Microbial Data Analysis Course Part 1

Ece Kartal

Contents

This vignette contains the course schedule and material of the **Microbiome Data Analysis & Visualization Course**. The estimated time to go over the content of the tutorial is of ~2 hours for theoretical part & 4 hours for practical session.

We will discuss the role of the microbiome in human body and the consequences of its dysfunction on the body - What is the microbiome, - How to study microbiome and their limitations - Evolution of microbiome over human lifespan - The human microbiome in health and disease - Applications based on microbiome

Please contact me at ece.kartal@uni-heidelberg.de. Feedback about the tutorial is also highly appreciated.

Before you start

This tutorial is intended to guide users through the statistical analysis of shotgun metagenomics data. We assume the following skills in the audience:

- Basic knowledge of the statistical programming language R.

- Basic knowledge about how to load and make use of external R packages, such as those included in the tidyverse and Bioconductor packages.
- Be familiar with concepts like omics data, biological databases, exploratory data analysis and hypothesis testing.

This tutorial requires:

- R >= 4.1.2 You can download an install R from this link.
- The following R packages are also required:
  - `tidyverse`
  - `knitr`
  - `ggrepel`
  - `pROC`
  - `vegan`
  - `reshape2`
  - `ggplot2`
  - `ggpubr`
  - `car`
  - `dplyr`
  - `plyr`
  - `SIAMCAT`
  - Rstudio is highly recommended to open, run and modify the R code that we will use in this tutorial. You can download and install Rstudio from this link. Please install RStudio Desktop, Open Source Edition, which is free thanks to its Open Source License.

Install and load required R libraries

- Please clone the content of the following repository in your computer **the day before course**: https://github.com/saezlab/Microbiome_analysis_course_2022.git
- The following code chunk takes care of checking and installing those packages in your R installation. Please run it from your computer:

Before we start with any analysis, it is good practice to load all required libraries. This will also immediately identify libraries that may be missing. Note that for this course, we pre-installed all libraries for you. When you run your own analysis, you have to check which libraries are already available, and which are not. We use `suppressPackageStartupMessages` here to suppress the output messages of the various packages for reasons of brevity.

When using functions that sample pseudorandom numbers, each time you execute them you will obtain a different result. For the purpose of this vignette, this is not what we want. We therefore set a particular seed value(here: 1880) so that the result will always be the same. For more information, check out this webpage that explains this general concept in more detail.

```
# CRAN packages
cran_packages <- c("tidyverse", "BiocManager", "knitr", "ggrepel", "pROC", "vegan",
                   "reshape2", "ggplot2", "ggpubr", "car",  "dplyr", "plyr")
for (i in cran_packages) {
  if (!require(i, character.only = TRUE))
```

```
    install.packages(i)
}

# BioC packages
bioc_packages <-
  c("SIAMCAT")
for (i in bioc_packages) {
  if (!require(i, character.only = TRUE))
    BiocManager::install(i, update = FALSE)
}
```

```
suppressPackageStartupMessages({
  library(knitr)
  library(tidyverse)
  library(ggplot2)
  library(ggpubr)
  library(car)
  library(vegan)
  #library(Rarefy)
})

set.seed(1880)


# Set parameters
PARAM <- list()
PARAM$folder.R <- paste0(getwd(), "/")
PARAM$folder <- gsub("src/", "", PARAM$folder.R)
PARAM$folder.data <- paste0(PARAM$folder, "data/")
PARAM$folder.output <- paste0(PARAM$folder, "output/")

statusColor=c("#8dd3c7", "#E69F00", "#fb9a99", "#1f78b4",
                  "#33a02c", "#beaed4")

levels=c("PC", "CTR")
threshold <- 10^-5
```

The biology of the tutorial

In this tutorial we will work with a subset of the shotgun metagenomics data available in the ENA entry PRJEB38625. This entry contains the data accompanying the publication entitled: "A faecal microbiota signature with high specificity for pancreatic cancer". For simplicity, in this tutorial we will compare the pancreatic cancer patients (N=57) to healthy controls (N=50) from a Spanish study cohort.

Metagenomics

Shotgun metagenomics sequencing captures the microbial communities contained in a sample. It allows us to understand the extraordinary diversity present within microbial communities which is limited via standard culturing approaches.

Load shotgun metagenomics data into R and setup output directories

Next, we read the data tables that contain the count matrix and samples' metadata. **count matrix** This is a matrix that contains samples as columns and feature names (species in our case) as rows.

```
# set the working directory
folder <- gsub("/src", "", getwd())
folder.results <- paste0(folder, "/results/")
file.path(folder, 'data/mobi.Rdata')
```

```
## [1] "/Users/ecekartal/Documents/Academics-Work/Teaching/Microbiome_analysis_course_2023/data/mobi.Rda
```

```
# load data
load(file=file.path(folder, 'data/mobi.Rdata'))
```

We can also explore the content of the metadata table, which contain the properties/annotations for each sample.

```
meta[1:10, 1:10]
```

```
##                 environment_material status jaundice jaundice_imp diabetes
## MMPC35551931ST feces [ENVO:00002003]     PC        1            0        0
## MMPC41376408ST feces [ENVO:00002003]     PC        1            0        0
## MMPC59659730ST feces [ENVO:00002003]     PC        1            0        0
## MMPC96048560ST feces [ENVO:00002003]     PC        1            1        0
## MMPC42296680ST feces [ENVO:00002003]     PC        1            0        0
## MMPC49776422ST feces [ENVO:00002003]     PC        1            1        1
## MMPC19228810ST feces [ENVO:00002003]     PC        1            1        0
## MMPC45272194ST feces [ENVO:00002003]     PC        1            0        1
## MMPC36793964ST feces [ENVO:00002003]     PC        1            0        0
## MMPC92286162ST feces [ENVO:00002003]     PC        1            0        0
##                library_size center gender periodontitis age
## MMPC35551931ST     11417135      2      0             1  79
## MMPC41376408ST     22532306      2      0             0  62
## MMPC59659730ST     22296806      2      1             0  69
## MMPC96048560ST     15812641      2      0             1  54
## MMPC42296680ST     12998378      2      1             0  68
## MMPC49776422ST     24362813      2      0             1  68
## MMPC19228810ST     15320194      2      0            NA  71
## MMPC45272194ST     19734216      2      0             0  84
## MMPC36793964ST     12388494      2      1             1  69
## MMPC92286162ST     12858226      2      0             1  66
```

```
meta$ID <- rownames(meta)
```

Quality control and pre-processing

Checking library size

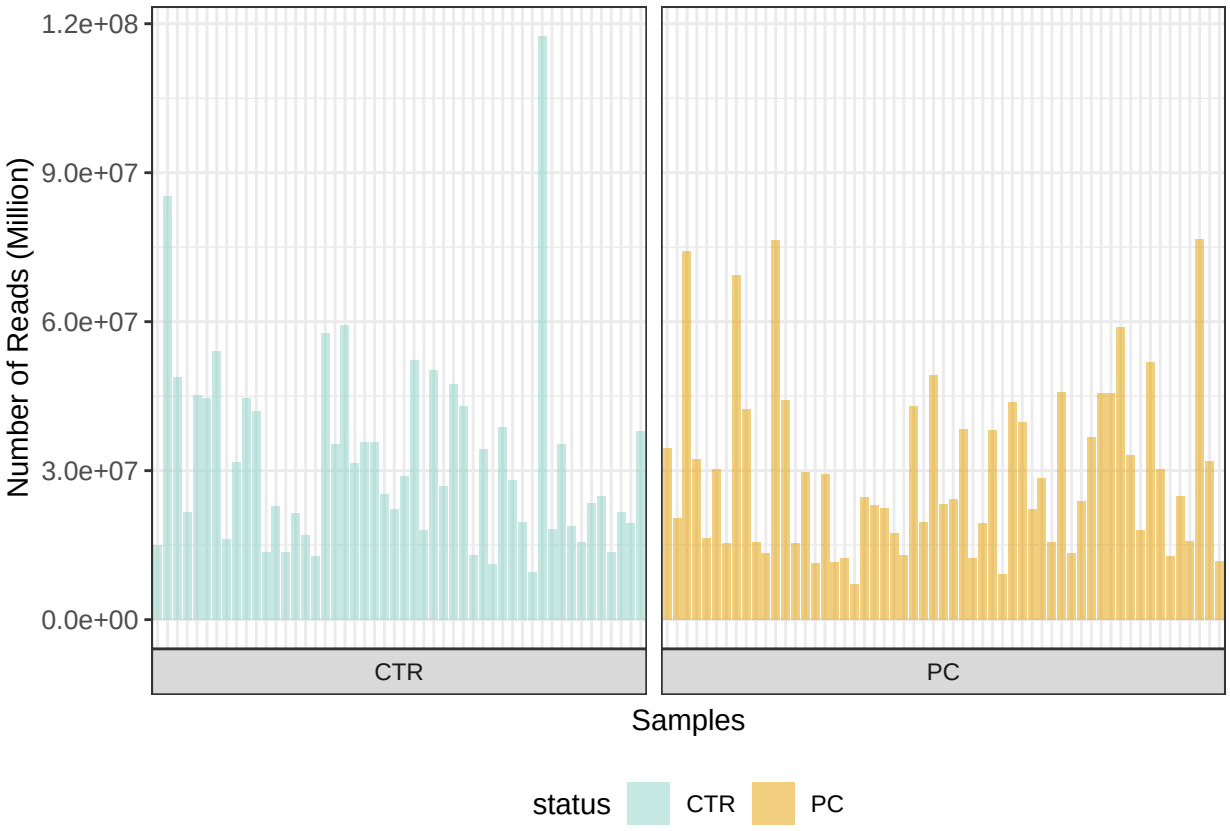
Before being able to compare data from both cancer and controls, we should perform basic quality control analyses and some pre-processing of the count matrix. In a first step, we can take a look to the total number

of sequences per sample, which is also known as library size. This provides information about the sequencing depth and is a good indicator of differences or batch effects in the sequencing process.

```
p.readcount <- meta %>%
  ggplot(aes(y = library_size,
             x = factor(ID),
             fill = as.factor(status))) +
  geom_col(stat = "identity",
           alpha = 0.5,
           position = position_dodge2(width = 0.9, preserve = "single")) +
  scale_fill_manual(name = "status",
                    values = statusColor) +
  facet_grid(. ~ status,
             scales = "free_x",
             space = "free_x",
             switch = "x") +
  xlab("Samples") +
  ylab("Number of Reads (Million)") +
  theme_bw() +
  theme(
    axis.text.x=element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.y = element_text(size = 10),
    legend.position = "bottom")
```

```
## Warning in geom_col(stat = "identity", alpha = 0.5, position =
## position_dodge2(width = 0.9, : Ignoring unknown parameters: ‘stat‘
```

```
p.readcount
```




```
ggsave(p.readcount,
       filename=paste0(PARAM$folder.output, "readcounts.pdf"),
       width = 16,
       height = 7)
```

Figure 1. Total read count per sample

NOTE: Inside this code chunk, we employ several functions. First, we create a data frame with the id of the samples and library size for each of them using the `data.frame()` function. Next, we pass this data frame object to the `ggplot()` function to start creating the plot. This is done thanks to the pipe operand `%>%`, which "sends" the data frame to the function positioned after it. For more information about the pipe operand please see this documentation page.

NOTE: The `ggplot2` package, which is part of the `tidyverse` collection, is here employed to create the plot. `ggplot2` comprises a powerful yet simple framework to create and edit high-quality graphics. For more information, please see `ggplot2` homepage.

As it can be observed in the plot, the library sizes range from 7 to 117 millions of sequences. This means that the ratio between the largest and the smallest libraries is of ~ 16. This is an acceptable value for most statistical approaches. When this ratio is higher than ~3, ad-hoc adjustments should be made to consider the heterogeneity in library sizes. We will talk more about this in the normalization section of the tutorial.

Filtering low abundant species and samples

Next, the count matrix data can be filtered to remove bacteria which are lowly abundant across conditions or not present at all. The reasons for this are biological as well as statistical.

Firstly, species which are abundant at low levels across the different samples are likely to arise from noise in the sequencing process, or are otherwise not likely to be biologically meaningful and are therefore best ignored in the downstream analysis. Secondly, removing species with low counts allows the mean-variance relationship to be more reliably estimated and reduces the number of statistical tests performed during differential analysis. Here, we will keeps species with 10^-5 abundance or more in a minimum number of two samples.

There are a number of way to decrease the number of features:

- Apply an abundance cutoff (such as only looking at taxa that are at least 1% abundance in at least one sample)
- Apply a frequency cutoff (such as only looking at taxa that occur in at least 2% of samples)

```
quantile(colSums(motu.abs))
```

```
##     0%    25%    50%    75%   100%
##    706   1930   6476  10220 154928
```

```
motu.abs.fil <- motu.abs[rowSums(motu.abs >= 10^-5) >= 2,
                         colSums(motu.abs) > 1000]
# print a message to show the number of species that are retrieved after filtering
message(
  "Initial count matrix contained ",
  nrow(motu.abs),
  " species and the resulting count matrix contains ",
  nrow(motu.abs.fil),
  " species.")
```

```
## Initial count matrix contained 14212 species and the resulting count matrix contains 1343 species.
```

Normalization

There are multiple factors that can result in libraries of different sizes. Those include experimental variations, batch effects or simply, different sequencing depths. We assume that, if it were not for these variations, all samples should have a similar range and distribution of abundance. Therefore, after data filtering, a normalization step is necessary to ensure that species abundance can be compared between samples and experimental conditions. Below, we use the relative abundance.

There are other normalisation approaches described in Pereira et al. 2018 & more recently Rico et al., 2021.

```
# apply relative abundance normalization
motu.rel <- prop.table(as.matrix(motu.abs), 2)
motu.fil.rel <- prop.table(as.matrix(motu.abs.fil), 2)
```

And have a look to how the normalized values look like:

```
motu.fil.rel[1:10,1:10]
```

```
##                                     MMPC41376408ST MMPC59659730ST
## Abiotrophia defectiva [r_04788]                  0   0.0000000000
## Absiella dolichum [r_03694]                      0   0.0000000000
```

```
## Acetobacter sp. [r_03587]                          0   0.0003984064
## Acholeplasma sp. CAG:878 [r_07583]                 0   0.0000000000
## Acidaminococcus fermentans [r_03588]               0   0.0000000000
## Acidaminococcus intestini [r_01949]                0   0.0000000000
## Acidaminococcus massiliensis [r_03590]             0   0.0000000000
## Acinetobacter sp. [r_02372]                        0   0.0000000000
## Acinetobacter sp. [r_03673]                        0   0.0000000000
## Acinetobacter species  [m_12662]                   0   0.0000000000
##                                       MMPC96048560ST MMPC42296680ST
## Abiotrophia defectiva [r_04788]          0.000000000              0
## Absiella dolichum [r_03694]              0.000000000              0
## Acetobacter sp. [r_03587]                0.004222147              0
## Acholeplasma sp. CAG:878 [r_07583]       0.000000000              0
## Acidaminococcus fermentans [r_03588]     0.000000000              0
## Acidaminococcus intestini [r_01949]      0.000000000              0
## Acidaminococcus massiliensis [r_03590]   0.000000000              0
## Acinetobacter sp. [r_02372]              0.004391831              0
## Acinetobacter sp. [r_03673]              0.000000000              0
## Acinetobacter species  [m_12662]         0.004601441              0
##                                       MMPC49776422ST MMPC19228810ST
## Abiotrophia defectiva [r_04788]         8.747529e-06              0
## Absiella dolichum [r_03694]             0.000000e+00              0
## Acetobacter sp. [r_03587]               0.000000e+00              0
## Acholeplasma sp. CAG:878 [r_07583]      0.000000e+00              0
## Acidaminococcus fermentans [r_03588]    0.000000e+00              0
## Acidaminococcus intestini [r_01949]     0.000000e+00              0
## Acidaminococcus massiliensis [r_03590]  0.000000e+00              0
## Acinetobacter sp. [r_02372]             0.000000e+00              0
## Acinetobacter sp. [r_03673]             0.000000e+00              0
## Acinetobacter species  [m_12662]        0.000000e+00              0
##                                       MMPC45272194ST MMPC36793964ST
## Abiotrophia defectiva [r_04788]         0.000000e+00     0.00000000
## Absiella dolichum [r_03694]             0.000000e+00     0.00000000
## Acetobacter sp. [r_03587]               9.148126e-06     0.01006971
## Acholeplasma sp. CAG:878 [r_07583]      0.000000e+00     0.00000000
## Acidaminococcus fermentans [r_03588]    0.000000e+00     0.00000000
## Acidaminococcus intestini [r_01949]     0.000000e+00     0.00000000
## Acidaminococcus massiliensis [r_03590]  0.000000e+00     0.00000000
## Acinetobacter sp. [r_02372]             0.000000e+00     0.00000000
## Acinetobacter sp. [r_03673]             0.000000e+00     0.00000000
## Acinetobacter species  [m_12662]        0.000000e+00     0.00697134
##                                       MMPC92286162ST MMPC23673679ST
## Abiotrophia defectiva [r_04788]          0.000000000    0.000000000
## Absiella dolichum [r_03694]              0.000000000    0.000000000
## Acetobacter sp. [r_03587]                0.000000000    0.000000000
## Acholeplasma sp. CAG:878 [r_07583]       0.000000000    0.000000000
## Acidaminococcus fermentans [r_03588]     0.000000000    0.000000000
## Acidaminococcus intestini [r_01949]      0.005805515    0.000000000
## Acidaminococcus massiliensis [r_03590]   0.000000000    0.000000000
## Acinetobacter sp. [r_02372]              0.000000000    0.006908538
## Acinetobacter sp. [r_03673]              0.000000000    0.000000000
## Acinetobacter species  [m_12662]         0.000000000    0.012200739
```

Diversity Analysis

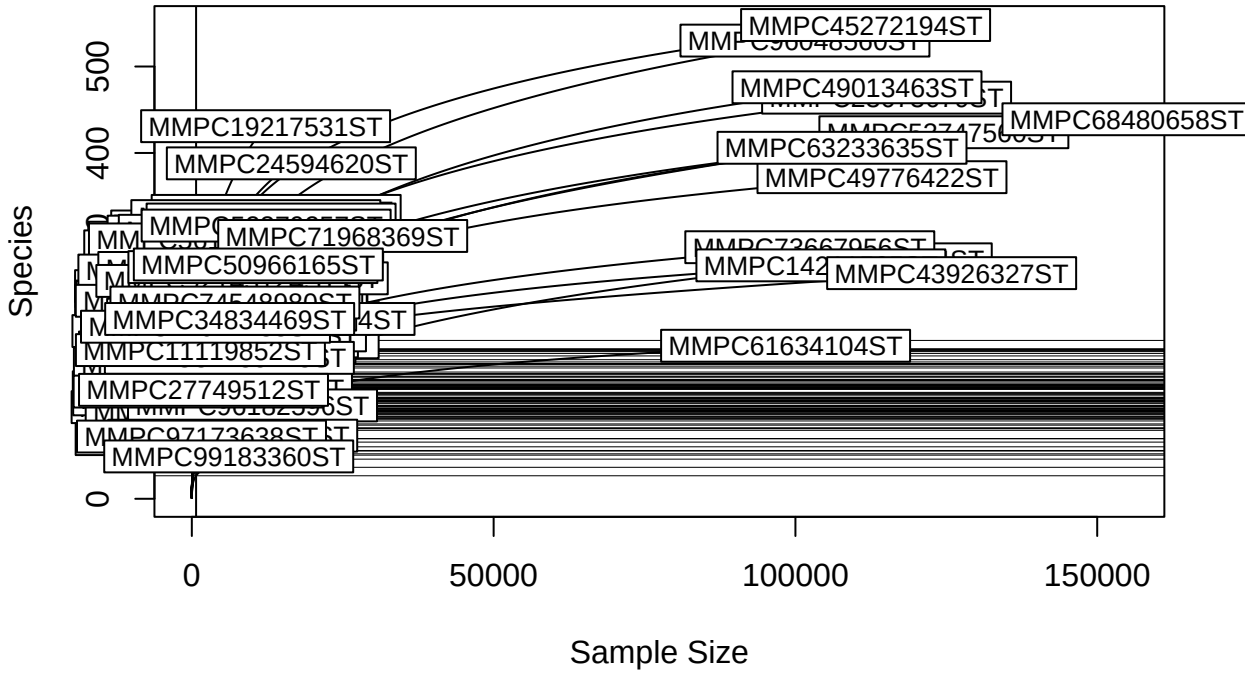
Diversity in the ecological sense is intuitively understood as the complexity of a community of organisms.The two main categories of methods are known as **alpha diversity** and **beta diversity**

Alpha (within sample) Diversity Analysis

Alpha diversity measures the diversity within a single sample and is generally based on the number and relative abundance of taxa at some rank **Shannon**: How difficult it is to predict the identity of a randomly chosen individual. **Simpson**: The probability that two randomly chosen individuals are the same species. **Inverse Simpson**: This is a bit confusing to think about. Assuming a theoretically community where all species were equally abundant, this would be the number of species needed to have the same Simpson index value for the community being analyzed.

The `diversity` function from the vegan package can be used to calculate the alpha diversity of a set of samples.

```
## [1] 1596  107
```




```
# transform count matrix
motu.t = t(motu.abs.fil)

shared= motu.t %>%
  as_tibble(rownames="ID") %>%
  pivot_longer(-ID)
```

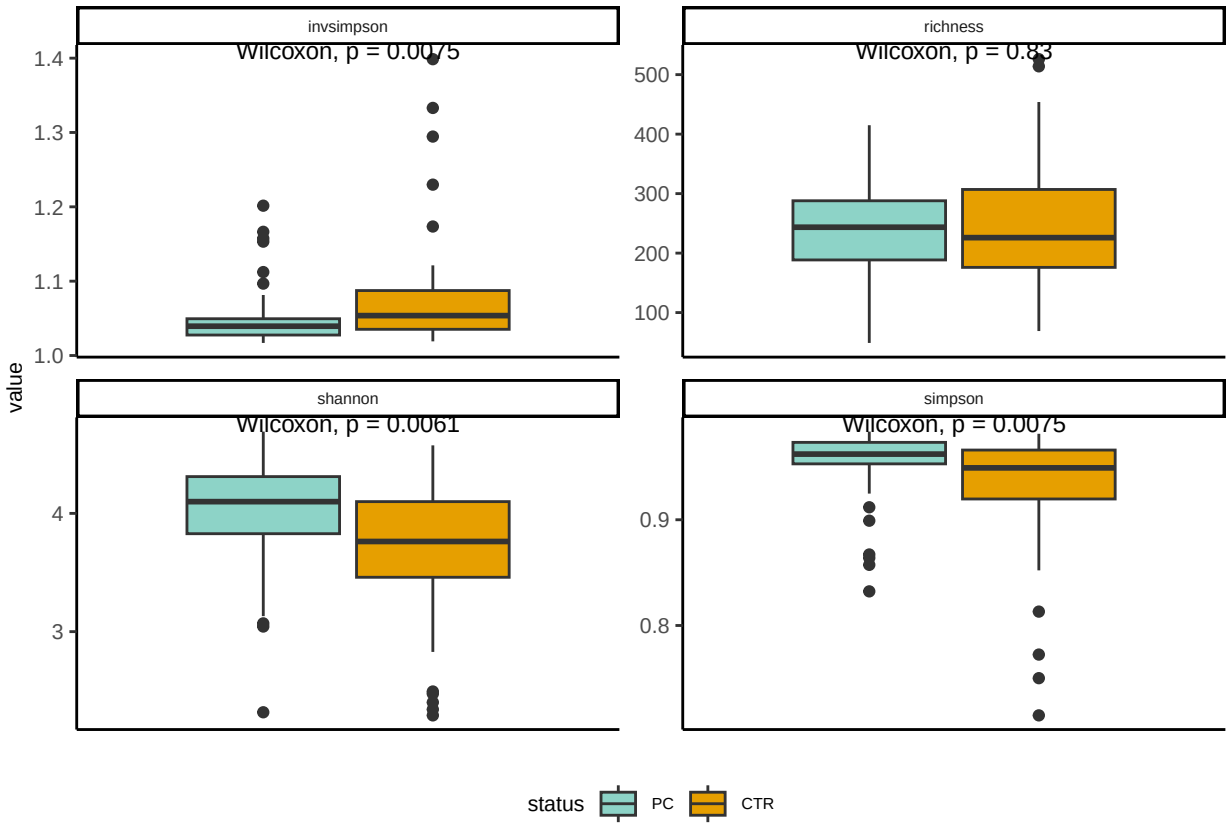
```r
alpha.div <- shared %>%
  dplyr::group_by(ID) %>%
  dplyr::summarize(richness = specnumber(value),
            shannon = diversity(value, index="shannon"),
            simpson = diversity(value, index="simpson"),
            invsimpson = 1/simpson,
            n = sum(value)) %>%
  pivot_longer(cols=c(richness, shannon, invsimpson, simpson), names_to="metric")


# add status info
alpha.div.com <- left_join(alpha.div, meta, by="ID")
```

Lets have a look how alpha diversity looks like

```r
# to compare indices between different patient status, boxplot is suitable
p.alpha <- alpha.div.com %>%
    ggplot(aes(x=metric, y=value, fill=status)) +
    geom_boxplot() +
    facet_wrap(. ~ metric, scale="free") +
  stat_compare_means(size=3) +# wilcox.test p-value
    theme_classic()+
    labs(x = "")+
    theme(text = element_text(size=8),
          axis.text.y = element_text(size=8),
          axis.title.x=element_blank(),
          axis.text.x=element_blank(),
          axis.ticks.x=element_blank(),
          legend.position = "bottom") +
    scale_fill_manual(name = "status",
                      labels = levels,
                      values=statusColor)

p.alpha
```



QUESTION: Did you notice a pattern for samples with high read counts?

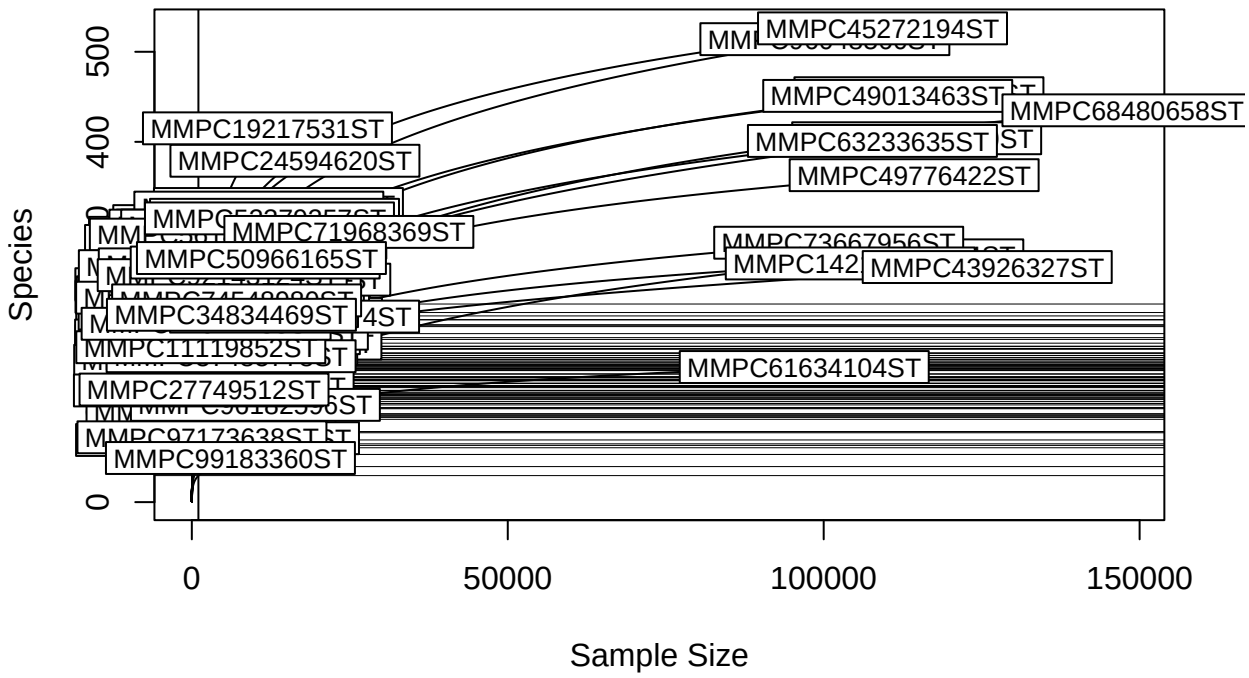
Rarefaction

Rarefaction is used to simulate even numbers of samples (i.e. reads). Even sampling is important:

- When comparing diversity of samples, more samples make it more likely to observe rare species. This will have a larger effect on some diversity indexes than others, depending on how they weigh rare species. Therefore, when comparing the diversity or similarity of samples, it is important to rarefy, or subsample, to a constant depth. Typically, the depth chosen is the minimum sample depth. If the minimum depth is very small, the samples with the smallest depth can be removed and the minimum depth of the remaining samples can be used.

```
# rarefy based on minimum number of reads
min_seq = min(rowSums(motu.t))

# plot rarefaction curves
rarecurve.data <- rarecurve(motu.t, step = 50, sample = min_seq)
```



We will use `rrarefy` function from vegan package for rarefaction

```
# transform, rarefy
feat.rare <- rrarefy(motu.t, min_seq)
feat.rare <- t(as.data.frame(feat.rare))
```

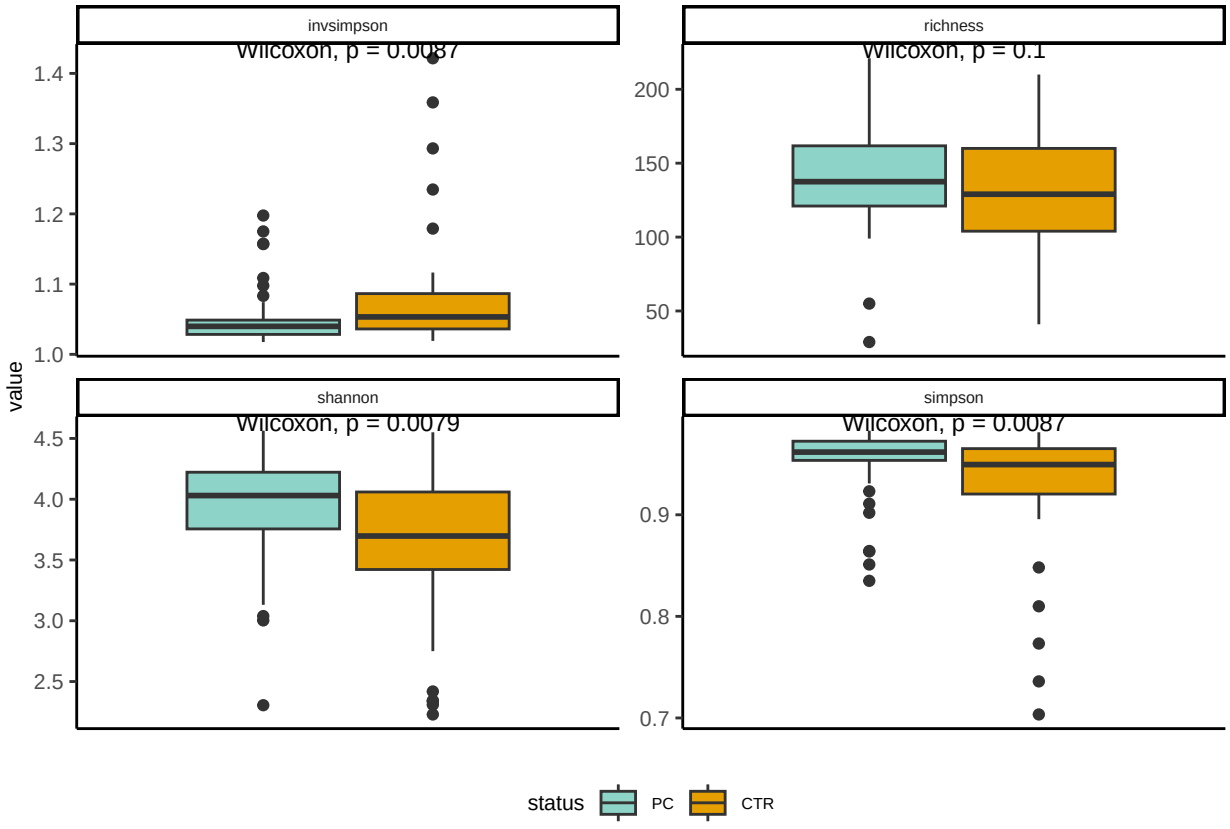
```
shared= t(feat.rare) %>% as_tibble(rownames="ID") %>% pivot_longer(-ID)

alpha.div.r <- shared %>%
  dplyr::group_by(ID) %>%
  dplyr::summarize(richness = specnumber(value),
            shannon = diversity(value, index="shannon"),
            simpson = diversity(value, index="simpson"),
            invsimpson = 1/simpson,
            n = sum(value)) %>%
  pivot_longer(cols=c(richness, shannon, invsimpson, simpson),
               names_to="metric")


# add status info
alpha.div.rar <- left_join(alpha.div.r, meta, by="ID")
```

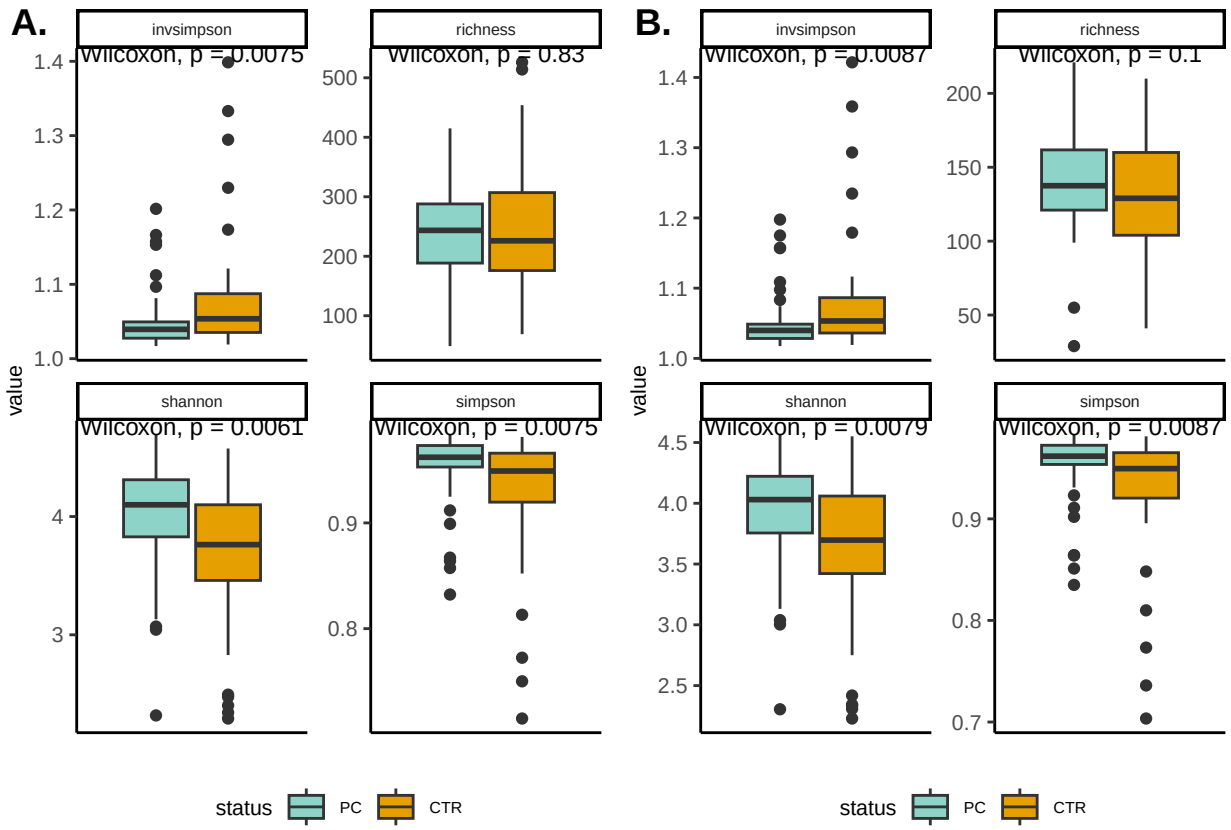
In general, you will see roughly normal distribution for Shannon's diversity as well as most richness metrics. Simpson's diversity, on the other hand, is usually skewed. So most will use inverse Simpson (1/Simpson) instead. This not only increases normalcy but also makes the output more logical as a higher inverse Simpson value corresponds to higher diversity.

```
# to compare indices between different patient status, boxplot is suitable.
# wilcox.test is a non-parametric test that doesn't make specific assumptions about the distribution, w
p.alpha.rar <- alpha.div.rar %>%
  ggplot(aes(x=metric, y=value, fill=status)) +
  geom_boxplot() +
  stat_compare_means(size=3) +
  facet_wrap(. ~ metric, scale="free") +
  theme_classic()+
  labs(x = "")+
  theme(text = element_text(size=8),
          axis.text.y = element_text(size=8),
          axis.title.x=element_blank(),
          axis.text.x=element_blank(),
          axis.ticks.x=element_blank(),
          legend.position = "bottom") +
    scale_fill_manual(name = "status",
                      labels = levels,
                      values=statusColor)
p.alpha.rar
```




```
# print both plots
figure.alpha <- ggarrange(p.alpha, p.alpha.rar,
                    labels = c("A.", "B."),
                    ncol = 2, nrow = 1)+
  xlab("")
```

```
figure.alpha
```




```
# save plot
ggsave(figure.alpha, filename=paste0(PARAM$folder.output,
                                     "alpha.div.pdf"),
       height = 11, width = 18, unit="cm")
```

We can use **analysis of variance (ANOVA)** to tell if at least one of the diversity means is different from the rest. Overall, for **alpha-diversity**:

- ANOVA, t-test, or general linear models with the normal distribution are used when the data is roughly normal
- Kruskal-Wallis, Wilcoxon rank sum test, or general linear models with another distribution are used when the data is not normal

However, our sample size is small and normalcy tests are very sensitive for small data-sets.

```
# Do ANOVA
metatest = c("smoking", "status","diabetes", "center", "gender", "status",
             "antibiotic", "periodontitis", "age")

alpha.div.anova <- alpha.div.com %>%
  pivot_wider(names_from = metric, values_from = value)
```

```
collect.confounders <- data.frame()

for (metavar in metatest) {
# calculate anova for stool
lm <- lm(substitute(richness~ as.factor(metavar),
                    list(metavar = as.name(metavar))),
         data = alpha.div.anova, na.action=na.omit)

aov <- Anova(lm) %>% broom::tidy() %>%
  mutate(metric="richness")
  aov <- aov[1,]
  aov$term <- metavar

  # collect data
  collect.confounders <- rbind(collect.confounders, aov)
}

# fdr correction
collect.confounders$p.adj <- p.adjust(collect.confounders$p.value)

# Lets have a look to results
collect.confounders
```

```
## # A tibble: 9 x 7
##   term            sumsq    df statistic p.value metric    p.adj
##   <chr>           <dbl> <dbl>     <dbl>   <dbl> <chr>     <dbl>
## 1 smoking         2734.     1     0.337  0.563  richness 1
## 2 status          2085.     1     0.248  0.620  richness 1
## 3 diabetes       34291.     1     4.21   0.0429 richness 0.343
## 4 center         53176.     1     6.71   0.0110 richness 0.0988
## 5 gender         33736.     1     4.16   0.0440 richness 0.343
## 6 status          2085.     1     0.248  0.620  richness 1
## 7 antibiotic      5381.     1     0.636  0.427  richness 1
## 8 periodontitis  17617.     1     2.12   0.149  richness 0.892
## 9 age           359361.    40     1.13   0.329  richness 1
```

Beta (between sample) Diversity Analysis

Beta diversity is a way to quantify the difference between two communities. There are many metrics that are used for this (`manhattan`, `euclidean`, `canberra`, `bray`, `kulczynski`, `jaccard`, `gower`, `altGower`, `morisita`, `horn`, `mountford`, `raup` , `binomial`, `chao`, `cao` or `mahalanobis`), but we will only mention a few of the more popular ones.

- Indexes used with presence/absence data: *Jaccard*: the number of species common to both communities divided by the number of species in either community. *Unifrac*: The fraction of the phylogenetic tree branch lengths shared by the two communities.
- Indexes used with count data: *Bray–Curtis*: The sum of lesser counts for species present in both communities divided by the sum of all counts in both communities. This can be thought of as a quantitative version of the Sørensen index. *Weighted Unifrac*: The fraction of the phylogenetic tree branch lengths shared by the two communities, weighted by the counts of organisms, so more abundant organisms have a greater influence.

The vegan function `vegdist` is used to compute dissimilarity indexes. Since this is a pairwise comparison, the output is a triangular matrix. In R, a matrix is like a data.frame, but all of the same type (e.g. all numeric), and has some different behavior.

Bray-Curtis takes into account species presence/absence, as well as abundance, whereas other measures (like Jaccard) only take into account presence/absence and UniFrac incorporates phylogenetic information.

```
# calculate not rarefied beta diversity
beta_dist <- vegan::vegdist(t(motu.abs.fil), index = "bray")
```

Non-metric Multi-dimensional Scaling (NMDS) is a way to condense information from multidimensional data (multiple variables/species/OTUs), into a 2D representation or ordination. In an NMDS plot generated using an count table the points are samples. The closer the points/samples are together in the ordination space, the more similar their microbial communities.

- NMDS plots are non-metric, meaning that among other things, they use data that is not required to fit a normal distribution. This is handy for microbial ecologists because the majority of our data has a skewed distribution with a long tail. In other words, there are only a few abundant species, and many, many species with low abundance (the long tail).
- What makes an NMDS plot non-metric is that it is rank-based. This means that instead of using the actual values to calculate distances, it uses ranks. So for example, instead of saying that sample A is 5 points away from sample B, and 10 from sample C, you would instead say that: sample A is the "1st" most close sample to B, and sample C is the "2nd" most close.

```
nmds <- metaMDS(beta_dist) %>% scores(display=c("sites")) %>% as_tibble(rownames="ID")
```

```
## Run 0 stress 0.1821367
## Run 1 stress 0.1631865
## ... New best solution
## ... Procrustes: rmse 0.05851582  max resid 0.3155062
## Run 2 stress 0.158168
## ... New best solution
## ... Procrustes: rmse 0.04127529  max resid 0.2820274
## Run 3 stress 0.1606996
## Run 4 stress 0.1835877
## Run 5 stress 0.1678957
## Run 6 stress 0.1536716
## ... New best solution
## ... Procrustes: rmse 0.05502511  max resid 0.3696543
## Run 7 stress 0.1576313
## Run 8 stress 0.1555251
## Run 9 stress 0.1711274
## Run 10 stress 0.1724841
## Run 11 stress 0.1533605
## ... New best solution
## ... Procrustes: rmse 0.05159401  max resid 0.3656845
## Run 12 stress 0.1706541
## Run 13 stress 0.1752657
## Run 14 stress 0.1532831
## ... New best solution
## ... Procrustes: rmse 0.0148756  max resid 0.1050825
## Run 15 stress 0.1606997
## Run 16 stress 0.1835861
```

```
## Run 17 stress 0.1589843
## Run 18 stress 0.176003
## Run 19 stress 0.1712891
## Run 20 stress 0.1656964
## *** Best solution was not repeated -- monoMDS stopping criteria:
##     19: stress ratio > sratmax
##      1: scale factor of the gradient < sfgrmin
```

```
# combine metadata and betadiv
meta_nmds <- dplyr::inner_join(meta, nmds)
```

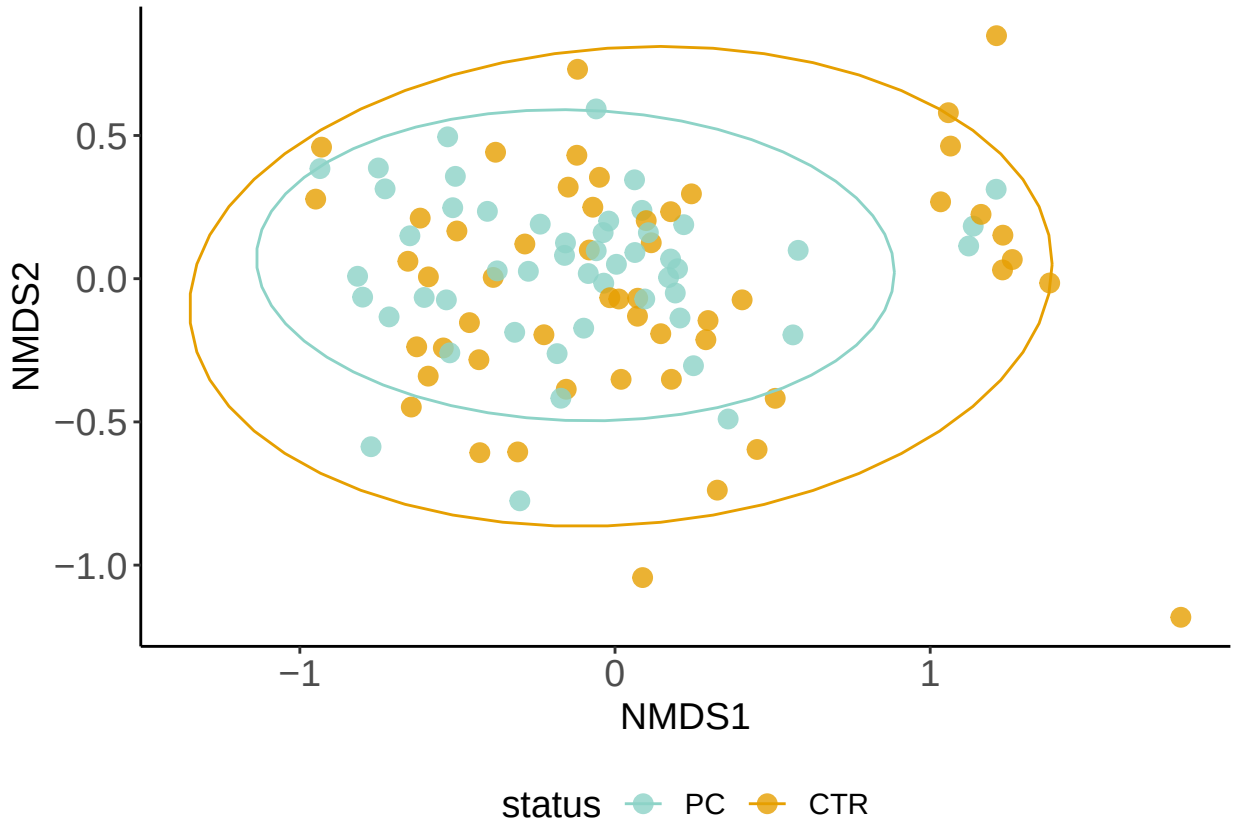
```
## Joining with `by = join_by(ID)`
```

Ordination

The typical way beta diversity is plotted is using ordination. Ordination is a way to display "high dimensional" data in a viable number of dimensions (2 to 3). Our data is "high dimensional" because we have many samples with many species and species can be considered a "dimension". If we had only two species, we could make a scatter plot of their abundance in each sample and get an idea of how the samples differ. With thousands of species, this is not possible. Instead, ordination is used to try to capture the information in many dimensions by in a smaller number of new "artificial" dimensions.

```
p.betadiv.ord <-   meta_nmds %>%
  ggplot( aes(x = NMDS1, y = NMDS2, color = status)) +
  geom_point(size=3, alpha=0.8) +
  scale_color_manual(name = "status",
                     values=statusColor,
                     labels = levels) +
  stat_ellipse() +
  theme_classic() +
  theme(legend.position = "bottom",
        text = element_text(size=14),
        axis.text.x = element_text(size=14),
        axis.text.y = element_text(size=14))

p.betadiv.ord
```



QUESTION: What happens when you change color according to center, gender, diabetes...

QUESTION: Does the difference significant?

```
## [1] 0.278
```

Calculating Rarefied Beta Diversity

`avgdist` function computes the dissimilarity matrix of a dataset multiple times using `vegdist` while randomly subsampling the dataset each time. All of the subsampled iterations are then averaged (mean) to provide a distance matrix that represents the average of multiple subsampling iterations.

```
# calculate rarefied beta diversity
beta_dist.rar <-t(motu.abs.fil) %>%
  vegan::avgdist(dmethod = "bray", sample = min_seq)
nmds.rar <- metaMDS(beta_dist.rar) %>% scores(display=c("sites")) %>% as.tibble(rownames="ID")
```

```
## Warning: ‘as.tibble()‘ was deprecated in tibble 2.0.0.
## i Please use ‘as_tibble()‘ instead.
## i The signature and semantics have changed, see ‘?as_tibble‘.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
## Run 0 stress 0.2386675
```

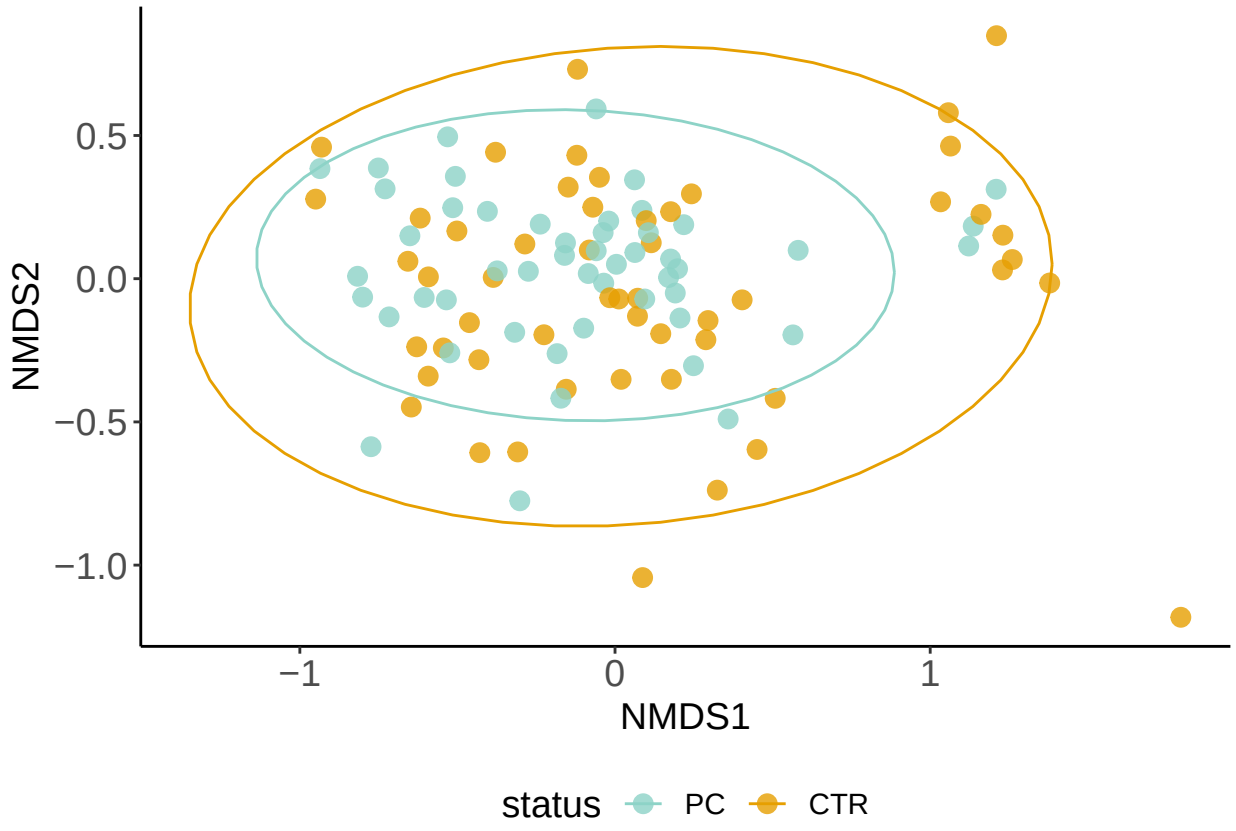
```
## Run 1 stress 0.2359546
## ... New best solution
## ... Procrustes: rmse 0.07855098  max resid 0.2337271
## Run 2 stress 0.2382179
## Run 3 stress 0.2304113
## ... New best solution
## ... Procrustes: rmse 0.07385392  max resid 0.500947
## Run 4 stress 0.2442723
## Run 5 stress 0.2405787
## Run 6 stress 0.23773
## Run 7 stress 0.242768
## Run 8 stress 0.240554
## Run 9 stress 0.2358042
## Run 10 stress 0.23902
## Run 11 stress 0.2349213
## Run 12 stress 0.2365697
## Run 13 stress 0.241056
## Run 14 stress 0.2339657
## Run 15 stress 0.2370215
## Run 16 stress 0.2363435
## Run 17 stress 0.2441011
## Run 18 stress 0.2434136
## Run 19 stress 0.2341675
## Run 20 stress 0.2355794
## *** Best solution was not repeated -- monoMDS stopping criteria:
##     20: stress ratio > sratmax
```

```
# combine metadata and betadiv
meta_nmds.rar <- dplyr::inner_join(meta, nmds.rar)
```

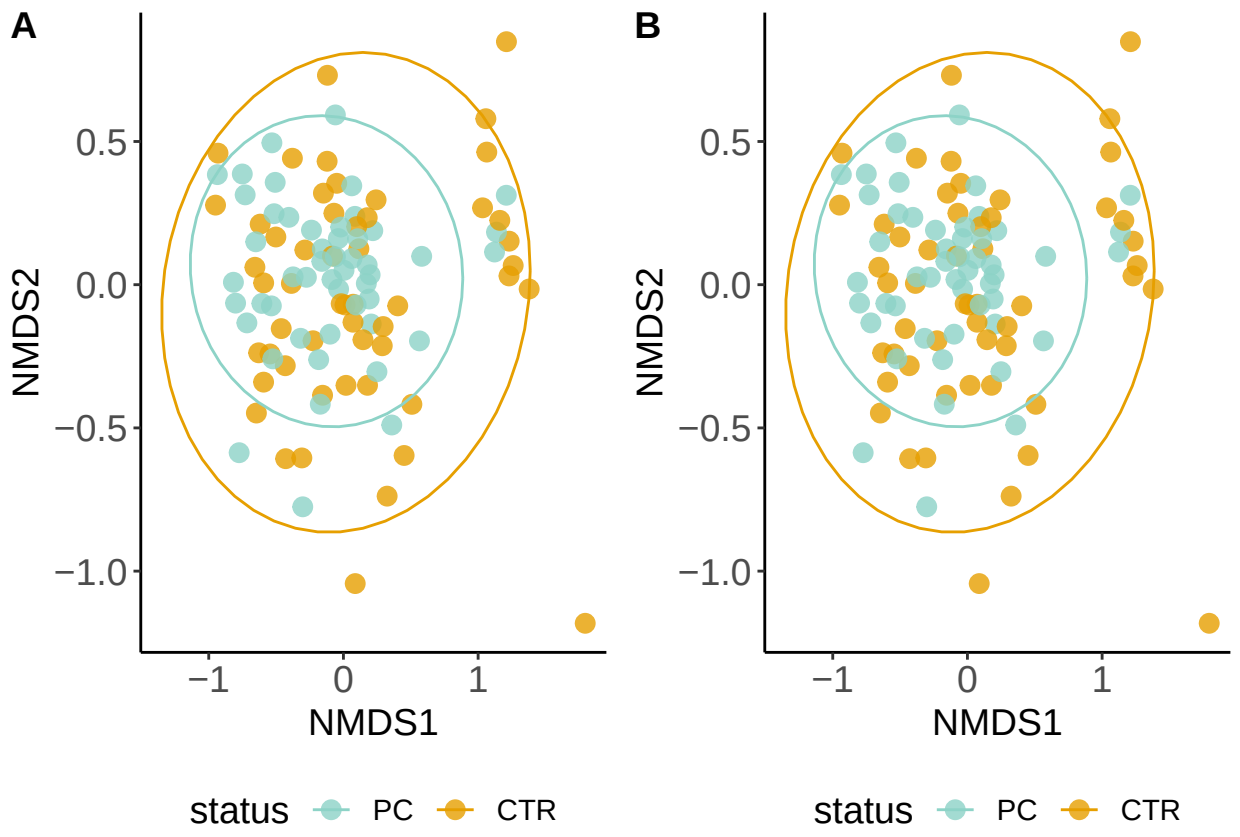
```
## Joining with `by = join_by(ID)`
```

```
p.betadiv.ord.rar <-   meta_nmds %>%
  ggplot( aes(x = NMDS1, y = NMDS2, color = status)) +
  geom_point(size=3, alpha=0.8) +
  scale_color_manual(name = "status",
                     values=statusColor,
                     labels = levels) +
  stat_ellipse() +
  theme_classic() +
  theme(legend.position = "bottom",
        text = element_text(size=14),
        axis.text.x = element_text(size=14),
        axis.text.y = element_text(size=14))

p.betadiv.ord.rar
```




```
# # print both plots
figure <- ggarrange(p.betadiv.ord, p.betadiv.ord.rar,
                    labels = c("A", "B"),
                    ncol = 2, nrow = 1)
figure
```




```
ggsave(figure,
       filename=paste0(PARAM$folder.output, "betadiv.pdf"),
       width = 15, height = 8)
```

QUESTION: What happens when you change color according to center, gender, diabetes...

QUESTION: Does the difference significant?

```
## [1] 0.002
```

QUESTION: What about other meta variables? Change the test variables and see what happens.

Session info

It is good practice to print the so-called session info at the end of an R script, which prints all loaded libraries, their versions etc. This can be helpful for reproducibility and recapitulating which package versions have been used to produce the results obtained above.

```
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Sonoma 14.0
##
```

```
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] vegan_2.6-4     lattice_0.21-9  permute_0.9-7   car_3.1-2
##  [5] carData_3.0-5   ggpubr_0.6.0    lubridate_1.9.3 forcats_1.0.0
##  [9] stringr_1.5.0   dplyr_1.1.3     purrr_1.0.2     readr_2.1.4
## [13] tidyr_1.3.0     tibble_3.2.1    ggplot2_3.4.4   tidyverse_2.0.0
## [17] knitr_1.44
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.4      xfun_0.40         rstatix_0.7.2     tzdb_0.4.0
##  [5] vctrs_0.6.4       tools_4.3.1       generics_0.1.3    parallel_4.3.1
##  [9] fansi_1.0.5       cluster_2.1.4     pkgconfig_2.0.3   Matrix_1.6-1.1
## [13] lifecycle_1.0.3   compiler_4.3.1    farver_2.1.1      textshaping_0.3.7
## [17] munsell_0.5.0     htmltools_0.5.6.1 yaml_2.3.7        crayon_1.5.2
## [21] pillar_1.9.0      MASS_7.3-60       abind_1.4-5       nlme_3.1-163
## [25] tidyselect_1.2.0  digest_0.6.33     stringi_1.7.12    labeling_0.4.3
## [29] splines_4.3.1     cowplot_1.1.1     fastmap_1.1.1     grid_4.3.1
## [33] colorspace_2.1-0  cli_3.6.1         magrittr_2.0.3    utf8_1.2.3
## [37] broom_1.0.5       withr_2.5.1       scales_1.2.1      backports_1.4.1
## [41] timechange_0.2.0  rmarkdown_2.25    ggsignif_0.6.4    ragg_1.2.6
## [45] hms_1.1.3         evaluate_0.22     mgcv_1.9-0        rlang_1.1.1
## [49] glue_1.6.2        rstudioapi_0.15.0 R6_2.5.1          systemfonts_1.0.5
```