R Problem Set 1: Loops, Functions, and Matrix Algebra

Your name goes here

Due date: September 12, 2017

Contents

Instructions **1**

Part 1 2

Part 2 3

Instructions

Please print and submit a hardcopy of your completed problem set (the knitted .pdf) to the next lab, 1:30 PM Tuesday, September 12, 2017. *Note*: please do *not* email files. It will be easier to grade your problem sets if given the hardcopy.

Please do the following problems. When you see a line of code like `set.seed(123)`, just leave it. It is to make your assignments more easily comparable from student to student.

How to do this problem set

You will get more out of this if you try the assignment on your own first. If you get stuck, search online for a solution. The best way to do this is to google your error messages, or describe what you're trying to do, along with the letter R. For instance, "how do I create a data frame in R?" or "Error in ‘[.data.frame‘(dat, 4) : undefined columns selected." If you get stuck, team work is permitted, but you must write up your own solutions.

R code goes in code chunks like this:

```r
print("Hello World")
```

```
## [1] "Hello World"
```

They begin with three backticks (the key below the escape key in the upper left part of your keyboard) and {r}. They end with three backticks, so that the markdown compiler knows to go back to printing text, rather than evaluating your code. You may in the future want to pass additional arguments to knitr (the engine that compiles R markdown documents) by including additional arguments in the curly braces. For now, you can use the defaults. A keyboard shortcut for creating a new code chunk is Command-Option-I (Mac) or Control-Alt-I (Windows).

Put your solutions below the questions in this .Rmd file. It is a good practice to periodically compile the document using the "Knit PDF" button. This lets you check on whether the document has any errors in it.

You should also test your code as you are writing it by pressing Command-Enter (Mac) or Ctrl-Enter (Windows). If you have code selected, it will run that block of code. If you do not have any code selected, it will run the line of code where your cursor is.

Part 1

Problem 1: Working with For Loops

The following code will create the initial data frame dat for this problem:

```r
# Create a vector x and assign to it values from -2 to 2 in increments of 0.1.
x <- seq(from=-2, to=2, by=0.1)
```

```r
## Create Data Frame
dat <- as.data.frame(matrix(nrow=length(x), ncol=3))
dat[,1] <- x
dat[,2] <- x + x
dat[,3] <- x * x

colnames(dat) <- c("x", "x.plus", "x.multiply")
```

a) Using a for loop, add a fourth column—"dev"—to your data frame that computes for each row j the average absolute deviation from the mean of each row: $\frac{1}{3}\sum\limits_{j=1}^{3} |x_{ij} - \bar{x}_j|$.

b) Create a vector "months" that contains the first four months of the year.

c) Add a fifth column—"month"—to your data frame that randomly assigns one of the four elements in the vector "months" to each observation. Start your code with the command "set.seed(123)".

```r
set.seed(123)
```

d) Using for loops, compute the means of the first (x) and fourth columns (dev) separately for each month. That is, you will compute eight different values (i.e., first column mean for January, fourth column mean for January, first column mean for February, fourth column mean for February, and so forth). Repeat this exercise with medians.

e) **BONUS**: We have not covered plotting in the first R lab yet. Therefore, this problem is completely optional and please do not worry if you decide not to solve it. However, the help function or the internet will be helpful to solve this problem. Use the "hist()" function to produce a histogram of the dev variable. Next, produce a scatterplot of the x.multiply (vertical axis) against the x variable (horizontal axis). For both plots, add an informative plot title and label both x and y axes.

Problem 2: Writing Functions

a) Load the R Data Frame "dta.Rdata". Figure out what this object is called using the ls() function, use head() to look at the data. It may be helpful to remove other objects in your workspace using rm(list=ls()).

b) Write a function called "average" to take the mean of the variable "independent.variable" in the data frame. This function should take a data frame as an input and return the average value of "independent.variable" in the data frame it is supplied with. Do **not** use R's "mean" function to perform this calculation. Instead, write your own function to do so. What is the mean of the "independent.variable" column in this data frame? Does this value correspond with the result you obtain when using R's canned "mean" function? Use a logic statement to answer this last question.

c) Write a second function, "average.two.obs" to take the mean of the variable "independent.variable" based on only the first two observations in the data frame that is passed to it. This function should return the average value of "independent.variable" based on just these two observations. What is the resulting estimate of the mean you obtain when you use this function on the data frame? Does this value correspond with the result you obtain when using R's canned "mean" function? Use a logic statement to answer this last question.

Problem 3: Combining Loops and Functions to Evaluate Consistency of Estimators

Set a seed of "8989". Use the following code to start the answer the subsequent questions, which draw upon the functions written in Problem 2.

```
#Load Data
load("../data/dta.Rdata")

## set seed
set.seed(8989)
```

a) Write a loop that applies the "average" and "average.two.obs" functions to an increasingly large portion of the overall dataframe. Specifically, apply your functions to every sample size between 10 and 500, in increments of 10. That is, start by applying the functions to the first ten rows of the dataframe only and save the resulting averages. Then apply the functions to the first 20 rows and save the resulting averages, and so forth until you include the first 500 rows of the data frame. Display the head `head(averages.part2) head(averages.two.obs.part2)` of both vectors created by the loop.

b) Plot the resulting information. On the x-axis plot the sample size used to estimate the mean (e.g., the first mean will be at 10 on the x-axis, the second at 20, and so on). On the y-axis plot the resulting estimate. Rather than points, plot these values as a solid line. On the same plot, graph the estimates from "average.two.obs" on the same plot using a line of a different color. Produce a title and label each of the axes in your plot. How does this compare to the true value we were supposed to get, 25? Draw a horizontal line at 25.

c) If an estimator gets closer to the value it is trying to estimate as the sample it is applied to grows in size, we call it consistent. Do either of these estimators appear to be consistent based on your graph?

Part 2

Introduction

We have already encountered cursory examples of ordinary least squares (OLS) regression. It turns out that as long as our data matrix, X is full rank, then the OLS estimator for β, the vector of slopes of the best-fit line, can be written in matrix form as follows:

$\hat{\beta} = (X'X)^{-1}X'Y$. In words: the inverse of the $X'X$ matrix, multiplied by the transpose of the X matrix, multipled by Y.

Typically, X is a matrix of predictor variables that includes a constant (a column of 1's), and Y is a vector of outcomes. The object $(X'X)^{-1}X'Y$ is a $kx1$ matrix of estimated coefficients—one for each unknown in the model, including the constant. So for example, if we want to estimate the following model:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

where X_1 and X_2 are both columns in the matrix X, $(X'X)^{-1}X'Y$ would return a vector of $k = 3$ coefficient estimates: $\hat{\alpha}$, $\hat{\beta}_1$ and $\hat{\beta}_2$ (we use "hats" here to convey that these are estimates and not the true parameter values; an estimate of ϵ, a vector of errors, is not included in this output vector).

This assignment will focus on the mechanics of applying this estimator in R using matrix operations. In short, you are going to write your own OLS function!

To do this, we are going to have you read in a real, uncleaned data set, clean it as needed, and then apply your OLS function to estimate an OLS model. After working through this, much of the mechanics in 450A should seem slightly less scary, or at least have a ring of familiarity when they arrive. This will also give you insight as to what is going on “under the hood” when we use canned OLS functions in R such as lm().

Problem 1: Pre-processing data

1) Read in the “commoncontent2012.RData” data, which is the raw 2012 CCES data. Relabel this data frame “dd”. Check the dimensions and examine the first few rows of the data. You will notice that all of these variables do not have intuitive names, and some of them contain weird values. So next we will need to pre-process these data.
2) We first want to identify the party of the respondents in these data. Let’s make a new variable (i.e., a new column in our data frame) called “dem” that takes a 1 if a respondent self-identified as a Democrat (see pid3), or said they leaned toward the Democratic party (see pid7others) in a follow-up question, and a 0 otherwise. Do the same thing for Republicans using the same two variables. Hint: the functions table() and class() are useful for determining which values a variable contains and what type of vector it is, respectively.
3) For those labeled “Skipped” or “Not asked” on pid3, code them as NA. For those labeled “Not sure”, “Skipped” or “Not asked” on pid7others, code them as NA as well. How many respondents that identify as Democrats and Republicans, respectively, do you identify in the dataset?
4) Make a new column in dd, age, that is a numeric equal to the respondent’s age in years. Do this using the variable birthyr, which is a factor vector that conveys the respondent’s year of birth. You may need to change the class of birthyr in order to accomplish this. Note that this survey was conducted in 2012. What is the mean age of all respondents in the dataset?
5) Create a new column—“female”—that equals 1 if the respondent is a female and 0 if the respondent is a male using the variable “gender”. What percent of the respondents is female?
6) Using the variable educ, create a column, BA, that equals 1 if the respondent has a Bachelor’s Degree or higher, and 0 otherwise. Be mindful of the class of the original variable. Make sure BA ends up as numeric. How many respondents hold at least a B.A.?
7) Construct a variable obama, that equals 1 if the respondent voted for President Obama, 0 if they voted for someone else, and NA if the did not vote or did not answer the question or are not sure. Use the variable CC410a. What percent of respondents voted for someone *other than* President Obama?

Problem 2: Writing an OLS Function using Matrix Algebra

1) Construct a matrix called X where the columns are: a vector of 1’s of the same length as the variables you just created, as well as the dem, rep, female, age, and BA variables—*in that order*. Make sure the column names remain the same after constructing the matrix; label the column of 1’s “constant”.
2) Construct a *matrix* Y that is just one column, obama. Again, make sure the column name remains the same.
3) Use your X and Y matrices to implement the OLS estimator—$(X'X)^{-1}X'Y$—to estimate the unknown parameters (the constant term and the betas) in the following regression:

$$obama = constant + \beta_1\text{dem} + \beta_2\text{rep} + \beta_3\text{female} + \beta_4\text{age} + \beta_5\text{ba} + \epsilon$$

4) Using what we know about how to write functions and how to perform matrix operations in R, write a function called “OLS.est” that takes as arguments a data frame, a character vector of the names of independent variables, and a character vector with the name of the dependent variable. Have the function subset the data frame to the variables of interest, compute the OLS estimator $(X'X)^{-1}X'Y$, and return

a kx1 matrix of estimated coefficients called “beta.hat”. Make sure that by default the function renders the first column of X a constant vector of 1’s, and give this column the name “(Intercept)” (the constant is often referred to as the intercept, and it is good to practice working with column names). Note: if an observation (a row) is missing on either an X variable or Y, that entire row cannot be included in the OLS model and must be deleted. Make sure your function accounts for this fact. Also, recall that the first column of the matrix of independent variables should be the constant term. You will have to add it inside the function.

Problem 3: Applying your function to actual data

1) Apply your new function to the data frame “dd” (that is, the whole CCES data frame that you pre-processed in Problem 1. Do not alter it any further prior to passing it to the function and have the subsetting to relevant variables occur within the function). Again, estimate the unknown parameters (the constant term and the betas) in the following regression:

$$obama = constant + \beta_1\text{dem} + \beta_2\text{rep} + \beta_3\text{female} + \beta_4\text{age} + \beta_5\text{ba} + \epsilon$$

2) Confirm these estimates are correct by estimating the same regression using the lm() function. Use the ? command or search online for how to use this function. Examples abound.

Problem 4

You will notice that the summary output of the lm() function contained standard errors. These are estimates of the standard deviations that distributions of these coefficients would possess if we took many samples of data and estimated these models many times. In other words, they are estimates of the variability in our estimates of these coefficients given this sample of data. Let’s use matrix operations to estimate these standard errors.

1) Revise your OLS.est function to calculate an additional object, a one-column matrix “e” that is equal to $Y - X\hat{\beta}$. This is a vector of residuals, which are estimates of the errors in the model. Still working inside the function, generate a new object which is equal to the sum of the squares of each of the elements in “e”, which should be a constant. Call this new object e.2 and make sure it is of class numeric. Have the function return beta.hat and e.2. Since you are returning multiple objects, have the output of the function be a list. Use your function to compute the same regression model as before.

2) Revise the function yet again to output a new kxk matrix, “var.cov”, that is equal to $\frac{e.2}{n-k} * (X'X)^{-1}$, where n is the number of observations that were included in the regression, k is the number of estimated parameters, including the constant, and X is the matrix of independent variables included in the regression.

3) Revise your function one last time to output an additional object, a vector called “ses”, that is equal to the square root of the diagonal elements of var.cov (you may find diag() helpful for this question). So now, the function should output beta.hat, e.2, var.cov and ses in a list. Compare the ses vector to the standard errors estimated by lm() above. Are they the same?

BONUS: For your own enjoyment

Note, if you find this daunting, don’t worry. Other courses will cover regression and programming in depth.

1) Interpret the coefficients on BA and female, respectively.

2) What is the predicted value of Y (whether or not someone voted for Obama) for an 95-year-old Democrat who is female and went to college? What about a 50-year-old Republican who is male and went to college? Hint: refer back to the equation for the model we estimated, and note that you now have estimated values for the unknown parameters.

3) For both predictions, do such people exist in the data set? If so, how many?