An Introduction to Data Analysis with R

By Vivienne Foroughirad & Jillian Wisse

What is R?

www.r-project.org

R is a free and open source program for conducting data analysis, including data manipulation, statistics, and data visualization.

What is RStudio?

http://www.rstudio.com/products/rstudio/

RStudio is an integrated development environment (IDE) for programming in R. If R is like notepad, then RStudio is Microsoft Word.

Getting started with R

The instructions you give R are called commands. The basic approach to using R is to type a command into the console and hit enter—R evaluates what you typed and prints the result.

```
> x<-2+2
> x
[1] 4
```

The arrow "<-" is the assignment operator. Operators are characters like "+", "/", "&" that stand for a specific mathematical or logical function. The direction of the arrow can be switched, or substituted with an equal sign "=" (but then order matters!)

For example:

```
> x=2*3
> x
[1] 6
```

or

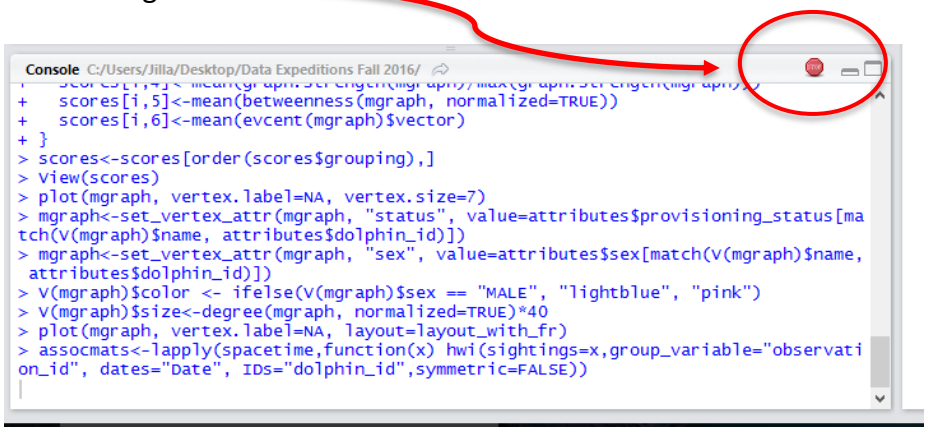
```
> 12/4->x
> x
[1] 3
```

but not

```
> 2+3=x
Error in 2 + 3 = x : target of assignment expands to non-language
object
```

When writing R code, it is usually best to type all code into a text editor, and then selectively run code when needed. That makes it easiest to go back and modify code and reuse code when needed. RStudio comes with a built in text editor called "R Script" which recognizes and formats R code and let's you easily transfer it to the console. Code can be transferred to the console by either copying and pasting, by hitting the "Run" icon at the top of the R script, or hitting Control + Enter. The latter two methods will either run the line that your cursor is on, or can run a highlighted section.

If the code runs successfully, you will see the code entered on your screen above the prompt ">" sign. If you see stop sign on the top right of the console, it means the code is still running.



If you see a plus sign instead of the ">" prompt, then the program is waiting for you to enter the rest of the line of code (often happens when you're missing a closing parenthesis, etc).

```
> x<-(3+5
+
```

Debugging hints
R is case sensitive, so X and x will be recognized as two different variables. R can also occasionally have trouble interpreting different fonts if you copy and paste from other sources. Quotation marks can be especially sensitive.

When programming in R there are several data types you should be aware of. Some of the most basic types are character, factor, numeric, and integer data. Find out the data type of a variable using the class( ) function.

```
> x<-"dolphin"
> class(x)
[1] "character"

> x<-4
> class(x)
[1] "integer"
```

```
> x<-3.3
> class(x)
[1] “numeric”
```

Factors represent categorical data. There are also more specific data types such as date and time objects. It is important when using operators or functions on different variables to make sure that they are compatible data types.

```
> x<-“3-Mar-13”
> y<-5

>class(x)
[1] “character”


>class(y)
[1] “numeric”

>x+y
Error in x + y : non-numeric argument to binary operator

>x<-as.Date(x, format=c(“%d-%b-%y”)
>x+y
 [1] "2013-03-08"
#Five days have been added
```

You can add comments to your code by prefacing the line with the pound sign ‘#’. R will then know that these are notes and won’t try to run them as code. These basic data types can be combined into datasets such as vectors, matrices, data frames, and lists. A vector is a set of a single data type, denoted by c( ).

```
#Create a vector
>x<-c(1,2,3)
>x
[1] 1 2 3

>y<-c(“Cat”, “Dog”, “Rabbit”)
>y
[1] "Cat"    "Dog"    "Rabbit"
```

Various operators and functions can then be used on vectors.

```
>x<-c(1,2,3)
>x
[1] 1 2 3
>x*3
[1] 3 6 9
>mean(x)
[1] 2
```

Vectors can be combined into tables called data frames.

```
>x<-c(1,2,3)
>y<-c("Cat", "Dog", "Rabbit")
>z<-data.frame(x,y)
>z
  x      y
1 1    Cat
2 2    Dog
```

```
3 3 Rabbit
```

Indexing can be used to select a specific value from a vector or data frame.

```
>x<-c(4,5,6)
>x[2] #Selects the 2nd element from x
5
```

A list is a grouping of data objects that may or may not be of the same type

```
>q<-list(x, z)
>q[[1]] # Use double brackets to access list elements
 x
[1] 1 2 3
```

Loops are programmatic statements that allow for repeated actions, or iterations.

```
>foo<- seq(1, 100, by=2) # Create a vector of odd numbers
>foo.squared<-NULL # Create an empty object
>for(i in 1:50) {
foo.squared[i] = foo[i]^2
}
>
>foo.squared
[1]    1    9   25   49   81  121  169  225  289  361  441  529
[13]  625  729  841  961 1089 1225 1369 1521 1681 1849 2025 2209
[25] 2401 2601 2809 3025 3249 3481 3721 3969 4225 4489 4761 5041
[37] 5329 5625 5929 6241 6561 6889 7225 7569 7921 8281 8649 9025
[49] 9409 9801
```

R packages

Packages are the fundamental units of reproducible R code. They include reusable R functions, the documentation that describes how to use them, and sample data. In this lab we will be using two pacakges, *ggplot2* and *plyr.* To install:

```
>install.packages(c("ggplot2", "plyr"))
>library(ggplot2)
>library(plyr)
```

That's it! You now have the tools to explore and analyze any number of data sets.

Good Luck!