Programming Assignment 1: Bonus Content

SOC Methods Camp

September 3rd, 2019

Your homework this week is not meant to be exceptionally challenging, but everyone is coming into camp with different levels of programming experience. This bonus content is to help make the first homework a little easier by pointing you in the direction of commands you might need or the structure a piece of code should have. The task numbers below correspond with the tasks on your assignment.

It is completely up to you if you want to reference this sheet for help with some, all, or none of the tasks- choose your own adventure! And remember- if you're finding yourself agonizing over a problem for hours, *don't.* Shoot Liv or Katie an email, and if it's late just put aside the assignment, get some sleep, and come in tomorrow morning with what you have done.

A helpful data wrangling dplyr guide can be found here: https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf.

Step zero: loading and examining the data

Task one: You will need the "class" command for this task. To review the types of vectors in R, go back to RStudio Primers: The Basics > Types.

Task two: You will need to use the command *one_of* inside the dply verb *select.*

Task three: In the codebook we can see that 1 = white, 2 = black, 3 = hispanic, and that 4, 5, 6, 7, and 8 are all "other" categories. Therefore the first three categories can keep their same values, and categories 5-8 should all be collapsed under category four. You will need the *mutate* command for this.

Task four: You will need the factor command from base R and the dplyr verb *mutate.*

Step two: logical statements and filtering

Task one: Remember that to subset a table by columns you can use *table_name[, -insert vector of desired columns here-].*

Task two: You will need the dply command *mutate* for this.

Task three: You will need the *filter* and *summarise* commands here. Remember, *summarise* is useful for spitting out a labeled summary table. The format for summarise is:

```
summarise(LabelOfOutput = OperationYouWantPerformed)
```

Task four: This will require filtering as in the previous task.

Task five: Remember that the structure for a ggplot, at its most basic, is:

```
plotname <- ggplot(YourDataHere, aes(x = YourXHere)) +
  geom_GEOMTYPE
```

Step three: look at education as a confounder

Task one: You will need the *group_by*, *summarise*, and *arrange* commands from dplyr.

Task two: Here you'll need the *group_by*, and *summarise* commands.

Task three: Once again you will need the *group_by*, *summarise*, and *arrange* commands.

Task four: You will need the *filter* and *summarise* commands.

Step four: for loops for sampling

For this step, remember that the structure of a for loop is like this:

```
#you need an empty vector for the for loop to fill
empty_vector <- c()

#you will tell R that you want it to do something to each individual i in a
#given number of instances
for(i in 1:length(NumerOfDataRowsOrColumns)){
  #And then inside the squiggly brackets you tell R what you want to do to
  #each i. This is where the "meat" of your for loop goes. At the end of the
  #"meat" section, you usually want to tell R to put some output into the ith
  #position of your empty vector
  empty_vector[i] <- MeatOutput
}
```

Substep one

Use dplyr's *filter* command for this substep.

Substep two

This is where we are writing the "meat" of our loop. The meat of the loop will do two things: remove a row, and calculate a mean. For substep two, just think about how to make R remove the first row of data and calculate a new mean based on that removal.

Substep three

First, think about how to edit the code you wrote in substep two to remove the ith row instead of the first row. Then reference the for loop structure up above to figure out how to put it all together into a for loop that does what you want it to.

Substep four

The *as.data.frame* command will be useful here. Then it's just a matter of making density plots, which you did in step two! To add a vertical line, you will need to look into the ggplot subcommand *geom_vline.*