DESeq_BioInfoCourse

Justin Anderson, Bjoern Pietzenuk and Lara Syllwasschy

2018

DESeq2: Differential Gene Expression Analysis

We will now be leveraging your newly formed R skills to analyze the gene count data you generated this week. To help us in the analysis process we will be using a package called DESeq2. A package is a collection of code that is used for a specific analysis. Once the package is loaded we can use it's functions just like we have used the base R functions, e.g. head() and hist(). The DESeq2 package was written in the following paper: Love MI, Huber W and Anders S (2014). "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2." Genome Biology, 15, pp. 550. doi: 10.1186/s13059-014-0550-8. If you want to learn more about DESeq2 you can visit their vignette page here: http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html#quick-start (http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html#quick-start) or http://bioconductor.jp/packages/2.14/bioc/vignettes/DESeq2/inst/doc/beginner.pdf (http://bioconductor.jp/packages/2.14/bioc/vignettes/DESeq2/inst/doc/beginner.pdf)

#Loading the DESeq2 package

```
#First we tell R where online this package can be found:
#install.packages("Rcpp")
#install.packages("BiocManager")
#install.packages("foghorn")
#BiocManager::install("affy")
#Then we tell R which package at this site we want to download to your computer
#BiocManager::install(c("DESeq2")) #Yes, you also want to update all the dependencies
#Lastly, we use the function library to load this package into our working memory.
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 objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colMeans,
##     colnames, colSums, dirname, do.call, duplicated, eval, evalq,
##     Filter, Find, get, grep, grepl, intersect, is.unsorted, lapply,
##     lengths, Map, mapply, match, mget, order, paste, pmax, pmax.int,
##     pmin, pmin.int, Position, rank, rbind, Reduce, rowMeans, rownames,
##     rowSums, sapply, setdiff, sort, table, tapply, union, unique,
##     unsplit, which, which.max, which.min
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges
```

```
##
## Attaching package: 'IRanges'
```

```
## The following object is masked from 'package:grDevices':
##
##     windows
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: GenomeInfoDb
```

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
## Loading required package: DelayedArray
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
```

```
## Loading required package: BiocParallel
```

```
##
## Attaching package: 'DelayedArray'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
```

```
## The following objects are masked from 'package:base':
##
##     aperm, apply
```

```
#A list of packages loaded can be viewed in the Packages tab of the lower right graphics pane
l.
#You only need to download each package a single time so if you return to this code later you
r can skip the "source" and "biocLite" steps above.

#Install the following packages and then load them into the working memory:
#biocLite("lazyeval")
#biocLite("ggplot2")
#biocLite("affy")

library(lazyeval)
library(ggplot2)
library(affy)
```

Load the count files and merge into a single dataframe

One way to load your count files is with the “Import Dataset” option in the Environment panel. Use the option “From CSV” or “From Text (base)”. In the popup window set the ‘Separator’ or ‘Delimiter’ option to Tab and uncheck the ‘First Row as Names’ option because the data does not have a header line. Clicking the Import button will add this file to your Environment. The code to load this file was automatically generated and run in the lower left Console panel. Copy the code that was created in the Console and save it in the following R code section. This way you can return to this code later and run it directly. Repeat for all count files, being sure to name them uniquely.

I suggest the following dataframe names: Noss_R1 Noss_R2 Noss_R3 Pais_R1 Pais_R2 Pais_R3

```
#File loading code here:
Noss_R1<-read.table("Noss_1_count.txt",sep="\t",header=F)
Noss_R2<-read.table("Noss_2_count.txt",sep="\t",header=F)
Noss_R3<-read.table("Noss_3_count.txt",sep="\t",header=F)
Pais_R1<-read.table("Pais_1_count.txt",sep="\t",header=F)
Pais_R2<-read.table("Pais_2_count.txt",sep="\t",header=F)
Pais_R3<-read.table("Pais_3_count.txt",sep="\t",header=F)




### Each row is a single gene. How many genes are annotated in the A. halleri genome?
# Answer: __32553_ genes

#Add column names to match the data
colnames(Noss_R1)=c("Gene","Counts_Noss_R1")
colnames(Noss_R2)=c("Gene","Counts_Noss_R2")
colnames(Noss_R3)=c("Gene","Counts_Noss_R3")
colnames(Pais_R1)=c("Gene","Counts_Pais_R1")
colnames(Pais_R2)=c("Gene","Counts_Pais_R2")
colnames(Pais_R3)=c("Gene","Counts_Pais_R3")
# Merge the data into a single dataframe
  # The function merge() combines two dataframes by the specified identical column name.
Counts=merge(Noss_R1,Noss_R2,by="Gene")
Counts=merge(Counts,Noss_R3,by="Gene")
Counts=merge(Counts,Pais_R1,by="Gene")
Counts=merge(Counts,Pais_R2,by="Gene")
Counts=merge(Counts,Pais_R3,by="Gene")
head(Counts)
```

```
##     Gene Counts_Noss_R1 Counts_Noss_R2 Counts_Noss_R3 Counts_Pais_R1
## 1 g00001            361            326            314            318
## 2 g00002            109            121             80            129
## 3 g00003             41             20             22             19
## 4 g00004              0              0              0             26
## 5 g00005            419            400            413            473
## 6 g00006            180            213            276            246
##   Counts_Pais_R2 Counts_Pais_R3
## 1            222            244
## 2            103             63
## 3             19              8
## 4             27             20
## 5            383            313
## 6            271            167
```

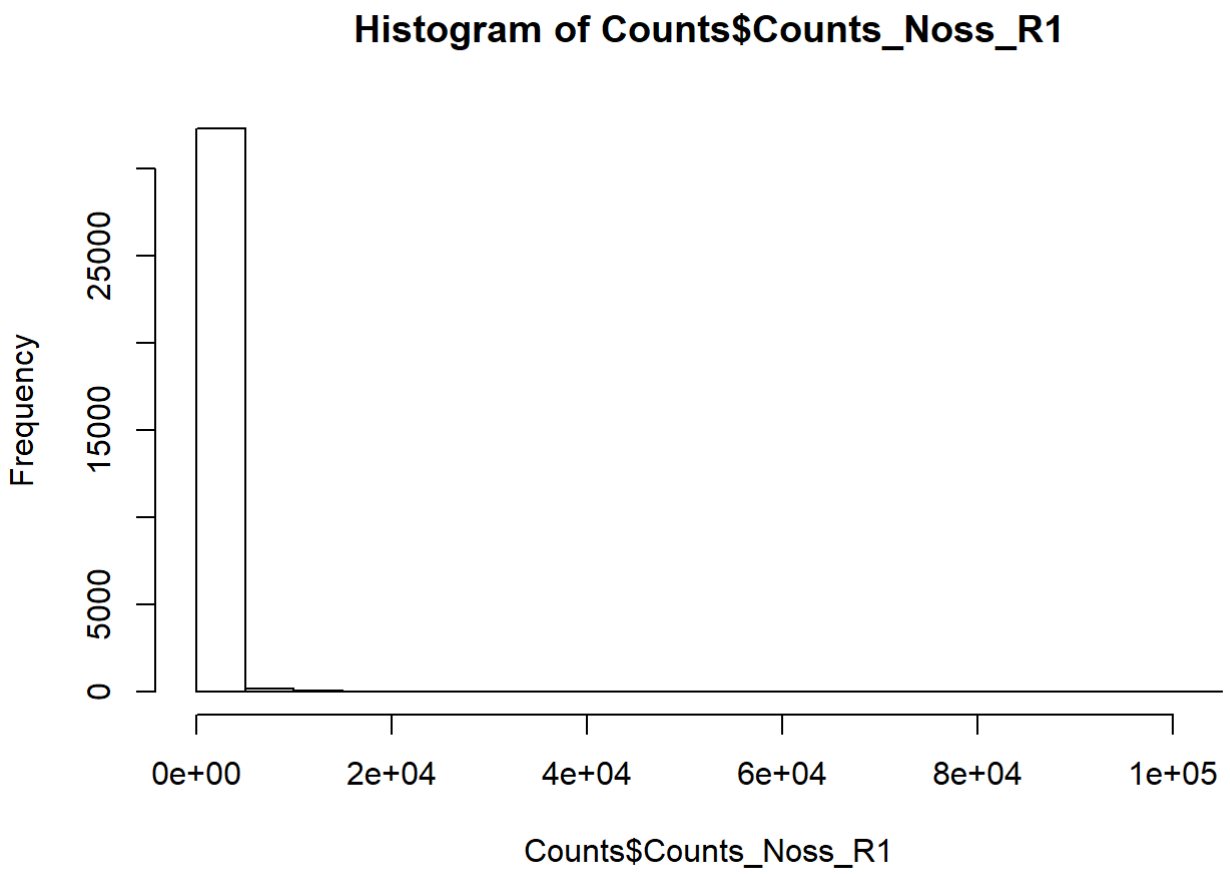
```
# Use the summary() function to determine the average number of counts per gene.
```

summary(Counts)

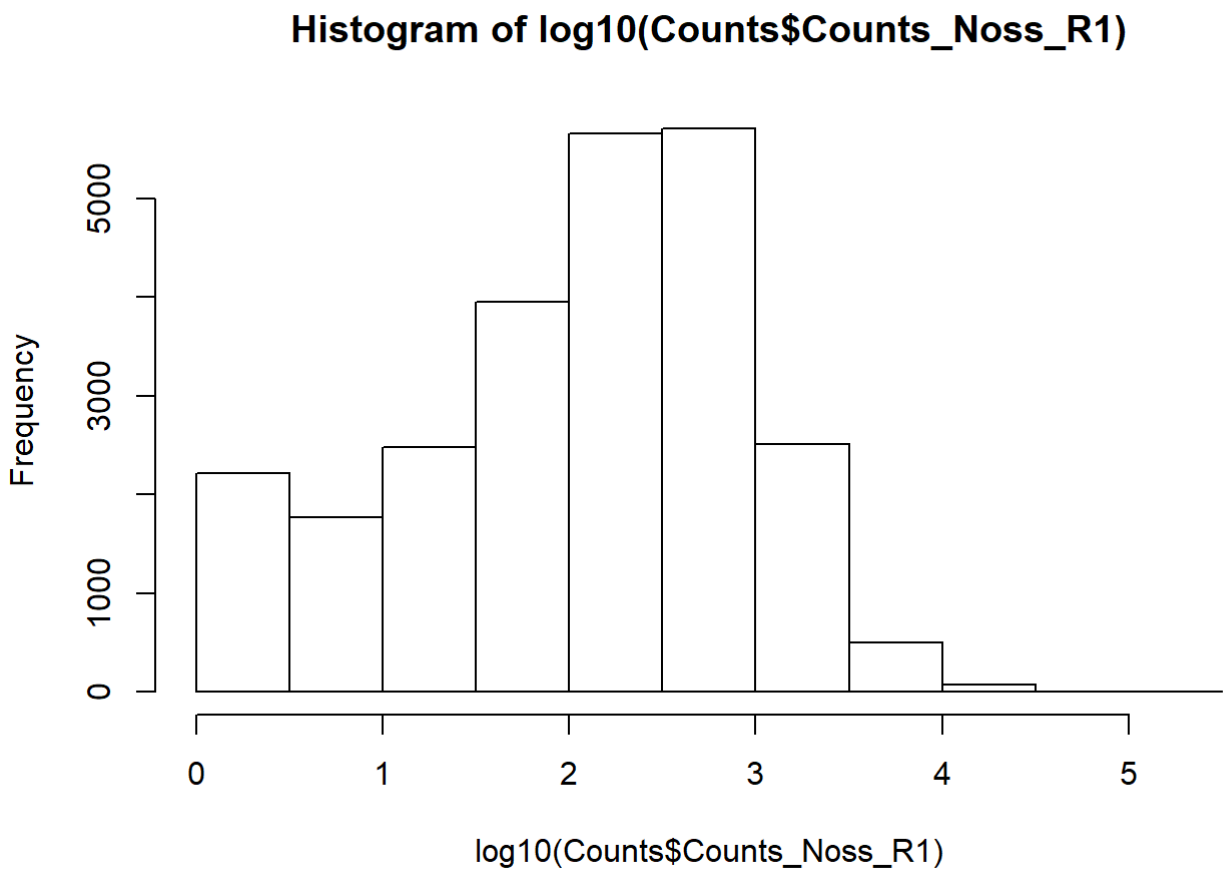
Explore the data

Before we jump into an analysis of differential expression with DESeq2 let’s explore our data using the R techniques we know.

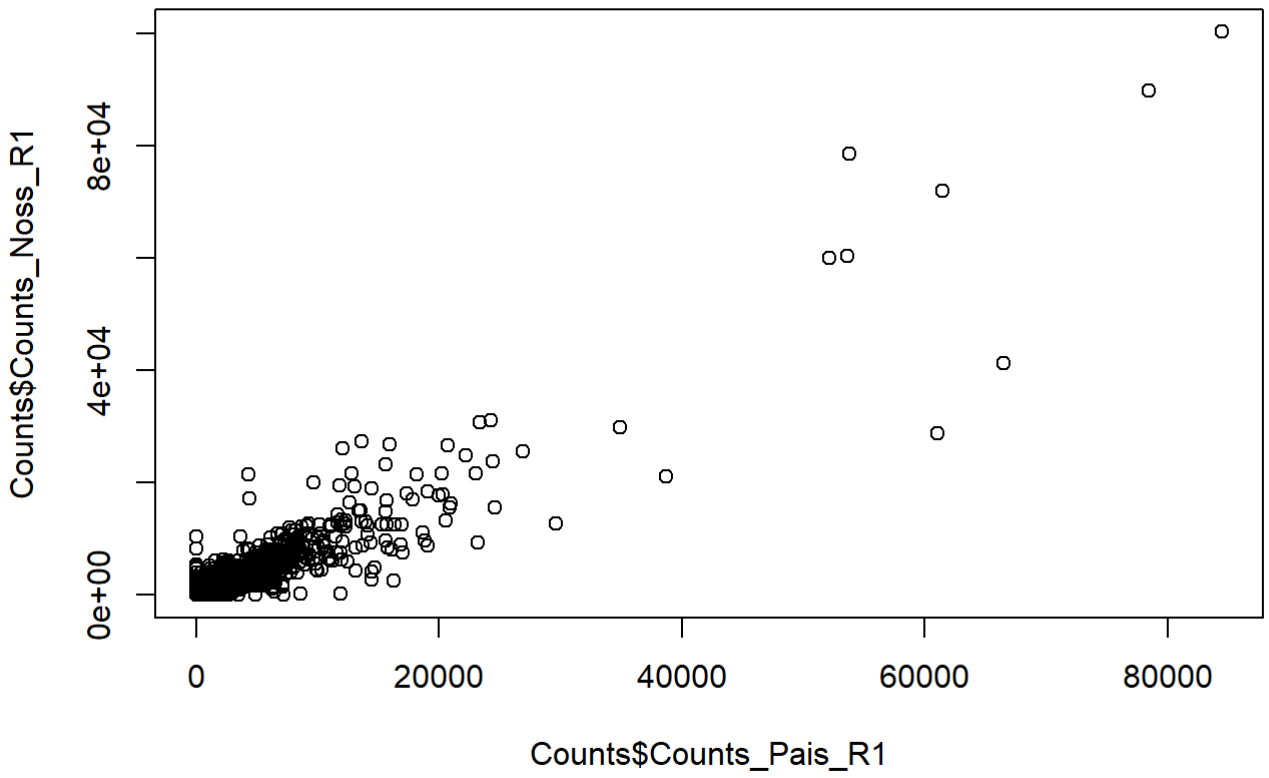
```
#Time to flex those R muscles!

# Generate a histogram of the Counts_Noss_R1 column. Log10 transforming the data might prove
 more informative.
hist(Counts$Counts_Noss_R1)
```




```
hist(log10(Counts$Counts_Noss_R1))
```




```
# Create a scatter plot of Counts_Noss_R1 counts vs Counts_Pais_R1 counts
plot(Counts$Counts_Noss_R1~Counts$Counts_Pais_R1)
```



```
# Sort the  data by Counts_Noss_R1 and see which genes have the highest number of counts in N
oss
Counts_sorted<-Counts[order(Counts$Counts_Noss_R1),]
tail(Counts_sorted)
```

```
##         Gene Counts_Noss_R1 Counts_Noss_R2 Counts_Noss_R3 Counts_Pais_R1
## 27322 g27322          59893          41224          11452          52121
## 23052 g23052          60305          53102          32636          53609
## 648   g00648          71862          39546           9528          61436
## 4721  g04721          78479          48491          28053          53787
## 18687 g18687          89737          85758          59568          78473
## 1268  g01268         100225          63082          27745          84473
##       Counts_Pais_R2 Counts_Pais_R3
## 27322          15235          15927
## 23052          19482          28919
## 648            14111          21321
## 4721           21368          29748
## 18687          35661          72848
## 1268           16973          25582
```

```
# How many genes have zero counts in Counts_Pais_R1?
  #Start by creating a dataframe called Zeros as a subset of the count data that only contain
individuals with a zero value in the Counts_Pais_R1 column. Then use nrow() to count the numb
er of rows.
Zeros<-Counts[Counts$Counts_Pais_R1==0,]
nrow(Zeros)
```

```
## [1] 8408
```

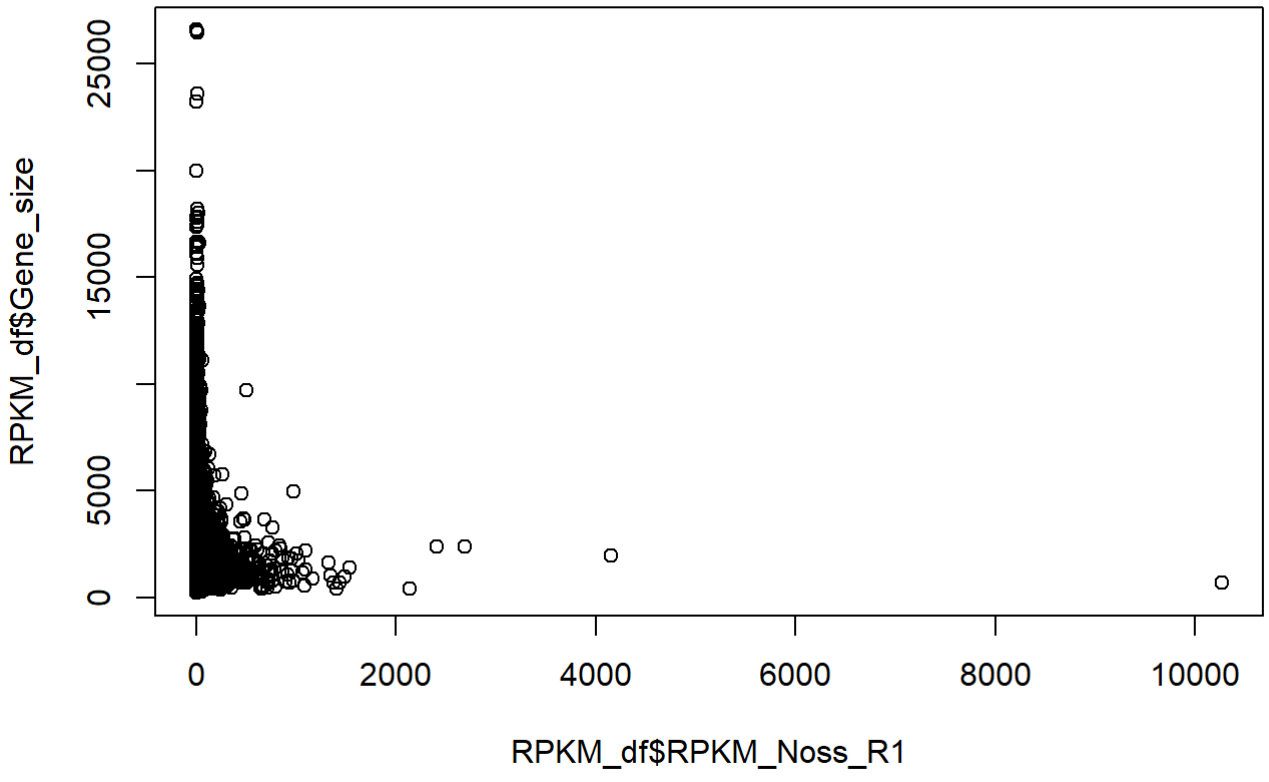
Compare across genes with RPKM

Scientists often want to compare the expression not only between conditions but between genes. Thus far we have only explored count data which is the number of reads per gene. If we want to compare between genes and between libraries we need to scale for gene size and for the total number of sequenced reads per library. If one library was sequenced more resulting in twice as many reads in the count results we don't want to falsely assume that all of the genes are expressed higher in that library. Similarly, if a gene's transcript is 1500bp long it could divide into ten 150bp reads equaling a count of 10 for that gene. A longer gene that was 3000bp could have the same number of total transcripts but could divide into twenty 150bp reads of the same length. A simple way to scale for these two issues at the same time is to calculate RPKM. RPKM is the Reads Per Kilobase of gene per Million mapped reads. Nowadays there are also more sophisiticated methods available but RPKM is a good approximation within the limits of this course. We will later use DeSeq2 which employs a more advanced technique.

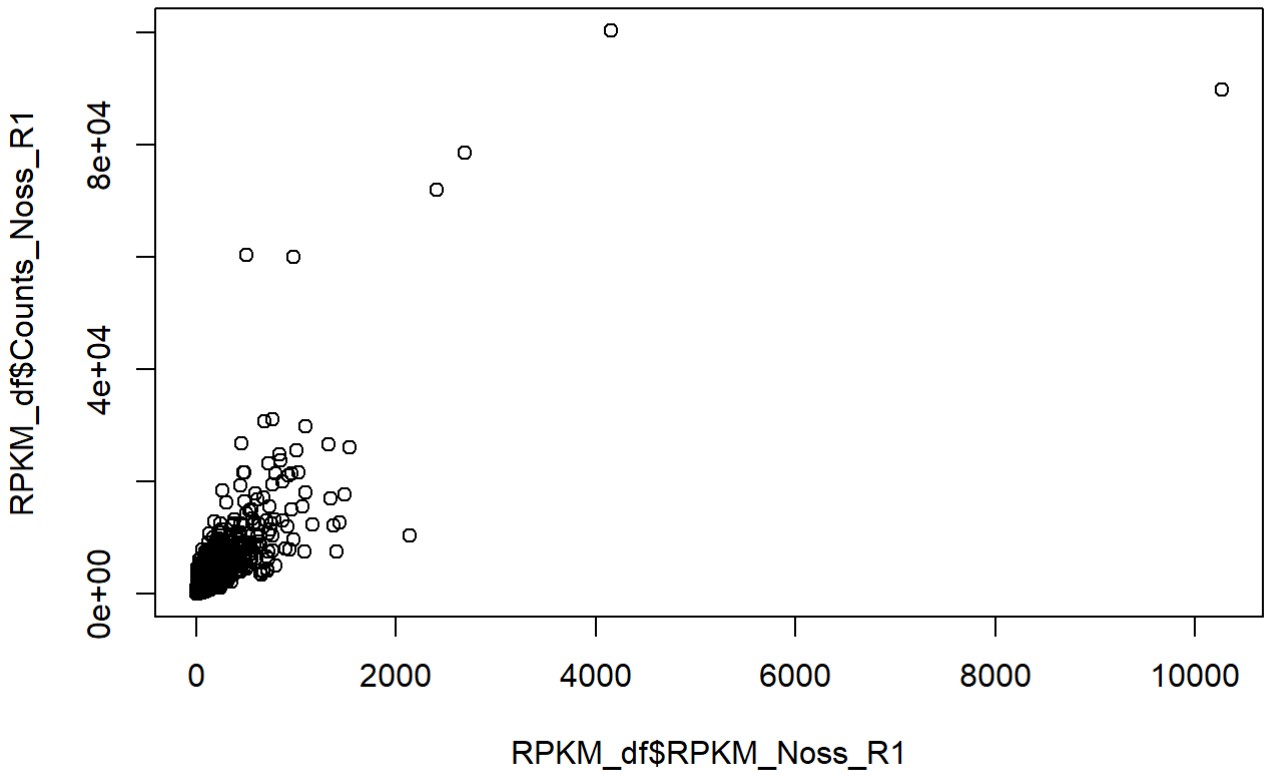
RPKM=(Counts/gene size in kb)/(number of mapped reads/1 million)

In the next section of R code: 1) Load the A. halleri gene annotation data and merge with the counts dataframe generating a new dataframe called "RPKM_df". 2) Calculate the RPKM of all counts adding them to a new column each called e.g. "Noss_R1_RPKM" 3) Plot the relationship between gene size and RPKM 4) Explore the results

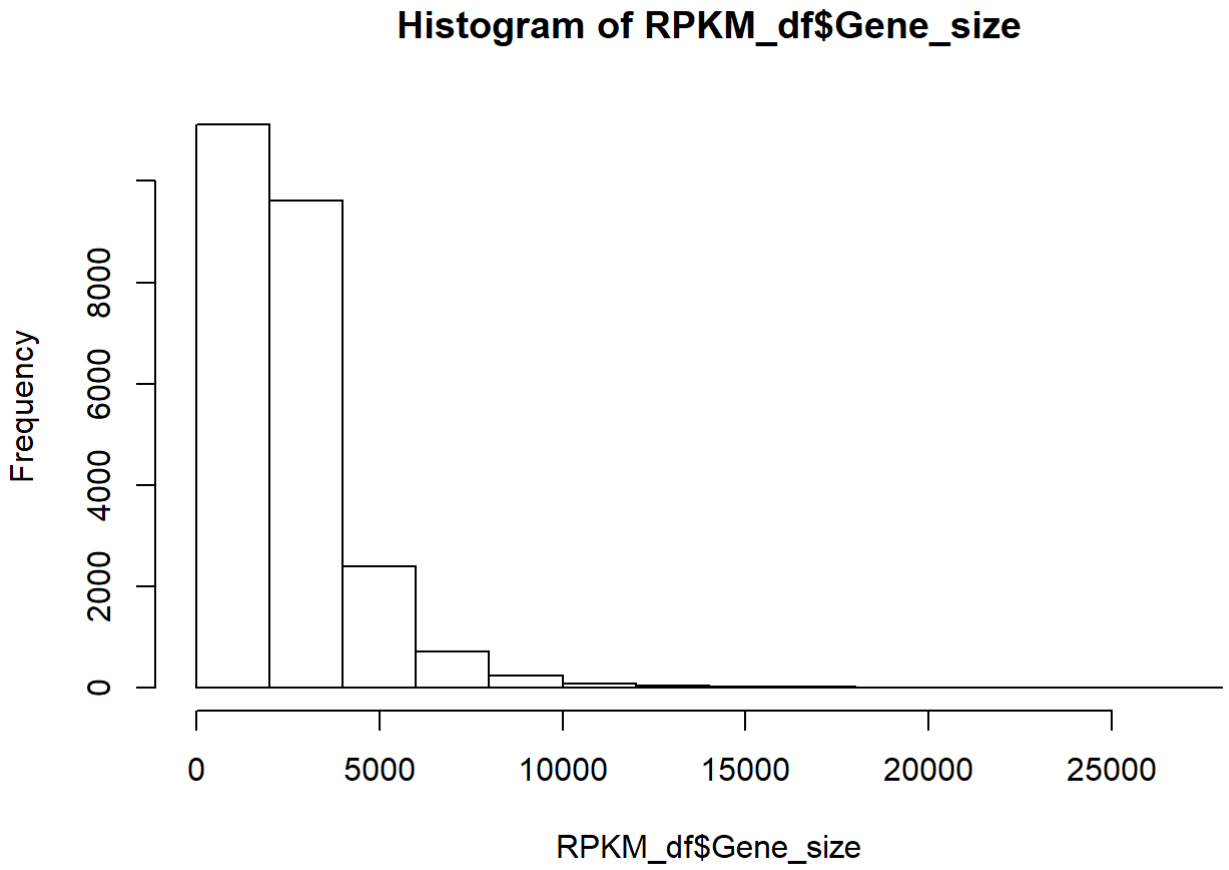
```
# 1) Load and merge
    # First load the A. halleri gene annotation data with A. thaliana orthologues (HallerotoT
halianaGenes.csv) with the Import Dataset button and copy and paste the code here:

Hatoath<-read.table("HalleritoThalianaGenes.csv",sep=",",header=T)

    # Then merge with the Counts dataframe using the column "Gene"
  RPKM_df = merge(Counts,Hatoath,by="Gene")

# 2) Calculate for all replicates
  # RPKM=(Counts/(gene size/1000))/(sum mapped reads/1 million)
  # The function sum() might be helpful here
RPKM_df$RPKM_Noss_R1=(RPKM_df$Counts_Noss_R1/(RPKM_df$Gene_size/1000))/(sum(RPKM_df$Counts_No
ss_R1)/1000000)
RPKM_df$RPKM_Noss_R2=(RPKM_df$Counts_Noss_R2/(RPKM_df$Gene_size/1000))/(sum(RPKM_df$Counts_No
ss_R2)/1000000)
RPKM_df$RPKM_Noss_R3=(RPKM_df$Counts_Noss_R3/(RPKM_df$Gene_size/1000))/(sum(RPKM_df$Counts_No
ss_R3)/1000000)
RPKM_df$RPKM_Pais_R1=(RPKM_df$Counts_Pais_R1/(RPKM_df$Gene_size/1000))/(sum(RPKM_df$Counts_Pa
is_R1)/1000000)
RPKM_df$RPKM_Pais_R2=(RPKM_df$Counts_Pais_R2/(RPKM_df$Gene_size/1000))/(sum(RPKM_df$Counts_Pa
is_R2)/1000000)
RPKM_df$RPKM_Pais_R3=(RPKM_df$Counts_Pais_R3/(RPKM_df$Gene_size/1000))/(sum(RPKM_df$Counts_Pa
is_R3)/1000000)

  # 3) Plot gene size vs RPKM and Counts_Noss_R1 vs RPKM
plot(RPKM_df$Gene_size~RPKM_df$RPKM_Noss_R1)
```



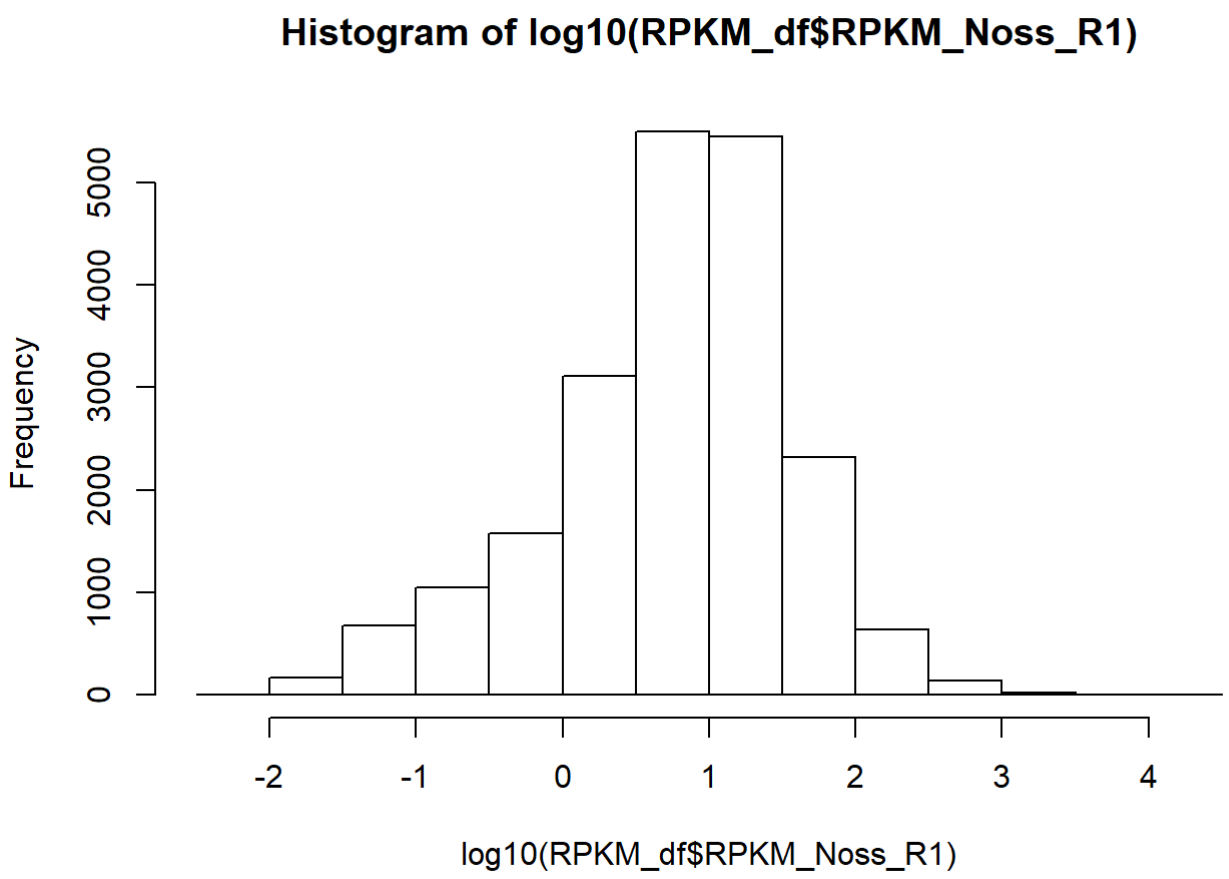

```
plot(RPKM_df$Counts_Noss_R1~RPKM_df$RPKM_Noss_R1)
```




```
# 4) Explore
  # What do you see? Is gene size related to expression?

  # Create a histogram of gene size. Approximately what is the size of the average A. halleri
gene?
hist(RPKM_df$Gene_size)
```




```
  # Create a histogram of expression (RPKM) using a log10 transformation of the Noss_R1_RPKM
 column.
hist(log10(RPKM_df$RPKM_Noss_R1))
```




```
 # What genes have the highest expression (RPKM) for Noss_R1?
tail(RPKM_df[order(RPKM_df$RPKM_Noss_R1),])
```

```
##         Gene Counts_Noss_R1 Counts_Noss_R2 Counts_Noss_R3 Counts_Pais_R1
## 4934  g05847          26109          12928           8900          12045
## 6234  g07445          10384           2803            977           8612
## 567   g00648          71862          39546           9528          61436
## 4002  g04721          78479          48491          28053          53787
## 1086  g01268         100225          63082          27745          84473
## 15084 g18687          89737          85758          59568          78473
##       Counts_Pais_R2 Counts_Pais_R3    Scaffold Gene_Start Gene_End Score
## 4934            2036           6327 scaffold_10     871127   872506  0.58
## 6234            1735           2102 scaffold_14    1253700  1254094  0.65
## 567            14111          21321  scaffold_1    2933343  2935753  0.30
## 4002           21368          29748  scaffold_8     341028   343392  0.31
## 1086           16973          25582  scaffold_2    1695825  1697779  0.42
## 15084          35661          72848 scaffold_72      72384    73091  0.36
##       Strand note                                Function Additional_info
## 4934       -    .  Universal stress protein family protein     ahrd-qc=***
## 6234       +    .                         Unknown protein
## 567        +    .                Pyruvate decarboxylase 1     ahrd-qc=***
## 4002       +    . Glyceraldehyde-3-phosphate dehydrogenase     ahrd-qc=***
## 1086       +    .                   Alcohol dehydrogenase     ahrd-qc=***
## 15084      -    .              Cysteine-rich venom protein     ahrd-qc=***
##       Gene_size A_thaliana_orthologue RPKM_Noss_R1 RPKM_Noss_R2 RPKM_Noss_R3
## 4934       1379             AT3G11930     1531.913     780.3264     577.0882
## 6234        394             AT5G65207     2132.439     592.1560     221.7251
## 567        2410             AT4G33070     2412.628    1365.8240     353.5096
## 4002       2364             AT1G13440     2686.050    1707.3513    1061.0804
## 1086       1954             AT1G77120     4150.111    2687.1389    1269.6284
## 15084       707             AT4G33710    10269.760   10096.3514    7533.7304
##       RPKM_Pais_R1 RPKM_Pais_R2 RPKM_Pais_R3
## 4934      713.7136     144.8716     519.4213
## 6234     1786.0320     432.0888     603.9802
## 567      2082.9906     574.5271    1001.5595
## 4002     1859.1365     886.9234    1424.6117
## 1086     3532.4399     852.3222    1482.1636
## 15084    9069.4775    4949.3006   11664.9917
```

```
  # What is the largest gene in the annotation? Is it expressed?
RPKM_df[RPKM_df$Gene_size==max(RPKM_df$Gene_size),]
```

```
##         Gene Counts_Noss_R1 Counts_Noss_R2 Counts_Noss_R3 Counts_Pais_R1
## 23336 g30473            801            690            791            799
##       Counts_Pais_R2 Counts_Pais_R3     Scaffold Gene_Start Gene_End Score
## 23336            767            598 scaffold_357      31086    57659  0.37
##       Strand note                            Function Additional_info Gene_size
## 23336      +    . Serine/threonine-protein kinase ATM     ahrd-qc=***     26573
##       A_thaliana_orthologue RPKM_Noss_R1 RPKM_Noss_R2 RPKM_Noss_R3 RPKM_Pais_R1
## 23336             AT3G48190     2.438934     2.161313     2.661659     2.456901
##       RPKM_Pais_R2 RPKM_Pais_R3
## 23336     2.832204     2.547691
```

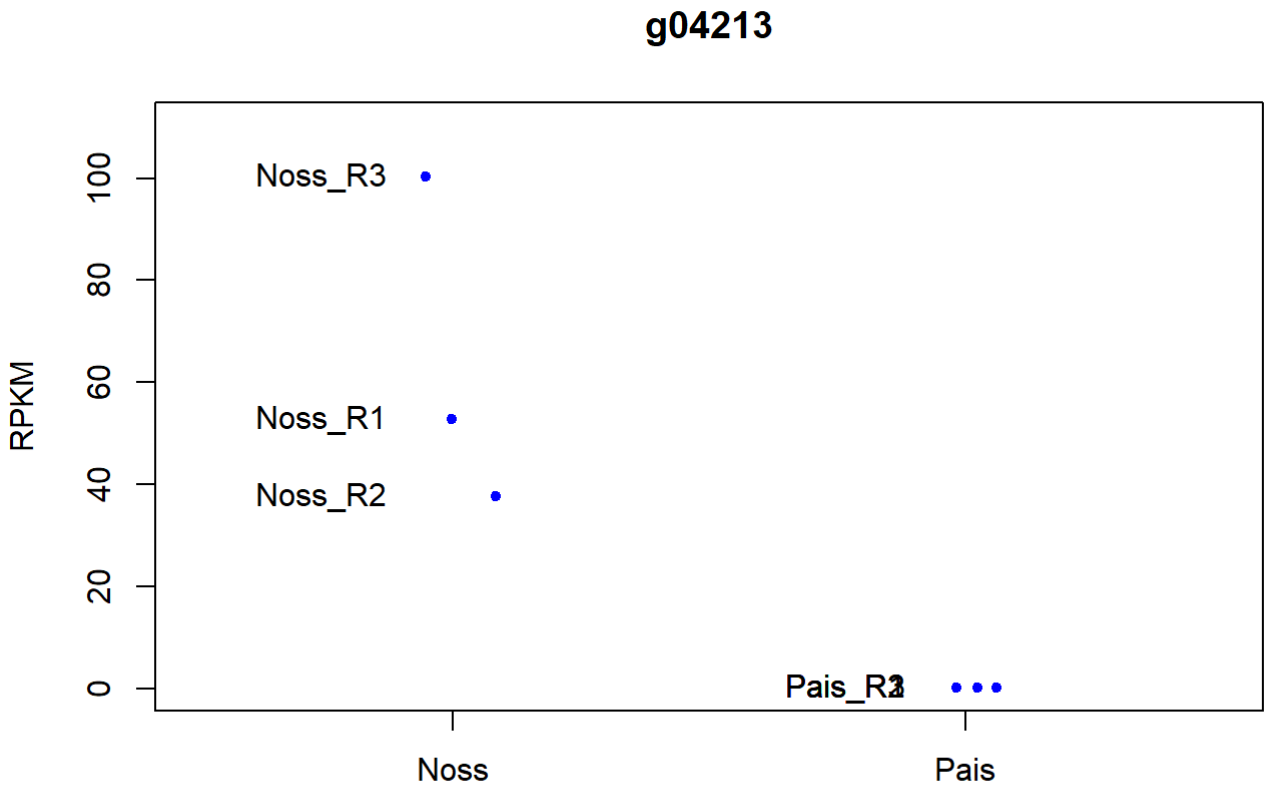
```
  # Are the highest expressed genes the same in Noss_R1 and Pais_R1 conditions?
tail(RPKM_df[order(RPKM_df$RPKM_Pais_R1),])
```

```
##         Gene Counts_Noss_R1 Counts_Noss_R2 Counts_Noss_R3 Counts_Pais_R1
## 4002  g04721          78479          48491          28053          53787
## 2785  g03287           9747          11634           7206          18801
## 567   g00648          71862          39546           9528          61436
## 504   g00571          12747          25088          32128          29655
## 1086  g01268         100225          63082          27745          84473
## 15084 g18687          89737          85758          59568          78473
##       Counts_Pais_R2 Counts_Pais_R3    Scaffold Gene_Start Gene_End Score
## 4002           21368          29748  scaffold_8     341028   343392  0.31
## 2785           17655           7286  scaffold_5     939896   940702  0.95
## 567            14111          21321  scaffold_1    2933343  2935753  0.30
## 504            44088          26607  scaffold_1    2649539  2650259  0.91
## 1086           16973          25582  scaffold_2    1695825  1697779  0.42
## 15084          35661          72848 scaffold_72      72384    73091  0.36
##       Strand note                                Function Additional_info
## 4002       +    . Glyceraldehyde-3-phosphate dehydrogenase     ahrd-qc=***
## 2785       -    .         Auxin-repressed 12.5 kDa protein     ahrd-qc=***
## 567        +    .                Pyruvate decarboxylase 1     ahrd-qc=***
## 504        -    .              Cysteine-rich venom protein     ahrd-qc=*-*
## 1086       +    .                   Alcohol dehydrogenase     ahrd-qc=***
## 15084      -    .              Cysteine-rich venom protein     ahrd-qc=***
##       Gene_size A_thaliana_orthologue RPKM_Noss_R1 RPKM_Noss_R2 RPKM_Noss_R3
## 4002       2364             AT1G13440    2686.0504     1707.351    1061.0804
## 2785        806             AT2G33830     978.4623     1201.443     799.4212
## 567        2410             AT4G33070    2412.6283     1365.824     353.5096
## 504         720             AT4G33720    1432.4638     2900.299    3989.9520
## 1086       1954             AT1G77120    4150.1106     2687.139    1269.6284
## 15084       707             AT4G33710   10269.7601    10096.351    7533.7304
##       RPKM_Pais_R1 RPKM_Pais_R2 RPKM_Pais_R3
## 4002      1859.136     886.9234     1424.612
## 2785      1906.019    2149.3266     1023.388
## 567       2082.991     574.5271     1001.559
## 504       3365.479    6008.3833     4183.595
## 1086      3532.440     852.3222     1482.164
## 15084     9069.477    4949.3006    11664.992
```

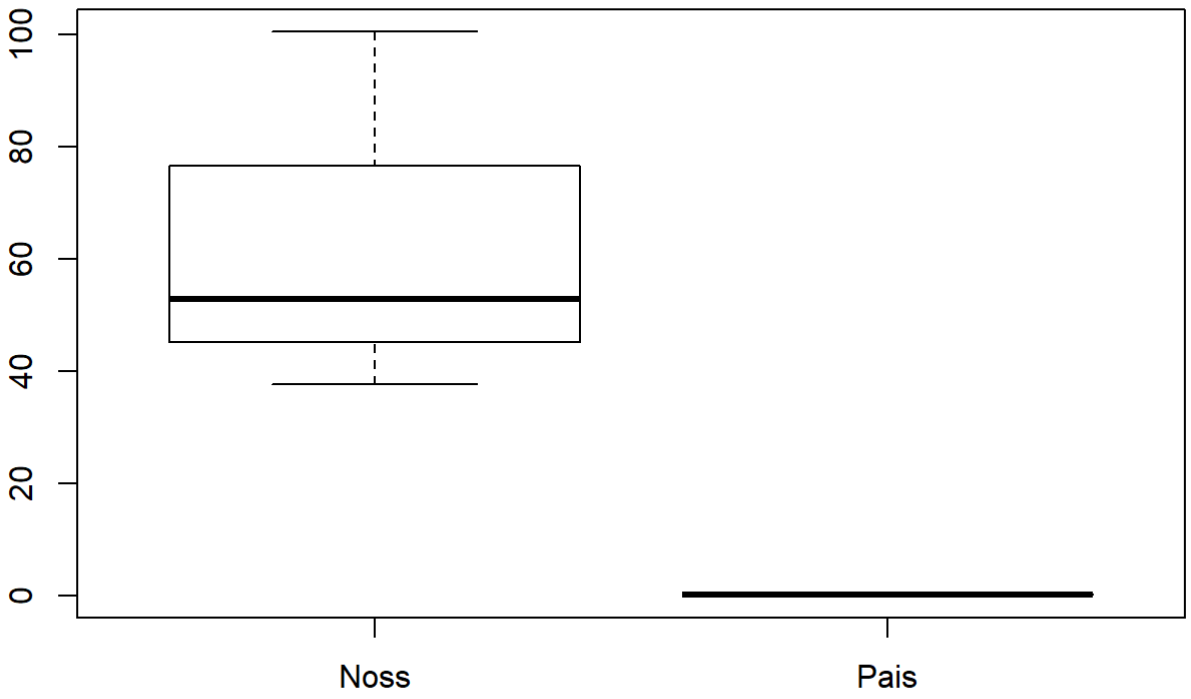
Significantly Different Expression?

In the following section we create a dataframe with RPKM data already calculated for two genes:g01794 and g04213. We will then plot the expression of each of these genes in our two populations and perform a Two Sample t-test to see if there is a significant difference in the expression.

```
# Run the following lines to build a dataframe called RPKM_df2
Population= c("Noss","Noss","Noss","Pais","Pais","Pais")
Rep= c("Noss_R1","Noss_R2","Noss_R3","Pais_R1","Pais_R2","Pais_R3")
g01794= c(10.53482, 19.88798, 19.00557, 12.80166, 13.49514, 16.33977)
g04213 =c(52.80253, 37.53903, 100.4055, 0.178579, 0.07127946, 0.1240432)
RPKM_df2=data.frame(Population,Rep, g01794, g04213)

# Run the following lines to plot the expression of g04213 in this data
stripchart(g04213 ~ Population, vertical = TRUE, data = RPKM_df2, pch = 20, main="g04213", yl
ab="RPKM", col = "blue", method="jitter", xlim=c(0.5,2.5), ylim=c(.9*min(g04213), 1.1*max(g04
213)))
text(RPKM_df2$g04213 ~ RPKM_df2$Population, labels=RPKM_df2$Rep, adj=1.5 )
```




```
boxplot(RPKM_df2$g04213 ~ RPKM_df2$Population)
```



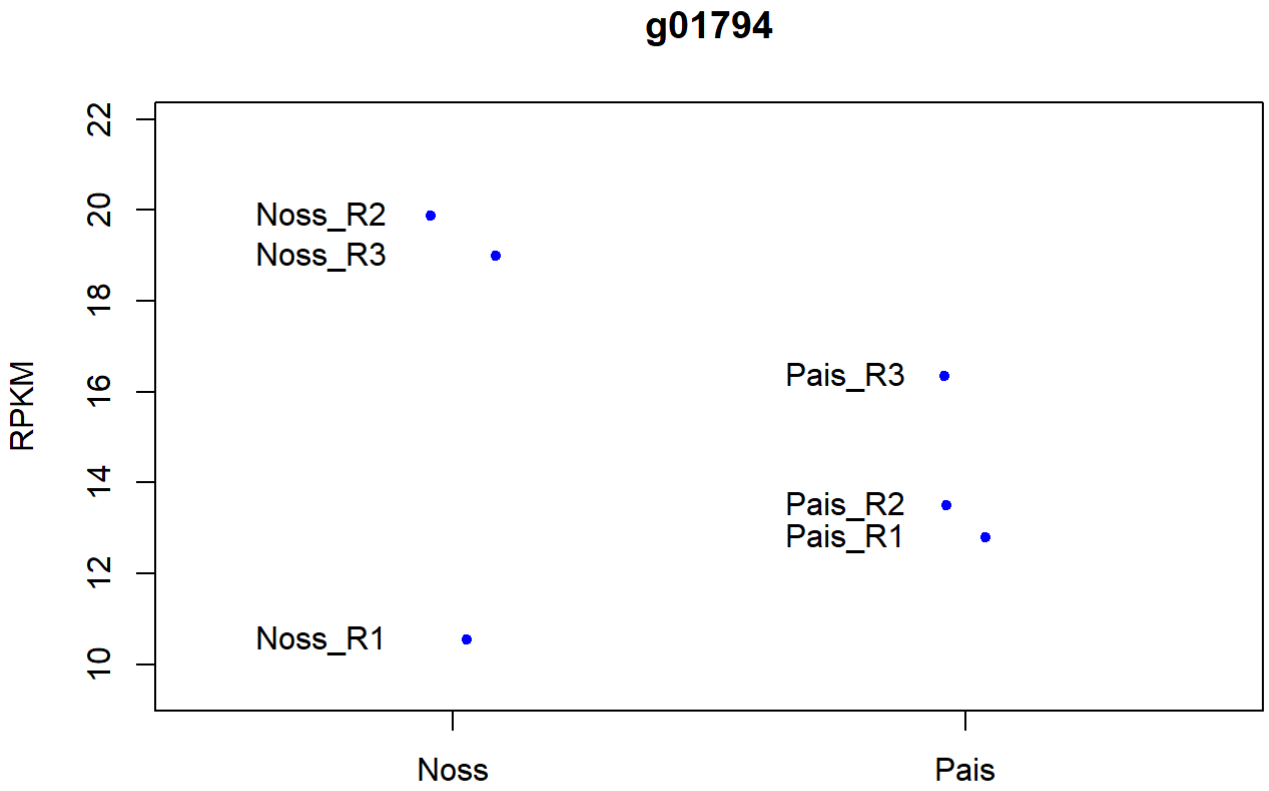
```
# As you can see, the three repetitions from the plants from Noss express this gene higher th
an the three repetitions from plants from Pais.

#Now we want to see if the mean expression between Noss and Pais is significantly different f
or this gene. In earlier times, a two sample t-test was used for that.

t.test(RPKM_df2$g04213 ~ RPKM_df2$Population)
```

```
##
##  Welch Two Sample t-test
##
## data:  RPKM_df2$g04213 by RPKM_df2$Population
## t = 3.352, df = 2, p-value = 0.07865
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -17.99751 144.91295
## sample estimates:
## mean in group Noss mean in group Pais
##         63.5823533          0.1246339
```

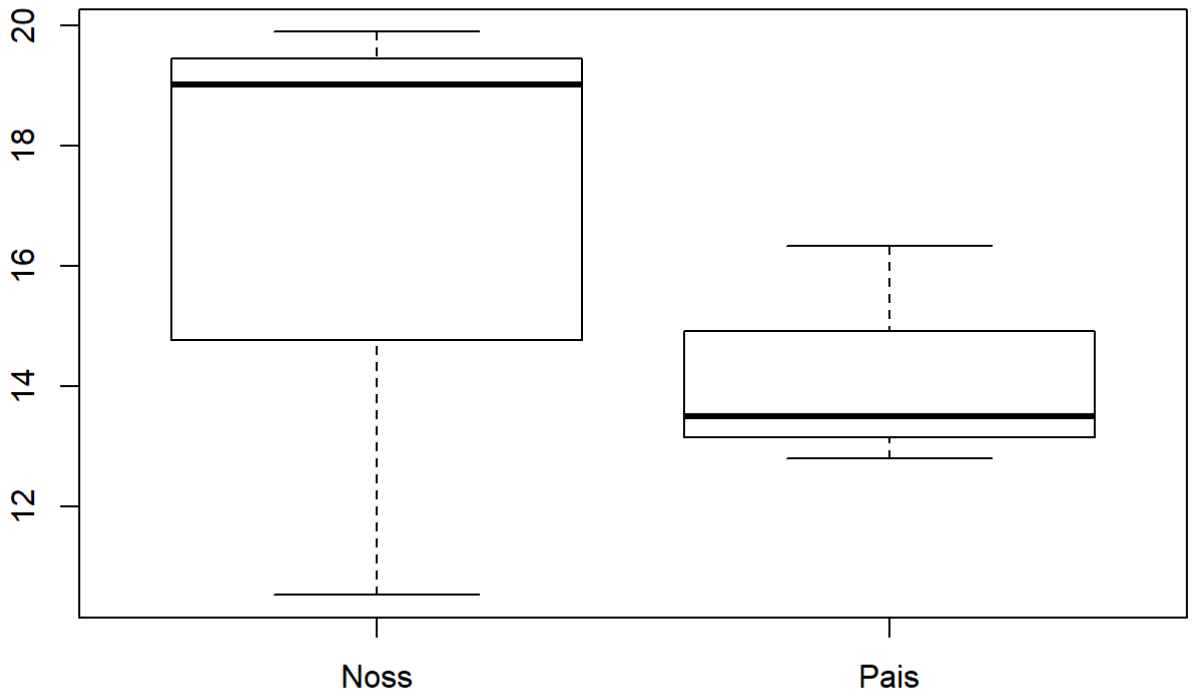
```
#In the output you see the p-value score. This is the probablity of observing this difference
in means or a greater difference by chance alone. Traditionally a p-value of less than 0.05 i
s considered to be a "significant" difference. According to this test the expression is not s
ignificantly different. However, the t-test relies on a lot of assumptions, e.g. normal distr
ibution and equal variances. You can ckeck if the variances are equal with the following func
tion:

var.test(RPKM_df2$g04213 ~ RPKM_df2$Population)
```

```
##
##  F test to compare two variances
##
## data:  RPKM_df2$g04213 by RPKM_df2$Population
## F = 373520, num df = 2, denom df = 2, p-value = 5.354e-06
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##      9577.455 14567309.263
## sample estimates:
## ratio of variances
##           373520.8
```

```
#The p-value tells us that the variances are not equal, therefore we can not use the t-test.
 We will employ DeSeq2 in the next section, which uses the Wald-test, which is able to incorp
orate differences in variances.

## Now time for g01794

# Run the following lines to plot the expression of g01794 in this data
stripchart(g01794 ~ Population, vertical = TRUE, data = RPKM_df2, pch = 20,main="g01794", yla
b="RPKM", col = "blue", xlim=c(0.5,2.5), method="jitter", ylim=c(.9*min(g01794), 1.1*max(g017
94)))
text(RPKM_df2$g01794 ~ RPKM_df2$Population , labels=RPKM_df2$Rep, adj=1.5)
```




```
boxplot(RPKM_df2$g01794 ~ RPKM_df2$Population)
```



```
#The plots show that the average expression of this gene is approximatly the same in Noss and Pais.

t.test(RPKM_df2$g01794 ~ RPKM_df2$Population)
```

```
##
##  Welch Two Sample t-test
##
## data:  RPKM_df2$g01794 by RPKM_df2$Population
## t = 0.71373, df = 2.5182, p-value = 0.5358
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -9.016255 13.544122
## sample estimates:
## mean in group Noss mean in group Pais
##           16.47612           14.21219
```

```
var.test(RPKM_df2$g01794 ~ RPKM_df2$Population)
```

```
##
##  F test to compare two variances
##
## data:  RPKM_df2$g01794 by RPKM_df2$Population
## F = 7.5868, num df = 2, denom df = 2, p-value = 0.2329
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##    0.1945336 295.8856176
## sample estimates:
## ratio of variances
##           7.586811
```

```
# The t-test tell us there is not a significant difference between these two conditions for this gene and here the assumption of equal variances is met.

# Expression studies often refer to a "log2 fold change". We can calculate that!
  # Step 1: Calculate the mean expression for each condition
g04213_Noss_Mean = mean(RPKM_df2[RPKM_df2$Population=="Noss",]$g04213)
g04213_Pais_Mean = mean(RPKM_df2[RPKM_df2$Population=="Pais",]$g04213)

  # Step 2: Calculate the ratio of the expression between the two populations
g04213_Ratio= g04213_Noss_Mean/g04213_Pais_Mean
g04213_Ratio #This is the "fold change" difference between the expression in these two populations.
```

```
## [1] 510.153
```

```
  #Step 3: Take the log2 of this ratio to get a "log2fold change"
log2(g04213_Ratio)
```

```
## [1] 8.994786
```

```
#Result: In Noss we found a 8.99 log2fold change in the expression of g04213.

#Now calculate the same for g01794.
log2(mean(RPKM_df2[RPKM_df2$Population=="Noss",]$g01794)/mean(RPKM_df2[RPKM_df2$Population==
"Pais",]$g01794))
```

```
## [1] 0.2132479
```

If you like, you can repeat this t-test and log2fold change calculation for the remaining genes. I would not suggest doing it this way. Alternatively, and more acurately, you can use DESeq2 to explore all the genes at the same time and determine which genes are differentially expressed between these two conditions.

##Time for DESeq2! This week you have heard the term “Differentially Expressed Genes”. This is a unique status given to only genes with a significant difference in their expression between the two populations. While the scaled values of an RPKM are excellent for basic summaries of expression, researchers have found that additional calibration is best included in the estimation of differential expression. These adjusments are already baked into the DESeq2 software. We just need to format our data to their specifications.

#Convert the counts dataframe to a count matrix The function DESeqDataSetFromMatrix() takes three inputs: countData, coldata, and design

```
# The countData is the Counts dataframe we have already been working with.

# The coldata is a dataframe that informs DESeq2 of our experimental design. Run the followin
g two lines to generate the coldata dataframe.

Population = factor(c(rep("Noss", 3), rep("Pais", 3)))
coldata = data.frame(row.names=colnames(Counts[2:7]), Population)

# Confirm that this new dataframe matches the column names of our count data and is in the sa
me order as the columns.
coldata
```

```
##                Population
## Counts_Noss_R1       Noss
## Counts_Noss_R2       Noss
## Counts_Noss_R3       Noss
## Counts_Pais_R1       Pais
## Counts_Pais_R2       Pais
## Counts_Pais_R3       Pais
```

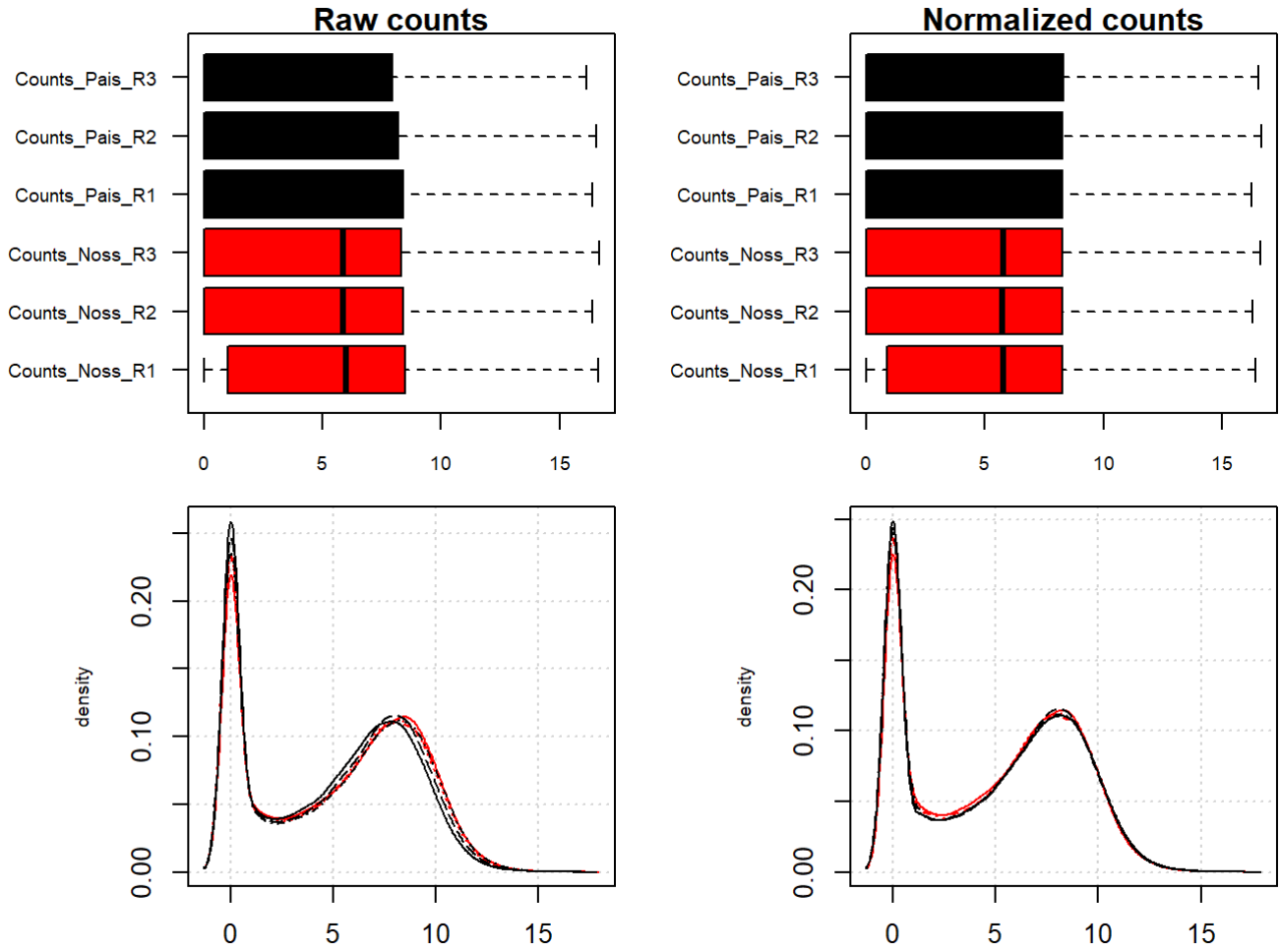
```
# Lastly is the design. For this experiment we are only interested in exploring the effect of
populations and therefore annotate design as ~Population. We then load the data with the func
tion DESeqDataSetFromMatrix() below and assign it the name dds.

dds = DESeqDataSetFromMatrix(countData=Counts, colData=coldata, design = ~Population,  tidy =
TRUE, ignoreRank = FALSE)
dds$Population <- factor(dds$Population, levels = c("Pais","Noss"))

## Check the general properties of the DESeq dataset
print(dds)
```

```
## class: DESeqDataSet
## dim: 32553 6
## metadata(1): version
## assays(1): counts
## rownames(32553): g00001 g00002 ... g32552 g32553
## rowData names(0):
## colnames(6): Counts_Noss_R1 Counts_Noss_R2 ... Counts_Pais_R2
##   Counts_Pais_R3
## colData names(1): Population
```

```
#### Normalization
# Normalizing for different numbers of aligned reads per library
dds.norm <-  estimateSizeFactors(dds)
sizeFactors(dds.norm)
```

```
## Counts_Noss_R1 Counts_Noss_R2 Counts_Noss_R3 Counts_Pais_R1 Counts_Pais_R2
##      1.1558284      1.0788687      1.0459773      1.1019961      0.9455573
## Counts_Pais_R3
##      0.7710586
```

```
# Checking the normalization
par(mfrow=c(2,2),mar=c(2,7,1,0.5),cex.lab=0.7)#set parameters for the plotting window
epsilon <- 1 # pseudo-count to avoid problems with log(0)
boxplot(log2(counts(dds.norm)+epsilon),col=dds$Population, cex.axis=0.7,
        las=1, xlab="log2(counts+1)", horizontal=TRUE, main="Raw counts")
boxplot(log2(counts(dds.norm, normalized=TRUE)+epsilon), col=dds$Population,cex.axis=0.7,
        las=1, xlab="log2(normalized counts)", horizontal=TRUE, main="Normalized counts")
plotDensity(log2(counts(dds.norm)+epsilon),
            xlab="log2(counts+1)", col=dds$Population,cex.lab=0.7, panel.first=grid())
plotDensity(log2(counts(dds.norm, normalized=TRUE)+epsilon),
            xlab="log2(normalized counts)",col=dds$Population, cex.lab=0.7, panel.first=grid
())
```




```
# Restore default parameters
par(mfrow=c(1,1), cex.lab=1,mar=c(5.1, 4.1, 4.1, 2.1))

# Performing estimation of dispersion parameter to account for different variances within pop
ulations for each gene
dds.disp <- estimateDispersions(dds.norm, fitType='local')
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
#### Calculate Differential Expression
alpha <- 0.05 #significance level for adjusted p-value, default is 0.1
waldTestResult <- nbinomWaldTest(dds.disp)
resultDESeq2 <- results(waldTestResult, alpha=alpha, pAdjustMethod="BH")

head(resultDESeq2)
```

```
## log2 fold change (MLE): Population Noss vs Pais
## Wald test p-value: Population Noss vs Pais
## DataFrame with 6 rows and 6 columns
##                baseMean     log2FoldChange             lfcSE                stat
##               <numeric>          <numeric>         <numeric>           <numeric>
## g00001 292.415619290729  0.125749456257421 0.159428446934926  0.788751685630789
## g00002 98.4398836712859 -0.127324691590693 0.252328335413533 -0.504599261046227
## g00003  20.459008238116  0.641953573373225 0.470944335731203   1.36311985232078
## g00004 13.0144176234505  -7.27097062022309  1.26256593688076  -5.75888387911559
## g00005 394.720674618388 -0.138190397023554 0.122470720042903  -1.12835457303709
## g00006 223.908277453116 -0.237486511066558 0.234100225880337  -1.01446510858111
##                     pvalue                 padj
##                  <numeric>            <numeric>
## g00001    0.430257152754396    0.663104998274879
## g00002    0.613840328965223    0.797147372652695
## g00003    0.172844741078142    0.383550884096319
## g00004 8.46719225505441e-09 1.75296029755631e-07
## g00005    0.259170204511947    0.492790847040914
## g00006    0.310360870296248      0.5483427106933
```

```
# Instead of normalizing and calculating differential expression stepwise, we can also run DE
Seq() on our "dds" data set.
# Run the DESeq calculation (it might take a couple of minutes)
dds = DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

DESeq2 uses the so-called Benjamini-Hochberg (BH) adjustment; in brief, this method calculates for each gene an adjusted p value which answers the following question: if one called significant all genes with a p value less than or equal to this gene's p value threshold, what would be the fraction of false positives (the false discovery rate, FDR) among them (in the sense of the calculation outlined above). These values, called the BH-adjusted p values, are given in the column padj of the results object. This multiple test correction of the p-value has to be performed when we test multiple hypotheses together, e.g. compare the expression of multiple genes between populations because with more tests the probability of false positives rises.

Inspecting the results

```
# Calling the results function extracts the estimated log2Fold Change and p values.

res = results( dds )
res
```

```
## log2 fold change (MLE): Population Noss vs Pais
## Wald test p-value: Population Noss vs Pais
## DataFrame with 32553 rows and 6 columns
##                baseMean     log2FoldChange             lfcSE               stat
##               <numeric>          <numeric>         <numeric>          <numeric>
## g00001 292.415619290729  0.124938841546534 0.196069742236019  0.637216329871738
## g00002 98.4398836712859 -0.125930980535974 0.286763795733168 -0.439145325908408
## g00003  20.459008238116  0.643923862836244 0.530429187771876   1.21396762787718
## g00004 13.0144176234505  -7.27149385704467  1.30852251354847   -5.5570261739905
## g00005 394.720674618388 -0.137596205500517 0.156783224432202 -0.877620714836222
## ...                 ...                ...               ...                ...
## g32549                0                 NA                NA                 NA
## g32550                0                 NA                NA                 NA
## g32551 24.9884378384706  -3.79368558057502 0.650845560139541  -5.82885681783196
## g32552 11.8129593495557  -2.92989229536927 0.816505297957221   -3.5883322529559
## g32553 1.26788236191877    1.2220920081517  2.13978681838286  0.571127926227387
##                    pvalue                 padj
##                 <numeric>            <numeric>
## g00001    0.523983942295063     0.77966200610239
## g00002     0.66055623825598    0.858735275200879
## g00003    0.224760086164226    0.506963473103293
## g00004 2.74409625226585e-08 6.61430001554801e-07
## g00005    0.380149583893787    0.668598852070112
## ...                     ...                  ...
## g32549                  NA                   NA
## g32550                  NA                   NA
## g32551 5.58083606269698e-09 1.49449667540227e-07
## g32552 0.000332800010589831  0.00335231548202329
## g32553    0.567912929708421    0.806715393769701
```

```
# The results file caries metadata we can view using mcols()
mcols(res, 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 (MLE): Population Noss vs Pais
## lfcSE               results         standard error: Population Noss vs Pais
## stat                results         Wald statistic: Population Noss vs Pais
## pvalue              results      Wald test p-value: Population Noss vs Pais
## padj                results                           BH adjusted p-values
```

The DESeq2 Package describes the results table as follows:

The first column, baseMean, is the average of the normalized count values, dividing by size factors, taken over all samples. The remaining four columns refer to a specific contrast, namely the comparison of the levels Pais versus Noss of the factor variable population. The column log2FoldChange is the effect size estimate. It tells

us how much the gene's expression seems to have changed due to population differences. This value is reported on a logarithmic scale to base 2: for example, a log2 fold change of 1.5 means that the gene's expression is increased by a multiplicative factor of 2^1.5 = 2.82.

Of course, this estimate has an uncertainty associated with it, which is available in the column lfcSE, the standard error estimate for the log2 fold change estimate. We can also express the uncertainty of a particular effect size estimate as the result of a statistical test. The purpose of a test for differential expression is to test whether the data provides sufficient evidence to conclude that this value is really different from zero. DESeq2 performs for each gene a hypothesis test to see whether evidence is sufficient to decide against the null hypothesis that there is no effect of the population on the gene and that the observed difference between populations was merely caused by experimental variability (i. e., the type of variability that you can just as well expect between different samples in the same population). As usual in statistics, the result of this test is reported as a p value, and it is found in the column pvalue. (Remember that a p value indicates the probability that a fold change as strong as the observed one, or even stronger, would be seen under the situation described by the null hypothesis.)

We note that a subset of the p values in res are NA ("not available""). This is DESeq's way of reporting that all counts for this gene were zero, and hence no test was applied. In addition, p values can be assigned NA if the gene was excluded from analysis because it contained an extreme count outlier. For more information, see the outlier detection section of the advanced vignette.

Sort out significance

In our case we are interested in first exploring the genes with the most significant change in expression between our two populations while also keeping the fraction of false positives low.

```
# The function sum() below counts how many genes have an adjusted p value of less than 0.1. A
djust this to only genes with a padj value less than 0.05. This is our number of differential
ly expressed genes!
sum( res$padj < 0.05, na.rm=TRUE )
```

```
## [1] 4217
```

```
  #How many differentially expressed genes are in the dataset?
    #Answer:4217

# We can create a dataframe of only those genes and view that data as follows:
resSig = res[ which(res$padj < 0.05 ), ]
View(resSig)

# Sorting this dataframe allows us to look at the genes with the strongest down-regulation
head( resSig[ order( resSig$log2FoldChange ), ] )
```

```
## log2 fold change (MLE): Population Noss vs Pais
## Wald test p-value: Population Noss vs Pais
## DataFrame with 6 rows and 6 columns
##                baseMean    log2FoldChange             lfcSE              stat
##               <numeric>         <numeric>         <numeric>         <numeric>
## g13816 956.222870083691 -13.4712728308532  1.19568636849377 -11.2665605177245
## g08786 2905.56988127731 -12.6356050372464 0.855843223893168 -14.7639248456837
## g00682 497.648121502874 -12.5301093866213  1.22969432235259 -10.1896131086052
## g03619 809.185324907922 -12.2682382974367   1.1955574663999 -10.2615212084947
## g00021 375.574694423789 -12.1242462355505  1.19804048875696 -10.1200638453631
## g23458 1818.01061763723 -11.9599390555225 0.864970279680394 -13.8269942175837
##                      pvalue                 padj
##                   <numeric>            <numeric>
## g13816 1.91921399258659e-29 4.41806582586998e-27
## g08786 2.50281684376788e-49  1.8470788307007e-46
## g00682 2.20640532072001e-24 3.84466127135463e-22
## g03619 1.05038820508076e-24 1.86924403133947e-22
## g00021  4.5012190585596e-24  7.5801737327099e-22
## g23458 1.75189225859926e-43 9.15801678182765e-41
```

```
# Or up-regulation
tail( resSig[ order( resSig$log2FoldChange ), ] )
```

```
## log2 fold change (MLE): Population Noss vs Pais
## Wald test p-value: Population Noss vs Pais
## DataFrame with 6 rows and 6 columns
##                baseMean   log2FoldChange             lfcSE              stat
##               <numeric>        <numeric>         <numeric>         <numeric>
## g32141 400.242213420486  11.982390537321  1.20242201124006  9.9652122343997
## g17236 1603.13153748053  12.144021193351   1.0497970427916 11.5679704727096
## g04939  896.25799396989 12.1831540071382  1.19890776038147 10.1618776771132
## g23681 4910.08354789353 13.1699409789948 0.873600659165259 15.0754705148448
## g21701 6214.79038261644 13.4935643178445 0.881086017498686 15.3146957843587
## g00955 2097.50475312781 14.3718690173876  1.19013890233449 12.0757913124231
##                      pvalue                 padj
##                   <numeric>            <numeric>
## g32141 2.16411153965653e-23  3.5260965424066e-21
## g17236 5.98824192957221e-31 1.47310751467476e-28
## g04939 2.93371254455034e-24 5.04196679231898e-22
## g23681 2.34836922461122e-51 1.96260559762235e-48
## g21701 6.09966022404317e-53 5.46616694077468e-50
## g00955 1.41793600860219e-33 3.95320559198292e-31
```

```
#Now add the A. halleri gene annotations and the A. thaliana orthologues by merging again. No
te that we give names for the "by" parameter here because the columns are named differently b
etween dataframes. "row.names" takes the row names (A. halleri gene identifiers) of the resSi
g dataframe.
Res_df = merge(Hatoath,as.data.frame(resSig),by.x="Gene",by.y="row.names")

#Because there is still a large number of genes differentially expressed, for this course we
 want to focus on the genes that are upregulated in Noss vs. Pais (positive log2 fold change)
and take the 6 most significant ones (lowest adjusted p-value) of these.
Res_NvsP<-Res_df[Res_df$log2FoldChange>0,]
Res_NvsP_Sig<-head(Res_NvsP[order(Res_NvsP$padj),])
Res_NvsP_Sig
```

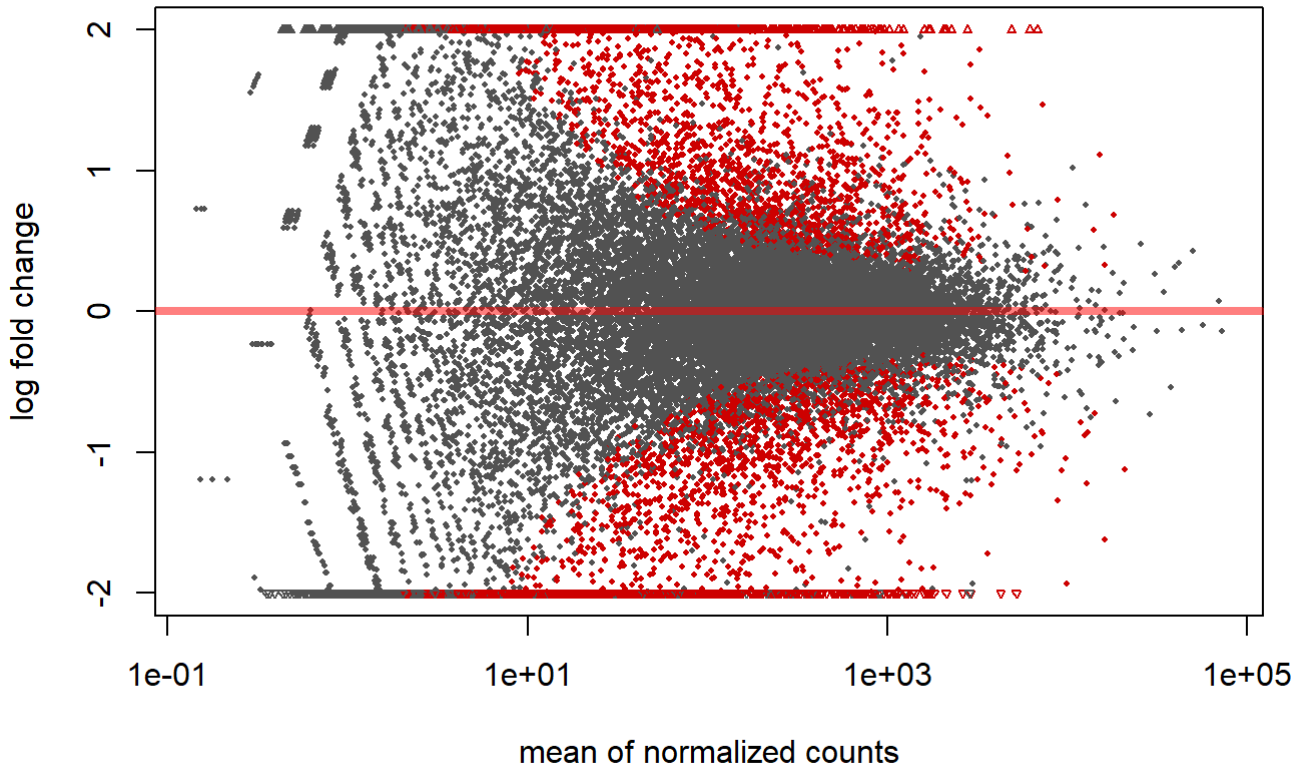
```
##        Gene     Scaffold Gene_Start Gene_End Score Strand note
## 401  g04213   scaffold_7     346001   351127  0.03      +    .
## 2614 g29390 scaffold_273          1     1527  0.59      +    .
## 1125 g12276  scaffold_31     503797   505791  0.32      +    .
## 2624 g29529 scaffold_281      69686    71433  0.71      +    .
## 510  g05342   scaffold_9     941712   942677  0.25      +    .
## 1719 g19050  scaffold_76      23199    26891  0.02      -    .
##                                      Function Additional_info Gene_size
## 401        Cadmium/zinc-transporting ATPase HMA2     ahrd-qc=***      5126
## 2614                   UPF0725 protein At2g19200     ahrd-qc=*-*      1526
## 1125                   Chaperone protein dnaJ 49     ahrd-qc=*-*      1994
## 2624                             Glutaredoxin-C2     ahrd-qc=***      1747
## 510                         AT3g62460/T12C14_160     ahrd-qc=***       965
## 1719 Pentatricopeptide repeat-containing protein     ahrd-qc=***      3692
##      A_thaliana_orthologue  baseMean log2FoldChange     lfcSE     stat
## 401              AT4G30110 2800.9558       8.917925 0.4632295 19.25164
## 2614                 <NA> 1024.5705       8.820770 0.4938545 17.86107
## 1125             AT5G62780  904.8083       5.425592 0.3599458 15.07336
## 2624             AT5G40370  961.8004       2.802270 0.1910150 14.67042
## 510              AT3G62460  435.7361       4.264175 0.3008661 14.17300
## 1719             AT5G08490  185.4775       4.431756 0.3150999 14.06461
##            pvalue         padj
## 401  1.367763e-82 3.119991e-79
## 2614 2.370507e-71 3.717547e-68
## 1125 2.424708e-51 1.962606e-48
## 2624 9.972646e-49 6.763071e-46
## 510  1.346275e-45 8.043032e-43
## 1719 6.267106e-45 3.494538e-42
```

```
#Notice that g04213 now is highly significantly differentially expressed!

# Saving the significant genes
  # You can use the write.table or write.csv function to save your list of genes as a .txt or
.csv file that can then be opened in other programs like excel.

write.csv( Res_df, file="results.csv" )

# or save a file of just your significant genes upregulated in Noss vs. Pais
write.csv( Res_NvsP_Sig, file="SignificantResults_NossvsPais.csv" )
```

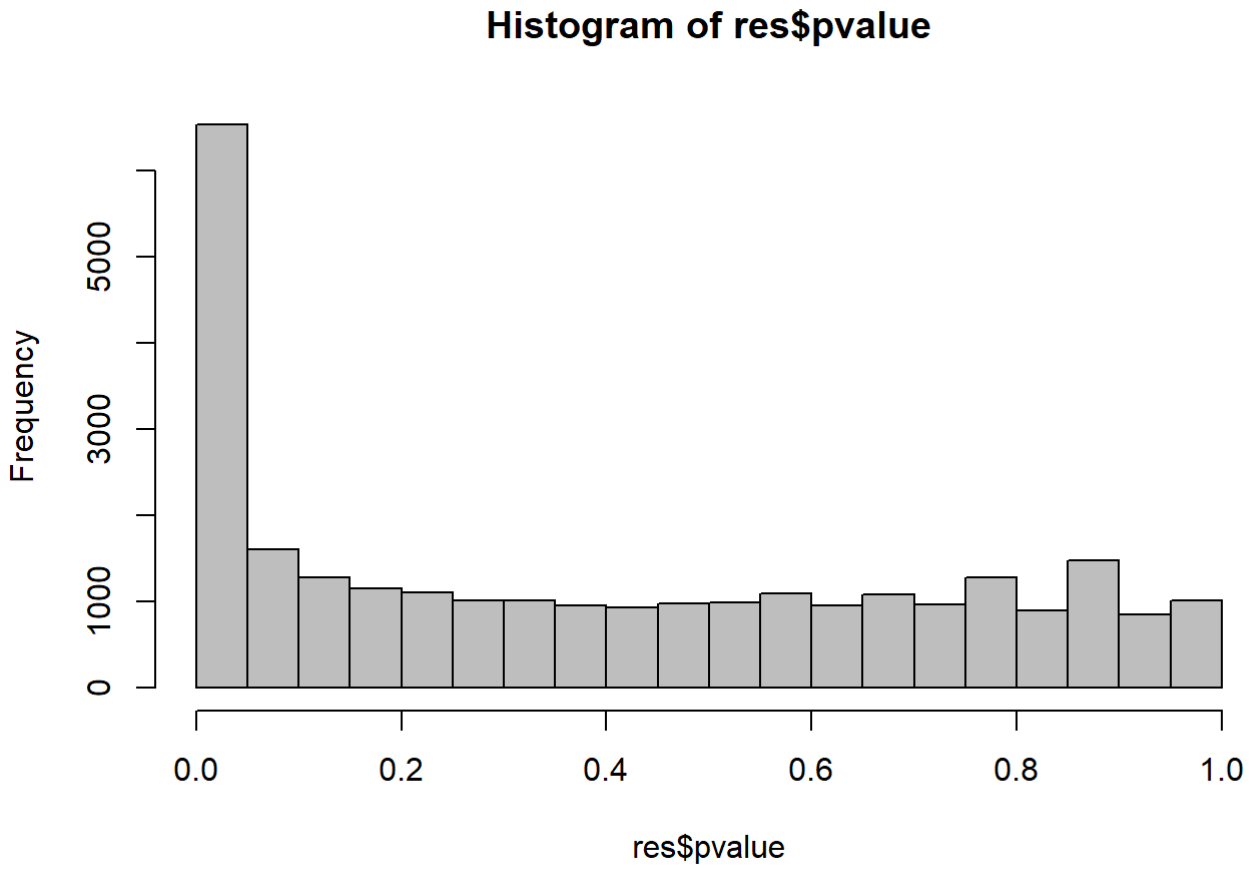
Diagnostic plots

Before exploring the list of differentially expressed genes we should run some basic diagnostics. The first is to plot mean expression vs log fold change to show our data's distribution. We want to be sure we don't have a typing error affecting our results. Second is to look at the distribution of p values to ensure they are relatively uniformly distributed.

```
# Conveniently, mean expression vs log fold change can be plotted using plotMA(). This plot r
epresents each gene with a dot. The x axis is the average expression over all samples, the y
 axis the log2 fold change between populations. Genes with an adjusted p value below a thresh
old (set by alpha) are shown in red. - DESeq2 Beginner Guide
plotMA( res,alpha =.1, ylim = c(-2, 2) )
```




```
hist( res$pvalue, breaks=20, col="grey" )
```



Gene Clustering

Sometimes expression changes of only a few genes leads to a strong phenotype. Most often however, phenotypes are polygenic and several differentially expressed genes contribute to an altered phenotype, e.g. the altered metal homeostasis of our two populations. Then it is interesting to look at the expression of several genes together. One common way of presenting expression data from multiple genes is with a heatmap. This next section of R code will walk you through the process of generating a heatmap of the differentially expressed genes.

```
# Start by adding the required packages first. You will need to install them either using bio
cLite as before or using the Tools drop down menu and clicking "Install Packages...". Then lo
ad the packages using library().

#biocLite("genefilter")
library( "genefilter" )
```

```
##
## Attaching package: 'genefilter'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     rowSds, rowVars
```

```
#BiocManager::install("gplots")
library( "gplots" )
```

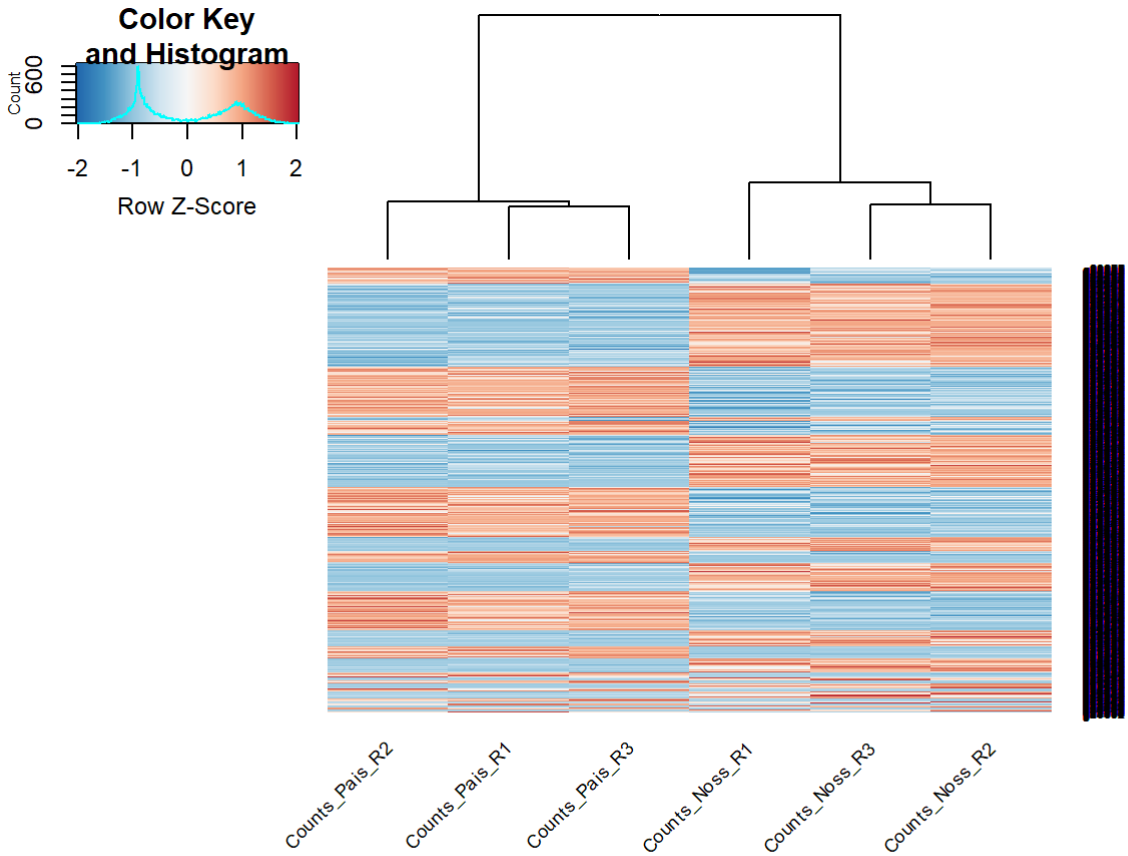
```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:IRanges':
##
##     space
```

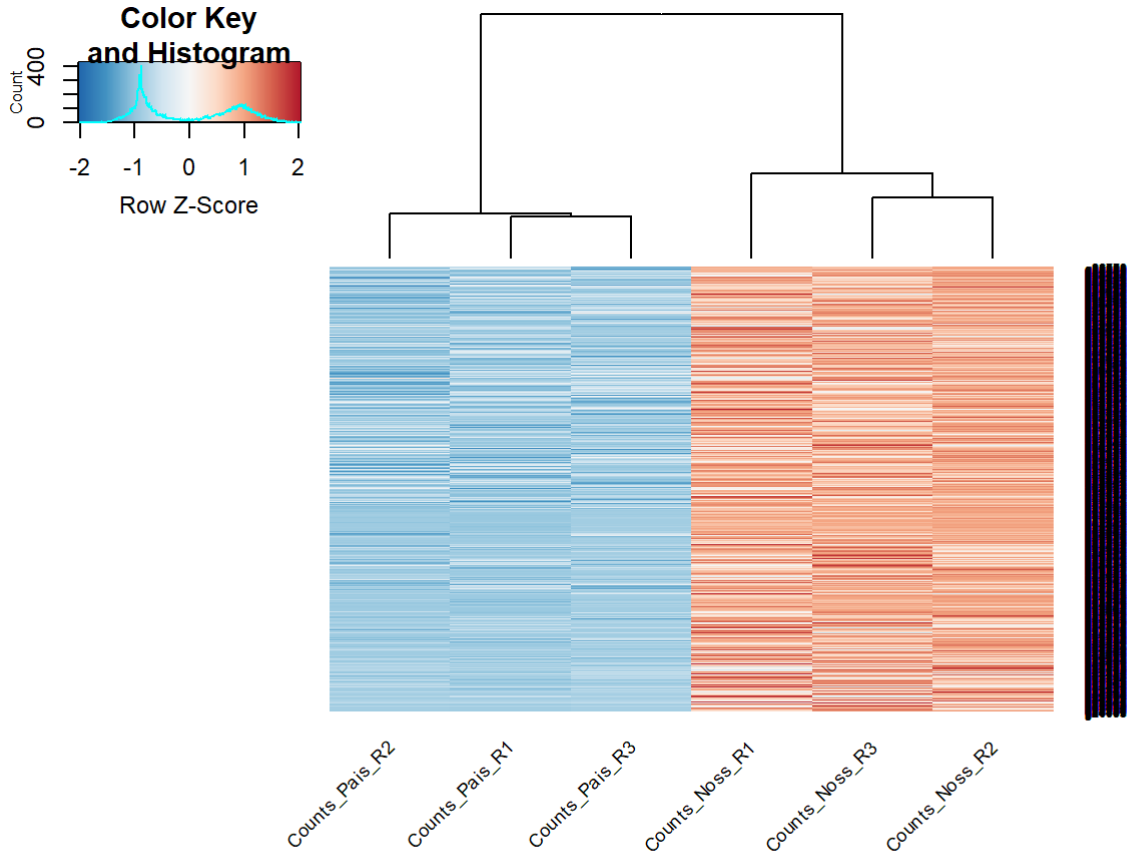
```
## The following object is masked from 'package:S4Vectors':
##
##     space
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library( "RColorBrewer" )

# Heatmaps are more comparable if the expression data has been transformed. We can transform
 our data using the function rlog()
rld = rlog( dds )

# We then generate a heatmap by assaying the rld data based on our adjusted p values. Other f
actors could also be used to subet the gene list.
heatmap.2( assay(rld)[ res$padj<0.05 & is.na(res$padj)==FALSE, ], scale="row", trace="none",
 dendrogram="column", col = colorRampPalette( rev(brewer.pal(9, "RdBu")) )(255), margins = c(
8,8), cexRow=.5, cexCol = .7, srtCol=45)
```



```
#In the top left you can see the color key, in the top right the dendrogram and beneath the h
eatmap.

heatmap.2( assay(rld)[ res$padj<0.05 & is.na(res$padj)==FALSE & res$log2FoldChange>0, ], scal
e="row", trace="none", dendrogram="column", col = colorRampPalette( rev(brewer.pal(9, "RdBu"
)) )(255), margins = c(8,8), cexRow=.5, cexCol = .7, srtCol=45)
```



You have now completed the coding part of this course.

What are these genes known for?

The next part of this analysis is to explore some of the genes you found to be differentially expressed. Because there is little information on most A. halleri genes, we will work with the A. thaliana orthologues. The two species are closely related, therefore gene function is usually similar and there is a lot more research published on A. thaliana as the model species for plant genetics. The gene names (i.e. AT4G30110) can be searched on the The Arabidopsis Information Resource (TAIR) database website: www.arabidopsis.org. On this website you will find detailed information about the gene's function, biological processes it is involved in, and publications that have helped expand our knowledge about this gene. Do the annotations on this website correspond with the biological expectations? Do any publications give more detail about the known function of these genes? Add the list of top 6 significant genes, their log2fold change, and a basic description of their annotation as a table in a results section.

Enrichment

Visualizing a heatmap and looking at the most significant genes answers the question, "What genes are expressed the most differently between these two conditions?" Alternatively, we can ask, "Of the differentially expressed genes, what gene families and biological processes do they come from?" and, "Is this a random

sampling of the known gene families and pathways or are specific groups of genes responding?" These types of questions are asking about enrichment. Conveniently, the www.arabidopsis.org website also provides an enrichment tool, although there are a number of alternative tools also available. Ideally you would choose a tool which has a good annotation for your species and phenotype. In our case data for Arabidopsis halleri or thaliana should be available and metal homeostasis and stress related genes should be well annotated. An enrichment analysis exceeds the restrictions of this course, but if you want to you can give it a try at home.

Congratulations! You have just successfully completed an RNA-seq analysis pipeline.