

1170:Lab3

September 17th, 2013

Goals For This Week

In this third lab we will further familiarize ourselves with R.

- Create a .R file script
- Learn some basic debugging and ways in which things can go wrong.

1 Writing Commands in a .R text file

After working from the command prompt screen in R for the last couple of weeks it should be clear that there are some annoyances that arise.

- In Unix it is difficult to save your commands
- When something goes wrong, or when you need to slightly change a command you may have to reenter many commands to avoid errors
- It can be difficult to determine which command it was that you entered that R first had a problem with

One of the ways that we can try to overcome this is to have all of our commands saved and stored outside of R.

To do this we will create a .R file type in a text editor, gedit.

To start first open your R folder in your home file network. Now somewhere in the open space in the folder right click and select "create new file".

You should now see a new empty file in the folder. Click on it and name it Lab3.R

Now open the file, when you do it should open into a program called gedit. gedit is a text editor like notepad in windows.

Now in our lab3.R file we can type the commands that we would like to enter in R. Gedit is not R, so we will need to then copy and paste our commands into R in the terminal. To do this first minimize gedit, then right click on the desktop and open a new terminal. Then type R and hit enter to open R.

To test this out let us pull back up our Lab3.R file and enter the following commands:

```
x<-seq(1,5,.01)
y<-x^2
plot(x,y,type='l')
```

Now copy and paste these lines from the .R file into R in the terminal. You can either do it one line at a time in descending order, or all at once.

Now to see the power of using a script let us change our domain of interest. Now we want to see the plot of $y = x^2$ from $x=[4,10]$.

To do this all we need to do is go back to our Lab3.R file in gedit and change the line where we defined x to:

```
x<-seq(4,10,.01)
```

Now copy and paste all 3 lines back into R in the terminal. Take a moment and think about how we would have had to do that if we only used the command prompts in the terminal.

2 Errors in R

Occasionally we will confuse R by either having a typo in our code, entering commands in an improper order, or by using syntax that R does not understand. Whenever we have a problem in our R code, we will get some sort of error response from R. When this happens we want to be able to determine what went wrong in a systematic way.

2.1 Typos

By far the most common type of error that we will encounter in R stems from a typo. There are many different ways that we can confuse R with typos:

- Imbalanced numbers of parenthesis
- Commas or Periods out of place
- Missing multiplication signs between numbers and variables
- Mistyping either a variable or a command

2.2 Improper Command Order

Sometimes we will properly enter the command itself but do so out of order so that R isn't sure what we mean

- Defining an output before defining the input variable
- Using the lines command before an original plot has been created
- Plotting a variable that has not been defined yet

2.3 Non-Mathematical Commands

Sometimes we tell R to do something properly, and in the correct order, but the result of which is non-mathematical

- Taking the square root of a negative number, or a function which goes negative
- Dividing by 0
- Taking the log of a function or number which is less than or equal to 0.

3 Debugging in R

These compose the common mistakes that we see in programming. When you have a problem with your code in order to debug we want to go through the following mental check-list:

- Did I correctly type my command in to the interface
 - If I am using a function are all the variables which the function uses properly defined?
 - Do I have the correct number of parentheses, or any commas out of place?
 - Do I have a typo in the command, did I use the command that I meant to?
 - Did I use the right variables in the right order for this command?
 - If I am plotting, are my inputs and outputs the same length?
- Was my command entered at the right time or in the right order?
 - If I am defining a variable (y) in terms of another variable (x), have I already defined what x is?
 - If I redefined my input did I also redefine my output?
 - If I am plotting, have I already defined both my input and output variables?
 - If I am using the lines command, do I already have a plot?
- Am I making R do something non-mathematical
 - Was there a NaNs error message? Then yes, you tried to make R break math

4 examples

Consider the following set of commands entered into R:

Example 1

```
x<- seq(1,5,.01)
y<- seq(0,4,.001)
plot(x,y)
```

What kind of error do you expect there to be? Why did we get this error? How could we fix this problem?

Example 2

```
y<- x+5
x<- 7
y
```

What kind of error do you expect from this code?

Example 3

```
x<- seq(4,5,.001)
y<- 4x-7
plot(y,x)
x<- seq(5,8)
z<- (3*y)+4)
line(x,z)
```

There are a lot of problems with this section of code, how many can you find?

5 Assignments for the Week

Consider the following problem from last week:

For the ball problem from section 2: The ball reaches its maximum height after (20/19.6) seconds. Show a progression of secant lines to approximate the velocity of the ball when it is at its maximum height. Include in your answer the graph of the function with of all of your secant lines (at least 2)

The following is a hypothetical attempt at solving this problem in R:

```

x<- seq(10,-10,.01)
y<- function(x){slope*(x-x1)+x1}
x1<- .5
y1<- f(.5)
s<- (y-y1)/(x-x1)
plot(x,y)
x2<- .75
y2<- y(x2)
line(x2,y2)

```

As you can see there are numerous things that are going to go wrong with this code.

For this weeks assignment go line by line through the code and attempt to fix the errors so that at the end you get a working code for 2 secant lines. Be sure to reference Lab2 as an aid to see how we had done the example problem. In your solutions include a .R file of your corrected code as well as a written statement outlining what was wrong with the original code and how you fixed it. To get you started here is the fix for the first line:

```

x<- seq(-10,10,.01)

```

Explanation: the order in the sequence was incorrect for a domain, we want our domain values to move from left to right or from -10 to 10. Additionally there is a typo with how the command was entered, it is impossible to go from positive 10 to negative 10 by a $+.01$ increment.