Microbiome data science with R/Bioconductor

Welcome to Radboud Summer School, July 2022

2022-07-11

Contents

Chapter 1

Overview

1.1 Contents and learning goals

This course will focus on **microbiome data analysis with R/Bioconductor**, a popular open source environment for scientific data analysis. You will get an overview of the reproducible data analysis workflows in microbiome research, with a focus on gut-brain axis studies.

After the course you will know how to approach new tasks in the analysis of taxonomic profiling data by taking advantage of available documentation and R tools.

The teaching follows the open online documentation created by the course teachers, extending the online book Orchestrating Microbiome Analysis (https://microbiome.github.io/OMA). The openly licensed teaching material will be available online during and after the course, following Finnish national recommendations on open education.

The training material walks you through the standard steps of biomedical data analysis covering data access, exploration, analysis, visualization, reproducible reporting, and best practices in open science. We will teach generic data analytical skills that are applicable to common data analysis tasks encountered in modern omics research. The teaching format allows adaptations according to the student's learning speed.

1.2 Schedule and organizers

Format In-person course. For detailed schedule, see the course website.

Venue University of Radboud. July 11-15, 2022.

Expected background

Target audience The course is primarily designed for advanced MSc and PhD students, Postdocs, and biomedical researchers who wish to learn new skills in scientific programming and biomedical data analysis. Academic students and researchers encouraged to apply. Priority will be given for local students. Some earlier experience with R or another programming language is recommended.

Preparation Advance preparation is expected. Online support is available. See section 3 for instructions.

1.3 Acknowledgments

Citation We thank all developers and contributors who have contributed open resources that supported the development of the training material. Kindly cite the course material as Borman et al. (2022)

Contact See https://microbiome.github.io

License and source code

Chapter 2

Code of Conduct

The Bioconductor community values an open approach to science that promotes the

- sharing of ideas, code, software and expertise
- collaboration
- diversity and inclusivity
- a kind and welcoming environment
- community contributions

We value your attendance and participation at Bioconductor events and in our community.

For the full version, enforcement, and reporting instructions, see the Bioconductor code of conduct.

Chapter 3

Getting started

3.1 Checklist (before the course)

3.1.1 Computer setup and installations

Setting up the system on your own computer is required to follow the full course and will be useful for later use if you intent to analyze microbiome data on your computer. The required software:

- R (version >4.2.0)
- 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 (see Section 3.3)
- After a successful installation you can start with the case study examples in this training material

3.2 Support and resources

- We recommend to have a look at the additional reading tips and try out online material listed in Section 10.
- **You can run the workflows by simply copy-pasting the examples.** For further, advanced material, you can test and modify further examples from the online book, and apply these techniques to your own data.
- Online support on installation and other matters, join us at Gitter

3.3 Installing and loading the required R packages

You may need the examples from this subsection if you are installing the environment on your own computer. If you need to add new packages, you can modify the examples below.

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

Download the file pkgs.csv. This contains the list of packages that we recommend to preinstall. This can be done with the following code.

```r
# List of packages that we need
pkg <- read.csv("pkgs.csv")[,1]

# List packages that are already installed
pkg_already_installed <- pkg[ pkg %in% installed.packages() ]

# List remaining packages that need to be installed
packages_to_install <- setdiff(pkg, pkg_already_installed)
```

```r
# If there are packages that need to be installed, install them
if( length(packages_to_install) ) {
   BiocManager::install(packages_to_install)
}
```

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
# Loading all packages into session. Returns true if package was successfully loaded.
loaded <- sapply(pkg, require, character.only = TRUE)
as.data.frame(loaded)
```

Chapter 4

Reproducible reporting with Rmarkdown

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

- 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
- Create a Rmarkdown template in RStudio, and render it into a document (markdown, PDF, docx or other format). In case you are new to Rmarkdown Rstudio provides resources to learn about the use cases and the basics of Rmarkdown.
- Further examples are tips for Rmarkdown are available in the online tutorial to reproducible reporting by Dr. C Titus Brown.

Chapter 5

Importing microbiome data

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

5.1 Data access

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.

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* container, 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* (TreeSE) Huang et al. (2020).

The data container provides the basis for downstream data analysis in the *miaverse* data science framework.

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).

5.2.1 Example solution

Let us first import the biom file into R / TreeSE container.

```r
# Load the mia R package
library(mia)

# Defining file paths
## Biom file (taxonomic profiles)
biom_file_path <- "data/Tengeler2020/Aggregated_humanization2.biom"

# Import the data into SummarizedExperiment container
se <- loadFromBiom(biom_file_path)

# Convert this data to TreeSE container (no direct importer exists)
tse <- as(se, "TreeSummarizedExperiment")
```

Check and clean up rowData (information on the taxonomic features).

```r
# Investigate the rowData of this data object
print(head(rowData(tse)))
```

```
## DataFrame with 6 rows and 6 columns
##              taxonomy1          taxonomy2           taxonomy3
##            <character>        <character>         <character>
## 1726470  "k__Bacteria   p__Bacteroidetes      c__Bacteroidia
## 1726471  "k__Bacteria   p__Bacteroidetes      c__Bacteroidia
## 17264731 "k__Bacteria   p__Bacteroidetes      c__Bacteroidia
## 17264726 "k__Bacteria   p__Bacteroidetes      c__Bacteroidia
## 1726472  "k__Bacteria p__Verrucomicrobia c__Verrucomicrobiae
## 17264724 "k__Bacteria   p__Bacteroidetes      c__Bacteroidia
##                   taxonomy4              taxonomy5           taxonomy6
##                 <character>            <character>         <character>
## 1726470       o__Bacteroidales      f__Bacteroidaceae     g__Bacteroides"
## 1726471       o__Bacteroidales      f__Bacteroidaceae     g__Bacteroides"
## 17264731      o__Bacteroidales  f__Porphyromonadaceae g__Parabacteroides"
## 17264726      o__Bacteroidales      f__Bacteroidaceae     g__Bacteroides"
## 1726472  o__Verrucomicrobiales f__Verrucomicrobiaceae     g__Akkermansia"
## 17264724      o__Bacteroidales      f__Bacteroidaceae     g__Bacteroides"
```

```r
# We notice that the rowData fields do not have descriptibve names.
# Hence, let us rename the columns in rowData
names(rowData(tse)) <- c("Kingdom", "Phylum", "Class", "Order",
                        "Family", "Genus")

# We also notice that the taxa names are of form "c__Bacteroidia" etc.
# Goes through the whole DataFrame. Removes '.*[kpcofg]__' from strings, where [kpcofg]
# is any character from listed ones, and .* any character.
rowdata_modified <- BiocParallel::bplapply(rowData(tse),
                                           FUN = stringr::str_remove,
                                           pattern = '.*[kpcofg]__')

# Genus level has additional '\"', so let's delete that also
rowdata_modified <- BiocParallel::bplapply(rowdata_modified,
                                           FUN = stringr::str_remove,
                                           pattern = '\"')

# rowdata_modified is a list, so convert this back to DataFrame format.
# and assign the cleaned data back to the TSE rowData
rowData(tse) <- DataFrame(rowdata_modified)

# Recheck rowData after the modifications
print(head(rowData(tse)))
```

```
## DataFrame with 6 rows and 6 columns
##              Kingdom          Phylum             Class              Order
##          <character>     <character>       <character>        <character>
## 1726470     Bacteria   Bacteroidetes       Bacteroidia      Bacteroidales
## 1726471     Bacteria   Bacteroidetes       Bacteroidia      Bacteroidales
## 17264731    Bacteria   Bacteroidetes       Bacteroidia      Bacteroidales
## 17264726    Bacteria   Bacteroidetes       Bacteroidia      Bacteroidales
## 1726472     Bacteria Verrucomicrobia Verrucomicrobiae Verrucomicrobiales
## 17264724    Bacteria   Bacteroidetes       Bacteroidia      Bacteroidales
##                       Family           Genus
##                  <character>     <character>
## 1726470       Bacteroidaceae     Bacteroides
## 1726471       Bacteroidaceae     Bacteroides
## 17264731  Porphyromonadaceae Parabacteroides
## 17264726      Bacteroidaceae     Bacteroides
## 1726472  Verrucomicrobiaceae     Akkermansia
## 17264724      Bacteroidaceae     Bacteroides
```

Next let us add sample information (colData) to our TreeSE object.

```
## Sample phenodata
sample_meta_file_path <- "data/Tengeler2020/Mapping_file_ADHD_aggregated.csv"

# Check what type of data it is
# read.table(sample_meta_file_path)

# It seems like a comma separated file and it does not include headers
# Let us read the file; note that sample names are in the first column
sample_meta <- read.table(sample_meta_file_path, sep=",", header=FALSE, row.names=1)

# Check the data
print(head(sample_meta))
```

```
##        V2       V3            V4   V5
## 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
```

```
# Add headers for the columns (as they seem to be missing)
colnames(sample_meta) <- c("patient_status", "cohort",
                           "patient_status_vs_cohort", "sample_name")

# Add this sample data to colData of the taxonomic data object
# Note that the data must be given in a DataFrame format (required for our purposes)
colData(tse) <- DataFrame(sample_meta)

# Check the colData after modifications
print(head(colData(tse)))
```

```
## DataFrame with 6 rows and 4 columns
##      patient_status      cohort patient_status_vs_cohort sample_name
##         <character> <character>              <character> <character>
## 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
```

Add phylogenetic tree (rowTree).

```
## Phylogenetic tree
tree_file_path <- "data/Tengeler2020/Data_humanization_phylo_aggregation.tre"

# Read the tree file
tree <- ape::read.tree(tree_file_path)

# Add tree to rowTree
rowTree(tse) <- tree
```

We can save the data object as follows.

```
saveRDS(tse, file="tse.rds")
```

5.2.2 Loading readily processed data

Alternatively, you can just load the already processed data set.

By using this readily processed data set we can skip the data import step, and assume that the data has already been appropriately preprocessed and available in the TreeSE container.

```
tse <- readRDS("data/Tengeler2020/tse.rds")
```

In addition to our example data, further demonstration data sets are available in the TreeSE data container: see OMA demo data sets.

```
## 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: MultiAssayExperiment

## 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.

Chapters 4-6 in OMA provide many more examples on exploring and manipulating microbiome data.

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%")
```

	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

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.

```
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.

	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

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.

```r
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.

```r
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.

```r
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

	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		

```
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.

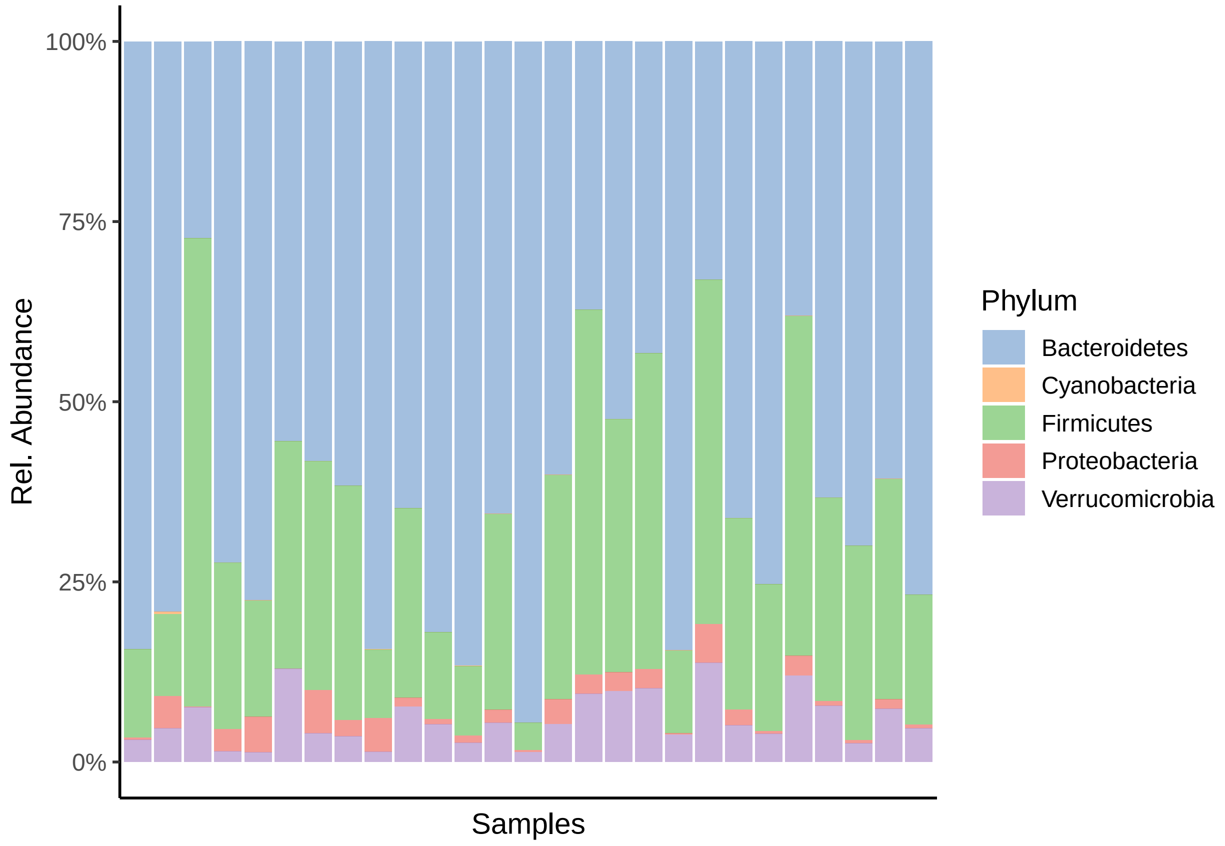
6.2 Visualization

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
```

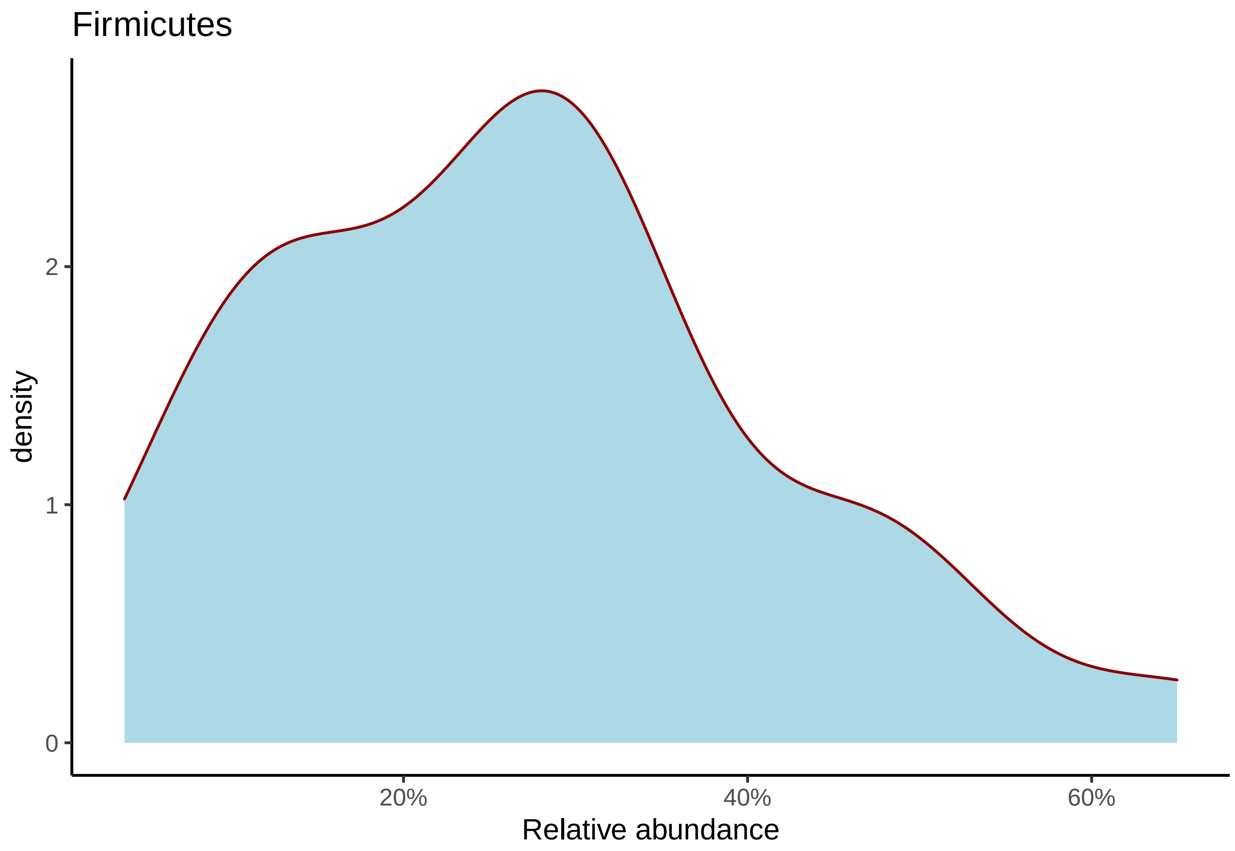


Density plot shows the overall abundance distribution for a given taxonomic group. Let us check the relative abundance of Firmicutes across the sample collection. The density plot is a smoothened version of a standard histogram.

The plot shows peak abundances around 30 %.

```r
# 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: subsetting in OMA
- **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.

6.4 Exploring data: example solutions

- **Abundance table** Retrieve the taxonomic abundance table from the example data set (TSE object).

```
# We show only part of it
assays(tse)$counts[1:6,1:6]
```

```
##           A110   A12  A15  A19  A21  A23
## 1726470  17722 11630    0 8806 1740 1791
## 1726471  12052     0 2679 2776  540  229
## 17264731     0   970    0  549  145    0
## 17264726     0  1911    0 5497  659    0
## 1726472   1143  1891 1212  584   84  700
## 17264724     0  6498    0 4455  610    0
```

- How many different samples and genus-level groups this phyloseq object has?

```
dim(rowData(tse))
```

```
## [1] 151   6
```

- What is the maximum abundance of Akkermansia in this data set?

```
# Agglomerating to Genus level
tse_genus <- agglomerateByRank(tse,rank="Genus")
# Retrieving the count for Akkermansia
Akkermansia_abund <- assays(tse_genus)$count["Genus:Akkermansia",]
max(Akkermansia_abund)
```

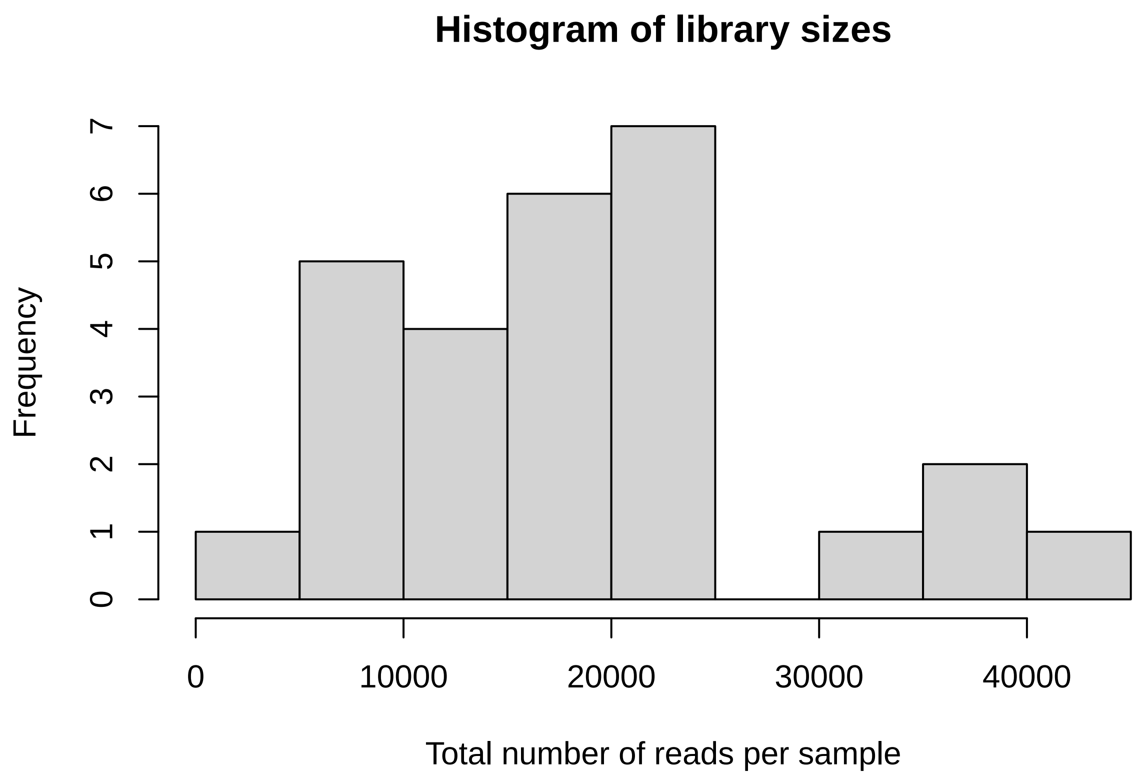
```
## [1] 2535
```

- Draw a histogram of library sizes (total number of reads per sample).

```
library(scater)
```

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

```
tse <- addPerCellQC(tse) # adding a new column "sum" to colData, of total counts/sample.
hist(colData(tse)$sum, xlab = "Total number of reads per sample", main = "Histogram of library sizes")
```



*

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?

```
head(rowData(tse))
```

```
## DataFrame with 6 rows and 6 columns
##                Kingdom         Phylum         Class          Order
##            <character>    <character>   <character>    <character>
## 1726470       Bacteria  Bacteroidetes   Bacteroidia  Bacteroidales
## 1726471       Bacteria  Bacteroidetes   Bacteroidia  Bacteroidales
## 17264731      Bacteria  Bacteroidetes   Bacteroidia  Bacteroidales
## 17264726    Bacteria   Bacteroidetes      Bacteroidia      Bacteroidales
## 1726472     Bacteria Verrucomicrobia Verrucomicrobiae Verrucomicrobiales
## 17264724    Bacteria   Bacteroidetes      Bacteroidia      Bacteroidales
##                       Family           Genus
##                  <character>     <character>
## 1726470       Bacteroidaceae     Bacteroides
## 1726471       Bacteroidaceae     Bacteroides
## 17264731  Porphyromonadaceae Parabacteroides
## 17264726      Bacteroidaceae     Bacteroides
## 1726472  Verrucomicrobiaceae     Akkermansia
## 17264724      Bacteroidaceae     Bacteroides
```

```
taxonomyRanks(tse) # The taxonomic ranks available
```

```
## [1] "Kingdom" "Phylum"  "Class"   "Order"   "Family"  "Genus"
```

```
unique(rowData(tse)["Phylum"]) # phylum-level groups
```

```
## DataFrame with 5 rows and 1 column
##                    Phylum
##               <character>
## 1726470     Bacteroidetes
## 1726472   Verrucomicrobia
## 17264729   Proteobacteria
## 172647189      Firmicutes
## 17264742    Cyanobacteria
```

```
length(unique(rowData(tse)["Phylum"])[,1]) # number of phylum-level groups
```

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

- **Sample metadata** Retrieve sample metadata. How many patient groups this data set has? Draw a histogram of sample diversities.

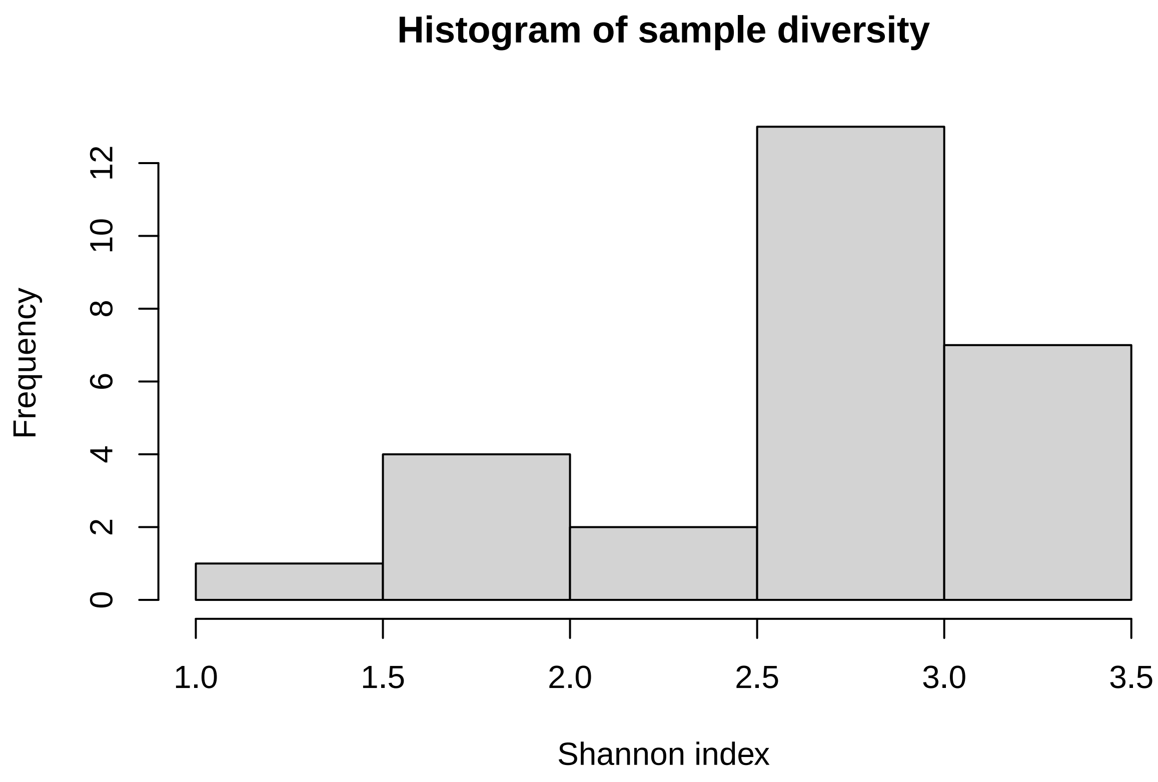
```
colData(tse) # samples metadata
```

```
## DataFrame with 27 rows and 7 columns
##      patient_status      cohort patient_status_vs_cohort sample_name       sum
##         <character> <character>              <character> <character> <numeric>
## A110           ADHD    Cohort_1            ADHD_Cohort_1        A110     37394
## A12            ADHD    Cohort_1            ADHD_Cohort_1         A12     40584
## A15            ADHD    Cohort_1            ADHD_Cohort_1         A15     16077
## A19            ADHD    Cohort_1            ADHD_Cohort_1         A19     39210
## A21            ADHD    Cohort_2            ADHD_Cohort_2         A21      6351
## ...             ...         ...                      ...         ...       ...
## A26         Control    Cohort_2         Control_Cohort_2         A26     20431
## A27         Control    Cohort_2         Control_Cohort_2         A27     14636
## A33         Control    Cohort_3         Control_Cohort_3         A33     13051
## A35         Control    Cohort_3         Control_Cohort_3         A35     12642
## A38         Control    Cohort_3         Control_Cohort_3         A38     15544
##       detected     total
##      <numeric> <numeric>
## A110        68     37394
## A12         51     40584
## A15         68     16077
## A19         62     39210
## A21         58      6351
## ...        ...       ...
## A26         64     20431
## A27         86     14636
## A33         78     13051
## A35         62     12642
## A38         74     15544
```

```
unique(colData(tse)$patient_status) # patient groups
```

```
## [1] "ADHD"    "Control"
```

```
# Example of sample diversity using Shannon index
tse <- mia::estimateDiversity(tse,
                             abund_values = "counts",
                             index = "shannon",
                             name = "shannon")
hist(colData(tse)$shannon, xlab = "Shannon index", main = "Histogram of sample diversity")
```



- **Subsetting** Pick a subset of the data object including only ADHD individuals from Cohort 1. How many there are?

```
sub_cohort_1 <- tse[, colData(tse)$cohort=="Cohort_1"]
sub_cohort_1_ADHD <- sub_cohort_1[, colData(sub_cohort_1)$patient_status=="ADHD"]
colData(sub_cohort_1_ADHD)
```

```
## DataFrame with 4 rows and 8 columns
##      patient_status      cohort patient_status_vs_cohort sample_name       sum
##         <character> <character>              <character> <character> <numeric>
## A110           ADHD    Cohort_1            ADHD_Cohort_1        A110     37394
## A12            ADHD    Cohort_1            ADHD_Cohort_1         A12     40584
## A15            ADHD    Cohort_1            ADHD_Cohort_1         A15     16077
## A19            ADHD    Cohort_1            ADHD_Cohort_1         A19     39210
##       detected     total   shannon
##      <numeric> <numeric> <numeric>
## A110        68     37394   1.76541
## A12         51     40584   2.71644
## A15         68     16077   3.17810
## A19         62     39210   2.89199
```

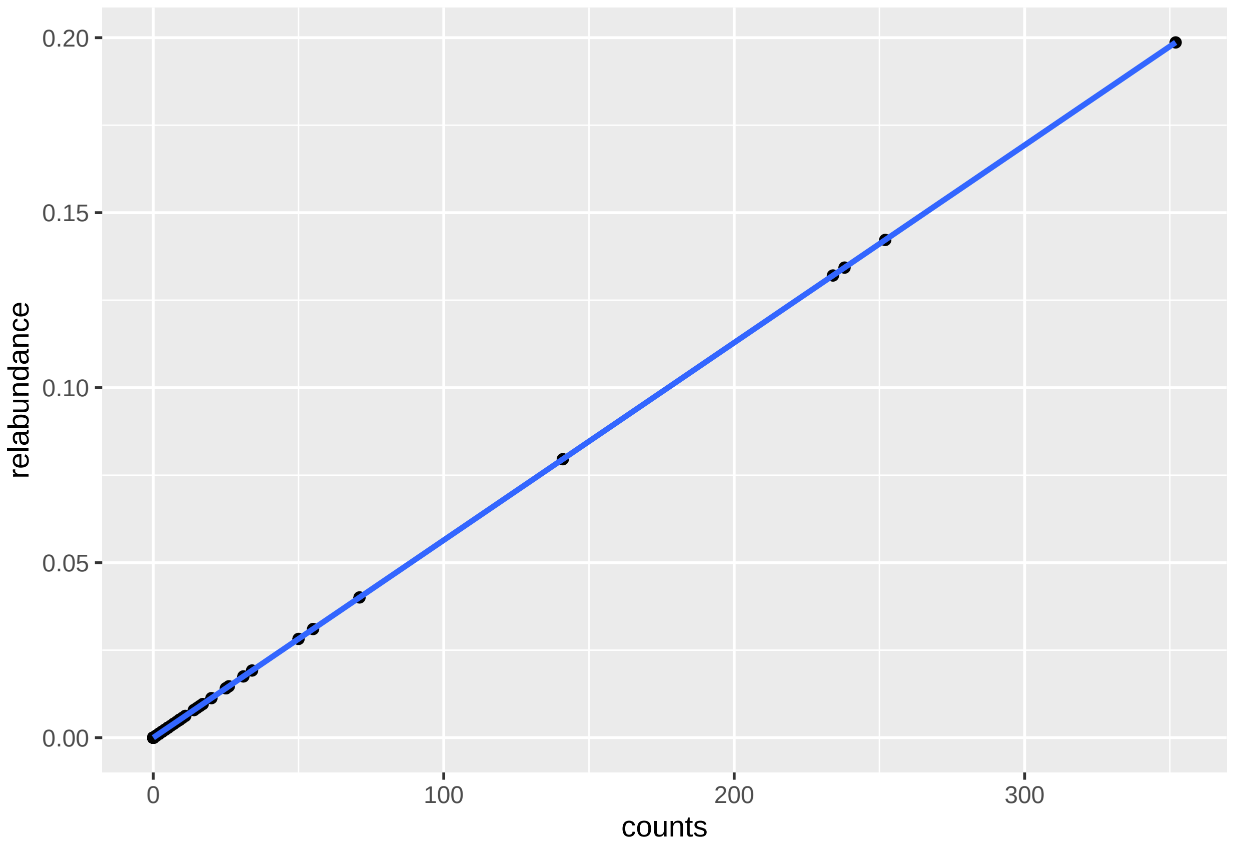
- **Transformations** The data contains read counts. We can convert these into relative abundances and other formats. Compare abundance of a given taxonomic group using the example data before and after the compositionality transformation (with a cross-plot, for instance). You can also compare the results to CLR-transformed data (see e.g. Gloor et al. 2017)

```
tse <- transformCounts(tse, method = "relabundance")
tse <- transformCounts(tse, method = "clr", abund_values = "counts",pseudocount = 1)
```

```
## Warning: All the total abundances of samples do not sum-up to a fixed constant.
## Please consider to apply, e.g., relative transformation in prior to CLR
## transformation.
```

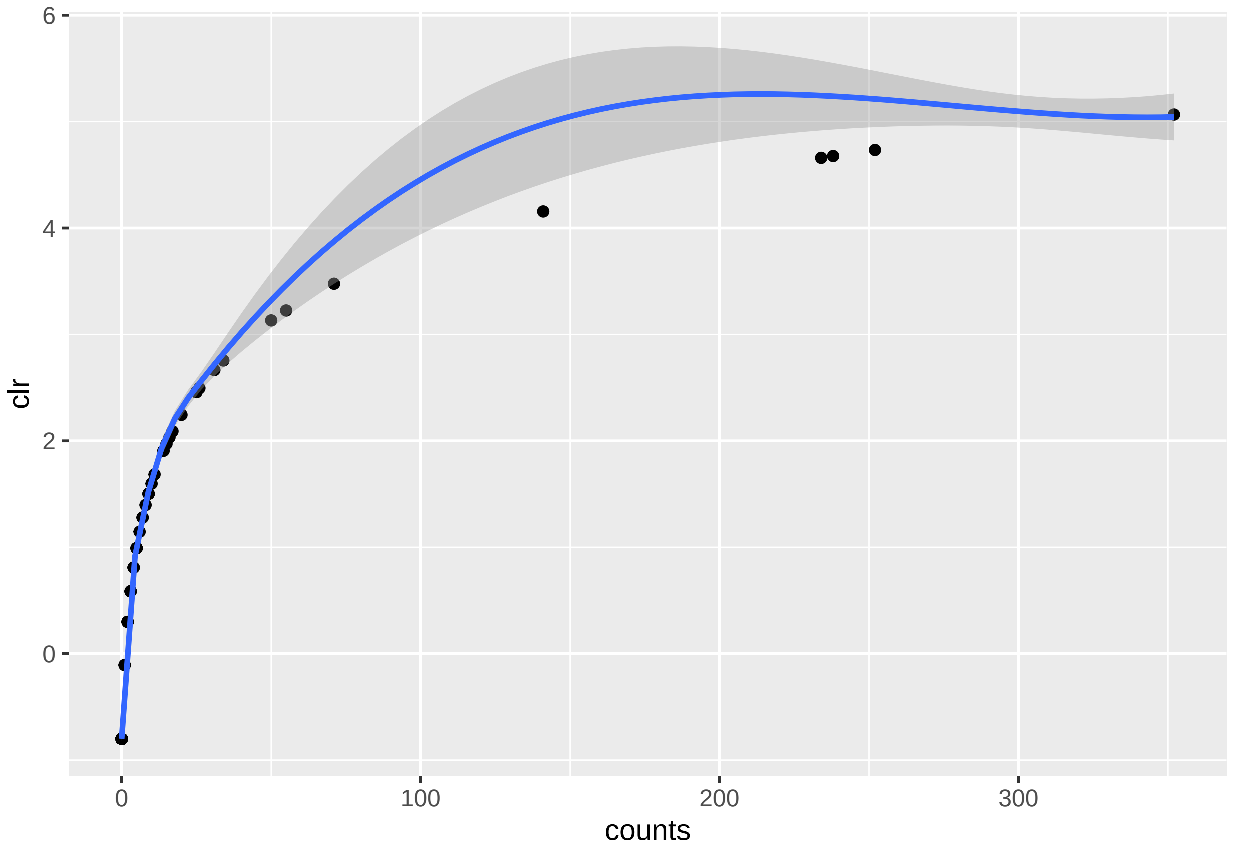
```
# Lets compare with taxa: A29
taxa <- "A29"
df <- as.data.frame(list(
                          counts=assays(tse)$counts[,taxa],
                          relabundance=assays(tse)$relabundance[,taxa],
                          clr=assays(tse)$clr[,taxa])
                        )
ggplot(df, aes(x=counts,y=relabundance))+
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



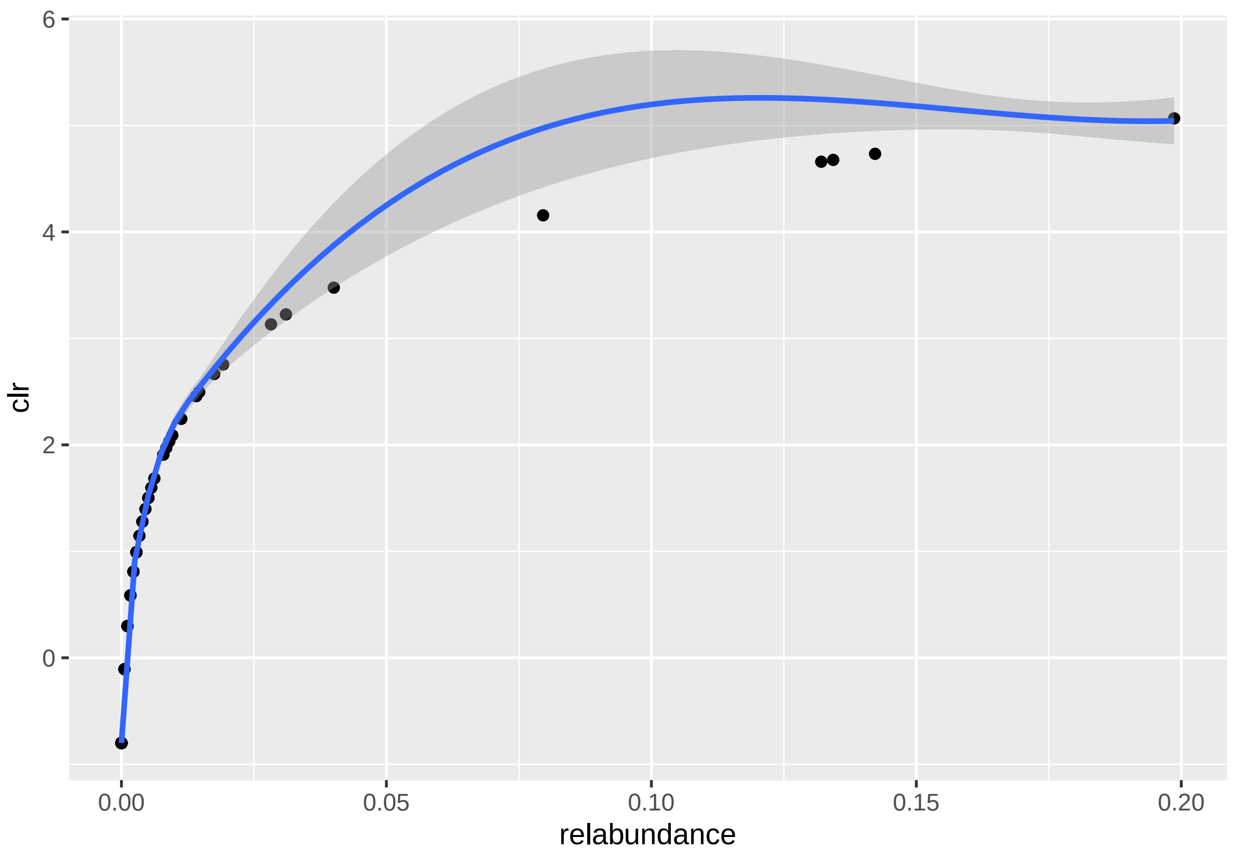
```
ggplot(df, aes(x=counts,y=clr))+
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```
ggplot(df, aes(x=relabundance,y=clr))+
  geom_point()+
  geom_smooth()
```

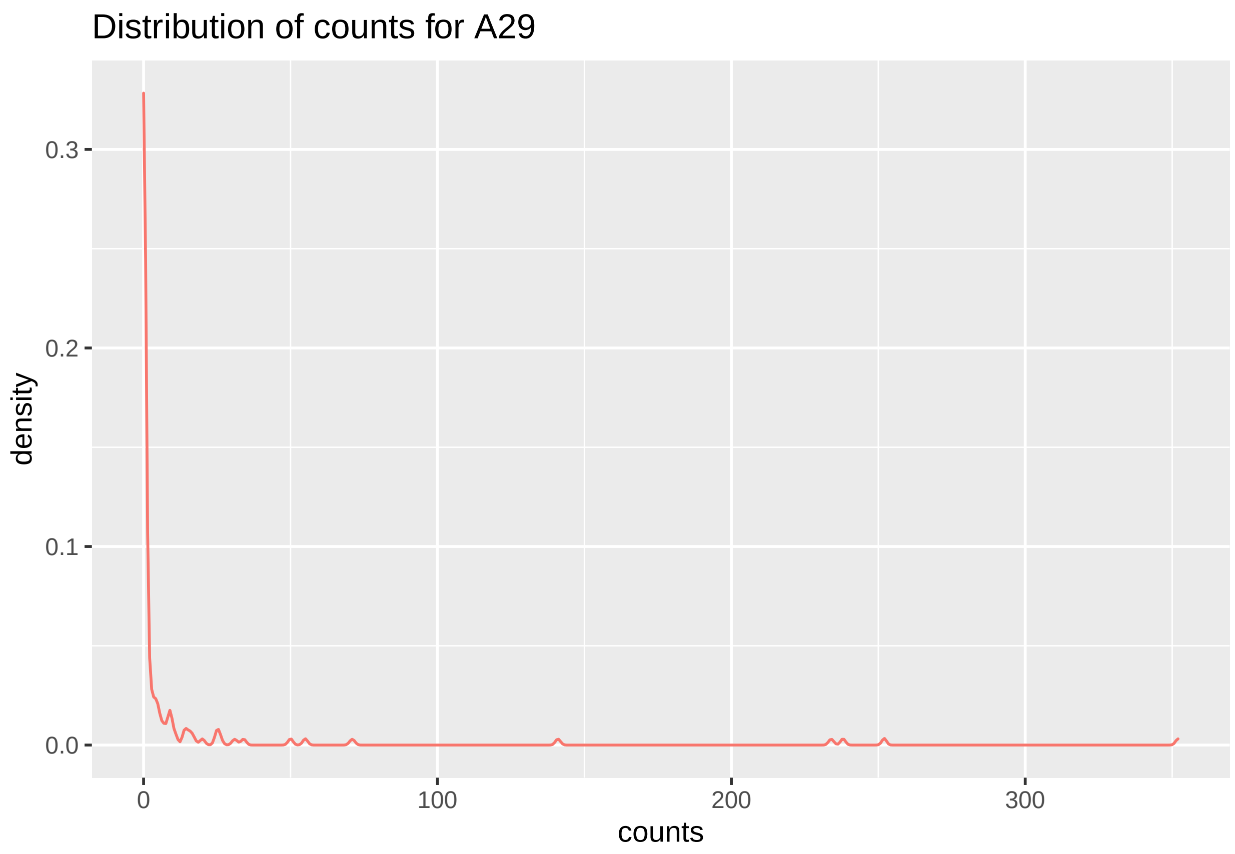
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



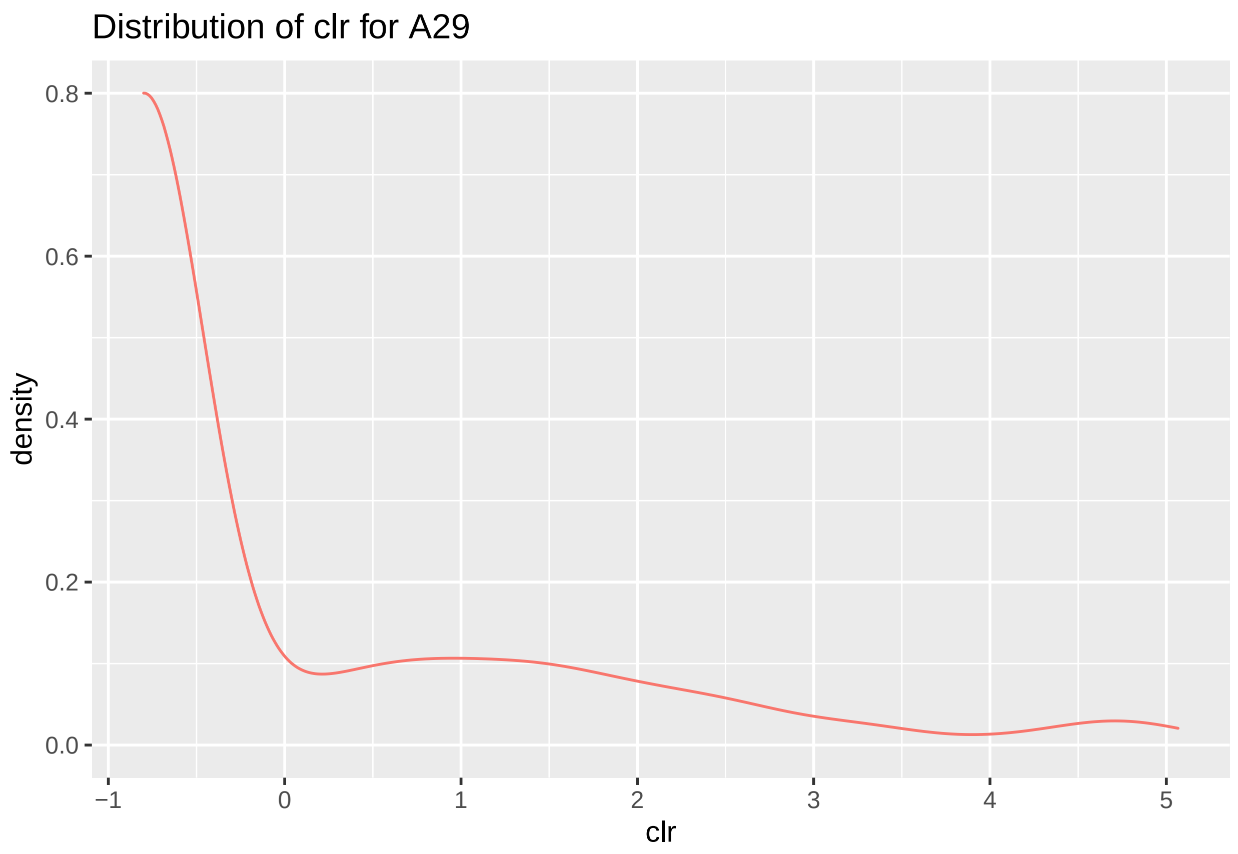
*

Visual exploration Visualize the population distribution of abundances for certain taxonomic groups. Do the same for CLR-transformed abundances.

```
# Same taxa used as earlier
ggplot(df, aes(x=counts,colour="blue")) +
    geom_density(alpha=.2)+
  theme(legend.position = "none")+
  labs(title =paste("Distribution of counts for",taxa, collapse = ": "))
```



```
ggplot(df, aes(x=clr,colour="red")) +
    geom_density(alpha=.2)+
  theme(legend.position = "none")+
  labs(title =paste("Distribution of clr for",taxa, collapse = ": "))
```


Distribution of clr for A29
density
0.0
0.2
0.4
0.6
0.8
−1
0
1
2
3
4
5
clr

Chapter 7

Alpha diversity demo

7.1 Alpha diversity estimation

First let's load the required packages and data set

```
library(mia)
library(miaViz)
library(tidyverse)
# library(vegan)

tse <- read_rds("data/Tengeler2020/tse.rds")

tse
```

```
## class: TreeSummarizedExperiment
## dim: 151 27
## metadata(0):
## assays(1): counts
## rownames(151): 1726470 1726471 ... 17264756 17264757
## rowData names(6): Kingdom Phylum ... Family Genus
## colnames(27): A110 A12 ... A35 A38
## colData names(4): patient_status cohort patient_status_vs_cohort
##   sample_name
## reducedDimNames(0):
## mainExpName: NULL
## altExpNames(0):
## rowLinks: a LinkDataFrame (151 rows)
## rowTree: 1 phylo tree(s) (151 leaves)
## colLinks: NULL
## colTree: NULL
```

Then let's estimate multiple diversity indices.

```
?estimateDiversity

tse <- estimateDiversity(tse,
                              index = c("shannon","gini_simpson","faith"),
                              name = c("shannon","gini_simpson","faith"))
head(colData(tse))
```

```
## DataFrame with 6 rows and 7 columns
##      patient_status      cohort patient_status_vs_cohort sample_name   shannon
##         <character> <character>              <character> <character> <numeric>
## A110           ADHD    Cohort_1            ADHD_Cohort_1        A110   1.76541
## A12            ADHD    Cohort_1            ADHD_Cohort_1         A12   2.71644
## A15            ADHD    Cohort_1            ADHD_Cohort_1         A15   3.17810
## A19            ADHD    Cohort_1            ADHD_Cohort_1         A19   2.89199
## A21            ADHD    Cohort_2            ADHD_Cohort_2         A21   2.84198
## A23            ADHD    Cohort_2            ADHD_Cohort_2         A23   2.79794
##      gini_simpson     faith
##         <numeric> <numeric>
## A110     0.669537   7.39224
## A12      0.871176   6.29378
## A15      0.930561   6.60608
## A19      0.899210   6.79708
## A21      0.885042   6.65110
## A23      0.859813   5.96246
```

We can see that the variables are included in the data. Similarly, let's calculate richness indices.

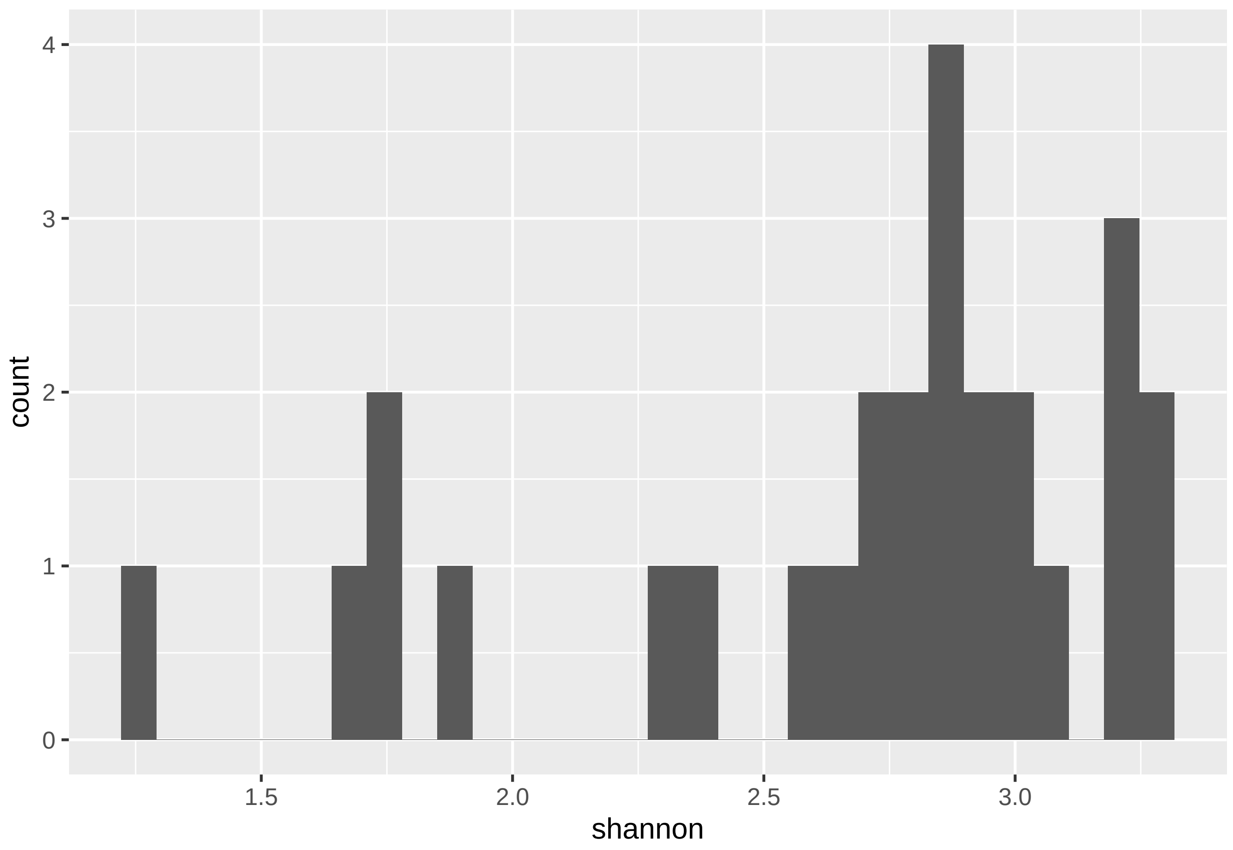
```
tse <- estimateRichness(tse,
                              index = c("chao1","observed"))
head(colData(tse))
```

```
## DataFrame with 6 rows and 10 columns
##      patient_status      cohort patient_status_vs_cohort sample_name   shannon
##         <character> <character>              <character> <character> <numeric>
## A110           ADHD    Cohort_1            ADHD_Cohort_1        A110   1.76541
## A12            ADHD    Cohort_1            ADHD_Cohort_1         A12   2.71644
## A15            ADHD    Cohort_1            ADHD_Cohort_1         A15   3.17810
## A19            ADHD    Cohort_1            ADHD_Cohort_1         A19   2.89199
## A21            ADHD    Cohort_2            ADHD_Cohort_2         A21   2.84198
## A23            ADHD    Cohort_2            ADHD_Cohort_2         A23   2.79794
##      gini_simpson     faith     chao1  chao1_se  observed
##         <numeric> <numeric> <numeric> <numeric> <numeric>
## A110     0.669537   7.39224        68  0.000000        68
## A12      0.871176   6.29378        51  0.000000        51
## A15      0.930561   6.60608        68  0.000000        68
## A19      0.899210   6.79708        62  0.000000        62
## A21      0.885042   6.65110        58  0.000000        58
## A23      0.859813   5.96246        61  0.247942        61
```

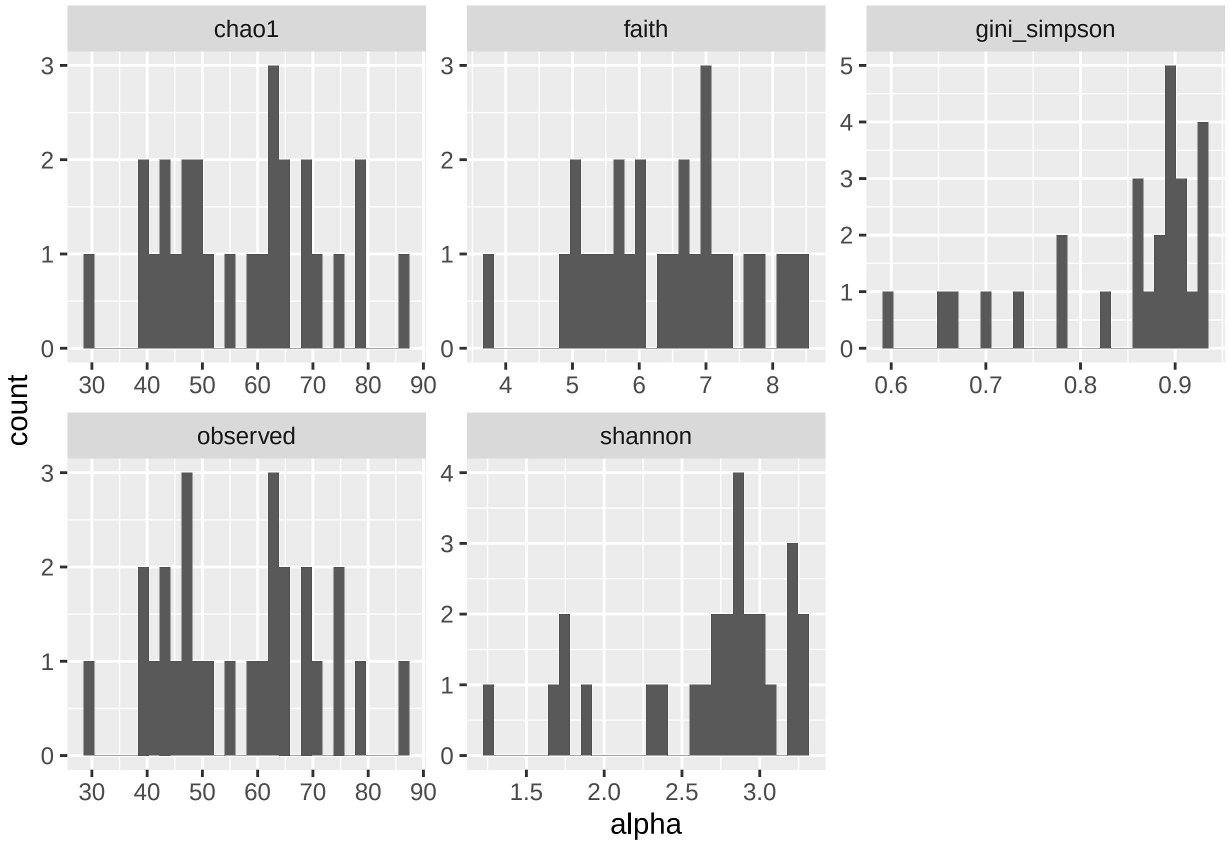
7.2 Visualizing alpha diversity

We can plot the distributions of individual indices:

```
#individual plot
p <- as_tibble(colData(tse)) %>%
  ggplot(aes(shannon)) +
  geom_histogram()

print(p)
```

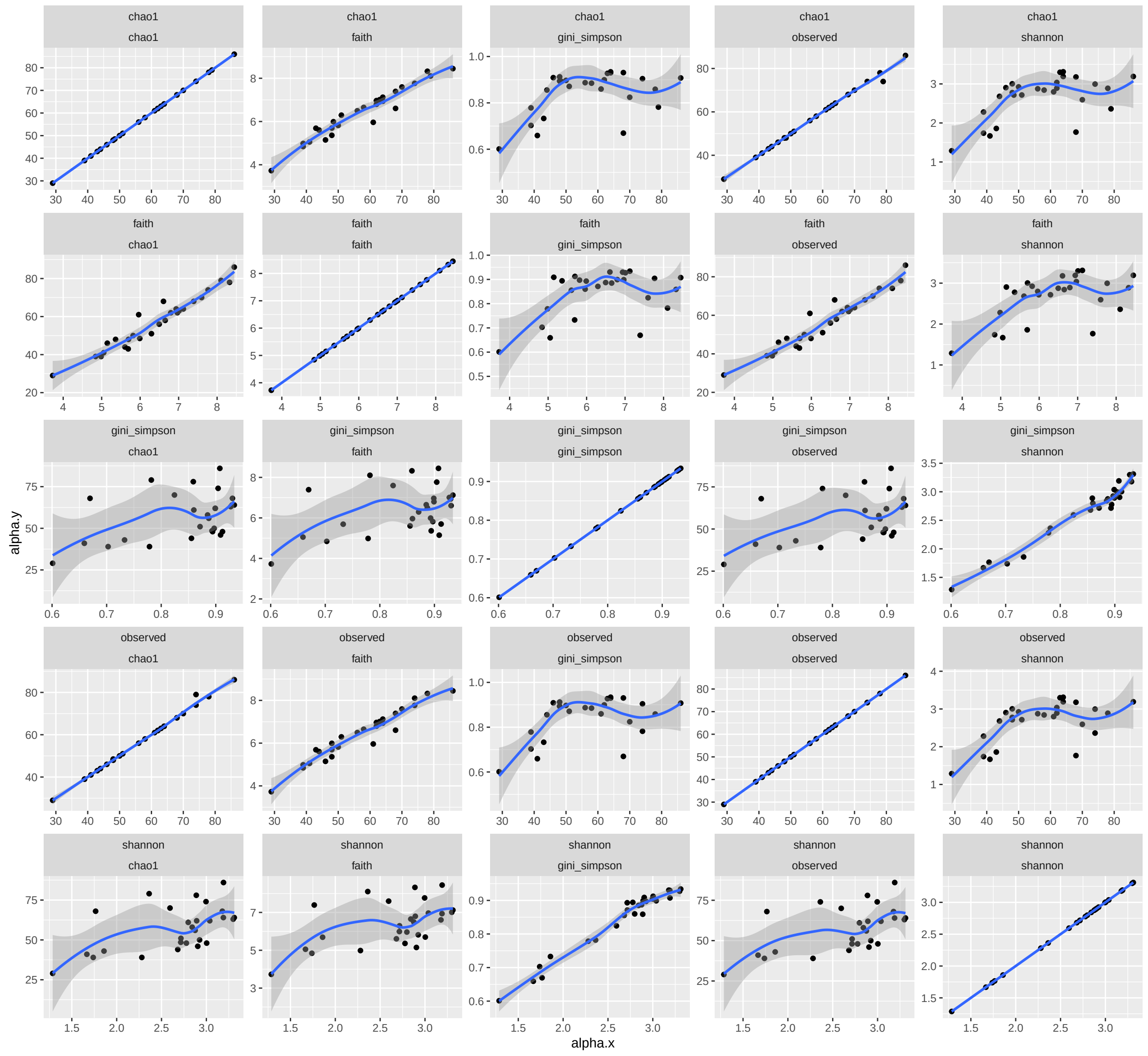


```
#multiple plots

p <- as_tibble(colData(tse)) %>%
  pivot_longer(cols = c("shannon","gini_simpson","faith","chao1","observed"), names_to = "index", values_to =
  ggplot(aes(alpha)) +
  geom_histogram() +
  facet_wrap(vars(index), scales = "free")


print(p)
```



and the correlation between indices:

```r
p <- as_tibble(colData(tse)) %>%
  pivot_longer(cols = c("shannon","gini_simpson","faith","chao1","observed"), names_to = "index", values_to =
  full_join(.,., by = "sample_name") %>%
  ggplot( aes(x = alpha.x, y = alpha.y)) +
  geom_point() +
  geom_smooth() +
  facet_wrap(index.x ~ index.y, scales = "free")

print(p)
```

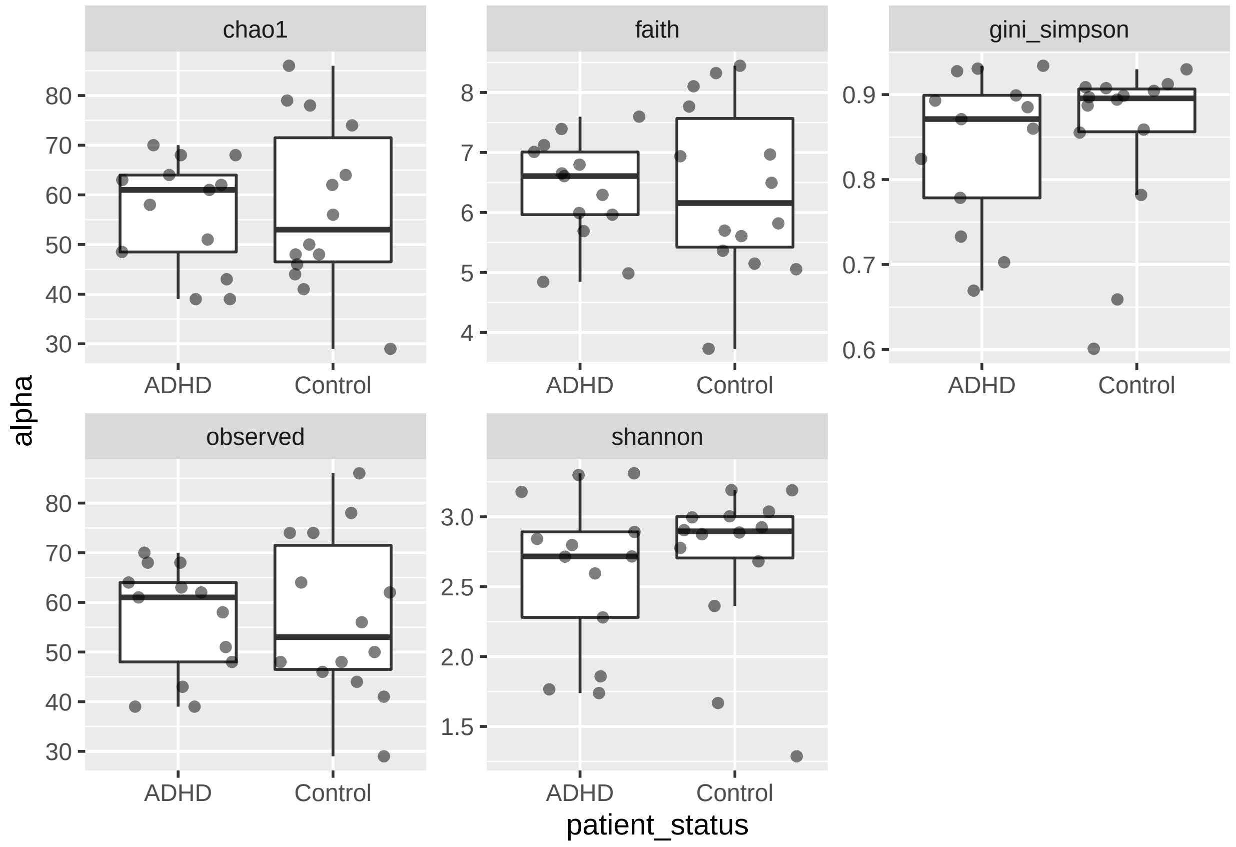


7.3 Comparing alpha diversity

It is often interesting to look for any group differences:

```
p <- as_tibble(colData(tse)) %>%
  pivot_longer(cols = c("shannon","gini_simpson","faith","chao1","observed"), names_to = "index", values_to =
  ggplot( aes(x = patient_status, y = alpha)) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(alpha =0.5) +
  facet_wrap(vars(index), scales = "free")
```

```
print(p)
```



Moreover, we can test the group differences by parametric or non-parametric tests:

```
df1 <- as_tibble(colData(tse)) %>%
  pivot_longer(cols = c("faith","chao1","observed"), names_to = "index", values_to = "alpha") %>%
  group_by(index) %>%
  nest() %>%
  mutate(test_pval = map_dbl(data, ~ t.test(alpha ~ patient_status, data = .x)$p.value)) %>%
  mutate(test = "ttest" )

df2 <- as_tibble(colData(tse)) %>%
  pivot_longer(cols = c("shannon","gini_simpson"), names_to = "index", values_to = "alpha") %>%
  group_by(index) %>%
  nest() %>%
  mutate(test_pval = map_dbl(data, ~ wilcox.test(alpha ~ patient_status, data = .x)$p.value))%>%
  mutate(test = "wilcoxon" )

df <- rbind(df1,df2) %>% select(-data) %>% arrange(test_pval) %>% ungroup()

df
```

```
## # A tibble: 5 x 3
##   index        test_pval test
##   <chr>            <dbl> <chr>
## 1 shannon          0.488 wilcoxon
## 2 gini_simpson     0.685 wilcoxon
## 3 chao1            0.856 ttest
## 4 observed         0.900 ttest
## 5 faith            0.983 ttest
```

End of the demo.

7.4 Exercises

Do “Alpha diversity basics” from the exercises.

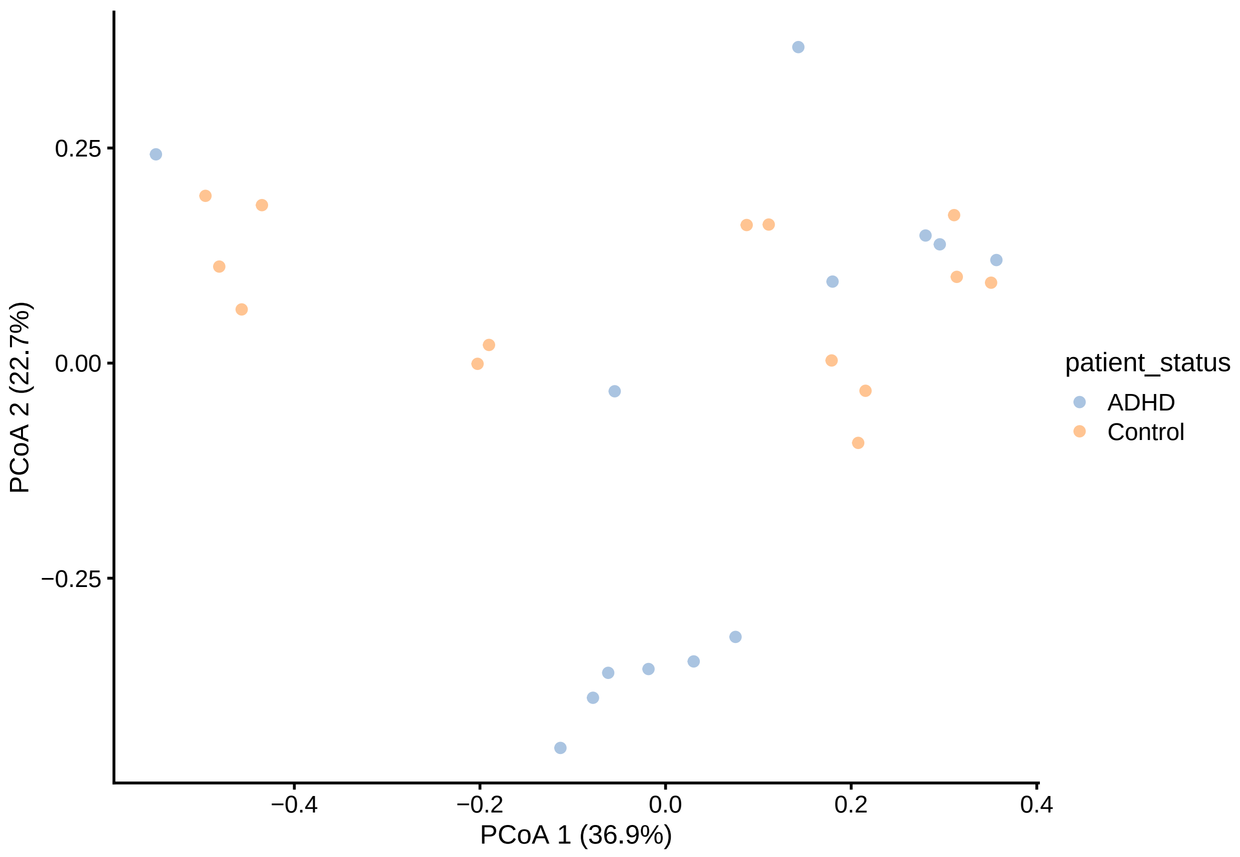
Chapter 8

Beta diversity demo

8.1 Visualizations

Lets generate ordination plots with different methods and transformations.

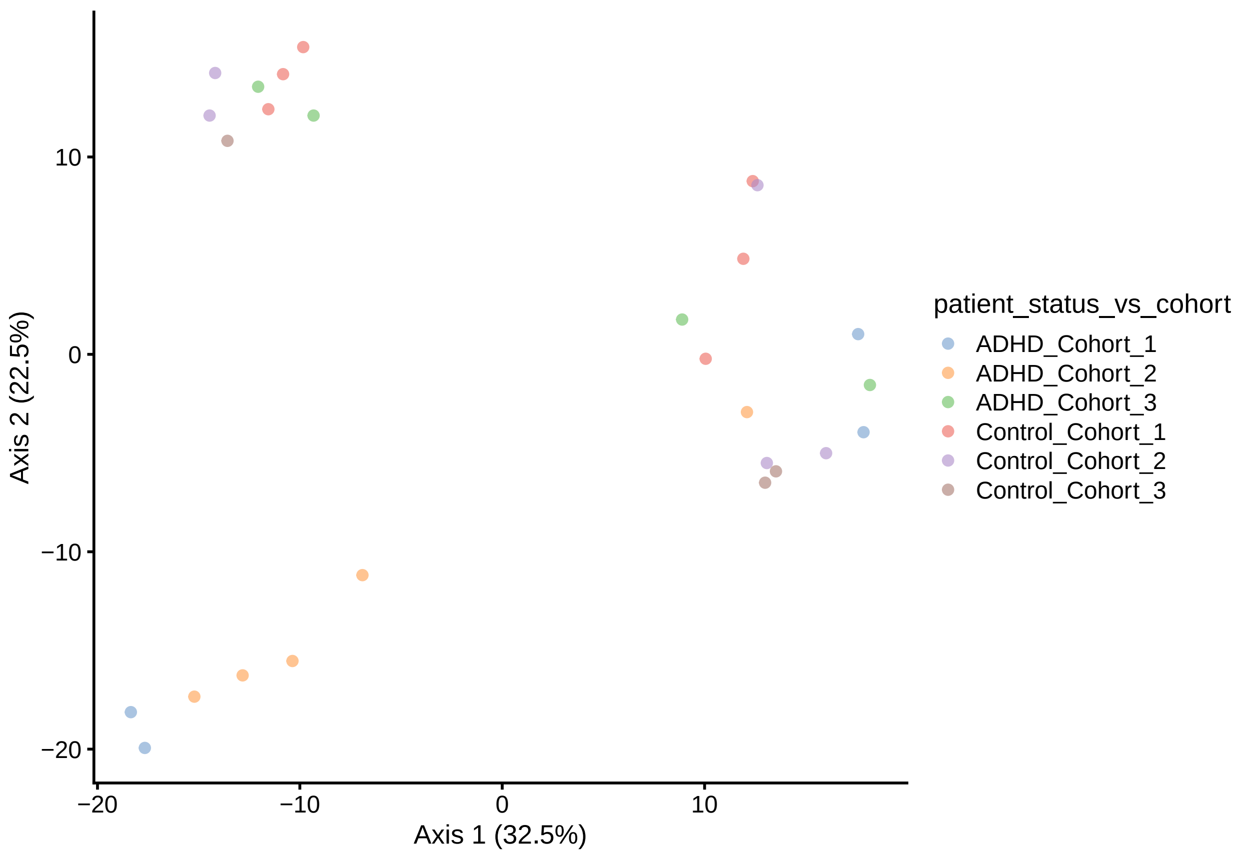
```r
#### calculating Bray Curtis dissimilarity and PCoA

tse <- transformSamples(tse, method = "relabundance")
tse <- runMDS(tse, FUN = vegan::vegdist, method = "bray", name = "PCoA_BC", exprs_values = "relabundance")


p <- plotReducedDim(tse, "PCoA_BC", colour_by = "patient_status")

# Add explained variance for each axis
e <- attr(reducedDim(tse, "PCoA_BC"), "eig");
rel_eig <- e/sum(e[e>0])
p <- p + labs(x = paste("PCoA 1 (", round(100 * rel_eig[[1]],1), "%", ")", sep = ""),
              y = paste("PCoA 2 (", round(100 * rel_eig[[2]],1), "%", ")", sep = ""))

print(p)
```



```
#### Aitchinson distances and PCA

tse <- transformSamples(tse, method = "clr", pseudocount = 1)
```

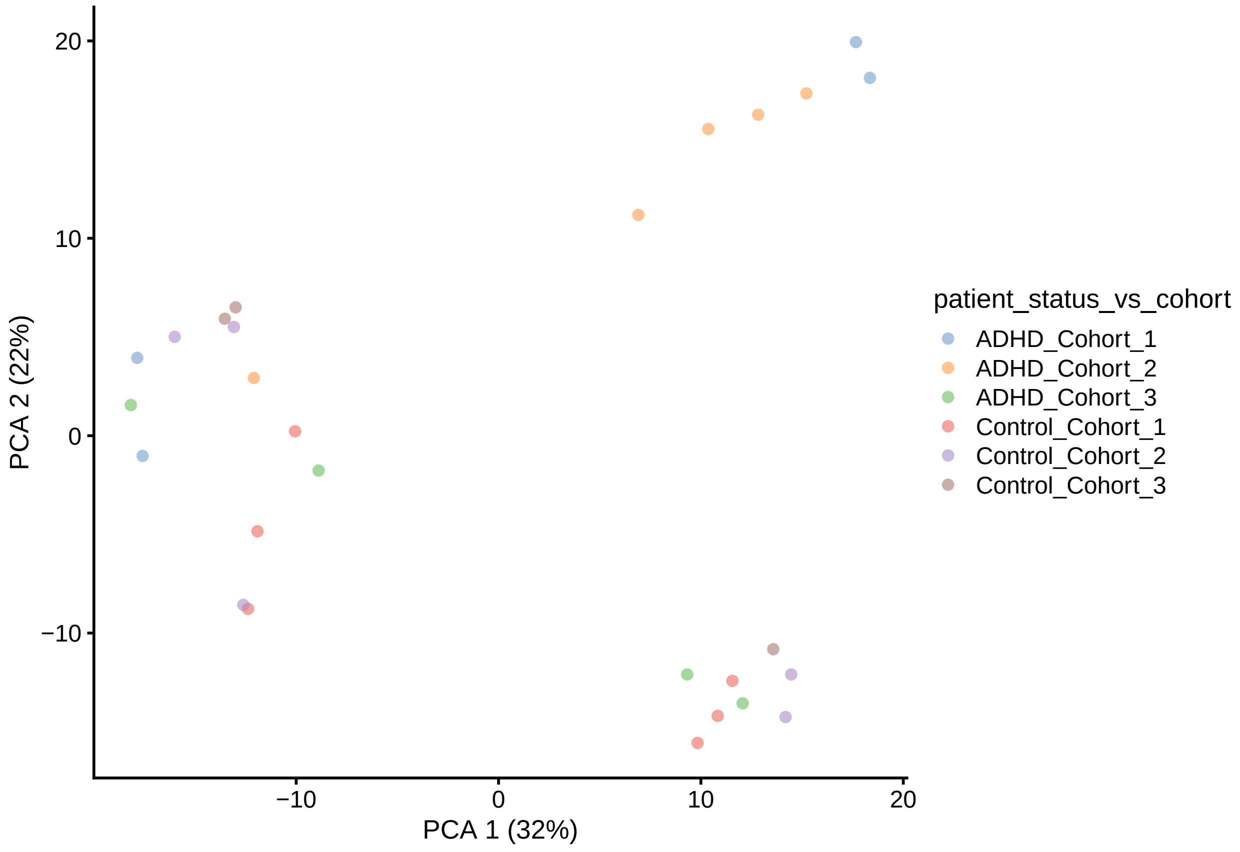
```
## Warning: All the total abundances of samples do not sum-up to a fixed constant.
## Please consider to apply, e.g., relative transformation in prior to CLR
## transformation.
```

```
tse <- runMDS(tse, FUN = vegan::vegdist, name = "MDS_euclidean",
              method = "euclidean", exprs_values = "clr")

p <- plotReducedDim(tse, "MDS_euclidean", colour_by = "patient_status_vs_cohort")

# Add explained variance for each axis
e <- attr(reducedDim(tse, "MDS_euclidean"), "eig");
rel_eig <- e/sum(e[e>0])
p <- p + labs(x = paste("Axis 1 (", round(100 * rel_eig[[1]],1), "%", ")", sep = ""),
              y = paste("Axis 2 (", round(100 * rel_eig[[2]],1), "%", ")", sep = ""))

print(p)
```



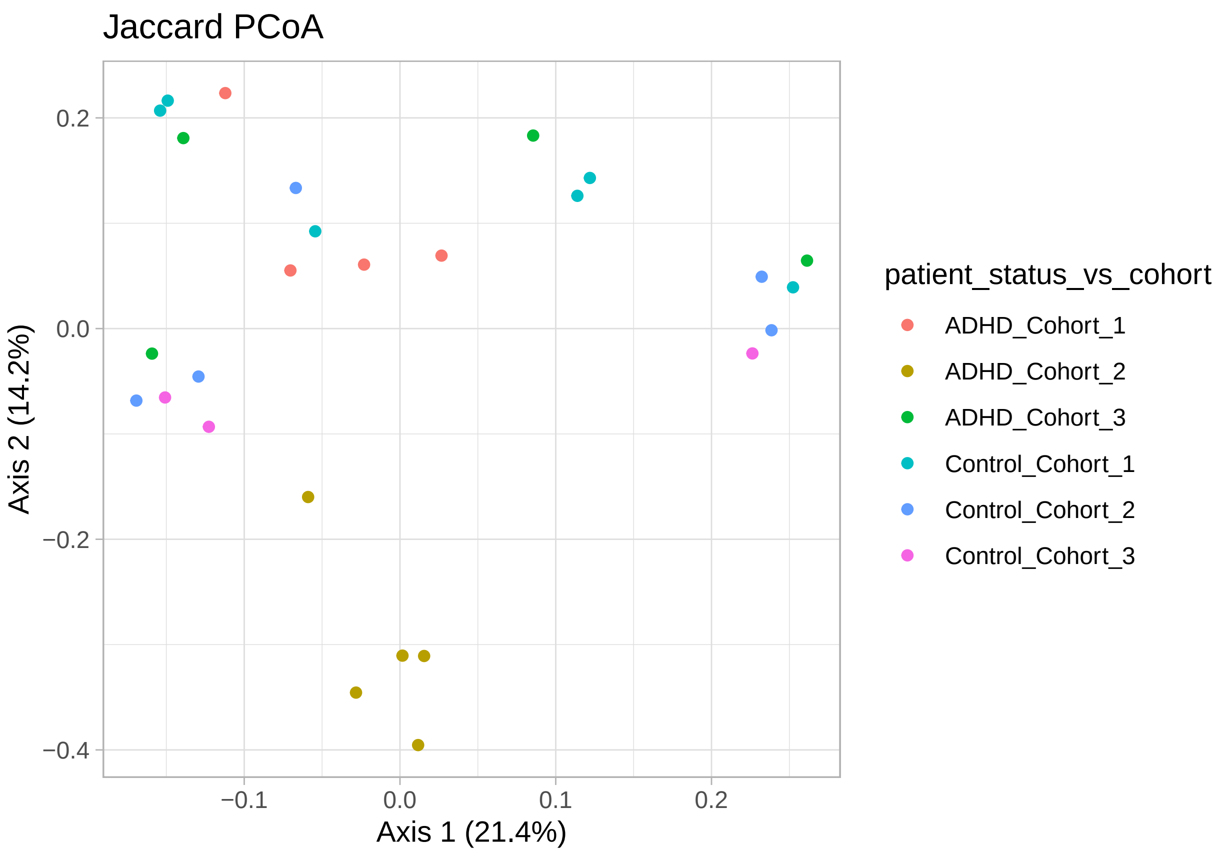
PCA is a subtype of MDS with Euclidean distances, below is a different alternative for running the same analysis.

```
# alternative method

tse <- runPCA(tse, name = "PCA", exprs_values = "clr", ncomponents = 10)
plotReducedDim(tse, "PCA", colour_by = "patient_status_vs_cohort")
```



One can use also ggplot for ordination plots for the flexible adaptablity.

```
dis <- vegan::vegdist(t(assays(tse)$counts), method = "jaccard")

# principal coordinate analysis
jaccard_pcoa <- ecodist::pco(dis)

# a data frame from principal coordinates and groupng variable
jaccard_pcoa_df <- data.frame(pcoa1 = jaccard_pcoa$vectors[,1],
                                pcoa2 = jaccard_pcoa$vectors[,2],
                              patient_status_vs_cohort =  colData(tse)$patient_status_vs_cohort)


# plot
jaccard_plot <- ggplot(data = jaccard_pcoa_df, aes(x=pcoa1, y=pcoa2, color = patient_status_vs_cohort)) +
  geom_point() +
  labs(x = paste("Axis 1 (", round(100 * jaccard_pcoa$values[[1]] / sum(jaccard_pcoa$values), 1), "%", ")", s
       y = paste("Axis 2 (", round(100 * jaccard_pcoa$values[[2]] / sum(jaccard_pcoa$values), 1), "%", ")", s
       title = "Jaccard PCoA") +
  theme(title = element_text(size = 12)) +
  theme_light()

jaccard_plot
```



8.2 Hypothesis testing

PERMANOVA with the function adonis is most commonly used to detect differences in multivariate data. adonis function was recently updated with slightly different functionality. Now the adonis2 allows independent analysis of terms.

```
variable_names <- c("patient_status", "cohort")

tse_genus <- agglomerateByRank(tse, "Genus")
```

```
## Warning: 'clr' includes negative values.
## Agglomeration of it might lead to meaningless values.
## Check the assay, and consider doing transformation again manually with agglomerated data.
```

```
# Apply relative transform
tse_genus <- transformSamples(tse_genus, method = "relabundance")


set.seed(12346)
# We choose 99 random permutations for speed. Consider applying more (999 or 9999)
```

```
assay <- t(assay(tse_genus,"relabundance"))

mod <- paste("assay ~", paste(variable_names, collapse="+")) %>% as.formula()

permanova2 <- vegan::adonis2(mod,
                     by = "margin", # each term analyzed individually
                     data = colData(tse),
                     method = "bray",
                     permutations = 99)

print(permanova2)
```

```
## Permutation test for adonis under reduced model
## Marginal effects of terms
## Permutation: free
## Number of permutations: 99
##
## vegan::adonis2(formula = mod, data = colData(tse), permutations = 99, method = "bray", by = "margin")
##                Df SumOfSqs      R2     F Pr(>F)
## patient_status  1   0.1885 0.05817 1.490   0.23
## cohort          2   0.1450 0.04474 0.573   0.75
## Residual       23   2.9104 0.89787
## Total          26   3.2414 1.00000
```

```
# older adonis for reference
permanova <- vegan::adonis(mod,
                            #by = "margin", # each term analyzed sequentially
                            data = colData(tse),
                            method = "bray",
                            permutations = 99)
```

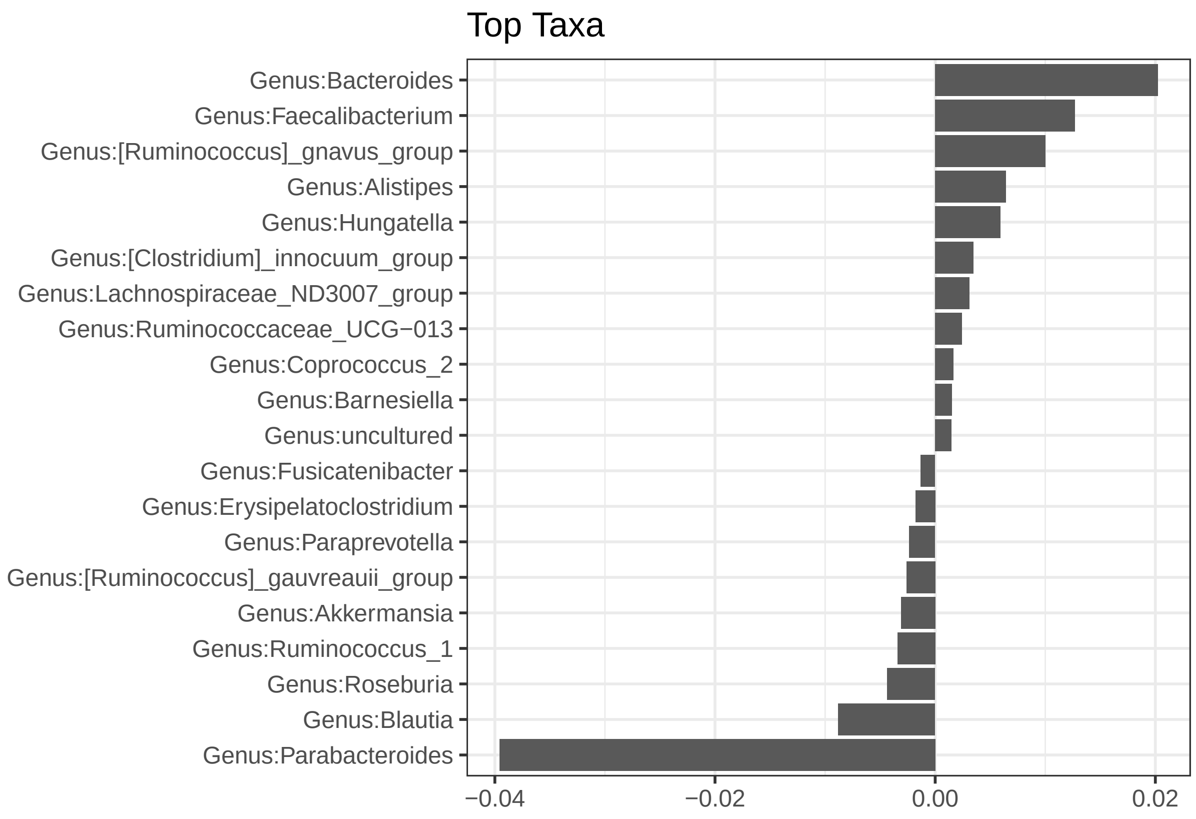
```
## 'adonis' will be deprecated: use 'adonis2' instead
```

```
permanova$aov.tab
```

```
## Permutation: free
## Number of permutations: 99
##
## Terms added sequentially (first to last)
##
##                Df SumsOfSqs  MeanSqs F.Model      R2 Pr(>F)
## patient_status  1    0.1860 0.186024 1.47011 0.05739   0.22
## cohort          2    0.1450 0.072503 0.57298 0.04474   0.79
## Residuals      23    2.9104 0.126537         0.89787
## Total          26    3.2414                  1.00000
```

With older adonis version one cam calculate top coefficients driving the differences between groups.

```r
# older adonis supplies the coefficients
coef <- coefficients(permanova)["cohort1",]
top.coef <- sort(head(coef[rev(order(abs(coef)))],20))

# plot
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
```



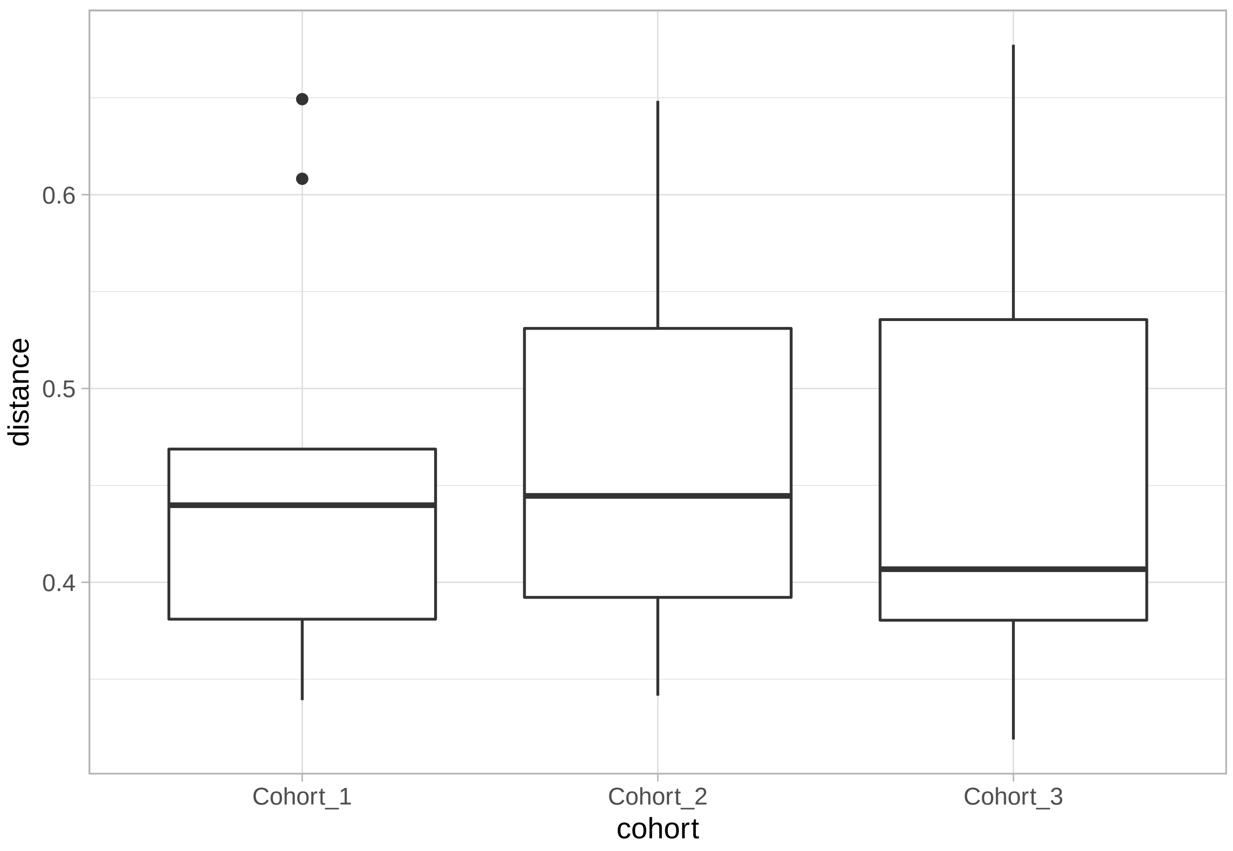
8.2.1 Testing the differences in dispersion

PEMRANOVA doesn't differentiate between different within-group variation, i.e. dispersion, or the mean differences between groups, i.e. the location of the centroid. Follow-up testing can be done with PERMDISP2 implemented in the vegan package.

```
dis <- vegan::vegdist(t(assays(tse)$counts), method = "bray")
b <- vegan::betadisper(dis, colData(tse)$cohort)
print(anova(b))
```

```
## Analysis of Variance Table
##
## Response: Distances
##           Df   Sum Sq   Mean Sq F value Pr(>F)
## Groups     2 0.000375 0.0001875  0.0166 0.9835
## Residuals 24 0.270795 0.0112831
```

```
# boxplor for distances to centroid
p <- cbind(distance = as.numeric(b$distances),
           cohort = colData(tse)$cohort) %>%
  as_tibble() %>%
  mutate(distance = as.numeric(distance)) %>%
  ggplot(aes(cohort, distance)) +
  geom_boxplot() +
  theme_light()

print(p)
```



End of the demo.

8.3 Exercises

Do "Beta diversity" from the exercises.

```r
library(mia)
library(patchwork)
library(tidySummarizedExperiment)
library(ANCOMBC)
library(ALDEx2)
library(Maaslin2)
library(knitr)
library(tidyverse)
# LinDA is a very new package that we therefore install separately
# directly from Github:
if (!"LinDA" %in% installed.packages()[, "Package"]) {
  devtools::install_github("zhouhj1994/LinDA")
}
library(LinDA)
```

```r
# import our dataset
tse <- read_rds("data/Tengeler2020/tse.rds")
```

Chapter 9

Differential abundance analysis demo

Here, we perform differential abundance analyses using four different methods: **Aldex2**, **ANCOMBC**, **MaAsLin2** and **LinDA**. We will analyse Genus level abundances.

We recommend to first have a look at the DAA section of the OMA book. This will give you a little repetition of the introduction and leads you through an example analysis with a different data set and more explanations of the code. Simultaneously, you may want to perform the same analysis on the dataset we have worked with so far. You can double check the steps below where we show how to perform the steps on our dataset.

Lets start with the prevalence filtering:

```r
# note that some tools perform prevalence filtering themselves
# so that this step is unncessary depending on the tool used
tse <- subsetByPrevalentTaxa(tse, detection = 0, prevalence = 0.1)
```

9.1 ALDEx2

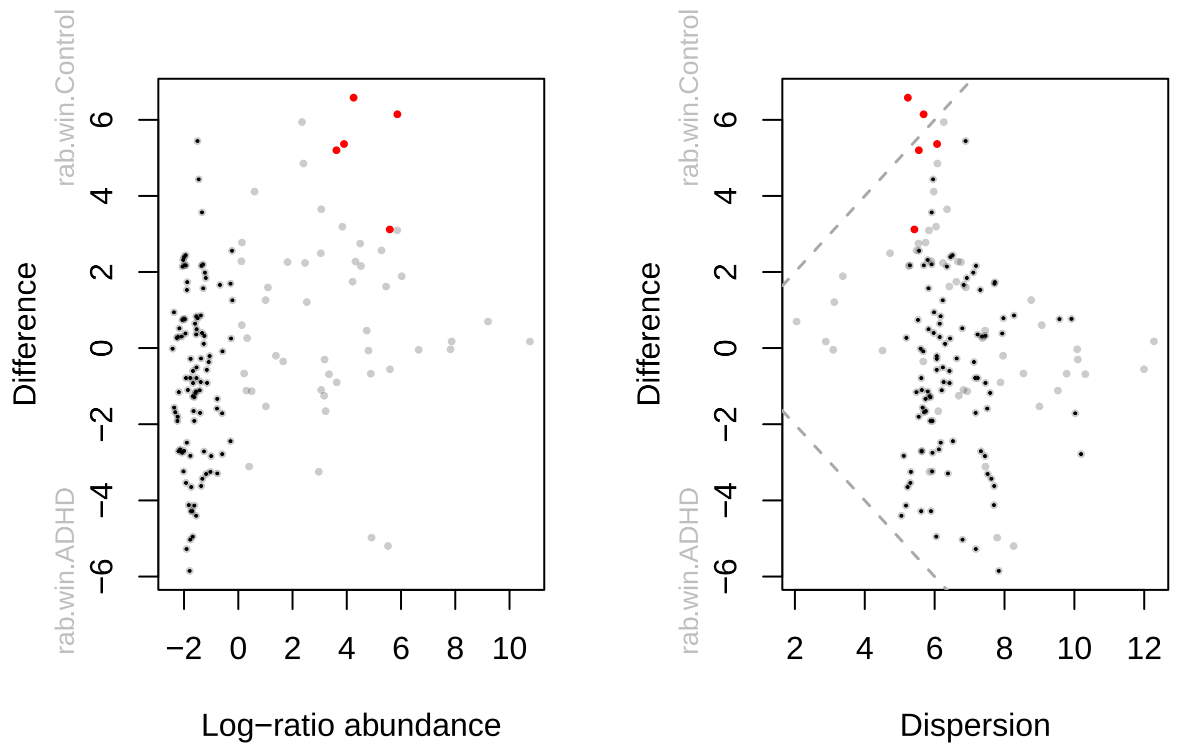
```r
# set seed to rule out different results from randomness
set.seed(1)
# Generate Monte Carlo samples of the Dirichlet distribution for each sample.
# Convert each instance using the centered log-ratio transform.
# This is the input for all further analyses.
x <- aldex.clr(
  reads = assay(tse),
  conds = colData(tse)$patient_status,
  # 128 recommened for ttest, 1000 for rigorous effect size calculation
  mc.samples = 128,
  denom = "all",
  verbose = FALSE
)
```

```
## operating in serial mode
```

```
## computing center with all features
```

```r
# calculates expected values of the Welch's t-test and Wilcoxon rank test on
# the data returned by aldex.clr
x_tt <- aldex.ttest(
  x,
  paired.test = FALSE,
  verbose = FALSE)
# determines the median clr abundance of the feature in all samples and in
# groups, the median difference between the two groups, the median variation
# within each group and the effect size, which is the median of the ratio
# of the between group difference and the larger of the variance within groups
x_effect <- aldex.effect(x, CI = TRUE, verbose = FALSE)
# combine all outputs
aldex_out <- data.frame(x_tt, x_effect)
```

```r
par(mfrow = c(1, 2))
  aldex.plot(
    aldex_out,
    type = "MA",
    test = "welch",
    xlab = "Log-ratio abundance",
    ylab = "Difference",
    cutoff = 0.05
  )
  aldex.plot(
    aldex_out,
    type = "MW",
    test = "welch",
    xlab = "Dispersion",
    ylab = "Difference",
    cutoff = 0.05
  )
```



9.2 ANCOM-BC

In case you get an error when running the function below, you may need to install the newest version of ANCOMBC (released a few days ago) by running the following line of code:

```
# library(devtools)
# install_github("FrederickHuangLin/ANCOMBC", ref = "develop")
```

```
# perform the analysis
out = ancombc(
  x = tse,
  formula = "patient_status",
  p_adj_method = "fdr",
  prv_cut = 0, # no prev filtering necessary anymore
  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
)


# store the results in res
res <- out$res
```

9.3 MaAsLin2

```
# first we rarefy data to stick to Nearing et al 2022
tse <- subsampleCounts(
  tse,
  abund_values = "counts",
  min_size = min(colSums2(assay(tse))),
  seed = runif(1, 0, .Machine$integer.max),
  replace = TRUE,
  name = "subsampled",
  verbose = TRUE
)
```

```
## Warning: Subsampling/Rarefying may undermine downstream analyses and have
## unintended consequences. Therefore, make sure this normalization is appropriate
## for your data.
```

```
## `set.seed(1347512162.87252)` was used to initialize repeatable random subsampling.
## Please record this for your records so others can reproduce.
```

```
## 0 features removed because they are not present in all samples after subsampling.
```

```
# maaslin expects features as columns and samples as rows
# for both the asv/otu table as well as meta data
asv <- t(assay(tse, "subsampled"))
meta_data <- data.frame(colData(tse))
# you can specifiy different GLMs/normalizations/transforms. We used similar
# settings as in Nearing et al. (2021) here:
fit_data <- Maaslin2(
  asv,
  meta_data,
  # A folder will be created that is called like the below specified output.
  # It contains also figures to visualize the difference between genera
  # for the significant ones.
  output = "DAA example",
  transform = "AST",
  fixed_effects = "patient_status",
  # random_effects = c(...), # you can also fit MLM by specifying random effects
  # specifying a ref is especially important if you have more than 2 levels
  reference = "patient_status,Control",
  normalization = "TSS",
  standardize = FALSE,
  min_prevalence = 0 # prev filterin already done
)
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
# which genera are identified as differentially abundant? (leave out "head" to
# see all)
kable(head(filter(fit_data$results, qval <= 0.05)))
```

feature	metadata	value	coef	stderr	pval	name	qval	N	N.not
X17264718	patient_status	Control	0.0728128	0.0125076	0.0000045	patient_statusControl	0.0006841	27	
X172647170	patient_status	Control	0.0675534	0.0147170	0.0001078	patient_statusControl	0.0081392	27	

9.3.1 LinDA

```
otu.tab <- as.data.frame(assay(tse))
meta <- as.data.frame(colData(tse)) %>% select(patient_status)
res <- linda(
  otu.tab,
  meta,
  formula = '~patient_status',
  alpha = 0.05,
  prev.cut = 0, # we already filtered
  lib.cut = 1000,
  winsor.quan = 0.97)
```

```
## Pseudo-count approach is used.
```

9.4 Summary of methods

```
summ <- full_join(
    rownames_to_column(aldex_out, "genus") %>%
      select(genus, aldex2 = wi.eBH),
    rownames_to_column(out$res$diff_abn, "genus") %>%
      select(genus, ancombc = patient_statusControl),
    by = "genus") %>%
  full_join(
    select(fit_data$results, genus = feature, maaslin2 = qval) %>%
      mutate(genus = str_remove(genus, "X")),
    by = "genus") %>%
  full_join(
    select(
      rownames_to_column(res$output$patient_statusControl, "genus"),
      genus, linda = reject),
      by = "genus") %>%
  mutate(
    across(c(aldex2, maaslin2), ~ .x <= 0.05),
    # the following line would be necessary without prevalence filtering
    # as some methods output NA
    #across(-genus, function(x) ifelse(is.na(x), FALSE, x)),
    score = rowSums(across(c(aldex2, ancombc, maaslin2, linda)))
  )

# This is how it looks like:
kable(head(arrange(summ, desc(score))))
```

genus	aldex2	ancombc	maaslin2	linda	score
17264718	TRUE	TRUE	TRUE	TRUE	4
172647170	TRUE	TRUE	TRUE	TRUE	4
17264734	FALSE	TRUE	FALSE	TRUE	2
1726479	FALSE	TRUE	FALSE	TRUE	2
17264728	FALSE	TRUE	FALSE	TRUE	2
17264733	FALSE	TRUE	FALSE	TRUE	2

Now we can answer our questions:

```
# how many genera were identified by each method?
summarise(summ, across(where(is.logical), sum)) %>%
  kable()
```

aldex2	ancombc	maaslin2	linda
2	73	2	23

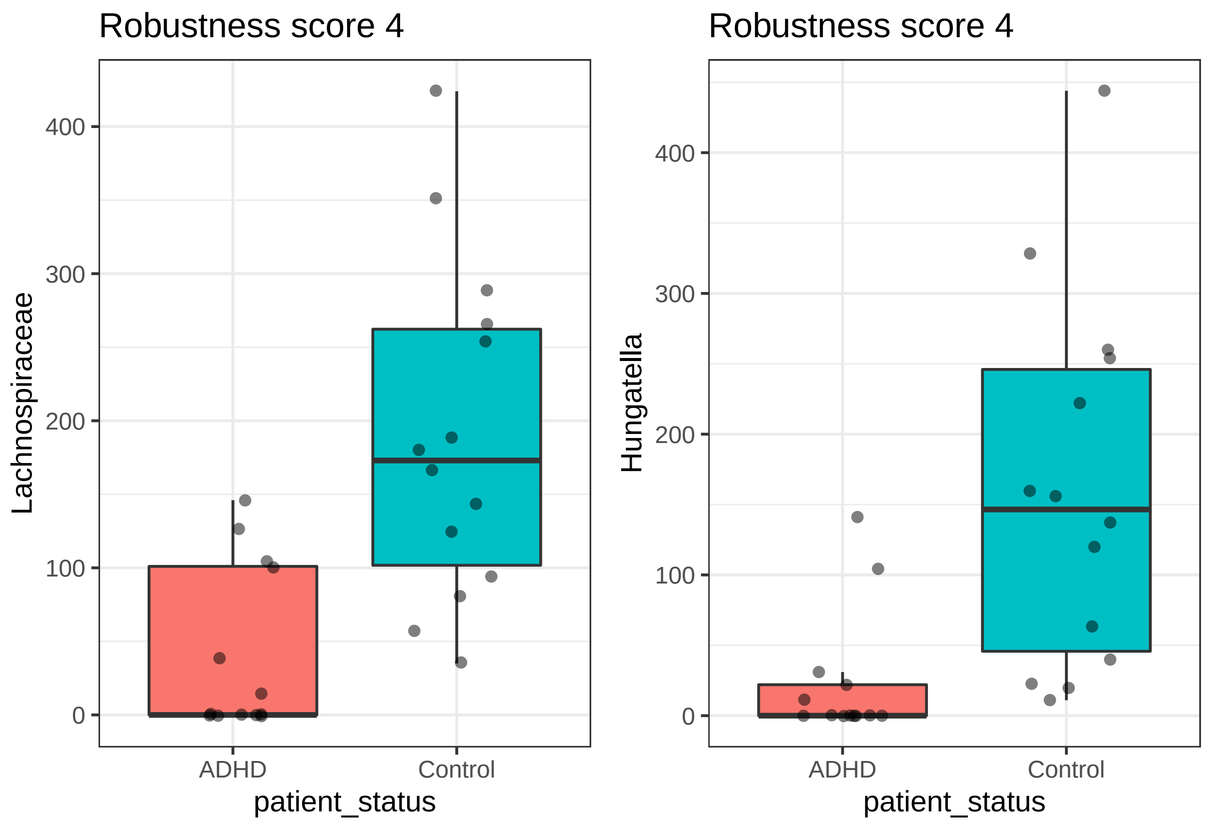
```
# which genera are identified by all methods?
filter(summ, score == 4) %>% kable()
```

genus	aldex2	ancombc	maaslin2	linda	score
17264718	TRUE	TRUE	TRUE	TRUE	4
172647170	TRUE	TRUE	TRUE	TRUE	4

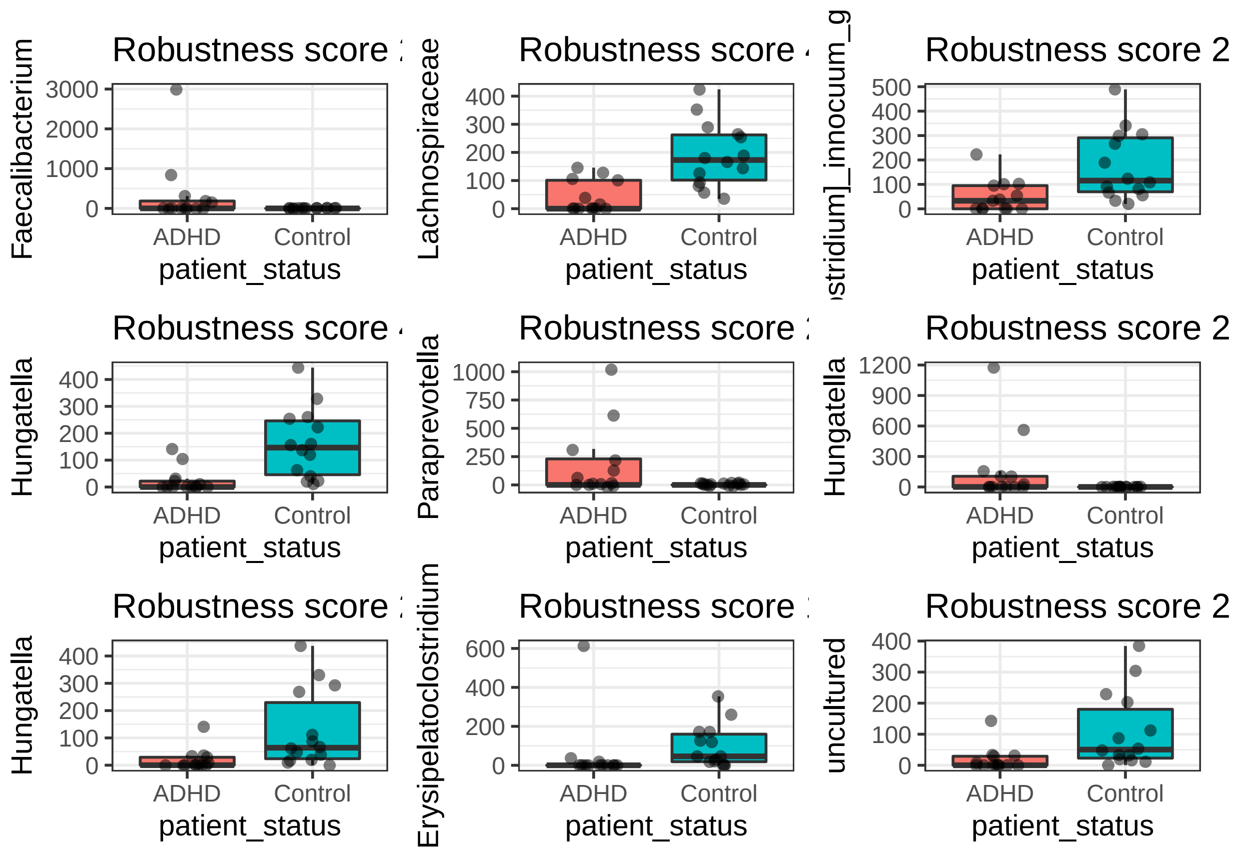
Lets create plots to vizualize differential abundance:

```
# to plot the highest phylogenetic taxon name
taxid_switch <- as.data.frame(rowData(tse)) %>%
  rownames_to_column("taxid") %>%
  mutate(genus = ifelse(Family == "", Class, ifelse(Genus == "", Family, Genus))) %>%
  select(taxid, genus) %>%
  column_to_rownames("taxid")

plot_data <- data.frame(t(assay(tse)))
plot_data$patient_status <- colData(tse)$patient_status
```

```
# create a plot for each genus where the score is indicated in the title
plots <- pmap(
  select(summ, genus, score),
  function(genus, score) {
  taxon <- taxid_switch[genus, "genus"]
  ggplot(plot_data, aes_string("patient_status", glue::glue("X{genus}"))) +
    geom_boxplot(aes(fill = patient_status), outlier.shape = NA) +
    geom_jitter(width = 0.2, alpha = 0.5) +
    ggtitle(glue::glue("Robustness score {score}")) +
    theme_bw() +
    theme(legend.position = "none") +
    ylab(taxon)
})

# now we can show only those genera that have at least score 4 (3, 2 or 1)
robust_plots <- plots[summ$score == 4]


# to display this nicely in the book we use patchwork here:
# (we show first 8)
robust_plots[[1]] +
  robust_plots[[2]]
```



```r
# or if we have most trust in any specific method we can show genera that
# are differentially abundant according to that method and then look in the
# title how many methods also identified it (we only show first 9 here):
linda_plots <- plots[summ$linda]
linda_plots[[1]] + linda_plots[[2]] +
  linda_plots[[3]] + linda_plots[[4]] +
  linda_plots[[5]] + linda_plots[[6]] +
  linda_plots[[7]] + linda_plots[[8]] +
  linda_plots[[9]] +
  plot_layout(ncol = 3)
```



9.5 Exercises

- Try to interpret above analyses.
- Do "Differential Abundance" from the exercises.

Chapter 10

Study material

10.1 Online tutorial

The course will utilize material from the online book (beta version) Orchestrating Microbiome Analysis with R/Bioconductor (OMA). We encourage to familiarize with this material and test examples already before the course.

10.2 Lecture slides

To be added.

10.3 Tasks

Seek guidance from the https://microbiome.github.io/OMA/

- Exercises
- Example solutions

10.4 Extra material on miaverse and R programming

Further information on the data science framework

Bibliography

Borman, T., Eckerman, H., Aatsinki, A., and Lahti, L. (2022). *Microbiome data science with R/Bioconductor. Radboud Summer School.*