11_17_21_DESeq2

Kelly_F

11/17/2021

Load packages

```
library(BiocManager)
```

```
## Bioconductor version '3.13' is out-of-date; the current release version '3.14'
##   is available with R version '4.1'; see https://bioconductor.org/install
```

```
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, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

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

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## 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")'.

##
## Attaching package: 'Biobase'
```

```
## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

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

```
library(ggplot2)
```

Import countData and colData

Data used in this tutorial is from Himes et al., 2014

```
counts <- read.csv("./airway_scaledcounts.csv", row.names = 1)
mdat <- read.csv("./airway_metadata.csv")
```

Q1. How many genes are in this dataset?

38,694

Q2. How many 'control' cell lines do we have?

4

```
# Investigate the number of genes in the counts df
dim(counts)
```

```
## [1] 38694     8
```

```
# Investigate how many control cell lines we have in mdat
length(grep(pattern= "control", x=mdat$dex))
```

```
## [1] 4
```

Check the correspondence of the metadata and the count data

```
# Do the sample id names in metadata match the column names in the counts table?
# 'all' function checks if all values of logical test are TRUE

all(mdat$id == colnames(counts))
```

```
## [1] TRUE
```

Calculate mean count per gene for control and treated samples

```
# Subset control & treated samples
control <- subset(mdat, dex=="control")
treated <- subset(mdat, dex=="treated")
# control <- metadata[metadata[,"dex"]=="control",] alternative way to subset control samples

# Subset counts for controls from counts df
control.counts <- counts[, control$id]
treated.counts <- counts[, treated$id]
```

Find mean counts per gene for control and treated samples

```
# Mean control counts
control.mean <- rowMeans(control.counts)
treated.mean <- rowMeans(treated.counts)
```

> Q3. How would you make the above code in either approach more robust?

I have implemented this in my code above. Instead of using 'rowSums(control.counts)/4' you can use rowMeans(control.counts) OR rowSums(control.counts)/length(control.counts). This makes the code more robust for future use if the number of control or treated samples changes.

> Q4. Follow the same procedure for the treated samples (i.e. calculate the mean per gene across drug treated samples and assign to a labeled vector called treated.mean)

see above code.

Compare the control and treated samples

> Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

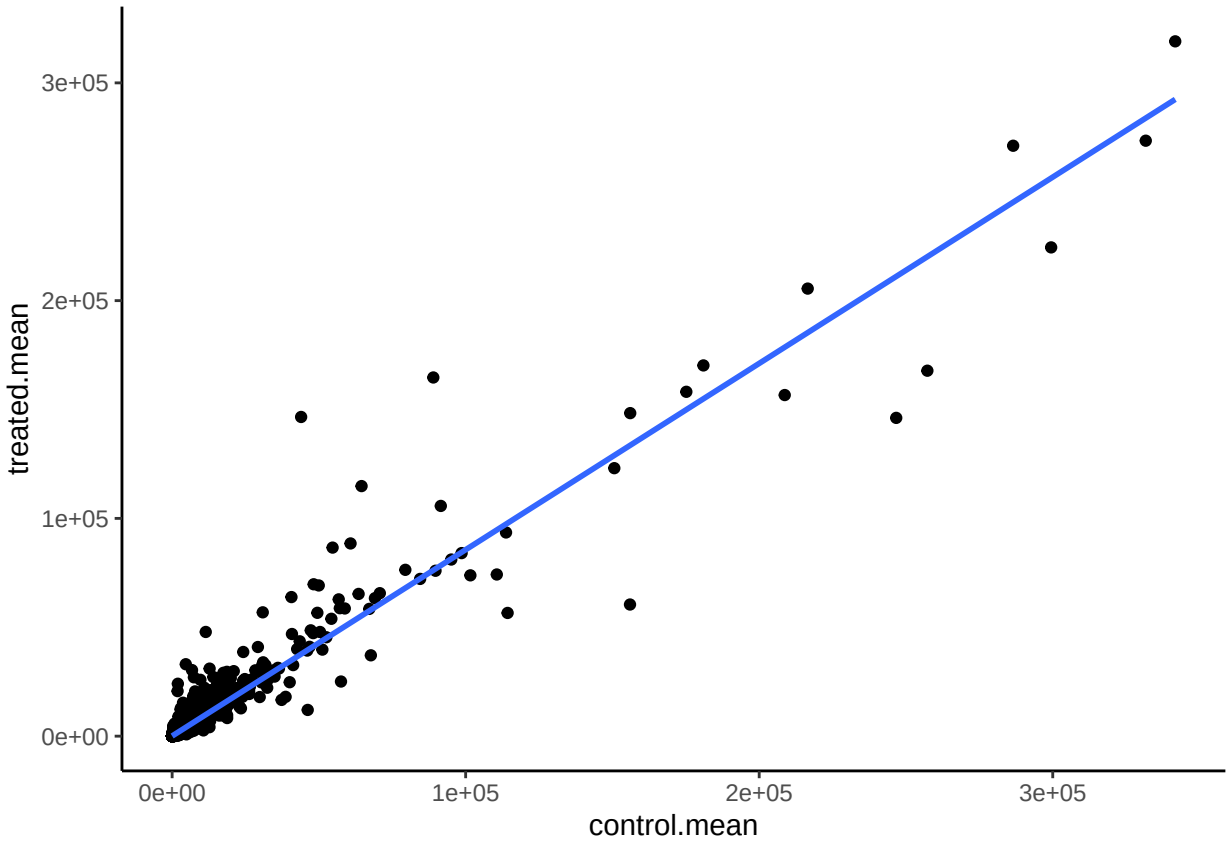
see below

> (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

geom_point (see below)

```
# Combine control and treated means
meancounts <- data.frame(control.mean, treated.mean)

# Graph
ggplot(meancounts, aes(control.mean, treated.mean)) +
  geom_point() +
  geom_smooth(method='lm', formula= y~x)+
  theme_classic()
```



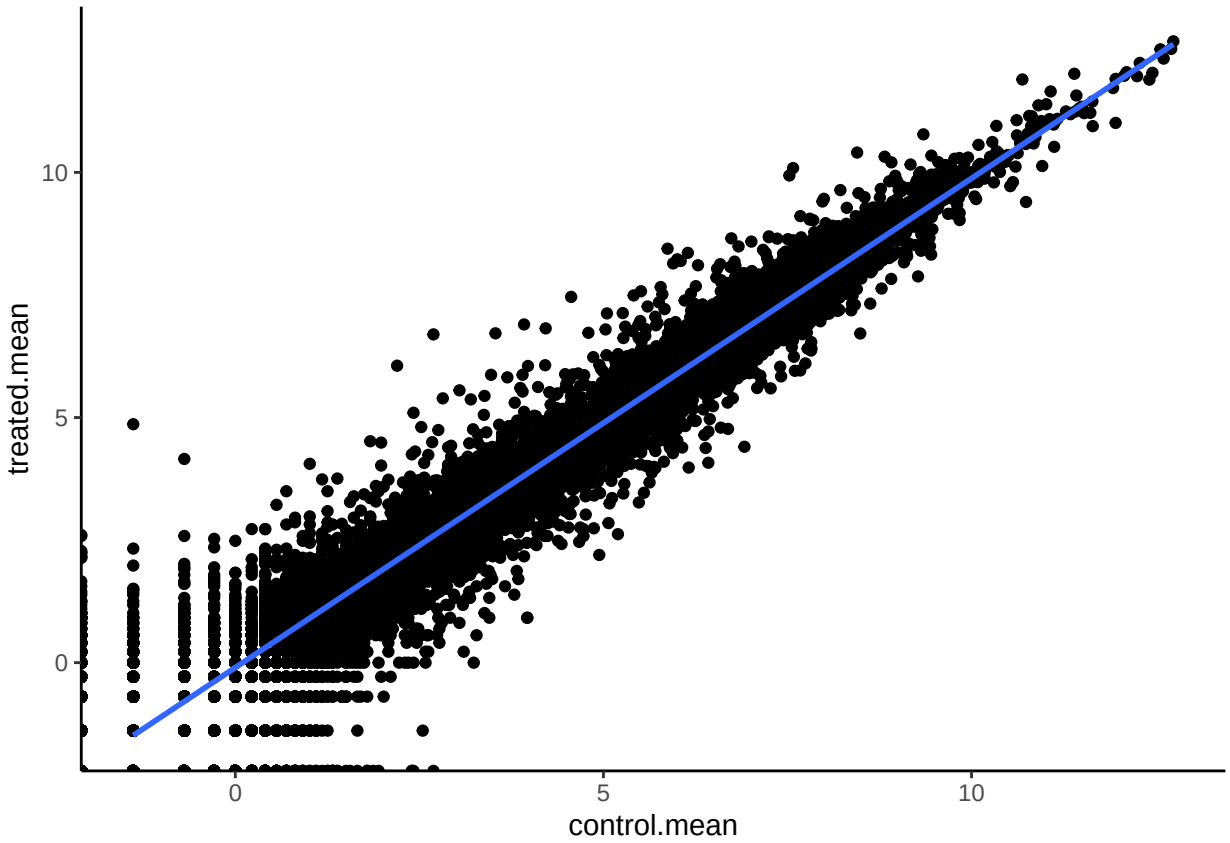
Log transform data and plot again, since the range of values for gene expression are quite large

> Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this?

log. Note, I am using ggplot instead of built in plot function.

```
# Graph
ggplot(log(meancounts), aes(control.mean, treated.mean)) +
  geom_point() +
  geom_smooth(method='lm', formula= y~x)+
  theme_classic()
```

```
## Warning: Removed 16877 rows containing non-finite values (stat_smooth).
```



Add log2 transform values to the meancounts df

```
meancounts$log2fc <- log2(meancounts$treated.mean/meancounts$control.mean)


# is.nan(meancounts[,'log2fc'])
# drop.na
```

```
# Remove zero counts to prevent non numeric values in mean counts table
zero.vals <- which(meancounts[,1:2]==0, arr.ind=TRUE)
to.rm <- unique(zero.vals[,1])
mycounts <- meancounts[-to.rm,]
head(mycounts)
```

```
##                 control.mean treated.mean      log2fc
## ENSG00000000003       900.75       658.00 -0.45303916
## ENSG00000000419       520.50       546.00  0.06900279
## ENSG00000000457       339.75       316.50 -0.10226805
## ENSG00000000460        97.25        78.75 -0.30441833
## ENSG00000000971      5219.00      6687.50  0.35769358
## ENSG00000001036      2327.00      1785.75 -0.38194109
```

Q7. What is the purpose of the arr.ind argument in the which() function call above? Why would we then take the first column of the output and need to call the unique() function?

arr.ind=TRUE indicates positions where there is a zero value. These are the positions (numerically) we want to remove. Unique will remove duplicate rows.

How many genes are upregulated at the log2 fold-change threshold of +2 or greater?

```
sum(mycounts$log2fc > +2)
```

```
## [1] 250
```

```
# Percentage
round((sum(mycounts$log2fc > +2)/nrow(mycounts))*100, 2)
```

```
## [1] 1.15
```

Q8. Using the up.ind vector above can you determine how many up regulated genes we have at the greater than 2 fc level?

```
sum(mycounts$log2fc > +2)
```

```
## [1] 250
```

Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level?

```
sum(mycounts$log2fc < -2)
```

```
## [1] 367
```

Q10. Do you trust these results? Why or why not?

No, because up until this point we have not done any tests for significance of the up & down regulated genes.

DESeq2 analysis

```
dds <- DESeqDataSetFromMatrix(countData=counts,
                              colData=mdat,
                              design=~dex)
```

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

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
# Run analysis pipeline
dds <- DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

Look at Deseq2 results

```
res <- results(dds)
head(res)
```

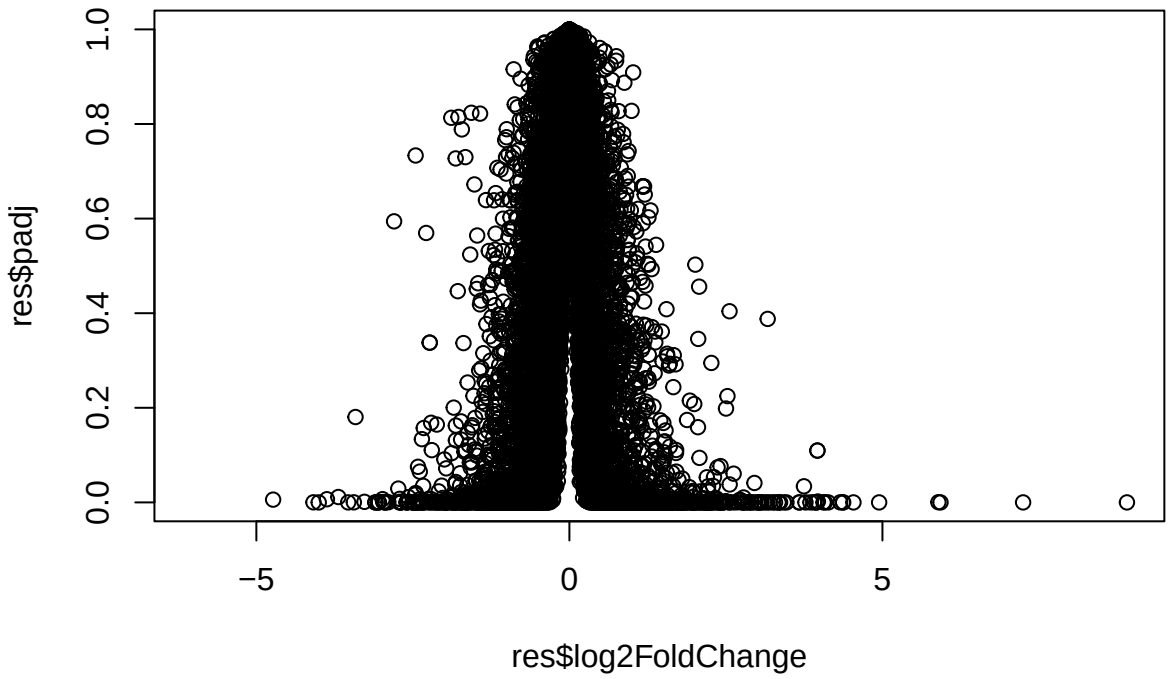
```
## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat    pvalue
##                  <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG00000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
## ENSG00000000005   0.000000             NA        NA        NA        NA
## ENSG00000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
## ENSG00000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
## ENSG00000000460  87.682625     -0.1471420  0.257007 -0.572521 0.5669691
## ENSG00000000938   0.319167     -1.7322890  3.493601 -0.495846 0.6200029
##                      padj
##                 <numeric>
## ENSG00000000003  0.163035
## ENSG00000000005        NA
## ENSG00000000419  0.176032
## ENSG00000000457  0.961694
## ENSG00000000460  0.815849
## ENSG00000000938        NA
```

```
# Look at summary of results
summary(res)
```

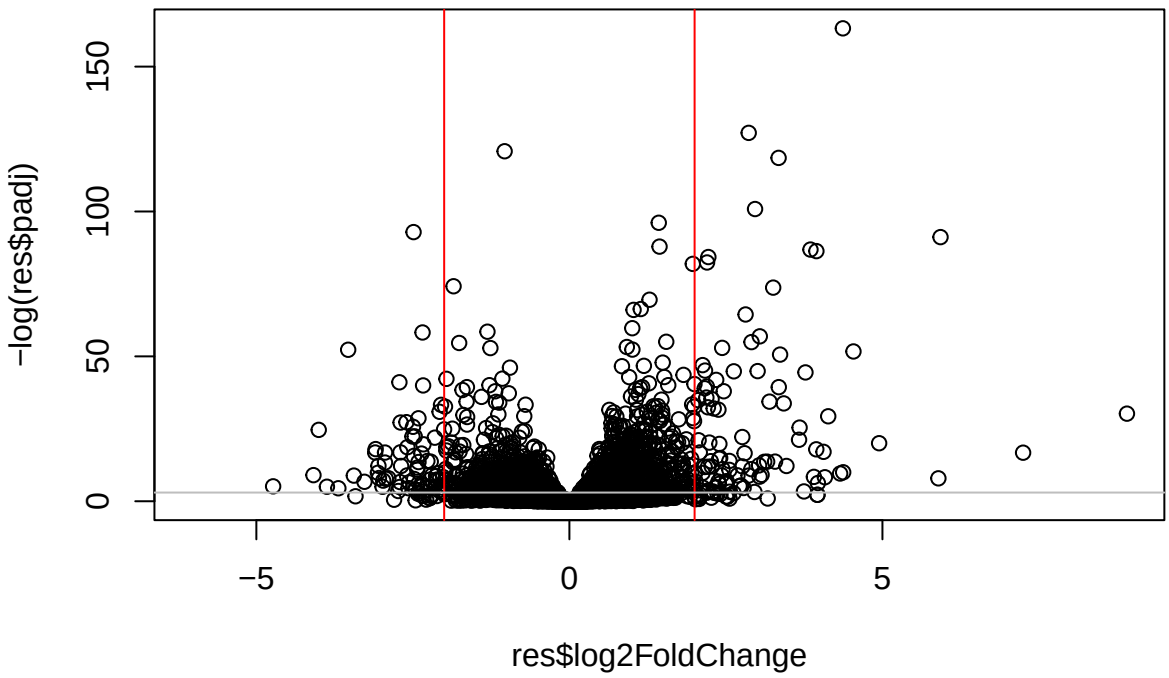
```
##
## out of 25258 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1563, 6.2%
## LFC < 0 (down)     : 1188, 4.7%
## outliers [1]       : 142, 0.56%
## low counts [2]     : 9971, 39%
## (mean count < 10)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Visualize results in volcano plot

```
plot(res$log2FoldChange, res$padj)
```




```
# Log transformed data
plot(res$log2FoldChange, -log(res$padj))
abline(v=c(-2, 2), col="red")
abline(h=-log(0.05), col="gray")
```



Adding annotation data

Want to add gene names to our dataset to make sense of what is going on. To accomplish we will use bioconductor packages.

```
# Load/install neccessary packages for annotation
library("AnnotationDbi")
#BiocManager::install("org.Hs.eg.db")
library("org.Hs.eg.db")
```

```
##
```

```
# Map symbol, the common gene name, to each ENSEMBL gene id
res$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="SYMBOL",          # The new format we want to add
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
head(res$symbol)
```

```
## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
##        "TSPAN6"          "TNMD"          "DPM1"         "SCYL3"      "C1orf112"
## ENSG00000000938
##           "FGR"
```

Write results to file

```
# Write results to csv
write.csv(res, file="./all my results.csv")
```

Pathway analysis

Add kegg pathway analysis details to our results table

```
# Check annotation types
columns(org.Hs.eg.db)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
## [16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
## [21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [26] "UNIPROT"
```

```
# Before we can use KEGG, we need to get gene identifiers in the correct format for KEGG, which is ENTR

res$entrez <- mapIds(org.Hs.eg.db, # Annotation package
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="ENTREZID",          # The new format we want to add
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
res$genenames <-mapIds(org.Hs.eg.db, # Annotation package
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="GENENAME",          # The new format we want to add
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# Find enriched pathways using gauge

# Create vector of fold changes w/ names of the values as ENTREZ IDs, which is required for gauge input

foldchanges <- res$log2FoldChange
names(foldchanges) <- res$entrez
head(foldchanges)
```

```
##        7105       64102        8813       57147       55732        2268
## -0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

Run gauge pathway analysis

```
library(pathview)
```

```
## ##############################################################################
## Pathview is an open source software package distributed under GNU General
## Public License version 3 (GPLv3). Details of GPLv3 is available at
## http://www.gnu.org/licenses/gpl-3.0.html. Particullary, users are required to
## formally cite the original Pathview paper (not just mention it) in publications
## or products. For details, do citation("pathview") within R.
##
## The pathview downloads and uses KEGG data. Non-academic uses may require a KEGG
## license agreement (details at http://www.kegg.jp/kegg/legal.html).
## ##############################################################################
```

```
library(gage)
```

```
##
```

```
library(gageData)
data(kegg.sets.hs)

keggres <- gage(foldchanges, gsets=kegg.sets.hs)

attributes(keggres)
```

```
## $names
## [1] "greater" "less"    "stats"
```

Pathview

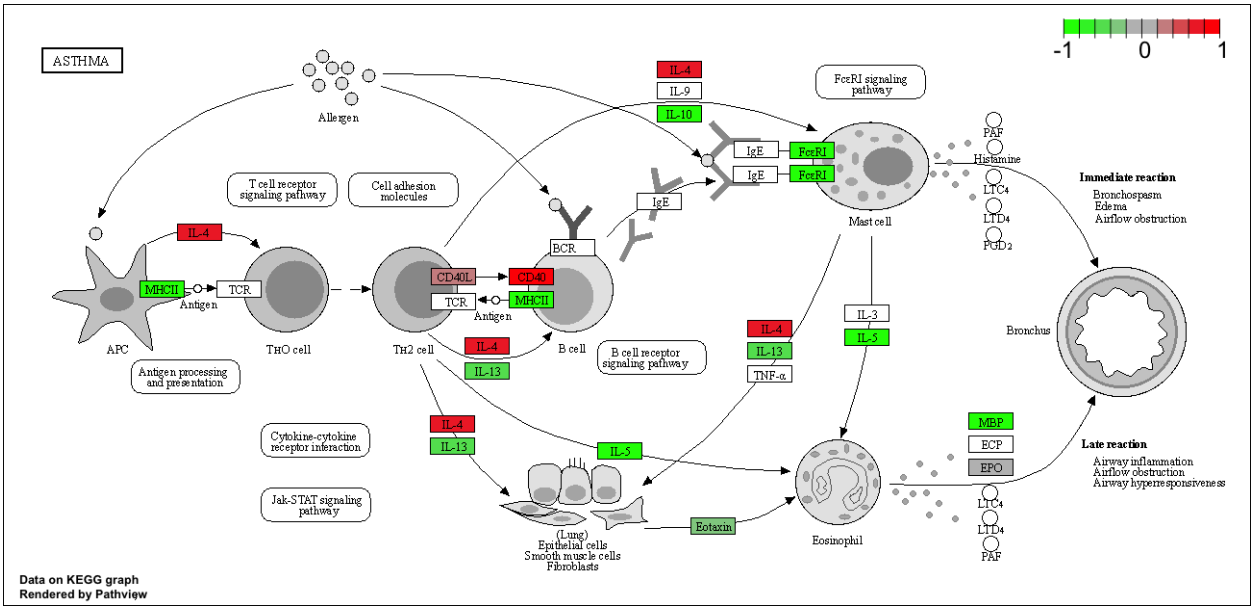
The pathview() function will add our genes to a KEGG pathway as colored entries.

```
pathview(gene.data=foldchanges, pathway.id="hsa05310")
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
## Info: Working in directory /Volumes/GoogleDrive/My Drive/GitHub/bggn_213/11_17_21_DESeq2
```

```
## Info: Writing image file hsa05310.pathview.png
```


ASTHMA
-1
0
1
Allergen
IL-4
IL-9
IL-10
FcεRI signaling pathway
IgE
FcεRI
IgE
FcεRI
Mast cell
PAF
Histamine
LTC4
LTD4
PGD2
Immediate reaction
Bronchospasm
Edema
Airflow obstruction
T cell receptor signaling pathway
Cell adhesion molecules
IL-4
MHCII
Antigen
TCR
APC
TH0 cell
Antigen processing and presentation
CD40L
CD40
TCR
MHCII
Antigen
TH2 cell
BCR
IgE
B cell
B cell receptor signaling pathway
IL-4
IL-13
IL-4
IL-13
TNF-α
IL-3
IL-5
Bronchus
Cytokine-cytokine receptor interaction
Jak-STAT signaling pathway
IL-4
IL-13
IL-5
(Lung) Epithelial cells
Smooth muscle cells
Fibroblasts
Eotaxin
Eosinophil
MBP
ECP
EPO
LTC4
LTD4
PAF
Late reaction
Airway inflammation
Airflow obstruction
Airway hyperresponsiveness
Data on KEGG graph
Rendered by Pathview