

Microbiome data science with R/Bioconductor

Welcome to Radboud Summer School, July 2022

2022-07-06

Contents

1	Overview	3
1.1	Contents and learning goals	3
1.2	Schedule and organizers	3
1.3	Acknowledgments	4
2	Code of Conduct	5
3	Getting started	6
3.1	Checklist (before the course)	6
3.2	Support and resources	6
3.3	Installing and loading the required R packages	6
4	Importing microbiome data	8
4.1	Data access	8
4.2	Importing microbiome data in R	8
4.3	Example solutions	9
5	Reproducible reporting with Rmarkdown	10
6	Alpha diversity demo	11
6.1	Alpha diversity estimation	11
6.2	Visualizing alpha diversity	15
6.3	Comparing alpha diversity	18
6.4	Exercises	20
7	Beta diversity demo	21
7.1	Visualizations	21
7.2	Hypothesis testing	25
7.3	Exercises	29

8	Study material	30
8.1	Online tutorial	30
8.2	Lecture slides	30
8.3	Tasks	30
8.4	Extra material on miaverse and R programming	30

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

All material is released under the open CC BY-NC-SA 3.0 License and available online during and after the course, following the recommendations on open teaching materials of the national open science coordination in Finland. The source code of this repository is reproducible and contains the Rmd files with executable code. See README.

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 not required for the course but it can be useful for later use. 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 8.
- **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.

```
# 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)

# 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.

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


Chapter 4

Importing microbiome data

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

4.1 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

Open data set of your own choice, different options are listed in OMA.

4.2 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).

4.3 Example solutions

Example code for data import: import.Rmd

Chapter 5

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 and tips for Rmarkdown are available in the online tutorial to reproducible reporting by Dr. C Titus Brown.

Chapter 6

Alpha diversity demo

6.1 Alpha diversity estimation

First let's load the required packages and data set

```
library(mia)
```

```
## 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

library(miaViz)

## Loading required package: ggplot2

## Loading required package: ggraph

library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.1 --

## v tibble  3.1.7      v dplyr    1.0.9
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::collapse() masks Biostrings::collapse(), IRanges::collapse()
## x dplyr::combine()  masks Biobase::combine(), BiocGenerics::combine()
## x purrr::compact()  masks XVector::compact()
## x dplyr::count()    masks matrixStats::count()
## x dplyr::desc()     masks IRanges::desc()
## x tidyr::expand()   masks S4Vectors::expand()
## x dplyr::filter()   masks stats::filter()
## x dplyr::first()    masks S4Vectors::first()
## x dplyr::lag()      masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()   masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()   masks S4Vectors::rename()
## x dplyr::slice()    masks XVector::slice(), IRanges::slice()
```

```
# 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
```

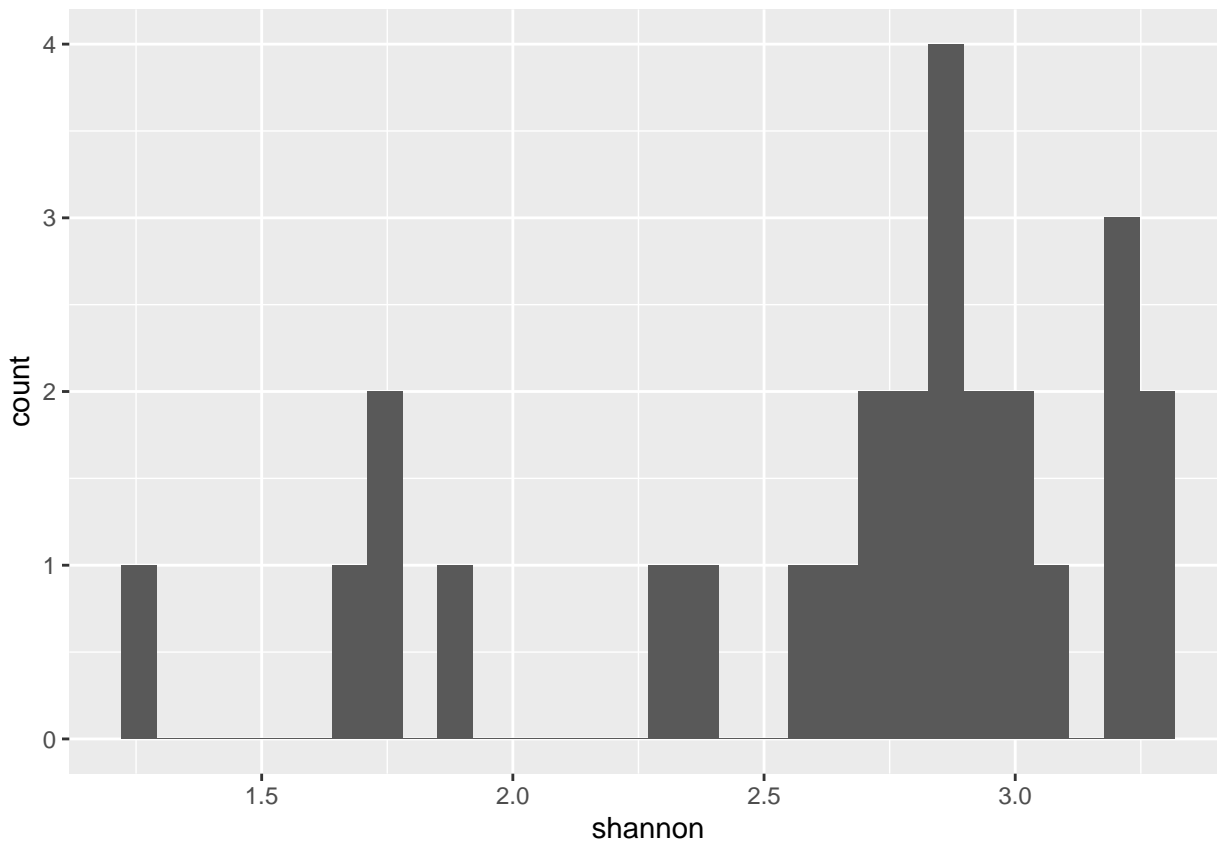
6.2 Visualizing alpha diversity

We can plot the distributions of individual indices:

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

print(p)
```

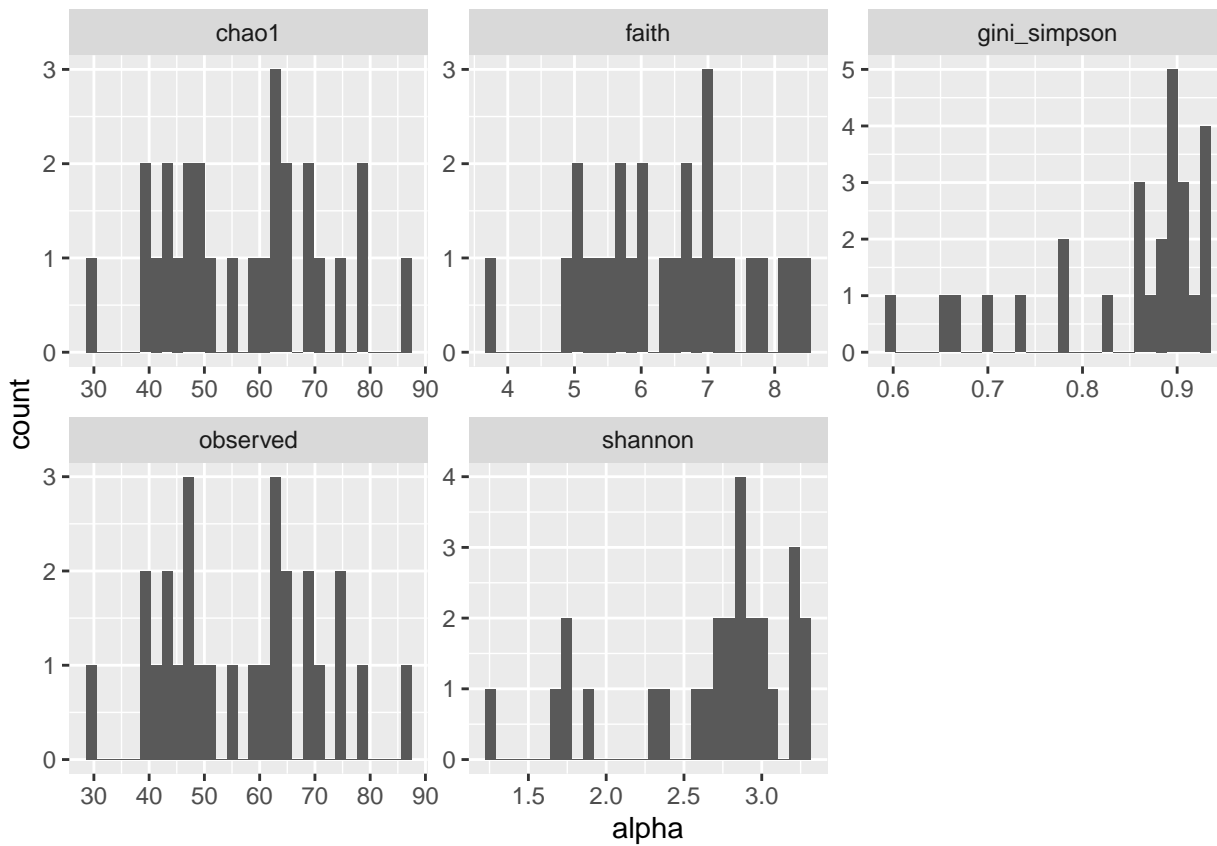
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
#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)
```

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

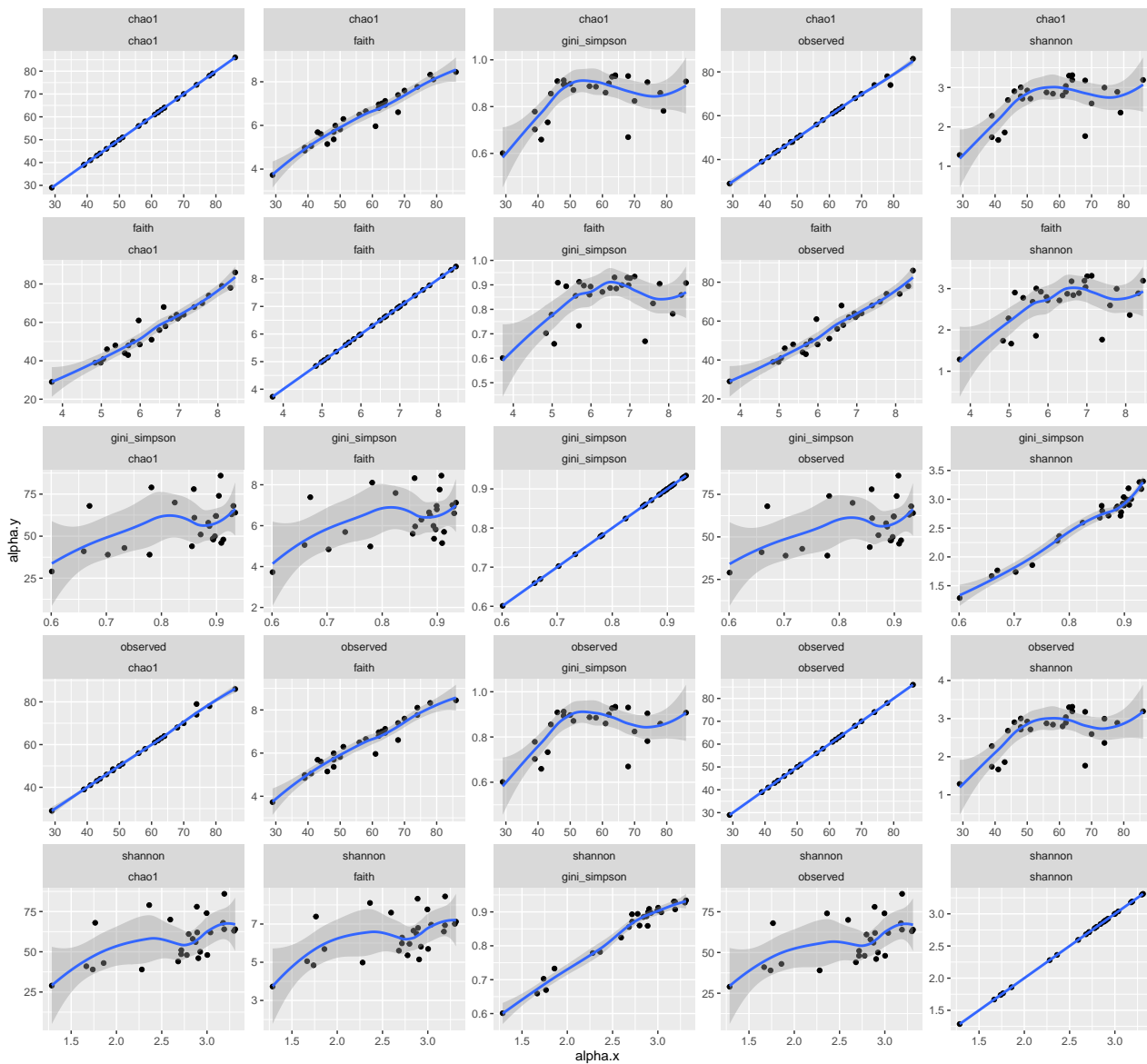


and the correlation between indices:

```
p <- as_tibble(colData(tse)) %>%
  pivot_longer(cols = c("shannon", "gini_simpson", "faith", "chao1", "observed"), names_to = "index", values_to = "value")
  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)
```

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

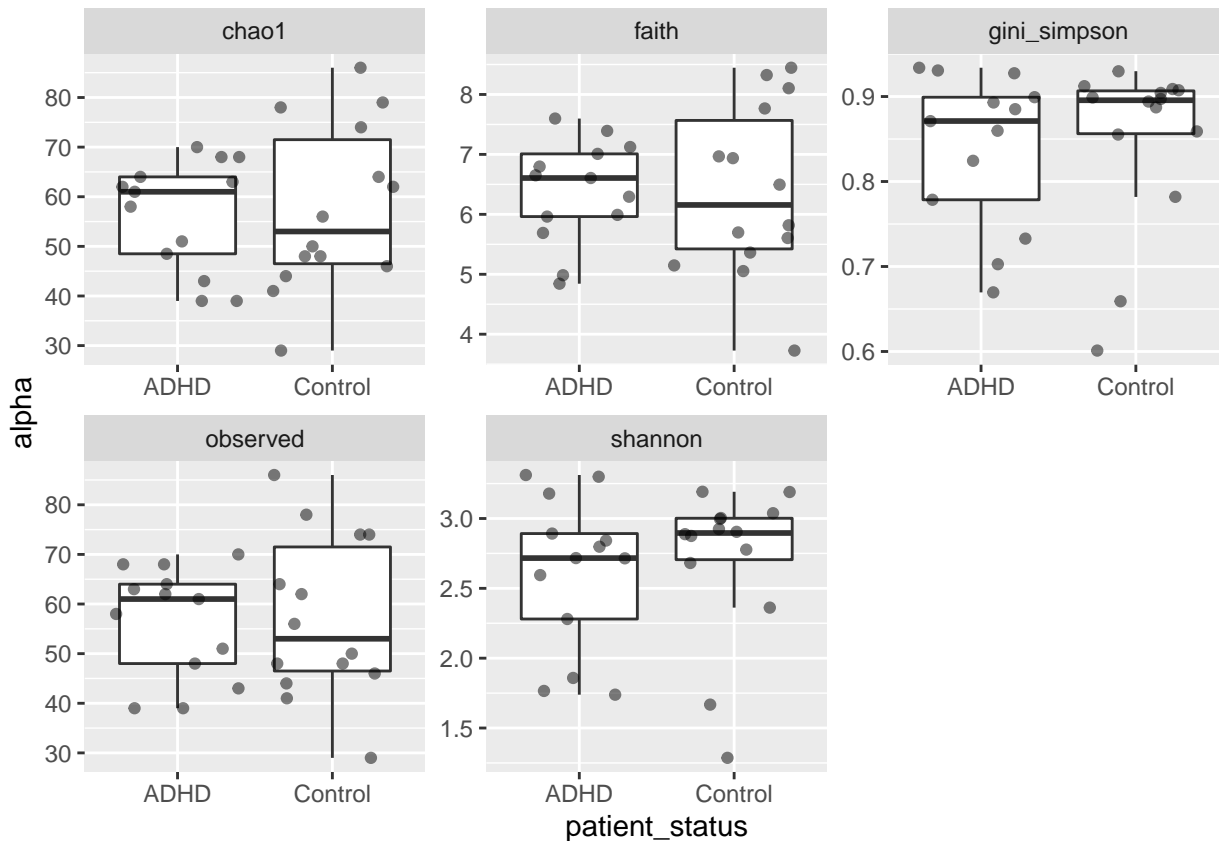


6.3 Comparing alpha diversity

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 = "alpha")
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.

6.4 Exercises

Do “Alpha diversity basics” from the exercises.

Chapter 7

Beta diversity demo

7.1 Visualizations

Lets generate ordination plots with different methods and transformations.

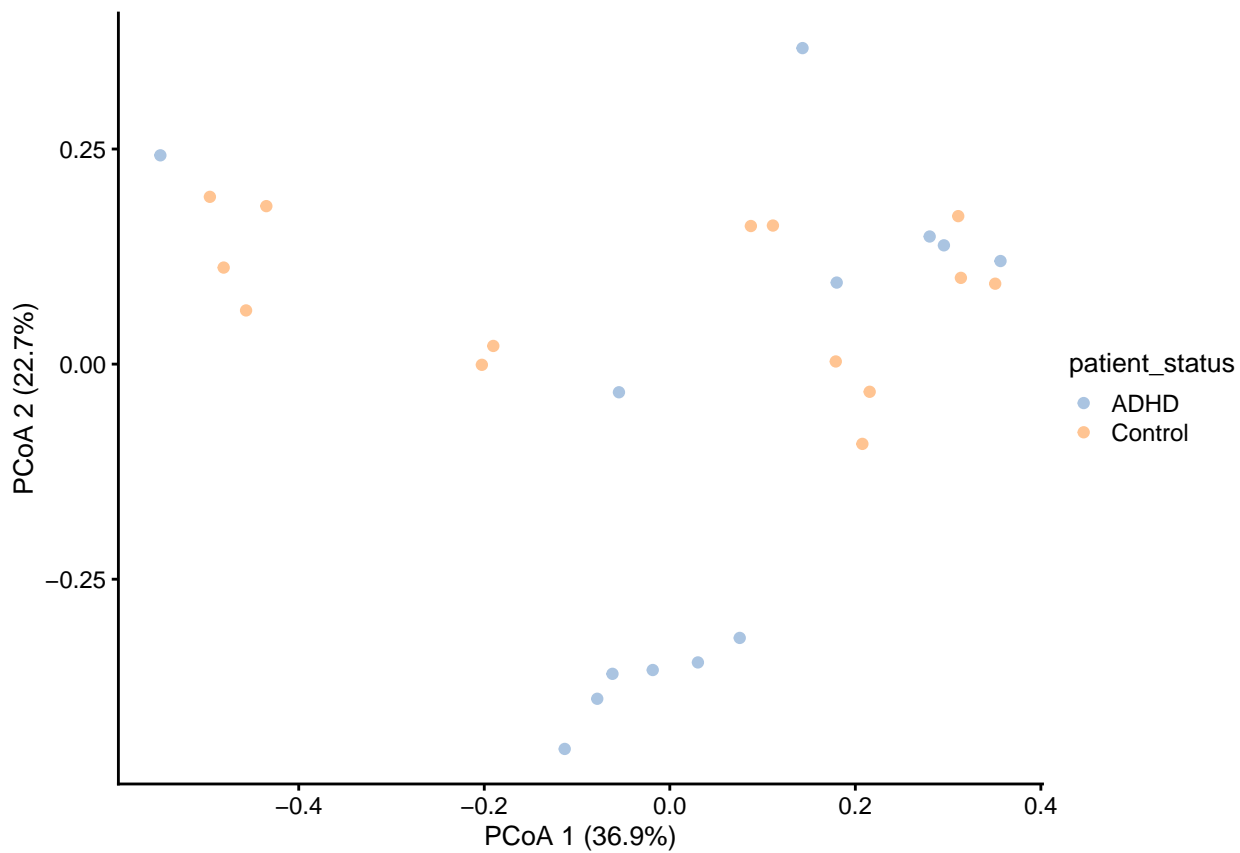
```
#### calculating Bray Curtis dissimilarity and PCoA

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")
```

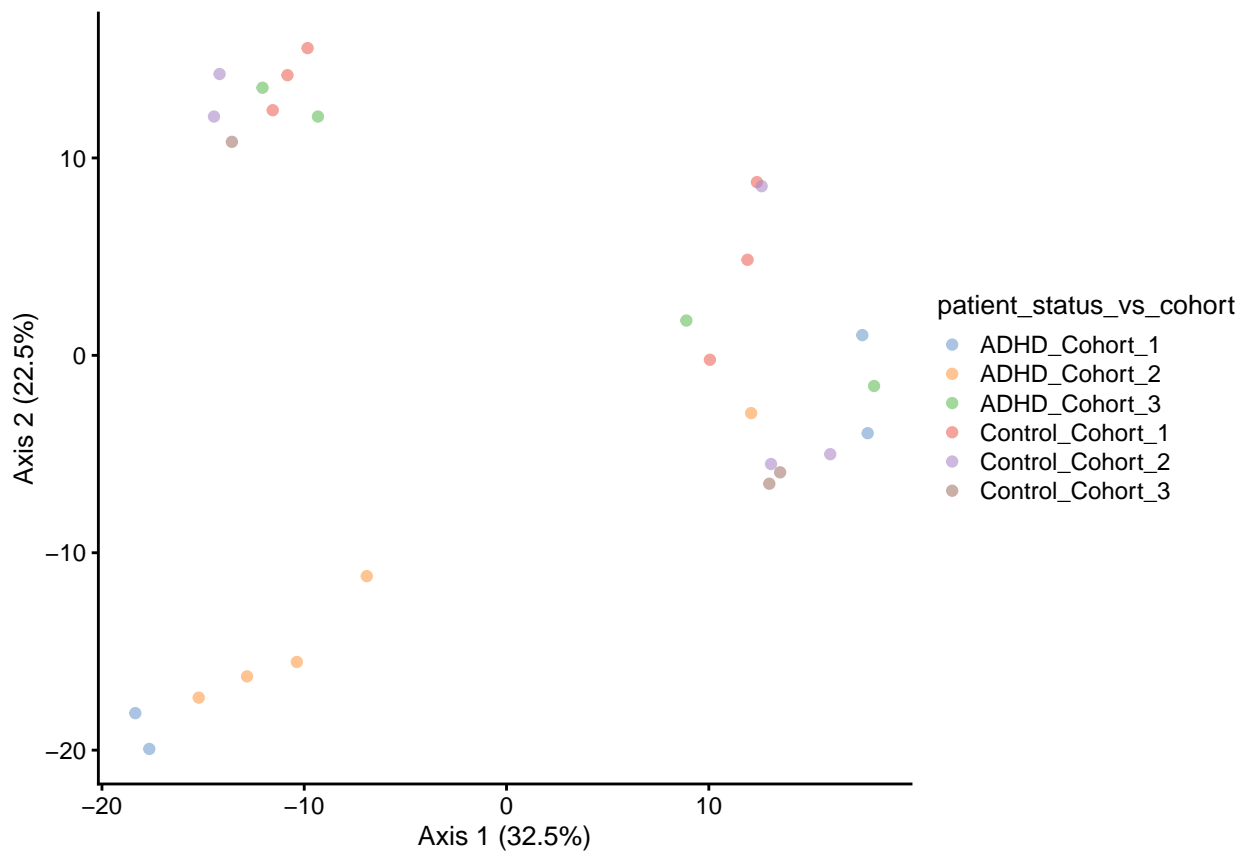
```
# Add explained variance for each axis
```

```
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 = ""))
```

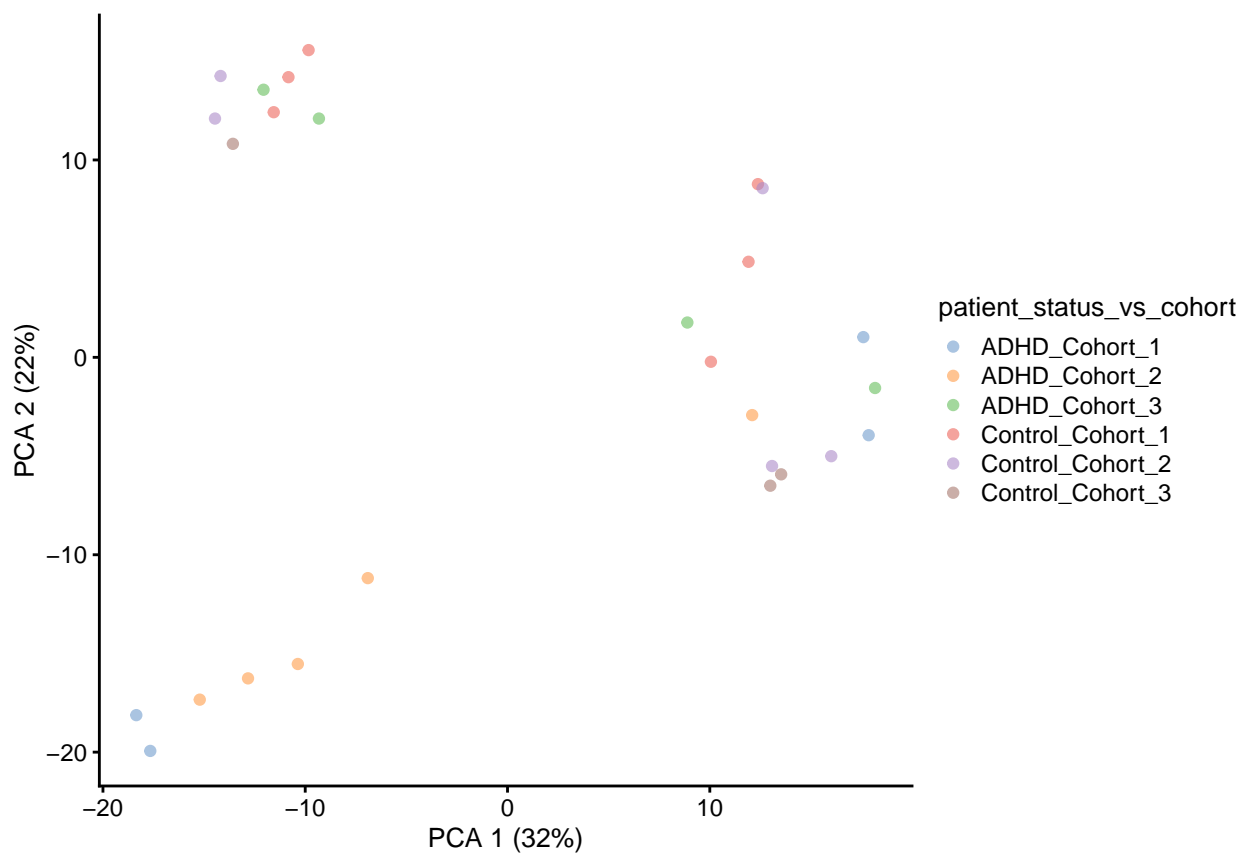
```
print(p)
```



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 adaptability.

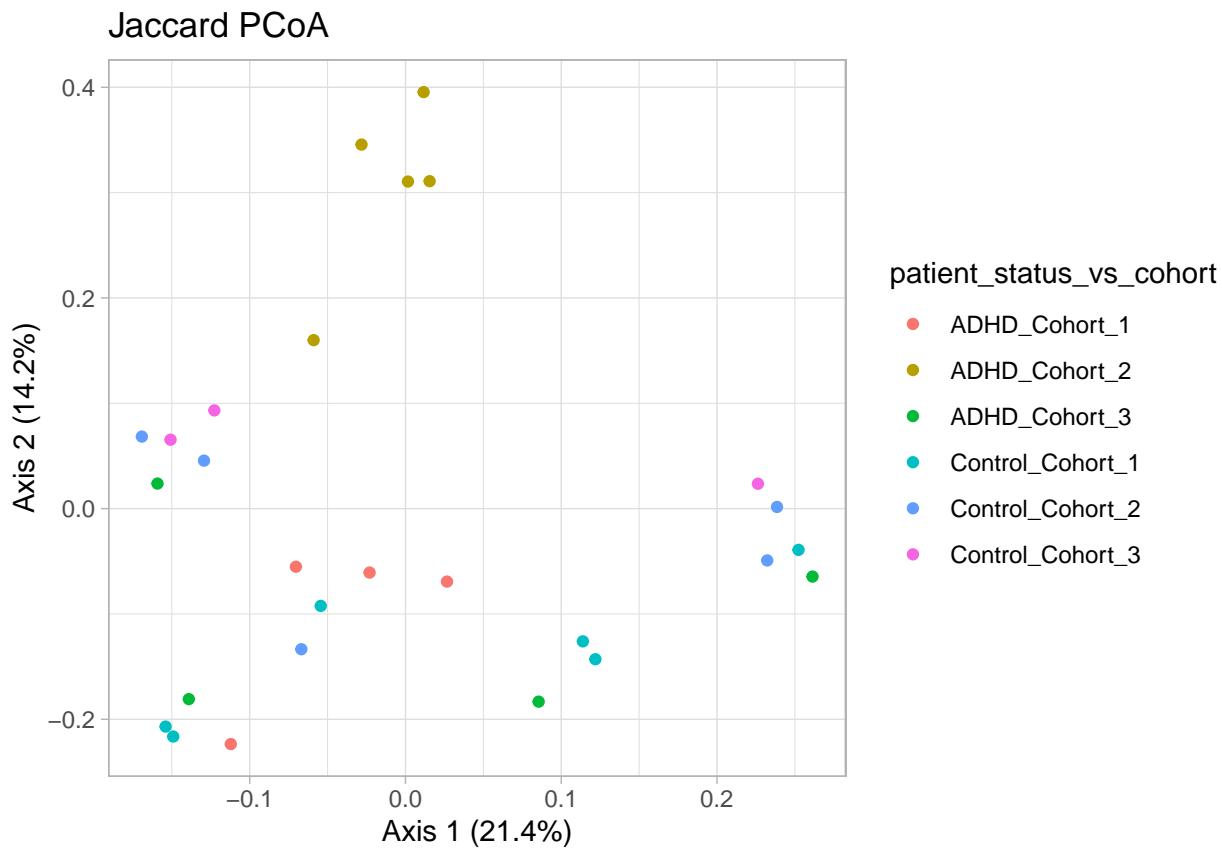
```
dis <- vegan::vegdist(t(assays(tse)$counts), method = "jaccard")

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

# a data frame from principal coordinates and grouping 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), "%", ")"),
       y = paste("Axis 2 (", round(100 * jaccard_pcoa$values[[2]] / sum(jaccard_pcoa$values), 1), "%", ")"),
       title = "Jaccard PCoA") +
  theme(title = element_text(size = 12)) +
  theme_light()

jaccard_plot
```



7.2 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

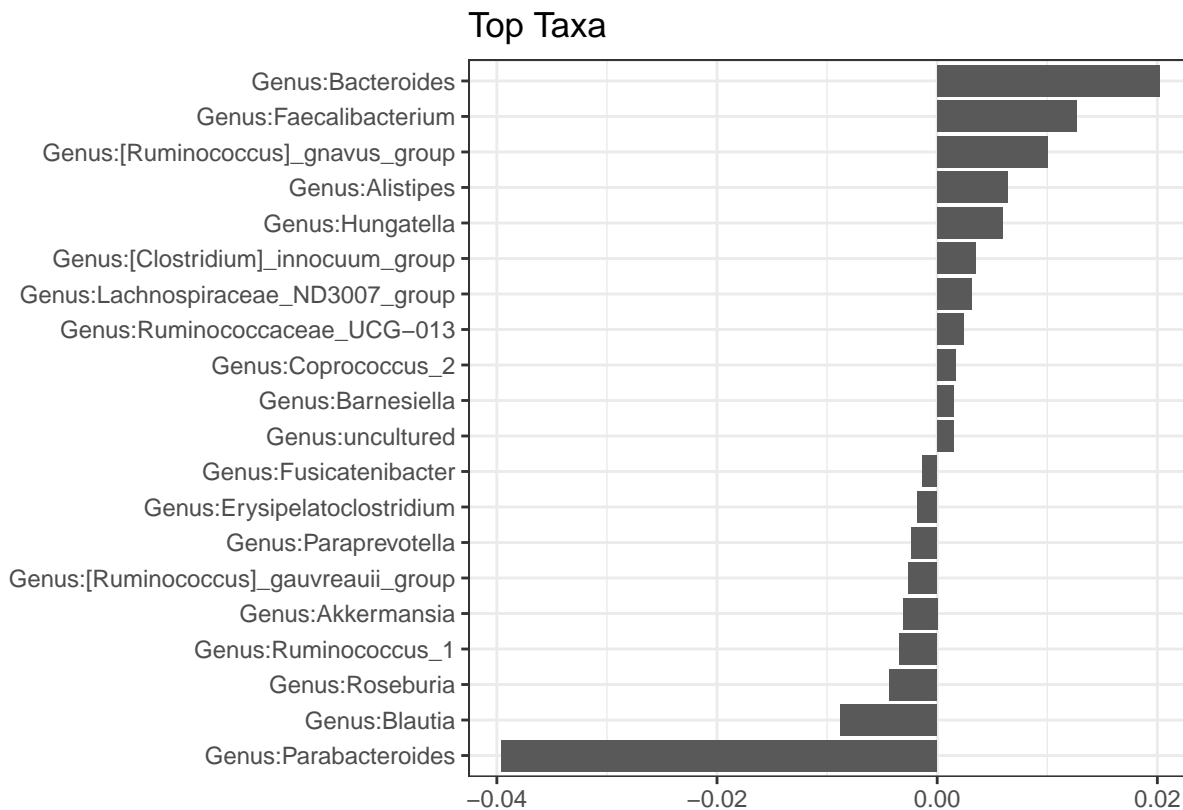
```

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

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

# plot
top_taxa_coefficient_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_coefficient_plot
```



7.2.1 Testing the differences in dispersion

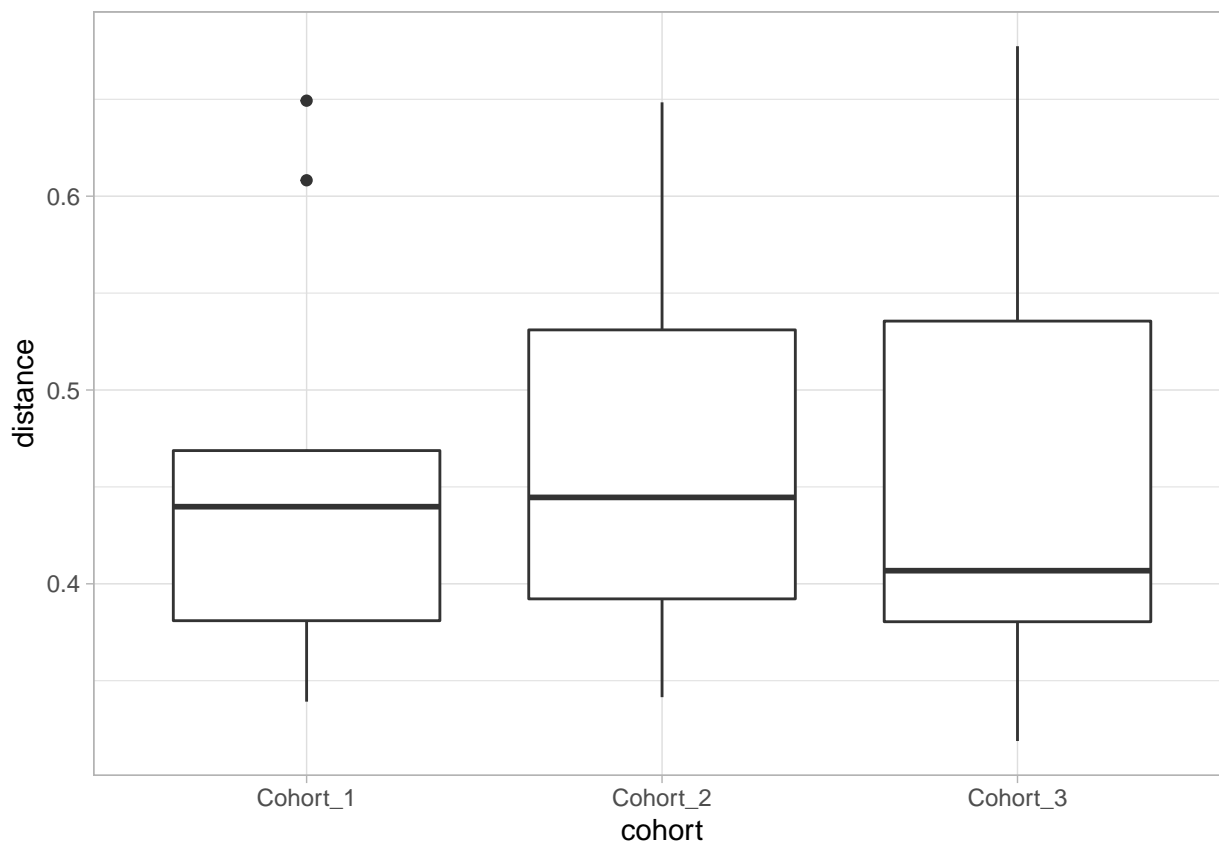
PERMANOVA doesn't differentiate between different within-group variation, i.e. dispersion, or the mean differences between groups, i.e. the location of the centroid. Follow-up testing can be done with PERMDISP2 implemented in the vegan package.

```
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.

7.3 Exercises

Do “Beta diversity” from the exercises.

Chapter 8

Study material

8.1 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.

8.2 Lecture slides

To be added.

8.3 Tasks

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

- Exercises
- Example solutions

8.4 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*.