

Class12

Hetian Su

Install Bioconductor

```
# install.packages('BiocManager')  
# BiocManager::install()
```

```
# BiocManager::install('DESeq2')  
library(BiocManager)
```

Bioconductor version '3.15' is out-of-date; the current release version '3.16' is available with R version '4.2'; see <https://bioconductor.org/install>

```
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

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 4.2.2

Attaching package: 'MatrixGenerics'

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

```
colAlls, colAnyNAs, colAnys, colAvgPerRowSet, 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, rowAvgPerColSet,
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
```

Read in data for input to DESeq2

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

```
head(counts)
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG000000000003	723	486	904	445	1170
ENSG000000000005	0	0	0	0	0
ENSG000000000419	467	523	616	371	582
ENSG000000000457	347	258	364	237	318
ENSG000000000460	96	81	73	66	118
ENSG000000000938	0	0	1	0	2

	SRR1039517	SRR1039520	SRR1039521
ENSG000000000003	1097	806	604
ENSG000000000005	0	0	0
ENSG000000000419	781	417	509
ENSG000000000457	447	330	324
ENSG000000000460	94	102	74
ENSG000000000938	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

```
nrow(counts)
```

```
[1] 38694
```

There are 38694 genes.

Q2

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

```
[1] 4
```

There are 4 control samples.

Toy differential gene expression analysis

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

```
ENSG000000000003 ENSG000000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460  
          900.75           0.00           520.50           339.75           97.25  
ENSG000000000938  
          0.75
```

Q3

We can make this approaches into methods with “control” or “treated” as input. Also, the 4 should be replaced by a code that extract number of samples automatically.

Q4

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

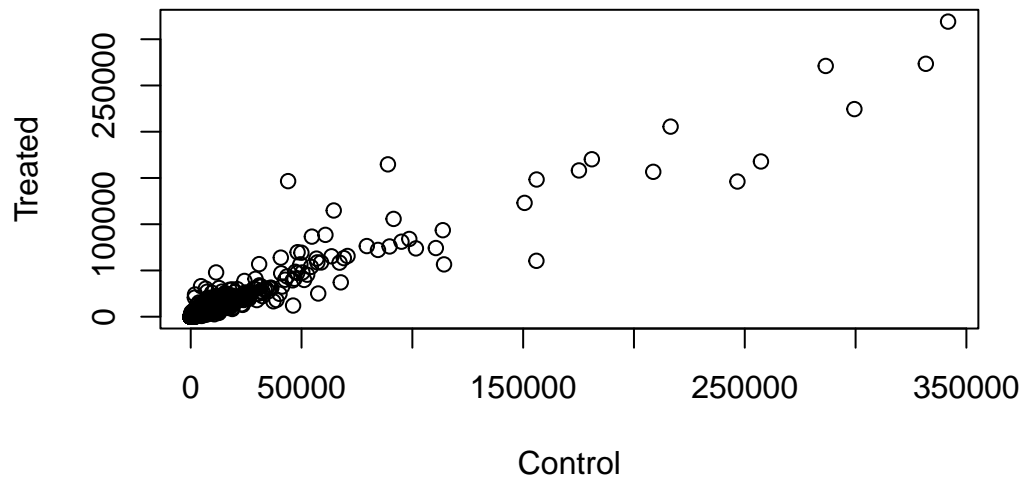
```
ENSG000000000003 ENSG000000000005 ENSG000000000419 ENSG000000000457 ENSG000000000460  
          658.00           0.00           546.00           316.50           78.75  
ENSG000000000938  
          0.00
```

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

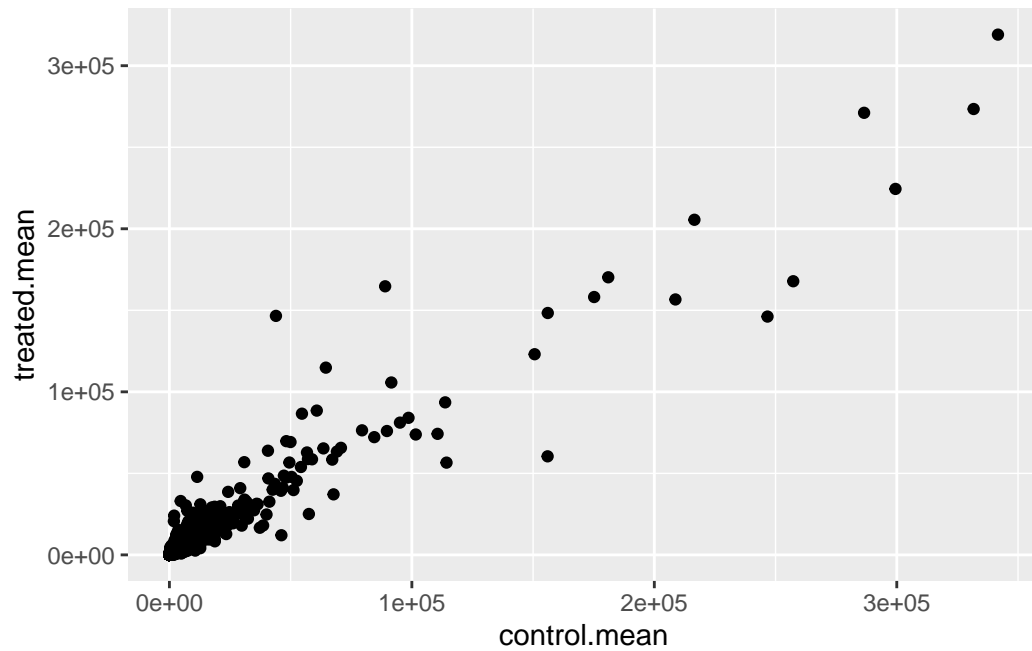
```
control.mean treated.mean
23005324      22196524
```

Q5

```
plot(control.mean, treated.mean, xlab='Control', ylab='Treated')
```



```
library(ggplot2)
ggplot(meancounts)+
  aes(control.mean, treated.mean)+
  geom_point()
```

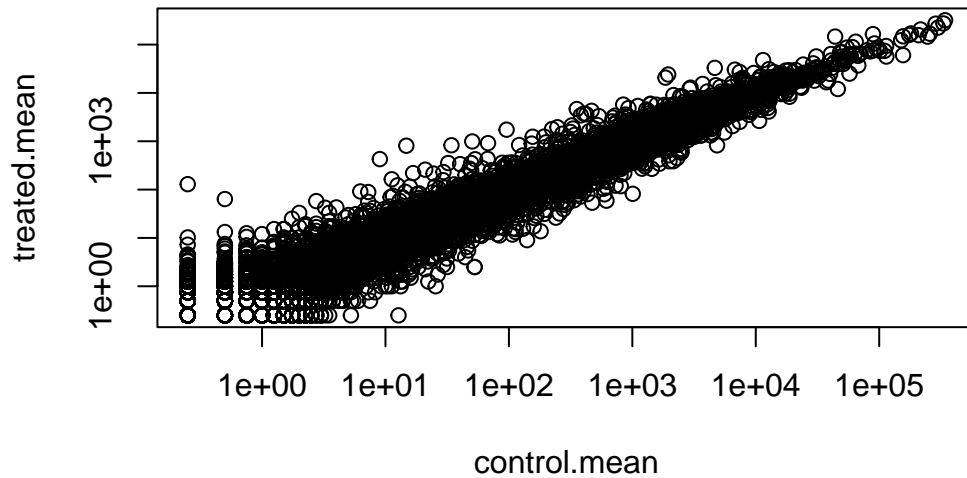


Q6

```
plot(control.mean, treated.mean, 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



```

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
ENSG000000000419	520.50	546.00	0.06900279
ENSG000000000457	339.75	316.50	-0.10226805
ENSG000000000460	97.25	78.75	-0.30441833
ENSG000000000938	0.75	0.00	-Inf

```

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
ENSG000000000003	900.75	658.00	-0.45303916
ENSG000000000419	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

`arr.ind` tells the function to return array indices instead of logical vectors. When array indices are returned, a gene that has 0 expression in both control and treated will give rise to duplicated array indices, so the `unique()` function is needed.

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

Q8

```
sum(up.ind)
```

```
[1] 250
```

There are **250** upregulated genes.

Q9

```
sum(down.ind)
```

```
[1] 367
```

There are **367** down-regulated genes.

Q10

No because the counts are not normalized.

DESeq Analysis

```
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): ENSG000000000003 ENSG000000000005 ... ENSG00000283120  
               ENSG00000283123  
rowData names(0):  
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521  
colData names(4): id dex celltype geo_id
```

```
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>
ENSG000000000003	747.1942	-0.3507030	0.168246	-2.084470	0.0371175
ENSG000000000005	0.0000	NA	NA	NA	NA
ENSG0000000000419	520.1342	0.2061078	0.101059	2.039475	0.0414026
ENSG0000000000457	322.6648	0.0245269	0.145145	0.168982	0.8658106
ENSG0000000000460	87.6826	-0.1471420	0.257007	-0.572521	0.5669691
...
ENSG00000283115	0.000000	NA	NA	NA	NA
ENSG00000283116	0.000000	NA	NA	NA	NA
ENSG00000283119	0.000000	NA	NA	NA	NA
ENSG00000283120	0.974916	-0.668258	1.69456	-0.394354	0.693319
ENSG00000283123	0.000000	NA	NA	NA	NA
	padj				
	<numeric>				
ENSG000000000003	0.163035				
ENSG000000000005	NA				
ENSG0000000000419	0.176032				
ENSG0000000000457	0.961694				
ENSG0000000000460	0.815849				
...	...				
ENSG00000283115	NA				
ENSG00000283116	NA				
ENSG00000283119	NA				
ENSG00000283120	NA				
ENSG00000283123	NA				

```
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
```