R: Scripts, Getting Data into R, Simple Lienar Regression, & Ripley's K

Michael L. Treglia

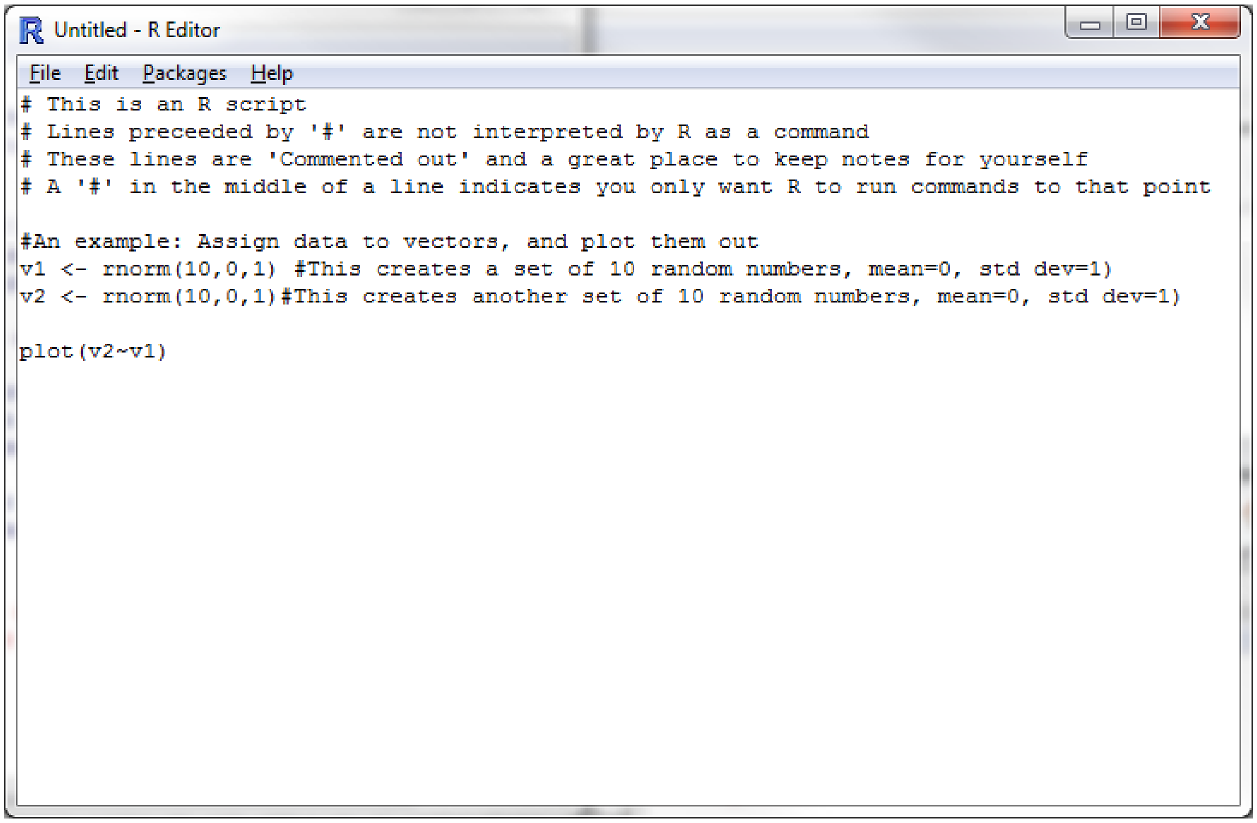
Material for Lab 3 of Landscape Analysis and Modeling, Spring 2015

This document is available online, with active hyperlinks, at:

In the first lab using R, we simply used the command line as an interactive interface. As you could see, you could carry out the basic functions quickly and easily in that way, though saving your commands and making your work repeatable might be difficult. Most simply, it seems you could copy and paste all of your commands into plain text editor (e.g., Notepad and Text Edit are built-in editors on Windows and Macs, respectively), although this is still cumbersome. Isn't there a better way?!?!

YES!!! The key to this is "scripts" - these are sets of code that you type, and can re-run as you desire, either a line at a time, or all-at-once. To create a new script in R, use the File menu at the top of your R window, and select "New Script".

A new window will appear within R; you can type your commands here, and use [CTRL]+R to run an individual line where your cursor is, or all lines that you have selected with your mouse. You can also run all of the code at once, by using the "Edit" menu, and selecting "Run All". You can save this file as a '.R' file, which you can open up from within R, or edit using a plain text editor of your choice. A helpful aspect of this is you can include "comments" in script files, which R will not run as code. To indicate a comment, simply use a '#' before the text. You can do this for an entire line, or for part of a line, after actual R code, as seen below. At this point, everything that you run from the Script will be displayed in the interactive terminal window. (Try copying the code displayed in the image, and run it from your own script window).



```
# This is an R script
# Lines preceeded by '#' are not interpreted by R as a command
# These lines are 'Commented out' and a great place to keep notes for yourself
# A '#' in the middle of a line indicates you only want R to run commands to that point

#An example: Assign data to vectors, and plot them out
v1 <- rnorm(10,0,1) #This creates a set of 10 random numbers, mean=0, std dev=1)
v2 <- rnorm(10,0,1)#This creates another set of 10 random numbers, mean=0, std dev=1)

plot(v2~v1)
```

Okay, lets do some quick examples

```
vector1 <- c(1,2,5,8,2)
```

Once you have used the assignment operator, R has information loaded into memory. You can see what R has loaded into memory by using:

```
ls()
```

Typing any item listed by that function will display that entire dataset.

Note that the 'c' before the list of numbers above is actually a function. So in this case, the function is being used to define the assignment (i.e, what is being called vector1). Functions precede opening parentheses, and have various arguments, depending on the specific function. To see specifics for a function, simply type help([function name]), as with this:

```
help(c)
```

The help file may seem a bit elusive at first, but just read the description carefully - as you get more familiar with R, it will start to make some sense. Furthermore, help has useful examples at the end of each section - simply scroll down to the end of the help page for a specific function and you will find them.

To see what is actually stored as the character or name, simply type the assignment

```
vector1
```

Furthermore, you can conduct mathematical operations on the assigned information, and get summary statistics. Try the following and see what happens:

```
mean(vector1)
vector1 * 4
summary(vector1)
length(vector1)
```

Question 1: What does the function 'length' do?

You can work with multiple vectors together too (try these and see what happens):

```
vector2 <- c(3,4,2,6,9)
vector1 * vector2
vector1 / vector2
cor(vector1,vector2)
```

Question 2: What does the function 'cor' do, and what value does it return in this case?

To store the result of an operation as a new entity, simply use an assignment:

```
vector3 <- vector1 * vector2
```

An important thing to be aware of is that data can be in various forms - it can be stored as integers, as strings (i.e, non-numerical characters), and factors (i.e., categories). You can define these yourself, though R has automatic interpretations. To figure out how R sees data, use the 'str' command (which stands for Structure). For example,

```r
str(vector1)
```

The result of that command should be 'num [1:5] 1 2 5 8 2'. If you are dealing with multiple variables in a dataset, as you will later in this exercise, the results will also list the variable names.

This indicates that it vector1 is numeric, contains 5 pieces of information (1 through 5), and lists the data. If it were a much longer vector, it would only list the first few pieces of information.

You can also plot data in various ways. For example, the simple commands 'boxplot([vector name]' and 'hist[vector name]' create boxplot and histograms respectively.

You can also plot two variables together.

Question 3: using the help for the function 'plot', produce a plot of the variables vector1 and vector2 together, and copy and paste it into your results.

There are some datasets you can play around with, built into R. To see what they are, type the command below and hit enter/return.

```r
data()
```

Load the 'iris' dataset by typing

```r
data(iris)
```

You can view the beginning of the dataset (i.e, the first few rows) using the 'head' function, and the end of the dataset using the 'tail function'. To view only a single column, you could use iris$[columnName]. You can also combine these, to view the beginning or end of a single variable in a dataset.

```r
head(iris$Sepal.Length)
```

*Note that spaces are not allowed in column names in R - they are replaced with periods. This is reflected when you view the dataset, and needs to be maintained when you use the data in functions.

Now work with this dataset to complete answer the remaining questions/complete the remaining tasks.

Question 4: Inspect the structure of the iris dataset. What variable is recognized by R as a Factor variable? Bonus Question: Can you figure out what categories exist for this factor variable?

Question 5: Make a histogram for Sepal Length, and paste it into your results. Here's a hint: to call on a single column of a dataset, you need to type 'datasetName$columnName'

Question 6: Use the 'subset' command to create a subset of the data, where the only species represented is 'setosa' (you can use 'head' and 'tail' to inspect the beginning and end of the resulting dataset, or view the entire result by simply typing the new name of the resulting dataset)

Question 7: For the entire iris dataset, calculate the correlation coefficient between Sepal Length and Petal Length.

Question 8: Plot the relationship between Sepal Length and Petal Length for the entire dataset. As an extra challenge, see if you can use symbols other than the default

Question 9: What is the maximum value for Petal Length across the entire dataset?

Question 10: For the subset of data that you took earlier (Question 6), create a boxplot of Sepal Width.