```
source("http://bioconductor.org/biocLite.R")
#source("http://bioconductor.org/workflows.R")
#setwd("/Users/leerkesm/Desktop/RNAseq/BATCH/class_data_ruv2_deseq2/")


#readline(prompt="Press [enter] to continue 0")

biocLite("airway")
library("airway")
data("airway")
se <- airway

dir <- system.file("extdata", package="airway", mustWork=TRUE)


#vignette("airway")

csvfile <- file.path(dir,"sample_table.csv")
list.files(dir)
csvfile <- file.path(dir,"sample_table.csv")
sampleTable <- read.csv(csvfile,row.names=1)

filenames <- file.path(dir, paste0(sampleTable$Run, "_subset.bam"))

biocLite("Rsamtools")
library("Rsamtools")

biocLite("GenomicFeatures")
library("GenomicFeatures")
gtffile <- file.path(dir,"Homo_sapiens.GRCh37.75_subset.gtf")

file.exists(filenames)

biocLite("GenomicRanges")
library("GenomicRanges")

bamfiles <- BamFileList(filenames, yieldSize=2000000)
seqinfo(bamfiles[1])

biocLite("GenomicAlignments")
library("GenomicAlignments")

txdb <- makeTxDbFromGFF(gtffile, format="gtf", circ_seqs=character())
ebg <- exonsBy(txdb, by="gene")

se <- summarizeOverlaps(features=ebg, reads=bamfiles,
                        mode="Union",
                        singleEnd=FALSE,
                        ignore.strand=TRUE,
                        fragments=TRUE )

assayNames(se)
head(assay(se), 3)
colSums(assay(se))

colSums(assay(se))
rowRanges(se)

data("airway")
se <- airway

str(metadata(rowRanges(se)))
colData(se)
colData(se) <- DataFrame(sampleTable)
se$cell
se$dex

str(se)

se$dex <- relevel(se$dex, "untrt")
se$dex

round( colSums(assay(se)) / 1e6, 1 )

colData(se)

biocLite("DESeq2")
library("DESeq2")

dds <- DESeqDataSet(se, design = ~ cell + dex)
countdata <- assay(se)
head(countdata, 3)
coldata <- colData(se)
ddsMat <- DESeqDataSetFromMatrix(countData = countdata,
                                 colData = coldata,
                                 design = ~ cell + dex)

nrow(dds)
dds <- dds[ rowSums(counts(dds)) > 1, ]
nrow(dds)

rld <- rlog(dds, blind=FALSE)
head(assay(rld), 3)

par( mfrow = c( 1, 2 ) )
dds <- estimateSizeFactors(dds)
```

```
#plot(log2(counts(dds, normalized=TRUE)[,1:2] + 1), pch=16, cex=0.3)
#readline(prompt="Press [enter] to continue 1")

#plot(assay(rld)[,1:2], pch=16, cex=0.3)
#readline(prompt="Press [enter] to continue 2")

sampleDists <- dist( t( assay(rld) ) )

biocLite("pheatmap")
library("pheatmap")

biocLite("RColorBrewer")
library("RColorBrewer")

#Sample distances

sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- paste( rld$dex, rld$cell, sep="-" )
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
#pheatmap(sampleDistMatrix,
#         clustering_distance_rows=sampleDists,
#         clustering_distance_cols=sampleDists,
#         col=colors)
#readline(prompt="Press [enter] to continue 3")

#check_outliers

plotPCA(rld, intgroup = c("dex", "cell"))
readline(prompt="Press [enter] to continue 4")

data <- plotPCA(rld, intgroup = c( "dex", "cell"), returnData=TRUE)

percentVar <- round(100 * attr(data, "percentVar"))

biocLite("ggplot2")
library("ggplot2")

#PCA plot using the rlog-transformed values.

#ggplot(data, aes(PC1, PC2, color=dex, shape=cell)) +
geom_point(size=3) +
#  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
#  ylab(paste0("PC2: ",percentVar[2],"% variance"))
#readline(prompt="Press [enter] to continue 5")

#MDS plot

#check_outliers
```

```
mdsData <- data.frame(cmdscale(sampleDistMatrix))
mds <- cbind(mdsData, as.data.frame(colData(rld)))
ggplot(mds, aes(X1,X2,color=dex,shape=cell)) + geom_point(size=3)
readline(prompt="Press [enter] to continue 6")

#Running the differential expression pipeline

dds <- DESeq(dds)

#Building the results table

res <- results(dds)

mcols(res, use.names=TRUE)

summary(res)

res.05 <- results(dds, alpha=.05)
table(res.05$padj < .05)

resLFC1 <- results(dds, lfcThreshold=1)
table(resLFC1$padj < 0.001)

#Other comparisons

results(dds, contrast=c("cell", "N061011", "N61311"))

sum(res$pvalue < 0.05, na.rm=TRUE)

sum(!is.na(res$pvalue))

sum(res$padj < 0.1, na.rm=TRUE)

resSig <- subset(res, padj < 0.1)
head(resSig[ order(resSig$log2FoldChange), ])

head(resSig[ order(resSig$log2FoldChange, decreasing=TRUE), ])

#Plotting results

topGene <- rownames(res)[which.min(res$padj)]
#plotCounts(dds, gene=topGene, intgroup=c("dex"))
#readline(prompt="Press [enter] to continue 7")

#Normalized counts for a single gene over treatment group.

data <- plotCounts(dds, gene=topGene, intgroup=c("dex","cell"),
returnData=TRUE)
```

```
#ggplot(data, aes(x=dex, y=count, color=cell)) +
#  scale_y_log10() +
#  geom_point(position=position_jitter(width=.1,height=0), size=3)
#readline(prompt="Press [enter] to continue 8")

#Normalized counts indicating cell line with color.

#ggplot(data, aes(x=dex, y=count, fill=dex)) +
#  scale_y_log10() +
#  geom_dotplot(binaxis="y", stackdir="center")
#readline(prompt="Press [enter] to continue 9")

#Normalized counts using a more structural arrangement. Here the
color indicates treatment.

#ggplot(data, aes(x=dex, y=count, color=cell, group=cell)) +
#  scale_y_log10() + geom_point(size=3) + geom_line()
#readline(prompt="Press [enter] to continue 10")

#An MA-plot

#plotMA(res, ylim=c(-5,5))
#readline(prompt="Press [enter] to continue 11")

#plotMA(resLFC1, ylim=c(-5,5))
#readline(prompt="Press [enter] to continue 12")
#topGene <- rownames(resLFC1)[which.min(resLFC1$padj)]
#with(resLFC1[topGene, ], {
#  points(baseMean, log2FoldChange, col="dodgerblue", cex=2, lwd=2)
#  text(baseMean, log2FoldChange, topGene, pos=2, col="dodgerblue")
#})

#Histogram of p values for genes with mean normalized count larger
than 1.

#hist(res$pvalue[res$baseMean > 1], breaks=0:20/20, col="grey50",
border="white")
#readline(prompt="Press [enter] to continue 13")

#Gene clustering

biocLite("genefilter")
library("genefilter")

topVarGenes <- head(order(rowVars(assay(rld)),decreasing=TRUE),20)

mat <- assay(rld)[ topVarGenes, ]
mat <- mat - rowMeans(mat)
df <- as.data.frame(colData(rld)[,c("cell","dex")])
```

```
#jpeg('rplotpheatmap.jpg')
pheatmap(mat, annotation_col=df)
readline(prompt="Press [enter] to continue pheatmap14")
#dev.off()


#Independent filtering

qs <- c(0, quantile(resLFC1$baseMean[resLFC1$baseMean > 0], 0:6/6))
bins <- cut(resLFC1$baseMean, qs)
levels(bins) <- paste0("~",round(signif(.5*qs[-1] + .5*qs[-
length(qs)],2)))
ratios <- tapply(resLFC1$pvalue, bins, function(p) mean(p < .05,
na.rm=TRUE))
#barplot(ratios, xlab="mean normalized count", ylab="ratio of small p
values")
#readline(prompt="Press [enter] to continue 15")

#Annotating and exporting results

biocLite("AnnotationDbi")
biocLite("org.Hs.eg.db")

library("AnnotationDbi")
library("org.Hs.eg.db")

columns(org.Hs.eg.db)

res$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(res),
                     column="SYMBOL",
                     keytype="ENSEMBL",
                     multiVals="first")
res$entrez <- mapIds(org.Hs.eg.db,
                     keys=row.names(res),
                     column="ENTREZID",
                     keytype="ENSEMBL",
                     multiVals="first")

resOrdered <- res[order(res$padj),]

#Exporting results

resOrderedDF <- as.data.frame(resOrdered)[1:100,]
#write.csv(resOrderedDF, file="myresults.csv")
#write.table(topTable(fit2, coef=1, adjust="fdr", sort.by="logFC",
number=nrow(d_vsn)), file="blm-cd11.xls", row.names=F, sep="\t")
write.table(resOrderedDF, file="myresults.xls")
```

```
#Plotting fold changes in genomic space
resGR <- results(dds, lfcThreshold=1, format="GRanges")
resGR$symbol <- mapIds(org.Hs.eg.db, names(resGR), "SYMBOL",
"ENSEMBL")

biocLite("Gviz")
library("Gviz")

window <- resGR[topGene] + 1e6
strand(window) <- "*"
resGRsub <- resGR[resGR %over% window]
naOrDup <- is.na(resGRsub$symbol) | duplicated(resGRsub$symbol)
resGRsub$group <- ifelse(naOrDup, names(resGRsub), resGRsub$symbol)

sig <- factor(ifelse(resGRsub$padj < .1 &
!is.na(resGRsub$padj),"sig","notsig"))

options(ucscChromosomeNames=FALSE)
g <- GenomeAxisTrack()
a <- AnnotationTrack(resGRsub, name="gene ranges", feature=sig)
d <- DataTrack(resGRsub, data="log2FoldChange", baseline=0,
               type="h", name="log2 fold change", strand="+")

#log2 fold changes in genomic region surrounding the gene with
smallest adjusted p value. Genes highlighted in pink have adjusted p
value less than 0.1.


#jpeg('plotTracks.jpg')
plotTracks(list(g,d,a), groupAnnotation="group", notsig="grey",
sig="hotpink")
readline(prompt="Press [enter] to continue pheatmap16a")
#dev.off()

readline(prompt="Press [enter] to continue 16 svaseq now")

#Below we obtain a matrix of normalized counts for which the average
count across
#samples is larger than 1. As we described above, we are trying to
recover any hidden
#batch effects, supposing that we do not know the cell line
information. So we use a
#full model matrix with the dex variable, and a reduced, or null,
model matrix with only
#an intercept term. Finally we specify that we want to estimate 2
surrogate variables.
#For more information read the manual page for the svaseq function by
typing ?svaseq.
```

```
biocLite("sva")
library("sva")
dat <- counts(dds, normalized=TRUE)
idx <- rowMeans(dat) > 1
dat <- dat[idx,]
mod <- model.matrix(~ dex, colData(dds))
mod0 <- model.matrix(~ 1, colData(dds))
svseq <- svaseq(dat, mod, mod0, n.sv=2)

svseq$sv

#Because we actually do know the cell lines, we can see how well the
SVA method did
#at recovering these variables (Figure below).

#jpeg('plotTracks.jpg')
#plotTracks(list(g,d,a), groupAnnotation="group", notsig="grey",
sig="hotpink")
#dev.off()

#jpeg('stripchart1b.jpg')
par(mfrow=c(2,1),mar=c(3,5,3,1))
stripchart(svseq$sv[,1] ~ dds$cell,vertical=TRUE,main="SV1")
abline(h=0)
readline(prompt="Press [enter] to continue 17 svaseq now")
#dev.off()

#jpeg('stripchart2b.jpg')
stripchart(svseq$sv[,2] ~ dds$cell,vertical=TRUE,main="SV2")
abline(h=0)
readline(prompt="Press [enter] to continue 18 svaseq now")
#dev.off()

#Surrogate variables 1 and 2 plotted over cell line. Here, we know
the hidden source of variation (cell line), and therefore can see how
the SVA procedure is able to identify a source of variation which is
correlated with cell line.
#Finally, in order to use SVA to remove any effect on the counts from
our surrogate variables, we simply add these two surrogate variables
as columns to the DESeqDataSet and then add them to the design:

ddssva <- dds
ddssva$SV1 <- svseq$sv[,1]
ddssva$SV2 <- svseq$sv[,2]
design(ddssva) <- ~ SV1 + SV2 + dex

#We could then produce results controlling for surrogate variables by
running DESeq with the new design:
```

```
ddssva <- DESeq(ddssva)
```