Class15 RNASeq Analysis

Gabrielle Meza (A13747395)

11/17/2021

#Background

Today we're examining a published RNA-seq experiment where irway smooth muscle cells were yreated with dexamethasone, a synthetic glucocorticoid steriod with anti-inflammatory effects (Himes et al.)

We need: 1) Count of Data 2) col of data

The count of data, is the actual data, with each gene as a column header. the col is the metadata, and each gene is a row header

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

```
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
```

> Q1. How many genes are in this dataset?

38694 total genes

```
nrow(counts)
```

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

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

4

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

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

Side note:

lets check the correspondence of the metadata and count data setup

```
metadata $id
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

```
colnames(counts)
```

```
## [1] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" "SRR1039516"
## [6] "SRR1039517" "SRR1039520" "SRR1039521"
```

We can use the `==` thing to see if they colums and rows are the same

```
metadata$id == colnames(counts)
```

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

We are going to wrap this in an `all` thing so it will tell us if they all are true or not. if there is a flase andwhere, then it would not give out TRUE. We would use this for a larger dataset potentially.

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

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

Compare control to treated

first we need to access all the control columns in our counts data.

```
metadata$dex == "control"
```

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

```
control.inds <- metadata$dex == "control"

#this is pulling out the rows that are control in the metadata sheet
metadata[ control.inds, ]
```

```
##          id     dex celltype     geo_id
## 1 SRR1039508 control   N61311 GSM1275862
## 3 SRR1039512 control  N052611 GSM1275866
## 5 SRR1039516 control  N080611 GSM1275870
## 7 SRR1039520 control  N061011 GSM1275874
```

```
#now use $id to get the ids that are controls and make it a new value

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

use these ids to access just the control columns of our `counts` data. Use head becuase it would be a large dataset

```
head(counts[ , control.ids])
```

```
##                 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(counts[ , control.ids])
head(control.mean)
```

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

> Q4. Follow the same procedure for the treated samples

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

```
## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
##          658.00            0.00          546.00          316.50           78.75
## ENSG00000000938
##            0.00
```

we will combine our meancount data for bookkeeping purposes

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

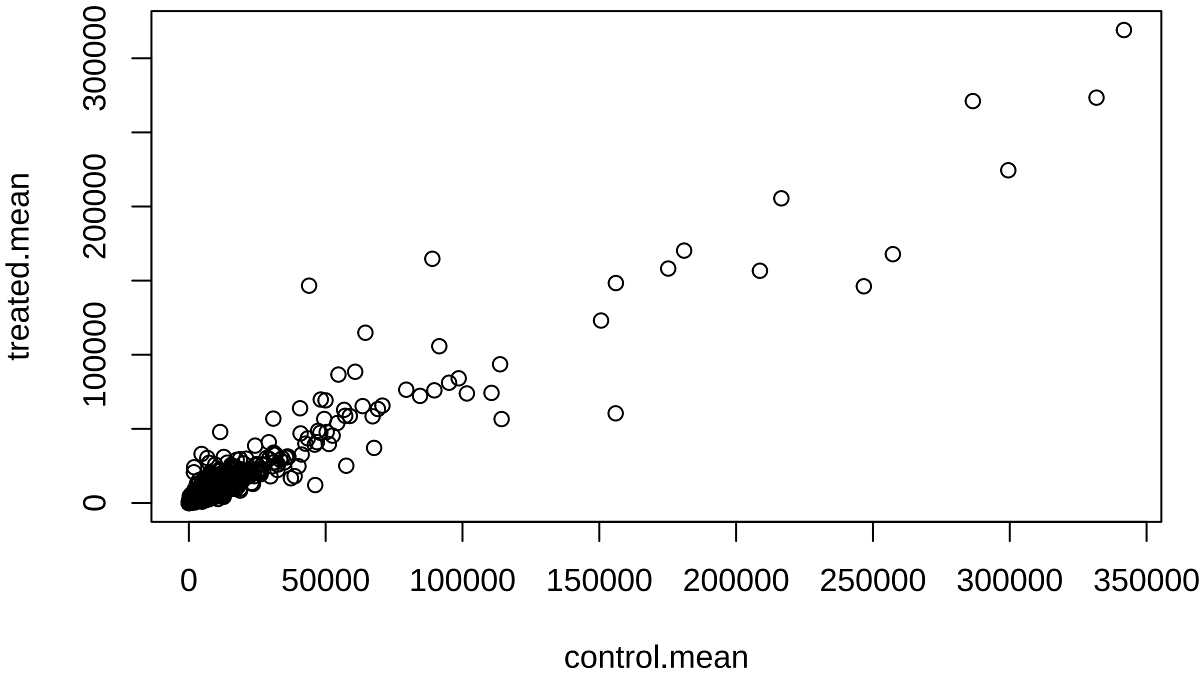
there are 38694 rows/genes in this dataset. You can do this int he printout and have it run the code in the printout with * but you would have r rcomand in the *

Compare the control and treated

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

Quick plot of our progress so far

```
plot(meancounts)
```

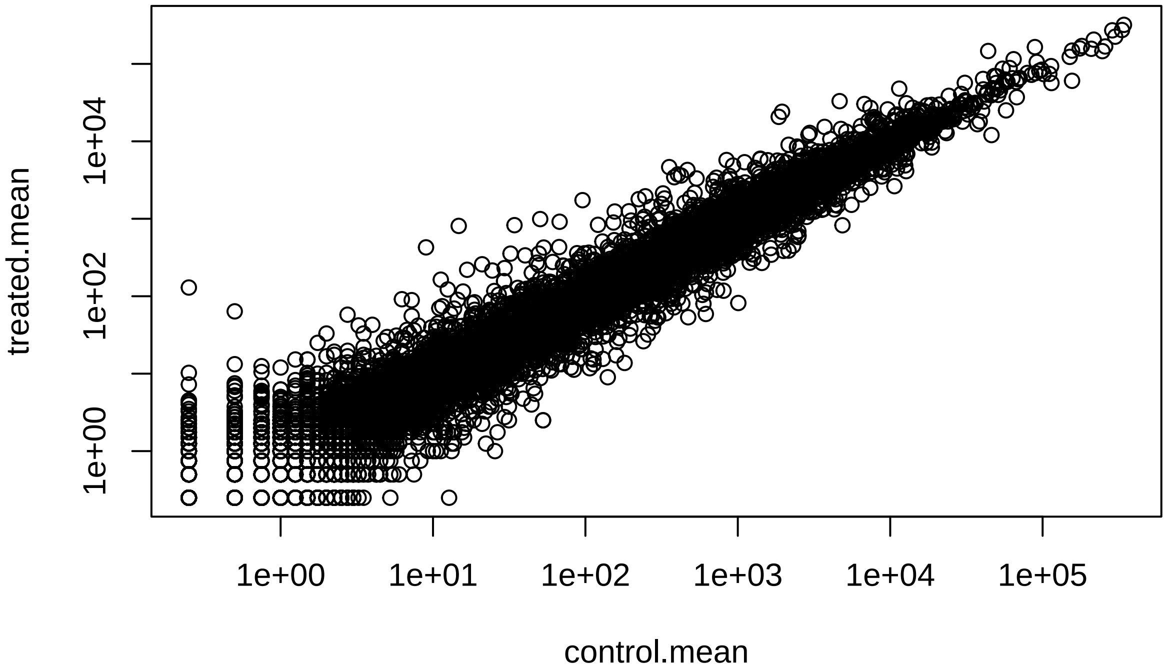


there should be many more points showen (38,000!) this could better be represented by a log scale

```
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 nicer in this world…

```
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
```

We need to drop the zero count genes/ rows!

```
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.

```
which (c(T,F,T))
```

```
## [1] 1 3
```

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

```
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). You then want to use **`unique`** to get the rows you want to look at

```
to.rm <- unique(inds[,1])
mycounts <- (meancounts[-to.rm,])
```

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

```
##                 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 genens are upregulated at the log 2 fold change threshold of +2 or greater?

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

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

what percentage is this?

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

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

How about the downreguolated genes?

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

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

DESeq2 analysis

```
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
```

We first need to setip the DESeq

design is where in the col data do we care about?

```
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
```

Run the DESeq anaylsis pipeline. it is doing a lot of other analysis, like p-value and stuff. it is being saved as a dds with all of the results. you can use the package default `results()` to read the data and out it into a new vairable to then to pull it up and read it all.

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

```
## estimating size factors
```

```
## estimating dispersions
```

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

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

```
## final dispersion estimates
```

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

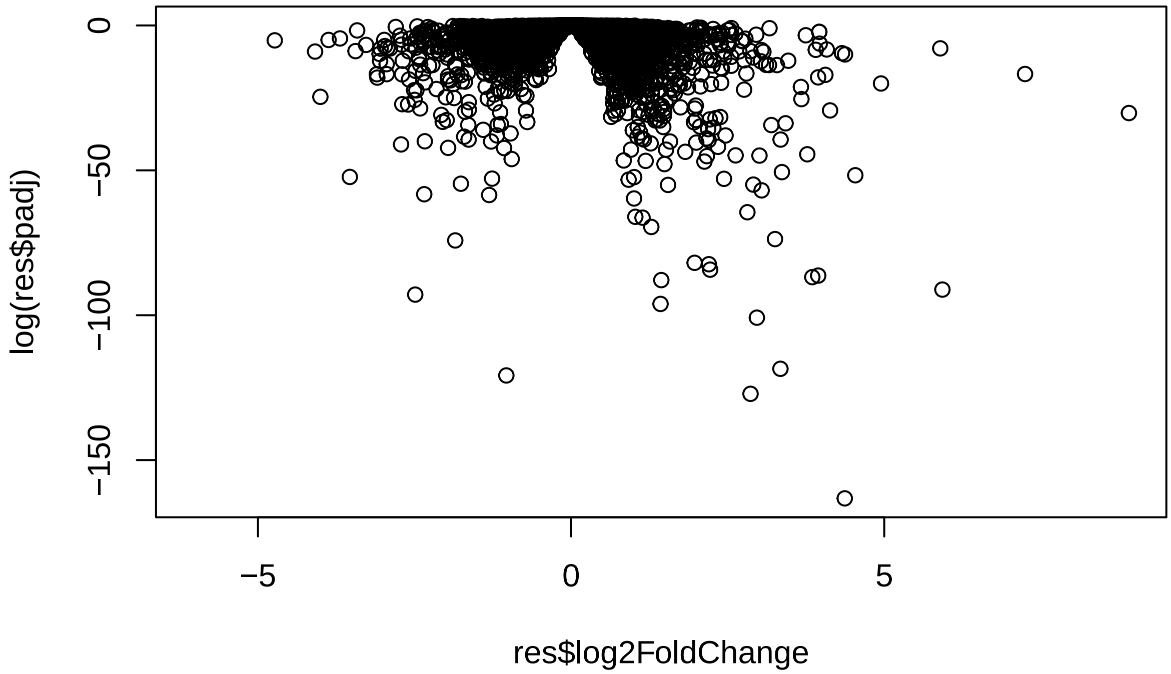
```
res <- results(dds)
head(res)
```

```
## 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. you can use base r package to plot this

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

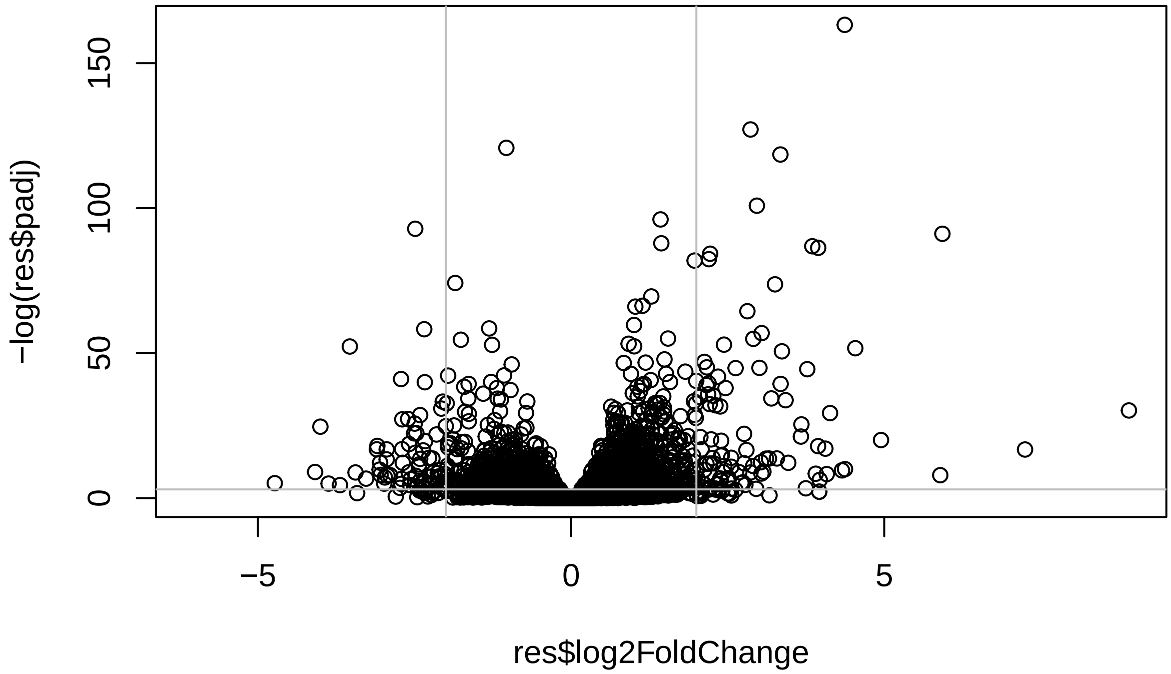


when you take a log, the values you are interested in at the very low numbers since it is a p-vaule. To make this easier visually, you can put a negative sign infront of log

with log, as you move up in expression, go more positive. as you go down in expression, go more negative

use abline to add a line at certain points. here we are making the line at the set pValue (horizontal) to be sugnificant. And then the cut off if there is a 2 fold change in either neg or positive direction.

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



Adding annotation data

Now lets color and annotate this data further! 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 be installing and using 2 more packages in biocManager. I installed these in the console. **AnnotationDbi** the other contains data we are going to map between and is **org.Hs.eg.db**

```
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"
```

Here we map to “SYMBOL” the comon gene name that the world understands and wants

```
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>
## 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
```

Let’s now save this data

Let’s finally save our results as a .csv file so we can use for a lter time, and use in different area if needed.

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

DAY 2- Pathway analysis

Let’s try to bring some biology insight back into this work. for this we will us KEGG. So let’s load up KEGG for pathway analysis.

For this, we will have to install some stuff 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)
```

```
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 gen identifiers in the correct format for KEGG, which is ENTREZ format in this case. So let slook into the dataset again to get familiar with th elanguage

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

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

```
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"
```

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)andwehavethefoldchangeresul*

```
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
```

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

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

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

```
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
```

There we go! now let's run the **gage()** pathway analysis. Get the results by doing keggres = and the gage function, and it saves the data as keggres

We can then look at the attributes() of this or indeed any R object

```
keggres = gage(foldchanges, gsets=kegg.sets.hs)
attributes(keggres)
```

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

```
head(keggres$greater)
```

```
##                                             p.geomean stat.mean       p.val
## hsa00500 Starch and sucrose metabolism    0.002822007  2.825461 0.002822007
## hsa00330 Arginine and proline metabolism 0.012317455  2.280002 0.012317455
## hsa04910 Insulin signaling pathway       0.017110962  2.129511 0.017110962
## hsa04510 Focal adhesion                  0.025239833  1.961955 0.025239833
## hsa04920 Adipocytokine signaling pathway 0.043426078  1.725063 0.043426078
## hsa00790 Folate biosynthesis             0.048254489  1.744387 0.048254489
##                                              q.val set.size        exp1
## hsa00500 Starch and sucrose metabolism    0.6010875       54 0.002822007
## hsa00330 Arginine and proline metabolism 0.7774866       54 0.012317455
## hsa04910 Insulin signaling pathway       0.7774866      138 0.017110962
## hsa04510 Focal adhesion                  0.7774866      200 0.025239833
## hsa04920 Adipocytokine signaling pathway 0.7774866       68 0.043426078
## hsa00790 Folate biosynthesis             0.7774866       11 0.048254489
```

```
head(res)
```

```
## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 9 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      entrez               genename
##                 <numeric> <character> <character>            <character>
## ENSG00000000003  0.163035      TSPAN6        7105          tetraspanin 6
## ENSG00000000005        NA        TNMD       64102            tenomodulin
## ENSG00000000419  0.176032        DPM1        8813 dolichyl-phosphate m..
## ENSG00000000457  0.961694       SCYL3       57147 SCY1 like pseudokina..
## ENSG00000000460  0.815849    C1orf112       55732 chromosome 1 open re..
## ENSG00000000938        NA         FGR        2268 FGR proto-oncogene, ..
```

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/Gabriellemeza/Desktop/UCSD/Bioinformatics/bggn213_github/Class15
```

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