class 14: RNASeq mini project

Eli Sobel 69027989

```
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)
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, aperm, 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, saveRDS, setdiff,
    table, tapply, union, unique, unsplit, which.max, which.min


Attaching package: 'S4Vectors'

The following object is masked from 'package:utils':

    findMatches

The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges


Attaching package: 'IRanges'

The following object is masked from 'package:grDevices':

    windows

Loading required package: GenomicRanges

Loading required package: GenomeInfoDb
```

```
Loading required package: SummarizedExperiment

Loading required package: MatrixGenerics

Loading required package: matrixStats

Warning: package 'matrixStats' was built under R version 4.4.2


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
```

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

```
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"
```

import data

We need two things - "Counts" and "Metadata" (what DESeq calls colData - as it describes the columns in Counts).

```
counts <- read.csv("GSE37704_featurecounts.csv", row.names=1)
metadata <- read.csv("GSE37704_metadata.csv")
head(counts)
```

```
                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
```

```
head(metadata)
```

```
         id     condition
1 SRR493366 control_sirna
2 SRR493367 control_sirna
3 SRR493368 control_sirna
4 SRR493369      hoxa1_kd
5 SRR493370      hoxa1_kd
6 SRR493371      hoxa1_kd
```

We want the columns in `counts` to match the rows in the `metadata`.

```
colnames(counts)
```

```
[1] "length"    "SRR493366" "SRR493367" "SRR493368" "SRR493369" "SRR493370"
[7] "SRR493371"
```

```
metadata$id
```

```
[1] "SRR493366" "SRR493367" "SRR493368" "SRR493369" "SRR493370" "SRR493371"
```

We can get rid of the first column in `counts` to make these match.

```
countData <- counts[,-1]
head(countData)
```

```
                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
```

```
colnames(countData)
```

```
[1] "SRR493366" "SRR493367" "SRR493368" "SRR493369" "SRR493370" "SRR493371"
```

```
metadata$id
```

```
[1] "SRR493366" "SRR493367" "SRR493368" "SRR493369" "SRR493370" "SRR493371"
```

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

```
[1] TRUE
```

```
all(c(T,T,T,T))
```

```
[1] TRUE
```

```
x <- c(T,T,T)
if(all(x)) { cat("Me happy")} else {cat("Me no happy")}
```

```
Me happy
```

```
head(counts)
```

```
                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
```

Filter out zero counts

It is standard practice to remove any genes/transcripts that we have no data for - i.e. zero counts in all columns.

```
to.keep.inds <- rowSums(countData) > 0
cleanCounts <- countData[to.keep.inds,]
head(cleanCounts)
```

```
                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
```

Setup for DESetup

```
library(DESeq2)
```

```
dds <- DESeqDataSetFromMatrix(countData = cleanCounts, colData = metadata, design = ~conditi
```

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

DESeq

```
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)
```

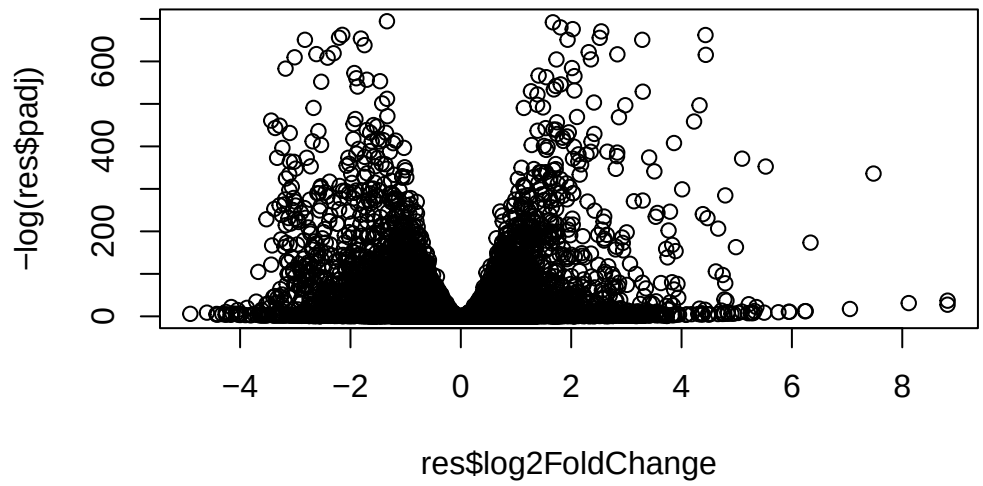
Inspect results

```
head(res)
```

```
log2 fold change (MLE): condition hoxa1 kd vs control sirna
Wald test p-value: condition hoxa1 kd vs control sirna
DataFrame with 6 rows and 6 columns
                 baseMean log2FoldChange     lfcSE       stat      pvalue
                <numeric>      <numeric> <numeric>  <numeric>   <numeric>
ENSG00000279457   29.9136      0.1792571 0.3248216   0.551863 5.81042e-01
ENSG00000187634  183.2296      0.4264571 0.1402658   3.040350 2.36304e-03
ENSG00000188976 1651.1881     -0.6927205 0.0548465 -12.630158 1.43990e-36
ENSG00000187961  209.6379      0.7297556 0.1318599   5.534326 3.12428e-08
ENSG00000187583   47.2551      0.0405765 0.2718928   0.149237 8.81366e-01
ENSG00000187642   11.9798      0.5428105 0.5215598   1.040744 2.97994e-01
                      padj
                 <numeric>
ENSG00000279457 6.86555e-01
ENSG00000187634 5.15718e-03
ENSG00000188976 1.76549e-35
ENSG00000187961 1.13413e-07
ENSG00000187583 9.19031e-01
ENSG00000187642 4.03379e-01
```

Data Viz

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



Pathway Analysis

```
head(res)
```

```
log2 fold change (MLE): condition hoxa1 kd vs control sirna
Wald test p-value: condition hoxa1 kd vs control sirna
DataFrame with 6 rows and 6 columns
                baseMean log2FoldChange     lfcSE       stat      pvalue
               <numeric>      <numeric> <numeric>  <numeric>   <numeric>
ENSG00000279457   29.9136      0.1792571 0.3248216   0.551863 5.81042e-01
ENSG00000187634  183.2296      0.4264571 0.1402658   3.040350 2.36304e-03
ENSG00000188976 1651.1881     -0.6927205 0.0548465 -12.630158 1.43990e-36
ENSG00000187961  209.6379      0.7297556 0.1318599   5.534326 3.12428e-08
ENSG00000187583   47.2551      0.0405765 0.2718928   0.149237 8.81366e-01
ENSG00000187642   11.9798      0.5428105 0.5215598   1.040744 2.97994e-01
                      padj
                 <numeric>
ENSG00000279457 6.86555e-01
ENSG00000187634 5.15718e-03
ENSG00000188976 1.76549e-35
ENSG00000187961 1.13413e-07
```

```
ENSG00000187583 9.19031e-01
ENSG00000187642 4.03379e-01
```

Annotation of genes

First I need to translate my Ensemble IDs in my `res` object to Entrez and gene symbol formats.

For this I will use the AnnotationDbi package and its `mapIds()` function.

Let's map to "SYMBOL", "ENTREZID", "GENENAME" from our "ENSEMBLE" ids.

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

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

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

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

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

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

```
colnames(res)
```

```
[1] "baseMean"       "log2FoldChange" "lfcSE"          "stat"
[5] "pvalue"         "padj"           "genename"       "ENTREZID"
[9] "symbol"
```

```
head(res)
```

```
log2 fold change (MLE): condition hoxa1 kd vs control sirna
Wald test p-value: condition hoxa1 kd vs control sirna
DataFrame with 6 rows and 9 columns
                 baseMean log2FoldChange     lfcSE       stat      pvalue
                <numeric>      <numeric> <numeric>  <numeric>   <numeric>
ENSG00000279457   29.9136      0.1792571 0.3248216   0.551863 5.81042e-01
ENSG00000187634  183.2296      0.4264571 0.1402658   3.040350 2.36304e-03
ENSG00000188976 1651.1881     -0.6927205 0.0548465 -12.630158 1.43990e-36
ENSG00000187961  209.6379      0.7297556 0.1318599   5.534326 3.12428e-08
ENSG00000187583   47.2551      0.0405765 0.2718928   0.149237 8.81366e-01
ENSG00000187642   11.9798      0.5428105 0.5215598   1.040744 2.97994e-01
                      padj               genename    ENTREZID      symbol
                 <numeric>            <character> <character> <character>
ENSG00000279457 6.86555e-01                     NA          NA          NA
ENSG00000187634 5.15718e-03 sterile alpha motif ..      148398      SAMD11
ENSG00000188976 1.76549e-35 NOC2 like nucleolar ..       26155       NOC2L
ENSG00000187961 1.13413e-07 kelch like family me..      339451      KLHL17
ENSG00000187583 9.19031e-01 pleckstrin homology ..       84069     PLEKHN1
ENSG00000187642 4.03379e-01 PPARGC1 and ESRR ind..       84808       PERM1
```

Before going any further let's focus in on a subset of "top" hits.

We can use as a starting point log2FC of +2/-2 and a adjusted p-value of 0.05.

```
top.inds <- (abs(res$log2FoldChange) > 2) & (res$padj < 0.05)
top.inds[is.na(top.inds)] <- FALSE
head(top.inds, 20)
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
c(F,T,T,F) &
c(T,T,F,NA)
```

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

Let's save our "top genes" to a CSV

```
top.genes <- res[top.inds,]
write.csv(top.genes, file="top_geneset.csv")
```

Now we can do some pathway analysis

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

The **gage** function wants a vetor of importance as input with gene names as labels - KEGG speaks ENTREZ

```
foldchanges <- res$log2FoldChange
names(foldchanges) <- res$entrez
head(foldchanges)
```

```
[1]  0.17925708  0.42645712 -0.69272046  0.72975561  0.04057653  0.54281049
```

Run gage with these values

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

```
attributes(keggres)
```

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

```
head(keggres$less)
```

```
                                      p.geomean stat.mean p.val q.val
hsa00232 Caffeine metabolism                 NA       NaN    NA    NA
hsa00983 Drug metabolism - other enzymes     NA       NaN    NA    NA
hsa00230 Purine metabolism                   NA       NaN    NA    NA
hsa04514 Cell adhesion molecules (CAMs)      NA       NaN    NA    NA
hsa04010 MAPK signaling pathway              NA       NaN    NA    NA
hsa04012 ErbB signaling pathway              NA       NaN    NA    NA
                                      set.size exp1
hsa00232 Caffeine metabolism                 0   NA
hsa00983 Drug metabolism - other enzymes     0   NA
```

```
hsa00230 Purine metabolism                      0   NA
hsa04514 Cell adhesion molecules (CAMs)         0   NA
hsa04010 MAPK signaling pathway                 0   NA
hsa04012 ErbB signaling pathway                 0   NA
```

hsa04110

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

```
Warning: None of the genes or compounds mapped to the pathway!
Argument gene.idtype or cpd.idtype may be wrong.

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

Info: Working in directory C:/Users/eliso/Desktop/Bioinformatics/class14

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

```
data(go.sets.hs)
data(go.subs.hs)

# Focus on Biological Process subset of GO
gobpsets = go.sets.hs[go.subs.hs$BP]

gores = gage(foldchanges, gsets=gobpsets)
```

```
head(gores$less)
```

```
                                          p.geomean stat.mean p.val q.val
GO:0000002 mitochondrial genome maintenance        NA       NaN    NA    NA
GO:0000003 reproduction                            NA       NaN    NA    NA
GO:0000012 single strand break repair              NA       NaN    NA    NA
GO:0000018 regulation of DNA recombination         NA       NaN    NA    NA
GO:0000019 regulation of mitotic recombination     NA       NaN    NA    NA
GO:0000022 mitotic spindle elongation              NA       NaN    NA    NA
                                          set.size exp1
GO:0000002 mitochondrial genome maintenance        0   NA
GO:0000003 reproduction                            0   NA
GO:0000012 single strand break repair              0   NA
GO:0000018 regulation of DNA recombination         0   NA
GO:0000019 regulation of mitotic recombination     0   NA
GO:0000022 mitotic spindle elongation              0   NA
```

To run reactome online, we need to make a little text file where we have one gene id per line.

```
sig_genes <- res[res$padj <= 0.05 & !is.na(res$padj), "symbol"]
print(paste("Total number of significant genes:", length(sig_genes)))
```

```
[1] "Total number of significant genes: 8147"
```

```
write.table(sig_genes, file="significant_genes.txt", row.names=FALSE, col.names=FALSE, quote=
```

```
head(sig_genes)
```

```
ENSG00000187634 ENSG00000188976 ENSG00000187961 ENSG00000188290 ENSG00000187608
       "SAMD11"          "NOC2L"         "KLHL17"           "HES4"          "ISG15"
ENSG00000188157
         "AGRN"
```