The Journal of Open Source Software

Syd: A package for making interactive data visualizations in python

Andrew T. Landau[1]

1 UCL Queen Square Institute of Neurology, University College London, London, United Kingdom

DOI: 10.21105/joss.08447

Software

- Review
- Repository
- Archive

Editor: Andrew Stewart

Reviewers:

- @ZeitgeberH
- @learningdebunked

Submitted: 01 May 2025
Published: 26 August 2025

Summary

Gaining an intuitive feel for the patterns in scientific data is fundamental to the scientific process. However, in the era of large datasets, it is not only challenging to understand the patterns in the data, but it is often a significant technical burden to simply plot the data in a satisfactory manner. For example, typical neuroscience datasets can have thousands to millions of neurons, spanning across recordings sessions and experimental subjects. Each neuron might contain a hint of a discovery. Similarly, studying the inner- workings of large-language models often comes down to looking at the activation patterns of thousands of neurons to understand how they represent text.

`Syd` represents a solution that supports interactive data analysis of rich datasets. It provides a python interface that enables the rapid construction of interactive plots, which enable scientists to quickly and easily look through all their data.

Statement of need

Although many packages enable the construction of graphical user interfaces in python, they all require some level of bespoke implementation and often depend on significant levels of boilerplate code. `Syd` relieves this need by providing a simple and opinionated interface for defining *what you want to plot* and *which parameters you want to be interactive*; it handles all the behind the scenes action required to make an interface.

`Syd` "viewers" are made by defining a plot function which accepts a `state` dictionary as input and returns a `matplotlib` `figure` object (Hunter, 2007). Then, parameters are added to the viewer with simple declarative functions. If required, `Syd` also provides an intuitive system for callbacks that greatly minimizes boilerplate GUI code. The simplicity of the interface means that interactive data visualization tools can become a fundamental part of a data analyst's typical workflow - as easy as making plots.

Because `Syd` handles the behind-the-scenes implementation code, it permits a user to "deploy" viewers in multiple environments without any changes to the code. Viewers can be deployed in jupyter notebooks for fast and local analysis or deployed on web browsers for sharing across local networks. These deployment environments depend on `ipywidgets` and `flask`, respectively (JupyterDevelopmentTeam, 2025; Pallets, 2024). In this way, a scientist can study their data at their desk, then use the same code to share the results interactively with their advisor or with their lab at different computers.

Why choose Syd?

There are many powerful packages in the python ecosystem for interactive data-visualization.

So, why use Syd?

Syd is a minimalist package that helps turn any matplotlib plot into an interactive data visualization tool with a few lines of code and minimal cognitive load. It is not a general-purpose, sophisticated dashboard framework. Researchers who need more advanced functionality or want to learn a more unique plotting package should turn to other options. However, Syd offers the simplest and most straightforward tools for enabling fast-paced, on-the-fly exploratory data analysis.

Capability / Package	**Syd**	**Panel**	**Plotly**	**Altair**	**Streamlit**
Interactive widgets (no full rerun)	Yes	Yes	Yes	Yes	No
Runs *inside* Jupyter/IPython	Yes	Yes	Yes	Yes	No
Boilerplate required for a simple GUI	**Tiny**	Low	Medium	Low	Low
Learning curve / cognitive load	**Tiny**	Low	Medium	Medium	Low
Native **Matplotlib** support	Yes	Yes	No	No	Yes
“Bring-your-own-plot function”	**First-class**	Yes	Yes	Partly	Yes
Separate web server needed	No	Yes	No	No	Yes
Primarily for interactive plots	Yes	No	Yes	Yes	No

Extra points

- Syd’s only abstraction is a state dict passed to your matplotlib function; nothing else to learn.
- Syd’s declarative style makes it’s code footprint tiny.

Example Viewer

This is a simple example that demonstrates how easy it is to make interactive visualization tools with Syd.

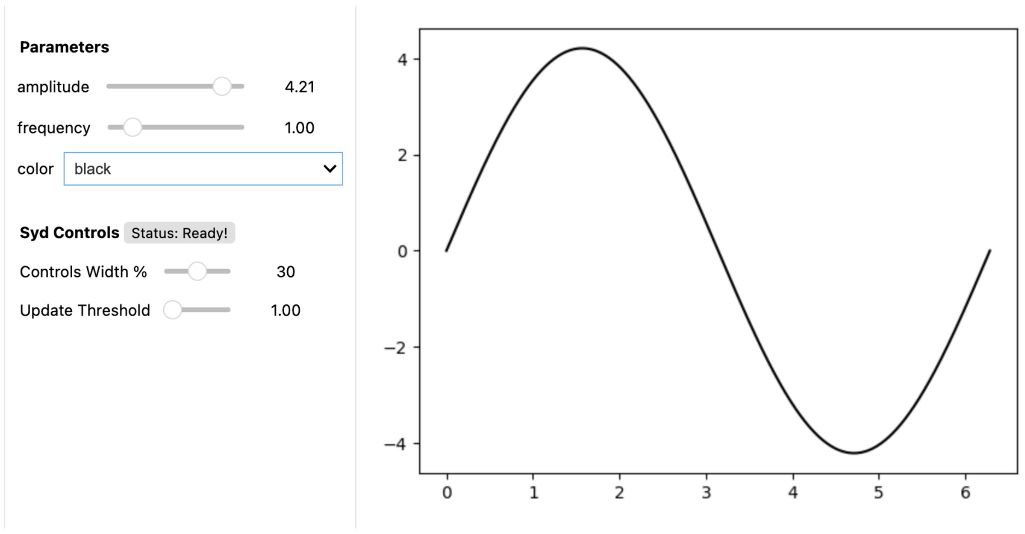
1. A plot function is defined that accepts `state` as input.
2. A viewer is created, and three parameters are added.
3. (Optional) Callback functions are defined and attached to parameters.
4. The viewer is deployed in the desired environment.

```python
import numpy as np
import matplotlib.pyplot as plt
from syd import make_viewer

# Plot function
def plot(state):
    amplitude = state["amplitude"]
    frequency = state["frequency"]
    color = state["color"]
    fig = plt.figure()
    t = np.linspace(0, 2 * np.pi, 1000)
    ax = plt.gca()
    ax.plot(t, amplitude * np.sin(frequency * t), color=color)
    return fig

# Viewer & Parameters
viewer = make_viewer(plot)
viewer.add_float("amplitude", value=1.0, min=0.1, max=5.0)
viewer.add_float("frequency", value=1.0, min=0.1, max=5.0)
viewer.add_selection("color", options=["black", "blue", "green", "red"])

# Deployment
viewer.show() # for viewing in a jupyter notebook
# viewer.share() # for viewing in a web browser
```



Parameters

amplitude 4.21

frequency 1.00

color black

Syd Controls Status: Ready!

Controls Width % 30

Update Threshold 1.00

Figure 1: Example Syd Viewer

Acknowledgements

We acknowledge the fantastic dependencies this package depends on, including `Matplotlib`, `ipywidgets`, and `Flask`. In addition, we acknowledge Jasmine Reggiani and members of the Cortexlab at UCL for their feedback.

References

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

JupyterDevelopmentTeam. (2025). *Ipywidgets*. https://github.com/jupyter-widgets/ipywidgets; GitHub.

Pallets. (2024). *Flask*. https://github.com/pallets/flask; GitHub.