Notes on Reading Data with R

I have decided that meeting once every two weeks is too slow a pace for learning R. So in between these sessions, I will send out some notes to help you along. I hope to do this several times a week. I would appreciate feed-back on what to cover next and of course I am willing to answer questions.

In our meeting it seemed that nearly everyone was interested in working with data. Because R is a complete programming language, it is possible to use R for simulations and modeling when you have no data, but for us, reading data seems like a good place to begin.

The “snook.r” script that you received from Lea has about simplest read statement you can use. To execute this function, first you start R by clicking the R icon on your desktop or toolbar . You should get something that looks like this –



Colors may vary due to customization.

Also open the Mat.R file with a text editor. I use boxer for text editing and it looks like this:



In you editor, you will need to revise the path name leading to the file “MAT\_06-08.csv". Remember that R uses this “/” folder delimiter while windows uses “\”. So if you cut and paste the path name from somewhere, these delimiters will have to be reversed.

Now using copy and paste, drop the 1st line of code in the R console window and press return. Hopefully your results will look like this:



The read.csv function assessed the data in the external file and created an R data-frame called ‘mat’. A data frame is a 2-d object that can be visualized as a spread-sheet. Typically the rows are observations and the columns are variables. You can refer to the rows and columns by name or by number. So for example, type “mat[2,5]” to show the contents of the 2nd row and 5th column like this:



or you can access the same datum by typing “mat[‘2’,’Month’]. The character string ‘2’ is the row name and the character string ‘Month’ is the column name. In computers, the character string ‘2’ and the number 2 are represented in memory by a difference sequence of bits even though in print they may look the same. Packages such as excel tend to blur this distinction.

To see the first 10 rows of data type ‘mat[1:10,]’. As before, mat is given with 2 indices inside the [,]’s. In this example 1:10 represents the sequential numbers 1, 2, . . . 10 which are the row numbers. Leaving the column subscript position blank means to show all columns.



In the read.csv function call given in full here:

mat <- read.csv("c:/Projects/CBP/Rcourse/MAT\_06-08.csv",header=TRUE)

the contents of the parentheses after read.csv contain the arguments of the function. The arguments are the information that the function needs to do it’s job. The first argument is a character string "c:/Projects/CBP/Rcourse/MAT\_06-08.csv" which tells read.csv where the file is. The second argument, which is separated from the first by a comma, is header = TRUE. This tells read.csv that the first line in the data file contains the names to use for the columns of the data frame. The “<-“ is a symbol for the assignment operator in R. Thus the output of read.csv is assigned to the object “mat”.

You should check out the other potential arguments for the read.csv function by involking R-help

Simply type (or copy and paste)

?read.csv

in the R-console.

You should get a new window something like this:



Note that this actually takes you to help for a function called ‘read.table’ and then shows ‘read.csv’ as a simplified version of read.table. Read through the options here and see if it is clear how to use them. There will be stuff here that you won’t understand, but hopefully as we progress, more will become clear.

Here is a list of problems I have had reading data – perhaps knowing about these will save you some time:

1. Not having the same number of columns on each row – solution, add delimiters until they are all equal.
2. If the data contain an unclosed quote, e.g. station = St. Mary’s River, this gets read.table very confused. Solution: get rid of quote in data, or redefine the quote argument of the read.table function, eg quote = “”, will define no quoting characters.
3. If the data contain a “#”, this will be interpreted as a comment indicator and the remainder of the line will be ignored (sometimes cause of problem 1). To fix this, use comment.char argument set to null characters, e.g, comment.char=””.
4. By default, R will redefine all character string variables as factors. Factors are a somewhat complicated variable type that we will deal with in the future. I always use the stringAsFactors=FALSE when I am reading data and convert strings to factors on an as needed basis using the as.factor() function.