

# 1170:Lab1

August 21st, 2012

## Goals For This Week

In the first week we will just be getting to know R. I'm going to assume that none of you have ever used anything like this before so we'll start out slow. Specifically, we should all be able to:

- Log in and Open R
- Use R to perform arithmetic
- Store variable names
- Plotting functions
- Save your work and submit it

## 1 Log in and Open R

To open R on the campus computers right click on the desktop and select *open terminal* from the menu. Then type

```
>R
```

And now you're using R! Excellent!

## 2 Using R like a calculator

R is an extremely adaptable program. It comes with many built-in functions and you can download tons of extensions that have been written by other people. At it's heart, though, R just loves to do arithmetic. Try to enter the following commands.

```
> 2+3  
> 2-3  
> 2^23
```

R also has a lot of built-in mathematical functions.

```
> sin(3)  
> exp(3)  
> log(3)
```

The *log* function is calculating the natural log, by the way.

### 3 Storing Variable Names

You should now see that we can use R like a calculator. If we want to do anything more complicated than that, however, we're probably going to need to assign values to variables for later use.

When you assign a variable name you have two options. You may either use the familiar equals sign(=) or the assignment operator (< -). For beginner programmers, I recommend using the assignment operator as it's a little more intuitive. Here's a basic example:

```
> a=3
> b=2
> a=b
```

After entering these commands, what do *a* and *b* equal? We can enter the following to find out.

```
> a
[1] 3
> b
[1] 3
```

This tells us that both variables are equal to 3 by the end. If this is confusing, let's look at it again but with the assignment operator instead of the equals sign.

```
> a<-3
> b<-2
> a<-b
```

There are two things to note about this sequence. The first is that we have defined *a* in two different ways. The most recent definition always takes precedence in this case. The second thing to note is that in the third line, the value for *b* is given to *a* (note the direction of the arrow) not vice versa.

Finally, let's see what happens when we change the definition of *b*

```
> b<-5
```

Now you might wonder whether or not *a* now equals 5.

```
> a
[1] 3
```

Nope, it still equals 3. The value of *a* will not change unless you explicitly tell R to change it. It's important to keep these facts in mind when you find yourself frustrated at R. Also, you often want to do things like this

```
> a<-1
> a<-a+1
> a<-a+1
```

One more nice thing about R is that variable names can be anything. It's generally a good idea to use longer names so that it's easier to understand what you are doing.

```
> N_usstates<-50
> feet_to_inches<-12
> ultimate_answer<-42
```

## 4 Plotting Functions

Let's say that we want to plot a sweet function like  $y = f(x) = e^{\sin(x^2-3)}$ . The first thing we need to know is the values of  $x$  for which we want to compute  $f(x)$ , i.e the domain. In R, the domain will be a list of numbers, a vector. Let's suppose that we want to plot the function between 0 and 10. We can use the 'c' command below to create the a vector with any numbers we want.

```
> x <- c(0,1,2,3,4,5,6,7,8,9,10)
```

Now we can calculate the y-values.

```
> y <- exp(sin(x^2-3))
```

Then we use the builtin plot function. The plot function requires at least 2 inputs: a list of  $x$  values and  $y$  values. We already have these defined as  $x$  and  $y$ .

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

This just looks like some randomly scattered points. We probably need to plot some more points in order to visualize this function. Instead of just taking integer values of  $x$ , let's try to plot the function at intervals of .01. It would be a lot of work to input all of those numbers manually, so we use a builtin shortcut.

```
> x<-seq(0,10,.01)
```

The seq command (think "sequence") takes three inputs: the low value, the high value and the spacing in between numbers. Enter in  $x$  to see the whole sequence.

```
> x
```

Now enter re-enter the plot command. It should now give an error. The vectors  $x$  and  $y$  are of different lengths! We have to update  $y$  the same way as before to calculate it at all of our intermediate points. Re-entering the plot command should now display a lot more points that seem to lie on a curve. R will connect these points with lines for you

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

You can add a title and label the  $x$  and  $y$  axis like so.

```
> title(main="This crazy function",xlab="x",ylab="y")
```

## 5 Saving

R can export graphics in many different formats, I'm going to recommend using jpegs.

```
> dev.copy(jpeg, 'Lab1plot1.jpg')
> dev.off()
```

This just saved the current graph in the working directory. To figure out where that is type.

```
> getwd()
```

You can now paste this into the word processor of your choice. The math departments computers have Open-Office Writer. Find it in the launch menu under *applications* and then office.

Once you've opened office, you can insert your figure by choosing *insert-picture-from file*.

## 6 Assignment for this week

1. Plot the function  $f(x) = e^{\sin(x^2-3)}$  between 0 and 10 in the manner described above. Hopefully you've already done this!
2. Suppose that a colony of bacteria grow according to  $B_1 = f(t) = 10000 \frac{e^t}{100+e^t}$ . There are two other colonies that grow according to  $B_2 = f(2t)$  and  $B_3 = 2f(t)$ . Create plots of each of these population for times between 0 and 15. Describe the differences between the plots. Comment on how these bacterial colonies might differ from each other. Label the x-axis "time" and the y-axis "number of bacteria"