Introduction to Statistical Computing

Susan Vanderplas

Spring 2022

Contents

Preface

This book is designed to demonstrate introductory statistical programming concepts and techniques. It is intended as a substitute for hours and hours of video lectures - watching someone code and talk about code is not usually the best way to learn how to code. It's far better to learn how to code by … coding.

I hope that you will work through this book week by week over the semester. I have included comics, snark, gifs, YouTube videos, extra resources, and more: my goal is to make this a collection of the best information I can find on statistical programming.

In most cases, this book includes **way more information** than you need. Everyone comes into this class with a different level of computing experience, so I've attempted to make this book comprehensive. Unfortunately, that means some people will be bored and some will be overwhelmed. Use this book in the way that works best for you - skip over the stuff you know already, ignore the stuff that seems too complex until you understand the basics. Come back to the scary stuff later and see if it makes more sense to you.

How to Use This Book

I've made an effort to use some specific formatting and enable certain features that make this book a useful tool for this class.

Special Sections

Watch Out

Watch out sections contain things you may want to look out for - common errors, etc.

Examples

Example sections contain code and other information. Don't skip them!

My Opinion

These sections contain things you should definitely not consider as fact and should just take with a grain of salt.

Go Read

Sometimes, there are better resources out there than something I could write myself. When you see this section, go read the enclosed link as if it were part of the book.

Try It Out

Try it out sections contain activities you should do to reinforce the things you've just read.

Learn More

Learn More sections contain other references that may be useful on a specific topic. Suggestions are welcome (email me to suggest a new reference that I should add), as there's no way for one person to catalog all of the helpful programming resources on the internet!

Note

Note sections contain clarification points (anywhere I would normally say "note that ....)

Expandable Sections

These are expandable sections, with additional information when you click on the line

This additional information may be information that is helpful but not essential, or it may be that an example just takes a LOT of space and I want to make sure you can skim the book without having to scroll through a ton of output.

Many times, examples will be in expandable sections

This keeps the code and output from obscuring the actual information in the textbook that I want you to retain. You can always look up the syntax, but you do need to absorb the details I've written out.

About This Book

This is a Quarto book. To learn more about Quarto books visit https://quarto.org/docs/books.

I have written this entire book using reproducible techniques, with R and python code and results included within the book's text.

Stat 151 will be offered for the first time in Spring 2022, as I'm writing this in Fall 2021. Initially, my goal is to write the book in R and include python as an additional option/example. Eventually, I hope to teach Stat 151 in R and Python at the same time.

1 Getting Started

Objectives

1. Understand the basics of how computers work
2. Understand the file system mental model for computers
3. Set up RStudio, R, Quarto, and python
4. Be able to run demo code in R and python

1.1 Computer Basics

It is helpful when teaching a topic as technical as programming to ensure that everyone starts from the same basic foundational understanding and mental model of how things work. When teaching geology, for instance, the instructor should probably make sure that everyone understands that the earth is a round ball and not a flat plate – it will save everyone some time later.

We all use computers daily - we carry them around with us on our wrists, in our pockets, and in our backpacks. This is no guarantee, however, that we understand how they work or what makes them go.

1.1.1 Hardware

Here is a short 3-minute video on the basic hardware that makes up your computer. It is focused on desktops, but the same components (with the exception of the optical drive) are commonly found in cell phones, smart watches, and laptops.

When programming, it is usually helpful to understand the distinction between RAM and disk storage (hard drives). We also need to know at least a little bit about processors (so that we know when we've asked our processor to do too much). Most of the other details aren't necessary (for now).

1.1.2 Operating Systems

Operating systems, such as Windows, MacOS, or Linux, are a sophisticated program that allows CPUs to keep track of multiple programs and tasks and execute them at the same time.

1.1.3 File Systems

Evidently, there has been a bit of generational shift as computers have evolved: the "file system" metaphor itself is outdated because no one uses physical files anymore. This article is an interesting discussion of the problem: it makes the argument that with modern search capabilities, most people use their computers as a laundry hamper instead of as a nice, organized filing cabinet.

Regardless of how you tend to organize your personal files, it is probably helpful to understand the basics of what is meant by a computer **file system** – a way to organize data stored on a hard drive. Since data is always stored as 0's and 1's, it's important to have some way to figure out what type of data is stored in a specific location, and how to interpret it.

That's not enough, though - we also need to know how computers remember the location of what is stored where. Specifically, we need to understand **file paths**.

When you write a program, you may have to reference external files - data stored in a .csv file, for instance, or a picture. Best practice is to create a file structure that contains everything you need to run your entire project in a single file folder (you can, and sometimes should, have sub-folders).

For now, it is enough to know how to find files using file paths, and how to refer to a file using a relative file path from your base folder. In this situation, your "base folder" is known as your **working directory** - the place your program thinks of as home.

2 Setting up your computer

In this section, I will provide you with links to set up various programs on your own machine. If you have trouble with these instructions or encounter an error, post on the class message board or contact me for help.

1. Download and run the R installer for your operating system from CRAN:

    - Windows: https://cran.rstudio.com/bin/windows/base/
    - Mac: https://cran.rstudio.com/bin/macosx/
    - Linux: https://cran.rstudio.com/bin/linux/ (pick your distribution)

    If you are on Windows, you should also install the Rtools4 package; this will ensure you get fewer warnings later when installing packages.

    More detailed instructions for Windows are available here

2. Download and install the latest version of python 3

    - Then, install Jupyter using the instructions here

3. Download and install the latest version of RStudio for your operating system. RStudio is a integrated development environment (IDE) for R - it contains a set of tools designed to make writing R code easier.

4. Download and install the latest version of Quarto for your operating system. Quarto is a command-line tool released by RStudio that allows Rstudio to work with python and other R specific tools in a unified way.

In class activity

Open RStudio on your computer and explore a bit.

- Can you find the R console? Type in `2+2` to make sure the result is `4`.
- Run the following code in the R console:

```r
install.packages(
  c("tidyverse", "rmarkdown", "knitr", "quarto")
)
```

- Can you find the text editor?
  - Create a new quarto document (File -> New File -> Quarto Document).
  - Paste in the contents of this document.
  - Compile the document and use the Viewer pane to see the result.
  - If this all worked, you have RStudio, Quarto, R, and Python set up correctly on your machine.

3 Scripts and Notebooks

In this class, we'll be using markdown notebooks to keep our code and notes in the same place. One of the advantages of both R and Python is that they are both scripting languages, but they can be used within notebooks as well. This means that you can have an R script file or a python script file, and you can run that file, but you can also create a document (like the one you're reading now) that has code AND text together in one place. This is called literate programming and it is a very useful workflow both when you are learning programming and when you are working as an analyst and presenting results.

3.1 Scripts

Before I show you how to use literate programming, let's look at what it replaces: scripts. **Scripts** are files of code that are meant to be run on their own. They may produce results, or format data and save it somewhere, or scrape data from the web – scripts can do just about anything.

Scripts can even have documentation within the file, using `#` characters (at least, in R and python) at the beginning of a line. `#` indicates a comment – that is, that the line does not contain code and should be ignored by the computer when the program is run. Comments are incredibly useful to help humans understand what the code does and why it does it.

3.1.0.1 Plotting a logarithmic spiral in R and python

This code will use concepts we have not yet introduced - feel free to tinker with it if you want, but know that you're not responsible for being able to **write** this code yet. You just need to read it and get a sense for what it does. I have heavily commented it to help with this process.

```r
# Define the angle of the spiral (polar coords)
# go around two full times (2*pi = one revolution)
theta <- seq(0, 4*pi, .01)
# Define the distance from the origin of the spiral
# Needs to have the same length as theta
r <- seq(0, 5, length.out = length(theta))
```

```
# Now define x and y in cartesian coordinates
x <- r * cos(theta)
y <- r * sin(theta)

plot(x, y, type = "l")
```

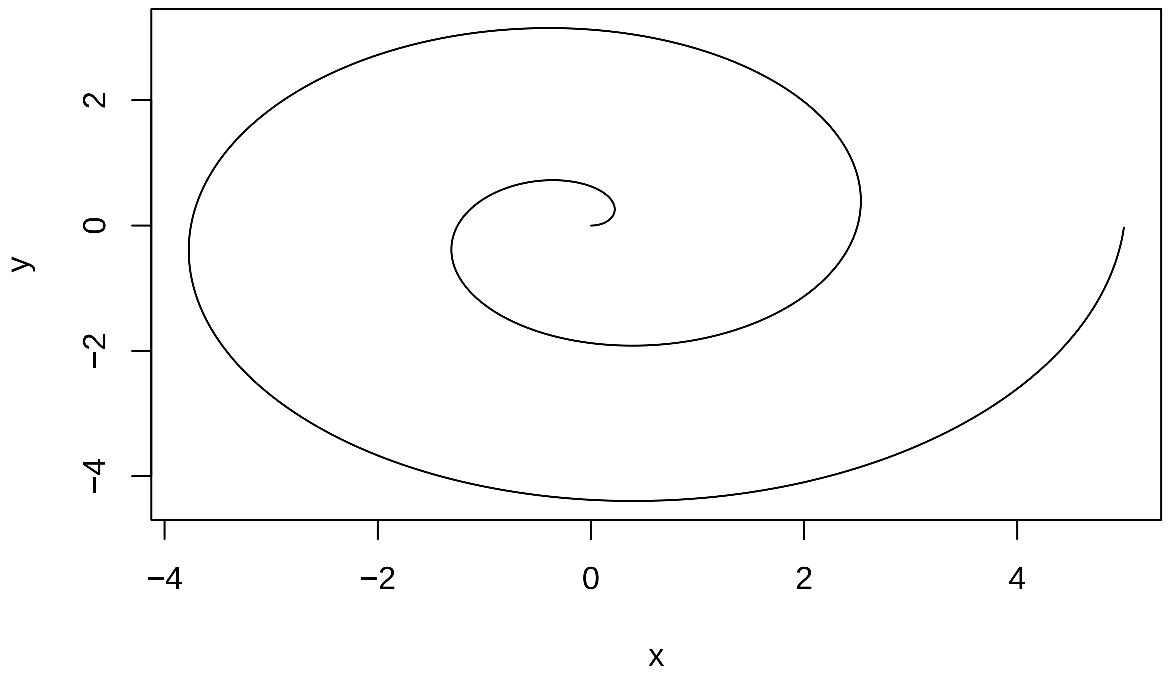


Figure 3.1: A Cartesian Spiral in R

I have saved this script here. You can download it and open it up in RStudio (File -> Open -> Navigate to file location).

```
import numpy as np
import matplotlib.pyplot as plt

# Define the angle of the spiral (polar coords)
# go around two full times (2*pi = one revolution)
theta = np.arange(0, 4 * np.pi, 0.01)
# Define the distance from the origin of the spiral
# Needs to have the same length as theta
# (get length of theta with theta.size,
#  and then divide 5 by that to get the increment)
r = np.arange(0, 5, 5/theta.size)
```

```
# Now define x and y in cartesian coordinates
x = r * np.cos(theta)
y = r * np.sin(theta)

# Define the axes
fig, ax = plt.subplots()
# Plot the line
ax.plot(x, y)
plt.show()
```

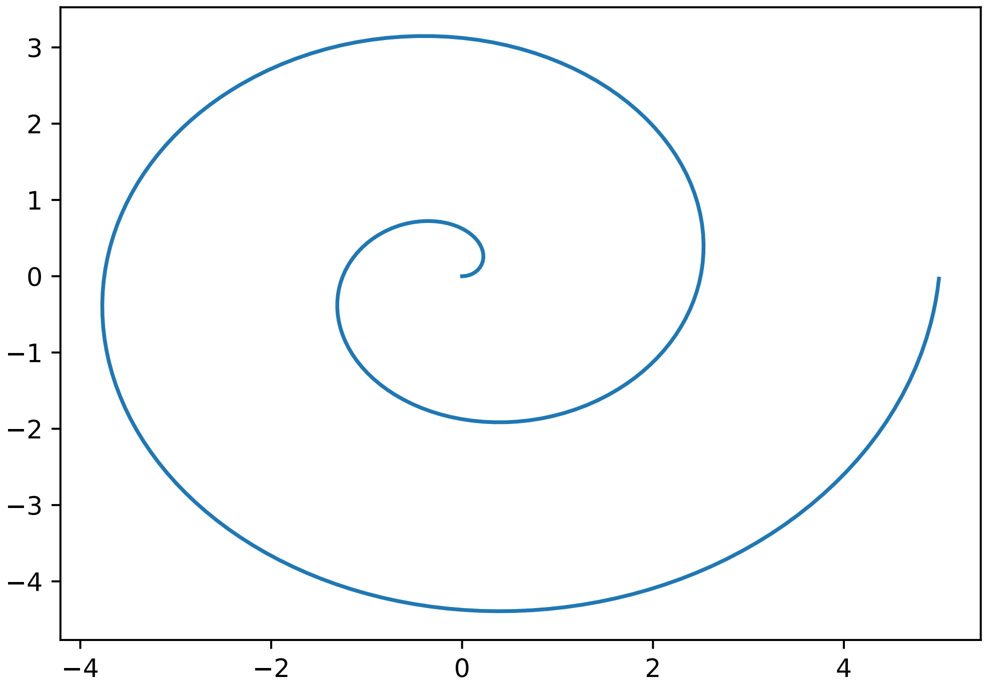


Figure 3.2: A Cartesian Spiral in python

I have saved this script here. You can download it and open it up in RStudio (File -> Open -> Navigate to file location).

Scripts can be run in Rstudio by clicking the Run button at the top of the editor window when the script is open.

3.1.0.2 Try it out!

- Download the R and python scripts in the above example, open them in RStudio, and run each script using the Run button. What do you see?
- (Advanced) Open a terminal in RStudio (Tools -> Terminal -> New Terminal) and see if you can run the R script from the terminal using `R CMD BATCH path/to/file/markdown-spiral-script.R` (You will have to modify this command to point to the file on your machine)
  Notice that two new files appear in your working directory: `Rplots.pdf` and `markdown-spiral-script.Rout`
- (Advanced) Open a terminal in RStudio (Tools -> Terminal -> New Terminal) and see if you can run the R script from the terminal using `python3 path/to/file/markdown-spiral-script.py` (You will have to modify this command to point to the file on your machine)
  This will require you to have python3 accessible to you on the command line, which may be a challenge if it is not set up in the way that I'm assuming it is. Feel free to make an appointment to see if we can figure it out, if this does not work the first time.

Most of the time, you will run scripts interactively - that is, you'll be sitting there watching the script run and seeing what the results are as you are modifying the script. However, one advantage to scripts over notebooks is that it is easy to write a script and schedule it to run without supervision to complete tasks which may be repetitive. I have a script that runs daily at midnight, 6am, noon, and 6pm to pull information off of the internet for a dataset I'm maintaining. I've set it up so that this all happens automatically and I only have to check the results when I am interested in working with that data.

3.2 Notebooks

Notebooks are an implementation of literate programming. Both R and python have native notebooks that are cross-platform and allow you to code in R or python. This book is written using Quarto markdown, which is an extension of Rmarkdown, but it is also possible to use jupyter notebooks to write R code.

In this class, we're going to use Quarto/R markdown, because it is a much better tool for creating polished reports than Jupyter (in my opinion). This matters because the goal is that you learn something useful for your own coding and then you can easily apply it when you go to work as an analyst somewhere to produce impressive documents. Jupyter notebooks are great for interactive coding, but aren't so wonderful for producing polished results. They also don't allow you to switch between languages mid-notebook, and since I'm trying to teach this class in both R and python, I want you to have both languages available.

There are some excellent opinions surrounding the use of notebooks in data analysis:

- Why I Don't Like Notebooks" by Joel Grus at JupyterCon 2018
- The First Notebook War by Yihui Xie (response to Joel's talk).
  Yihui Xie is the person responsible for `knitr` and `Rmarkdown`.

3.2.0.1 Try it out - R markdown

Take a look at the R markdown sample file I've created to go with the R script above. You can see the HTML file it generates here.

- Download the Rmd file and open it with RStudio.
- Change the output to `output: word_document` and hit the Render button

.
  Can you find the markdown-demo.docx file that was generated? What does it look like?
- Change the output to `output: pdf_document` and hit the Render button

.
  Can you find the markdown-demo.pdf file that was generated? What does it look like?

Rmarkdown tries very hard to preserve your formatted text appropriately regardless of the output document type. While things may not look exactly the same, the goal is to allow you to focus on the content and the formatting will "just work".

3.2.0.2 Try it out - Jupyter

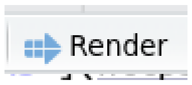
Take a look at the jupyter notebook sample file I've created to go with the R script above. You can see the HTML file it generates here.

- Download the ipynb file and open it with jupyter.
- Export the notebook as a pdf file (File -> Save as -> PDF via HTML). Can you find the jupyter-demo.pdf file that was generated? What does it look like?
- Export the notebook as an html file (File -> Save as -> HTML). Can you find the jupyter-demo.html file that was generated? What does it look like?

3.2.0.3 Try it out - Quarto markdown

The nice thing about quarto is that it will work with python and R seamlessly, and you can compile the document using python or R. R markdown will also allow you to use python chunks, but you must compile the document in R.

Take a look at the Qmd notebook sample file I've created to go with the scripts above. You'll notice that it is basically the script portion of this textbook – that's because I'm writing the textbook in Quarto.

- Download the qmd file and open it with RStudio
- Try to compile the file by hitting the Render button 

- (Advanced) In the terminal, type in `quarto render path/to/file/quarto-demo.qmd`. Does that render the HTML file?
  One advantage of this is that using quarto to render the file doesn't require R at the command line. As the document contains R chunks, R is still required to compile the document, but the biggest difference between qmd and rmd is that qmd files are workflow agnostic - you can generate them in e.g. MS Visual Studio Code, compile them in that workflow, and never have to use RStudio.

Notes:

Show code R file, code python file, R notebook file, and jupyter notebook file that do similar things. Then show a quarto notebook file that runs the R and python chunks together in a single document.

4 Using R and Python as Calculators

5 Basic Data Types

integer/numeric/string/double/etc.

6 Vectors, Arrays, Indexing, and Loops

Vectors, Matrices, and Numpy arrays. Logical indexing. For loops.

(probably 2 weeks of content)

7 Data Structures

data frames in R and Pandas

8 Reading in Data

External data files - csv, excel, etc.

References