Class 12

Courtney Anderson (PID:A69038035)

Table of contents

Background

Today we will analyze some RNAseq data from Himes et. al on the effects of a common steroid (dexmethansone also called "dex") on airway smooth muscle cells (ASMs).

For this analysis we need two main inputs:

-`countData`: a table of **counts** per gene (in rows) across expriments (in columns) -`colData`: **metadata** about the design of experiments. The rows here must mat h the columns in 'countData'

```
library(BiocManager)
library(DESeq2)
```

```
Loading required package: S4Vectors
```

```
Loading required package: stats4
```

```
Loading required package: BiocGenerics
```

```
Loading required package: generics


Attaching package: 'generics'

The following objects are masked from 'package:base':

    as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
    setequal, union


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, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, is.unsorted, lapply, Map, mapply, match, mget,
    order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
    rbind, Reduce, rownames, sapply, saveRDS, table, tapply, unique,
    unsplit, which.max, which.min


Attaching package: 'S4Vectors'

The following object is masked from 'package:utils':

    findMatches

The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges

Loading required package: GenomicRanges
```

```
Loading required package: Seqinfo

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
```

Data Import

```
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
```

Q1a. How many "genes" are in this dataset?

```
nrow(counts)
```

```
[1] 38694
```

Q1b. How many experiments (i.e columns in `counts` or rows in `metadata`) are there

```
ncol(counts)
```

```
[1] 8
```

Q2. How many control experiments are there in the database?

```
sum(metadata$dex== "control")
```

```
[1] 4
```

Toy analysis example

Q3. How would you make the above code in either approach more robust? Is there a function that could help here? I would use the rowMeans() funcition instead of rowSums().

1. Extract the "control" columns from `counts`

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

2. Calculate the mean value for each gene in these "control" columns

```
control.mean <- rowMeans(control.counts)
```

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)

3-4. Co the same for the treated columns

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

5. Compare these mean values for each gene. For ease of book-keeping we can store these together in one data frame called `meancounts`

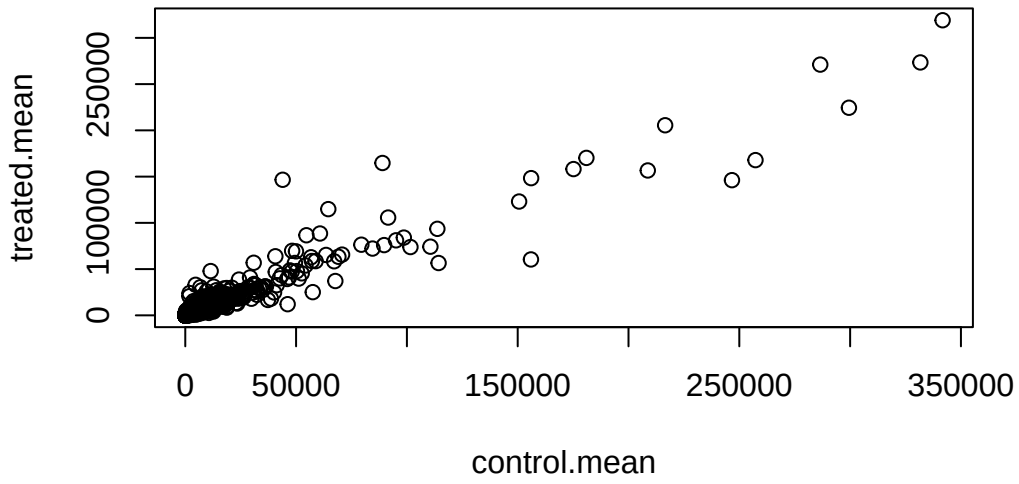
```
meancounts <- data.frame(control.mean, treated.mean)
head(meancounts)
```

```
                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
```

Q5 (a). Create a scatter plot showing the mean of the treated samples against the mean of the control samples. Your plot should look something like the following.

Plot these against eachother

```
plot(meancounts)
```



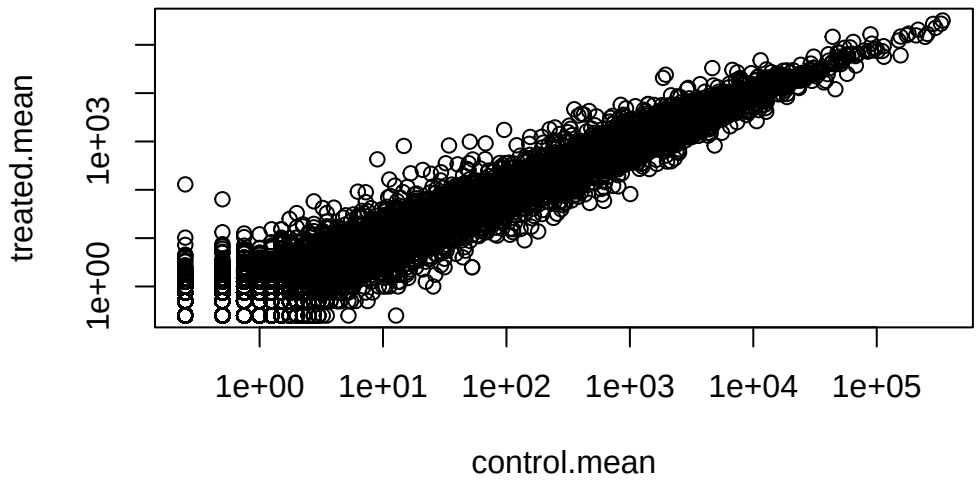
Q5 (b).You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?] I would use geom_point() function.

This is screaming at me to log it

```
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
```



Q6. Try plotting both axes on a log scale. What is the argument to plot() that allows you to do this? Log2

We use log2 "fold change" as a way to compare

```
#treated/control
log2(10/10)
```

```
[1] 0
```

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

```
[1] 1
```

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

```
[1] -1
```

```
log2(40/10)
```

```
[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
```

```
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? When arr.ind=TRUE, it will return the row and columns that are TRUE. We would need to take the first column od the output to call it a unique function so we dont count any row twice.

A common rule of thumb threshold for calling somthing"up" regulated is a log2-fold change of +2 or greater. For down regulated -2 or less.

```
up.ind <- mycounts$log2fc > 2
down.ind <- mycounts$log2fc < (-2)
```

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(up.ind)
```

```
[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(down.ind)
```

```
[1] 367
```

Q10. Do you trust these results? Why or why not? No I dont because we haven't investigated any significance. Time to get p-values involved.

DESeq analysis

Let's do this properly DESeq and put some stats behing these numbers.

```
library(DESeq2)
citation("DESeq2")
```

```
To cite package 'DESeq2' in publications use:

  Love, M.I., Huber, W., Anders, S. Moderated estimation of fold change
  and dispersion for RNA-seq data with DESeq2 Genome Biology 15(12):550
  (2014)
```

```
A BibTeX entry for LaTeX users is

  @Article{,
    title = {Moderated estimation of fold change and dispersion for RNA-
seq data with DESeq2},
    author = {Michael I. Love and Wolfgang Huber and Simon Anders},
    year = {2014},
    journal = {Genome Biology},
    doi = {10.1186/s13059-014-0550-8},
    volume = {15},
    issue = {12},
    pages = {550},
  }
```

DESEq wants 3 things for analysis, countDATA, colDATA and design

```
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
```

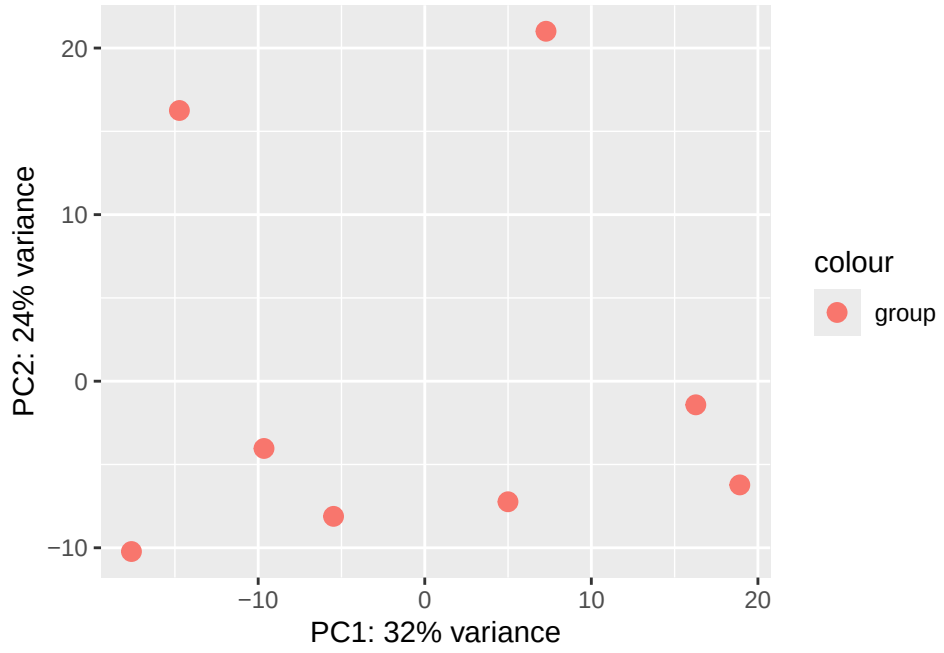
```
dds
```

```
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
```

The main fucntion in the DESeq package to run analysis is called `DESeq()`

```
vsd <- vst(dds, blind = FALSE)
plotPCA(vsd, intgroup = c("dex"))
```

```
using ntop=500 top features by variance
```




```
pcaData <- plotPCA(vsd, intgroup=c("dex"), returnData=TRUE)
```

```
using ntop=500 top features by variance
```

```
head(pcaData)
```

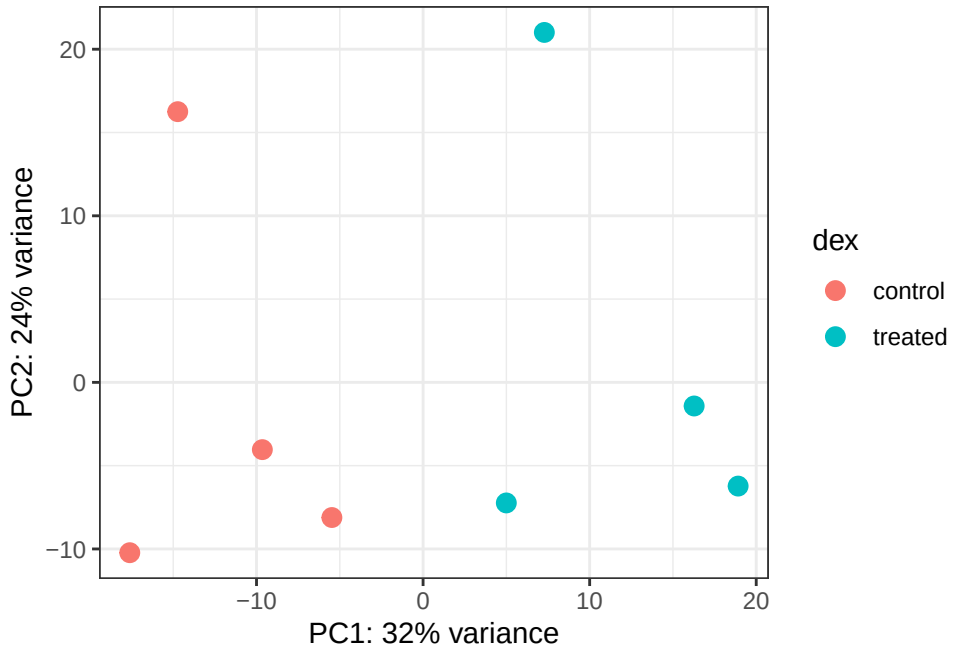
```
                PC1        PC2   group       name         id     dex celltype
SRR1039508 -17.607922 -10.225252 control SRR1039508 SRR1039508 control   N61311
SRR1039509   4.996738  -7.238117 treated SRR1039509 SRR1039509 treated   N61311
SRR1039512  -5.474456  -8.113993 control SRR1039512 SRR1039512 control  N052611
SRR1039513  18.912974  -6.226041 treated SRR1039513 SRR1039513 treated  N052611
SRR1039516 -14.729173  16.252000 control SRR1039516 SRR1039516 control  N080611
SRR1039517   7.279863  21.008034 treated SRR1039517 SRR1039517 treated  N080611
               geo_id sizeFactor
SRR1039508 GSM1275862  1.0193796
```

```
SRR1039509 GSM1275863  0.9005653
SRR1039512 GSM1275866  1.1784239
SRR1039513 GSM1275867  0.6709854
SRR1039516 GSM1275870  1.1731984
SRR1039517 GSM1275871  1.3929361
```

```
# Calculate percent variance per PC for the plot axis labels
percentVar <- round(100 * attr(pcaData, "percentVar"))
```

```
library(ggplot2)

ggplot(pcaData) +
  aes(x = PC1, y = PC2, color = dex) +
  geom_point(size =3) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  coord_fixed() +
  theme_bw()
```




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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

```
results(dds)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003  747.1942      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005    0.0000             NA        NA        NA        NA
ENSG00000000419  520.1342       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457  322.6648       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460   87.6826      -0.147143  0.256995 -0.572550 0.5669497
...                    ...            ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916       -0.66825   1.69441 -0.394385  0.693297
ENSG00000283123  0.000000             NA        NA        NA        NA
                     padj
                <numeric>
ENSG00000000003  0.163017
ENSG00000000005        NA
ENSG00000000419  0.175937
ENSG00000000457  0.961682
ENSG00000000460  0.815805
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

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

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 38694 rows and 6 columns
                 baseMean log2FoldChange     lfcSE      stat    pvalue
                <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003  747.1942      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005    0.0000             NA        NA        NA        NA
ENSG00000000419  520.1342       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457  322.6648       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460   87.6826      -0.147143  0.256995 -0.572550 0.5669497
...                    ...            ...       ...       ...       ...
ENSG00000283115  0.000000             NA        NA        NA        NA
ENSG00000283116  0.000000             NA        NA        NA        NA
ENSG00000283119  0.000000             NA        NA        NA        NA
ENSG00000283120  0.974916       -0.66825   1.69441 -0.394385  0.693297
ENSG00000283123  0.000000             NA        NA        NA        NA
                     padj
                <numeric>
ENSG00000000003  0.163017
ENSG00000000005        NA
ENSG00000000419  0.175937
ENSG00000000457  0.961682
ENSG00000000460  0.815805
...                   ...
ENSG00000283115        NA
ENSG00000283116        NA
ENSG00000283119        NA
ENSG00000283120        NA
ENSG00000283123        NA
```

```
summary(res)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 1564, 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
```

```
res05 <- results(dds, alpha=0.05)
summary(res05)
```

```
out of 25258 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 1237, 4.9%
LFC < 0 (down)     : 933, 3.7%
outliers [1]       : 142, 0.56%
low counts [2]     : 9033, 36%
(mean count < 6)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

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

```
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)
```

```
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      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
                     padj      symbol
                <numeric> <character>
ENSG00000000003  0.163017      TSPAN6
ENSG00000000005        NA        TNMD
ENSG00000000419  0.175937        DPM1
ENSG00000000457  0.961682       SCYL3
ENSG00000000460  0.815805       FIRRM
ENSG00000000938        NA         FGR
```

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

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

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

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

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

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

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

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

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
                     padj      symbol      entrez     uniprot
                <numeric> <character> <character> <character>
ENSG00000000003  0.163017      TSPAN6        7105  A0A087WYV6
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.175937        DPM1        8813      H0Y368
ENSG00000000457  0.961682       SCYL3       57147      X6RHX1
ENSG00000000460  0.815805       FIRRM       55732      A6NFP1
ENSG00000000938        NA         FGR        2268      B7Z6W7
                          genename
                       <character>
ENSG00000000003          tetraspanin 6
```

```
ENSG00000000005            tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 FIGNL1 interacting r..
ENSG00000000938 FGR proto-oncogene, ..
```

```
ord <- order( res$padj )
#View(res[ord,])
head(res[ord,])
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                 baseMean log2FoldChange     lfcSE      stat      pvalue
                <numeric>      <numeric> <numeric> <numeric>   <numeric>
ENSG00000152583   954.771        4.36836 0.2371306   18.4217 8.79214e-76
ENSG00000179094   743.253        2.86389 0.1755659   16.3123 8.06568e-60
ENSG00000116584  2277.913       -1.03470 0.0650826  -15.8983 6.51317e-57
ENSG00000189221  2383.754        3.34154 0.2124091   15.7316 9.17960e-56
ENSG00000120129  3440.704        2.96521 0.2036978   14.5569 5.27883e-48
ENSG00000148175 13493.920        1.42717 0.1003811   14.2175 7.13625e-46
                       padj      symbol      entrez     uniprot
                  <numeric> <character> <character> <character>
ENSG00000152583 1.33157e-71     SPARCL1        8404      B4E2Z0
ENSG00000179094 6.10774e-56        PER1        5187      A2I2P6
ENSG00000116584 3.28806e-53     ARHGEF2        9181  A0A8Q3SIN5
ENSG00000189221 3.47563e-52        MAOA        4128      B4DF46
ENSG00000120129 1.59896e-44       DUSP1        1843      B4DRR4
ENSG00000148175 1.80131e-42        STOM        2040      F8VSL7
                            genename
                         <character>
ENSG00000152583           SPARC like 1
ENSG00000179094 period circadian reg..
ENSG00000116584 Rho/Rac guanine nucl..
ENSG00000189221    monoamine oxidase A
ENSG00000120129 dual specificity pho..
ENSG00000148175               stomatin
```

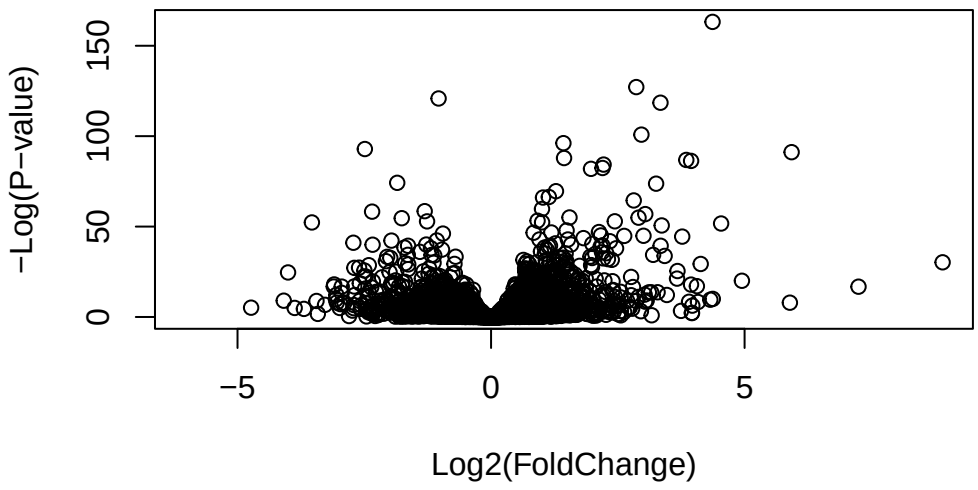
```
write.csv(res[ord,], "deseq_results.csv")
```

```
# Convert DESeq2 results object to a regular data frame
res.df <- as.data.frame(res)
```

Volcano Plot

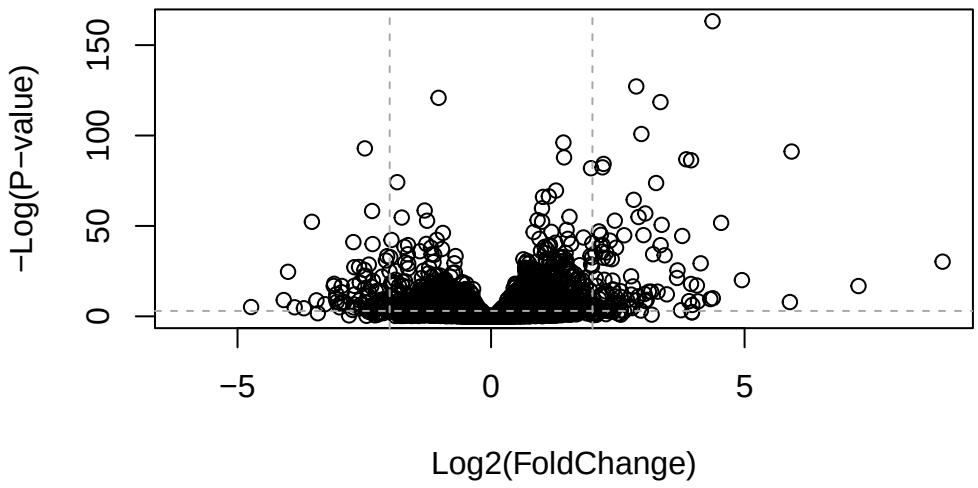
This is a plot of log2FC vs adjusted p-value

```
plot( res$log2FoldChange,  -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
```




```
plot( res$log2FoldChange,  -log(res$padj),
 ylab="-Log(P-value)", xlab="Log2(FoldChange)")

# Add some cut-off lines
abline(v=c(-2,2), col="darkgray", lty=2)
abline(h=-log(0.05), col="darkgray", lty=2)
```

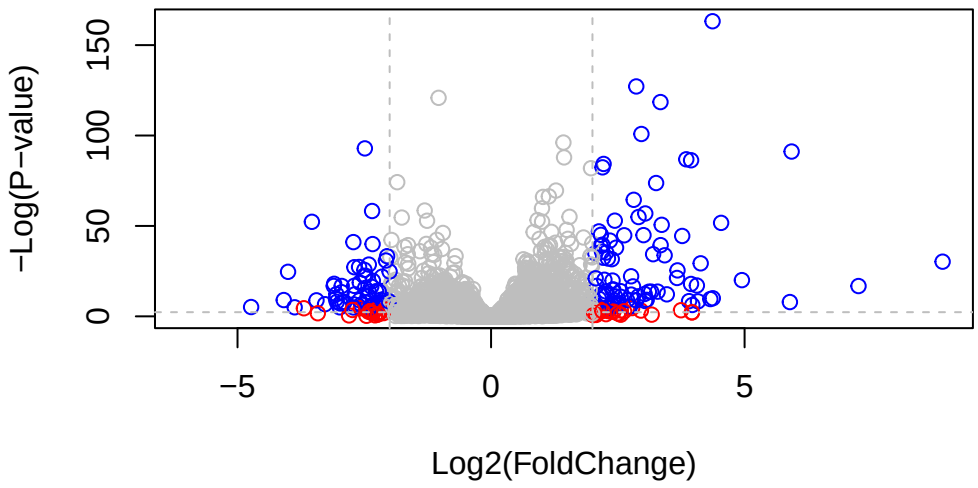



```
# Setup our custom point color vector
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "red"

inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"

# Volcano plot with custom colors
plot( res$log2FoldChange,  -log(res$padj),
 col=mycols, ylab="-Log(P-value)", xlab="Log2(FoldChange)" )

# Cut-off lines
abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)
```



A nicer ggplot volcano plot

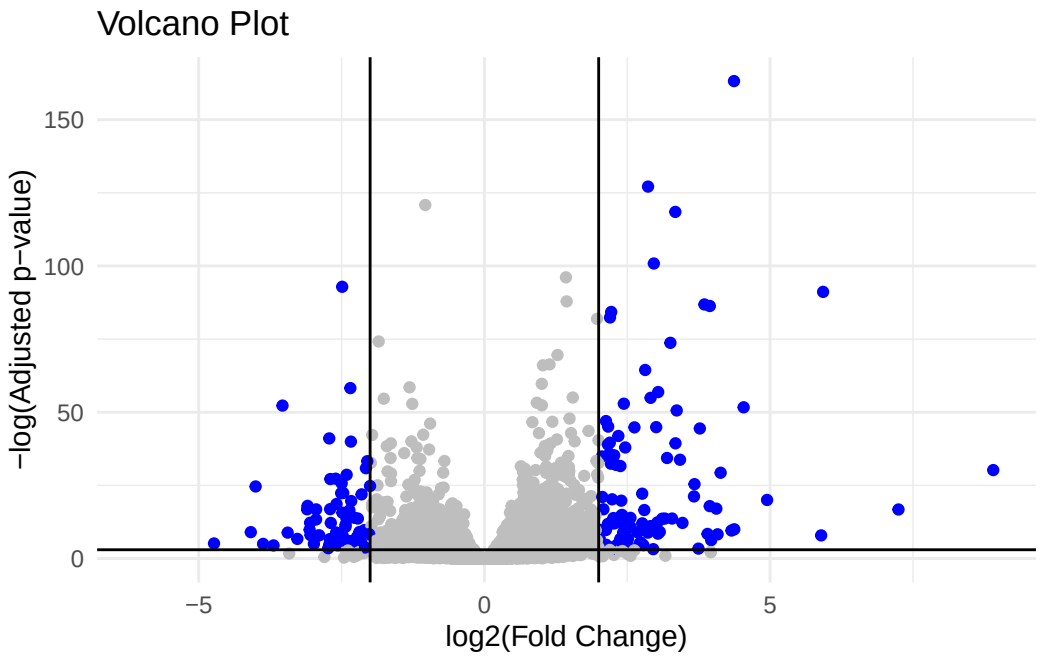
```
library(ggplot2)

mycols <- rep("gray", nrow(res))
mycols[abs(res$log2FoldChange) >= 2 & res$padj < 0.05] <- "blue"  # significant and large cha
mycols[res$padj >= 0.05] <- "gray"  # non-significant (optional, ensures gray)

# Volcano plot
ggplot(res, aes(x = log2FoldChange, y = -log(padj))) +
  geom_point(color = mycols) +
  geom_vline(xintercept = c(-2, 2), color = "black") +
  geom_hline(yintercept = -log(0.05), color = "black") +
  labs(
    x = "log2(Fold Change)",
    y = "-log(Adjusted p-value)",
    title = "Volcano Plot"
  ) +
  theme_minimal()
```

```
Warning: Removed 23549 rows containing missing values or values outside the scale range
```

(`geom_point()`).




```
colnames(res)
```

```
[1] "baseMean"       "log2FoldChange" "lfcSE"          "stat"
[5] "pvalue"         "padj"           "symbol"         "entrez"
[9] "uniprot"        "genename"
```

```
head(res)
```

```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 10 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195      -0.350703  0.168242 -2.084514 0.0371134
ENSG00000000005   0.000000             NA        NA        NA        NA
ENSG00000000419 520.134160       0.206107  0.101042  2.039828 0.0413675
ENSG00000000457 322.664844       0.024527  0.145134  0.168996 0.8658000
ENSG00000000460  87.682625      -0.147143  0.256995 -0.572550 0.5669497
ENSG00000000938   0.319167      -1.732289  3.493601 -0.495846 0.6200029
```

```
                    padj      symbol      entrez     uniprot
                <numeric> <character> <character> <character>
ENSG00000000003  0.163017      TSPAN6        7105  A0A087WYV6
ENSG00000000005        NA        TNMD       64102      Q9H2S6
ENSG00000000419  0.175937        DPM1        8813      H0Y368
ENSG00000000457  0.961682       SCYL3       57147      X6RHX1
ENSG00000000460  0.815805       FIRRM       55732      A6NFP1
ENSG00000000938        NA         FGR        2268      B7Z6W7
                           genename
                          <character>
ENSG00000000003          tetraspanin 6
ENSG00000000005            tenomodulin
ENSG00000000419 dolichyl-phosphate m..
ENSG00000000457 SCY1 like pseudokina..
ENSG00000000460 FIGNL1 interacting r..
ENSG00000000938 FGR proto-oncogene, ..
```

```
library(EnhancedVolcano)
```

```
Loading required package: ggrepel
```

```
x <- as.data.frame(res)
head(x)
```

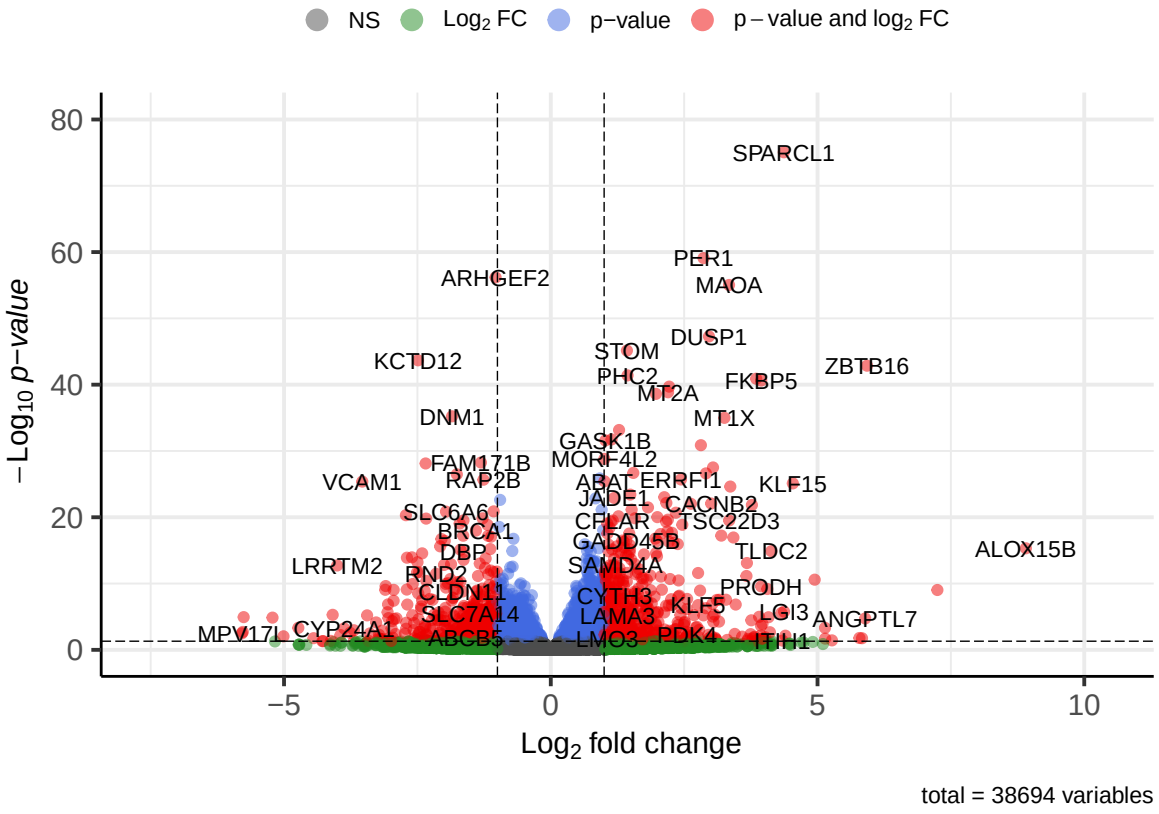
```
                  baseMean log2FoldChange     lfcSE       stat     pvalue
ENSG00000000003 747.1941954    -0.35070296 0.1682421 -2.0845139 0.03711345
ENSG00000000005   0.0000000             NA        NA         NA         NA
ENSG00000000419 520.1341601     0.20610728 0.1010415  2.0398279 0.04136747
ENSG00000000457 322.6648439     0.02452701 0.1451339  0.1689958 0.86579996
ENSG00000000460  87.6826252    -0.14714263 0.2569954 -0.5725496 0.56694971
ENSG00000000938   0.3191666    -1.73228897 3.4936010 -0.4958463 0.62000288
                     padj symbol entrez    uniprot
ENSG00000000003 0.1630172 TSPAN6   7105 A0A087WYV6
ENSG00000000005        NA   TNMD  64102     Q9H2S6
ENSG00000000419 0.1759366   DPM1   8813     H0Y368
ENSG00000000457 0.9616825  SCYL3  57147     X6RHX1
ENSG00000000460 0.8158052  FIRRM  55732     A6NFP1
ENSG00000000938        NA    FGR   2268     B7Z6W7
                                                    genename
ENSG00000000003                                 tetraspanin 6
ENSG00000000005                                   tenomodulin
```

```
ENSG00000000419 dolichyl-phosphate mannosyltransferase subunit 1, catalytic
ENSG00000000457                                SCY1 like pseudokinase 3
ENSG00000000460   FIGNL1 interacting regulator of recombination and mitosis
ENSG00000000938              FGR proto-oncogene, Src family tyrosine kinase
```

```
EnhancedVolcano(
    x,
    lab = res$symbol,                     # gene labels
    x = 'log2FoldChange',                # x-axis: fold change
    y = 'pvalue',                        # y-axis: p-value
    title = 'Differential Expression (Condition A vs B)',
    subtitle = 'DESeq2 results',
    xlab = bquote(~Log[2]~ 'fold change'),
    ylab = bquote(~-Log[10]~italic('p-value')),
    pCutoff = 0.05                       # significance cutoff (no trailing comma!)
)
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
i The deprecated feature was likely used in the EnhancedVolcano package.
  Please report the issue to the authors.
```

```
Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
i Please use the `linewidth` argument instead.
i The deprecated feature was likely used in the EnhancedVolcano package.
  Please report the issue to the authors.
```



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)

# 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"
```

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

res$entrez <- mapIds(org.Hs.eg.db,
                     keys = rownames(res),    # your Ensembl IDs
                     column = "ENTREZID",
                     keytype = "ENSEMBL",
                     multiVals = "first")
```

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

```
sum(!is.na(res$entrez))   # should now be most genes
```

```
[1] 27451
```

```
head(res$entrez)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         "7105"          "64102"           "8813"          "57147"          "55732"
ENSG00000000938
         "2268"
```

```
head(res$entrez)
```

```
ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
         "7105"          "64102"           "8813"          "57147"          "55732"
ENSG00000000938
         "2268"
```

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

```
       7105       64102        8813       57147       55732        2268
-0.35070296          NA  0.20610728  0.02452701 -0.14714263 -1.73228897
```

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

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

```
# Look at the first three down (less) pathways
head(keggres$less, 3)
```

```
                                    p.geomean stat.mean        p.val
hsa05332 Graft-versus-host disease 0.0004250607 -3.473335 0.0004250607
hsa04940 Type I diabetes mellitus  0.0017820379 -3.002350 0.0017820379
hsa05310 Asthma                    0.0020046180 -3.009045 0.0020046180
                                       q.val set.size         exp1
hsa05332 Graft-versus-host disease 0.09053792       40 0.0004250607
hsa04940 Type I diabetes mellitus  0.14232788       42 0.0017820379
hsa05310 Asthma                    0.14232788       29 0.0020046180
```

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

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

```
Info: Working in directory /Users/courtneyanderson/Desktop/R_Files/BGGN213/Class 12
```

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

```
library(png)
library(grid)

# Read the image file (replace 'my_image.png' with your filename)
img <- readPNG("hsa05310.pathview.png")

# Display the image
grid.raster(img)
```