Supplementary methods

Benchmarking of different methods for detection of DTGs on simulated data

```
library(polyester)
library(DESeq2)
library(xtail)
library(edgeR)
library(ggplot2)
library(riborex)
library(RColorBrewer)
```

Simulating Ribo-seq and RNA-seq counts with batch effect

We simulated counts based on a published rat RNA-seq and Ribo-seq dataset with 4 control samples, randomly assigning two as condition 1 and two as condition 2. The fold changes were sampled from a gamma distribution (k=0.6, =0.5) and sample batch effect with varying batch coefficient from 0 to 1 was introduced using Polyester package10. We randomly assigned 10% of the genes a fold change of 1.5 higher in Ribo-seq or RNA-seq in one of the two conditions and those 10% of genes were also assigned a batch coefficient. This sampling and simulation was carried out 50 times to obtain mean and standard deviations for each batch effect setting.

Detecting differential translation genes (DTGs)

We used the following tools to identify DTGs:
1) DESeq2
2) Riborex (DESeq2, edgeR, edgeRD, voom)
3) Xtail
4) RiboDiff

Xtail and Riborex require the Ribo-seq and RNA-seq count matrices and a vector with conditions for each sample in the same order as in the count matrix. RiboDiff also requires a sample information file which including condition information, and sequencing type information (Ribo-seq, RNA-seq) per sample in the same order. DESeq2 interaction term model was used as follows: ~batch + condition + sequencing + condition:sequencing, where sequencing is ribo-seq or rna-seq, condition is control/treated and batch is any covariate in the data

```
get_DTG=function(counts_ribo, counts_rna, bcoeff, bgenes, te_genes, num_samples,
                 conditions=2, all_methods = "TRUE"){
    par(mfrow=c(1,1))

    # Number of DTGs to simulate
    dTE_genes = round(te_genes*nrow(counts_ribo)/100,0)
    # Number of genes to simulate with batch
    bgenes = round(bgenes*nrow(counts_ribo)/100,0)

    # Number of conditions and samples
    group = rep(seq(0,conditions-1,by=1),each=num_samples)

    # Batch information for each sample (here it is a sample batch
```

```
# effect so each sample is assigned one batch)
batch = rep(seq(1,num_samples,by=1),conditions)

# Sampling fold changes from gamma distribution with shape=0.6 and scale=0.5
gcoeffs = c(rgamma(nrow(counts_ribo),
                   shape=0.6,scale=0.5)*sample(c(-1,1),nrow(counts_ribo),
                                               replace=T))

# Randomly selecting genes to assign as DTG
select = sample(1:nrow(counts_ribo),dTE_genes)

# Assigning batch effect coefficient to the DTGs and others depending on bgenes supplied
bcoeffs = rep(0,nrow(counts_ribo))
if(bgenes <= length(select)){
  bselect <- sample(select,bgenes)
}else if(bgenes > length(select)){
  bselect <- c(select,sample(setdiff(1:nrow(counts_ribo),select),bgenes-length(select)))
}
bcoeffs[bselect] = bcoeff*sample(c(1,-1),bgenes,replace=TRUE)

# Preparing coefficient vector and model for simulation
coeffs = cbind(bcoeffs,gcoeffs)
mod = model.matrix(~-1 + batch + group)

## Ribo-seq count simulation based on params from known dataset (counts_ribo) ##
params = get_params(counts_ribo)
sim_ribo = create_read_numbers(params$mu,params$fit,params$p0,beta=coeffs,
                               mod=mod,seed=32332)

# Randomly add or subtract 1.5 log fold change to RNA with respect
# to Ribo-seq fold change coefficient
gcoeffs2 = gcoeffs
gcoeffs2[select] = gcoeffs[select] + sample(c(1.5,-1.5),replace=T,length(select))
gcoeffs = gcoeffs2

# Assign those genes selected as DTG positive
labels = rep(0,nrow(counts_ribo))
labels[select] = 1
coeffs = cbind(bcoeffs,gcoeffs)

## RNA-seq count simulation based on params from known dataset (counts_rna) ##
params = get_params(counts_rna)
sim_rna = create_read_numbers(params$mu,params$fit,
                              params$p0,
                              beta=coeffs,mod=mod,seed=32332)


### Detecting DTGs
# Merge matrices
merged <- cbind(sim_ribo,sim_rna)
colnames(merged) <- seq(1,(ncol(sim_ribo)+ncol(sim_rna)),by=1)
colnames(merged)[1:ncol(sim_ribo)] <- sub("$","_Ribo",colnames(merged)[1:ncol(sim_ribo)])
colnames(merged)[(ncol(sim_ribo)+1):(ncol(sim_ribo) + ncol(sim_rna))] <-
```

```
  sub("$","_RNA",colnames(merged)[(ncol(sim_ribo)+1):(ncol(sim_ribo) + ncol(sim_rna))])

coldata <- as.data.frame(cbind(colnames(merged),group,colnames(merged),batch))
colnames(coldata) <- c("sample","group","seq_type","batch")

rownames(coldata) <- coldata[,1]
coldata[,3] <- sub(".*_","",coldata[,3])
coldata[,3] <- as.factor(coldata[,3])


# Filtering very low counts

filter <- apply(sim_ribo, 1, function(x) length(x[x>5])>=2)
sim_rna <- as.matrix(sim_rna[filter,])
sim_ribo <- as.matrix(sim_ribo[filter,])
merged <- as.matrix(merged[filter,])
labels <- labels[filter]

filter <- apply(sim_rna, 1, function(x) length(x[x>5])>=2)
sim_rna <- as.matrix(sim_rna[filter,])
sim_ribo <- as.matrix(sim_ribo[filter,])
merged <- as.matrix(merged[filter,])
labels <- labels[filter]

# DESeq2 interaction term
ddsMat <- DESeqDataSetFromMatrix(countData = merged,
                                 colData = coldata,
                                design =~ batch + group + seq_type + group:seq_type)

ddsMat <- DESeq(ddsMat)
res <- results(ddsMat, contrast=list("group1.seq_typeRNA"))
length(which(res$padj < 0.05))
DESeq2::plotMA(res)

# Check batch effects based on PC1 and PC2 (Principal component analysis)
group1 <- coldata$group[which(coldata$seq_type == "Ribo")]
batch1 <- coldata$batch[which(coldata$seq_type == "Ribo")]

vsd <- vst(sim_ribo,nsub=900)
plotPCA(vsd,1,2,coldata,"Before batch correction")

vsd_nobatch <- removeBatchEffect(vsd, batch1)
plotPCA(vsd_nobatch,1,2,coldata, "After batch correction")

res_all <- matrix(nrow=nrow(res),ncol=4)
res_all[,1] = res$padj

### Running Riborex
rnacond <- c(rep("BN",num_samples),rep("SHR",num_samples))
rownames(sim_rna) <- seq(1,nrow(sim_rna),by=1)
rownames(sim_ribo) <- seq(1,nrow(sim_ribo),by=1)
res.riborex.deseq2 <- riborex(sim_rna,sim_ribo, rnacond, rnacond,engine="DESeq2")
length(which(res.riborex.deseq2$padj< 0.05))
```

```
    res_all[,2] = res.riborex.deseq2$padj[order(as.numeric(rownames(res.riborex.deseq2)))]

    if(all_methods == TRUE){
      ### Running Xtail
      xtail <- xtail(sim_rna,sim_ribo,rnacond)
      length(which(xtail$resultsTable$pvalue.adjust < 0.05))
      res_all[,3] = xtail$resultsTable$pvalue.adjust

      ### Running RiboDiff
      counts_ribodiff = cbind(seq(1,nrow(merged),by=1),merged)
      colnames(counts_ribodiff)[1] = "ID"
      write.table(counts_ribodiff,"counts_forRibodiff.txt",quote=F,row.names=F,col.names=T,sep="\t")
system("python ~/bin/RiboDiff/scripts/TE.py -e coldata.txt -c counts_forRibodiff.txt -o output_RiboDiff
      ribodiff <- read.delim("output_RiboDiff.txt",sep="\t")
      length(which(ribodiff$padj < 0.05))
      res_all[,4] = ribodiff$padj
      system("rm output_RiboDiff.txt")
      system("rm output_RiboDiff.pkl")
      system("rm counts_forRibodiff.txt")
    }

    plot_ss(res_all,labels,dTE_genes,bcoeff,bgenes)
}
```

Calculating and plotting Sensitivity and Specificity

Sensitivity is defined as the true positive rate =TP/ (TP+FN), and Specificity is defined as the true negative rate =TN / (TN+FP) where TP, FN, TN, FP are the true positives, false negatives, true negatives and the false positives respectively. Sensitivity and Specificity were calculated for various FDR thresholds (0 to 0.2).

```
plot_ss=function(res_all,labels,dTE_genes,bcoeff,bgenes){
  breaks_per_order = 100
  lowest_order = 5

  # FDR thresholds
  FDR_cutoffs=seq(0,0.2,by=0.001)
  plot(FDR_cutoffs,rep(0,length(FDR_cutoffs)),
    ylim=c(0,1),col="white",type="l",xlab="FDR cutoff",
    ylab="Sensitivity and Specificity",main=paste("Batch coeff: ",bcoeff,sep=""))

  library(RColorBrewer)
  cols <- c("darkgreen","orange","yellow","purple")
  roc = matrix(nrow=length(FDR_cutoffs),ncol=2)
  roc_all = matrix(nrow=length(FDR_cutoffs),ncol=0)

  for(j in 1:ncol(res_all)){
    roc = matrix(nrow=length(FDR_cutoffs),ncol=2)
    for(i in 1:nrow(roc)){

      ## Checking for true positives (hits) and false negatives (missed)
      trues <- res_all[,j][labels==1]
      tp <- length(which(trues <= FDR_cutoffs[i]))
```

```
      fn <- length(which(trues > FDR_cutoffs[i])) + length(which(is.na(trues)))

      ## Checking for true negatives (correct rejection) and false positives (false hit)
      false <- res_all[,j][labels==0]
      fp <- length(which(false <= FDR_cutoffs[i]))
      tn <- length(which(false > FDR_cutoffs[i])) + length(which(is.na(false)))

      ## Calculating Sensitivity and Specificity
      sens <- tp/(tp+fn)
      sens <- tp/length(trues)
      spec <- tn/length(false)
      roc[i,] = c(sens,spec)
    }
    lines(FDR_cutoffs,roc[,1],type="l",col=cols[j],lwd=2)
    lines(FDR_cutoffs,roc[,2],type="l",col=cols[j],lty=2,lwd=2)
    roc_all = cbind(roc_all,roc)
  }
  legend("topright",c("DESeq2","Riborex: DESeq2","Xtail","RiboDiff"),
    fill=cols,bty="n",border=NA,cex = 0.8)

  fdr = 0.05
  tp_fn = matrix(ncol=ncol(res_all),nrow=2)
  tn_fp = matrix(ncol=ncol(res_all),nrow=2)
  for(j in 1:ncol(res_all)){
    trues <- res_all[,j][labels==1]
    tp <- length(which(trues <= fdr))
    fn <- length(which(trues > fdr)) + length(which(is.na(trues)))
    false <- res_all[,j][labels==0]
    fp <- length(which(false <= fdr))
    tn <- length(which(false > fdr)) + length(which(is.na(false)))
    tp_fn[1,j] = tp
    tp_fn[2,j] = fn
    tn_fp[1,j] = tn
    tn_fp[2,j] = fp
  }

  ## Plotting the true positives and false positives
  par(mar=c(8,4,1,1))
  barplot(tp_fn[1,],col="darkgreen",ylab="# True positives",border=NA, ylim=c(0,dTE_genes))
  axis(1, at=seq(1,7,by=1),labels=c("DESeq2","Riborex: DESeq2","Riborex: edgeR",
    "Riborex: edgeRD","Riborex: voom","Xtail","RiboDiff"), col.axis="black", las=2)

  par(mar=c(8,4,1,1))
  barplot(tn_fp[2,],col="red",border=NA, ylim=c(0,dTE_genes), ylab="# False positives")
  axis(1, at=seq(1,7,by=1),labels=c("DESeq2","Riborex: DESeq2","Riborex: edgeR",
    "Riborex: edgeRD","Riborex: voom","Xtail","RiboDiff"), col.axis="black", las=2)
}
```

```
plotPCA <- function(data, comp1, comp2, coldata,title){
pca_all <- (prcomp(t(data)))
pca_all_data <- pca_all$x
pca_all_data <- cbind(as.data.frame(pca_all_data), coldata[1:8,])
percentVar <- pca_all$sdev^2 / sum(pca_all$sdev^2)
```

```
pca_all_data$pch <- pca_all_data$group
pca_all_data$col_type <- pca_all_data$batch

pca_all_data$col_type <- sub("1", "darkmagenta", pca_all_data$col_type)
pca_all_data$col_type <- sub("3", "darkgreen", pca_all_data$col_type)
pca_all_data$col_type <- sub("2", "turquoise3", pca_all_data$col_type)
pca_all_data$col_type <- sub("4", "orange", pca_all_data$col_type)

pca_all_data$pch <- sub("0", 2, pca_all_data$pch)
pca_all_data$pch <- sub("1", 3, pca_all_data$pch)

plot(pca_all_data[,comp1], pca_all_data[,comp2], col=pca_all_data$col_type,
     pch=as.numeric(pca_all_data$pch), xlab=paste("PC", comp1, ": ",
                round(percentVar[comp1], 2)*100, "% variance", sep=""),
     ylab=paste("PC", comp2, ": ", round(percentVar[comp2], 2)*100, "% variance", sep=""),
     main=title,cex=2)
}
```

Using real dataset counts to simulate

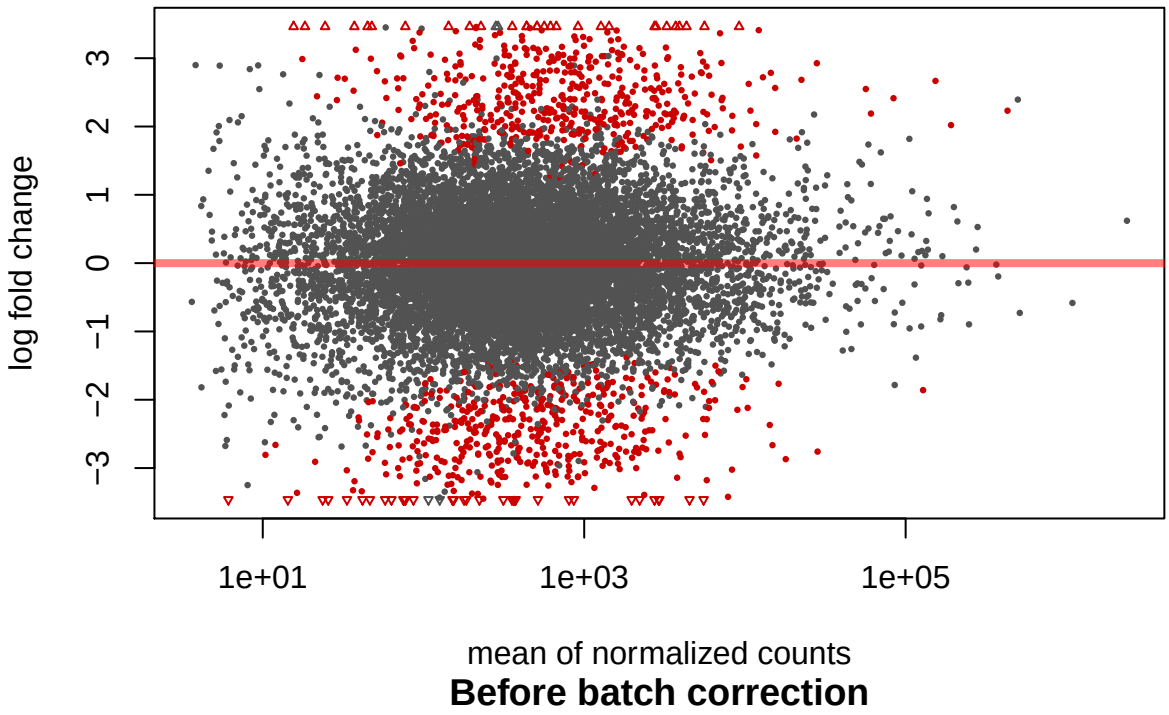
Ribo-seq and RNA-seq data from the following paper was used to simulate read counts:
Schafer, S. et al. Translational regulation shapes the molecular landscape of complex disease phenotypes. Nat. Commun. 6, 7200 (2015).

Simulations were carried out 50 times each for different batch coefficients.

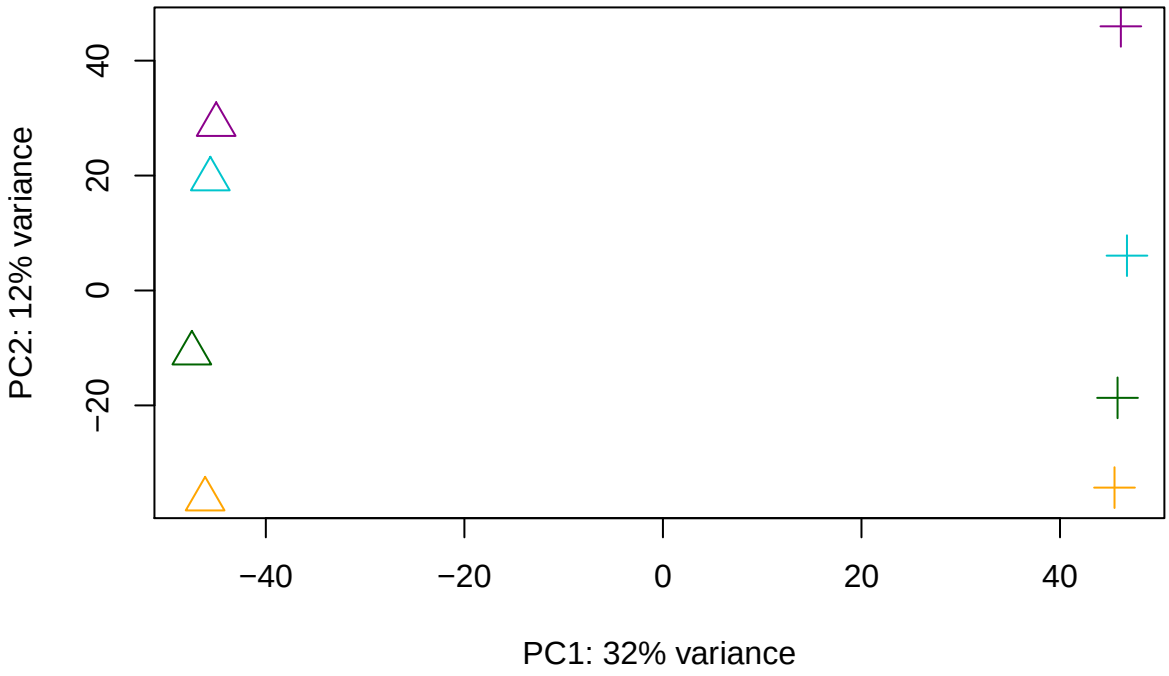
```
### Ribo-seq data
files_ribo <-list.files("counts_natcom",pattern="RPF",recursive=TRUE,full.names = T)
tmp <- read.delim(files_ribo[1],head=F,sep="")
ribo <- tmp
for(i in 2:length(files_ribo)){
  tmp <- read.delim(files_ribo[i],head=F,sep="")
  colnames(tmp)[2] = sub(".*/","",files_ribo[i])
  ribo <- merge(ribo,tmp,by="V1",all.x=TRUE)
}

colnames(ribo)[2] <- sub(".*/","",files_ribo[1])
ribo <- ribo[-grep("ENSRN",ribo[,1],invert = T),]
rownames(ribo) <- ribo[,1]
ribo <- ribo[,-1]

### RNA-seq data
files_rna <-list.files("counts_natcom",pattern="polya",recursive=TRUE,full.names = T)

tmp <- read.delim(files_rna[1],head=F,sep="")
rna <- tmp
for(i in 2:length(files_rna)){
  tmp <- read.delim(files_rna[i],head=F,sep="")
  colnames(tmp)[2] = sub(".*/","",files_rna[i])
  rna <- merge(rna,tmp,by="V1",all.x=TRUE)
}

colnames(rna)[2] <- sub(".*/","",files_rna[1])
rna <- rna[-grep("ENSRN",rna[,1],invert = T),]
```

```
rownames(rna) <- rna[,1]
rna <- rna[,-1]

## Using only one tissue type - liver, and control cases BN for simulation
ribo <- ribo[,grep("liver",colnames(ribo))]
ribo <- ribo[,grep("BN",colnames(ribo))]
rna <- rna[,grep("liver",colnames(rna))]
rna <- rna[,grep("BN",colnames(rna))]

filter <- apply(ribo, 1, function(x) length(x[x>5])>=2)
counts_ribo <- as.matrix(ribo[filter,])

filter <- apply(rna, 1, function(x) length(x[x>5])>=2)
counts_rna <- as.matrix(rna[filter,])

## Calling the function for simulation and plotting

#pdf("batch-pca.pdf")
# You can run this script N times to obtain mean and standard deviation for the sensitivity
# and specificity curves
# for(j in 1:50){
# For various batch coefficients increasing from 0 (No batch)
# to 1.3 (batch accounting for 40% variance)
par_bcoeff = c(0,seq(0.1,1.3,by=0.2))
for(i in 1:length(par_bcoeff)){
  results = get_DTG(counts_ribo, counts_rna, bcoeff=par_bcoeff[i], bgenes = 10,
                    te_genes=10, num_samples=4, conditions=2)
}
```

```
## converting counts to integer mode

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## converting counts to integer mode
```

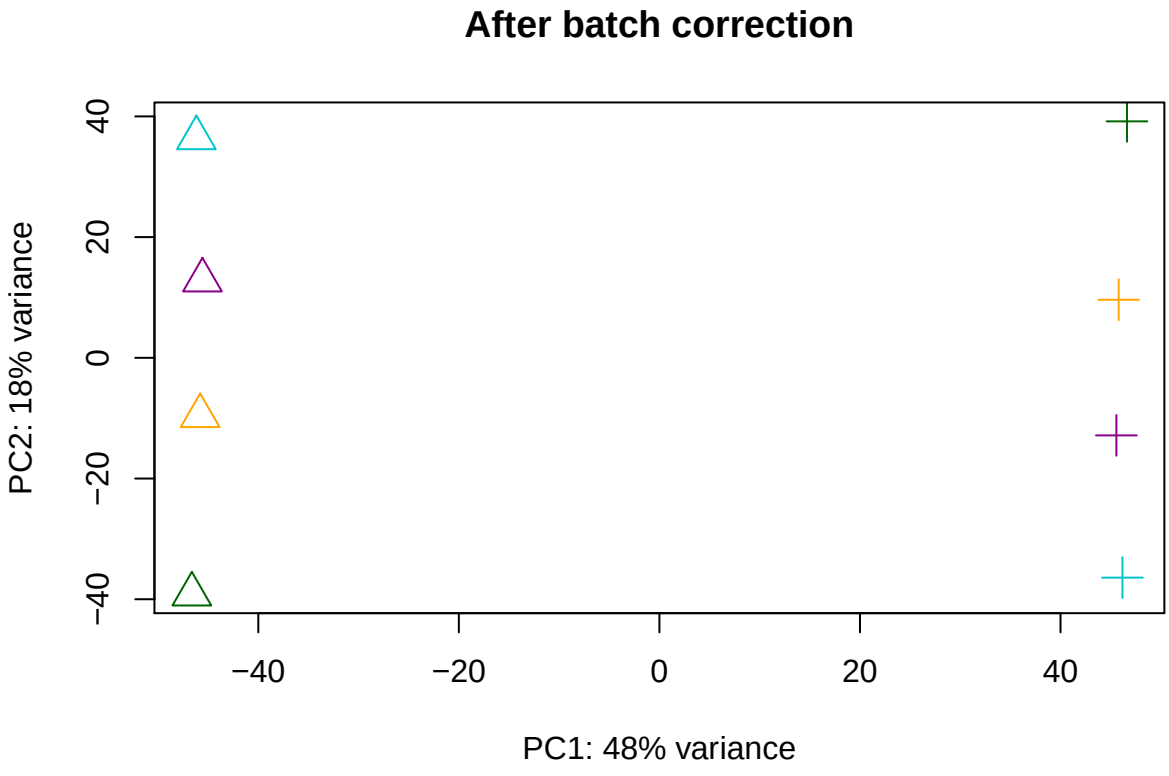


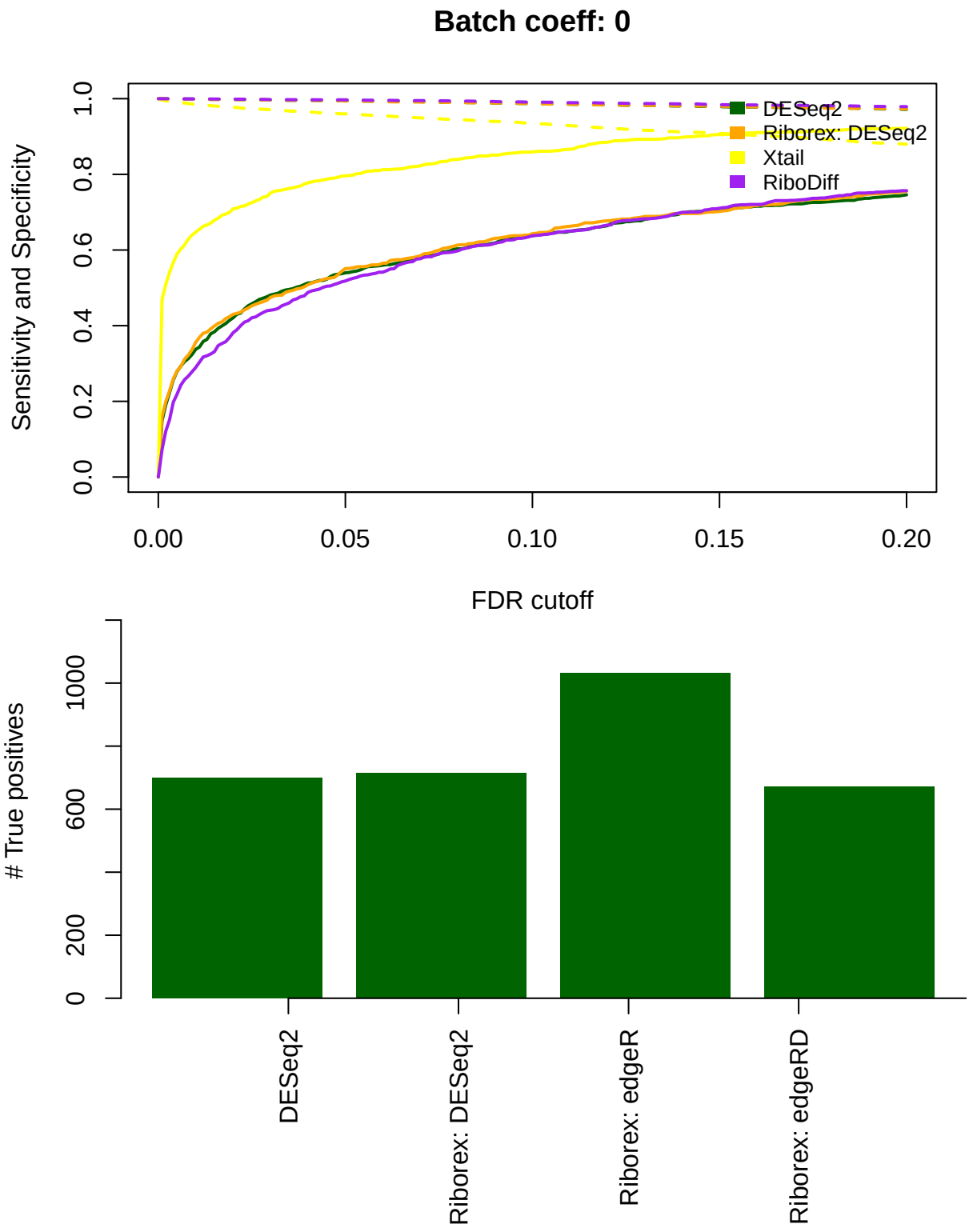
Before batch correction




```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
```

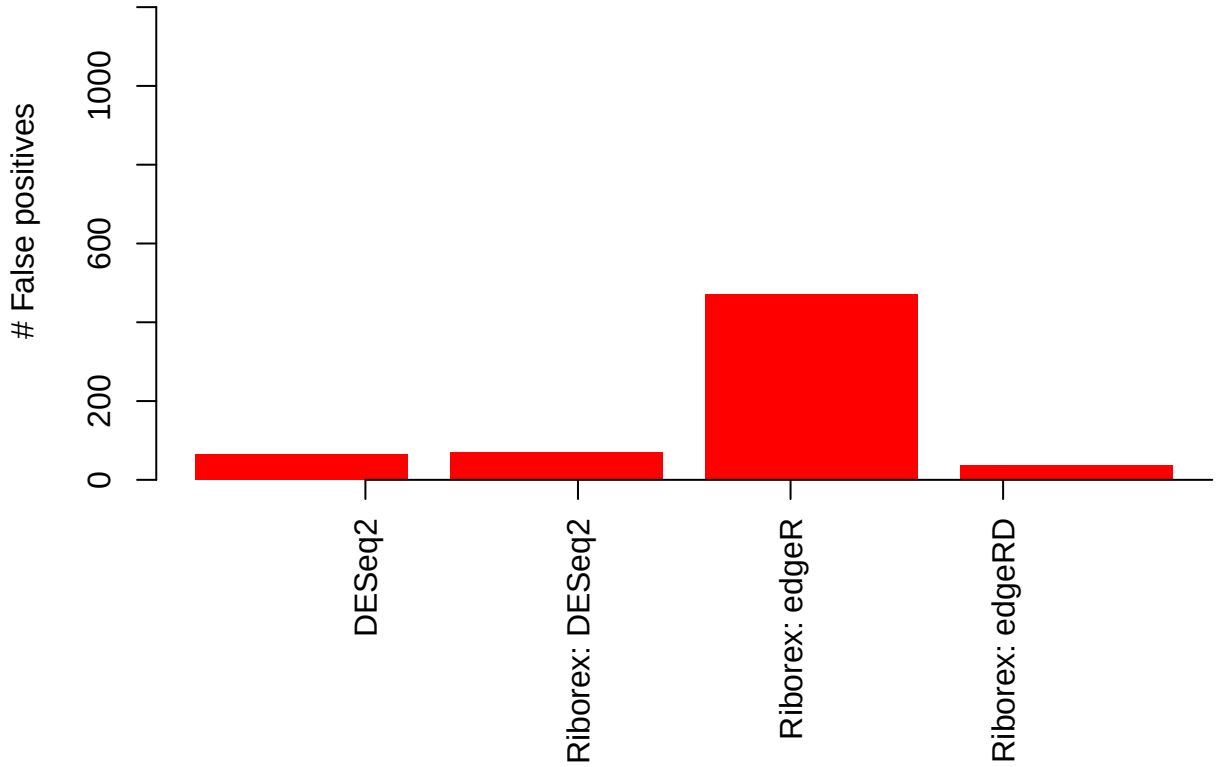
```
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6807
##  log2R: 6187
```



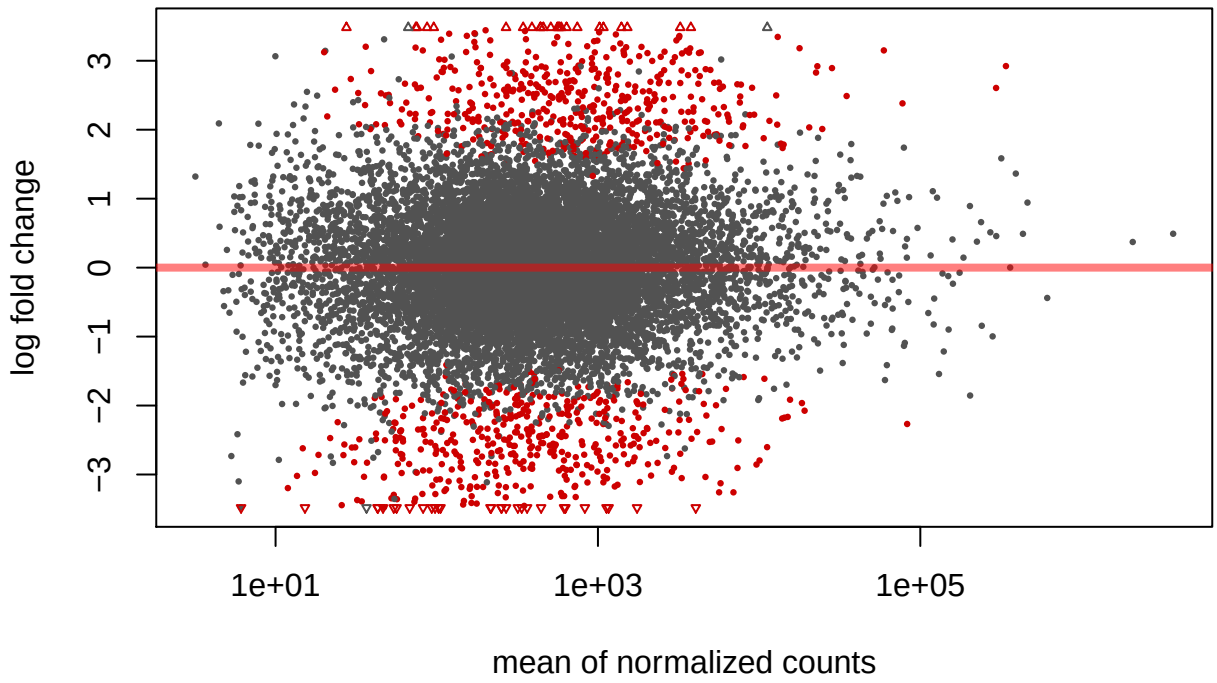


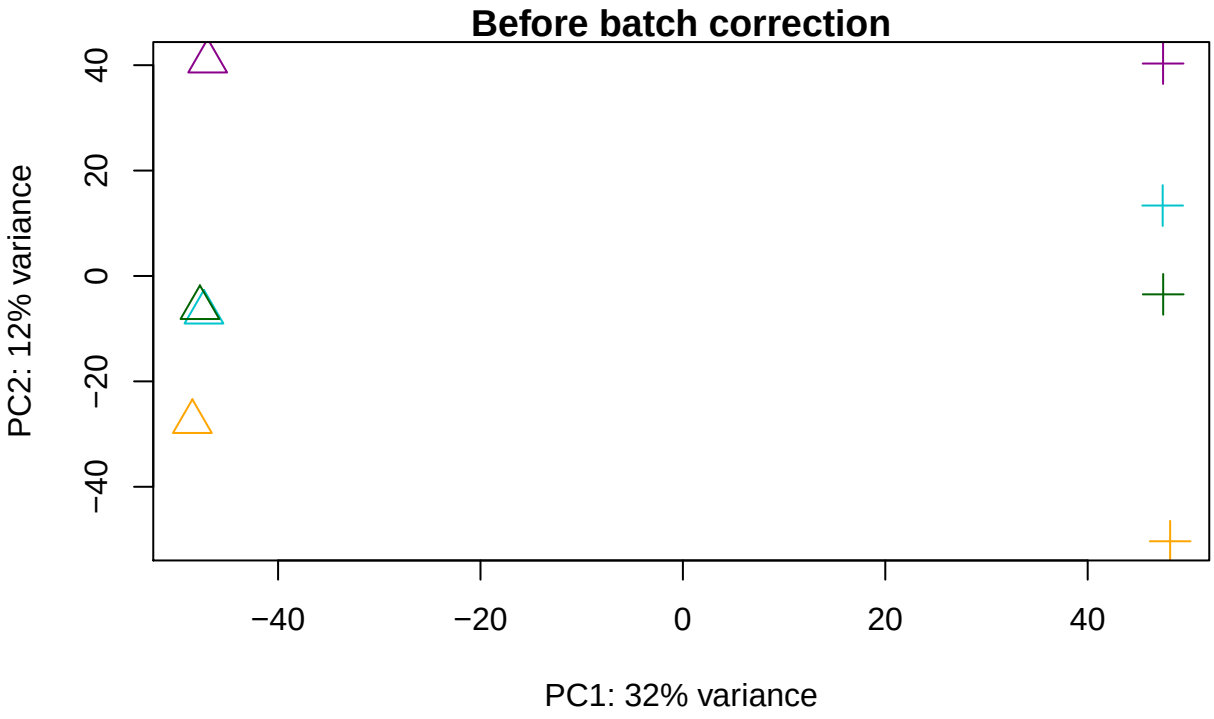

```
## converting counts to integer mode
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```



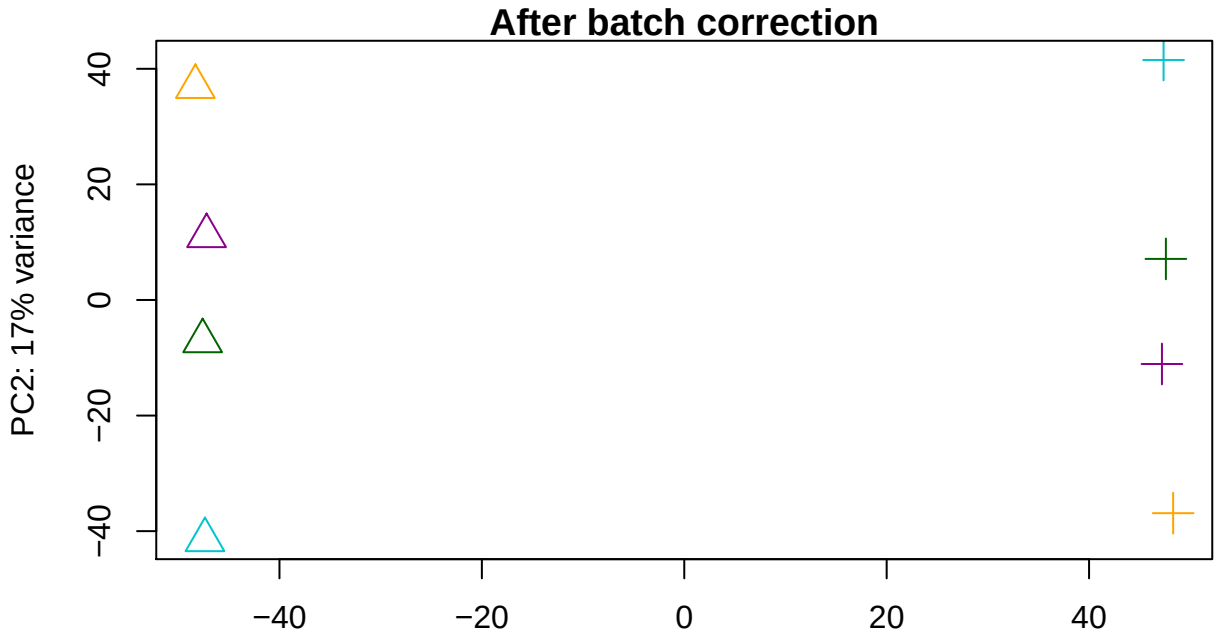

```
## converting counts to integer mode
```

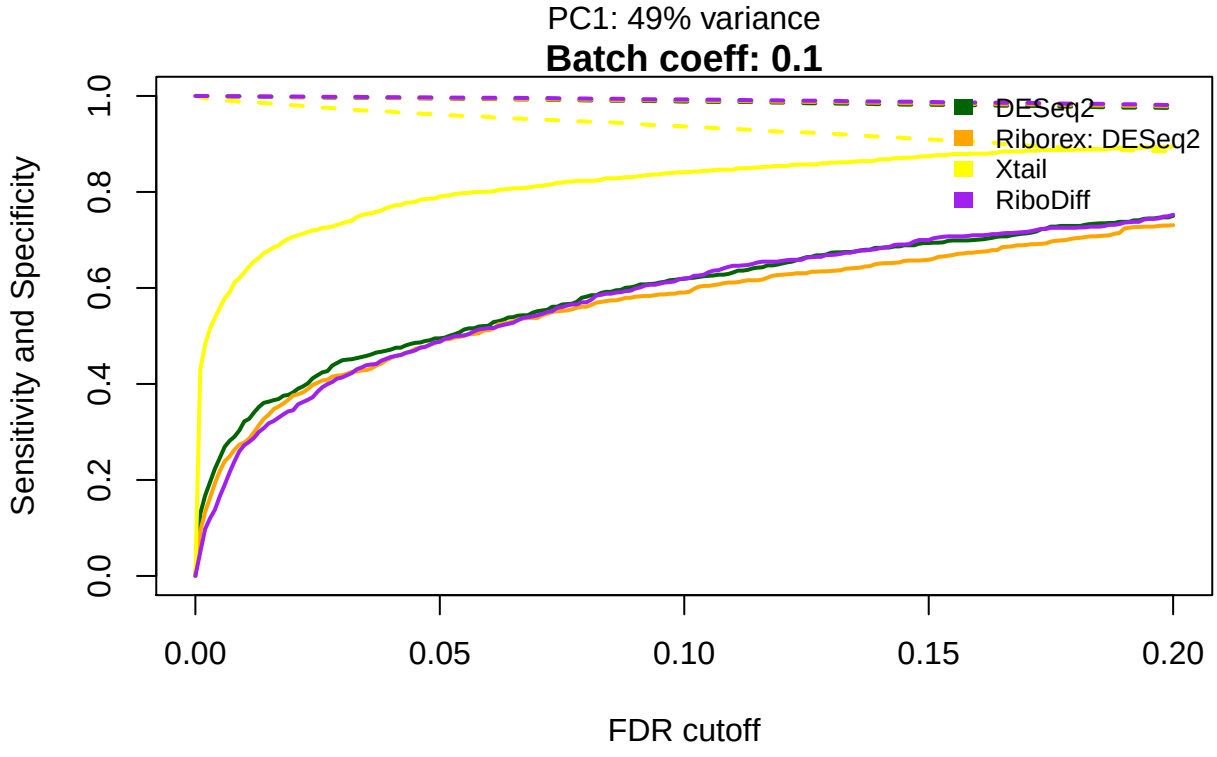


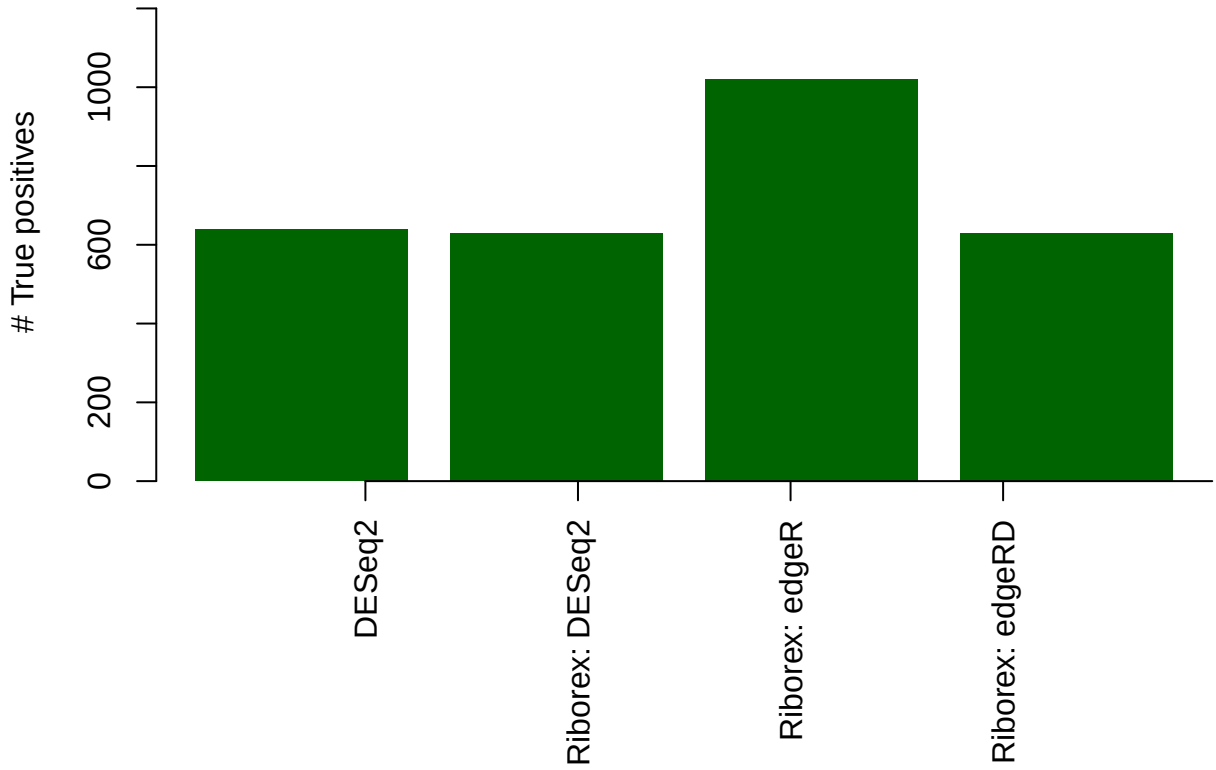



```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
```

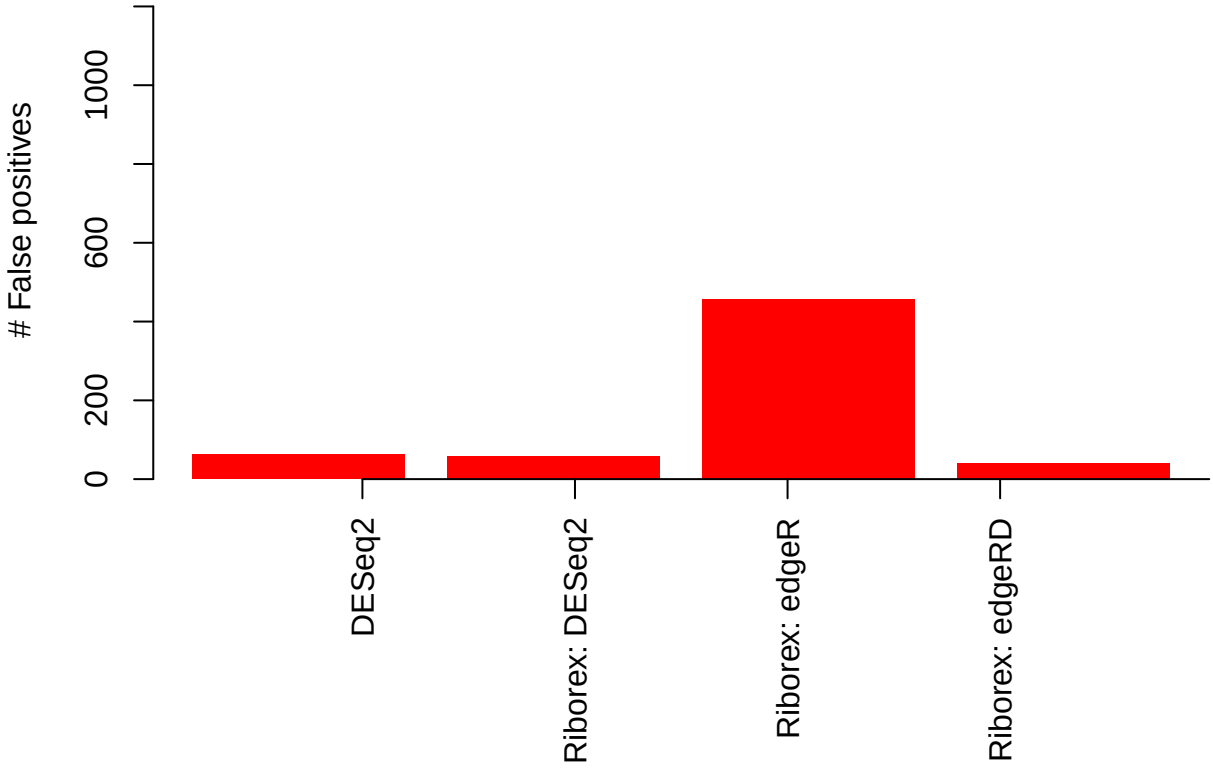
```
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6724
##  log2R: 6264
```



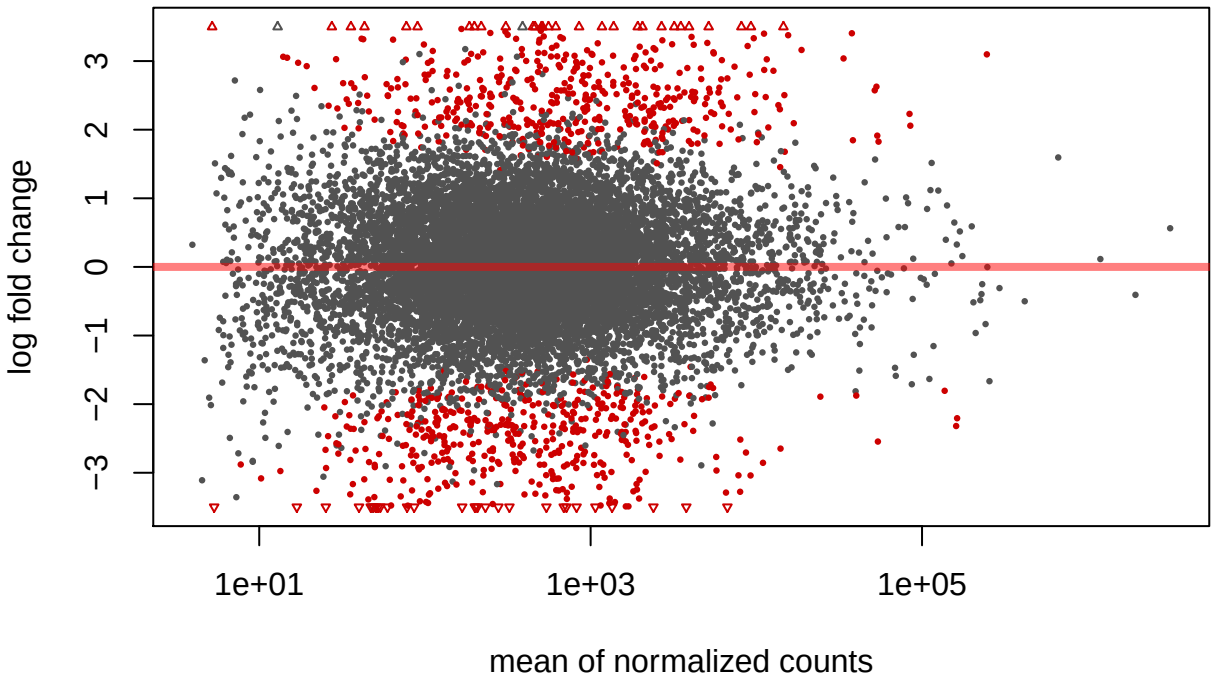


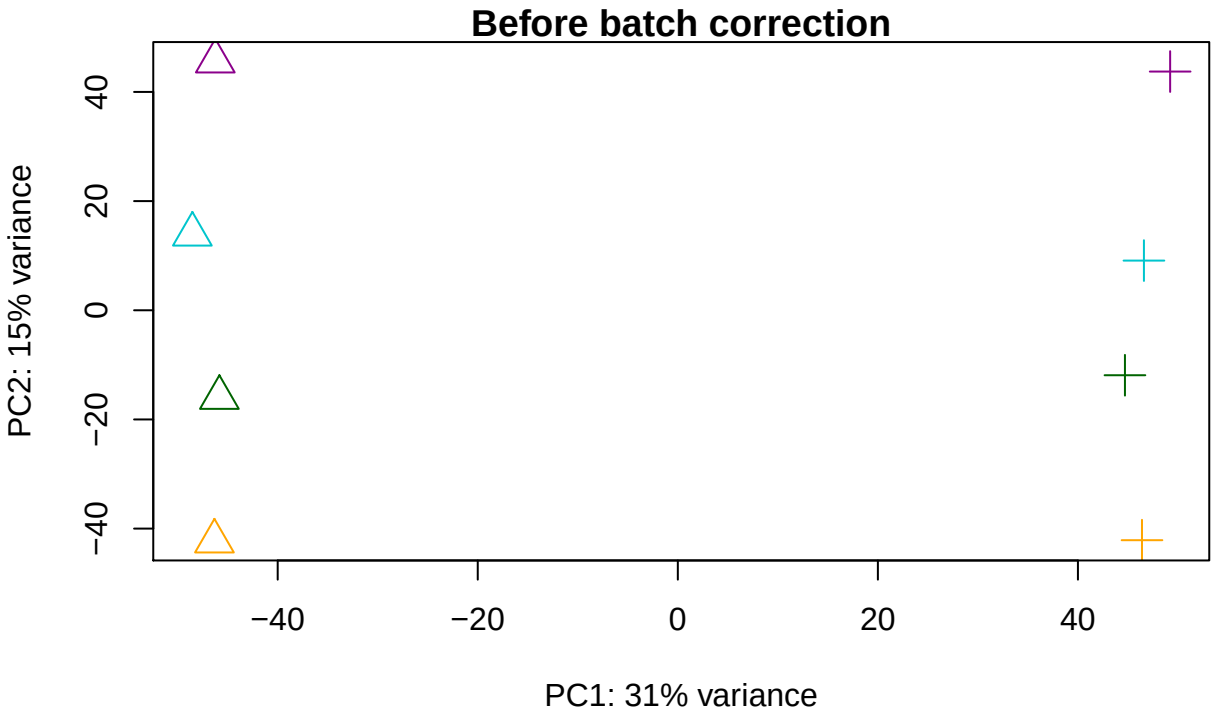



```
## converting counts to integer mode
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```



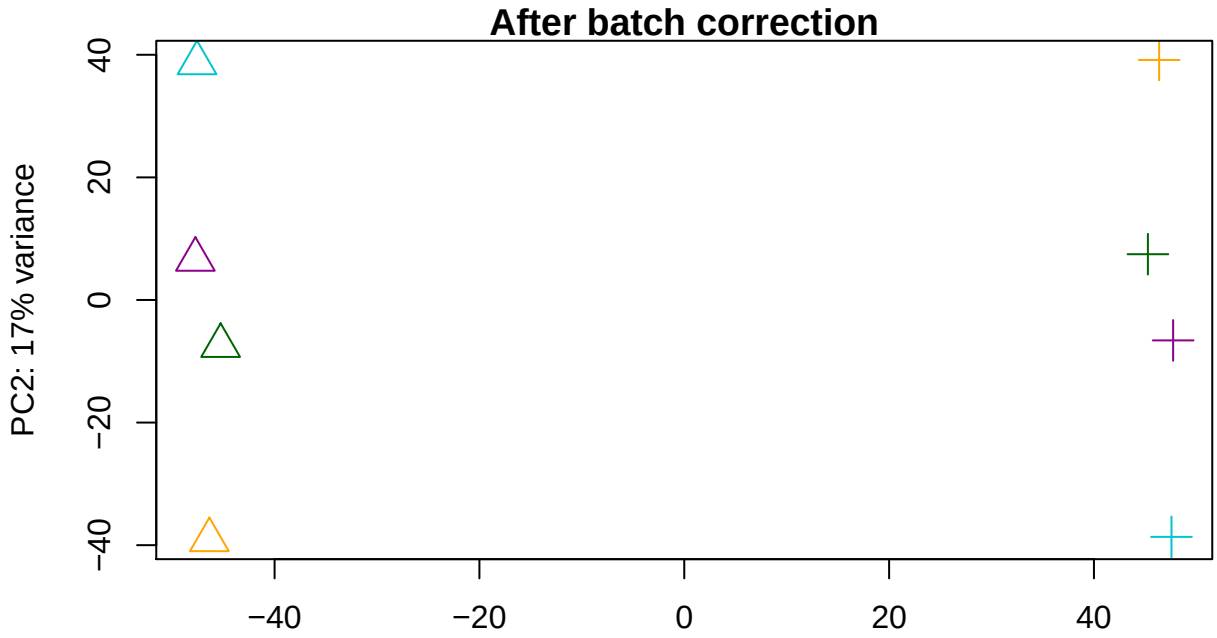

```
## converting counts to integer mode
```

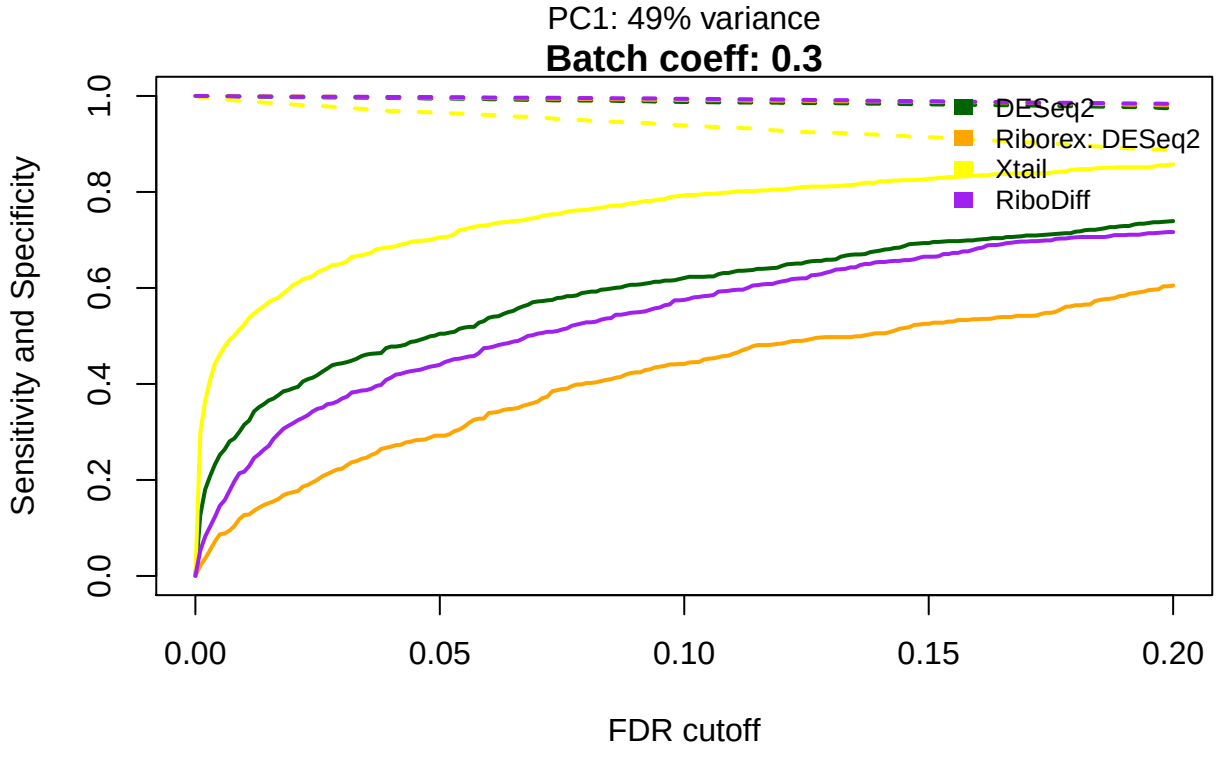


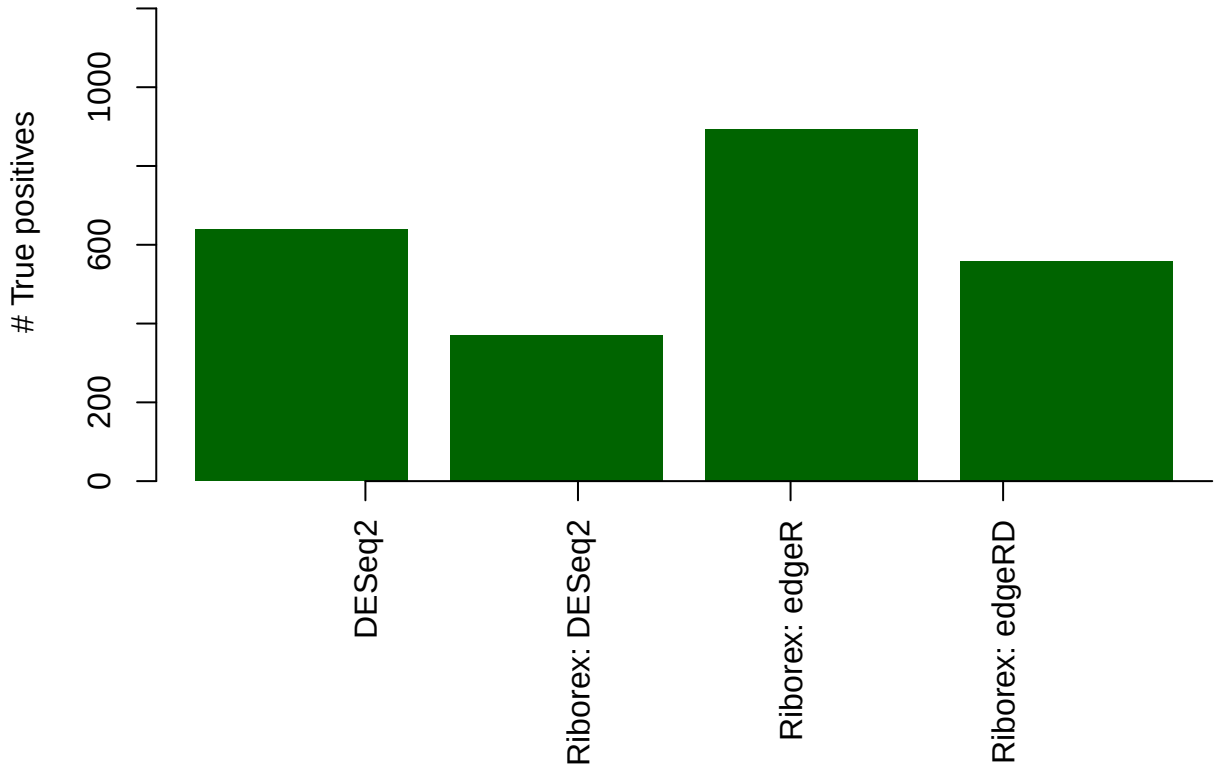



```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
```

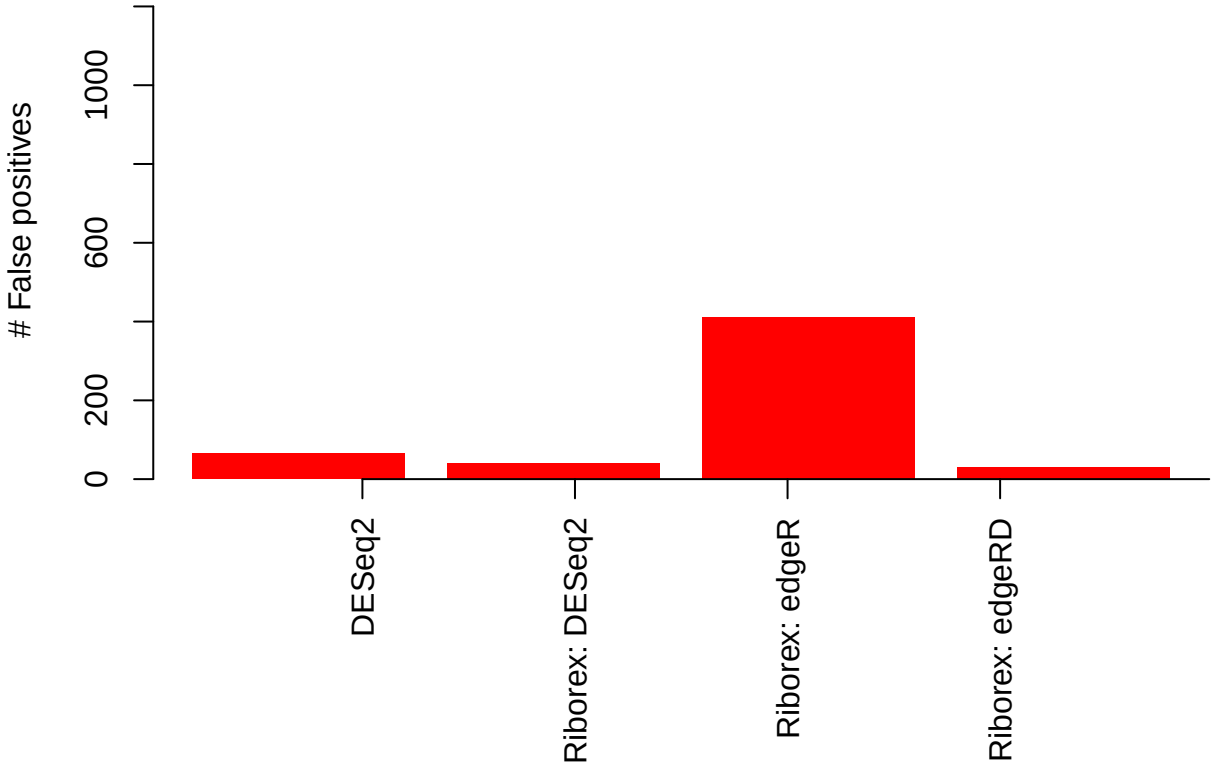
```
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6771
##  log2R: 6192
```



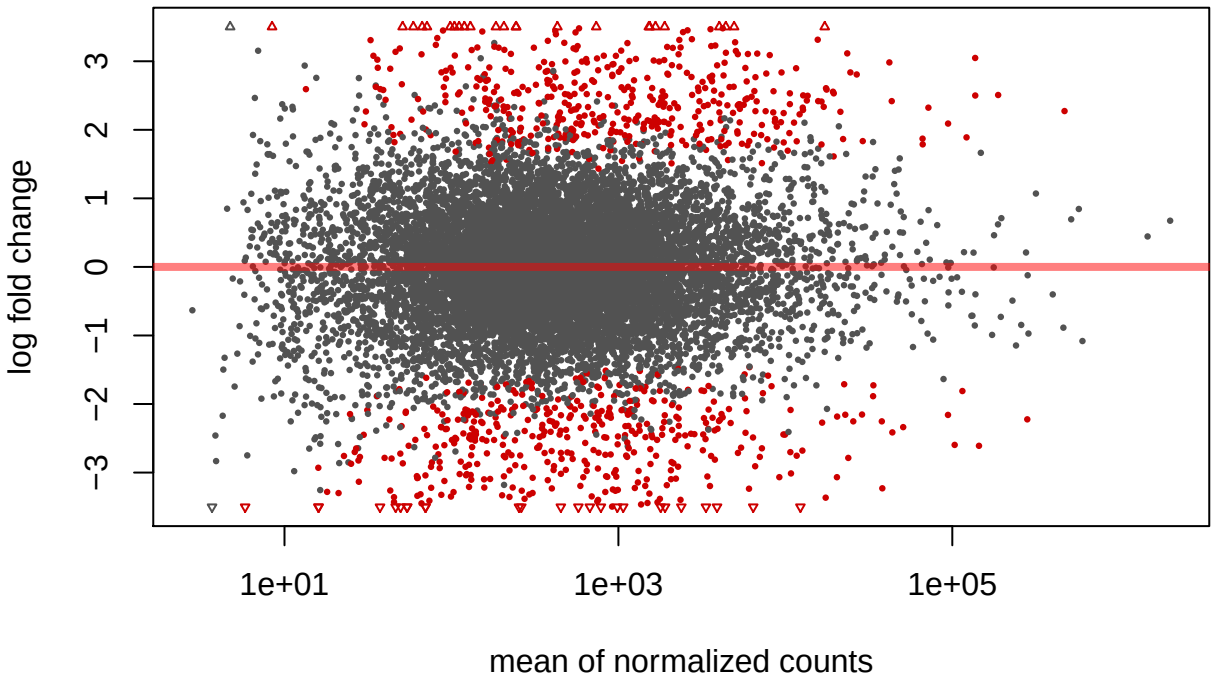


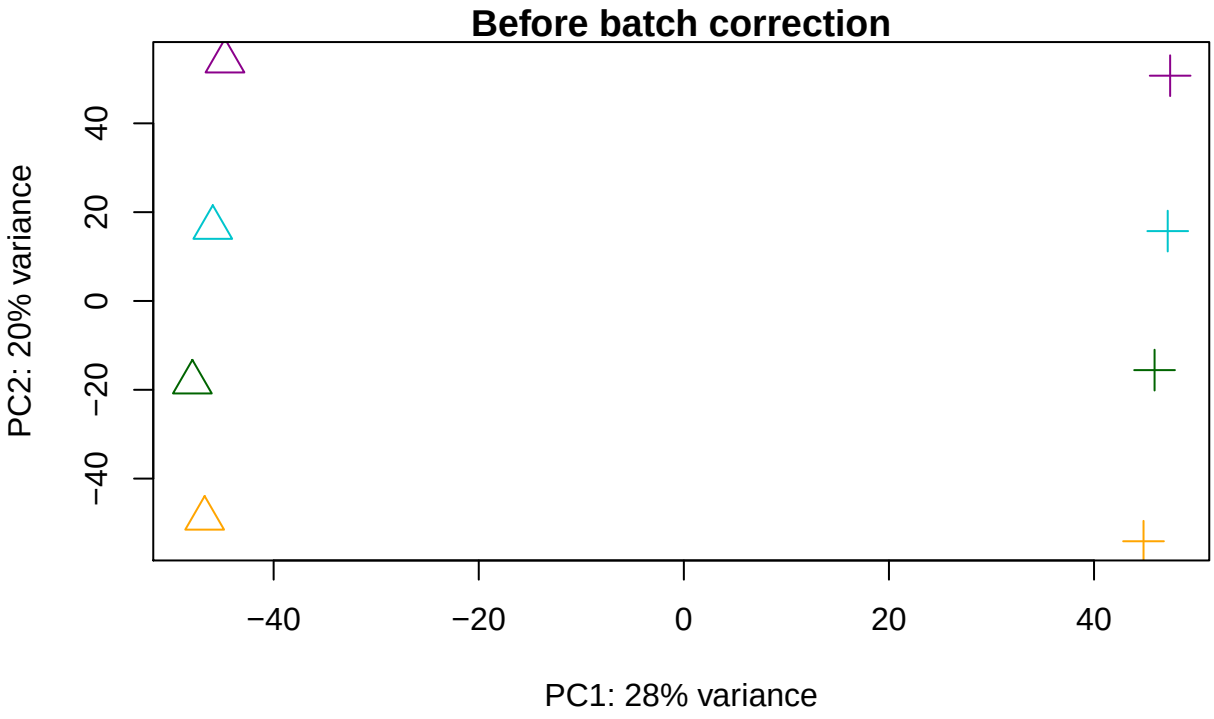



```
## converting counts to integer mode
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```



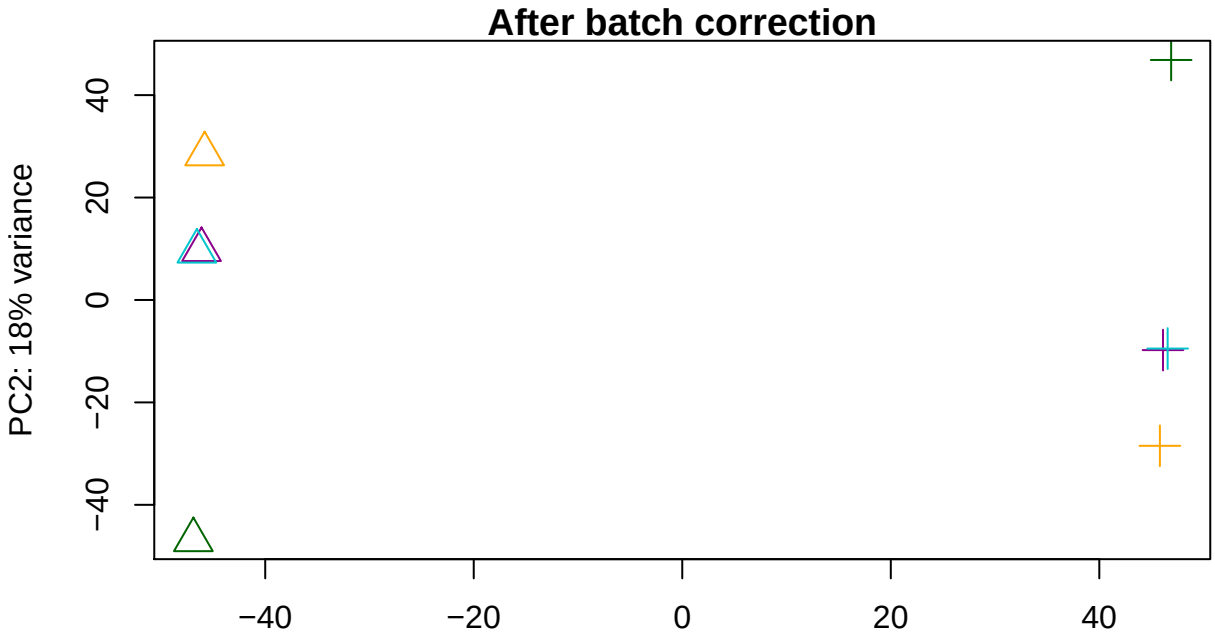

```
## converting counts to integer mode
```

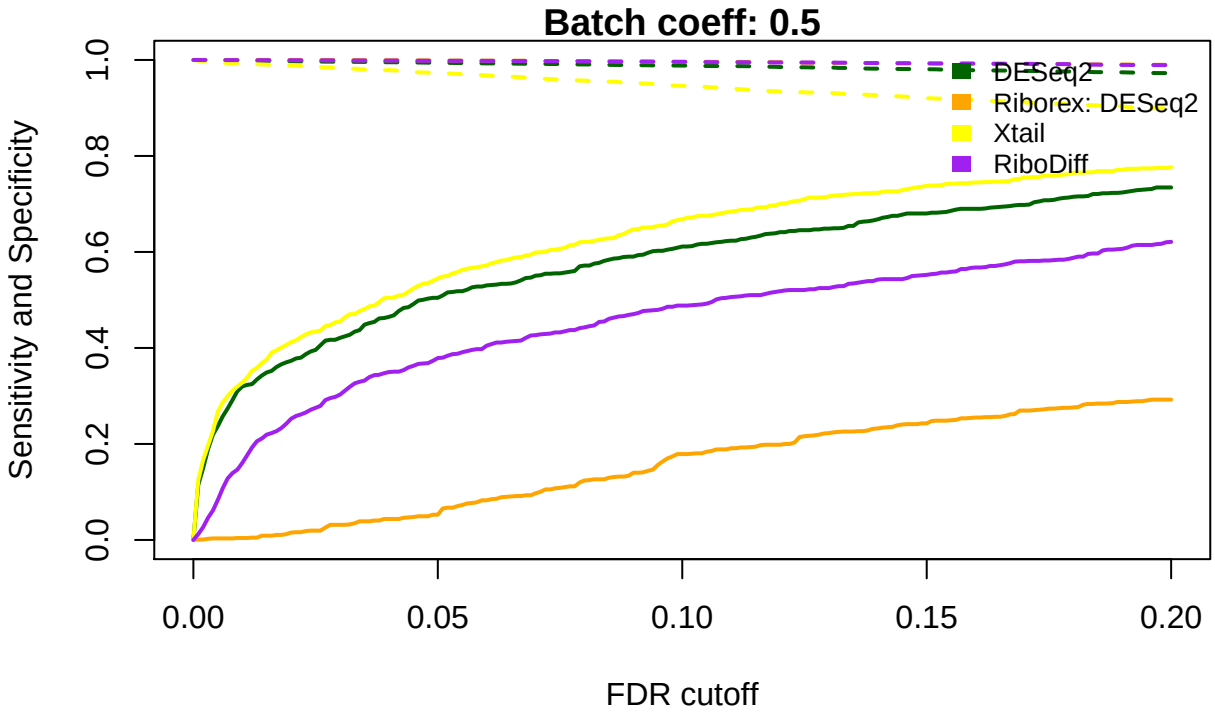


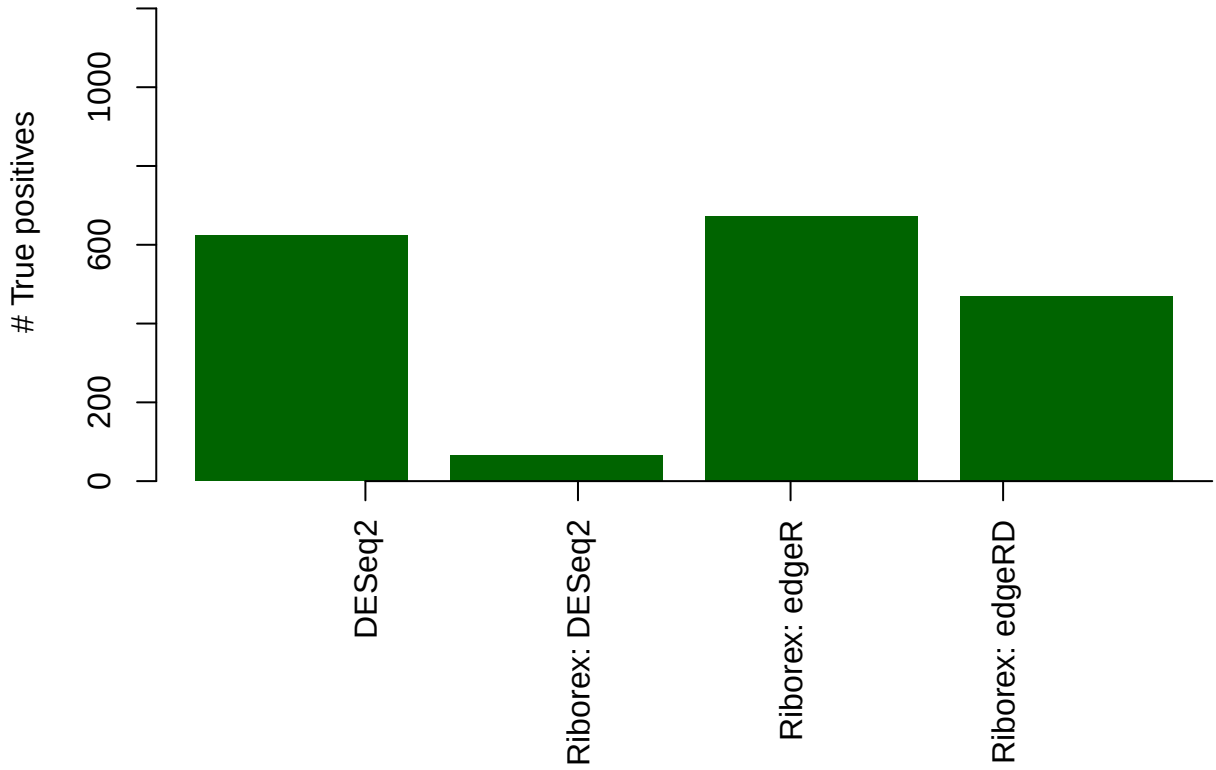



```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
```

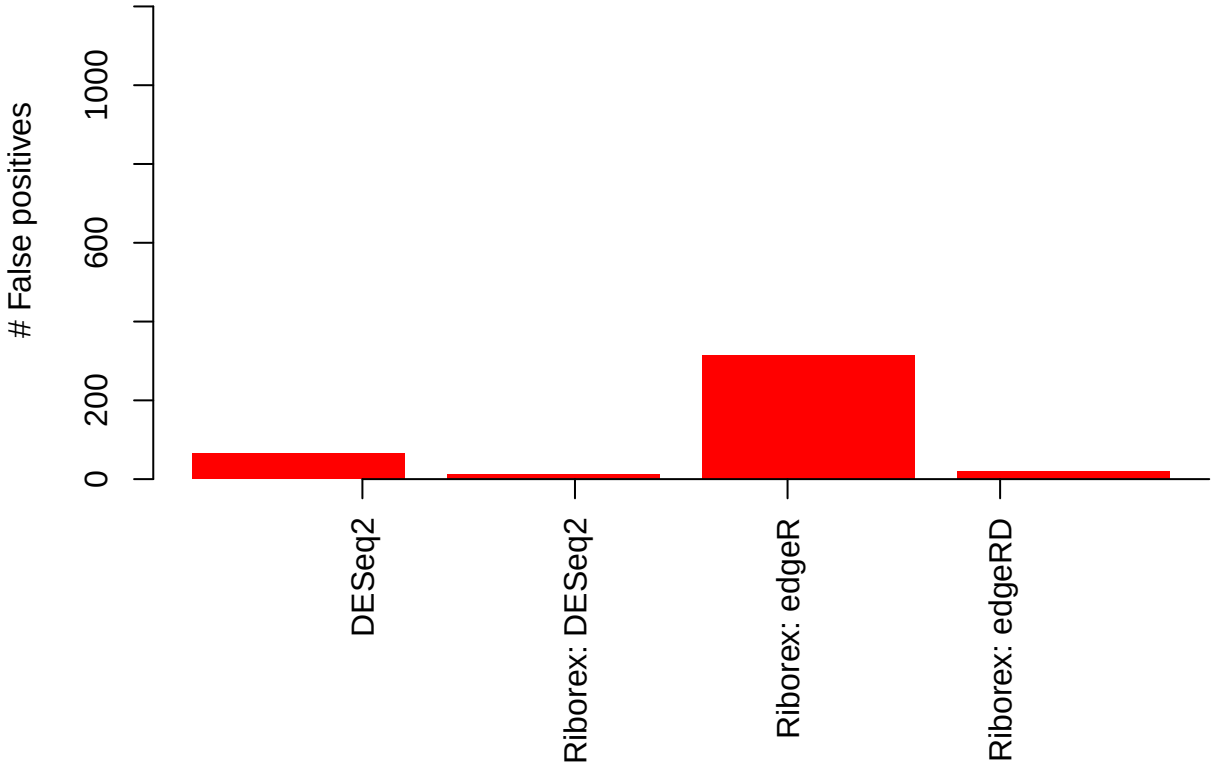
```
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6713
##  log2R: 6223
```



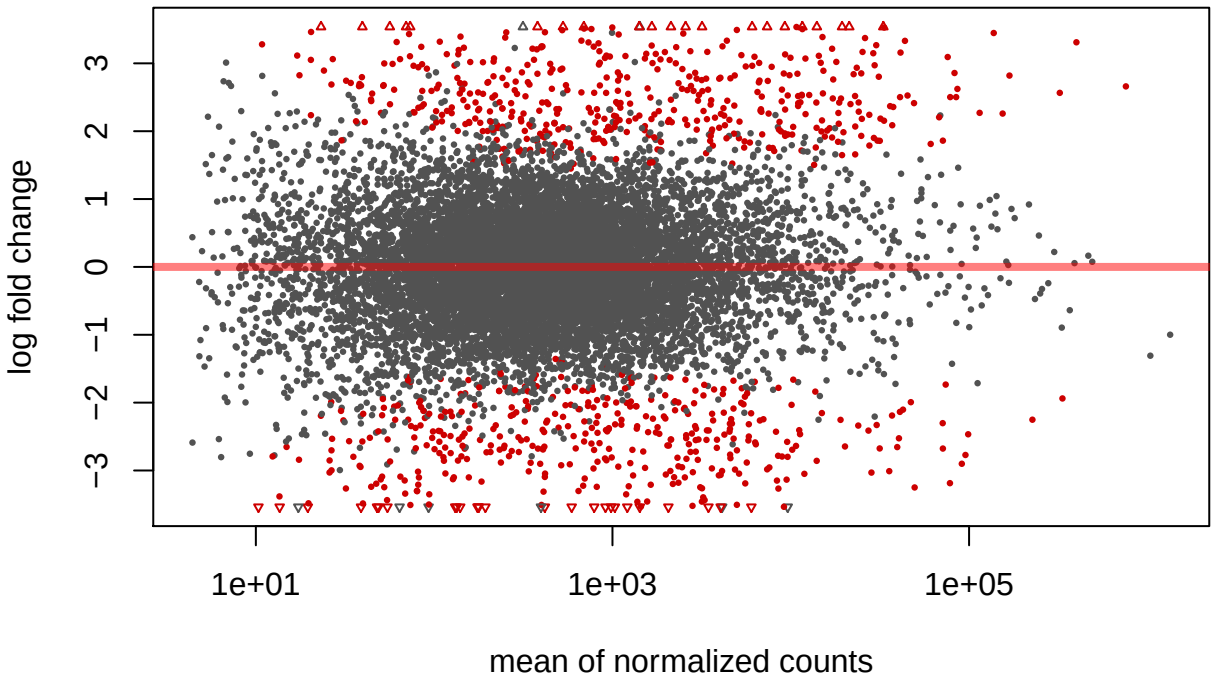


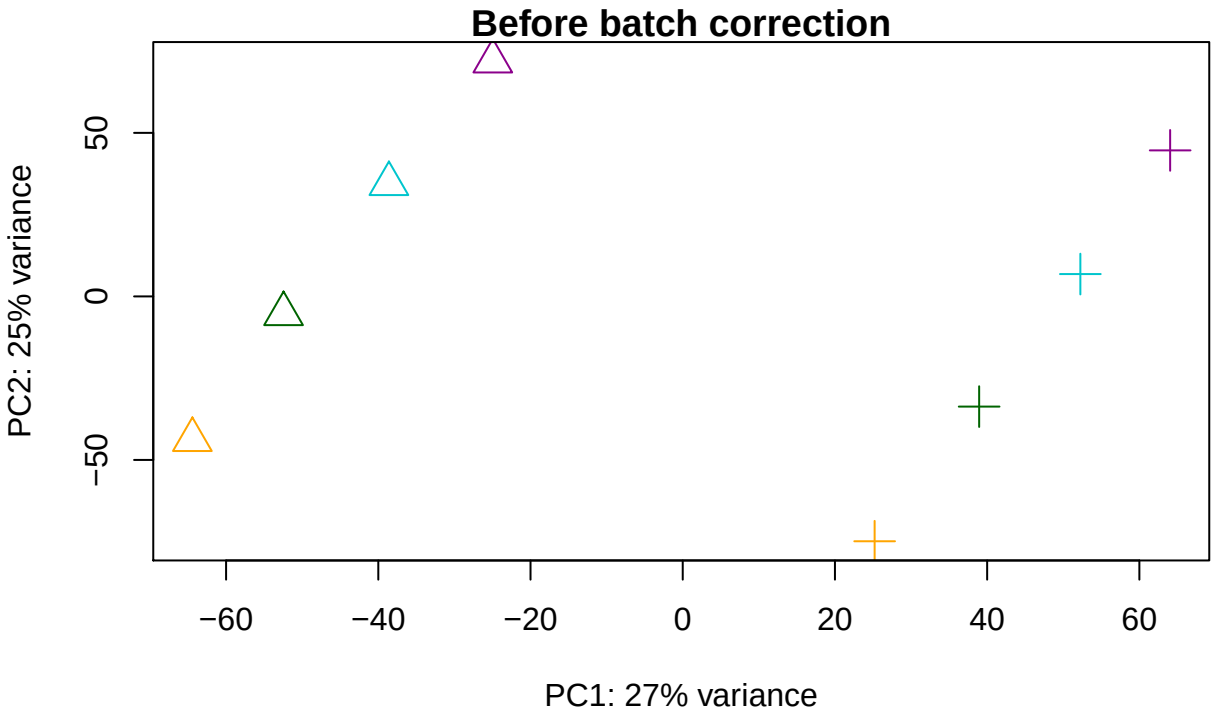



```
## converting counts to integer mode
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```



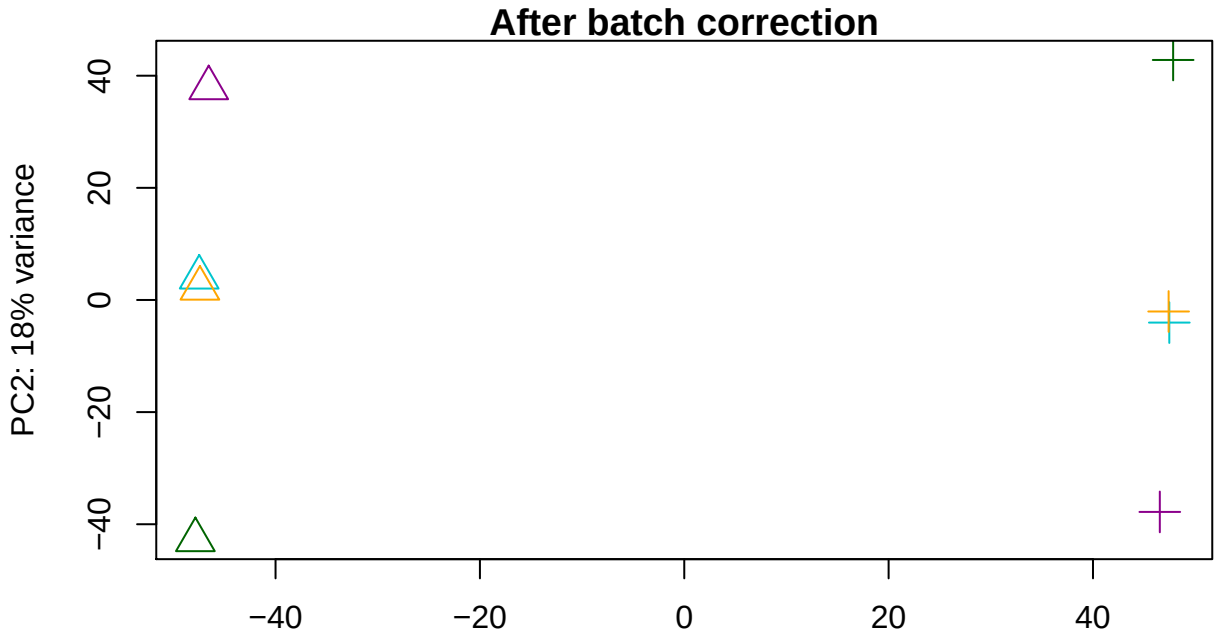

```
## converting counts to integer mode
```

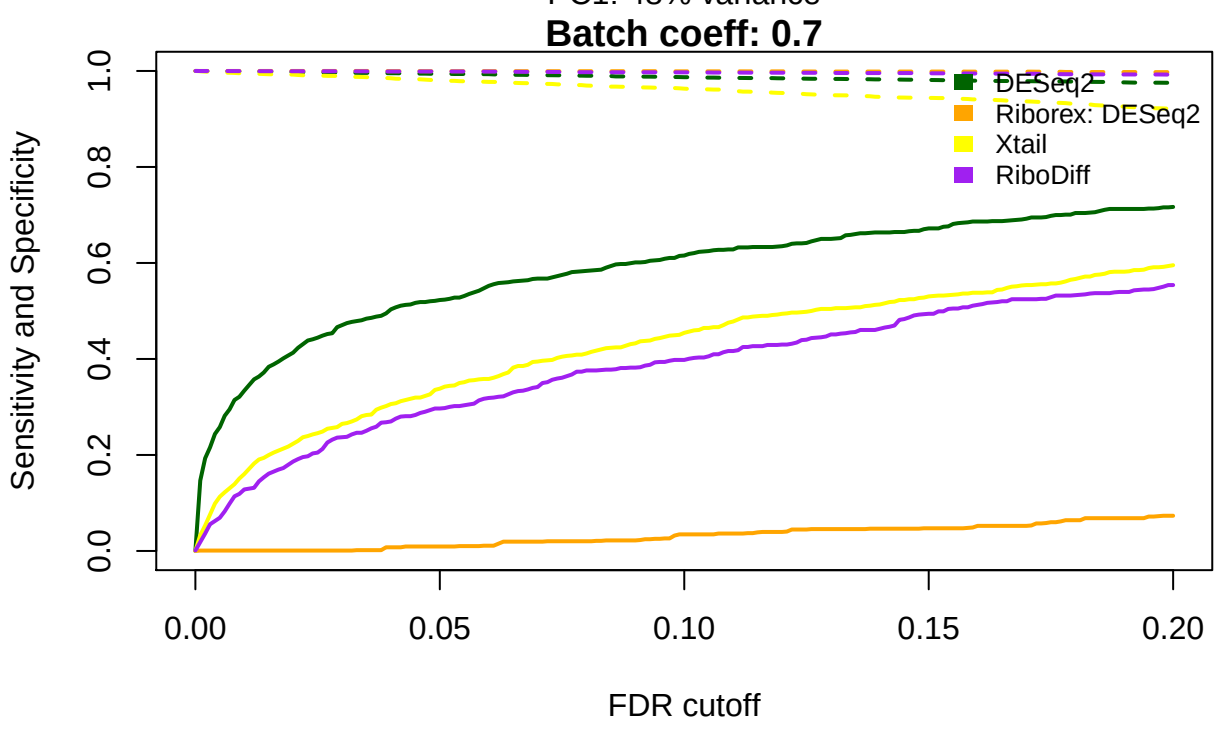


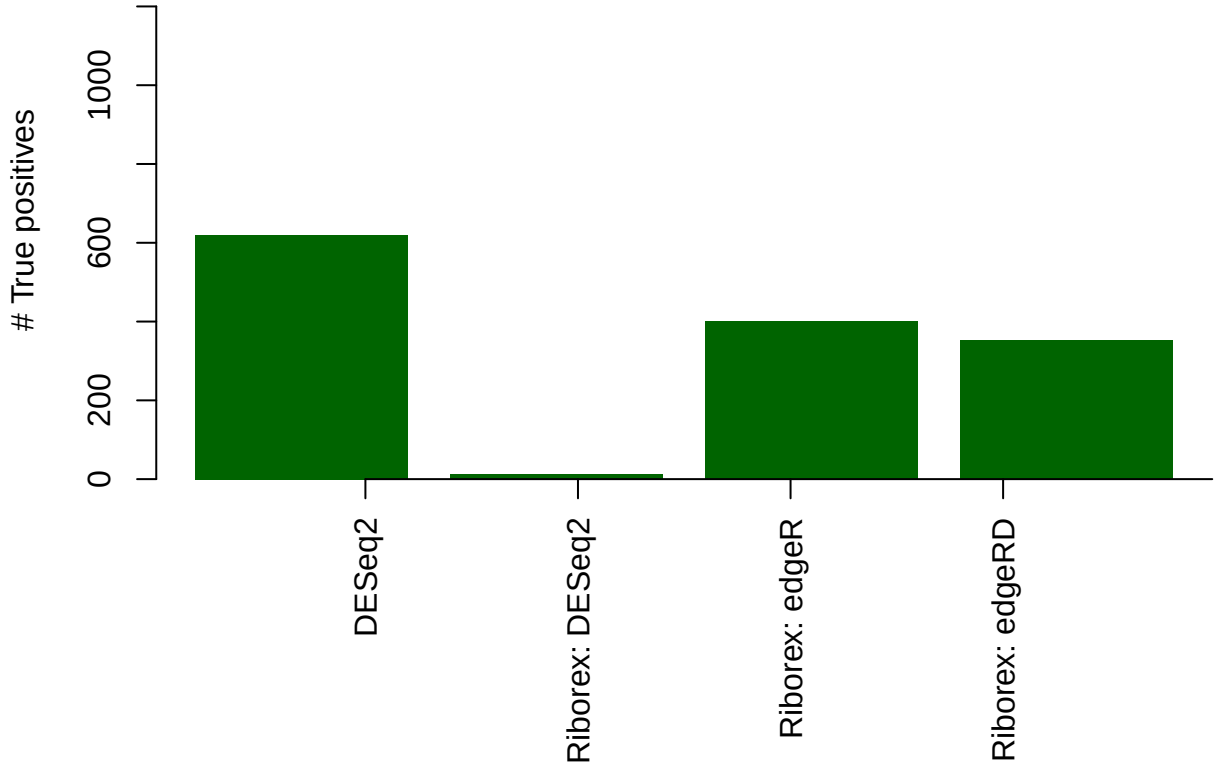



```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
```

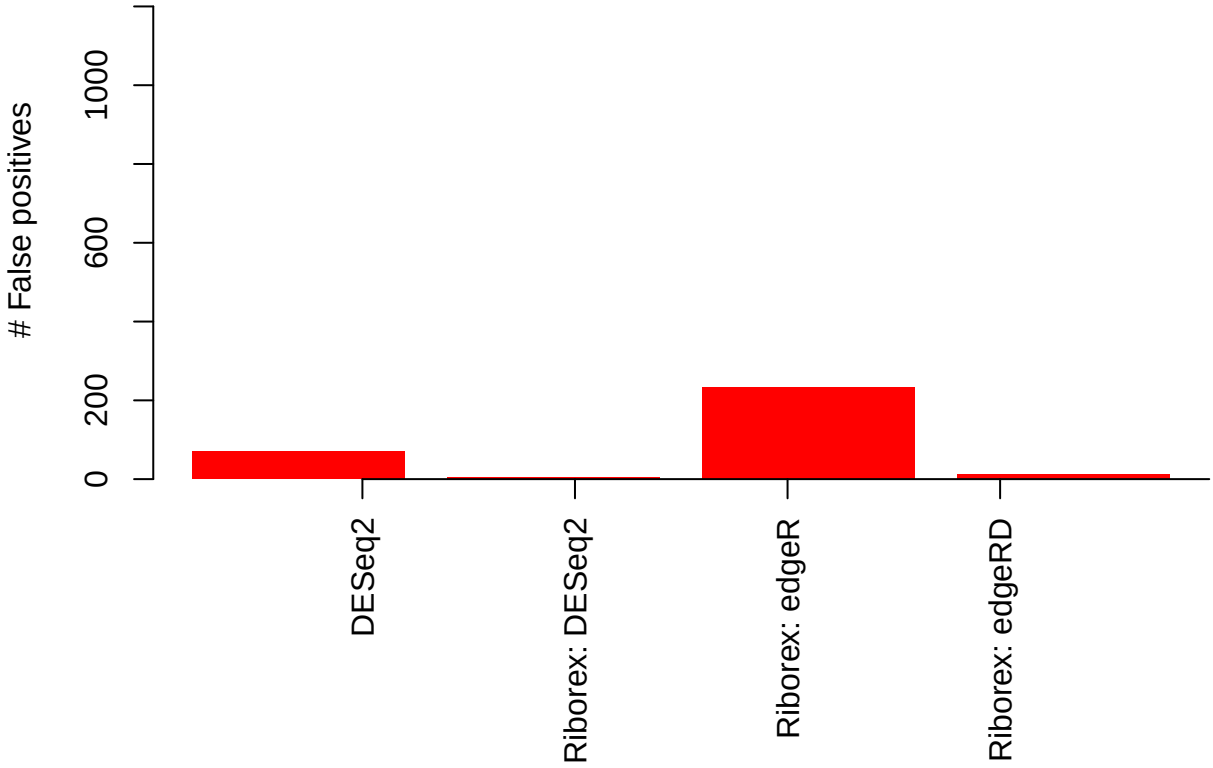
```
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6649
##  log2R: 6226
```



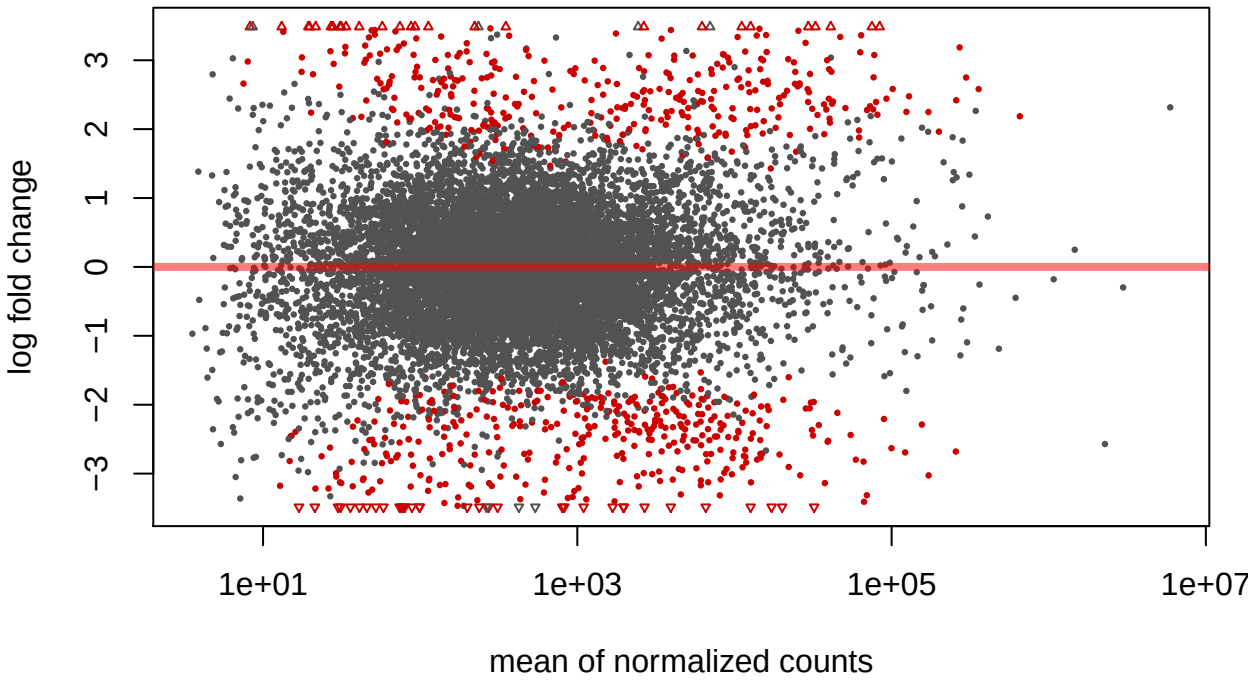


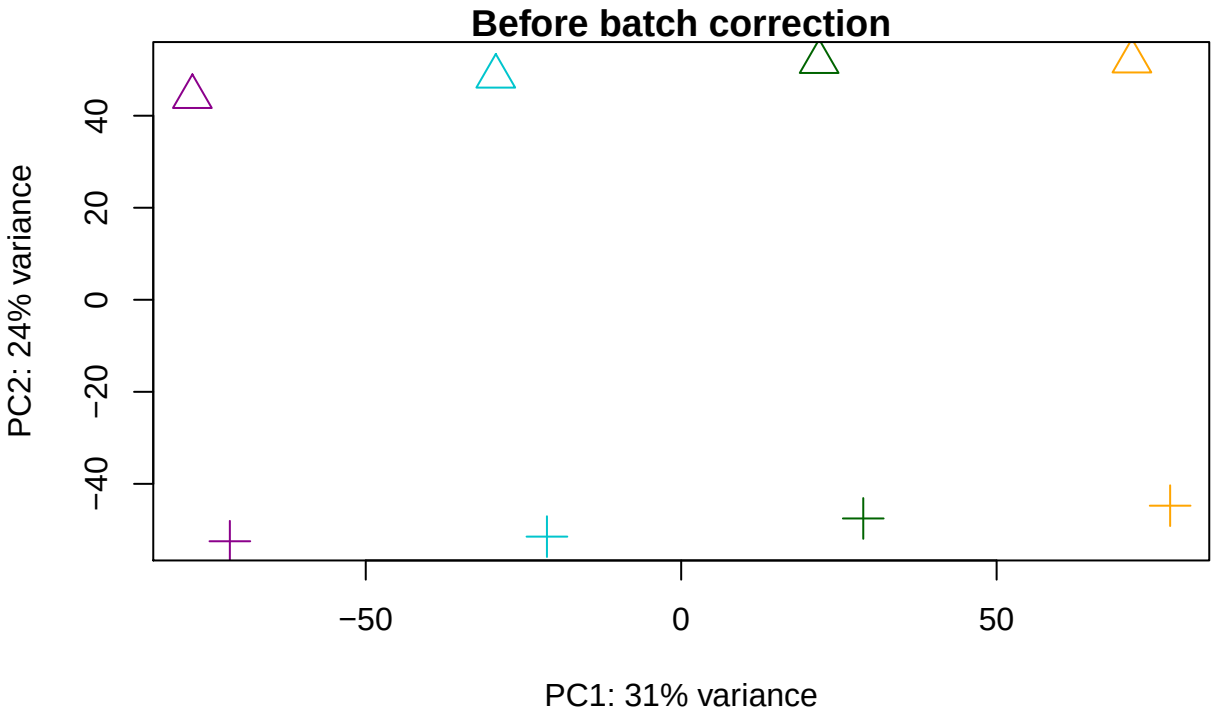



```
## converting counts to integer mode
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```



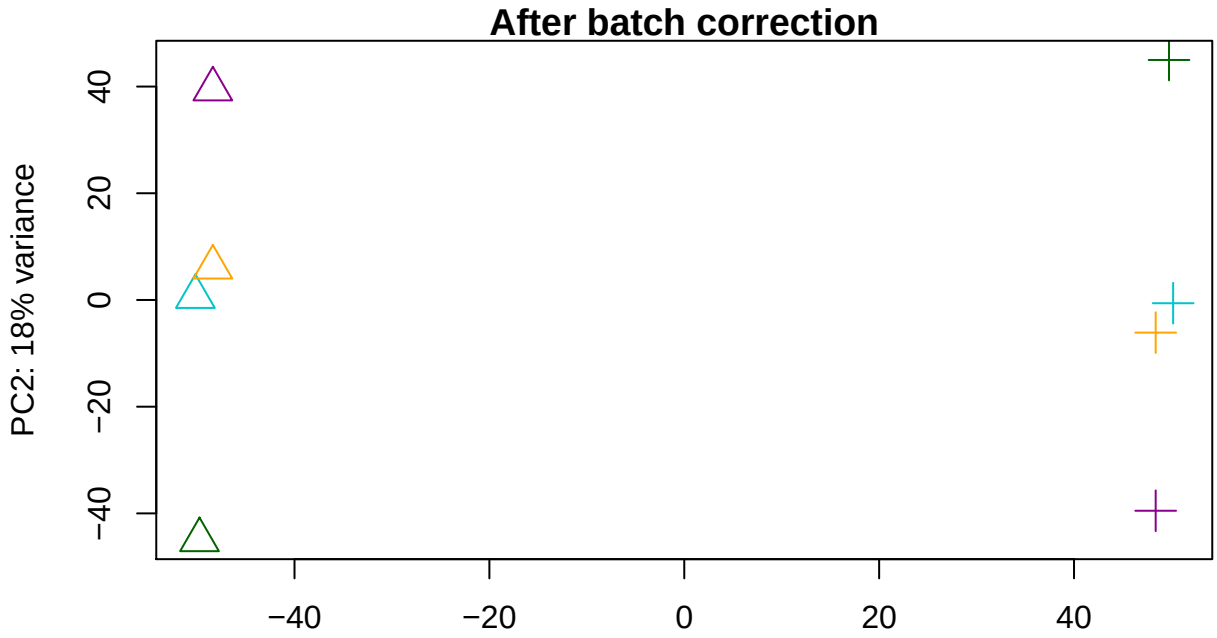

```
## converting counts to integer mode
```

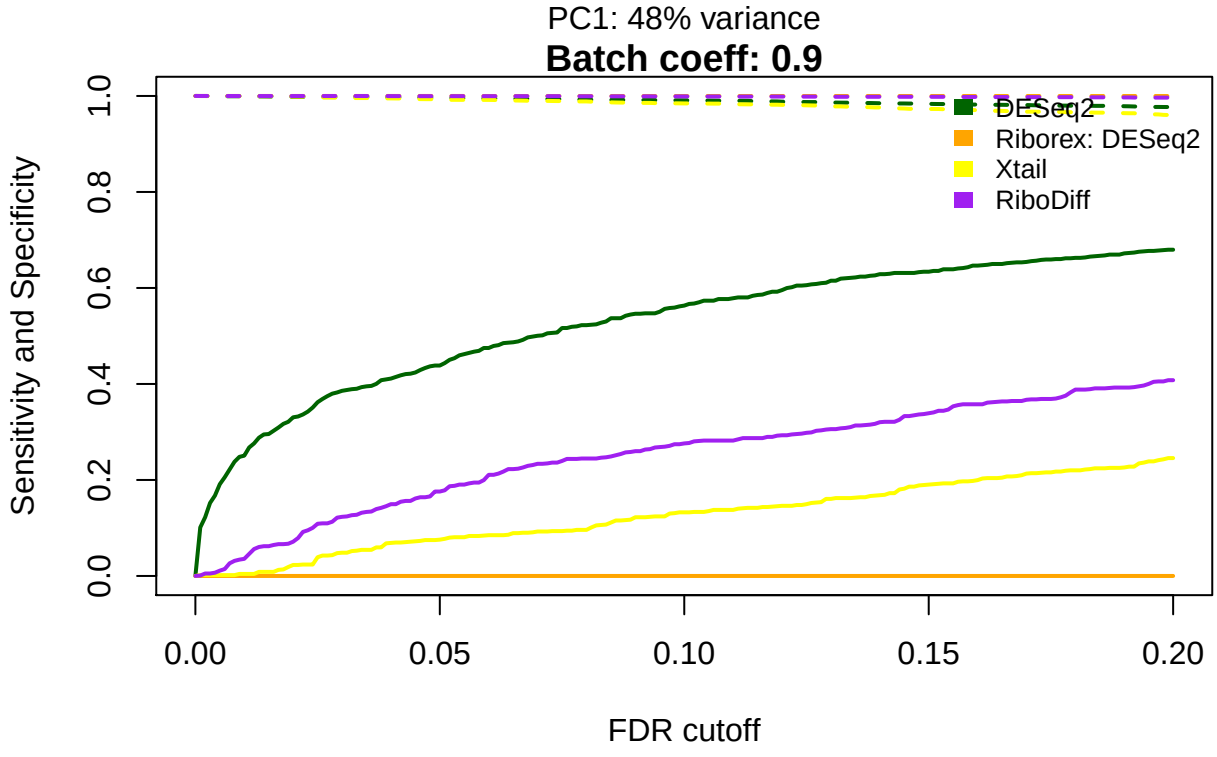


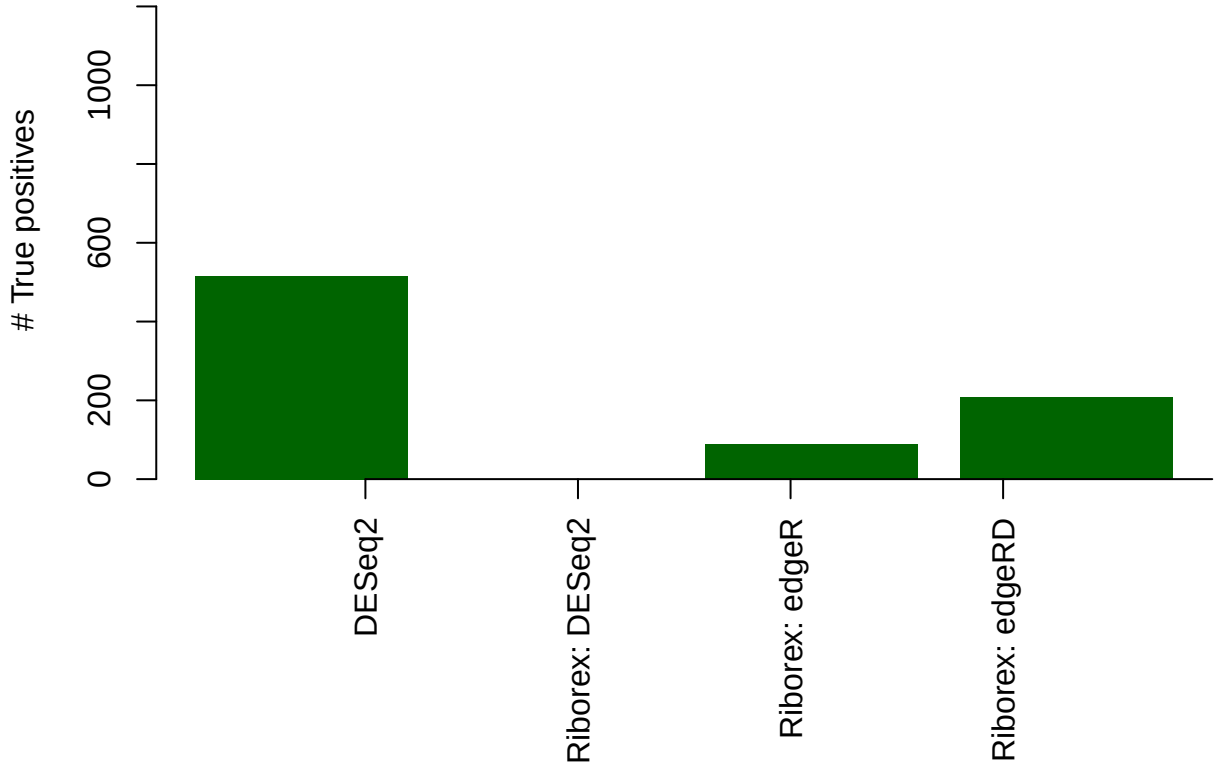



```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
```

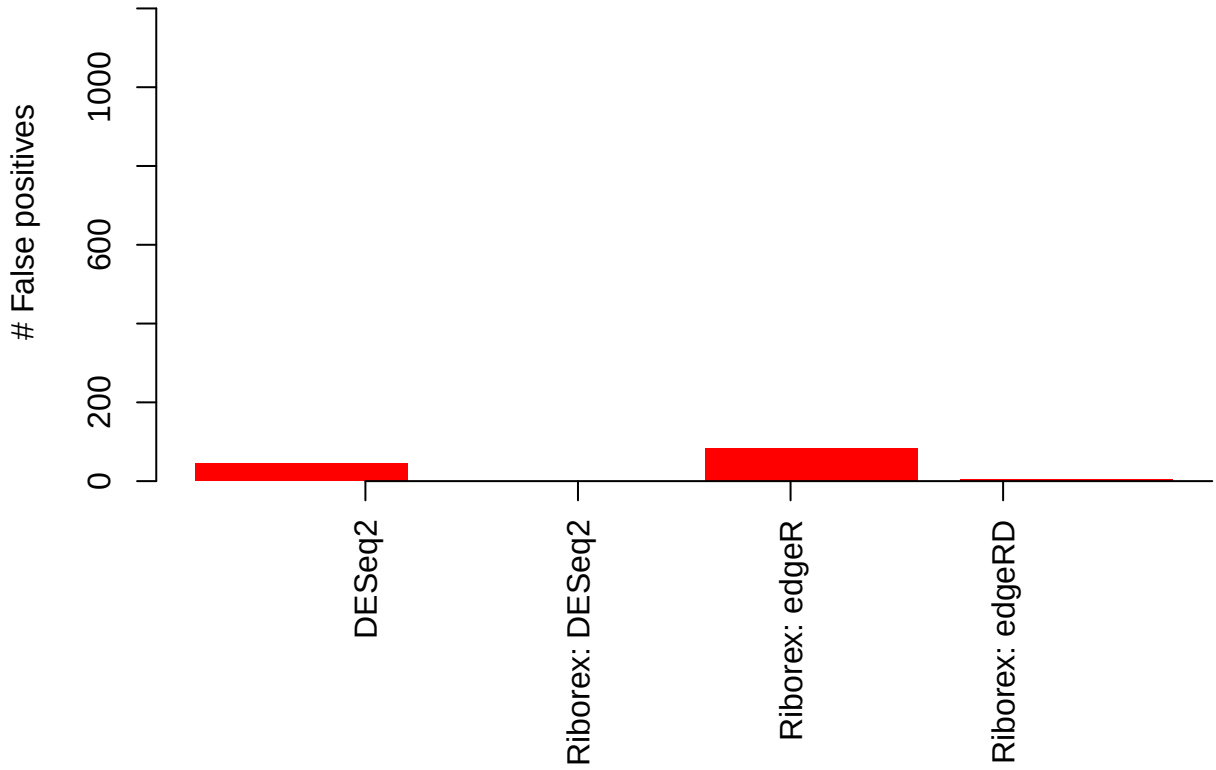
```
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6663
##  log2R: 6211
```



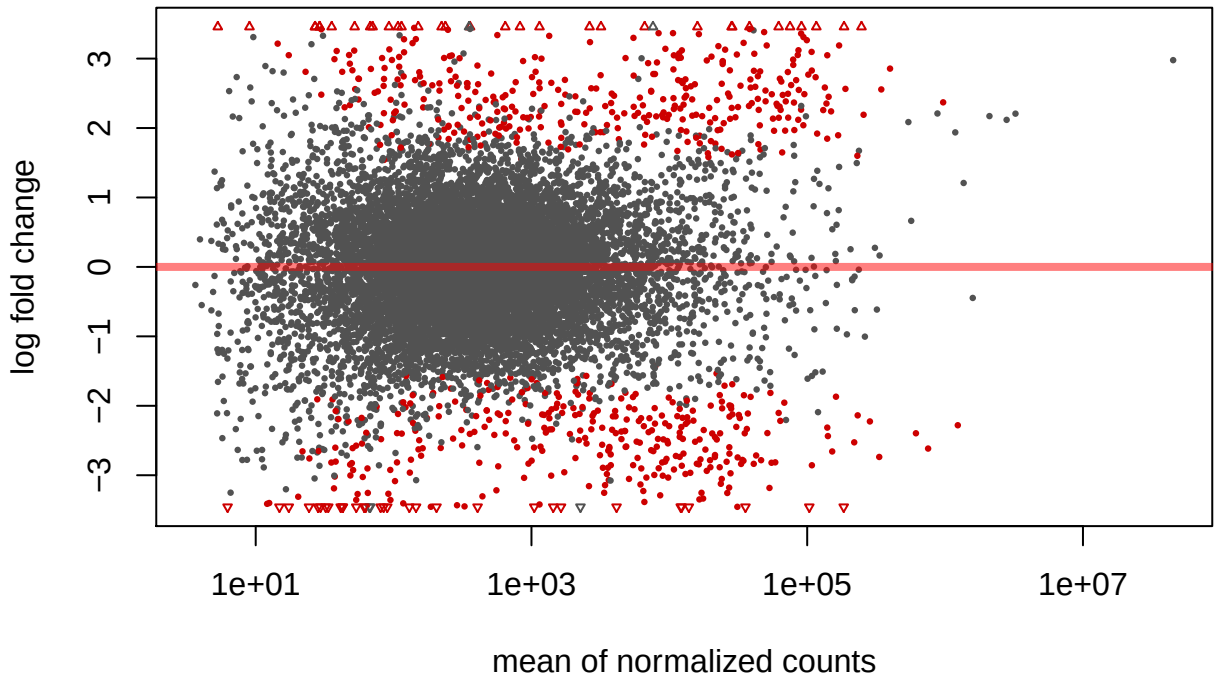


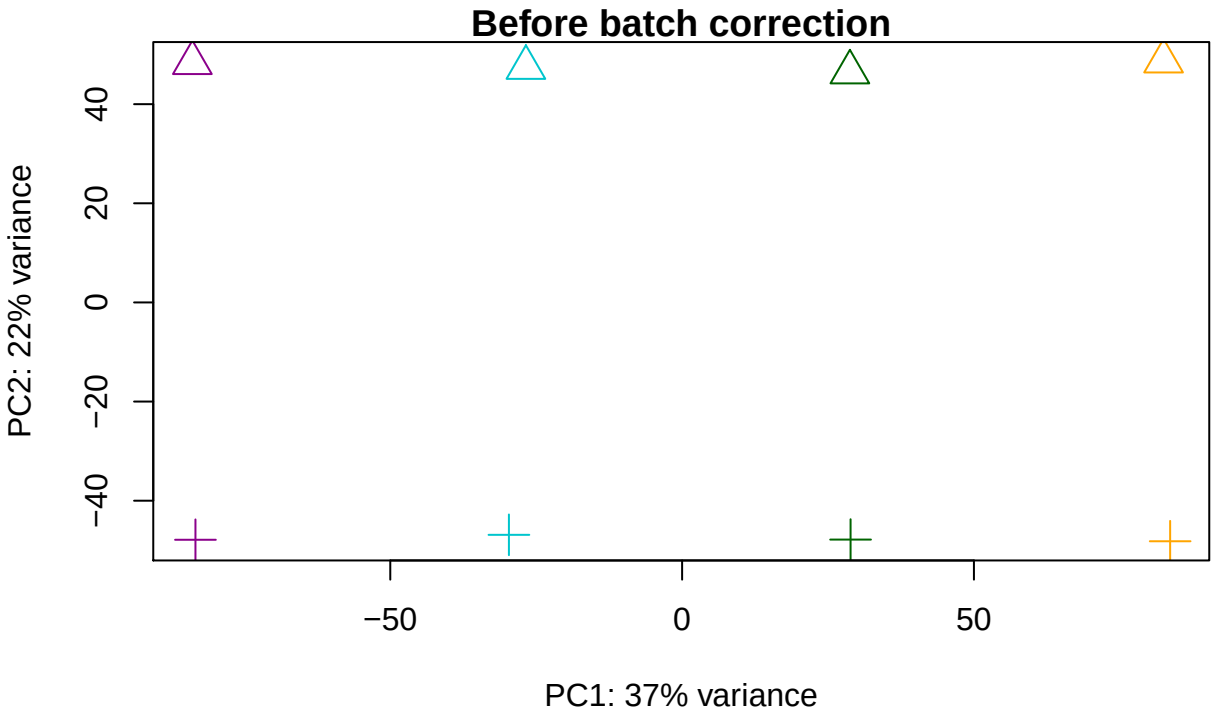



```
## converting counts to integer mode
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```



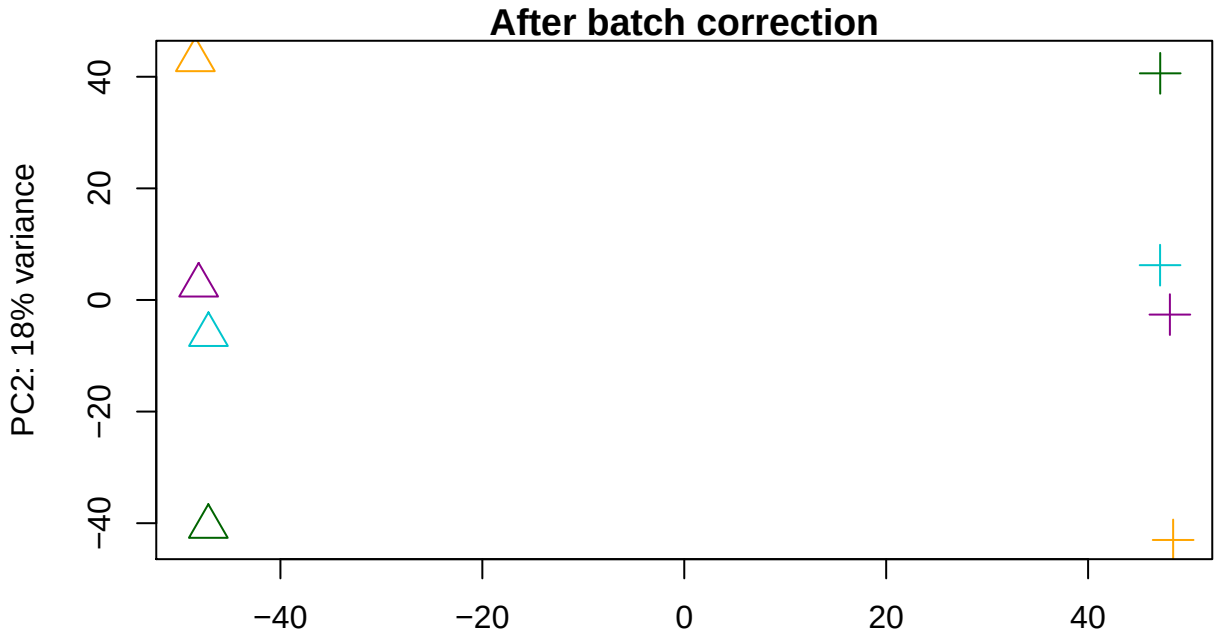

```
## converting counts to integer mode
```

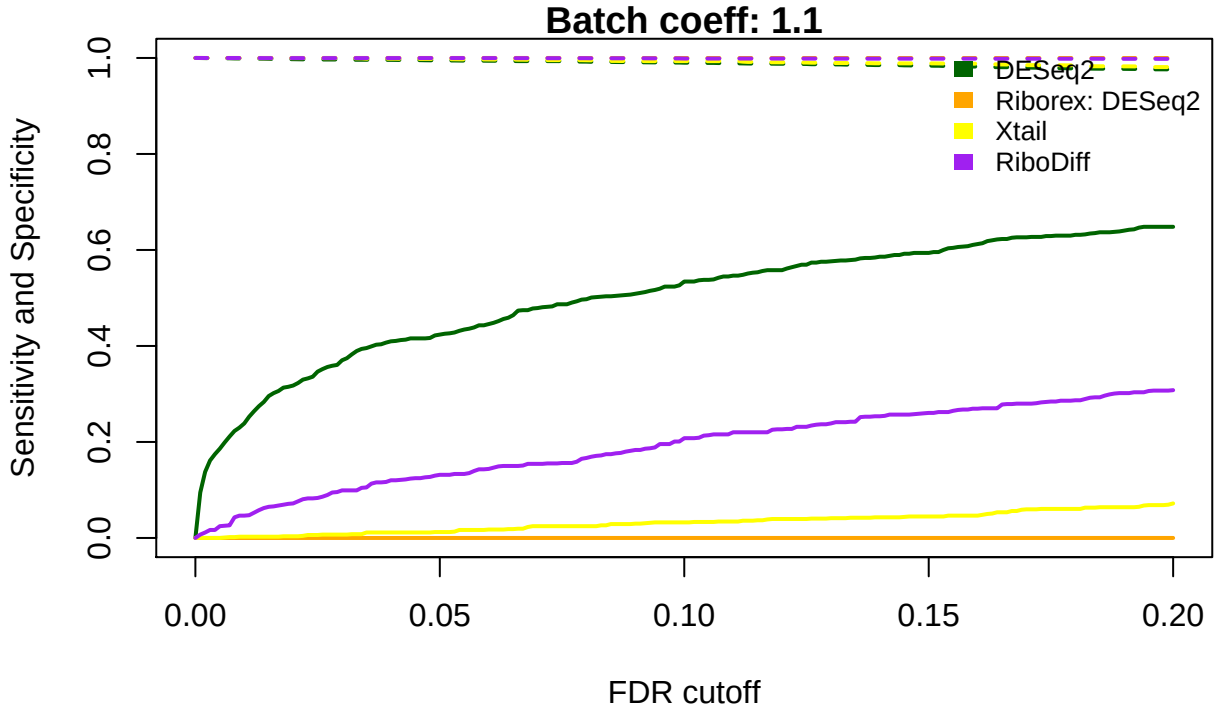


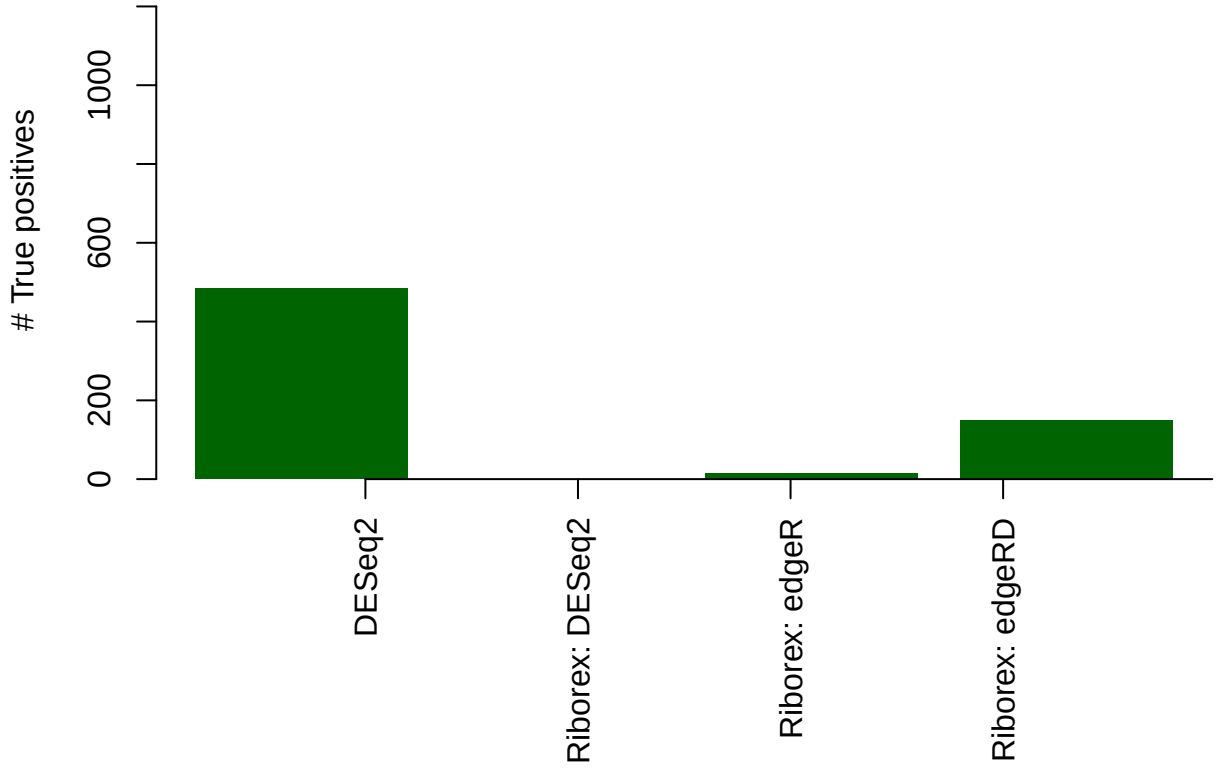



```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
```

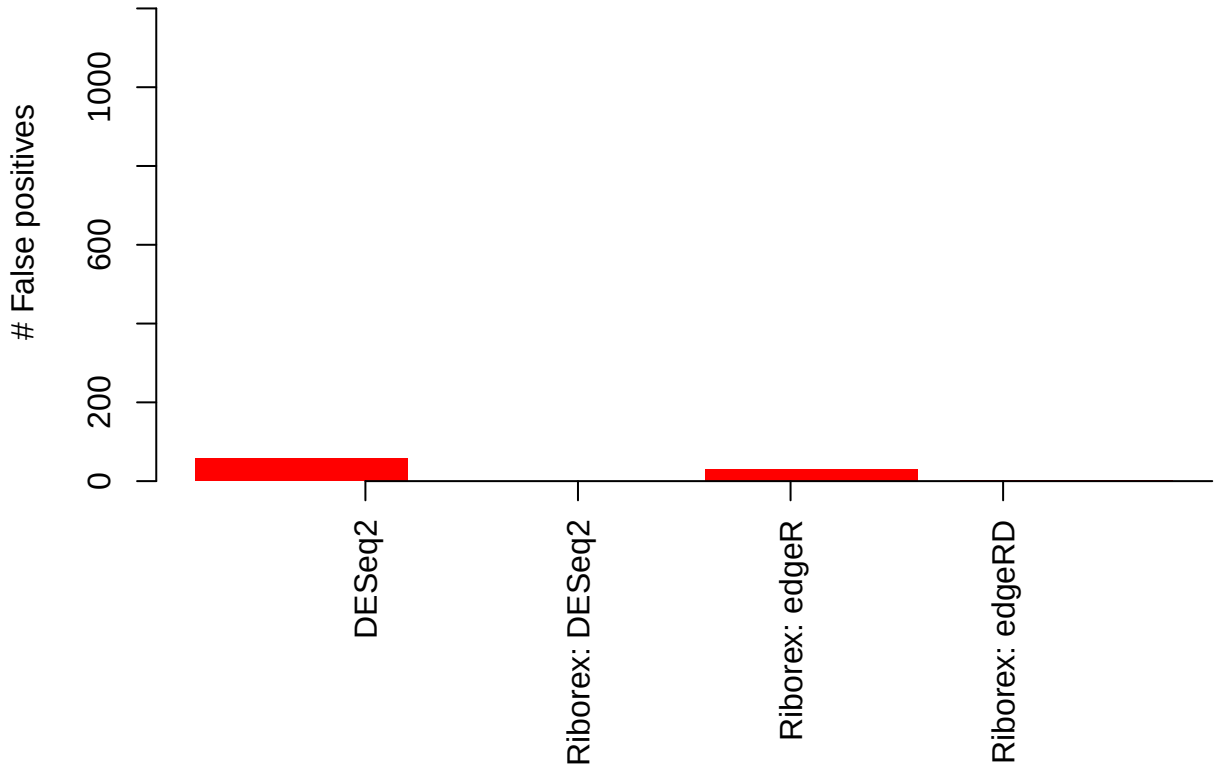
```
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6703
##  log2R: 6138
```



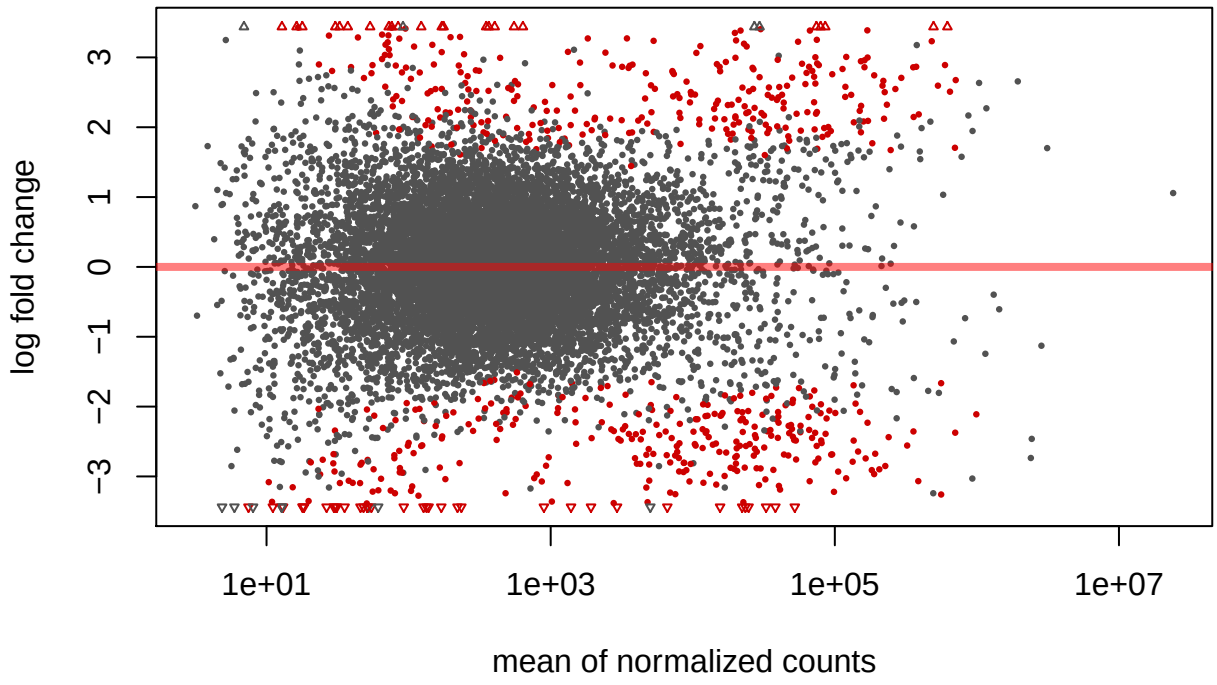


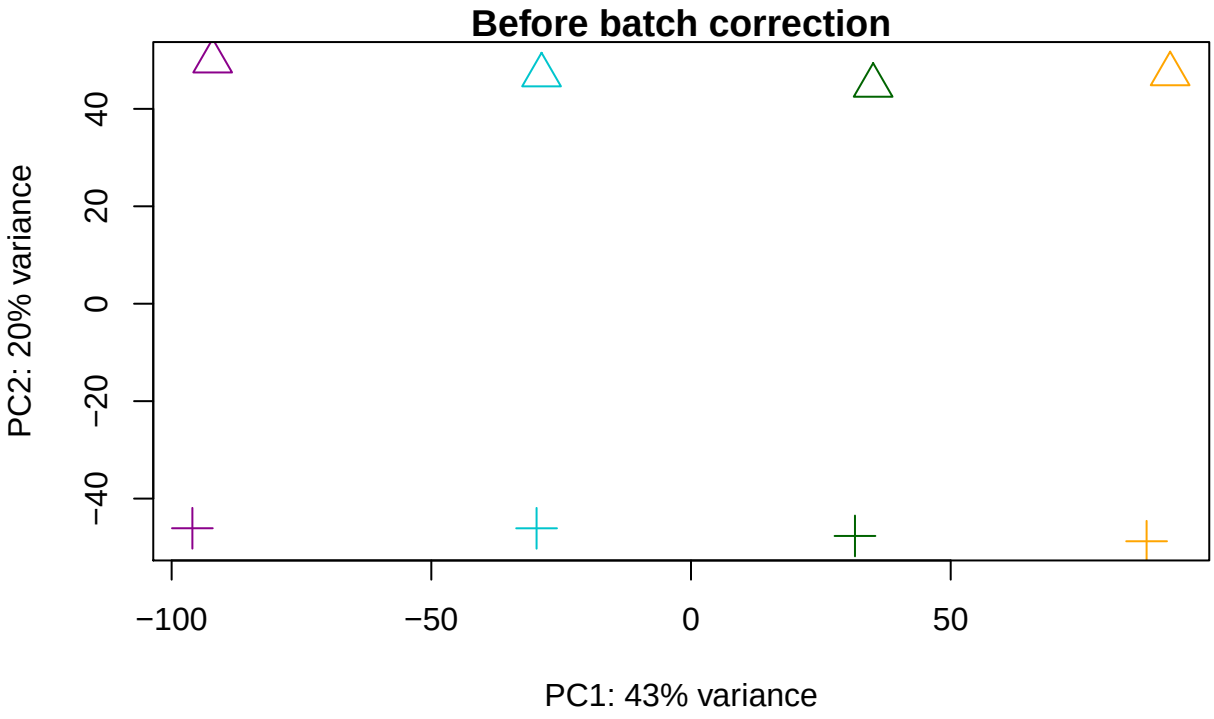



```
## converting counts to integer mode
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```



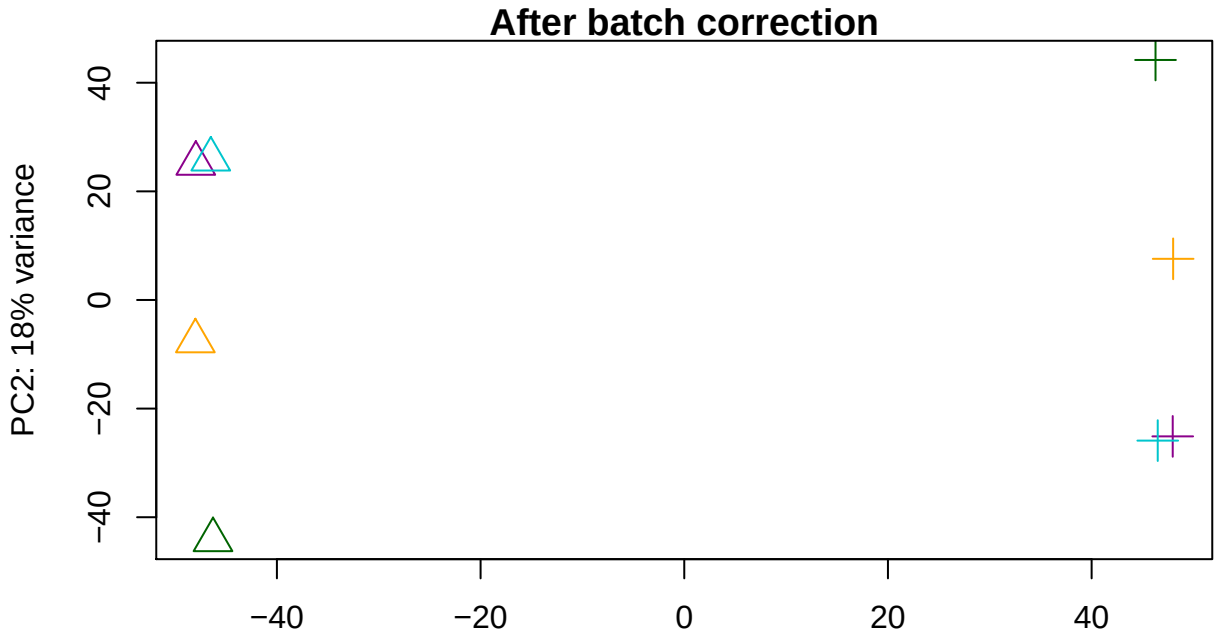

```
## converting counts to integer mode
```

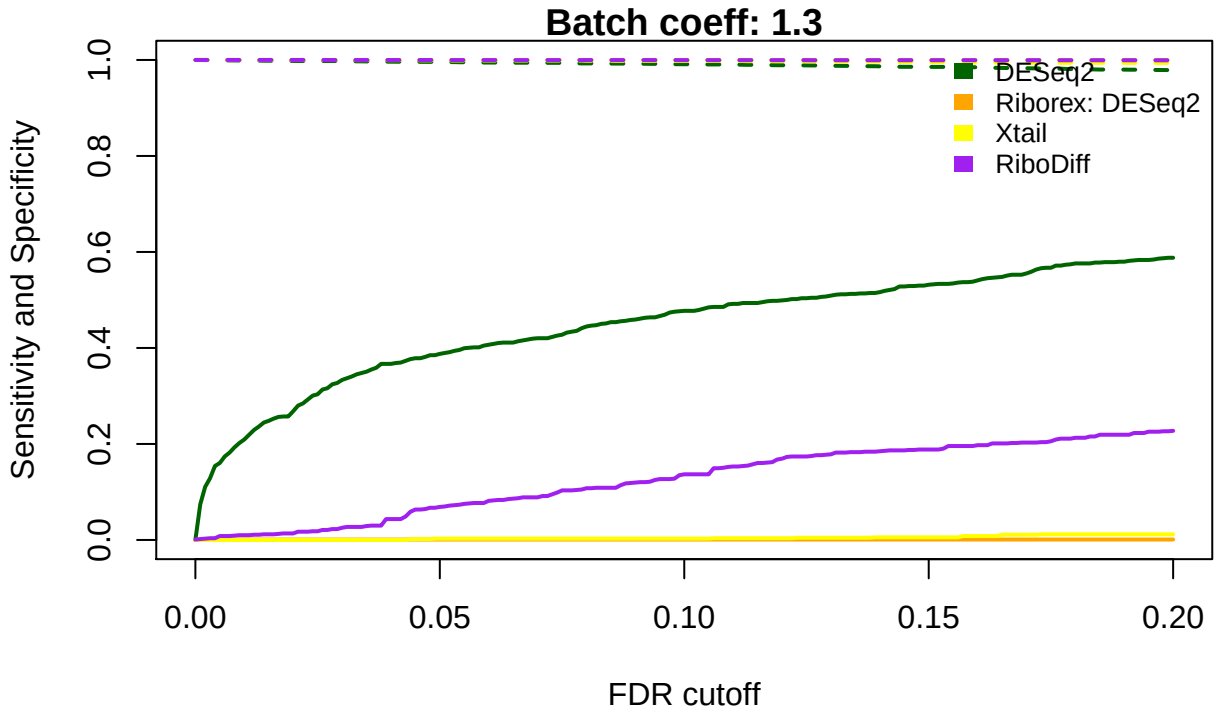


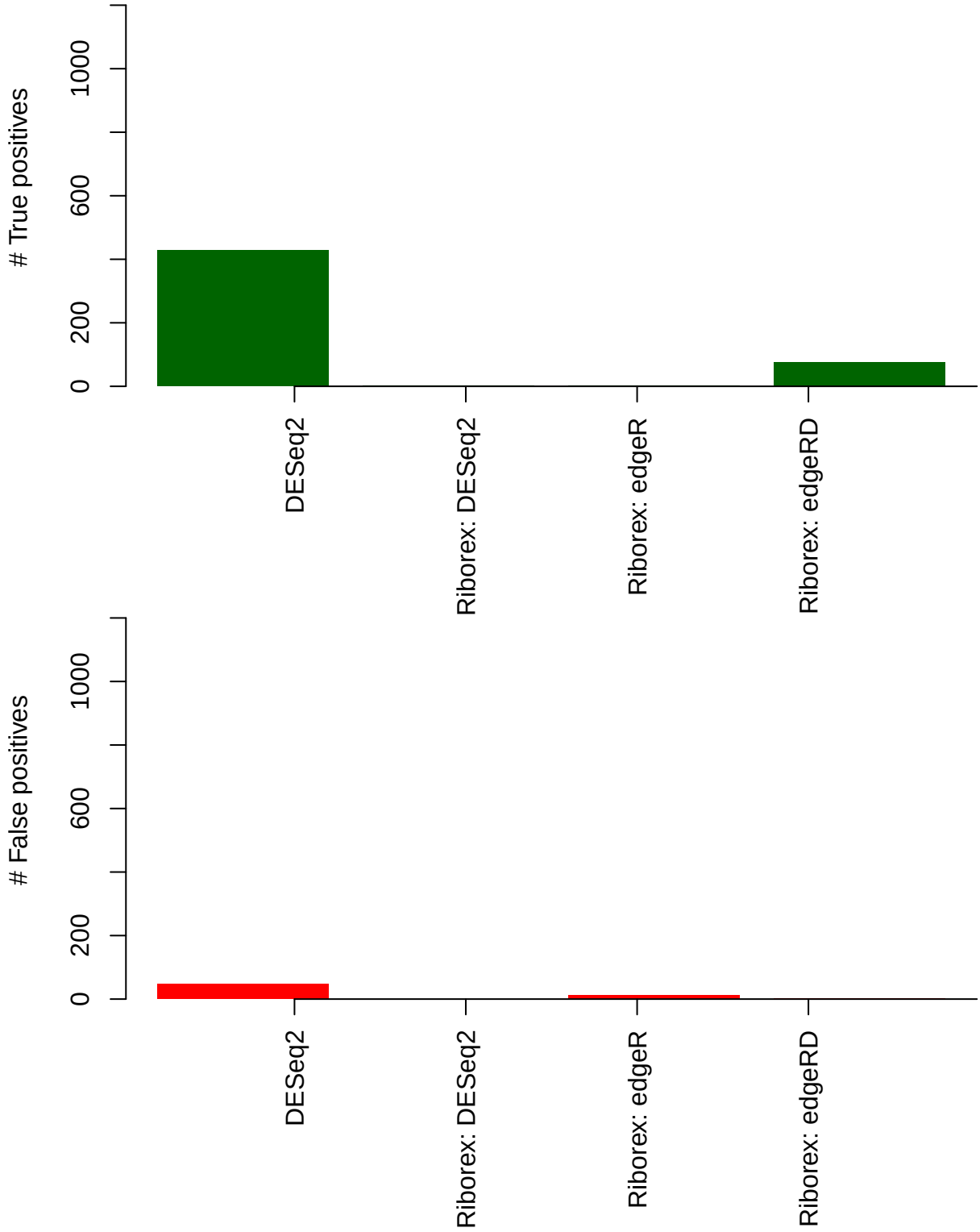



```
## DESeq2 mode selected
## combining design matrix
## converting counts to integer mode
## applying DESeq2 to modified design matrix
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
## Calculating the library size factors
## 1. Estimate the log2 fold change in mrna
## converting counts to integer mode
## 2. Estimate the log2 fold change in rpf
## converting counts to integer mode
## 3. Estimate the difference between two log2 fold changes
## 4. Estimate the log2 ratio in first condition
## converting counts to integer mode
## 5. Estimate the log2 ratio in second condition
## converting counts to integer mode
## 6. Estimate the difference between two log2 ratios
```

```
## Number of the log2FC and log2R used in determining the final p-value
##  log2FC: 6954
##  log2R: 5847
```






True positives
0
200
600
1000
DESeq2
Riborex: DESeq2
Riborex: edgeR
Riborex: edgeRD
False positives
0
200
600
1000
DESeq2
Riborex: DESeq2
Riborex: edgeR
Riborex: edgeRD

```
#}
#dev.off()
```