In-class Lab 3

ECON 425 (Justin Heflin, West Virginia University)

January 27, 2023

The purpose of this in-class lab is to continue to familiarize yourself with RStudio, more specifically today we will be learning about data frames, data files, and how to plot graphics. The lab may be completed as a group, but each student should turn in their own work. To get credit, upload your .R script to the appropriate place on eCampus ("In-Class Labs'' folder).

For starters

Open up a new R script (named `ICL3_XYZ.R`, where `XYZ` are your initials)

Importing Data

- Download the texas_crime data-set from the In-Class Labs folder on eCampus
- Save it on your desktop, a folder on your laptop, somewhere where you can easily find it
- Save the data-set as texas_crime.csv
- Import the texas_crime dataset
- Set your working directory `setwd()` by going into the bottom-right panel
- Select the `Files` tab (should be the tab on the far left in that panel)
- Find where you saved the texas_crime data-set
- Select the "gear" `More` dropdown menu, then click `Set As Working Directory`

For instance, I have saved the data-set to my desktop so I will set my working directory as my desktop. If you have a Econ 425 folder on your desktop or somewhere on your laptop and you want to set your working directory to that folder because that's where you saved the data-set that is completely fine. Just follow the steps to set your working directory to that particular folder. Once you have set your working directory you can proceed to importing the data-set into R by using the following line of code:

```r
texas_crime <- read.csv("texas_crime.csv")
#View(texas_crime)
```

Note: if you saved the data-set as something else, whatever that something else is it must be what's written in between the quotation marks, "Texas_Crime_Econ_425",… Remember R is case sensitive, meaning what is between the quotation marks has to be identical to the file name you gave it when you downloaded it.

You can use the `View()` function to ensure the data-set has properly been imported using the following code: `View(texas_crime)` or whatever name you called the data-set.

This is an example of a panel data-set, we have a variety of data at the county level for all counties in the state of course repeated on an annual basis from 1993 through 2017

Exploring Data-Set

```
colnames(texas_crime)
#Returns the name of the variables in the data-set

head(texas_crime)
#Returns the first 6 rows of the data-set

str(texas_crime)
#Returns the structure of the data-set such as number of observations, variable names
#Type of variable, and the first few values of each variable

summary(texas_crime)
#Provides summary statistics such mean, median, minimum, maximum, and the number of NA's
#We will drop/subset the NA observations from the data-set later
```

In your R script comment out the variable type for the county and burglary columns, the average robbery number, and the maximum number for aggravated assault.

Dropping Observations, Subsetting

If you have viewed the data-set you probably noticed for each county in the year 2017 we have no data on any variable. So, we need to drop those rows. To do so we will use the following code:

```
texas_crime_2 <- texas_crime[!texas_crime$year %in% 2017,] #View(texas_crime_2)
#The lines of code directly above and below this comment accomplish the exact same thing
texas_crime_3 <- subset(texas_crime, texas_crime$year<="2016") #View(texas_crime_3)
```

The `!` means "not". For example, `x!=y` which stands for x is not equal to y.

Creating dummy/binary variables using the ifelse function with one condition

Imagine a unit such as a county or entire state passing and implementing a policy. One way to signal that a policy has been implemented is the use of a dummy variable where if the dummy variable is equal to 1 that indicates the policy is in effect. If the dummy variable is equal to 0, that implies the policy has not been implemented (think pre-treatment period). To create a dummy/binary variable in R we will use the `ifelse()` function.

The `ifelse()` function takes three parameters. First the single or multiple condition (the first example below is a single condition), second the value to be returned when the condition evaluates `TRUE` and third the value to return when the condition evaluates `FALSE`.

Our Texas crime data-set covers every county in the state of Texas from 1993 through 2016. Firearm ownership in Texas is relatively high. To be proactive the state of Texas implements a state-wide 10 day waiting period when someone purchases a firearm. Texas implemented this policy in 2000.

Below is the code used to create the dummy variable titled "Waiting_Period_10_Days"

```
texas_crime_2$Waiting_Period_10_Days <- ifelse(texas_crime_2$year >= 2000, 1, 0)
```

First, we had to create a new variable in our data frame using the $ operator. Then, inside the `ifelse` function we had to define the conditions. Since this policy is state-wide, meaning every county in the state had to adopt it we only need to have one condition, the timing of the policy. Essentially, for every county in the state of Texas, starting in the year 2000, we should have a 1 in the variable column named "Waiting_Period_10_Days" and 0's for every county from 1993 through 1999.

Basically, if the year in the "year" column is greater than or equal to 2000 (the year the policy was implemented) then it should have a 1. If not the year in the "year" column is *not* greater than or equal to 2000 then it should have a 0.

Creating dummy/binary variables using the ifelse function with two conditions, only one unit

Suppose Tyler county passed and implemented a magazine restriction, where consumers are not permitted to own magazines that hold more than 10 rounds at a time. Tyler county implemented this policy in 2011.

```r
texas_crime_2$Magazine_Restriction <- ifelse(texas_crime_2$county_name == "tyler" &
                                                texas_crime_2$year >= 2011, 1, 0)
```

The `&` symbol in the previous line of code allows us to add a second condition. In this example both conditions must be true in order for the dummy variable to return a 1. If one or both of those conditions if false then the function will return a 0.

Creating dummy/binary variables using the ifelse function with two conditions, with multiple units

Suppose a few counties wanted to go a step further on the waiting period policy and have people wait two weeks before they could pick up the firearm they purchased. Austin, Baylor, Dallas, and Trinity passed and implemented a two week waiting period in 2004.

```r
#Create a character vector of the four counties that implemented the two week waiting period
Counties_2004 <- c("austin", "baylor", "dallas", "trinity")

#Create the dummy variable (with two conditions) for the two week waiting period
texas_crime_2$Waiting_Period_14_Days <- ifelse(texas_crime_2$county_name%in%Counties_2004
                                                & texas_crime_2$year >= 2004, 1, 0)
```

The `&` symbol in the previous line of code allows us to add a second condition. In this example both conditions must be true in order for the dummy variable to return a 1. If one or both of those conditions if false then the function will return a 0.

If you wanted to create the dummy variable county by county here is how you could do it:

```r
texas_crime_2$Waiting_Period_14_Days_2 <- ifelse(texas_crime_2$county_name=="austin" &
                                          texas_crime_2$year >= 2004, 1, 0) +
                                          ifelse(texas_crime_2$county_name=="baylor" &
                                          texas_crime_2$year >= 2004, 1, 0) +
                                          ifelse(texas_crime_2$county_name=="dallas" &
                                          texas_crime_2$year >= 2004, 1, 0) +
                                          ifelse(texas_crime_2$county_name=="trinity" &
                                          texas_crime_2$year >= 2004, 1, 0)
```