

DESeq Mini Project

Chloe J. Welch

11/19/2021

The workflow for this mini project will be as follows:

- 1) Data import/read (countdata and coldata)
- 2) PCA (QC)
- 3) DESeq analysis
- 4) Volcano plot

```
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, colAveragesPerRowSet, 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, rowAveragesPerColSet,
##   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

```

```
metaFile <- "GSE37704_metadata.csv"
countFile <- "GSE37704_featurecounts.csv"
```

Import metadata and take a peak

```
colData = read.csv(metaFile, row.names=1)
head(colData)
```

```
##           condition
## SRR493366 control_sirna
## SRR493367 control_sirna
## SRR493368 control_sirna
## SRR493369      hoxa1_kd
## SRR493370      hoxa1_kd
## SRR493371      hoxa1_kd
```

Import countdata

```
countData = read.csv(countFile, row.names=1)
head(countData)
```

```
##           length SRR493366 SRR493367 SRR493368 SRR493369 SRR493370
## ENSG00000186092      918         0         0         0         0         0
## ENSG00000279928      718         0         0         0         0         0
## ENSG00000279457     1982        23        28        29        29        28
## ENSG00000278566      939         0         0         0         0         0
## ENSG00000273547      939         0         0         0         0         0
## ENSG00000187634     3214        124        123        205        207        212
##           SRR493371
## ENSG00000186092         0
## ENSG00000279928         0
## ENSG00000279457        46
## ENSG00000278566         0
## ENSG00000273547         0
## ENSG00000187634       258
```

```
# Note we need to remove the odd first $length col
countData1 <- as.matrix(countData[,-1])
head(countData1)
```

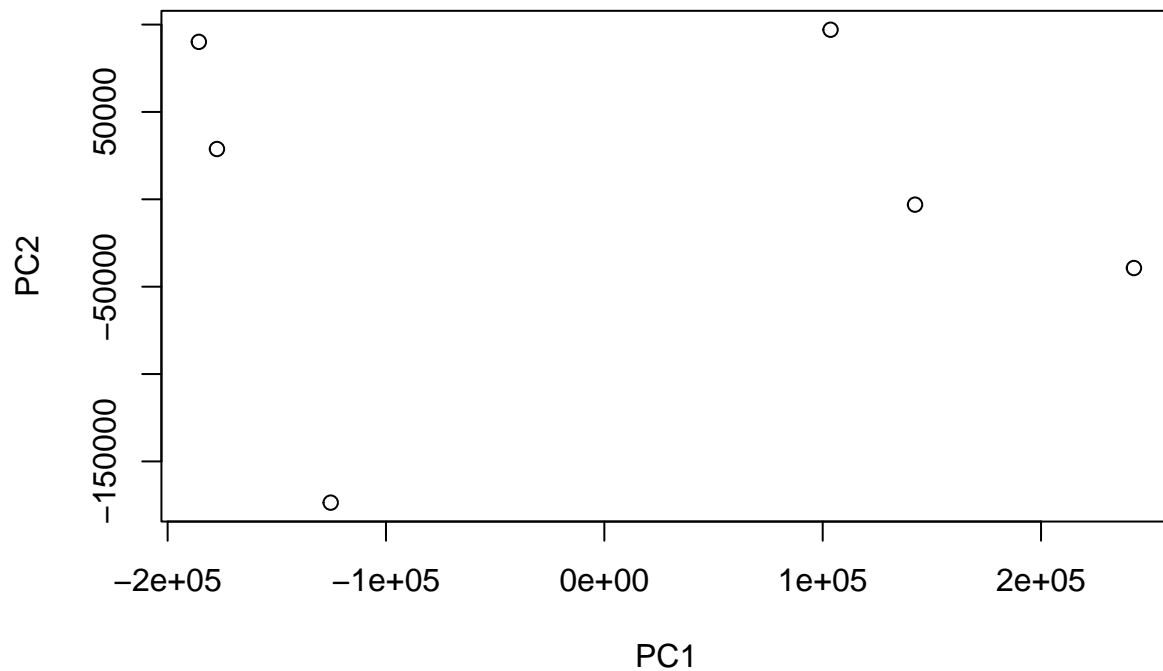
```
##           SRR493366 SRR493367 SRR493368 SRR493369 SRR493370 SRR493371
## ENSG00000186092         0         0         0         0         0         0
## ENSG00000279928         0         0         0         0         0         0
## ENSG00000279457        23        28        29        29        28        46
## ENSG00000278566         0         0         0         0         0         0
## ENSG00000273547         0         0         0         0         0         0
## ENSG00000187634       124       123       205       207       212       258
```

```
# Filter count data where you have 0 read count across all samples.
countData1 = countData1[-which(rowSums(countData1) == 0), ]
head(countData1)
```

```
##           SRR493366 SRR493367 SRR493368 SRR493369 SRR493370 SRR493371
## ENSG00000279457      23       28       29       29       28       46
## ENSG00000187634     124      123      205      207      212      258
## ENSG00000188976    1637     1831     2383     1226     1326     1504
## ENSG00000187961     120      153      180      236      255      357
## ENSG00000187583      24       48       65       44       48       64
## ENSG00000187642       4        9       16       14       16       16
```

Now, let's try to run PCA

```
pca <- prcomp(t(countData1))
plot(pca$x[,1:2])
```



Now, let's set up the `DESeqDataSet` object required for the `DESeq()` function and then run the DESeq pipeline. This is again similar to our last day's hands-on session.

```
dds = DESeqDataSetFromMatrix(countData=countData1,
                              colData=colData,
                              design=~condition)

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

dds = DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

dds

## class: DESeqDataSet
## dim: 15975 6
## metadata(1): version
## assays(4): counts mu H cooks
## rownames(15975): ENSG00000279457 ENSG00000187634 ... ENSG00000276345
##      ENSG00000271254
## rowData names(22): baseMean baseVar ... deviance maxCooks
## colnames(6): SRR493366 SRR493367 ... SRR493370 SRR493371
## colData names(2): condition sizeFactor
```

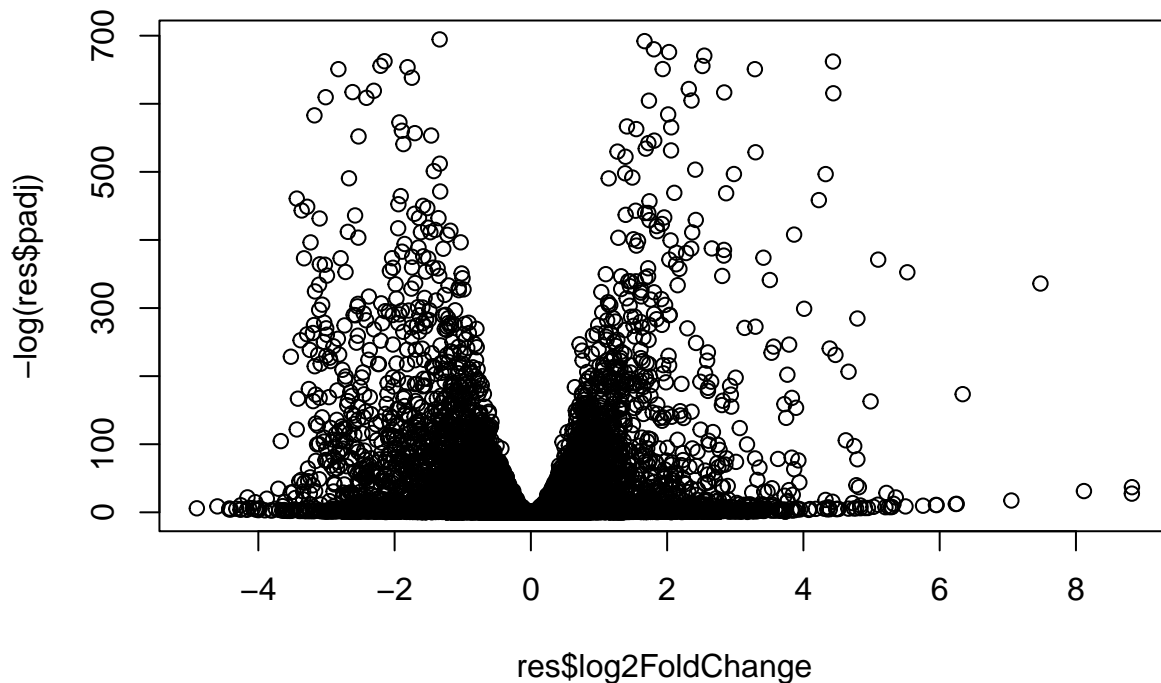
Next, get results for the HoxA1 knockdown versus control siRNA (remember that these were labeled as “hoxa1_kd” and “control_sirna” in our original `colData` metaFile input to `DESeq`; you can check this above and by running the `resultsNames(dds)` command).

```
res = results(dds)
summary(res)
```

```
##
## out of 15975 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 4349, 27%
## LFC < 0 (down)    : 4396, 28%
## outliers [1]      : 0, 0%
## low counts [2]    : 1237, 7.7%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Now we will make a volcano plot, a commonly produced visualization from this type of data that we introduced last day. Basically, it's a plot of log2 fold change vs -log adjusted p-value.

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



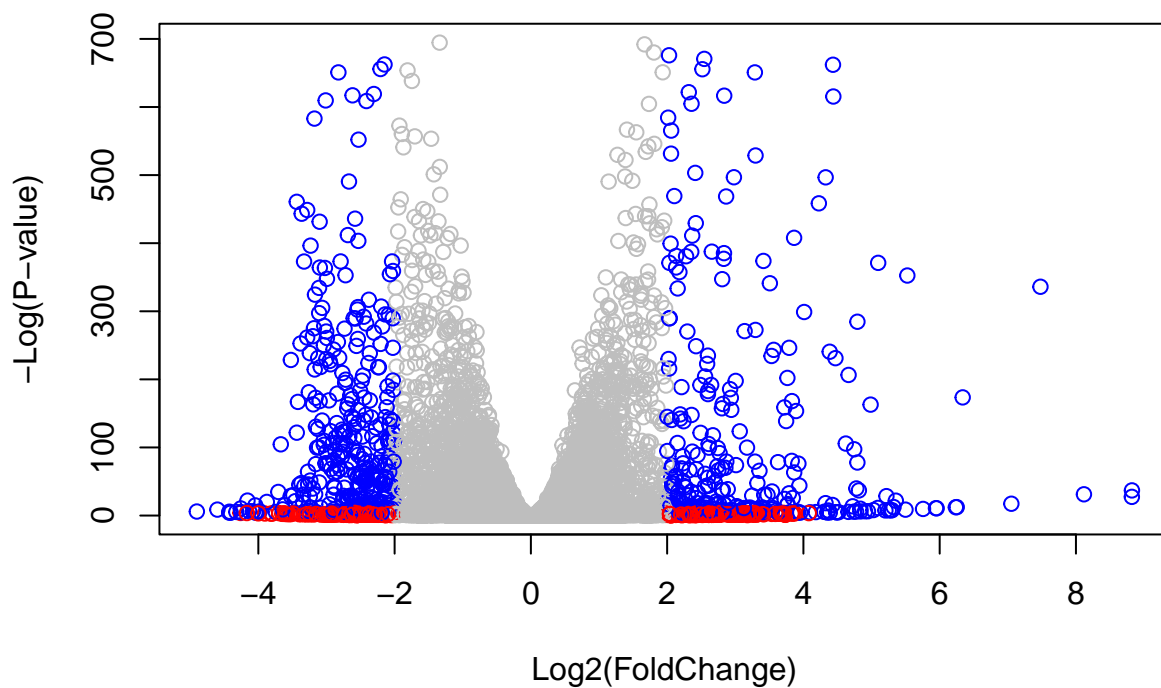
Now, let's add some color to this volcano plot.

```
# Make a color vector for all genes
mycols <- rep("gray", nrow(res) )

# Color red the genes with absolute fold change above 2
mycols[ abs(res$log2FoldChange) > 2 ] <- "red"

# Color blue those with adjusted p-value less than 0.01
# and absolute fold change more than 2
inds <- (res$pvalue < 0.01) & (abs(res$log2FoldChange) > 2 )
mycols[ inds ] <- "blue"

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



Since we mapped and counted against the Ensembl annotation, our results only have information about Ensembl gene IDs. However, our pathway analysis downstream will use KEGG pathways, and genes in KEGG pathways are annotated with Entrez gene IDs. So, let's add them as we did the last day.

Use the `mapIds()` function multiple times to add SYMBOL, ENTREZID and GENENAME annotation to our results by completing the code below.

```
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),
                    keytype="ENSEMBL",
                    column="SYMBOL",
                    multiVals="first")
```

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

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

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



```
head(res, 10)
```

```
## log2 fold change (MLE): condition hoxa1 kd vs control sirna
## Wald test p-value: condition hoxa1 kd vs control sirna
## DataFrame with 10 rows and 9 columns
##
```

	baseMean	log2FoldChange	lfcSE	stat	pvalue
##	<numeric>	<numeric>	<numeric>	<numeric>	<numeric>
## ENSG00000279457	29.913579	0.1792571	0.3248216	0.551863	5.81042e-01
## ENSG00000187634	183.229650	0.4264571	0.1402658	3.040350	2.36304e-03
## ENSG00000188976	1651.188076	-0.6927205	0.0548465	-12.630158	1.43990e-36
## ENSG00000187961	209.637938	0.7297556	0.1318599	5.534326	3.12428e-08
## ENSG00000187583	47.255123	0.0405765	0.2718928	0.149237	8.81366e-01
## ENSG00000187642	11.979750	0.5428105	0.5215598	1.040744	2.97994e-01
## ENSG00000188290	108.922128	2.0570638	0.1969053	10.446970	1.51282e-25
## ENSG00000187608	350.716868	0.2573837	0.1027266	2.505522	1.22271e-02
## ENSG00000188157	9128.439422	0.3899088	0.0467163	8.346304	7.04321e-17
## ENSG00000237330	0.158192	0.7859552	4.0804729	0.192614	8.47261e-01

```
##
```

	padj	symbol	entrez	name
##	<numeric>	<character>	<character>	<character>
## ENSG00000279457	6.86555e-01	WASH9P	102723897	WAS protein family h..
## ENSG00000187634	5.15718e-03	SAMD11	148398	sterile alpha motif ..
## ENSG00000188976	1.76549e-35	NOC2L	26155	NOC2 like nucleolar ..
## ENSG00000187961	1.13413e-07	KLHL17	339451	kelch like family me..
## ENSG00000187583	9.19031e-01	PLEKHN1	84069	pleckstrin homology ..
## ENSG00000187642	4.03379e-01	PERM1	84808	PPARGC1 and ESRR ind..
## ENSG00000188290	1.30538e-24	HES4	57801	hes family bHLH tran..
## ENSG00000187608	2.37452e-02	ISG15	9636	ISG15 ubiquitin like..
## ENSG00000188157	4.21963e-16	AGRN	375790	agrin
## ENSG00000237330	NA	RNF223	401934	ring finger protein ..

Finally for this section, let's reorder these results by adjusted p-value and save them to a CSV file in your current project directory.

```
res = res[order(res$pvalue),]
write.csv(res, file="deseq_results.csv")
```

Here, we are going to use the gage package for pathway analysis. Once we have a list of enriched pathways, we're going to use the pathview package to draw pathway diagrams, shading the molecules in the pathway by their degree of up/down-regulation.

The gageData package has pre-compiled databases mapping genes to KEGG pathways and GO terms for common organisms. kegg.sets.hs is a named list of 229 elements. Each element is a character vector of member gene Entrez IDs for a single KEGG pathway. (See also go.sets.hs). The sigmet.idx.hs is an index of numbers of signaling and metabolic pathways in kegg.set.gs. In other words, KEGG pathway include other types of pathway definitions, like "Global Map" and "Human Diseases", which may be undesirable

in a particular pathway analysis. Therefore, `kegg.sets.hs[sigmet.idx.hs]` gives you the “cleaner” gene sets of signaling and metabolic pathways only.

Side-Note: While there are many freely available tools to do pathway analysis, and some like gage are truly fantastic, many of them are poorly maintained or rarely updated. The DAVID tool that a lot of folks use for simple gene set enrichment analysis was not updated at all between Jan 2010 and Oct 2016.

First, we need to do our one time install of these required bioconductor packages:

```
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)
data(sigmet.idx.hs)
```

```
# Focus on signaling and metabolic pathways only
kegg.sets.hs = kegg.sets.hs[sigmet.idx.hs]
```

```
# Examine the first 3 pathways
head(kegg.sets.hs, 3)
```

```
## $'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"
```

```
##
## $'hsa00230 Purine metabolism'
## [1] "100" "10201" "10606" "10621" "10622" "10623" "107" "10714"
## [9] "108" "10846" "109" "111" "11128" "11164" "112" "113"
## [17] "114" "115" "122481" "122622" "124583" "132" "158" "159"
## [25] "1633" "171568" "1716" "196883" "203" "204" "205" "221823"
## [33] "2272" "22978" "23649" "246721" "25885" "2618" "26289" "270"
## [41] "271" "27115" "272" "2766" "2977" "2982" "2983" "2984"
## [49] "2986" "2987" "29922" "3000" "30833" "30834" "318" "3251"
## [57] "353" "3614" "3615" "3704" "377841" "471" "4830" "4831"
## [65] "4832" "4833" "4860" "4881" "4882" "4907" "50484" "50940"
## [73] "51082" "51251" "51292" "5136" "5137" "5138" "5139" "5140"
## [81] "5141" "5142" "5143" "5144" "5145" "5146" "5147" "5148"
## [89] "5149" "5150" "5151" "5152" "5153" "5158" "5167" "5169"
## [97] "51728" "5198" "5236" "5313" "5315" "53343" "54107" "5422"
## [105] "5424" "5425" "5426" "5427" "5430" "5431" "5432" "5433"
## [113] "5434" "5435" "5436" "5437" "5438" "5439" "5440" "5441"
## [121] "5471" "548644" "55276" "5557" "5558" "55703" "55811" "55821"
## [129] "5631" "5634" "56655" "56953" "56985" "57804" "58497" "6240"
## [137] "6241" "64425" "646625" "654364" "661" "7498" "8382" "84172"
## [145] "84265" "84284" "84618" "8622" "8654" "87178" "8833" "9060"
## [153] "9061" "93034" "953" "9533" "954" "955" "956" "957"
## [161] "9583" "9615"
```

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

```
##      1266      54855      1465      51232      2034      2317
## -2.422719  3.201955 -2.313738 -2.059631 -1.888019 -1.649792
```

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

```
attributes(keggres)
```

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

```
# Look at the first few down (less) pathways
head(keggres$less)
```

```
##                p.geomean stat.mean      p.val
## hsa04110 Cell cycle      8.995727e-06 -4.378644 8.995727e-06
## hsa03030 DNA replication  9.424076e-05 -3.951803 9.424076e-05
## hsa03013 RNA transport    1.375901e-03 -3.028500 1.375901e-03
## hsa03440 Homologous recombination 3.066756e-03 -2.852899 3.066756e-03
## hsa04114 Oocyte meiosis    3.784520e-03 -2.698128 3.784520e-03
## hsa00010 Glycolysis / Gluconeogenesis 8.961413e-03 -2.405398 8.961413e-03
##                q.val set.size      exp1
## hsa04110 Cell cycle      0.001448312      121 8.995727e-06
```

```
## hsa03030 DNA replication          0.007586381      36 9.424076e-05
## hsa03013 RNA transport            0.073840037     144 1.375901e-03
## hsa03440 Homologous recombination 0.121861535      28 3.066756e-03
## hsa04114 Oocyte meiosis           0.121861535     102 3.784520e-03
## hsa00010 Glycolysis / Gluconeogenesis 0.212222694     53 8.961413e-03
```

Each `keggresless` and `keggresgreater` object is data matrix with gene sets as rows sorted by p-value.

The top “less/down” pathways is “Cell cycle” with the KEGG pathway identifier `hsa04110`.

Now, let’s try out the `pathview()` function from the `pathview` package to make a pathway plot with our RNA-Seq expression results shown in color. To begin with let’s manually supply a `pathway.id` (namely the first part of the “`hsa04110` Cell cycle”) that we could see from the print out above.

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

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

```
## Info: Working in directory /Users/chloewelch/Desktop/BGGN213Projects/bggn213_github/class16
```

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

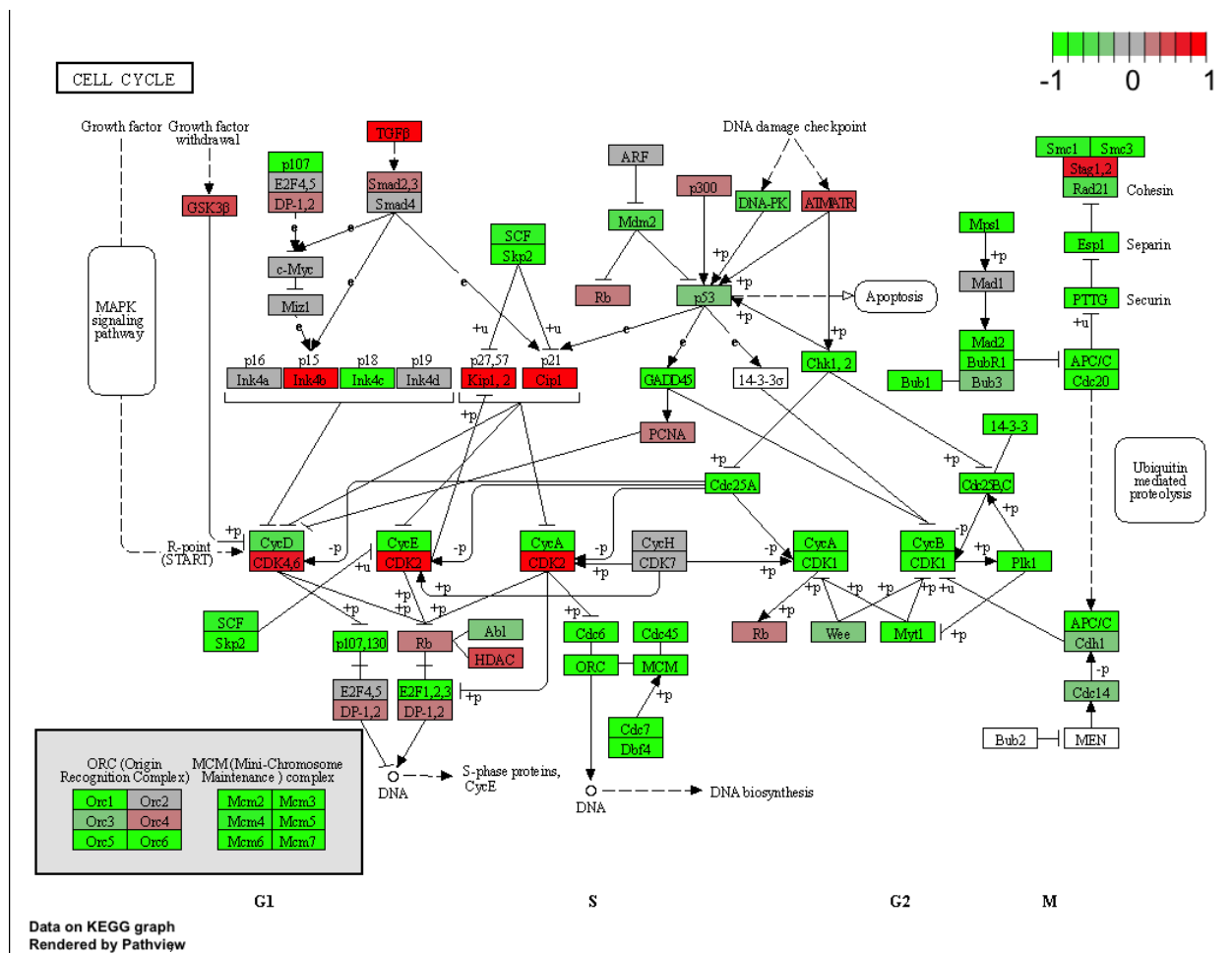
A different PDF based output of the same data

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

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

```
## Info: Working in directory /Users/chloewelch/Desktop/BGGN213Projects/bggn213_github/class16
```

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



Now, let's process our results a bit more to automatically pull out the top 5 upregulated pathways, then further process that just to get the pathway IDs needed by the `pathview()` function. We'll use these KEGG pathway IDs for pathview plotting below.

```
## Focus on top 5 upregulated pathways here for demo purposes only
keggrespathways <- rownames(keggres$greater)[1:5]
```

```
# Extract the 8 character long IDs part of each string
keggresids = substr(keggrespathways, start=1, stop=8)
keggresids
```

```
## [1] "hsa04640" "hsa04630" "hsa00140" "hsa04142" "hsa04330"
```

Finally, let's pass these IDs in `keggresids` to the `pathview()` function to draw plots for all the top 5 pathways.

```
pathview(gene.data=foldchanges, pathway.id=keggresids, species="hsa")
```

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

```
## Info: Working in directory /Users/chloewelch/Desktop/BGGN213Projects/bgg213_github/class16
```

```
## Info: Writing image file hsa04640.pathview.png
```

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

## Info: Working in directory /Users/chloewelch/Desktop/BGGN213Projects/bgg213_github/class16

## Info: Writing image file hsa04630.pathview.png

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

## Info: Working in directory /Users/chloewelch/Desktop/BGGN213Projects/bgg213_github/class16

## Info: Writing image file hsa00140.pathview.png

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

## Info: Working in directory /Users/chloewelch/Desktop/BGGN213Projects/bgg213_github/class16

## Info: Writing image file hsa04142.pathview.png

## Info: some node width is different from others, and hence adjusted!

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

## Info: Working in directory /Users/chloewelch/Desktop/BGGN213Projects/bgg213_github/class16

## Info: Writing image file hsa04330.pathview.png
```