# Class15_RNASeq

Caitriona Brennan

11-17-21

```r
library(BiocManager)
```

```r
library(DESeq2)
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## 
## Attaching package: 'BiocGenerics'
```

```
## 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
```

Today we examine a published RNA-seq experiment where airway smooth muscle cells were treated with dexam

We need tow things : -1:count data 2: col data

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

```
head(counts)
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003        723        486        904        445       1170
## ENSG00000000005          0          0          0          0          0
## ENSG00000000419        467        523        616        371        582
## ENSG00000000457        347        258        364        237        318
## ENSG00000000460         96         81         73         66        118
## ENSG00000000938          0          0          1          0          2
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003       1097        806        604
## ENSG00000000005          0          0          0
## ENSG00000000419        781        417        509
## ENSG00000000457        447        330        324
## ENSG00000000460         94        102         74
## ENSG00000000938          0          0          0
```

```
head(metadata)
```

```
##           id     dex celltype     geo_id
## 1 SRR1039508 control   N61311 GSM1275862
## 2 SRR1039509 treated   N61311 GSM1275863
## 3 SRR1039512 control  N052611 GSM1275866
## 4 SRR1039513 treated  N052611 GSM1275867
## 5 SRR1039516 control  N080611 GSM1275870
## 6 SRR1039517 treated  N080611 GSM1275871
```

Side note: Let's check the corespondance of the metadata nad count data setup. i.e. check if the first column of metadata is the same as the counts column headers

```
metadata[1]
```

```
##           id
## 1 SRR1039508
## 2 SRR1039509
## 3 SRR1039512
## 4 SRR1039513
## 5 SRR1039516
## 6 SRR1039517
## 7 SRR1039520
## 8 SRR1039521
```

```
colnames(counts)==metadata[1]
```

```
##        id
## [1,] TRUE
## [2,] TRUE
## [3,] TRUE
## [4,] TRUE
## [5,] TRUE
## [6,] TRUE
## [7,] TRUE
## [8,] TRUE
```

wrap the above code in all to tell us if all the outputs are true

```
all(colnames(counts)==metadata[1])
```

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

##compare control to treated First we need to access all the control columns in our counts data. This is in the column "dex" using control.inds in [ ] gets that row of Trues and "1" gets the first col of this

```
control.inds <- metadata$dex=="control"
metadata[control.inds,1]
```

```
## [1] "SRR1039508" "SRR1039512" "SRR1039516" "SRR1039520"
```

Use these ids to access just the control columns of our 'counts' data

```
control.ids <- metadata[control.inds,]$id
control.ids
```

```
## [1] "SRR1039508" "SRR1039512" "SRR1039516" "SRR1039520"
```

```
control.mean <- rowMeans(counts[,control.ids])
head(control.mean)
```

```
## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
##          900.75            0.00          520.50          339.75           97.25
## ENSG00000000938
##            0.75
```

##Do the same for the drug tested

```
treated.id <- metadata[metadata$dex == "treated",]$id
treated.mean <- rowMeans(counts[,treated.id])
```

We will combine our meancount data for bookkeeping purposes.

```
meancounts <- data.frame(control.mean, treated.mean)
```

Use the ' beside the 1 in the keyboard to also use code in this r,markdown
it will only show up in the r script: There are 38694 rows/genes in this dataset

##how many genes in this dataset

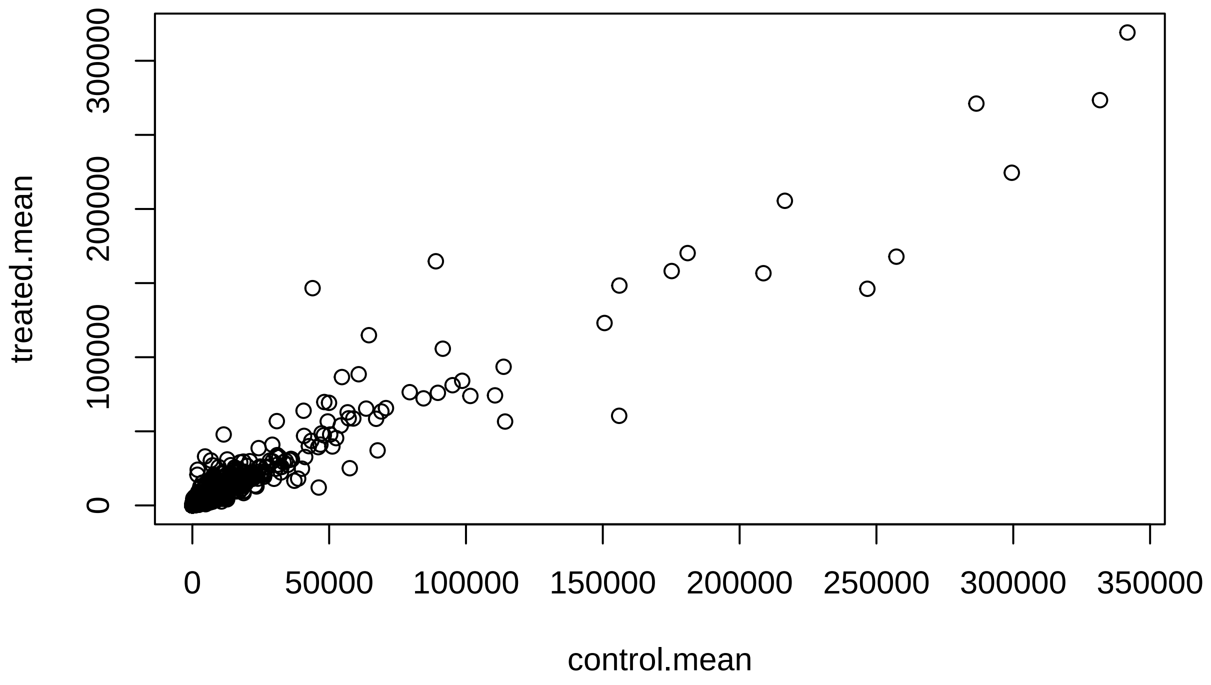
```
nrow(counts)
```

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

##Compare the control and treated

A quick plot of our progress so far

```
plot(meancounts)
```

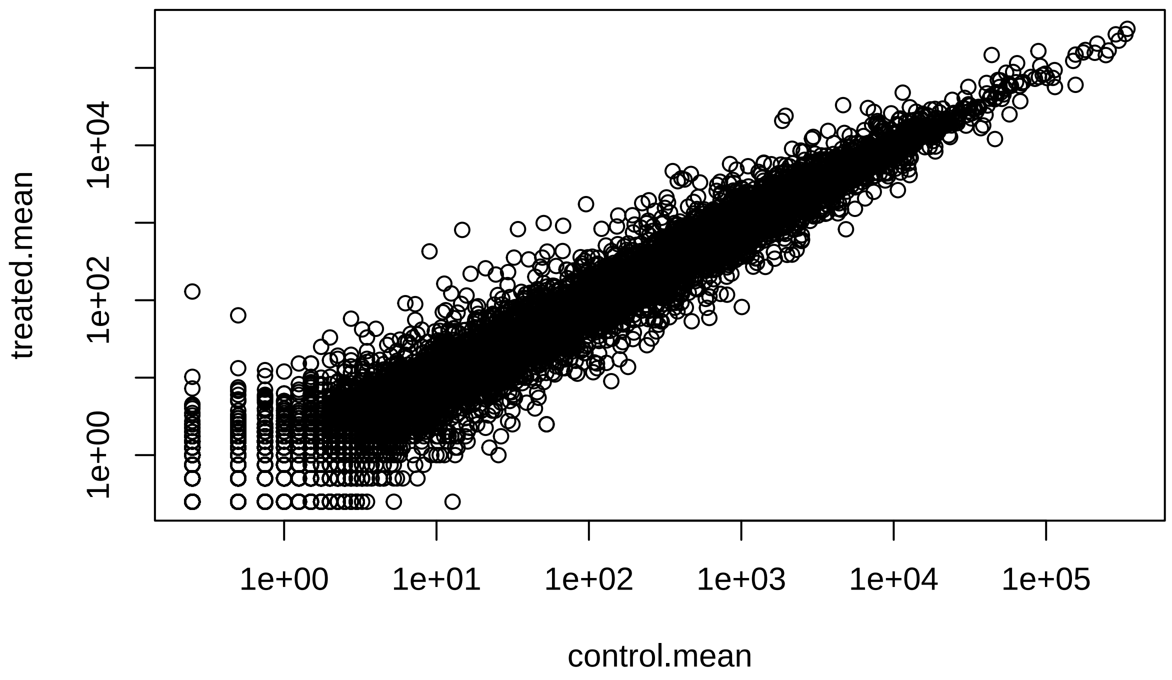


This plot (above) needs to be altered to make it clearer. A log would be v useful

```
plot(meancounts, log = "xy")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15032 x values <= 0 omitted
## from logarithmic plot
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15281 y values <= 0 omitted
## from logarithmic plot
```



We often use log transformations as they make life much easier…

```
log2(10/20)
```

```
## [1] -1
```

```
log2(80/20)
```

```
## [1] 2
```

```
meancounts$log2fc <- log2(meancounts[,"treated.mean"]/meancounts[,"control.mean"])
head(meancounts)
```

```
##                 control.mean treated.mean      log2fc
## ENSG00000000003       900.75       658.00 -0.45303916
## ENSG00000000005         0.00         0.00         NaN
## ENSG00000000419       520.50       546.00  0.06900279
## ENSG00000000457       339.75       316.50 -0.10226805
## ENSG00000000460        97.25        78.75 -0.30441833
## ENSG00000000938         0.75         0.00        -Inf
```

Let's look for zeros in the meancounts

```r
head(meancounts[,1:2])
```

```
##                 control.mean treated.mean
## ENSG00000000003       900.75       658.00
## ENSG00000000005         0.00         0.00
## ENSG00000000419       520.50       546.00
## ENSG00000000457       339.75       316.50
## ENSG00000000460        97.25        78.75
## ENSG00000000938         0.75         0.00
```

```r
head(meancounts[,1:2] == 0)
```

```
##                 control.mean treated.mean
## ENSG00000000003        FALSE        FALSE
## ENSG00000000005         TRUE         TRUE
## ENSG00000000419        FALSE        FALSE
## ENSG00000000457        FALSE        FALSE
## ENSG00000000460        FALSE        FALSE
## ENSG00000000938        FALSE         TRUE
```

The which() function tells us the indices of TRUE entries in a logical vector

However, it is not useful in default mode on our type of multi column input…

```r
inds <- which(meancounts[,1:2] == 0, arr.ind = TRUE)
head(inds)
```

```
##                 row col
## ENSG00000000005   2   1
## ENSG00000004848  65   1
## ENSG00000004948  70   1
## ENSG00000005001  73   1
## ENSG00000006059 121   1
## ENSG00000006071 123   1
```

I only care about the rows here (if there is a zero in any column I will exclude this row eventually).

```r
to.rm <- unique(sort(inds[,"row"]))
```

```r
head(meancounts[to.rm,])
```

```
##                 control.mean treated.mean log2fc
## ENSG00000000005         0.00         0.00    NaN
## ENSG00000000938         0.75         0.00   -Inf
## ENSG00000004848         0.00         0.25    Inf
## ENSG00000004948         0.00         0.00    NaN
## ENSG00000005001         0.00         0.00    NaN
## ENSG00000005102         1.00         0.00   -Inf
```

```
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
```

We now have 21817 genes remaining

```
nrow(mycounts)
```

```
## [1] 21817
```

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

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

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

What is the percentage of this?

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

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

The calculate down regulated

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

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

```
round((sum(mycounts$log2fc < -2) / nrow(mycounts)) *100,2)
```

```
## [1] 1.68
```

#DESeq2 analysis We first need to set up the DESeq input object

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

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

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

```
dds
```

```
## class: DESeqDataSet
## dim: 38694 8
## metadata(1): version
## assays(1): counts
## rownames(38694): ENSG00000000003 ENSG00000000005 ... ENSG00000283120
##   ENSG00000283123
## rowData names(0):
## colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
## colData names(4): id dex celltype geo_id
```

Run the DESeq analysis pipeline

```
dds <- DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

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

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

```
## final dispersion estimates
```

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

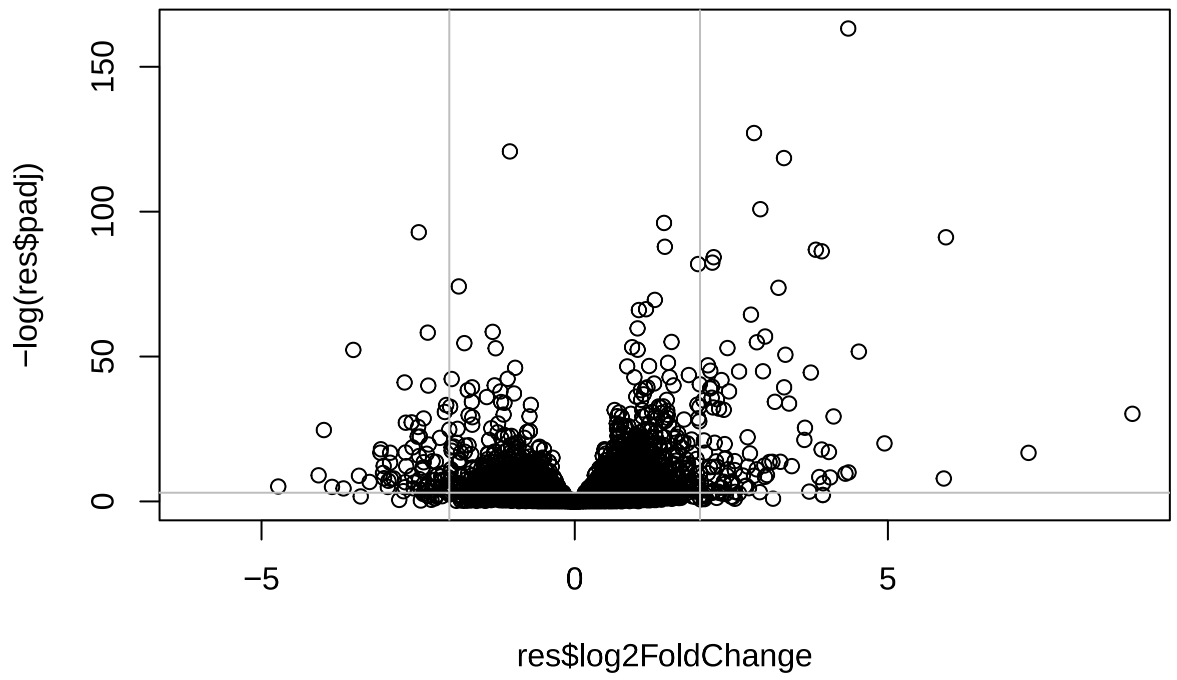
```
res <-  results(dds)
head(res)
```

```
## 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
```

## A Volcano plot

This is a very common data viz of this type of data that does not really look like a volcano abline creates a line

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



##Adding annotation data We want to add meaningful gene names to our dataset so we can make some sense of what is going on here..

For this we will use two bioconductor packages, one does the work and is called **AnnotationDbi**. The other contains the data we are going to map between and is called **org.Hs.eg.db**

```
library("AnnotationDbi")
```

```
## Warning: package 'AnnotationDbi' was built under R version 4.1.2
```

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

```
##
```

```
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"
```

Here we map to "SYMBOL" the common gene name that the world understands and wants

```
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"
```

```
head(res)
```

```
## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 7 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      symbol
##                 <numeric> <character>
## ENSG00000000003  0.163035      TSPAN6
## ENSG00000000005        NA        TNMD
## ENSG00000000419  0.176032        DPM1
## ENSG00000000457  0.961694       SCYL3
## ENSG00000000460  0.815849    C1orf112
## ENSG00000000938        NA         FGR
```

Lets save our results to date.

```
write.csv(res, file = "class15_allmyresults.csv")
```

##Pathway analysis

Let's try bring some insight back into this work. For this we will start with KEGG.

```
library(pathview)
library(gage)
library(gageData)

data(kegg.sets.hs)

# Examine the first 2 pathways in this kegg set for humans
head(kegg.sets.hs, 2)
```

```
## $`hsa00232 Caffeine metabolism`
## [1] "10"   "1544" "1548" "1549" "1553" "7498" "9"
##
## $`hsa00983 Drug metabolism - other enzymes`
##  [1] "10"     "1066"   "10720"  "10941"  "151531" "1548"   "1549"   "1551"
##  [9] "1553"   "1576"   "1577"   "1806"   "1807"   "1890"   "221223" "2990"
## [17] "3251"   "3614"   "3615"   "3704"   "51733"  "54490"  "54575"  "54576"
## [25] "54577"  "54578"  "54579"  "54600"  "54657"  "54658"  "54659"  "54963"
## [33] "574537" "64816"  "7083"   "7084"   "7172"   "7363"   "7364"   "7365"
## [41] "7366"   "7367"   "7371"   "7372"   "7378"   "7498"   "79799"  "83549"
## [49] "8824"   "8833"   "9"      "978"
```

Before we can use KEGG we need to get our gene identifiers in the correct format for KEGG, which is ENTREZ format in this case

```
head(rownames(res))
```

```
## [1] "ENSG00000000003" "ENSG00000000005" "ENSG00000000419" "ENSG00000000457"
## [5] "ENSG00000000460" "ENSG00000000938"
```

need to translate the rownames above into entrez format

```
head(columns(org.Hs.eg.db))
```

```
## [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
## [6] "ENTREZID"
```

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

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

The main gage() function requires a named vector of fold changes, where the names of the values are the Entrez gene IDs.

Note that we used the mapIDs() function above to obtain Entrez gene IDs (stored in res*entrez)andwehavethefoldchangeresult*

```
foldchanges = res$log2FoldChange
head(foldchanges)
```

```
## [1] -0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

assign names to this vector that are the gene IDs that KEGG wants

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

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

Now we can pass this to the gage() function

```
#get the results
keggres = gage(foldchanges, gsets=kegg.sets.hs)
```

We can look at the attributes() of an object

```
attributes(keggres)
```

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

```
head(keggres$less)
```

```
##                                                          p.geomean stat.mean
## hsa05332 Graft-versus-host disease                    0.0004250461 -3.473346
## hsa04940 Type I diabetes mellitus                     0.0017820293 -3.002352
## hsa05310 Asthma                                       0.0020045888 -3.009050
## hsa04672 Intestinal immune network for IgA production 0.0060434515 -2.560547
## hsa05330 Allograft rejection                          0.0073678825 -2.501419
## hsa04340 Hedgehog signaling pathway                   0.0133239547 -2.248547
##                                                              p.val      q.val
## hsa05332 Graft-versus-host disease                    0.0004250461 0.09053483
## hsa04940 Type I diabetes mellitus                     0.0017820293 0.14232581
## hsa05310 Asthma                                       0.0020045888 0.14232581
## hsa04672 Intestinal immune network for IgA production 0.0060434515 0.31387180
## hsa05330 Allograft rejection                          0.0073678825 0.31387180
## hsa04340 Hedgehog signaling pathway                   0.0133239547 0.47300039
##                                                       set.size         exp1
## hsa05332 Graft-versus-host disease                          40 0.0004250461
## hsa04940 Type I diabetes mellitus                           42 0.0017820293
## hsa05310 Asthma                                             29 0.0020045888
## hsa04672 Intestinal immune network for IgA production       47 0.0060434515
## hsa05330 Allograft rejection                                36 0.0073678825
## hsa04340 Hedgehog signaling pathway                         56 0.0133239547
```

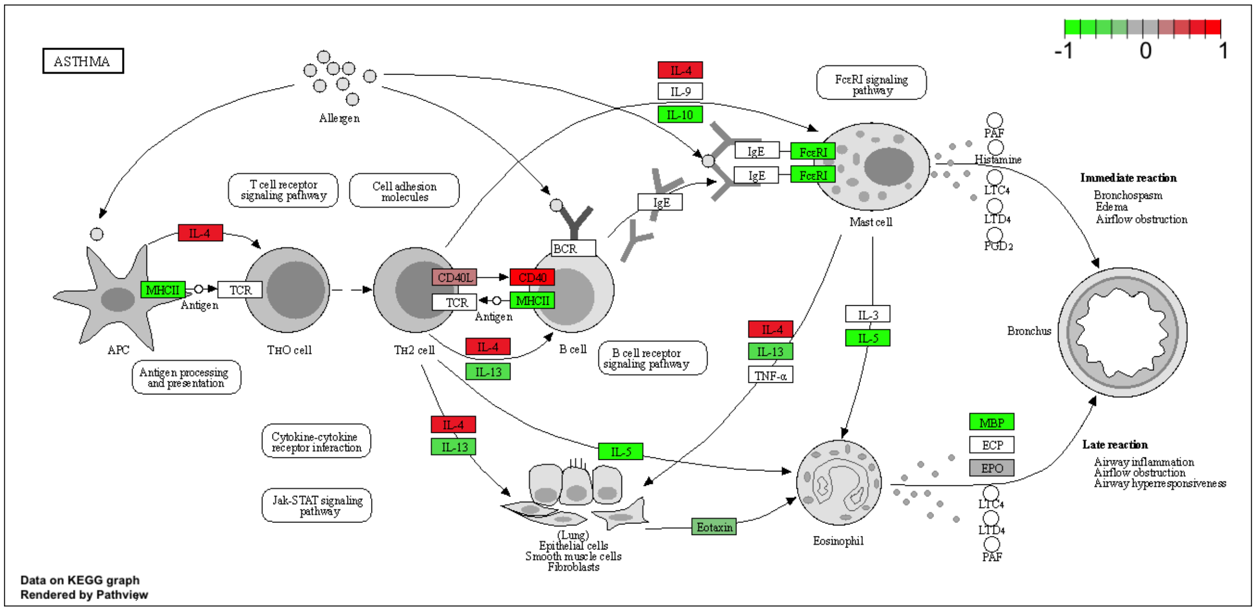
The pathview() function will add our genes to a KEGG pathway as colored entries:

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

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

```
## Info: Working in directory /Users/caitrionabrennan/Documents/Bioinformatics _213/R Studio Class/bggn2
```

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



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