Computational Tools

This text uses the Python 3 programming language, along with a standard set of numerical and data visualization tools that are used widely in commercial applications, scientific experiments, and open-source projects. Python has recruited enthusiasts from many professions that use data to draw conclusions. By learning the Python language, you will join a million-person-strong community of software developers and data scientists.

A programming language is, loosely speaking, a structured subset of natural language (words) and special characters (e.g. , or {) that allow humans to describe operations they would like their computer to perform on their behalf.

The programming language translates these words and symbols into instructions the computer can execute.

Why Python?

Among the hundreds of programming languages to choose from, we chose to teach you Python for the following reasons:

- Easy to learn and use (relative to other programming languages).
- Designed with readability in mind.
- Excellent tools for handling data efficiently and succinctly.
- Cemented as the world's third most popular programming language, the most popular scripting language, and an increasing standard for data analysis in industry.
- General purpose: Initially you will learn Python for data analysis, but it can also used for websites, database management, web scraping, financial modeling, data visualization, etc. In particular, it is the world's best language for gluing those different pieces together.

However, the general purpose nature of Python comes at a cost: it is often said that Python is "the best language for nothing but the second best language for everything".

We aren't sure this is true, but a more optimistic view of that quote is that Python is a great language to have in your toolbox to solve all sorts of problems and patch them together.

A versatile "second-best" language might be the best one to learn first.

Some other languages to consider:

- R has an impressive ecosystem of statistical packages, and is defensible as a choice for pure data science. It could be a useful second language to learn for projects that are entirely statistical.
- Matlab has much more natural notation for writing linear algebra heavy code. However, it is: (a) expensive; (b) poor at dealing with data analysis; (c) grossly inferior to Python as a language; and (d) being left behind as Python and Julia ecosystems expand to more packages.
- Julia is in part a far better version of Matlab, which can be as fast as Fortran or C. However, it has a young and immature environment and is currently more appropriate for academics and scientific computing specialists.

Another consideration for programming language choice is runtime performance. On this dimension, Python, R, and Matlab can be slow for certain types of tasks.

Luckily, this will not be an issue for data science and the types of analysis we will do in this course, because most of the data analytics packages in Python (and R) rely on high-performance code written in other languages in the background.

Why Open Source?

Software development has changed radically in the last decade, increasingly becoming a process of stitching together both established high quality libraries, and state-of-the-art research projects.

A major disadvantage of Matlab, Stata, and other proprietary languages is that they are not open-source, and unable to work within this new paradigm.

Forgetting the cost for a moment, the benefits of using an open-source language are pragmatic rather than ideological.

- Open source languages are easier for everyone in the world to write and share packages because the code is accessible and available.
- With the right kinds of open source licenses; academics, businesses, and hobbyists all have incentives to contribute.
- Because open-source languages are managed on publicly accessible sites (e.g. GitHub), it is easier to build a community and collaborate.
- Package management systems (i.e. a way to find, download, install, and upgrade packages) in open-source languages can be very open and accessible since they don’t need to deal with proprietary software licenses.