Batch effects

Looking for batch effects

This document provides some analyses looking for unwanted data trends (batch effects).

```
library(mbtools)
```

Based on IDs

IDs are sometimes a proxy for the temporal order of samples since many researches assign IDs consecutively to subjects. For that we will first get the reads stratified by sequence and ID.

```
ps <- readRDS("../data/taxonomy.rds")

reads <- taxa_count(ps, lev=NA)[order(sample)]
```

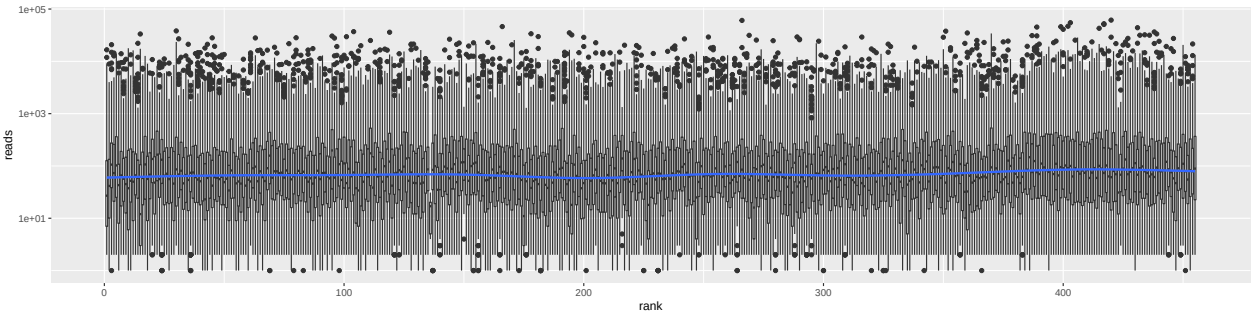
Now we can plot the marginal distributions of the ordered samples. IDs are not evenly spaced so we will rather use their rank.

```
library(ggplot2)

reads$rank <- as.numeric(factor(reads$sample))

ggplot(reads[reads > 0], aes(x=rank, y=reads)) +
    geom_boxplot(aes(group=rank)) + stat_smooth() + scale_y_log10()
```

```
## `geom_smooth()` using method = 'gam'
```



There is nothing striking here. Maybe a tiny effect where low IDs have a higher fraction of low abundance sequences.

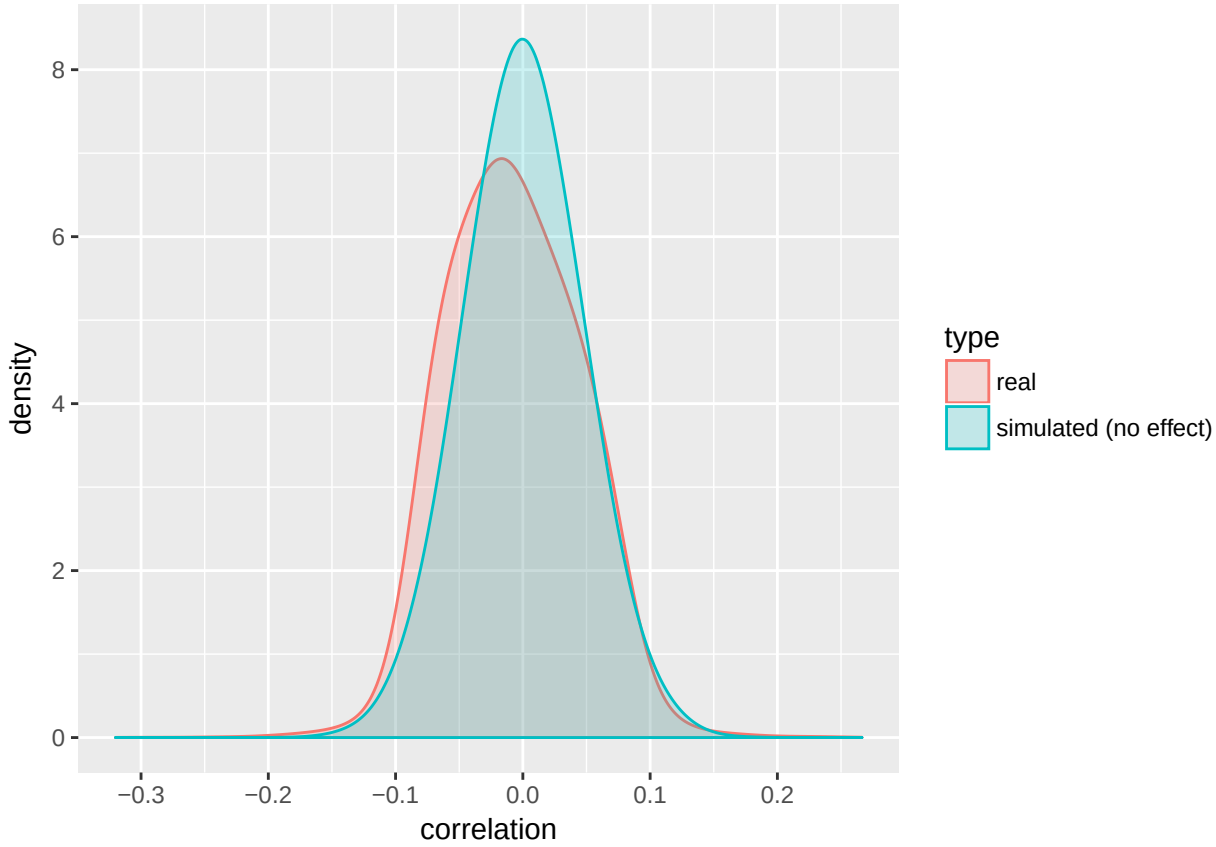
We can check whether individual sequences are correlated with their IDs. To get an idea how those correlations would distribute under a model with no association between ID rank and reads we will also simulate read counts under a Poisson model.

```
cors <- reads[, cor(reads, rank, method="spearman"), by=taxa][, V1]

# simulate reads for 20,000 sequences across 500 samples
sim_reads <- matrix(rpois(500*20000, mean(reads$reads)), ncol=20000)
sim_cors <- apply(sim_reads, 2, function(x) cor(x, 1:500, method="spearman"))

data <- rbind(data.frame(correlation=cors, type="real"),
              data.frame(correlation=sim_cors, type="simulated (no effect)"))
```

```
ggplot(data, aes(x=correlation, fill=type, col=type)) +
  geom_density(adjust=3, alpha=0.2)
```



Maximum correlations are ~0.2 which is not really substantial.

Based on Runs

Runs are identified from the directory structure in our DADA2 pipeline. We can use the same function here.

```
library(stringr)

annotate_files <- function(path) {
    fwd <- list.files(path, pattern = "R1.+\\.fastq.gz", full.names = TRUE,
                      recursive = TRUE)
    ids_fwd <- str_match(fwd, "run_(\\d+)/(.+)_S")
    bwd <- list.files(path, pattern = "R2.+\\.fastq.gz", full.names = TRUE,
                      recursive = TRUE)
    ids_rev <- str_match(bwd, "run_(\\d+)/(.+)_S")
    files_fwd <- data.table(run = ids_fwd[, 2],
                            sample = ids_fwd[, 3], forward = fwd)
    files_rev <- data.table(run = ids_rev[, 2],
                            sample = ids_rev[, 3], reverse = bwd)
    files <- files_fwd[files_rev, on = .(run, sample)]

    return(files)
}
```

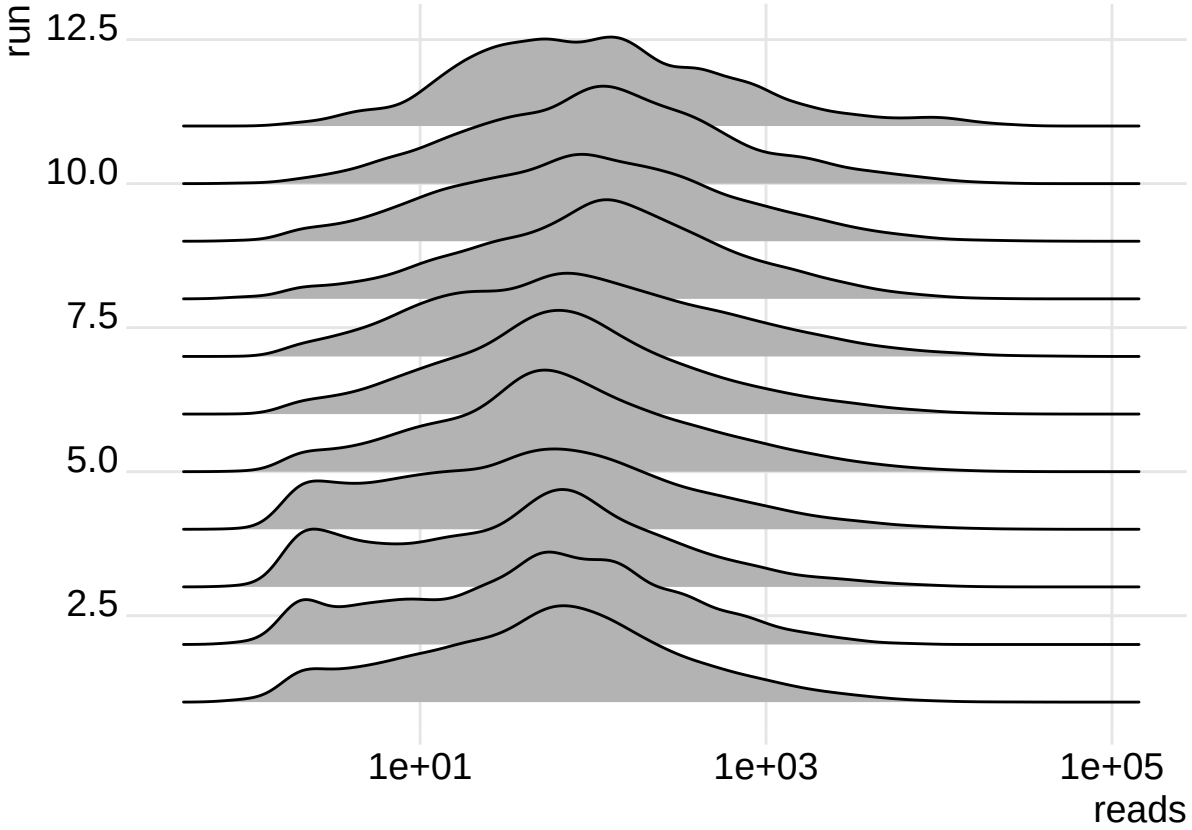
```
files <- annotate_files("../data/filtered")
files[, sample := basename(forward)]
reads <- reads[files, on="sample"]
reads[, run := as.integer(run)]
```

Now we can stratify the samples by run.

```
library(ggridges)

ggplot(reads[reads > 0], aes(x=reads, y=run, group=run)) +
  geom_density_ridges() + scale_x_log10() + theme_ridges()
```

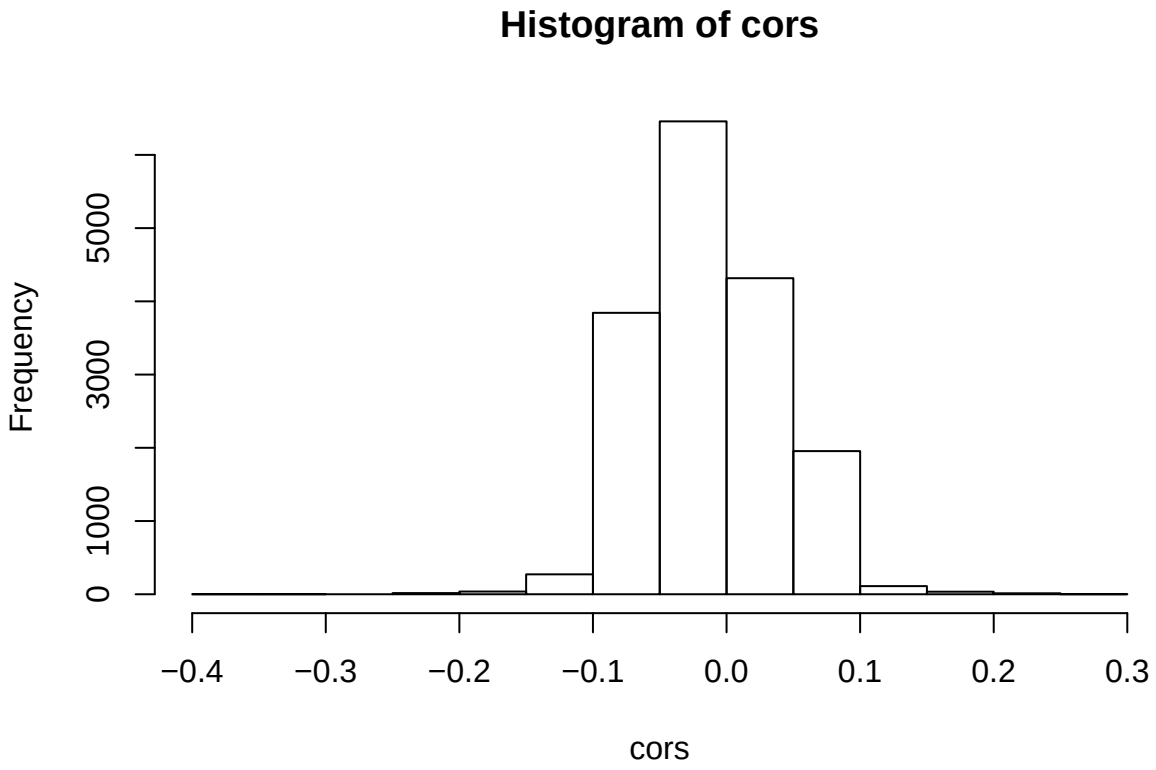
```
## Picking joint bandwidth of 0.123
```



Again nothing really striking. The first 4 runs have a bit higher fractions of low abundance sequences again. Might be noteworthy that those are removed prior to the association analysis anyways.

We can also check the correlation again.

```
cors <- reads[, cor(reads, run, method="spearman"), by=taxa][, V1]
hist(cors)
```



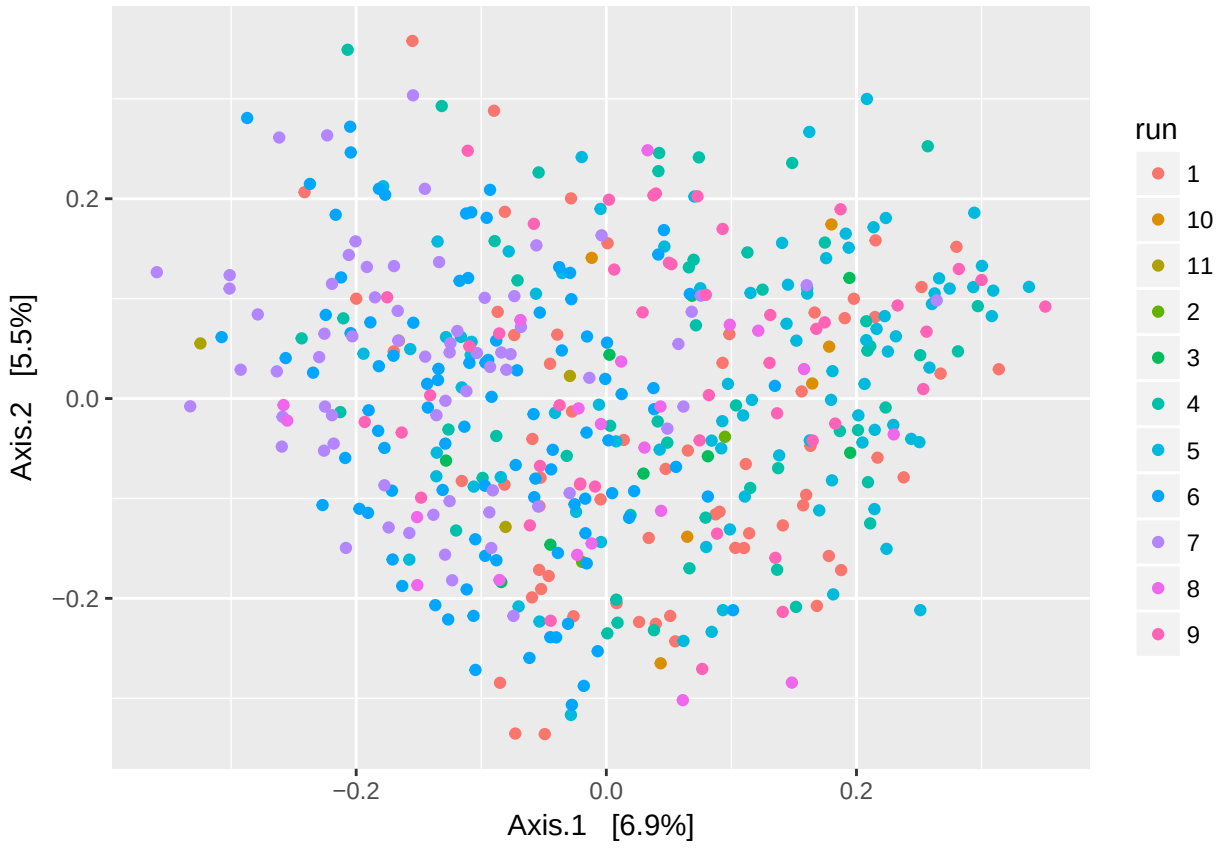
Alternatively we can also check whether the ordination shows dependencies on the runs. For that we will first add our file metadata to the phyloseq object.

```
files <- as.data.frame(files)
rownames(files) <- files$sample
sample_data(ps) <- files
```

With PCoA

First using a linear transformation.

```
ord <- ordinate(ps, "PCoA")
plot_ordination(ps, ord, color="run")
```

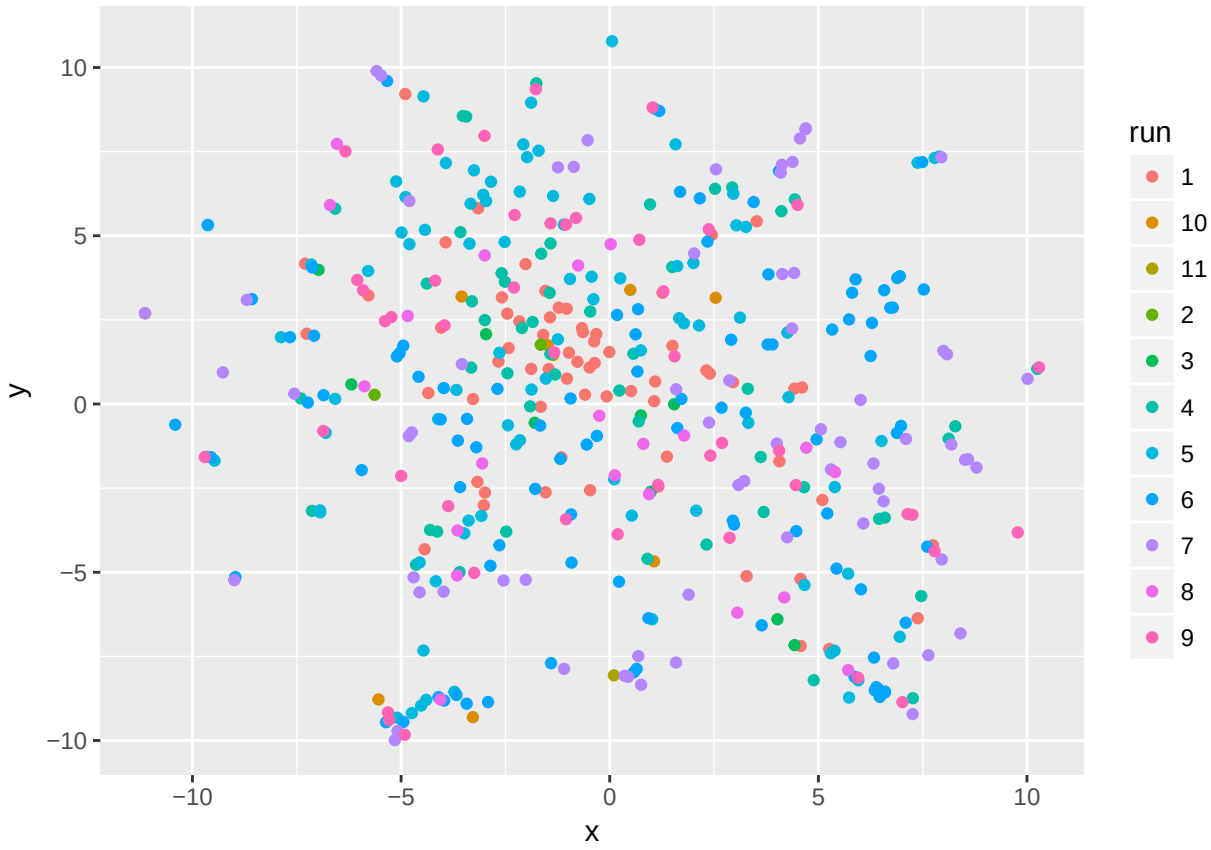


With t-SNE

And also with a non-linear transformation. t-SNE essentially searches for a spatial arrangement of the points in 2 dimensions that has the same distribution of inter-point distances as the original data.

```
library(Rtsne)

tsne <- Rtsne(otu_table(ps), pca=FALSE)
data <- data.frame(x=tsne$Y[, 1], y=tsne$Y[,2], run=files$run)
ggplot(data, aes(x=x, y=y, col=run)) + geom_point()
```



There is no obvious separation of points by run.