Plotly & Dash 500

9 min read · Dec 15, 2022

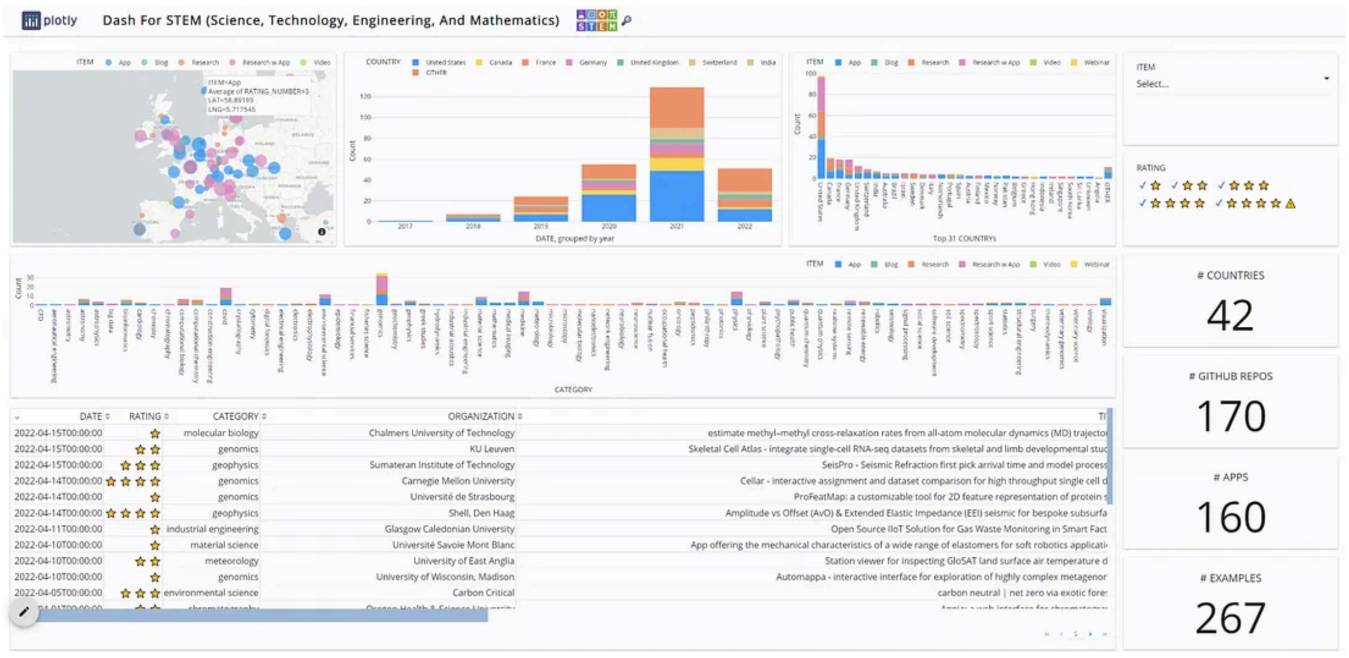
Plotly Follow

Written by: Daniel Anton Suchy

Hello, world! My name is Daniel Anton Suchy, and I am a data scientist at Plotly. Today, I'd like to share some information about the Plotly and Dash 500 (P&D500) app I recently created.

From real-world problems to Dash

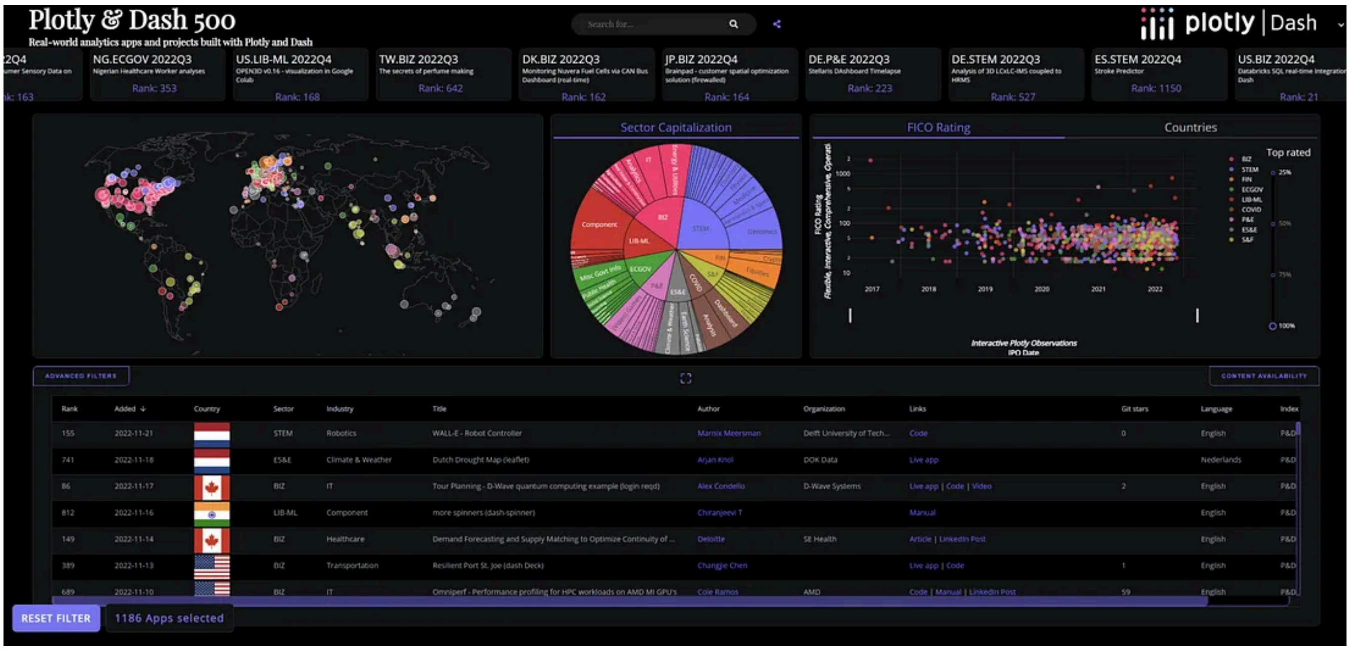
The idea of the "P&D500" was born from Dave Gibbon's strong initiative to create a project that would showcase the practical abilities of Dash and Plotly. Over the years, Dave has been engaging with app authors, collecting and promoting the best Dash apps he has seen, all the while slowly building this amazing dataset of real-world Dash applications.



DATE	RATING	CATEGORY	ORGANIZATION	TI
2022-04-15T00:00:00	★	molecular biology	Chalmers University of Technology	estimate methyl-methyl cross-relaxation rates from all-atom molecular dynamics (MD) trajecto
2022-04-15T00:00:00	★ ★	genomics	KU Leuven	Skeletal Cell Atlas - integrate single-cell RNA-seq datasets from skeletal and limb developmental stuc
2022-04-15T00:00:00	★ ★ ★	geophysics	Sumateran Institute of Technology	SeisPro - Seismic Refraction first pick arrival time and model process
2022-04-14T00:00:00	★ ★ ★ ★	genomics	Carnegie Mellon University	Cellar - interactive assignment and dataset comparison for high throughput single cell d
2022-04-14T00:00:00	★	genomics	Université de Strasbourg	ProFeatMap: a customizable tool for 2D feature representation of protein s
2022-04-14T00:00:00	★ ★ ★ ★	geophysics	Shell, Den Haag	Amplitude vs Offset (AvO) & Extended Elastic Impedance (EEI) seismic for bespoke subsurfa
2022-04-11T00:00:00	★	industrial engineering	Glasgow Caledonian University	Open Source IIoT Solution for Gas Waste Monitoring in Smart Fact
2022-04-10T00:00:00	★	material science	Université Savoie Mont Blanc	App offering the mechanical characteristics of a wide range of elastomers for soft robotics applicati
2022-04-10T00:00:00	★ ★	meteorology	University of East Anglia	Station viewer for inspecting GloSAT land surface air temperature d
2022-04-10T00:00:00	★	genomics	University of Wisconsin, Madison	Automappa - interactive interface for exploration of highly complex metagenor
2022-04-05T00:00:00	★ ★ ★	environmental science	Carbon Critical	carbon neutral | net zero via exotic fore:

P&D500 predecessor, created by Dave Gibbon using Dashboard Engine.

Dave doesn't have formal programming training, but he was able to use Plotly's simple drag-and-drop Dash Enterprise Dashboard Engine interface to build an impressive app. He received an immense amount of positive feedback, so he asked me to help him expand on it. That's how the current iteration of the "Plotly and Dash 500" project was born!



Rank	Added	Country	Sector	Industry	Title	Author	Organization	Links	Git stars	Language	Index
155	2022-11-21		STEM	Robotics	WALL-E - Robot Controller	Marnix Meersman	Delft University of Tech...	Code	0	English	P&D
741	2022-11-18		ES&E	Climate & Weather	Dutch Drought Map (leaflet)	Arjan Knol	DOK Data	Live app		Nederlands	P&D
86	2022-11-17		BIZ	IT	Tour Planning - D-Wave quantum computing example (login reqd)	Alex Condello	D-Wave Systems	Live app | Code | Video	2	English	P&D
812	2022-11-16		LIB-ML	Component	more spinners (dash-spinner)	Chiranjeevi T		Manual		English	P&D
149	2022-11-14		BIZ	Healthcare	Demand Forecasting and Supply Matching to Optimize Continuity of ...	Deloitte	SE Health	Article | LinkedIn Post		English	P&D
389	2022-11-13		BIZ	Transportation	Resilient Port St. Joe (dash Deck)	Changjie Chen		Live app | Code	1	English	P&D
689	2022-11-10		BIZ	IT	Omniperf - Performance profiling for HPC workloads on AMD MI GPU's	Cole Ramos	AMD	Code | Manual | LinkedIn Post	59	English	P&D

The main view in the P&D500 app.

The dataset currently stands at over 1,200 hand-picked apps. Here's a quick overview of Dave's process:

- The data collection process consisted of extensive repeated manual searches over a long period (GitHub, google, LinkedIn, research periodical search tools, etc.)
- The data curation process was manual (eye test, GitHub stars, the sophistication of workflow, the sophistication of visualizations, uniqueness, geographic/country of origin, the language of the app, etc.)
- For app ranking, part of it was qualitative, while the other part was quantitative

The first time I saw this dataset, I was blown away. Even as a data scientist, I had never seen such a clean dataset filled with such rich information! It was incredibly easy to work with: well-organized, clearly-defined data fields, followed good etiquette with the locations data which made it very simple to geocode, and the data itself is very rich. I was excited to see what we could do with it.

The Art of Possible

Are you interested in a specific topic? Simply type it into the search bar, and you will be presented with a list of related apps.

Want a wider range of apps related to your industry? Select the sector, and you will be presented with a selection of relevant applications to explore and take inspiration from.

Are you looking for an application for a particular region? You can choose from a list of over 20 languages!

Over 850 applications have their source code available on GitHub. Looking at existing code is a great way to learn and get inspiration, helping developers to further improve their skills and build better apps suited to their needs.

For 600+ apps, you will also be able to find a link to the live app, which is an easy way to see how the app looks and functions, without having to run the code locally.

Technical bit

So, how does the app work?

The simple answer is that 4 main callbacks enable the following:

1. Data filtering based on 16 filters

2. Snapshot saving and sharing
3. Modal displays when map/scatter/marquee is clicked
4. Sunburst chart transformation into an interactive medium

Now to the details.

1. **Data filtering**

Currently, 16 different filters can be used simultaneously to filter the data. That translates to a lot of visualization possibilities, but if managed incorrectly, applying all filters every time a user selects a new filter could slow down the app considerably.

This is remedied by two simple solutions:

- Add a simple "If" statement so that when a filter is not selected, code skips creating a mask for the Pandas df.
- Where possible, use the Redis cache.

The main example of the Redis cache is the free text search:

```
mask = df[["title", "organization"]].apply(lambda row:
row.str.contains(type_filter, case=False).any(), axis=1)
```

This expression can be computationally heavy since it loops through the whole DataFrame to find the apps that mention the text the user wishes to see.

We simply make a function out of it and use the `@cache.memoize()` decorator that will cache the results of the user's search.

```
@cache.memoize()
def text_filter(text):
  mask = df.apply(lambda row: row.str.contains(text, case=False).any(), axis=1)
  return mask
```

You might think, "How does this speed things up if the user searches for a different expression"? In that case, the cache does not exist yet, and the Pandas mask will be calculated normally. However, once the user is satisfied after having narrowed the data down, and uses a different filter, the cache for `text_filter()` will be used, and the expensive Pandas operation will be skipped.

Using Redis cache is a great way to speed up your app and make it more responsive. By adding a couple of lines of code, the response time of a large SQL call/data operation/function becomes milliseconds instead of seconds for subsequent queries.

Any Redis database can be used, however, if you have a Dash Enterprise license, it can be simply added via the App Manager. You can read more in the Redis documentation.

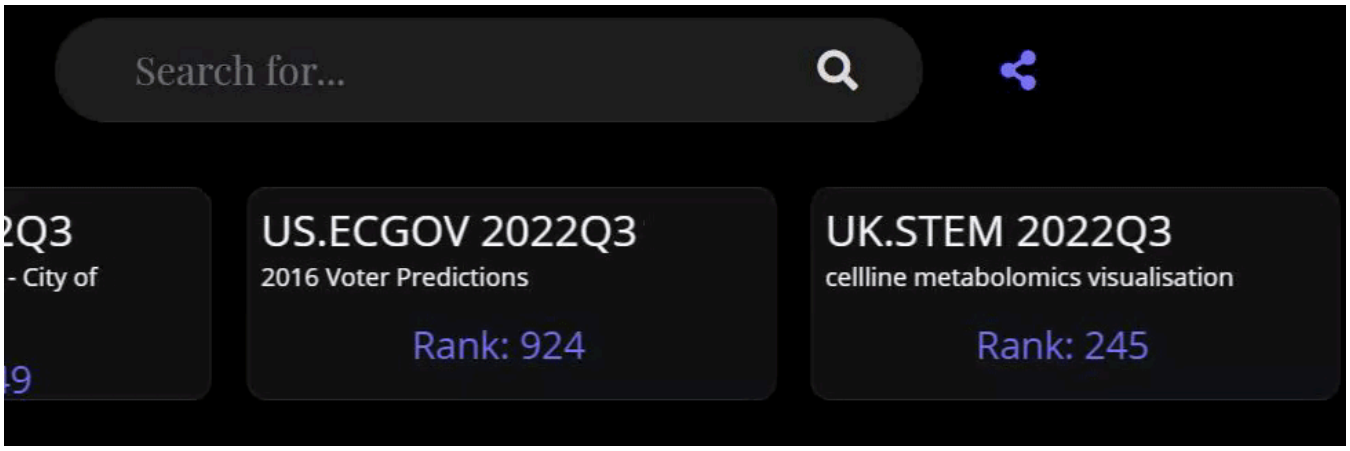
Another trick to speed up loading times is the Dash Ag-Grid table. While Ag-Grid has many great features, the main feature I utilized was pagination.

When pagination is enabled, the data passed into the table won't be fully loaded when initialized. The data is subsequently loaded as the user scrolls through the table. This behavior can be modified to suit the needs, however, you can expect response time in milliseconds.

How does this help? The data of the DataFrame you want to display has to be sent to the user and then processed by the browser. If you try to send a DataFrame that has 1MB, 10, 100… it will take a long time, and the experience can be varied among users with different computer specs, behavior that you have no control over. Pagination mitigates all of that, as well as the initial callback wait, and makes it virtually instantaneous.

2. Snapshot saving and sharing

You might have noticed a share button in the header of the app. When you click on it, your current app view (including all the filters) will be saved, and you will be given a shareable link.



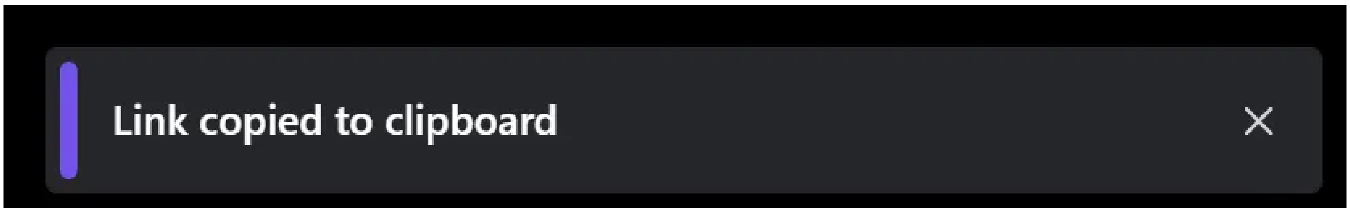
Share your current view of the application by clicking a single button.

There are two components to this functionality:

- Saving the state of the app — done with the usage of **`dash_snapshots`** Dash Enterprise package that simply allows us to run a one-line function **`snapshot_save`**`(content_to_save)` that will do the magic and store the content in the Redis database. This can then be retrieved whenever the unique link is later opened.
- Copying the link to the clipboard — is done through the following client-side callback:

```
clientside_callback(
  """
  function copyShareableLinkToClipboard (text) {
    var el = document.createElement('textarea');
    el.value = text;
    el.setAttribute('readonly', '');
    el.style = {position: 'absolute', left: '-9999px'};
    document.body.appendChild(el);
    el.select();
    document.execCommand('copy');
    document.body.removeChild(el);
  }
  """,
  Output("copy-to-clipboard-output-confirmation", "children"),
  Input("share-clipboard-trigger-text", "storage"),
)
```

Where **`share-clipboard-trigger-text`** is a unique link created by **`snapshot_save`**`(content_to_save)`.


Link copied to clipboard

Updating the user with a notification.

3. Modal displays

This modal contains more information about the app & author. It is triggered by the `clickData` callback when map/scatter/marquee is clicked.

When the figures are created, we store the index of the DataFrame inside the figure so that we can use it later to retrieve the data about that particular app when it is clicked on. An example of how to store additional data in figures can be found in the Plotly graphing library documentation, in our case we pass custom data as follows: `custom_data=[df.index]`.

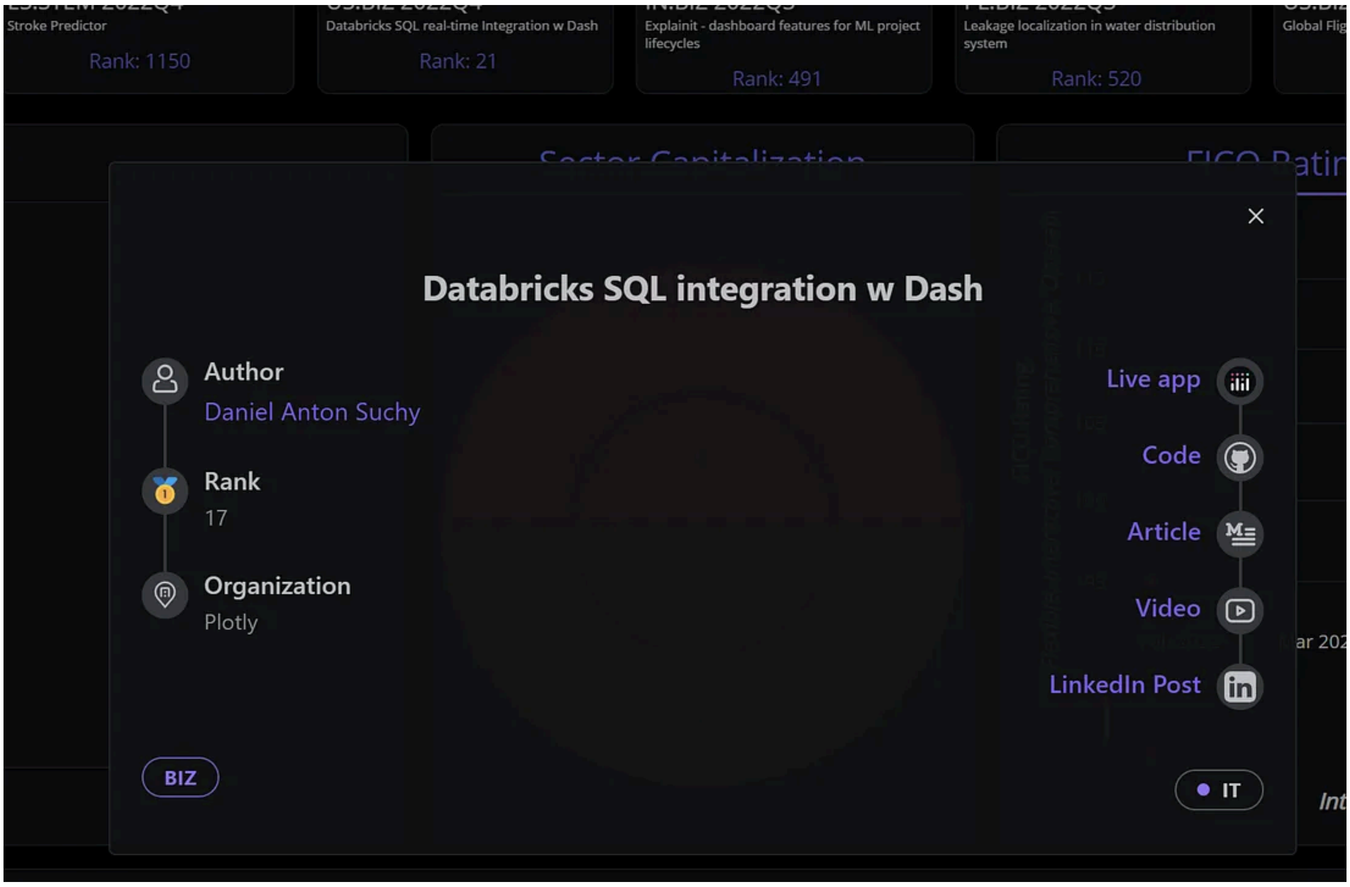
To determine which component was clicked `dash.callback_context.triggered` is used, and then depending on the source clicked, the appropriate data is retrieved:

```
ctx = dash.callback_context.triggered_id
if "fig-map" == ctx:
  index = clickData["points"][0]["customdata"][0]
elif "fig-fico-rating" == ctx:
  index = scatter_clickData["points"][0]["customdata"][0]
else: # ( marquee click)
  index = marquee_clickData["points"][0]["customdata"][0]
df_temp = df[df.index == index]
```

Plotly has a thorough explanation of how to use `dash.callback_context.triggered` in the Dash documentation, I would highly recommend trying it out, as it is a very powerful tool that lets you implement complex logic in your callbacks.

Once we have the data, we can populate the modal with it. The modal component is created using `dmc.Modal()` component and is populated in such a way that only the data (author LinkedIn, code, live app…) that is available for the application is displayed.

I decided to use the Mantine modal because of the synergy with `dmc.Timeline()` component which is a great way to display the data in an informative and visually appealing way!


Stroke Predictor
Rank: 1150
Databricks SQL real-time Integration w Dash
Rank: 21
Explainit - dashboard features for ML project lifecycles
Rank: 491
Leakage localization in water distribution system
Rank: 520
Global Flig

Databricks SQL integration w Dash

Author
Daniel Anton Suchy

Rank
17

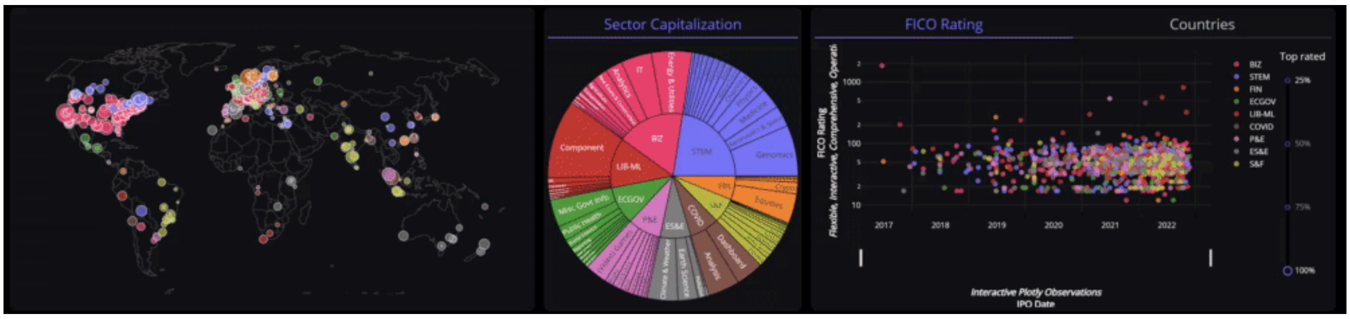
Organization
Plotly

Modal with more information about the clicked user app, created using Dash Mantine Components.

4. Sunburst chart

A sunburst chart is a great way to see how the data is distributed and what are the most popular topics.

To take this amazing figure type to the next level, we will utilize `Input("sunburst", "clickData")` callback to enhance the figure, and transform it into a filter that will allow us to filter the data based on the topic we are interested in.



Sunburst figure modified to act as a selector that cross-filters the data.

This is achieved by exploring the clicked data through `pie_click["points"][0]["currentPath"]` and `pie_click["points"][0]["entry"]`, and then comparing it to previously selected topics which are stored between callbacks in

```
dcc.Store(id="store-selected-sector"),
dcc.Store(id="store-selected-industry"),
```

Once the selection is made, we also alert the user that the data has been filtered by displaying a notification.

Notification at the bottom right of the user’s screen, updating the user about the action taken.

A few tips

When you are designing your app, keep the user experience top of mind!

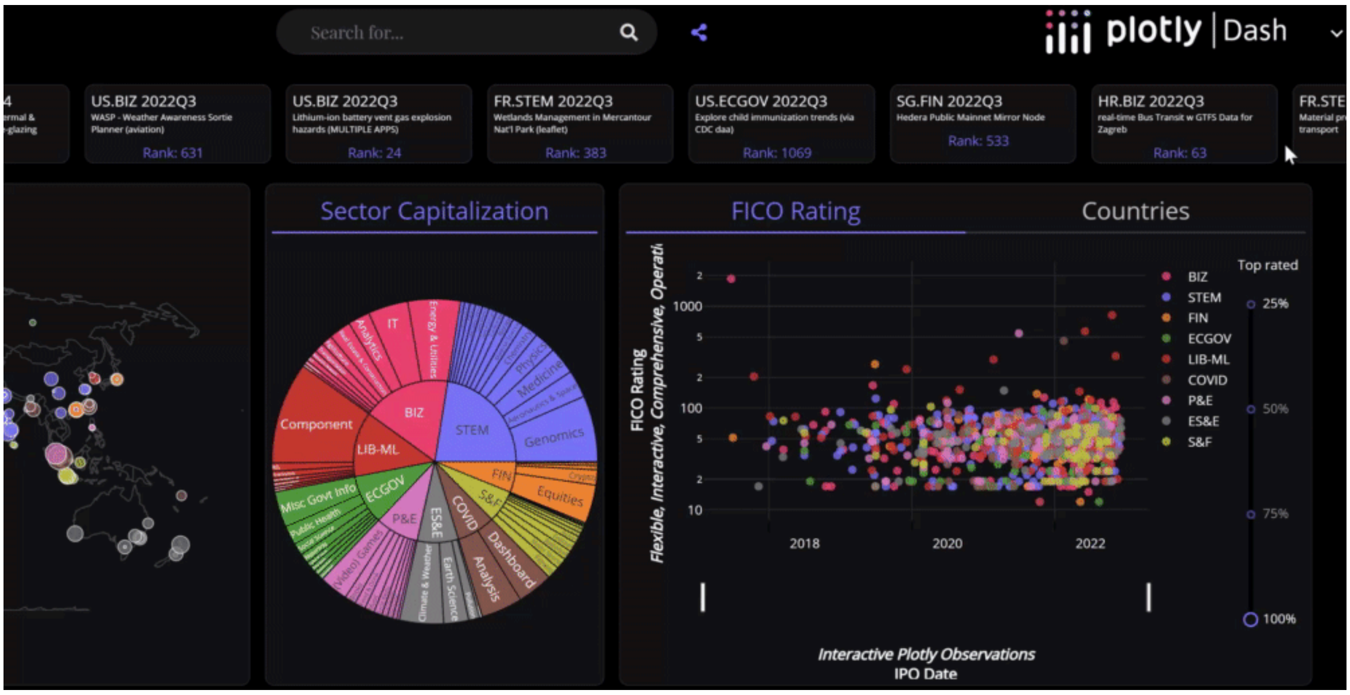
- Figure legend/axes titles
- Does it need to be shown? If yes, name and position them properly.
- Make sure the colors used remain consistent between the figures and the app itself.
- Control panel
- Figure controls are intuitive enough so that the user does not need to use the control panel, so consider removing it — it looks better when the never-used control buttons don’t phase in and out on hover!
- If you require the user to take a screenshot of the figure, remove the other controls and just keep the screenshot button!
- Hover Information Take your time to write your custom *hover template* and don’t forget to use **bold**, *italic*, underline, or even color!

- Notifications and overlays
- If you have a callback that re-plots a figure, use a loading overlay over the figure, to let the user know "look, this figure is now being changed because of the button you clicked!" at a single glance. The overlay will also remove the worry we all have on some pages where nothing seemingly happens and you start wondering if you really did click that button or not.
- Consider using notifications/tooltips to inform the user!
- Custom CSS
- Small things like changing the border color on hover can inform the user "look, this is intractable".
- You can easily create custom behavior for components (see marquee components in the P&D500 app).
- Also, make sure the fonts and colors are consistent throughout the app!

Thank you!

Whether you have found your app inside P&D500, or whether you just intend to use it, we want to thank you for taking part in this journey, and we hope you will find it helpful.

We will be adding more features to the whole experience and creating walkthrough videos too, so look forward to it!



You can submit your application at the top-right of the page.

If you have an app that you would like to be featured, you are welcome to submit it via the application's submission menu, found under a dropdown on the top right of the page. We are so excited to see what you will create!

For technical questions, please reach out to me on LinkedIn.
For other questions/suggestions, please reach out to Dave.

Data Visualization · Web App Development · Python · Dash · Plotly

Follow

Published in Plotly

4K followers · Last published 13 hours ago

Plotly is a data visualization company that makes it easy to build, test, and deploy beautiful interactive web apps, charts and graphs—in any programming language.

Follow

Written by Plotly

32K followers · 973 following

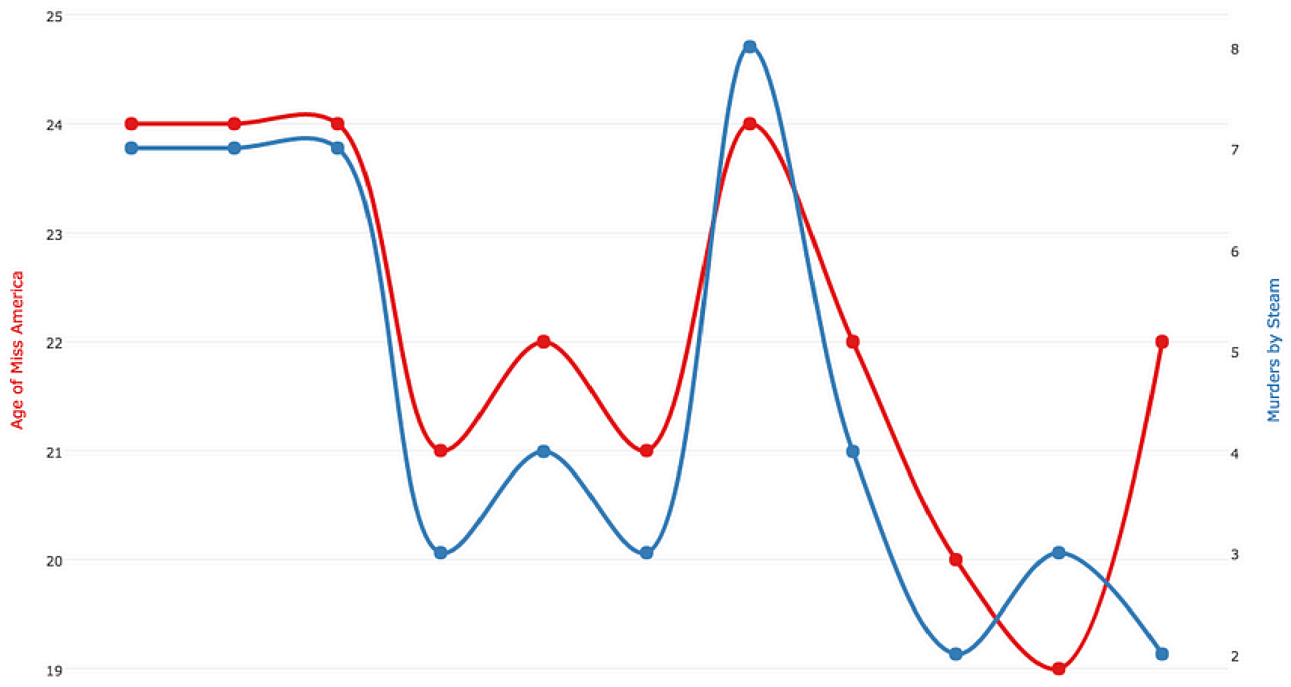
The low-code framework for rapidly building interactive, scalable data apps in Python.

No responses yet

Felipegenovese

What are your thoughts?

More from Plotly and Plotly

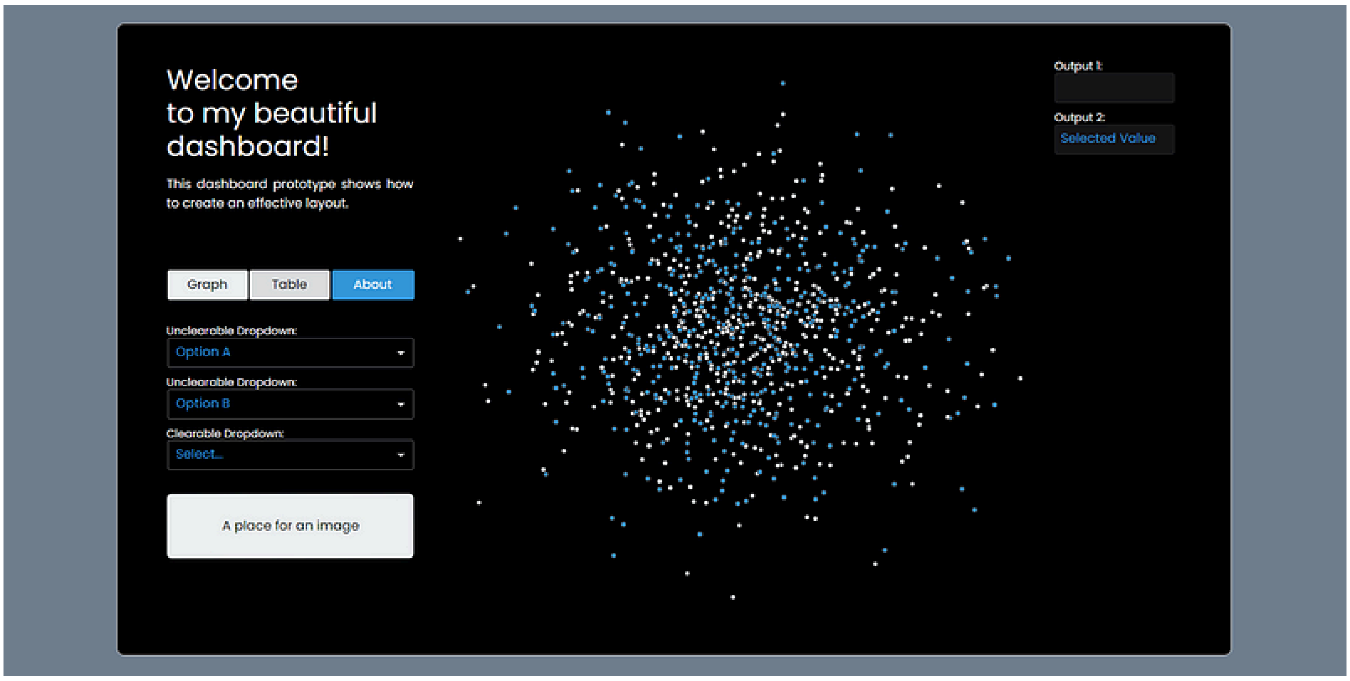


Plotly

Spurious Correlations

Correlation does not imply causation. These bizarre correlations prove it!

Jul 4, 2016 227 1



In Plotly by Tanya Lomskaya

How to create a beautiful, interactive dashboard layout in Python with Plotly Dash

When created in Python, a dashboard can have an impressive design, unique interactivity, and the highest processing speed. It sounds great...

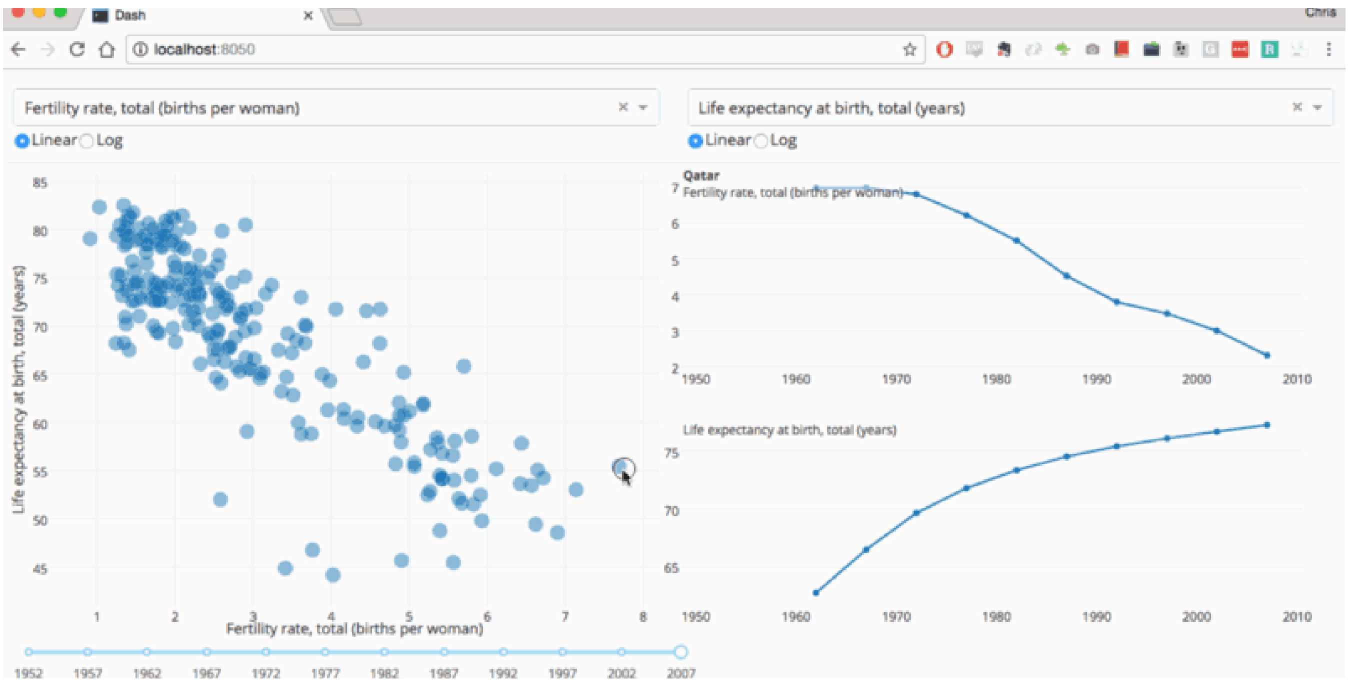
Apr 19, 2024 787 9

In Plotly by Plotly

How Plotly is Leading with AI, and Keeping People at the Center

Plotly's People Operations team sheds light on how Plotly is multiplying and democratizing the use of AI into every part of how we work.

Jun 30 1


Dash
localhost:8050
Fertility rate, total (births per woman)
Linear Log
Life expectancy at birth, total (years)
Linear Log
Life expectancy at birth, total (years)
Fertility rate, total (births per woman)
Qatar
Fertility rate, total (births per woman)
Life expectancy at birth, total (years)
1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007

In Plotly by Plotly

🌟 Introducing Dash 🌟

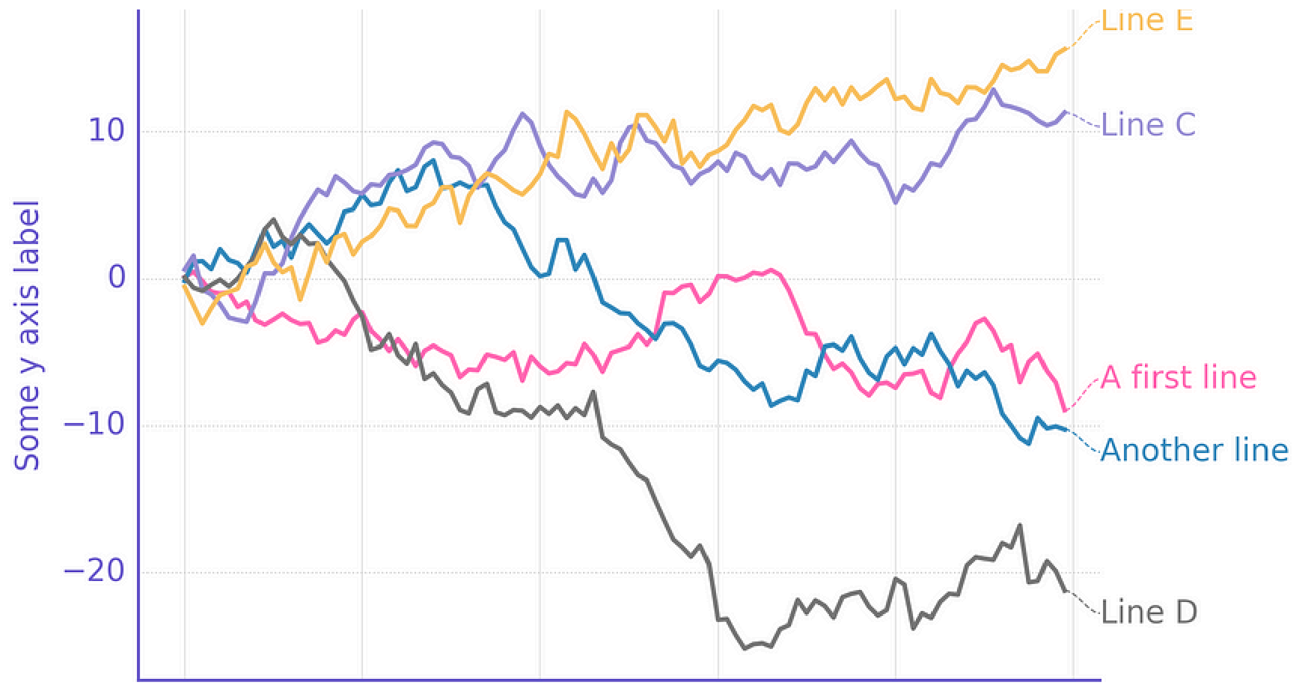
Create Reactive Web Apps in pure Python

Jun 21, 2017 7.3K 22

See all from Plotly

See all from Plotly

Recommended from Medium



PY In Python in Plain English by Zlatan B

Making Line Plots Delightful with Optimized Direct Labeling

Improve your Matplotlib line plots by replacing the default legend

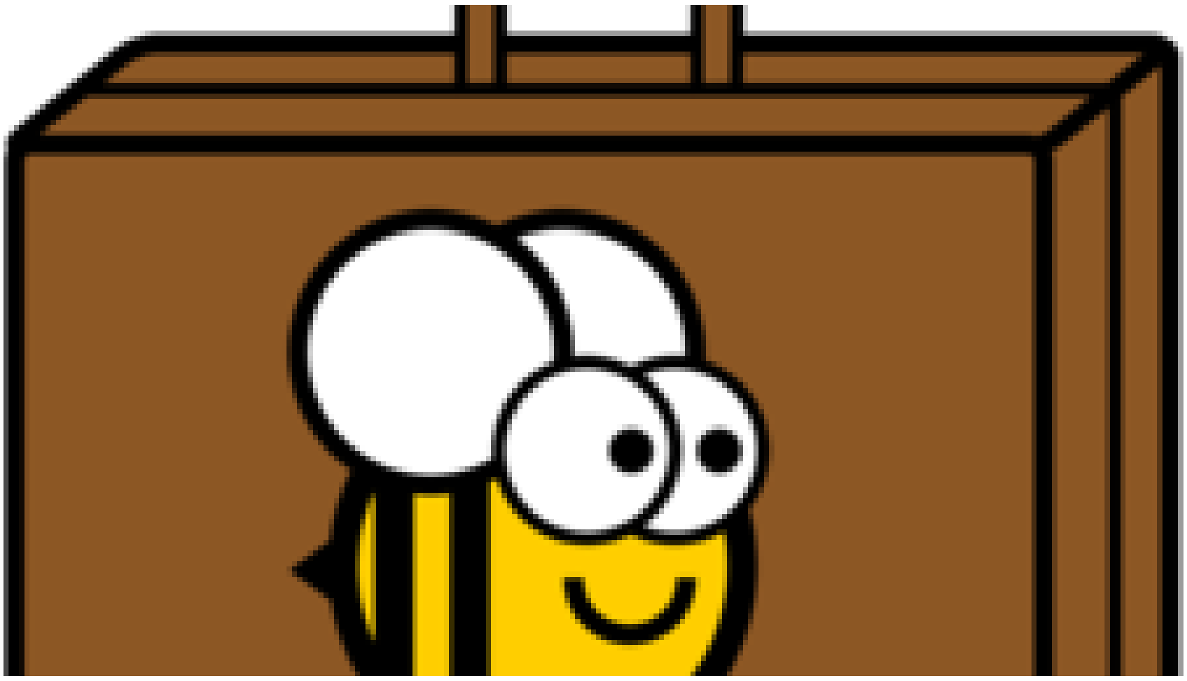
Jun 17 5

Linking

Streamlit + Matplotlib: Visualization Dashboards

Hello,

Apr 21

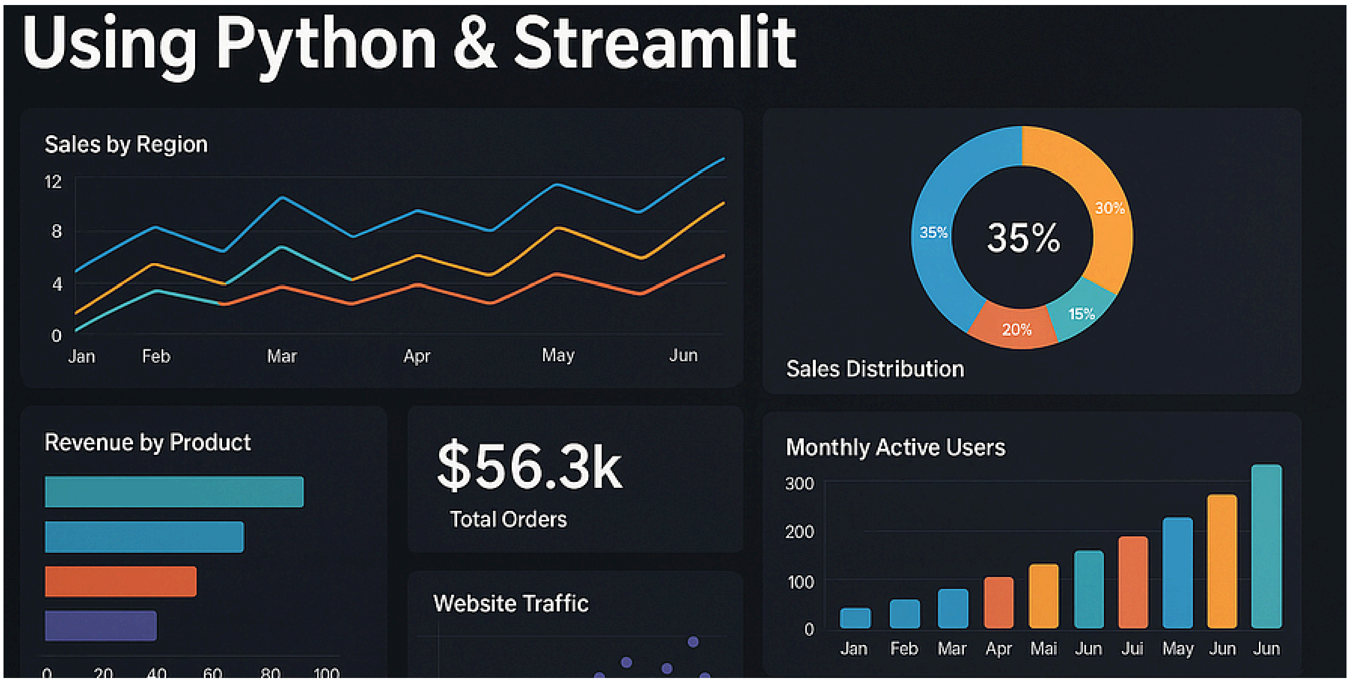


muyiwa akin-ogundeji

How to package a python desktop app for Windows with briefcase

This is part 2 in a 3-part series where I share my experiences building a Python desktop app using pywebview and flask, and then packaging…

Feb 4 1 2

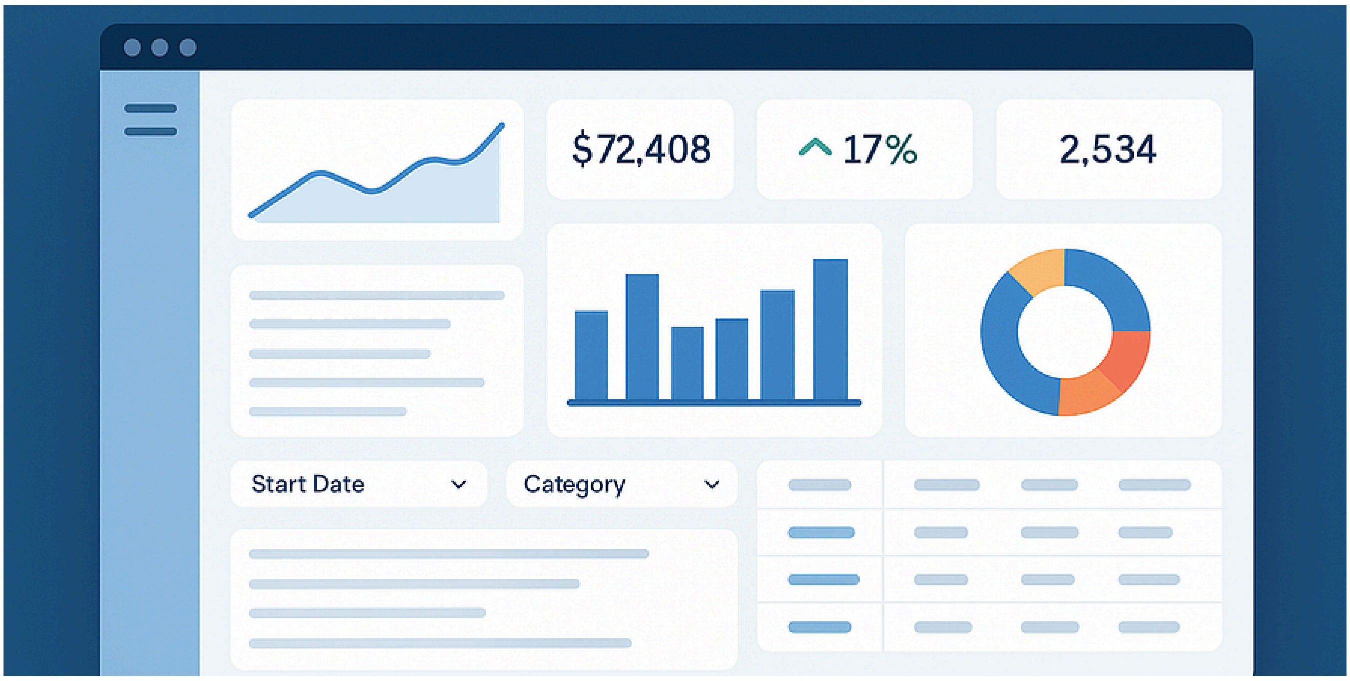


Nikulsinh Rajput

The Dashboard I Built in 20 Minutes Using Python & Streamlit

How Streamlit turned my messy CSV files into a polished, interactive data app—with almost no web dev.

Jun 18 46 3

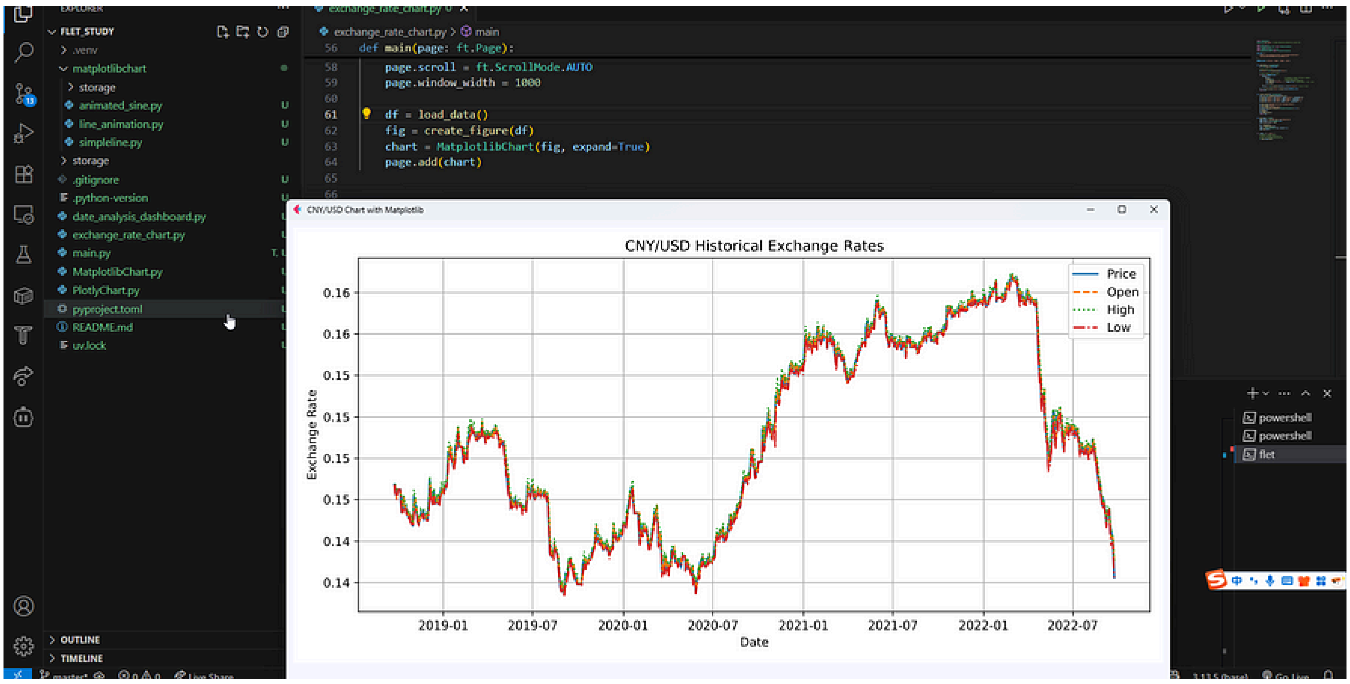


In Level Up Coding by Thomas Reid

Building a modern data dashboard

Using Python and the Gradio library

Jul 7 89 2



```
def main(page: ft.Page):
    page.scroll = ft.ScrollMode.AUTO
    page.window_width = 1000

    df = load_data()
    fig = create_figure(df)
    chart = MatplotlibChart(fig, expand=True)
    page.add(chart)
```

exchange_rate_chart.py
CNY/USD Chart with Matplotlib
CNY/USD Historical Exchange Rates
Exchange Rate
Date
Price
Open
High
Low

Dr. Shouke Wei

Visualizing Real-World Data with Python and Flet

Pandas, Matplotlib, Flet, MatplotlibChart, Exchange Rate Dataset

Jul 2 10

See more recommendations