In R, when you use row.names=1 as an argument in functions like read.csv or read.tables, it tells R to treat the forst column of the input data as arow names rather than as a regular data column.

Class 13: RNAseq mini project

Neha (A17567541)

In today's class, we will explore and analyze data from a published RNA-seq experiment where airway smooth muscle cells were treated with dexamethasone, a synthetic glucocorticoid steroid with anti-inflammatory effects (Himes et al. 2014).

Importing the data:

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

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

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

```
[1] 4
```

```
#This gives us a logical output, which we can take the sum of to get the true values.
colnames(counts)
```

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

Toy differential gene expression:

First, we have to check whether the metadata id column matches the columns in our count-data.

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

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

```
colnames(counts)
```

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

To check that all elements of a vector are TRUE we can use the `all()` function.

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

```
[1] TRUE
```

To start I will calculate the `control`.mean and `treated`.mean values and compare them.

-Identify and extract the `control` only columns -Determine the mean value for each gene (ie. row) -Do the same for `treated`

```
control.inds<-metadata$dex=="control"
control.counts<-counts[,control.inds]
control.mean<-apply(control.counts,1,mean)

treated.inds<-metadata$dex=="treated"
treated.counts<-counts[,treated.inds]
treated.mean<-apply(treated.counts,1,mean)

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

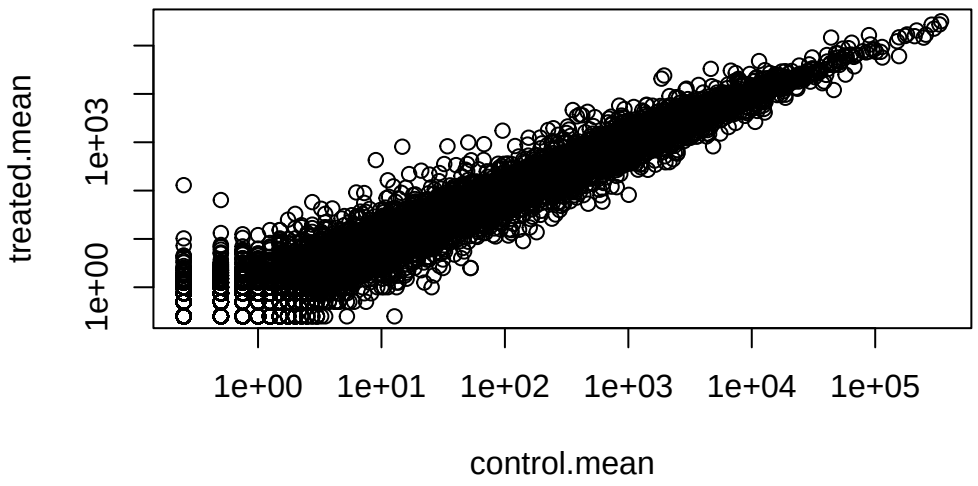
Have a quick view of this data:

We take the log function because it is very heavily skewed and over a really wide range. Log functions are the inverse of exponents, and they are used to simplify a lot of calculations.

```
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
```



I want to compare the treated and the control values here and we will use fold change in log2 units to do this. log2(Treated/Control)

```
log2fc<-log2(meancounts$treated.mean/meancounts$control.mean)
meancounts$log2fc<-log2fc       adding a column to an existing data frame
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
```

```
20/10
```

```
[1] 2
```

```
log2(20/10)
```

```
[1] 1
```

```
#This means that there is some difference, doubling in the treated
log2(20/20)
```

```
[1] 0
```

```
#This means that there is no difference
log2(10/20)
```

```
[1] -1
```

Differentially expressed of the treated as compared to the control is what we are looking for

```
#Halving in the treated
```

A common rule of thumb cut-off for calling a gene "differentially expressed" is a log2 fold-change value of either >+2 or <-2 for "up regulated" and "down regulated" respectively.

We first need to remove zero count genes- as we can't say anything about these genes anyway and their division of log values are messing things up (divide by zero) or the -infinity log problem.

```
to.rm.ind<-rowSums(meancounts[,1:2]==0)>0
mycounts<-meancounts[!to.rm.ind,]
```

> Q. How many genes do we have left that we can say something about ie they don't have 0 counts?

```
nrow(mycounts)
```

```
[1] 21817
```

```
sum(meancounts$log2fc >+2, na.rm=T)
```

```
[1] 1846
```

> Q. How many genes are upregulated and how many are downregulated ?

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

```
[1] 250
```

```
down.ind
```

```
[1] 367
```

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

No we are missing stats!!! Are these differences significant?

DESeq analysis

Let's do this properly with the help of the DESeq2 package.

```
#|message:false
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, setdiff, sort,
    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
```

```
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 have to use a specific data object for working with DESeq

```
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 our main analysis with the `DESeq()` function.

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

```
estimating size factors
```

```
estimating dispersions
```

```
gene-wise dispersion estimates
```

```
mean-dispersion relationship
```

```
final dispersion estimates
```

```
fitting model and testing
```

To get the results out of our `dds` object, we can use the DESeq function called `results()`

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

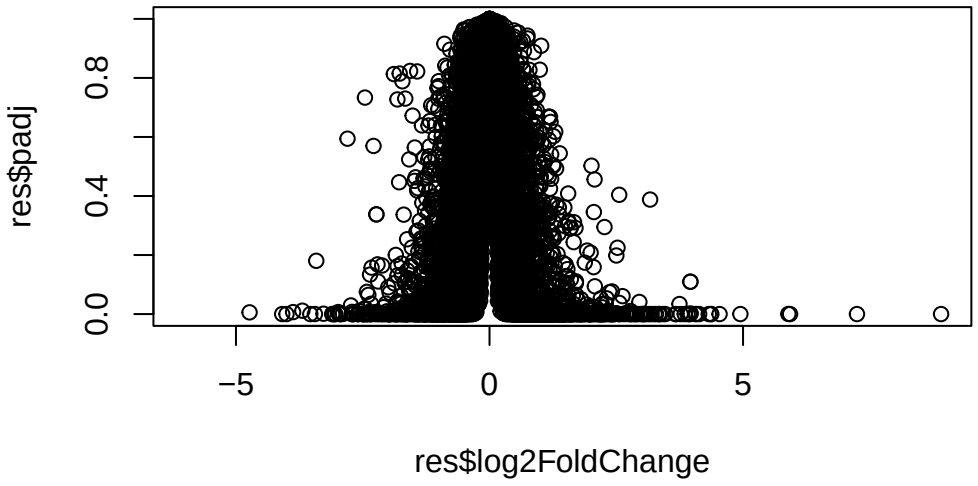
```
log2 fold change (MLE): dex treated vs control
Wald test p-value: dex treated vs control
DataFrame with 6 rows and 6 columns
                  baseMean log2FoldChange     lfcSE      stat    pvalue
                 <numeric>      <numeric> <numeric> <numeric> <numeric>
ENSG00000000003 747.194195     -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
```

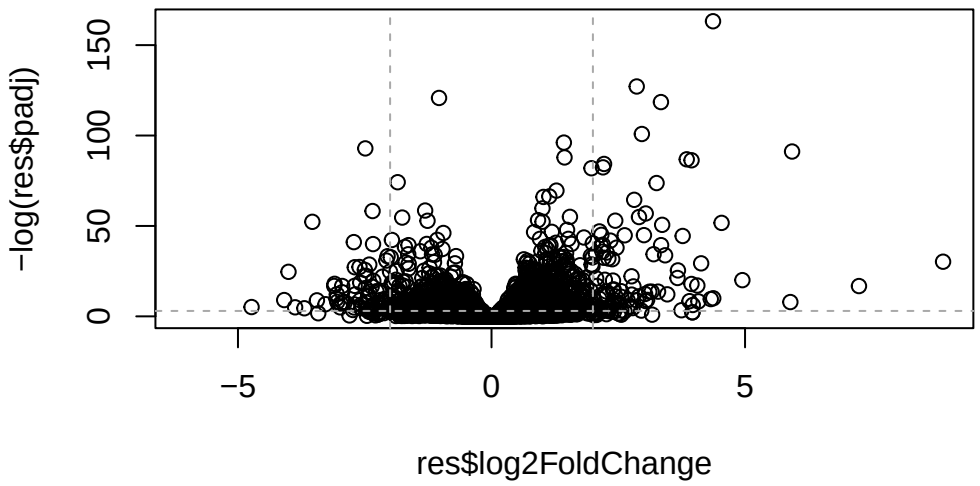
padj is the adjusted p-value. We are dealing with a dataset of about 20,000, of which 5% is a lot. So we make an adjusted p-value to account for the "torture" on the dataset by running a test multiple times.

A very common and useful summary figure for this type of analysis is called a volcano plot- a plot of log2FC vs P-value. We use the `padj` the adjusted P-value for multiple testing.

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



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

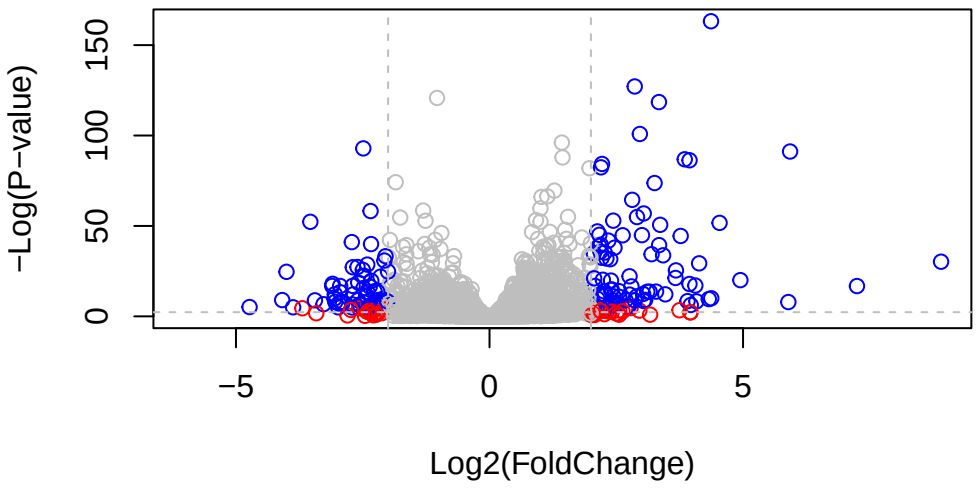



```r
mycols <- rep("gray", nrow(res))
mycols[ abs(res$log2FoldChange) > 2 ]  <- "red"

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

```
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
```

```
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), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     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>
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       FIRRM
ENSG00000000938        NA         FGR
```

I also want an entrez column

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="ENTREZID",          # 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 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       FIRRM       55732
ENSG00000000938        NA         FGR        2268
```

```
res$entrez <- mapIds(org.Hs.eg.db,
                     keys=row.names(res), # Our genenames
                     keytype="ENSEMBL",        # The format of our genenames
                     column="GENENAME",          # 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 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          tetraspanin 6
ENSG00000000005        NA        TNMD            tenomodulin
ENSG00000000419  0.176032        DPM1 dolichyl-phosphate m..
ENSG00000000457  0.961694       SCYL3 SCY1 like pseudokina..
ENSG00000000460  0.815849       FIRRM FIGNL1 interacting r..
ENSG00000000938        NA         FGR FGR proto-oncogene, ..
```

Pathway analysis

Now that I have added the necessary annotation data, I can talk to different databases that use these IDs.

We will use the `gage` package to do geneset analysis (a.k.a pathway analysis, geneset enrichment, overlap 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)

data(kegg.sets.hs)
```

We will use KEGG first ()

```
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"
```

The main `gage()` function requires a named vector of fold changes, where the names of the values are the Entrez gene IDs.

```
foldchange <- res$log2FoldChange
names(foldchange) <- res$symbol
head(foldchange)
```

```
     TSPAN6        TNMD        DPM1       SCYL3       FIRRM         FGR
-0.35070302          NA  0.20610777  0.02452695 -0.14714205 -1.73228897
```

Run the analysis

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

Let's look at what's in our results here

```
attributes(keggres)
```

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

```
head(keggres$less, 3)
```

```
                                      p.geomean stat.mean p.val q.val
hsa00232 Caffeine metabolism                 NA       NaN    NA    NA
hsa00983 Drug metabolism - other enzymes     NA       NaN    NA    NA
hsa01100 Metabolic pathways                  NA       NaN    NA    NA
                                      set.size exp1
hsa00232 Caffeine metabolism                 0   NA
hsa00983 Drug metabolism - other enzymes     0   NA
hsa01100 Metabolic pathways                  0   NA
```

I can now use the returned pathway IDs from KEGG as input to the `pathview` package to make pathway figures with our DEG's highlighted.

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

```
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/neha2/Desktop/Winter 2024/BIMM 143/Class 13

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