Computational Sociology (PGSP11583)

Christopher Barrie

2022-08-17T00:00:00+03:00

Table of contents

Computational Sociology

This is the course book we will be using for Computational Sociology (PGSP11583).

```
print("Computational Sociology")
```

```
[1] "Computational Sociology"
```

> The book is a "live" document meaning I will be updating as we progress together through the course.

License

Learning outcomes

This course will give students training in the use of various computational methods in the social sciences. The course will prepare students for dissertation work that uses digital trace data and/or computational methods and will provide hands-on training in the use of the R programming language and (some) Python.

The course will provide a venue for seminar discussion of examples using these methods in the empirical social sciences as well as lectures on the technical and/or statistical dimensions of their application.

Course structure

We will be using this online book for the ten-week course in "Computational Sociology" (PGSP11583). Each chapter contains the readings for that week. The book also includes worksheets with example code for how to conduct some of the text analysis techniques we discuss each week.

Each week (with the partial exception of week 1), we will be discussing, alternately, the substantive and technical dimensions of published research in the empirical social sciences.

Course theme

In order to discipline the course, I have decided to focus on one theme that is currently making headlines: the political consequences of social media.

With this in mind, every two weeks we will be studying a different phenomenon that has gained media attention: e.g., echo chambers, misinformation, violence.

On alternate weeks, we will discuss how these phenomena have been studied—using both computational and non-computational methods. In the subsequent week, we will then go through some of the technical dimensions of the methods used in the papers we study.

Course pre-preparation

NOTE: Before the lecture in Week 2, students should complete two introductory R exercises.

1. First, you should consult the worksheet here, which is an introduction to setting up and understanding the very basics of working in R. S
2. Second, Ugur Ozdemir has provided such a more comprehensive introductory R course for the Research Training Centre at the University of Edinburgh and you can follow the instructions here to access this.

Reference sources

There are two main reference texts that will be of use during this course:

- Wickham, Hadley and Garrett Grolemund. R for Data Science: https://r4ds.had.co.nz/
- Salganik, Matt. Bit by Bit: Social Research in the Digital Age: https://www.bitbybitbook.com/

Assessment

Fortnightly worksheets

Each fortnight, I will provide you with one worksheet that walks you through how to implement a different computational technique. At the end of these worksheets you will find a set of questions. **You should buddy up with someone else in your class and go through these together.**

This is called "pair programming" and there's a reason we do this. Firstly, coding can be an isolating and difficult thing—it's good to bring a friend along for the ride! Secondly, if there's something you don't know, maybe your buddy will. This saves you both time. Thirdly, your buddy can check your code as you write it, and vice versa. Again, this means both of you are working together to produce and check something as you go along.

At the subsequent week's lecture, I will pick on a pair at random to answer each one of that worksheet's questions (i.e., there is ~1/3 chance you're going to get picked each week). I will ask you to walk us through your code. And remember: it's also fine if you struggled and didn't get to the end! If you encountered an obstacle, we can work through that together. All that matters to me is that you **try**.

Fortnightly flash talks

On the weeks where you are not going to be tasked with a coding assignment, you're not off the hook… I will again be selecting a pair at random (the same as your coding pair) to talk me through one of the readings. I will pick a different pair for each reading (i.e., ~ 1/3 chance again).

Don't let this be cause of great anguish: I just want **two or three minutes** where you lay out for me: 1) the research question; 2) the data source; 3) the method; 4) the findings; 5) what you thought its limitations were. The main portion of your flash talk should focus on element 5).

For this last one, you will want to think about 1)-4); i.e., you will want to think about whether it really answered the research question, whether the data was appropriate for answering that question, whether the method was appropriate for answering that question, and whether the results show what the author claims they show. **I will provide you wish an example flash talk at the first lecture.**

Final assessment

Assessment takes the form of **one** summative assessment. This will be a 4000 word essay on a subject of your choosing (with prior approval by me). For this, you will be required to select from a range of data sources I will provide. You may also suggest your own data source.

You will be asked to: a) formulate a research question; b) use at least one computational technique that we have studied; c) conduct an analysis of the data source you have provided; d) write up the initial findings; and e) outline potential extensions of your analysis.

You will then provide the code you used in reproducible (markdown) format and will be assessed on both the substantive content of your essay contribution (the social science part) as well as your demonstrated competency in coding and text analysis (the computational part).

Introduction to R

This section is designed to ensure you are familiar with the R environment.

Getting started with R at home

Given that we're all working from home these days, you'll need to download R and RStudio onto your own devices. R is the name of the programming language that we'll be using for coding exercises; RStudio is the IDE ("Integrated Development Environment"), i.e., the piece of software that almost everyone uses when working in R.

You can download both of these on Windows and Mac easily and for free. This is one of the first reasons to use an "open-source" programming language: it's free and everyone can contribute!

IT Services at the University of Edinburgh have provided a walkthrough of what is needed for you to get started. I also break this down below:

1. Install R for Mac from here: https://cran.r-project.org/bin/macosx/. Install R for Windows from here: https://cran.r-project.org/bin/windows/base/.
2. Download RStudio for Windows or Mac from here: https://rstudio.com/products/rstudio/download/, choosing the Free version: this is what most people use and is more than enough for all of our needs.

All programs are free. Make sure to load everything listed above for your operating system or R will not work properly!

Some basic information

- A script is a text file in which you write your commands (code) and comments.
- If you put the # character in front of a line of text this line will not be executed; this is useful to add comments to your script!
- R is case sensitive, so be careful when typing.

- To send code from the script to the console, highlight the relevant line of code in your script and click on Run, or select the line and hit ctrl+enter on PCR or cmd+enter on Mac
- Access help files for R functions by preceding the name of the function with ? (e.g., ?table)
- By pressing the up key, you can go back to the commands you have used before
- Press the tab key to auto-complete variable names and commands

Getting Started in RStudio

Begin by opening RStudio (located on the desktop). Your first task is to create a new script (this is where we will write our commands). To do so, click:

```
File --> NewFile --> RScript
```

Your screen should now have four panes:

- the Script (top left)
- the Console (bottom left)
- the Environment/History (top right)
- Files/Plots/Packages/Help/Viewer (bottom right)

A simple example

The Script (top left) is where we write our commands for R. You can try this out for a first time by writing a small snipped of code as follows:

```
x <- "I can't wait to learn Computational Text Analysis" #Note the quotation marks!
```

To tell R to run the command, highlight the relevant row in your script and click the Run button (top right of the Script) - or hold down ctrl+enter on Windows or cmd+enter on Mac - to send the command to the Console (bottom left), where the actual evaluation and calculations are taking place. These shortcut keys will become very familiar to you very quickly!

Running the command above creates an object named 'x', that contains the words of your message.

You can now see 'x' in the Environment (top right). To view what is contained in x, type in the Console (bottom left):

```
print(x)
```

```
[1] "I can't wait to learn Computational Text Analysis"
```

```
# or alternatively you can just type:

x
```

```
[1] "I can't wait to learn Computational Text Analysis"
```

Loading packages

The 'base' version of R is very powerful but it will not be able to do everything on its own, at least not with ease. For more technical or specialized forms of analysis, we will need to load new packages.

This is when we will need to install a so-called 'package'—a program that includes new tools (i.e., functions) to carry out specific tasks. You can think of them as 'extensions' enhancing R's capacities.

To take one example, we might want to do something a little more exciting than print how excited we are about this course. Let's make a map instead.

This might sound technical. But the beauty of the packaged extensions of R is that they contain functions to perform specialized types of analysis with ease.

We'll first need to install one of these packages, which you can do as below:

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

After the package is installed, we then need to load it into our environment by typing library(). Note that, here, you don't need to wrap the name of the package in quotation marks. So this will do the trick:

```
library(tidyverse)
```

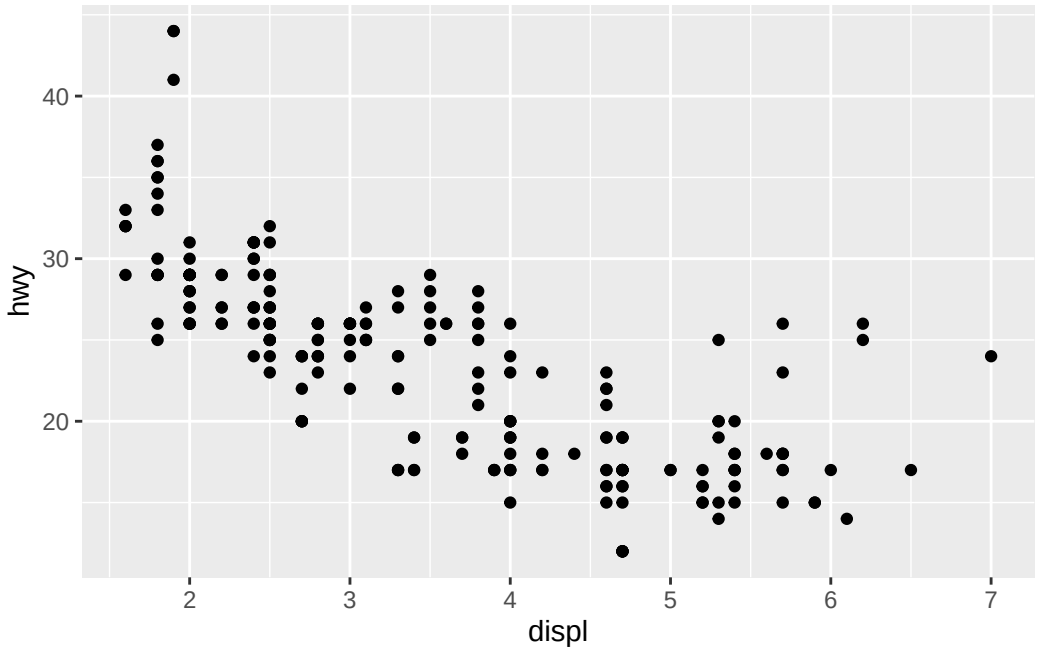
```
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
v ggplot2 3.3.6     v purrr   0.3.4
v tibble  3.1.7     v dplyr   1.0.9
v tidyr   1.2.0     v stringr 1.4.0
v readr   2.1.2     v forcats 0.5.1

-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

What now? Well, let's see just how easy it is to visualize some data using ggplot which is a package that comes bundled into the larger tidyverse package.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



If we wanted to save where we'd got to with making our plots, we would want to save our scripts, and maybe the data we used as well, so that we could return to it at a later stage.

Saving your objects, plots and scripts

- Saving scripts: To save your script in RStudio (i.e. the top left panel), all you need to do is click File –> Save As (and choose a name for your script). Your script will be

something like: myfilename.R.

- Saving plots: If you have made any plots you would like to save, click Export (in the plotting pane) and choose a relevant file extension (e.g. .png, .pdf, etc.) and size.
- To save **individual** objects (for example x from above) from your environment, run the following command (choosing a suitable filename):

```
save(x,file="myobject.RData")
load(file="myobject.RData")
```

- To save **all** of your objects (i.e. everything in the top right panel) at once, run the following command (choosing a suitable filename):

```
save.image(file="myfilname.RData")
```

- Your objects can be re-loaded into R during your next session by running:

```
load(file="myfilename.RData")
```

There are many other file formats you might use to save any output. We will encounter these as the course progresses.

Knowing where R saves your documents

If you are at home, when you open a new script make sure to check and set your working directory (i.e. the folder where the files you create will be saved). To check your working directory use the getwd() command (type it into the Console or write it in your script in the Source Editor):

```
getwd()
```

To set your working directory, run the following command, substituting the file directory of your choice. Remember that anything following the '#' symbol is simply a clarifying comment and R will not process it.

```
## Example for Mac
setwd("/Users/Documents/mydir/")
## Example for PC
setwd("c:/docs/mydir")
```

Practicing in R

The best way to learn R is to use it. These workshops on text analysis will not be the place to become fully proficient in R. They will, however, be a chance to conduct some hands-on analysis with applied examples in a fast-expanding field. And the best way to learn is through doing. So give it a shot!

For some further practice in the R programming language, look no further than (**wickham_r_2017?**) and, for tidy text analysis, (**silge_text_2017?**).

- The free online book by Hadley Wickham "R for Data Science" is available here
- The free online book by Julia Silge and David Robinson "Text Mining with R" is available here
- For more practice with R, you may want to consult a set of interactive tutorials, available through the package "learnr." Once you've installed this package, you can go through the tutorials yourselves by calling:

```
library(learnr)

available_tutorials() # this will tell you the names of the tutorials available

run_tutorial(name = "ex-data-basics", package = "learnr") #this will launch the interactiv
```

One final note

Once you've dipped into the "R for Data Science" book you'll hear a lot about the so-called tidyverse in R. This is essentially a set of packages that use an alternative, and more intuitive, way of interacting with data.

The main difference you'll notice here is that, instead of having separate lines for each function we want to run, or wrapping functions inside functions, sets of functions are "piped" into each other using "pipe" functions, which look have the appearance: `%>%`.

I will be using "tidy" syntax in the weekly exercises for these computational text analysis workshops. If anything is unclear, I can provide the equivalents in "base" R too. But a lot of the useful text analysis packages are now composed with 'tidy' syntax.

1 Week 1

This week will be dedicated to a more general introduction to computational social science and what it means to think "computationally."

In essence, this means getting used to different types of data. These might be:

- Social media data;
- Image data;
- Sound data;
- Video data;
- Remote sensing data;
- Call detail records

These data are "found" or "trace" data. They are not custom-made for social science; they *can* be repurposes, though, for answering questions in the social sciences.

1.1 Essential reading:

- Salganik (2017)
- D. M. J. Lazer et al. (2020)
- D. Lazer et al. (2021)

1.2 Slides

Slides for week one are available here

2 Week 2

3 Week 3

4 Week 4

5 Week 5

References

Lazer, David M. J., Alex Pentland, Duncan J. Watts, Sinan Aral, Susan Athey, Noshir Contractor, Deen Freelon, et al. 2020. "Computational Social Science: Obstacles and Opportunities." *Science* 369 (6507): 1060–62. https://doi.org/10.1126/science.aaz8170.

Lazer, David, Eszter Hargittai, Deen Freelon, Sandra Gonzalez-Bailon, Kevin Munger, Katherine Ognyanova, and Jason Radford. 2021. "Meaningful Measures of Human Society in the Twenty-First Century." *Nature* 595 (7866): 189–96. https://doi.org/10.1038/s41586-021-03660-7.

Salganik, Matthew J. 2017. *Bit by Bit: Social Research in the Digital Age*. Princeton, NJ.: Princeton University Press.