

Plan for the first part of Day 2

- Learn how to read data sets into R.
Files: `data1.txt`, etc.
- Calculate summary statistics for data analysis.
- Use basic plotting features to visualize data.
- Export objects out of R.
- Manage R Workspace
- Save `.Rhistory` and `.RData` files

Getting Data Into R

For today's example, download file `data1.txt` from

<http://tiny.cc/BootcampData1>

For rectangular (flat files) data, use `read.table`.

```
> fileName <- "data1.txt"
```

```
> d1 <- read.table(fileName, header=T, sep="\t")
```

File has column labels.



Denotes the delimiter. `"\t"` for tab-delimited files, `" , "` for comma-separated.



How many rows does this data set have?

How many columns?

```
> ncol(d1)
```

```
> nrow(d1)
```

Summary Statistics

```
> normNumbers <- rnorm(1000)
```

```
> summary(normNumbers)
```

To get these functions individually:

```
> mean(normNumbers)
```

```
> median(normNumbers)
```

```
> min(normNumbers)
```

```
> max(normNumbers)
```

```
> sd(normNumbers)
```

What about the quartiles?

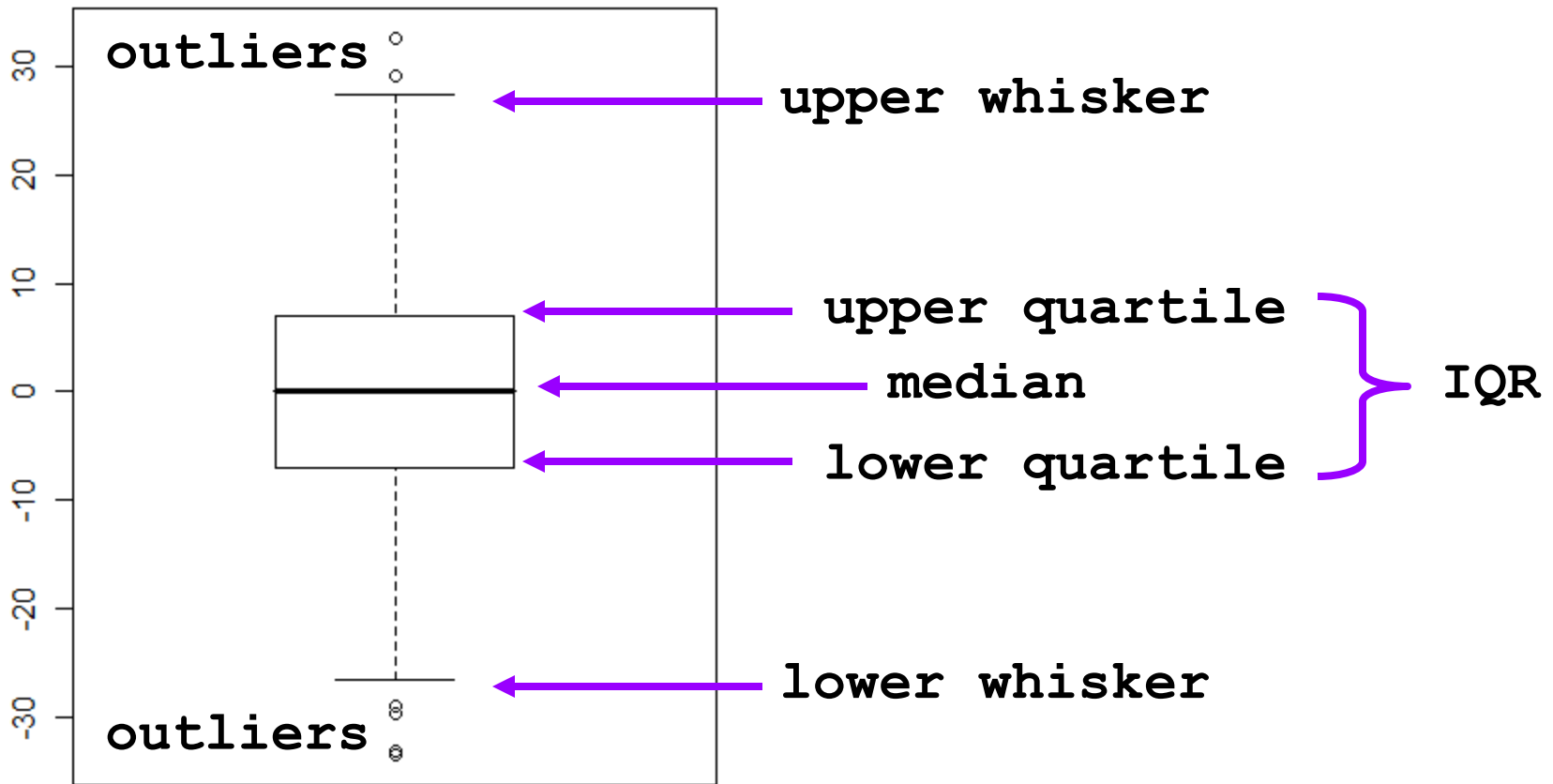
```
> quantile(normNumbers, 0.25)
```

```
> quantile(normNumbers, 0.75)
```

Note: `quantile` computes sample quantiles.

Compute summary
statistics for the data
you just
downloaded!

Boxplots



Visualizing Distributions with Boxplots

Starting with a standard normal random values:

```
> xy <- rnorm(1000)
```

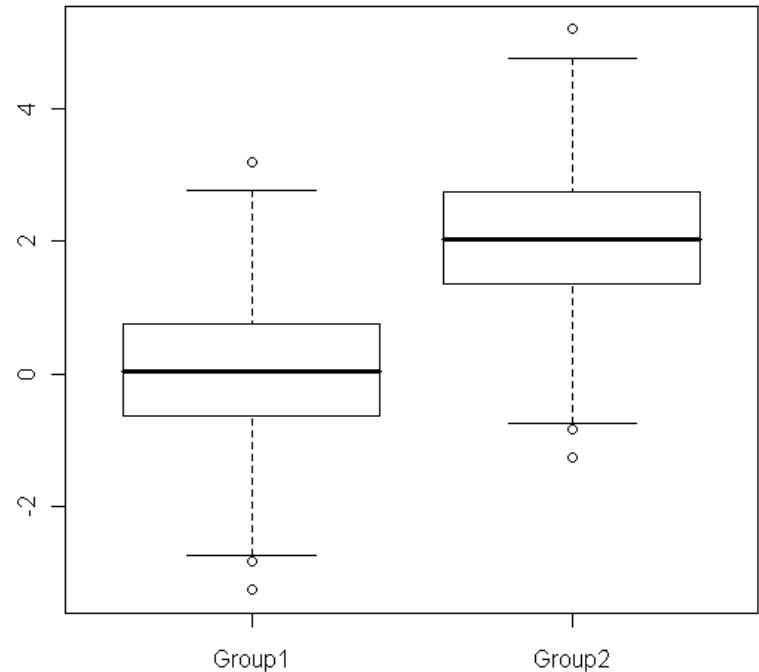
Boxplots

```
> boxplot(xy)
```

```
> zz <- rt(1000, df=2)
```

```
> boxplot(xy, zz)
```

Make boxplots for
the data you just
downloaded!



Tweaking the `boxplot` function

Adding x and y-axis labels:

```
> boxplot(xy, xlab="Group1", ylab="Scores",  
main="Title")  
> boxplot(xy, zz, names=c("Group1", "Group2"),  
ylab="Scores")
```

Box-plotting data frame objects:

```
> myDF <- data.frame(Monday=xy, Saturday=zz)  
> boxplot(myDF, ylab="Scores")
```

The column names automatically appear as labels on the boxplot.

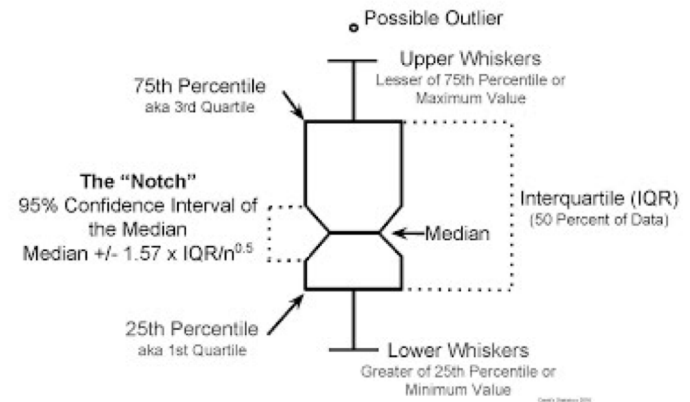
Boxplots to compare between groups

Boxplot of MPG by Car Cylinders

```
boxplot(  
    mpg~cyl,  
    data=mtcars,  
    main="Car Milage Data",  
    xlab="Number of Cylinders",  
    ylab="Miles Per Gallon")
```

Notched Boxplots to compare between groups with median comparison

```
# Boxplot of Toothgrowth  
boxplot(  
  len~supp*dose,  
  data=ToothGrowth,  
  notch=TRUE,  
  col=(c("gold", "darkgreen")),  
  main="Tooth Growth",  
  xlab="Supplement and Dose")
```



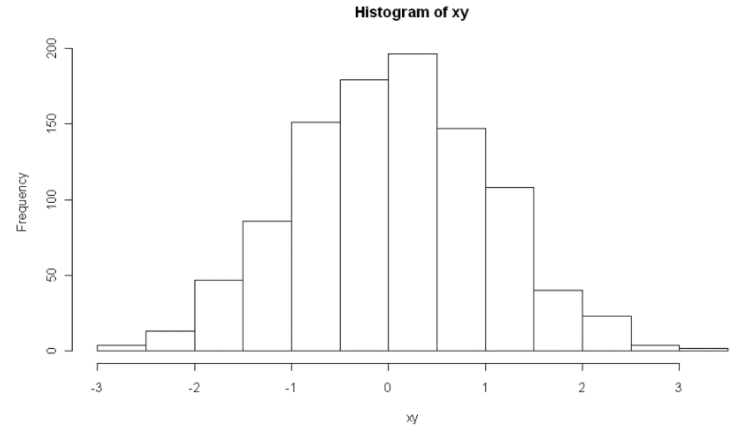
Notch defines the median

Visualizing Distributions with Histograms

By default, R will plot histograms as a representation of frequencies.

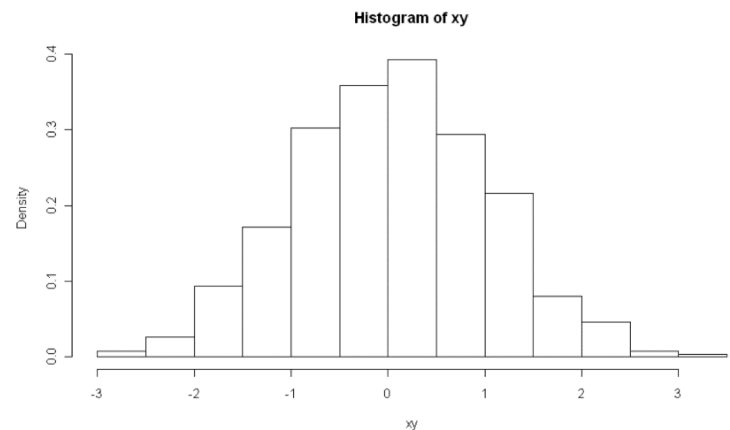
```
> xy <- rnorm(1000)
```

```
> hist(xy)
```



To plot a histogram of probability densities instead, set the **freq** argument to **F**.

```
> hist(xy, freq=F)
```



Make histograms for
the data you just
downloaded!

Let's put what we've learnt into practice...

Download the two other files `data2.txt` and `data3.txt` from <http://tiny.cc/BootcampData2> and <http://tiny.cc/BootcampData3>

Compute summary statistics for these new data sets.
Compare them to the summary statistics obtained from the first data set.

Make boxplots comparing all three sets of data.
Make histograms comparing the distributions of all three sets of data.

A handy line of code:

```
> par(mfrow=c(1,2)) # puts two plots in the one frame
```

Using the `plot` function

```
> xVals <- 1:10  
> yDat <- 1:10  
> plot(xVals, yDat)
```

We can add labels on the x and y-axis:

```
> plot(xVals, yDat, xlab="X", ylab="Y")
```

and a title:

```
> plot(xVals, yDat, main="my title goes here")
```

Try these different plotting styles out:

```
> plot(xVals, yDat, type="l") # line plot  
> plot(xVals, yDat, type="p") # just points  
> plot(xVals, yDat, type="h") # histogram-like
```

Exporting Data out of R

Extra Challenge
Can you merge all
three data sets and
write this out as a
single file?

Recall our object **d1**:

Say we're ready to export this as a text file.

```
> write.table(d1, file="myd1.txt", sep="\t",  
  quote=F, col.names=T, row.names=F)
```

This produces a file called **myd1.txt**, that has:

- Columns are tab-delimited
- Character strings don't have quotation marks.
- Column labels are preserved.
- There are no row labels.

See **?write.table** for more ways to customize the output.

Variations on `write.table`

- If we turn on the `quote` argument:

```
> write.table(d1, file="myd1_a.txt", sep="\t",  
  quote=T, col.names=T, row.names=F)
```

we get a text file with quotation marks (") around each entry.

- If we change the `sep` argument:

```
> write.table(bio.df, file="myd1_b.txt",  
  sep=",", quote=F, col.names=T, row.names=F)
```

we get a text file that has columns separated by a comma (,).

- If we turn on the `row.names` argument:

```
> write.table(d1, file="myd1_c.txt", sep=",",  
  quote=F, col.names=T, row.names=T)
```

we get a text file that has row names (as well as column names).

Managing Your Workspace

All objects that you create are stored in the workspace.

To list all objects:

```
> ls()
```

To remove objects from the workspace:

```
> rm("names")
```

R sees the rest of the world (your computer) from a local directory. This is called the working directory.

```
> getwd()
```

To change this to something else:

```
> setwd("C:/Temp")
```

Making (.R) History

Keeping a copy of the R code you've written, even while just learning is valuable.

R automatically keeps track of all the commands you've entered during the session.

```
> history()
```

Note: `?history` tells us that to see all the commands, use

```
> history(max.show=Inf)
```

We can save these commands as a `.Rhistory` file.

```
> savehistory(file=".Rhistory")
```

In a new session, we can load an existing `.Rhistory` file.

```
> loadhistory(file=".Rhistory")
```

Saving Your R Session

You can also save the R objects that you generated during an R session.

```
> save.image(file=".RData")
```

In a new R session, you can load these objects back in.

```
> load(".RData")
```

These **.RData** files can get quite huge, so it might be more sensible to save only a few key objects (you probably don't need everything anyway).

```
> save(list=c("bmi", "height", "weight",  
"names"), file="bmi.RData")
```

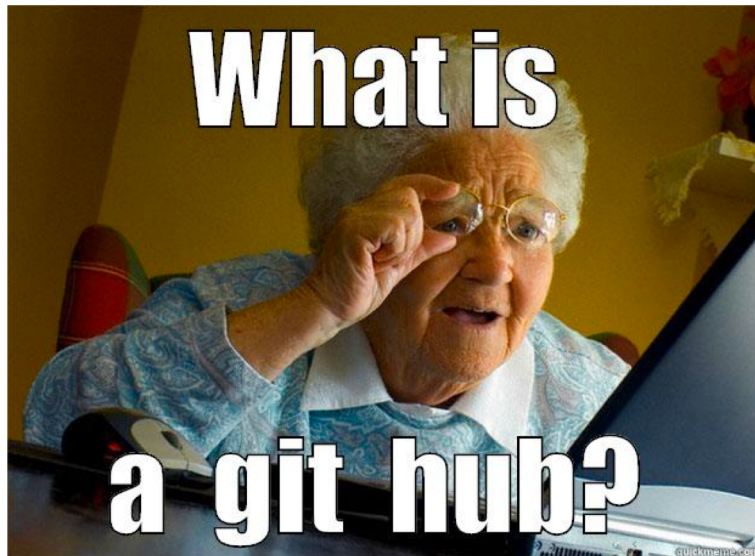
To use these objects in a different R session,

```
> load("bmi.RData")
```


Intro to Git/GitHub

<https://github.com/join>

<https://education.github.com/pack>



GitHub

What is Git & GitHub?

Git is an example of **version control system**

- Revert files/project to the original state
- Compare changes over time
- See who modified what? Control modifications by collaborators




Github is a **user-friendly repository hosting service** for Git




- Web-based interface that works on top of Git
- Social platform to share knowledge and work (Reproducibility in research)
- Collaboration between scientists and developers




Homepage and repositories



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)





 ameya225


Repositories


New repository


Find a repository...


 [ameya225/R_Bootcamp_E...](#)


 [ameya225/agingpapers](#)

 [ameya225/learn-co-sand...](#)

 [jmarlab/R_Bootcamp_O...](#)

 [ameya225/useful_functions](#)


 [ameya225/R_Bootcamp_O...](#)

 [ameya225/GTEx](#)

Show more

Your teams

Find a team...

 [learn-co-students/data-scie...](#)

Browse activity

[Discover repositories](#)

 jmarlab created a repository [jmarlab/R_Bootcamp_Orientation](#) 6 hours ago

[jmarlab/R_Bootcamp_Orientation](#)

★ Star

This is the repo for Einstein's R Bootcamp (2018 1st year PhD orientation)

Updated Aug 7

 rhiever starred [jhfhfhfj1/autokeras](#) 4 days ago

[jhfhfhfj1/autokeras](#)

★ Star

This is an automated machine learning (AutoML) package.

 Python ★ 1.8k Updated Aug 7

 rhiever starred [fivethirtyeight/russian-troll-tweets](#) 6 days ago

[fivethirtyeight/russian-troll-tweets](#)

★ Star

★ 371 Updated Aug 7

Walk-through of GitHub's basic concepts

Creating a repo

Creating a repository for multiple people to work together

Master in a repository

This is the final version that is considered ready to use by anybody in the team or outside if repository is public.

Creating a Branch

- Create a branch in your project, for an environment where you can try out new ideas.
- Changes you make on a branch don't affect the master unless pull request is accepted.
- Changes committed to branch reflects for you to keep track of different versions

Adding Commits

- Keeps track of your progress as you work on a branch or master.
- Creates a transparent history that others can follow to understand what you've done and why.

Forking a repo Fork

- It creates a copy for you to work on independently without any changes to theirs.
- Submit a pull request to owner so that the owner can incorporate changes.

Github concepts continued...

Pull requests

- Pull Requests initiates discussion about your commits or changes made to a code.
- See exactly what changes would be merged if pull request is accepted.
- Use GitHub's @mention system in your Pull Request message to ask for feedback from specific people or teams, or for someone to review your work

Issues

Markdown syntax

Watch and Star

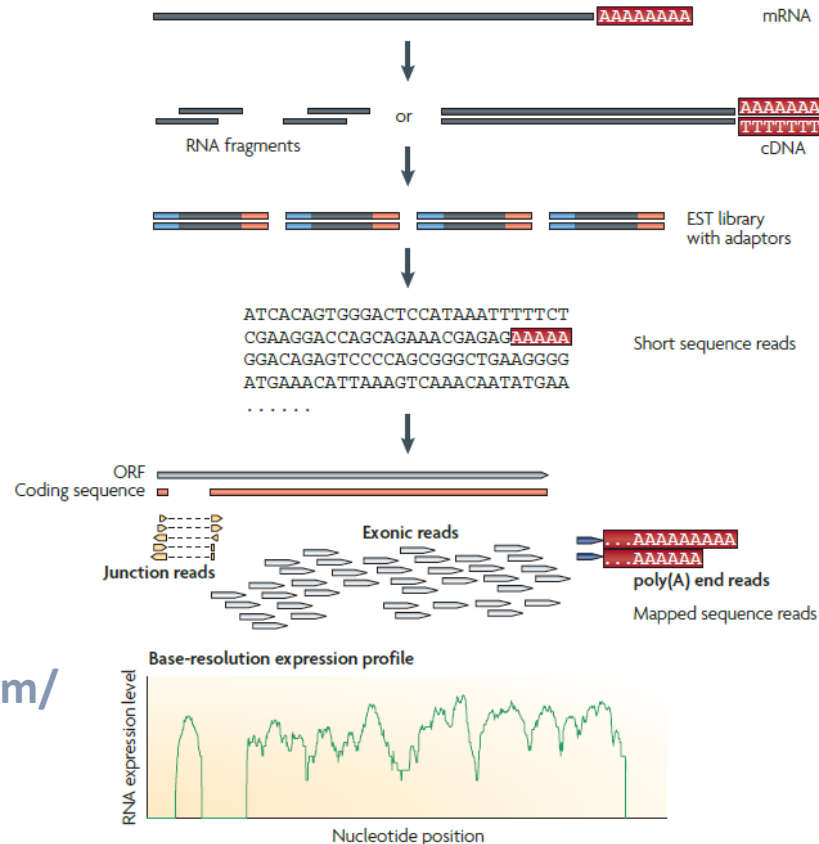
Use Rstudio to pull, commit and push for GitHub

Generally easier in the terminal (5 basic commands)

Can also use GitHub Web Interface

https://github.com/jmarlab/R_Bootcamp_Orientation.git

RNA-sequencing Technology

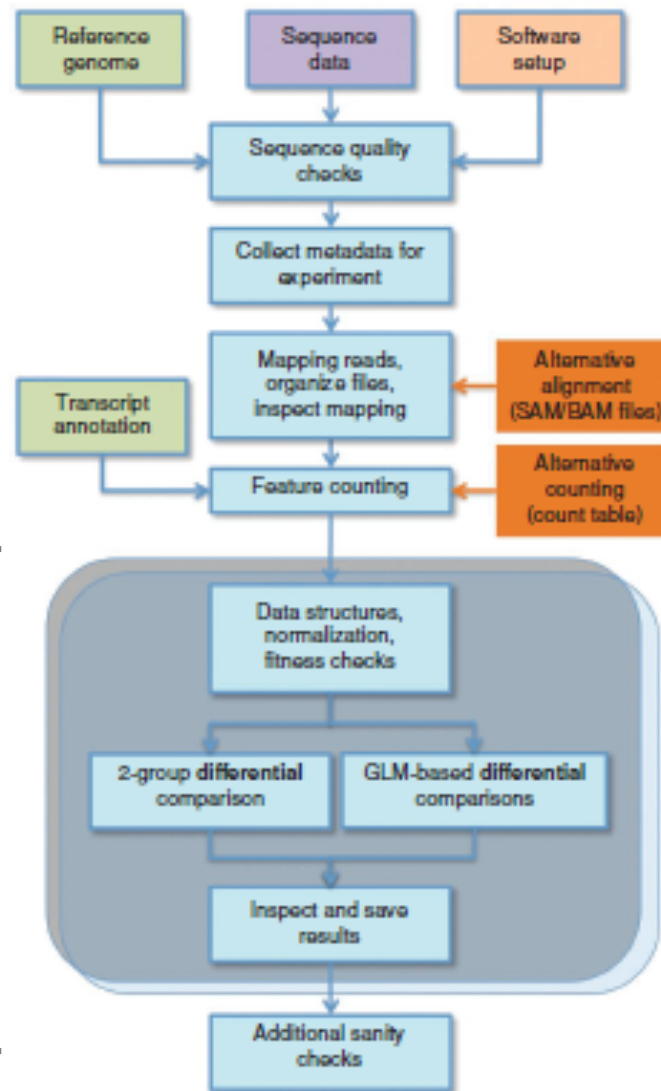


Illumina video:
<https://www.youtube.com/watch?v=fCd6B5HRaZ8>

Figure 1 | **A typical RNA-Seq experiment.** Briefly, long RNAs are first converted into a library of cDNA fragments through either RNA fragmentation or DNA fragmentation (see main text). Sequencing adaptors (blue) are subsequently added to each cDNA fragment and a short sequence is obtained from each cDNA using high-throughput sequencing technology. The resulting sequence reads are aligned with the reference genome or transcriptome, and classified as three types: exonic reads, junction reads and poly(A) end-reads. These three types are used to generate a base-resolution expression profile for each gene, as illustrated at the bottom; a yeast ORF with one intron is shown.

Wang et al. (2009). Nature Reviews Genetics. RNA-seq: a revolutionary tool for transcriptomics.

RNA-seq Analysis Pipeline



Bioconductor R
packages:

Limma, edgeR

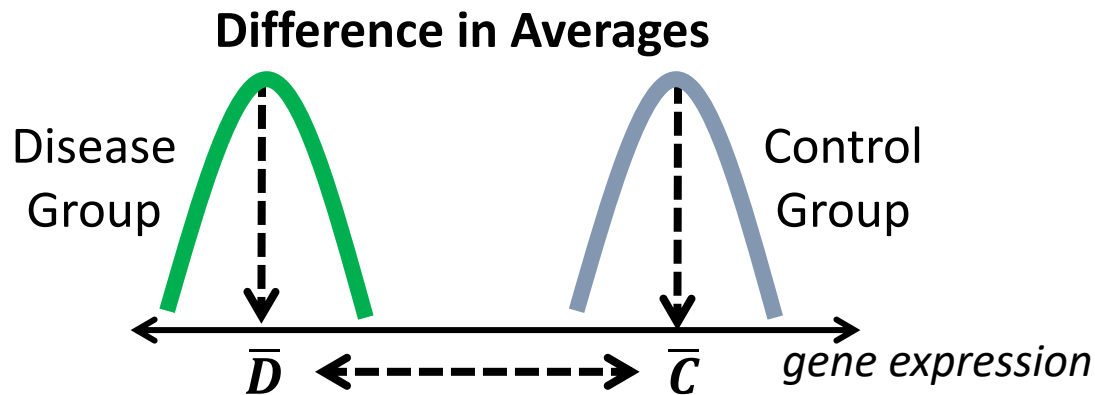
DESeq, DESeq2

Specialized Platforms:
monocle (single cell)

Assessing Differential Expression

The goal is to assess differential gene expression
between conditions-
Disease vs Normal; Treated vs Untreated

$$T_{(gene)} = \frac{\bar{D} - \bar{C}}{f(Var(D, C))}$$



What do we know about our patient data?

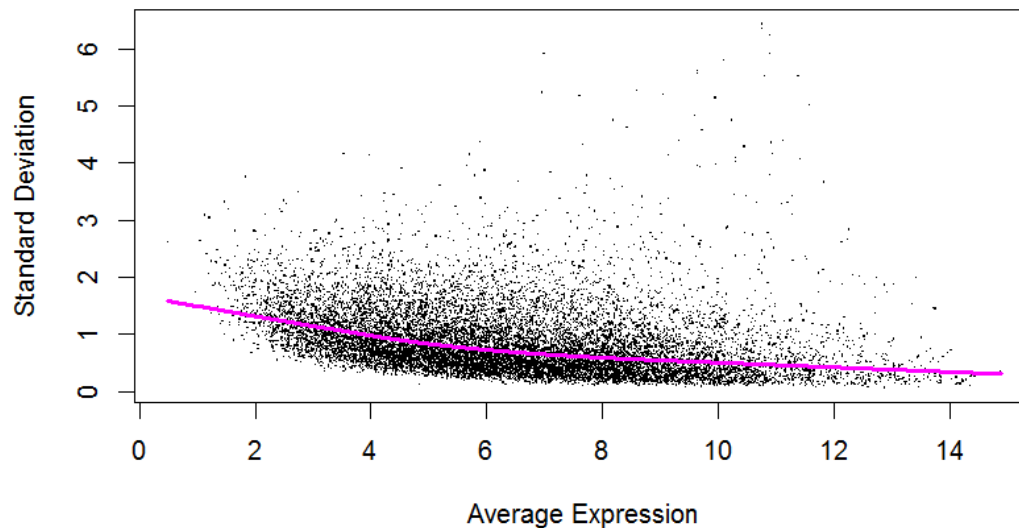
Where is this information stored?

How can we identify which columns correspond to which patients?

Testing for differential expression using limma*

- When dealing with –omic level platforms, we are working with high-dimensional data, and tiny quantities of biological material.
- Noisy data and false positives are therefore bound to occur.
- Limma uses an empirical Bayes method to estimate differential expression by minimizing the variance estimate.
- This results in a moderated T-statistic:

$$T_{(gene)} = \frac{\bar{D} - \bar{C}}{f(Var(D, C) + \alpha)}$$



*limma is a R/Bioconductor package that is used for microarray and RNA-seq data analysis.

Assessing differential expression with limma

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("limma")          # install package

> library(limma)
> design <- model.matrix(~0+status)

> fit <- lmFit(object = edat, design = design)
> my.contrasts <- makeContrasts(
  Pri_vs_Nor = Primary-Normal,
  levels = design)

> fit2 <- contrasts.fit(fit = fit, contrasts =
my.contrasts)
> fit2 <- eBayes(fit2)
> deg_Pri_vs_Nor <- topTable(fit2, coef = "Pri_vs_Nor",
adjust.method = "BH", p.value = 0.001, number =
nrow(fit2))
```

Each team downloads one of these datasets

- Lung cancer in non-smokers (Tumor vs Normal)
 - <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE87340>
- Alzheimer's Disease- Human brain samples (Mount Sinai Brain Bank)
 - <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE53697>
- Huntington's Disease vs isogenic controls (Neural Stem cells and iPSCs)
 - <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE74201>