Introduction to R

## Learning Objectives

* Familiarize participants with R syntax
* Understand the concepts of objects and assignment
* Understand the concepts of vector and data types
* Get exposed to a few functions

## Creating objects

You can get output from R simply by typing in math in the console

3 + 5  
12/7

However, to do useful and interesting things, we need to save the numbers as something we can remember and use later. This is known as assigning *values* to *objects* or *variables*. To create an object, we need to give it a name followed by the assignment operator <- and the value we want to give it:

weight\_kg <- 55

*Note*: There is a subtle difference between <- and =. Both will work to assign operators, but traditionalists preferr to use <- because = is also use to specify parameters in functions (which we'll learn about later).

Objects can be given any name such as x, current\_temperature, or subject\_id. You want your object names to be explicit and not too long. R is case sensitive.

When assigning a value to an object, R does not print anything. You can force to print the value by using parentheses or by typing the name after you've assigned it.

(weight\_kg <- 55)  
weight\_kg

Now R has weight\_kg in memory (or in the *environment*). Find it in the environment window and see it's value. Now that weight\_kg has a value, we can do arithmetic with it. For instance, we may want to convert this weight in pounds (weight in pounds is 2.2 times the weight in kg):

2.2 \* weight\_kg

This will evaluate the line of code as if you had typed 2.2 \* 55

We can store this value as a new variable with the assignment operator

weight\_lb <- 2.2 \* weight\_kg

This evaluates the code on the right hand side of the <- and stores a copy of that value in an object called weight\_lb

We can change the value of a variable's by assigning it a new one:

weight\_kg <- 57.5  
(2.2 \* weight\_kg)

Note that assigning a new value to one variable does not change the values of other variables that once depended on it.

What do you think is the current content of the object weight\_lb? 126.5 or 121?

### Exercises

What are the values after each statement in the following?

mass <- 47.5 # mass?  
age <- 122 # age?  
mass <- mass \* 2.0 # mass?  
age <- age - 20 # age?  
massIndex <- mass/age # massIndex?

Note the use of the pound sign / hashtag / number sign #. This is the R symbol for comments. Nothing after this symbol will be evaluated as R script. This is very useful for when you want to write notes to yourself in plain English (or your language of choice!).

## Vectors and data types

The power from R is that it is an object oriented language, meaning that we can work with object types that are more complex than just single values.

A vector is the most common and basic data structure in R, and is pretty much the workhorse of R. It's a group/list of values, mainly either numbers or characters. You can assign this list of values to a variable, just like you would for one item. For example we can create a vector of animal weights:

weights <- c(50, 60, 65, 82)  
weights

The advantage here is that you can do arithmetic with the entire vector at once, similar to copying an equation in Excel:

lb\_weights <- 2.2\*weights  
lb\_weights

A vector can also contain characters:

animals <- c("mouse", "rat", "dog")  
animals

There are many functions that allow you to inspect the content of a vector. length() tells you how many elements are in a particular vector:

length(weights)  
length(animals)

class() indicates the class (the type of element) of an object:

class(weights)  
class(animals)

You can add elements to your vector simply by using the c() function:

weights <- c(weights, 90) # adding at the end  
weights <- c(30, weights) # adding at the beginning  
weights

What happens here is that we take the original vector weights, and we are adding another item first to the end of the other ones, and then another item at the beginning. We can do this over and over again to build a vector or a dataset. As we program, this may be useful to autoupdate results that we are collecting or calculating.

We just saw 2 of the 6 **data types** that R uses. This list is:

* character for words
* numeric for decimals
* logical for TRUE and FALSE (the boolean data type)
* integer for integer numbers (e.g., 2L, the L indicates to R that it's an integer)
* complex to represent complex numbers with real and imaginary parts (e.g., 1+4i) and that's all we're going to say about them
* raw that we won't discuss further

Vectors are one of the many **data structures** that R uses. Other important ones are lists (list), matrices (matrix), data frames (data.frame) and factors (factor).

We are now going to use our "surveys" dataset to explore the data.frame data structure.