R HANDOUTS – HANDS ON!

Why R? Should I use the R statistical software for introductory statistics?
I assume you know that statistics is done with software and that learning a reputable software package should be a part of any applied statistics course, so that the question is which software?

Advantages

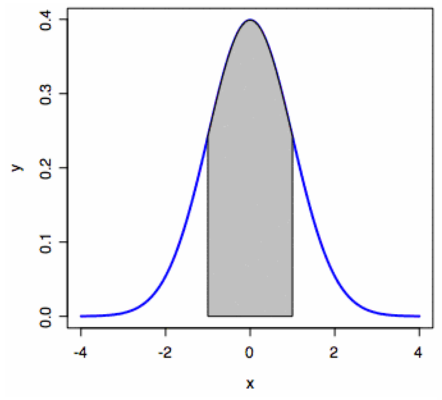
1. R is free. In addition to helping your school's budget, this means no whining for money, and immediate installation on any machine under your control It also means students can take it home and install it on their own machines. High school students can take the software to college with them.

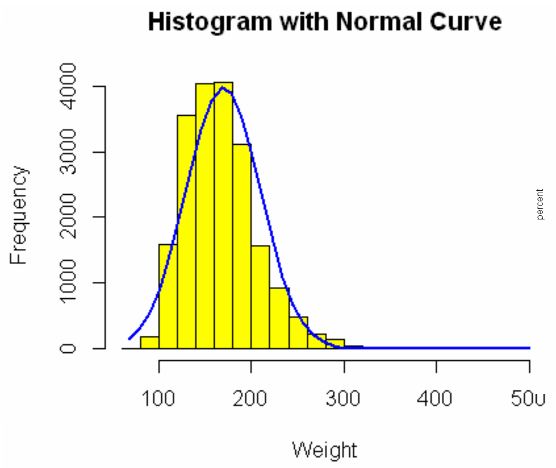
2. R runs on Windows, Mac and Linux/Unix platforms, so it probably runs on the university hardware as well as students' laptops .

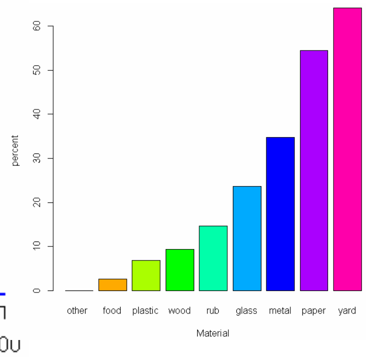
3. R is professional quality software used by professional statisticians. This sets it apart from Excel and various education programs.

4. R includes a vast array of statistical procedures. Few students will ever outgrow it no matter how many statistics courses they take. This sets it apart from graphing calculators which are a terminal technology supporting only the first course.

5. R was written for advanced graphical data analysis and has *knock-out* graphical output.

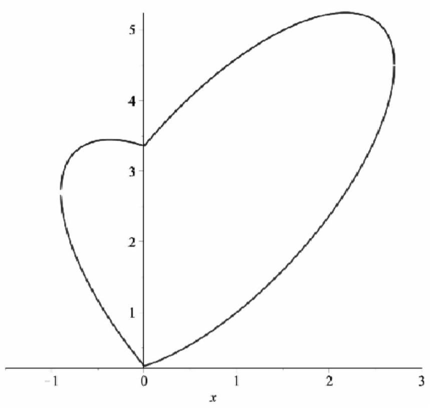






Disadvantages

1. The usual documentation for R assumes you already have a Ph.D. in statistics and extensive programming experience. However, you do not need all that to *use* R! All you need are the 13 handouts for my intro course. You can use them to learn R. No Problem!



Remember: Curves Are Beautiful!

2. R has a command-line interface. There is an optional graphical interface called R Commander. The R Supplement for IPS is a document that has more information on using RCmdr. We will be using RStudio for the most part.

3. R is relatively new, which means there are not a lot of people using R in introductory courses but R is so powerful R-specific resources are growing rapidly, and is becoming the world standard in business and industry. Why? Once again: ***R is free! And if it's for free, it's for me!***

DOWNLOAD R

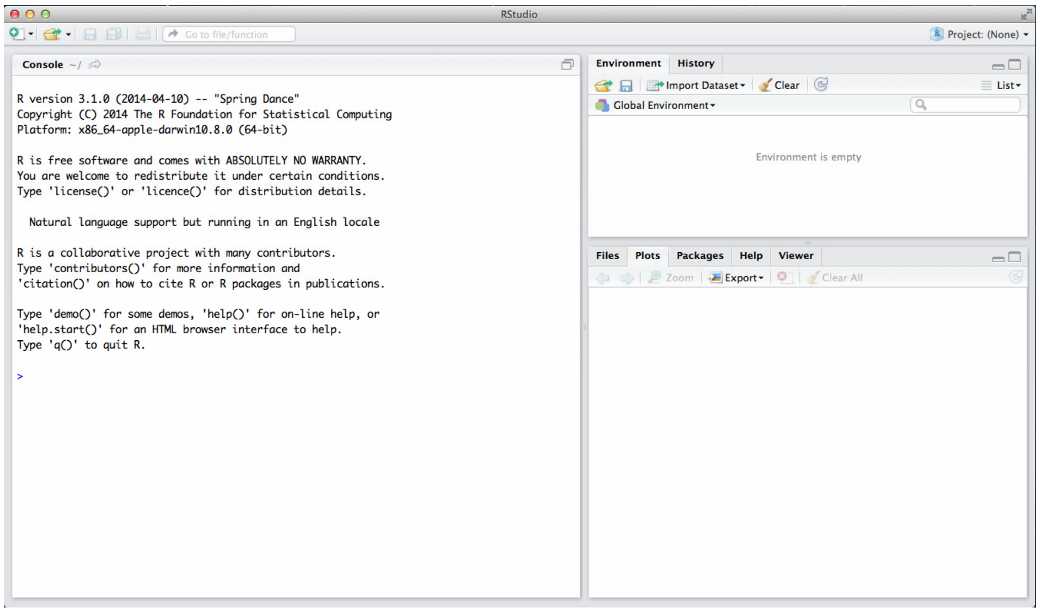
Go to the R home page http://www.R-Project.org and download and install the R software for your platform. Information on doing this can be found at the website.

RSTUDIO

The goal of this lab is to introduce you to R and RStudio, which you'll be using throughout the course both to learn the statistical concepts discussed in the texbook and also to analyze real data and come to informed conclusions. To straighten out which is which: R is the name of the programming language itself and RStudio is a convenient interface.

Go to http://www.rstudio.com/products/rstudio/download/ and download RStudio for your platform, either Windows or Mac.

As the labs progress, you are encouraged to explore beyond what the labs dictate; a willingness to experiment will make you a much better programmer. Before we get to that stage, however, you need to build some basic fluency in R. Today we begin with the fundamental building blocks of R and RStudio: the interface, reading in data, and basic commands.



```
R version 3.1.0 (2014-04-10) -- "Spring Dance"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

The panel in the upper right contains your workspace as well as a history of the commands that you've previously entered. Any plots that you generate will show up in the panel in the lower right corner. The panel on the left is where the action happens. It's called the console. Everytime you launch RStudio, it will have the same text at the top of the console telling you the version of R that you're running. Below that information is the prompt. As its name suggests, this prompt is really a request, a request for a command. Initially, interacting with R is all about typing commands and interpreting the output. These commands and their syntax have evolved over decades (literally) and now provide what many users feel is a fairly natural way to access data and organize, describe, and invoke statistical computations.

Lab 0: Introduction to R

This section will begin a very gentle introduction to plotting in R. The goal is to show how one can draw the plot of a function using R and annotate the resulting plot. The tutorial is easy to follow and it gives you a nice overview of the plotting capabilities of R.

R as a Calculator

Start R on your system. This will open R's interactive shell. Let's start by performing a basic calculation in the shell. Enter the expression `2+2`, then press the Enter key. The result follows.

```
> 2+2
[1] 4
```

Sweet! R can add two and two! Let's try something a bit harder, such as *sin π /2*.

```
> sin(pi/2)
[1] 1
```

Good start! Let's produce a list of numbers, storing the result in the vector **x**.

```
> x=0:5
```

We can see what's stored in the variable **x** by typing its name and hitting the Enter key.

```
> x
[1] 0 1 2 3 4 5
```

Let's add 3 to each entry in **x**.

```
> x+3
[1] 3 4 5 6 7 8
```

Let's square each entry in **x**.

```
> x^2
[1] 0 1 4 9 16 25
```

These calculations are pretty typical of the way in which R deals with lists of numbers. Whatever you do to the list is applied to each element in the list.

Let's do something a bit more substantial. First, let's create a list of numbers using R's `seq` command.

```
> x=seq(0,2*pi,by=pi/2)
> x
[1] 0.000000 1.570796 3.141593 4.712389 6.283185
```

Note the syntax, `seq(a,b,by=increment)` produces a list of numbers that starts at `a`, increments by `increment`, and finishes at `b`. Hence the command `x=seq(0,2*pi,by=pi/2)` produces a list of numbers that starts at zero, increments by π /2, and stops at 2 π. Now, let's take the sine of each number in the list stored in **x**.

```
> sin(x)
[1]  0.000000e+00  1.000000e+00  1.224647e-16 -1.000000e+00 -2.449294e-16
```

A little strange, but note the scientific notation. This result is approximately **0, 1, 0, -1, 0**, which is correct. The small errors are due to round-off error.
Let's apply what we've learned to sketch the graph of a sinusoid.

Plotting a Sinusoid

In this first activity, let's try to draw the graph of a sinusoid with the following equation.

```
y = 2 sin (2π x - π /2)
```

In a trigonometry class, the first step would be to identify the amplitude, period, and phase shift. To that effort, we factor out a 2π.

```
y = 2 sin 2π (x - 1/4)
```

Comparing this equation with the more general form, $y = A \sin B (x - \varphi)$, we see that $A = 2$, $B = 2\pi$, and $\varphi = 1/4$. Thus, we have the following facts:

- The amplitude of the sinusoid is $|A| = |2| = 2$.
- The phase shift of the sinusoid is $\varphi = 1/4$.
- The period of the sinusoid is $T = 2\pi /B = 2\pi /2\pi = 1$.

Thus, when we draw the graph of the sinusoid, we should see it complete one full cycle every 1 second, it should bounce up and down between 2 and -2, and it should be shifted 1/4 units to the right. With these thoughts in mind, let's sketch 2 periods of the sinusoid, which means that we should sketch the graph of the sinusoid over the domain [0, 2].

Coding

We begin. We first create a list of domain values and store them in **x**. The following command starts at 0, increments by 0.01, and stops at 2, giving us a large number of points in the domain [0,2].

```
> x=seq(0,2,by=0.01)
```

We now need to evaluate the sinusoid $y = 2 \sin 2\pi (x - 1/4)$ at each point of the vector **x**. This is a simple matter in R.

```
> y=2*sin(2*pi*(x-1/4))
```

The above command evaluates the sinusoid at each point in the vector **x** and stores the result in the vector **y**. All that remains is to plot the points in the vector **y** versus the points in the vector **x**.

```
> plot(x,y)
```

The result of this last command on our system is shown in Figure 1. Results may differ on different systems depending on the windowing system used by your version of R.

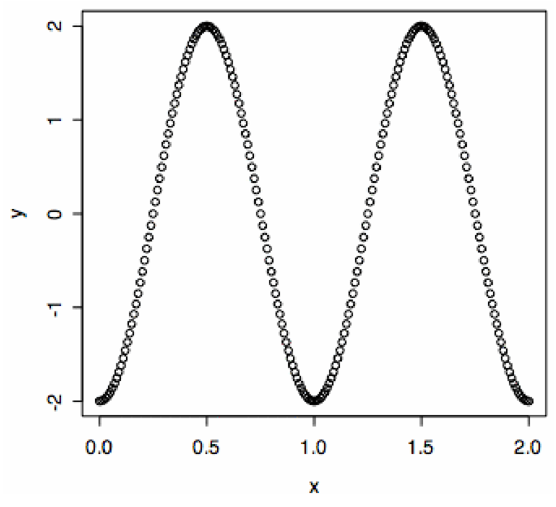


Figure 1. *A plot of y = 2 sin 2π (x - 1/4) on the interval [0, 2].*

Note that R's default behavior is to plot the individual points. We can change the "type" of the plot as follows.

```
> plot(x,y,type='l')
```

The result is shown in Figure 2. Note that the points are no longer marked and we have a "smooth curve" instead. Actually, what's going on behind the scenes is each consecutive pair of points is being joined with a small line segment. Because we've plotted a "lot of points," the result has the appearance of a "smooth curve."

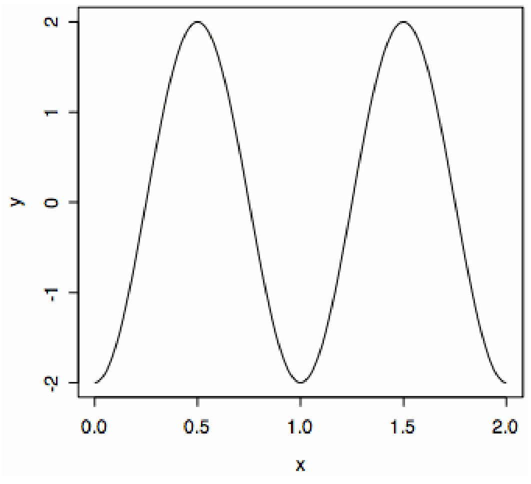


Figure 2. *Changing the type, giving the appearance of a "smooth curve."*

You can add axis labels and a title with the commands that follow below. After typing the first line, namely `plot(x,y,type='l',` (that's a lower case "L") press the Enter key. The plus sign (+) on the next line is R's "line continuation" character and is automatically supplied by R's interactive shell. Without the "line continuation" character, lines would be too long to include in this document.

```
> plot(x,y,type='l',
+ xlab='x-axis',
+ ylab='y-axis',
+ main='A plot of a sinusoid.')
```

It's somewhat difficult to type a number of consecutive lines correctly, so if you make a mistake you will need to start over. However, you can use the "up-arrow" on your keyboard to replay lines you've already typed. Pressing the "up-arrow" a number of times take you back through the "history" of lines you've entered in R's interactive shell. When you get the line you want, press Enter, or edit the line and then press Enter.
The result of this code is shown in Figure 3. Note the addition of axis labels and a "main" title.

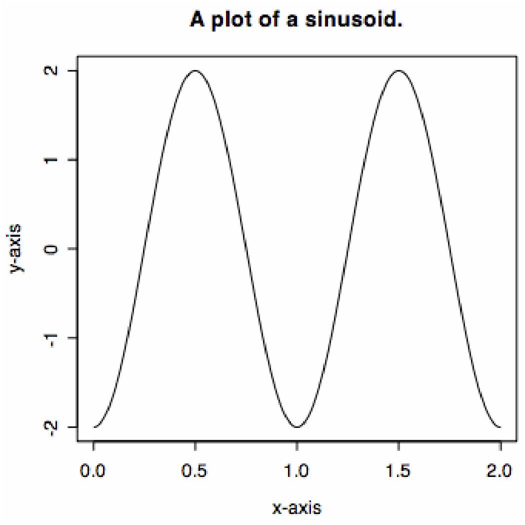


Figure 3. *Adding axis labels and a title to the figure..*

Enjoy! I hope you've enjoyed this small introduction to plotting in R. To find out more about R's `plot` command, type the following command in R's interactive shell.

```
> ?plot
```

This will open a help file on the `plot` where you can read more about this powerful command.