

Zobitz and Zimmerman

Our name



Table of contents

Welcome	v
Welcome	v
1 Environments for Data Science	1
1.1 Choosing different languages	1
1.2 Our recommendations	5
1.3 The future: why choose?	6
1.4 Exercises	6
References	7
References	7





Welcome

Welcome



1

Environments for Data Science

You have data from an experiment, or perhaps have accessed data from an ecological network such as NEON. You are now ready to explore your data through visualization, conduct statistical tests, and perhaps increase the explanatory power beyond just the data collection itself. Your next step is to choose a computational environment for analysis. As described in the **?@sec-02data**, half the battle is loading in data into your computer and connecting up these datasets. A typical next step is exploratory data analysis.

This chapter provides contrasts three of the most common programming languages for data science when faced with a common task. Borrowing language from ecology, an environment for data science as an interconnected system with a particular programming language as the foundation. Just like natural environments, environments for data science differ depending on the programming language. Let's begin.

1.1 Choosing different languages

We define an environment for data science as a programming language. Examples of programming languages include R, python, julia briefly described in Table 1.1

Table 1.1: Comparison of different programming languages used in data science.

Name	First Available	Description	Reference
R	1993	Started as a statistical programming language. Contributed packages allow for the extension of the base language into other data science programs.	https://www.r-project.org/

Name	First Available	Description	Reference
python	1991	Started as a computer programming language that supports object-oriented programming. Similar to R, contributed modules extend the core python language to machine learning, data visualization, and other areas of science, engineering, and mathematics.	https://www.python.org/
julia	2012	with the goal of combining speed and parallel language operation, essentially taking the best aspects of different programming languages and combining into one (al n.d.).	https://julialang.org/

As data scientists we are fortunate the variety of these programming languages exist! While one programming language may be preferable in some contexts than others, we should pick the tool that is suited to gain the best insight from that for what we need in the moment.

This textbook takes a neutral approach on *which* language makes the most sense (admittedly both of us have our favorites). Rather, let's examine differences between common environmental data science languages using the power of chatGPT.

Generative AI is an offshoot of machine learning methods from data science, and so provides a good case study to examine differences environments for data science. These artificial intelligence tools (e.g. chatGPT¹ and others) have rapidly transformed our daily lives (especially post 2023) and how we interact with the internet. For scientific research disclosing the use of generative AI tools is recognized as maintaining scientific integrity (Bockting et al. 2023; "Best Practices for Generative AI in Research | AJE" n.d.; Bertolo and Antonelli 2024). Let's use them here to contrast how the different environments in Table 1.1 produce output.

If we chose R, a prompt to chatGPT might be the following Figure 1.1:

Figure 1.2 shows its response, which (admittedly) a well-organized (and documented!) explanation of starter code:

The provided code loads up the correct library (tidyverse), converts the time

¹<https://chat.openai.com/>

**You**

write code to plot halfhourly temperature data using R with tidyverse syntax

Figure 1.1: An innocent question to chatGPT.

```
R Copy code

# Load the tidyverse package
library(tidyverse)

# Assuming your data frame is named temperature_data
# Make sure the 'timestamp' column is in datetime format
temperature_data$timestamp <- as.POSIXct(temperature_data$timestamp)

# Create a half-hourly plot
temperature_data %>%
  ggplot(aes(x = timestamp, y = temperature)) +
  geom_line() +
  scale_x_datetime(labels = scales::date_format("%H:%M")) +
  labs(title = "Half-Hourly Temperature Plot",
       x = "Time",
       y = "Temperature (°C)")
```

Figure 1.2: How chatGPT responded to our question using R.

to the POSIXct format (which makes working with dates and times easier) and generates a well-labeled plot. Not too shabby. Based on your knowledge of R, we would also award extra credit points for using the tidyverse pipe (%>%) in the code, but perhaps not full credit because of the adoption of the base R pipe² (|>).

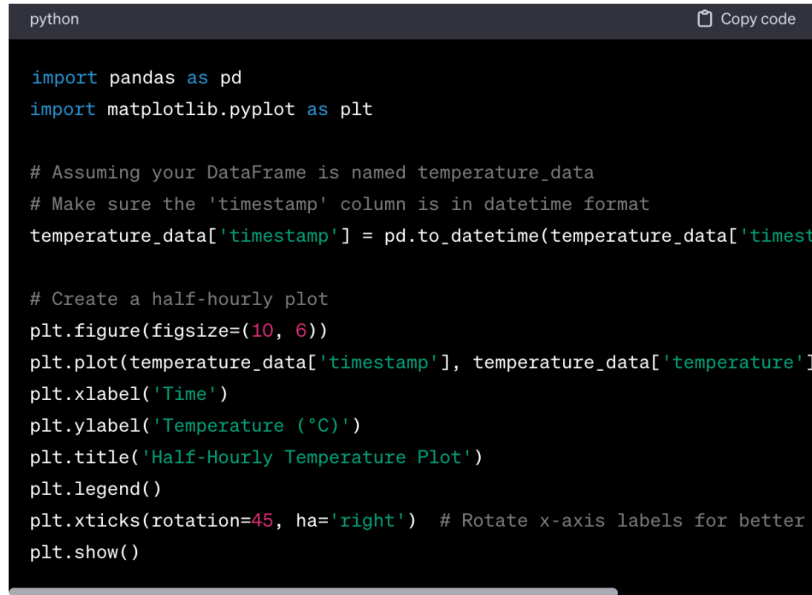
Now let's give the same prompt with python (Figure 1.3):

Examining the code seems like a beat for beat rehash of the same code with R, but with just python (almost like Star Wars: The Force Awakens³, but don't @ us!). There are some differences to note:

- Similar to R, in python libraries are defined at the start of the code (with import). However, those libraries need to be referred to when you wish to use a command from a particular library (e.g. the function `to_datetime` is a function in the pandas library). Thankfully in python you can abbreviate the names as `pd` - whichever makes sense for you. (NOTE: In R if you

²<https://www.tidyverse.org/blog/2023/04/base-vs-magrittr-pipe/>

³<https://www.vox.com/2015/12/26/10664834/star-force-awakens-derivative>



```
python Copy code

import pandas as pd
import matplotlib.pyplot as plt

# Assuming your DataFrame is named temperature_data
# Make sure the 'timestamp' column is in datetime format
temperature_data['timestamp'] = pd.to_datetime(temperature_data['timestamp'])

# Create a half-hourly plot
plt.figure(figsize=(10, 6))
plt.plot(temperature_data['timestamp'], temperature_data['temperature'])
plt.xlabel('Time')
plt.ylabel('Temperature (°C)')
plt.title('Half-Hourly Temperature Plot')
plt.legend()
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better
plt.show()
```

Figure 1.3: How chatGPT responded to our question with python.

only want to use a particular function in a library, then refer to it with the double colon (::, e.g. `PACKAGE::FUNCTION`).

- Python doesn't have the native pipe operator (`|>`) like R. The assignment operator is equals (`=`) versus a left facing arrow (`<—`).

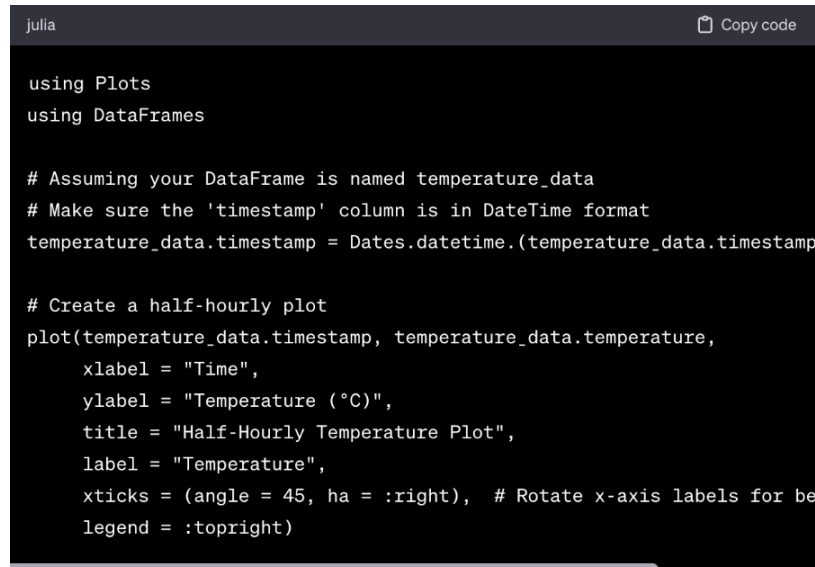
We finally asked chatGPT the same prompt, this time with julia (Figure 1.4):

Can you spot the differences (and similarities) with the Julia output in Figure 1.4 compared to R (Figure 1.2) or Python (Figure 1.3)? For all practical purposes, it comes down to preference - which one you are more familiar with.

So which should you choose? R is good for statistics and data manipulation. There is a strong user base in ecology (Lai et al. 2019). At the same time, the tidyverse and associated packages have been developed with data science in mind (Wickham and Grolemund 2017; Wickham et al. 2019).

The same case could be made for python - which at its core is a programming language (R was initially designed for statistical analysis which has morphed into wider applications for data science). If you are familiar with C or C++, python may feel very familiar to you.

julia is a very promising language with a user base growing in biology (Roesch et al. 2023). We believe there is longevity in this language, although it has a smaller user community in the environmental sciences.

A screenshot of a Julia REPL window. The window has a dark background with light-colored text. In the top right corner, there is a tab labeled 'julia' and a 'Copy code' button. The code displayed is as follows:

```
using Plots
using DataFrames

# Assuming your DataFrame is named temperature_data
# Make sure the 'timestamp' column is in DateTime format
temperature_data.timestamp = Dates.datetime.(temperature_data.timestamp)

# Create a half-hourly plot
plot(temperature_data.timestamp, temperature_data.temperature,
     xlabel = "Time",
     ylabel = "Temperature (°C)",
     title = "Half-Hourly Temperature Plot",
     label = "Temperature",
     xticks = (angle = 45, ha = :right), # Rotate x-axis labels for better readability
     legend = :topright)
```

Figure 1.4: How chatGPT responded to our question using julia.

A non open-source alternative to programming is MATLAB⁴, which for the authors was the first foray into programming languages in graduate school. This software is used in engineering and industrial applications. Because it is a commercial and proprietary language, it is not as translatable for open science applications. Octave⁵ is an open-source alternative as MATLAB. From John's experience working with Octave, at the time it was a good substitute for MATLAB, but it was hard to share code with his collaborators in biology.

1.2 Our recommendations

We believe that the best position to take is one of openness to learning new tools and software as your needs will invariably evolve. It is okay to dabble! John started out using MATLAB, then transitioned to Octave because it was open-source. He then moved to work with R (at the same time as the tidyverse was growing in use) as a fresh start to learn new tools and techniques for managing data.

Naupaka learned C++⁶ in high school, then didn't program much until

⁴<https://www.mathworks.com/products/matlab.html>

⁵<https://octave.org/>

⁶<https://en.wikipedia.org/wiki/C%2B%2B>

grad school, where I learned perl⁷ in a genomics course. When I started analyzing my own microbial community ecology data for my dissertation, I needed the statistical tools only available in R so started teaching myself. Then along the way I taught myself SQL, bash⁸, python, and a little bit of [lisp]([https://en.wikipedia.org/wiki/Lisp_\(programming_language\)](https://en.wikipedia.org/wiki/Lisp_(programming_language))).

1.3 The future: why choose?

Our experiences with programming languages reflect the times we developed as scientists and the growth of different languages. While we can't predict the future, but we can safely say at some point you may use programming language not even mentioned here. Our graduate school advisors programmed in C⁹ or Fortran¹⁰ - which while still in use, would not what we would recommend to our students as a first programming language. Organisms in a biological environment adapt and evolve to change - and so will you.

Siloing yourself in an particular programming language is becoming less relevant with the proliferation of online tools such as quarto¹¹, jupyter notebooks¹², and google colab¹³. These alternatives borrow strengths from each language. We recognize that code switching can be challenge ("Comparison of Programming Languages (Syntax)" 2024). The best environments for data science shares a common theme: they are built to support your success and longevity.

1.4 Exercises

1. Make an inventory of programming languages you have learned.
2. Interview your advisor about what programming languages they used, and what they would recommend.
3. Give a simple table and give a prompt to import in as a csv and plot with a programming language. Compare output. Is it what you would expect?

⁷<https://www.perl.org/>

⁸<https://www.gnu.org/software/bash/>

⁹[https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))

¹⁰<https://fortran-lang.org/>

¹¹<https://quarto.org/>

¹²<https://jupyter.org/>

¹³<https://colab.research.google.com/>

References

- al, Viral Shah, Stefan Karpinski. n.d. “Why We Created Julia.” <https://julialang.org/blog/2012/02/why-we-created-julia/>. Accessed April 19, 2024.
- Bertolo, Riccardo, and Alessandro Antonelli. 2024. “Generative AI in Scientific Publishing: Disruptive or Destructive?” *Nature Reviews Urology* 21 (1): 1–2. <https://doi.org/10.1038/s41585-023-00836-w>.
- “Best Practices for Generative AI in Research | AJE.” n.d. <https://www.aje.com/arc/best-practices-generative-ai-in-research/>. Accessed February 3, 2024.
- Bockting, Claudi L., Eva A. M. van Dis, Robert van Rooij, Willem Zuidema, and Johan Bollen. 2023. “Living Guidelines for Generative AI — Why Scientists Must Oversee Its Use.” *Nature* 622 (7984): 693–96. <https://doi.org/10.1038/d41586-023-03266-1>.
- “Comparison of Programming Languages (Syntax).” 2024. *Wikipedia*, April.
- Lai, Jiangshan, Christopher J. Lortie, Robert A. Muenchen, Jian Yang, and Keping Ma. 2019. “Evaluating the Popularity of R in Ecology.” *Ecosphere* 10 (1): e02567. <https://doi.org/10.1002/ecs2.2567>.
- Roesch, Elisabeth, Joe G. Greener, Adam L. MacLean, Huda Nassar, Christopher Rackauckas, Timothy E. Holy, and Michael P. H. Stumpf. 2023. “Julia for Biologists.” *Nature Methods* 20 (5): 655–64. <https://doi.org/10.1038/s41592-023-01832-z>.
- Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolemond, et al. 2019. “Welcome to the Tidyverse.” *Journal of Open Source Software* 4 (43): 1686. <https://doi.org/10.21105/joss.01686>.
- Wickham, Hadley, and Garrett Grolemond. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st edition. Sebastopol, CA: O’Reilly Media.



Index

julia, [1](#)

python, [1](#)

R, [1](#)