Data Visualization

First point: use the Source pane in RStudio. It's good practice as it saves your work, and it makes things MUCH easier when it comes to running and debugging code.

type a command in the source pane
move the cursor to the line with the command you want to run
hit command-"enter" to run the code, and you'll see the result in the console.

Obviously, we can summarize data into tables and look at pathways and p-values . . . but to really understand your experiment you need to look at the data at every stage of processing.

This is well illustrated by Anscombe's quartet, which is one of the (MANY) pre-loaded datasets in R.

```
> ?datasets
```

In RStudio, look for information about Anscombe:

```
> ?anscombe
```

(note that ?Anscombe gives you a "not found" message, but ??Anscombe will find it)

loading data:

```
> rm(list=ls())
```

(this clears your workspace of all objects and data, and is a good thing to do before you start a new project)

```
> library(stats)
> ls()
character(0)
> data(anscombe)
> ls()
```

These are admittedly pathological datasets but are a good illustration of why looking at data is important. You could think of these datasets as gene expression over time points.

Each of the values below is the same for each set:
number of points
average x
average y
regression line
standard error of slope
sum of squares
residual sum of squares
correlation coefficient r2

```
> anscombe
> str(anscombe)
> anscombe$x1
```

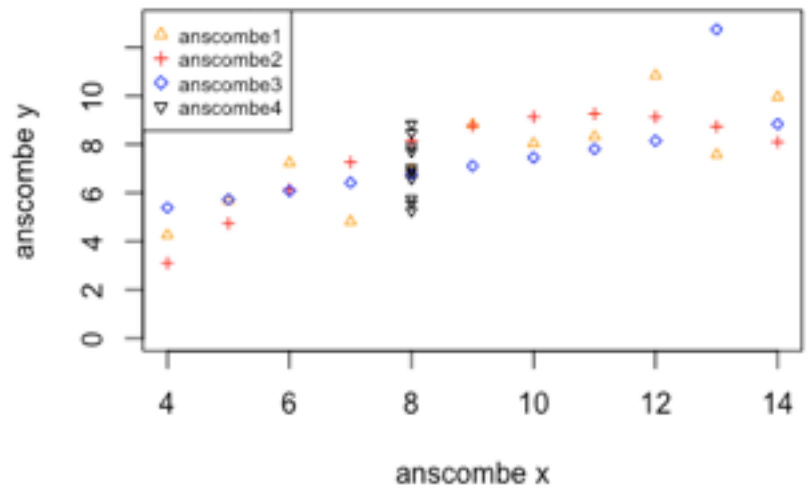
Make some plots:

```
> plot(anscombe$x1, anscombe$y1)
> plot(anscombe$x2, anscombe$y2)
> plot(anscombe$x3, anscombe$y3)
> plot(anscombe$x4, anscombe$y4)
```

Make a plot with all of the data:

```
> plot(anscombe$x1, anscombe$y1)
> points(anscombe$x2, anscombe$y2)
> points(anscombe$x3, anscombe$y3)
> points(anscombe$x4, anscombe$y4)
```

Exercise:
Now make prettier scatterplots. Explore the plot function to figure out how to use different plotting symbols and make nice axis labels and a title, and create a legend. You will need to look at the help for 'par' to get details on most of the parameters.



Find and explore a real dataset from GEO:
https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE93783

Unzip the data file using gunzip and read it into R:

```
> tab <- read.table("GSE93783_igenomes_known_GENES_counts_Fibro.tab",
header=T, row.names=1)
> head(tab)
> dim(tab)
```

normalize the gene expression (we'll use quantile normalization for now, though more sophisticated methods are available and often more appropriate)

```
> ??quantile
> library(limma)
> nltab <- normalizeQuantiles(tab)

> ?heatmap
> heatmap(nltab)
```

hmm . . . not the best idea. Let's just use the data with the most variance.

```
> ?var
```

ASIDE: *apply functions

```
> var(nltab[1,])
> nlmat <- as.matrix(nltab)
> var(nlmat[1,])
> dim(nlmat)
> nlvar <- vector(length=23710)
> for (i in 1:length(nlmat[,1]))
  { nlvar[i] <- var(nlmat[i,])}


> nlvar <- apply(nltab, 2, var)
> nlvar <- apply(nltab, 1, var)
> summary(nlvar)
```

What did the normalization do, by the way? try a boxplot.

```
> boxplot(tab)
> boxplot(tab, outline=F)
> boxplot(nltab)
> boxplot(nltab, outline=F)
```

Exercise:
Write a statement to subset the nltab object into a new object, highvartab, that contains only genes that have variance above 1000000. (this is a pretty crude cutoff) Make a boxplot of this object.

Make a heat map of our new highvartab object:

```
> ?heatmap
> heatmap(highvartab)
> heatmap(as.matrix(highvartab))
```

Is this the right way to look at the data?
(no)
Use edgeR, as it provides the right model for looking at data like these, with replicates.

```
> biocLite("edgeR")
> library(edgeR)
> data <- tab
> libSizes <- as.vector(colSums(data))
> d <- DGEList(counts=data,group=whichtime,lib.size=libSizes)
> d <- calcNormFactors(d)
> d <- estimateCommonDisp(d)
> d <- estimateTagwiseDisp(d)
> de.com <- exactTest(d)
> results <- topTags(de.com,n = length(data[,1]))
> str(results)
> head(as.matrix(results$table))
```

Note that this is only using the first two time points. ANOVA or something similar can account for several time points simultaneously.

Viewing sequencing data

We'll use samtools to manipulate the rawseqs.sam file (get statistics etc) and convert it into something viewable.

First we need to convert the sam file to a bam file (binary format):

```
$ samtools view -bS rawseqs.sam > rawseqs.bam
```

and then sort the bam file:

```
$ samtools sort rawseqs.bam rawseqs_sorted
```

and finally index it:

```
$ samtools index rawseqs_sorted.bam
```

Look at some mapping stats:

```
$ samtools flagstat rawseqs_sorted.bam
$ samtools stats rawseqs_sorted.bam
```

and samtools can even generate output that is read by a variant caller:

```
$ samtools mpileup rawseqs_sorted.bam
```

IGV
Open IGV, select the correct genome build (hg38) and load rawseqs_sorted.bam (the index .bai file has to be in the same directory but isn't loaded). The alignments should match the gene CYP2D6, so navigate to that region. You should be able to see the alignments and zoom in to see any mismatches.

Exercises:
Write a shell script that will reproduce the samtools commands that you just ran. Start the script with #!/bin/bash and then just type the commands as you would on the command line, and save it as runsam.sh. You can execute the script by typing sh runsam.sh

Revisit your parsesam.py python program, which takes the name of a sam file as an argument and then does parsing. Add this program to the steps performed by your shell script.

R/Gviz
```
> source("https://bioconductor.org/biocLite.R")
> biocLite("Gviz")
> browseVignettes("Gviz")
```

Exercise:
Work through gviz user guide examples in R

RCircos, circlize etc. Not bioconductor packages
https://cran.r-project.org/

https://cran.r-project.org/web/packages/circlize/

installing CRAN packages:
```
> help("install.packages")
> install.packages(“circlize")
```

Exercise:
Work through circlize user guide examples in R