Resources for Getting Started with Python, R, and Text Analysis

Jason T. Kiley [*]
Oklahoma State University

August 7, 2015

Software

As you begin with Python and R, I suggest getting the following two free multi–platform apps. A good text editor is a great tool for text analysis in general, and RStudio makes R a little more user friendly.

1. Atom is a free general purpose text editor. Among (many) other things, a good text editor will highlight the syntax of programming languages and give you a lot of control over how the text displays and wraps. This makes it easier to read, write, and edit code. Atom is relatively new, but it has a large community building add-on packages for it, and it has full time developers who work for GitHub.
2. RStudio is an integrated development environment ("IDE") for R. While I am partial to Atom for editing `.R` scripts, I like RStudio for entering commands in the interpreter, keeping track of objects, and exploring data.

Python

Python is a general purpose programming language with many built–in features and a relatively easy learning curve. It is useful for a variety of data prep and analysis tasks, and it is relatively easy to produce data as a CSV that can be readily imported into your favorite analysis software.

[*]My thanks to Laura K. Nelson, who suggested several of the resources listed here.

One issue that often comes up with Python is the choice between Python 2.x and 3.x. As the Python Software Foundation will tell you, "Python 2.x is legacy; Python 3.x is the present and future of the language." Unless you need to use a module that has not been updated for Python 3.x, I suggest that you target Python 3.x. Among other things, it handles Unicode text gracefully, and that can save a lot of work with some sources of text.

1. Python is freely available from the Python Software Foundation.
2. Introduction to Computer Science and Programming Using Python on edX. This is a free online course on edX by MIT's computer science department. This has more computer science theory than other learning options, but that can be helpful for understanding why you want to use certain language features.
3. *Learning Python* by Mark Lutz. This book focuses more on coverage of Python as a programming language and less on computer science theory. Lutz has been teaching Python for many years, and his experience shows. With an eye toward doing text analysis, you may be more interested in chapters 1–20, 22–24, 33–34, and 37. Get an ebook; the print version is huge.
4. Python Documentation. The Library Reference has concise explanations of how various modules work. In general, I look here when I think that there should be a function for the thing I want to do. For working with text and exporting CSVs, `re`, `os.path`, `fileinput`, `glob`, and `csv` are helpful.
5. Stack Overflow is a free site for enthusiast programmers, and it covers many languages, including Python. Given the design, it is very helpful when you want to do a specific thing in Python. I often search for what I want to accomplish with Google (e.g. python read text files) and look for results from Stack Overflow. If the example is not exactly what you are looking for, look at the related section in the side bar to the right.
6. Udacity's Intro to Computer Science course uses Python and seems to have a similar approach to the edX course above.
7. Neal Caren's Learning Python for Social Scientists is a website with a great collection of Python tutorials that are particularly relevant to our field.
8. I'm not typically a fan of the IDEs for Python for everyday use, but you may find PyCharm Educational Edition and its tutorials helpful when learning.

R

R is an open source programming language and environment that is aimed at statistical and graphics applications. It's useful and powerful, though the learning curve can be quite high. While I often use Stata for running analyses, I use R for assembling and cleaning up data sets.

1. R is freely available from the R Project.
2. *R for Everyone* by Jared Lander is my favorite book for learning R, though you should expect to spend a lot of time searching for specific solutions. If you frequently use another statistical computing environment, you can often find equivalent commands by using Google to search (e.g. `stata merge in R`).
3. For Stata users in particular, *Getting Started in R~Stata* is a great resource that provides a lot of coverage of equivalent commands in R and Stata.
4. I rarely find the built–in R documentation to be particularly helpful. There are exceptions, but you will likely find other resources to be more helpful. Instead, try searching on Rseek.
5. R-bloggers is a blog aggregator that links to hundreds of blogs that cover R. It's less useful for learning directly, but it is very helpful for discovering new packages or seeing useful applications of packages that you're familiar with.

Text Analysis and Processing

Text analysis is a large field that cuts across computer science, linguistics, and a variety of social science domains. With that in mind, this is a selected list of libraries and resources that I have either used or seen applied in management or similar fields.

1. TextBlob is a Python library that makes is relatively easy for beginners to use natural language processing and get meaningful results quickly. To install it, type `pip3 install textblob` at the command line (e.g. Terminal on a Mac).[1]
2. NLTK is the premier Python library for working with text. The authors have a free book, *Natural Language Processing with Python*. It is widely used and has a large community, so it is likely that you can find solutions and samples for many uses by searching. To install it, type `pip3 install nltk` at the command line, though, if you already installed `textblob`, it installed NLTK for you.

[1] Using `pip3` assumes that you are using Python 3.x. If you are using 2.x, the proper command may be `pip`. For more advanced users, consider using Python's virtual environment feature for each of your projects.

3. For gathering data from web pages, Beautiful Soup is a Python library for web–scraping and page parsing. Given its popularity, examples (including ones for particularly thorny problems) are readily available by searching.

4. Justin Grimmer's syllabus for his course on text analysis has a nice reading list covering a variety of text analysis topics.

5. `topicmodels` is an R package that implements LDA models for discovering topics from a corpus. As an example, see Tim Hannigan and Robert Vesco's topic model using data from last year's AoM program.

Additional Tools and Concepts

The tools below are very helpful once you have a grasp of the programming basics above.

1. Linters analyze code in real time and provide errors and warnings that help you spot and address errors quickly and conform to community norms for the style of your code. Atom has an optional `linter` package that is required for using the various linters available. For Python, I prefer `flake8`, but there are several options available.

2. Version control software allows you to capture snapshots of your project at various times with comments of what your changes are intended to accomplish. Most versioning software is intended for software engineering, so the complexity is somewhat high. That said, you can learn to use it for academic projects in less than a day. I prefer Git, and I use it with GitHub (free Micro plan for academics) and the GUI app SourceTree. For learning to use Git, I like *Version Control with Git* by Jon Loeliger and Matthew McCullough.

3. Learn to use exceptions in Python. At some point when processing large amounts of data, you will run across the rare issue that causes your program to stop with an error. Some of these can be hunted down and fixed in the raw data, but it is often easier to let an exception handler take care of corner cases that arise once every 10,000 or 100,000 documents. If you're new to Python, this may not be clear, but read up on exceptions when you find yourself with this problem.

4. Find a visual `diff` tool that you like for comparing two (or three files). `diff` is a command line utility that compares files line by line, and a visual diff tool provides a nice graphical interface. Primarily, a tool like this is helpful for comparing versions of your code, but it is also very helpful for comparing two similar texts. This is often useful for comparing multiple texts from a single source (e.g. press releases from one firm) when exploring your raw data.