Managing Batch Effects in Microbiome Data

Yiwen Wang, Kim-Anh Lê Cao

2019-10-22

Contents

Chapter 1

Examples of microbiome studies with batch effects

This vignette provides all the analyses performed in the paper 'Managing Batch Effects in Microbiome Data' by Yiwen Wang and Kim-Anh Lê Cao.

Packages installation and loading

First, you will need to install then load the following packages from the CRAN and Bioconductor:

```r
# cran.packages <- c('knitr', 'mixOmics', 'xtable', 'ggplot2', 'vegan', 'cluster',
#                    'gridExtra', 'pheatmap', 'ruv', 'lmerTest', 'bapred')
# install.packages(cran.packages)
# bioconductor.packages <- c('sva', 'limma', 'AgiMicroRna',
#                            'variancePartition', 'pvca')
# if (!requireNamespace('BiocManager', quietly = TRUE))
#     install.packages('BiocManager')
# BiocManager::install(bioconductor.packages, version = '3.8')

library(knitr)
library(xtable) # table
library(mixOmics)
library(sva) # ComBat
library(ggplot2) # PCA sample plot with density
library(gridExtra) # PCA sample plot with density
library(limma) # removeBatchEffect (LIMMA)
library(vegan) # RDA
library(AgiMicroRna) # RLE plot
library(cluster) # silhouette coefficient
library(variancePartition) # variance calculation
library(pvca) # PVCA
library(pheatmap) # heatmap
library(ruv) # RUVIII
library(lmerTest) # lmer
library(bapred) # FAbatch
```

1.1 Study description

1.1.1 Sponge *Aplysina aerophoba* study

Sacristán-Soriano *et al.* studied the potential involvement of bacterial communities from the sponge species *A. aerophoba* in the biosynthesis of brominated alkaloids (BAs) (Sacristán-Soriano et al., 2011). They compared the microbial composition and BA concentration in two different tissues (ectosome and choanosome) to investigate the relationship between bacterial composition and BA concentration. The authors concluded that differences in bacterial profiles were not only due to tissue variation (the main effect of interest), but also because the samples were run on two separate denaturing gradient gels during processing. Gel thus acted as a technical batch effect as described in Table 1 below.

Table 1. Overview of exemplar datasets with batch effects. We considered microbiome studies from sponge *Aplysina aerophoba*; organic matter in anaerobic digestion (AD) and mice models with Huntington's disease (HD).

	Sponge data			AD data			HD data		
No. of OTUs	24			231			368		
No. of samples	32			75			13		
Design type	Balanced			Approx. balanced			Unbalanced		
Organism	Sponge samples			Organic matter			Fecal samples		
Batch sources	Gel (sample processing)			Date (sample processing and sequencing)			Cage (housing)		
Batch × treatment design		Ectosome	Choanosome		0-0.5	1-2		HD	WT
	Gel 1	8	8	09/04/2015	4	5	Cage F	0	3
				14/04/2016	4	12	Cage G	3	0
				01/07/2016	8	13	Cage H	3	0
	Gel 2	8	8	14/11/2016	8	9	Cage J	0	4
				21/09/2017	2	10			

1.1.2 Anaerobic digestion study

Anaerobic Digestion (AD) is a microbiological process of organic matter degradation that produces a biogas used in electrical and thermal energy production. However, the AD bioprocess undergoes inhibition during its developmental stage that is not well characterised: Chapleur *et al.* explored microbial indicators that could improve the AD bioprocess's efficacy and prevent its failure (Chapleur et al., 2016). They profiled the microbiota of 75 AD samples in various conditions. Here we consider two different ranges of phenol concentration as treatments. The experiment was conducted at different dates (5), which constitutes a technical source of unwanted variation (Table 1).

1.1.3 Huntington's disease study

In their study, Kong *et al.* reported differences in microbial composition between Huntington's disease (HD) and wild-type (WT) mice (Kong et al., 2018). However, the establishment of microbial communities was also driven by biological batch effects: the cage environment and sex. Here we consider only female mice to illustrate a special case of a batch × treatment unbalanced design. The HD data include 13 faecal mice samples hosted across 4 cages (Table 1).

We load the data and functions that are provided *outside* the packages.

```r
# load the data
load(file = './datasets/microbiome_datasets.RData')

# load the extra functions
```

```
source(file = './Functions.R')
dim(sponge.tss)
```

```
## [1] 32 24
```

```
dim(ad.count)
```

```
## [1]  75 567
```

```
dim(hd.count)
```

```
## [1]  13 368
```

Note: the AD data and HD data loaded are raw counts, while sponge data are total sum scaling (TSS) scaled data calculated on raw counts, with no offset.

1.2 Data processing

Here are the processing steps for the **raw count** microbiome data:

1. Prefilter the count data to remove features with excess zeroes across all samples

2. Add an offset of 1 to the whole data matrix — note that this is not ideal but provides a pracitical way to handle zero counts.

3. Log-ratio transformation with Centered Log Ratio (CLR)

1.2.1 Prefiltering

We use a prefiltering step to remove OTUs for which the sum of counts are below a set threshold (0.01%) compared to the total sum of all counts (Arumugam et al., 2011).

```
# ad data
ad.index.keep <- which(colSums(ad.count)*100/(sum(colSums(ad.count))) > 0.01)
ad.count.keep <- ad.count[, ad.index.keep]
dim(ad.count.keep)
```

```
## [1]  75 231
```

```
# hd data
hd.count.keep <- hd.count
dim(hd.count.keep)
```

```
## [1]  13 368
```

Note: The HD data only include 13 samples, which are a small part of a big dataset that has already been prefiltered. We retained all the OTUs in the data and did not redo the prefiltering again.

1.2.2 Adding offset

We need to add an offset of 1 to all count data to handle zeroes for the CLR transformation. As the sponge data were TSS scaled, a small offset is added in this specific case. According to scale invariance principle (Aitchison, 1986), it returns the same results with CLR transformation on raw counts or TSS data. However, we recommend starting from the raw counts (not TSS) for those analyses.

```r
# sponge data
sponge.tss <- sponge.tss + 0.01

# ad data
ad.count.keep <- ad.count.keep + 1

# hd data
hd.count.keep <- hd.count.keep + 1
```

1.2.3 Centered log-ratio transformation

Microbiome data are compostional and with different library sizes. Using standard statistical methods on such data may lead to spurious results and therefore the data must be further transformed. The CLR is the transformation of choice.

```r
# sponge data
sponge.tss.clr <- logratio.transfo(sponge.tss, logratio = 'CLR')
class(sponge.tss.clr) <- 'matrix'

# ad data
ad.clr <- logratio.transfo(ad.count.keep, logratio = 'CLR')
class(ad.clr) <- 'matrix'

# hd data
hd.clr <- logratio.transfo(hd.count.keep, logratio = 'CLR')
class(hd.clr) <- 'matrix'
```

The final CLR data of sponge study, AD study and HD study contain 32 samples and 24 OTUs, 75 samples and 231 OTUs, 13 samples and 368 OTUs, respectively as described in Table 1.

Chapter 2

Batch effect detection

In this chapter, we apply qualitative methods and diagnostic plots to visually assess the presence of batch effects.

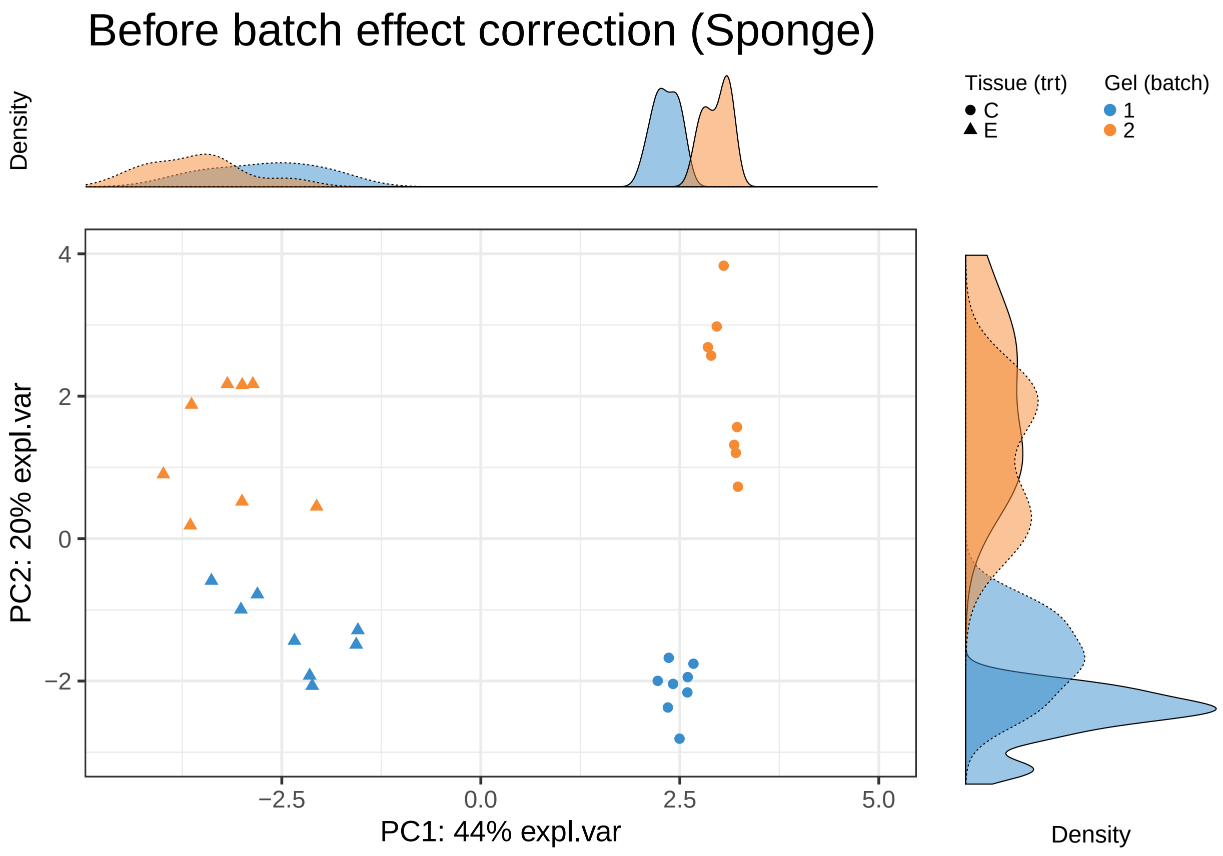
2.1 Principal component analysis (PCA) with density plot per component

PCA is an unsupervised method used to explore the data variance structure by reducing its dimensions to a few principal components (PC) that explain the greatest variation in the data. Density plots are a complementary way to visualise batch effects per PC through examining the distributions of all samples.

First, we run a good old PCA on the data, to assess whether major sources of variation can be explained by batch effects: in cases where batch effects account for a large source of variation in the data, the scatter plot of the top PCs should highlight a separation of the samples due to different batches. Plotting density plots on each component helps to visualise whether it is the case: samples within a batch will show similar distributions, and samples across different batches will show different distributions, if there is a batch effect.

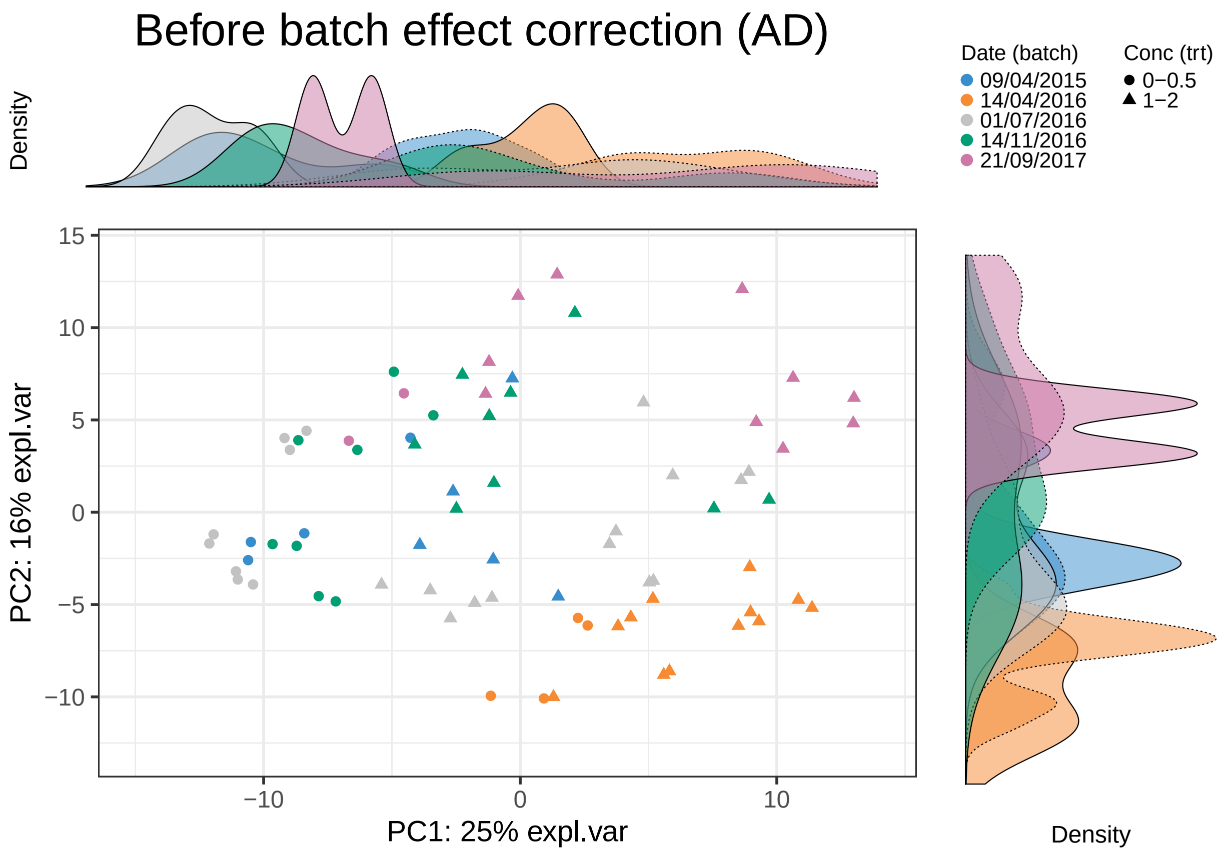
```r
# sponge data
sponge.pca.before <- pca(sponge.tss.clr, ncomp = 3)

# ad data
ad.pca.before <- pca(ad.clr, ncomp = 3)

# hd data
hd.pca.before <- pca(hd.clr, ncomp = 3)
```

```r
# sponge data
Scatter_Density(data = sponge.pca.before$variates$X, batch = sponge.batch,
                trt = sponge.trt, expl.var = sponge.pca.before$explained_variance,
                xlim = c(-4.5,5), ylim = c(-3,4),
                batch.legend.title = 'Gel (batch)',
                trt.legend.title = 'Tissue (trt)',
                title = 'Before batch effect correction (Sponge)')
```



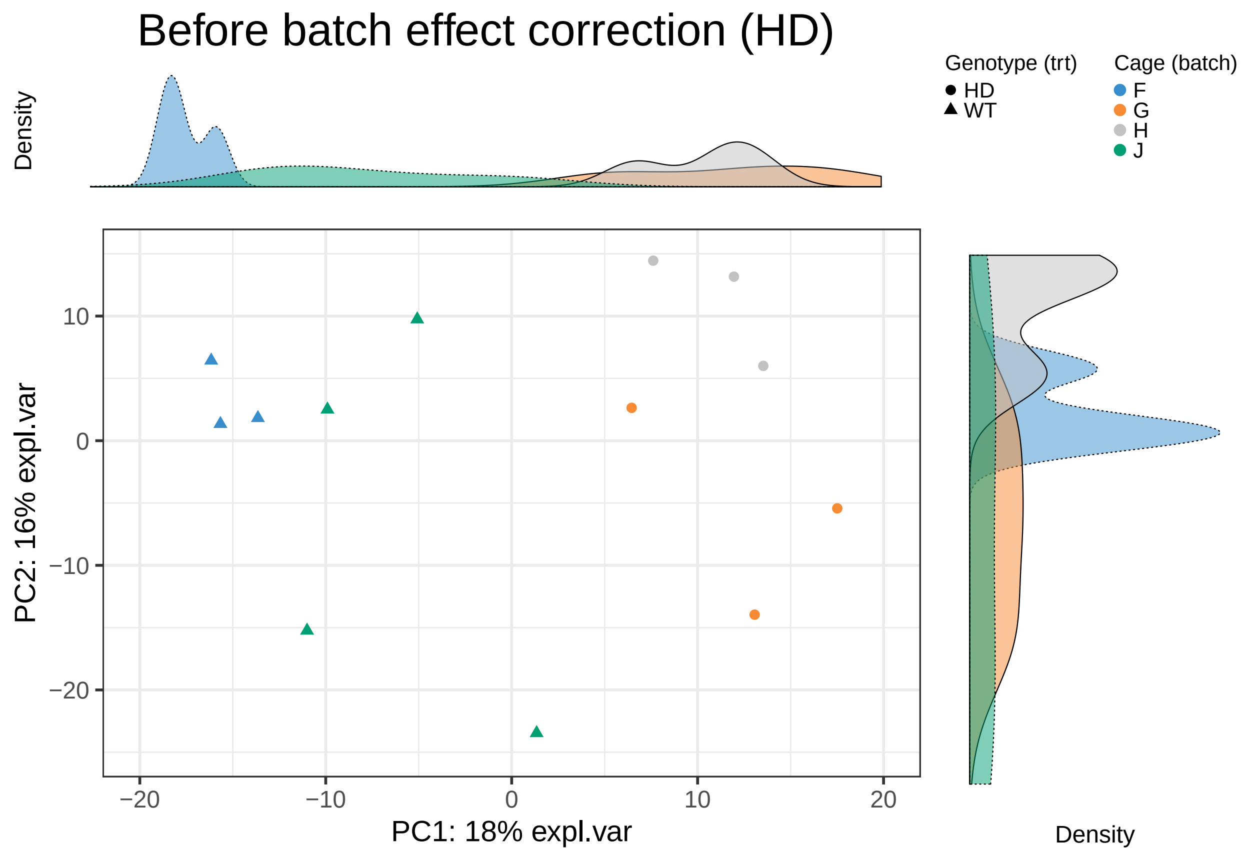
In sponge data, the first PC (explaining the largest source of variation) shows variation between samples from different tissues (the effect of interest), while the second PC (explaining the second largest source of variation) displays sample differences due to different batches, as also highlighted in the density plots per component. Therefore, PCA plots can inform not only of the presence of batch effects, but also which variation is the largest in the data. In this particular dataset, the effect of interest variation is larger than batch variation.

```r
# ad data
Scatter_Density(data = ad.pca.before$variates$X, batch = ad.batch,
                trt = ad.trt, expl.var = ad.pca.before$explained_variance,
                xlim = c(-15,14), ylim = c(-13,14),
                batch.legend.title = 'Date (batch)',
                trt.legend.title = 'Conc (trt)',
                title = 'Before batch effect correction (AD)')
```



In AD data, we observe a separation of samples from batch 14/04/2016. The batch variation is mostly explained by the second PC.

```r
# hd data
Scatter_Density(data = hd.pca.before$variates$X, batch = hd.batch,
                trt = hd.trt, expl.var = hd.pca.before$explained_variance,
                xlim = c(-20,20), ylim = c(-25,15),
                batch.legend.title = 'Cage (batch)',
                trt.legend.title = 'Genotype (trt)',
                title = 'Before batch effect correction (HD)')
```



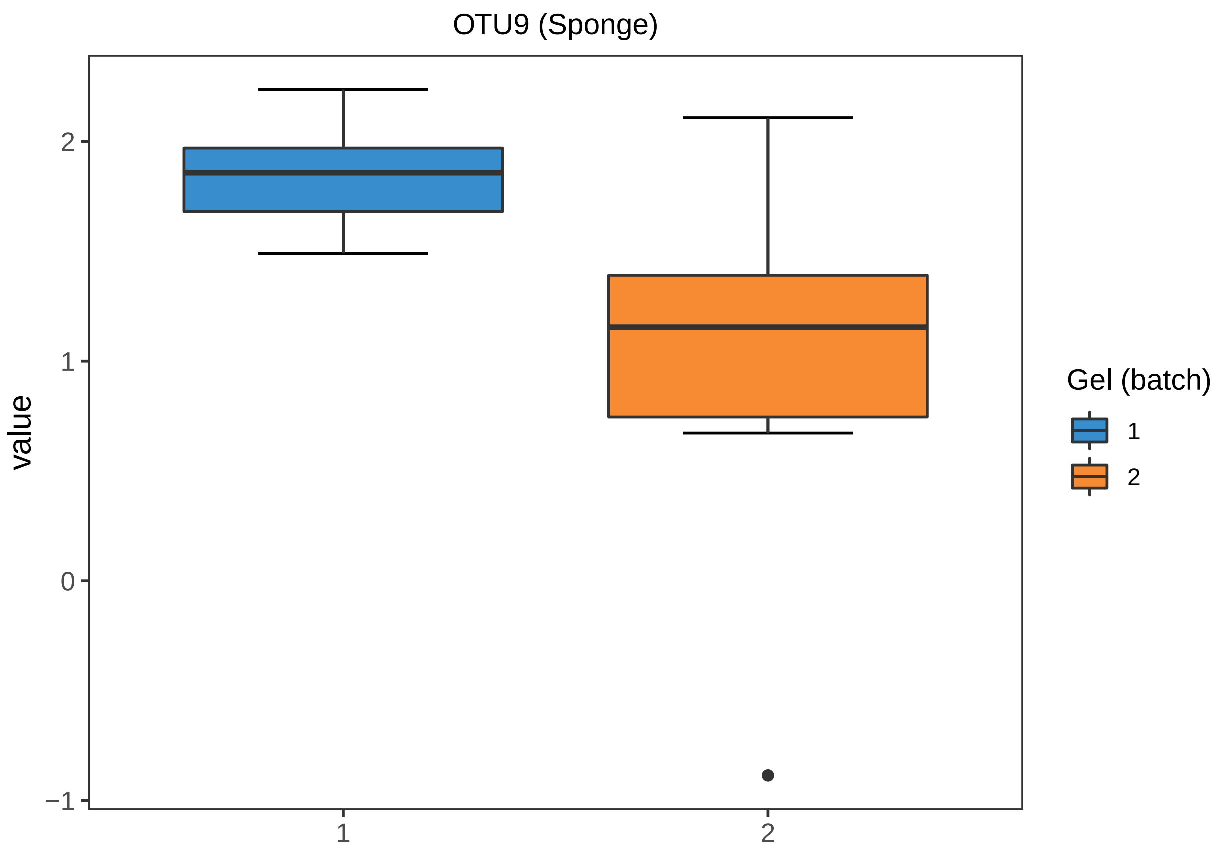
In HD data, batch effect is due to different cages and obvious. The batch variation is both explained by the first and second PC.

2.2 Density plot and box plot

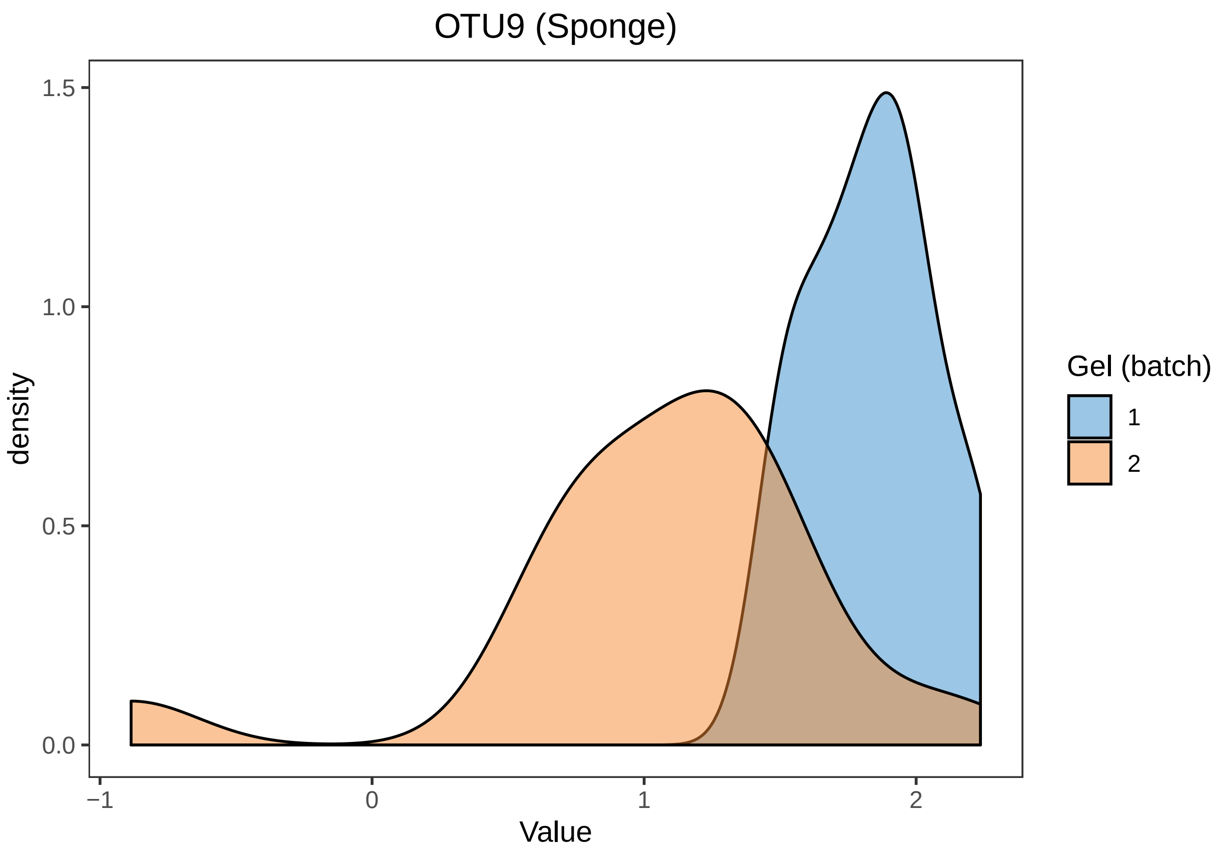
For non systematic batch effects, it is useful to visualise a few OTUs individually. We apply density plots and box plots on OTUs, one at a time from each dataset to visualise batch effects. But only one OTU each dataset is selected as examples.

We randomly select OTU9 in sponge data, and generate density plots and box plots separately across samples within each batch to observe whether batch effects serve as a major source of variation.

```
# sponge data
sponge.before.df <- data.frame(value = sponge.tss.clr[,9], batch = sponge.batch)

box_plot_fun(data = sponge.before.df, x = sponge.before.df$batch,
             y = sponge.before.df$value, title = 'OTU9 (Sponge)',
             batch.legend.title = 'Gel (batch)')
```



```r
ggplot(sponge.before.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU9 (Sponge)', x = 'Value', fill = 'Gel (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())
```



For the proportional abundance of OTU9, the samples within different batches are very distinct, indicating a strong batch effect in sponge data.

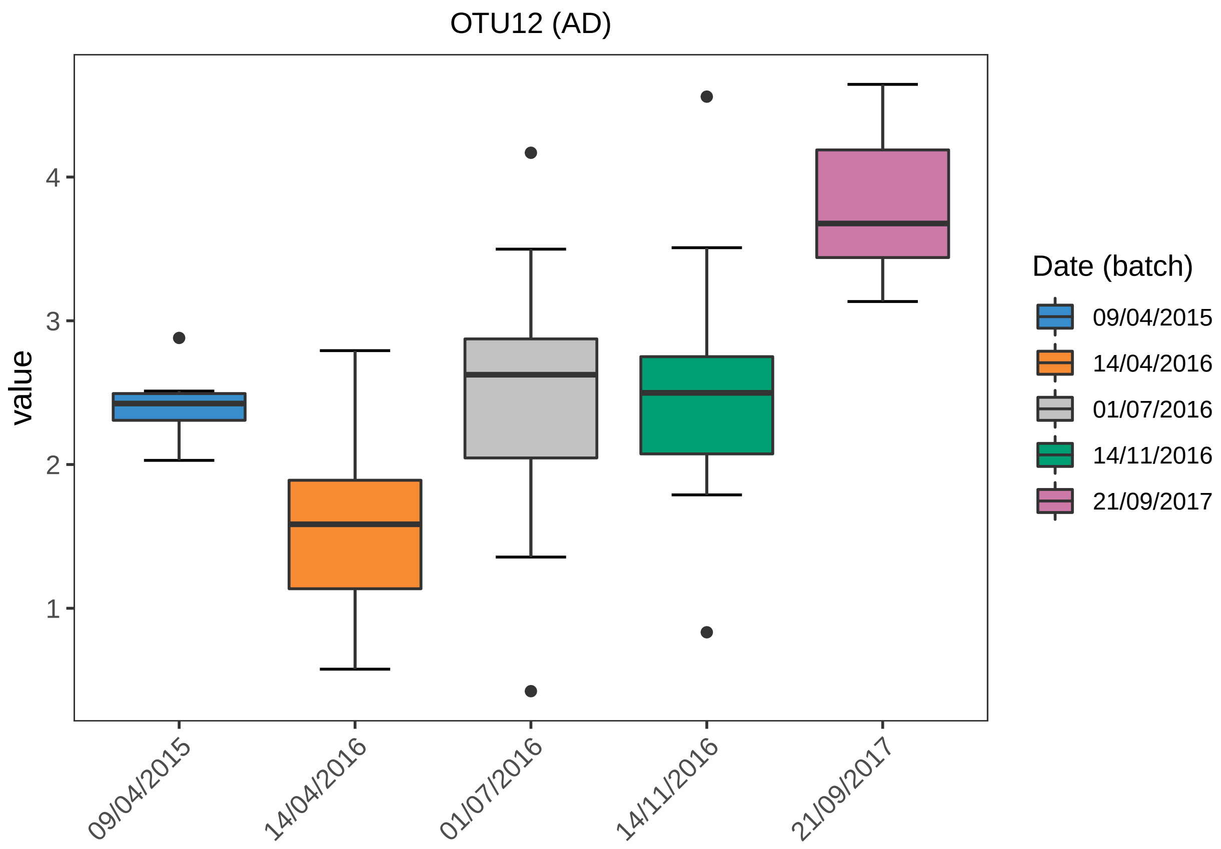
Assuming the data fit the distribution of the linear model (which is the case here after CLR transformation), we can also assess the effect of batch in a linear model on that particular OTU9:

```
sponge.lm <- lm(sponge.tss.clr[,9] ~ sponge.trt + sponge.batch)
summary(sponge.lm)
```

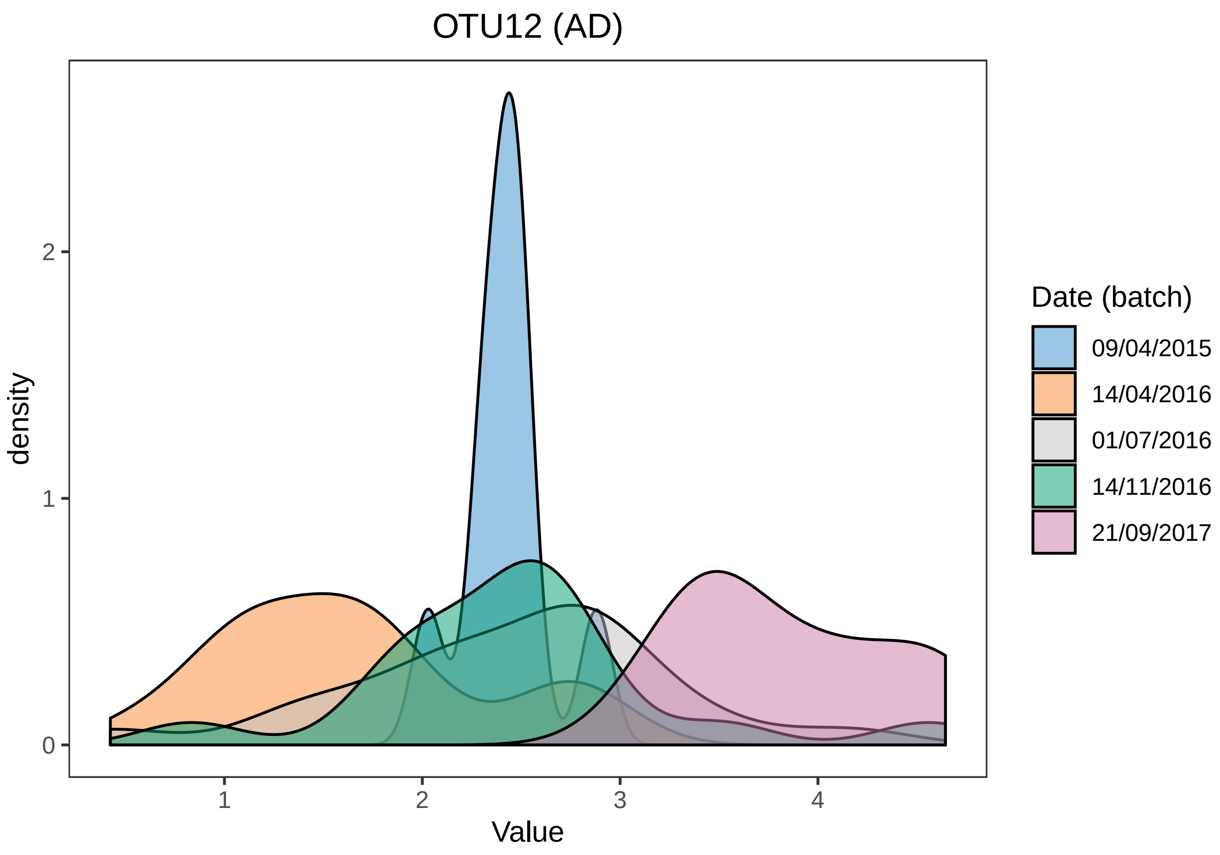
```
##
## Call:
## lm(formula = sponge.tss.clr[, 9] ~ sponge.trt + sponge.batch)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.87967 -0.24705  0.04588  0.24492  1.00757
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.7849     0.1497  11.922 1.06e-12 ***
## sponge.trtE     0.1065     0.1729   0.616    0.543
## sponge.batch2  -0.7910     0.1729  -4.575 8.24e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.489 on 29 degrees of freedom
## Multiple R-squared:  0.4236, Adjusted R-squared:  0.3839
## F-statistic: 10.66 on 2 and 29 DF,  p-value: 0.0003391
```

The batch (gel) effect is statistically significant ($P < 0.001$), as indicated in the **sponge.batch2** row.

```r
# ad data
ad.before.df <- data.frame(value = ad.clr[,1], batch = ad.batch)

box_plot_fun(data = ad.before.df,x = ad.before.df$batch,
             y = ad.before.df$value, title = 'OTU12 (AD)',
             batch.legend.title = 'Date (batch)',
             x.angle = 45, x.hjust = 1)
```



```r
ggplot(ad.before.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 (AD)',x = 'Value',fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())
```



Batch effects in AD data are also easily visualised.

```
ad.lm <- lm(ad.clr[,1] ~ ad.trt + ad.batch)
anova(ad.lm)
```

```
## Analysis of Variance Table
##
## Response: ad.clr[, 1]
##           Df Sum Sq Mean Sq F value    Pr(>F)
## ad.trt     1  1.460  1.4605  3.1001   0.08272 .
## ad.batch   4 32.889  8.2222 17.4532 6.168e-10 ***
## Residuals 69 32.506  0.4711
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

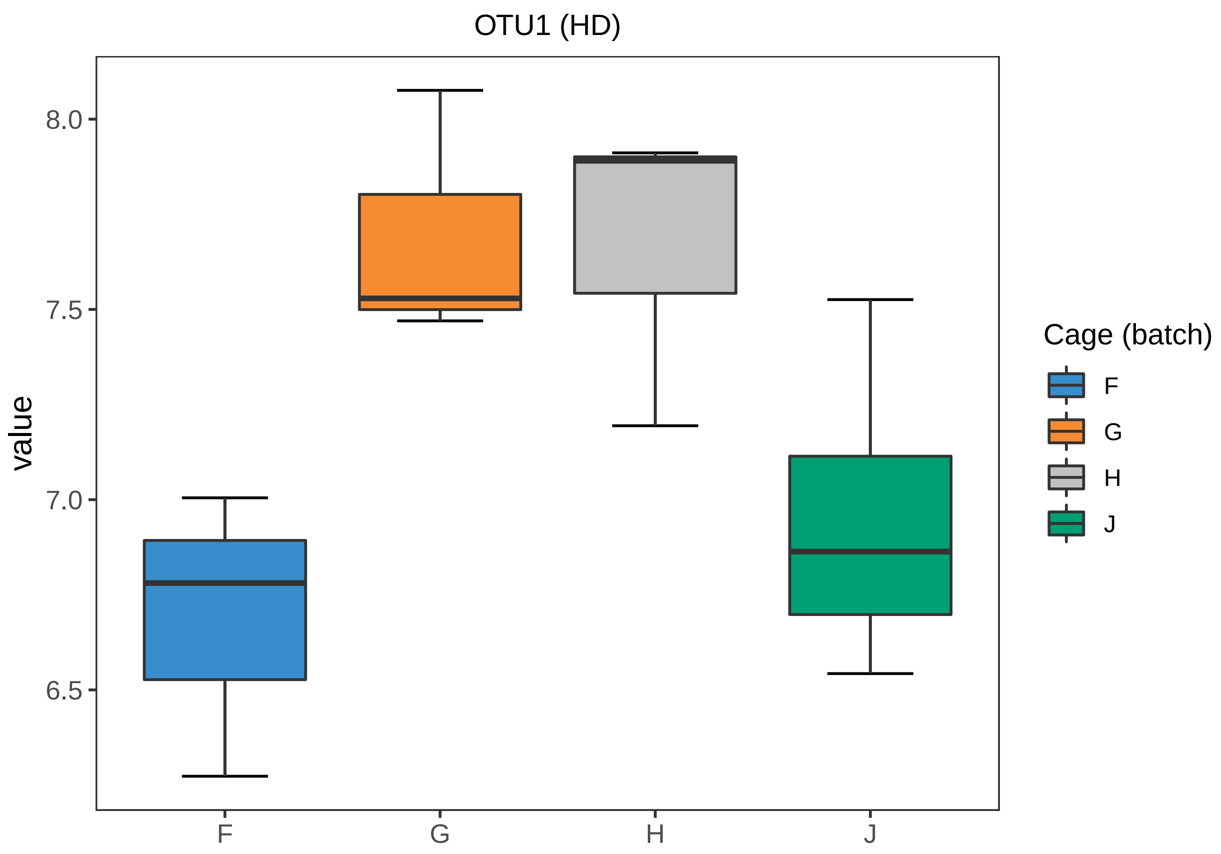
In AD data, the difference between batches (dates) is statistically significant (P < 0.001, as tested with ANOVA). We can also obtain P values between each two batch categories.

```
summary(ad.lm)
```

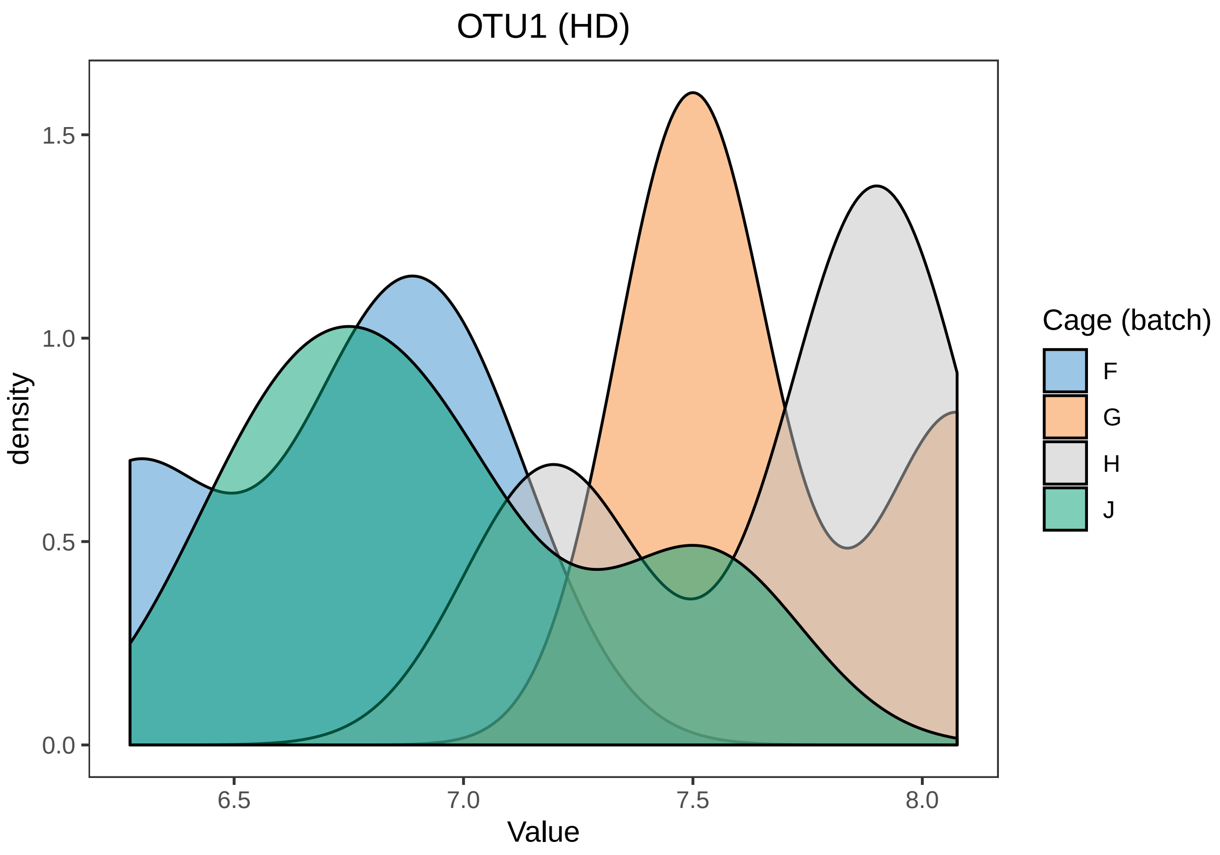
```
##
## Call:
## lm(formula = ad.clr[, 1] ~ ad.trt + ad.batch)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09885 -0.39613 -0.00381  0.36645  1.98185
##
## Coefficients:
```

```
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.311213   0.247768   9.328 7.57e-14 ***
## ad.trt1-2            0.203619   0.171183   1.189  0.23833
## ad.batch14/04/2016 -0.828100   0.287918  -2.876  0.00535 **
## ad.batch01/07/2016  0.007239   0.273672   0.026  0.97897
## ad.batch14/11/2016  0.062689   0.282978   0.222  0.82533
## ad.batch21/09/2017  1.361132   0.306373   4.443 3.30e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6864 on 69 degrees of freedom
## Multiple R-squared:  0.5138, Adjusted R-squared:  0.4786
## F-statistic: 14.58 on 5 and 69 DF,  p-value: 9.665e-10
```

```
# hd data
hd.before.df <- data.frame(value = hd.clr[,1], batch = hd.batch)

box_plot_fun(data = hd.before.df, x = hd.before.df$batch,
             y = hd.before.df$value,title = 'OTU1 (HD)',
             batch.legend.title = 'Cage (batch)')
```



```
ggplot(hd.before.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU1 (HD)',x = 'Value',fill = 'Cage (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())
```



In HD data, we easily detect the differences between samples within different batches.

```
hd.lm <- lm(hd.clr[,1] ~ hd.batch)
anova(hd.lm)
```

```
## Analysis of Variance Table
##
## Response: hd.clr[, 1]
##           Df Sum Sq Mean Sq F value  Pr(>F)
## hd.batch   3 2.4108 0.80359  5.2569 0.02276 *
## Residuals  9 1.3758 0.15286
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the batch x treatment design of HD data is nested and unbalanced, the linear model with both treatment (genotype) and batch (cage) is unable to fit. We therefore fit a linear model with batch effect only. The difference between cages is statistically significant ($P < 0.05$). But the difference may also be influenced by treatment and we are unable to exclude treatment influence.

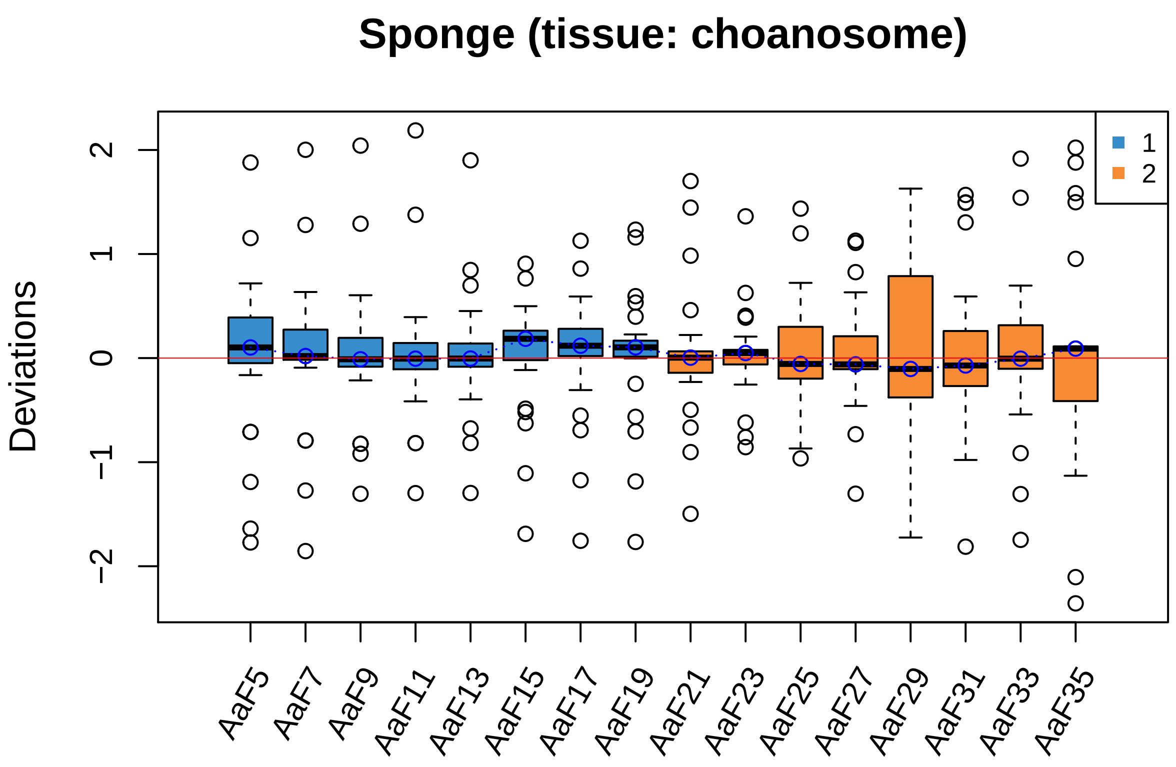
2.3 RLE plots

RLE plots can be plotted using ‘RleMicroRna’ in R package ‘AgiMicroRna’. Here, we made some changes on the function ‘RleMicroRna’, called ‘RleMicroRna2’ and available on our extra functions ‘Functions.R’.

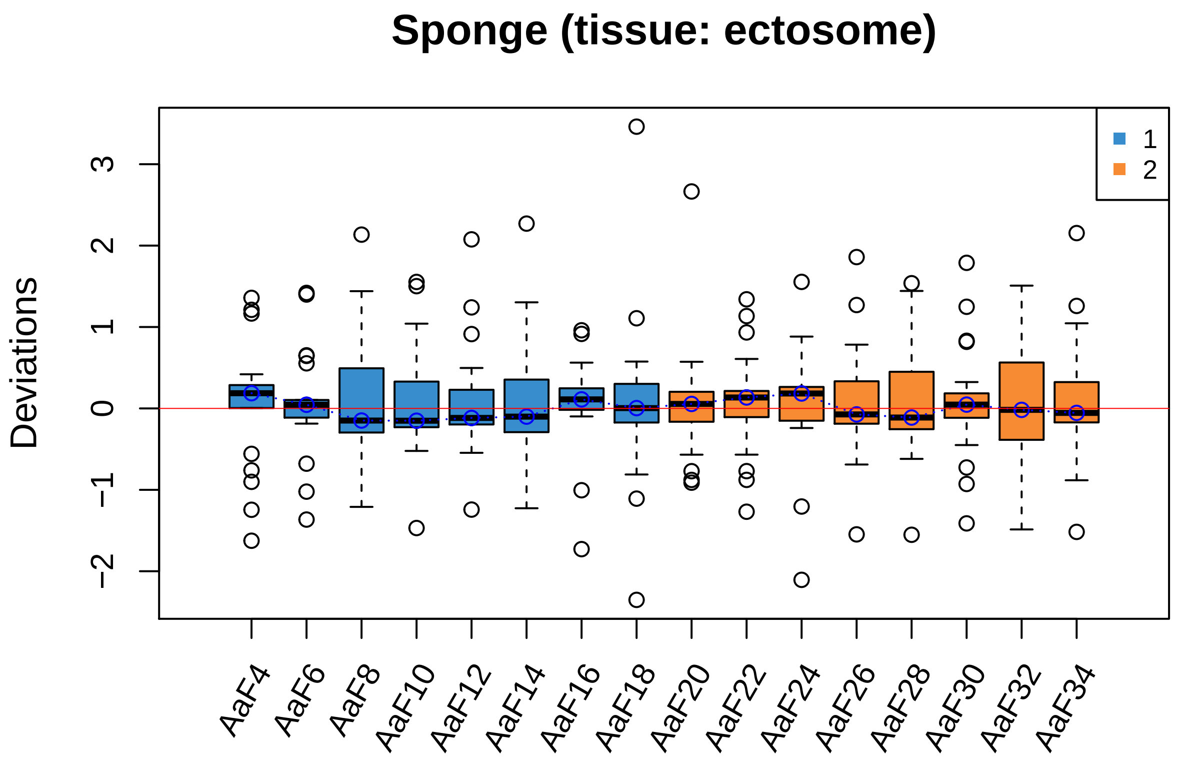
RLE plots are based on the assumption that the majority of microbial variables are unaffected by the effect of interest, and therefore any sample heterogeneity observed - i.e. different distributions and their variances, and medians different from zero, should indicate the presence of batch effects. In our case studies, the treatment information is known, so we generate multiple RLE plots per treatment group, as suggested by (Lin et al., 2018).

In sponge data, we group the samples according to the tissue (choanosome / ectosome) and generate two RLE plots:

```r
# sponge data
sponge.batch_c <- sponge.batch[sponge.trt == 'C']
sponge.batch_e <- sponge.batch[sponge.trt == 'E']

sponge.before_c <- sponge.tss.clr[sponge.trt == 'C', ]
sponge.before_e <- sponge.tss.clr[sponge.trt == 'E', ]


RleMicroRna2(object = t(sponge.before_c), batch = sponge.batch_c,
             maintitle = 'Sponge (tissue: choanosome)')
```

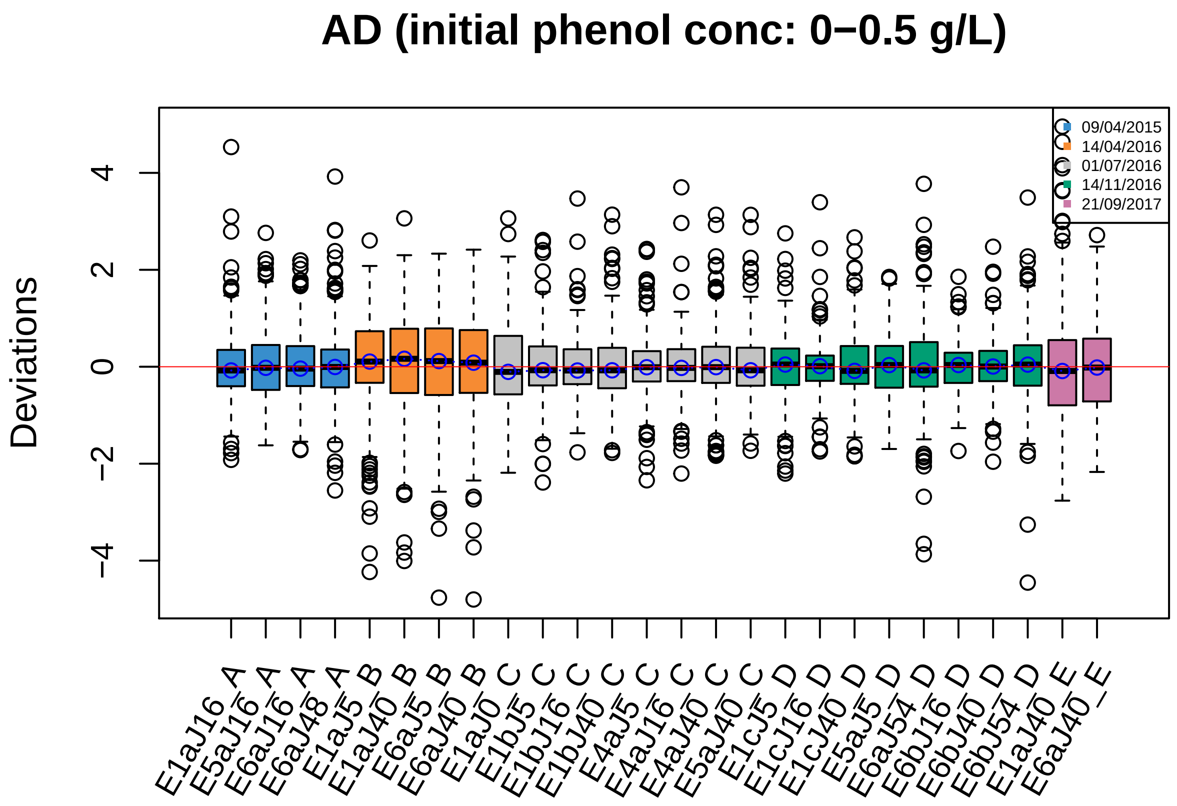


```r
RleMicroRna2(object = t(sponge.before_e), batch = sponge.batch_e,
             maintitle = 'Sponge (tissue: ectosome)')
```

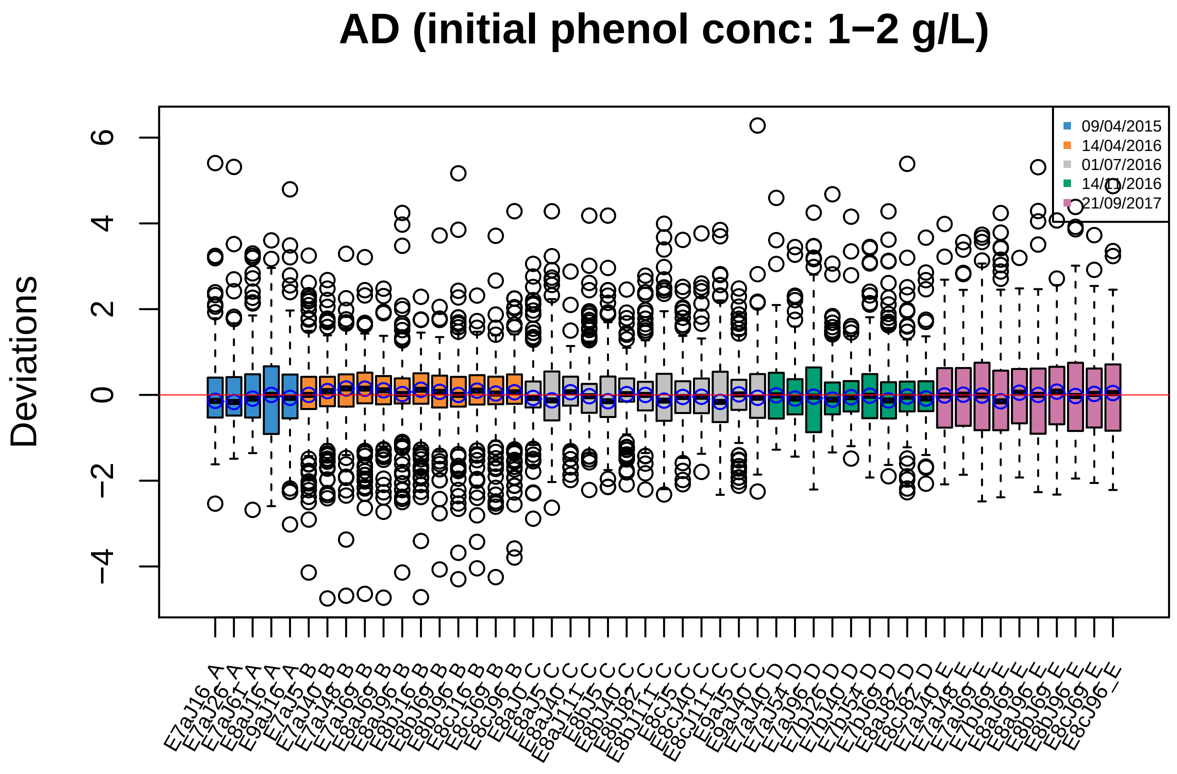


For both RLE plots with samples from different tissues, the batch effect is not obvious as all medians of samples are close to zero, but Gel2 has a greater interquartile range (IQR) than the other samples.

```
# ad data
ad.batch_05 <- ad.batch[ad.trt == '0-0.5']
ad.batch_2 <- ad.batch[ad.trt == '1-2']

ad.before_05 <- ad.clr[ad.trt == '0-0.5', ]
ad.before_2 <- ad.clr[ad.trt == '1-2', ]

RleMicroRna2(object = t(ad.before_05), batch = ad.batch_05,
             maintitle = 'AD (initial phenol conc: 0-0.5 g/L)',
             legend.cex = 0.5)
```

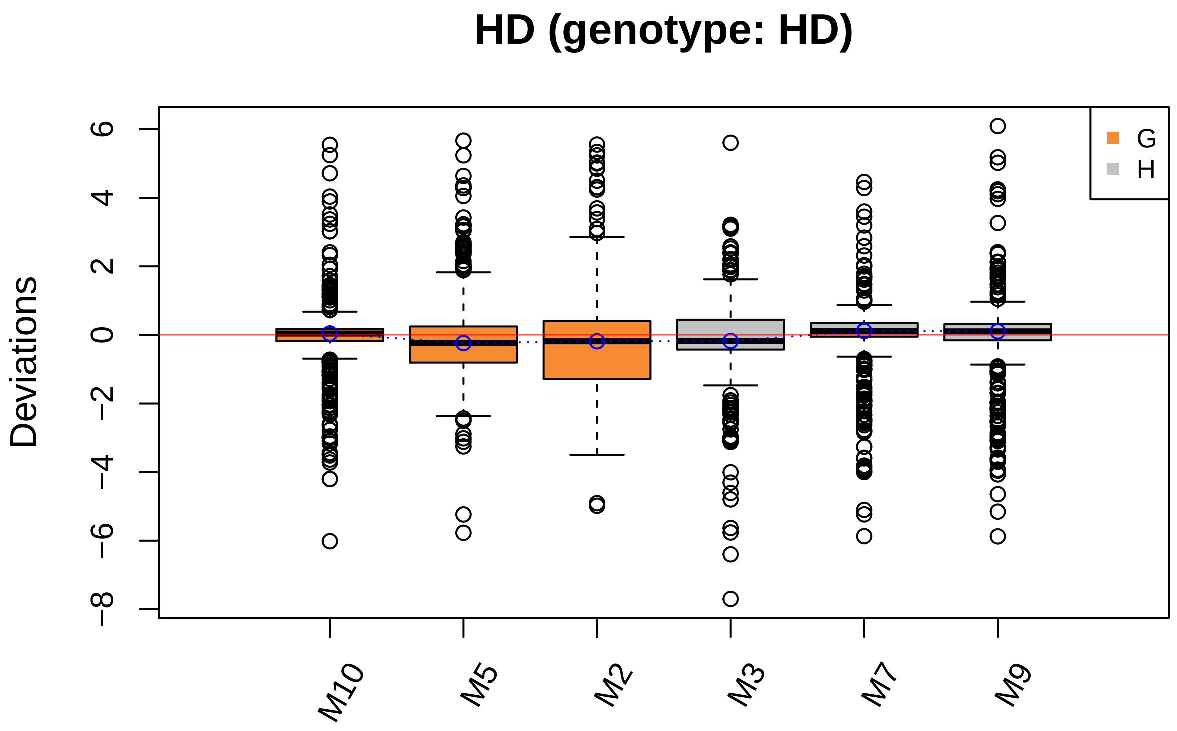


```r
RleMicroRna2(object = t(ad.before_2), batch = ad.batch_2,
             maintitle = 'AD (initial phenol conc: 1-2 g/L)',
             cex.xaxis = 0.7, legend.cex = 0.5)
```

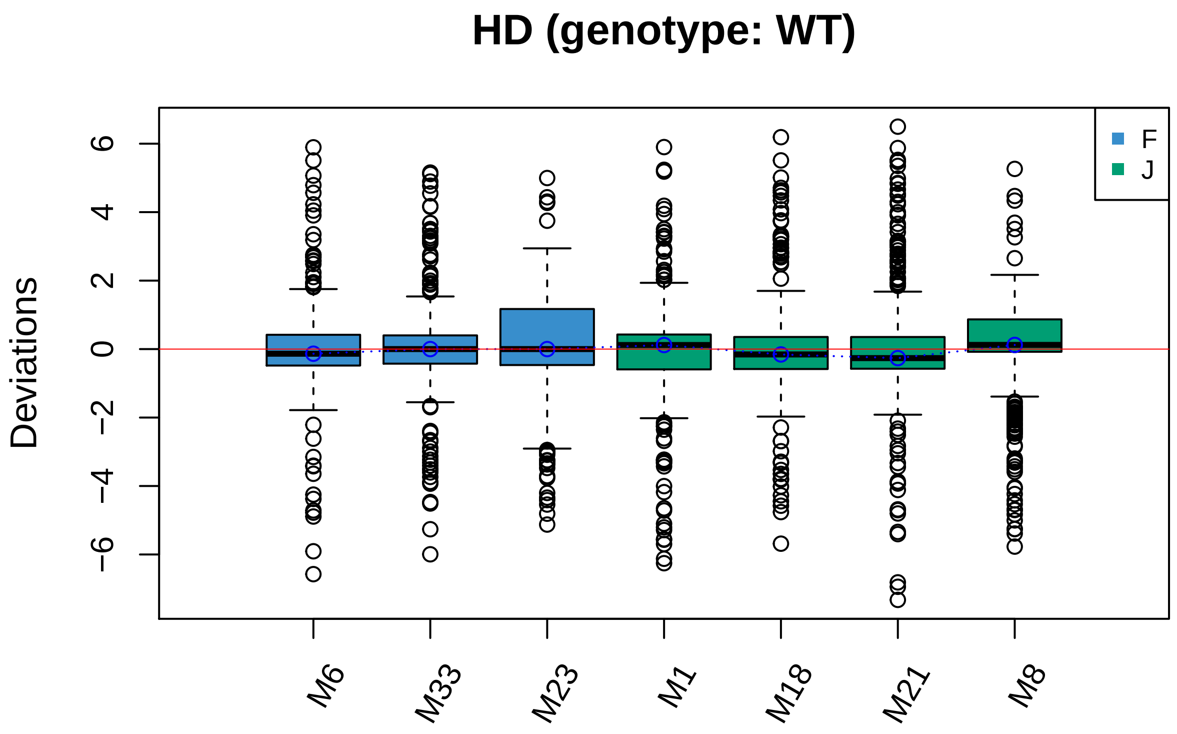


In RLE plots for the AD data, the batch effect is also not obvious as all medians of samples are close to zero, but the samples dated 14/04/2016 may be affected by batch as they have a greater IQR than the other samples.

```
# hd data
hd.batch_h <- hd.batch[hd.trt == 'HD']
hd.batch_w <- hd.batch[hd.trt == 'WT']

hd.before_h <- hd.clr[hd.trt == 'HD', ]
hd.before_w <- hd.clr[hd.trt == 'WT', ]

RleMicroRna2(object = t(hd.before_h), batch = hd.batch_h,
             maintitle = 'HD (genotype: HD)')
```



```
RleMicroRna2(object = t(hd.before_w), batch = hd.batch_w,
             maintitle = 'HD (genotype: WT)')
```

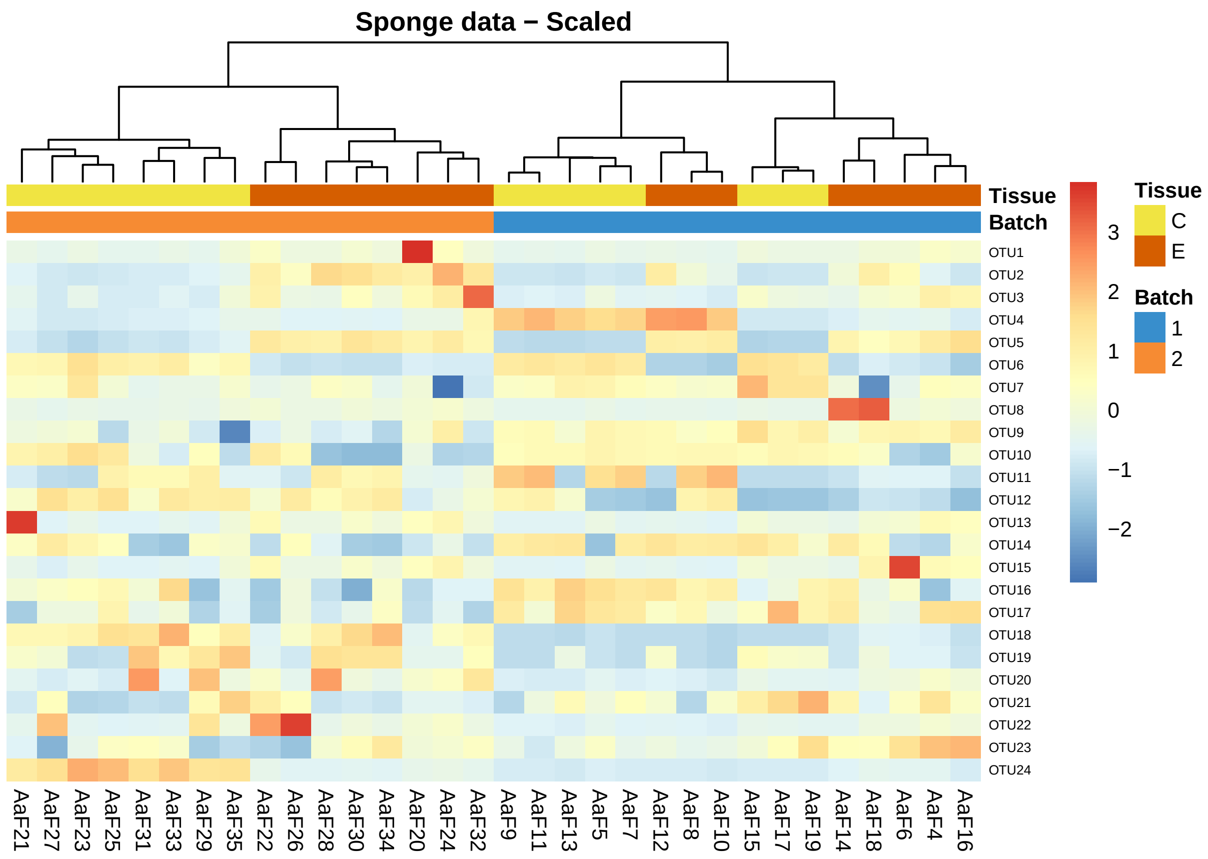


The batch effect in HD data is not easily detected, but Cage G has a greater IQR than the other samples, which may indicate a batch effect.

2.4 Heatmap

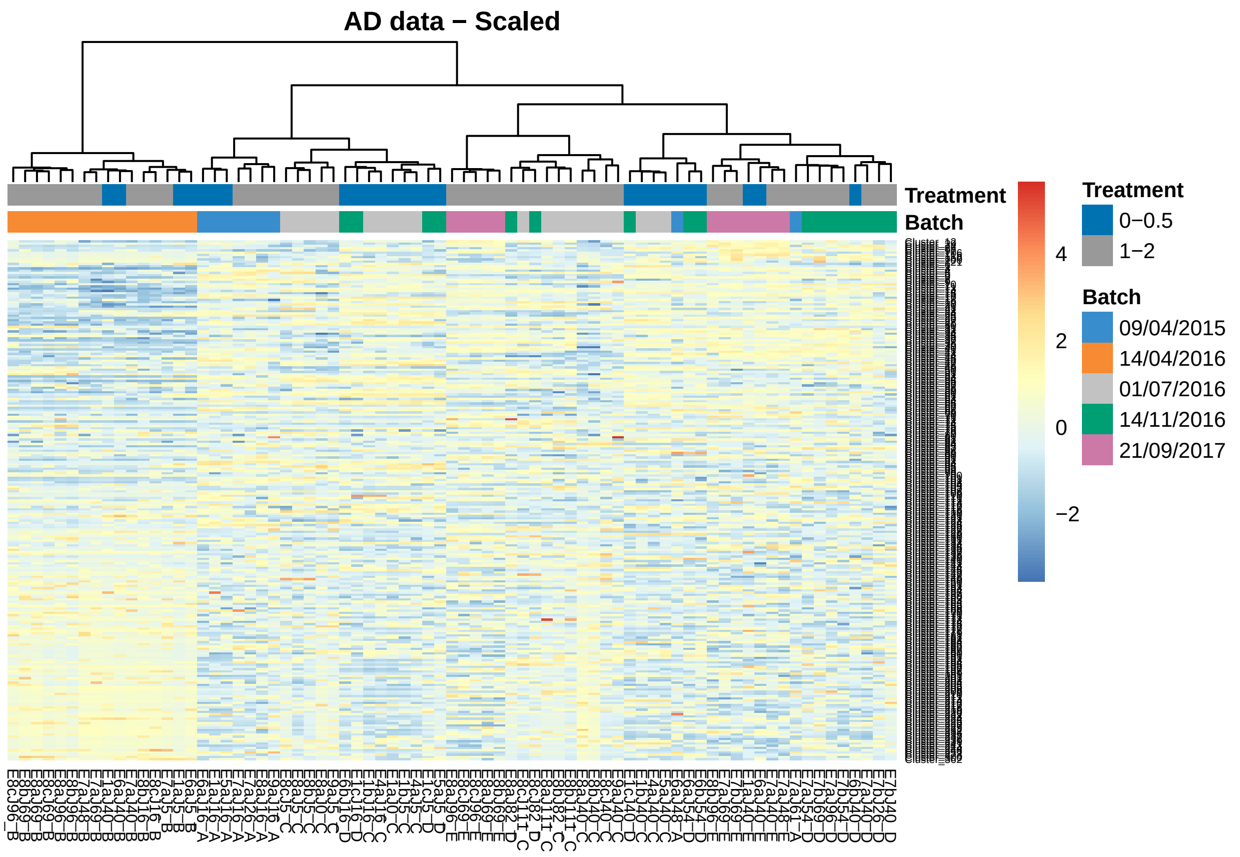
Clustering analysis can be used to detect batch effects. Ideally samples with the same treatment will be clustered together, data clustered by batches instead of treatments indicate a batch effect. Heatmaps and dendrograms are two common approaches to visualise the clusters.

```
# Sponge data
# scale on OTUs
sponge.tss.clr.scale <- scale(sponge.tss.clr, center = T, scale = T)
# scale on samples
sponge.tss.clr.scale <- scale(t(sponge.tss.clr.scale), center = T, scale = T)

sponge.anno_col <- data.frame(Batch = sponge.batch, Tissue = sponge.trt)
sponge.anno_metabo_colors <- list(Batch = c('1' = '#388ECC', '2' = '#F68B33'),
                                  Tissue = c(C = '#F0E442', E = '#D55E00'))


pheatmap(sponge.tss.clr.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 5, fontsize_col = 8,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = sponge.anno_col,
         annotation_colors = sponge.anno_metabo_colors,
         border_color = 'NA',
         main = 'Sponge data - Scaled')
```



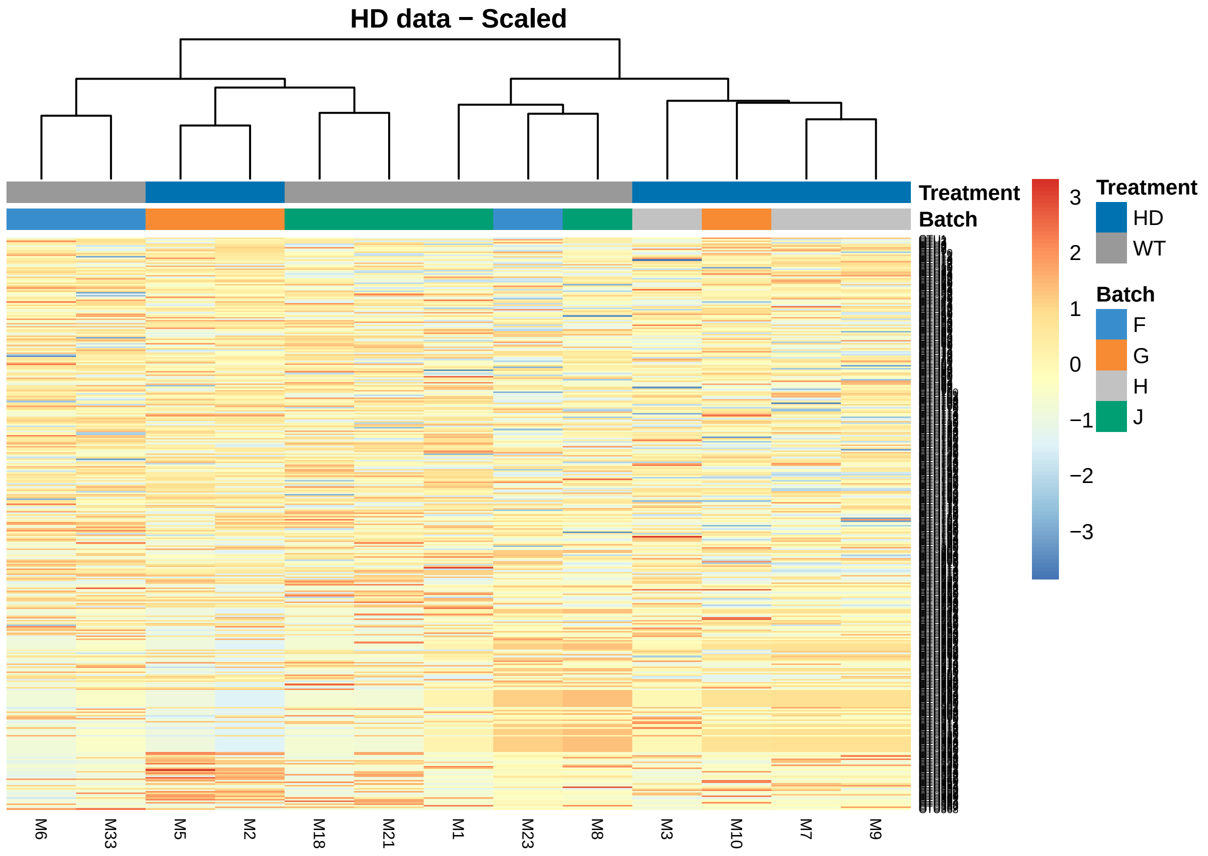
In sponge data, samples are preferentially clustered by batch instead of tissue type, indicating a batch effect.

```
# AD data
ad.clr.scale <- scale(ad.clr,center = T, scale = T)
ad.clr.scale <- scale(t(ad.clr.scale), center = T, scale = T)

ad.anno_col <- data.frame(Batch = ad.batch, Treatment = ad.trt)
ad.anno_metabo_colors <- list(Batch = c('09/04/2015' = '#388ECC',
                                        '14/04/2016' = '#F68B33',
                                        '01/07/2016' = '#C2C2C2',
                                        '14/11/2016' = '#009E73',
                                        '21/09/2017' = '#CC79A7'),
                              Treatment = c('0-0.5' = '#0072B2', '1-2' = '#999999'))


pheatmap(ad.clr.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - Scaled')
```



In AD data, samples within batch 14/04/2016 are clustered and distinct from other samples, also indicating a batch effect.

```
# HD data
hd.clr.scale <- scale(hd.clr,center = T, scale = T)
hd.clr.scale <- scale(t(hd.clr.scale), center = T, scale = T)

hd.anno_col <- data.frame(Batch = hd.batch, Treatment = hd.trt)
hd.anno_metabo_colors <- list(Batch = c('F' = '#388ECC', 'G' = '#F68B33',
                                         'H' = '#C2C2C2', 'J' = '#009E73'),
                              Treatment = c('HD' = '#0072B2', 'WT' = '#999999'))


pheatmap(hd.clr.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = hd.anno_col,
         annotation_colors = hd.anno_metabo_colors,
         border_color = 'NA',
         main = 'HD data - Scaled')
```



The batch effect in HD data is not obvious in this heatmap, as we observe not clustering according to batch.

Chapter 3

Batch effect adjustment

3.1 Accounting for batch effects

Methods that account for batch effects estimate unknown batch effects through matrix decomposition and / or assign a known or estimated batch as a covariate with linear models.

3.1.1 Linear model and linear mixed model

LM and LMM are suitable for known batch effects, and can consider batch x treatment interaction and deal with unbalanced batch x treatment design. But they are univariate and rely on a Gaussian likelihood assumption, which may not apply to zero-inflated microbiome data despite CLR transformation.

We fit a linear model with effect of interest and batch effect for each OTU in both sponge and AD data. The P-value for the regression coefficient associated with the *effect of interest* in a linear model is then extracted and multiple testing corrected with “FDR”.

```
# Sponge data
sponge.trt_p <- apply(sponge.tss.clr, 2, FUN = function(x){
  res.lm <- lm(x ~ sponge.trt + sponge.batch)
  summary.res <- summary(res.lm)
  p <- summary.res$coefficients[2,4]
})

sponge.trt_adjp <- p.adjust(sponge.trt_p, method = 'fdr')

# AD data
ad.trt_p <- apply(ad.clr, 2, FUN = function(x){
  res.lm <- lm(x ~ ad.trt + ad.batch)
  summary.res <- summary(res.lm)
  p <- summary.res$coefficients[2,4]
})

ad.trt_adjp <- p.adjust(ad.trt_p, method = 'fdr')
```

As the batch x treatment design of HD data is unbalaced, we fit a linear mixed model considering batch (cage) as random effects.

```
# HD data
hd.trt_p <- apply(hd.clr, 2, FUN = function(x){
  res.lmm <- lmer(x ~ hd.trt + (1|hd.batch))
  summary.res <- summary(res.lmm)
  p <- summary.res$coefficients[2,5]
})

hd.trt_adjp <- p.adjust(hd.trt_p, method = 'fdr')
```

3.1.2 SVA

SVA can account for unknown batch effects, but is a univariate approach that relies on a Gaussian likelihood assumption and implicitly introduces a correlation between treatment and batch.

The *sva* function performs two different steps. First it identifies the number of latent factors that need to be estimated. The number of factors can be estimated using the function *num.sv*.

We first fit a full model with effect of interest and a null model with no effect, and use the full model in the function *num.sv* to estimate the number of latent factors.

```
# sponge data
sponge.mod <- model.matrix( ~ sponge.trt) # full model
sponge.mod0 <- model.matrix( ~ 1,data = sponge.trt) # null model
sponge.sva.n <- num.sv(dat = t(sponge.tss.clr), mod = sponge.mod)
```

Next we apply the *sva* function to estimate the surrogate variables with both full and null models:

```
sponge.sva <- sva(dat = t(sponge.tss.clr), mod = sponge.mod,
                  mod0 = sponge.mod0, n.sv = sponge.sva.n)
```

```
## Number of significant surrogate variables is:  3
## Iteration (out of 5 ):1  2  3  4  5
```

We then include the estimated surrogate variables in both the null and full models. The reason is that we want to adjust for the surrogate variables, so we treat them as adjustment variables that must be included in both models. The *f.pvalue* function is then used to calculate parametric F-test P-values and Q-values (adjusted P-values) for each OTU of sponge data.

```
sponge.mod.bat <- cbind(sponge.mod, sponge.sva$sv)
sponge.mod0.bat <- cbind(sponge.mod0, sponge.sva$sv)

sponge.sva.trt_p <- f.pvalue(t(sponge.tss.clr), sponge.mod.bat, sponge.mod0.bat)
sponge.sva.trt_adjp <- p.adjust(sponge.sva.trt_p, method='fdr')
```

Now these P-values and Q-values are accounting for surrogate variables (estimated batch effects).

We also apply SVA on both AD and HD data.

```
# ad data
ad.mod <- model.matrix( ~ ad.trt)
ad.mod0 <- model.matrix( ~ 1, data = ad.trt)
ad.sva.n <- num.sv(dat = t(ad.clr), mod = ad.mod)
ad.sva <- sva(t(ad.clr), ad.mod, ad.mod0, n.sv = ad.sva.n)
```

```
## Number of significant surrogate variables is:  6
## Iteration (out of 5 ):1  2  3  4  5
```

```
ad.mod.bat <- cbind(ad.mod, ad.sva$sv)
ad.mod0.bat <- cbind(ad.mod0, ad.sva$sv)
ad.sva.trt_p <- f.pvalue(t(ad.clr), ad.mod.bat, ad.mod0.bat)
ad.sva.trt_adjp <- p.adjust(ad.sva.trt_p, method = 'fdr')
```

```
# hd data
hd.mod <- model.matrix( ~ hd.trt)
hd.mod0 <- model.matrix( ~ 1,data = hd.trt)
hd.sva.n <- num.sv(dat = t(hd.clr), mod = hd.mod)
hd.sva <- sva(t(hd.clr), hd.mod, hd.mod0, n.sv = hd.sva.n)
```

```
## Number of significant surrogate variables is:  3
## Iteration (out of 5 ):1  2  3  4  5
```

```
hd.mod.bat <- cbind(hd.mod, hd.sva$sv)
hd.mod0.bat <- cbind(hd.mod0, hd.sva$sv)
hd.sva.trt_p <- f.pvalue(t(hd.clr), hd.mod.bat, hd.mod0.bat)
hd.sva.trt_adjp <- p.adjust(hd.sva.trt_p, method = 'fdr')
```

3.1.3 RUV2

RUV2 estimates and accounts for unknown batch effects, but the method requires negative control variables that are affected by batch effects but not treatment effects.

Practically, negative control variables are chosen prior during the experimental design so that they are not affected by treatments (and assume they are affected by batch effects). RUV2 can only account for the difference captured by these controls.

Since our three datasets do not have negative control variables, we use a linear model (or linear mixed model) to identify OTUs that are not affected by treatment, those will be considered as *pseudo negative control variables* to fit the assumptions of RUV2. The P-values of treatment effects calculated in section 'linear model and linear mixed model' are therefore used here.

Note: This analysis is not optimal as we use *pseudo negative control variables*, but we wish to provide this as a practical example.

```
# sponge data
sponge.nc <- sponge.trt_adjp > 0.05

# AD data
ad.nc <- ad.trt_adjp > 0.05

# HD data
hd.nc <- hd.trt_p > 0.05
```

Note: In HD data, there is no OTU having statistically siginificant difference between genotypes. We use P-values instead of adjusted P-values to set the negative controls.

We then apply the *RUV2* function. This function needs to specify the number of unwanted factors (components of negative controls), which we arbitrarily set to k = 3. We then extract the P-values of treatment effects considering unwanted batch effects after FDR adjustment.

```
# sponge data
sponge.ruv2 <- RUV2(Y = sponge.tss.clr, X = sponge.trt,
                    ctl = sponge.nc, k = 3) # k is subjective
sponge.ruv2.trt_p <- sponge.ruv2$p
sponge.ruv2.trt_adjp <- p.adjust(sponge.ruv2.trt_p, method="fdr")

# AD data
ad.ruv2 <- RUV2(Y = ad.clr, X = ad.trt,
                ctl = ad.nc, k = 3) # k is subjective
```

```r
ad.ruv2.trt_p <- ad.ruv2$p
ad.ruv2.trt_adjp <- p.adjust(ad.ruv2.trt_p, method="fdr")

# HD data
hd.ruv2 <- RUV2(Y = hd.clr, X = hd.trt,
                ctl = hd.nc, k = 3) # k is subjective
hd.ruv2.trt_p <- hd.ruv2$p
hd.ruv2.trt_adjp <- p.adjust(hd.ruv2.trt_p, method="fdr")
```

3.1.4 RUV4

RUV4 is an updated version of RUV2 that uses negative control variables and the residual matrix that has no treatment effect to estimate unwanted batch effects.

RUV4 also requires to specify the number of unwanted factors as *RUV2*, which we estimate with the function *'getK'* (from the *ruv* package). This function is only for *RUV4* and the estimated k is not always suitable. If the estimated k is 0, k can be set to 1 instead to account for unwanted variation captured from negative controls.

```r
# sponge data
sponge.k.obj <- getK(Y = sponge.tss.clr, X = sponge.trt, ctl = sponge.nc)
sponge.k <- sponge.k.obj$k
sponge.k <- ifelse(sponge.k !=0, sponge.k, 1)
sponge.ruv4 <- RUV4(Y = sponge.tss.clr, X = sponge.trt, ctl = sponge.nc, k = sponge.k)
sponge.ruv4.trt_p <- sponge.ruv4$p
sponge.ruv4.trt_adjp <- p.adjust(sponge.ruv4.trt_p, method="fdr")

# AD data
ad.k.obj <- getK(Y = ad.clr, X = ad.trt, ctl = ad.nc)
ad.k <- ad.k.obj$k
ad.k <- ifelse(ad.k !=0, ad.k, 1)
ad.ruv4 <- RUV4(Y = ad.clr, X = ad.trt, ctl = ad.nc, k = ad.k)
ad.ruv4.trt_p <- ad.ruv4$p
ad.ruv4.trt_adjp <- p.adjust(ad.ruv4.trt_p, method="fdr")

# HD data
hd.k.obj <- getK(Y = hd.clr, X = hd.trt, ctl = hd.nc)
hd.k <- hd.k.obj$k
hd.k <- ifelse(hd.k !=0, hd.k, 1)
hd.ruv4 <- RUV4(Y = hd.clr, X = hd.trt, ctl = hd.nc, k = hd.k)
hd.ruv4.trt_p <- hd.ruv4$p
hd.ruv4.trt_adjp <- p.adjust(hd.ruv4.trt_p, method="fdr")
```

3.2 Correcting for batch effects

An alternative approach to manage batch effects is to remove batch effects from the original microbiome data, then use the corrected data in any subsequent data analysis. Compared with methods accounting for batch effects, batch effect correction methods are practical and enable broader application in a variety of analyses. However, the methods do not consider the correlation (linear) and interaction (non-linear) between batch and treatment effects and may result in over-adjusted batch effects, statistical power loss, and an inability to detect variables associated with the treatment effect.

3.2.1 BMC (batch mean centering)

We center data within a batch across all variables. As a result, each batch mean is standardised to zero. BMC has several disadvantages: it is a univariate method and it is not optimal for non-Gaussian distributed microbiome data.

```
# Sponge data
sponge.b1 <- scale(sponge.tss.clr[sponge.batch == 1, ], center = TRUE, scale = FALSE)
sponge.b2 <- scale(sponge.tss.clr[sponge.batch == 2, ], center = TRUE, scale = FALSE)
sponge.bmc <- rbind(sponge.b1, sponge.b2)
sponge.bmc <- sponge.bmc[rownames(sponge.tss.clr), ]
```

```
# AD data
ad.b1 <- scale(ad.clr[ad.batch == '09/04/2015', ], center = TRUE, scale = FALSE)
ad.b2 <- scale(ad.clr[ad.batch == '14/04/2016', ], center = TRUE, scale = FALSE)
ad.b3 <- scale(ad.clr[ad.batch == '14/11/2016', ], center = TRUE, scale = FALSE)
ad.b4 <- scale(ad.clr[ad.batch == '01/07/2016', ], center = TRUE, scale = FALSE)
ad.b5 <- scale(ad.clr[ad.batch == '21/09/2017', ], center = TRUE, scale = FALSE)
ad.bmc <- rbind(ad.b1, ad.b2, ad.b3, ad.b4, ad.b5)
ad.bmc <- ad.bmc[rownames(ad.clr), ]
```

3.2.2 ComBat

ComBat requires known and systematic batch effects. If the treatment information is known, the option *mod* can be fitted with a full model having treatment informationn to efficiently maintain enough treatment variation (see examples below for sponge and AD data). The *mod* can also be NULL if the treatment information is unknown.

```
# Sponge data
sponge.combat <- t(ComBat(t(sponge.tss.clr), batch = sponge.batch,
                          mod = sponge.mod, par.prior = F, prior.plots = F))
```

```
## Found2batches

## Adjusting for1covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding nonparametric adjustments

## Adjusting the Data
```

```
# AD data
ad.combat <- t(ComBat(t(ad.clr), batch = ad.batch,
                      mod = ad.mod, par.prior = F, prior.plots = F))
```

```
## Found5batches

## Adjusting for1covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors

## Finding nonparametric adjustments

## Adjusting the Data
```

3.2.3 removeBatchEffect

removeBatchEffect is a function implemented in the LIMMA package that fits a linear model for each variable given a series of conditions as explanatory variables, including the batch effect and treatment effect. Contrary to a standard linear or linear mixed models (see section 'linear model and linear mixed model') that simultaneously estimate treatment and batch effects, *removeBatchEffect* subtracts the batch effect from the original data, resulting in a residual matrix that contains any and only treatment effect. The option *design* is the same as option *mod* in ComBat.

```
# Sponge data
sponge.limma <- t(removeBatchEffect(t(sponge.tss.clr), batch = sponge.batch,
                                    design = sponge.mod))
```

```
ad.limma <- t(removeBatchEffect(t(ad.clr), batch = ad.batch,
                                design = ad.mod))
```

3.2.4 FAbatch

FAbatch is a combination of location-scale adjustment and factor analysis. We fit *y* with the treatment information and *batch* with batch information. Since this method only accepts numeric variables, the levels of variables are changed to be numeric (e.g. '1', '2' and so on). However, on our example, FAbatch did not converge, potentially leading to suboptimal results. We therefore did not compare the results from FAbatch with those from other methods.

```
# sponge data
sponge.fabatch.obj <- fabatch(x = sponge.tss.clr,
                              y = as.factor(as.numeric(sponge.trt)),
                              batch = sponge.batch)
sponge.fabatch <- sponge.fabatch.obj$xadj

# ad data
ad.fabatch.obj <- fabatch(x = ad.clr,
                          y = as.factor(as.numeric(ad.trt)),
                          batch = as.factor(as.numeric(ad.batch)))
ad.fabatch <- ad.fabatch.obj$xadj
```

3.2.5 Percentile normalisation

Percentile normalisation (PN) was developed for microbiome data integration. For each batch, it transforms the relative abundance of control samples to their own percentiles while converting the relative abundance of case samples into the percentiles of their corresponding control distribution. The method thus only applies to case-control studies, and may lose a large amount of information as it uses percentiles rather than the original values.

Note: PN applies on TSS scaled data (Gibbons et al., 2018). As we added an offset on sponge TSS scaled data, we re-applied the TSS.

```
# sponge data
sponge.tss <- t(apply(sponge.tss, 1, function(x){x/sum(x)}))
sponge.percentile <- percentile_norm(data = sponge.tss, batch = sponge.batch,
                                    trt = sponge.trt)



# ad data
# TSS
ad.tss <- t(apply(ad.count.keep, 1, function(x){x/sum(x)}))
```

```r
ad.percentile <- percentile_norm(data = ad.tss, batch = ad.batch,
                                 trt = ad.trt)
```

3.2.6 SVD-based method

We center and scale the data before SVD. After SVD, we deflate the first component, which has the highest variation and is assumed to be related to batch effects.

```r
# sponge data
sponge.sd <- apply(sponge.tss.clr, 2, sd) # calculate standard deviation
sponge.mean <- apply(sponge.tss.clr, 2, mean) # calculate mean
sponge.X <- scale(sponge.tss.clr, center = T, scale = T) # center and scale

sponge.m <- crossprod(sponge.X) # generate a square matrix
sponge.m.svd <- svd(sponge.m) # SVD
# barplot(sponge.m.svd$d)

# extract 1st singular vectors
sponge.a1 <- sponge.m.svd$u[ ,1]
sponge.b1 <- sponge.m.svd$v[ ,1]

# deflate component 1 from the data
sponge.t1 <- sponge.X %*% sponge.a1 / drop(sqrt(crossprod(sponge.a1)))
sponge.c1 <- crossprod(sponge.X, sponge.t1) / drop(crossprod(sponge.t1))
sponge.svd.defl.matrix1  <- sponge.X - sponge.t1 %*% t(sponge.c1)

# add back mean and standard deviation
sponge.svd <- sponge.svd.defl.matrix1
sponge.svd[1:nrow(sponge.svd), 1:ncol(sponge.svd)] = NA
for(i in 1:ncol(sponge.svd.defl.matrix1)){
  for(j in 1:nrow(sponge.svd.defl.matrix1)){
    sponge.svd[j,i] = sponge.svd.defl.matrix1[j,i]*sponge.sd[i] + sponge.mean[i]
  }
}
```

```r
# ad data
ad.sd <- apply(ad.clr, 2, sd) # calculate standard deviation
ad.mean <- apply(ad.clr, 2, mean) # calculate mean
ad.X <- scale(ad.clr,center = T, scale = T) # center and scale

ad.m <- crossprod(ad.X) # generate a square matrix
ad.m.svd <- svd(ad.m) # SVD
# barplot(ad.m.svd$d)

# extract 1st singular vectors
ad.a1 <- ad.m.svd$u[ ,1]
ad.b1 <- ad.m.svd$v[ ,1]

# deflate component 1 from the data
ad.t1 <- ad.X %*% ad.a1 / drop(sqrt(crossprod(ad.a1)))
ad.c1 <- crossprod(ad.X, ad.t1) / drop(crossprod(ad.t1))
ad.svd.defl.matrix1  <- ad.X - ad.t1 %*% t(ad.c1)
```

```r
# add back mean and standard deviation
ad.svd <- ad.svd.defl.matrix1
ad.svd[1:nrow(ad.svd), 1:ncol(ad.svd)] = NA
for(i in 1:ncol(ad.svd.defl.matrix1)){
  for(j in 1:nrow(ad.svd.defl.matrix1)){
    ad.svd[j,i] = ad.svd.defl.matrix1[j,i]*ad.sd[i] + ad.mean[i]
  }
}
```

3.2.7 RUVIII

RUVIII needs not only negative control variables as RUV2 and RUV4, but also technical sample replicates. As only AD data include sample replicates, we illustrate RUVIII on these data only.

In contrast to RUV2 and RUV4, RUVIII is a multivariate method that accounts for the dependency between microbial variables, but it is currently limited to the correction of technical and computational sources of batch effects.

Note: RUVIII requires the number of negative control variables to be larger than the sample size in order to fully use the sample information.

```r
# ad data only
ad.replicates <- ad.metadata$sample_name.data.extraction
ad.replicates.matrix <- replicate.matrix(ad.replicates)

ad.ruvIII <- RUVIII(Y = ad.clr, M = ad.replicates.matrix, ctl = ad.nc)
rownames(ad.ruvIII) <- rownames(ad.clr)
```

Chapter 4

Methods evaluation

After batch effect adjustment, it is essential to evaluate its effectiveness. For simplicity, we focus only on strategies to assess batch effect *correction* methods, as methods that account for the batch effect often imply it has been assessed internally in the statistical model.

4.1 Diagnostic plots

Diagnostic plots are useful to visually evaluate for post-correction batch effects, as presented earlier in section 'batch effect detection'.

4.1.1 Principal component analysis (PCA) with density plot per component

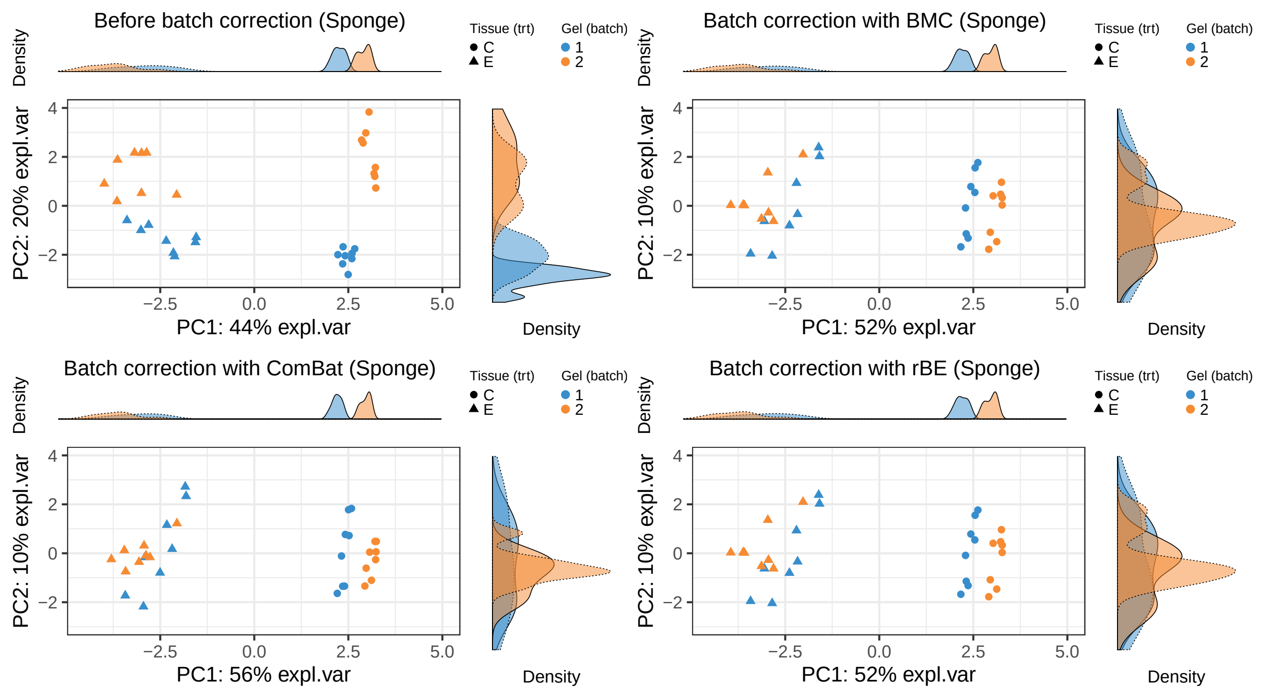
We apply PCA on both sponge and AD data before and after correction with different methods.

```
# sponge data
sponge.pca.before <- pca(sponge.tss.clr, ncomp = 3)
sponge.pca.bmc <- pca(sponge.bmc, ncomp = 3)
sponge.pca.combat <- pca(sponge.combat, ncomp = 3)
sponge.pca.limma <- pca(sponge.limma, ncomp = 3)
sponge.pca.percentile <- pca(sponge.percentile, ncomp = 3)
sponge.pca.svd <- pca(sponge.svd, ncomp = 3)

# ad data
ad.pca.before <- pca(ad.clr, ncomp = 3)
ad.pca.bmc <- pca(ad.bmc, ncomp = 3)
ad.pca.combat <- pca(ad.combat, ncomp = 3)
ad.pca.limma <- pca(ad.limma, ncomp = 3)
ad.pca.percentile <- pca(ad.percentile, ncomp = 3)
ad.pca.svd <- pca(ad.svd, ncomp = 3)
ad.pca.ruv <- pca(ad.ruvIII, ncomp = 3)
```

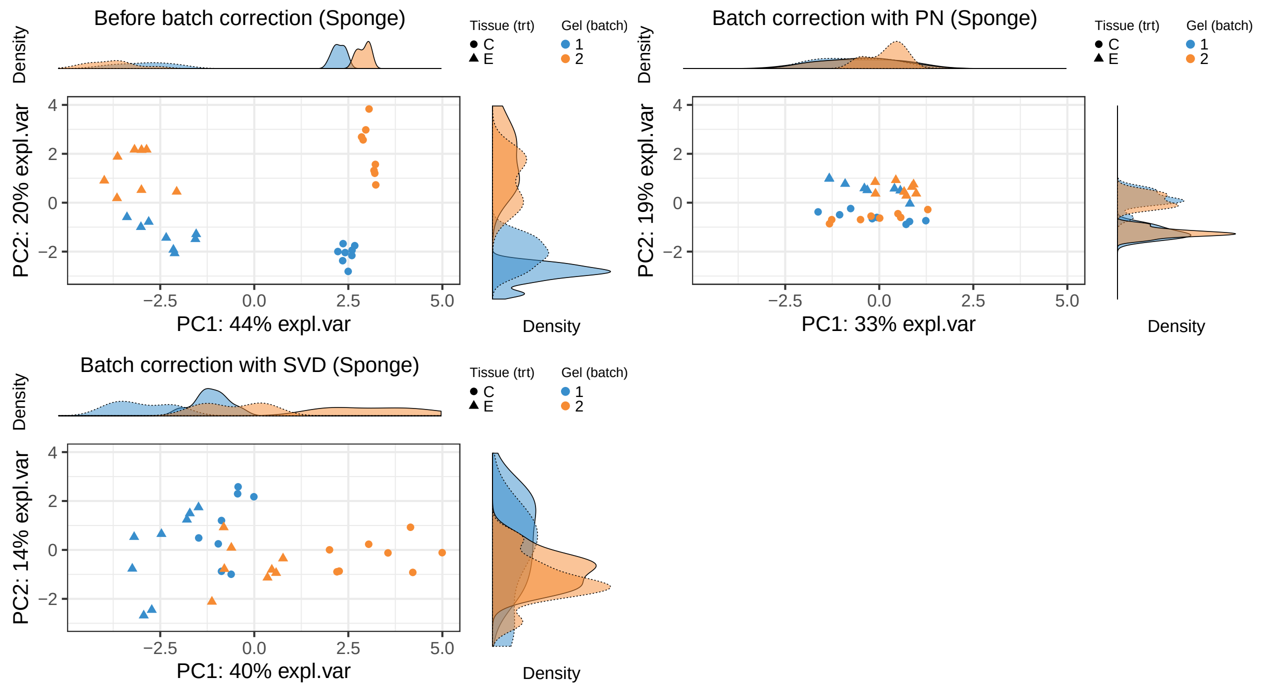
We then plot these PCA sample plots with denisty plots per PC.

Note: BMC: batch mean centering; PN: percentile normalisation; rBE: removeBatchEffect; SVD: singular value decomposition

```
grid.arrange(sponge.pca.plot.before, sponge.pca.plot.bmc,
             sponge.pca.plot.combat, sponge.pca.plot.limma, ncol = 2)
```

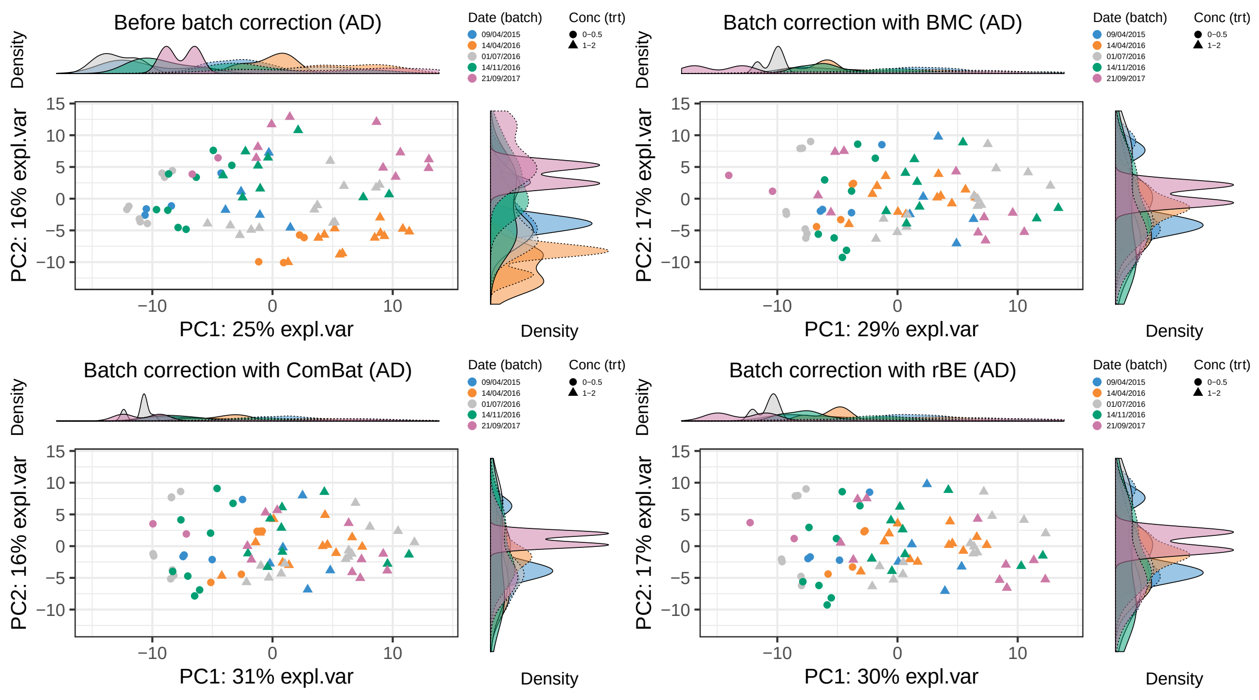


```r
grid.arrange(sponge.pca.plot.before, sponge.pca.plot.percentile,
             sponge.pca.plot.svd, ncol = 2)
```

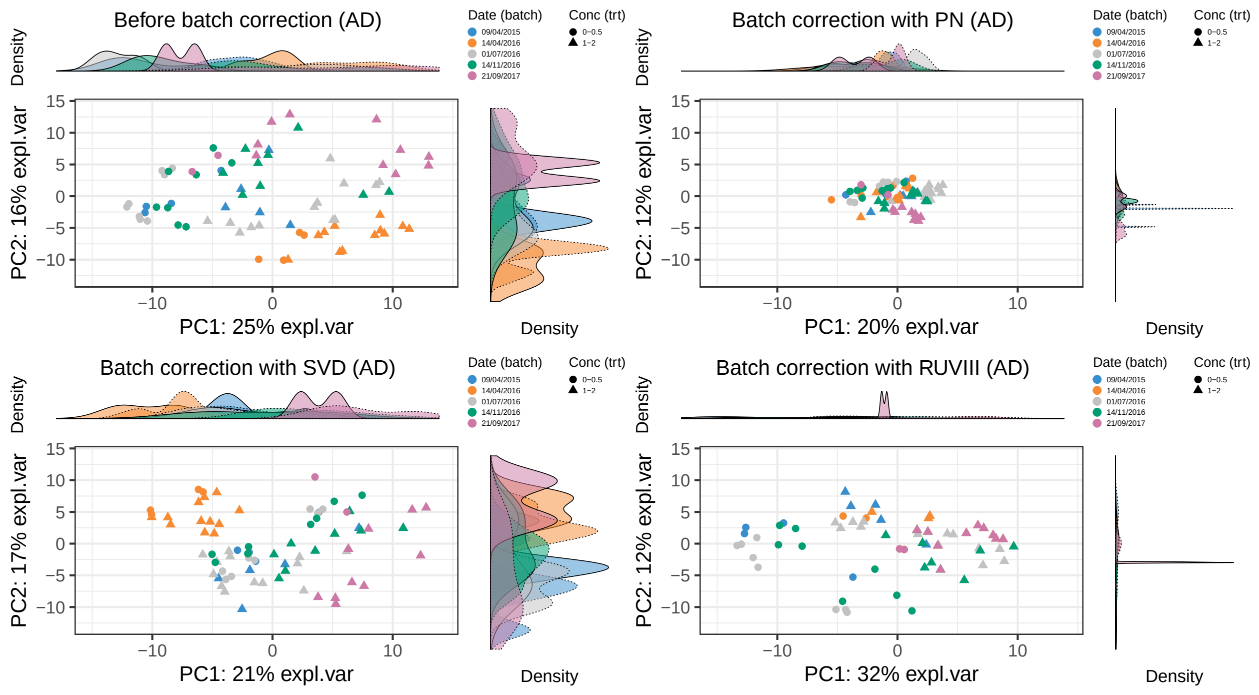


In sponge data, SVD performed the worse, compared to BMC, removeBatchEffect and percentile normalisation, as it did not remove batch effects but removed tissue variation instead.

```r
grid.arrange(ad.pca.plot.before, ad.pca.plot.bmc,
             ad.pca.plot.combat, ad.pca.plot.limma, ncol = 2)
```



```r
grid.arrange(ad.pca.plot.before, ad.pca.plot.percentile,
             ad.pca.plot.svd, ad.pca.plot.ruv, ncol = 2)
```

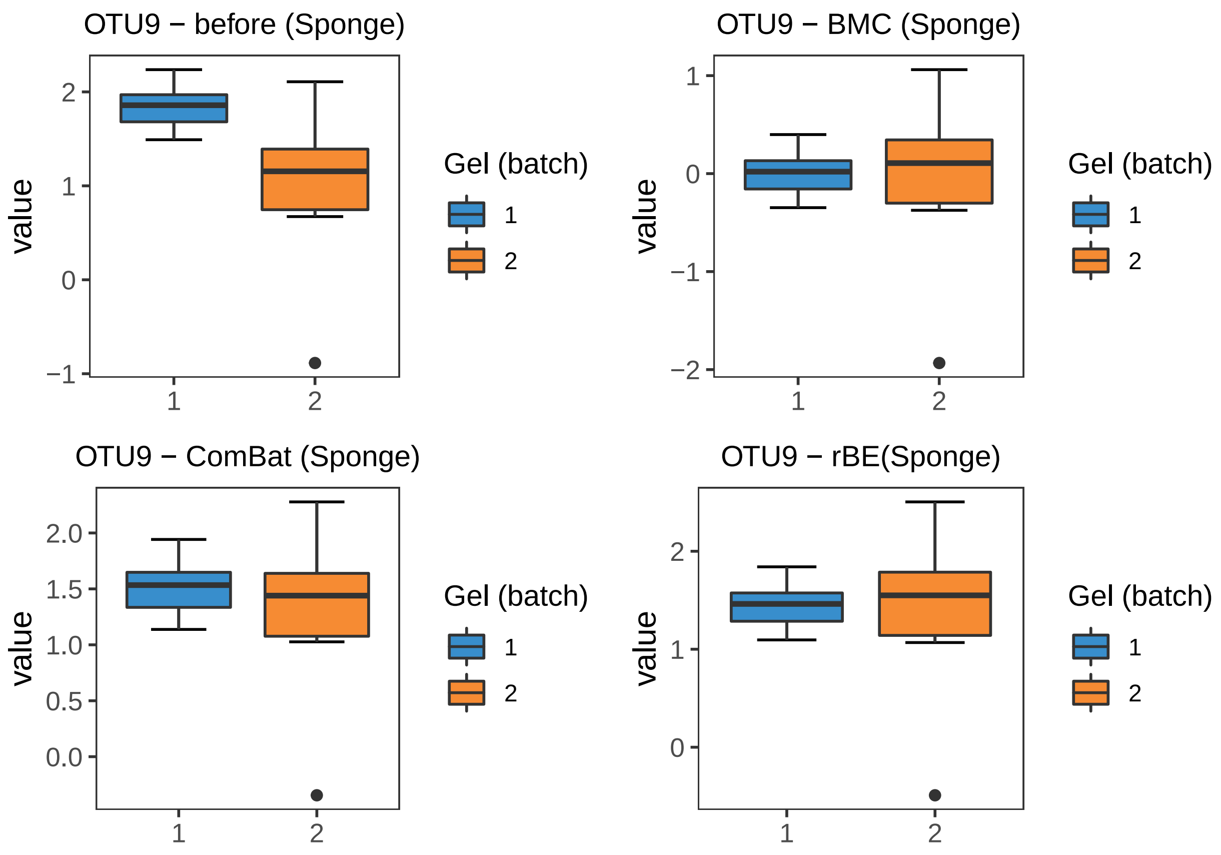


In AD data, BMC, removeBatchEffect, percentile normalisation and RUVIII removed the batch effects, and maintained the treatment effects. SVD did not removed the batch effect but decreased the treatment effects.

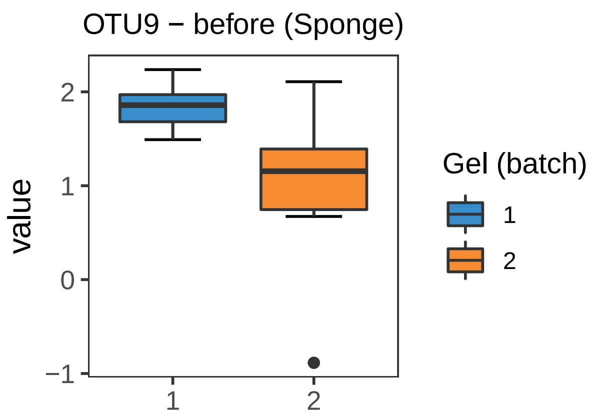
4.1.2 Density plot and box plot

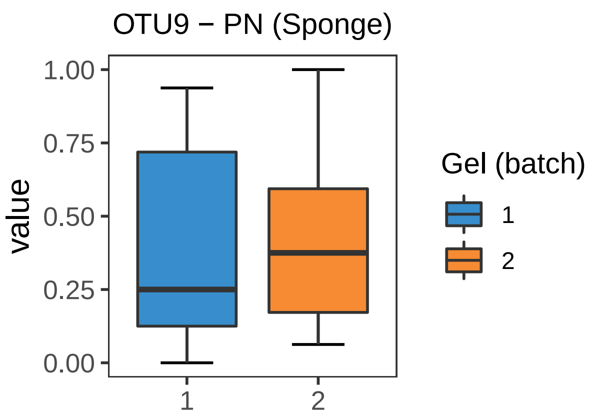
```r
# sponge data
sponge.before.df <- data.frame(value = sponge.tss.clr[ ,9], batch = sponge.batch)
sponge.boxplot.before <- box_plot_fun(data = sponge.before.df,
                                      x = sponge.before.df$batch,
                                      y = sponge.before.df$value,
                                      title = 'OTU9 - before (Sponge)',
                                      batch.legend.title = 'Gel (batch)')

sponge.bmc.df <- data.frame(value = sponge.bmc[ ,9], batch = sponge.batch)
sponge.boxplot.bmc <- box_plot_fun(data = sponge.bmc.df,
                                   x = sponge.bmc.df$batch,
                                   y = sponge.bmc.df$value,
                                   title = 'OTU9 - BMC (Sponge)',
                                   batch.legend.title = 'Gel (batch)')


sponge.combat.df <- data.frame(value = sponge.combat[ ,9], batch = sponge.batch)
sponge.boxplot.combat <- box_plot_fun(data = sponge.combat.df,
                                      x = sponge.combat.df$batch,
                                      y = sponge.combat.df$value,
                                      title = 'OTU9 - ComBat (Sponge)',
                                      batch.legend.title = 'Gel (batch)')


sponge.limma.df <- data.frame(value = sponge.limma[ ,9], batch = sponge.batch)
sponge.boxplot.limma <- box_plot_fun(data = sponge.limma.df,
                                     x = sponge.limma.df$batch,
                                     y = sponge.limma.df$value,
                                     title = 'OTU9 - rBE(Sponge)',
                                     batch.legend.title = 'Gel (batch)')

sponge.percentile.df <- data.frame(value = sponge.percentile[ ,9], batch = sponge.batch)
sponge.boxplot.percentile <- box_plot_fun(data = sponge.percentile.df,
                                          x = sponge.percentile.df$batch,
                                          y = sponge.percentile.df$value,
                                          title = 'OTU9 - PN (Sponge)',
                                          batch.legend.title = 'Gel (batch)')

sponge.svd.df <- data.frame(value = sponge.svd[ ,9], batch = sponge.batch)
sponge.boxplot.svd <- box_plot_fun(data = sponge.svd.df,
                                   x = sponge.svd.df$batch,
                                   y = sponge.svd.df$value,
                                   title = 'OTU9 - SVD (Sponge)',
                                   batch.legend.title = 'Gel (batch)')
```

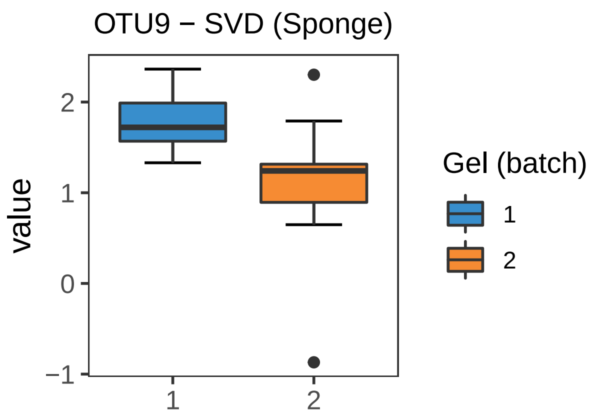
```r
grid.arrange(sponge.boxplot.before, sponge.boxplot.bmc,
             sponge.boxplot.combat, sponge.boxplot.limma, ncol = 2)
```



```r
grid.arrange(sponge.boxplot.before, sponge.boxplot.percentile,
             sponge.boxplot.svd, ncol = 2)
```





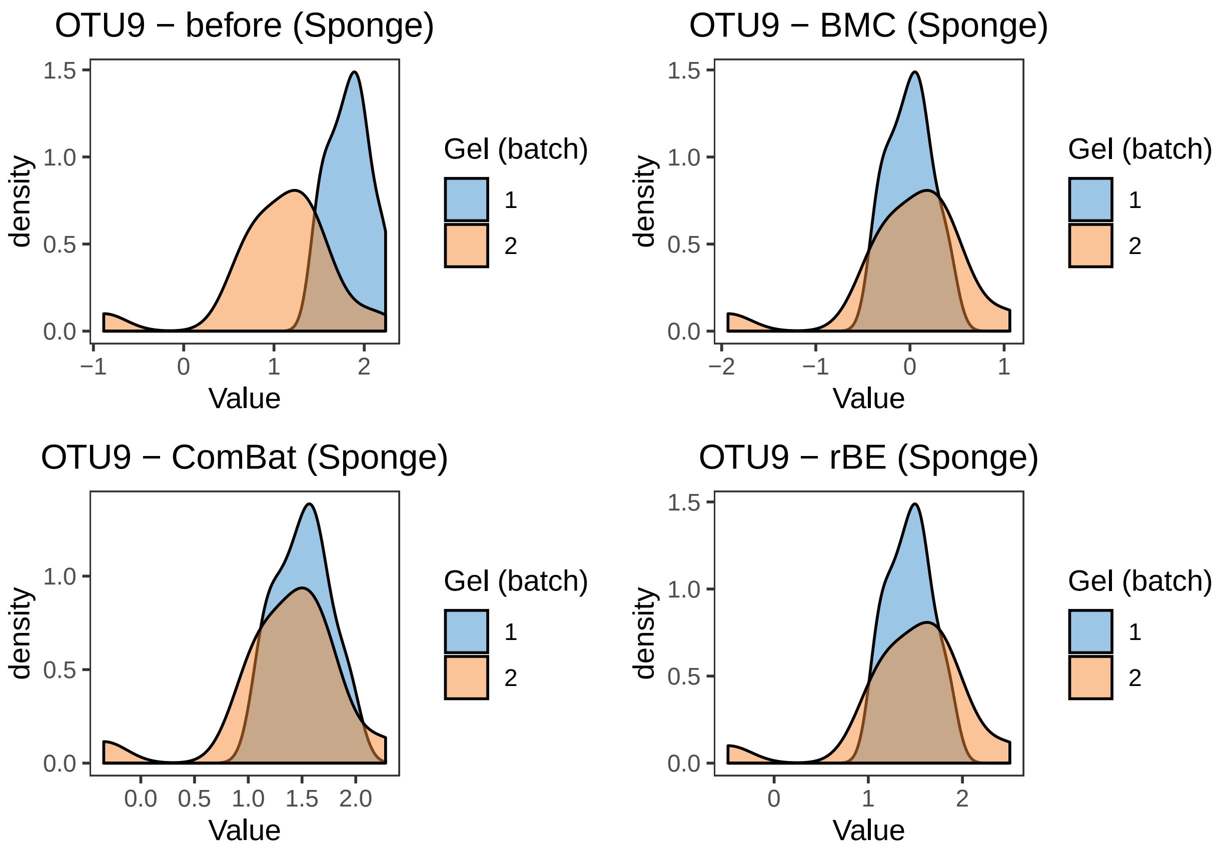


From the boxplots of OTU9 in sponge data, the difference between two batches are removed by BMC, ComBat, removeBatchEffect and percentile normalisation correction. Among these methods, percentile normalisation does not remove as much batch difference as the other methods.

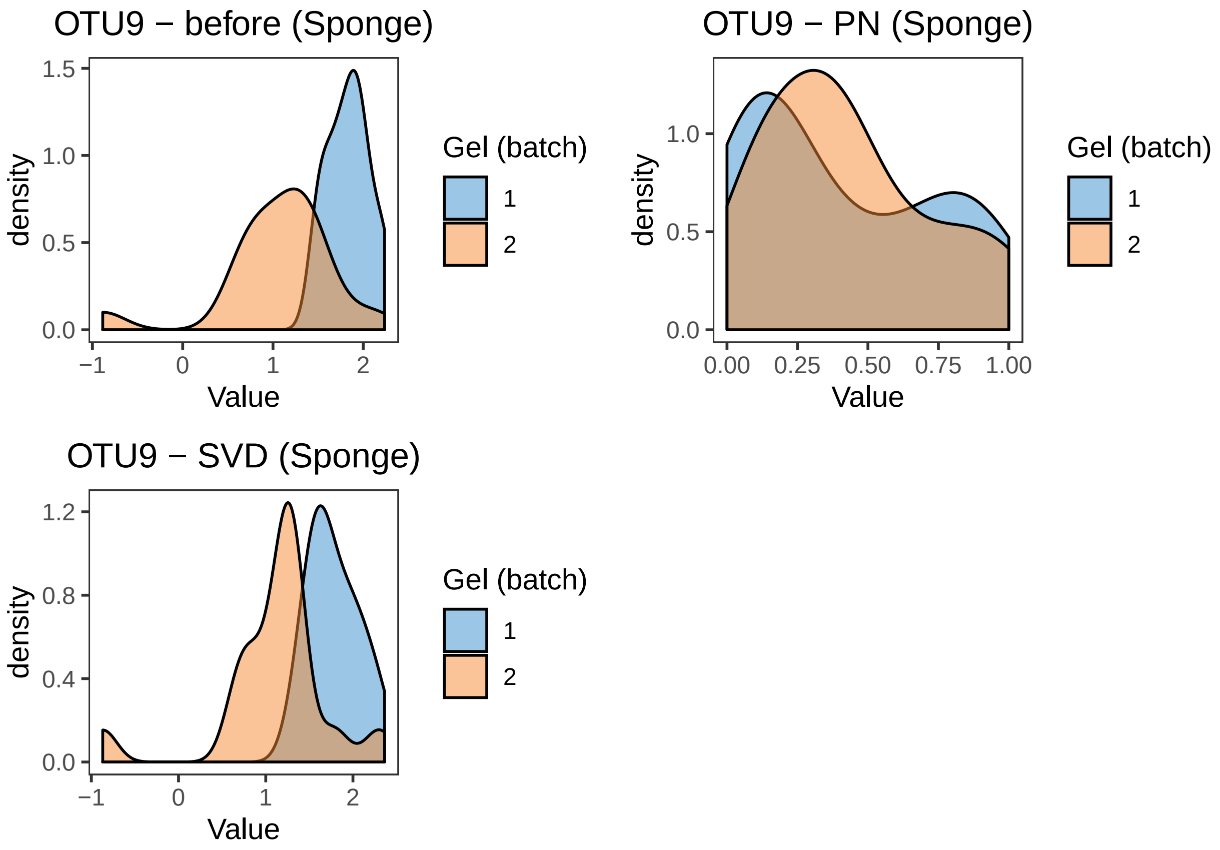
```
# density plot
# before
sponge.dens.before <- ggplot(sponge.before.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU9 - before (Sponge)', x = 'Value', fill = 'Gel (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())

# BMC
sponge.dens.bmc <- ggplot(sponge.bmc.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU9 - BMC (Sponge)', x = 'Value', fill = 'Gel (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())


# ComBat
sponge.dens.combat <- ggplot(sponge.combat.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU9 - ComBat (Sponge)', x = 'Value', fill = 'Gel (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())
```

```
# removeBatchEffect
sponge.dens.limma <- ggplot(sponge.limma.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU9 - rBE (Sponge)', x = 'Value', fill = 'Gel (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())


# percentile normal
sponge.dens.percentile <- ggplot(sponge.percentile.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU9 - PN (Sponge)', x = 'Value', fill = 'Gel (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())


# SVD
sponge.dens.svd <- ggplot(sponge.svd.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU9 - SVD (Sponge)', x = 'Value', fill = 'Gel (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())
```

```
grid.arrange(sponge.dens.before, sponge.dens.bmc,
             sponge.dens.combat, sponge.dens.limma, ncol = 2)
```



```
grid.arrange(sponge.dens.before, sponge.dens.percentile,
             sponge.dens.svd, ncol = 2)
```



We can also assess the effect of batch in a linear model before and after batch correction.

```
# P-values
sponge.lm.before <- lm(sponge.tss.clr[ ,9] ~ sponge.trt + sponge.batch)
summary(sponge.lm.before)
```

```
##
## Call:
## lm(formula = sponge.tss.clr[, 9] ~ sponge.trt + sponge.batch)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.87967 -0.24705  0.04588  0.24492  1.00757
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.7849     0.1497  11.922 1.06e-12 ***
## sponge.trtE     0.1065     0.1729   0.616    0.543
## sponge.batch2  -0.7910     0.1729  -4.575 8.24e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.489 on 29 degrees of freedom
## Multiple R-squared:  0.4236, Adjusted R-squared:  0.3839
## F-statistic: 10.66 on 2 and 29 DF,  p-value: 0.0003391
```

Before correction, the batch effect is statistically significant (P $<$ 0.001), as indicated in the **sponge.batch2** row.

```
sponge.lm.bmc <- lm(sponge.bmc[ ,9] ~ sponge.trt + sponge.batch)
summary(sponge.lm.bmc)
```

```
##
## Call:
## lm(formula = sponge.bmc[, 9] ~ sponge.trt + sponge.batch)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.87967 -0.24705  0.04588  0.24492  1.00757
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5.323e-02  1.497e-01  -0.356    0.725
## sponge.trtE    1.065e-01  1.729e-01   0.616    0.543
## sponge.batch2 -3.925e-17  1.729e-01   0.000    1.000
##
## Residual standard error: 0.489 on 29 degrees of freedom
## Multiple R-squared:  0.01291,    Adjusted R-squared:  -0.05517
## F-statistic: 0.1896 on 2 and 29 DF,  p-value: 0.8283
```

After BMC correction, the batch effect is not statistically significant (P $>$ 0.05), as indicated in the **sponge.batch2** row.

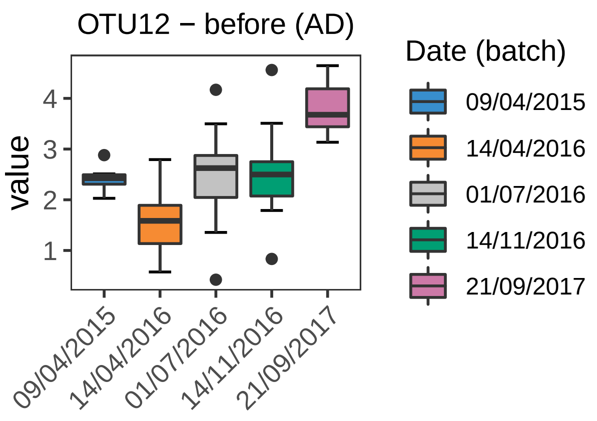
The results from other batch correction methods are similar as BMC correction.

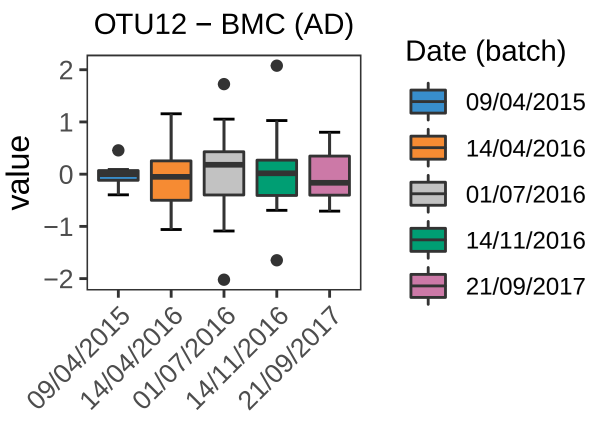
```
sponge.lm.combat <- lm(sponge.combat[ ,9] ~ sponge.trt + sponge.batch)
summary(sponge.lm.combat)

sponge.lm.limma <- lm(sponge.limma[ ,9] ~ sponge.trt + sponge.batch)
summary(sponge.lm.limma)

sponge.lm.percentile <- lm(sponge.percentile[ ,9] ~ sponge.trt + sponge.batch)
summary(sponge.lm.percentile)

sponge.lm.svd <- lm(sponge.svd[ ,9] ~ sponge.trt + sponge.batch)
summary(sponge.lm.svd)
```

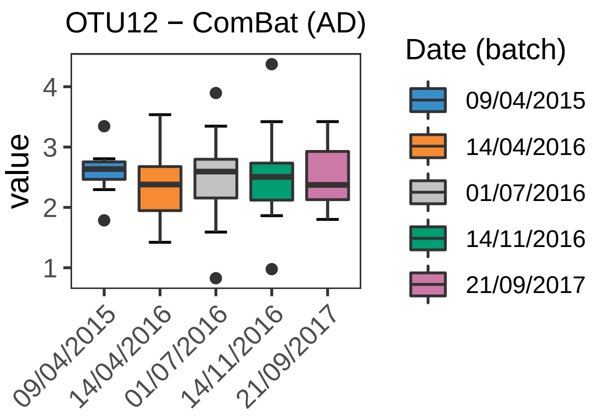
In sponge data, the results of density plots and P-values of batch effects from linear models reach a consesus with boxplots, the difference between two batches are removed by BMC, ComBat, removeBatchEffect and percentile normalisation correction. We observe that percentile normalisation modifies the distribution of the samples from the two batches.

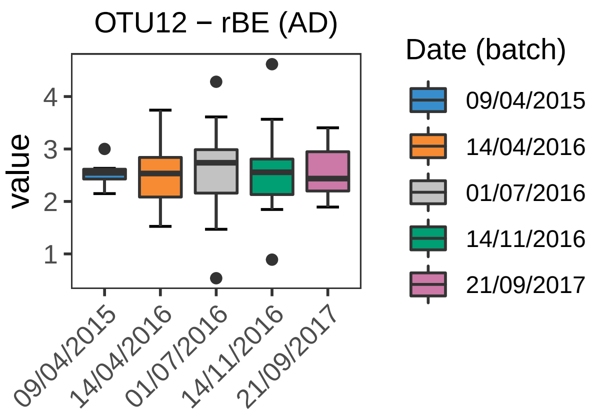
```
# ad data
# boxplot
ad.before.df <- data.frame(value = ad.clr[ ,1], batch = ad.batch)
ad.boxplot.before <- box_plot_fun(data = ad.before.df, x = ad.before.df$batch,
                                  y = ad.before.df$value,
                                  title = 'OTU12 - before (AD)',
                                  batch.legend.title = 'Date (batch)',
                                  x.angle = 45, x.hjust = 1)



ad.bmc.df <- data.frame(value = ad.bmc[ ,1], batch = ad.batch)
ad.boxplot.bmc <- box_plot_fun(data = ad.bmc.df,x = ad.bmc.df$batch,
                               y = ad.bmc.df$value,
                               title = 'OTU12 - BMC (AD)',
                               batch.legend.title = 'Date (batch)',
                               x.angle = 45, x.hjust = 1)


ad.combat.df <- data.frame(value = ad.combat[ ,1], batch = ad.batch)
ad.boxplot.combat <- box_plot_fun(data = ad.combat.df, x = ad.combat.df$batch,
                                  y = ad.combat.df$value,
                                  title = 'OTU12 - ComBat (AD)',
                                  batch.legend.title = 'Date (batch)',
                                  x.angle = 45, x.hjust = 1)

ad.limma.df <- data.frame(value = ad.limma[ ,1], batch = ad.batch)
ad.boxplot.limma <- box_plot_fun(data = ad.limma.df,x = ad.limma.df$batch,
                                 y = ad.limma.df$value,
                                 title = 'OTU12 - rBE (AD)',
                                 batch.legend.title = 'Date (batch)',
                                 x.angle = 45, x.hjust = 1)


ad.percentile.df <- data.frame(value = ad.percentile[ ,1], batch = ad.batch)
ad.boxplot.percentile <- box_plot_fun(data = ad.percentile.df,x = ad.percentile.df$batch,
                                      y = ad.percentile.df$value,
                                      title = 'OTU12 - PN (AD)',
                                      batch.legend.title = 'Date (batch)',
                                      x.angle = 45, x.hjust = 1)

ad.svd.df <- data.frame(value = ad.svd[ ,1], batch = ad.batch)
ad.boxplot.svd <- box_plot_fun(data = ad.svd.df,x = ad.svd.df$batch,
                               y = ad.svd.df$value,
                               title = 'OTU12 - SVD (AD)',
                               batch.legend.title = 'Date (batch)',
                               x.angle = 45, x.hjust = 1)

ad.ruv.df <- data.frame(value = ad.ruvIII[ ,1], batch = ad.batch)
ad.boxplot.ruv <- box_plot_fun(data = ad.ruv.df,x = ad.ruv.df$batch,
                               y = ad.ruv.df$value,
                               title = 'OTU12 - RUVIII (AD)',
                               batch.legend.title = 'Date (batch)',
                               x.angle = 45, x.hjust = 1)
```

```
grid.arrange(ad.boxplot.before, ad.boxplot.bmc,
             ad.boxplot.combat, ad.boxplot.limma, ncol = 2)
```

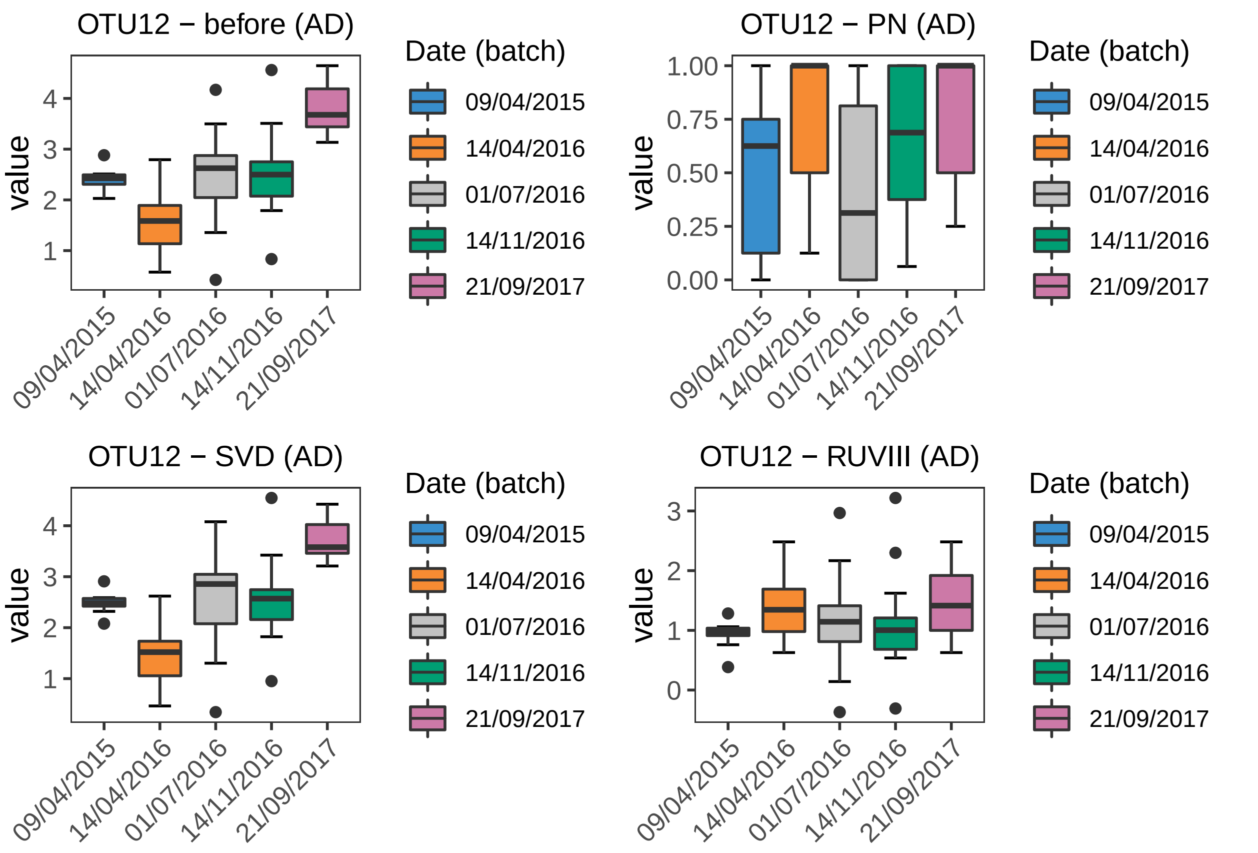








```
grid.arrange(ad.boxplot.before, ad.boxplot.percentile,
             ad.boxplot.svd, ad.boxplot.ruv, ncol = 2)
```

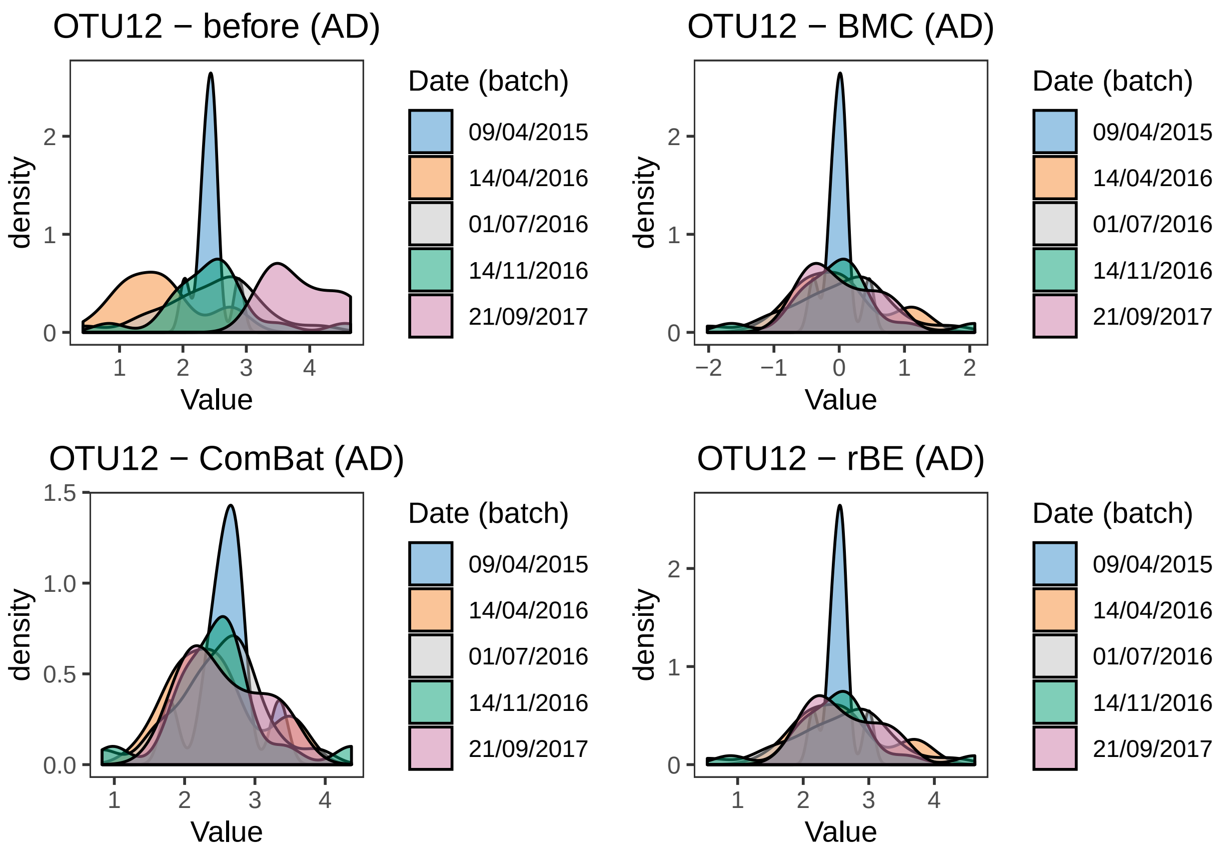


From the boxplots of OTU9 in AD data, the difference between five batches are removed by BMC, ComBat, removeBatchEffect and RUVIII correction. Among these methods, RUVIII did not remove as much batch difference as the other methods.

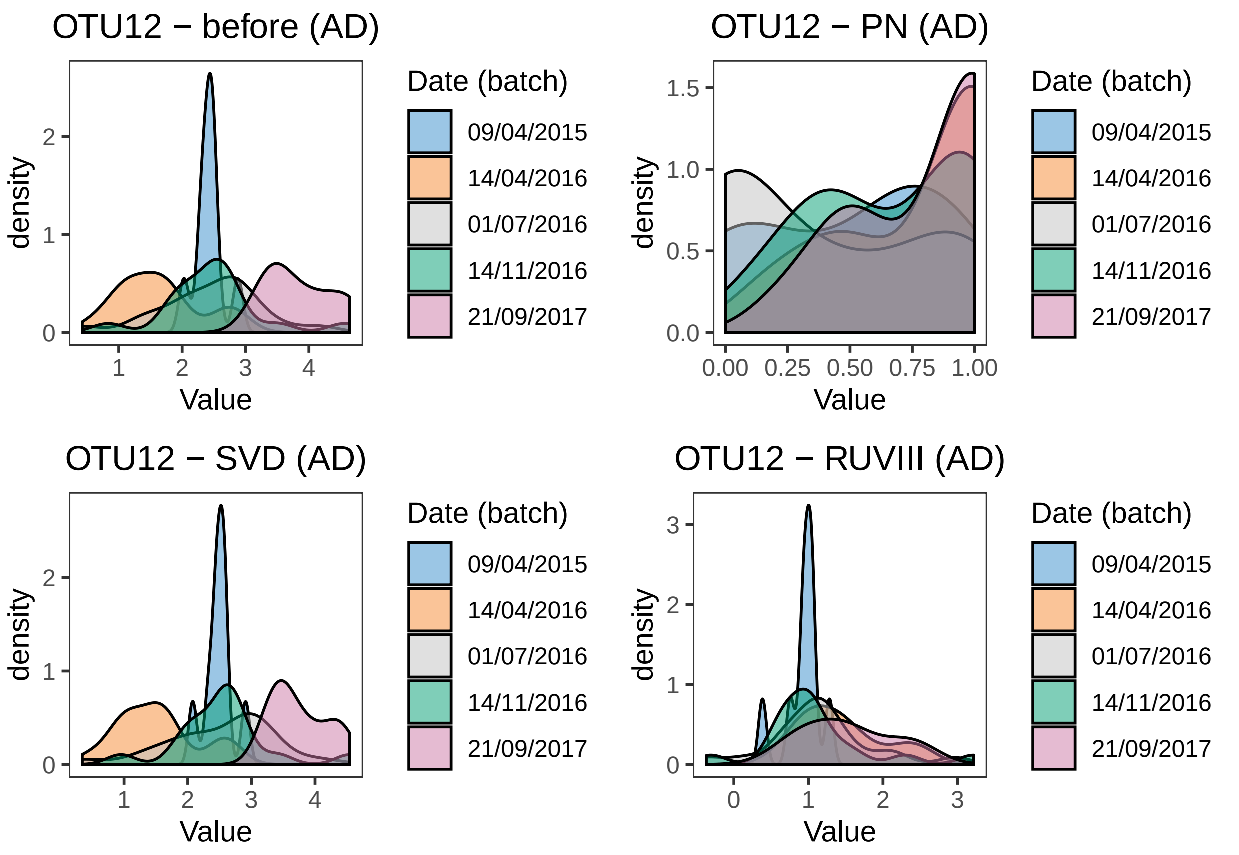
```
# density plot
# before
ad.dens.before <- ggplot(ad.before.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 - before (AD)', x = 'Value', fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())

# BMC
ad.dens.bmc <- ggplot(ad.bmc.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 - BMC (AD)', x = 'Value', fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())


# ComBat
ad.dens.combat <- ggplot(ad.combat.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 - ComBat (AD)', x = 'Value', fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())
```

```r
# removeBatchEffect
ad.dens.limma <- ggplot(ad.limma.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 - rBE (AD)', x = 'Value', fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())


# percentile norm
ad.dens.percentile <- ggplot(ad.percentile.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 - PN (AD)', x = 'Value', fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())


# SVD
ad.dens.svd <- ggplot(ad.svd.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 - SVD (AD)', x = 'Value', fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())


# RUVIII
ad.dens.ruv <- ggplot(ad.ruv.df, aes(x = value, fill = batch)) +
  geom_density(alpha = 0.5) + scale_fill_manual(values = color.mixo(1:10)) +
  labs(title = 'OTU12 - RUVIII (AD)', x = 'Value', fill = 'Date (batch)') +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5),
                     panel.grid = element_blank())
```

```r
grid.arrange(ad.dens.before, ad.dens.bmc,
             ad.dens.combat, ad.dens.limma, ncol = 2)
```



```r
grid.arrange(ad.dens.before, ad.dens.percentile,
             ad.dens.svd, ad.dens.ruv, ncol = 2)
```



Similar as sponge data, we can also assess the batch effect in AD data using linear model.

```
#p-values
ad.lm.before <- lm(ad.clr[ ,1] ~ ad.trt + ad.batch)
anova(ad.lm.before)
```

```
## Analysis of Variance Table
##
## Response: ad.clr[, 1]
##            Df Sum Sq Mean Sq F value    Pr(>F)
## ad.trt      1  1.460  1.4605  3.1001   0.08272 .
## ad.batch    4 32.889  8.2222 17.4532 6.168e-10 ***
## Residuals 69 32.506  0.4711
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#summary(ad.lm.before)
```

Before correction, the batch effect is statistically significant ($P < 0.001$), as indicated in the **ad.batch** row.

```
ad.lm.bmc <- lm(ad.bmc[ ,1] ~ ad.trt + ad.batch)
anova(ad.lm.bmc)
```

```
## Analysis of Variance Table
##
## Response: ad.bmc[, 1]
##            Df Sum Sq Mean Sq F value Pr(>F)
## ad.trt      1  0.631 0.63084  1.3391 0.2512
## ad.batch    4  0.036 0.00893  0.0190 0.9993
## Residuals 69 32.506 0.47110
```

```
#summary(ad.lm.bmc)
```

After BMC correction, the batch effect is not statistically significant ($P > 0.05$), as indicated in the **ad.batch** row.The results from other batch correction methods are similar as BMC correction.

```
ad.lm.combat <- lm(ad.combat[ ,1] ~ ad.trt + ad.batch)
anova(ad.lm.combat)
#summary(ad.lm.combat)

ad.lm.limma <- lm(ad.limma[ ,1] ~ ad.trt + ad.batch)
anova(ad.lm.limma)
#summary(ad.lm.limma)

ad.lm.percentile <- lm(ad.percentile[ ,1] ~ ad.trt + ad.batch)
anova(ad.lm.percentile)
#summary(ad.lm.percentile)

ad.lm.svd <- lm(ad.svd[ ,1] ~ ad.trt + ad.batch)
anova(ad.lm.svd)
#summary(ad.lm.svd)


ad.lm.ruv <- lm(ad.ruvIII[ ,1] ~ ad.trt + ad.batch)
anova(ad.lm.ruv)
#summary(ad.lm.ruv)
```

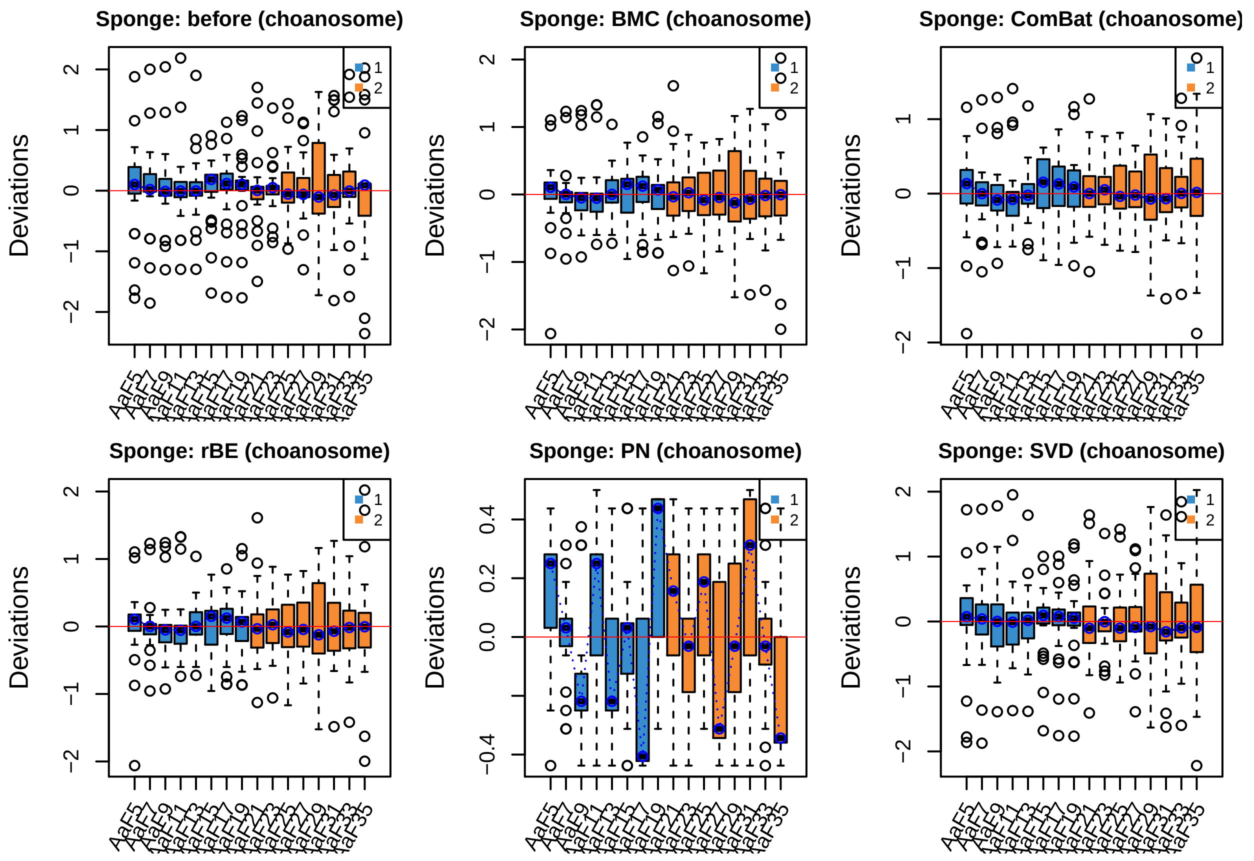
The results of density plots and P-values of batch effects from linear models reach a consesus with boxplots, the difference between two batches are removed by BMC, ComBat, removeBatchEffect and RUVIII correction.

4.1.3 RLE plots

In sponge data, we group the samples according to the tissue (choanosome / ectosome) and generate two RLE plots respectively in all datasets before and after batch correction with different methods:

```
# sponge data
# BMC
sponge.bmc_c <- sponge.bmc[sponge.trt == 'C', ]
sponge.bmc_e <- sponge.bmc[sponge.trt == 'E', ]

# ComBat
sponge.combat_c <- sponge.combat[sponge.trt == 'C', ]
sponge.combat_e <- sponge.combat[sponge.trt == 'E', ]

# rBE
sponge.limma_c <- sponge.limma[sponge.trt == 'C', ]
sponge.limma_e <- sponge.limma[sponge.trt == 'E', ]

# PN
sponge.percentile_c <- sponge.percentile[sponge.trt == 'C', ]
sponge.percentile_e <- sponge.percentile[sponge.trt == 'E', ]

# SVD
sponge.svd_c <- sponge.svd[sponge.trt == 'C', ]
sponge.svd_e <- sponge.svd[sponge.trt == 'E', ]
```

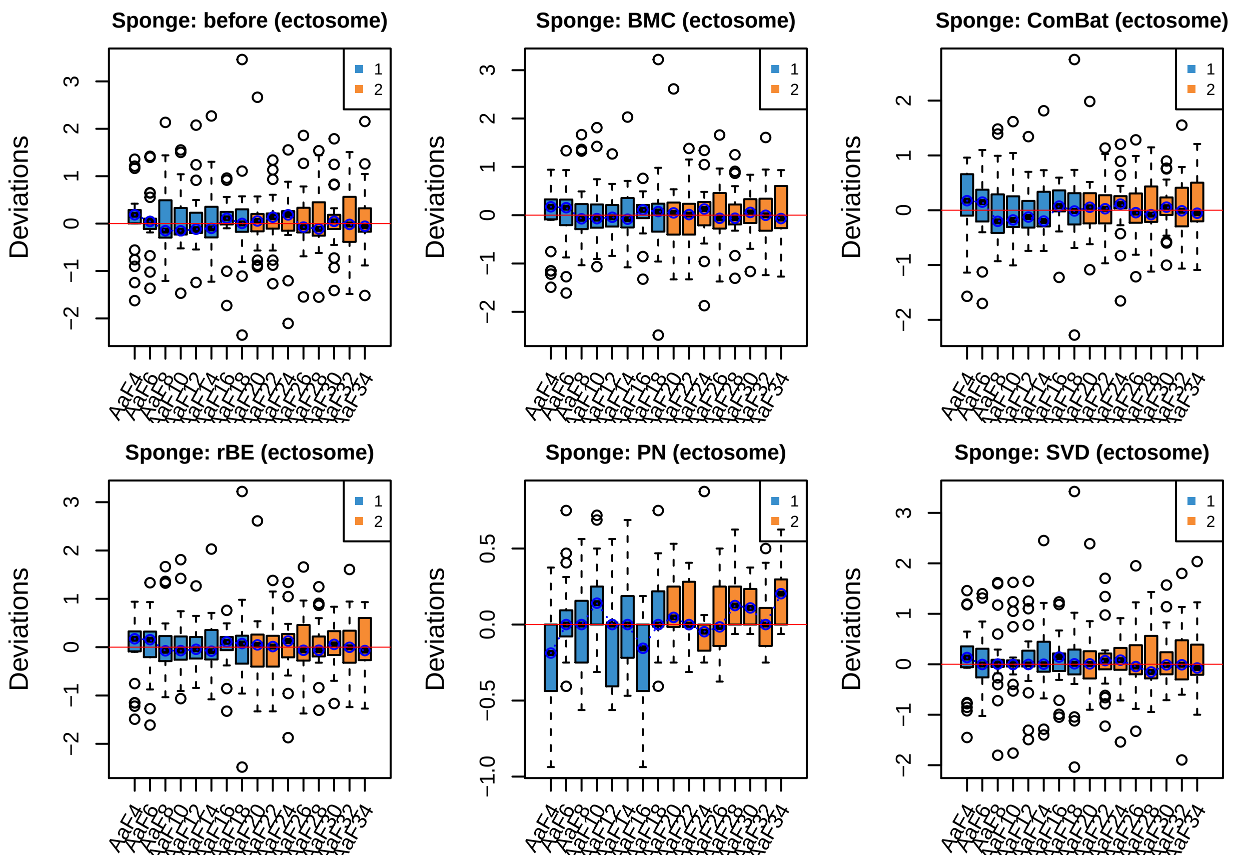
```r
par(mfrow = c(2,3), mai = c(0.4,0.6,0.3,0.1))

RleMicroRna2(object = t(sponge.before_c), batch = sponge.batch_c,
             maintitle = 'Sponge: before (choanosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.bmc_c), batch = sponge.batch_c,
             maintitle = 'Sponge: BMC (choanosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.combat_c), batch = sponge.batch_c,
             maintitle = 'Sponge: ComBat (choanosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.limma_c), batch = sponge.batch_c,
             maintitle = 'Sponge: rBE (choanosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.percentile_c), batch = sponge.batch_c,
             maintitle = 'Sponge: PN (choanosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.svd_c), batch = sponge.batch_c,
             maintitle = 'Sponge: SVD (choanosome)', title.cex = 1)
```



```r
par(mfrow = c(1,1))
```

In tissue choanosome of sponge data, the batch effect before correction is not obvious as all medians of samples are close to zero, but Gel2 has a greater interquartile range (IQR) than the other samples. Percentile normalisation increased batch effects, as the samples after correction are deviated from zero and the IQR is increased. Among other methods, the IQR of all the box plots (samples) from different batches is consistent after ComBat correction.

```r
par(mfrow = c(2,3), mai = c(0.4,0.6,0.3,0.1))

RleMicroRna2(object = t(sponge.before_e), batch = sponge.batch_e,
             maintitle = 'Sponge: before (ectosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.bmc_e), batch = sponge.batch_e,
             maintitle = 'Sponge: BMC (ectosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.combat_e), batch = sponge.batch_e,
             maintitle = 'Sponge: ComBat (ectosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.limma_e), batch = sponge.batch_e,
             maintitle = 'Sponge: rBE (ectosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.percentile_e), batch = sponge.batch_e,
             maintitle = 'Sponge: PN (ectosome)', title.cex = 1)

RleMicroRna2(object = t(sponge.svd_e), batch = sponge.batch_e,
             maintitle = 'Sponge: SVD (ectosome)', title.cex = 1)
```



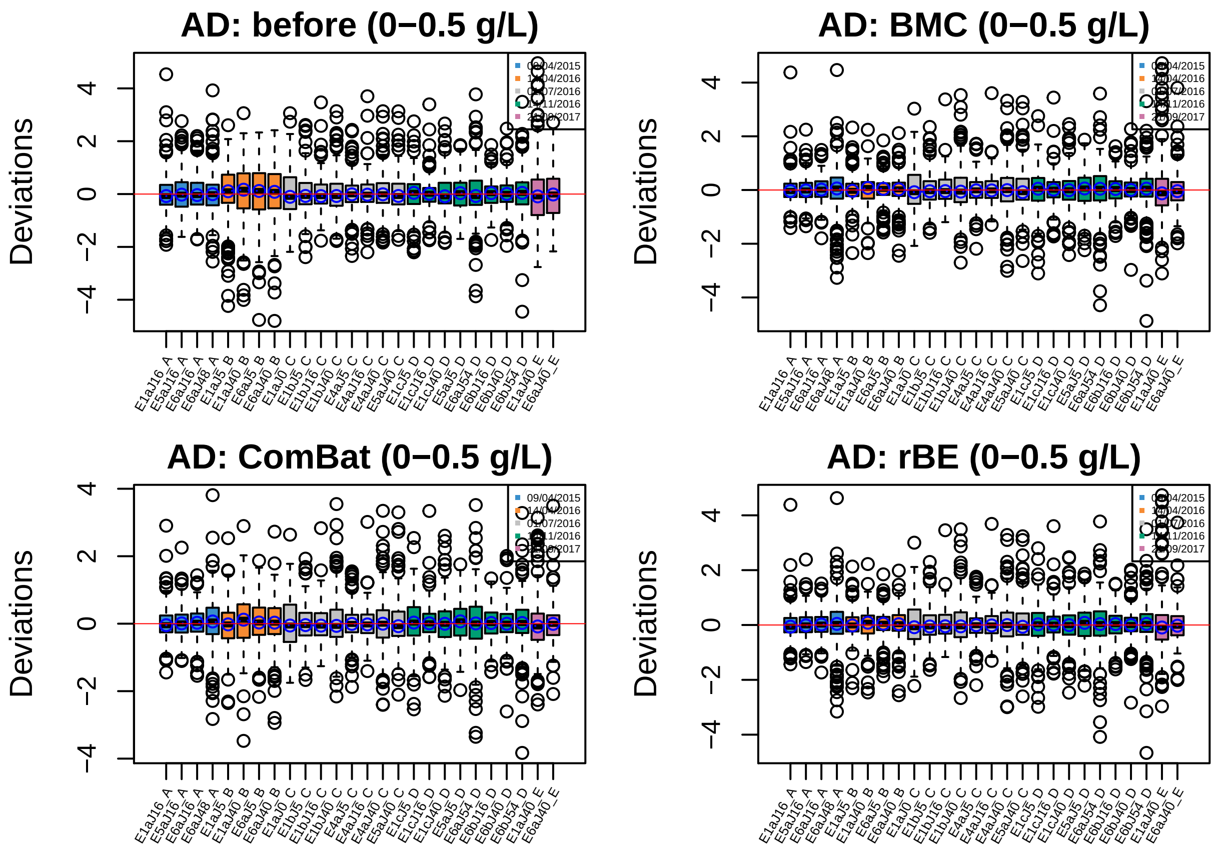
```r
par(mfrow = c(1,1))
```

In tissue ectosome of sponge data, batch effect is not easily detected, but percentile normalisation increased batch variation.

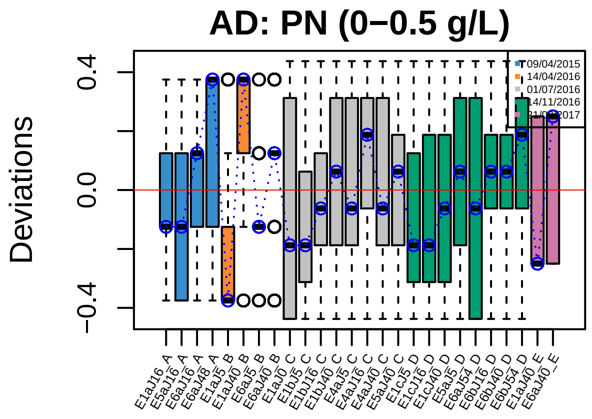
In AD data, we group the samples according to the inital phenol concentration (0-0.5 g/L / 1-2 g/L) and generate two RLE plots respectively in all datasets before and after batch correction, as sponge data:

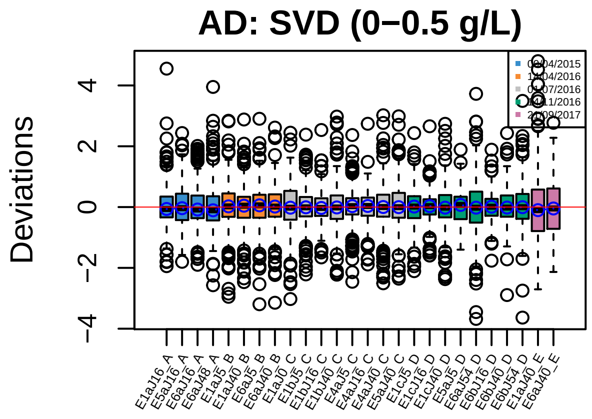
```r
# ad data
# BMC
ad.bmc_05 <- ad.bmc[ad.trt == '0-0.5', ]
ad.bmc_2 <- ad.bmc[ad.trt == '1-2', ]

# ComBat
ad.combat_05 <- ad.combat[ad.trt == '0-0.5', ]
ad.combat_2 <- ad.combat[ad.trt == '1-2', ]

# rBE
ad.limma_05 <- ad.limma[ad.trt == '0-0.5', ]
ad.limma_2 <- ad.limma[ad.trt == '1-2', ]

# PN
ad.percentile_05 <- ad.percentile[ad.trt == '0-0.5', ]
ad.percentile_2 <- ad.percentile[ad.trt == '1-2', ]

# SVD
ad.svd_05 <- ad.svd[ad.trt == '0-0.5', ]
ad.svd_2 <- ad.svd[ad.trt == '1-2', ]

# RUVIII
ad.ruv_05 <- ad.ruvIII[ad.trt == '0-0.5', ]
ad.ruv_2 <- ad.ruvIII[ad.trt == '1-2', ]
```

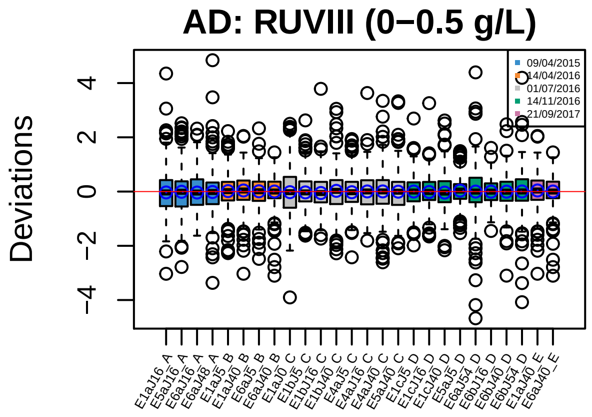
```r
par(mfrow = c(2,2), mai = c(0.5,0.8,0.3,0.1))

RleMicroRna2(object = t(ad.before_05), batch = ad.batch_05,
             maintitle = 'AD: before (0-0.5 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.5)

RleMicroRna2(object = t(ad.bmc_05), batch = ad.batch_05,
             maintitle = 'AD: BMC (0-0.5 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.5)

RleMicroRna2(object = t(ad.combat_05), batch = ad.batch_05,
             maintitle = 'AD: ComBat (0-0.5 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.5)

RleMicroRna2(object = t(ad.limma_05), batch = ad.batch_05,
             maintitle = 'AD: rBE (0-0.5 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.5)
```



```
RleMicroRna2(object = t(ad.percentile_05), batch = ad.batch_05,
             maintitle = 'AD: PN (0-0.5 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.5)

RleMicroRna2(object = t(ad.svd_05), batch = ad.batch_05,
             maintitle = 'AD: SVD (0-0.5 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.5)

RleMicroRna2(object = t(ad.ruv_05), batch = ad.batch_05,
             maintitle = 'AD: RUVIII (0-0.5 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.5)

par(mfrow = c(1,1))
```

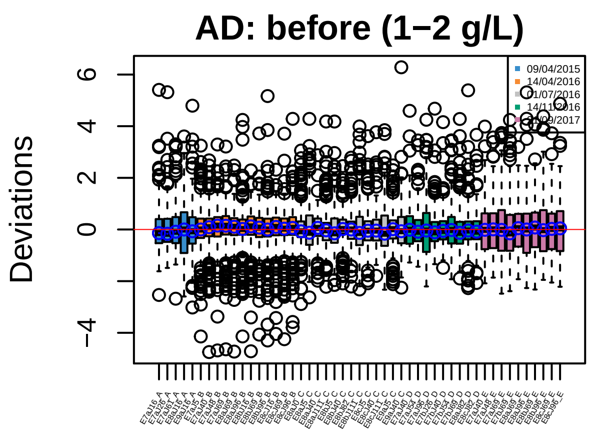


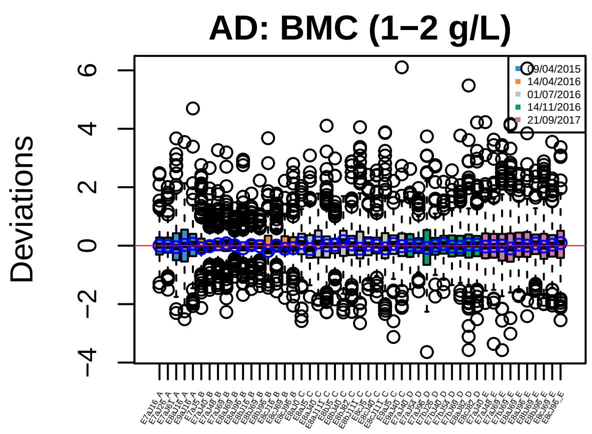


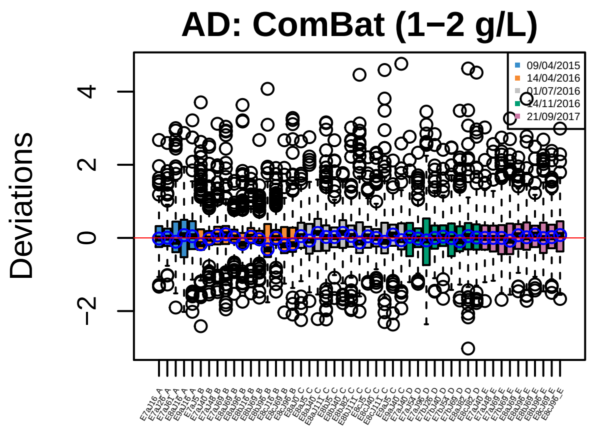


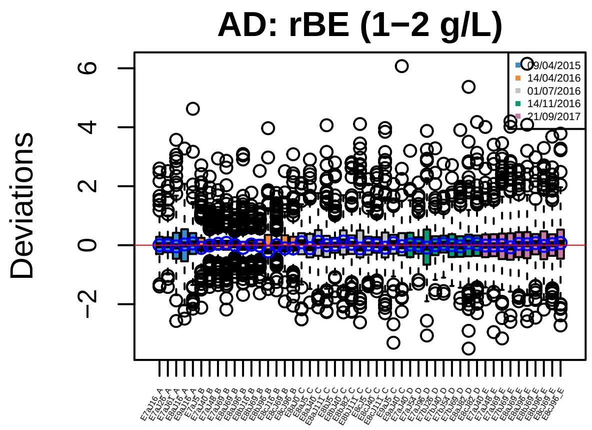
In AD data with initial phenol concentration between 0-0.5 g/L, batch effect is not easily detected, but percentile normalisation increased batch variation.

```
par(mfrow = c(2,2), mai = c(0.35,0.8,0.3,0.1))

RleMicroRna2(object = t(ad.before_2), batch = ad.batch_2,
             maintitle = 'AD: before (1-2 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.3)

RleMicroRna2(object = t(ad.bmc_2), batch = ad.batch_2,
             maintitle = 'AD: BMC (1-2 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.3)

RleMicroRna2(object = t(ad.combat_2), batch = ad.batch_2,
             maintitle = 'AD: ComBat (1-2 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.3)

RleMicroRna2(object = t(ad.limma_2), batch = ad.batch_2,
             maintitle = 'AD: rBE (1-2 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.3)
```

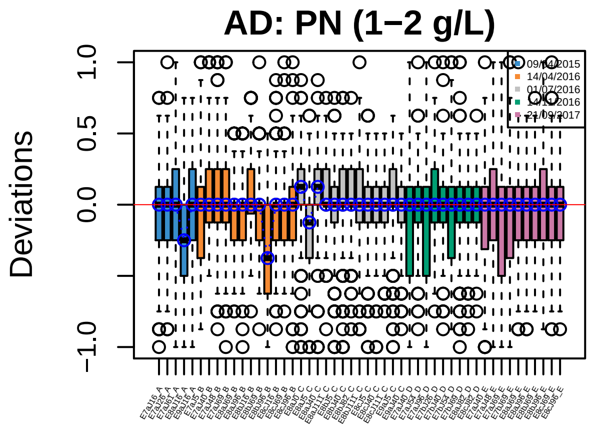


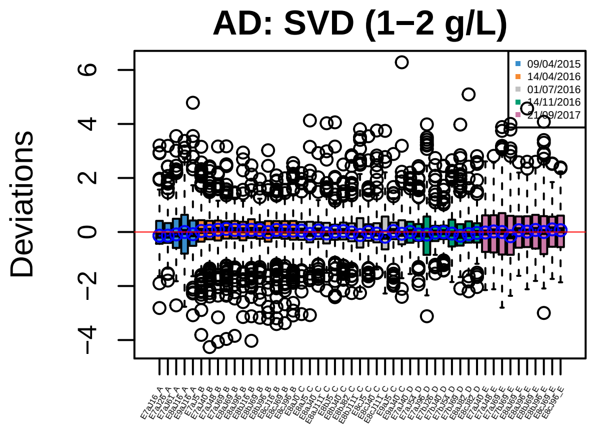


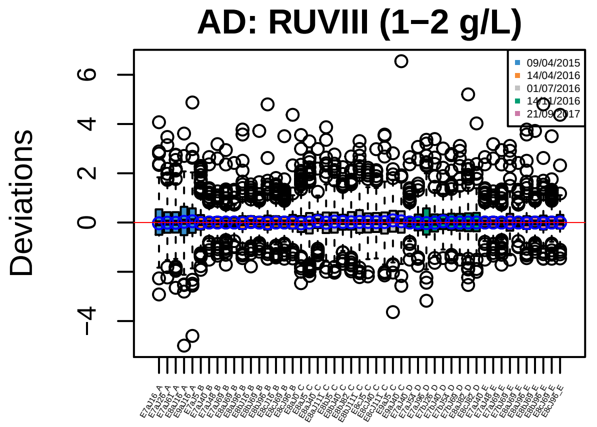




```
RleMicroRna2(object = t(ad.percentile_2), batch = ad.batch_2,
             maintitle = 'AD: PN (1-2 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.3)

RleMicroRna2(object = t(ad.svd_2), batch = ad.batch_2,
             maintitle = 'AD: SVD (1-2 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.3)

RleMicroRna2(object = t(ad.ruv_2), batch = ad.batch_2,
             maintitle = 'AD: RUVIII (1-2 g/L)', legend.cex = 0.4,
             cex.xaxis = 0.3)

par(mfrow = c(1,1))
```





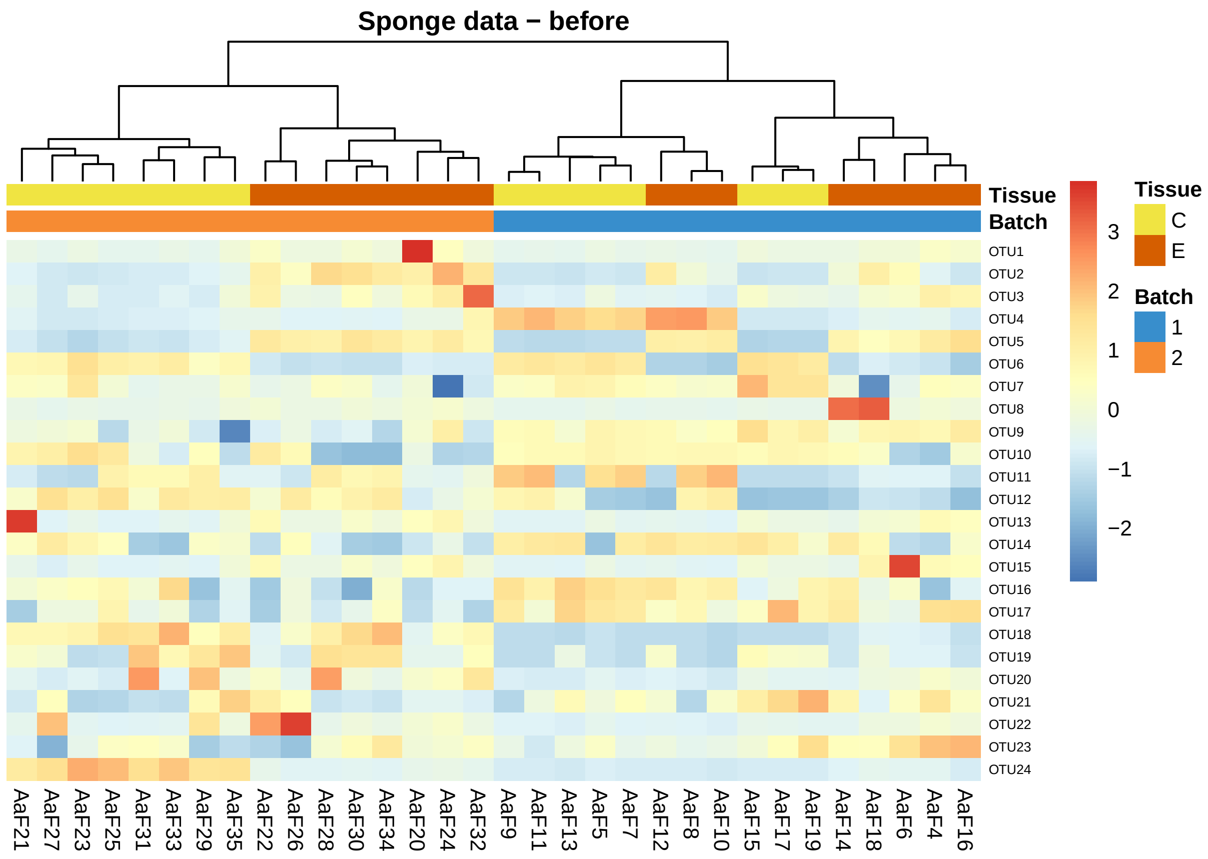


In AD data with initial phenol concentration between 1-2 g/L, batch effect is not easily detected, but percentile normalisation increased batch variation.

4.1.4 Heatmap

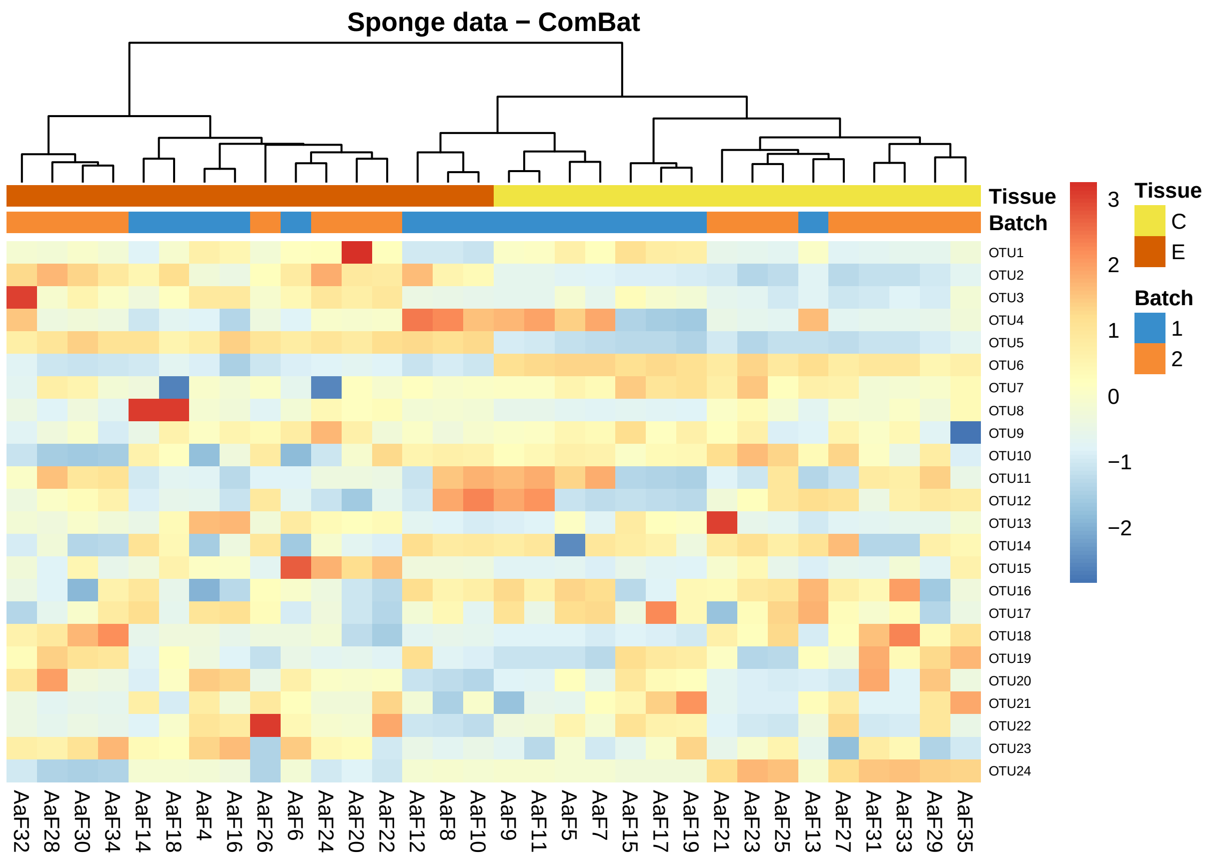
We use the heatmap to visualise data clusters. Samples clustered by batches instead of treatments indicate a batch effect.

```
# Sponge data
# before
sponge.tss.clr.scale <- scale(sponge.tss.clr, center = T, scale = T)
# scale on OTUs
sponge.tss.clr.scale <- scale(t(sponge.tss.clr.scale), center = T, scale = T)
# scale on samples

sponge.anno_col <- data.frame(Batch = sponge.batch, Tissue = sponge.trt)
sponge.anno_metabo_colors <- list(Batch = c('1' = '#388ECC', '2' = '#F68B33'),
                                  Tissue = c(C = '#F0E442', E = '#D55E00'))


pheatmap(sponge.tss.clr.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 5, fontsize_col = 8,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = sponge.anno_col,
         annotation_colors = sponge.anno_metabo_colors,
         border_color = 'NA',
         main = 'Sponge data - before')
```



Before correction, samples in sponge data are preferentially clustered by batch instead of tissue type, indicating a batch effect.

```
# ComBat
sponge.combat.scale <- scale(sponge.combat, center = T, scale = T)
# scale on OTUs
sponge.combat.scale <- scale(t(sponge.combat.scale), center = T, scale = T)
# scale on samples

pheatmap(sponge.combat.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 5, fontsize_col = 8,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = sponge.anno_col,
         annotation_colors = sponge.anno_metabo_colors,
         border_color = 'NA',
         main = 'Sponge data - ComBat')
```



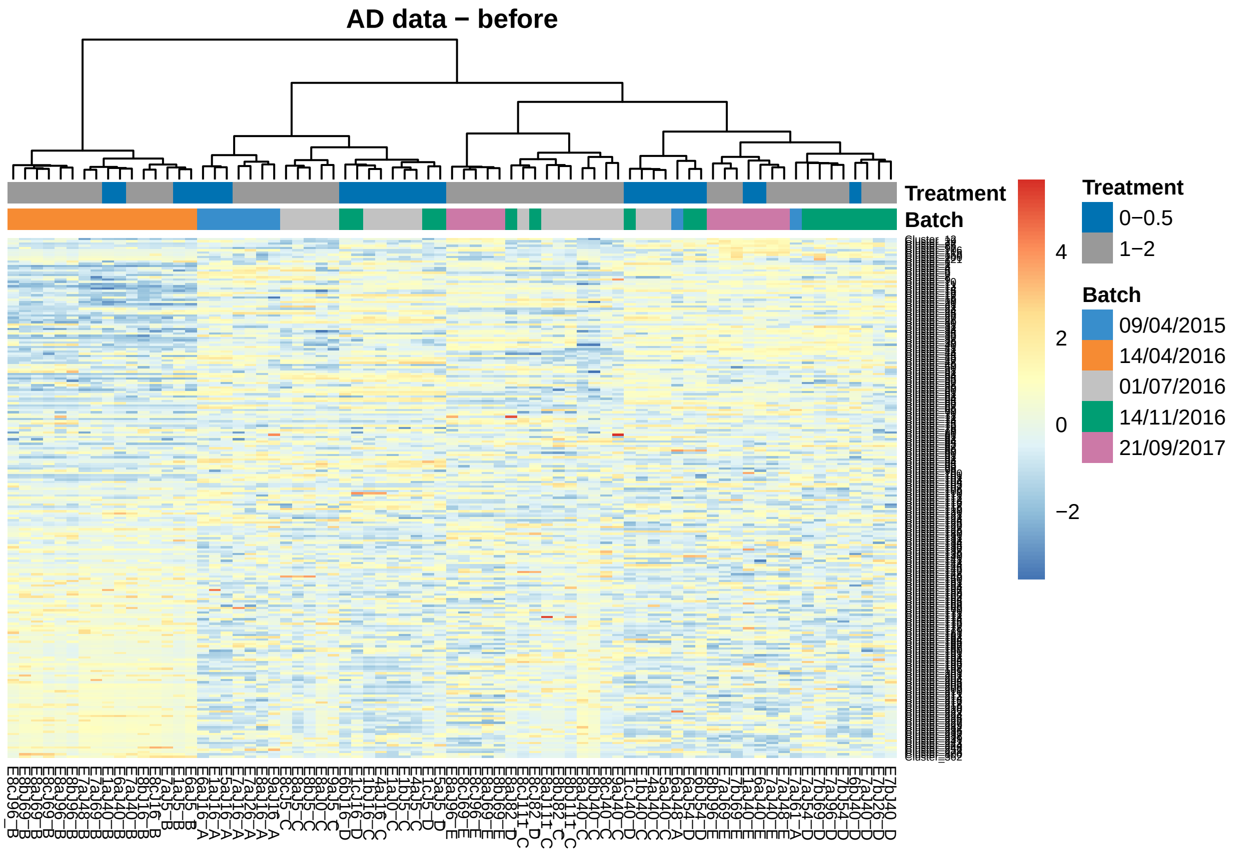
After batch correction with ComBat, the data are clustered by tissues rather than batches, therefore, ComBat not only removed batch effects, but also disengaged the effect of interest. The results from batch correction with other methods are similar.

```
# BMC
sponge.bmc.scale <- scale(sponge.bmc, center = T, scale = T)
# scale on OTUs
sponge.bmc.scale <- scale(t(sponge.bmc.scale), center = T, scale = T)
# scale on samples

pheatmap(sponge.bmc.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 5, fontsize_col = 8,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = sponge.anno_col,
         annotation_colors = sponge.anno_metabo_colors,
         border_color = 'NA',
         main = 'Sponge data - BMC')
```

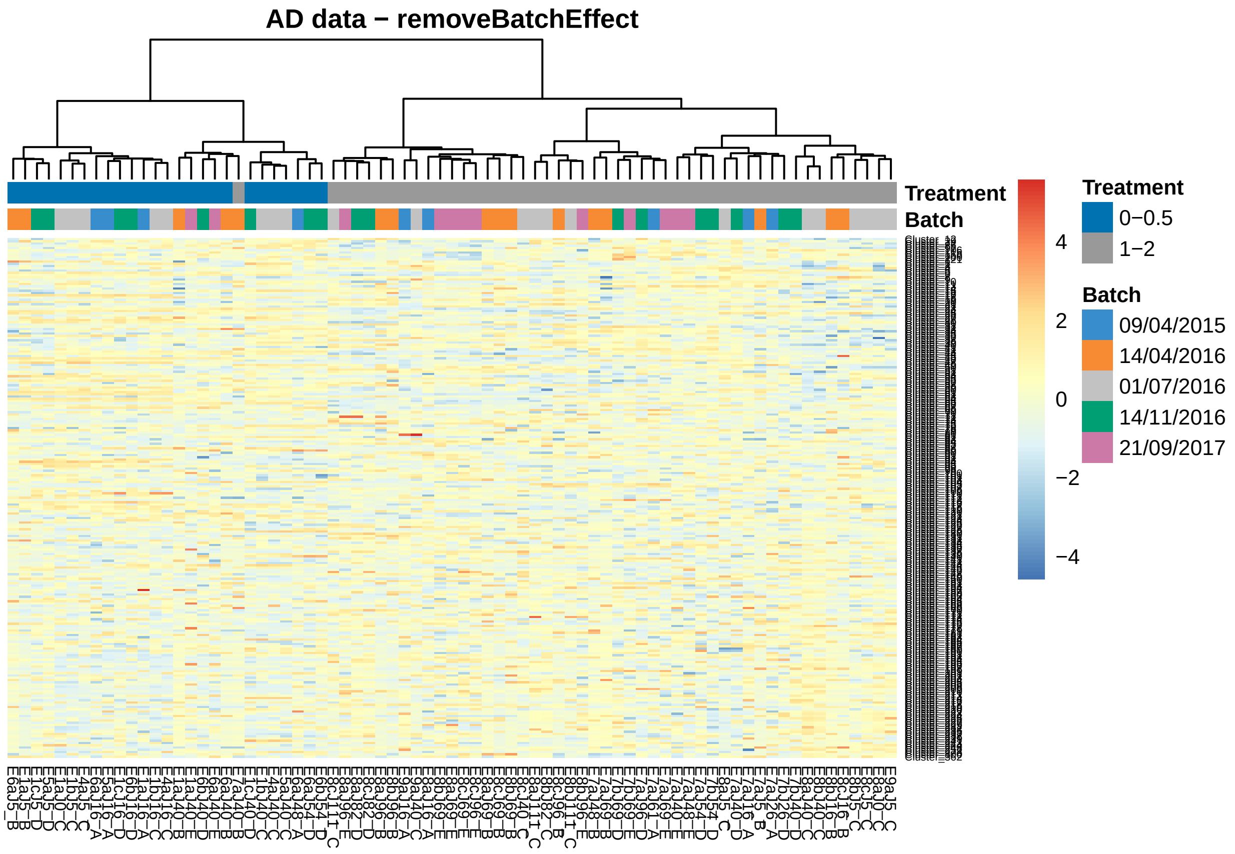
```r
# removeBatchEffect
sponge.limma.scale <- scale(sponge.limma, center = T, scale = T)
# scale on OTUs
sponge.limma.scale <- scale(t(sponge.limma.scale), center = T, scale = T)
# scale on samples

pheatmap(sponge.limma.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 5, fontsize_col = 8,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = sponge.anno_col,
         annotation_colors = sponge.anno_metabo_colors,
         border_color = 'NA',
         main = 'Sponge data - removeBatchEffect')

# percentile normalisation
sponge.percentile.scale <- scale(sponge.percentile, center = T, scale = T)
# scale on OTUs
sponge.percentile.scale <- scale(t(sponge.percentile.scale), center = T, scale = T)
# scale on samples

pheatmap(sponge.percentile.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 5, fontsize_col = 8,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = sponge.anno_col,
         annotation_colors = sponge.anno_metabo_colors,
         border_color = 'NA',
         main = 'Sponge data - percentile norm')


# SVD
sponge.svd.scale <- scale(sponge.svd, center = T, scale = T)
# scale on OTUs
sponge.svd.scale <- scale(t(sponge.svd.scale), center = T, scale = T)
# scale on samples

pheatmap(sponge.svd.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 5, fontsize_col = 8,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = sponge.anno_col,
         annotation_colors = sponge.anno_metabo_colors,
         border_color = 'NA',
         main = 'Sponge data - SVD')
```

In the heatmap of AD data before batch correction, samples within batch 14/04/2016 are clustered and distinct from other samples, indicating a batch effect.

```
# AD data
# before
ad.clr.scale <- scale(ad.clr, center = T, scale = T) # scale on OTUs
ad.clr.scale <- scale(t(ad.clr.scale), center = T, scale = T) # scale on samples

ad.anno_col <- data.frame(Batch = ad.batch, Treatment = ad.trt)
ad.anno_metabo_colors <- list(Batch = c('09/04/2015' = '#388ECC',
                                       '14/04/2016' = '#F68B33',
                                       '01/07/2016' = '#C2C2C2',
                                       '14/11/2016' = '#009E73',
                                       '21/09/2017' = '#CC79A7'),
                              Treatment = c('0-0.5' = '#0072B2', '1-2' = '#999999'))


pheatmap(ad.clr.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - before')
```



```r
# removeBatchEffect
ad.limma.scale <- scale(ad.limma, center = T, scale = T) # scale on OTUs
ad.limma.scale <- scale(t(ad.limma.scale), center = T, scale = T) # scale on samples

pheatmap(ad.limma.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - removeBatchEffect')
```



After batch correction with removeBatchEffect, the data are almost clustered by treatments rather than batches, therefore, removeBatchEffect not only removed batch effects, but also disengaged the treatment effects. The results from batch correction with other methods are similar.

```
# BMC
ad.bmc.scale <- scale(ad.bmc, center = T, scale = T) # scale on OTUs
ad.bmc.scale <- scale(t(ad.bmc.scale), center = T, scale = T) # scale on samples

pheatmap(ad.bmc.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - BMC')

# ComBat
ad.combat.scale <- scale(ad.combat, center = T, scale = T) # scale on OTUs
ad.combat.scale <- scale(t(ad.combat.scale), center = T, scale = T) # scale on samples

pheatmap(ad.combat.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - ComBat')



# percentile normalisation
ad.percentile.scale <- scale(ad.percentile, center = T, scale = T) # scale on OTUs
ad.percentile.scale <- scale(t(ad.percentile.scale), center = T, scale = T) # scale on samples

pheatmap(ad.percentile.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - percentile norm')

# SVD
ad.svd.scale <- scale(ad.svd, center = T, scale = T) # scale on OTUs
ad.svd.scale <- scale(t(ad.svd.scale), center = T, scale = T) # scale on samples

pheatmap(ad.svd.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - SVD')


# RUVIII
```

```r
ad.ruv.scale <- scale(ad.ruvIII, center = T, scale = T) # scale on OTUs
ad.ruv.scale <- scale(t(ad.ruv.scale), center = T, scale = T) # scale on samples

pheatmap(ad.ruv.scale,
         scale = 'none',
         cluster_rows = F,
         cluster_cols = T,
         fontsize_row = 4, fontsize_col = 6,
         fontsize = 8,
         clustering_distance_rows = 'euclidean',
         clustering_method = 'ward.D',
         treeheight_row = 30,
         annotation_col = ad.anno_col,
         annotation_colors = ad.anno_metabo_colors,
         border_color = 'NA',
         main = 'AD data - RUVIII')
```

4.2 Variance calculation

Compared to diagnostic plots, quantitative approaches are more objective and direct. The evaluation is done quantitatively before and after batch effect removal and the biological (treatment) effect must also be assessed before and after the batch effect correction to ensure it has been preserved.

4.2.1 Linear model per variable

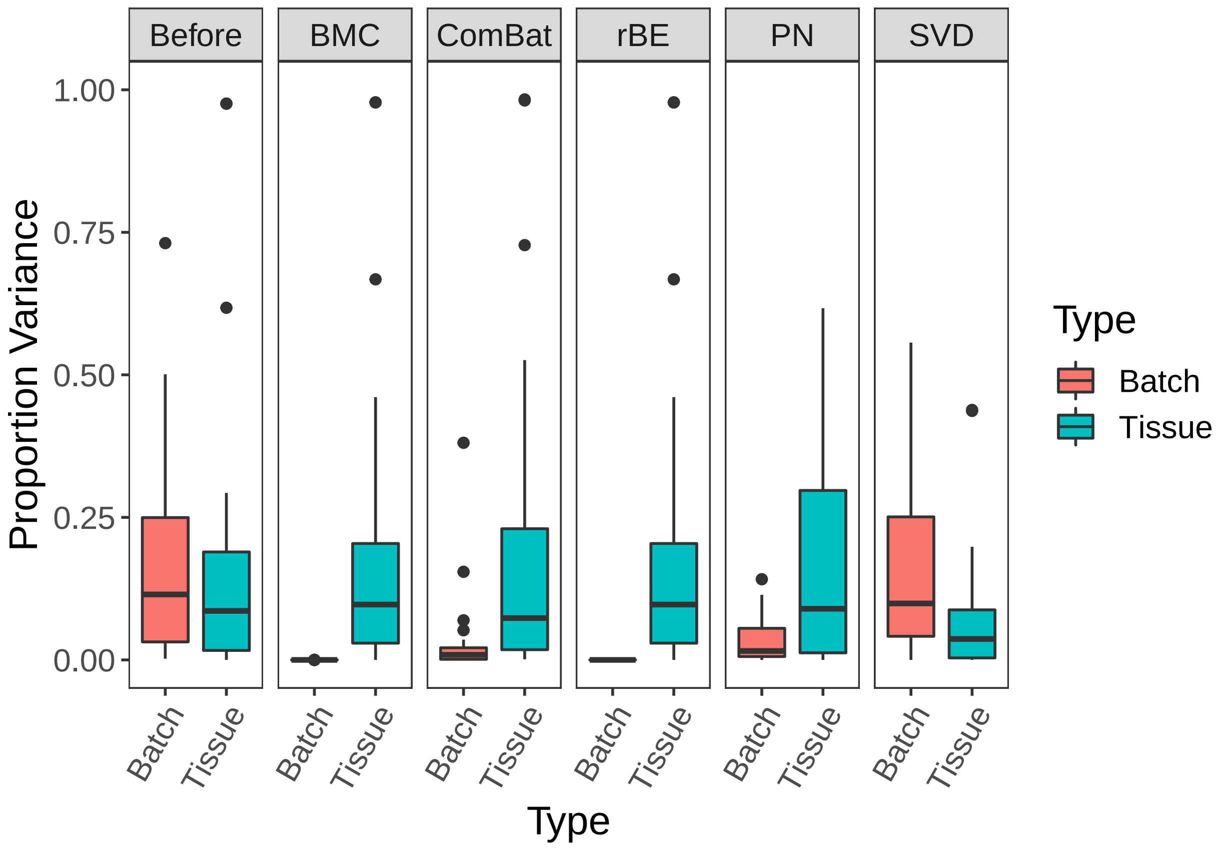
The variance explained by batch or treatment per OTU can be calculated by a linear model. Box plots can then be used to visualise the variance calculated from each OTU. Following batch effect correction, the percentage of variance explained by the treatment should be greater than the batch.

We use a function *fitExtractVarPartModel* to calculate and extract the variance of batch and treatment per OTU.

```r
# Sponge data
sponge.form <- ~ sponge.trt + sponge.batch
sponge.info <- as.data.frame(cbind(rownames(sponge.tss.clr), sponge.trt, sponge.batch))
rownames(sponge.info) <- rownames(sponge.tss.clr)

# before
sponge.varPart.before <- fitExtractVarPartModel(exprObj = t(sponge.tss.clr),
                                                formula = sponge.form,
                                                data = sponge.info)

# BMC
sponge.varPart.bmc <- fitExtractVarPartModel(exprObj = t(sponge.bmc),
                                             formula = sponge.form,
                                             data = sponge.info)

# combat
sponge.varPart.combat <- fitExtractVarPartModel(exprObj = t(sponge.combat),
                                                formula = sponge.form,
                                                data = sponge.info)

# removeBatchEffect
```

```
sponge.varPart.limma <- fitExtractVarPartModel(exprObj = t(sponge.limma),
                                               formula = sponge.form,
                                               data = sponge.info)

# percentile normalisation
sponge.varPart.percentile <- fitExtractVarPartModel(exprObj = t(sponge.percentile),
                                                    formula = sponge.form,
                                                    data = sponge.info)

# svd
sponge.varPart.svd <- fitExtractVarPartModel(exprObj = t(sponge.svd),
                                             formula = sponge.form,
                                             data = sponge.info)
```

```
# extract the variance of trt and batch
# before
sponge.varmat.before <- as.matrix(sponge.varPart.before[ ,1:2])
# BMC
sponge.varmat.bmc <- as.matrix(sponge.varPart.bmc[ ,1:2])
# ComBat
sponge.varmat.combat <- as.matrix(sponge.varPart.combat[ ,1:2])
# removeBatchEffect
sponge.varmat.limma <- as.matrix(sponge.varPart.limma[ ,1:2])
# percentile normalisation
sponge.varmat.percentile <- as.matrix(sponge.varPart.percentile[ ,1:2])
# SVD
sponge.varmat.svd <- as.matrix(sponge.varPart.svd[ ,1:2])

# merge results
sponge.variance <- c(as.vector(sponge.varmat.before), as.vector(sponge.varmat.bmc),
                     as.vector(sponge.varmat.combat), as.vector(sponge.varmat.limma),
                     as.vector(sponge.varmat.percentile), as.vector(sponge.varmat.svd))

# add batch, trt and methods info
sponge.variance <- cbind(variance = sponge.variance,
                         Type = rep(c('Tissue', 'Batch'), each = ncol(sponge.tss.clr)),
                         method = rep(c('Before', 'BMC', 'ComBat', 'rBE',
                                        'PN', 'SVD'), each = 2*ncol(sponge.tss.clr)))
# reorder levels
sponge.variance <- as.data.frame(sponge.variance)
sponge.variance$method <- factor(sponge.variance$method,
                                 levels = unique(sponge.variance$method))
sponge.variance$variance <- as.numeric(as.character(sponge.variance$variance))

ggplot(sponge.variance, aes(x = Type, y = variance, fill = Type)) +
  geom_boxplot() + facet_grid(cols = vars(method)) + theme_bw() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        strip.text = element_text(size = 12), panel.grid = element_blank(),
        axis.text = element_text(size = 12), axis.title = element_text(size = 15),
        legend.title = element_text(size = 15), legend.text = element_text(size = 12)) +
  labs(x = 'Type', y = 'Proportion Variance', name = 'Type') + ylim(0,1)
```



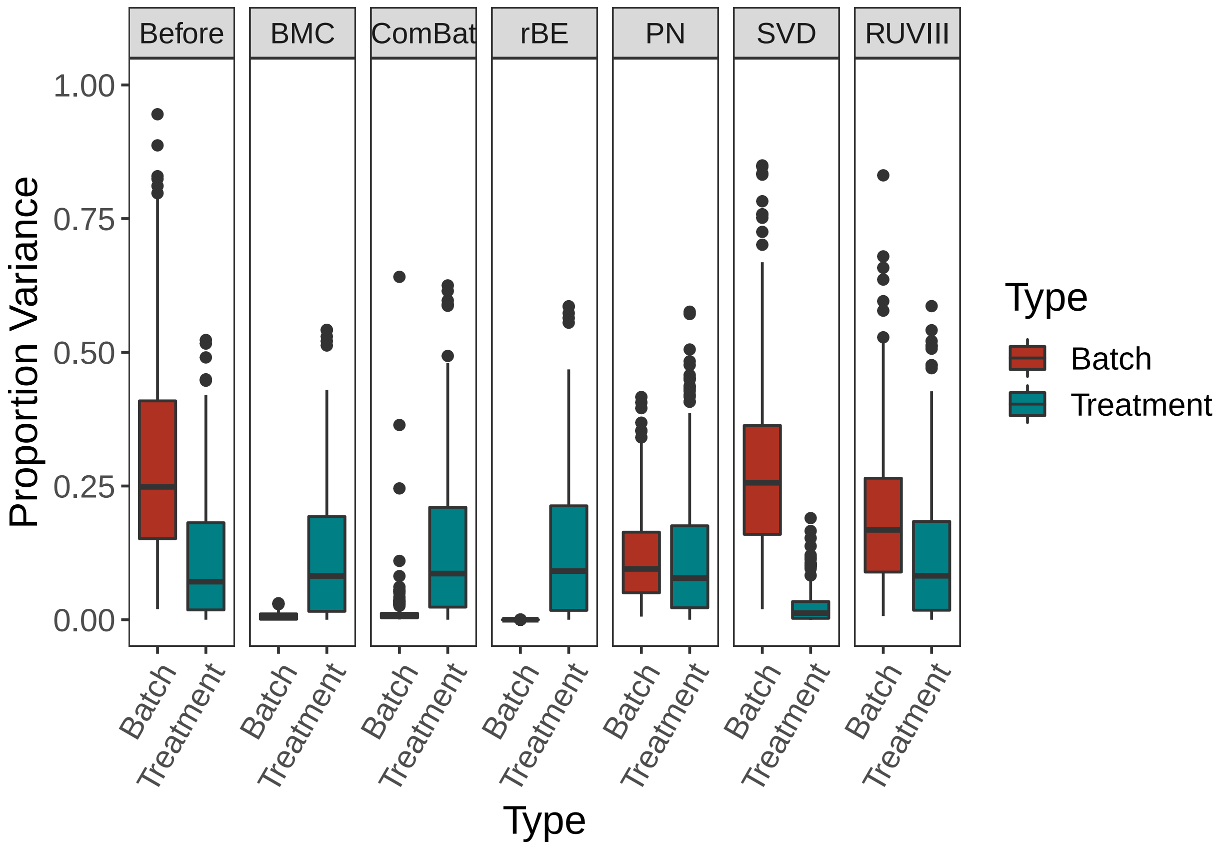
BMC, ComBat and removeBatchEffect successfully removed batch variation while preserving treatment variation, but SVD performed poorly. Percentile normalisation preserved sufficient treatment variation, but removed less batch effect variation than BMC, ComBat and removeBatchEffect. Percentile normalisation used percentiles instead of actual values. This leads to information loss. Moreover, it was designed for case-control microbiome studies, which differ from our example studies. This method would therefore be more effective with control samples in each batch. SVD was inefficient at targeting batch effects, as it assumes that batch effects cause the largest variation in the data and should therefore appear as the first component. However, in sponge data the batch effect is mainly present on the second component as was shown in PCA sample plots with density, which results in a miscorrection of batch effects.

We also calculate the variance for AD data.

```r
# AD data
ad.form <- ~ ad.trt + ad.batch
ad.info <- as.data.frame(cbind(rownames(ad.clr), ad.trt,ad.batch))
rownames(ad.info) <- rownames(ad.clr)

# before
ad.varPart.before <- fitExtractVarPartModel(exprObj = t(ad.clr),
                                            formula = ad.form, data = ad.info)

# BMC
ad.varPart.bmc <- fitExtractVarPartModel(exprObj = t(ad.bmc),
                                         formula = ad.form, data = ad.info)

# combat
ad.varPart.combat <- fitExtractVarPartModel(exprObj = t(ad.combat),
                                            formula = ad.form, data = ad.info)
```

```r
# removeBatchEffect
ad.varPart.limma <- fitExtractVarPartModel(exprObj = t(ad.limma),
                                           formula = ad.form, data = ad.info)

# percentile normalisation
ad.varPart.percentile <- fitExtractVarPartModel(exprObj = t(ad.percentile),
                                                formula = ad.form, data = ad.info)

# svd
ad.varPart.svd <- fitExtractVarPartModel(exprObj = t(ad.svd),
                                         formula = ad.form, data = ad.info)

# ruv
ad.varPart.ruv <- fitExtractVarPartModel(exprObj = t(ad.ruvIII),
                                         formula = ad.form, data = ad.info)
```

```r
# extract the variance of trt and batch
# before
ad.varmat.before <- as.matrix(ad.varPart.before[ ,1:2])
# BMC
ad.varmat.bmc <- as.matrix(ad.varPart.bmc[ ,1:2])
# ComBat
ad.varmat.combat <- as.matrix(ad.varPart.combat[ ,1:2])
# removeBatchEffect
ad.varmat.limma <- as.matrix(ad.varPart.limma[ ,1:2])
# percentile normalisation
ad.varmat.percentile <- as.matrix(ad.varPart.percentile[ ,1:2])
# SVD
ad.varmat.svd <- as.matrix(ad.varPart.svd[ ,1:2])
# RUVIII
ad.varmat.ruv <- as.matrix(ad.varPart.ruv[ ,1:2])


# merge results
ad.variance <- c(as.vector(ad.varmat.before), as.vector(ad.varmat.bmc),
                 as.vector(ad.varmat.combat), as.vector(ad.varmat.limma),
                 as.vector(ad.varmat.percentile), as.vector(ad.varmat.svd),
                 as.vector(ad.varmat.ruv))

# add batch, trt and methods info
ad.variance <- cbind(variance = ad.variance,
                     Type = rep(c( 'Treatment', 'Batch'), each = ncol(ad.clr)),
                     method = rep(c('Before', 'BMC', 'ComBat', 'rBE', 'PN',
                                    'SVD', 'RUVIII'), each = 2*ncol(ad.clr)))
# reorder levels
ad.variance <- as.data.frame(ad.variance)
ad.variance$method <- factor(ad.variance$method,
                            levels = unique(ad.variance$method))
ad.variance$variance <- as.numeric(as.character(ad.variance$variance))

ggplot(ad.variance, aes(x = Type, y = variance, fill = Type)) +
  geom_boxplot() + facet_grid(cols = vars(method)) + theme_bw() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        strip.text = element_text(size = 11), panel.grid = element_blank(),
        axis.text = element_text(size = 12), axis.title = element_text(size = 15),
        legend.title = element_text(size = 15), legend.text = element_text(size = 12)) +
  labs(x = "Type", y = "Proportion Variance", name = 'Type') +
  scale_fill_hue(l = 40) + ylim(0,1)
```



Similar results as in sponge data, BMC, ComBat and removeBatchEffect successfully removed batch variation while preserving treatment variation, but SVD performed poorly. Percentile normalisation and RUVIII preserved sufficient treatment variation, but removed less batch effect variation than BMC, ComBat and removeBatchEffect. It is possible the negative control variables or technical replicate samples did not entirely capture the batch effects. SVD was also inefficient at targeting batch effects, as in AD data the batch effect is present on both first and second components, which results in a miscorrection of batch effects.

4.2.2 RDA

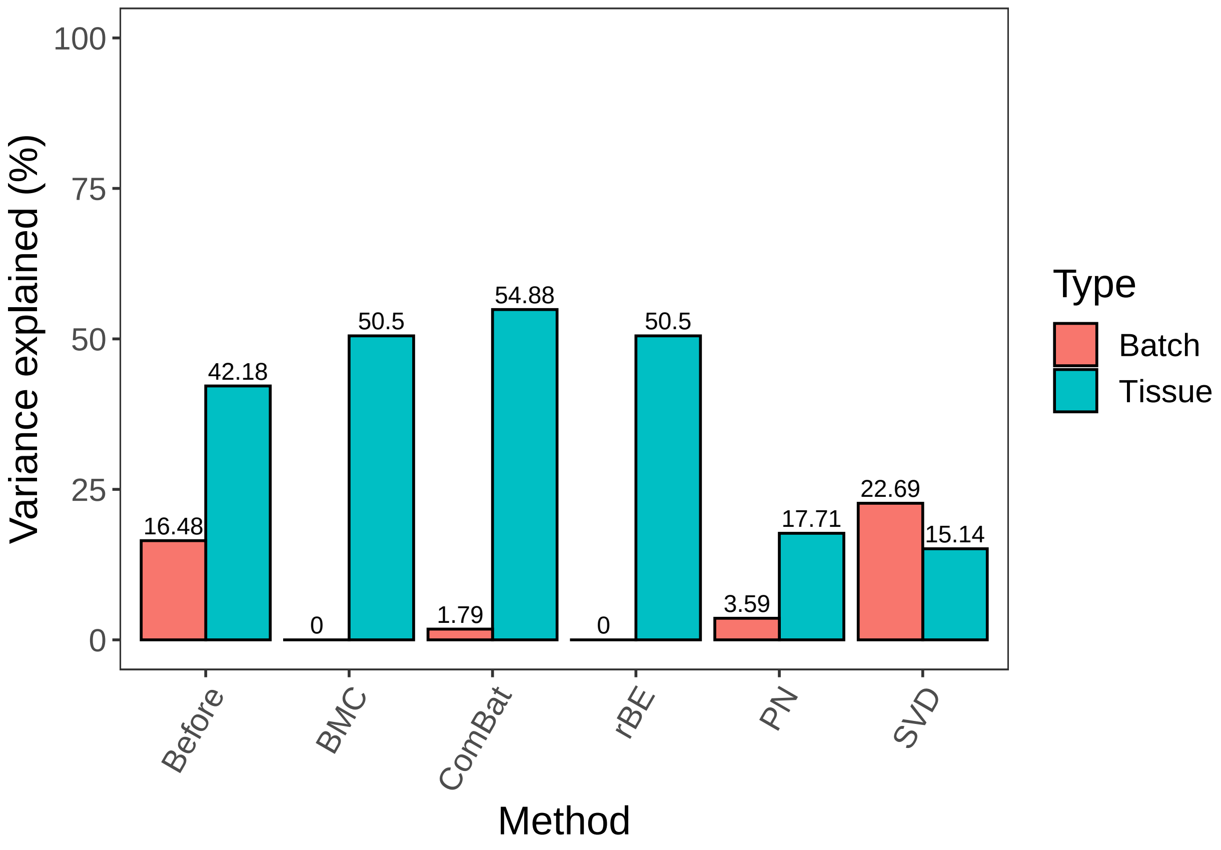
The pRDA method is a multivariate method to assess *globally* the effect of batch and treatment (two separate models).

```
# Sponge data
sponge.data.design <- numeric()
sponge.data.design$group <- sponge.trt
sponge.data.design$batch <- sponge.batch

# before
# conditioning on a batch effect
sponge.rda.before1 <- rda(sponge.tss.clr ~ group + Condition(batch),
                          data = sponge.data.design)
sponge.rda.before2 <- rda(sponge.tss.clr ~ batch + Condition(group),
                         data = sponge.data.design)

# amount of variance
sponge.rda.bat_prop.before <- sponge.rda.before1$pCCA$tot.chi*100/sponge.rda.before1$tot.chi
sponge.rda.trt_prop.before <- sponge.rda.before2$pCCA$tot.chi*100/sponge.rda.before2$tot.chi

# BMC
# conditioning on a batch effect
sponge.rda.bmc1 <- rda(sponge.bmc ~ group + Condition(batch),
                      data = sponge.data.design)
sponge.rda.bmc2 <- rda(sponge.bmc ~ batch + Condition(group),
                      data = sponge.data.design)

# amount of variance
sponge.rda.bat_prop.bmc <- sponge.rda.bmc1$pCCA$tot.chi*100/sponge.rda.bmc1$tot.chi
sponge.rda.trt_prop.bmc <- sponge.rda.bmc2$pCCA$tot.chi*100/sponge.rda.bmc2$tot.chi


# combat
# conditioning on a batch effect
sponge.rda.combat1 <- rda(sponge.combat ~ group + Condition(batch),
                         data = sponge.data.design)
sponge.rda.combat2 <- rda(sponge.combat ~ batch + Condition(group),
                         data = sponge.data.design)

# amount of variance
sponge.rda.bat_prop.combat <- sponge.rda.combat1$pCCA$tot.chi*100/sponge.rda.combat1$tot.chi
sponge.rda.trt_prop.combat <- sponge.rda.combat2$pCCA$tot.chi*100/sponge.rda.combat2$tot.chi


# limma
# conditioning on a batch effect
sponge.rda.limma1 <- rda(sponge.limma ~ group + Condition(batch),
                        data = sponge.data.design)
sponge.rda.limma2 <- rda(sponge.limma ~ batch + Condition(group),
                        data = sponge.data.design)

# amount of variance
sponge.rda.bat_prop.limma <- sponge.rda.limma1$pCCA$tot.chi*100/sponge.rda.limma1$tot.chi
sponge.rda.trt_prop.limma <- sponge.rda.limma2$pCCA$tot.chi*100/sponge.rda.limma2$tot.chi


# percentile
# conditioning on a batch effect
sponge.rda.percentile1 <- rda(sponge.percentile ~ group + Condition(batch),
                             data = sponge.data.design)
sponge.rda.percentile2 <- rda(sponge.percentile ~ batch + Condition(group),
                             data = sponge.data.design)

# amount of variance
sponge.rda.bat_prop.percentile <- sponge.rda.percentile1$pCCA$tot.chi*100/sponge.rda.percentile1$tot.ch
sponge.rda.trt_prop.percentile <- sponge.rda.percentile2$pCCA$tot.chi*100/sponge.rda.percentile2$tot.ch
```

```r
# SVD
# conditioning on a batch effect
sponge.rda.svd1 <- rda(sponge.svd ~ group + Condition(batch),
                      data = sponge.data.design)
sponge.rda.svd2 <- rda(sponge.svd ~ batch + Condition(group),
                      data = sponge.data.design)

# amount of variance
sponge.rda.bat_prop.svd <- sponge.rda.svd1$pCCA$tot.chi*100/sponge.rda.svd1$tot.chi
sponge.rda.trt_prop.svd <- sponge.rda.svd2$pCCA$tot.chi*100/sponge.rda.svd2$tot.chi
```

We now represent the amount of variance explained by batch and treatment estimated with pRDA:

```r
# proportion
sponge.rda.prop.before <- c(sponge.rda.bat_prop.before,
                           sponge.rda.trt_prop.before)
sponge.rda.prop.bmc <- c(sponge.rda.bat_prop.bmc,
                        sponge.rda.trt_prop.bmc)
sponge.rda.prop.combat <- c(sponge.rda.bat_prop.combat,
                           sponge.rda.trt_prop.combat)
sponge.rda.prop.limma <- c(sponge.rda.bat_prop.limma,
                          sponge.rda.trt_prop.limma)
sponge.rda.prop.percentile <- c(sponge.rda.bat_prop.percentile,
                               sponge.rda.trt_prop.percentile)
sponge.rda.prop.svd <- c(sponge.rda.bat_prop.svd,
                        sponge.rda.trt_prop.svd)

# merge results
sponge.rda.prop.val <- c(sponge.rda.prop.before, sponge.rda.prop.bmc,
                        sponge.rda.prop.combat, sponge.rda.prop.limma,
                        sponge.rda.prop.percentile, sponge.rda.prop.svd)

# add batch, trt and method info
sponge.rda.prop <- data.frame(prop = sponge.rda.prop.val,
                             prop.r = round(sponge.rda.prop.val, 2),
                             Method = rep(c('Before', 'BMC', 'ComBat',
                                            'rBE', 'PN', 'SVD'), each = 2),
                             Type = rep(c('Batch', 'Tissue'), 6))

# reorder levels
sponge.rda.prop$Method <- factor(sponge.rda.prop$Method,
                                levels = unique(sponge.rda.prop$Method))

ggplot(data = sponge.rda.prop, aes(x = Method, y = prop, fill = Type)) +
  geom_bar(stat = "identity", position = 'dodge', colour = 'black') +
  geom_text(data = sponge.rda.prop, aes(Method, prop + 2.5, label = prop.r),
           position = position_dodge(width = 0.9), size = 3) + theme_bw() +
  labs(y = "Variance explained (%)") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        panel.grid = element_blank(), axis.text = element_text(size = 12),
        axis.title = element_text(size = 15), legend.title = element_text(size = 15),
        legend.text = element_text(size = 12)) + ylim(0,100)
```



In sponge data, pRDA shows that BMC, ComBat and removeBatchEffect were more efficient at removing batch variation while preserving treatment variation. This result is in agreement with the proportional variance calculated using a linear model in the previous section 'linear model per variable'. ComBat removed relatively less batch variation compared with the other two methods, which implies that the batch effect is not purely systematic. The low efficiency of percentile normalisation is more obvious in the variance calculated with pRDA. It did not remove enough batch variation, nor preserve enough treatment variation in sponge data.

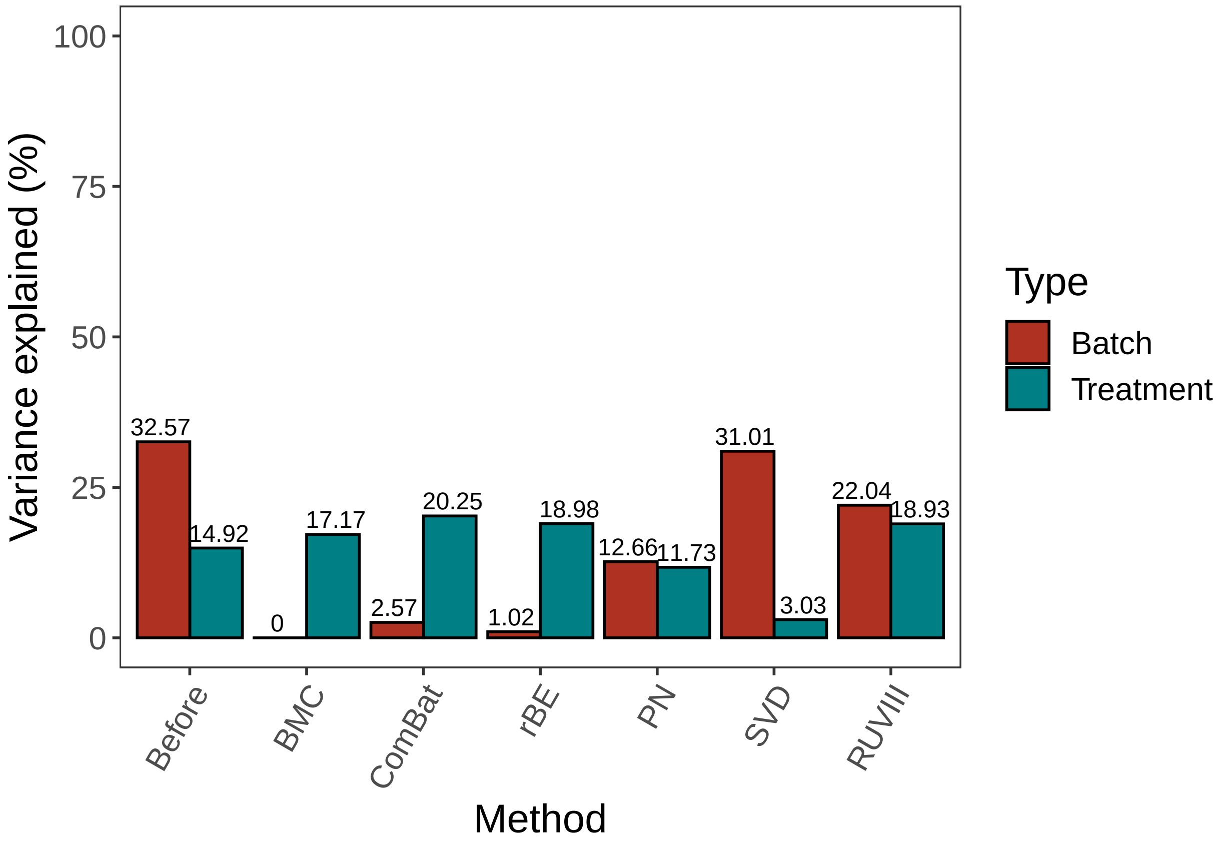
We also apply pRDA on AD data.

```r
# AD data
ad.data.design <- numeric()
ad.data.design$group <- ad.trt
ad.data.design$batch <- ad.batch

# before
# conditioning on a batch effect
ad.rda.before1 <- rda(ad.clr ~ group + Condition(batch), data = ad.data.design)
ad.rda.before2 <- rda(ad.clr ~ batch + Condition(group), data = ad.data.design)

# amount of variance
ad.rda.bat_prop.before <- ad.rda.before1$pCCA$tot.chi*100/ad.rda.before1$tot.chi
ad.rda.trt_prop.before <- ad.rda.before2$pCCA$tot.chi*100/ad.rda.before2$tot.chi

# BMC
# conditioning on a batch effect
ad.rda.bmc1 <- rda(ad.bmc ~ group + Condition(batch), data = ad.data.design)
ad.rda.bmc2 <- rda(ad.bmc ~ batch + Condition(group), data = ad.data.design)
```

```
# amount of variance
ad.rda.bat_prop.bmc <- ad.rda.bmc1$pCCA$tot.chi*100/ad.rda.bmc1$tot.chi
ad.rda.trt_prop.bmc <- ad.rda.bmc2$pCCA$tot.chi*100/ad.rda.bmc2$tot.chi


# combat
# conditioning on a batch effect
ad.rda.combat1 <- rda(ad.combat ~ group + Condition(batch), data = ad.data.design)
ad.rda.combat2 <- rda(ad.combat ~ batch + Condition(group), data = ad.data.design)

# amount of variance
ad.rda.bat_prop.combat <- ad.rda.combat1$pCCA$tot.chi*100/ad.rda.combat1$tot.chi
ad.rda.trt_prop.combat <- ad.rda.combat2$pCCA$tot.chi*100/ad.rda.combat2$tot.chi


# limma
# conditioning on a batch effect
ad.rda.limma1 <- rda(ad.limma ~ group + Condition(batch), data = ad.data.design)
ad.rda.limma2 <- rda(ad.limma ~ batch + Condition(group), data = ad.data.design)

# amount of variance
ad.rda.bat_prop.limma <- ad.rda.limma1$pCCA$tot.chi*100/ad.rda.limma1$tot.chi
ad.rda.trt_prop.limma <- ad.rda.limma2$pCCA$tot.chi*100/ad.rda.limma2$tot.chi


# percentile
# conditioning on a batch effect
ad.rda.percentile1 <- rda(ad.percentile ~ group + Condition(batch),
                          data = ad.data.design)
ad.rda.percentile2 <- rda(ad.percentile ~ batch + Condition(group),
                          data = ad.data.design)

# amount of variance
ad.rda.bat_prop.percentile <- ad.rda.percentile1$pCCA$tot.chi*100/ad.rda.percentile1$tot.chi
ad.rda.trt_prop.percentile <- ad.rda.percentile2$pCCA$tot.chi*100/ad.rda.percentile2$tot.chi


# SVD
# conditioning on a batch effect
ad.rda.svd1 <- rda(ad.svd ~ group + Condition(batch), data = ad.data.design)
ad.rda.svd2 <- rda(ad.svd ~ batch + Condition(group), data = ad.data.design)

# amount of variance
ad.rda.bat_prop.svd <- ad.rda.svd1$pCCA$tot.chi*100/ad.rda.svd1$tot.chi
ad.rda.trt_prop.svd <- ad.rda.svd2$pCCA$tot.chi*100/ad.rda.svd2$tot.chi


# RUVIII
# conditioning on a batch effect
ad.rda.ruv1 <- rda(ad.ruvIII ~ group + Condition(batch), data = ad.data.design)
ad.rda.ruv2 <- rda(ad.ruvIII ~ batch + Condition(group), data = ad.data.design)

# amount of variance
```

```
ad.rda.bat_prop.ruv <- ad.rda.ruv1$pCCA$tot.chi*100/ad.rda.ruv1$tot.chi
ad.rda.trt_prop.ruv <- ad.rda.ruv2$pCCA$tot.chi*100/ad.rda.ruv2$tot.chi
```

```
# proportion
ad.rda.prop.before <- c(ad.rda.bat_prop.before, ad.rda.trt_prop.before)
ad.rda.prop.bmc <- c(ad.rda.bat_prop.bmc, ad.rda.trt_prop.bmc)
ad.rda.prop.combat <- c(ad.rda.bat_prop.combat, ad.rda.trt_prop.combat)
ad.rda.prop.limma <- c(ad.rda.bat_prop.limma, ad.rda.trt_prop.limma)
ad.rda.prop.percentile <- c(ad.rda.bat_prop.percentile, ad.rda.trt_prop.percentile)
ad.rda.prop.svd <- c(ad.rda.bat_prop.svd, ad.rda.trt_prop.svd)
ad.rda.prop.ruv <- c(ad.rda.bat_prop.ruv, ad.rda.trt_prop.ruv)

# merge results
ad.rda.prop.val <- c(ad.rda.prop.before, ad.rda.prop.bmc,
                    ad.rda.prop.combat, ad.rda.prop.limma,
                    ad.rda.prop.percentile, ad.rda.prop.svd,
                    ad.rda.prop.ruv)

# add batch, trt and method info
ad.rda.prop <- data.frame(prop = ad.rda.prop.val, prop.r = round(ad.rda.prop.val, 2),
                         Method = rep(c('Before', 'BMC', 'ComBat', 'rBE',
                                        'PN', 'SVD', 'RUVIII'), each = 2),
                         Type = rep(c('Batch', 'Treatment'), 7))

# reorder levels
ad.rda.prop$Method <- factor(ad.rda.prop$Method, levels = unique(ad.rda.prop$Method))

ggplot(data = ad.rda.prop, aes(x = Method, y = prop, fill = Type)) +
  geom_bar(stat = "identity", position = 'dodge', colour = 'black') +
  geom_text(data = ad.rda.prop, aes(Method, prop + 2.5, label = prop.r),
            position = position_dodge(width = 1), size = 3) + theme_bw() +
  labs(y = "Variance explained (%)") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        panel.grid = element_blank(), axis.text = element_text(size = 12),
        axis.title = element_text(size = 15), legend.title = element_text(size = 15),
        legend.text = element_text(size = 12)) + scale_fill_hue(l = 40) + ylim(0,100)
```



Similar as results from sponge data, BMC, ComBat and removeBatchEffect were more efficient at removing batch variation while preserving treatment variation. ComBat removed relatively less batch variation compared with the other two methods. Percentile normalisation did not remove enough batch variation, nor preserve enough treatment variation in AD data. RUVIII preserved enough treatment variation but did not remove enough batch variation.

In sponge data, the results of BMC and removeBatchEffect are the same, while in AD data, removeBatchEffect removed less batch variation but preserved more treatment variation. This indicates some linear correlation exists between the batch and treatment effects, which might originate from the batch x treatment design.

4.2.3 PVCA

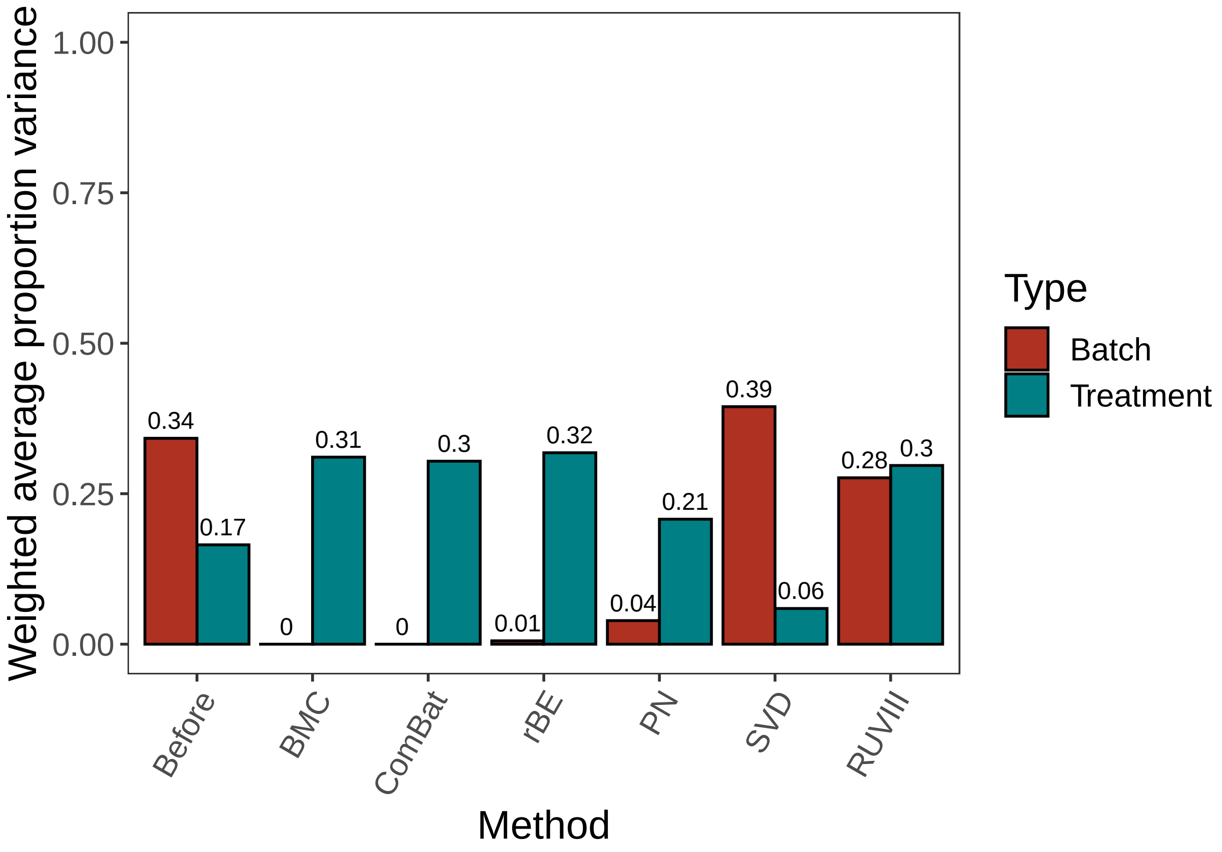
PVCA can be applied as a validation method that complements pRDA, but it requires 42 OTUs minimum. Therefore, it did not apply on sponge data.

```r
# AD data
ad.PVCA.score <- data.frame(Interaction = NA, Batch = NA,
                            Treatment = NA, Residuals = NA)

ad.Bat_Int.factors <- data.frame(Batch = ad.batch, Treatment = ad.trt)
rownames(ad.Bat_Int.factors) <- rownames(ad.clr)
pdata <- AnnotatedDataFrame(ad.Bat_Int.factors)

# before
ad.eset.X.before <- new("ExpressionSet", exprs = t(ad.clr), phenoData = pdata)
ad.pvcaObj.before <- pvcaBatchAssess(ad.eset.X.before, c('Batch', 'Treatment'), 0.6)
ad.values.before <- ad.pvcaObj.before$dat
ad.PVCA.score[1, ] <- ad.values.before
```

```
# bmc
ad.eset.X.bmc <- new("ExpressionSet", exprs = t(ad.bmc), phenoData = pdata)
ad.pvcaObj.bmc <- pvcaBatchAssess(ad.eset.X.bmc, c('Batch', 'Treatment'), 0.6)
ad.values.bmc <- ad.pvcaObj.bmc$dat
ad.PVCA.score[2, ] <- ad.values.bmc


# combat

ad.eset.X.combat <- new("ExpressionSet", exprs = t(ad.combat), phenoData = pdata)
ad.pvcaObj.combat <- pvcaBatchAssess(ad.eset.X.combat, c('Batch', 'Treatment'), 0.6)
ad.values.combat <- ad.pvcaObj.combat$dat
ad.PVCA.score[3, ] <- ad.values.combat

# limma

ad.eset.X.limma <- new("ExpressionSet", exprs = t(ad.limma), phenoData = pdata)
ad.pvcaObj.limma <- pvcaBatchAssess(ad.eset.X.limma, c('Batch', 'Treatment'), 0.6)
ad.values.limma <- ad.pvcaObj.limma$dat
ad.PVCA.score[4, ] <- ad.values.limma


# PN

ad.eset.X.percentile <- new("ExpressionSet", exprs = t(ad.percentile),
                            phenoData = pdata)
ad.pvcaObj.percentile <- pvcaBatchAssess(ad.eset.X.percentile,
                                         c('Batch', 'Treatment'), 0.6)
ad.values.percentile <- ad.pvcaObj.percentile$dat
ad.PVCA.score[5, ] <- ad.values.percentile


# svd

ad.eset.X.svd <- new("ExpressionSet", exprs = t(ad.svd), phenoData = pdata)
ad.pvcaObj.svd <- pvcaBatchAssess(ad.eset.X.svd, c('Batch', 'Treatment'), 0.6)
ad.values.svd <- ad.pvcaObj.svd$dat
ad.PVCA.score[6, ] <- ad.values.svd


# RUVIII

ad.eset.X.ruv <- new("ExpressionSet", exprs = t(ad.ruvIII), phenoData = pdata)
ad.pvcaObj.ruv <- pvcaBatchAssess(ad.eset.X.ruv, c('Batch', 'Treatment'), 0.6)
ad.values.ruv <- ad.pvcaObj.ruv$dat
ad.PVCA.score[7, ] <- ad.values.ruv

rownames(ad.PVCA.score) <- c('Before', 'BMC', 'ComBat', 'rBE', 'PN', 'SVD', 'RUVIII')
```

```
# merge results
ad.pvca.prop.val <- c(ad.PVCA.score$Batch, ad.PVCA.score$Treatment)

# add batch, trt and method info
```

```r
ad.pvca.prop <- data.frame(prop = ad.pvca.prop.val, prop.r = round(ad.pvca.prop.val, 2),
                          Method = rep(c('Before', 'BMC', 'ComBat', 'rBE',
                                         'PN', 'SVD', 'RUVIII'), 2),
                          Type = rep(c('Batch', 'Treatment'), each = 7))

# reorder levels
ad.pvca.prop$Method <- factor(ad.pvca.prop$Method, levels = unique(ad.pvca.prop$Method))

ggplot(data = ad.pvca.prop, aes(x = Method, y = prop, fill = Type)) +
  geom_bar(stat = "identity", position = 'dodge', colour = 'black') +
  geom_text(data = ad.pvca.prop, aes(Method, prop + 0.03, label = prop.r),
            position = position_dodge(width = 0.9), size = 3) + theme_bw() +
  labs(y = "Weighted average proportion variance") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        panel.grid = element_blank(),axis.text = element_text(size = 12),
        axis.title = element_text(size = 15), legend.title = element_text(size = 15),
        legend.text = element_text(size = 12)) + scale_fill_hue(l = 40) + ylim(0,1)
```



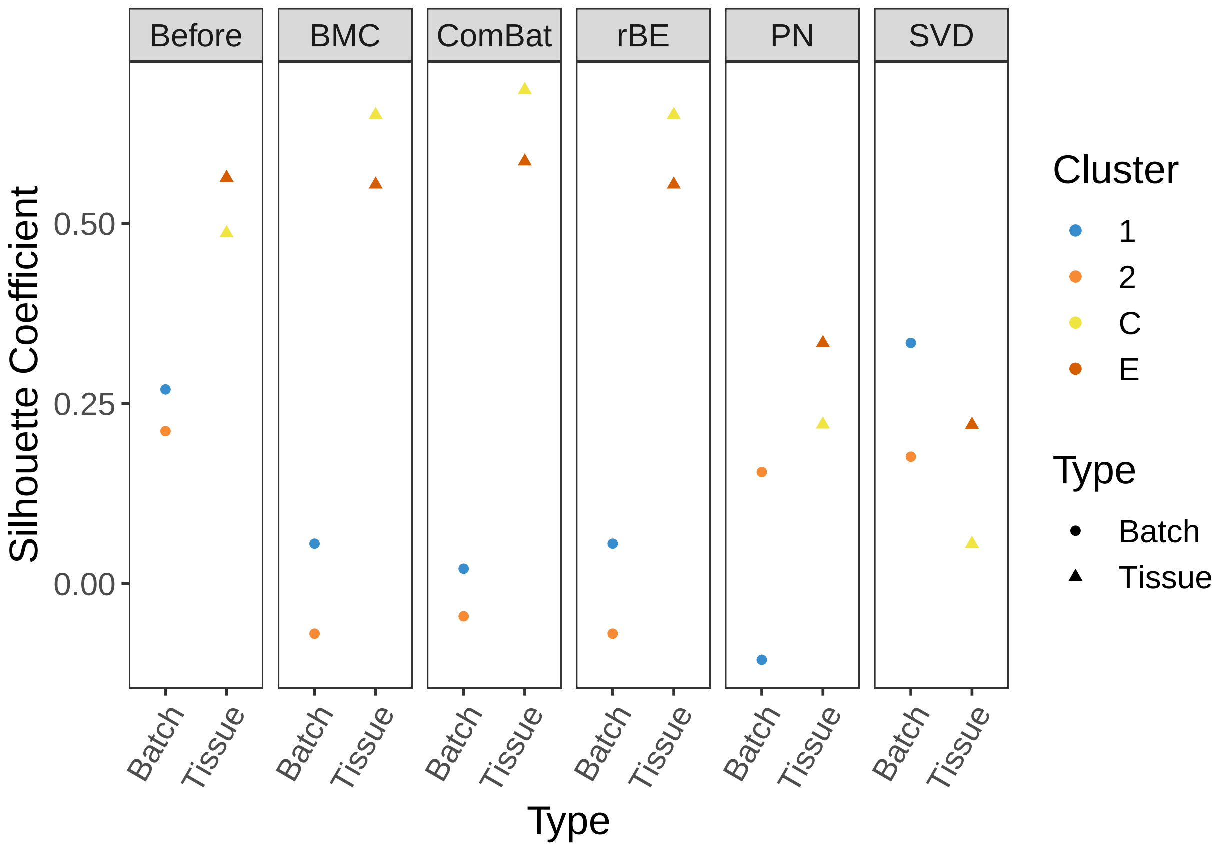
Proportionally, BMC, ComBat and removeBatchEffect removed more of the variance explained by batch and maintained more variance explained by treatment than other methods in AD data.

4.2.4 Silhouette coefficient

Finally we assess the quality of the clusters, based on either batch or treatment information using the silhouette coefficient and the PC components. This coefficient measures how cohesive a sample is to its own cluster compared to other clusters. It has been adapted to assess the consistency of sample groups based on treatment or batch effects.

The average silhouette coefficient width across all samples in one group (batch $\bar{s}_b$ or treatment $\bar{s}_t$) is calculated and plotted here. If $\bar{s}_b$ is close to 0 there is no batch effect, and if $\bar{s}_t$ is close to 1 or -1 there is a treatment effect.

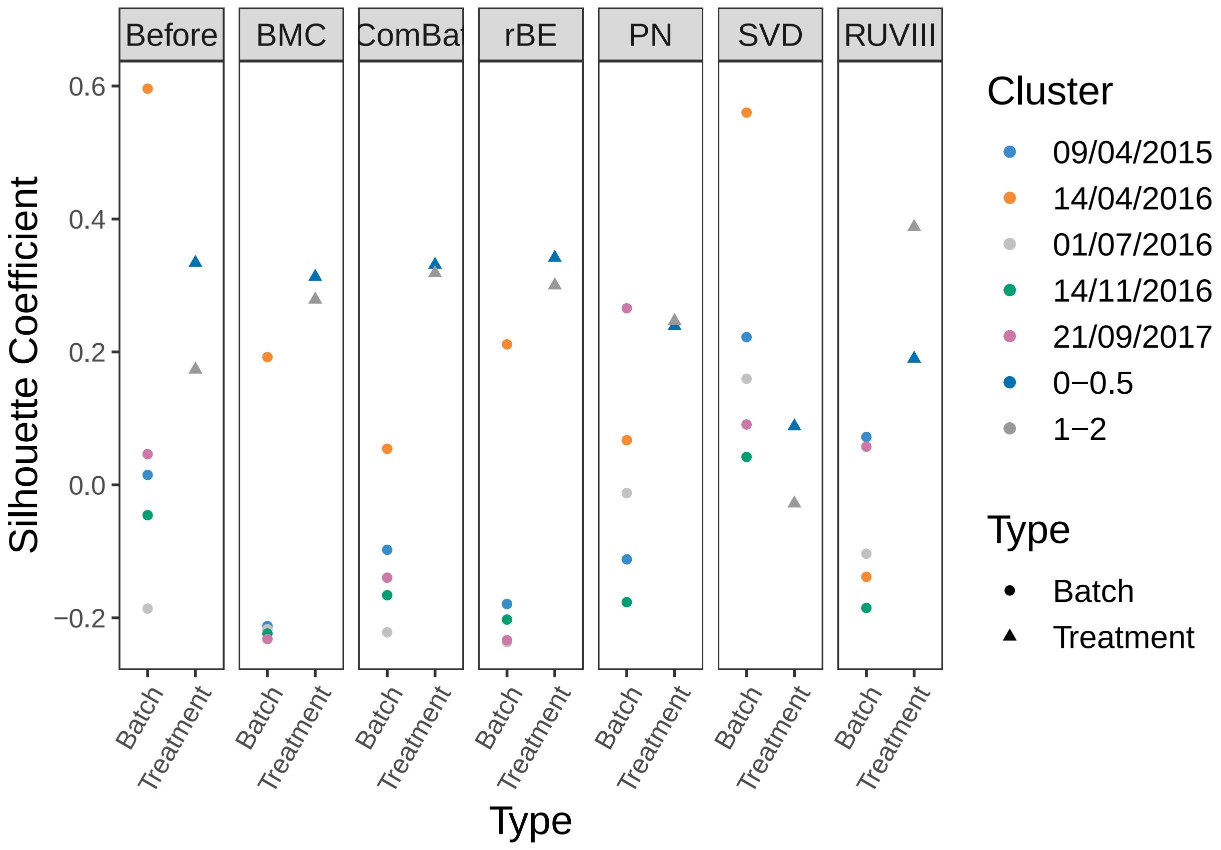
```r
# Sponge data
sponge.silh.before <- calc.sil(sponge.pca.before$variates$X,
                              y1 = sponge.batch, y2 = sponge.trt,
                              name.y1 = 'Batch', name.y2 = 'Tissue')
sponge.silh.bmc <- calc.sil(sponge.pca.bmc$variates$X,
                           y1 = sponge.batch, y2 = sponge.trt,
                           name.y1 = 'Batch', name.y2 = 'Tissue')
sponge.silh.combat <- calc.sil(sponge.pca.combat$variates$X,
                              y1 = sponge.batch, y2 = sponge.trt,
                              name.y1 = 'Batch', name.y2 = 'Tissue')
sponge.silh.limma <- calc.sil(sponge.pca.limma$variates$X,
                             y1 = sponge.batch, y2 = sponge.trt,
                             name.y1 = 'Batch', name.y2 = 'Tissue')
sponge.silh.percentile <- calc.sil(sponge.pca.percentile$variates$X,
                                  y1 = sponge.batch, y2 = sponge.trt,
                                  name.y1 = 'Batch', name.y2 = 'Tissue')
sponge.silh.svd <- calc.sil(sponge.pca.svd$variates$X,
                           y1 = sponge.batch, y2 = sponge.trt,
                           name.y1 = 'Batch', name.y2 = 'Tissue')


sponge.silh.plot <- rbind(sponge.silh.before, sponge.silh.bmc, sponge.silh.combat,
                         sponge.silh.limma, sponge.silh.percentile, sponge.silh.svd)
sponge.silh.plot$method <- c(rep('Before', nrow(sponge.silh.before)),
                            rep('BMC', nrow(sponge.silh.bmc)),
                            rep('ComBat', nrow(sponge.silh.combat)),
                            rep('rBE', nrow(sponge.silh.limma)),
                            rep('PN', nrow(sponge.silh.percentile)),
                            rep('SVD', nrow(sponge.silh.svd))
)
sponge.silh.plot$method <- factor(sponge.silh.plot$method,
                                  levels = unique(sponge.silh.plot$method))
sponge.silh.plot$Cluster <- factor(sponge.silh.plot$Cluster,
                                    levels = unique(sponge.silh.plot$Cluster))
sponge.silh.plot$Type <- factor(sponge.silh.plot$Type,
                                levels = unique(sponge.silh.plot$Type))


ggplot(sponge.silh.plot, aes(x = Type, y = silh.coeff, color = Cluster, shape = Type)) +
  geom_point() + facet_grid(cols = vars(method)) + theme_bw() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        strip.text = element_text(size = 12), panel.grid = element_blank(),
        axis.text = element_text(size = 12), axis.title = element_text(size = 15),
        legend.title = element_text(size = 15), legend.text = element_text(size = 12)) +
  scale_color_manual(values = c('#388ECC','#F68B33','#F0E442','#D55E00')) +
  labs(x = 'Type', y = 'Silhouette Coefficient', name = 'Type')
```



$\bar{s}_b$ of different batches in sponge data decreased to 0 after correction with BMC, ComBat and removeBatchEffect, while $\bar{s}_t$ of different tissues increased or maintained at the same level as before correction.

We also calculate the silhouette coefficients for AD data.

```r
# AD data
ad.silh.before <- calc.sil(ad.pca.before$variates$X, y1 = ad.batch,
                           y2 = ad.trt, name.y1 = 'Batch', name.y2 = 'Treatment')
ad.silh.bmc <- calc.sil(ad.pca.bmc$variates$X, y1 = ad.batch,
                        y2 = ad.trt, name.y1 = 'Batch', name.y2 = 'Treatment')
ad.silh.combat <- calc.sil(ad.pca.combat$variates$X, y1 = ad.batch,
                           y2 = ad.trt, name.y1 = 'Batch', name.y2 = 'Treatment')
ad.silh.limma <- calc.sil(ad.pca.limma$variates$X, y1 = ad.batch,
                          y2 = ad.trt, name.y1 = 'Batch', name.y2 = 'Treatment')
ad.silh.percentile <- calc.sil(ad.pca.percentile$variates$X, y1 = ad.batch,
                               y2 = ad.trt, name.y1 = 'Batch', name.y2 = 'Treatment')
ad.silh.svd <- calc.sil(ad.pca.svd$variates$X, y1 = ad.batch,
                        y2 = ad.trt, name.y1 = 'Batch', name.y2 = 'Treatment')
ad.silh.ruv <- calc.sil(ad.pca.ruv$variates$X, y1 = ad.batch,
                        y2 = ad.trt, name.y1 = 'Batch', name.y2 = 'Treatment')


ad.silh.plot <- rbind(ad.silh.before, ad.silh.bmc, ad.silh.combat,
                      ad.silh.limma, ad.silh.percentile, ad.silh.svd, ad.silh.ruv)
ad.silh.plot$method <- c(rep('Before', nrow(ad.silh.before)),
                         rep('BMC', nrow(ad.silh.bmc)),
                         rep('ComBat', nrow(ad.silh.combat)),
                         rep('rBE', nrow(ad.silh.limma)),
                        rep('PN', nrow(ad.silh.percentile)),
                        rep('SVD', nrow(ad.silh.svd)),
                        rep('RUVIII', nrow(ad.silh.ruv))
)
ad.silh.plot$method <- factor(ad.silh.plot$method, levels = unique(ad.silh.plot$method))
ad.silh.plot$Cluster <- factor(ad.silh.plot$Cluster, levels = unique(ad.silh.plot$Cluster))
ad.silh.plot$Type <- factor(ad.silh.plot$Type, levels = unique(ad.silh.plot$Type))

ggplot(ad.silh.plot, aes(x = Type, y = silh.coeff, color = Cluster, shape = Type)) +
  geom_point() + facet_grid(cols = vars(method)) + theme_bw() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        strip.text = element_text(size = 12), panel.grid = element_blank(),
        axis.text = element_text(size = 10), axis.title = element_text(size = 15),
        legend.title = element_text(size = 15), legend.text = element_text(size = 12)) +
  scale_color_manual(values = c('#388ECC', '#F68B33', '#C2C2C2', '#009E73',
                                '#CC79A7', '#0072B2', '#999999')) +
  labs(x = 'Type', y = 'Silhouette Coefficient', name = 'Type')
```



In AD data that includes five batches, the interpretation of the results is challenging. After correction, BMC, ComBat, removeBatchEffect, percentile normalisation and RUVIII decreased the batch silhouette coefficients for the batch dated 14/04/2016, but increased the coefficients for the other batches. Therefore it is difficult to assess the efficiency of these methods in this particular case.

Chapter 5

Assessment of the nature of batch effects

5.1 Simulations

5.1.1 Mean = 5, and unequal variance

'Systematic' refers to homogeneous change amongst all microbial variables (OTUs) due to having the same source of variation. We illustrate this concept in a linear model framework where batch and treatment regression coefficients are estimated simultaneously on each OTU. For a given batch effect and a given OTU, we can formulate the systematic assumption as:

$$\beta_j \sim N(\mu, \sigma^2)$$

where β_j is the batch regression coefficient of OTU$_j$ $(j = 1, ..., p)$. Here we consider the simplest case of a linear model with one batch predictor, but this formulation could be extended to a model with multiple batch predictors where the batch regression coefficients can represent more than two batch levels. In a univariate model that tests each OTU individually, then the distribution of the batch coefficients of all OTUs is Gaussian with a mean μ, and standard deviation σ. This indicates that the batch effect has a similar, though not necessarily identical, influence on all OTUs.

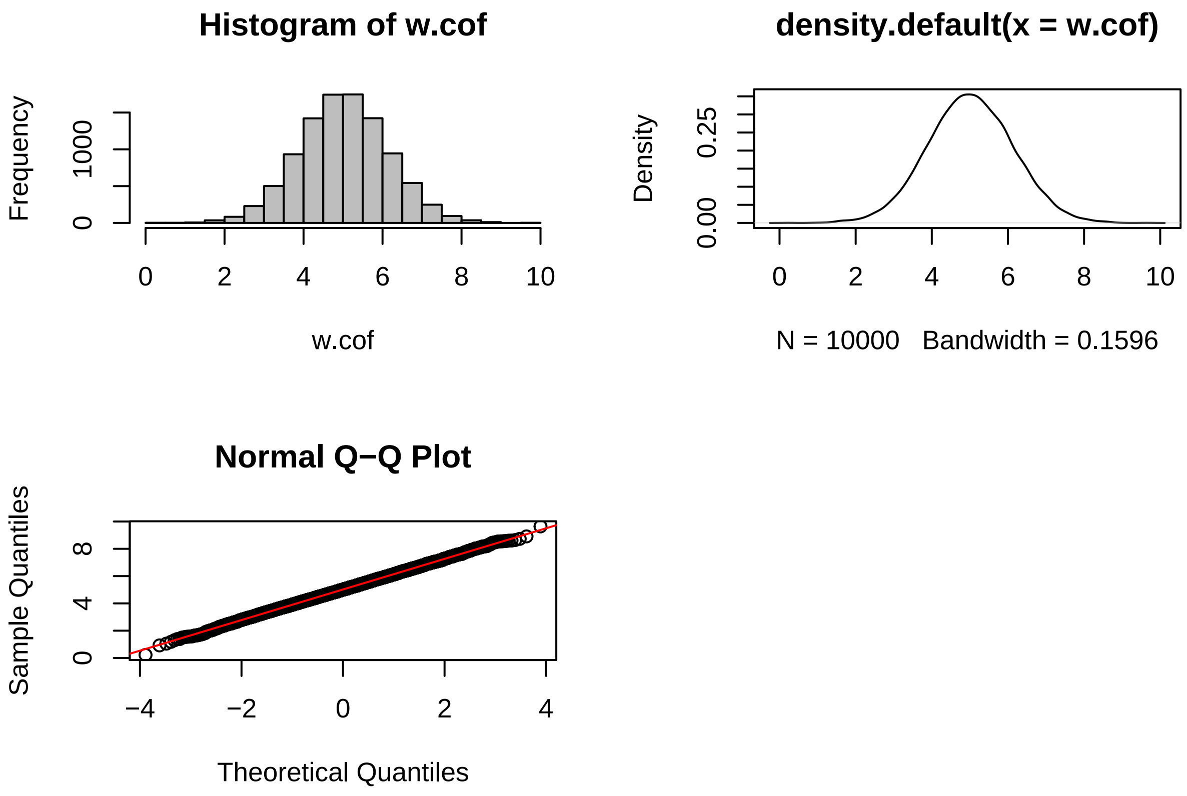
To illustrate the type of batch effects, we simulated a set of data with 50 samples and 10,000 OTUs each, based on the simulation approach from (Gagnon-Bartsch et al., 2013).

The dataset with a systematic batch effect:

- $\beta_j \sim N(5, 1^2)$ for $j = 1, ..., p$ OTUs;
- $\sigma_j \sim N(0, 2^2)$ for $j = 1, ..., p$ OTUs; This variance per OTU is aimed to simulate data as realistically as possible.
- $\beta_{ij} \sim N(\beta_j, \sigma_j^2)$ for $i = 1, ..., n$ samples.

```
# Create the simulated data
m <- 50
n <- 10000
nc <- 1000 # negative controls without treatment effects
p <- 1
k <- 1
ctl <- rep(FALSE, n)
ctl[1:nc] <- TRUE
# treatment effect
X <- matrix(c(rep(0, floor(m/2)), rep(1, ceiling(m/2))), m, p)
beta <- matrix(rnorm(p*n, 5, 1), p, n) #treatment coefficients
```

```
beta[ ,ctl] <- 0
# batch effect
W <- as.matrix(rep(0, m), m, k)
W[c(1:12,38:50), 1] <-  1
alpha <- matrix(rnorm(k*n, 5, 1), k, n)
Y_alpha <- sapply(alpha, function(alpha){rnorm(m, mean =  alpha,
                                              abs(rnorm(1, mean = 0, sd = 2)))})
YY_alpha <- apply(Y_alpha, 2, function(x){x*W})

epsilon <- matrix(rnorm(m*n, 0, 1), m, n)
Y <- X%*%beta + YY_alpha + epsilon


# estimate batch coefficient for each OTU
w.cof <- c()
for(i in 1:ncol(Y)){
  res <- lm(Y[ ,i] ~ X + W)
  sum.res <- summary(res)
  w.cof[i] <- sum.res$coefficients[3,1]
}

par(mfrow = c(2,2))
hist(w.cof,col = 'gray')
plot(density(w.cof))
qqnorm(w.cof)
qqline(w.cof, col = 'red')
par(mfrow = c(1,1))
```



The histogram, density plot and quantile–quantile plot are plotted using estimated batch regression coefficients $\hat{\beta}_j$ for each

OTU j. These plots display the distribution of batch coefficients is very close to normal, indicating a systematic batch effect.

5.1.2 Mean = 0 or 5, and unequal variance

Non-systematic batch effects have a heterogeneous influence on microbial variables. Using a linear model framework, as described previously, we can formulate this non-systematic assumption as:

$$\beta'_j \sim \begin{cases} N(0, \delta^2) & \text{for OTUs with no batch effect,} \\ N(\mu, \sigma^2) & \text{for OTUs with batch effect.} \end{cases}$$

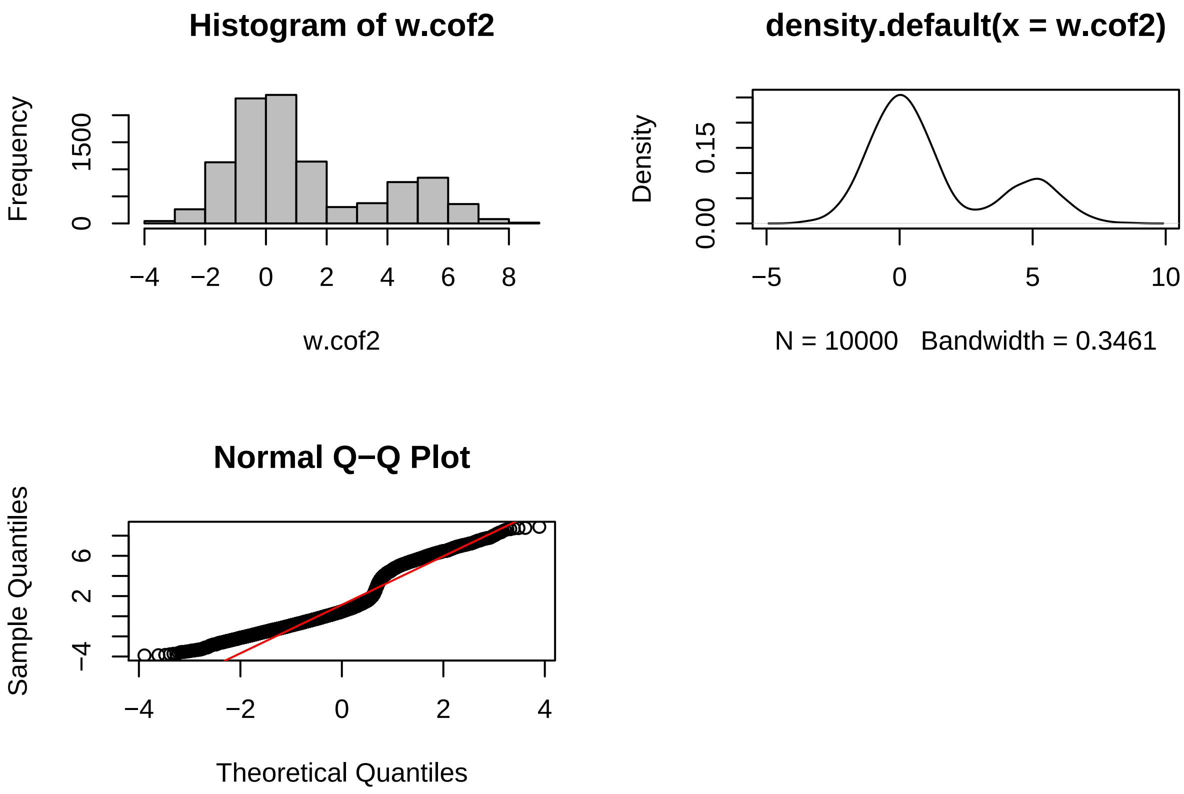
Therefore, the batch regression coefficients β'_j may follow skewed distributions with several modes.

The dataset with non-systematic batch effect:

- $\beta'_t \sim N(0, 1^2)$ and $\beta'_k \sim N(5, 1^2)$ for $t = 1, ..., T$ OTUs, $k = 1, ..., K$ OTUs and $T = \frac{3}{4}p$, $K = \frac{1}{4}p$;
- $\sigma'_j \sim N(0, 2^2)$ for $j = 1, ..., p$ OTUs;
- $\beta'_{ij} \sim N(\beta'_j, \sigma'^2_j)$ for $i = 1, ..., n$ samples.

```r
# Create the simulated data
m <- 50
n <- 10000
nc <- 1000 # negative controls without treatment effects
p <- 1
k <- 1
ctl <- rep(FALSE, n)
ctl[1:nc] <- TRUE
# treatment effect
X <- matrix(c(rep(0, floor(m/2)), rep(1, ceiling(m/2))), m, p)
beta <- matrix(rnorm(p*n, 5, 1), p, n) #treatment coefficients
beta[ ,ctl] <- 0
# batch effect
W <- as.matrix(rep(0, m), m, k)
W[c(1:12,38:50), 1] <-  1
alpha2 <- matrix(sample(c(rnorm(k*(3*n/4), 0, 1),rnorm(k*(n/4), 5, 1)), n), k, n)
Y_alpha2 <- sapply(alpha2, function(alpha){rnorm(m, mean =  alpha,
                                                 sd = abs(rnorm(1, mean = 0, sd = 2)))})
YY_alpha2 <- apply(Y_alpha2, 2, function(x){x*W})

epsilon <- matrix(rnorm(m*n, 0, 1), m, n)
Y2 <- X%*%beta + YY_alpha2 + epsilon





w.cof2 <- c()
for(i in 1:ncol(Y2)){
  res <- lm(Y2[ ,i] ~ X + W)
  sum.res <- summary(res)
  w.cof2[i] <- sum.res$coefficients[3,1]
}
```

```r
par(mfrow = c(2,2))
hist(w.cof2, col = 'gray')
plot(density(w.cof2))
qqnorm(w.cof2)
qqline(w.cof2, col = 'red')
par(mfrow = c(1,1))
```



The histogram, density plot and quantile–quantile plot are plotted using estimated batch regression coefficients $\hat{\beta}_j$ for each OTU j, showing a bi-modal distribution. This distribution indicates a non-systematic batch effect.

We observed similar patterns in our real case studies, suggesting that the batch effects are mixed with multiple sources and are non-systematic (see the examples below).

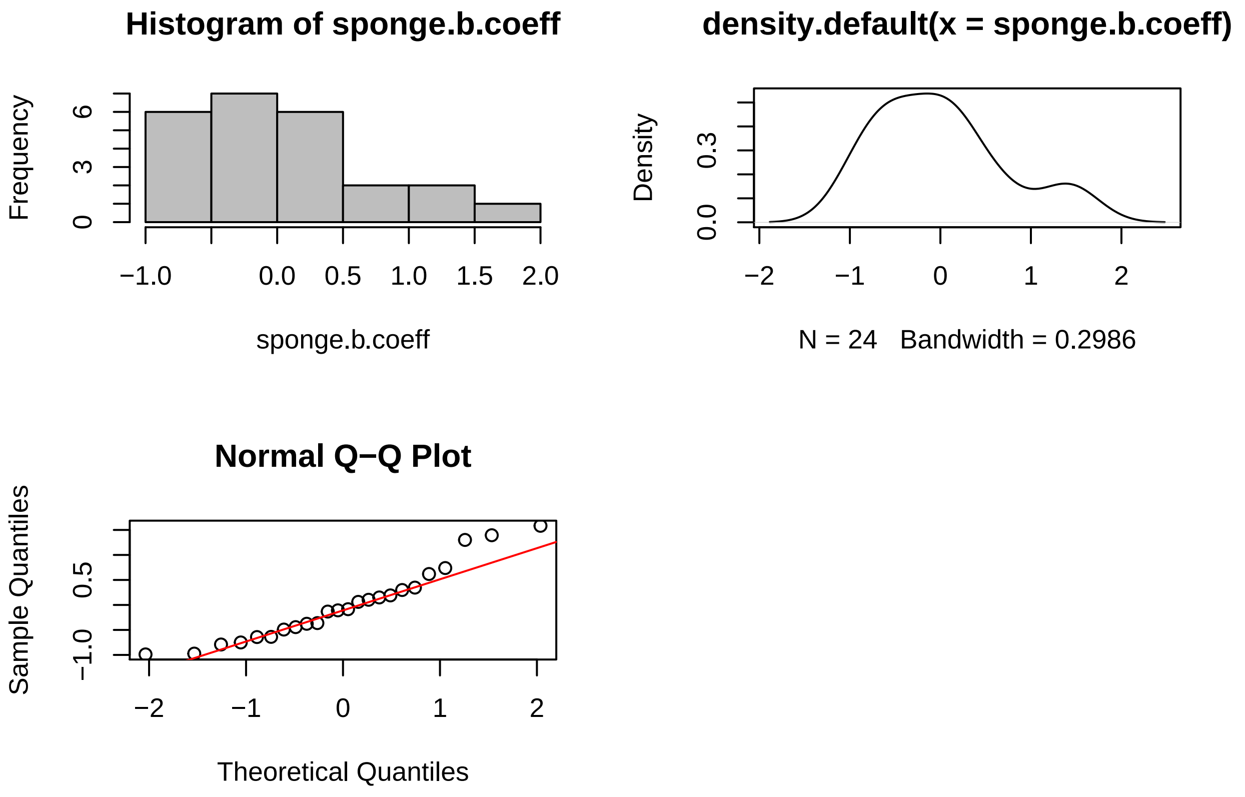
5.2 Real data

We also estimate the batch regression coefficients for each OTU in real datasets to assess the nature of batch effects.

5.2.1 Sponge data

```r
sponge.b.coeff <- c()
for(i in 1:ncol(sponge.tss.clr)){
  res <- lm(sponge.tss.clr[ ,i] ~ sponge.trt + sponge.batch)
  sum.res <- summary(res)
  sponge.b.coeff[i] <- sum.res$coefficients[3,1]
}
```

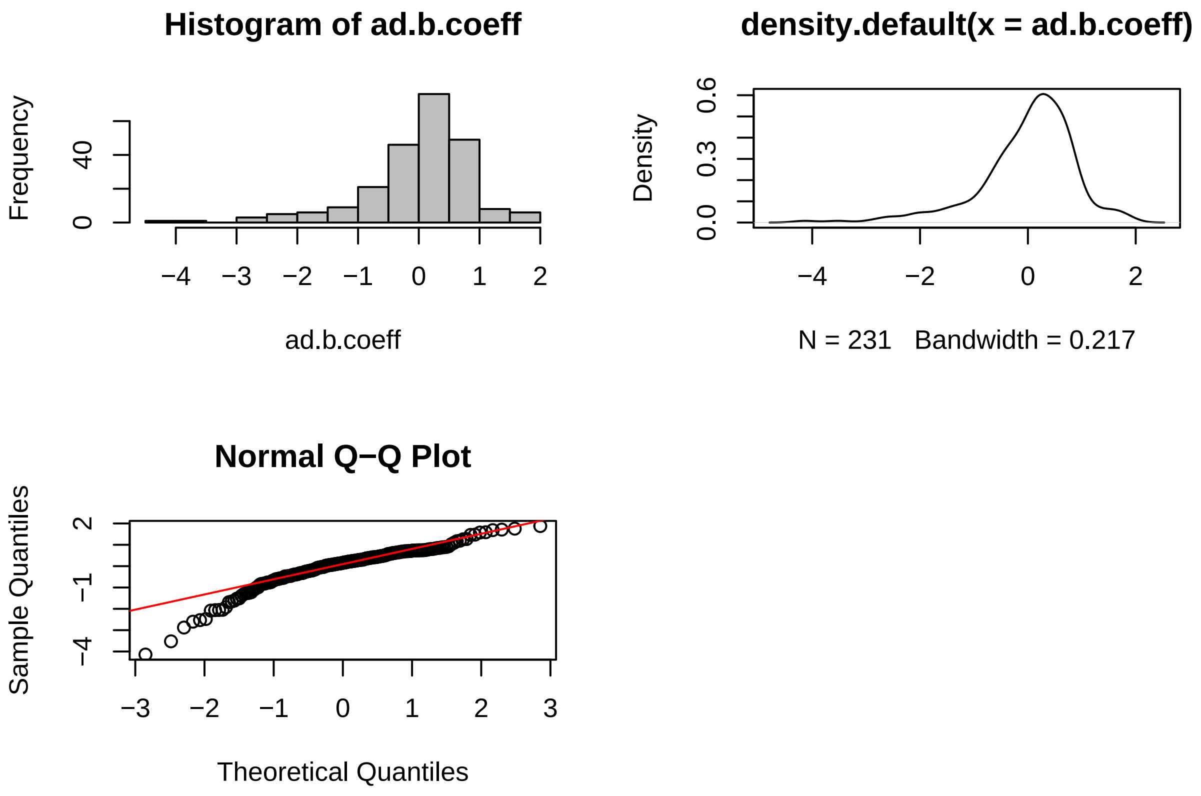
```
par(mfrow = c(2,2))
hist(sponge.b.coeff,col = 'gray')
plot(density(sponge.b.coeff))
qqnorm(sponge.b.coeff)
qqline(sponge.b.coeff, col='red')
par(mfrow = c(1,1))
```



In sponge data, the histogram, density plot and quantile–quantile plot are also plotted using estimated batch regression coefficients $\hat{\beta}_j$ for each OTU j. The bi-modal distribution indicates a non-systematic batch effect, which is similar as the results from simulation with non-systematic batch effects.

5.2.2 AD data

```
ad.b.coeff <- c()
ad.batch.relevel <- relevel(ad.batch, '01/07/2016')
for(i in 1:ncol(ad.clr)){
  res <- lm(ad.clr[,i] ~ ad.trt + ad.batch.relevel)
  sum.res <- summary(res)
  ad.b.coeff[i] <- sum.res$coefficients[4,1]
}

par(mfrow = c(2,2))
hist(ad.b.coeff,col = 'gray')
plot(density(ad.b.coeff))
qqnorm(ad.b.coeff)
qqline(ad.b.coeff, col='red')
par(mfrow = c(1,1))
```



We observe similar results in AD data. The distribution of estimated batch regression coefficients is not normal. The batch effects therefore are non-systematic.

Bibliography

Aitchison, J. (1986). *The statistical analysis of compositional data*. Chapman and Hall London.

Arumugam, M., Raes, J., Pelletier, E., Le Paslier, D., Yamada, T., Mende, D. R., Fernandes, G. R., Tap, J., Bruls, T., Batto, J.-M., et al. (2011). Enterotypes of the human gut microbiome. *nature*, 473(7346):174.

Chapleur, O., Madigou, C., Civade, R., Rodolphe, Y., Mazéas, L., and Bouchez, T. (2016). Increasing concentrations of phenol progressively affect anaerobic digestion of cellulose and associated microbial communities. *Biodegradation*, 27(1):15–27.

Gagnon-Bartsch, J. A., Jacob, L., and Speed, T. P. (2013). Removing unwanted variation from high dimensional data with negative controls. *Berkeley: Tech Reports from Dep Stat Univ California*, pages 1–112.

Gibbons, S. M., Duvallet, C., and Alm, E. J. (2018). Correcting for batch effects in case-control microbiome studies. *PLoS Computational Biology*, 14(4):e1006102.

Kong, G., Lê Cao, K.-A., Judd, L. M., Li, S., Renoir, T., and Hannan, A. J. (2018). Microbiome profiling reveals gut dysbiosis in a transgenic mouse model of Huntington's disease. *Neurobiology of Disease*.

Lin, Y., Ghazanfar, S., Wang, K., Gagnon-Bartsch, J. A., Lo, K. K., Su, X., Han, Z.-G., Ormerod, J. T., Speed, T. P., Yang, P., et al. (2018). scMerge: Integration of multiple single-cell transcriptomics datasets leveraging stable expression and pseudo-replication. *BioRxiv*, page 393280.

Sacristán-Soriano, O., Banaigs, B., Casamayor, E. O., and Becerro, M. A. (2011). Exploring the links between natural products and bacterial assemblages in the sponge Aplysina aerophoba. *Applied and Environmental Microbiology*, 77(3):862–870.