

Class_15.Rmd

Kaito Tanaka

11/23/2021

2. Import countData and colData

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

Have a look at each

```
head(counts)
```

```
##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG000000000003      723       486       904       445      1170
## ENSG000000000005       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
## ENSG000000000003     1097       806       604
## ENSG000000000005       0         0         0
## ENSG00000000419      781       417       509
## ENSG00000000457      447       330       324
## ENSG00000000460       94        102        74
## ENSG00000000938       0         0         0
```

```
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
## 7 SRR1039520 control  N061011 GSM1275874
## 8 SRR1039521 treated  N061011 GSM1275875
```

Q1. How many genes are in this dataset?

```
nrow(counts)

## [1] 38694

38694
```

Q2. How many ‘control’ cell lines do we have?

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

## [1] 4
```

4

First I need to extract all of the “control” columns. Then I need to take the row-wise mean to get the average count values for all genes in these four experiments.

```
control inds <- metadata$dex == "control"
control counts <- counts[, control inds]
head(control counts)
```

```
##          SRR1039508 SRR1039512 SRR1039516 SRR1039520
## ENSG00000000003     723      904     1170      806
## ENSG00000000005      0        0        0        0
## ENSG00000000419    467      616      582      417
## ENSG00000000457    347      364      318      330
## ENSG00000000460     96       73      118      102
## ENSG00000000938      0        1        2        0
```

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

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

To make the above code in either approach more robust, we could use the general approach in calculating the mean, such that the mean will be correct even when our dataset 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)

```
treated inds <- metadata$dex == "treated"
treated counts <- counts[, treated inds]
head(treated counts)
```

```
##          SRR1039509 SRR1039513 SRR1039517 SRR1039521
## ENSG00000000003     486      445     1097      604
## ENSG00000000005      0        0        0        0
## ENSG00000000419    523      371      781      509
## ENSG00000000457    258      237      447      324
## ENSG00000000460     81       66      94       74
## ENSG00000000938      0        0        0        0
```

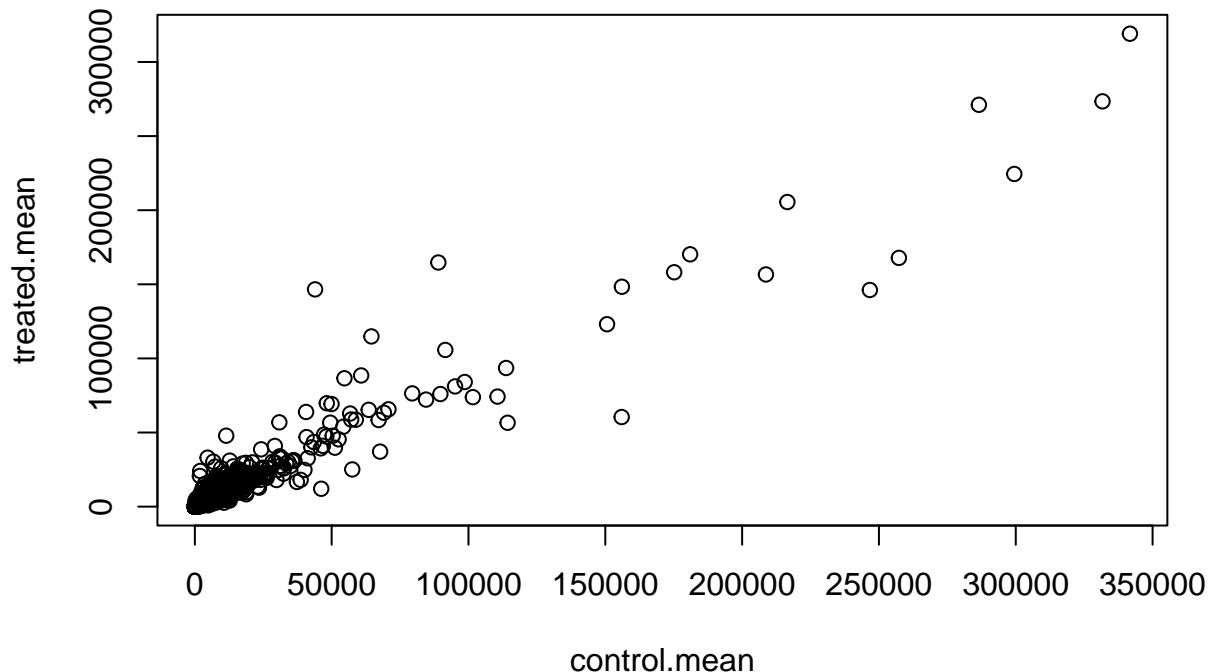
```
treated.mean <- rowMeans(treated.counts)
```

Then combine our meancounts data

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

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. Make a plot of control vs treated

```
plot(meancounts)
```



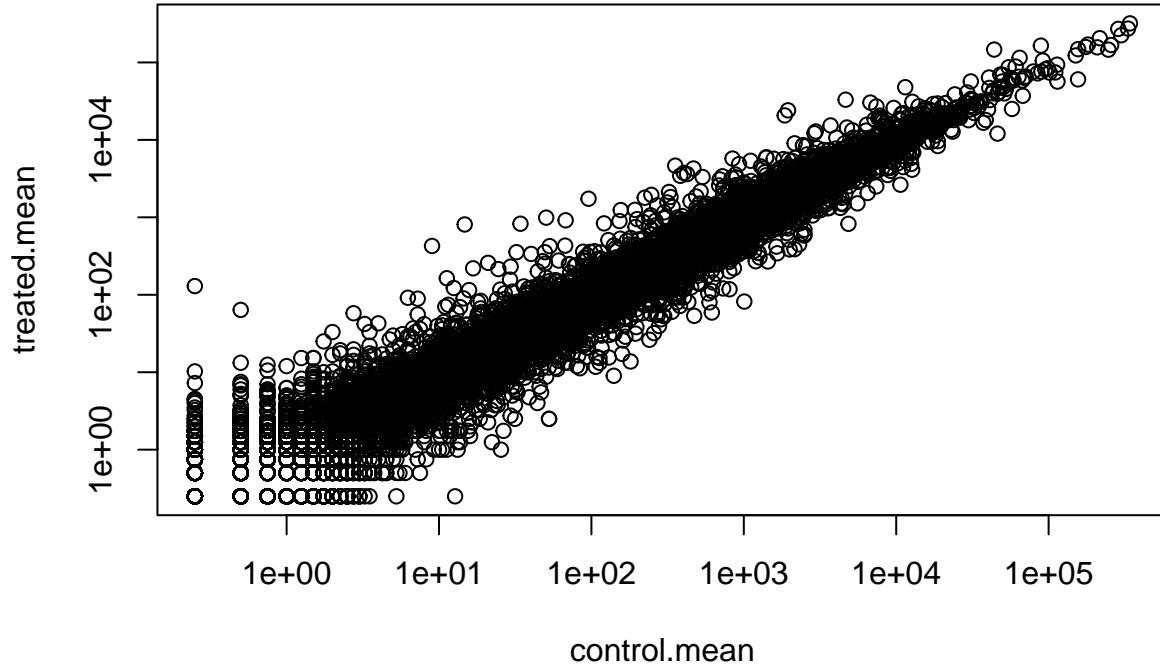
Q5 (b). You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot? geom_point

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

```
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 `log2` in this field because it has nice math properties that make interpretation easier

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

```
## [1] 0
```

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

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

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

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

```
log2(5/10)
```

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

We see 0 values for no change, + values for increases, and - values for decreases. This nice property leads us to work with `log2(fold-change)` all the time in genomics and proteomics field.

Now add the `log2(fold-change)` values to the `meancounts` data frame.

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

```
##                                     control.mean treated.mean      log2fc
## ENSG000000000003            900.75     658.00 -0.45303916
## ENSG000000000005             0.00      0.00       NaN
## ENSG000000000419            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
```

We see some infinities in the data so we need to filter them out.

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

```
##                                     control.mean treated.mean
## ENSG000000000003            FALSE      FALSE
## ENSG000000000005             TRUE      TRUE
## ENSG000000000419            FALSE      FALSE
## ENSG00000000457            FALSE      FALSE
## ENSG00000000460            FALSE      FALSE
## ENSG00000000938            FALSE      TRUE
```

I can use the **which()** function with the ‘arr.ind=TRUE’ argument to get the columns and rows where the TRUE values are (i.e. the zero counts)

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

```
##           row col
## ENSG000000000005    2   1
## ENSG00000004848   65   1
## ENSG00000004948   70   1
## ENSG00000005001   73   1
## ENSG00000006059  121   1
## ENSG00000006071  123   1
```

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?

The arr.ind argument will allow the function to return the rows and columns that have TRUE values (the zeros in our data). The unique() function is used to ensure that the function doesn’t count a row twice if both the control and treated have TRUE.

```
to.rm <- unique(zero.vals[, "row"])
head(sort(to.rm))
```

```
## [1] 2 6 65 70 73 81
```

Now we can remove these from our ‘meancounts’ data frame.

```
head(meancounts)
```

```
## control.mean treated.mean log2fc
## ENSG000000000003 900.75 658.00 -0.45303916
## ENSG000000000005 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
```

```
mycounts <- meancounts[-to.rm,]
head(mycounts)
```

```
## control.mean treated.mean log2fc
## ENSG000000000003 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
```

How many genes are left?

```
nrow(mycounts)
```

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

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

250

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

367

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

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

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

I do not trust these results because I am not sure of whether these changes are statistically significant. We aren't accounting for fold changes and significance, and thus could be a cause of statistical insignificance.

4. DeSEQ Analysis

Lets do this the right way. DESeq2 is an R-package specifically for analyzing count based NGS data like RNA-seq. It is available from Bioconductor.

```
library(DESeq2)

## Loading required package: S4Vectors

## Warning: package 'S4Vectors' was built under R version 4.1.2

## 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

## Warning: package 'GenomicRanges' was built under R version 4.1.2

## 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

citation("DESeq2")

```

```

##
## 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},
## }
##
```

```

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
```

Now run the DESeq analysis

```

dds <- DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

Results using the DESeq ‘results()’ function:

```

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>
## ENSG000000000003  747.1942    -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005   0.0000       NA        NA        NA        NA
## ENSG000000000419  520.1342    0.2061078  0.101059  2.039475 0.0414026
## ENSG000000000457  322.6648    0.0245269  0.145145  0.168982 0.8658106
## ENSG000000000460   87.6826   -0.1471420  0.257007 -0.572521 0.5669691
## ...
##           ...      ...
## ENSG00000283115   0.000000       NA        NA        NA        NA
## ENSG00000283116   0.000000       NA        NA        NA        NA
## ENSG00000283119   0.000000       NA        NA        NA        NA
## ENSG00000283120   0.974916   -0.668258  1.69456 -0.394354 0.693319
## ENSG00000283123   0.000000       NA        NA        NA        NA
##           padj
##          <numeric>
## ENSG000000000003  0.163035
## ENSG000000000005   NA
## ENSG000000000419  0.176032
## ENSG000000000457  0.961694
## ENSG000000000460  0.815849
## ...
##           ...
## ENSG00000283115   NA
## ENSG00000283116   NA
## ENSG00000283119   NA
## ENSG00000283120   NA
## ENSG00000283123   NA

```

5. Adding Annotation Data

```
BiocManager::install("AnnotationDbi")
```

Bioconductor version 3.14 (BiocManager 1.30.16), R 4.1.1 (2021-08-10)

```
## Warning: package(s) not installed when version(s) same as current; use 'force = TRUE' to
##   re-install: 'AnnotationDbi'
```

```
BiocManager::install("org.Hs.eg.db")
```

Bioconductor version 3.14 (BiocManager 1.30.16), R 4.1.1 (2021-08-10)

```
## Warning: package(s) not installed when version(s) same as current; use 'force = TRUE' to
##   re-install: 'org.Hs.eg.db'
```

```

library("AnnotationDbi")

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

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

```

##

Examine what is in `org.Hs.eg.db`

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

```

## [1] "ACCCNUM"          "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"

```

Use the `mapIDs()` function to add individual columns to our results table.

```

res$symbol <- mapIDs(org.Hs.eg.db,
                      keys=row.names(res),
                      keytype="ENSEMBL",
                      column="SYMBOL",
                      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>
## ENSG000000000003  747.194195    -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005   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>
## ENSG000000000003   0.163035    TSPAN6
## ENSG000000000005      NA        TNMD
## ENSG00000000419   0.176032    DPM1
## ENSG00000000457   0.961694    SCYL3
## ENSG00000000460   0.815849    C1orf112
## ENSG00000000938      NA        FGR

```

Q11. Run the mapIds() function two more times to add the Entrez ID and UniProt accession and GENENAME as new columns called res\$entrez, res\$uniprot and res\$genename.

```

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

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>
## ENSG000000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005   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      entrez      uniprot
##          <numeric> <character> <character> <character>
## ENSG000000000003  0.163035    TSPAN6       7105 AOA024RCIO
## ENSG000000000005   NA          TNMD       64102 Q9H2S6
## ENSG00000000419   0.176032    DPM1        8813 060762
## ENSG00000000457   0.961694    SCYL3       57147 Q8IZE3
## ENSG00000000460   0.815849    C1orf112    55732 AOA024R922
## ENSG00000000938   NA          FGR        2268 P09769
##          genename
##          <character>
## ENSG000000000003   tetraspanin 6

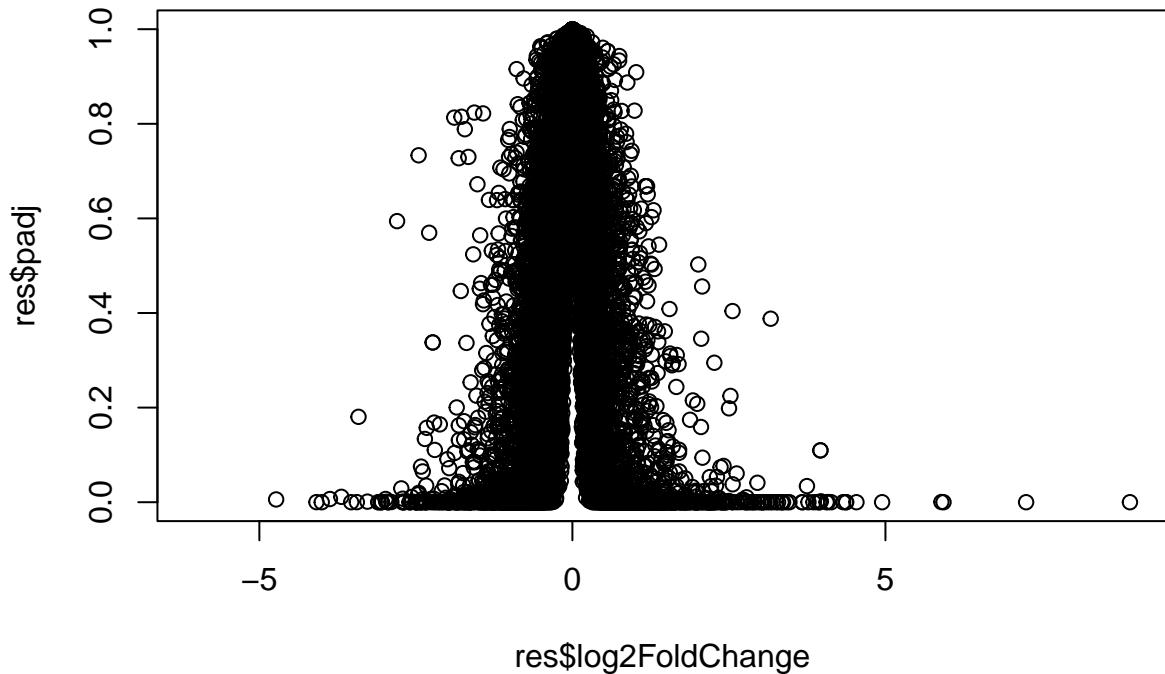
```

```
## ENSG00000000005      tenomodulin
## ENSG00000000419 dolichyl-phosphate m..
## ENSG00000000457 SCY1 like pseudokina..
## ENSG00000000460 chromosome 1 open re..
## ENSG00000000938 FGR proto-oncogene, ..
```

6. Make a Volcano plot Let's make a commonly produced visualization from this data, namely a so-called Volcano plot. These summary figures are frequently used to highlight the proportion of genes that are both significantly regulated and display a high fold change.

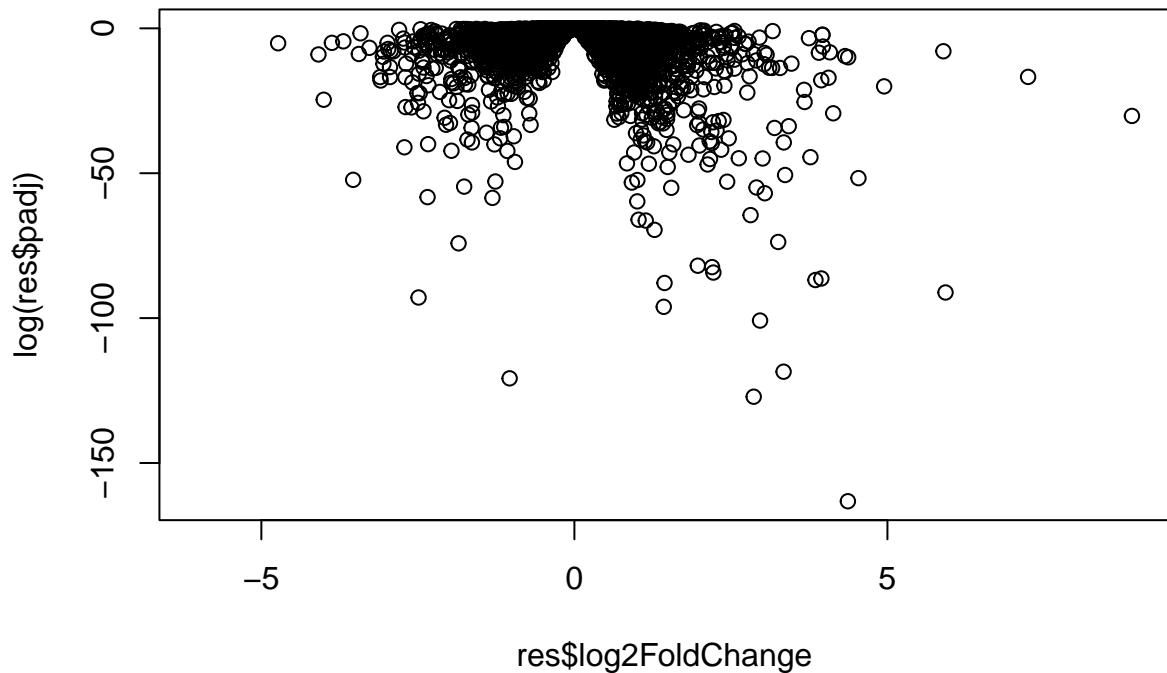
Let's use base R

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



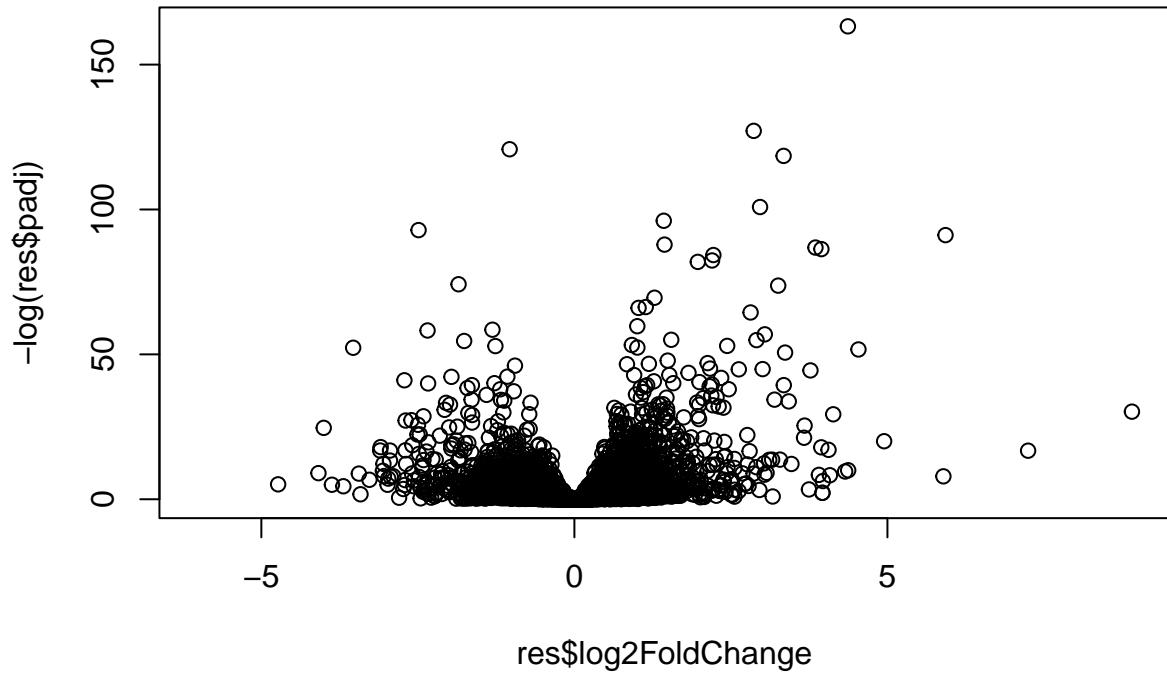
The above plot is not so useful because all smaller p-values are hidden at the bottom of the plot, and as a result, is hard to see. use log!

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



Then, use $-\log$ to flip the p-value.

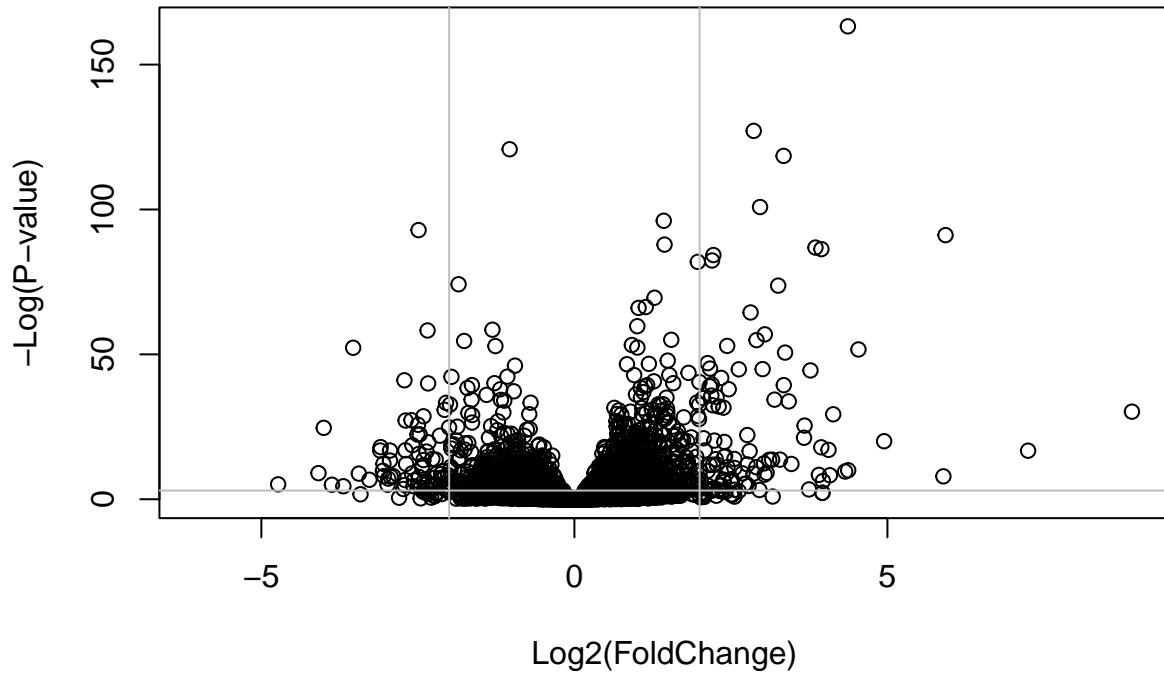
```
plot(res$log2FoldChange, -log(res$padj))
```



Now let's add guidelines: labels and colors, to the plot to draw attention to the genes (points on the graph) we care about. These are the ones that have large fold-changes and low p-values.

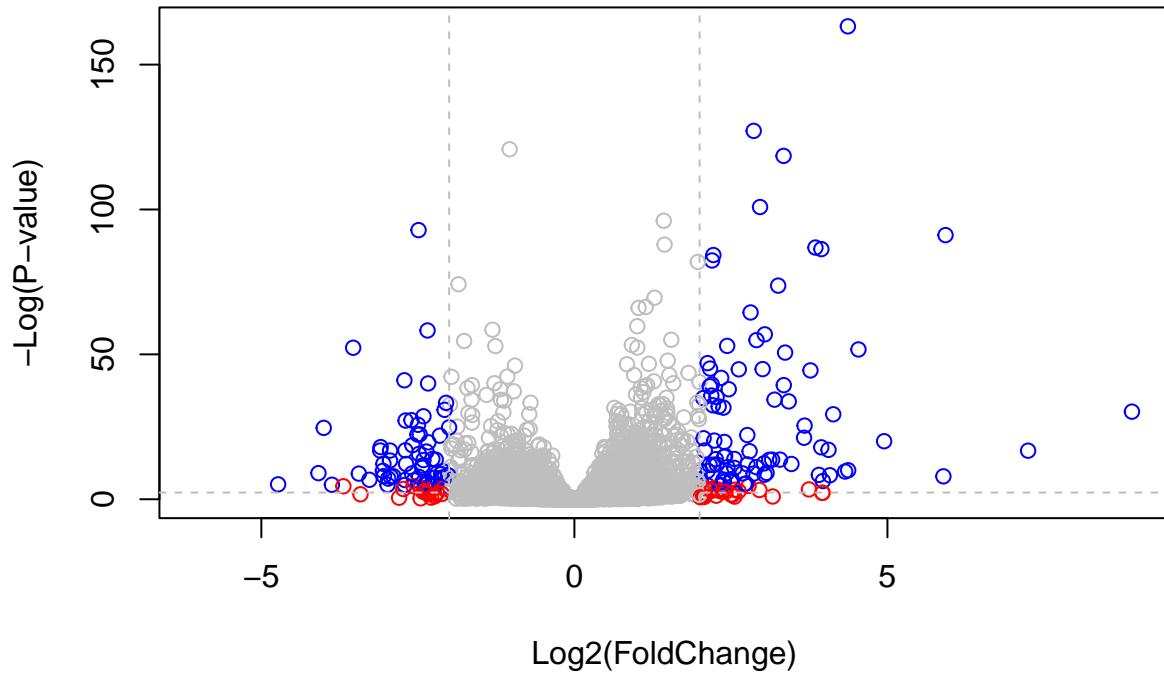
first, labels

```
plot(res$log2FoldChange, -log(res$padj), ylab="-Log(P-value)", xlab="Log2(FoldChange)")
abline(v=c(-2, +2), col="gray")
abline(h=-log(0.05), col="gray")
```



Now add color!

```
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"
inds <- (res$padj < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"
plot( res$log2FoldChange, -log(res$padj),
col=mycols, ylab="-Log(P-value)", xlab="Log2(FoldChange)")
abline(v=c(-2,2), col="gray", lty=2)
abline(h=-log(0.1), col="gray", lty=2)
```



7. Pathway Analysis

```
#BiocManager::install( c("pathview", "gage", "gageData") )

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"

```

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$entrez`) and we have the *foldchange results from DESeq2 analysis* (stored in `res$log2FoldChange`).

```

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

```

```

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

```

Now lets look at the object returned from `gage()`.

```
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 0.0004250461 -3.473346 0.0004250461
## hsa04940 Type I diabetes mellitus 0.0017820293 -3.002352 0.0017820293
## hsa05310 Asthma                  0.0020045888 -3.009050 0.0020045888
##                                     q.val set.size      exp1
## hsa05332 Graft-versus-host disease 0.09053483      40 0.0004250461
## hsa04940 Type I diabetes mellitus 0.14232581      42 0.0017820293
## hsa05310 Asthma                  0.14232581      29 0.0020045888

```

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

```

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

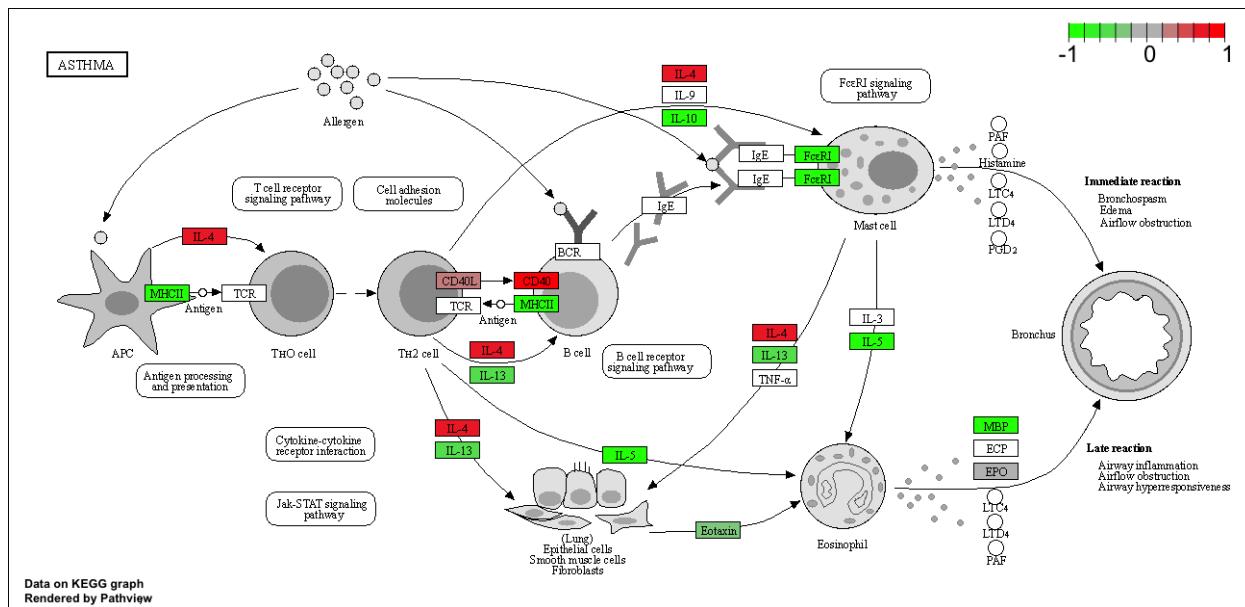
```

```

## Info: Working in directory /Users/kaito2427/Desktop/BIMM 143 Lab/Class 15

```

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



```
# A different PDF based output of the same data
pathview(gene.data=foldchanges, pathway.id="hsa05310", kegg.native=FALSE)
```

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

```
## Info: Working in directory /Users/kaito2427/Desktop/BIMM 143 Lab/Class 15
```

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

Q12. Can you do the same procedure as above to plot the pathview figures for the top 2 down-regulated pathways?

```
pathview(gene.data=foldchanges, pathway.id="hsa04110")
```

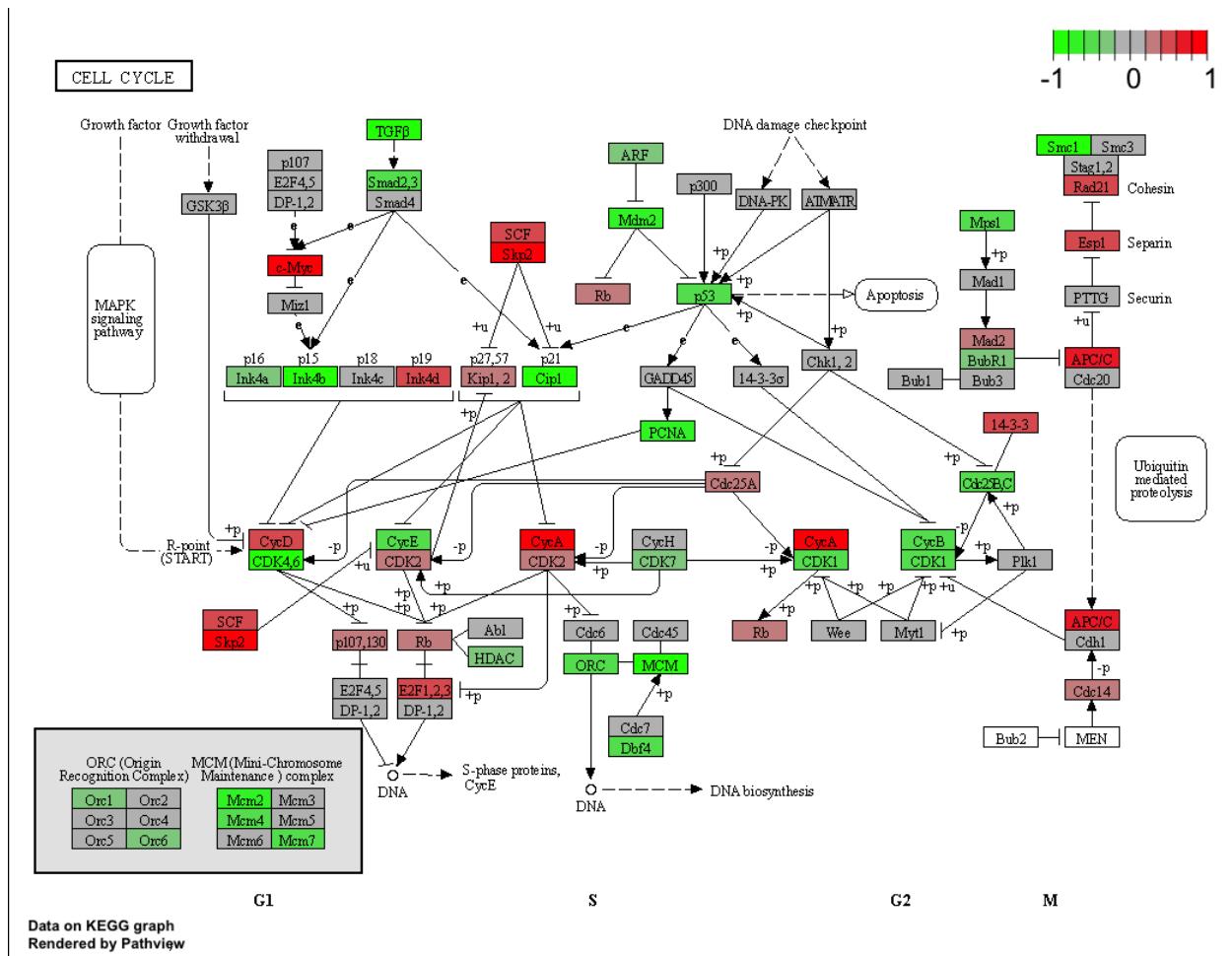
```
## Info: Downloading xml files for hsa04110, 1/1 pathways..
```

```
## Info: Downloading png files for hsa04110, 1/1 pathways..
```

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

```
## Info: Working in directory /Users/kaito2427/Desktop/BIMM 143 Lab/Class 15
```

```
## Info: Writing image file hsa04110.pathview.png
```



```
pathview(gene.data=foldchanges, pathway.id="hsa03030")
```

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

```
## Info: Working in directory /Users/kaito2427/Desktop/BIMM 143 Lab/Class 15
```

```
## Info: Writing image file hsa03030.pathview.png
```

