Data Analysis for Policy Research Using R

Introduction and R Basics

Harold Stolper

1. Introductions
2. What is R and How Will We Use It?
3. Syllabus / Questions?
4. R Projects and Directory Structure
5. R Basics
6. Assignment 1

Introductions

Student introductions

1. Preferred name
2. Pronouns (if comfortable)
3. Previous R exposure/experience
4. And answer one of these two questions:
   - What's something you would eventually like to learn how to do in R?
   - What's something that you have observed or think is important that people in your field aren't paying attention to?
5. (optional) One piece of culture you are excited about right now
   - e.g. music, writing, fashion, a meme, other art, sports, etc.

Liam Tay Kearney, Teaching Assistant

Harold Stolper, instructor

Graduated from SIPA many moons ago, returned to Columbia for my PhD in economics.

Recent background:

- This is my 7th year teaching quant courses at SIPA.
- Worked as the economist for a non-profit doing research and advocacy to promote upward mobility for low-income NYers.
- Recent focus on documenting racist police enforcement of fare evasion, and other topics related to policing, neighborhood change, and transit accessibility.

Transitioned from Stata to R after years of using and teaching Stata.

What is R and How Will We Use It?

What is R?

- R is "an alternative to traditional statistical packages such as SPSS, SAS, and Stata such that it is an extensible, open-source language and computing environment for Windows, Macintosh, UNIX, and Linux platforms." (ICPSR)
- "R is an integrated suite of software facilities for data manipulation, calculation and graphical display." (R-project.org)

How will we use R?

- RStudio is a powerful user interface for R.
  - After installing R and RStudio, we'll be working entirely in the RStudio interface.
- R Markdown files are used in RStudio to "both save and execute code generate high quality reports that can be shared with an audience."
  - This lecture was created using R Markdown.
  - Beginning next week, everything you submit for this class will be a document generated with R Markdown.

Base R vs. user-defined R packages

R uses "packages" as a bundle of code, data and documention.

There are default base packages that come with R. Some examples:

- `base`
- `stats`
- `utils`

And there are R packages developed and shared by others. Some R packages include:

- `tidyverse`
- `ggplot2`

More about these in later weeks...

Installing and loading R packages

You only need to install a package once. To install an R package use `install.package()` function.

```
install.packages("tidyverse")
```

But you need to load a package every time you plan to use it using the `library()` function.

```
library(tidyverse)
#> -- Attaching packages --------------------------------------- tidyverse 1.3.1
#> v ggplot2 3.3.5     v purrr   0.3.4
#> v tibble  3.1.3     v dplyr   1.0.7
#> v tidyr   1.1.3     v stringr 1.4.0
#> v readr   2.0.0     v forcats 0.5.1
#> -- Conflicts ------------------------------------------ tidyverse_conflicts()
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
```

What can you do with R+RStudio+RMarkdown?

Things you can also do using Stata:

- Data cleaning and manipulation
- Statistical analysis and plots

Things you can't do in Stata:

- Generate reports and presentations
- Generate interactive content
  - Maps
  - Graphs
  - Dashboards

What will we be doing in this class?

We'll be learning how to use R to explore data to inform policy.

That means we'll be spending a lot of time working through R, but also thinking about how/when to use the methods and concepts we've learned in Quant I and II:

- **Research design:** understanding how data structure impacts analysis and causal inference
- **Data management:** cleaning and structuring data for analysis
- **Exploratory analysis:** identifying and analyzing key factors in your analysis
- **Explanatory analysis:** estimating relationships between variables to inform policy
- **Data visualization & presentation:** conveying findings to your target audience
- **Policy writing & interpretation:** translating statistical analysis in accessible terms
- **Data advocacy**: thinking critically about using *data for good*

Syllabus / Questions?

R Projects and Directory Structure

Working directory

R looks for files in your **working directory**

Function `getwd()` shows the current working directory (also shown at the top of the RStudio console.)

```
getwd()
#> [1] "C:/Users/hstol/My Drive/SIPA/R - Data Analysis/U6614/Lectures/Lecture1"
```

Function `list.files()` lists all files located in working directory

```
list.files()
```

So what is the working directory?

When you run code from the **R Console** or an **R Script**, or from **code chunks** in an RMarkdown file (.rmd), the working directory is...

- the folder your file is saved in, or ...
- if you are working within an **R Project**, the working directory is the main directory for the project (more on that shortly!)

```
getwd()
#> [1] "C:/Users/hstol/My Drive/SIPA/R - Data Analysis/U6614/Lectures/Lecture1"
```

- For this class we'll be using R projects to keep organized.

The path is how we refer to a directory

Absolute file path: a complete list of directories needed to locate a file or folder.

```
setwd("C:/Users/Harold Stolper/Google Drive/SIPA/R - Data Analysis/Fall 2020/Le
```

Relative file path: a way of indicating a given file location relative to your working directory (note that they might be the same!)

- Assuming your current working directory is in the "lecture2" folder and you want to go up 2 levels to the "Fall 2021" folder, your relative file path would look something like this:

```
setwd("../../")
```

File path shortcuts:

Key	Description
~	tilde is a shortcut for user's home directory (mine is my name pm)
../	moves up a level
../../	moves up two level

What is an R project? Why are we using them?

What is an "R project"?

- A file that keeps all "project" files organized together: – input data, R scripts, analytical results, and figures.
- When you open an R project, your working directory is automatically set to the directory where your R project lives.

Why will we be asking you to create and work with R projects?

- We want you to be able to run the R Markdown files (.rmd) used to generate each lecture.
- Often times these .rmd files point to data files or sub-folders
- You can create or download an R project with directory structure (i.e. organizing files and sub-folders in a particular way).
- That way you'll be able to run .rmd files from your own computer that point to files in sub-folders without making any changes to file-paths.

R Basics

Executing code in R

Three ways to execute commands in R

1. **Console:** type/paste commands to run "on the fly"
2. **Code chunks** in R Markdown (.rmd files)
   - Can execute one command at a time, one chunk at a time, or "knit" the entire file into a document (e.g. html or pdf)
3. **R scripts** (.r files)
   - Just a text file full of R commands
   - Can execute one command at a time, several commands at a time, or the entire script

Shortcuts for executing commands

Three ways to execute commands in R

1. Type/paste commands directly into the "console" (and press ENTER)
2. 'code chunks' in RMarkdown (.Rmd files)
   - **Cmd/Ctrl + Enter**: execute highlighted line(s) within chunk
   - **Cmd/Ctrl + Shift + k**: "knit" entire document
3. R scripts (.R files)
   - **Cmd/Ctrl + Enter**: execute highlighted line(s)
   - **Cmd/Ctrl + Shift + Enter** (without highlighting any lines): run entire script

Assignment

Assignment means assigning a value/set of values to an "object"

- `<-` is the assignment operator
  - in other languages `=` is the assignment operator
- good practice to put a space before and after assignment operator

```
# Create an object a and assign value
a <- 5
a
#> [1] 5

# Create an object b and assign value
b <- "yay!"
b
#> [1] "yay!"
```

Note 1: comments start with one or more `#` symbols

Note 2: R is caps sensitive!

Objects and assignment

R stores information in objects (like all "object-oriented" programming languages).

Some objects:

- numbers
- character strings
- vectors
- matrices
- lists
- functions
- plots
- dataframes (the datasets of R!)

Functions

Functions do things to different objects. They often accept arguments – we "pass" arguments to functions.

Functions are also objects themselves that can be "called" to do things like:

- calculate and display statistics
- generate output
- display part or all of objects (e.g. show some data)
- manipulate objects (e.g. create a new column of data)
- extract information from objects (e.g. the number of rows of data)

Base R includes lots of functions. We'll be working with additional packages that include some handy functions.

Let's jump in!

Our goals for today's R workshop example are very modest:

- Create an R project including R script.
- Look around and get our bearings.
- Install and load a package (gapminder).
- Use base R functions to inspect a dataframe included with this package.
- Use some functions to perform some very basic analysis.
- Assign results from our analysis to new objects and display them.

Assignment 1

Assignment 1: submit an R script via CW by 9am next Tuesday

Create an R script called assignment1.r that includes code and answers (as comments) for the following:

0. Create a new R project called assignment1.
   - This is for your internal project management, do not submit your R project file.
1. Load the gapminder data using the library function.
   - You'll need to install the gapminder package if you didn't follow in class today.
2. Show the data structure of the gapminder dataframe in the gapminder package.
3. What is the average gdpPercap across all observations in the dataset?
   - Use ?gapminder to access gapminder documentation and find the units for gdpPercap.
   - How would you interpret this mean? i.e. what is it the mean of?
4. Plot year (x-axis) vs. gdpPercap (y-axis).
   - Describe what the plot says about economic growth over time.
5. Create a barplot showing the number of observations in each continent.
   - Start by using the table function with continents as its argument.
   - Next pass the object created by this function to the barplot function.

General assignment guidance

- Use blank spaces liberally, code is hard to read and spaces help!
- Use comments liberally throughout your R script to describe your steps.
- Troubleshooting is a skill! Here are some tips and resources:
  - Consult the R script from today's class for examples.
  - Get used to using R's built in documentation by using ?
  - Use Google liberally for examples that work.
  - When you're stuck, focus on finding examples to get your own code to work, even if you don't feel comfortable with all the syntax just yet.
- Consulting with others is good! Copying, however, is not the way to learn to code.
  - **Copied assignment submissions will result in a 0 for all parties.**