Mechanics of Data Manipulation in R

RAZ

December 7, 2017

Learning Objectives

1. Students will be able to assign values to objects.
2. Students will be able to recognize and describe the different types of atomic vectors.
3. Students will be able to select desired entries from a vector.
4. Students will be able to identify a function in R, and describe how to pass arguments to a function.
5. Students will be able to predict the outcome of applying mathematical operators and functions to a vector.

POGIL Group Setup:

1. Conceptor (Manager)- In addition to managerial duties - does initial constructions of the what the figure should look like. Draws things, conceptualizes how the data will be presented.
2. Coder/Recorder - the person writing the bulk of the code. Importing packages, annotating code throughout. Recording ideas.
3. Error Checker (Reflector) - In addition to reflector duties, this person responsible for debugging code, using helpfiles, etc.
4. Explainer/Presenter - the person that will do the speaking out. Be able to explain the code and the rationale of each bit to the rest of the class. Should also explain the consensus of the group about what the data show us.

Part 1 - Introduction to objects and vectors - (45 minutes)

In the space below, save a number to an object, and inspect the object

In the space below, divide your object by 2, and add your object to itself

In the space below, generate code that will save the result of multiplying your object by 3 into a new object (REPORT OUT)

In the space below, create a new object, named "weight", and assign the vector provided to this object

```
##Below you see a function, known as concatenate(), or just c(), which takes all of the numbers and puts
c(39, 48, 36, 26, 20, 47, 35, 31, 21, 34)
```

```
##  [1] 39 48 36 26 20 47 35 31 21 34
```

Below, inspect the vector you just made, and perform the same mathematical operations that you used earlier on your vector. Compare and contrast the results of each, record this, and report out.

Make two new vectors named "parasites" and "sex" that contains the values provided on your handout

We discussed them briefly above, but in R, most of the work you will perform uses commands known as "functions". Functions typically are a word followed by parentheses (). One that is very useful is the function

"mean()", which calculates the mean of whatever object you specify as an argument inside the function. In the space below calculate the mean of the parasites vector.

In addition to an argument specifying the target object for a function, most functions have additional arguments that modify the behavior of the function. You can pass additional arguments to a function by separating them with a comma ",". Below is some code to calculate the mean number of parasites, excluding the first and last value. Can you modify the code to exclude the first two and last two values?

```
#mean(parasites, trim=1)
```

You can also save the output of a function to a new object. Below, calculate the mean number of parasites and store it in a new object. Then, uset that object and divide each value of "weight" by the mean number of parasites. REPORT OUT YOUR CODE

Most of the operations you will perform in R (statistical analysis, graphing), use "data frames", which is a fancy way of referring to a collection of columns and rows. There are specifics about the parts of a data frame I'll explain later To make a data frame from a group of vectors is easy! Simply type data.frame() and fill in the names of the vectors in the parenthesis. In the space below, make and save a data frame that contains the "sex" and "parasites" vectors.

Inspect the data frame, and use the functions summary(), str(), and head() on your new data frame.

Sometimes you want to add new things to an existing data frame. The command cbind() can be used to do so. First, add a column containing the vector "weight" to your data frame.

Remember, though, you can do math in R! Add another column to your data frame containing the log values of "weights" by using log() and cbind().

There are many ways to get at different parts of your data frame, once you've made it. One way is to add the "$" modifier after the name of your data frame. Another way is to use brackets with commas and numbers. Try calling parts of your data frame with the dollar sign, and experiment by using [], [1,], [,1], and [1,1] to extract different parts of your frame. REPORT OUT YOUR RESULTS

For your final task, using everything you've learned today, add a column to your data frame which contains a vector of the weight of each entry divided by the number of parasites in that entry. REPORT OUT YOUR CODE