# WGCNA

## Function for redering rmd

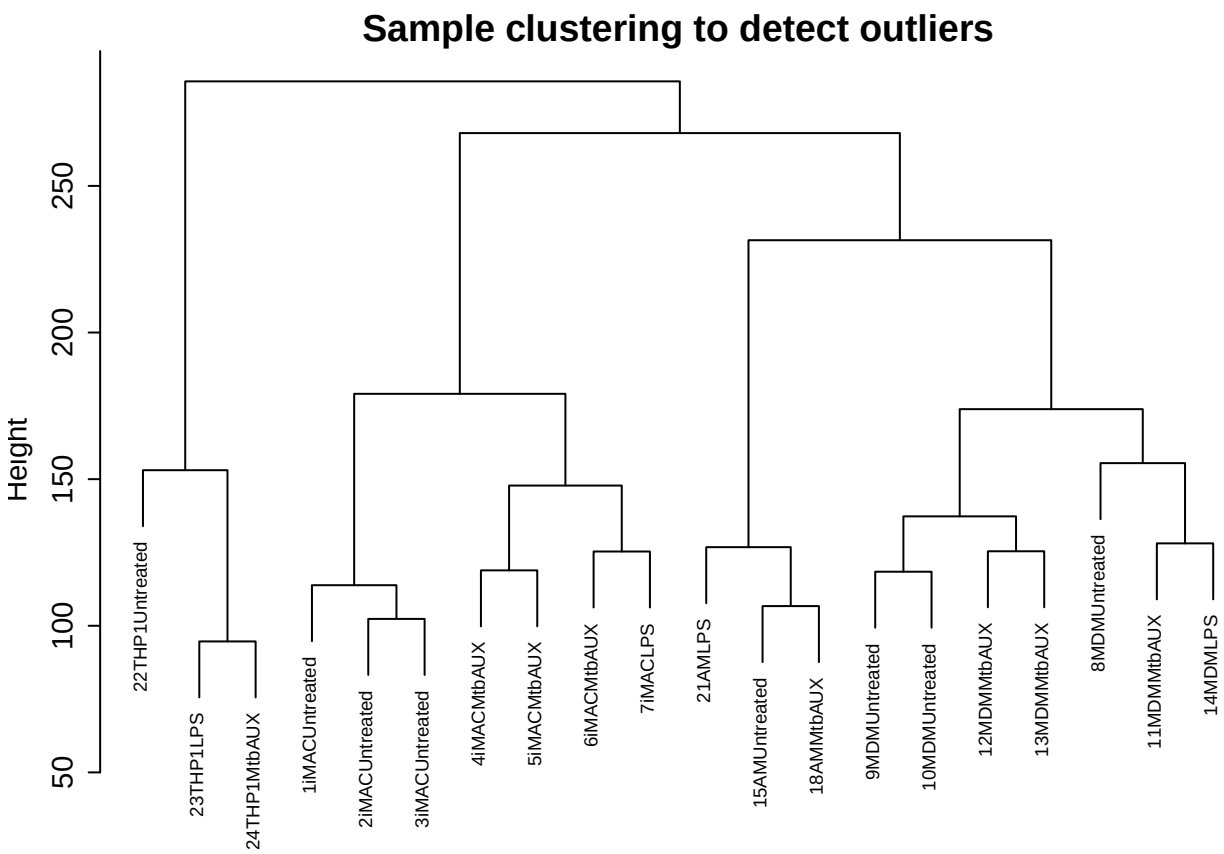
```
source_rmd = function(file, ...) {
  tmp_file = tempfile(fileext=".R")
  on.exit(unlink(tmp_file), add = TRUE)
  knitr::purl(file, output=tmp_file)
  source(file = tmp_file, ...)
}
```

Reading in the raw data and the functions

```
options(knitr.duplicate.label = "allow")
source_rmd("rawdata_normalization.rmd")
source_rmd("functions.rmd")
```

```
# Transposing the normalized counts for WGCNA
m <- t(norm_exp_matrix_am_rm)
```

```
#Group data in a dendogram to check outliers
sampleTree = hclust(dist(m), method = "average")

par(cex = 0.6)
par(mar = c(0, 4, 2, 0))
plot(
  sampleTree,
  main = "Sample clustering to detect outliers",
  sub = "",
  xlab = "",
  cex.lab = 1.5,
  cex.axis = 1.5,
  cex.main = 2
)
```




Finding sof threshold for WGCNA

Plotting the threshold picks

```
par(mfrow = c(1, 2))

cex1 = 0.9


plot(
  sft$fitIndices[, 1],-sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
  xlab = "Soft Threshold (power)",
  ylab = "Scale Free Topology Model Fit, signed R^2",
  main = paste("Scale independence")
)
text(
  sft$fitIndices[, 1],-sign(sft$fitIndices[, 3]) * sft$fitIndices[, 2],
  labels = powers,
  cex = cex1,
  col = "red"
)
abline(h = 0.90, col = "red")
plot(
  sft$fitIndices[, 1],
  sft$fitIndices[, 5],
  xlab = "Soft Threshold (power)",
  ylab = "Mean Connectivity",
```

```
  type = "n",
  main = paste("Mean connectivity")
)
text(
  sft$fitIndices[, 1],
  sft$fitIndices[, 5],
  labels = powers,
  cex = cex1,
  col = "red"
)
```

Running WGCNA - Takes a couple of minutes

Results generated from the following chunk

```
netwk <- readRDS("wgcna_netwk.rdata")

#netwk <- readRDS("wgcna_results_am_rm.rdata")
```

```
# Prevents namespace error
temp_cor <- cor
cor <- WGCNA::cor
picked_power <- 14

# blockwise module function
netwk <- blockwiseModules(
  m,
  # <= input here
  # simmilarity matrix options
  corType = "pearson",
  # spearman does not work for signed networks

  # == Adjacency Function ==
  power = 12,
  # <= power here
  networkType = "signed",
  # should the network represent
  #only positive or both negative and positive corralation

  # TOM options
  TOMType = "signed",

  # == Tree and Block Options ==
  deepSplit = 1,
  ## The bigger this number the larger
  ##the pathways the modules will describe

  pamRespectsDendro = F,
  detectCutHeight = 0.9,
  minModuleSize = 8,
  ## How small the modules can be, somewhere between 5 and 30
  maxBlockSize = 4000,

  # == Module Adjustments ==
```

```
  reassignThreshold = 0,
  mergeCutHeight = 0.25,

  # == TOM == Archive the run results in TOM file (saves time)
  saveTOMs = T,
  saveTOMFileBase = "WGCNA/ER",

  # == Output Options
  numericLabels = T,
  verbose = 3
)

#write_rds(netwk, "wgcna_netwk.rdata")
```

Merging close modules

```
moduleColors <- labels2colors(netwk$colors)


merged <-
  mergeCloseModules(m, moduleColors, cutHeight = 0.25, verbose = 3)
```

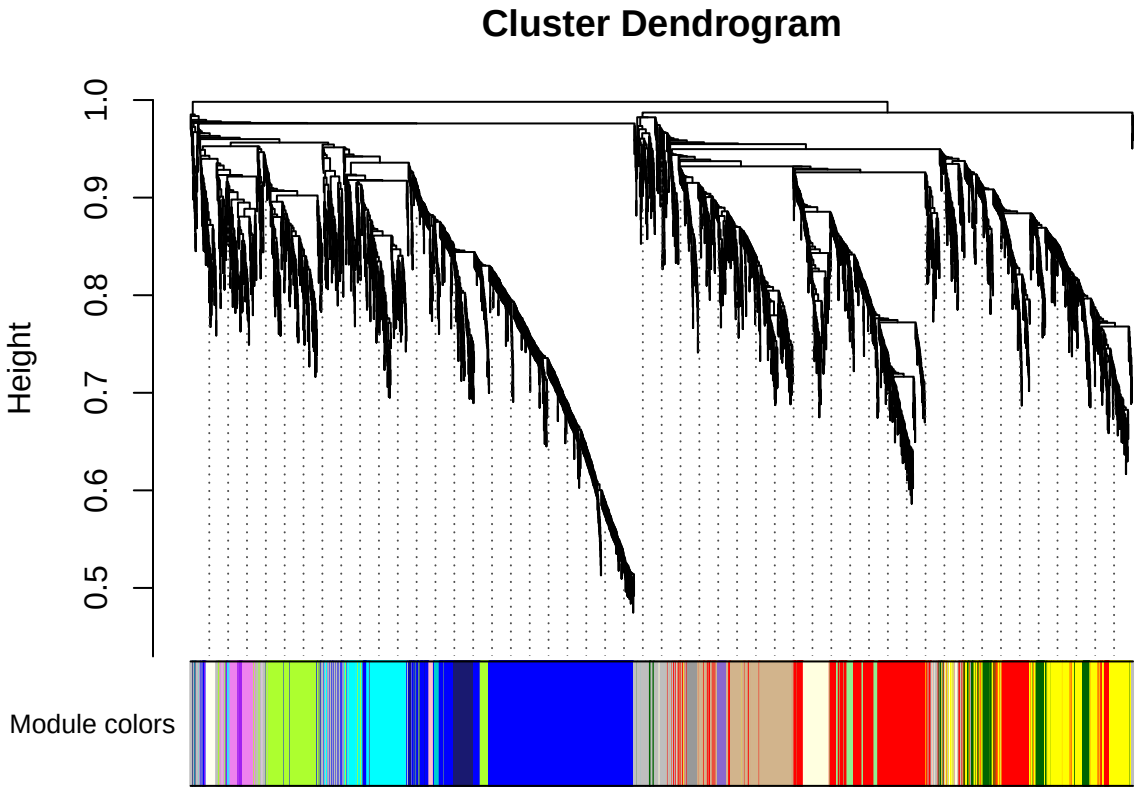
```
##  mergeCloseModules: Merging modules whose distance is less than 0.25
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 49 module eigengenes in given set.
##    Calculating new MEs...
##    multiSetMEs: Calculating module MEs.
##      Working on set 1 ...
##      moduleEigengenes: Calculating 49 module eigengenes in given set.
```

```
#Grouping module colors
mergedColors = merged$colors

#Eigengenes of new grouped modules
mergedMEs = merged$newMEs

#Renaming the module colors
moduleColors = mergedColors

#Building numeric labels corresponding to the colors
colorOrder = c("grey", standardColors(50))
moduleLabels = match(moduleColors, colorOrder) - 1
MEs = mergedMEs
#dim(dissTOM)
```

Plotting module dendrogram

```
geneTree = netwk$dendrograms[[1]]
# Plot the dendrogram and the module colors underneath
plotDendroAndColors(
  netwk$dendrograms[[1]],
```

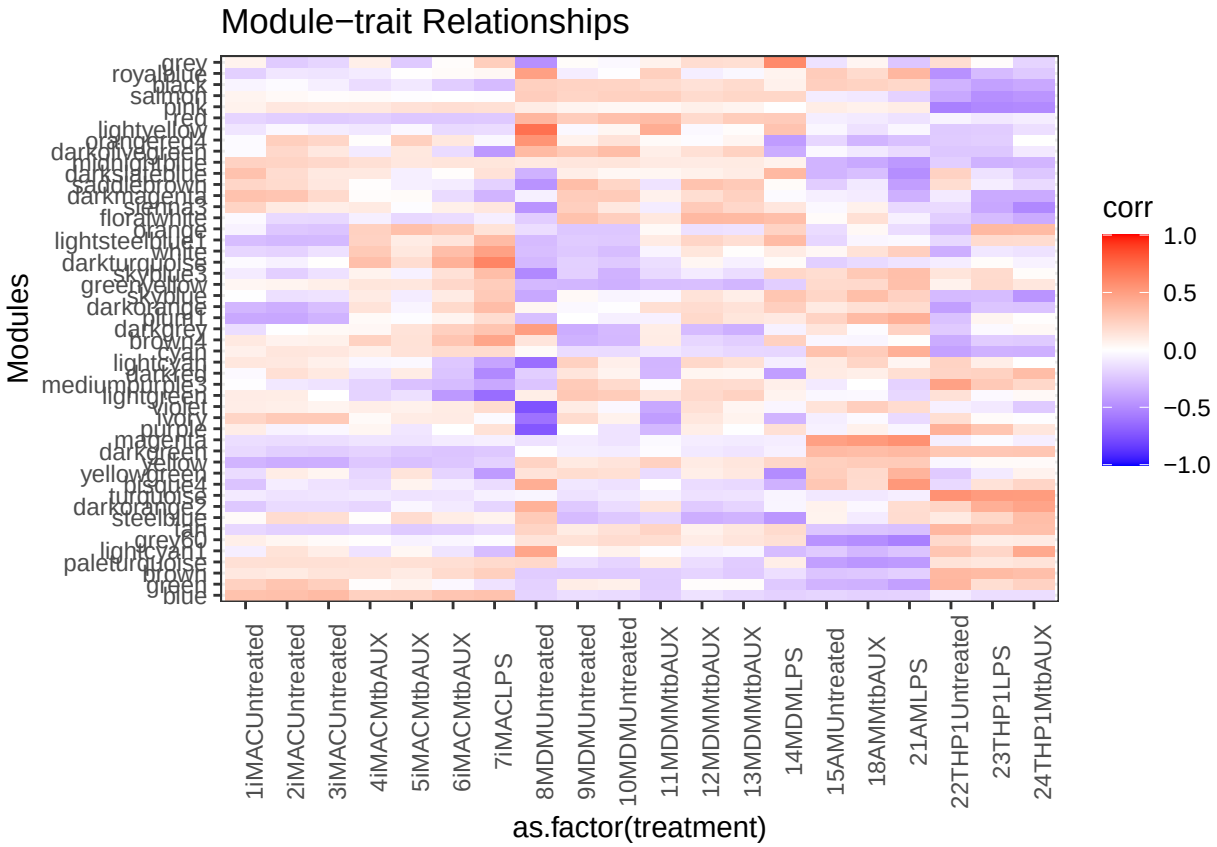
```
  mergedColors[netwk$blockGenes[[1]]],
  "Module colors",
  dendroLabels = FALSE,
  hang = 0.03,
  addGuide = TRUE,
  guideHang = 0.05
)
```




Defining the module eigengenes and plotting their corralation to the treatments for visual inspection

```
module_order = names(MEs) %>% gsub("ME", "", .)
MEs$treatment = row.names(MEs)


mME = MEs %>%
  pivot_longer(-treatment) %>%
  mutate(name = gsub("ME", "", name),
         name = factor(name, levels = module_order))

mME$treatment <- factor(mME$treatment, levels = row.names(MEs))

mME %>% ggplot(., aes(
  x = as.factor(treatment),
  y = name,
  fill = value
)) +
```

```
  geom_tile() +
  theme_bw() +
  scale_fill_gradient2(
    low = "blue",
    high = "red",
    mid = "white",
    midpoint = 0,
    limit = c(-1, 1)
  ) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title = "Module-trait Relationships", y = "Modules", fill = "corr")
```



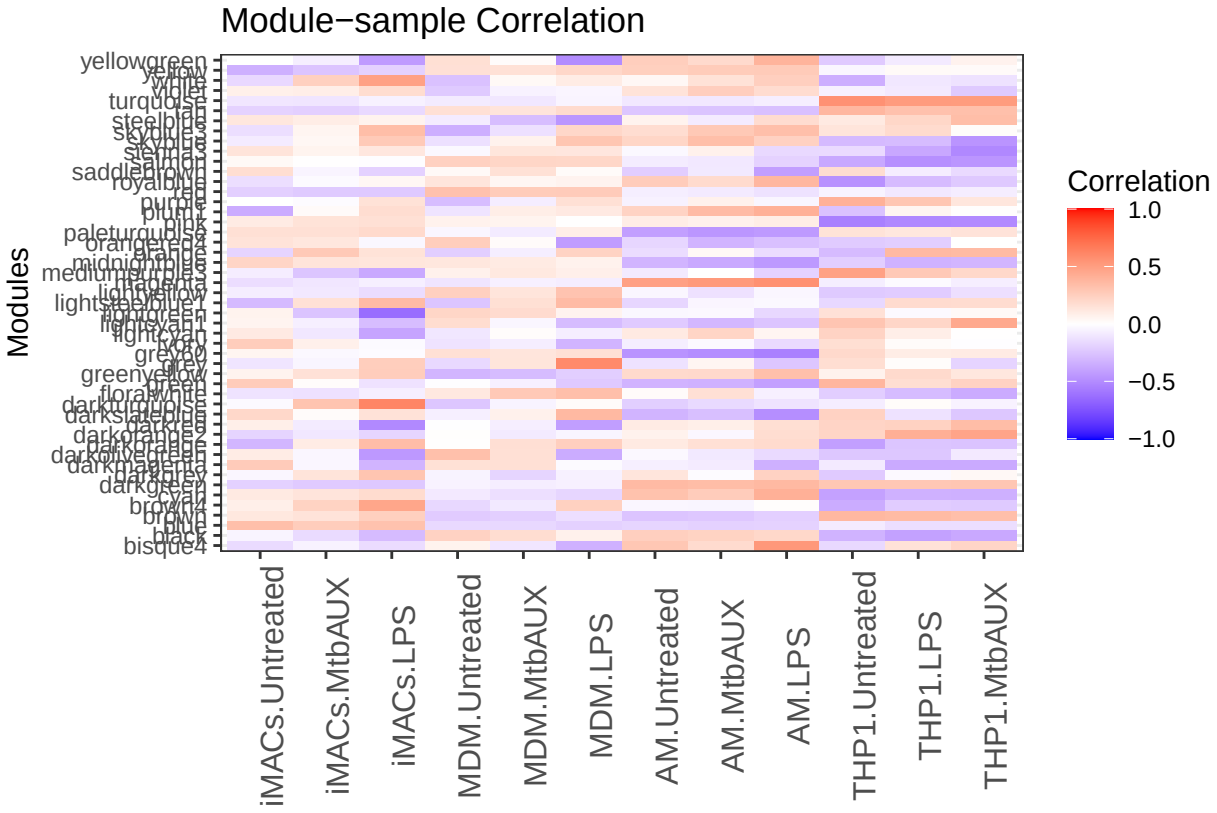

```
# Ordering the mME according to the ME number
mME <- mME[order(mME$name), ]
# Adding condition to the mME dataframe for averaging
mME$condition <-
  rep(sample.info$Condition, times = length(unique(mME$name)))
```

Creating another heatmap for visual inspection but collapsing the replicates by averaging values

Plotting the collapsed heatmap

```
mME_collapsed_replicates$condition <-
  factor(mME_collapsed_replicates$condition,
         levels = unique(mME_collapsed_replicates$condition))
```

```
mME_collapsed_replicates %>% ggplot(., aes(
  x = as.factor(condition),
  y = name,
  fill = average_value
)) +
  geom_tile() +
  theme_bw() +
  scale_fill_gradient2(
    low = "blue",
    high = "red",
    mid = "white",
    midpoint = 0,
    limit = c(-1, 1)
  ) +
  theme(axis.text.x = element_text(angle = 90, size = 12)) +
  labs(title = "Module-sample Correlation",
       y = "Modules",
       x = "",
       fill = "Correlation")
```




Finding differentially expressed modules using linear modeling

```
MEs0 = moduleEigengenes(m, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
```

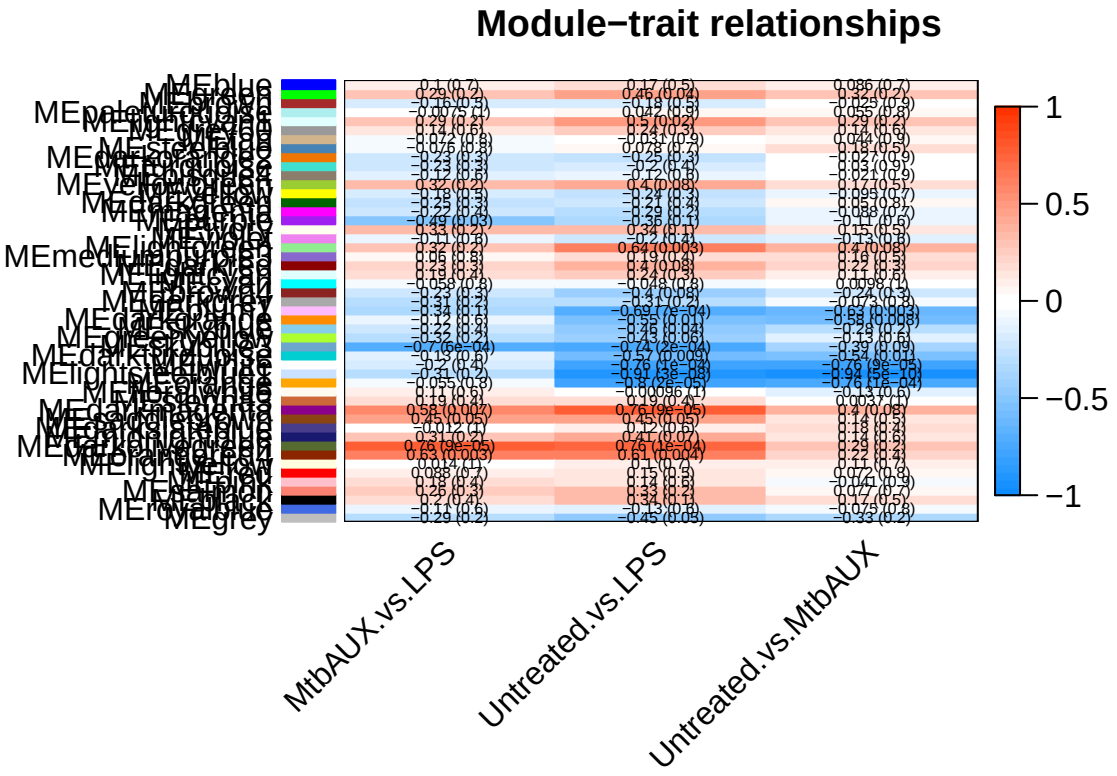
```r
# Creating a binary variable for untreated vs treated (Mtb and LPS)
sample.info$status <-
  c(rep(c(rep(0, times = 3), rep(1, times = 4)), times = 2), 0, 1, 1, 0, 1, 1)
#sample.info$status[c(15,16,17)] <- 2

mm <- model.matrix( ~ sample.info$status)

# Linear modeling
fit <- lmFit(t(MEs), design = mm)



efit3 <- limma::eBayes(fit)

# Creating dataframe of modules with highest difference across all samples
stats_df <- topTable(efit3, number = ncol(MEs)) %>%
  rownames_to_column("module")

stats_df %>% head()
```

```
##              module     logFC       AveExpr        t      P.Value   adj.P.Val
## 1 MElightsteelblue1 0.4108089  2.688821e-17 4.046784 5.648202e-05 0.002767619
## 2          MEorange 0.3354772 -5.551115e-18 3.304708 9.890839e-04 0.017734868
## 3           MEwhite 0.3327800 -9.540979e-18 3.278139 1.085808e-03 0.017734868
## 4           MEplum1 0.3041867 -1.491862e-17 2.996473 2.807705e-03 0.034394388
## 5      MEdarkorange 0.2572963  1.882175e-17 2.534566 1.143057e-02 0.112019562
## 6        MEskyblue3 0.2388516 -7.979728e-18 2.352872 1.884784e-02 0.147569198
##           B
## 1  1.577566
## 2 -1.073314
## 3 -1.158301
## 4 -2.017093
## 5 -3.258538
## 6 -3.690047
```

Finding differentially expressed modules based on correlation

```r
#Relating modules to characteristics and identifying important genes
#Defining the number of genes and samples
nGenes = ncol(m)
nSamples = nrow(m)


binary <-
  binarizeCategoricalVariable(sample.info$Treatment, includePairwise = TRUE)

rownames(binary) <- sample.info$Sample_ID
binary <- as.data.frame(binary)


moduleTraitCor = cor(MEs, binary, use = "p")
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples)
```

```
#Displaying correlations and its p-values
textMatrix =  paste(signif(moduleTraitCor, 2),
                    " (",
                    signif(moduleTraitPvalue, 1),
                    ")",
                    sep = "")
dim(textMatrix) = dim(moduleTraitCor)
par(mar = c(8, 10.5, 3, 3))

#Displaying the correlation values in a heatmap plot
labeledHeatmap(
  Matrix = moduleTraitCor,
  xLabels = colnames(binary),
  yLabels = names(MEs),
  ySymbols = names(MEs),
  colorLabels = FALSE,
  colors = blueWhiteRed(50),
  textMatrix = textMatrix,
  setStdMargins = FALSE,
  cex.text = 0.5,
  zlim = c(-1, 1),
  main = paste("Module-trait relationships")
)
```



```
# Creating a dataframe to identify the module number to the module color
colorh <- labels2colors(netwk$colors)
module_colors_number <-
  data.frame(
    "gene_id" = names(netwk$colors),
    "module_number" = unname(netwk$colors),
    "color" = colorh
  )
module_colors_number <-
  module_colors_number %>% select(c(2, 3)) %>% unique()
module_colors_number <-
  module_colors_number[order(module_colors_number$module_number), ]
```

Creating a dataframe of the genes and their respective module

Creating dataframes of the modules

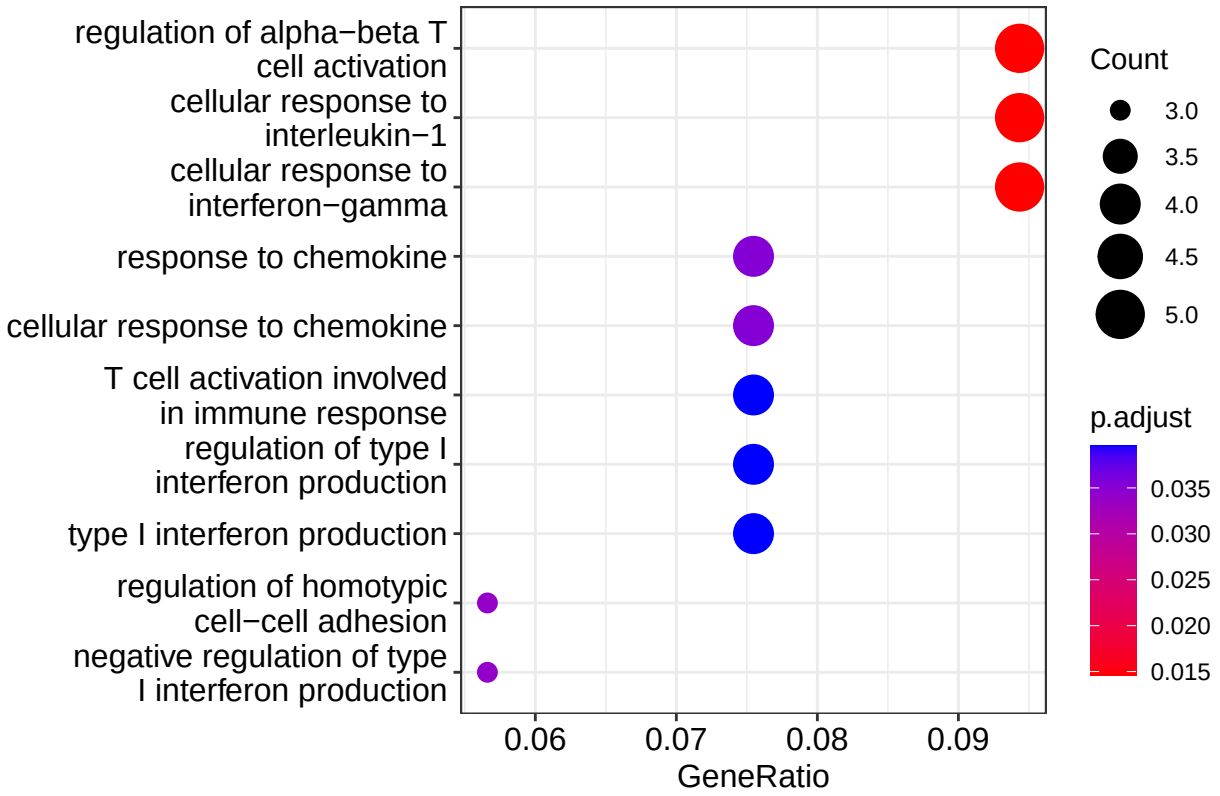
```
gene_module_key <-
  enframe(netwk$colors, name = "Gene_ID", value = "module") %>%
  mutate(module = paste0("ME", module))

ME41 <- gene_module_key %>%
  filter(module == "ME41") %>% inner_join(dplyr::select(gene.info, c(gene_source, Gene_ID)))

ME25 <- gene_module_key %>%
  filter(module == "ME25") %>% inner_join(dplyr::select(gene.info, c(gene_source, Gene_ID)))

ME27 <- gene_module_key %>%
  filter(module == "ME27") %>% inner_join(dplyr::select(gene.info, c(gene_source, Gene_ID)))
```

```
module_df <- MEs %>%
  rownames_to_column("Sample_ID") %>%
  # Here we are performing an inner join with a subset of metadata
  inner_join(sample.info %>%
               dplyr::select(Sample_ID, Treatment),
             by = "Sample_ID")

# Adding sample.info
module_df <-
  module_df %>% inner_join(sample.info, by = "Sample_ID") %>% dplyr::select(c(-"Treatment.x", "status"))
```

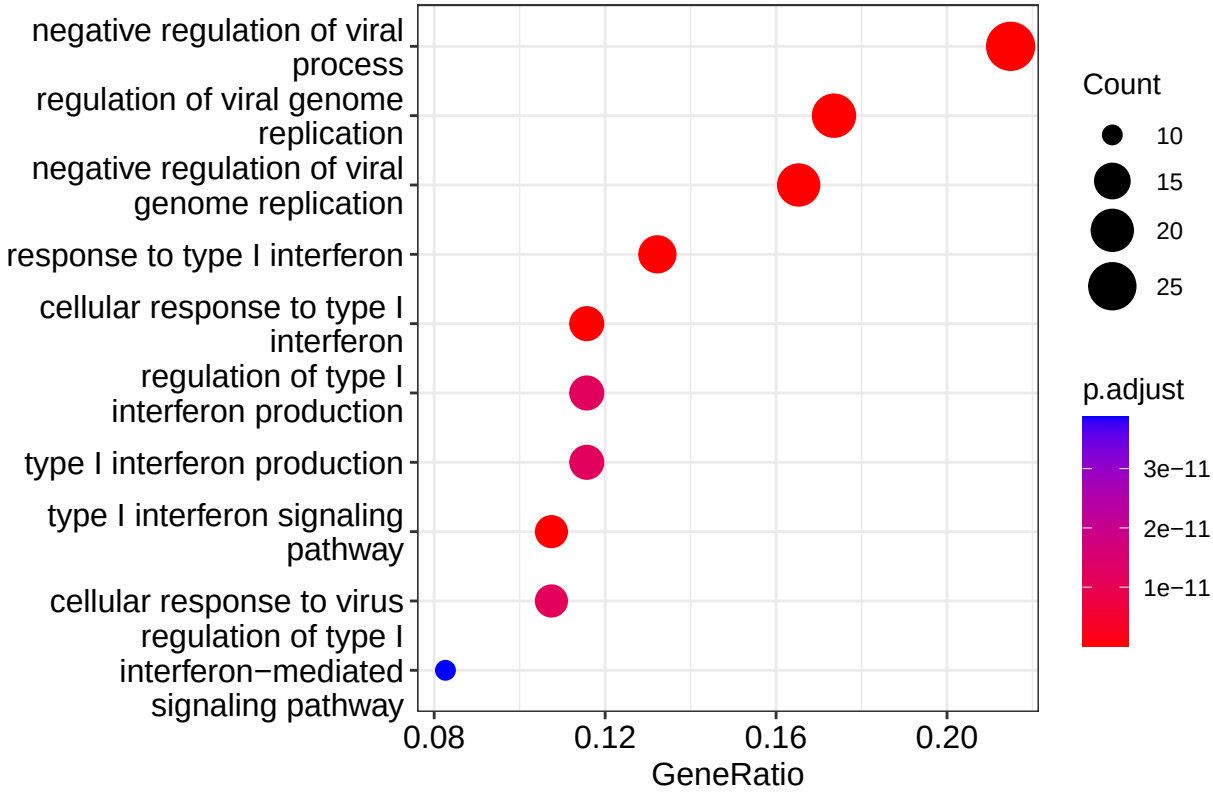
Creating a function to find the BP enriched in the different modules

```
module_go_enrichment <- function(deg_df, module_nr) {
  module_df <- gene_module_key %>%
    filter(module == module_nr) %>% inner_join(dplyr::select(gene.info, c(gene_source, Gene_ID)))

  # all genes for background
  all_background <- deg_df$Gene_ID %>% as.character()

  module_degs <-
    module_df %>% inner_join(deg_df, by = "gene_source") %>% dplyr::select(c("Gene_ID.x", "gene_source"
```

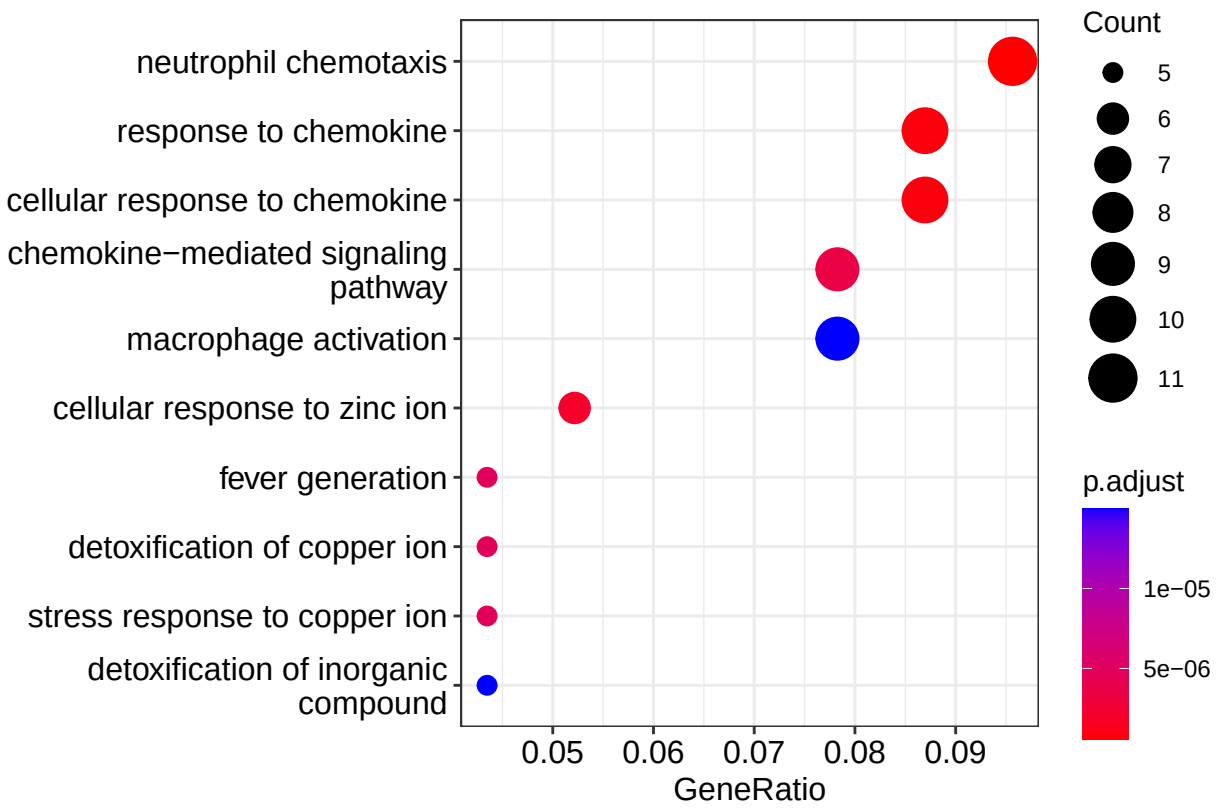
```
  sig_genes <- module_degs #%>% filter(P.Value < 0.05)
  sig_genes <- sig_genes$Gene_ID.x


  ego <- enrichGO(
    gene = sig_genes,
    universe = all_background,
    keyType = "ENSEMBL",
    OrgDb = org.Hs.eg.db,
    ont = "BP",
    maxGSSize = 100,
    pAdjustMethod = "BH",
    qvalueCutoff = 0.05,
    readable = TRUE
  )

  ## Output results from GO analysis to a table
  cluster_summary <- data.frame(ego)
  ego
}


enrich_module_41 <-
  module_go_enrichment(results$iMACs.MtbAUXvsiMACs.Untreated, "ME41")

enrich_module_25 <-
  module_go_enrichment(results$iMACs.MtbAUXvsiMACs.Untreated, "ME25")

enrich_module_27 <-
  module_go_enrichment(results$AM.MtbAUXvsAM.Untreated, "ME27")


dotplot(enrich_module_41)
```




```
dotplot(enrich_module_25)
```




```
dotplot(enrich_module_27)
```




```
extract_genes_in_term <- function(enrichment_df) {
  no_terms <- 10

  enriched_pathway_genes <-
    data.frame("term" = NULL, "genes" = NULL)

  for (term in 1:no_terms) {
    str <- str_split(enrichment_df$geneID[term], "/")
    enriched_pathway_genes <-
      rbind(
        data.frame("term" = enrichment_df$Description[term], "genes" = str[[1]]),
        enriched_pathway_genes
      )

  }
  names(enriched_pathway_genes) <- c("terms", "gene_source")
  enriched_pathway_genes
}
```

```
module_number_df <- enrich_module_41
heatmap_title <- "test"

module_enrichr_heatmap <-
  function(module_number_df, heatmap_title) {
    #Getting the correct order of the terms according to gene ratio and adj p value
    dotplot_df <- dotplot(module_number_df, showCategory = 10)
```

```
dotplot_df_ordered <-
  dotplot_df$data[with(dotplot_df$data, order(-GeneRatio, pvalue)), ]
factor(dotplot_df_ordered$Description, levels = dotplot_df_ordered$Description)

# Function to extract the genes from the go enriched terms created with the module genes
extract_genes_in_term <- function(enrichment_df) {
  no_terms <- 10

  enriched_pathway_genes <-
    data.frame("term" = NULL, "genes" = NULL)

  for (term in 1:no_terms) {
    str <- str_split(enrichment_df$geneID[term], "/")
    enriched_pathway_genes <-
      rbind(
        data.frame("term" = enrichment_df$Description[term], "genes" = str[[1]]),
        enriched_pathway_genes
      )

  }
  names(enriched_pathway_genes) <- c("terms", "gene_source")
  enriched_pathway_genes
}


z_transformed_avg_norm_exp_am_rm <-
  z_transformed_avg_norm_exp_am_rm %>%
  as.data.frame() %>%
  rownames_to_column("Gene_ID") %>%
  inner_join(select(gene.info, "Gene_ID", "gene_source"), by = "Gene_ID")


# Use the function for extracting the genes in the different terms
extracted_genes <- extract_genes_in_term(module_number_df)

# Use the sample.info file to add some metadata for plotting
tmp <- sample.info[, c(2, 4)]
colnames(tmp) = c("celltype", "Sample_ID")

# Converting the z-transformed averaged normalized data into tidy data for plotting
## This is the final dataframe for heatmap plotting
heatmap_df_long_tidy <-
  z_transformed_avg_norm_exp_am_rm %>% right_join(extracted_genes, by = "gene_source") %>% pivot_lo


  right_join(tmp, by = "Sample_ID") %>% unique()

# awkward way of changing position of THP1 LPS and MTB since for some reason LPs is before Mtb unli
## Used for facet
tmp_tmp <- tmp[20,]
tmp[20,] <- tmp[19,]
tmp[19,] <- tmp_tmp
```

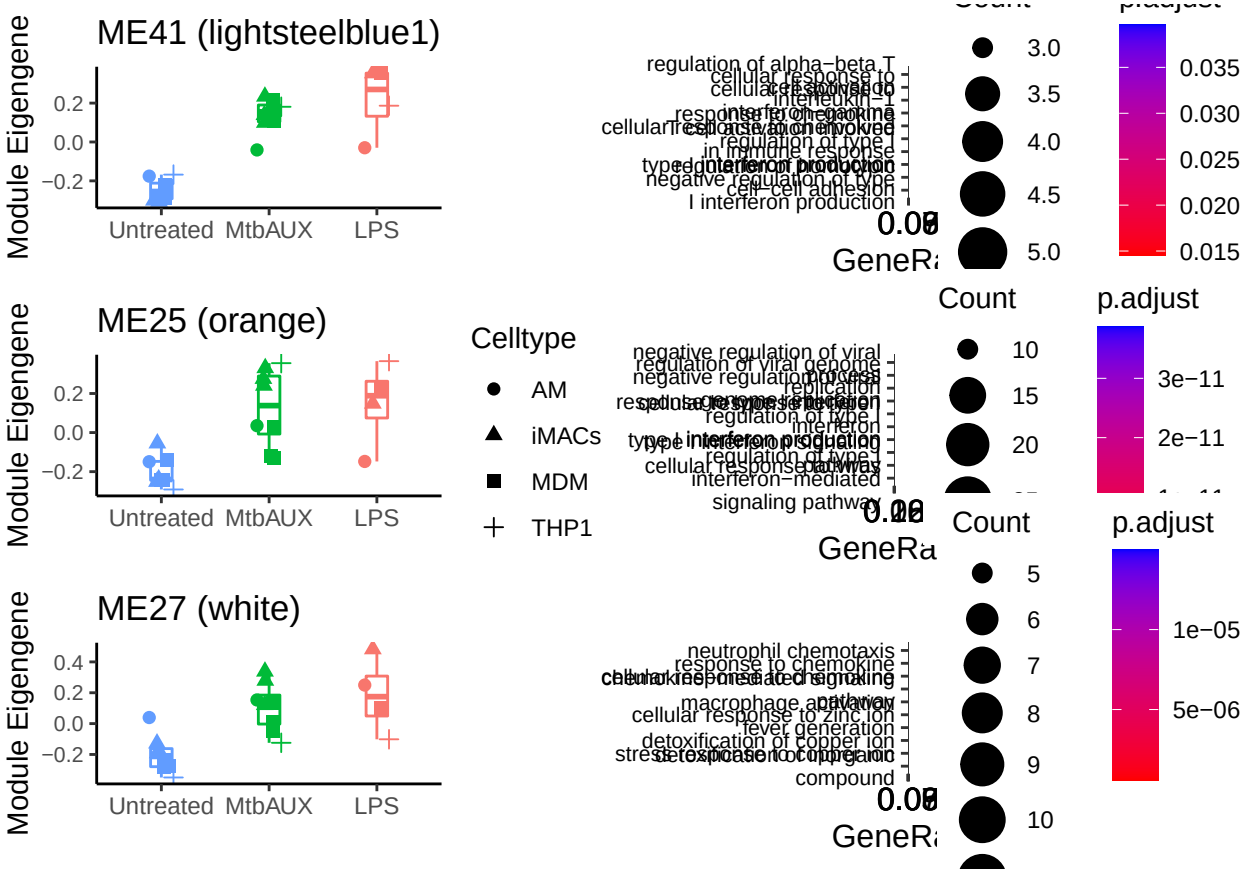
```
    # creating levels for the sample order
    x_level <-
      factor(unique(heatmap_df_long_tidy$Sample_ID),
             levels = unique(as.vector(tmp$Sample_ID)))

    # Plotting the heatmap
    ggplot(heatmap_df_long_tidy,
           mapping = aes(factor(Sample_ID, levels = levels(x_level)), gene_source, fill = zscore)) +
      geom_tile() +
      facet_grid(
        factor(heatmap_df_long_tidy$terms, levels = dotplot_df_ordered$Description) ~
          factor(
            heatmap_df_long_tidy$celltype,
            levels = unique(heatmap_df_long_tidy$celltype)
          ),
        scales = "free",
        space = "free",
        labeller = label_wrap_gen()
      ) +
      scale_fill_continuous(low = "#56B1F7", high = "#132B43") +
      theme(
        axis.text.x = element_text(
          angle = 90,
          vjust = 0.5,
          hjust = 1
        ),
        strip.text.y = element_text(angle = 360, size = 7),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank()
      ) +
      labs(title = heatmap_title, x = "", y = "Genes")

    #heatmap_df_long_tidy
  }

module_enrichr_heatmap(enrich_module_27, "Module ")
module_enrichr_heatmap(iMAC_module_22, "Module 22")
```

Plotting the BP enrichment and boxplots of module trait association together

```
fontsize = 8
theme_box <- theme(
  #legend.direction = "horizontal",
  legend.position = "top",
  legend.box = "vertical",
  axis.text.y = element_text(size = fontsize)
)




one <- ggplot(module_df,
              aes(
```

```
                x = factor(Treatment.y, levels = c("Untreated", "MtbAUX", "LPS")),
                y = MElightsteelblue1,
                color = Treatment.y
              )) +
  geom_boxplot(width = 0.2, outlier.shape = NA) +
  labs(title = "ME41 (lightsteelblue1)", shape = "Celltype", x = "") +
  geom_sina(
    maxwidth = 0.3,
    aes(shape = Sample_Group),
    position = "dodge",
    show.legend = TRUE,
    size = 2
  ) +
  theme_classic() +
  labs(y = "Module Eigengene") +
  guides(color = FALSE) +
  theme_box


two <- ggplot(module_df,
              aes(
                x = factor(Treatment, levels = c("Untreated", "MtbAUX", "LPS")),
                y = MEorange,
                color = Treatment
              )) +
  geom_boxplot(width = 0.2, outlier.shape = NA) +
  labs(title = "ME25 (orange)", shape = "Celltype", x = "") +
  geom_sina(
    maxwidth = 0.3,
    aes(shape = Sample_Group),
    position = "dodge",
    show.legend = TRUE,
    size = 2
  ) +
  theme_classic() +
  labs(y = "Module Eigengene") +
  guides(color = FALSE) +
  theme_box


three <- ggplot(module_df,
                aes(
                  x = factor(Treatment, levels = c("Untreated", "MtbAUX", "LPS")),
                  y = MEwhite,
                  color = Treatment
                )) +
  geom_boxplot(width = 0.2, outlier.shape = NA) +
  labs(title = "ME27 (white)", shape = "Celltype", x = "") +
  geom_sina(
    maxwidth = 0.3,
    aes(shape = Sample_Group),
    position = "dodge",
    show.legend = TRUE,
```

```
    size = 2
  ) +
  theme_classic() +
  labs(y = "Module Eigengene") +
  guides(color = FALSE) +
  theme_box


fontsize <- 8

theme <- theme(
  #legend.direction = "horizontal",
  legend.position = "right",
  legend.box = "horizontal",
  axis.text.y = element_text(size = fontsize)
)

one1 <- dotplot(enrich_module_41, showCategory = 10) +
  #scale_size(range = c(1,5)) +
  #scale_color(range = c(1,5)) +
  theme

two2 <- dotplot(enrich_module_25, showCategory = 10) +
  #scale_size(range = c(1,5)) +
  #scale_color(range = c(1,5)) +
  theme

three3 <- dotplot(enrich_module_27 , showCategory = 10) +
  #scale_size(range = c(1,5)) +
  #scale_color(range = c(1,5)) +
  theme


box <-
  ggarrange(
    one,
    two,
    three,
    common.legend = TRUE,
    legend = "right",
    nrow = 3
  )
dot <-
  ggarrange(
    one1,
    two2,
    three3,
    common.legend = FALSE,
    legend = "right",
    nrow = 3
  )

ggarrange(box, dot)
```



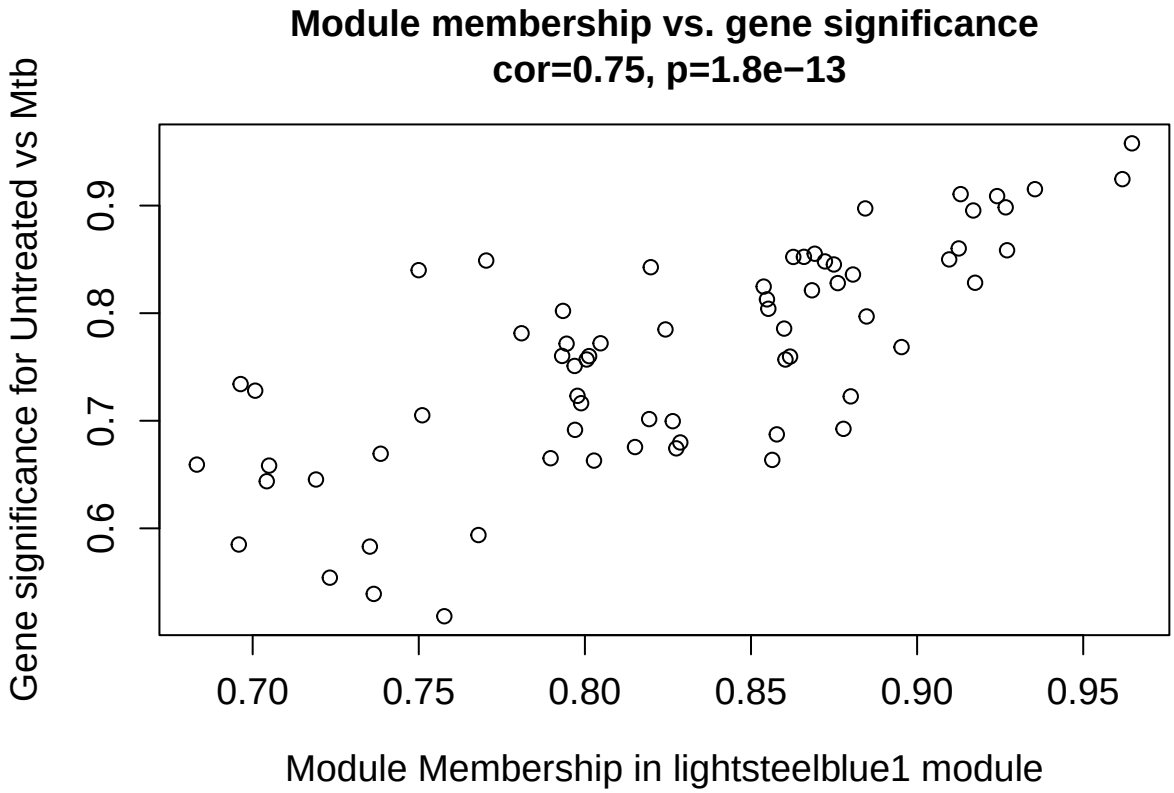

Assessing module membership and gene significance of the modules

```
#Defining the variable Peso10dias containing the column Peso10dias of datTrait
unvsmtb = as.data.frame(binary$Untreated.vs.MtbAUX)
names(unvsmtb) = "UntreatedvsMtbAUX"

#names (colors) of the modules
modNames = substring(names(MEs), 3)

geneModuleMembership = as.data.frame(cor(m, MEs, use = "p"))
MMPvalue = as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership), nSamples))

names(geneModuleMembership) = paste("MM", modNames, sep = "")
names(MMPvalue) = paste("p.MM", modNames, sep = "")

geneTraitSignificance = as.data.frame(cor(m, unvsmtb, use = "p"))
GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance), nSamples))

names(geneTraitSignificance) = paste("GS.", names(unvsmtb), sep = "")
names(GSPvalue) = paste("p.GS.", names(unvsmtb), sep = "")

module = "lightsteelblue1" ##########################putting the color below the plot
# module = "white"
# module = "orange"
column = match(module, modNames)
moduleGenes = moduleColors == module
```

```
modNames
```

```
##  [1] "blue"            "green"           "brown"           "paleturquoise"
##  [5] "lightcyan1"      "grey60"          "tan"             "steelblue"
##  [9] "darkorange2"     "turquoise"       "bisque4"         "yellowgreen"
## [13] "yellow"          "darkgreen"       "magenta"         "purple"
## [17] "ivory"           "violet"          "lightgreen"      "mediumpurple3"
## [21] "darkred"         "lightcyan"       "cyan"            "brown4"
## [25] "darkgrey"        "plum1"           "darkorange"      "skyblue"
## [29] "greenyellow"     "skyblue3"        "darkturquoise"   "white"
## [33] "lightsteelblue1" "orange"          "floralwhite"     "sienna3"
## [37] "darkmagenta"     "saddlebrown"     "darkslateblue"   "midnightblue"
## [41] "darkolivegreen"  "orangered4"      "lightyellow"     "red"
## [45] "pink"            "salmon"          "black"           "royalblue"
## [49] "grey"
```

```
#sizeGrWindow(7, 7)
par(mfrow = c(1, 1))
verboseScatterplot(
  abs(geneModuleMembership[moduleGenes, column]),
  abs(geneTraitSignificance[moduleGenes, 1]),
  xlab = paste("Module Membership in", module, "module"),
  ylab = "Gene significance for Untreated vs Mtb",
  main = paste("Module membership vs. gene significance\n"),
  cex.main = 1.2,
  cex.lab = 1.2,
  cex.axis = 1.2,
  col = "black"
)
```




making a dataframe of the MM and GS for all modules and genes

```
# Filter the results dataframes for SDEGS
sdeg_am <- sdeg_extraction(results$AM.MtbAUXvsAM.Untreated)
sdeg_mdm <- sdeg_extraction(results$MDM.MtbAUXvsMDM.Untreated)
sdeg_imac <- sdeg_extraction(results$iMACs.MtbAUXvsiMACs.Untreated)
sdeg_thp1 <- sdeg_extraction(results$THP1.MtbAUXvsTHP1.Untreated)

# Create a shared DF of all SDEGs
all_sdegs <- bind_rows(sdeg_imac, sdeg_am, sdeg_mdm) %>% dplyr::select(-2)


options(scipen = 999)
#Display the gene names inside the module
#colnames(expression0)[moduleColors=="pink"]

#Identifying most important genes for one determined characteristic inside of the cluster
geneInfo0 = data.frame(Gene_ID = colnames(m),
                       moduleColor = moduleColors,
                       geneTraitSignificance,
                       GSPvalue)


modOrder = order(-abs(cor(MEs, unvsmtb, use = "p")))
for (mod in 1:ncol(geneModuleMembership))
{
```

```
  oldNames = names(geneInfo0)
  geneInfo0 = data.frame(geneInfo0, geneModuleMembership[, modOrder[mod]],
                         MMPvalue[, modOrder[mod]])

  names(geneInfo0) = c(oldNames,
                       paste("MM.", modNames[modOrder[mod]], sep = ""),
                       paste("p.MM.", modNames[modOrder[mod]], sep = ""))
}


geneOrder = order(geneInfo0$moduleColor,-abs(geneInfo0$GS.UntreatedvsMtbAUX))
geneInfo = geneInfo0[geneOrder,]




write_csv(
  geneInfo %>% filter(moduleColor == "lightsteelblue1") %>%
    inner_join(gene.info[c(1, 11)], by = "Gene_ID") %>% relocate(103) %>%
    dplyr::select(1:7) %>% filter(gene_source %in% all_sdegs$gene_source),
  "lightsteelblue.csv"
)



write_csv(
  geneInfo %>% filter(moduleColor == "white") %>%
    inner_join(gene.info[c(1, 11)], by = "Gene_ID") %>%
    relocate(103) %>% dplyr::select(c(1:5, "MM.white", "p.MM.white")) %>%
    filter(gene_source %in% all_sdegs$gene_source),
  "white.csv"
)

write_csv(
  geneInfo %>% filter(moduleColor == "orange") %>%
    inner_join(gene.info[c(1, 11)], by = "Gene_ID") %>%
    relocate(103) %>% dplyr::select(c(1:5, "MM.orange", "p.MM.orange")) %>%
    filter(gene_source %in% all_sdegs$gene_source),
  "orange.csv"
)
```

```
steelblue_genes <- geneInfo %>%
  filter(moduleColor == "lightsteelblue1") %>%
  inner_join(gene.info[c(1,11)], by = "Gene_ID") %>%
  dplyr::select(gene_source)
steelblue_genes$original_names <- steelblue_genes[[1]]

white_genes <- geneInfo %>%
  filter(moduleColor == "white") %>%
  inner_join(gene.info[c(1,11)], by = "Gene_ID") %>%
  dplyr::select(gene_source)

orange_genes <- geneInfo %>%
```

```
  filter(moduleColor == "orange") %>%
  inner_join(gene.info[c(1,11)], by = "Gene_ID") %>%
  dplyr::select(gene_source)
```