



Python Data Visualization -- Huge leaps forward!

James A. Bednar, Senior Manager of Technical Services, Anaconda, Inc.

Day job: Managing our Services projects with clients

Night job: Managing our open-source visualization tools available at [HoloViz.org](https://holoviz.org), including [Panel](https://panel.holoviz.org), [hvPlot](https://hvplot.holoviz.org), [HoloViews](https://holoviews.org), [GeoViews](https://datashader.org), [Datashader](https://datashader.org), and [Colorcet](https://colorcet.holoviz.org).

Side job: Running [PyViz.org](https://pyviz.org), covering all viz and dashboarding tools.

It's been an exciting year since ACON 2019!

Major improvements in Python data visualization and dashboarding:

1. Several major new libraries for Python dashboarding
2. Widget library unification (Bokeh/Panel/ipywidgets)
3. Ultra-fast server-side rendering/aggregation
4. Use the APIs you already know
5. Annotators (Plotly/Bokeh/HoloViews)
6. Other new developments
7. What limitations does Python have for viz?
8. Exploring further

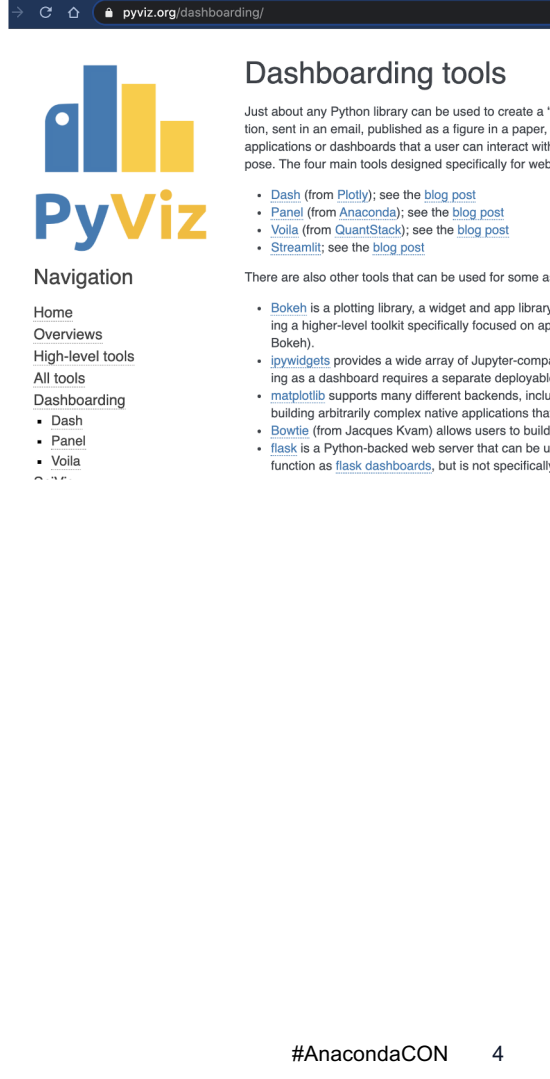
We'll cover each of these in turn.

1. Dashboarding

People used to say Python lagged R/Shiny for easy dashboard building tools, but now we have:

- Dash
- Panel
- Voilà
- Streamlit

(see pyviz.org/dashboarding/)



The screenshot shows the PyViz dashboarding page. At the top, there's a navigation menu with links to Home, Overviews, High-level tools, All tools, and Dashboarding. The Dashboarding section is expanded, showing links to Dash, Panel, and Voilà. Below the navigation menu, there's a section titled "Dashboarding tools" which explains that just about any Python library can be used to create a dashboard. It lists four main tools: Dash (from Plotly), Panel (from Anaconda), Voilà (from QuantStack), and Streamlit. There's also a section for "Navigation" and a list of other tools like Bokeh, ipywidgets, matplotlib, and Bowtie.

PyViz

Dashboarding tools

Just about any Python library can be used to create a dashboard, sent in an email, published as a figure in a paper, applications or dashboards that a user can interact with. The four main tools designed specifically for web:

- [Dash](#) (from [Plotly](#)); see the [blog post](#)
- [Panel](#) (from [Anaconda](#)); see the [blog post](#)
- [Voilà](#) (from [QuantStack](#)); see the [blog post](#)
- [Streamlit](#); see the [blog post](#)

Navigation

- Home
- Overviews
- High-level tools
- All tools
- Dashboarding
 - [Dash](#)
 - [Panel](#)
 - [Voilà](#)

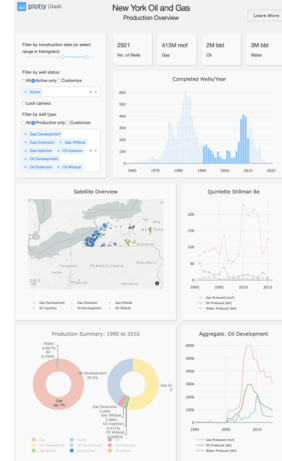
There are also other tools that can be used for some applications:

- [Bokeh](#) is a plotting library, a widget and app library, providing a higher-level toolkit specifically focused on applications (Bokeh).
- [ipywidgets](#) provides a wide array of Jupyter-compatible widgets, but using as a dashboard requires a separate deployable application.
- [matplotlib](#) supports many different backends, including building arbitrarily complex native applications that can be deployed as a dashboard.
- [Bowtie](#) (from Jacques Kvam) allows users to build dashboards as a Python-backed web server that can be used as a function as [flask dashboards](#), but is not specifically designed for dashboarding.

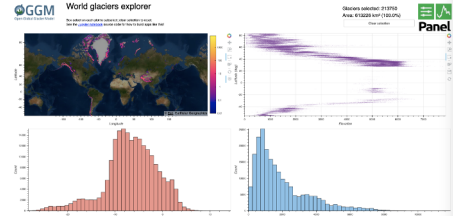
1. Dashboarding: Plotly Dash (2016)

Dash is now the old timer:

- Scalable server for Plotly plots
- Requires relatively heavy HTML/CSS knowledge
- Typically complex apps with callback structure
- Historically focused only on deployed apps
- [jupyter-dash](#) now allows incremental dashboard building
- [Dash Oil and Gas Demo](#) ([repo](#))

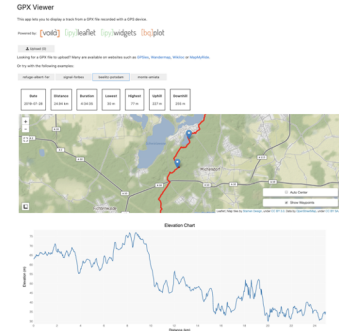


1. Dashboarding: Panel (Jun 2019)

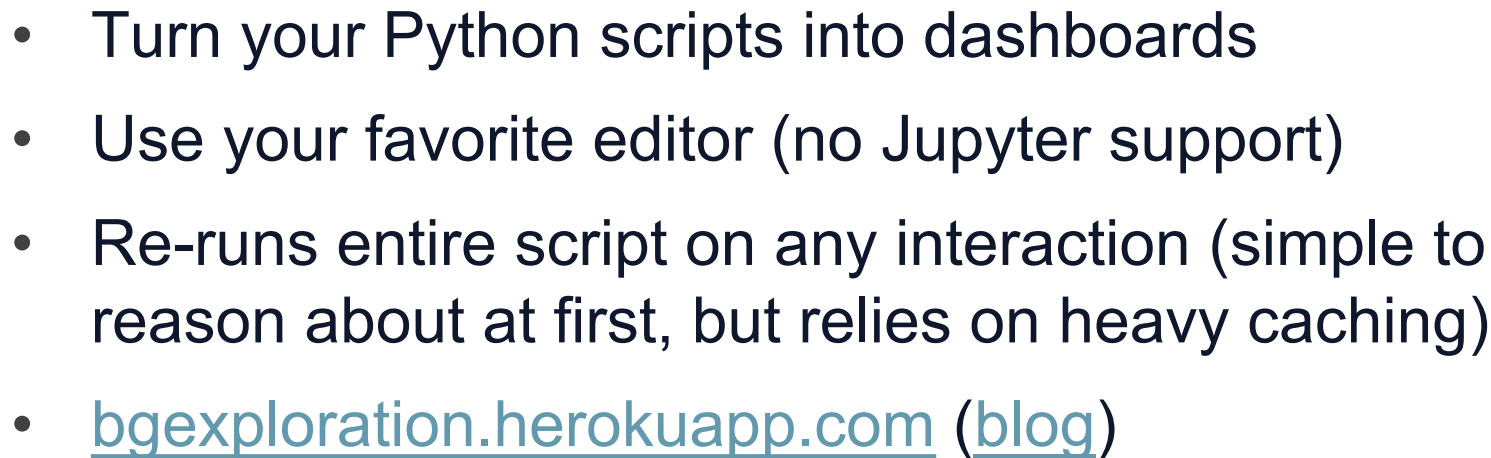


- Easy dashboarding for any plotting library
- Deep (but optional!) Jupyter integration for easy iteration/development
- Zero cost to switch from interactive prototype to deployed app and back
- Easy static HTML/CSS output with live widgets
- glaciers.pyviz.demos.anaconda.com

1. Dashboarding: Voilà (Jun 2019)





- Turns a Jupyter notebook into a shareable dashboard
- Deploys with a full Jupyter kernel (not scalable)
- To build complex layouts, uses ipywidgets or templates (or now Panel!)
- voila-gpx-viewer.pyviz.demo.anaconda.com



2. Widget library unification


- Previously: Jupyter supported ipywidgets, separate from Bokeh and Dash widgets
- Painful to switch between libraries
- Complex widgets (ipyvolume, VTK) usable with only one ecosystem
- Caused unnecessary split between user groups

2. Widget library unification

- New: [ipywidgets](#)  and [jupyter bokeh](#) 
- Wrap Bokeh models as ipywidgets, and vice versa
- Deploy Bokeh or Panel apps in Voila
- Deploy ipyvolume or bqplot in Bokeh Server
- Now everyone can get along:
 - Different notebook/deployment technologies have different strengths
 - Use whichever ones meet your needs!

3. Ultrafast server-side rendering/ aggregation




- [Datashader](#)  and [Vaex](#) (2016):
 - Initially standalone projects
 - Render billions of points to a pixelated representation
 - Only a fixed-size array is sent to the local browser for display
- Datashader now supports GPU/Dask arrays/dataframes
- Vaex renders points to 1D/2D/3D
- Datashader renders to 2D, now including [points](#)/[lines](#)/[areas](#)/[trimesh](#)/[quadmesh](#)/[rasters](#)/[polygons](#)

3. Ultrafast server-side rendering/ aggregation

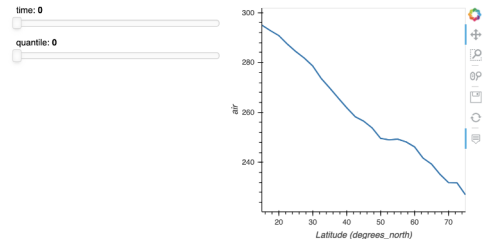
Server-side rendering now integrated into many plotting packages:


- vaex supports [bqplot](#), [matplotlib](#)
- datashader supported by [HoloViews](#)+Bokeh, [hvPlot](#), and now [Plotly](#)
- datashader also integrated into Nvidia [cuxfilter](#), [umap](#)
- [mpl_scatter_density](#) also available (points only) for mpl

4. Use the APIs you already know

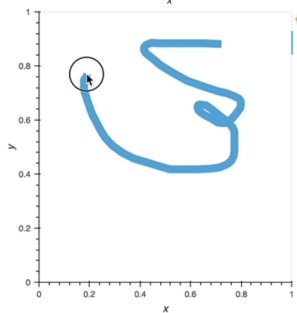
- Pandas Matplotlib .plot() API becoming even more widely supported as a standard:
 - Pandas plotting backend is now [officially configurable](#)
 - Now (partially) [supported by Plotly Express](#)
 - See pyviz.org/tools.html#high-level-shared-api
- [hvPlot](#)  now supports .plot() API for Pandas, Xarray, Dask, Streamz, Intake, GeoPandas, and NetworkX objects -- learn API once, use it everywhere!

4. Use the APIs you already know



- HoloViews  now offers .df and .xr accessors
 - Calls native Pandas or Xarray methods directly on data
 - Avoids need to learn HoloViews data API
- Upcoming: interactive support for building entire interactive apps directly from Pandas or Xarray APIs:
 - `time, q = IntSlider(...), FloatSlider(...)`
 - `ds.air.interactive(loc='left_top').isel(time=time).quantile(q=q, dim='lon').plot()`

5. Annotators/Drawing tools



- Both [Bokeh](#) and [Plotly](#) now offer shape-drawing tools:
 - user-generated shapes on top of plots
 - useful for capturing user annotations, labeling ML data, setting up analyses or simulations, ...
- Example for ML annotations:
examples.pyviz.org/ml_annotators

6. Other new developments

- New spatial-processing libraries: [xarray spatial](#), [spatialpandas](#), [pygeos](#), [scikit-geometry](#), [cuSpatial](#)
- New editor support for interactive plots in VSCode
- Colorcet: [256-color categorical colormaps](#)
- HoloViews: [automatic linked brushing](#)

7. What limitations does Python have for viz?

- Not much, now!
- Of course, some users will always want GUI BI tools (Tableau, Excel, etc.)
- Direct development of JavaScript apps will always be useful for the most scalable and responsive websites
- R users will prefer R solutions, of course!

8. Exploring further

- See full list of all Python viz tools at pyviz.org/tools
- See reproducible larger examples at examples.pyviz.org, awesome-panel.org, and awesome-streamlit.org
- See each package's website for more details
 - holoviz.org, panel.holoviz.org, hvplot.holoviz.org, datashader.org
 - plotly.com/python, plotly.com/dash
 - matplotlib.org, bokeh.org, altair-viz.github.io

 | ***Thank you.***