Introduction to multi-omics data analysis

University of Turku

2022-01-11

Contents

Chapter 1

Overview

Welcome to the multi-omics data analysis workshop

Figure source: Moreno-Indias *et al.* (2021) Statistical and Machine Learning Techniques in Human Microbiome Studies: Contemporary Challenges and Solutions. Frontiers in Microbiology 12:11.

1.1 Introduction

This course is based on *miaverse* (mia = **MI**crobiome **A**nalysis) is an R/Bioconductor framework for microbiome data science. It extends another popular framework, phyloseq.

The miaverse consists of an efficient data structure, an associated package ecosystem, demonstration data sets, and open documentation. These are explained in more detail in the online book Orchestrating Microbiome Analysis.

This workshop material walks you through example workflows for multi-omics data analysis covering data access, exploration, analysis, visualization and reproducible reporting. **You can run the workflow by simply copy-pasting the examples.** For advanced material, you can test and modify further examples from the OMA book, or try to apply the techniques to your own data.

1.2 Learning goals [TO DO]

This workshop provides an overview of bioinformatics tools for multi-omics studies, ranging from data preprocessing to statistical analysis and reproducible reporting.

Target audience Advanced students and applied researchers who wish to develop their skills in microbial community analysis. [TO DO]

Venue [TO DO]

1.3 Acknowledgments

Citation “Introduction to microbiome data science (2021). URL: https://microbiome.github.io”.

Borman et al. (2022)

We thank Felix Ernst, Sudarshan Shetty, and other miaverse developers who have contributed open resources that supported the development of the training material.

Contact Leo Lahti, University of Turku, Finland

Source code

The source code of this repository is fully reproducible and contains the Rmd files with executable code. All files can be rendered at one go by running the file main.R. You can check the file for details on how to clone the repository and convert it into a gitbook, although this is not necessary for the training.

- Landing page (html): workshop teaching material
- Source code (github): workshop teaching material

Chapter 2

Program

The workshop takes place on the 13th and 14th of January from 9am – 5pm (CET). Short breaks will be scheduled between sessions.

The practical sessions consists of a set of example multi-omics analysis workflows. It is assumed that you have already installed the required software. Do not hesitate to ask support from the course assistants.

2.1 Day 1

Lectures

- 9:15-10:00 - Welcome and introduction - Leo Lahti, Associate professor (UTU)
- 10:00-10:15 - Break
- 10:15-11:00 - Metagenomics - Katariina Pärnänen, Postdoctoral researcher (UTU)
- 11:00-11:15 - Break
- 11:15-12:00 - Metabolomics - Pande Putu Erawijantari, Postdoctoral researcher (UTU)
- 12:00-12:15 - Break
- 12:15-13:00 - Multiomics - Leo Lahti, Associate professor (UTU)
- 13:00-14:15 - Lunch break

Practical

- 14:15-17:00 - Tuomas Borman and Chouaib Benchraka, Research assistants (UTU)
- Topics:
  - Data import and data structures
  - Microbiome data exploration
  - Visualization

2.2 Day 2

Lectures

- 9:15-11:00 (including a short break) - Unsupervised and supervised machine learning - Matti Ruuskanen, Postdoctoral researcher (UTU)
- 11:00-11:15 - Break
- 11:15-12:00 - Individual-based modeling - Gergely Boza, Research fellow (CER)
- 12:00-12:15 - Break
- 12:15-13:00 - Data integration - Leo Lahti, Associate professor (UTU)
- 13:00-14:15 - Lunch break

Practical

- 14:15-17:00 - Tuomas Borman, Matti Ruuskanen and Chouaib Benchraka (UTU)
- Topics:
  - Unsupervised learning: Beta-diversity and biclustering
  - Supervised learning: Regression and classification with random forests
  - Validation and interpretation of black box models

Chapter 3

Getting started

3.1 Checklist (before the workshop)

Install the following software in advance in order to avoid unnecessary delays and leaving more time for the workshop contents.

- R (version >4.1.0)
- RStudio; choose "Rstudio Desktop" to download the latest version. Optional but preferred. For further details, check the Rstudio home page.
- Install and load the required R packages

3.2 Support and resources

For online support on installation and other matters, you can join us at:

- Users: miaverse Gitter channel
- Developers: Bioconductor Slack #microbiomeexperiment channel (ask for an invitation)

3.3 Installing and loading the required R packages

This section shows how to install and load all required packages into the R session. Only uninstalled packages are installed.

```r
# List of packages that we need from cran and bioc
cran_pkg <- c("BiocManager", "bookdown", "dplyr", "ecodist", "ggplot2",
              "gridExtra", "kableExtra", "knitr", "scales", "vegan", "caret",
              "ranger", "stringr", "pheatmap", "patchwork", "pdp", "biclust")
bioc_pkg <- c("mia", "miaViz", "microbiomeDataSets")

# Gets those packages that are already installed
```

```r
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)
```

```r
# 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)
}
```

```r
# 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.

```r
# Reorders bioc packages, so that mia and miaViz are first
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.
loaded <- sapply(c(bioc_pkg, cran_pkg), require, character.only = TRUE)
as.data.frame(loaded)
```

```
##                     loaded
## mia                   TRUE
## miaViz                TRUE
## microbiomeDataSets    TRUE
## BiocManager           TRUE
## bookdown              TRUE
## dplyr                 TRUE
## ecodist               TRUE
## ggplot2               TRUE
## gridExtra             TRUE
## kableExtra            TRUE
## knitr                 TRUE
## scales                TRUE
## vegan                 TRUE
## caret                 TRUE
## ranger                TRUE
## stringr               TRUE
## pheatmap              TRUE
## patchwork             TRUE
## pdp                   TRUE
## biclust               TRUE
```

Chapter 4

Data

This section demonstrates how to import data in R.

4.1 Data structure

Such analysis using the miaverse framework, are based upon core data structures including SingleCellExperiment (SCE), SummarizedExperiment (SE), TreeSummarizedExperiment (TreeSE) and MultiAssayExperiment (MAE) (resources).

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.

4.2 Example data

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.

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.

4.3 Importing data in R

```
library(stringr)

# Load the data
```

```
mae <- microbiomeDataSets::HintikkaXOData()

# 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 data to flat files
```

Chapter 5

Microbiome data exploration

Now we have loaded the data set into R. Next, let us walk through some basic operations for data exploration to confirm that the data has all the necessary components.

5.1 Data structure

Let us now investigate how taxonomic profiling data is organized in R.

Dimensionality tells us how many taxa and samples the data contains. As we can see, there are 12613 taxa and 40 samples.

```
# mae[[1]]: indexing/retrieving the taxonomic data experiment
dim(mae[[1]])
```

```
## [1] 12613    40
```

The `rowData` slot contains a taxonomic table. This includes taxonomic information for each of the 12613 entries. With the `head()` command, we can print just the beginning of the table.

The `rowData` seems to contain information from 7 different taxonomy classes.

```
knitr::kable(head(rowData(mae[[1]]))) %>%
  kableExtra::kable_styling("striped",
                            latex_options="scale_down") %>%
  kableExtra::scroll_box(width = "100%")
```

The colData slot contains sample metadata. It contains information for all 40 samples. However, here only the 6 first samples are shown as we use the `head()` command. There are 6 columns, that contain information, e.g., about patients' status, and cohort.

	Phylum	Class	Order	Family	Genus	Species	OTU
GAYR01026362.62.2014	Proteobacteria	Alphaproteobacteria	Rickettsiales	Mitochondria	Solanum melongena (eggplant)	Solanum melongena (eggplant)	GAYR01026362.62.2014
CVJT01000011.50.2173	Firmicutes	Bacilli	Bacillales	Staphylococcaceae	Staphylococcus	Staphylococcus aureus	CVJT01000011.50.2173
KF625183.1.1786	Proteobacteria	Gammaproteobacteria	Enterobacteriales	Enterobacteriaceae	Klebsiella	Klebsiella oxytoca	KF625183.1.1786
AYSG01000002.292.2076	Firmicutes	Bacilli	Lactobacillales	Streptococcaceae	Streptococcus	Streptococcus thermophilus TH1435	AYSG01000002.292.2076
CCPS01000022.154.1916	Proteobacteria	Gammaproteobacteria	Enterobacteriales	Enterobacteriaceae	Escherichia-Shigella	Escherichia coli	CCPS01000022.154.1916
KJ923794.1.1762	Firmicutes	Bacilli	Bacillales	Staphylococcaceae	Staphylococcus	Staphylococcus aureus	KJ923794.1.1762

	Sample	Rat	Site	Diet	Fat	XOS
C1	C1	1	Cecum	High-fat	High	0
C2	C2	2	Cecum	High-fat	High	0
C3	C3	3	Cecum	High-fat	High	0
C4	C4	4	Cecum	High-fat	High	0
C5	C5	5	Cecum	High-fat	High	0
C6	C6	6	Cecum	High-fat	High	0

```
# 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")

knitr::kable(head(colData(mae))) %>%
  kableExtra::kable_styling("striped",
                            latex_options="scale_down") %>%
  kableExtra::scroll_box(width = "100%")
```

From here, we can draw summaries of the sample (column) data, for instance to see what is the diet distribution.

The command `colData(mae)$Diet` fetches the data from the column, and `table()` creates a table that shows how many times each class is present, and `sort()` sorts the table to ascending order.

There are 20 samples from mice having High-fat, and 20 Low-fat.

```
sort(table(colData(mae)$Diet))
```

```
##
## High-fat  Low-fat
##       20       20
```

5.1.1 Transformations

Microbial abundances are typically 'compositional' (relative) in the current microbiome profiling data sets. This is due to technical aspects of the data generation process (see e.g. Gloor et al., 2017).

The next example calculates relative abundances as these are usually easier to interpret than plain counts. For some statistical models we need to transform the data into other formats as explained in above link (and as we will see later).

```
# Calculates relative abundances, and stores the table to assays
mae[[1]] <- transformCounts(mae[[1]], method = "relabundance")
```

A variety of standard transformations for microbiome data are available through mia R package.

	Phylum	Class	Order	Family	Genus	Species	OTU
Proteobacteria	Proteobacteria	NA	NA	NA	NA	NA	GAYR01026362.62.2014
Firmicutes	Firmicutes	NA	NA	NA	NA	NA	CVJT01000011.50.2173
Cyanobacteria	Cyanobacteria	NA	NA	NA	NA	NA	GEMN01027092.33.1623
Tenericutes	Tenericutes	NA	NA	NA	NA	NA	AM277369.1.1548
Deferribacteres	Deferribacteres	NA	NA	NA	NA	NA	AYGZ01000001.327.1863
Actinobacteria	Actinobacteria	NA	NA	NA	NA	NA	JGZF01000005.1.1534

5.1.2 Aggregation

Microbial species can be called at multiple taxonomic resolutions. We can easily agglomerate the data based on taxonomic ranks. Here, we agglomerate the data at Phylum level.

```r
se_phylum <- agglomerateByRank(mae[[1]], rank = "Phylum")

# Show dimensionality
dim(se_phylum)
```

```
## [1] 13 40
```

Now there are 13 taxa and 40 samples, meaning that there are 13 different Phylum level taxonomic groups. Looking at the `rowData` after agglomeration shows all Firmicutes are combined together, and all lower rank information is lost.

From the assay we can see that all abundances of taxa that belong to Firmicutes are summed up.

```r
knitr::kable(head(rowData(se_phylum))) %>%
  kableExtra::kable_styling("striped",
                            latex_options="scale_down") %>%
  kableExtra::scroll_box(width = "100%")
```

If you are sharp, you have by now noticed that all the aggregated values in the above example are NA's (missing data). This is because the agglomeration is missing abundances for certain taxa, and in that case the sum is not defined by default (`na.rm = FALSE`). We can ignore the missing values in summing up the data by setting `na.rm = TRUE`; then the taxa that do not have information in specified level will be removed. Those taxa that do not have information in specified level are agglomerated at lowest possible level that is left after agglomeration.

```r
temp <- rowData(agglomerateByRank(mae[[1]], rank = "Genus"))

# Prints those taxa that do not have information at the Genus level (NA)
knitr::kable(head(temp[which(is.na(temp$Genus)),])) %>%
  kableExtra::kable_styling("striped",
                            latex_options="scale_down") %>%
  kableExtra::scroll_box(width = "100%")
```

Here agglomeration is done similarly, but na.rm = TRUE

	Phylum	Class	Order	Family	Genus	Species	OTU
Family:uncultured	Proteobacteria	Alphaproteobacteria	Rhodospirillales	uncultured	NA	NA	JRJTB:01000:00983
Family:Ruminococcaceae	Firmicutes	Clostridia	Clostridiales	Ruminococcaceae	NA	NA	JRJTB:00751:00256
Order:Clostridiales	Firmicutes	Clostridia	Clostridiales	NA	NA	NA	JRJTB:03059:01977
Family:Lachnospiraceae	Firmicutes	Clostridia	Clostridiales	Lachnospiraceae	NA	NA	JRJTB:00738:02832
Family:Peptostreptococcaceae	Firmicutes	Clostridia	Clostridiales	Peptostreptococcaceae	NA	NA	JRJTB:01731:00274
Family:Pasteurellaceae	Proteobacteria	Gammaproteobacteria	Pasteurellales	Pasteurellaceae	NA	NA	JRJTB:01960:01703

```r
temp2 <- rowData(agglomerateByRank(mae[[1]], rank = "Genus", na.rm = TRUE))

print(paste0("Agglomeration with na.rm = FALSE: ", dim(temp)[1], " taxa."))
```

```
## [1] "Agglomeration with na.rm = FALSE: 277 taxa."
```

```r
print(paste0("Agglomeration with na.rm = TRUE: ", dim(temp2)[1], " taxa."))
```

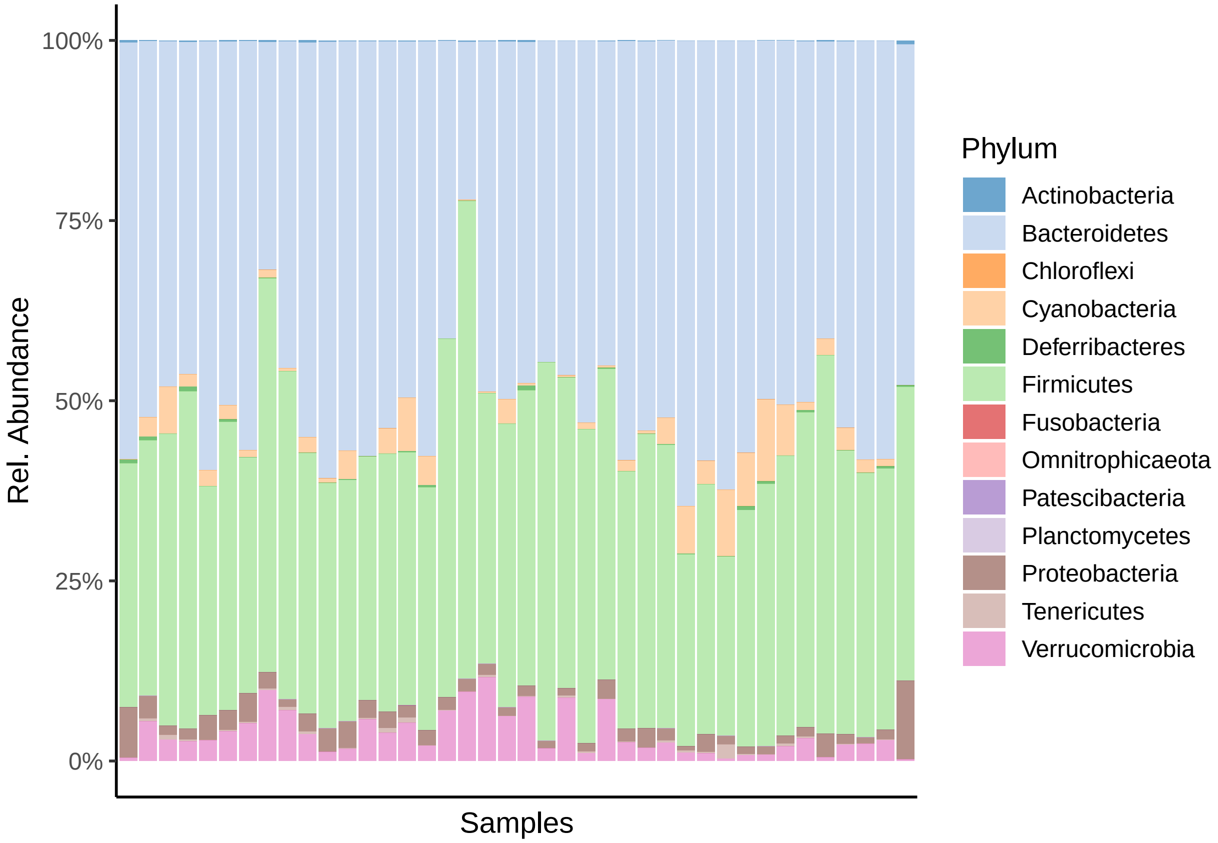
```
## [1] "Agglomeration with na.rm = TRUE: 262 taxa."
```

The mia package contains further examples on various data agglomeration and splitting options.

5.2 Visualization

The miaViz package facilitates data visualization. Let us plot the Phylum level abundances.

```r
# Here we specify "relabundance" to be abundance table that we use for plotting.
# Note that we can use agglomerated or non-agglomerated mae[[1]] as an input, because
# the function agglomeration is built-in option.

# Legend does not fit into picture, so its height is reduced.
plot_abundance <- plotAbundance(mae[[1]], abund_values="relabundance", rank = "Phylum") +
  theme(legend.key.height = unit(0.5, "cm")) +
  scale_y_continuous(label = scales::percent)

plot_abundance
```

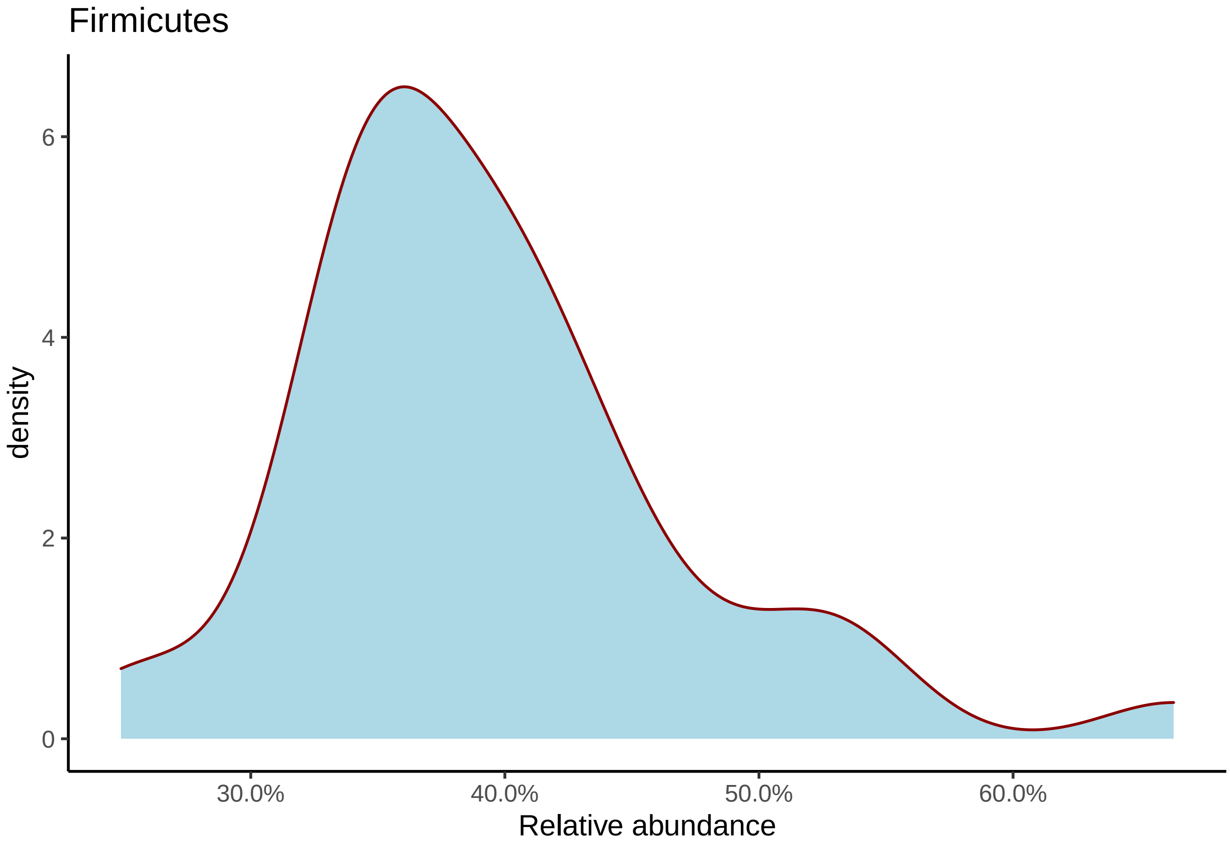


Density plot shows the overall abundance distribution for a given taxonomic group. Let us check the relative abundance of Firmicutes across the sample collection. The density plot is a smoothened version of a standard histogram.

The plot shows peak abundances around 30 %.

```
# Subset data by taking only Firmicutes
se_firmicutes <- se_phylum["Firmicutes"]

# Gets the abundance table
abundance_firmicutes <- assay(se_firmicutes, "relabundance")

# Creates a data frame object, where first column includes abundances
firmicutes_abund_df <- as.data.frame(t(abundance_firmicutes))
# Rename the first and only column
colnames(firmicutes_abund_df) <- "abund"

# Creates a plot. Parameters inside feom_density are optional. With
# geom_density(bw=1000), it is possible to adjust bandwidth.
firmicutes_abund_plot <- ggplot(firmicutes_abund_df, aes(x = abund)) +
  geom_density(color="darkred", fill="lightblue") +
  labs(x = "Relative abundance", title = "Firmicutes") +
  theme_classic() + # Changes the background
  scale_x_continuous(label = scales::percent)
```

```
firmicutes_abund_plot
```



For more visualization options and examples, see the miaViz vignette.

Chapter 6

Beta diversity

Beta diversity is another name for sample dissimilarity. It quantifies differences in the overall taxonomic composition between two samples.

Common indices include Bray-Curtis, Unifrac, Jaccard index, and the Aitchison distance. For more background information and examples, you can check the dedicated section in online book.

6.1 Examples of PCoA with different settings

Beta diversity estimation generates a (dis)similarity matrix that contains for each sample (rows) the dissimilarity to any other sample (columns).

This complex set of pairwise relations can be visualized in informative ways, and even coupled with other explanatory variables. As a first step, we compress the information to a lower dimensionality, or fewer principal components, and then visualize sample similarity based on that using ordination techniques, such as Principal Coordinate Analysis (PCoA). PCoA is a non-linear dimension reduction technique, and with Euclidean distances it is is identical to the linear PCA (except for potential scaling).

We typically retain just the two (or three) most informative top components, and ignore the other information. Each sample has a score on each of these components, and each component measures the variation across a set of correlated taxa. The top components are then easily visualized on a two (or three) dimensional display.

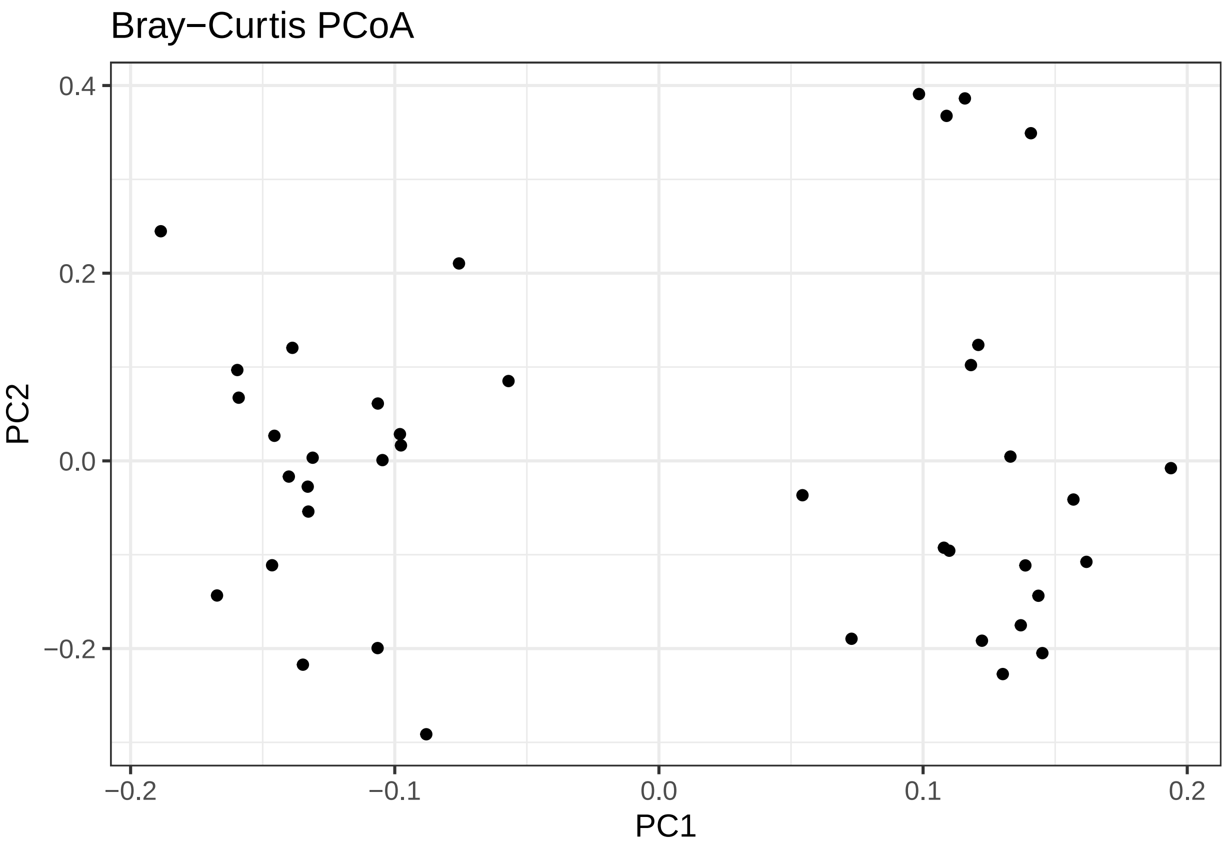
Let us next look at some concrete examples.

6.1.1 PCoA for ASV-level data with Bray-Curtis

Let us start with PCoA based on a Bray-Curtis dissimilarity matrix.

```
# Pick the relative abundance table
rel_abund_assay <- assays(mae[[1]])$relabundance

# Calculates Bray-Curtis distances between samples. Because taxa is in
# columns, it is used to compare different samples. We transpose the
# assay to get taxa to columns
bray_curtis_dist <- vegan::vegdist(t(rel_abund_assay), method = "bray")
```

```
# PCoA
bray_curtis_pcoa <- ecodist::pco(bray_curtis_dist)

# All components could be found here:
# bray_curtis_pcoa$vectors
# But we only need the first two to demonstrate what we can do:
bray_curtis_pcoa_df <- data.frame(pcoa1 = bray_curtis_pcoa$vectors[,1],
                                  pcoa2 = bray_curtis_pcoa$vectors[,2])

# Create a plot
bray_curtis_plot <- ggplot(data = bray_curtis_pcoa_df, aes(x=pcoa1, y=pcoa2)) +
  geom_point() +
  labs(x = "PC1",
       y = "PC2",
       title = "Bray-Curtis PCoA") +
  theme_bw(12) # makes titles smaller

bray_curtis_plot
```

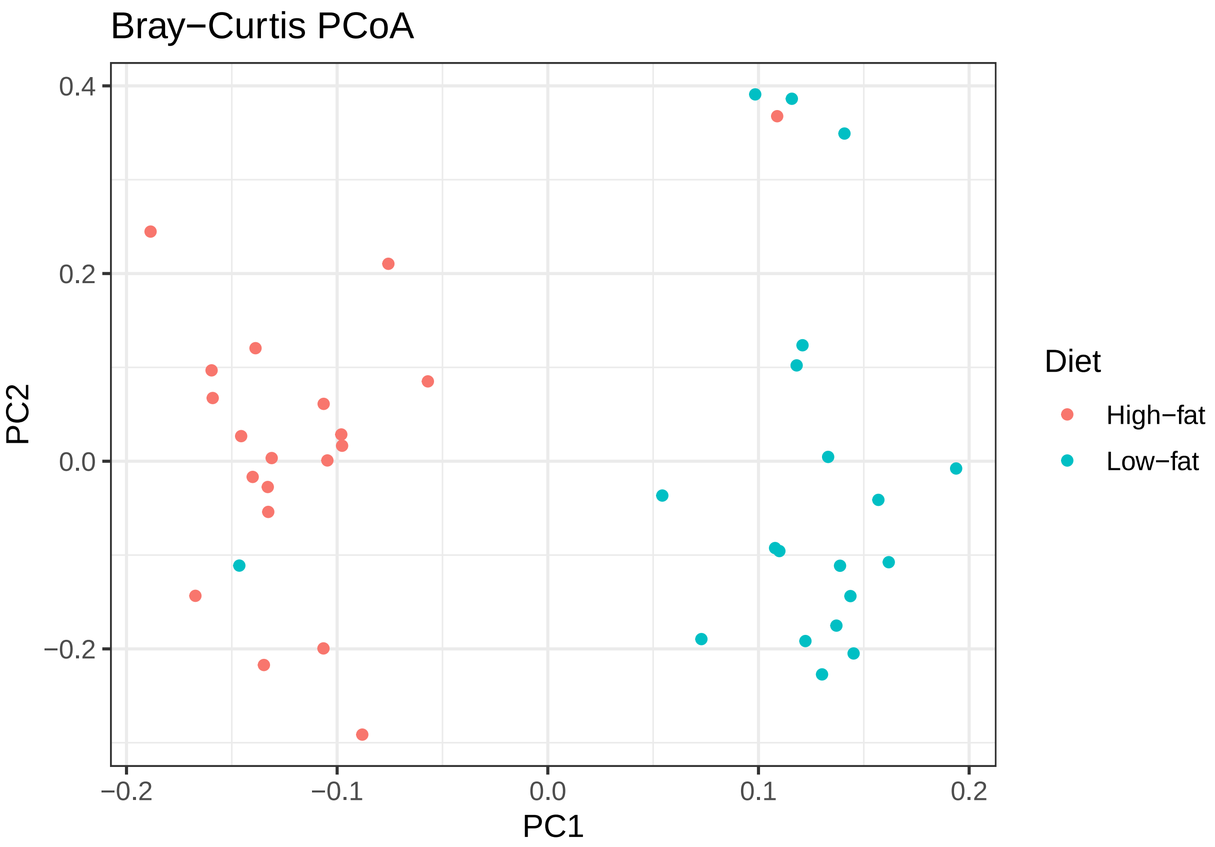


6.2 Highlighting external variables

We can map other variables on the same plot for example by coloring the points accordingly.

The following is an example with a discrete grouping variable (Diet) shown with colors:

```
# Add diet information to data.frame
bray_curtis_pcoa_df$Diet <- colData(mae)$Diet

# Creates a plot
plot <- ggplot(data = bray_curtis_pcoa_df, aes_string(x = "pcoa1", y = "pcoa2", color = "Diet")) +
  geom_point() +
  labs(x = "PC1",
       y = "PC2",
       title = "Bray-Curtis PCoA") +
  theme_bw(12)

plot
```



6.3 Estimating associations with an external variable

Next to visualizing whether any variable is associated with differences between samples, we can also quantify the strength of the association between community composition (beta diversity) and external factors.

The standard way to do this is to perform a so-called permutational multivariate analysis of variance (PERMANOVA). This method takes as input the abundance table, which measure of distance you want to base the test on and a formula that tells the model how you think the variables are associated with each other.

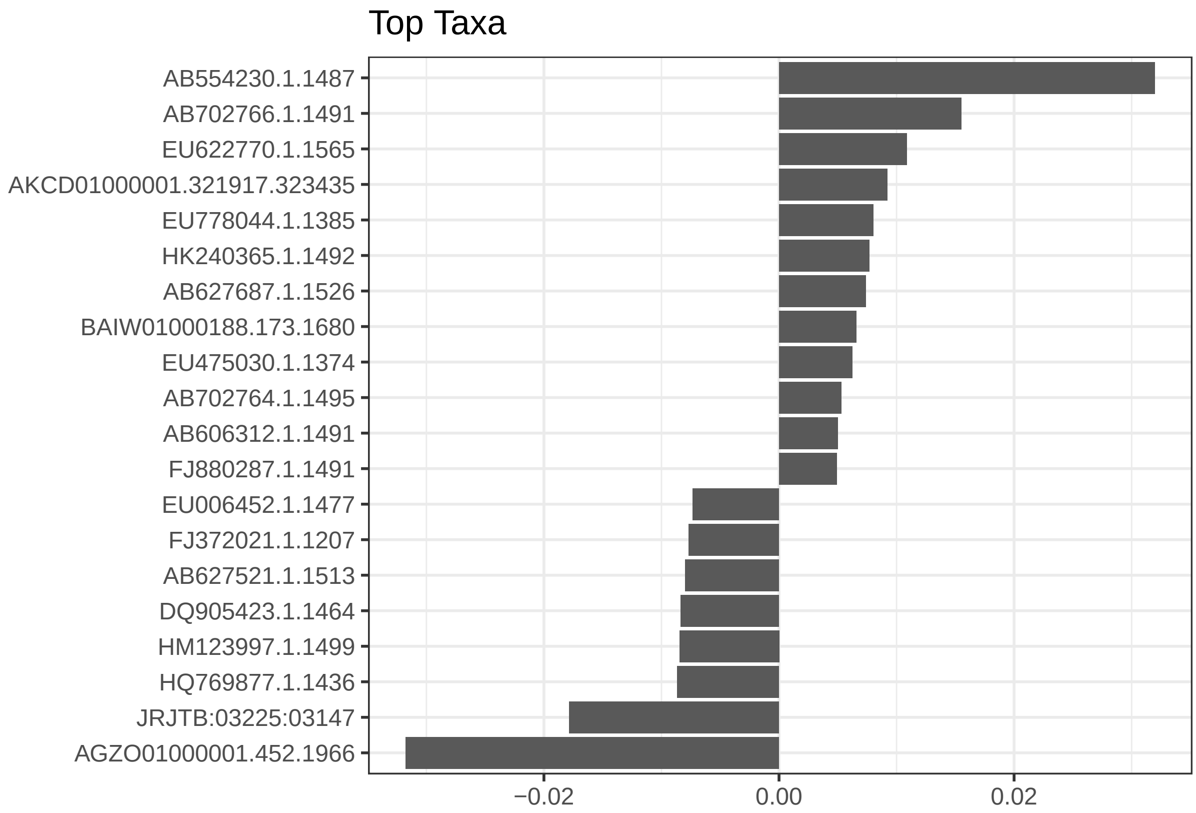
```
# First we get the relative abundance table
rel_abund_assay <- assays(mae[[1]])$relabundance

# again transpose it to get taxa to columns
rel_abund_assay <- t(rel_abund_assay)

# then we can perform the method
permanova_diet <- vegan::adonis(rel_abund_assay ~ Diet,
                                  data = colData(mae),
                                  permutations = 99)

# we can obtain a the p value for our predictor:
print(paste0("The test result p-value: ",
              as.data.frame(permanova_diet$aov.tab)["Diet", "Pr(>F)"]))
```

```
## [1] "The test result p-value: 0.01"
```

The diet variable is significantly associated with microbiota composition (p-value is less than 0.05).

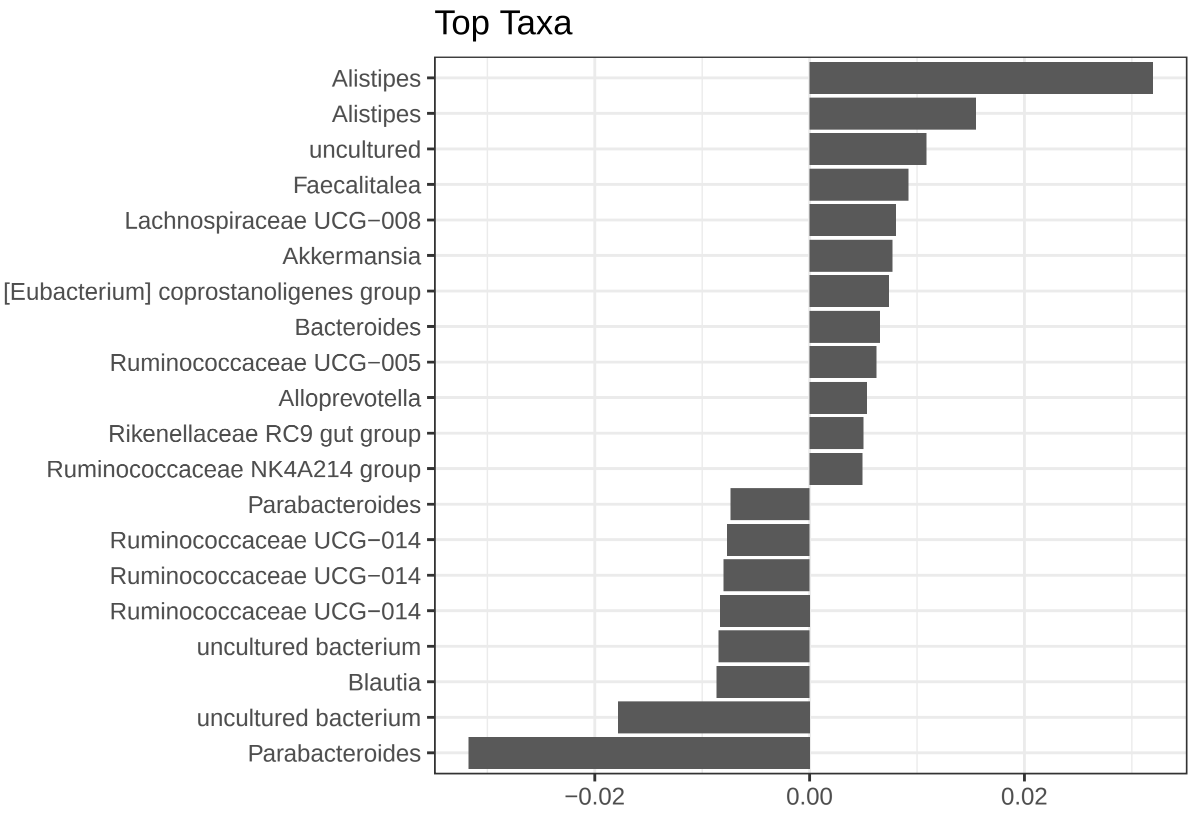
We can visualize those taxa whose abundances drive the differences between diets. We first need to extract the model coefficients of taxa:

```
# Gets the coefficients
coef <- coefficients(permanova_diet)["Diet1",]

# Gets the highest coefficients
top.coef <- sort(head(coef[rev(order(abs(coef)))],20))

# Plots the coefficients
top_taxa_coeffient_plot <- ggplot(data.frame(x = top.coef,
                                             y = factor(names(top.coef),
                         unique(names(top.coef)))),
                                  aes(x = x, y = y)) +
  geom_bar(stat="identity") +
  labs(x="", y="", title="Top Taxa") +
  theme_bw()

top_taxa_coeffient_plot
```



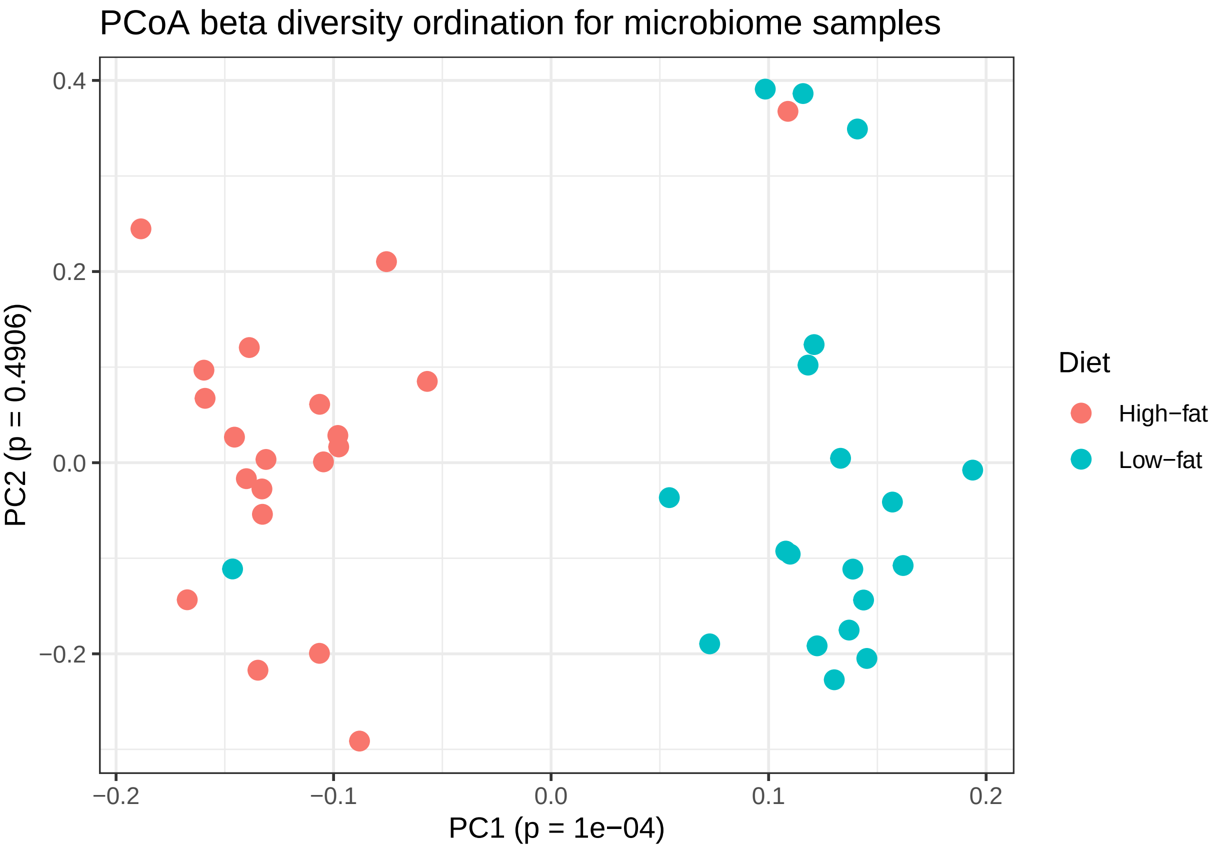
The above plot shows taxa as code names, and it is hard to tell which bacterial groups they represent. However, it is easy to add human readable names. We can fetch those from our rowData. Here we use Genus level names:

```r
# Gets corresponding Genus level names and stores them to top.coef
names <- rowData(mae[[1]])[names(top.coef), ][,"Genus"]

# Adds new labels to the plot
top_taxa_coeffient_plot <- top_taxa_coeffient_plot +
  scale_y_discrete(labels = names) # Adds new labels
top_taxa_coeffient_plot
```



The same test can be conducted using the ordination from PCoA as follows:

```
bray_curtis_pcoa_df$Diet <- colData(mae)$Diet
p_values <- list()
for(pc in c("pcoa1", "pcoa2")){
  # 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 = "pcoa1", y = "pcoa2", color = "Diet")) +
  geom_point(size = 3) +
  labs(title = paste0("PCoA beta diversity ordination for microbiome samples"), x = paste0("PC1 (p = ", p_val
  theme_bw()

plot
```



There are many alternative and complementary methods for analysing community composition. For more examples, see a dedicated section on beta diversity in the online book.

6.4 Community typing

A dedicated section presenting examples on community typing is in the online book.

Chapter 7

Unsupervised learning

Unsupervised learning is a part of machine learning where we try to find information from unknown data. It is also called data mining. Usually this means finding of clusters, for instance. Cluster refers to group of samples/features that are similar between each other. For example, based on clinical data we can try to find patient groups that have similar response to used drug.

7.1 Biclustering

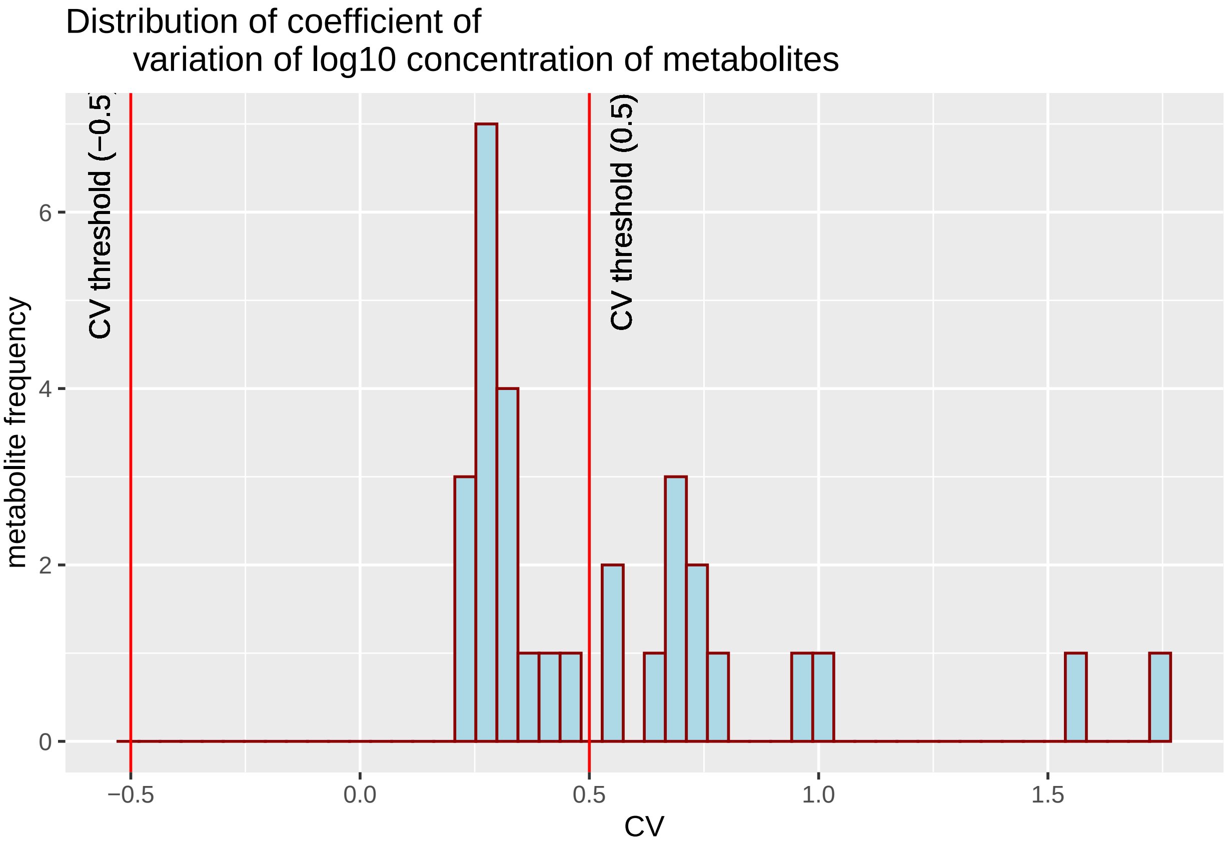
Biclustering is a clustering method, which simultaneously clusters rows and columns. In this example, the aim is to find clusters where subset of taxa share similar pattern over subset of metabolites. In our case, we try to find clusters where taxa and metabolites correlate similarly.

Check more from OMA which has dedicated chapter on biclustering.

First, we subset metabolite data so that it includes fewer metabolites. We use coefficient of variation as a subset criteria. Meaning of this is that, if concentration of metabolite does not vary, concentration does not differ between samples.

By doing that, we avoid unnecessary multiple testing, and we focus only on interesting metabolites.

```
library(ggplot2)
# 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 coefficient 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))

plot
```



Subsetting metabolomic data:

```
# 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")]
```

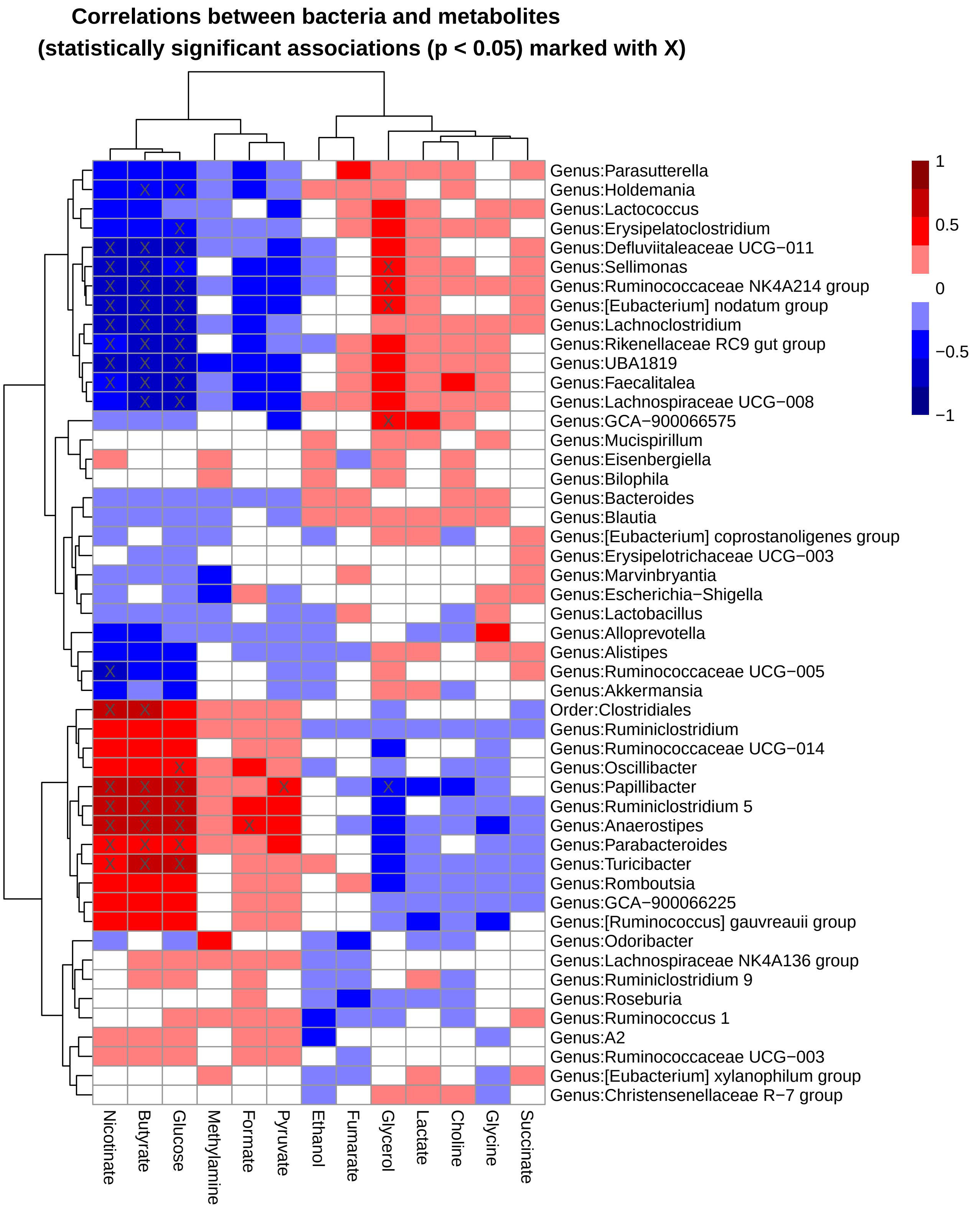
After that, we subset microbiome data by filtering rarest taxa off-

```
rank <- "Genus"
prevalence <- 0.2
detection <- 0.001
taxa_trans <-  "clr"

# 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)

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

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

# Remove uncultured and ambiguous(as it's hard to interpret their results)
taxa_tse <- taxa_tse[-grep("uncultured|Ambiguous_taxa", names(taxa_tse)),]
```

After subsetting, we can perform cross-correlation analysis. With cross-correlation analysis we can answer the question “If the abundance of this taxon is high, is the concentration of metabolite high?”, for instance.

Check OMA book’s dedicated chapther on multi-assay analyses.

```
library(pheatmap)

# Cross correlate data sets
correlations <- testExperimentCrossCorrelation(taxa_tse, metabolite_tse,
                                               abund_values1 = "clr", abund_values2 = "nmr",
                                               method = "spearman", mode = "matrix")

# For plotting purpose, convert p-values, under 0.05 are marked with "X"
p_threshold <- 0.01
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
pheatmap(correlations$cor, display_numbers = p_values,
               main = paste0("Correlations between bacteria and metabolites
              (statistically significant associations (p < 0.05) marked with X)"),
               fontsize = 10,
               breaks = breaks,
               color = colors,
               fontsize_number = 10)
```


Correlations between bacteria and metabolites
(statistically significant associations (p < 0.05) marked with X)
Genus:Parasutterella
Genus:Holdemania
Genus:Lactococcus
Genus:Erysipelatoclostridium
Genus:Defluviitaleaceae UCG−011
Genus:Sellimonas
Genus:Ruminococcaceae NK4A214 group
Genus:[Eubacterium] nodatum group
Genus:Lachnoclostridium
Genus:Rikenellaceae RC9 gut group
Genus:UBA1819
Genus:Faecalitalea
Genus:Lachnospiraceae UCG−008
Genus:GCA−900066575
Genus:Mucispirillum
Genus:Eisenbergiella
Genus:Bilophila
Genus:Bacteroides
Genus:Blautia
Genus:[Eubacterium] coprostanoligenes group
Genus:Erysipelotrichaceae UCG−003
Genus:Marvinbryantia
Genus:Escherichia−Shigella
Genus:Lactobacillus
Genus:Alloprevotella
Genus:Alistipes
Genus:Ruminococcaceae UCG−005
Genus:Akkermansia
Order:Clostridiales
Genus:Ruminiclostridium
Genus:Ruminococcaceae UCG−014
Genus:Oscillibacter
Genus:Papillibacter
Genus:Ruminiclostridium 5
Genus:Anaerostipes
Genus:Parabacteroides
Genus:Turicibacter
Genus:Romboutsia
Genus:GCA−900066225
Genus:[Ruminococcus] gauvreauii group
Genus:Odoribacter
Genus:Lachnospiraceae NK4A136 group
Genus:Ruminiclostridium 9
Genus:Roseburia
Genus:Ruminococcus 1
Genus:A2
Genus:Ruminococcaceae UCG−003
Genus:[Eubacterium] xylanophilum group
Genus:Christensenellaceae R−7 group
Nicotinate
Butyrate
Glucose
Methylamine
Formate
Pyruvate
Ethanol
Fumarate
Glycerol
Lactate
Choline
Glycine
Succinate
1
0.5
0
−0.5
−1
X

Finally, we can find biclusters from cross-correlation data.

```
# Load package
library(biclust)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: grid

##
## Attaching package: 'grid'

## The following object is masked from 'package:Biostrings':
##
##     pattern

## Loading required package: colorspace

## Loading required package: lattice
```

```
# Find biclusters
bc <- biclust(correlations$cor, method=BCPlaid(), fit.model = y ~ m,
              background = TRUE, shuffle = 100, back.fit = 0, max.layers = 10,
              iter.startup = 10, iter.layer = 100, verbose = FALSE)

bc
```

```
##
## An object of class Biclust
##
## call:
##  biclust(x = correlations$cor, method = BCPlaid(), fit.model = y ~
##      m, background = TRUE, shuffle = 100, back.fit = 0, max.layers = 10,
##      iter.startup = 10, iter.layer = 100, verbose = FALSE)
##
## Number of Clusters found:  4
##
## First  4  Cluster sizes:
##                       BC 1 BC 2 BC 3 BC 4
## Number of Rows:        12   13    8    5
## Number of Columns:      3    5    4    5
```

To get cluster information into right format, we can use functions from OMA book. They add cluster for those features that were not assigned to any cluster.

```r
# Functions for obtaining biclust information

# Get clusters for rows and columns
.get_biclusters_from_biclust <- function(bc, assay){
  # Get cluster information for columns and rows
  bc_columns <- t(bc@NumberxCol)
  bc_columns <- data.frame(bc_columns)
  bc_rows <- bc@RowxNumber
  bc_rows <- data.frame(bc_rows)

  # Get data into right format
  bc_columns <- .manipulate_bc_data(bc_columns, assay, "col")
  bc_rows <- .manipulate_bc_data(bc_rows, assay, "row")

  return(list(bc_columns = bc_columns, bc_rows = bc_rows))
}

# Input clusters, and how many observations there should be, i.e., the number of samples or features
.manipulate_bc_data <- function(bc_clusters, assay, row_col){
  # Get right dimension
  dim <- ifelse(row_col == "col", ncol(assay), nrow(assay))
  # Get column/row names
  if( row_col == "col" ){
    names <- colnames(assay)
  } else{
    names <- rownames(assay)
  }

  # If no clusters were found, create one. Otherwise create additional cluster which
  # contain those samples that are not included in clusters that were found.
  if( nrow(bc_clusters) != dim ){
      bc_clusters <- data.frame(cluster = rep(TRUE, dim))
  } else {
      # Create additional cluster that includes those samples/features that
      # are not included in other clusters.
      vec <- ifelse(rowSums(bc_clusters) > 0, FALSE, TRUE)
      # If additional cluster contains samples, then add it
      if ( any(vec) ){
          bc_clusters <- cbind(bc_clusters, vec)
      }
  }
  # Adjust row and column names
  rownames(bc_clusters) <- names
  colnames(bc_clusters) <- paste0("cluster_", 1:ncol(bc_clusters))
  return(bc_clusters)
}
```

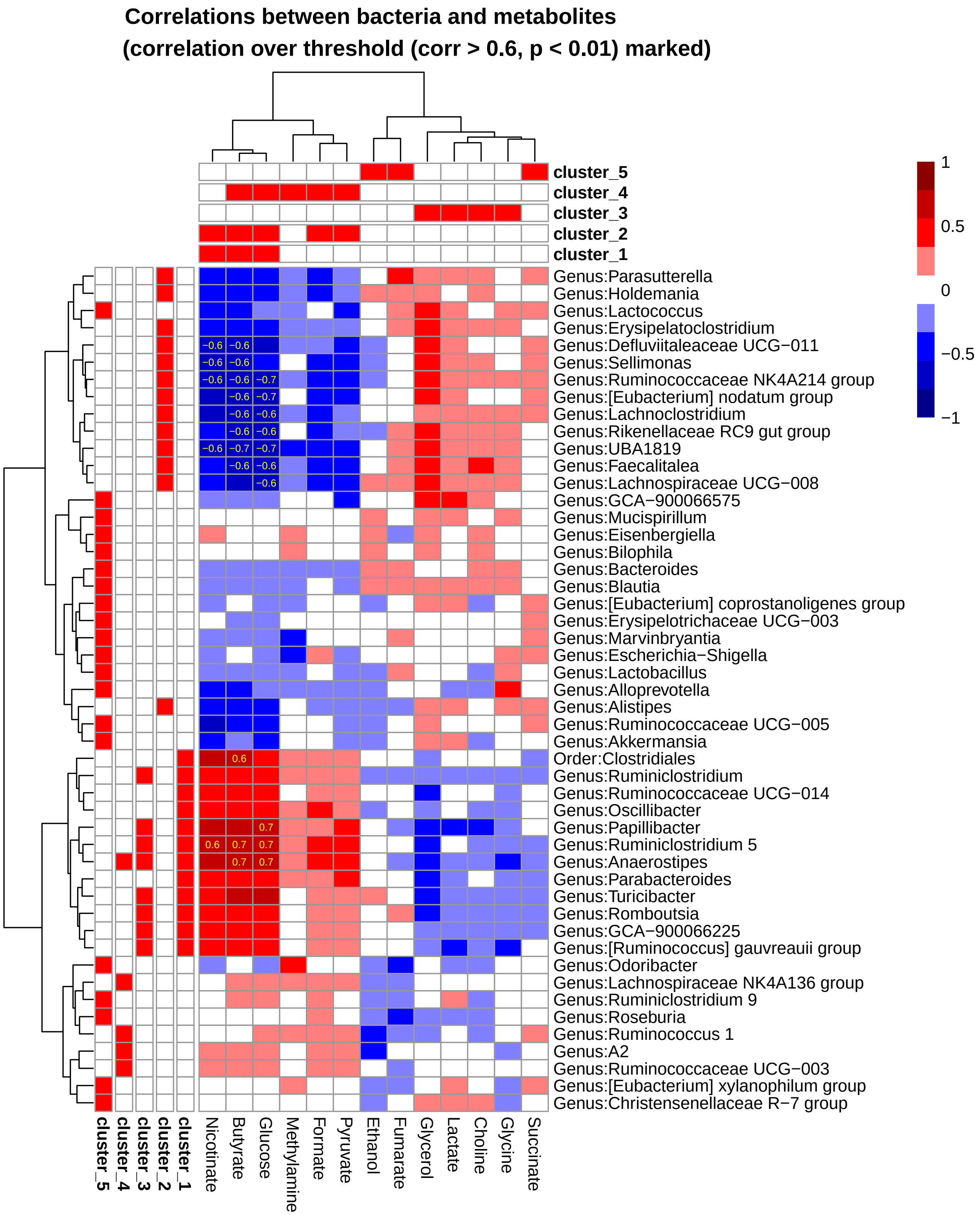
Then we can use the functions.

```
# Get biclusters
bcs <- .get_biclusters_from_biclust(bc, correlations$cor)

bicluster_rows <- bcs$bc_rows
bicluster_columns <- bcs$bc_columns

# Print biclusters for rows
head(bicluster_rows)
```

```
##                               cluster_1 cluster_2 cluster_3 cluster_4 cluster_5
## Genus:Escherichia-Shigella         FALSE     FALSE     FALSE     FALSE      TRUE
## Genus:Ruminiclostridium 5           TRUE     FALSE      TRUE     FALSE     FALSE
## Genus:Lactobacillus                FALSE     FALSE     FALSE     FALSE      TRUE
## Genus:Ruminococcaceae UCG-014       TRUE     FALSE     FALSE     FALSE     FALSE
## Genus:Lactococcus                  FALSE     FALSE     FALSE     FALSE      TRUE
## Genus:Mucispirillum                FALSE     FALSE     FALSE     FALSE      TRUE
```

Now, we can add bicluster information into the heatmap that we already made.

```
# Convert boolean values into factors
bicluster_columns <- data.frame(apply(bicluster_columns, 2, as.factor))
bicluster_rows <- data.frame(apply(bicluster_rows, 2, as.factor))

# Adjust colors for all clusters
if( ncol(bicluster_rows) > ncol(bicluster_columns) ){
  cluster_names <- colnames(bicluster_rows)
} else {
  cluster_names <- colnames(bicluster_columns)
}
annotation_colors <- list()
for(name in cluster_names){
  annotation_colors[[name]] <- c("TRUE" = "red", "FALSE" = "white")
}

# Get correlation values that are over thresholds
p_threshold <- 0.01
corr_threshold <- 0.6
corr_values <- ifelse(correlations$p_adj<p_threshold &
                          abs(correlations$cor)>corr_threshold , round(correlations$cor,1), "")

# Create a heatmap
pheatmap(correlations$cor,
         annotation_col = bicluster_columns,
         annotation_row = bicluster_rows,
         annotation_colors = annotation_colors,
         display_numbers = corr_values,
               main = paste0("Correlations between bacteria and metabolites
              (correlation over threshold (corr > ", corr_threshold,
              ", p < ", p_threshold,") marked)"),
               fontsize = 10,
               breaks = breaks,
               color = colors,
               fontsize_number = 6,
         number_color = "yellow",
         annotation_legend = FALSE)
```


Correlations between bacteria and metabolites
(correlation over threshold (corr > 0.6, p < 0.01) marked)
cluster_5
cluster_4
cluster_3
cluster_2
cluster_1
Genus:Parasutterella
Genus:Holdemania
Genus:Lactococcus
Genus:Erysipelatoclostridium
Genus:Defluviitaleaceae UCG−011
Genus:Sellimonas
Genus:Ruminococcaceae NK4A214 group
Genus:[Eubacterium] nodatum group
Genus:Lachnoclostridium
Genus:Rikenellaceae RC9 gut group
Genus:UBA1819
Genus:Faecalitalea
Genus:Lachnospiraceae UCG−008
Genus:GCA−900066575
Genus:Mucispirillum
Genus:Eisenbergiella
Genus:Bilophila
Genus:Bacteroides
Genus:Blautia
Genus:[Eubacterium] coprostanoligenes group
Genus:Erysipelotrichaceae UCG−003
Genus:Marvinbryantia
Genus:Escherichia−Shigella
Genus:Lactobacillus
Genus:Alloprevotella
Genus:Alistipes
Genus:Ruminococcaceae UCG−005
Genus:Akkermansia
Order:Clostridiales
Genus:Ruminiclostridium
Genus:Ruminococcaceae UCG−014
Genus:Oscillibacter
Genus:Papillibacter
Genus:Ruminiclostridium 5
Genus:Anaerostipes
Genus:Parabacteroides
Genus:Turicibacter
Genus:Romboutsia
Genus:GCA−900066225
Genus:[Ruminococcus] gauvreauii group
Genus:Odoribacter
Genus:Lachnospiraceae NK4A136 group
Genus:Ruminiclostridium 9
Genus:Roseburia
Genus:Ruminococcus 1
Genus:A2
Genus:Ruminococcaceae UCG−003
Genus:[Eubacterium] xylanophilum group
Genus:Christensenellaceae R−7 group
cluster_5
cluster_4
cluster_3
cluster_2
cluster_1
Nicotinate
Butyrate
Glucose
Methylamine
Formate
Pyruvate
Ethanol
Fumarate
Glycerol
Lactate
Choline
Glycine
Succinate
1
0.5
0
−0.5
−1

Chapter 8

Supervised learning

8.1 Random Forests

Creating a dataframe for modeling butyrate levels:

```
butyrate_df <- data.frame(cbind(y, x))
butyrate_df <- butyrate_df[,which(colnames(butyrate_df) %in% c("Butyrate", colnames(x)))]
```

Performing nested cross validation, making train and test (validation) sets:

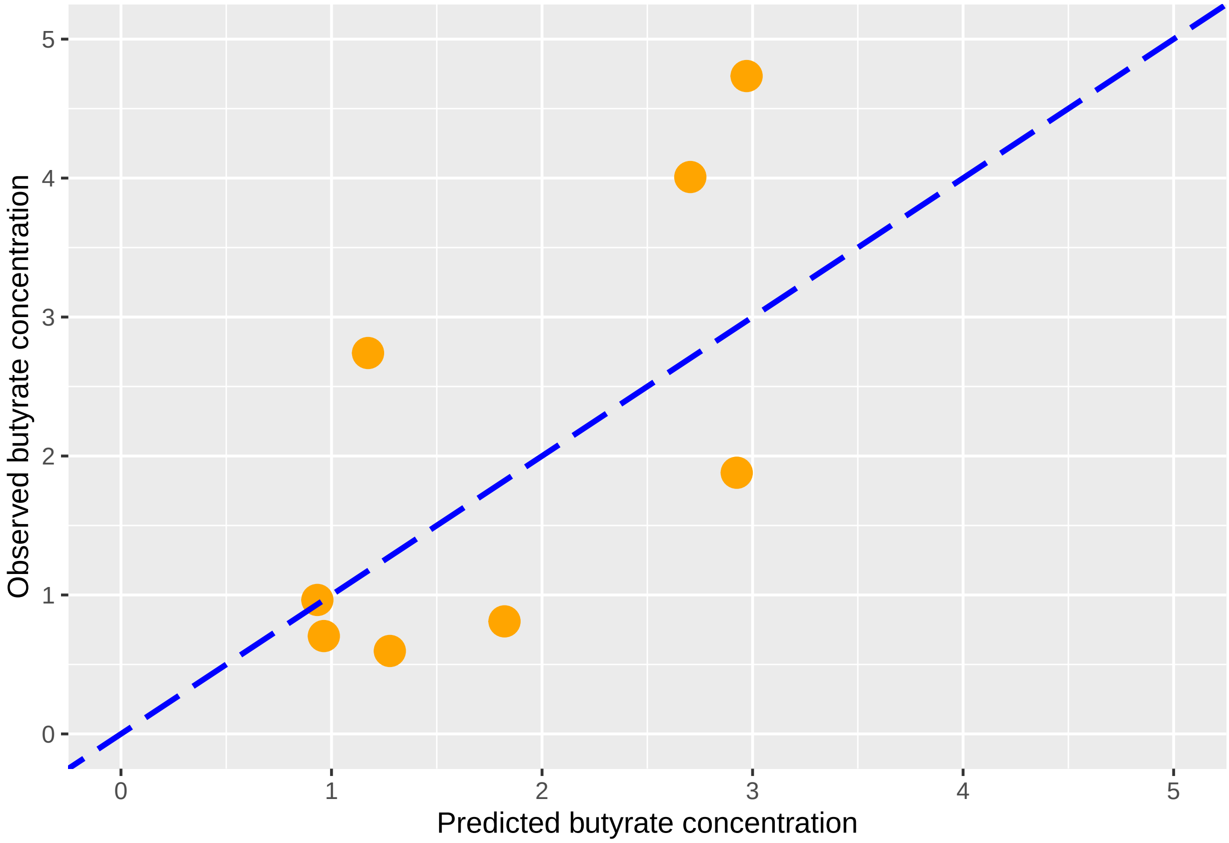
```
library(caret)
set.seed(42)
trainIndex <- createDataPartition(butyrate_df$Butyrate, p = .8, list = FALSE, times = 1)
butyrate_df_train <- butyrate_df[trainIndex,]
butyrate_df_test <- butyrate_df[-trainIndex,]
```

Train models with 5-fold CV repeated 5 times:

```
set.seed(42)
fitControl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
rfFit1 <- train(Butyrate ~ ., data = butyrate_df_train,
                method = "ranger",
                trControl = fitControl,
                importance = "permutation")
```

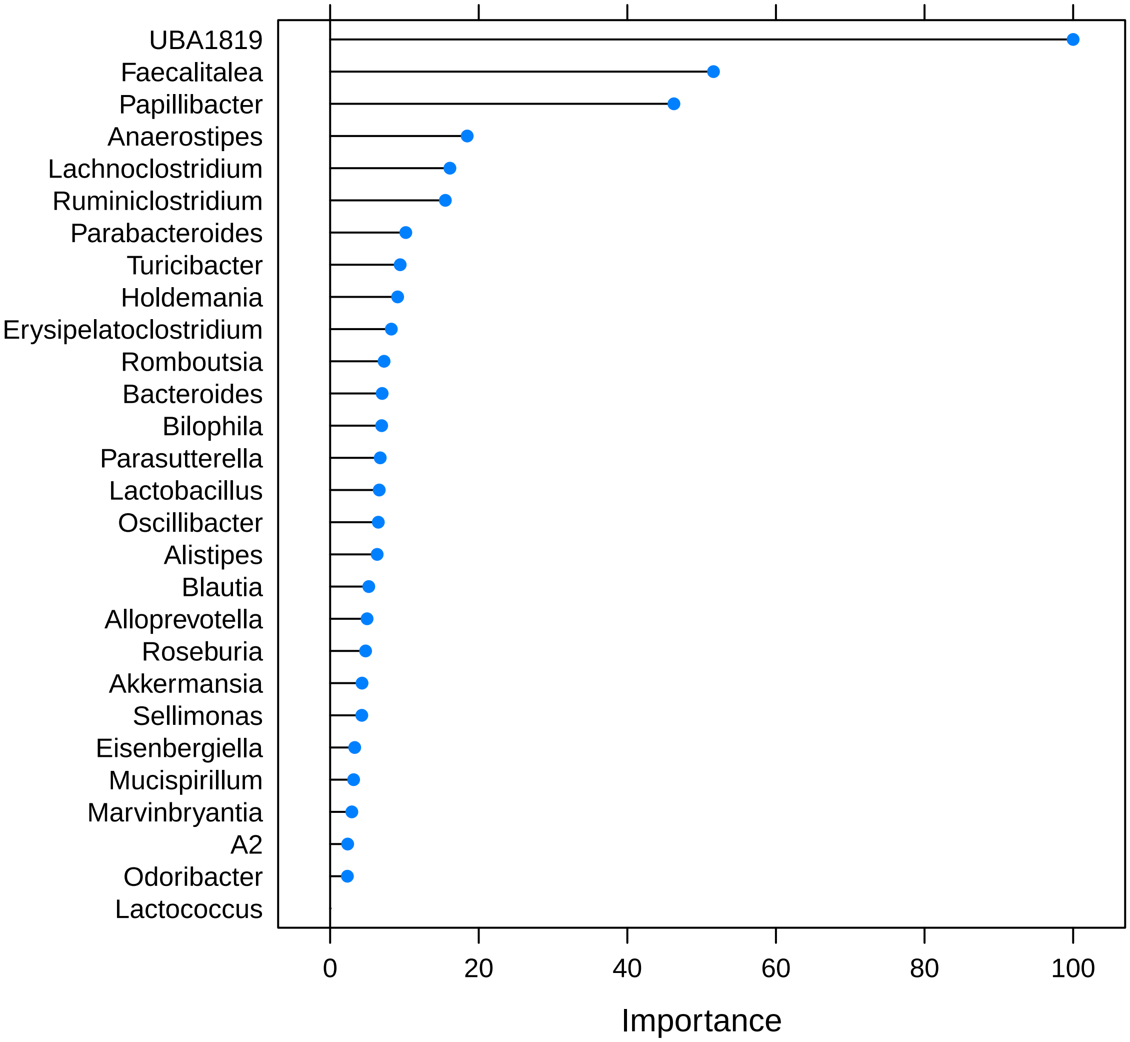
Measure performance by prediction on test data:

```
test_predictions <- predict(rfFit1, newdata = butyrate_df_test)

# Plot predicted vs observed
pred_obs <- data.frame(predicted = test_predictions, observed = butyrate_df_test$Butyrate)
ggplot(data = pred_obs, aes(x=predicted, y=observed)) + geom_point(size = 5, color = "orange") +
  xlab("Predicted butyrate concentration") + ylab("Observed butyrate concentration") +
  lims(x = c(0,5), y = c(0,5)) +
  geom_abline(linetype = 5, color = "blue", size = 1) # Plot a perfect fit line
```



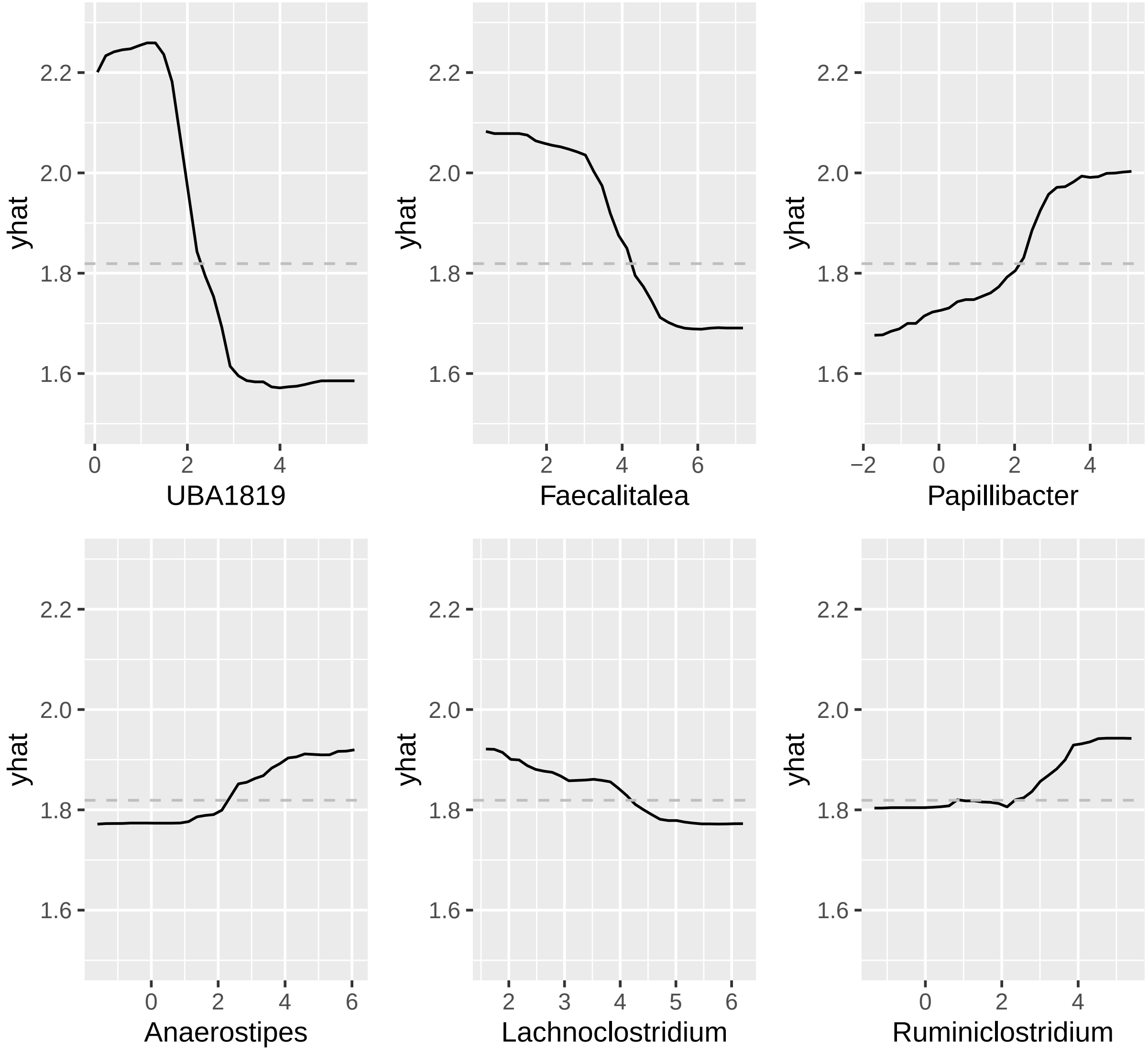
Examining model by ploting feature importance:

```
plot(varImp(rfFit1))
```



We could investigate association directions by making partial dependence plots:

```r
library(patchwork)
library(pdp)
# Calculate and plot partial dependence
top_features <- rownames(varImp(rfFit1)$importance)[order(varImp(rfFit1)$importance[,"Overall"], decreasing =
pd_plots <- list(NULL)
for (feature in 1:length(top_features)) {
  pd_plots[[feature]] <- partial(rfFit1, pred.var = top_features[feature], rug = TRUE) %>% autoplot() +
    geom_hline(yintercept = mean(butyrate_df_train$Butyrate), linetype = 2, color = "gray") + # Show the mean
    scale_y_continuous(limits=c(1.5,2.3)) # Harmonize the scale of yhat on all plots
}
wrap_plots(pd_plots)
```


yhat
2.2
2.0
1.8
1.6
0
2
4
UBA1819
2
4
6
Faecalitalea
−2
0
2
4
Papillibacter
0
2
4
6
Anaerostipes
2
3
4
5
6
Lachnoclostridium
0
2
4
Ruminiclostridium

Chapter 9

Model selection and evaluation

Coming up...

Bibliography

Borman, T., Eckermann, H., Benchraka, C., and Lahti, L. (2022). *Introduction to multi-omics data analysis.*