

BIMM 143 Class 15

Anika Bhattacharjya (A15459876)

Bioconductor and DESeq2 setup

```
#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

```

Import countData and colData

```

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

head(counts)

##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003      723       486       904       445      1170
## ENSG00000000005        0         0         0         0         0
## ENSG00000000419      467       523       616       371      582
## ENSG00000000457      347       258       364       237      318
## ENSG00000000460       96        81        73        66      118
## ENSG00000000938        0         0         1         0         2
##          SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003     1097       806       604

```

```

## ENSG00000000005      0      0      0
## ENSG00000000419    781    417   509
## ENSG00000000457    447    330   324
## ENSG00000000460     94    102    74
## ENSG00000000938      0      0      0

head(metadata)

##           id      dex celltype    geo_id
## 1 SRR1039508 control  N61311 GSM1275862
## 2 SRR1039509 treated  N61311 GSM1275863
## 3 SRR1039512 control  N052611 GSM1275866
## 4 SRR1039513 treated  N052611 GSM1275867
## 5 SRR1039516 control  N080611 GSM1275870
## 6 SRR1039517 treated  N080611 GSM1275871

```

Q1. How many genes are in this dataset?

```
nrow(counts)
```

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

38694 genes

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

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

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

4 control cell lines

Toy differential gene expression

Need to extract all the ‘control’ columns before taking the row-wise mean to get the average count values for all genes in these 4 experiments:

```

control <- metadata[metadata[, "dex"] == "control",]
control.counts <- counts[, control$id]
control.mean <- rowMeans(control.counts)
head(control.mean)

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

```

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

Get rid of the ‘/4’ and just take rowMeans because the number of data points will change over time and may not always be 4. In class we used rowMeans already so I never used the “control.mean <- rowSums(control.counts)/4” code.

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)

Finding ‘treated’ mean:

```

treated <- metadata[metadata[, "dex"] == "treated",]
treated.counts <- counts[, treated$id]

```

```

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

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

Combine meancount data for bookkeeping purposes
meancounts <- data.frame(control.mean, treated.mean)

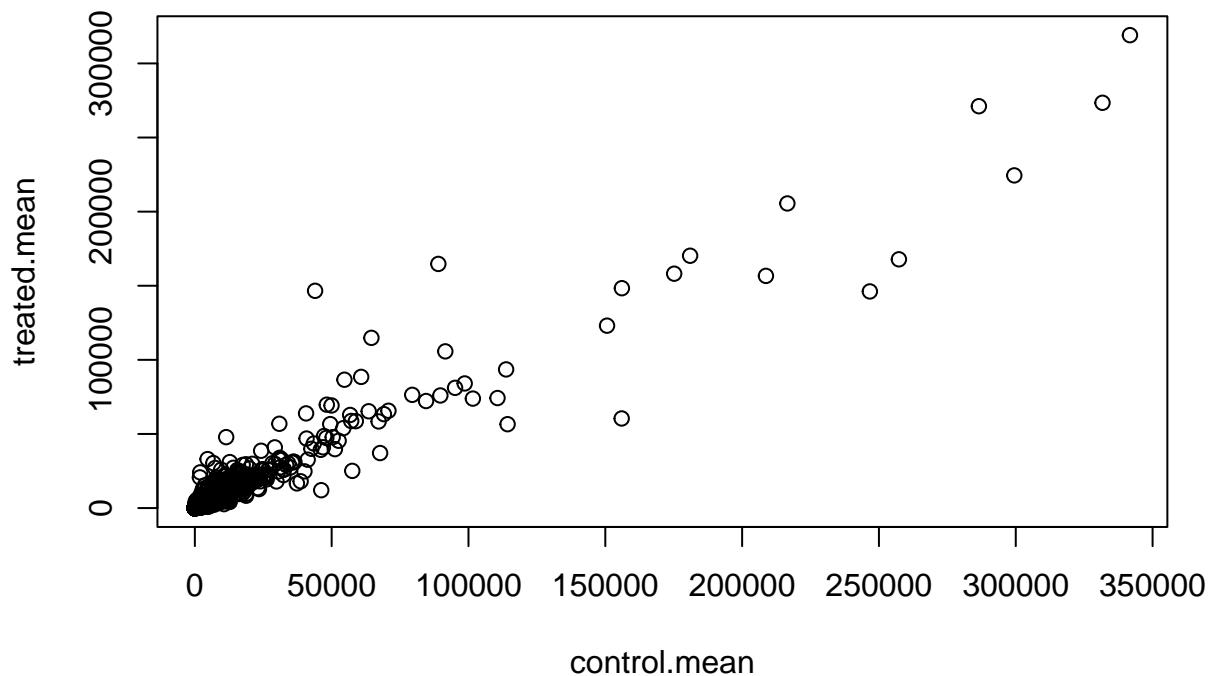
# Sum of mean for all counts
colSums(meancounts)

## control.mean treated.mean
##      23005324     22196524

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

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?

You would use geom_point.

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

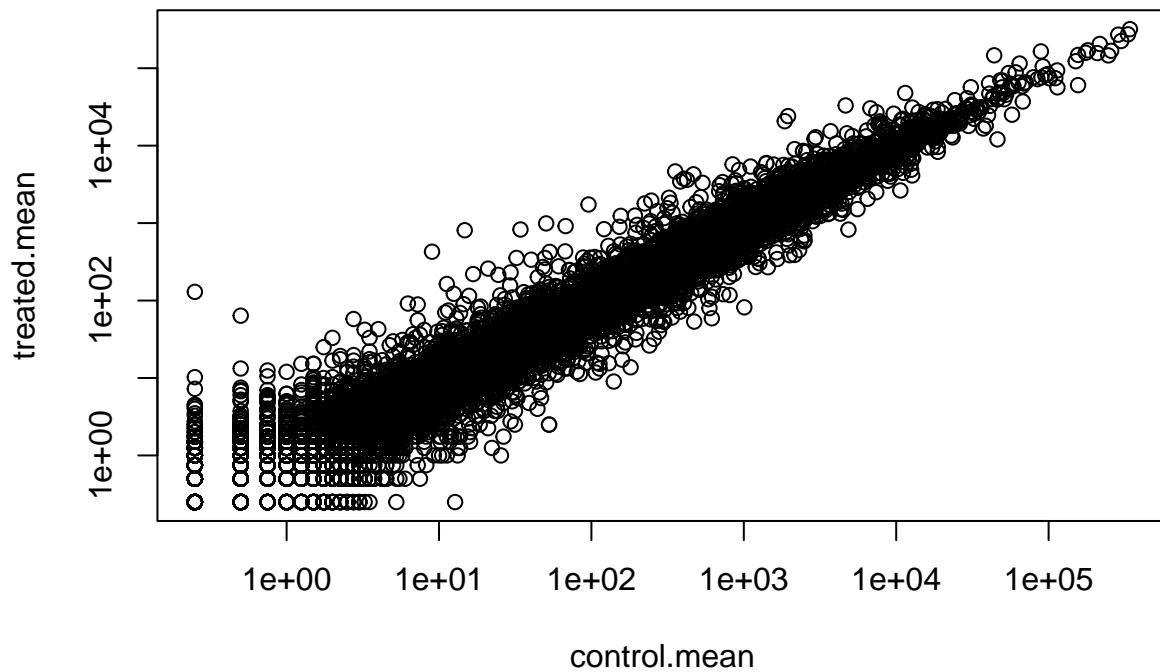
```

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 more straightforward.

```

log2(40/10)

## [1] 2

log2(10/10)

## [1] 0

log2(5/10)

## [1] -1

```

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

Let's add `log2(fold-change)` values to `meancounts`

```

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
## 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

The NaN result happens when you divide by zero and try to take the log. The -Inf result is given when you try to take the log of zero.

We need to exclude the genes with zero counts because we can't say anything about them.

#meancounts[,1:2] == 0

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

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

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

## [1] 2 6 65 70 73 81

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

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?

array.ind tells you which values are returned when the variable equals an specific array, in this case which values are TRUE. unique makes sure we don't count any values twice.

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

up.ind <- mycounts$log2fc > 2
which(up.ind, arr.ind = TRUE)

```

```

## [1] 59 149 203 290 338 344 576 1048 1055 1196 1618 1680
## [13] 1781 1939 1981 2038 2050 2118 2355 2358 2443 2514 2528 2599
## [25] 2686 3186 3265 3459 3548 3788 3836 4052 4128 4129 4244 4453
## [37] 4567 4659 4899 5124 5180 5181 5290 5386 5423 5484 5496 5498
## [49] 5584 5725 5964 6055 6167 6246 6277 6288 6364 6453 6518 6553
## [61] 6636 6869 6959 7034 7107 7108 7115 7183 7306 7348 7369 7422
## [73] 7617 7790 7913 7970 7989 8330 8365 8396 8530 8706 8714 8723
## [85] 8729 8741 8934 9175 9176 9213 9215 9270 9297 9392 9608 9753
## [97] 9828 9924 9940 9968 9971 10079 10146 10174 10229 10234 10515 10577
## [109] 10631 10790 10955 10981 11019 11143 11177 11215 11341 11497 11520 11650
## [121] 11798 11899 12087 12153 12237 12254 12269 12365 12637 12742 12828 12894
## [133] 12922 12952 13071 13363 13426 13604 13813 13822 13832 13833 13851 13938
## [145] 14099 14140 14151 14419 14472 14493 14519 14700 14820 14908 15111 15511
## [157] 15526 15564 15567 15675 15766 15807 16310 16341 16361 16365 16415 16416
## [169] 16421 16494 16630 16631 16638 16725 16929 16996 17115 17132 17206 17211
## [181] 17285 17318 17332 17338 17357 17385 17485 17530 17550 17779 17918 17996
## [193] 18061 18120 18163 18186 18246 18456 18493 18576 18650 18744 18839 18914
## [205] 18960 19045 19052 19078 19129 19139 19150 19296 19297 19462 19530 19550
## [217] 19666 19803 19915 20073 20183 20214 20314 20374 20384 20432 20445 20465
## [229] 20553 20668 20766 20771 20774 20984 21093 21113 21119 21189 21190 21216
## [241] 21266 21283 21317 21321 21391 21403 21409 21454 21544 21603

```

250 genes

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

```

down.ind <- mycounts$log2fc < (-2)
which(down.ind, arr.ind = TRUE)

## [1] 341 365 418 443 451 623 624 662 684 825 925 1124
## [13] 1142 1234 1238 1315 1351 1518 1551 1576 1623 1887 1911 1928
## [25] 1965 1974 2131 2245 2342 2375 2670 3015 3020 3091 3279 3320
## [37] 3332 3334 3355 3419 3794 3899 4001 4108 4286 4288 4331 4348
## [49] 4504 4605 4906 4933 4984 5009 5015 5053 5067 5154 5282 5320
## [61] 5397 5501 5518 5525 5535 5550 5555 5608 5684 5689 5724 5749
## [73] 5756 5779 5811 5873 5890 5904 5950 6073 6116 6293 6310 6370
## [85] 6504 6533 6684 6766 7000 7150 7189 7332 7396 7438 7521 7609
## [97] 7627 7858 7869 7936 8003 8013 8049 8052 8081 8114 8210 8212
## [109] 8237 8407 8492 8501 8653 8667 8762 8875 9043 9189 9199 9274
## [121] 9457 9600 9628 9807 9843 9853 9859 9983 9998 10020 10040 10176
## [133] 10272 10361 10371 10487 10495 10526 10569 10586 10779 11138 11147 11159
## [145] 11204 11237 11243 11332 11345 11417 11424 11436 11642 11710 11769 11848
## [157] 12032 12240 12290 12395 12456 12466 12520 12549 12566 12618 12640 12671
## [169] 12678 12687 12741 12826 12915 12964 13027 13193 13197 13211 13230 13324
## [181] 13340 13348 13408 13410 13451 13464 13473 13484 13627 13677 13682 13812
## [193] 13991 14033 14207 14226 14239 14285 14336 14350 14380 14406 14544 14628
## [205] 14666 14720 14898 15147 15275 15282 15325 15388 15406 15420 15492 15655
## [217] 15677 15731 15759 15787 15789 15840 15981 16063 16144 16234 16266 16283
## [229] 16318 16343 16371 16395 16500 16502 16662 16709 16860 16876 16932 16949
## [241] 16976 17024 17043 17069 17121 17162 17168 17196 17259 17326 17327 17344
## [253] 17488 17543 17546 17587 17652 17667 17682 17709 17745 17808 17931 17932
## [265] 17944 17957 18018 18051 18059 18197 18210 18299 18482 18490 18519 18521
## [277] 18546 18578 18582 18618 18652 18763 18785 18812 18834 18841 18922 18968
## [289] 19064 19094 19114 19187 19228 19243 19246 19270 19302 19311 19338 19347
## [301] 19363 19443 19499 19551 19554 19613 19616 19654 19669 19801 19813 19814

```

```

## [313] 19830 19890 19959 19976 20018 20074 20078 20096 20114 20128 20138 20233
## [325] 20242 20254 20258 20270 20273 20277 20390 20404 20413 20434 20480 20483
## [337] 20543 20559 20561 20599 20624 20631 20698 20709 20809 20813 20821 20894
## [349] 20969 21009 21084 21118 21148 21154 21188 21249 21280 21335 21371 21416
## [361] 21621 21627 21669 21673 21713 21744 21764

```

367 genes

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

Not really because fold change could be large without being significant. We need to normalize our data first before we can see any real trends.

DESeq2 analysis

```

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

## 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

```

Getting results:

```
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)
res

## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 38694 rows and 6 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
##          <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG00000000003   747.1942    -0.3507030  0.168246 -2.084470  0.0371175
## ENSG00000000005    0.0000       NA        NA        NA        NA
## ENSG00000000419   520.1342    0.2061078  0.101059  2.039475  0.0414026
## ENSG00000000457   322.6648    0.0245269  0.145145  0.168982  0.8658106
## ENSG00000000460    87.6826   -0.1471420  0.257007 -0.572521  0.5669691
## ...
##           ...       ...       ...       ...       ...
## ENSG0000283115   0.000000       NA        NA        NA        NA
## ENSG0000283116   0.000000       NA        NA        NA        NA
## ENSG0000283119   0.000000       NA        NA        NA        NA
## ENSG0000283120   0.974916   -0.668258  1.69456  -0.394354  0.693319
## ENSG0000283123   0.000000       NA        NA        NA        NA
##           padj
##          <numeric>
## ENSG0000000003   0.163035
## ENSG00000000005    NA
## ENSG00000000419   0.176032
## ENSG00000000457   0.961694
## ENSG00000000460   0.815849
## ...
##           ...
## ENSG0000283115    NA
## ENSG0000283116    NA
## ENSG0000283119    NA
## ENSG0000283120    NA
## ENSG0000283123    NA
```

Summarize results

```
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

Customize results table Focus in on genes with a small p-value (significant change)

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

```

Save our results

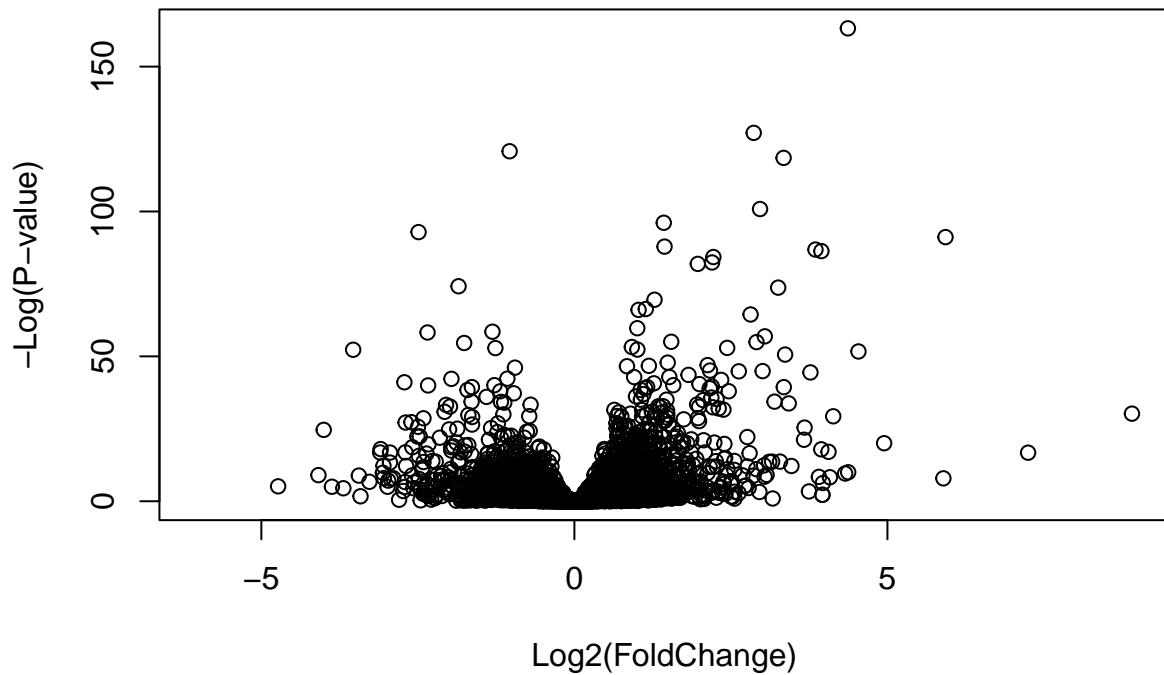
Write out whole results data set (including genes that don't change significantly)

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

Data Visualization

Volcano plots

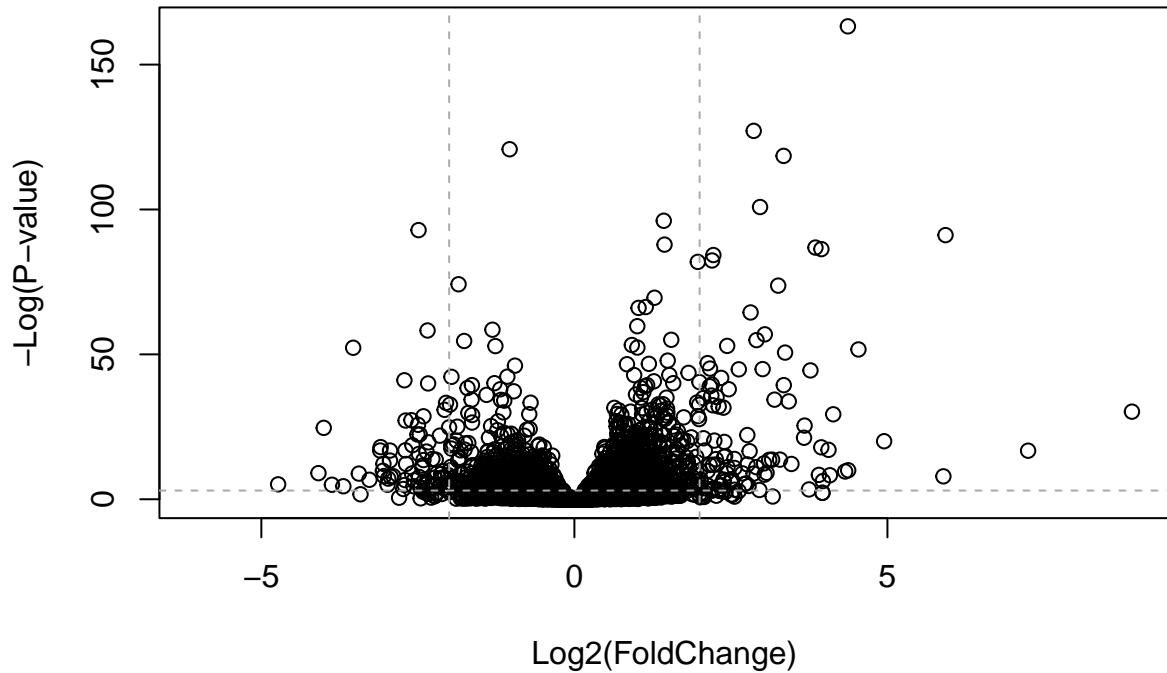
```
plot( res$log2FoldChange, -log(res$padj),
      xlab="Log2(FoldChange)",
      ylab="-Log(P-value)")
```



Make the plots more useful

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

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



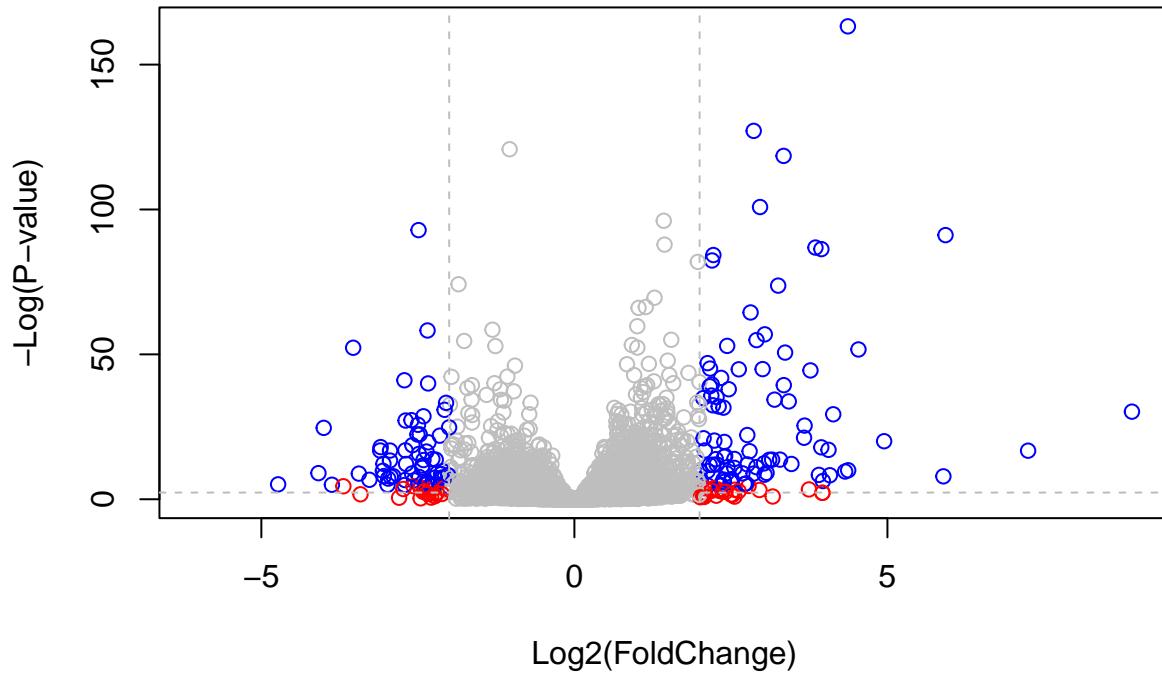
Set up custom color vector

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

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

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

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



Add Annotation data

We need two bioconductor packages

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

##
```

	ACCNUM	ALIAS	ENSEMBL	ENSEMLPROT	ENSEMLTRANS
[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				

We will use `mapIDs` function to translate between identifiers from different data bases

```
res$symbol <- mapIDs(org.Hs.eg.db,
                      keys=row.names(res), # Our gene names
                      keytype="ENSEMBL",      # The format of our gene names
                      column="SYMBOL",        # The new format we want to add
                      multiVals="first")

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

```

head(res)

## log2 fold change (MLE): dex treated vs control
## Wald test p-value: dex treated vs control
## DataFrame with 6 rows and 7 columns
##           baseMean log2FoldChange      lfcSE      stat     pvalue
## <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG000000000003 747.194195 -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005  0.000000      NA       NA       NA       NA
## ENSG000000000419 520.134160  0.2061078  0.101059  2.039475 0.0414026
## ENSG000000000457 322.664844  0.0245269  0.145145  0.168982 0.8658106
## ENSG000000000460 87.682625 -0.1471420  0.257007 -0.572521 0.5669691
## ENSG000000000938 0.319167 -1.7322890  3.493601 -0.495846 0.6200029
##           padj      symbol
## <numeric> <character>
## ENSG000000000003 0.163035    TSPAN6
## ENSG000000000005      NA      TNMD
## ENSG000000000419 0.176032    DPM1
## ENSG000000000457 0.961694    SCYL3
## ENSG000000000460 0.815849    C1orf112
## ENSG000000000938      NA      FGR

```

Enhanced Volcano Plots

Download didn't work, Professor Grant said to skip this section.

Pathway Analysis

```

library(pathview)

## #####
## Pathview is an open source software package distributed under GNU General
## Public License version 3 (GPLv3). Details of GPLv3 is available at
## http://www.gnu.org/licenses/gpl-3.0.html. Particullary, users are required to
## formally cite the original Pathview paper (not just mention it) in publications
## or products. For details, do citation("pathview") within R.
##
## The pathview downloads and uses KEGG data. Non-academic uses may require a KEGG
## license agreement (details at http://www.kegg.jp/kegg/legal.html).
## #####
library(gage)

##
library(gageData)

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

```

```

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>
## ENSG000000000003 747.194195     -0.3507030  0.168246 -2.084470 0.0371175
## ENSG000000000005  0.000000        NA       NA       NA       NA
## ENSG000000000419 520.134160      0.2061078  0.101059  2.039475 0.0414026
## ENSG000000000457 322.664844      0.0245269  0.145145  0.168982 0.8658106
## ENSG000000000460 87.682625      -0.1471420  0.257007 -0.572521 0.5669691
## ENSG000000000938 0.319167      -1.7322890  3.493601 -0.495846 0.6200029
##           padj      symbol      entrez
##           <numeric> <character> <character>
## ENSG000000000003 0.163035      TSPAN6      7105
## ENSG000000000005        NA      TNMD      64102
## ENSG000000000419 0.176032      DPM1      8813
## ENSG000000000457 0.961694      SCYL3      57147
## ENSG000000000460 0.815849      C1orf112    55732
## ENSG000000000938        NA      FGR      2268

```

Load Kegg data set

```
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

```
foldchange <- res$log2FoldChange
names(foldchange) <- res$entrez
```

```
head(foldchange)
```

```

##      7105      64102      8813      57147      55732      2268
## -0.35070302        NA 0.20610777 0.02452695 -0.14714205 -1.73228897

```

Get the results

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

```
attributes(keggres)
```

```
## $names
```

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

This separates results by “greater” and “less”

```
# 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=foldchange, pathway.id="hsa05310")
```

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

Info: Working in directory /home/a6bhattacharya/R/bimm143_github/BIMM_143_Class_15

Info: Writing image file hsa05310.pathview.png

