Introduction to microbiome data science

University of Turku

2021-09-14

Contents

Chapter 1

Overview

Welcome to Radboud Summer School, July 2021

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.

The training material walks you through an example workflow that shows the standard steps of taxonomic data analysis covering data access, exploration, analysis, visualization and reporoducible 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

This course provides an overview of the standard bioinformatics workflow in taxonomic profiling studies, ranging from data preprocessing to statistical analysis and reproducible reporting, with a focus on examples from human gut microbiota studies. You will become familiar with standard bioinformatics concepts and methods in taxonomic profiling studies of the human microbiome. This includes better understanding of the specific statistical challenges, practical hands-on experience with the commonly used methods, and reproducible research with R.

After the course you will know how to approach new tasks in microbiome data science by utilizing available documentation and R tools.

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

Venue Radboud University / Online, Nijmegen. July 5-16, 2021, with contributions by University of Turku, Finland.

1.3 Acknowledgments

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

Borman et al. (2021)

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.

- Source code (github): miaverse teaching material
- Course page (html): miaverse teaching material

Chapter 2

Program

The course takes place on each working day from 9am – 1pm (CEST). Short breaks will be scheduled between sessions.

The hands-on sessions consists of a set of questions and example data. Solve the exercises by taking advantage of the online examples and resources that are pointed out in the study material. There is often more than one way to solve a given task. 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: from raw sequences to ecological data analysis

Lectures

- Microbiota analysis: association studies vs. causality; microbiota sequencing methods (16S, shotgun, metagenomics) - by dr. Tom Ederveen (Radboud UMC Nijmegen, The Netherlands)
- DNA isolation and 16S rRNA gene sequencing; bioinformatics step 1: from raw sequences to OTU table in a biom file – by Tom Ederveen (Radboudumc Nijmegen, The Netherlands)

Demo & Practical

- Importing data to R for interactive data analysis
- Task: initialize reproducible report

2.2 Day 2 - Alpha diversity

Demo

- Microbiome data exploration

Lecture

- Key concepts in microbiome data science

Practical

- Alpha diversity: estimation, analysis, and visualization

2.3 Day 3 - Beta diversity

Demo

- Community similarity

Practical

- Beta diversity: estimation, analysis, and visualization

2.4 Day 4- Differential abundance

Lecture

- Differential abundance analysis methods

Practical

- Differential abundance in practice

Lecture

- Overview of microbiota data science methods & concepts

2.5 Day 5 : Presentations & closing

Student presentations on microbiome data analytics

Chapter 3

Getting started

3.1 Checklist (before the course)

Install the following software in advance in order to avoid unnecessary delays and leaving more time for the course 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
- After a successful installation you can start with the case study examples in this training material

3.2 Support and resources

For additional reading and online material, see Material section

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.

```
# List of packages that we need from cran and bioc
cran_pkg <- c("BiocManager", "bookdown", "dplyr", "ecodist", "ggplot2",
              "gridExtra", "kableExtra", "knitr", "scales", "vegan")
bioc_pkg <- c("ANCOMBC", "ape", "DESeq2",  "DirichletMultinomial", "mia", "miaViz")
```

```
# 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
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
## ANCOMBC                 TRUE
## ape                     TRUE
## DESeq2                  TRUE
## DirichletMultinomial    TRUE
## BiocManager             TRUE
## bookdown                TRUE
## dplyr                   TRUE
## ecodist                 TRUE
## ggplot2                 TRUE
## gridExtra               TRUE
## kableExtra              TRUE
## knitr                   TRUE
## scales                  TRUE
## vegan                   TRUE
```

Chapter 4

Reproducible reporting with Rmarkdown

Reproducible reporting is the starting point for robust interactive data science. Perform the following tasks:

- Create a Rmarkdown template in RStudio, and render it into a document (markdown, PDF, docx or other format)
- If you are entirely new to Markdown, take this 10 minute tutorial to get introduced to the most important functions within Markdown. Then experiment with different options with Rmarkdown

In case you are new to Rmarkdown Rstudio provides resources to learn about the use cases and the basics of Rmarkdown. A good online tutorial to reproducible reporting is available by Dr. C Titus Brown.

Chapter 5

Importing microbiome data

This section demonstrates how to import microbiome profiling data in R.

5.1 Data access

Option 1

ADHD-associated changes in gut microbiota and brain in a mouse model

Tengeler AC *et al.* (2020) **Gut microbiota from persons with attention-deficit/hyperactivity disorder affects the brain in mice**. Microbiome 8:44.

In this study, mice are colonized with microbiota from participants with ADHD (attention deficit hyperactivity disorder) and healthy participants. The aim of the study was to assess whether the mice display ADHD behaviors after being inoculated with ADHD microbiota, suggesting a role of the microbiome in ADHD pathology.

Download the data from data subfolder.

Option 2

Western diet intervention study in serotonin transporter modified mouse model

Mice received three weeks of Western (high fat, high sugar) or Control diet in three Genotype groups, a wild-type group and 2 groups with either partial or complete knock-out of the serotonin transporter (SERT), leading to an excess of serotonin in the gut, brain and blood circulation. Mice were measured before and after the intervention. In total there are three factors in this design, Time (pre-,post-intervention), Diet (Control, Western diet) and Genotype group (WT, HET, KO). There are about 6-8 mice per group.

Data can be found from Summer School's Brightspace.

Option 3

Open data set of your own choice, see e.g.:

- Bioconductor microbiomeDataSets

5.2 Importing microbiome data in R

Import example data by modifying the examples in the online book section on data exploration and manipulation. The data files in our example are in *biom* format, which is a standard file format for microbiome data. Other file formats exist as well, and import details vary by platform.

Here, we import *biom* data files into a specific data container (structure) in R, *TreeSummarizedExperiment* (TSE) Huang et al. (2020). This provides the basis for downstream data analysis in the *miaverse* data science framework.

In this course, we focus on downstream analysis of taxonomic profiling data, and assume that the data has already been appropriately preprocessed and available in the TSE format. In addition to our example data, further demonstration data sets are readily available in the TSE format through microbiomeDataSets.

Figure sources:

Original article - Huang R *et al.* (2021) TreeSummarizedExperiment: a S4 class for data with hierarchical structure. F1000Research 9:1246.

Reference Sequence slot extension - Lahti L *et al.* (2020) Upgrading the R/Bioconductor ecosystem for microbiome research F1000Research 9:1464 (slides).

5.3 Example solutions

- Example code for data import: import.Rmd

```
## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars
```

```
## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians
```

```
## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians

## Loading required package: SingleCellExperiment

## Loading required package: TreeSummarizedExperiment

## Loading required package: Biostrings

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit

## Loading required package: ggplot2

## Loading required package: ggraph

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:Biostrings':
##
##     collapse, intersect, setdiff, setequal, union

## The following object is masked from 'package:XVector':
##
##     slice

## The following object is masked from 'package:Biobase':
##
##     combine

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect
```

```
## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following object is masked from 'package:matrixStats':
##
##     count

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Chapter 6

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.

6.1 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 151 taxa and 27 samples.

```
dim(tse)
```

```
## [1] 151  27
```

The `rowData` slot contains a taxonomic table. This includes taxonomic information for each of the 151 entries. With the `head()` command, we can print just the beginning of the table.

The `rowData` seems to contain information from 6 different taxonomy classes.

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

The colData slot contains sample metadata. It contains information for all 27 samples. However, here only the 6 first samples are shown as we use the `head()` command. There are 4 columns, that contain information, e.g., about patients' status, and cohort.

	Kingdom	Phylum	Class	Order	Family	Genus
1726470	Bacteria	Bacteroidetes	Bacteroidia	Bacteroidales	Bacteroidaceae	Bacteroides
1726471	Bacteria	Bacteroidetes	Bacteroidia	Bacteroidales	Bacteroidaceae	Bacteroides
17264731	Bacteria	Bacteroidetes	Bacteroidia	Bacteroidales	Porphyromonadaceae	Parabacteroides
17264726	Bacteria	Bacteroidetes	Bacteroidia	Bacteroidales	Bacteroidaceae	Bacteroides
1726472	Bacteria	Verrucomicrobia	Verrucomicrobiae	Verrucomicrobiales	Verrucomicrobiaceae	Akkermansia
17264724	Bacteria	Bacteroidetes	Bacteroidia	Bacteroidales	Bacteroidaceae	Bacteroides

	patient_status	cohort	patient_status_vs_cohort	sample_name
A110	ADHD	Cohort_1	ADHD_Cohort_1	A110
A12	ADHD	Cohort_1	ADHD_Cohort_1	A12
A15	ADHD	Cohort_1	ADHD_Cohort_1	A15
A19	ADHD	Cohort_1	ADHD_Cohort_1	A19
A21	ADHD	Cohort_2	ADHD_Cohort_2	A21
A23	ADHD	Cohort_2	ADHD_Cohort_2	A23

```
knitr::kable(head(colData(tse))) %>%
  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 patient status distribution.

The command `colData(tse)$patient_status` 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 13 samples from patients having ADHD, and 14 control samples.

```
sort(table(colData(tse)$patient_status))
```

```
##
##    ADHD Control
##      13      14
```

6.1.1 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
tse <- transformCounts(tse, method = "relabundance")
```

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

6.1.2 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.

	Kingdom	Phylum	Class	Order	Family	Genus
Bacteroidetes	Bacteria	Bacteroidetes	NA	NA	NA	NA
Verrucomicrobia	Bacteria	Verrucomicrobia	NA	NA	NA	NA
Proteobacteria	Bacteria	Proteobacteria	NA	NA	NA	NA
Firmicutes	Bacteria	Firmicutes	NA	NA	NA	NA
Cyanobacteria	Bacteria	Cyanobacteria	NA	NA	NA	NA

	Kingdom	Phylum	Class	Order	Family	Genus
Family:Lachnospiraceae	Bacteria	Firmicutes	Clostridia	Clostridiales	Lachnospiraceae	
Order:Bacteroidales	Bacteria	Bacteroidetes	Bacteroidia	Bacteroidales		
Order:Clostridiales	Bacteria	Firmicutes	Clostridia	Clostridiales		
Family:Enterobacteriaceae	Bacteria	Proteobacteria	Gammaproteobacteria	Enterobacteriales	Enterobacteriaceae	
Order:Gastranaerophilales	Bacteria	Cyanobacteria	Melainabacteria	Gastranaerophilales		

```
tse_phylum <- agglomerateByRank(tse, rank = "Phylum")

# Show dimensionality
dim(tse_phylum)
```

```
## [1]  5 27
```

Now there are 5 taxa and 27 samples, meaning that there are 5 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.

```
knitr::kable(head(rowData(tse_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.

```
temp <- rowData(agglomerateByRank(tse, rank = "Genus"))

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

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

```
temp2 <- rowData(agglomerateByRank(tse, rank = "Genus", na.rm = TRUE))

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

```
## [1] "Agglomeration with na.rm = FALSE: 54 taxa."
```

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

```
## [1] "Agglomeration with na.rm = TRUE: 49 taxa."
```

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

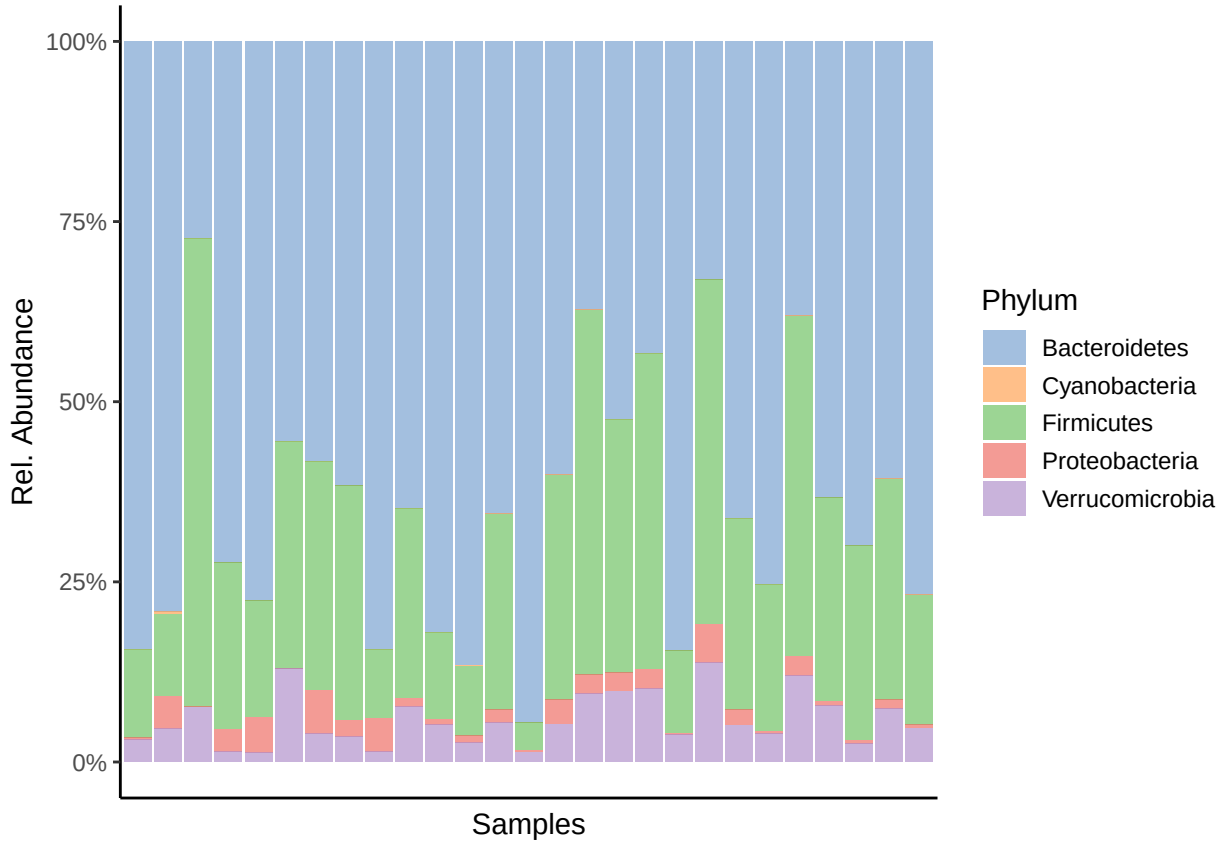
6.2 Visualization

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

```
# Here we specify "relabundance" to be abundance table that we use for plotting.
# Note that we can use agglomerated or non-agglomerated tse 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(tse, abund_values="relabundance", rank = "Phylum") +
  theme(legend.key.height = unit(0.5, "cm")) +
  scale_y_continuous(label = scales::percent)
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

```
plot_abundance
```

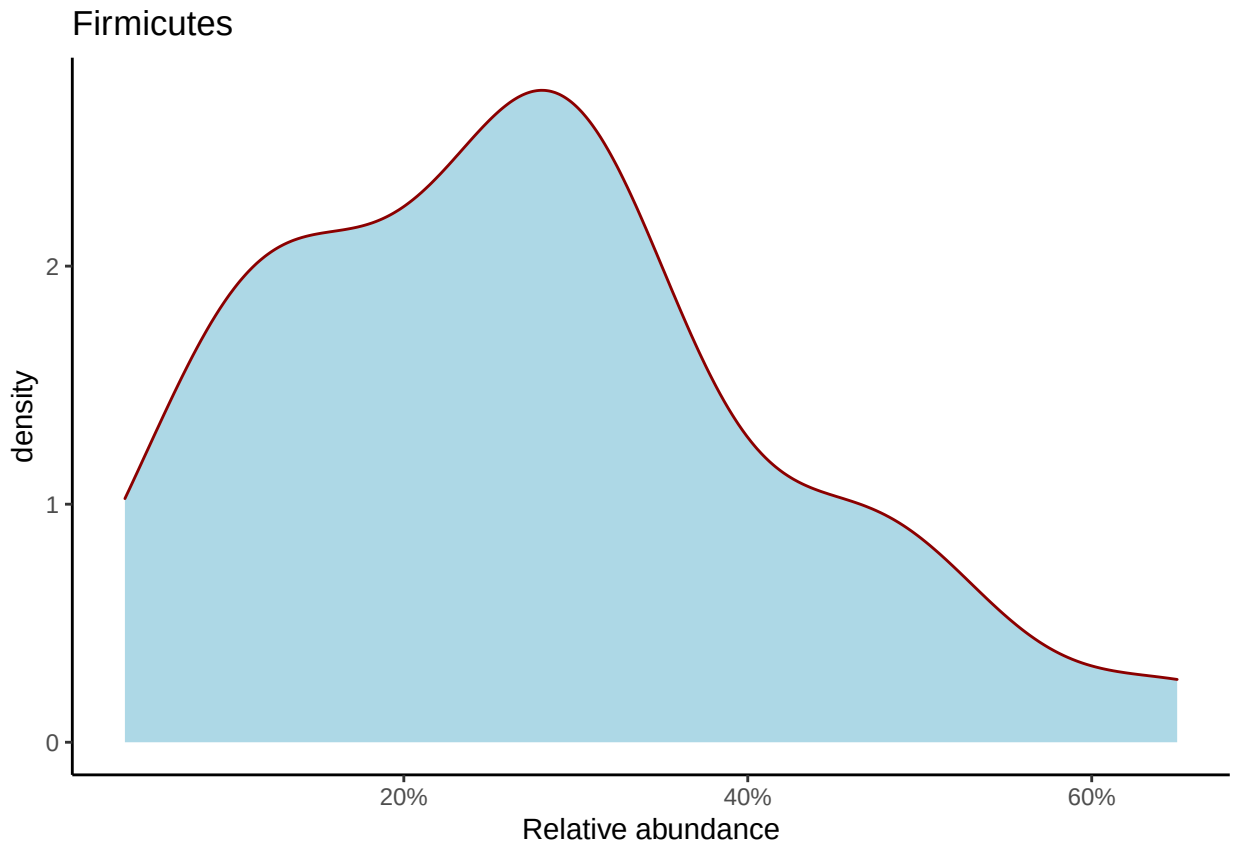


Density plot shows the overall abundance distribution for a given taxonomic group. Let us check the relative abundance of Firmicutes across the sample collection. The density plot is a smoothened version of a standard histogram.

The plot shows peak abundances around 30 %.

```
# Subset data by taking only Firmicutes
tse_firmicutes <- tse_phylum["Firmicutes"]

# Gets the abundance table
abundance_firmicutes <- assay(tse_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.

6.3 Exercises (optional)

Explore some of the following questions on your own by following online examples. Prepare a reproducible report (Rmarkdown), and include the code that you use to import the data and generate the analyses.

- **Abundance table** Retrieve the taxonomic abundance table from the example data set (TSE object). Tip: check "assays" in data import section
- How many different samples and genus-level groups this phyloseq object has? Tips: see dim(), rowData()
- What is the maximum abundance of Akkermansia in this data set? Tip: aggregate the data to Genus level with agglomerateByRank, pick abundance assay, and check a given genus (row) in the assay
- Draw a histogram of library sizes (total number of reads per sample). Tip: Library size section in OMA. You can use the available function, or count the sum of reads per sample by using the colSums command applied on the abundance table. Check Vandeputte et al. 2017 for further discussion on the differences between absolute and relative quantification of microbial abundances.

- **Taxonomy table** Retrieve the taxonomy table and print out the first few lines of it with the R command head(). Investigate how many different phylum-level groups this phyloseq object has? Tips: rowData, taxonomicRanks in OMA.
- **Sample metadata** Retrieve sample metadata. How many patient groups this data set has? Draw a histogram of sample diversities. Tips: colData
- **Subsetting** Pick a subset of the data object including only ADHD individuals from Cohort 1. How many there are? Tips: subsetSamples
- **Transformations** The data contains read counts. We can convert these into relative abundances and other formats. Compare abundance of a given taxonomic group using the example data before and after the compositionality transformation (with a cross-plot, for instance). You can also compare the results to CLR-transformed data (see e.g. Gloor et al. 2017)
- **Visual exploration** Visualize the population distribution of abundances for certain taxonomic groups. Do the same for CLR-transformed abundances. Tip: assays, transformCounts
- Experiment with other data manipulation tools from OMA.
- Example solution: Solutions

```
## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4
```

```
## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
## Loading required package: SingleCellExperiment

## Loading required package: TreeSummarizedExperiment

## Loading required package: Biostrings

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit

## Loading required package: ggplot2

## Loading required package: ggraph

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:Biostrings':
##
##     collapse, intersect, setdiff, setequal, union

## The following object is masked from 'package:XVector':
##
##     slice

## The following object is masked from 'package:Biobase':
##
##     combine

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union
```

```
## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following object is masked from 'package:matrixStats':
##
##     count

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Chapter 7

Alpha diversity

This section demonstrates the analysis of alpha diversity. This quantity measures microbial diversity within each sample. Higher numbers of unique taxa, and more even abundance distributions within a sample yield larger values for alpha diversity.

Alpha diversity is a key quantity in a microbiome research. The *mia* package provides access to a wide variety of alpha diversity indices. For more background information and examples with various alpha diversity indices, see the online book.

Let us show how to calculate to different diversity indices, Shannon and Faith. Shannon index reflects how many different taxa there are and how evenly they are distributed within a sample. Faith index additionally takes into account the phylogenetic relations into account.

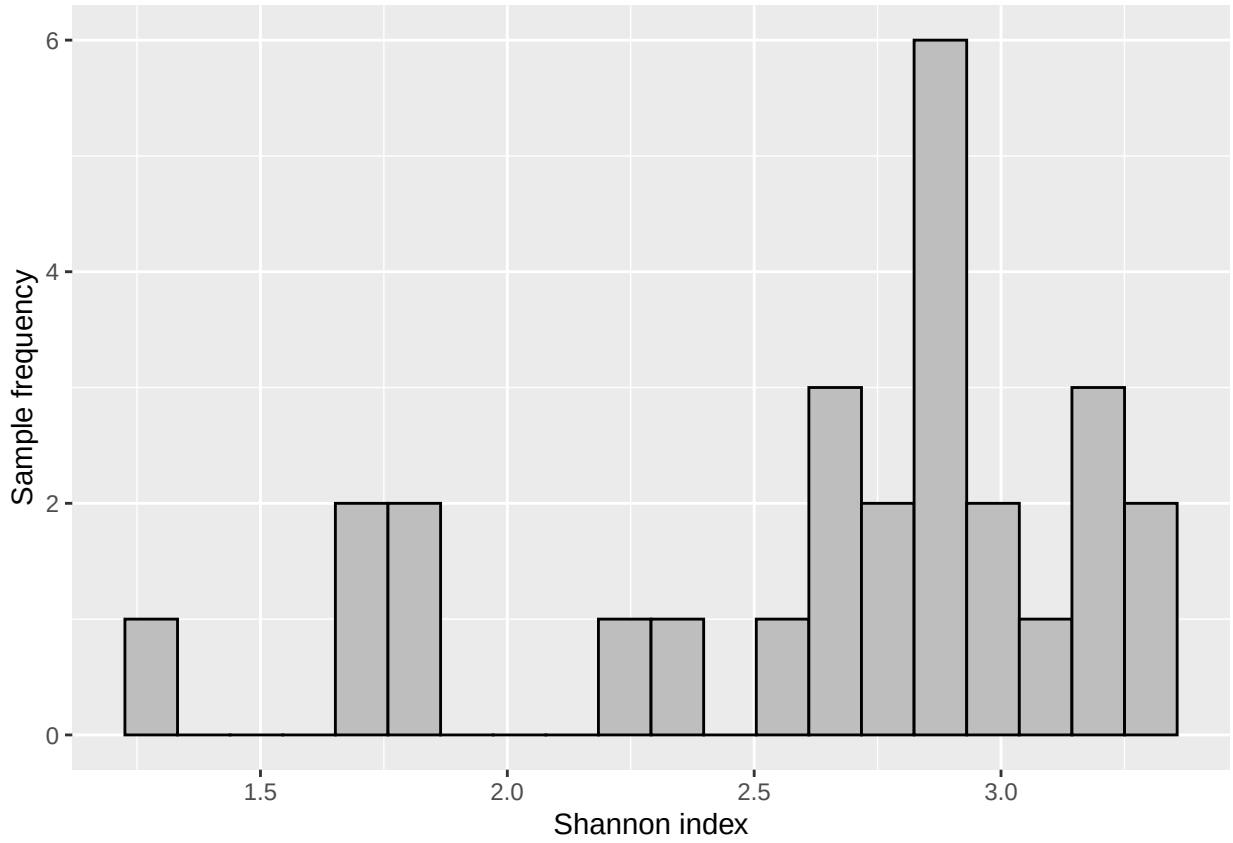
```
# Indices to be calculated.
# Every index is calculated by default if we don't specify indices.
indices <- c("shannon", "faith")

# Indices are stored in colData (i.e., sample metadata). We can specify the name
# of column, or we can use the default name which is the name of index
# (i.e., "shannon" and "faith").
names <- c("Shannon_diversity", "Faith_diversity")

# Calculates indices
tse <- estimateDiversity(tse, index = indices, name = names)

# Shows the calculated indices
knitr::kable(head(colData(tse)[names])) %>%
  kableExtra::kable_styling("striped",
                            latex_options="scale_down") %>%
  kableExtra::scroll_box(width = "100%")
```

Next we can visualize Shannon index with histogram.

```
# ggplot needs data.frame format as input.
# Here, colData is DataFrame, therefore it needs to be converted to data.frame
shannon_hist <- ggplot(as.data.frame(colData(tse)),
```

	Shannon_diversity	Faith_diversity
A110	1.765407	7.39224
A12	2.716438	6.29378
A15	3.178103	6.60608
A19	2.891987	6.79708
A21	2.841979	6.65110
A23	2.797942	5.96246

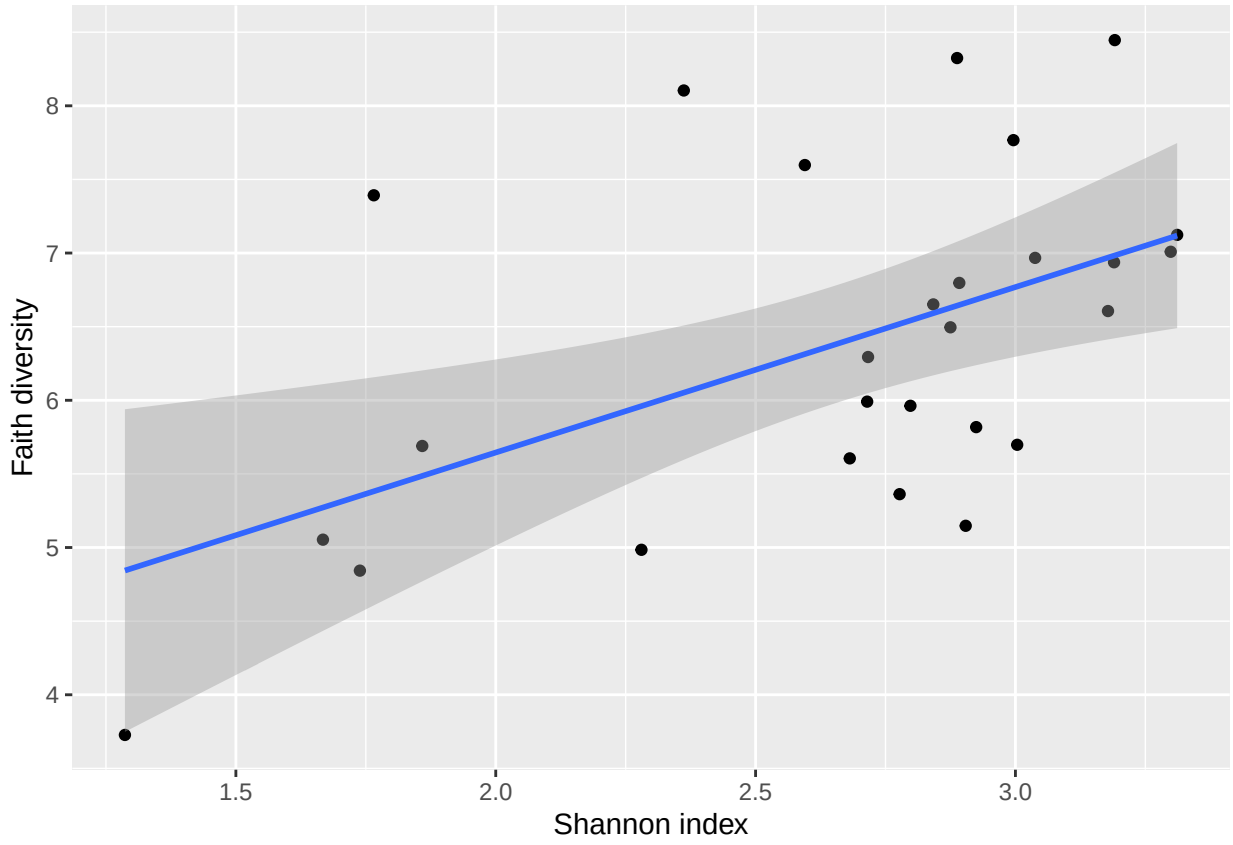
```
                      aes(x = Shannon_diversity)) +
  geom_histogram(bins = 20, fill = "gray", color = "black") +
  labs(x = "Shannon index", y = "Sample frequency")

shannon_hist
```



Next, let us compare the indices based on a scatter-plot.

```
cross_plot <- ggplot2::ggplot(as.data.frame(colData(tse)),
                              aes(x = Shannon_diversity, y = Faith_diversity)) +
  geom_point() + # Adds points
  geom_smooth(method=lm) + # Adds regression line
  labs(x = "Shannon index", y = "Faith diversity")

cross_plot
```

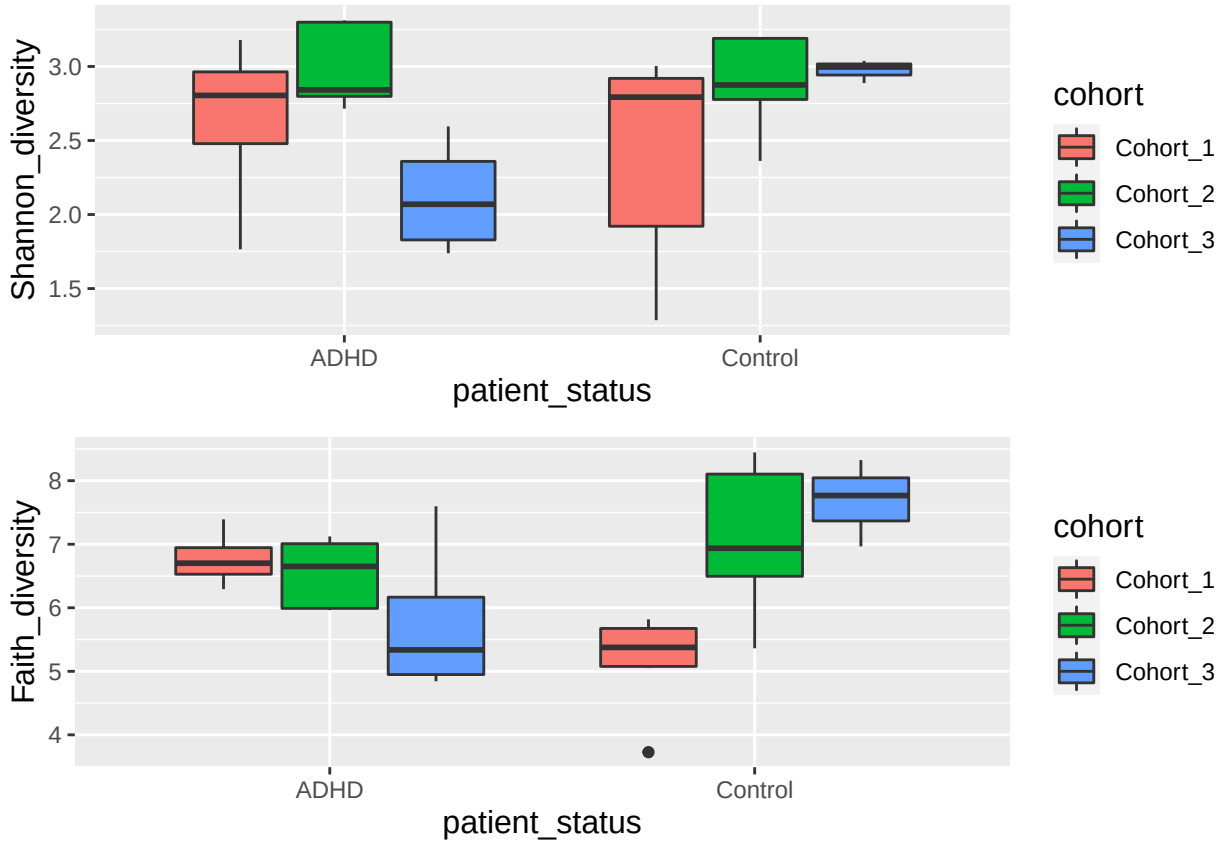
```
## `geom_smooth()` using formula 'y ~ x'
```



7.1 Visualization

Next let us compare indices between different patient status and cohorts. Boxplot is suitable for that purpose.

```
# Creates Shannon boxplot
shannon_box <- ggplot(as.data.frame(colData(tse)),
  aes(x = patient_status,
      y = Shannon_diversity,
      fill = cohort)) +
  geom_boxplot() +
  theme(title = element_text(size = 12)) # makes titles smaller

# Creates Faith boxplot
faith_box <- ggplot(as.data.frame(colData(tse)), aes(x = patient_status,
                                                     y = Faith_diversity,
                                                     fill = cohort)) +
  geom_boxplot() +
  theme(title = element_text(size = 12)) # makes titles smaller

# Puts them into same picture
gridExtra::grid.arrange(shannon_box, faith_box, nrow = 2)
```



For an alternative visualization, see examples with scater::plotColData.

7.2 Statistical testing and comparisons

To further investigate if patient status could explain the variation of Shannon index, let's do a Wilcoxon test. This is a non-parametric test that doesn't make specific assumptions about the distribution, unlike popular parametric tests, such as the t test, which assumes normally distributed observations.

Wilcoxon test can be used to estimate whether the differences between two groups is statistically significant. Here the ADHD and control groups are not significantly different between groups (p-value is over 0.05).

```
# Wilcoxon test, where Shannon index is the variable that we are comparing.
# Patient status - ADHD or control - is the factor that we use for grouping.
wilcoxon_shannon <- wilcox.test(Shannon_diversity ~ patient_status, data = colData(tse))

wilcoxon_shannon
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  Shannon_diversity by patient_status
## W = 76, p-value = 0.4879
```

```
## alternative hypothesis: true location shift is not equal to 0
```

Another test that we can make is to test if ADHD samples differs between different cohorts. From boxplot that we made in previous step, we can see that there might be statistically significant difference between different cohorts.

Let's compare Shannon index of ADHD samples between cohort 2 and cohort 3.

As we can see, there is statistically significant difference between the cohorts.

```
# Takes subset of colData. Takes only ADHD samples
ADHD_shannon <- colData(tse)[ colData(tse)[, "patient_status"] == "ADHD" , ]

# Takes subset of colData. Takes only samples that are in cohort 2 or cohort 3.
ADHD_shannon <- ADHD_shannon[ ADHD_shannon[, "cohort"] %in% c("Cohort_2", "Cohort_3") , ]

# Wilcoxon test, where Shannon index is the variable that we are comparing.
# Cohort - 2 or 3 - is the factor that we use for grouping.
wilcoxon_shannon_ADHD_cohorts <- wilcox.test(Shannon_diversity ~ cohort, data = ADHD_shannon)

wilcoxon_shannon_ADHD_cohorts
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  Shannon_diversity by cohort
## W = 20, p-value = 0.01587
## alternative hypothesis: true location shift is not equal to 0
```

For more examples, see a dedicated section on alpha diversity in the online book.

7.3 Exercises

Add the following in the reproducible summary report.

- Estimate alpha diversity for each sample and draw a histogram. Tip: estimateDiversity
- Compare the results between two or more alpha diversity indices (visually and/or statistically).
- See online book for further examples.
- Example Solutions

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians

## Loading required package: SingleCellExperiment

## Loading required package: TreeSummarizedExperiment

## Loading required package: Biostrings

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit

## Loading required package: ggplot2

## Loading required package: ggraph
```

Chapter 8

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. Each of these (dis)similarity measures emphasizes different aspects. For example, UniFrac incorporates phylogenetic information, and Jaccard index ignores exact abundances and considers only presence/absence values. For more background information and examples, you can check the dedicated section in online book.

8.1 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.

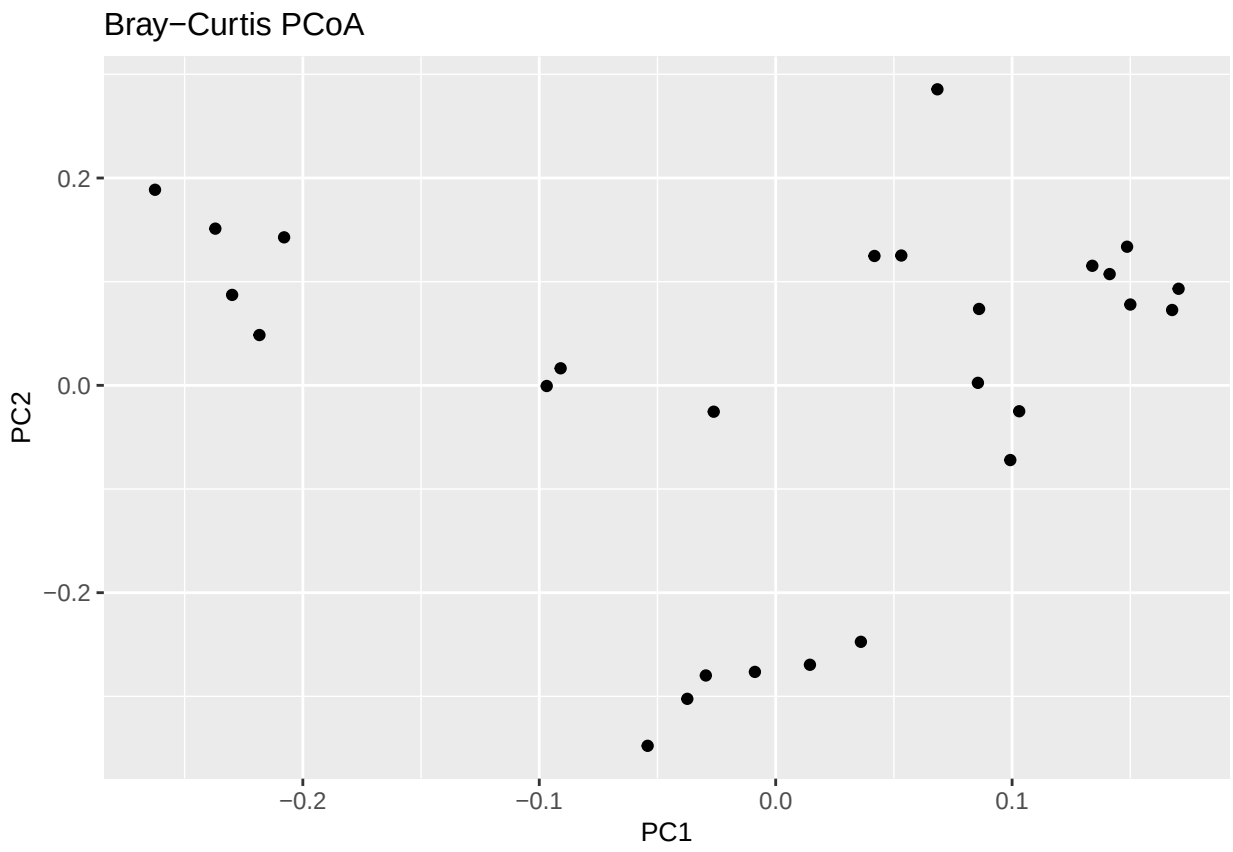
Let us next look at some concrete examples.

8.1.1 PCoA for ASV-level data with Bray-Curtis

Let us start with PCoA based on a Bray-Curtis dissimilarity matrix calculated at Genus level abundances.

```
# Pick the relative abundance table
rel_abund_assay <- assays(tse)$relabundance

# Calculates Bray-Curtis distances between samples. Because taxa is in
# columns, it is used to compare different samples. We transpose the
```

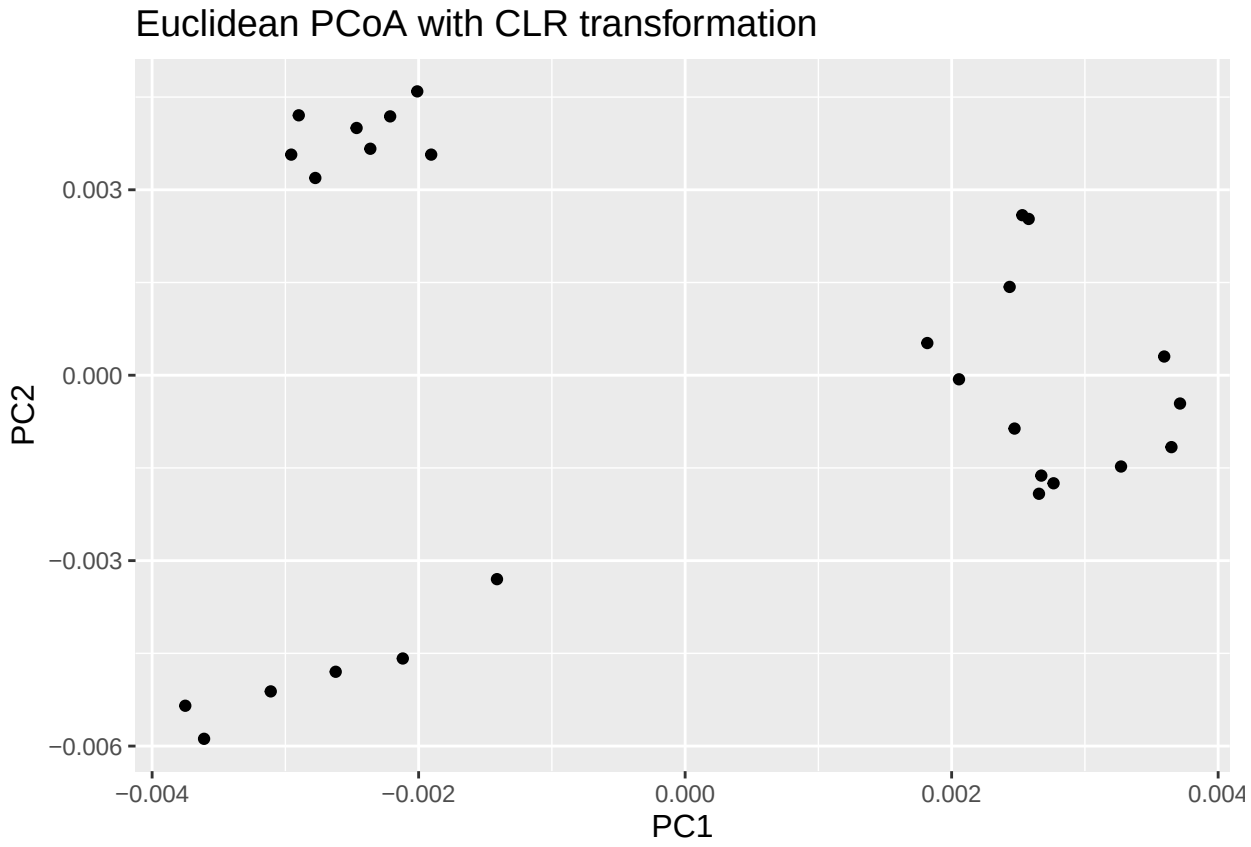
```r
# assay to get taxa to columns
bray_curtis_dist <- vegan::vegdist(t(rel_abund_assay), method = "bray")

# 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(title = element_text(size = 10)) # makes titles smaller

bray_curtis_plot
```



8.1.2 PCoA for ASV-level data with Aitchison distance

Now the same using Aitchison distance. This metric corresponds to Euclidean distances between CLR transformed sample abundance vectors.

```
# Does clr transformation. Pseudocount is added, because data contains zeros.
tse <- transformCounts(tse, method = "clr", pseudocount = 1)

# Gets clr table
clr_assay <- assays(tse)$clr

# Transposes it to get taxa to columns
clr_assay <- t(clr_assay)

# Calculates Euclidean distances between samples. Because taxa is in columns,
# it is used to compare different samples.
euclidean_dist <- vegan::vegdist(clr_assay, method = "euclidean")

# Does principal coordinate analysis
euclidean_pcoa <- ecodist::pco(euclidean_dist)

# Creates a data frame from principal coordinates
euclidean_pcoa_df <- data.frame(pcoa1 = euclidean_pcoa$vectors[,1],
                                pcoa2 = euclidean_pcoa$vectors[,2])

# Creates the plot
euclidean_plot <- ggplot(data = euclidean_pcoa_df, aes(x=pcoa1, y=pcoa2)) +
  geom_point() +
  labs(x = "PC1",
       y = "PC2",
       title = "Euclidean PCoA with CLR transformation") +
  theme(title = element_text(size = 12)) # makes titles smaller

euclidean_plot
```

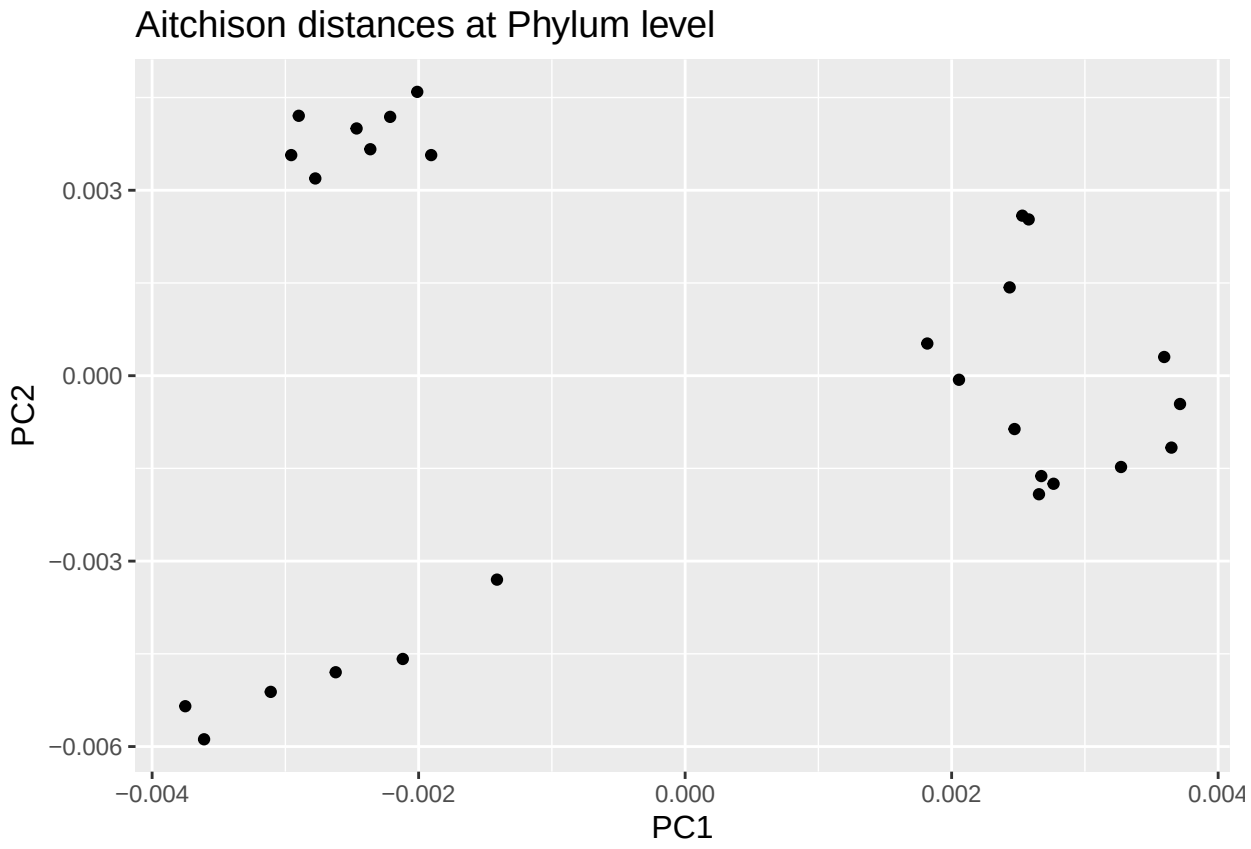


8.1.3 PCoA aggregated to Phylum level

We use again the Aitchison distances in this example but this time applied to the phylum level.

```
# Does clr transformation. Psuedocount is added, because data contains zeros.
tse_phylum <- transformCounts(tse_phylum, method = "clr", pseudocount = 1)

# Gets clr table
clr_phylum_assay <- assays(tse_phylum)$clr

# Transposes it to get taxa to columns
clr_phylum_assay <- t(clr_phylum_assay)

# Calculates Euclidean distances between samples. Because taxa is in columns,
# it is used to compare different samples.
euclidean_phylum_dist <- vegan::vegdist(clr_assay, method = "euclidean")

# Does principal coordinate analysis
euclidean_phylum_pcoa <- ecodist::pco(euclidean_phylum_dist)

# Creates a data frame from principal coordinates
euclidean_phylum_pcoa_df <- data.frame(
```

```
  pcoa1 = euclidean_phylum_pcoa$vectors[,1],
  pcoa2 = euclidean_phylum_pcoa$vectors[,2])

# Creates a plot
euclidean_phylum_plot <- ggplot(data = euclidean_phylum_pcoa_df,
  aes(x=pcoa1, y=pcoa2)) +
  geom_point() +
  labs(x = "PC1",
       y = "PC2",
       title = "Aitchison distances at Phylum level") +
  theme(title = element_text(size = 12)) # makes titles smaller

euclidean_phylum_plot
```

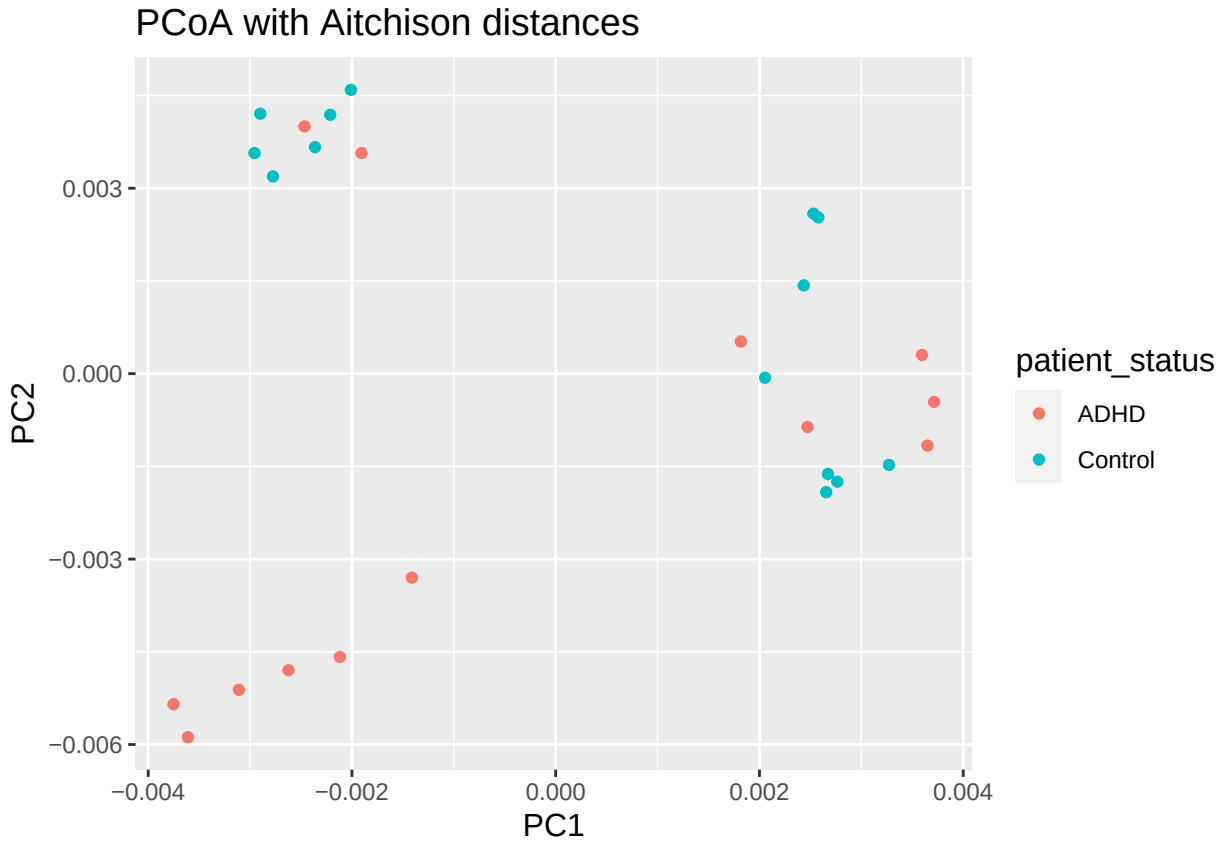


8.2 Highlighting external variables

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

8.2.1 Discrete grouping variable shown with colors

```
# Adds the variable we later use for coloring to the data frame
euclidean_patient_status_pcoa_df <- cbind(euclidean_pcoa_df,
                            patient_status = colData(tse)$patient_status)

# Creates a plot
euclidean_patient_status_plot <- ggplot(data = euclidean_patient_status_pcoa_df,
                                        aes(x=pcoa1, y=pcoa2,
                                            color = patient_status)) +
  geom_point() +
  labs(x = "PC1",
       y = "PC2",
       title = "PCoA with Aitchison distances") +
  theme(title = element_text(size = 12)) # makes titles smaller

euclidean_patient_status_plot
```



8.2.2 PCoA plot with continuous variable

We can also overlay a continuous variable on a PCoA plot. E.g. let us use the alcohol study dataset from curatedMetagenomicData. Perform PCoA and use the BMI as our continuous variable:

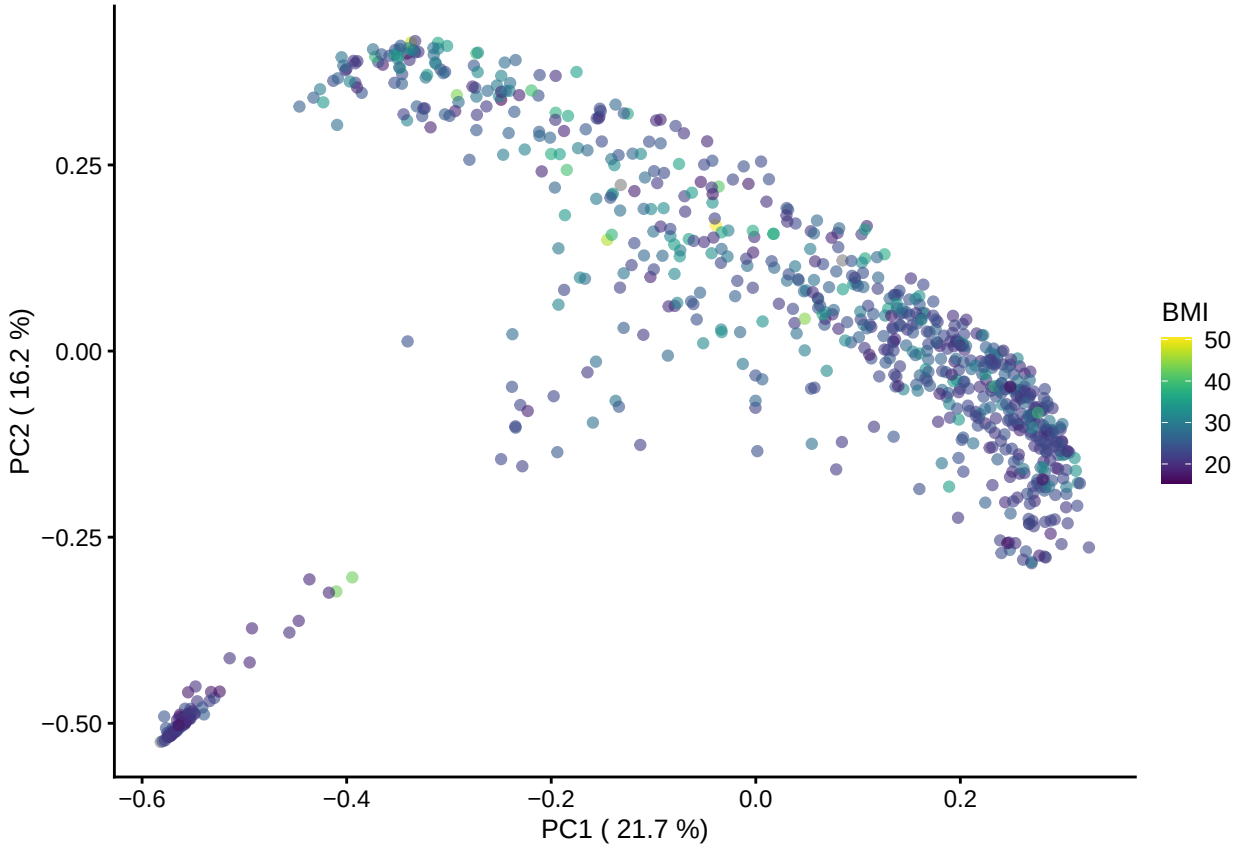
```
# Retrieving data as a TreeSummarizedExperiment object, and agglomerating to a genus level.
library(curatedMetagenomicData)
library(dplyr)
library(DT)
# Querying the data
tse_data <- sampleMetadata %>%
    filter(age >= 18) %>% # taking only data of age 18 or above
    filter(!is.na(alcohol)) %>% # excluding missing values
    returnSamples("relative_abundance")

# Agglomeration
tse_genus <- agglomerateByRank(tse_data, rank="Genus")
```

Performing PCoA with Bray-Curtis dissimilarity.

```
library(scater)
```

```
## Loading required package: scuttle
```

```
tse_genus <- runMDS(tse_genus, FUN = vegan::vegdist,
              name = "PCoA_BC", exprs_values = "relative_abundance")

# Retrieving the explained variance
e <- attr(reducedDim(tse_genus, "PCoA_BC"), "eig");
var_explained <- e/sum(e[e>0])*100

# Visualization
plot <-plotReducedDim(tse_genus,"PCoA_BC", colour_by = "BMI")+
  labs(x=paste("PC1 (",round(var_explained[1],1),"%)"),
       y=paste("PC2 (",round(var_explained[2],1),"%)"),
       color="")
plot
```



8.3 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(tse)$relabundance

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

# then we can perform the method
permanova_cohort <- vegan::adonis(rel_abund_assay ~ cohort,
                                  data = colData(tse),
                                  permutations = 9999)

# we can obtain a the p value for our predictor:
print(paste0("Different different cohorts and variance of abundance ",
```

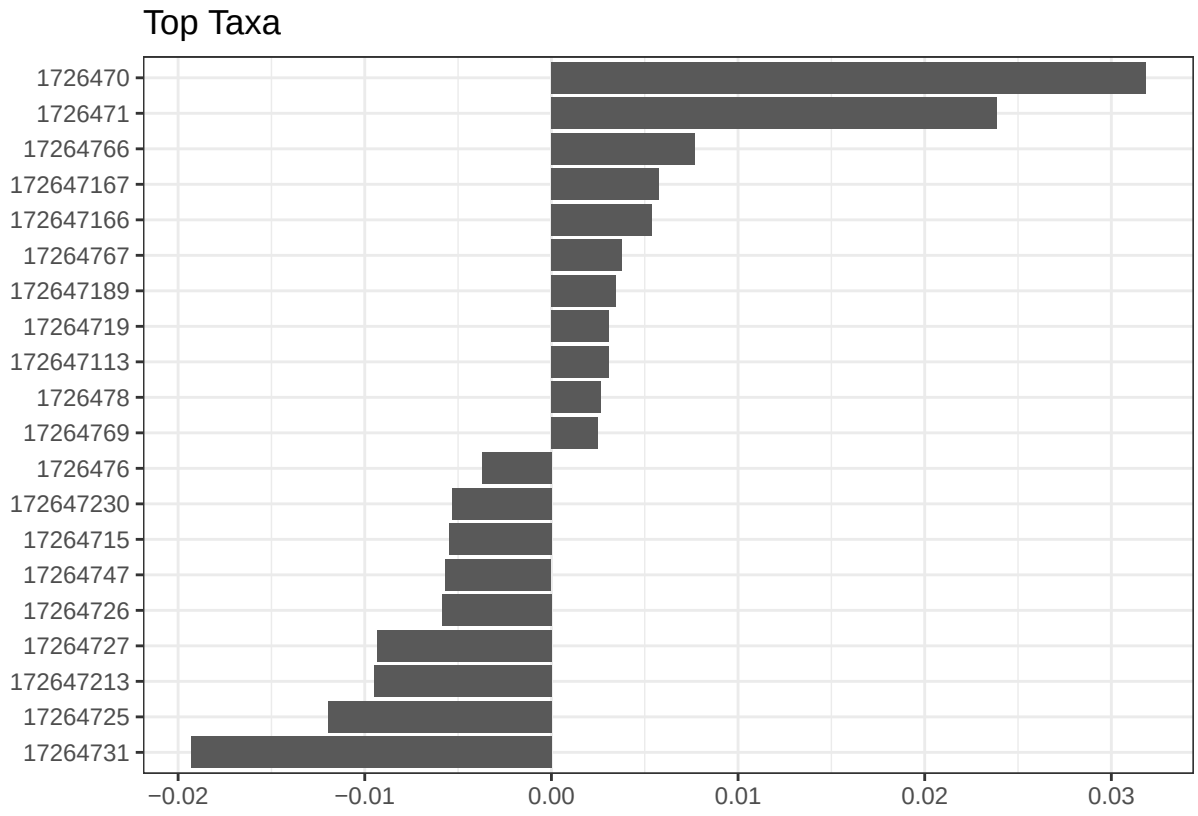
```
              "between samples, p-value: ",
              as.data.frame(permanova_cohort$aov.tab)["cohort", "Pr(>F)"]))
```

```
## [1] "Different different cohorts and variance of abundance between samples, p-value: 0.7489"
```

The cohort variable is not significantly associated with microbiota composition (p-value is over 0.05).

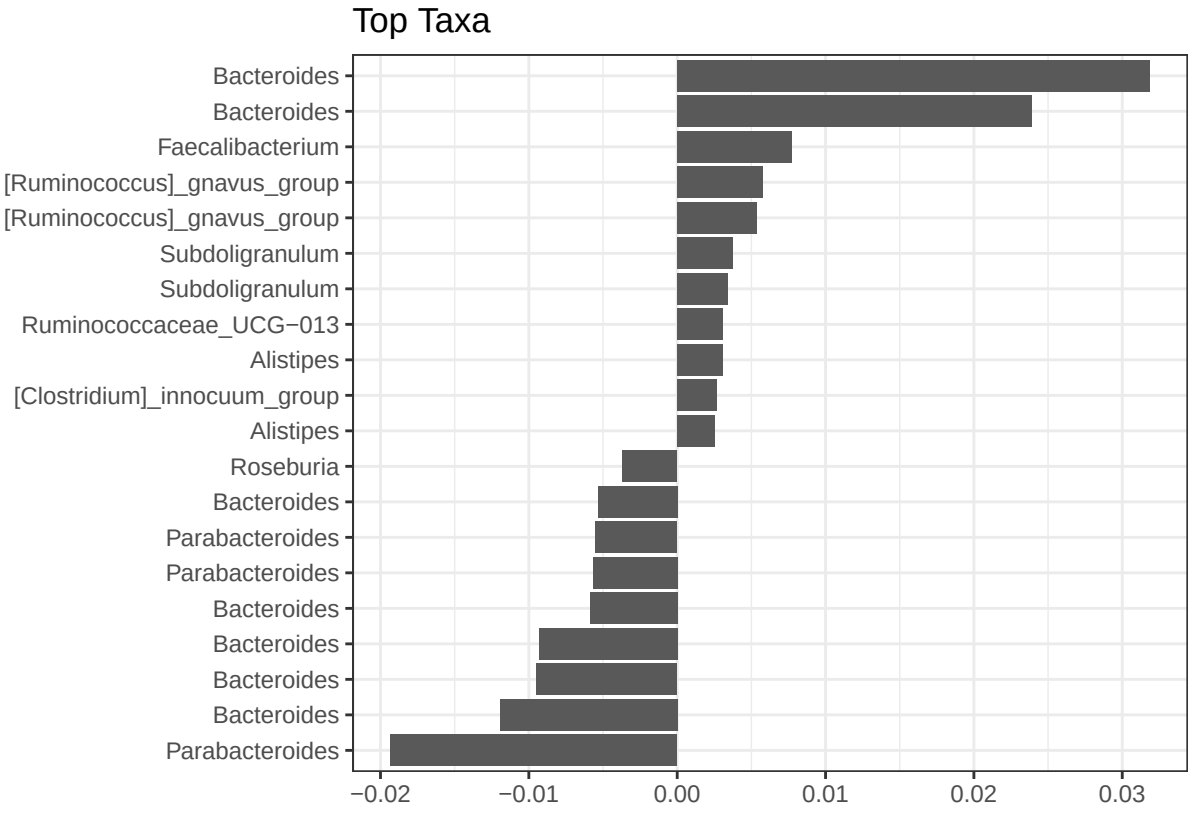
We can, however, visualize those taxa whose abundances drive the differences between cohorts. We first need to extract the model coefficients of taxa:

```
# Gets the coefficients
coef <- coefficients(permanova_cohort)["cohort1",]

# 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:

```
# Gets corresponding Genus level names and stores them to top.coef
names <- rowData(tse)[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
```



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.

8.4 Community typing

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

8.5 Exercises

- Visualize community variation with different methods (PCA, MDS, t-SNE…) by using the options in the alternative method, plotReducedDim OMA. Compare results obtained with different dissimilarities (Euclidean, Bray-Curtis, Unifrac..) and transformations (CLR, compositional..) of your own choice."
- Investigate the influence of the data transformations on statistical analysis: Visualize community variation with PCoA with the following options: 1) Bray-Curtis distances for compositional data; 2) Euclidean distances for CLR-transformed data.
- Community-level comparisons: Use PERMANOVA to investigate whether the community composition differs between two groups of individuals (e.g. males and females, or some other grouping of your choice). You can also include covariates such as age and gender, and see how this affects the results.

- Perform community typing for the data using the DMM method OMA
- Example Solutions

Chapter 9

Differential abundance analysis

Here, we analyse abundances with two different methods: **Wilcoxon test** (CLR), and **ANCOM-BC**. All of these test statistical differences between groups. We will analyse Genus level abundances.

We might want to first perform prevalence filtering to reduce the amount of multiple tests. In this particular dataset, all genera pass a prevalence threshold of 10%, therefore, we do not perform filtering.

9.1 Wilcoxon test

A Wilcoxon test estimates the difference in an outcome between two groups. It is a non-parametric alternative to a t-test, which means that the Wilcoxon test does not make any assumptions about the data.

Let's first combine the data for the testing purpose.

```
# Agglomerates data to Genus level
tse_genus <- agglomerateByRank(tse, rank = "Genus")

# Perform clr transformation. A Pseudocount of 1 needs to be added,
# because the data contains zeros and the clr transformation includes a
# log transformation.
tse_genus <- transformCounts(tse_genus, method = "clr", pseudocount = 1)

# Does transpose, so samples are in rows, then creates a data frame.
abundance_analysis_data <- data.frame(t(assay(tse_genus, "clr")))
# We will analyse whether abundances differ depending on the"patient_status".
# There are two groups: "ADHD" and "control".
# Let's include that to the data frame.
abundance_analysis_data <- cbind(
  abundance_analysis_data,
  patient_status = colData(tse_genus)$patient_status
)
```

Now we can start with the Wilcoxon test. We test all the taxa by looping through columns, and store individual p-values to a vector. Then we create a data frame from collected data.

The code below does the Wilcoxon test only for columns that contain abundances, not for columns that contain patient status.

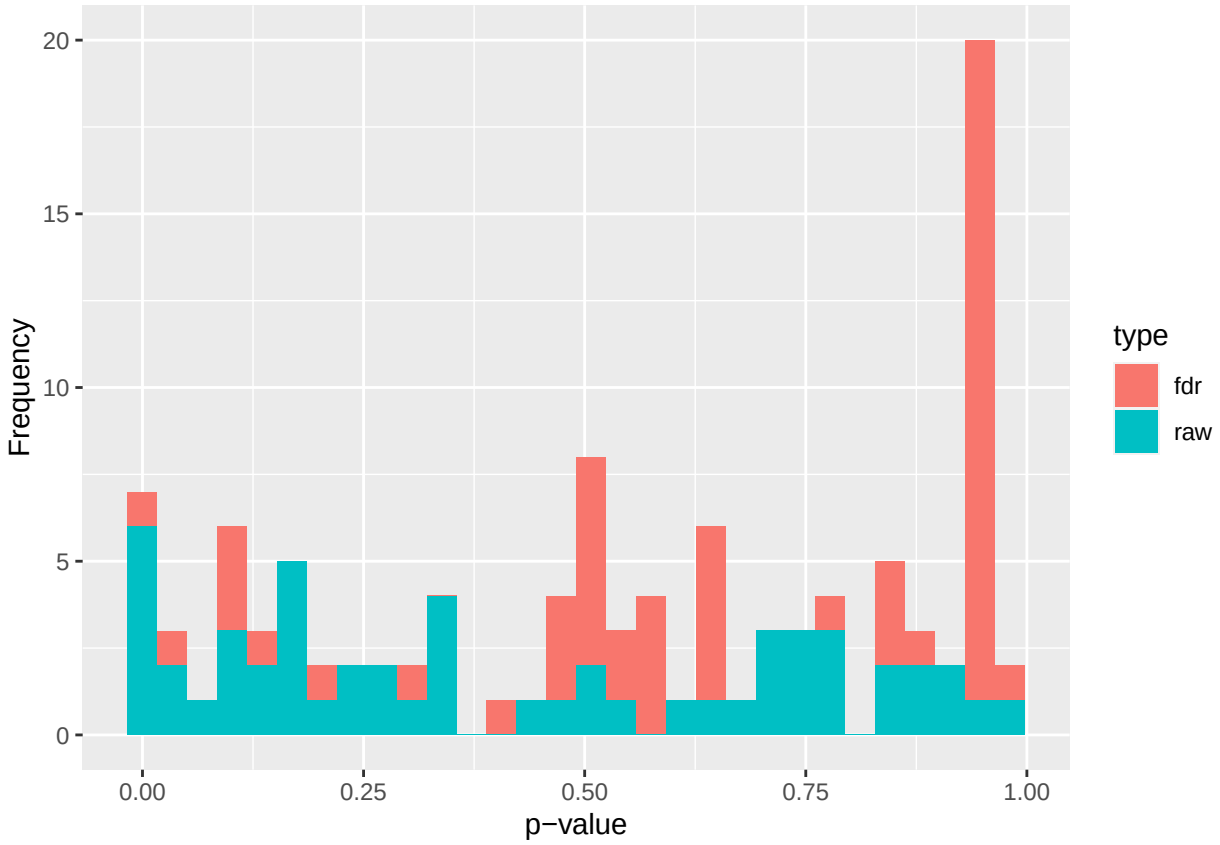
```
genera <- names(abundance_analysis_data[, !names(abundance_analysis_data) %in% "patient_status"])

wilcoxon_p <- c() # Initialize empty vector for p-values

# Do "for loop" over selected column names
for (i in genera) {

  result <- wilcox.test(abundance_analysis_data[, i] ~ patient_status,
                        data = abundance_analysis_data)

  # Stores p-value to the vector with this column name
  wilcoxon_p[[i]]  <- result$p.value

}

wilcoxon_p <- data.frame(taxa =  names(wilcoxon_p),
                         p_raw = unlist(wilcoxon_p))
```

Multiple tests were performed. These are not independent, so we need to adjust p-values for multiple testing. Otherwise, we would increase the chance of a type I error drastically depending on our p-value threshold. By applying a p-value adjustment, we can keep the false positive rate at a level that is acceptable. What is acceptable depends on our research goals. Here we use the fdr method, but there are several other methods as well.

```
wilcoxon_p$p_adjusted <- p.adjust(wilcoxon_p$p_raw, method = "fdr")
```

```
# prepare a dataframe to plot p values
df <- data.frame(x = c(wilcoxon_p$p_raw, wilcoxon_p$p_adjusted),
                 type=rep(c("raw", "fdr"),
        c(length(wilcoxon_p$p_raw),
          length(wilcoxon_p$p_adjusted))))

# make a histrogram of p values and adjusted p values
wilcoxon_plot <- ggplot(df) +
  geom_histogram(aes(x=x, fill=type)) +
  labs(x = "p-value", y = "Frequency")

wilcoxon_plot
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



9.2 ANCOM-BC

The analysis of composition of microbiomes with bias correction (ANCOM-BC) is a recently developed method for differential abundance testing. It is based on an earlier published approach. The former version of this method could be recommended as part of several approaches: A recent study compared several mainstream methods and found that among another method, ANCOM produced the most consistent results and is probably a conservative approach. Please note that based on this and other comparisons, no single method can be recommended across all datasets. Rather, it could be recommended to apply several methods and look at the overlap/differences.

As the only method, ANCOM-BC incorporates the so called *sampling fraction* into the model. The latter term could be empirically estimated by the ratio of the library size to the microbial load. Variations in this sampling fraction would bias differential abundance analyses if ignored. Furthermore, this method provides p-values, and confidence intervals for each taxon. It also controls the FDR and it is computationally simple to implement.

As we will see below, to obtain results, all that is needed is to pass a phyloseq object to the `ancombc()` function. Therefore, below we first convert our `tse` object to a `phyloseq` object. Then, we specify the formula. In this formula, other covariates could potentially be included to adjust for confounding. Please check the function documentation to learn about the additional arguments that we specify below. Also, see here for another example for more than 1 group comparison.

	patient_statusControl
172647198	FALSE
1726478	FALSE
172647201	FALSE
17264798	FALSE
172647195	FALSE
1726472	FALSE

```
# currently, ancombc requires the phyloseq format, but we can convert this easily
pseq <- makePhyloseqFromTreeSummarizedExperiment(tse)
pseq_genus <- phyloseq::tax_glom(pseq, taxrank = "Genus")

out = ancombc(
  phyloseq = pseq_genus,
  formula = "patient_status",
  p_adj_method = "fdr",
  zero_cut = 0.90, # by default prevalence filter of 10% is applied
  lib_cut = 0,
  group = "patient_status",
  struc_zero = TRUE,
  neg_lb = TRUE,
  tol = 1e-5,
  max_iter = 100,
  conserve = TRUE,
  alpha = 0.05,
  global = TRUE
)
res <- out$res
```

The object `out` contains all relevant information. Again, see the documentation of the function under **Value** for an explanation of all the output objects. Our question can be answered by looking at the `res` object, which now contains dataframes with the coefficients, standard errors, p-values and q-values. Conveniently, there is a dataframe `diff_abn`. Here, we can find all differentially abundant taxa. Below we show the first 6 entries of this dataframe:

```
knitr::kable(head(res$diff_abn)) %>% kableExtra::kable_styling("striped") %>%
  kableExtra::scroll_box(width = "100%")
```

In total, this method detects 14 differentially abundant taxa.

```
print(paste0("Wilcoxon test p-values under 0.05: ", sum(wilcoxon_p$p_adjusted<0.05, na.rm = TRUE), "/", lengt
```

```
## [1] "Wilcoxon test p-values under 0.05: 2/54"
```

```
print(paste0("ANCOM p-values under 0.05: ", sum(out$res$diff_abn$patient_statusControl), "/", length(out$res$
```

```
## [1] "ANCOM p-values under 0.05: 14/49"
```

9.3 Comparison of the methods

TO DO: More difference abundance methods are upcoming. Comparison will be made later.

9.4 Comparison of abundance

In previous steps, we got information which taxa vary between ADHD and control groups. Let's plot those taxa in the boxplot, and compare visually if abundances of those taxa differ in ADHD and control samples. For comparison, let's plot also taxa that do not differ between ADHD and control groups.

Let's first gather data about taxa that have highest p-values.

```
#we used the results from the wilcoxn test
df <- wilcoxon_p
# renaming row names and taxa column
df$taxa <- rownames(assay(tse_genus,"clr"))

# There are some taxa that do not include Genus level information. They are
# excluded from analysis.
# str_detect finds if the pattern is present in values of "taxon" column.
# Subset is taken, only those rows are included that do not include the pattern.
df <- df[ !stringr::str_detect(df$taxa, "Genus:uncultured"), ]

# Sorts p-values in decreasing order. Takes 3 first ones. Takes those rows that match
# with p-values. Takes taxa.
highest3 <- df[df$p_adjusted %in% sort(df$p_adjusted, decreasing = TRUE)[1:3], ]$taxa

# From clr transformed table, takes only those taxa that had highest p-values
highest3 <- assay(tse_genus, "clr")[highest3, ]

# Transposes the table
highest3 <- t(highest3)

# Adds colData that includes patient status infomation
highest3 <- data.frame(highest3, as.data.frame(colData(tse_genus)))

# Some taxa names are that long that they don't fit nicely into title. So let's add there
# a line break after e.g. "Genus". Here the dot after e.g. Genus is replaced with
# ": \n"
colnames(highest3)[1:3] <- lapply(colnames(highest3)[1:3], function(x){
  # Replaces the first dot
  temp <- stringr::str_replace(x, "[.]", ": ")

  # Replace all other dots and underscores with space
  temp <- stringr::str_replace_all(temp, c("[.]" = " ", "_" = " "))

  # Adds line break so that 25 characters is the maximal width
  temp <- stringr::str_wrap(temp, width = 25)
})
```

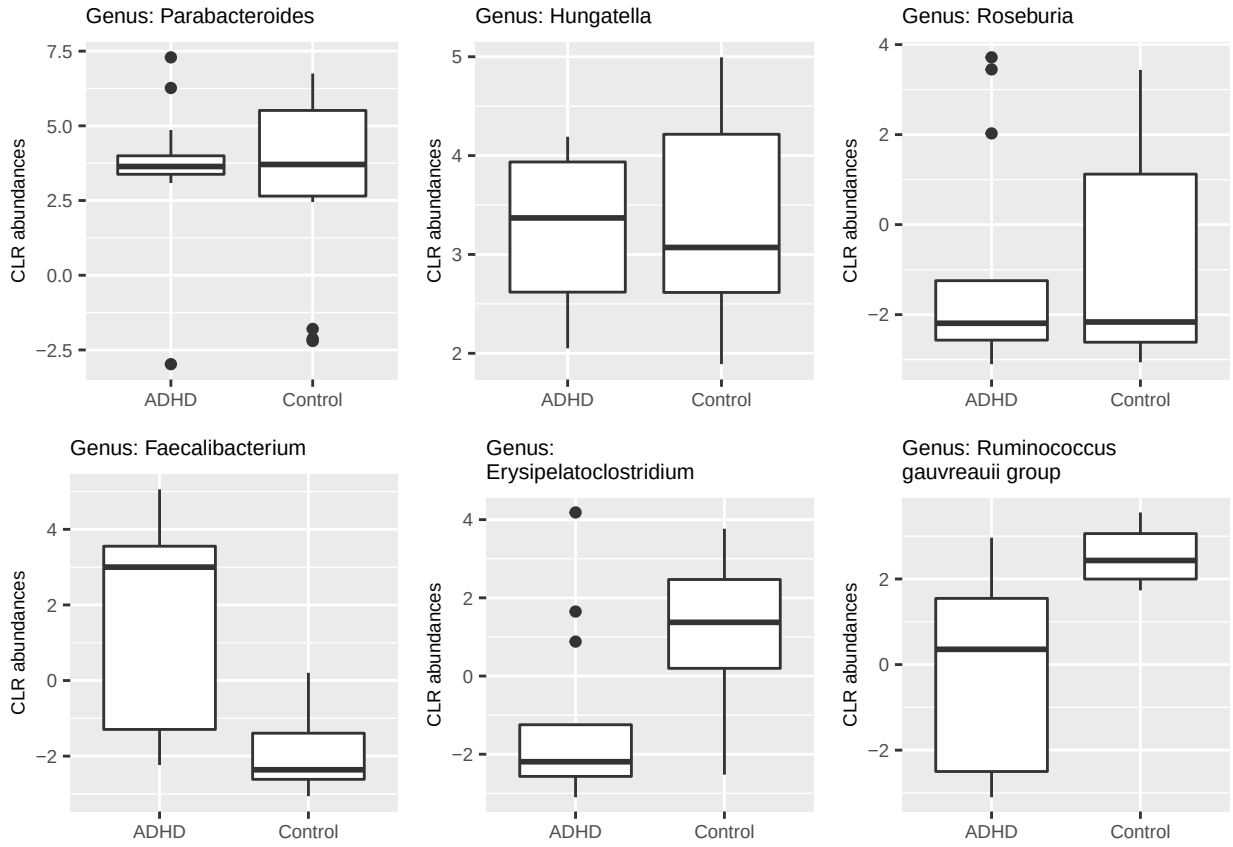
Next, let's do the same but for taxa with lowest p-values.

```
# Sorts p-values in increasing order. Takes 3rd first ones. Takes those rows that match
# with p-values. Takes taxa.
lowest3 <- df[df$p_adjusted %in% sort(df$p_adjusted, decreasing = FALSE)[1:3], ]$taxa


# From clr transformed table, takes only those taxa that had lowest p-values
lowest3 <-assay(tse_genus, "clr")[lowest3, ]

# Transposes the table
lowest3 <- t(lowest3)

# Adds colData that includes patient status infomation
lowest3 <- data.frame(lowest3, as.data.frame(colData(tse_genus)))

# Some taxa names are that long that they don't fit nicely into title. So let's add there
# a line break after e.g. "Genus". Here the dot after e.g. Genus is replaced with
# ": \n"
colnames(lowest3)[1:3] <- lapply(colnames(lowest3)[1:3], function(x){
  # Replaces the first dot
  temp <- stringr::str_replace(x, "[.]", ": ")

  # Replace all other dots and underscores with space
  temp <- stringr::str_replace_all(temp, c("[.]" = " ", "_" = " "))

  # Adds line break so that 25 characters is the maximal width
  temp <- stringr::str_wrap(temp, width = 25)
})
```

Then we can plot these six different taxa. Let's arrange them into the same picture.

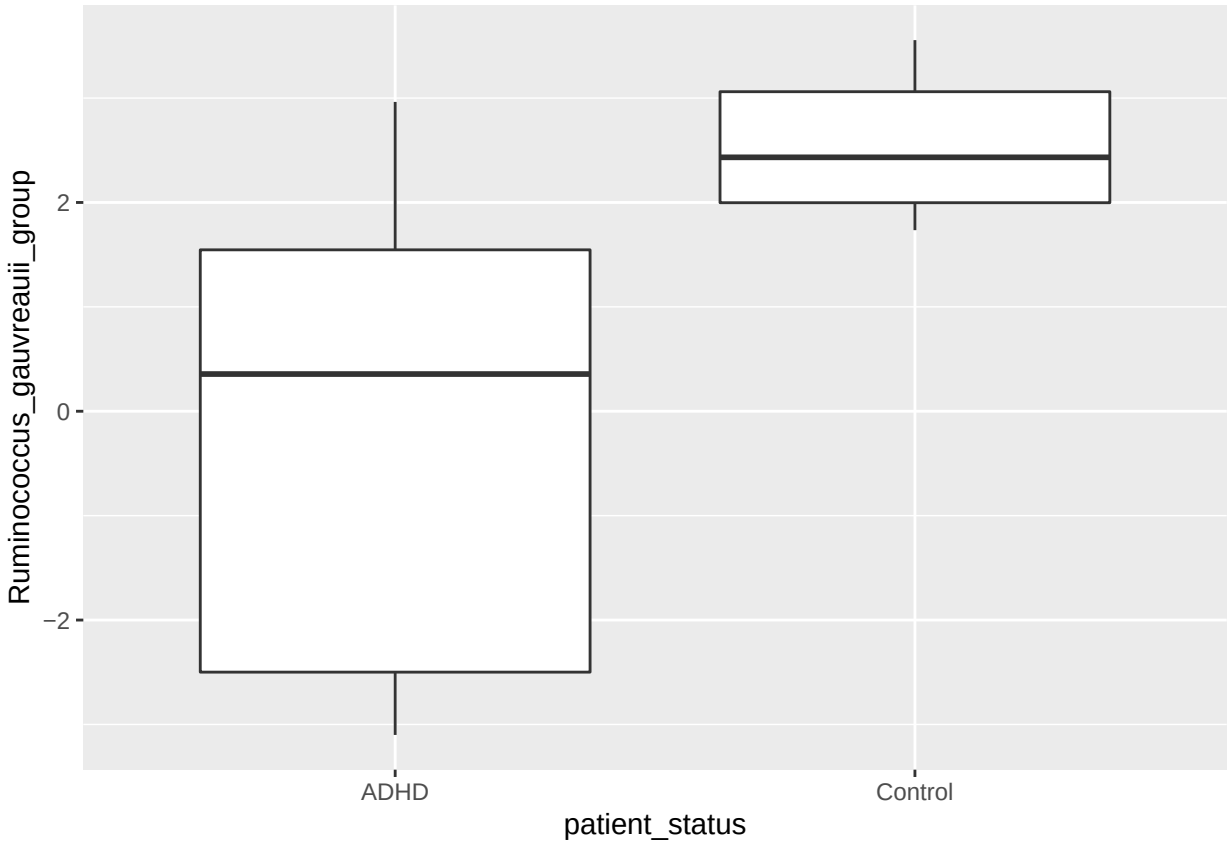
```
# Puts plots in the same picture
gridExtra::grid.arrange(

  # Plot 1
  ggplot(highest3, aes(x = patient_status, y = highest3[,1])) +
    geom_boxplot() +
    ylab("CLR abundances") + # y axis title
    ggtitle(names(highest3)[1]) + # main title
    theme(title = element_text(size = 7),
          axis.text = element_text(size = 7),
          axis.title.x=element_blank()), # makes titles smaller, removes x axis title

  # Plot 2
  ggplot(highest3, aes(x = patient_status, y = highest3[,2])) +
    geom_boxplot() +
    ylab("CLR abundances") + # y axis title
    ggtitle(names(highest3)[2]) + # main title
    theme(title = element_text(size = 7),
          axis.text = element_text(size = 7),
```

```
          axis.title.x=element_blank()), # makes titles smaller, removes x axis title

  # Plot 3
  ggplot(highest3, aes(x = patient_status, y = highest3[,3])) +
    geom_boxplot() +
    ylab("CLR abundances") + # y axis title
    ggtitle(names(highest3)[3]) + # main title
    theme(title = element_text(size = 7),
          axis.text = element_text(size = 7),
          axis.title.x=element_blank()), # makes titles smaller, removes x axis title

  # Plot 4
  ggplot(lowest3, aes(x = patient_status, y = lowest3[,1])) +
    geom_boxplot() +
    ylab("CLR abundances") + # y axis title
    ggtitle(names(lowest3)[1]) + # main title
    theme(title = element_text(size = 7),
          axis.text = element_text(size = 7),
          axis.title.x=element_blank()), # makes titles smaller, removes x axis title

  # Plot 5
  ggplot(lowest3, aes(x = patient_status, y = lowest3[,2])) +
    geom_boxplot() +
    ylab("CLR abundances") + # y axis title
    ggtitle(names(lowest3)[2]) + # main title
    theme(title = element_text(size = 7),
          axis.text = element_text(size = 7),
          axis.title.x=element_blank()), # makes titles smaller, removes x axis title

  # Plot 6
  ggplot(lowest3, aes(x = patient_status, y = lowest3[,3])) +
    geom_boxplot() +
    ylab("CLR abundances") + # y axis title
    ggtitle(names(lowest3)[3]) + # main title
    theme(title = element_text(size = 7),
          axis.text = element_text(size = 7),
          axis.title.x=element_blank()), # makes titles smaller, removes x axis title

  # 3 columns and 2 rows
  ncol = 3,
  nrow = 2
)
```



We plotted those taxa that have the highest and lowest p values according to DESeq2. Can you create a plot that shows the difference in abundance in "[Ruminococcus]_gauvreauii_group", which is the other taxon that was identified by all tools. Try for yourself! Below you find one way how to do it.

```
select(
  abundance_analysis_data,
  patient_status,
  Ruminococcus_gauvreauii_group = contains("gauvreauii_group")) %>%
  ggplot(aes(patient_status, Ruminococcus_gauvreauii_group)) +
  geom_boxplot()
```


Ruminococcus_gauvreauii_group
2
0
−2
ADHD
Control
patient_status

Chapter 10

Study material

10.1 Lecture slides

See slides.

10.2 R programming resources

R progamming

- Base R cheat sheet
- Base R cheat sheet 2
- Cheat sheet collection

Visualization

- ggplot2 cheat sheet
- R graphics cookbook

Rmarkdown

- Rmarkdown tips

RStudio

- RStudio cheat sheet

10.3 Resources for TreeSummarizedExperiment

SummarizedExperiment

- Publication
- Project page

TreeSummarizedExperiment

- Publication
- Project page

SingleCellExperiment

- Publication
- Project page

10.4 Resources for phyloseq

- List of R tools for microbiome analysis
- phyloseq
- microbiome tutorial
- microbiomeutilities
- Bioconductor Workflow for Microbiome Data Analysis: from raw reads to community analyses (Callahan et al. F1000, 2016).

10.5 Further reading

- Data Analysis and Visualization in R for Ecologists by Data Carpentry
- Modern Statistics for Modern Biology. Holmes & Huber (2018) for background in statistical analysis
- Microbiome Data Science. Shetty & Lahti, 2019

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
```

```
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

## Loading required package: GenomeInfoDb

## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians

## Loading required package: SingleCellExperiment

## Loading required package: TreeSummarizedExperiment

## Loading required package: Biostrings

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':
##
##     strsplit

## Loading required package: ggplot2

## Loading required package: ggraph

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:Biostrings':
##
##     collapse, intersect, setdiff, setequal, union

## The following object is masked from 'package:XVector':
##
##     slice
```

```
## The following object is masked from 'package:Biobase':
##
##     combine

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union

## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following object is masked from 'package:matrixStats':
##
##     count

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Chapter 11

Miscellaneous material

11.1 Shapiro-Wilk test

If necessary, it is possible to assess normality of the data with Shapiro-Wilk test.

```
# Does Shapiro-Wilk test. Does it only for columns that contain abundances, not for
# column that contain Groups.

normality_test_p <- c()

for (column in
     abundance_analysis_data[, !names(abundance_analysis_data) %in% "patient_status"]){
  # Does Shapiro-Wilk test
  result <- shapiro.test(column)

  # Stores p-value to vector
  normality_test_p <- c(normality_test_p, result$p.value)
}

print(paste0("P-values over 0.05: ", sum(normality_test_p>0.05), "/",
             length(normality_test_p)))
```

```
## [1] "P-values over 0.05: 7/54"
```

11.2 Deseq details

1. Raw counts are normalized by log-based scaling.

2. Taxa-wise variance is estimated. These values tell how much each taxa varies between samples.

3. A curve is fitted over all those taxa-wise variance estimates that we got in the last step.
   This model tells how big the variance is in a specific abundance level.

4. The model is used to shrink those individual variance estimates to avoid the effect of, e.g., small sample size and higher variance. This reduces the likelihood to get false positives.

5. Variance estimates are used to compare different groups. We receive a result that shows whether the variance is explained by groups.

Chapter 12

Exercise Solutions

This section includes exemplary solutions to the exercises presented earlier.

12.1 Section 5

Link:

- Rmd

12.2 Section 6

Links:

- Rmd

12.3 Section 7

Links:

- Rmd

12.4 Section 8

Links:

- Rmd

Bibliography

Borman, T., Eckermann, H., Benchraka, C., and Lahti, L. (2021). *Introduction to microbiome data science with miaverse. Radboud Summer School.*