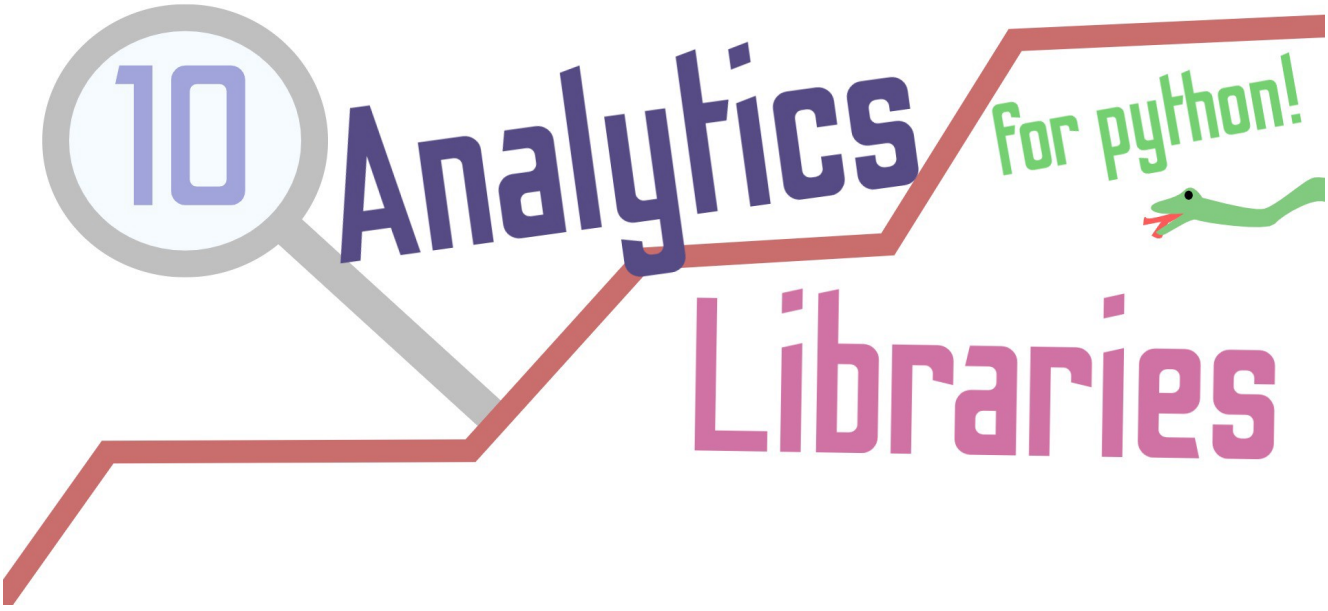
10 Of My Favorite Python Libraries For Data Analysis

A quick rundown of some great analytical packages you should be using in Python.

Emmett Boudreau



(Image by author)

Introduction

The most popular general purpose programming language on planet Earth right now is the Python programming language. This is not only because Python is incredibly easy, and relatively fast, but also that Python has a killer ecosystem with tools to fit every discipline

from business to finance and science. One area that this is certainly proven to be true is in the wonderful world of data science.

Of course, Python is also the most popular programming language used today for data science. While most scientists working with the language might be familiar with a lot of well-known and widely used packages such as Scipy, Sklearn, and Matplotlib, there are some packages which most data scientists have never even heard of that are also quite awesome!

№1: Plot.ly

To start us off, I am going with a library for data visualization that is pretty well-known, but some might have never heard of. Plot.ly is a graphing library that takes interactivity to a whole new level. I would genuinely advise using Plot.ly over something like Matplotlib or Seaborn. This is because Plot.ly comes with a multitude of different tools that most scientists can certainly come to appreciate. Just how much is in Plot.ly?

> Well...

Basic Graphing

Plot.ly comes preloaded with all of the fantastic tools that your average data scientist or even just computer programmer might expect. Scatter plots, bar charts, and

line charts are all staples of the Plot.ly module. While Matplotlib can accomplish similar goals, Plot.ly has the same functionality while also having default styling and Java-script interactivity that makes it a lot more fun and a lot easier to explore data. On top of that, presentations are certainly a thing that could be served a benefit from using Plot.ly over many of its competitors.

On top of all of that, Plot.ly also has some lesser-known charts and graphs that you would be hard-pressed to find in most other data visualization packages. Funnel charts, pie charts, violin charts, and tree maps are just a few examples of some unique and fun ways to explore data using the Plot.ly library.

3D

On top of the beautiful styling for basic plots, Plot.ly comes with a fully-featured 3D visualization system that rivals some of the best available with modern technology. One great advantage to the way that Plot.ly handles 3D is performance. Although 3D visualizations are of course quite complex and intensive, Plot.ly seems to handle a lot of these applications with relative ease. 3D visualizations are also awesome because the new axis allows for a greater amount of understanding by exploring data's positional values in 3D space. Furthermore, the Z axis can also represent a variety of features — making it possible to view several different correlations at once.

Maps

One thing that has become surprising in the recent development of technology is geo-data. Geo-data has evolutionized with FIPS to such an extent primarily because of smart-phones and global positioning systems (GPS,) and this has given way to a strong foundation for data science to build on top of. This is of course because data is now more readily available than it ever has been for free with geo-data included.

It is hard to argue with Plot.ly's beautiful approach to maps, as well. Plot.ly's clorepleths in particular are an absolute joy to use and can make compelling visualizations in a matter of seconds.

Financial

Although the financial side of things is certainly not where most of my domain knowledge falls, Plot.ly has great support for several financial visualizations. This goes to show Plot.ly's versatility as well, as visualizations rely both in and out of many scientist's domain. Like everything in Plot.ly, financial charts are easy to use and beautiful — a while ago I worked on a project where we tracked the value of currencies and created a full web-app with an LSTM model that would retrain itself and predict new data based on what it had learnt. Plot.ly really came in handy with its easy to use and Javascript integrated visualizations for candlestick data in this circumstance.

Statistical

To add to the unreasonable power of Plot.ly, Plot.ly also supports several different types of statistical plots. Furthermore, Plot.ly's statistical plotting is incredibly mature and makes things like plotting distributions relatively simple and easy.

Multi-language

Another great thing about Plot.ly is that it is written in C. As a result, it uses the LLVM compiler libraries and is compatible with a whole array of programming languages with just a simple API. Not only is Plot.ly available for Python users, but also R users, C users, and any other language's user base that is willing to implement it. There is even an implementation for the Julia language using the Plots.jl package.

Open-source

The last great thing about Plot.ly is that it is open-source. While this isn't as big of a benefit to some, being able to make changes if required or understand how code is being ran on your machine is always a plus. Being open-source software also means that Plot.ly can be used in practical business applications, making it valuable not only to those who program as a hobby, but also for real-world professionals.

№2: GGplot

The second great tool I would recommend for data analysis with Python is GGPlot.py. Any scientist who has experience in the R programming language has likely used GGPlot or GGPlot2. Both the R and Python package are fantastic and make plotting a breeze and furthermore in-depth. While the R implementation is certainly worth checking out, the Pythonic version is certainly pretty awesome as well.

Statistically-focused

A significant difference between the GGPlot package and the Plot.ly package for Python is that GGPlot is much more statistically-focused. Plot.ly is a great tool for data visualization, but is also targeted at more typical data visualization, rather than specifically statistical plotting.

Geometry-based

Another significant thing to note about GGPlot is its approach to graphics and visualization. Generally, GGPlot is more geometrically modular. This means that compositions can often be added to or altered quite easily. This is further beneficial to data science, as often there is a certain idea or data point that might want to be emphasized more than others.

№3: Bokeh

Bokeh is another interactive plotting library that was built with modern web-browsing and computing in mind. Similarly to Plot.ly, Bokeh might not be as extendable as something like GGPlot, but presents a host of benefits over the former by being incredibly integrated with Javascript. Every data scientist loves interactive visualizations, and Bokeh often takes this to a very high-level and simple API with stunning results.

Similar to Plot.ly

Bokeh as a visualization tool is incredibly similar to Plot.ly, and that isn't a bad thing at all. This means that Bokeh can be used to create beautiful and interactive visualizations incredibly easily.

Graphs

While Bokeh is similar to Plot.ly in a lot of ways, this is certainly not the case in what is possible with Bokeh. Not only is there support for all of the typical visualization techniques, but Bokeh also allows a user to create interesting and interactive visualizations of nearly anything. One relatively common use for Bokeh is the visualization of network graphs, which is really cool! On top of that, it does come with the geo-data visualizations that one might expect from the Plot.ly library.

№4: SymPy

Steering away from the world of visualization is a fantastic Python package called SymPy. MATLAB, Mathematica, and Julia programmers, hold on to your hats — SymPy is a Python module that allows Pythonic programmers and scientists to use more mathematical bindings. This allows the language to shift from a typically more software engineering approach to mathematics to the other end of the spectrum where code is frequently written as math.

Cool libraries

Many awesome implementations of Pythonic code and math use SymPy for mathematical calculations on the back-end. Here are some notable examples pulled straight from their homepage:

- **Cadabra**: Tensor algebra and (quantum) field theory system using SymPy for scalar algebra.
- **ChemPy**: A package useful for chemistry written in Python.
- **Lcapy**: Experimental Python package for teaching linear circuit analysis.
- **Spyder**: The Scientific Python Development Environment, a Python equivalent to Rstudio or MATLAB; full SymPy support can be enabled in Spyder's IPython Consoles.

Lambdify

Easily the coolest thing that SymPy has to offer is the ability to lambdify any expression or function inside of the language. I have talked before about why I think Lambda is one of the greatest tools available for Python developers. If you would like to read an article all about that topic, you can check it out here:

Scientific Python With Lambda

In the last ten years, the Python programming language has brought itself into the minds of many in the domain of...

While Python's lambda is great as an in-line argument on its own, it becomes even more powerful with the Lambdify function from SymPy. Here is how it works:

First, you can create an expression to be used as a mathematical function, in this example sin(x).

```
import numpya = numpy.arange(10)expr = sin(x)
```

That function can then be put through lambdify and become a very functional version of itself:

```
f = lambdify(x, expr, "numpy")
```

This is the basis for all of SymPy and makes scientific computing with the Python programming language a lot

easier. The use of this function can actually allow Python to explore one of the benefits to the Julia language, syntactical expressions. If you'd like to learn more about how the Julia language uses expressions, you can check out a full tutorial on it here:

How To Use Syntactical Expressions And Dispatch In Julia

The bread and butter of Julia: Syntactical expressions and multiple dispatch

№5: Blaze

The Blaze Ecosystem is a set of Pythonic libraries that make it far easier to query and process data in the Python programming language. The Blaze ecosystem actually consists of several different packages:

- Blaze
- Dask
- DataShape
- DyND
- Odo

These are all fantastic, useful, and well-made tools for the Python programming language. However, I will be focusing on the two that I have used the most and found to be the most valuable as it pertains to scientific computing.

Blaze

Blaze is an interface for querying all sorts of data on entirely different storage systems. This is incredibly useful for retrieving and analyzing big data dumps that might need to be taken apart by an algorithm, and furthermore might be in separate parts of different storage mediums. It is an incredibly useful tool for those who work in those particular circumstances, and has saved my life multiple times.

Dask

Dask is an incredibly unique package that allows for simple and most of all mutable parallel computing in Python. Parallel computing in Python has always been somewhat of a challenge in the language, and furthermore, optimizations have become incredibly difficult with many of the biggest packages available to the language because the language was not built with the idea of parallel computing in mind.

That being said, as the language itself moves more into this direction, Dask is a great tool for consistency with GPU support and working with CuArrays. Dask also features dynamic task scheduling that is explicitly coded by the programmer involved and big data collections.

№6: Orange

Orange is a Python library for data mining. How does this pertain to data analytics? Orange has a whole ecosystem of awesome tools for collecting data that can later be used for analysis, and analysis cannot happen without data. That being said, data wrangling is a very important step in the process of data science, and Orange helps to make that process a little more simpler.

Even cooler, the module comes with a few classification and regression models that you can use on your freshly mined data!

№7: Gensim

Genism is a Python library for topic modeling. What is great about Genism is that it is both easy to use, and very powerful. Using Gensim, you can create scalable statistical semantics that take up more memory than your computer has that can also be deployed into a real production environment. Furthermore, Genism can easily be implemented to perform natural language processing.

№8: Theano

Theano is another mathematical Python library that allows a programmer to work with copious amounts of data efficiently. While it might not match the mathematics-centric nature of something like SymPy, it does a lot really cool and interesting things that make linear algebra in

Python a lot more fun. On top of that, Theano is very accurate and relatively quick compared to a lot of similar solutions. On top of that, the package is tightly integrated with Numpy and dynamically integrates C code to make Python run faster.

№9: SciPy

If you've been using Python for statistics for even a week, it is likely that you have used SciPy. SciPy is the go-to package for statistical testing in the Python language. SciPy is not only generally a fast package to use, but it is also venerable and has been proven overtime to work incredibly well. The combination of SciPy, Pandas, and Numpy are what make Python a great language for general scientists and even more-so data scientists to work in.

Inclusive

One thing that can definitely be said about SciPy is that the module is very inclusive. While this might mean that occasionally it will be hard to find exactly what you need, it also means that everything you need is at your fingertips — a good trade, in my opinion. The SciPy library brings everything from statistics to distributions and even differential equations into easy to understand and simple functions that anyone could use!

№10: Seaborn

The last awesome tool I would recommend for data analysis in the Python programming language is a Pythonic classic:

> Seaborn.

Seaborn is an extension of MatPlotLib.PyPlot that integrates statistical plotting into the module. Seaborn provides a high-level interface for drawing, manipulating, and working with beautiful statistical graphics. While the appearance of Seaborn might pale in comparison to something like Bokeh or Plot.ly, it is incredibly significant in the margins of speed. When working with datasets with a lot of observations, it might be a good idea to try and use Seaborn rather than Plot.ly or Bokeh for your data visualizations. The simplicity of Seaborn is its greatest benefit, as it makes it fast and easy to use with familiar bindings from MatPlotLib.

Conclusion

If one language used by modern developers has the best ecosystem for data analysis and visualization, it is most likely Python. Python has tools that are not only great for statistical observations, but are also great for making the language act more statistical in itself. Furthermore, in terms of visualization, few languages rival the amazing

modules that the Python programming languages gives to Python-based scientists. That being said, these are some modules I really like — but there are a lot more, so it would certainly be interesting to know more about those in the responses!