
                  legend.position = c(0.5, 0.5))+
        coord_polar(start = 0)+
        annotate("text", x = 0, y = -0.25, label = i,
                 size = 5, fontface = "bold")

    print(tmp_plot)
    circle_barplots[[i]] <- tmp_plot

}
```

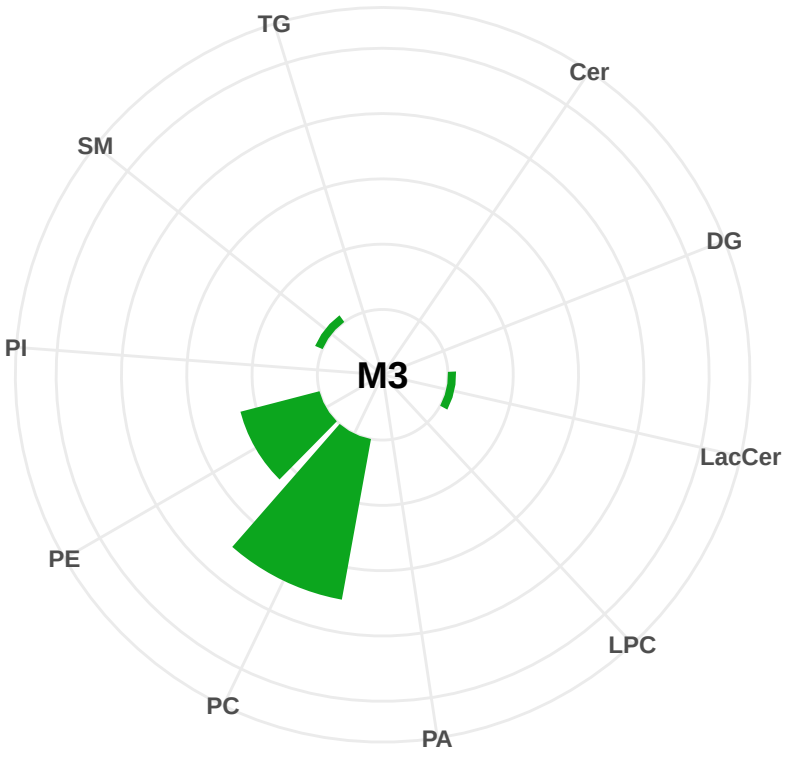
```
## Warning: Expected 2 pieces. Additional pieces discarded in 4 rows [32, 33, 34,
## 36].
```

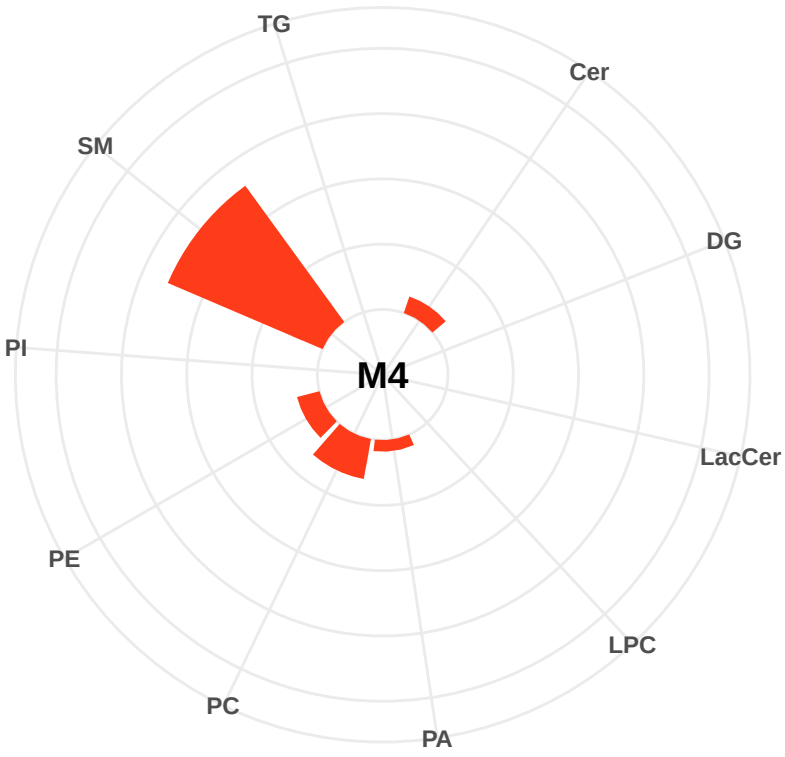


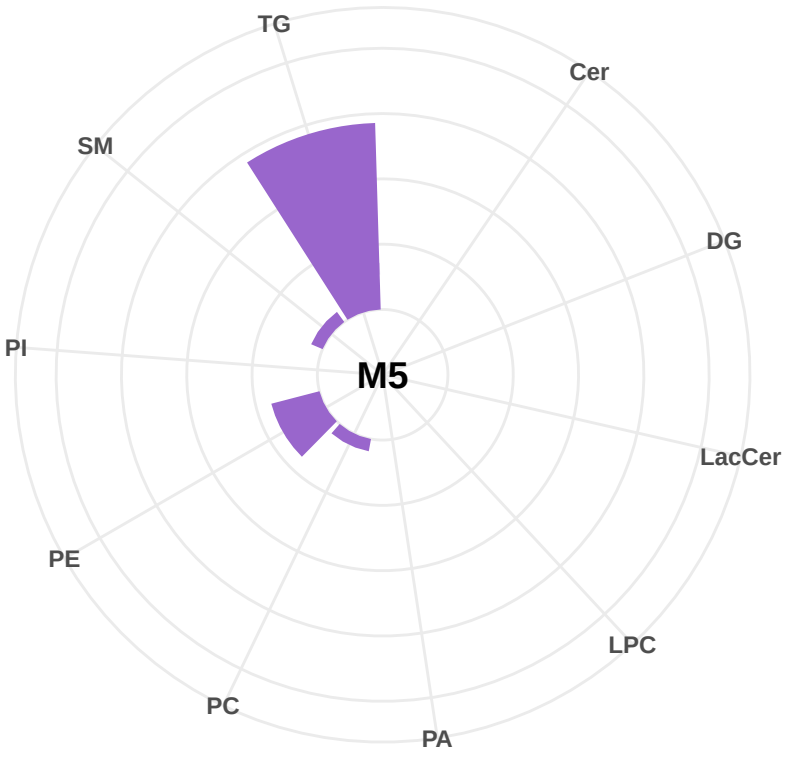

```
## Warning: Expected 2 pieces. Additional pieces discarded in 10 rows [13, 14, 15, 16, 17,
## 18, 19, 20, 21, 31].
```

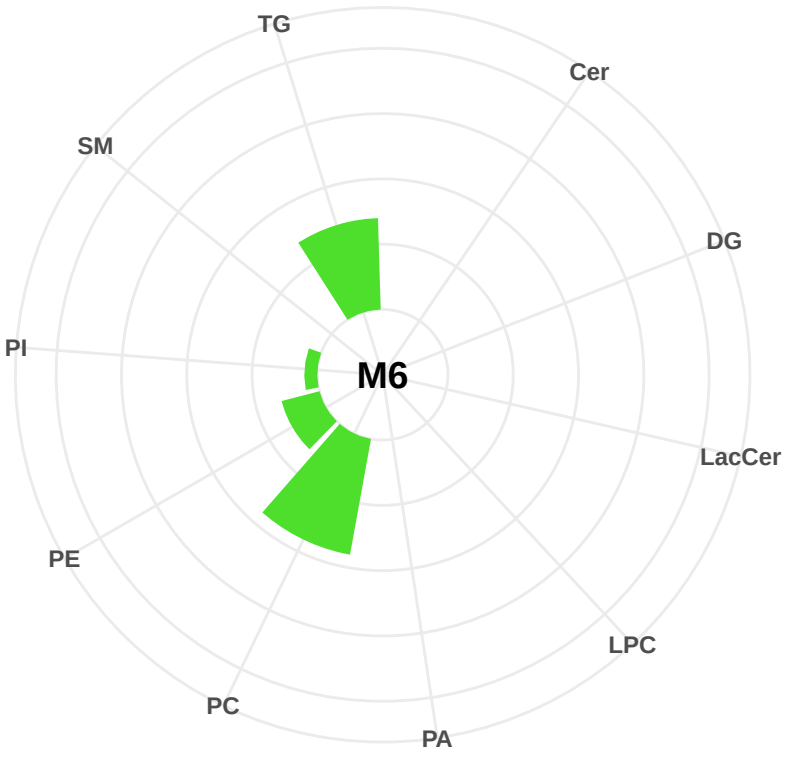


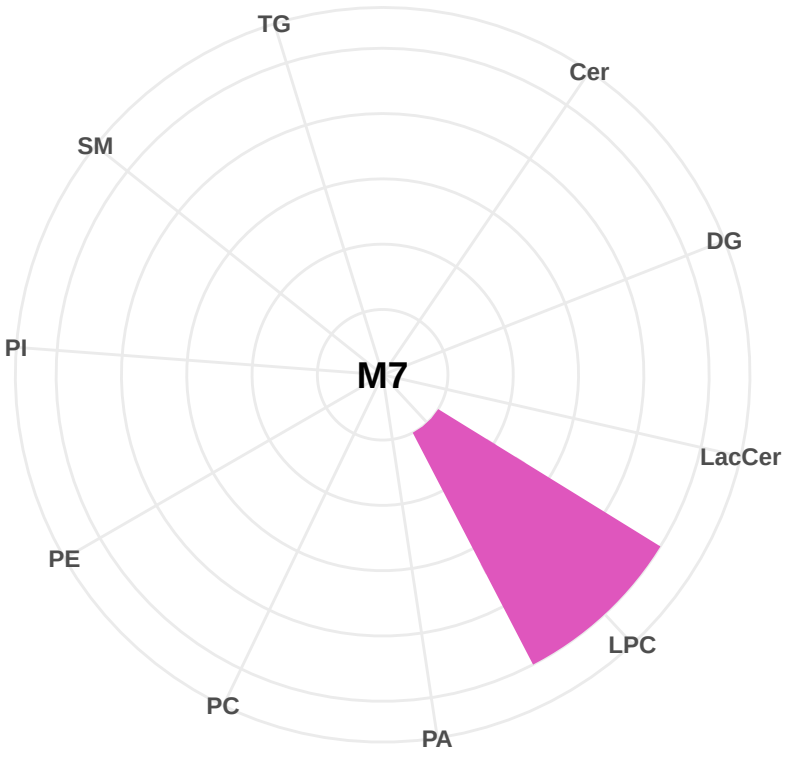

```
## Warning: Expected 2 pieces. Additional pieces discarded in 9 rows [5, 6, 7, 8, 9, 10,
## 43, 44, 45].
```


TG
Cer
SM
DG
PI
M3
LacCer
PE
LPC
PC
PA

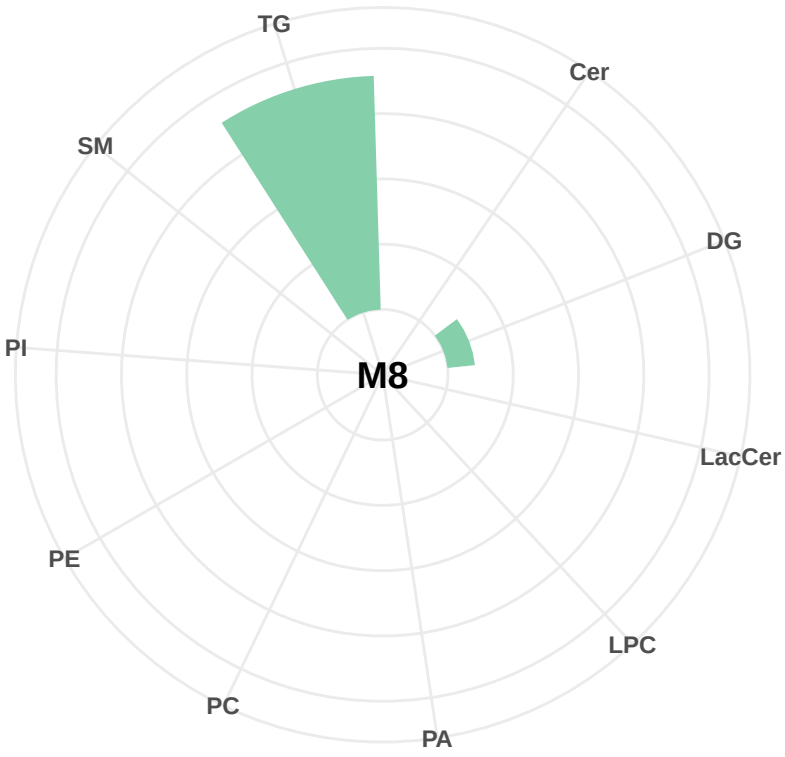

TG
Cer
SM
DG
PI
M4
LacCer
PE
LPC
PC
PA

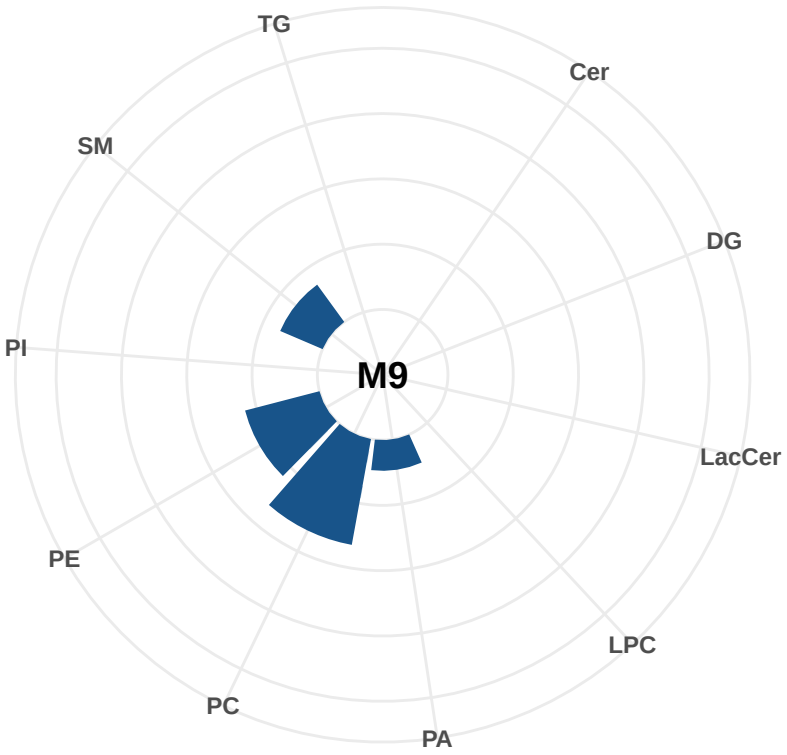

TG
Cer
SM
DG
PI
M5
LacCer
PE
LPC
PC
PA


TG
Cer
SM
DG
PI
M6
LacCer
PE
LPC
PC
PA

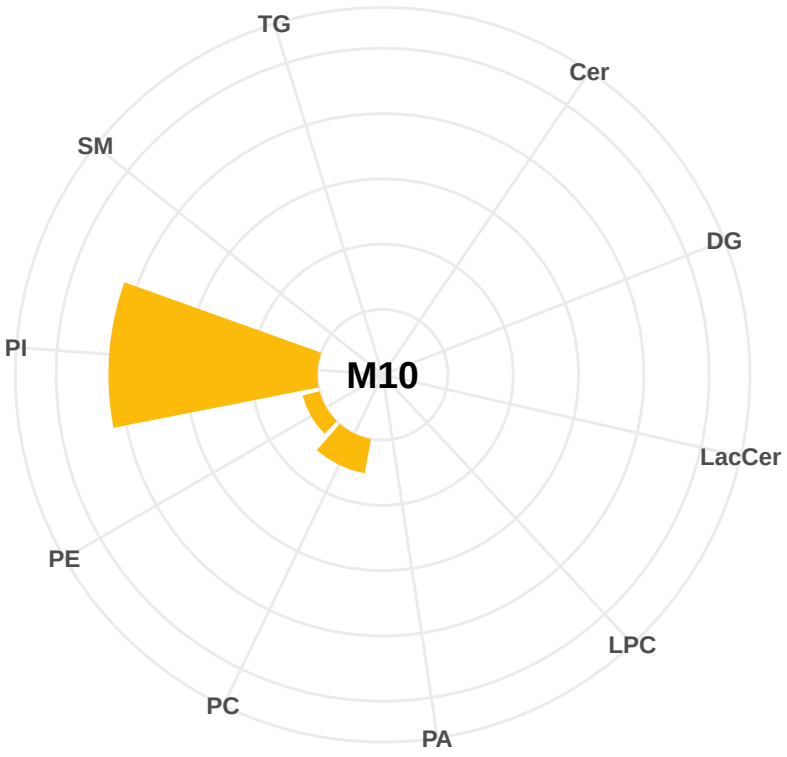


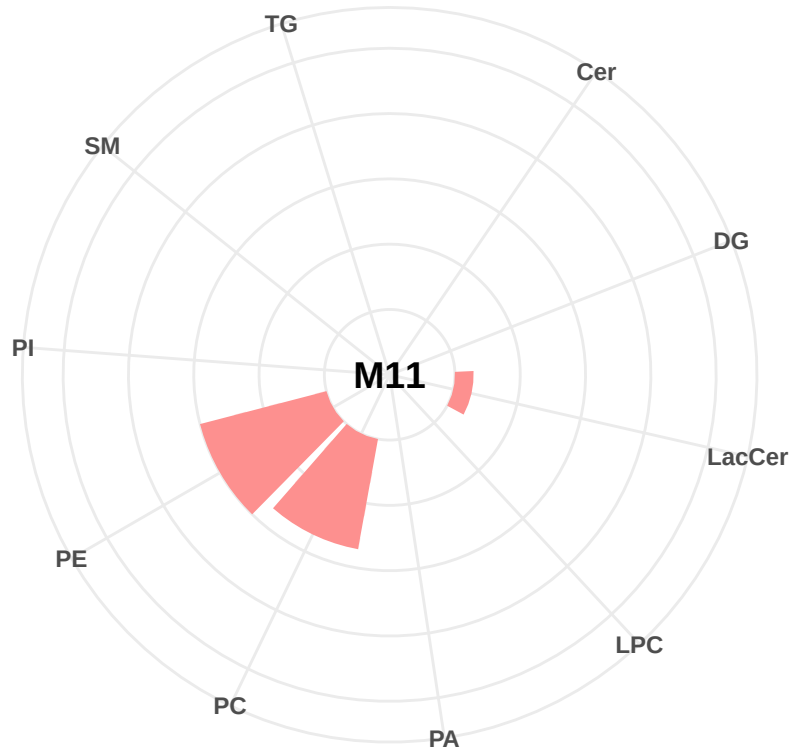

```
## Warning: Expected 2 pieces. Additional pieces discarded in 3 rows [7, 16, 17].
```


TG
Cer
SM
DG
PI
M8
LacCer
PE
LPC
PC
PA




```
## Warning: Expected 2 pieces. Additional pieces discarded in 8 rows [4, 5, 6, 7, 9, 10,
## 11, 12].
```


TG
Cer
SM
DG
PI
M10
LacCer
PE
LPC
PC
PA




```
#Clean
rm(tmp_lipids_excluded, tmp_lipids_included, tmp_plot, i, module_color, module_composistion, module_x)
```

```
#Calculate module eigengenes for each observation
module_eigengenes <- data.frame(moduleEigengenes(data_lipid, colors = lipid_merged_modules$Name)$eigeng

#Fix column names
colnames(module_eigengenes) <- gsub("ME", "", colnames(module_eigengenes))

#Fix module order
module_eigengenes <- module_eigengenes[,order(as.numeric(gsub("M", "", colnames(module_eigengenes))))]

#Set merging ID from rownames
module_eigengenes$ID <- rownames(module_eigengenes)

#Merge with data
data_regression <- data %>%
    left_join(., module_eigengenes)
```

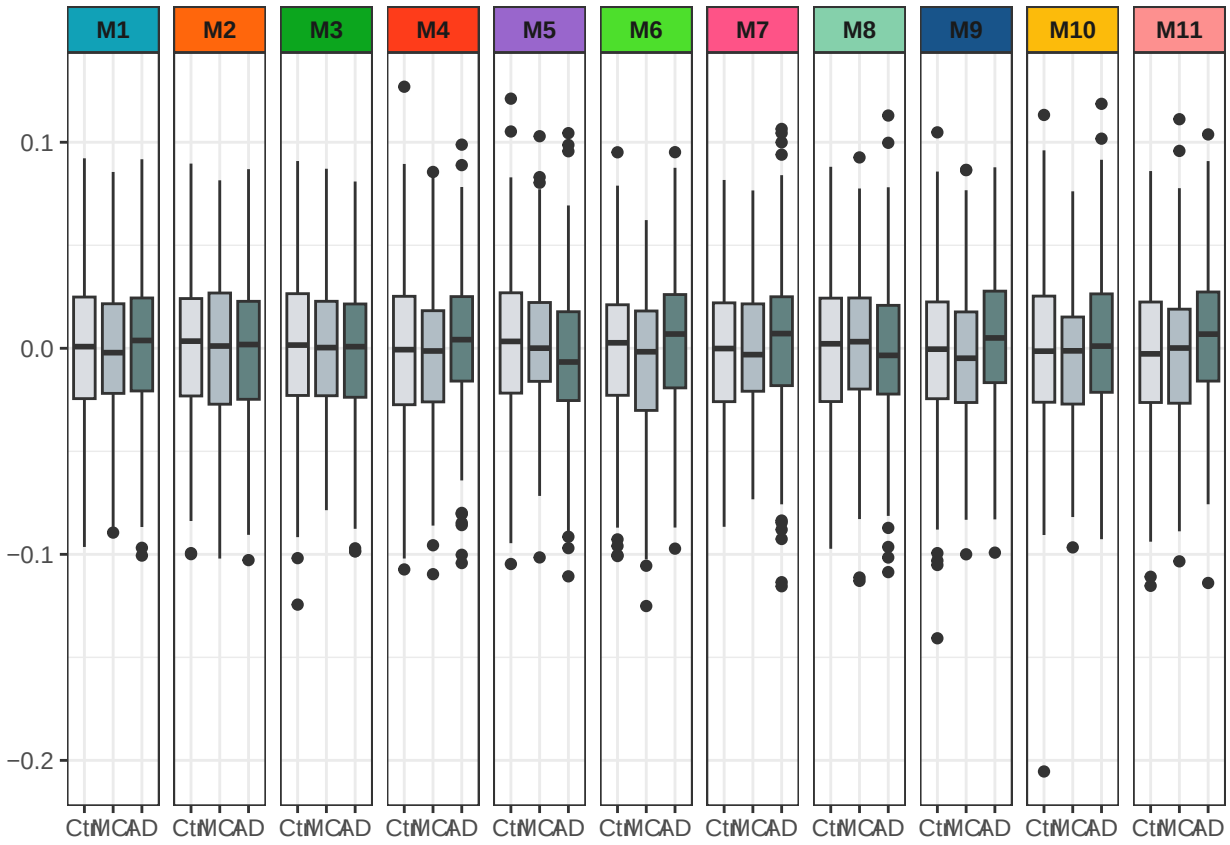
```
## Joining with `by = join_by(ID)`
```

```
#Clean
rm(module_eigengenes)
```

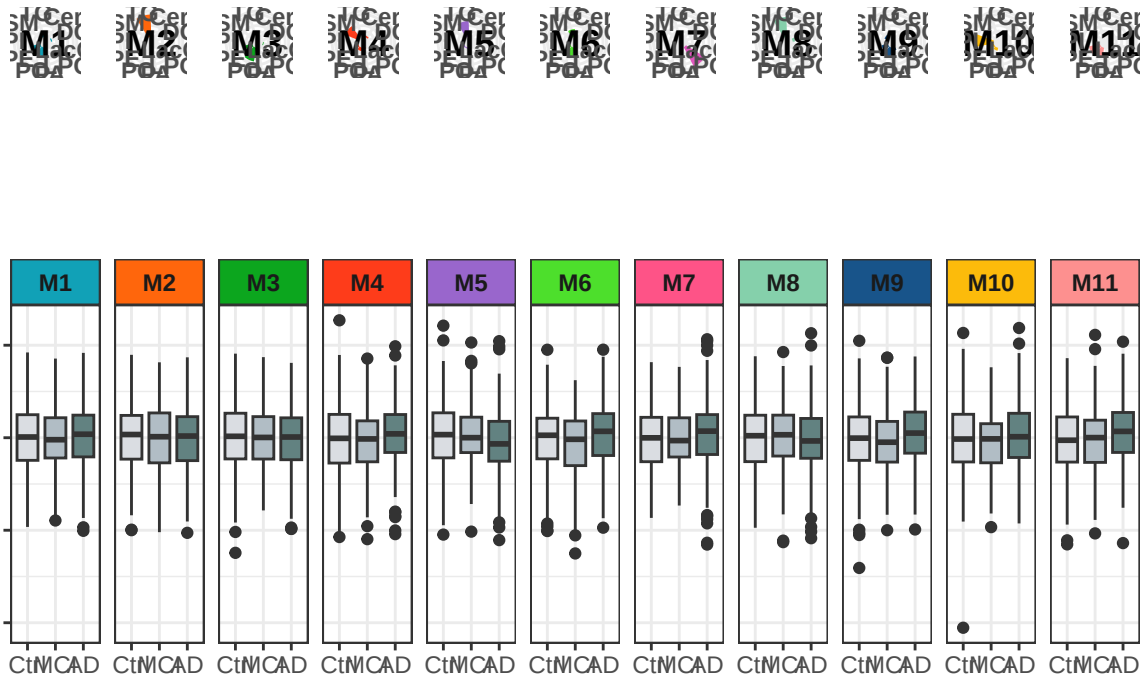
```
library(ggh4x)
```

```
## Warning: pakke 'ggh4x' blev bygget under R version 4.3.1
```

```
#Set facet strip color to the module colors
tmp_strip <- strip_themed(background_x = elem_list_rect(fill = color_palette[as.numeric(gsub("M", "", m

#Boxplot of module eigengenes distribution by AD status
tmp_boxplot <- data_regression %>%
    select(-Marital_Status, -MMSE_Total) %>%
    pivot_longer(starts_with("M"),
                 names_to = "Module",
                 values_to = "Eigengene") %>%
    mutate(Module = factor(Module, levels = module_order)) %>%
    mutate(Status = factor(Status, levels = c("CTL", "MCI", "ADC"),
                           labels = c("Ctrl", "MCI", "AD"))) %>%
    ggplot(aes(x = Status, y = Eigengene, fill = Status))+
    geom_boxplot() +
    scale_fill_manual(values = c("#dadde2",
                                 "#b1bdc5",
                                 "#628281"))+
    facet_wrap2(.~Module, ncol = length(module_order),
                strip = tmp_strip)+
    theme_bw()+
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          legend.position = "none",
          strip.text = element_text(face = "bold"))


# #Save figure
# pdf(here("figures/Boxplot_eigengenes_res_v1.2.pdf"),
#     width = 8, height = 4)

tmp_boxplot
```




```
# dev.off()

#Combined boxplot and circle barplots
library(patchwork)

# #Save figure
# pdf(here("figures/Sup1_Eigengene_boxplot_circlebar_res_v1.1.pdf"),
#     width = 12, height = 4)

(circle_barplots[["M1"]] |
    circle_barplots[["M2"]] |
    circle_barplots[["M3"]] |
    circle_barplots[["M4"]] |
    circle_barplots[["M5"]] |
    circle_barplots[["M6"]] |
    circle_barplots[["M7"]] |
    circle_barplots[["M8"]] |
    circle_barplots[["M9"]] |
    circle_barplots[["M10"]] |
    circle_barplots[["M11"]])/
    tmp_boxplot
```




```
# dev.off()

rm(tmp_strip, tmp_boxplot)
```

```
library(tableone)

tmp_vars <- append(c("Age", "Sex", "APOE4", "Disease_Duration"), module_order)

table_eigen <- data_regression %>%
    mutate(e4_c = factor(e4_c, levels = c(0,1,2),
                         labels = c("None", "Hertrozygote", "Homozygote"))) %>%
    rename("APOE4" = e4_c) %>%
    CreateTableOne(vars = tmp_vars,
                   strata = "Site",
                   addOverall = TRUE)
table_eigen
```

```
##                              Stratified by Site
##                               Overall       Kuopio        Lodz
##   n                             827           126           78
##   Age (mean (SD))             76.24 (6.76)  74.96 (6.69)  73.90 (6.42)
##   Sex = Male (%)                343 (41.5)     59 (46.8)     21 (26.9)
##   APOE4 (%)
##      None                       450 (60.4)     59 (46.8)     43 (59.7)
##      Hertrozygote               242 (32.5)     50 (39.7)     25 (34.7)
```

```
##      Homozygote                    53 ( 7.1)     17 (13.5)      4 ( 5.6)
##   Disease_Duration (mean (SD))  3.69 (3.50)   4.15 (3.41)   4.37 (2.86)
##   M1 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.04)
##   M2 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.04)
##   M3 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.03)
##   M4 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.04)
##   M5 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.04)
##   M6 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.03)
##   M7 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.03)
##   M8 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.03)
##   M9 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.03)
##   M10 (mean (SD))               0.00 (0.03)   0.00 (0.03)   0.00 (0.03)
##   M11 (mean (SD))               0.00 (0.03)   0.00 (0.04)   0.00 (0.04)
##                               Stratified by Site
##                                London        London_DCR    Perugia
##   n                              103           272           130
##   Age (mean (SD))              79.80 (5.94)  77.96 (6.95)  75.49 (5.70)
##   Sex = Male (%)                 40 (38.8)    118 (43.4)     53 (40.8)
##   APOE4 (%)
##      None                        68 (66.7)    127 (62.0)     78 (61.4)
##      Hertrozygote                27 (26.5)     65 (31.7)     44 (34.6)
##      Homozygote                   7 ( 6.9)     13 ( 6.3)      5 ( 3.9)
##   Disease_Duration (mean (SD))  3.71 (3.65)   3.37 (3.53)   3.24 (2.82)
##   M1 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.04)
##   M2 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.04)
##   M3 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.03)
##   M4 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.03)
##   M5 (mean (SD))                0.00 (0.03)   0.00 (0.03)   0.00 (0.04)
##   M6 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.03)
##   M7 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.03)
##   M8 (mean (SD))                0.00 (0.03)   0.00 (0.04)   0.00 (0.04)
##   M9 (mean (SD))                0.00 (0.04)   0.00 (0.04)   0.00 (0.03)
##   M10 (mean (SD))               0.00 (0.03)   0.00 (0.04)   0.00 (0.04)
##   M11 (mean (SD))               0.00 (0.04)   0.00 (0.04)   0.00 (0.03)
##                               Stratified by Site
##                                Thessaloniki  Toulouse      p      test
##   n                               93            25
##   Age (mean (SD))              73.54 (4.69)  70.40 (8.43)  <0.001
##   Sex = Male (%)                 42 (45.2)     10 (40.0)   0.144
##   APOE4 (%)                                                0.048
##      None                        56 (63.6)     19 (76.0)
##      Hertrozygote                27 (30.7)      4 (16.0)
##      Homozygote                   5 ( 5.7)      2 ( 8.0)
##   Disease_Duration (mean (SD))  3.10 (3.42)   5.53 (5.82)   0.066
##   M1 (mean (SD))                0.00 (0.04)   0.00 (0.04)   1.000
##   M2 (mean (SD))                0.00 (0.04)   0.00 (0.04)   1.000
##   M3 (mean (SD))                0.00 (0.04)   0.00 (0.04)   1.000
##   M4 (mean (SD))                0.00 (0.04)   0.00 (0.03)   1.000
##   M5 (mean (SD))                0.00 (0.04)   0.00 (0.04)   1.000
##   M6 (mean (SD))                0.00 (0.04)   0.00 (0.03)   1.000
##   M7 (mean (SD))                0.00 (0.04)   0.00 (0.03)   1.000
##   M8 (mean (SD))                0.00 (0.03)   0.00 (0.03)   1.000
##   M9 (mean (SD))                0.00 (0.03)   0.00 (0.03)   1.000
##   M10 (mean (SD))               0.00 (0.03)   0.00 (0.02)   1.000
```

```
##   M11 (mean (SD))               0.00 (0.04)   0.00 (0.03)   1.000
```

```
# write.csv(print(table_eigen, printToggle = FALSE), here("data/table_eigen_v1.1.csv"))

rm(tmp_vars, table_eigen)
```

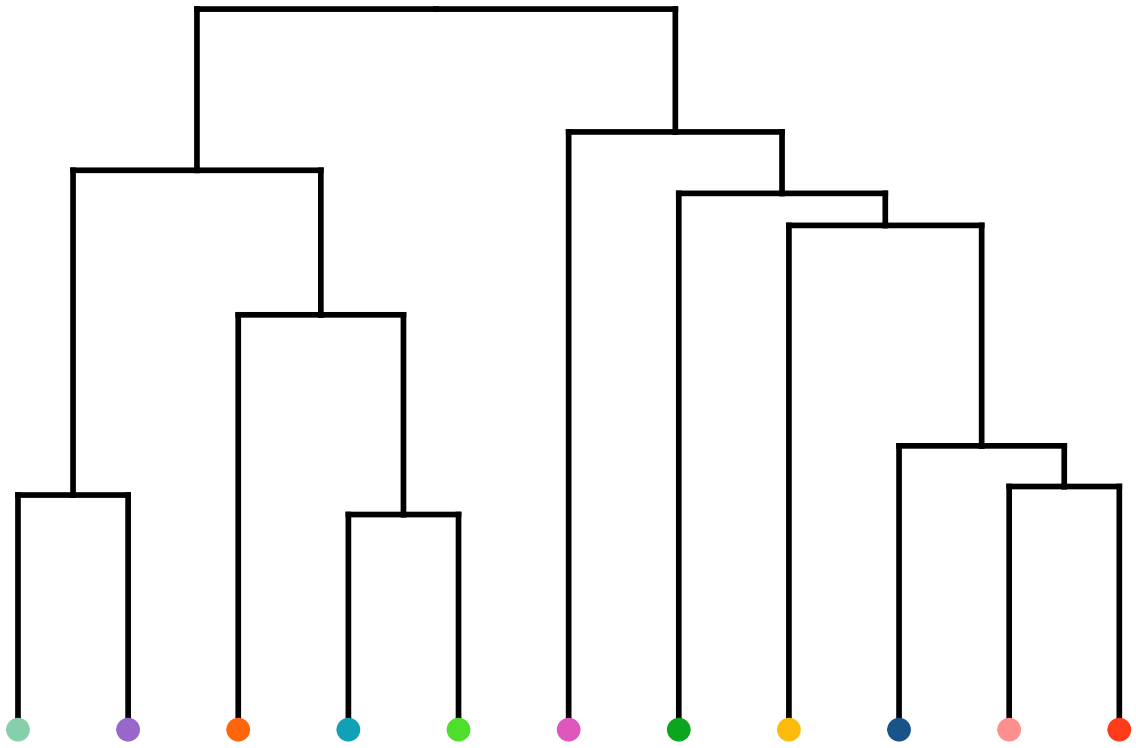
```
#Subset only AD and control samples
data_regression_AD <- data_regression %>%
    mutate(AD_CTL = if_else(Status == "ADC", 1, NA)) %>%
    mutate(AD_CTL = if_else(Status == "CTL", 0, AD_CTL)) %>%
    relocate(AD_CTL, .after = Status) %>%
    filter(!is.na(AD_CTL)) %>%
    data.frame()

#Subset only MCI and control samples
data_regression_MCI <- data_regression %>%
    mutate(MCI_CTL = if_else(Status == "MCI", 1, NA)) %>%
    mutate(MCI_CTL = if_else(Status == "CTL", 0, MCI_CTL)) %>%
    relocate(MCI_CTL, .after = Status) %>%
    filter(!is.na(MCI_CTL)) %>%
    data.frame()

#Subset by Sex
data_regression_sex <- data_regression %>%
    mutate(Sex_binary = if_else(Sex == "Female", 1, NA)) %>%
    mutate(Sex_binary = if_else(Sex == "Male", 0, Sex_binary)) %>%
    relocate(Sex_binary, .after = Status) %>%
    filter(!is.na(Sex_binary)) %>%
    data.frame()

#Subset by APOE4
data_regression_APOE4 <- data_regression %>%
    filter(!is.na(e4_c)) %>%
    data.frame()

#Subset only AD and control samples in Women
data_regression_women <- data_regression %>%
    filter(!is.na(Sex)) %>%
    filter(Sex == "Female") %>%
    mutate(AD_CTL = if_else(Status == "ADC", 1, NA)) %>%
    mutate(AD_CTL = if_else(Status == "CTL", 0, AD_CTL)) %>%
    relocate(AD_CTL, .after = Status) %>%
    filter(!is.na(AD_CTL)) %>%
    data.frame()

#Subset only AD and control samples in Men
data_regression_men <- data_regression %>%
    filter(!is.na(Sex)) %>%
    filter(Sex == "Male") %>%
    mutate(AD_CTL = if_else(Status == "ADC", 1, NA)) %>%
    mutate(AD_CTL = if_else(Status == "CTL", 0, AD_CTL)) %>%
    relocate(AD_CTL, .after = Status) %>%
    filter(!is.na(AD_CTL)) %>%
    data.frame()
```

```r
#Create a list containing all info needed for each model
regression_models <- list(
    Model_1 = c(name = "ADraw",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = "",
                data = "data_regression_AD",
                family = "binomial"),
    Model_2 = c(name = "ADadj",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = " + Sex + e4_c",
                data = "data_regression_AD",
                family = "binomial"),
    Model_3 = c(name = "MCIraw",
                formula_prefix = "MCI_CTL ~ ",
                formula_sufix = "",
                data = "data_regression_MCI",
                family = "binomial"),
    Model_4 = c(name = "MCIadj",
                formula_prefix = "MCI_CTL ~ ",
                formula_sufix = " + Sex + e4_c",
                data = "data_regression_MCI",
                family = "binomial"),
    Model_5 = c(name = "Sexraw",
                formula_prefix = "Sex_binary ~ ",
                formula_sufix = "",
                data = "data_regression_sex",
                family = "binomial"),
    Model_6 = c(name = "Sexadj",
                formula_prefix = "Sex_binary ~ ",
                formula_sufix = " + Status + e4_c",
                data = "data_regression_sex",
                family = "binomial"),
    Model_7 = c(name = "APOE4raw",
                formula_prefix = "e4_c ~ ",
                formula_sufix = "",
                data = "data_regression_APOE4",
                family = "gaussian"),
    Model_8 = c(name = "APOE4adj",
                formula_prefix = "e4_c ~ ",
                formula_sufix = " + Status + Sex",
                data = "data_regression_APOE4",
                family = "gaussian"),
    Model_9 = c(name = "Femaleraw",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = "",
                data = "data_regression_women",
                family = "binomial"),
    Model_10 = c(name = "Femaleadj",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = " + e4_c",
                data = "data_regression_women",
                family = "binomial"),
    Model_11 = c(name = "Menraw",
```

```
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = "",
                data = "data_regression_men",
                family = "binomial"),
    Model_12 = c(name = "Menadj",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = " + e4_c",
                data = "data_regression_men",
                family = "binomial"))

#Data frame to populate
regression_summary <- data.frame("tmp" = c())

#Loop over each regression model
for (j in names(regression_models)){

    #Loop over each lipid
    for (i in module_order){

        #Formula
        tmp_formula <- as.formula( paste0(regression_models[[j]][["formula_prefix"]], i,
       regression_models[[j]][["formula_sufix"]]))

        #Data
        tmp_data <- eval(as.symbol(regression_models[[j]][["data"]]))

        #Model type
        tmp_family <- regression_models[[j]][["family"]]

        #Fit logistic regression model
        model_fit <- glm(formula = tmp_formula,
                         data = tmp_data,
                         family = tmp_family)

        #Model suffix
        tmp_model_suffix <- regression_models[[j]][["name"]]

        #Extract summary statistics
regression_summary[i, paste0("Estimate_", tmp_model_suffix)] <- summary(model_fit)$coefficients[i, "Esti

regression_summary[i, paste0("StdError_", tmp_model_suffix)] <- summary(model_fit)$coefficients[i, "Std

regression_summary[i, paste0("Pval_", tmp_model_suffix)] <- summary(model_fit)$coefficients[i, which(gr

regression_summary[i, paste0("Model_", tmp_model_suffix)] <- format(tmp_formula)

    }

    #Adjust for multiple testing
regression_summary[, paste0("PvalFDR_", tmp_model_suffix)] <- p.adjust(regression_summary[, paste0("Pval
         method = "fdr")

}
```

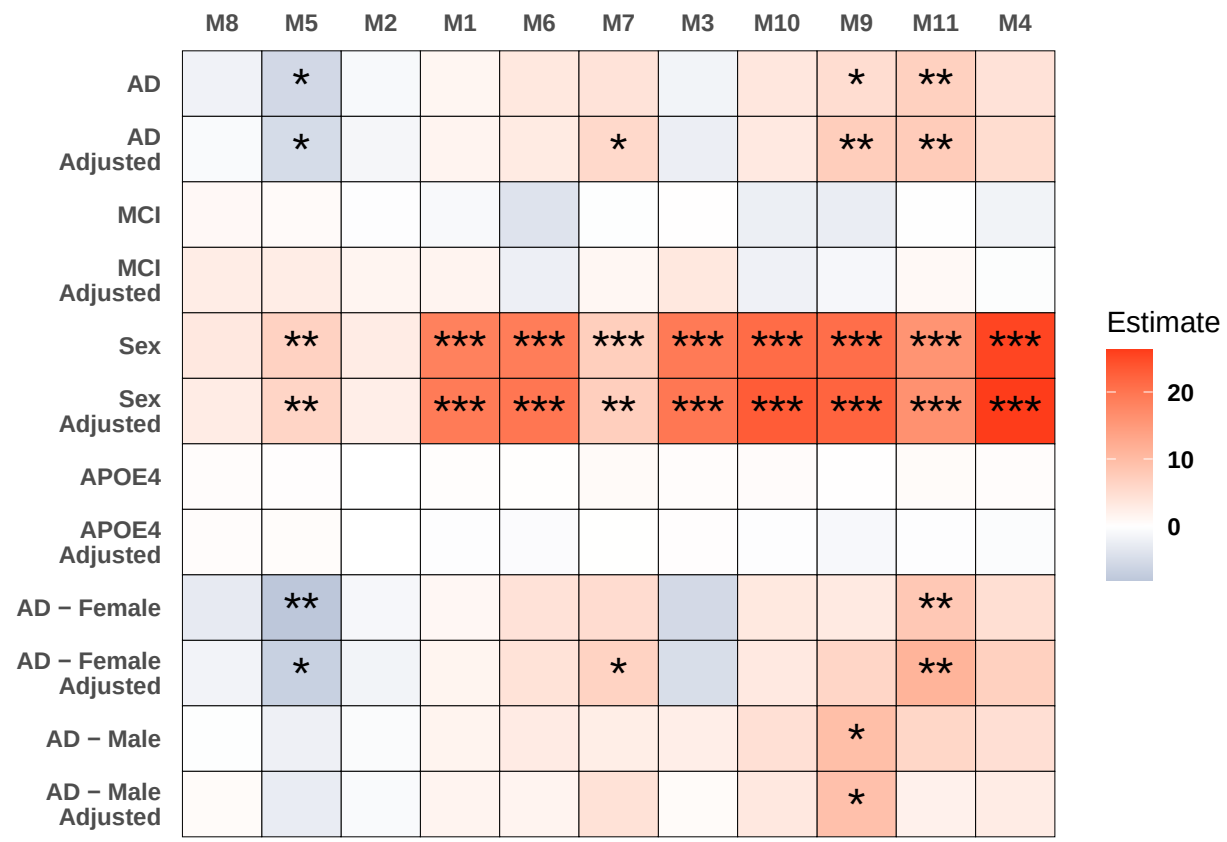
```r
#clean
rm(i, j, tmp_formula, tmp_model_suffix, model_fit, tmp_data, data_regression, data_regression_sex, data_
```

```r
# 1) Dendrogram
fig1_dendro
```



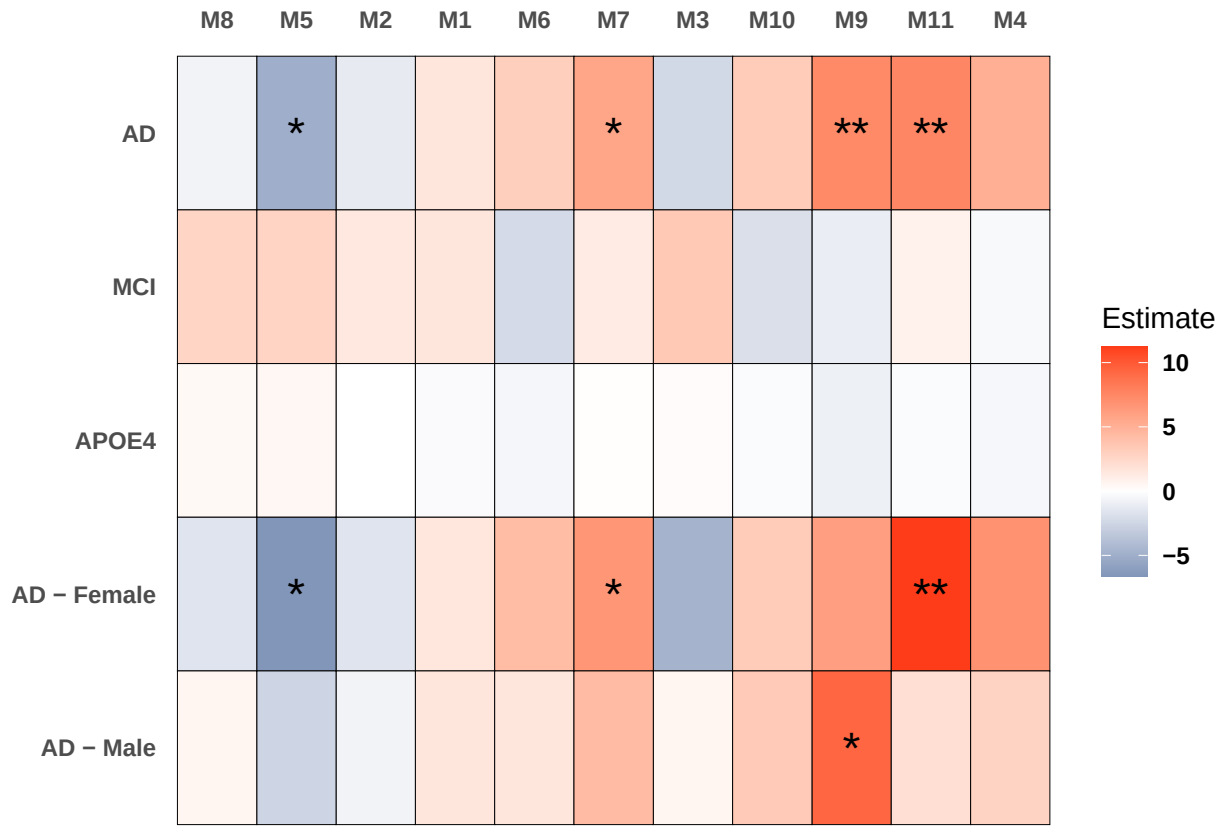
```r
#2) heatmap
regression_summary$Modules <- rownames(regression_summary)

#Empty tibble to populate
tmp_tibble <- c()
tmp_order <- c()

#Stack summary results for heatmap
for (j in names(regression_models)){
    tmp_tibble <- tmp_tibble %>%
        rbind(.,
        tibble(regression_summary) %>%
        select(contains(regression_models[[j]][["name"]]), Modules) %>%
        rename_with(~gsub(paste0("_", regression_models[[j]][["name"]]), "", .)) %>%
        mutate(Outcome = regression_models[[j]][["name"]]))
    tmp_order <- append(tmp_order, regression_models[[j]][["name"]])
}
```

```r
#reorganize
tmp_tibble <- tmp_tibble %>%
    relocate(Model, .after = Outcome)

tmp_order <- gsub("raw", "", tmp_order)
tmp_order <- gsub("adj", "\nAdjusted", tmp_order)
tmp_order <- gsub("Men", "AD - Male", tmp_order)
tmp_order <- gsub("Female", "AD - Female", tmp_order)

#Fix module_order_dendtrgram (Swap M4 and M11)
module_order_dendrogram <- replace(module_order_dendrogram,
        c(which(module_order_dendrogram %in% "M4"),
          which(module_order_dendrogram %in% "M11")),
        module_order_dendrogram[c(which(module_order_dendrogram %in% "M11"),
                                  which(module_order_dendrogram %in% "M4"))])

#Fix module_order_dendtrgram (Swap M5 and M8)
module_order_dendrogram <- replace(module_order_dendrogram,
        c(which(module_order_dendrogram %in% "M5"),
          which(module_order_dendrogram %in% "M8")),
        module_order_dendrogram[c(which(module_order_dendrogram %in% "M8"),
                                  which(module_order_dendrogram %in% "M5"))])

#plot heatmap full
tmp_tibble  %>%
    mutate(Pstar = if_else(Pval < 0.05, "*", "")) %>%
    mutate(Pstar = if_else(Pval < 0.01, "**", Pstar)) %>%
    mutate(Pstar = if_else(Pval < 0.001, "***", Pstar)) %>%
    mutate(Modules = factor(Modules, levels = module_order_dendrogram)) %>%
    mutate(Outcome = gsub("Men", "AD - Male", Outcome)) %>%
    mutate(Outcome = gsub("Female", "AD - Female", Outcome)) %>%
    mutate(Outcome = gsub("raw", "", Outcome)) %>%
    mutate(Outcome = gsub("adj", "\nAdjusted", Outcome)) %>%
    mutate(Outcome = factor(Outcome, levels = rev(tmp_order))) %>%
    ggplot(aes(x = Modules, y = Outcome, fill = Estimate))+
    geom_tile(color = "black") +
    scale_fill_gradient2(#limits = c(-9,9),
                         low = "#18548A",
                         mid = "#FFFFFF",
                         high = "#FE3C1A")+
    geom_text(aes(label = Pstar),
                  color = "black", size = 6)+
    theme_minimal()+
    scale_x_discrete(position = "top") +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          panel.grid.major = element_blank(),
          axis.text = element_text(face = "bold"),
          legend.text = element_text(face = "bold"))
```




```
#plot heatmap adjusted only
fig1_heatmap <- tmp_tibble  %>%
    mutate(Pstar = if_else(Pval < 0.05, "*", "")) %>%
    mutate(Pstar = if_else(Pval < 0.01, "**", Pstar)) %>%
    mutate(Pstar = if_else(Pval < 0.001, "***", Pstar)) %>%
    mutate(Modules = factor(Modules, levels = module_order_dendrogram)) %>%
    mutate(Outcome = gsub("Men", "AD - Male", Outcome)) %>%
    mutate(Outcome = gsub("Female", "AD - Female", Outcome)) %>%
    filter(!grepl("raw", Outcome)) %>%
    filter(!grepl("Sex", Outcome)) %>%
    mutate(Outcome = gsub("adj", "", Outcome)) %>%
    mutate(Outcome = factor(Outcome, levels = rev(tmp_order))) %>%
    ggplot(aes(x = Modules, y = Outcome, fill = Estimate))+
    geom_tile(color = "black") +
    scale_fill_gradient2(#limits = c(-9,9),
                         low = "#18548A",
                         mid = "#FFFFFF",
                         high = "#FE3C1A")+
    geom_text(aes(label = Pstar),
                  color = "black", size = 6)+
    theme_minimal()+
    scale_x_discrete(position = "top") +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          panel.grid.major = element_blank(),
          axis.text = element_text(face = "bold"),
```

```
          legend.text = element_text(face = "bold"))


#Save figure
#pdf(here("figures/Heatmap_regression_v2.4.pdf"), width = 9, height = 3)

fig1_heatmap
```




```
#dev.off()

# #Save Table for supplementary
# vroom_write(tmp_tibble, here("data/sup_table_module_eigengene_regression_redisual_v1.2.csv"))

regression_summary_fig1 <- regression_summary

rm(regression_models, tmp_tibble, j, tmp_order, regression_summary)
```

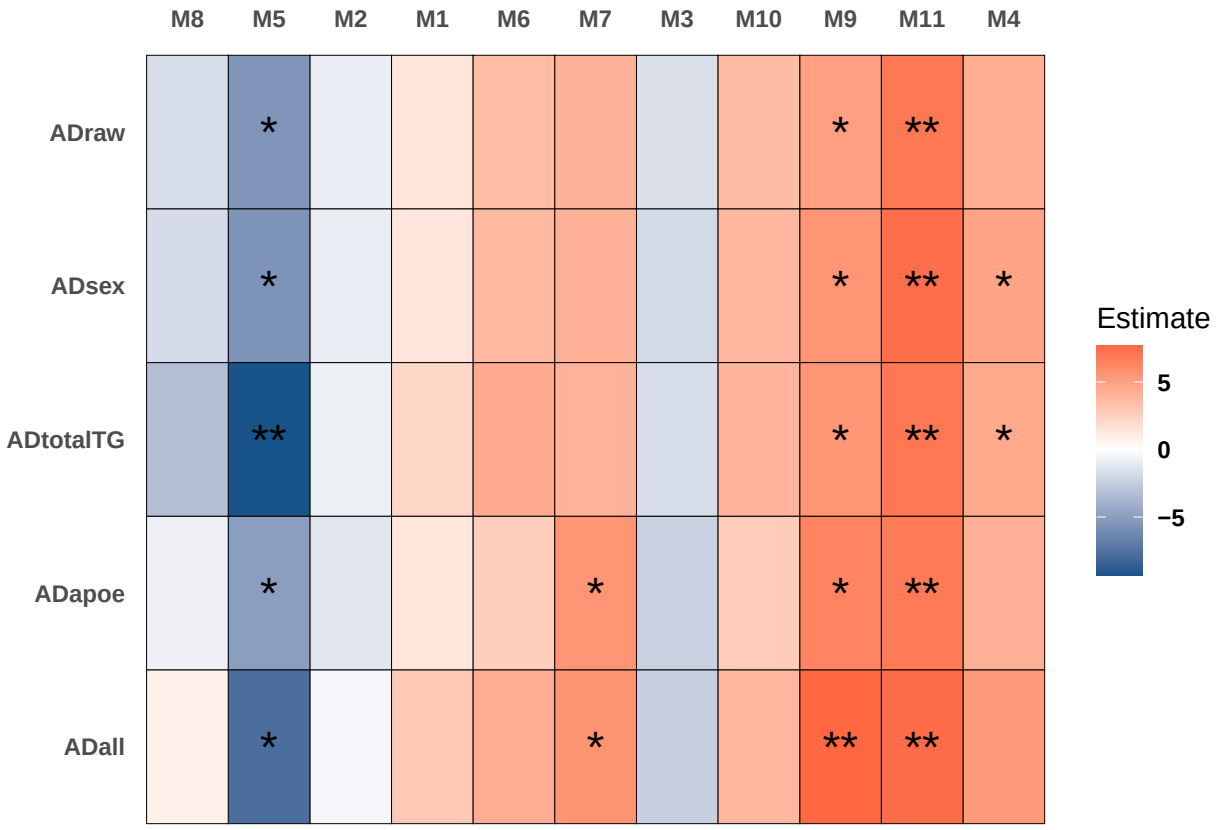
```
#Create a list containing all info needed for each model
regression_models <- list(
    Model_1 = c(name = "ADraw",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = "",
                data = "data_regression_AD"),
    Model_2 = c(name = "ADsex",
                formula_prefix = "AD_CTL ~ ",
```

```
                formula_sufix = " + Sex",
                data = "data_regression_AD"),
    Model_3 = c(name = "ADtotalTG",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = " + Total_TG",
                data = "data_regression_AD"),
    Model_4 = c(name = "ADapoe",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = " + e4_c",
                data = "data_regression_AD"),
    Model_5 = c(name = "ADall",
                formula_prefix = "AD_CTL ~ ",
                formula_sufix = " + Sex + e4_c + Total_TG",
                data = "data_regression_AD"))


#Data frame to populate
regression_summary <- data.frame("tmp" = c())

#Loop over each regression model
for (j in names(regression_models)){

    #Loop over each module
    for (i in module_order){

        #Formula
        tmp_formula <- as.formula( paste0(regression_models[[j]][["formula_prefix"]], i,
       regression_models[[j]][["formula_sufix"]]))

        #Data
        tmp_data <- eval(as.symbol(regression_models[[j]][["data"]]))

        #Fit logistic regression model
        model_fit <- glm(formula = tmp_formula,
                         data = tmp_data,
                         family = "binomial")

        #Model suffix
        tmp_model_suffix <- regression_models[[j]][["name"]]

        #Extract summary statistics
regression_summary[i, paste0("Estimate_", tmp_model_suffix)] <- summary(model_fit)$coefficients[i, "Esti

regression_summary[i, paste0("StdError_", tmp_model_suffix)] <- summary(model_fit)$coefficients[i, "Std

regression_summary[i, paste0("Pval_", tmp_model_suffix)] <- summary(model_fit)$coefficients[i, "Pr(>|z|)

regression_summary[i, paste0("Model_", tmp_model_suffix)] <- format(tmp_formula)

    }

    #Adjust for multiple testing
regression_summary[, paste0("PvalFDR_", tmp_model_suffix)] <- p.adjust(regression_summary[, paste0("Pval
```

```
         method = "fdr")

}


#Clean
rm(data_regression_AD, data_regression_MCI, model_fit, tmp_formula, tmp_model_suffix, tmp_data, i, j, tm
```

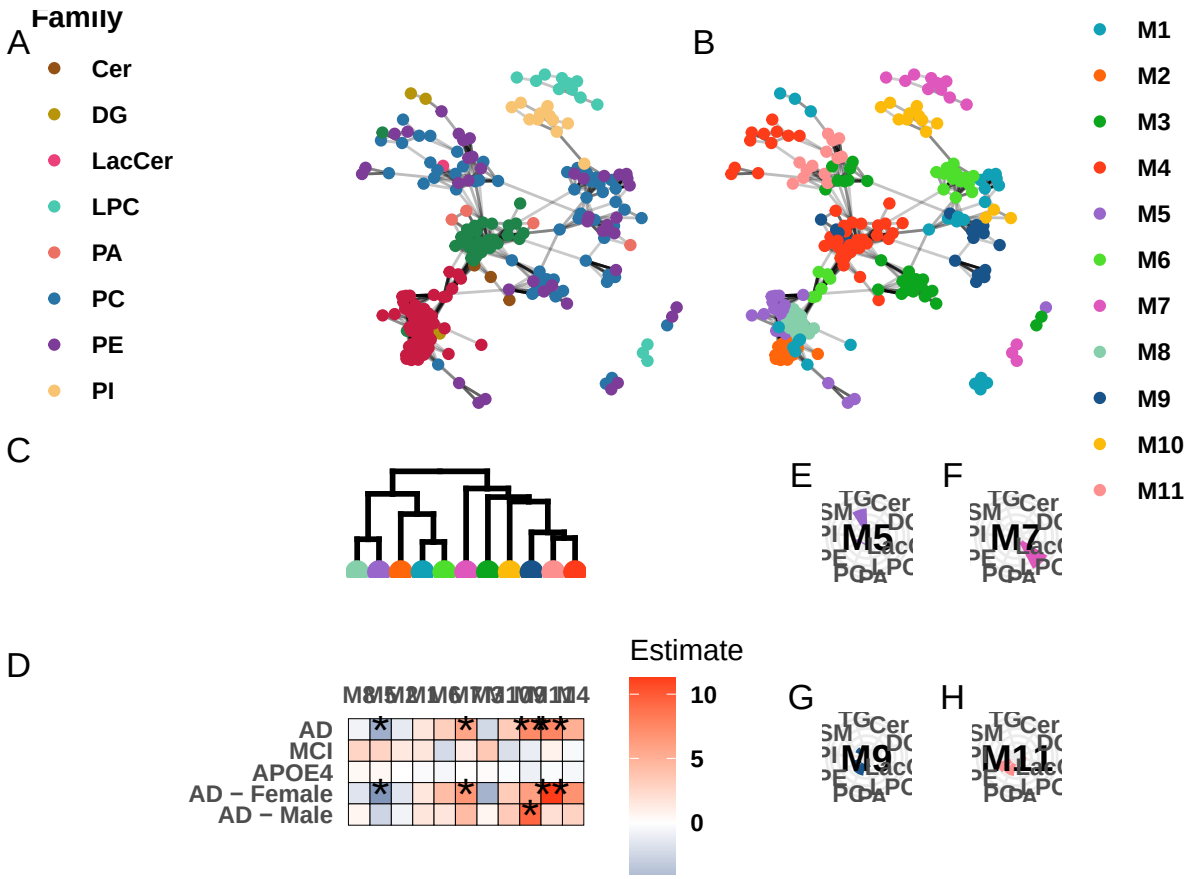
```
regression_summary$Modules <- rownames(regression_summary)

#Empty tibble to populate
tmp_tibble <- c()
tmp_order <- c()

#Stack summary results for heatmap
for (j in names(regression_models)){
    tmp_tibble <- tmp_tibble %>%
        rbind(.,
        tibble(regression_summary) %>%
        select(contains(regression_models[[j]][["name"]]), Modules) %>%
        rename_with(~gsub(paste0("_", regression_models[[j]][["name"]]), "", .)) %>%
        mutate(Outcome = regression_models[[j]][["name"]]))
    tmp_order <- append(tmp_order, regression_models[[j]][["name"]])
}

#reorganize
tmp_tibble <- tmp_tibble %>%
    relocate(Model, .after = Outcome)

# #Save figure
# pdf(here("figures/Heatmap_module_regression_sensitivity_analysis_res_v1.2.pdf"), width = 5, height = 

#plot heatmap full
tmp_tibble  %>%
    mutate(Pstar = if_else(Pval < 0.05, "*", "")) %>%
    mutate(Pstar = if_else(Pval < 0.01, "**", Pstar)) %>%
    mutate(Pstar = if_else(Pval < 0.001, "***", Pstar)) %>%
    mutate(Modules = factor(Modules, levels = module_order_dendrogram)) %>%
    mutate(Outcome = gsub("Men", "Male", Outcome)) %>%
    mutate(Outcome = factor(Outcome, levels = rev(tmp_order))) %>%
    ggplot(aes(x = Modules, y = Outcome, fill = Estimate))+
    geom_tile(color = "black") +
    scale_fill_gradient2(#limits = c(-9,9),
                         low = "#18548A",
                         mid = "#FFFFFF",
                         high = "#FE3C1A")+
    geom_text(aes(label = Pstar),
                  color = "black", size = 6)+
    theme_minimal()+
    scale_x_discrete(position = "top") +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          panel.grid.major = element_blank(),
```

```
        axis.text = element_text(face = "bold"),
        legend.text = element_text(face = "bold"))
```




```
#dev.off()

# #Save Table for supplementary
# vroom_write(tmp_tibble, here("data/sup_table_module_eigengene_sensitivity_redisual_v1.1.csv"))

rm(regression_models, tmp_tibble, j, regression_summary, tmp_order)
```

```
library(patchwork) #https://patchwork.data-imaginist.com/

## NOTE: Module circle plots are added manually, making it vulnerable to changes

# #Save figure
# pdf(here("figures/Figure_1_WCNA_Residual_v1.7.pdf"),
#     width = 10, height = 7)

#Combine multible plots for figure 1
(fig1_network_lipids | fig1_network_module) /
    ((fig1_dendro / fig1_heatmap) |
         ((circle_barplots[["M5"]] | circle_barplots[["M7"]]) /
          (circle_barplots[["M9"]] | circle_barplots[["M11"]] | circle_barplots[["M12"]])))+
    plot_annotation(tag_levels = "A")
```




```
# dev.off()


rm(circle_barplots, fig1_dendro, fig1_heatmap, fig1_network_lipids, fig1_network_module)
```

```
#Modules of interest
selected_modules <- regression_summary_fig1 %>%
    filter(Pval_ADadj < 0.05) %>%
    select(Pval_ADadj) %>%
    rownames_to_column("Modules") %>%
    pull(Modules)

#Calculate eigengenes (first principal component) for each observation
eigengenes_merged <- moduleEigengenes(data_lipid, colors = lipid_merged_modules$Name)$eigengenes

#Fix names
colnames(eigengenes_merged) <- gsub("ME", "", names(eigengenes_merged))

#Calculate the correlation value between each metabolite and each module
eigengenes_lipid_cor <- as.data.frame(cor(data_lipid, eigengenes_merged, use = "p"))

#Select lipids based on kME threshold r > 0.7

#Empty vector to populate
selected_lipids <- c()
tmp_corresponding_module <- c()
```

```
for (i in selected_modules){

    #filterlipids based on kME threshold r > 0.7
    tmp_selected_lipids <- eigengenes_lipid_cor %>%
        rownames_to_column("Lipids") %>%
        tibble() %>%
        filter(get(i) > 0.7) %>%
        pull(Lipids)

    #filter only lipids within selected module
    tmp_selected_lipids <-
        tmp_selected_lipids[tmp_selected_lipids %in%
        lipid_merged_modules$Lipid[lipid_merged_modules$Name == i]]

    #append selected lipids
    selected_lipids <- append(selected_lipids,
                              tmp_selected_lipids)

    #append corresponding module information
    tmp_corresponding_module <- append(tmp_corresponding_module,
                            rep(i, length(tmp_selected_lipids)))

}

selected_lipids <- data.frame(
    "Module" = tmp_corresponding_module,
    "Lipid" = selected_lipids)

selected_lipids <- selected_lipids %>%
    left_join(., lipid_merged_modules) %>%
    select(-Name) %>%
    data.frame()
```

```
## Joining with `by = join_by(Lipid)`
```

```
rm(regression_summary_fig1, eigengenes_lipid_cor,
   eigengenes_merged, selected_modules, tmp_selected_lipids,
   tmp_corresponding_module, i)

# #Select lipids based on top 90th percentile
# eigengenes_lipid_cor %>%
#     select(selected_modules) %>%
#     filter(M5 > quantile(eigengenes_lipid_cor$M5,
#                          probs = c(0.1, 0.9))[["90%"]] |
#            M8 > quantile(eigengenes_lipid_cor$M8,
#                          probs = c(0.1, 0.9))[["90%"]] |
#            M10 > quantile(eigengenes_lipid_cor$M10,
#                           probs = c(0.1, 0.9))[["90%"]] |
#            M11 > quantile(eigengenes_lipid_cor$M11,
#                           probs = c(0.1, 0.9))[["90%"]]) %>%
#     rownames()

# #Select lipids based on top 10 most correlated lipids
```

```
# #Empty data frame to populate with top 10 correlated lipids
# top_cor_lipids <- data.frame("tmp" = 1:10)
#
# #Loop through each module and select the 10 most correlated lipids
# for (i in module_order){
#     top_cor_lipids[, i] <- eigengenes_lipid_cor %>%
#         arrange(desc(pick(matches(i)))) %>%
#         filter(c(rep(TRUE, 10),
#                  rep(FALSE, nrow(.)-10))) %>%
#         rownames()
# }
#
# #Select only top lipids from modules of interest
# selected_lipids <- top_cor_lipids %>%
#     select(all_of(selected_modules)) %>%
#     pivot_longer(everything(), names_to = "Module", values_to = "Lipid")
#
# rm(regression_summary, selected_modules, eigengenes_merged, eigengenes_lipid_cor, top_cor_lipids, i)
```

```
# #Export lipids for next part of the analysis
# vroom_write(selected_lipids, here("data/ANM_module_selected_lipids_res_v1.1.csv"))
```