DESeq\_analysis\_kiwifruit\_flowers

Sarah Pilkington and Lara Brian

2 March 2016

Raw files used for FastQC diagnostic, then clipped, trimmed and filtered using FASTQMCF and FQ2TRIMMED. The files were then quality checked again and mapped to reference using STAR. The reference used in this analysis was the Chromosome 25 from the PS1v68 assembly and the gff3 file used was Aug68at\_unmasked. The reads were then counted from the .BAM output files using HTSeqCount. The count files were then used as the input file for this DESeq2 analysis to create one file with gene IDs in column A followed by the results from the count files in order.

First the packages needed for the analysis must be loaded:

library("DESeq2")

## Loading required package: S4Vectors  
## Loading required package: stats4  
## Loading required package: BiocGenerics  
## Loading required package: parallel  
##   
## Attaching package: 'BiocGenerics'  
##   
## The following objects are masked from 'package:parallel':  
##   
## clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
## clusterExport, clusterMap, parApply, parCapply, parLapply,  
## parLapplyLB, parRapply, parSapply, parSapplyLB  
##   
## The following object is masked from 'package:stats':  
##   
## xtabs  
##   
## The following objects are masked from 'package:base':  
##   
## anyDuplicated, append, as.data.frame, as.vector, cbind,  
## colnames, do.call, duplicated, eval, evalq, Filter, Find, get,  
## intersect, is.unsorted, lapply, Map, mapply, match, mget,  
## order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,  
## rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,  
## table, tapply, union, unique, unlist, unsplit  
##   
## Creating a generic function for 'nchar' from package 'base' in package 'S4Vectors'  
## Loading required package: IRanges  
## Loading required package: GenomicRanges  
## Loading required package: GenomeInfoDb  
## Loading required package: Rcpp  
## Loading required package: RcppArmadillo

library("ggplot2")

Then set the working directory and where the results should be saved to. Check it has set correctly using *getwd()*:

setwd("~/RNA-seq/R")  
getwd()

## [1] "C:/Users/hrasmp/Documents/RNA-seq/R"

Then define the input file:

RawData <- read.delim("kiwifruit\_flower\_counts\_delim.csv",sep=",",header=TRUE)  
dim(RawData)

## [1] 1623 4

head(RawData)

## Gene\_ID X1 X2 X3  
## 1 CHR25.AtP.um.g40633 83 61 60  
## 2 CHR25.AtP.um.g40634 667 677 658  
## 3 CHR25.AtP.um.g40635 249 240 225  
## 4 CHR25.AtP.um.g40636 228 250 225  
## 5 CHR25.AtP.um.g40637 449 422 430  
## 6 CHR25.AtP.um.g40638 241 245 267

cbind(colnames(RawData))

## [,1]   
## [1,] "Gene\_ID"  
## [2,] "X1"   
## [3,] "X2"   
## [4,] "X3"

Define the COUNT data and set it as a matrix:

DATA <- as.matrix(RawData[,c(2:4)])  
head(DATA)

## X1 X2 X3  
## [1,] 83 61 60  
## [2,] 667 677 658  
## [3,] 249 240 225  
## [4,] 228 250 225  
## [5,] 449 422 430  
## [6,] 241 245 267

To define the row names:

DATA2 <- DATA  
rownames(DATA2) <- RawData[,1]  
head(DATA2)

## X1 X2 X3  
## CHR25.AtP.um.g40633 83 61 60  
## CHR25.AtP.um.g40634 667 677 658  
## CHR25.AtP.um.g40635 249 240 225  
## CHR25.AtP.um.g40636 228 250 225  
## CHR25.AtP.um.g40637 449 422 430  
## CHR25.AtP.um.g40638 241 245 267

Group samples into various subsets of interest so they can be made into a colData dataframe:

individual <- as.factor(rep(seq(1:3)))

Make a colData Dataframe with the info:

colData <- DataFrame(individual,row.names=seq(1:3))

Make a DeseqDataSet object (an S3 class object can be made if the row data table holds multiple values that you would like queried):

dds<-DESeqDataSetFromMatrix(countData = DATA2,colData = colData,design = ~ individual)

#### Multifactor design

Using dds with previous design:

ddsMF <- DESeq(dds)

## estimating size factors  
## estimating dispersions

## Warning in checkForExperimentalReplicates(object, modelMatrix): same number of samples and coefficients to fit,  
## estimating dispersion by treating samples as replicates.  
## read the ?DESeq section on 'Experiments without replicates'

## gene-wise dispersion estimates  
## mean-dispersion relationship  
## final dispersion estimates  
## fitting model and testing

resMF <- results(ddsMF)  
head(resMF)

## log2 fold change (MAP): individual 3 vs 1   
## Wald test p-value: individual 3 vs 1   
## DataFrame with 6 rows and 6 columns  
## baseMean log2FoldChange lfcSE stat  
## <numeric> <numeric> <numeric> <numeric>  
## CHR25.AtP.um.g40633 68.01495 -0.073158223 0.05958964 -1.22770035  
## CHR25.AtP.um.g40634 667.58528 -0.012505939 0.07914655 -0.15800991  
## CHR25.AtP.um.g40635 238.08718 -0.057816597 0.08131244 -0.71104244  
## CHR25.AtP.um.g40636 234.43681 -0.007203577 0.08107792 -0.08884758  
## CHR25.AtP.um.g40637 433.80789 -0.034140006 0.08267963 -0.41291920  
## CHR25.AtP.um.g40638 251.08428 0.061023711 0.08171633 0.74677498  
## pvalue padj  
## <numeric> <numeric>  
## CHR25.AtP.um.g40633 0.2195595 1  
## CHR25.AtP.um.g40634 0.8744490 1  
## CHR25.AtP.um.g40635 0.4770579 1  
## CHR25.AtP.um.g40636 0.9292030 1  
## CHR25.AtP.um.g40637 0.6796658 1  
## CHR25.AtP.um.g40638 0.4551994 1

resultsNames(ddsMF)

## [1] "Intercept" "individual1" "individual2" "individual3"

mcols will bring up the meaning of the columns:

mcols(resMF, use.names=TRUE)

## DataFrame with 6 rows and 2 columns  
## type description  
## <character> <character>  
## baseMean intermediate mean of normalized counts for all samples  
## log2FoldChange results log2 fold change (MAP): individual 3 vs 1  
## lfcSE results standard error: individual 3 vs 1  
## stat results Wald statistic: individual 3 vs 1  
## pvalue results Wald test p-value: individual 3 vs 1  
## padj results BH adjusted p-values

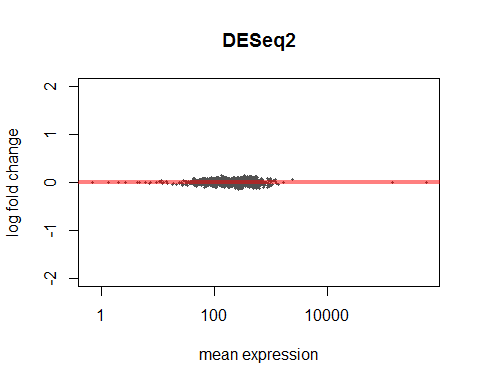
To bring up a summary of the results:

summary(resMF)

##   
## out of 1608 with nonzero total read count  
## adjusted p-value < 0.1  
## LFC > 0 (up) : 0, 0%   
## LFC < 0 (down) : 0, 0%   
## outliers [1] : 0, 0%   
## low counts [2] : 0, 0%   
## (mean count < 0.7)  
## [1] see 'cooksCutoff' argument of ?results  
## [2] see 'independentFiltering' argument of ?results

Plot the MA-plot to show log2 fold changes attributable to a given variable over the mean of normalised counts:

plotMA(resMF, main="DESeq2", ylim=c(-2,2))



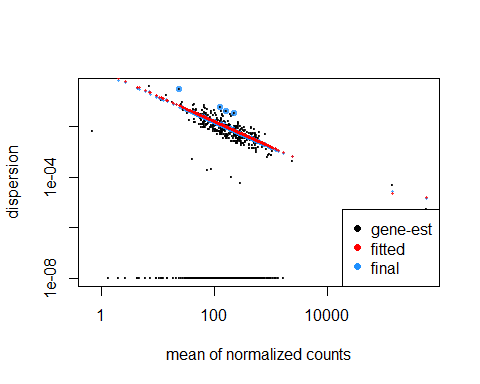
To add the Maximum Likelihood Estimate for the log2 fold change as a column in the results:

resMLE <- results(ddsMF, addMLE=TRUE)  
head(resMLE, 4)

## log2 fold change (MAP): individual 3 vs 1   
## Wald test p-value: individual 3 vs 1   
## DataFrame with 4 rows and 7 columns  
## baseMean log2FoldChange lfcMLE lfcSE  
## <numeric> <numeric> <numeric> <numeric>  
## CHR25.AtP.um.g40633 68.01495 -0.073158223 -0.46776874 0.05958964  
## CHR25.AtP.um.g40634 667.58528 -0.012505939 -0.01921957 0.07914655  
## CHR25.AtP.um.g40635 238.08718 -0.057816597 -0.14584103 0.08131244  
## CHR25.AtP.um.g40636 234.43681 -0.007203577 -0.01872912 0.08107792  
## stat pvalue padj  
## <numeric> <numeric> <numeric>  
## CHR25.AtP.um.g40633 -1.22770035 0.2195595 1  
## CHR25.AtP.um.g40634 -0.15800991 0.8744490 1  
## CHR25.AtP.um.g40635 -0.71104244 0.4770579 1  
## CHR25.AtP.um.g40636 -0.08884758 0.9292030 1

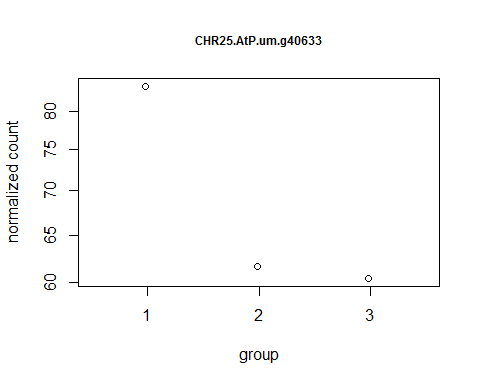
To plot the dispersion estimates:

plotDispEsts(ddsMF)



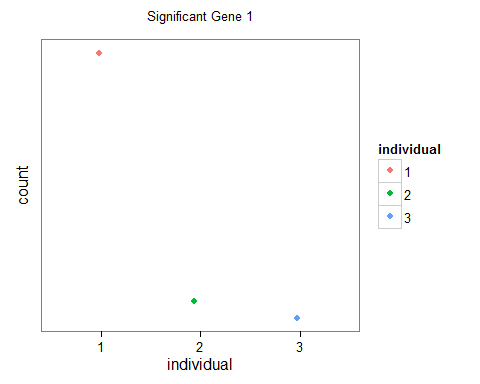
To plot the counts for a single gene across the groups:

plotCounts(ddsMF, gene=which.min(resMF$padj), intgroup="individual", cex.main=0.75)



To customise the counts plot:

data <- plotCounts(ddsMF, gene=which.min(resMF$padj), intgroup=c("individual"), cex.main=0.75, returnData=TRUE)  
ggplot(data, aes(x=individual, y=count, color=individual)) +  
 theme\_bw() +  
 scale\_y\_log10() +   
 geom\_point(position=position\_jitter(width=.1,height=0)) +  
 ggtitle("Significant Gene 1") +  
 theme(plot.title=element\_text(size=10, vjust=2), panel.grid.major=element\_blank(), panel.grid.minor=element\_blank())



#### PCA Analysis

Make transformations to data for PCA analysis:

rld <- rlog(ddsMF)  
vsd <- varianceStabilizingTransformation(ddsMF)

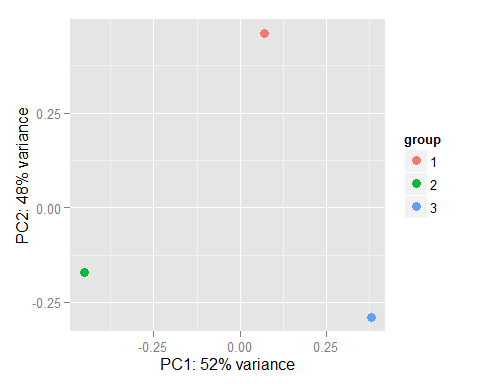
Take a quick look at a PCA plot of variance stabilised data:

pdf(file="VarianceStabl\_PCA2\_n1500.pdf", onefile= TRUE, paper="a4")  
  
plotPCA(vsd, intgroup=c("individual"),ntop = 1500)  
  
dev.off()

## png   
## 2

Take a quick look at a PCA plot of log normalised data:

plotPCA(rld, intgroup=c("individual"),ntop = 1500)



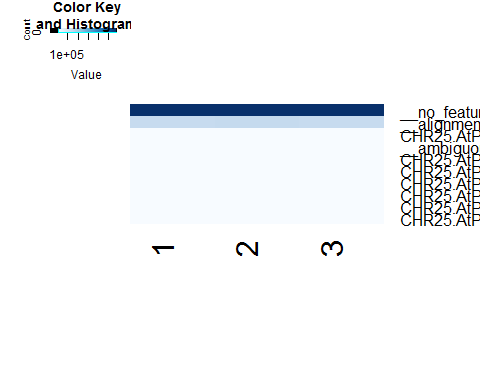
#### Heatmaps

Call the libraries "RColorBrewer" and "gplots" and build the heatmaps:

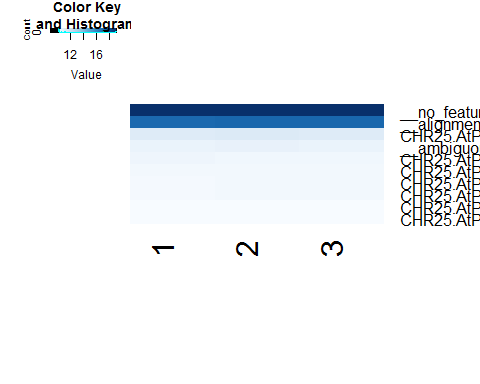
library("RColorBrewer")  
library("gplots")

##   
## Attaching package: 'gplots'  
##   
## The following object is masked from 'package:IRanges':  
##   
## space  
##   
## The following object is masked from 'package:stats':  
##   
## lowess

select <- order(rowMeans(counts(ddsMF, normalized=TRUE)),decreasing=TRUE)[1:10]  
hmcol <- colorRampPalette(brewer.pal(9, "Blues"))(100)  
heatmap.2(counts(ddsMF,normalized=TRUE)[select,], col = hmcol,  
 Rowv = FALSE, Colv = FALSE, scale="none",  
 dendrogram="none", trace="none", margin=c(10,6))



heatmap.2(assay(rld)[select,], col = hmcol,  
 Rowv = FALSE, Colv = FALSE, scale="none",  
 dendrogram="none", trace="none", margin=c(10,6))



heatmap.2(assay(vsd)[select,], col = hmcol,  
 Rowv = FALSE, Colv = FALSE, scale="none",  
 dendrogram="none", trace="none", margin=c(10,6))

