Python visualization resources

Eric E Monson, PhD
Duke University Libraries
Center for Data and Visualization Sciences
askdata@duke.edu

Below you will find a quick guide to:

1. Resources to help you customize your Altair charts to better tell stories with your data
2. My workflow for producing final charts when it's not practical for me to do everthing through code
3. Alternatives to Altair when plotting with Python, along with a few of the reasons I would use those modules

Vega-Altair

Vega-Altair project home

> "Vega-Altair is a declarative visualization library for Python. Its simple, friendly and consistent API, built on top of the powerful Vega-Lite grammar, empowers you to spend less time writing code and more time exploring your data."

(You will commonly see this module referred to just as Altair, but the official name is Vega-Altair.)

Altair is what I most often use for visualization in Python, because like Tableau, it makes it quick and easy to explore the patterns in data and try out different visual representations so I can figure out how to best tell my story. It's also easier for me to remember how to use because it has a consistent syntax for visualizations rather than forcing me to remember a different name for every type of plot. If you come to my workshop, this is what I'll be teaching you.

- **Getting Started** for a nice overview
- **User Guide** has a lot of detail is very well written
- **Gallery of Examples** has an amazing variety of visualizations with code for each

Customizing Altair visualizations

Altair User Guide: Customizing Visualizations

From the above documentation:

> "Altair's goal is to automatically choose useful plot settings and configurations so that the user is free to think about the data rather than the mechanics of plotting. That said, once you have a useful visualization, you will often want to adjust certain aspects of it. This section of the documentation outlines some of the ways to make these adjustments."

On that page, they detail the three different levels at which mark and plot configuration can happen, plus show you how to customize things like:

- titles and subtitles
- axis ranges and axis labels
- legends
- bar widths
- how to create a "theme"

Typical workflow for final charts to tell stories

While I think Altair is a great tool to begin telling my data story in Python because it helps me explore my data to find patterns, easily try out different visual representations, and use good visualization principles, it was not really created to "tell stories". My typical workflow when creating my final figures is to:

- **Get as close as possible to the final figure in the software** I'm using, so I get as close as possible to my final product in a reproducible way. That means in the code:
    - Applying proper colors for lines, bars, points, etc., including appropriate continuous or categorical color scales, plus using color to highlight some graphical marks over others
    - Setting line widths
    - Adding data labels, if needed
    - Rotating and sizing text so it's readable
- Then **export to SVG or PDF** to preserve vector nature of the graphical elements instead of creating pixels
- Import into **PowerPoint or Adobe Illustrator to add additional explanatory graphical or text annotations**

It's much easier to place blocks of explanatory text, add lines and boxes, and arrange groups of visualizations in a program like PowerPoint or Illustrator rather than in code, so even though it's not reproducible, it saves me a lot of time and lets me tell my story in a more effective way.

- If I'm just adding annotations, PowerPoint works great, and many people already know how to use it.
- If I have to manipulate the basic graphic elements of the figure (e.g., change fonts or font sizes, change line widths or colors, delete unwanted elements like parts of axes or borders, etc.), then I almost always use Adobe Illustrator. (Although, you can now bring SVG graphics into PowerPoint, Convert to Shape, Ungroup, and then manipulate low-level elements – it's just not as easy as in Illustrator.) If you want to learn more about how to use Illustrator for these tasks, watch my workshop video on Adobe Illustrator for Charts and Graphs

For quick general guidance on designing your visualizations to tell stories, watch my Effective Data Visualization presentation, and for a talk that includes graphic design principles, watch my Effective Academic posters video.

Alternative Python plotting modules

I want you to know a little about the visualization landscape in Python. While I use Altair, because, like Tableau, it makes it easy to quickly explore data and potential visual representations for best telling my story, there are a lot of alternatives, depending on your preferences and application.

Pandas plotting

Pandas plotting documentation

If your data is in a DataFrame, and you want to quickly visualize different columns as separate series in basic plots, Pandas has quite a bit of built-in plotting capability. Pandas expects your data to not be "tidy", though, which is very similar to Excel, if you've ever made charts there.

Seaborn

Seaborn documentation

I like Seaborn a lot for plotting "tidy" data. What it has, that is nice, is a lot of complex statistical plots which are fast and easy to make if that's exactly the type of visualization you want to make. It has Matplotlib under the hood, but has a much easier high-level interface. For example, it has plots with regressions built in, or heatmaps which have been clustered using hierarchical clustering, plus things like "swarm plots" for showing all the individual points in a distribution without overlap. The main reason I don't use it all the time is that there is a different name for every type of plot, so I always have to look up how to make each plot.

Plotly

Plotly

Plotly is a Javascript library in the background, but they have written interfaces for R, Python, Matlab, etc, to allow you to create nice visualizations which are interactive out of the box. Besides all of the basic plots, they also have a lot of interesting chart types, like Sankey diagrams, plus things like tables, which you can't make with Altair.

Plotly uses "tidy" data, which is nice, but it relies on your having your data aggregated to the proper level of detail before plotting, unlike Tableau and Altair.

ggplot in Python with plotnine

plotnine documentation

I've never used this module, because I don't know R, but it would make it handy to go back and forth between R and Python using the same plotting syntax.

HoloViz

HoloViz home page

Holoviz is a group of related projects including Datashader for visualizing very large data, and a couple of high-level libraries for plotting, hvPlot and HoloViews. I have trouble keeping straight how all of these fit together, but they are their own ecosystem for visualization, including dealing with **large data that doesn't fit in memory**.

Bokeh

Bokeh documentation

Bokeh is a very good visualization module that is for, "creating interactive visualizations for modern web browsers". It was originally developed by the same people that created NumPy, iPython, SciPy, Matplotlib, which started as Continuum Analytics, then Anaconda, now NumFOCUS. So, it has strong roots in the world of high-performance data analytics in Python. It is a very good package, but I have never used it enough to become very comfortable with it, and it always feels like it has its own ecosystem which I have to re-learn every time I try it out again. For example, Bokeh has its own ColumnDataSource, which you can create from a Pandas DataFrame, but never seems to feel like it fits with Pandas out of the box like some other modules do. There are higher-level modules like hvPlot which make it quicker and easier to get up and running with Bokeh plotting.

They have a description in their User Guide about how Bokeh fits in with other modules, including Panel, and Datashader.

Matplotlib

While it is the grandfather of the Python plotting world, and it underlies other modules like Seaborn, I find Matplotlib too low-level to be useful when doing my own data visualization. I never use it directly, unless there is a specific visualization that I find code for which I can't make in other tools.

Revision: Fall 2023