Clinical microbiome data science - example workflow

This in an example data analysis for poster. Poster was presented in conference Microbiome Interactions in Health and Disease. It was held on 13-15 October 2021.

Other material about poster and the *miaverse* project you can find by following the links below

- Poster
  - poster
  - abstract
  - lightning talk
- *miaverse*
  - homepage
  - Orchestrating Microbiome Analysis

Example workflow

Multi-omics means that we integrate data from multiple sources. For example, we can integrate microbial abundances in the gut with biomolecular profiling data from blood samples. This kind of integrative multi-omic approaches can support the analysis of microbiome dysbiosis and facilitate the discovery of novel biomarkers for health and disease.

The data containers that the *miaverse* utilizes are scalable and they can contain different types of data in a same container. Because of that, the *miaverse* is well-suitable for multi-assay microbiome data which incorporates different types of complementary data sources in a single reproducible workflow.

This is a reproducible workflow. **You can run this example workflow**

- Copy-paste code chunks and run them
- Download this R markdown file and run it

Install and load required packages

Here, all required packages are installed and loaded into the session. If packages are already installed, installation step is skipped; only uninstalled packages are installed.

```
# List of packages that we need from cran and bioc
cran_pkg <- c("BiocManager", "ggplot2", "pheatmap", "stringr")
bioc_pkg <- c("microbiome", "microbiomeDataSets", "mia")

# Gets those packages that are already installed
cran_pkg_already_installed <- cran_pkg[ cran_pkg %in% installed.packages() ]
bioc_pkg_already_installed <- bioc_pkg[ bioc_pkg %in% installed.packages() ]

# Gets those packages that need to be installed
cran_pkg_to_be_installed <- setdiff(cran_pkg, cran_pkg_already_installed)
bioc_pkg_to_be_installed <- setdiff(bioc_pkg, bioc_pkg_already_installed)
```

```
# If there are packages that need to be installed, installs them from CRAN
if( length(cran_pkg_to_be_installed) ) {
   install.packages(cran_pkg_to_be_installed)
}
```

```
# If there are packages that need to be installed, installs them from Bioconductor
if( length(bioc_pkg_to_be_installed) ) {
   BiocManager::install(bioc_pkg_to_be_installed, ask = F)
}
```

Now all required packages are installed, so let's load them into the session. Some function names occur in multiple packages. That is why miaverse's packages mia and miaViz are prioritized. Packages that are loaded first have higher priority.

```
# Reorders bioc packages, so that mia and miaViz are first and have higher priority
bioc_pkg <- c(bioc_pkg[ bioc_pkg %in% c("mia", "miaViz") ],
              bioc_pkg[ !bioc_pkg %in% c("mia", "miaViz") ] )

# Loading all packages into session. Returns true if package was successfully loaded.
sapply(c(bioc_pkg, cran_pkg), require, character.only = TRUE)
```

```
##                mia          microbiome microbiomeDataSets         BiocManager
##               TRUE                TRUE               TRUE                TRUE
##            ggplot2           pheatmap             stringr
##               TRUE                TRUE               TRUE
```

Load data

Multi-assay data can be stored in altExp slot of TreeSE or MAE data container.

Different data sets are first imported into SE or TreeSE data container similarly to the case when only one data set is present. After that different data sets are combined into the same data container. Result is one TreeSE object with alternative experiment in altExp slot, or MAE object with multiple experiment in its experiment slot.

As an example data, we use data from following publication: Hintikka L *et al.* (2021) Xylo-oligosaccharides in prevention of hepatic steatosis and adipose tissue inflammation: associating taxonomic and metabolomic patterns in fecal microbiotas with biclustering.

In this article, mice were fed with high-fat and low-fat diets with or without prebiotics. The purpose of this was to study if prebiotics would reduce the negative impacts of high-fat diet.

This example data can be loaded from microbiomeDataSets. The data is already in MAE format. It includes three different experiments: microbial abundance data, metabolite concentrations, and data about different biomarkers. Help for importing data into SE object you can find from here.

For the sake of simplicity, we compare only fat-contents of diets.

```
# Load the data
mae <- microbiomeDataSets::HintikkaXOData()

# For simplicity, classify all high-fat diets as high-fat, and all the low-fat diets as low-fat diets
colData(mae)$Diet <- ifelse(colData(mae)$Diet == "High-fat" |
                              colData(mae)$Diet == "High-fat + XOS", "High-fat", "Low-fat")

# Drop off those bacteria that do not include information in Phylum or lower levels
mae[[1]] <- mae[[1]][!is.na(rowData(mae[[1]])$Phylum), ]

# Clean taxonomy data, so that names do not include addtional characters
rowData(mae[[1]]) <- DataFrame(apply(rowData(mae[[1]]), 2, str_remove, pattern = "._[0-9]__"))

mae
```

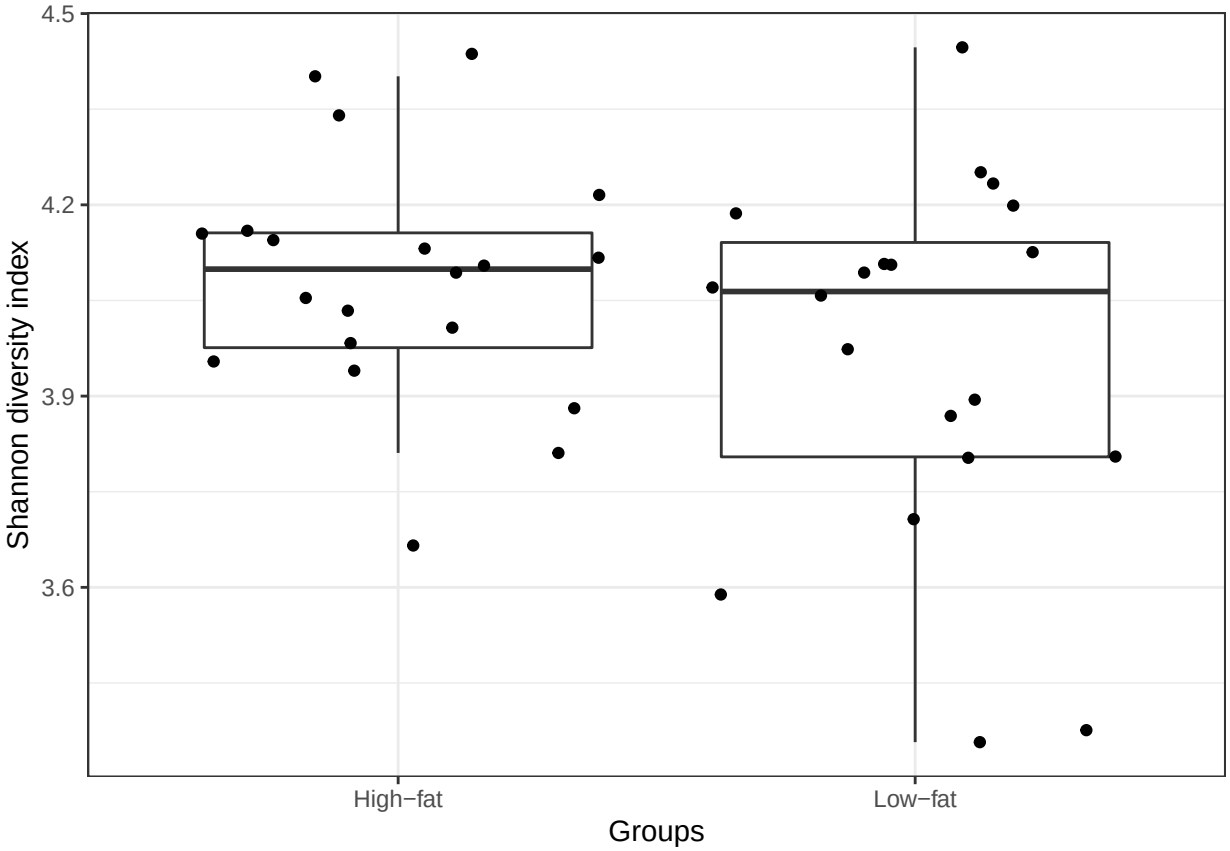
```
## A MultiAssayExperiment object of 3 listed
##  experiments with user-defined names and respective classes.
##  Containing an ExperimentList class object of length 3:
##  [1] microbiota: SummarizedExperiment with 12613 rows and 40 columns
##  [2] metabolites: SummarizedExperiment with 38 rows and 40 columns
##  [3] biomarkers: SummarizedExperiment with 39 rows and 40 columns
## Functionality:
##  experiments() - obtain the ExperimentList instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample coordination DataFrame
##  `$`, `[`, `[[` - extract colData columns, subset, or experiment
##  *Format() - convert into a long or wide DataFrame
##  assays() - convert ExperimentList to a SimpleList of matrices
##  exportClass() - save all data to files
```

Alpha diversity

Alpha diversity indices represent how abundances are distributed between different species. More information about alpha diversity you can find from here.

We can see that there is no statistically significant difference between high-fat and low-fat diets.

```
# Add sample meta data to microbiome data
colData(mae[[1]]) <- colData(mae)
# Calculate shannon index
mae[[1]] <- estimateDiversity(mae[[1]], index = "shannon")
# Get sample meta data as data frame
df <- as.data.frame(colData(mae[[1]]))
# Plot shannon index, compare diets
ggplot(df, aes_string(x = "Diet", y = "shannon")) +
  geom_boxplot(outlier.shape = NA) + # Remove outliers because jitter plot is also plotted
  geom_jitter() +
  xlab("Groups") +
  ylab("Shannon diversity index") +
  theme_bw()
```



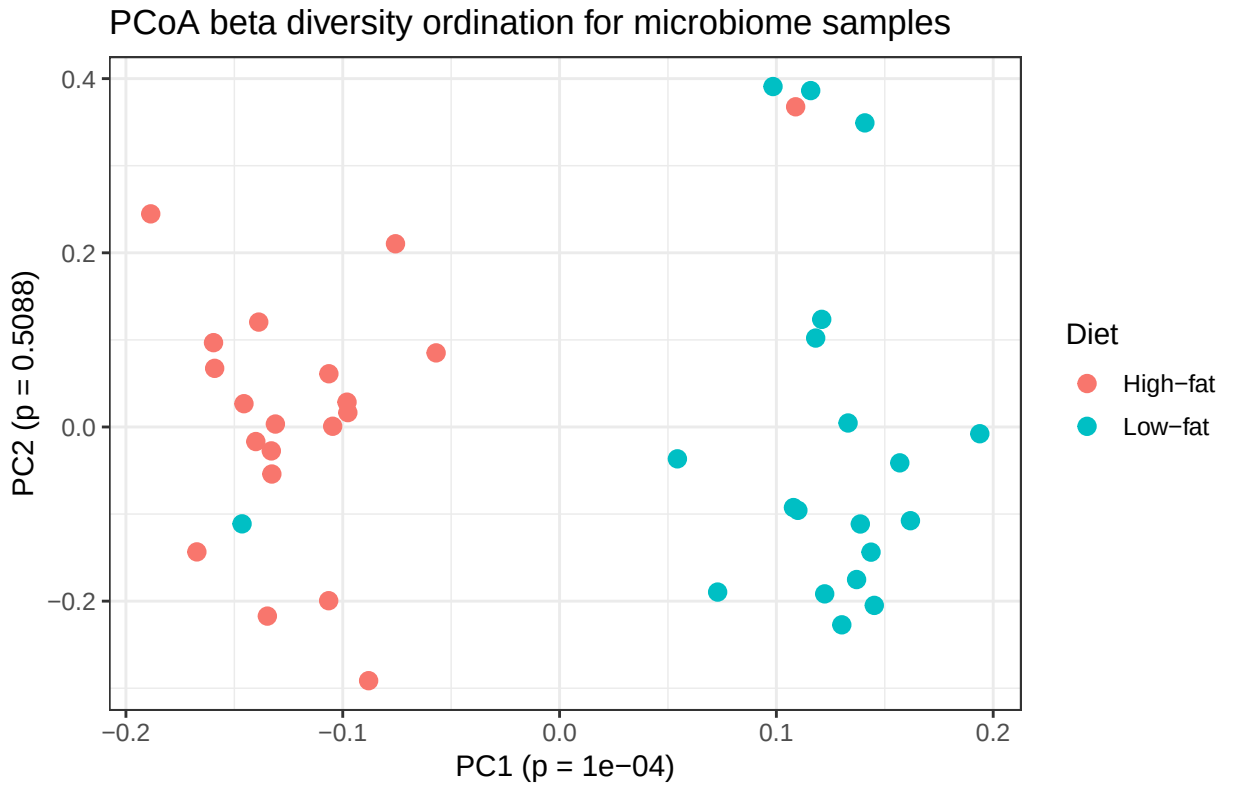
Beta diversity

Beta diversity measures the diffrence between samples. More information about beta diversity you can find from here.

We can see that data is clustered into 3 clusters. Bacterial composition of high-fat diet is different from bacterial composition of low-fat diet.

```
# Gets relative abundances
mae[[1]] <- transformSamples(mae[[1]], method = "relabundance")
# Relative abundance table
rel_abund_assay <- assays(mae[[1]])$relabundance

# Transposes it to get taxa to columns
rel_abund_assay <- t(rel_abund_assay)
# Calculates Bray-Curtis dissimilarities between samples. Because taxa is in columns,
# it is used to compare different samples.
bray_curtis_dis <- vegan::vegdist(rel_abund_assay, method = "bray")

# Does principal coordinate analysis
bray_curtis_pcoa <- ecodist::pco(bray_curtis_dis)

# Creates a data frame from principal coordinates and colData?
bray_curtis_pcoa_df <- data.frame(PC1 = bray_curtis_pcoa$vectors[,1],
                                  PC2 = bray_curtis_pcoa$vectors[,2],
                                  colData(mae[[1]]))
```

```
# Does the permanova analysis
p_values <- list()
for(pc in c("PC1", "PC2")){
  # Creates a formula from objects
  formula <- as.formula(paste0(pc, " ~ ", "Diet"))
  # Does the permanova analysis
  p_values[[pc]] <- vegan::adonis(formula, data = bray_curtis_pcoa_df,
                                  permutations = 9999, method = "euclidean"
                                  )$aov.tab["Diet", "Pr(>F)"]
}

# Creates a plot
plot <- ggplot(data = bray_curtis_pcoa_df, aes_string(x = "PC1", y = "PC2", color = "Diet")) +
  geom_point(size = 3) +
  labs(title = paste0("PCoA beta diversity ordination for microbiome samples"),
       x = paste0("PC1 (p = ", p_values[["PC1"]], ")"),
       y = paste0("PC2 (p = ", p_values[["PC2"]], ")")) +
  theme_bw(12)

print(plot)
```



Cross-correlation

Next we can do cross-correlation analysis. With it we can analyse, if one feature correlates with other feature. Here we analyse if individual bacteria genera correlate with concentrations of individual metabolites. “If this bacteria is present, is this metabolite’s concentration then low or high”?

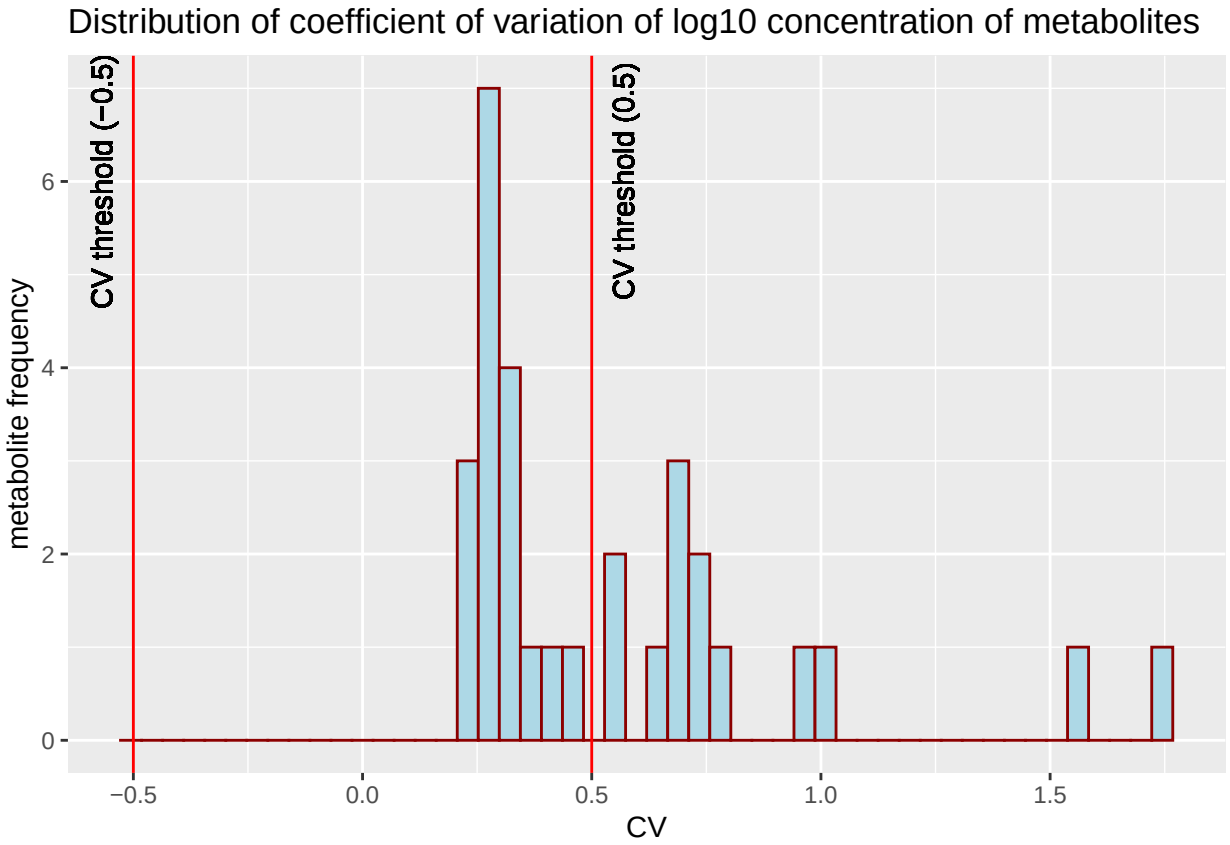
Because we are doing multiple testing, it needs to be taken into account by adjusting the p-values. Because

of the probability, sometimes unlikely thing happens if same thing is done multiple times. Same thing with p-values: if we test multiple times, it is likely that we get statistically significant result even though really there is no statistically significant difference.

Because of p-value adjustment strictens p-value threshold, individual differences need to be even more significant. That is why we usualy want to avoid doing "unnecessary" tests.

Here, we subset metabolites and take only those that vary the most. If their variation between samples is small, it is unlikely that we will find statistically significant differences between samples.

```
# Threshold: metabolites whose (cv > +threshold or cv < -threshold), will be included
cv_threshold <- 0.5
metabolite_trans <- "nmr"

# Get the data
metabolite_tse <- mae[[2]]

# Calculate coeffieicnt of variation of individual metabolites
df <- data.frame(cv = apply(assay(metabolite_tse, metabolite_trans), 1, function(x){sd(x)/mean(x)}))

# Plot them as a histogram, and show a line that is used as a threshold
plot <- ggplot(df, aes(x = cv)) +
  geom_histogram(bins = 50, color="darkred", fill="lightblue") +
  labs(x = "CV", y = "metabolite frequency",
       title = "Distribution of coefficient of variation of log10 concentration of metabolites") +
  geom_vline(xintercept = cv_threshold, color = "red") +
  geom_text(aes(cv_threshold, 6, label =
                  paste0("CV threshold (", cv_threshold, ")"), vjust = 2, angle=90)) +
  geom_vline(xintercept = -cv_threshold, color = "red") +
  geom_text(aes(-cv_threshold, 6, label =
                  paste0("CV threshold (", -cv_threshold, ")"), vjust = -1, angle=90))

print(plot)
```



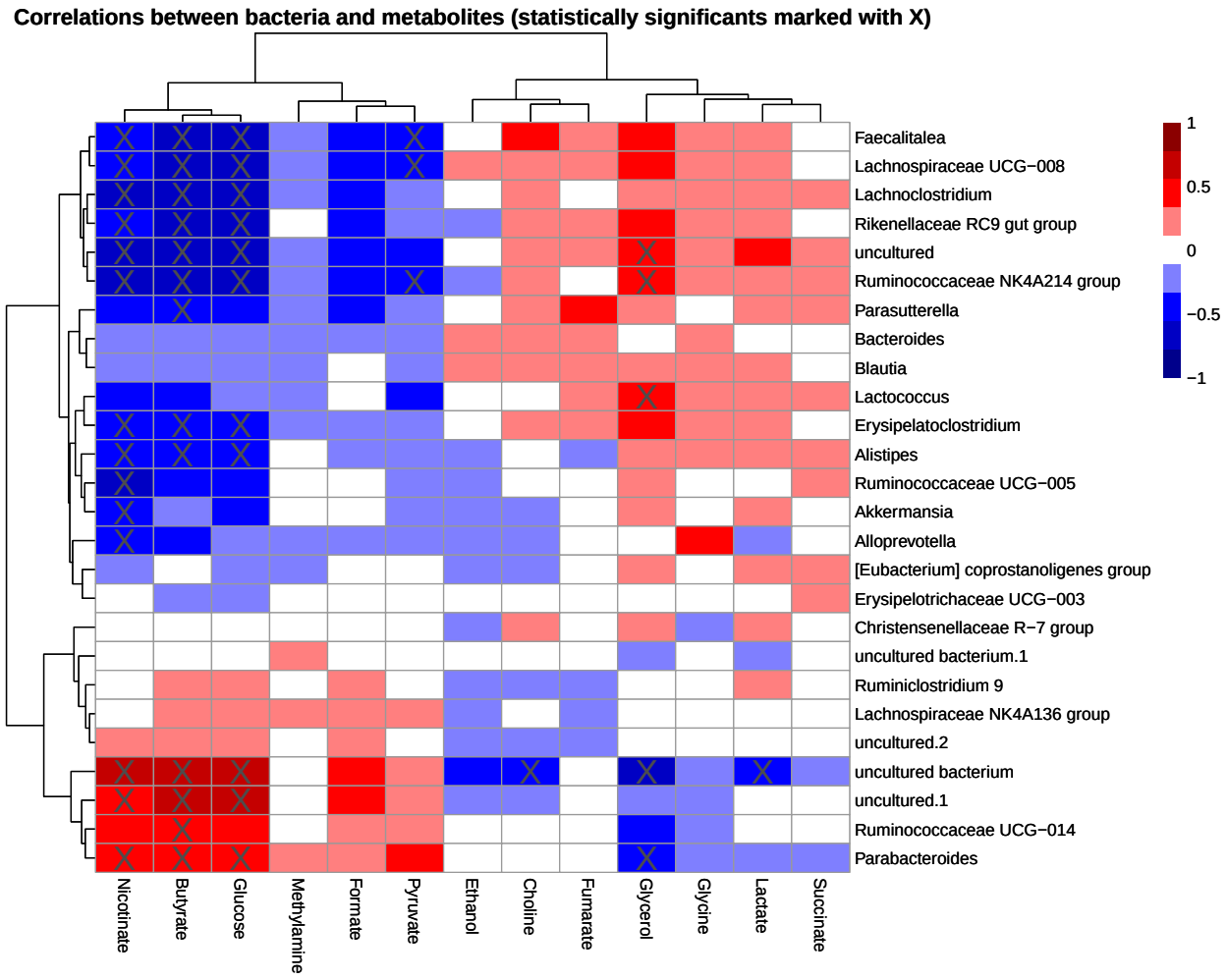

```
# Get those metabolites that are over threshold
metabolites_over_th <- rownames(df[df$cv > cv_threshold | df$cv < -cv_threshold, , drop = FALSE])
# Ignore those metabolites that do not have name / are NA
metabolites_over_th <- metabolites_over_th[!str_detect(metabolites_over_th, "NA")]
```

Next we can do the cross-correlation heatmap. From the heatmap we can see that certain bacteria correlate with certain metabolites statistically significantly.

For example, we can see that when the abundance of Ruminiclostridium 5 is high, the concentration of nicotinate tends to be relatively higher. Also we can see that when concentration butyrate is low, then abudnance of Lachnoclostridium tends to be higher or vice versa.

```
rank <- "Genus"
prevalence <- 0.2
detection <- 0.01
taxa_trans <-  "clr"
metabolite_trans <- "nmr"

# Get bacterial data
taxa_tse <- mae[[1]]
# Agglomerate at Genus level
taxa_tse <- agglomerateByRank(taxa_tse, rank = rank)
# Do CLR transformation
taxa_tse <- transformSamples(taxa_tse, method = "clr", pseudocount = 1)

# Get metabolite data
metabolite_tse <- mae[[2]]
```

```r
# Subset metabolite data
metabolite_tse <- metabolite_tse[metabolites_over_th, ]

# Subset bactrial data by its prevalence. Bacteria whose prevalences are over threshold are included
taxa_tse <- subsetByPrevalentTaxa(taxa_tse, prevalence = prevalence, detection = detection)

# Define data sets to cross-correlate
x <- t(assay(taxa_tse, taxa_trans))
y <- t(assay(metabolite_tse, "nmr"))
# If there are duplicated taxa names, makes them unique
colnames(x) <- str_remove(colnames(x), paste0(rank, ":"))
colnames(x) <- make.unique(colnames(x))

# Cross correlate data sets
correlations <- microbiome::associate(x, y, method = "spearman", mode = "matrix")

# For plotting purpose, convert p-values, under 0.05 are marked with "X"
p_threshold <- 0.05
p_values <- ifelse(correlations$p.adj<p_threshold, "X", "")

# Scale colors
breaks <- seq(-ceiling(max(abs(correlations$cor))), ceiling(max(abs(correlations$cor))),
              length.out = ifelse( max(abs(correlations$cor))>5,
                                   2*ceiling(max(abs(correlations$cor))), 10 ) )
colors <- colorRampPalette(c("darkblue", "blue", "white", "red", "darkred"))(length(breaks)-1)

# Create a heatmap
print(pheatmap(correlations$cor, display_numbers = p_values,
main = paste0("Correlations between bacteria and metabolites (statistically significants marked with X)"
         fontsize = 10,
         breaks = breaks,
         color = colors,
fontsize_number = 20) )
```




```
sessionInfo()
```

```
## R version 4.1.0 alpha (2021-04-26 r80229)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/local/lib/R/lib/libRblas.so
## LAPACK: /usr/local/lib/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=fi_FI.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=fi_FI.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=fi_FI.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=fi_FI.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
```

```
## other attached packages:
##  [1] stringr_1.4.0                   pheatmap_1.0.12
##  [3] BiocManager_1.30.16             microbiomeDataSets_1.1.5
##  [5] MultiAssayExperiment_1.19.10    microbiome_1.15.0
##  [7] ggplot2_3.3.5                   phyloseq_1.37.0
##  [9] mia_1.1.13                      TreeSummarizedExperiment_2.1.4
## [11] Biostrings_2.61.2               XVector_0.33.0
## [13] SingleCellExperiment_1.15.2     SummarizedExperiment_1.23.4
## [15] Biobase_2.53.0                  GenomicRanges_1.45.0
## [17] GenomeInfoDb_1.29.8             IRanges_2.27.2
## [19] S4Vectors_0.31.3                BiocGenerics_0.39.2
## [21] MatrixGenerics_1.5.4            matrixStats_0.60.1
##
## loaded via a namespace (and not attached):
##   [1] AnnotationHub_3.1.5           BiocFileCache_2.1.1
##   [3] plyr_1.8.6                    igraph_1.2.6
##   [5] lazyeval_0.2.2                splines_4.1.0
##   [7] BiocParallel_1.27.5           scater_1.21.3
##   [9] digest_0.6.27                 foreach_1.5.1
##  [11] htmltools_0.5.2               viridis_0.6.1
##  [13] fansi_0.5.0                   magrittr_2.0.1
##  [15] memoise_2.0.0                 ScaledMatrix_1.1.0
##  [17] cluster_2.1.2                 DECIPHER_2.21.0
##  [19] colorspace_2.0-2              blob_1.2.2
##  [21] rappdirs_0.3.3                ggrepel_0.9.1
##  [23] xfun_0.25                     dplyr_1.0.7
##  [25] crayon_1.4.1                  RCurl_1.98-1.4
##  [27] jsonlite_1.7.2                survival_3.2-13
##  [29] iterators_1.0.13              ape_5.5
##  [31] glue_1.4.2                    gtable_0.3.0
##  [33] zlibbioc_1.39.0               DelayedArray_0.19.2
##  [35] BiocSingular_1.9.1            Rhdf5lib_1.15.2
##  [37] scales_1.1.1                  DBI_1.1.1
##  [39] Rcpp_1.0.7                    viridisLite_0.4.0
##  [41] xtable_1.8-4                  decontam_1.13.0
##  [43] tidytree_0.3.5                bit_4.0.4
##  [45] rsvd_1.0.5                    ecodist_2.0.7
##  [47] httr_1.4.2                    RColorBrewer_1.1-2
##  [49] ellipsis_0.3.2                farver_2.1.0
##  [51] pkgconfig_2.0.3               scuttle_1.3.1
##  [53] dbplyr_2.1.1                  utf8_1.2.2
##  [55] labeling_0.4.2                tidyselect_1.1.1
##  [57] rlang_0.4.11                  reshape2_1.4.4
##  [59] later_1.3.0                   AnnotationDbi_1.55.1
##  [61] munsell_0.5.0                 BiocVersion_3.14.0
##  [63] tools_4.1.0                   cachem_1.0.6
##  [65] DirichletMultinomial_1.35.0   generics_0.1.0
##  [67] RSQLite_2.2.8                 ExperimentHub_2.1.4
##  [69] ade4_1.7-17                   evaluate_0.14
##  [71] biomformat_1.21.0             fastmap_1.1.0
##  [73] yaml_2.2.1                    knitr_1.34
##  [75] bit64_4.0.5                   purrr_0.3.4
##  [77] KEGGREST_1.33.0               nlme_3.1-153
##  [79] sparseMatrixStats_1.5.3       mime_0.11
```

```
##  [81] compiler_4.1.0               png_0.1-7
##  [83] beeswarm_0.4.0               filelock_1.0.2
##  [85] curl_4.3.2                   interactiveDisplayBase_1.31.2
##  [87] treeio_1.17.2                tibble_3.1.4
##  [89] stringi_1.7.4                highr_0.9
##  [91] lattice_0.20-44              Matrix_1.3-4
##  [93] vegan_2.5-7                  permute_0.9-5
##  [95] multtest_2.49.0              vctrs_0.3.8
##  [97] pillar_1.6.2                 lifecycle_1.0.0
##  [99] rhdf5filters_1.5.0           BiocNeighbors_1.11.0
## [101] data.table_1.14.0            bitops_1.0-7
## [103] irlba_2.3.3                  httpuv_1.6.2
## [105] R6_2.5.1                     promises_1.2.0.1
## [107] gridExtra_2.3                vipor_0.4.5
## [109] codetools_0.2-18             MASS_7.3-54
## [111] assertthat_0.2.1             rhdf5_2.37.0
## [113] withr_2.4.2                  GenomeInfoDbData_1.2.6
## [115] mgcv_1.8-36                  parallel_4.1.0
## [117] grid_4.1.0                   beachmat_2.9.1
## [119] tidyr_1.1.3                  rmarkdown_2.10
## [121] DelayedMatrixStats_1.15.4    Rtsne_0.15
## [123] shiny_1.6.0                  ggbeeswarm_0.6.0
```