class11R

Katelyn Brown

2/22/2022

Install BioConductor

```
# install.packages("BiocManager)
# BiocManager :: install()
# BiocManager :: install("DESeq2")
library(BiocManager)
library(DESeq2)
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges
```

```
## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

Upload the count data and meta files.

```
counts <- read.csv("https://bioboot.github.io/bimm143_W18/class-material/airway_scaledcounts.csv", row.
metadata <-  read.csv("https://bioboot.github.io/bimm143_W18/class-material/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
```

```
control <- metadata[metadata[,"dex"]=="control",]
control.counts <- counts[ ,control$id]
control.mean <- rowSums( control.counts )/4
head(control.mean)
```

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

Q1. How many genes are in this dataset?

There are 38694 genes in the dataset.

Q2. How many 'control' cell lines do we have?

There are 4 control cell lines.

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

You can make the above code more robust by using summary() instead of just finding the mean.

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 <- metadata[metadata[,"dex"]=="treated",]
treated.counts <- counts[ ,treated$id]
treated.mean <- rowSums( treated.counts )/4
head(treated.mean)
```

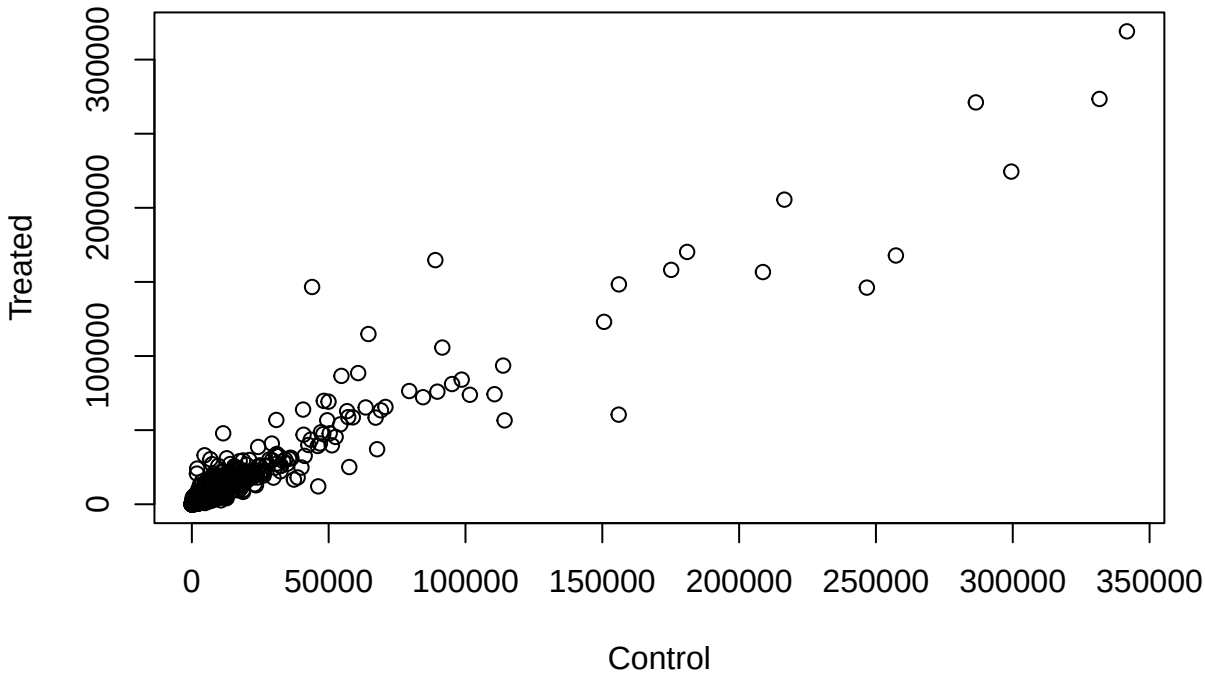
```
## ENSG00000000003 ENSG00000000005 ENSG00000000419 ENSG00000000457 ENSG00000000460
##          658.00            0.00          546.00          316.50           78.75
## ENSG00000000938
##            0.00
```

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

```
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
```

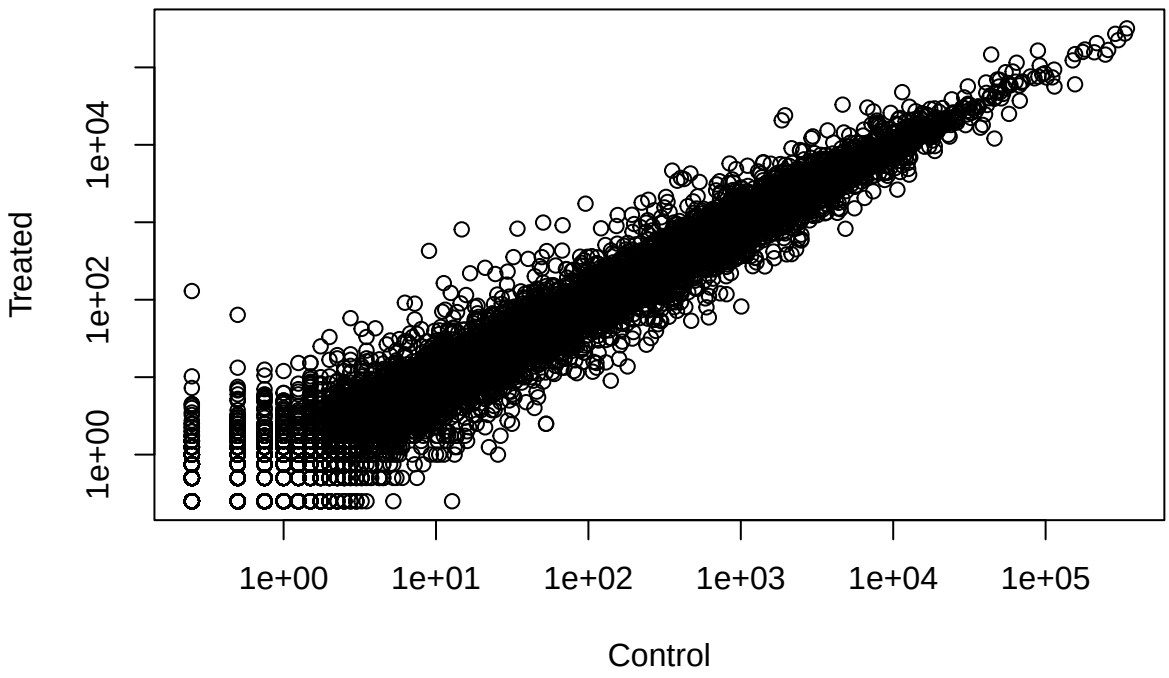
```
plot(meancounts[,1],meancounts[,2], xlab="Control", ylab="Treated")
```




```
plot(meancounts[,1],meancounts[,2], log = "xy", xlab="Control", ylab="Treated")
```

```
## 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
```




```
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
```

Q5b. You could also use the ggplot2 package to make this figure producing the plot below. What geom_?() function would you use for this plot?

You would use geom_point() to use ggplot.

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

You would use log() to plot the axes on a log scale.

```
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?

The purpose of the arr.ind argument is which array indices should be returned when x is an array.

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

> Q9. Using the down.ind vector above can you determine how many down regulated genes we have at the greater than 2 fc level?

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

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

```
count(down.ind)
```

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

There are 250 up regulated genes at a greater than 2 fc, and 367 down regulated genes at a greater than 2 fc level.

> Q10. Do you trust these results?

```
library(DESeq2)
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 <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

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

```
##
## out of 25258 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1563, 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)       : 1236, 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
```

Based on the summary of the results, I do trust these results and DESeq analysis.

Adding annotation data

To help interpret our results we need to understand what the differentially expressed genes are. The first step is to get the gene names (ie. gene SYMBOLs).

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

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

```
## '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
```

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), keytype = "ENSEMBL", column = "ENTREZID", mul
```

```
## '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 8 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
##                 <numeric> <character> <character>
## ENSG00000000003  0.163035      TSPAN6        7105
## ENSG00000000005        NA        TNMD       64102
## ENSG00000000419  0.176032        DPM1        8813
## ENSG00000000457  0.961694       SCYL3       57147
## ENSG00000000460  0.815849    C1orf112       55732
## ENSG00000000938        NA         FGR        2268
```

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

```
## '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 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     uniprot
##                 <numeric> <character> <character> <character>
## ENSG00000000003  0.163035      TSPAN6        7105  A0A024RCI0
## ENSG00000000005        NA        TNMD       64102      Q9H2S6
## ENSG00000000419  0.176032        DPM1        8813      O60762
## ENSG00000000457  0.961694       SCYL3       57147      Q8IZE3
## ENSG00000000460  0.815849    C1orf112       55732  A0A024R922
## ENSG00000000938        NA         FGR        2268      P09769
```

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

```
## '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     -0.3507030  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     uniprot
##                 <numeric> <character> <character> <character>
## ENSG00000000003  0.163035      TSPAN6        7105  A0A024RCI0
## ENSG00000000005        NA        TNMD       64102      Q9H2S6
## ENSG00000000419  0.176032        DPM1        8813      O60762
## ENSG00000000457  0.961694       SCYL3       57147      Q8IZE3
## ENSG00000000460  0.815849    C1orf112       55732  A0A024R922
## ENSG00000000938        NA         FGR        2268      P09769
##                            genename
##                         <character>
## ENSG00000000003          tetraspanin 6
## ENSG00000000005            tenomodulin
## ENSG00000000419 dolichyl-phosphate m..
## ENSG00000000457 SCY1 like pseudokina..
## ENSG00000000460 chromosome 1 open re..
## ENSG00000000938 FGR proto-oncogene, ..
```

Pathway Analysis

Install and load packages, then look at the first two pathways in KEGG.

```
# 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"
```

Need a vector of fold-change labeled with the names of genes in ENTREZ format.

```
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
```

Now we can run the GAGE analysis in the foldchange vector and the KEGG datasets we are interested in.

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

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

```
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
```

Now I can map the foldchange results onto a KEGG pathway. Do this manually by selecting a pathway ID from above.

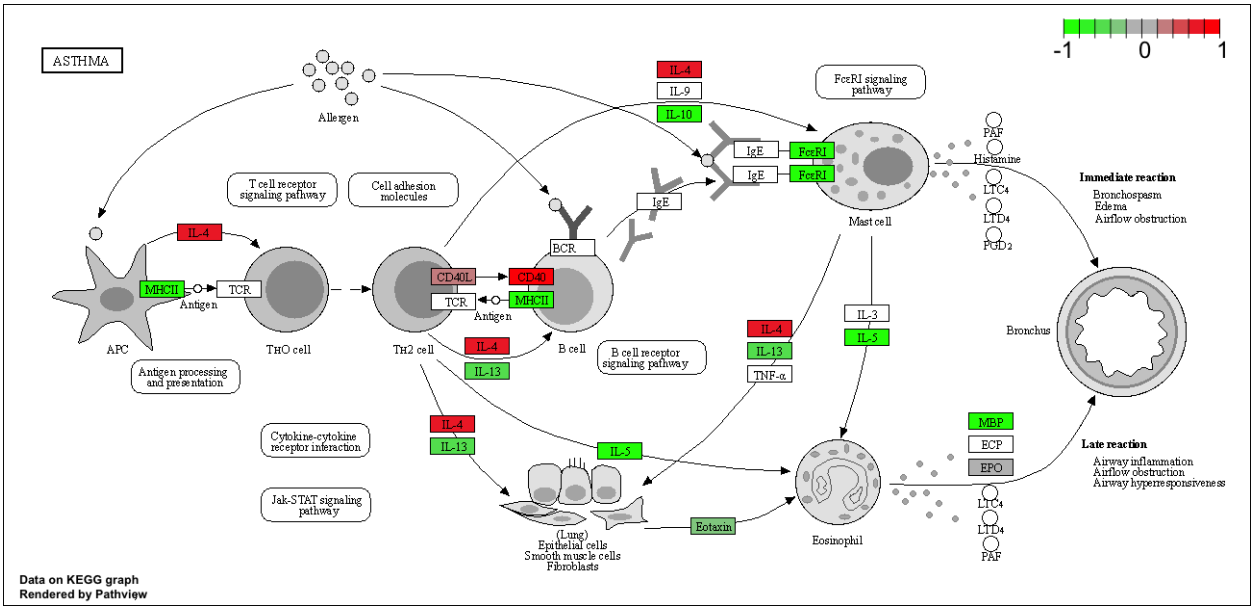
```
pathview(gene.data = foldchanges, pathway.id = "hsa05310")
```

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

```
## Info: Working in directory /Users/katybrown/Downloads/BIMM 143 R/class11
```

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

```
# Save results
write.csv(res, file = "deseqresults.csv")
```



Use a different 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/katybrown/Downloads/BIMM 143 R/class11
```

```
## 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?

```
head(keggres$greater, 2)
```

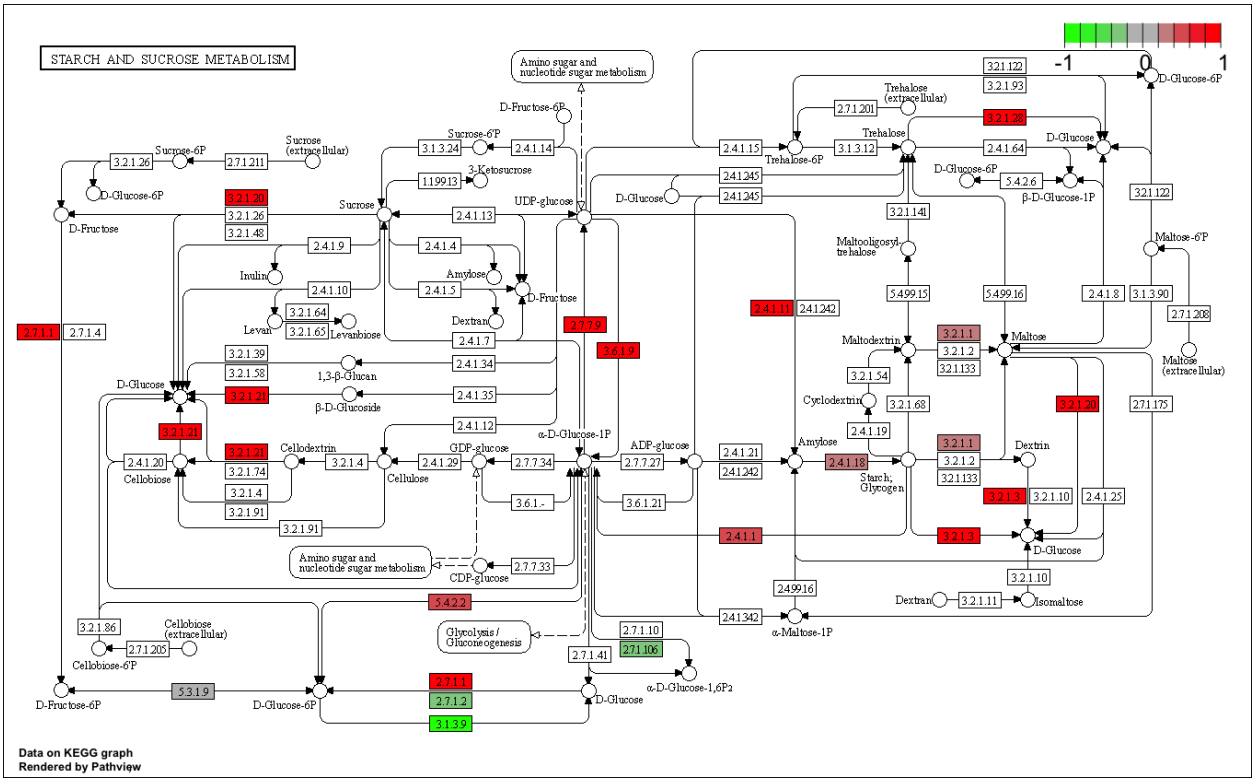
```
##                                      p.geomean stat.mean       p.val
## hsa00500 Starch and sucrose metabolism 0.003306262  2.772644 0.003306262
## hsa00330 Arginine and proline metabolism 0.012317455  2.280002 0.012317455
##                                            q.val set.size        exp1
## hsa00500 Starch and sucrose metabolism 0.7042337       52 0.003306262
## hsa00330 Arginine and proline metabolism 0.7774866       54 0.012317455
```

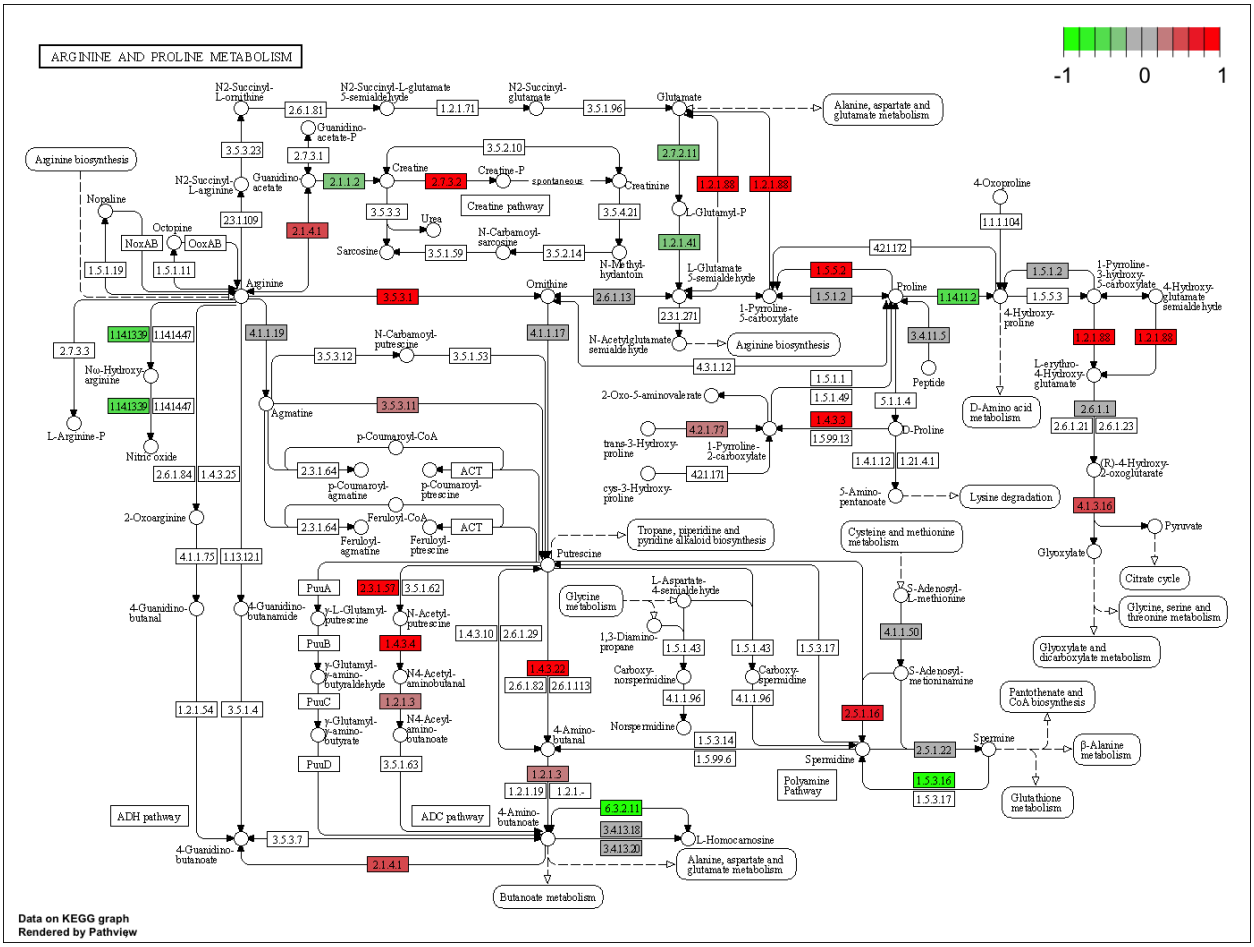
```
pathview(gene.data = foldchanges, pathway.id = "hsa00500")
```

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

## Info: Working in directory /Users/katybrown/Downloads/BIMM 143 R/class11

## Info: Writing image file hsa00500.pathview.png

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

```
pathview(gene.data = foldchanges, pathway.id = "hsa00330")
```

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

## Info: Working in directory /Users/katybrown/Downloads/BIMM 143 R/class11

## Info: Writing image file hsa00330.pathview.png
```




ARGININE AND PROLINE METABOLISM
-1
0
1
Arginine biosynthesis
Alanine, aspartate and glutamate metabolism
Glutamate
N2-Succinyl-L-ornithine
N2-Succinyl-L-glutamate 5-semialdehyde
N2-Succinyl-glutamate
Guanidino-acetate-P
Creatine
Creatine-P
spontaneous
Creatinine
Creatine pathway
L-Glutamyl-P
4-Oxoproline
Nopaline
Octopine
NoxAB
OoxAB
N2-Succinyl-L-arginine
Guanidino-acetate
Urea
Sarcosine
N-Carbamoyl-sarcosine
N-Methyl-hydantoin
L-Glutamate 5-semialdehyde
Arginine
Ornithine
Proline
1-Pyrroline-3-hydroxy-3-carboxylate
4-Hydroxy-glutamate semialdehyde
4-Hydroxy-proline
N-Acetylglutamate semialdehyde
Arginine biosynthesis
1-Pyrroline-5-carboxylate
Peptide
D-Amino acid metabolism
L-erythro-4-Hydroxy-glutamate
Nω-Hydroxy-arginine
L-Arginine-P
Nitric oxide
Agmatine
N-Carbamoyl-putrescine
2-Oxo-5-aminovalerate
trans-3-Hydroxy-proline
cys-3-Hydroxy-proline
D-Proline
5-Amino-pentanoate
Lysine degradation
(R)-4-Hydroxy-2-oxoglutarate
p-Coumaroyl-CoA
p-Coumaroyl-agmatine
p-Coumaroyl-ptrescine
ACT
Feruloyl-CoA
Feruloyl-agmatine
Feruloyl-ptrescine
2-Oxoarginine
Tropane, piperidine and pyridine alkaloid biosynthesis
Putrescine
Cysteine and methionine metabolism
Pyruvate
Glyoxylate
Citrate cycle
Glycine, serine and threonine metabolism
PuuA
PuuB
PuuC
PuuD
4-Guanidino-butanal
4-Guanidino-butanamide
γ-L-Glutamyl-putrescine
N-Acetyl-putrescine
γ-Glutamyl-amino-butyraldehyde
N4-Acetyl-aminobutanal
γ-Glutamyl-amino-butyrate
N4-Acetyl-amino-butanoate
Glycine metabolism
L-Aspartate-4-semialdehyde
1,3-Diamino-propane
Carboxy-norspermidine
Carboxy-spermidine
S-Adenosyl-L-methionine
S-Adenosyl-methioninamine
Glyoxylate and dicarboxylate metabolism
Pantothenate and CoA biosynthesis
4-Amino-butanal
Norspermidine
Spermidine
Spermine
β-Alanine metabolism
Polyamine Pathway
Glutathione metabolism
ADH pathway
ADC pathway
4-Amino-butanoate
4-Guanidino-butanoate
L-Homocarnosine
Alanine, aspartate and glutamate metabolism
Butanoate metabolism
Data on KEGG graph
Rendered by Pathview