Notes for Stat 430 – Design of Experiments – Lab #1 Sept. 10, 2012.

Later weeks: Input by hand, install packages. Aov command.

Design of Experiments (sometimes called DoE or DoX for short), can involve a lot of tedious arithmetic. Although being familiar with the math that goes into an analysis is important, doing it by hand is a waste of time. To get around that, we'll be using the software package R in Stat 430 to do the tedious parts.

The lab portion of Stat 430 will involve extra practice questions to do by hand, and occasionally serve as an R tutorial.

If you are planning to bring a laptop to work along (not required), then please have R installed before the beginning of class. If there are issues with installing R, please e-mail me, Jack Davis, at jackd@sfu.ca , or the professor, Dr. Steve Thompson at thompson@sfu.ca .

There are many good secondary resources at

Quick-R http://statmethods.net/ (good start)

Comprehensive R Archive Network http://cran.r-project.org/doc/manuals/R-intro.html (tons of info)

SAS and R blog http://sas-and-r.blogspot.ca/ (more advanced)

What is R?

R is a statistical software package that is…

- **Command line based.** Using R involves writing code instead of clicking on a set of buttons. In contrast programs like SPSS and JMP are Graphic Interface based, and most of the work is done by a series of menus. If you have experience with programming, especially experience programming SAS, it will be useful (but not necessary) in working with R. Experience with Excel also helps a little.

- **GNU Open source.** R is legally free for anyone to download and use, AND MODIFY. That means that anyone can not only use it, but tinker around with it to make improvements.

- **Available on Linux, Windows, and Mac platforms.**

- Considered the standard for academic statistics.

Installing R.

Direct download links:

http://cran.r-project.org/bin/windows/base/R-2.15.1-win.exe

http://cran.r-project.org/bin/macosx/R-2.15.1-signed.pkg (I think)

For more information, go to

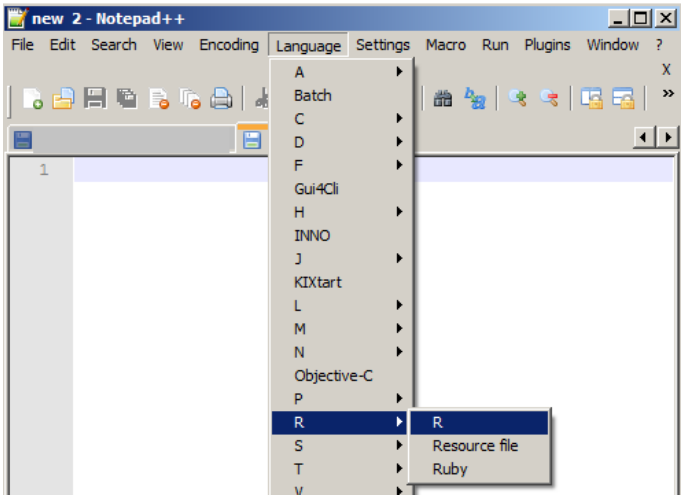
http://cran.r-project.org/

Installing Notepad++ (optional)

Plain old Notepad is good enough for writing R code, but I personally prefer to use something that color-codes my work for me, like Notepad++ There are lots of alternatives, this is just what I use.

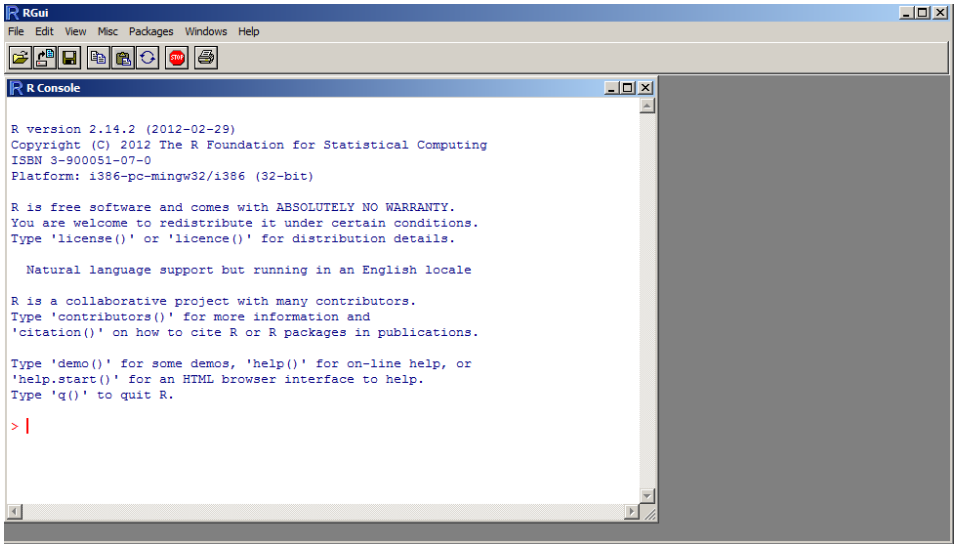
http://notepad-plus-plus.org/

Just make sure to use Language → R → R to get the proper colour coding.



Learning how to Learn, the ? and ?? commands.

When you open R, you should see a window like this. It will look a little different for Mac users.



The white sub-window with text that takes up most of the program space is called the CONSOLE. Any R code found anywhere is meant to be typed into the console or copy-pasted into the console from a notepad program.

We can use ??<*term*> to search for commands related to a specific term. For example, entering

```
??histograms
```

…will bring up a list of commands related to histograms, like this:

Help pages:

ggplot2::geom_histogram	Histogram
graphics::hist.POSIXt	Histogram of a Date or Date-Time Object
graphics::hist	Histograms
graphics::plot.histogram	Plot Histograms
grDevices::nclass.Sturges	Compute the Number of Classes for a Histogram
KernSmooth::dpih	Select a Histogram Bin Width
lattice::histogram	Histograms and Kernel Density Plots
lattice::panel.histogram	Default Panel Function for histogram

The first part is the name of the package that has the command, and the second part, after the ::, is code to do that command. We're going to try the command **`hist`** in the graphics package.

You can click on graphics::hist to get information the hist command, including examples, or you can use ?<command> to get information about single command. So

```
?hist
```

Brings up information about the hist() command. (Commands are often written with () at the end to show that there are functions, the input or parameters go inside the () usually.)

Since we can't take a histogram of nothing, we'll stop here until we've input some data.

Inputting data from a file.

First, make sure you have the Beardies.txt file downloaded before continuing. Use the menu at the top of the R window and choose File → Change Dir and navigate to the folder that contains "Beardies.txt" (data about some bearded dragons) and click OK.

Then enter the following into the console.

```
Beard = read.table("Beardies.txt",header=TRUE,sep= ' ')
```

Make sure these is a capital B in Beardies.txt.

- **read.table** is the command.
- **"Beardies.txt"** is the filename that contains the info you want.
- **header=TRUE** tells the console that the first line of the file is the variable names.
- **sep = ' '** tells the console to look for a space between the entries. Sometimes this is a tab or a comma, so it helps to open new datasets in notepad to see how they're set up before putting them in R.
- **Beard** is the name the dataset when it's in R. This can be any name without spaces and starting with a letter.

We can look at the first few lines of data in R using the **head()** command.

```
head(Beard)
```

You can refer to specific variables in the dataset with $.

Beard$breed will give you the breeds of all the bearded dragons in this set.

Beard$size will list the sizes.

This raw data isn't the most useful, we can also make a frequency table.

```
table( Beard$breed )
```

Frequency tables aren't useful for continuous data like size, but we can take a histogram instead.

```
hist( Beard$size )
```

We could change the number of bars that are shown by setting the parameter n.

```
hist( Beard$size, n=10)
```

We could also add a title with the parameter **main**…

```
hist( Beard$size, n=10, main="Dragon Sizes")
```

…and labels on the x and y axes with **xlab and ylab**.

```
hist(Beard$size,n=10, main="Dragon Sizes", xlab="Size in cm",
ylab="No. of Dragons")
```

Data can be input manually as well, but we'll deal with that another week.

Simple Analysis Function

R is also useful as a graphing calculator

You can produce mathematical results using + - * / ^ (and)

```
5^3
12 + (7/4)
sqrt(49)
```

There are also simple summary statistics:

`mean(Beard$size)` gives the mean

`sd(Beard$weight)` gives the standard deviation

`median(Beard$size)` gives the median

`length(Beard$size)` gives the number of elements, also known as the sample size.

T-Tests

The basic command for a t-test in R is **t.test()** , but there are several t-tests.

We can test the hypothesis that the mean size of all the beardies is 20 (A one-sample, two-sided test)

```
t.test(Beard$size, mu=20)
```

We can change the alternative to either side

```
t.test(Beard$size, mu=20, alternative="less")
```

We can also do two-sample tests, but we need to break the sample up first. The 1^{st}, 2^{nd}, 3^{rd}, … , 20^{th} beardies are of breed A, so we'll make that one group.

`groupA = size[1:20]` groupA now contains the sizes of the first 20 beardies

The 21^{st}, 22^{nd}, … , 40^{th} beardies are breed B, so we'll make that the other group.

```
groupB = size[21:40]
```

We can test if dragons in these groups have the same mean size (assuming normality within each group)

```
t.test(groupA, groupB)
```

We could include the assumption of equal variance

```
t.test(groupA, groupB, var.equal=TRUE)
```

Or treat this data as paired because the sample sizes are equal. (Just because you CAN doesn't mean you SHOULD. If these are 20 animals each of two breeds, a paired sample isn't implied)

```
t.test(groupA, groupB, paired=TRUE)
```

The t-test by formulas.

Finally, try to predict which of the t.test functions above will give you the same answer as t from the following code.

```
n1 = length(groupA)

n2 = length(groupB)

meanA = mean(groupA)

meanB = mean(groupB)

sdA = sd(groupA)

sdB = sd(groupB)


t = (meanA - meanB) / sqrt( sdA^2/n1 + sdB^2/n2)
```

When you're ready to check, type **t** into the console and press enter. Why would this give you the same answer for t as one of the t.test functions? *Hint: Page 48 of 8th edition of text.*