Frontera Hacks
A Dashboard

Kristen Hallas

PhD Student @ UTRGV

GitHub Repo

- https://github.com/kristen-hallas/UTRGV-FronteraHacks23-Dash

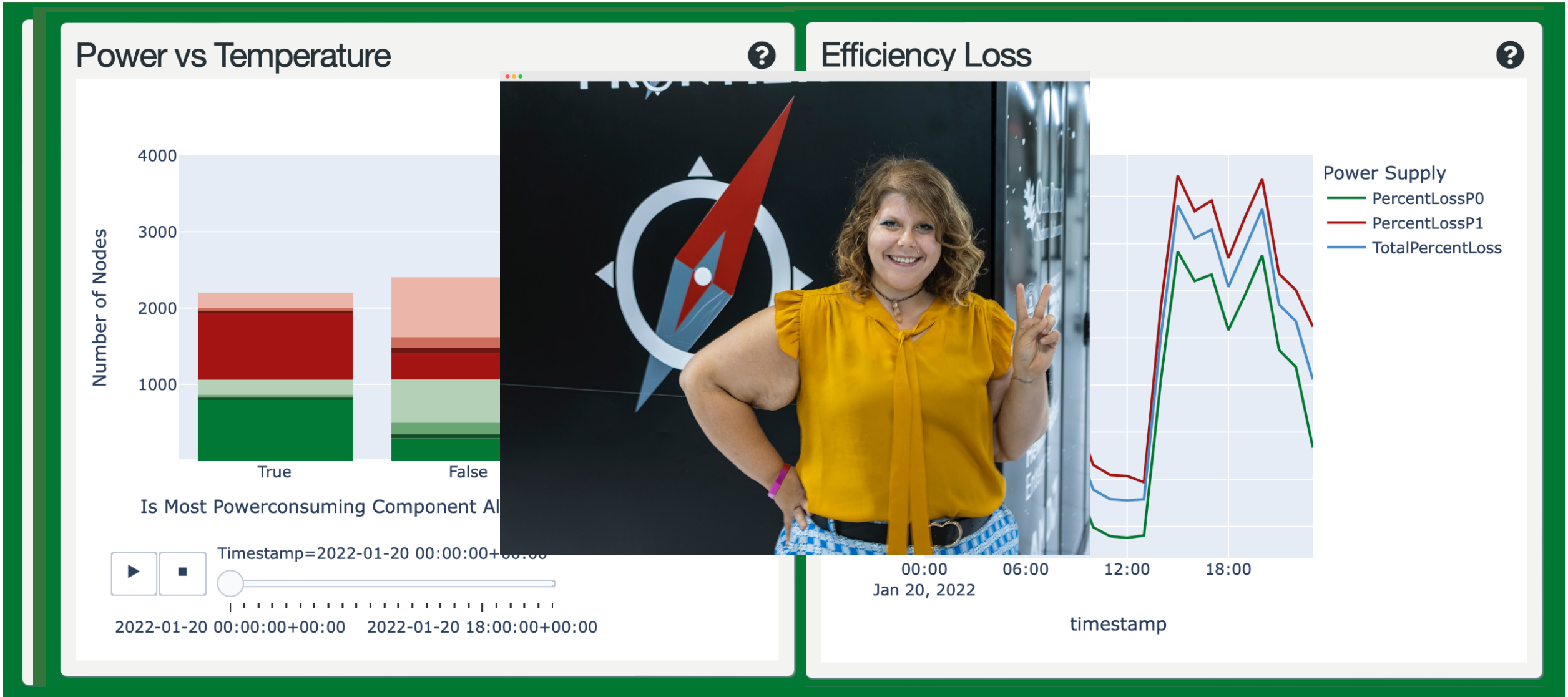
Contents:

- These slides
- Jupyter Notebook examples
- Ready-to-load Plotly Dash app


BRACE YOURSELVES
BAD DASHBOARD MEMES ARE COMING
imgflip.com

Dynamically Colored Summit Map

Kristen Hallas, PhD Student at University of Texas Rio Grande Valley



kristenhallas.com

Visualizing Supercomputer Data @ Oak Ridge National Lab

About this project

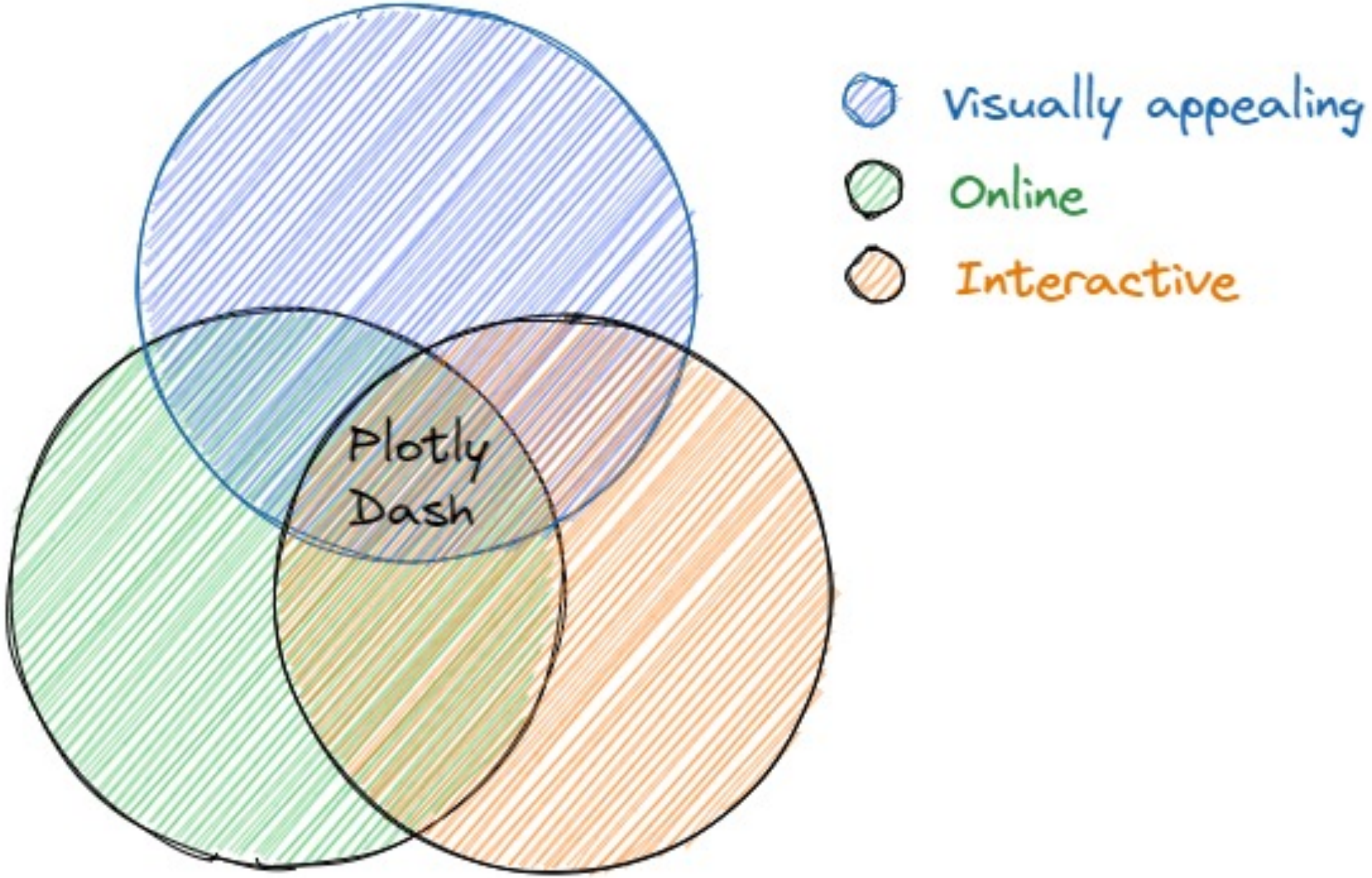
In Summer 2022, through the U.S. Department of Energy's Omni Technology Alliance Internship Program, I was appointed to work on real-world challenges related to cybersecurity and information technology at Oak Ridge National Lab. To this end, I employed Python to develop a novel dynamically colored supercomputer map using a full-stack, interactive visualization framework.

I presented my project at the 5th Annual Smoky Mountains Computational Sciences Data Challenge (SMCDC22) and was awarded as a Runner Up for Best Lightning Talk. At SIAM's Conference on Mathematics of Data Science (MDS22), I presented a poster, which is also linked

python™

pandas

plotly | Dash


Visually appealing
Online
Interactive
Plotly
Dash


YOU GET A DASHBOARD, *YOU*
GET A DASHBOARD
YOU GET A DASHBOARD!
Imgflip.com

Frontera Hacks A Dashboard

KRISTEN HALLAS

Application overview

Welcome to all those participating in Frontera Hacks, or any other visitors who may have stumbled here! If you are seeing this on some IP address, then congratulations for successfully launching this Plotly Dash app!

This app is ideal for someone who has a data set they want to explore and share with the world. I love Plotly Dash because it is interactive, it looks nice, you can customize so much, and the best part is if you need to do anything special, or even machine learning, the fact that this is all written in Python makes it possible. Thus, I designed this app with two goals in mind: 1) minimize the time it takes to launch something with a bunch of features that looks nice and 2) maximize the time you can spend processing and showcasing data that is interesting to you.

Some notable features include

- this layout, which is compatible with Markdown
- baked-in help modals (click the ? icon!)
- callback function buttons that swap out datasets (but can do so much more - definitely worth the research)
- samples of some of the coolest Plotly plots available to you

Click the buttons to load sample data sets, explore the interactivity of the Plotly plots, and enjoy customizing this dashboard!

MUSIC AND MENTAL HEALTH | WORLD HAPPINESS DATA 2015 | WORLD HAPPINESS DATA 2019 | WORLD HAPPINESS DATA 15-19

Histogram Example

Density Map Example

GitHub Repo

https://github.com/kristen-hallas/UTRGV-FronteraHacks23-Dash

Contents:

- These slides
- Jupyter Notebook examples
- Ready-to-load Plotly Dash app

To get started, you will need:

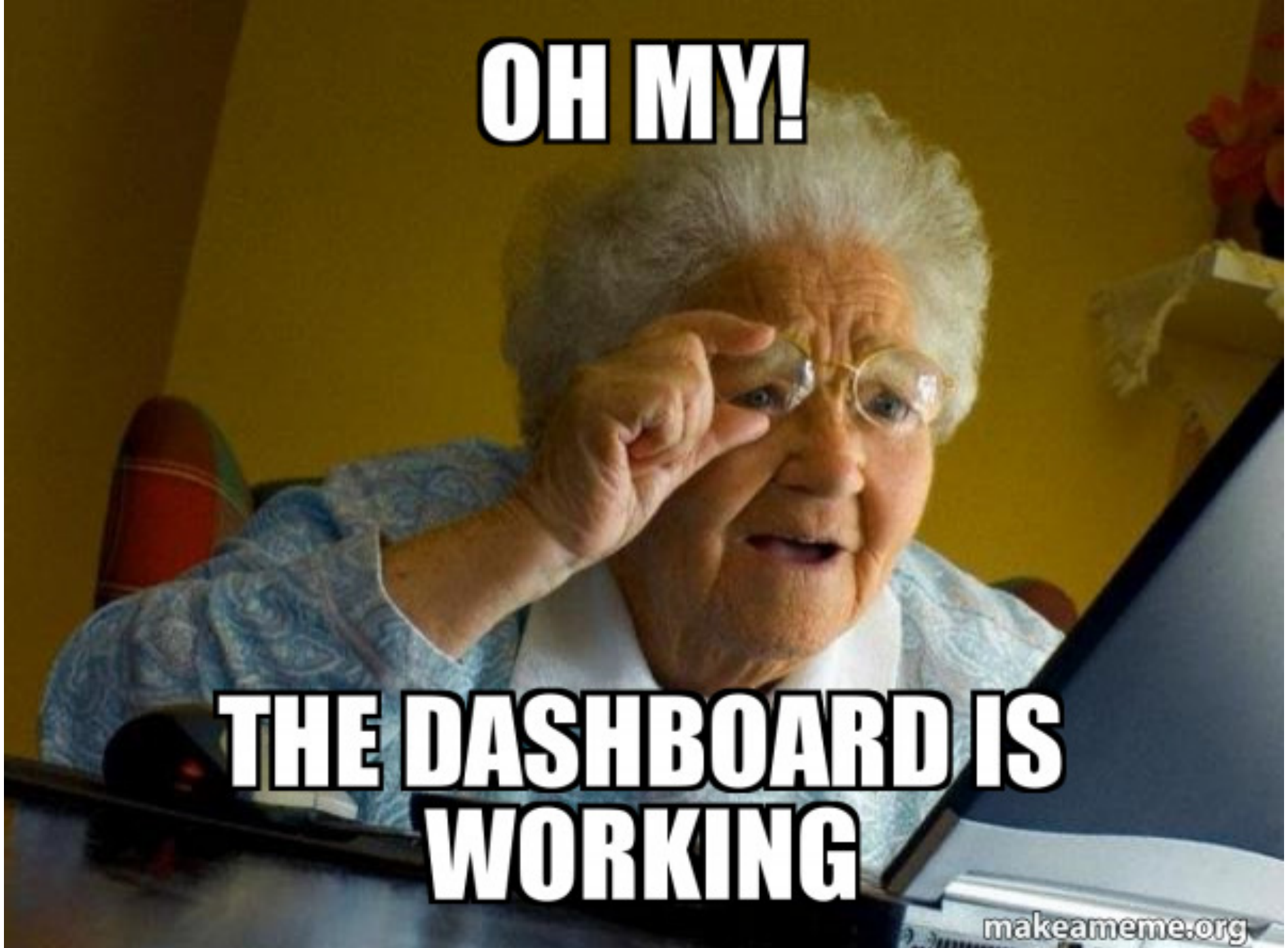
- Python
- Anaconda
- Your favorite Python IDE (PyCharm is great!)
- Jupyter Notebook (recommended)
- A dataset to visualize (Kaggle is a great resource!)

Python libraries required for this code

- Numpy
- Pandas
- Plotly
- Dash

Some helpful Terminal commands include:

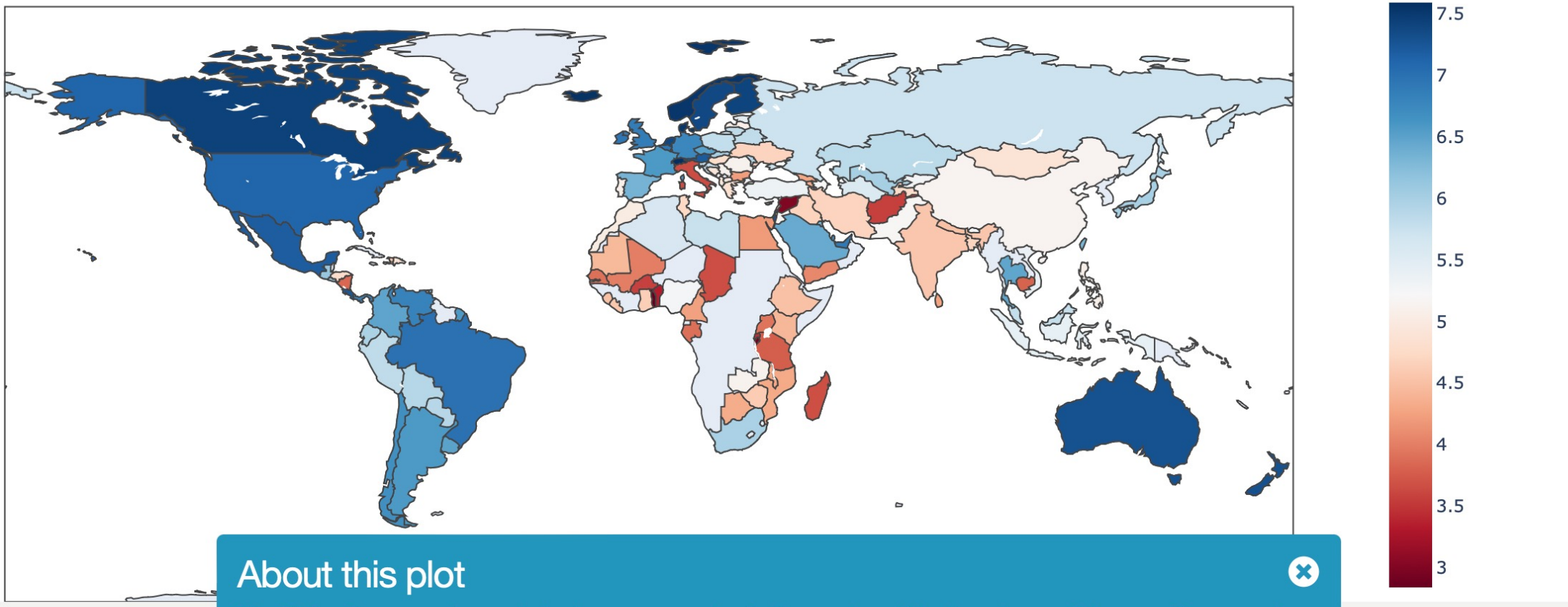
- Make directory = `mkdir`
- Change directory = `cd`
- Go up a directory = `cd ..`
- Run a Python app = `python3 app.py`
- Run Jupyter Notebook = `jupyter notebook`
- Quit a running Python app or Jupyter notebook = `ctrl + c`
- Install a library = `pip install dash`


OH MY!
THE DASHBOARD IS WORKING
makeameme.org

Modify main background

```
106
107  /* Base Styles
108  ------------------------------------------------- */
109  /* NOTE
110  html is set to 62.5% so that all the REM measurements throughout Skeleton
111  are based on 10px sizing. So basically 1.5rem = 15px :) */
112  html {
113    font-size: 62.5%; }
114  body {
115    font-size: 1.5em; /* currently ems cause chrome bug misinterpreting rems on body element */
116    line-height: 1.6;
117    font-weight: 400;
118    font-family: "Open Sans", "HelveticaNeue", "Helvetica Neue", Helvetica, Arial, sans-serif;
119    color: #263238;
120    background-color: #2596be;  /* BACKGROUND COLOR GOES HERE!*/
121    margin: 5%;
122  }
123
```

Main Plot Example



About this plot

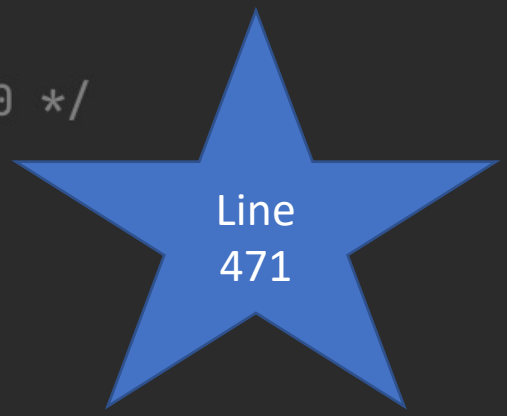
The ***Main plot example*** panel displays an example of a larger plot within the scheme of the app. You can edit this panel to contain information that is relevant to the plot you created.

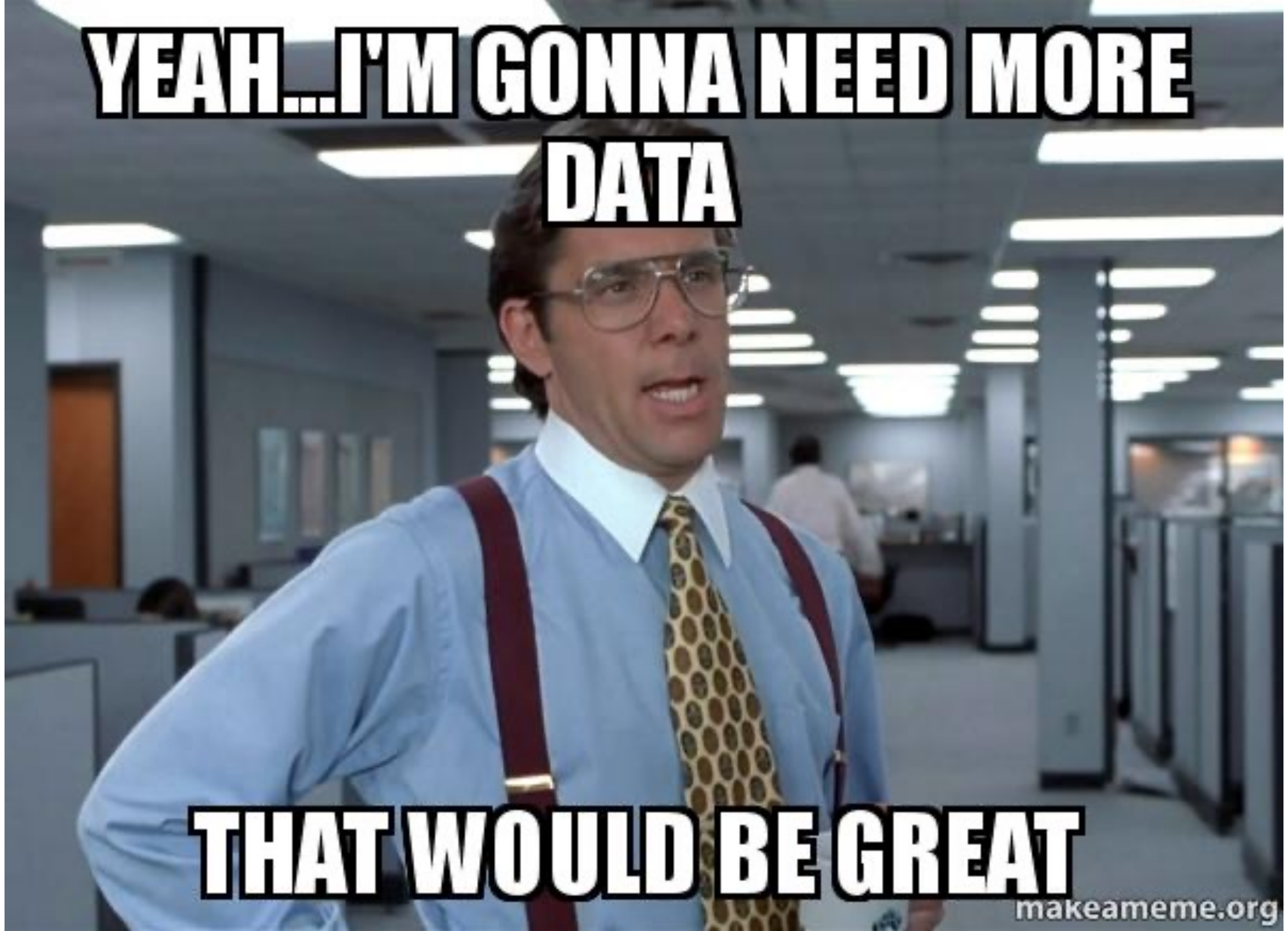
Plotly has all sorts of examples for plots, you can see all of them here..

Pie Example

Modify modal background

```
465
466     .modal-content {
467         z-index: 1004; /* Sit on top, including modebar which has z=1001 */
468         position: fixed;
469         left: 0;
470         width: 60%;
471         background-color: #2596be;  /* Material indigo 600 */
472         color: white;
473         border-radius: 5px;
474         margin-left: 20%;
475         margin-bottom: 2%;
476         margin-top: 2%;
477     }
478
```




YEAH...I'M GONNA NEED MORE DATA
THAT WOULD BE GREAT
makeameme.org

Insert your datasets here

```
73
74  # import all data for figures below
75  # this is from the starting dataset
76  musicdf = pd.read_csv("assets/mxmh_survey_results.csv")
77  # this is from the datasets that'll be added later
78  whddf = pd.read_csv("assets/WHD.csv")
79  whd15df = whddf.set_index('Country').query("Year==2015")
80  whd15df = whd15df.drop(columns=['Year'])
81  whd15df["Happiness Ratio"] = 1/whd15df["Happiness Rank"]
82  whd19df = whddf.set_index('Country').query("Year==2019")
83  whd19df = whd19df.drop(columns=['Year'])
84  whd19df["Happiness Ratio"] = 1/whd19df["Happiness Rank"]
```

Insert your data processing

```
# add all constants and code associated with data churning below
musicdf = musicdf.drop(columns=['Timestamp', 'Permissions'])
# you can decide how to handle missing data per column
# for example, with age you might want to impute a value based on the mean
musicdf["Age"] = musicdf["Age"].fillna(value=round(musicdf.Age.mean()))
# for something like music effects, you might want to assume no effect since it was not reported
musicdf["Music effects"] = musicdf["Music effects"].fillna(value="No effect")
musicdf["While working"] = musicdf["While working"].fillna(value="No")
musicdf["Instrumentalist"] = musicdf["Instrumentalist"].fillna(value="No")
musicdf["Composer"] = musicdf["Composer"].fillna(value="No")
musicdf["Primary streaming service"] = musicdf["Primary streaming service"].fillna(value="I do not use a streaming service.")
# feel free to impute missing values however you wish for your data
# you can also make new values
musicdf["Mental health severity"] = musicdf["Anxiety"] + musicdf["Depression"] + musicdf["Insomnia"] + musicdf["OCD"]
# data churning is done!!!
```

Insert your figures

```
# by default, we will initialize the music / mental health data set
# if you want to change the figures that load on start-up, edit these figures below
# Please be *very* careful with changing the names of these figures, as they are referenced multiple times in the code
# if you change any of the IDs above, you need to change where they are referenced below
# change figure names at your own risk
mainFig = go.Figure()
mainFig.add_trace(go.Violin(x=musicdf['Music effects'][musicdf['Music effects'] != 'No effect'],
                            y=musicdf['Depression'][musicdf['Music effects'] != 'No effect'],
                            legendgroup='Depression/10', scalegroup='Depression/10', name='Depression/10',
                            line_color='hotpink', box_visible=True)
                      )
```

Customize your app

```
169     html.Div(
170         children=[ # this contains the intro text and the toggle buttons
171             html.H4("Application overview", style={"margin-top": "0"}),
172             dcc.Markdown(
173                 """
174                 Welcome to all those participating in Frontera Hacks, or any other visitors who may have stumbled here!
175                 If you are seeing this on some IP address, then congratulations for successfully launching
176                 this Plotly Dash app!
177
178                 This app is ideal for someone who has a data set they want to explore and share with the world. I love
179                 Plotly Dash because it is interactive, it looks nice, you can customize so much, and the best part is
180                 if you need to do anything special, or even machine learning, the fact that this is all written
181                 in Python makes it possible. Thus, I designed this app with two goals in mind: 1) minimize the time
182                 it takes to launch something with a bunch of features that looks nice and 2) maximize the time
183                 you can spend processing and showcasing data that is interesting to you.
184
185                 Some notable features include
186                 * this layout, which is compatible with Markdown
187                 * baked-in help modals (click the ? icon!)
188                 * callback function buttons that swap out datasets (but can do so much more -
189                 [definitely worth the research](https://dash.plotly.com/basic-callbacks))
190                 * samples of some of the coolest Plotly plots available to you
191
192                 Click the buttons to load sample data sets, explore the interactivity of the Plotly plots, and enjoy
193                 customizing this dashboard!
194                 """
195                 ),
196                 html.Div(
```

```
196                 html.Div(
197                     children=[
198                         html.Button(
199                             "Music And Mental Health",
200                             id="mxmh",
201                             className="button",
202                             style={"padding-left": "10px", "padding-right": "10px",
203                                    "margin-left": "10px", "margin-right": "10px"}
204                         ),
```

```
484 # create callback to show each graph. you can't do multiple callbacks that use the same output/input ID
485 # so that's why we do it in a big function like this
486 # if you need help understanding each of the graph plots I comment about them up at their initialization
487 @app.callback([
488     Output("histo-graph", "figure"),
489     Output("dense-graph", "figure"),
490     Output("main-graph", "figure"),
491     Output("pie-graph", "figure"),
492     Output("scatter-graph", "figure")],
493     [Input("mxmh", "n_clicks"),
494      Input("whd15", "n_clicks"),
495      Input("whd19", "n_clicks"),
496      Input("whd", "n_clicks")], prevent_initial_call=True)
```

```
498 # you need the number of input in update_graphs to match the number of buttons you have updating graphs
499 def update_graphs(b1, b2, b3, b4):
500     triggered_id = ctx.triggered[0]['prop_id']
501     if 'mxmh.n_clicks' == triggered_id:
502         return update_mxmh()
503     elif 'whd15.n_clicks' == triggered_id:
504         return update_whd15()
505     elif 'whd19.n_clicks' == triggered_id:
506         return update_whd19()
507     else:
508         return update_whd()
509
```

Callback buttons

```
509
510 # the output should be returning the figures you wanted to update
511 def update_mxmh():
512     mainFig = go.Figure()
513     mainFig.add_trace(go.Violin(x=musicdf['Music effects'][musicdf['Music effects'] != 'No effect'],
514                                 y=musicdf['Depression'][musicdf['Music effects'] != 'No effect'],
515                                 legendgroup='Depression/10', scalegroup='Depression/10', name='Depression/10',
516                                 line_color='hotpink', box_visible=True)
517                       )
518     mainFig.add_trace(go.Violin(x=musicdf['Music effects'][musicdf['Music effects'] != 'No effect'],
519                                 y=musicdf['Anxiety'][musicdf['Music effects'] != 'No effect'],
520                                 legendgroup='Anxiety/10', scalegroup='Anxiety/10', name='Anxiety/10',
521                                 line_color='green', box_visible=True)
522                       )
523     mainFig.add_trace(go.Violin(x=musicdf['Music effects'][musicdf['Music effects'] != 'No effect'],
524                                 y=musicdf['OCD'][musicdf['Music effects'] != 'No effect'],
525                                 legendgroup='OCD/10', scalegroup='OCD/10', name='OCD/10',
526                                 line_color='blue', box_visible=True)
527                       )
528     mainFig.add_trace(go.Violin(x=musicdf['Music effects'][musicdf['Music effects'] != 'No effect'],
529                                 y=musicdf['Insomnia'][musicdf['Music effects'] != 'No effect'],
530                                 legendgroup='Insomnia/10', scalegroup='Insomnia/10', name='Insomnia/10',
531                                 line_color='purple', box_visible=True)

551     return histoFig, denseFig, mainFig, pieFig, scatterFig
```

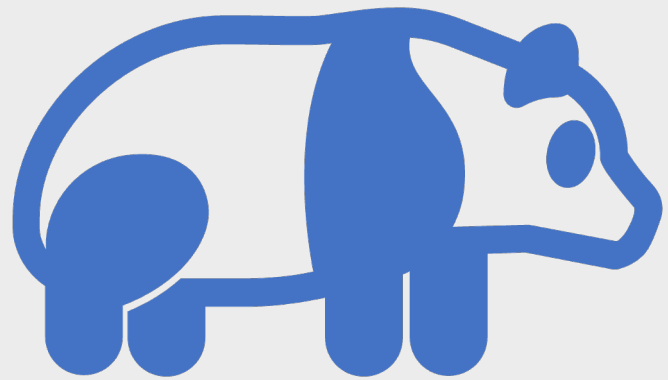
Be Careful of Renaming

```
# first function create show/hide callbacks for each info modal
# if you change any of the IDs above, you need to change them her
# change at your own risk
for id in ["histo", "dense", "main", "pie", "scatter"]:

    @app.callback(
        [Output(f"{id}-modal", "style"), Output(f"{id}-div", "sty
        [Input(f"show-{id}-modal", "n_clicks"), Input(f"close-{id
    )
    def toggle_modal(n_show, n_close):
        ctx = dash.callback_context
        if ctx.triggered and ctx.triggered[0]["prop_id"].startswi
            return {"display": "block"}, {"zIndex": 1003}
        else:
            return {"display": "none"}, {"zIndex": 0}
```

```
                build_modal_info_overlay(
                    "histo",
                    "bottom",
                    dedent(
                        """
            The _**Histogram example**_ panel displ
            contain information that is relevant to

            [You can learn more about Histograms at
            """
```

```
                        [
                            "Histogram Example",~# top left chart
                            html.Img(
                                id="show-histo-modal",
                                src="assets/question-circle-solid.svg
                                n_clicks=0,
                                className="info-icon",
                            ),
                        ],
                        className="container_title",
                    ),
                    dcc.Loading(
                        dcc.Graph(
                            id="histo-graph",
                            figure=histoFig,
                            config={"displayModeBar": False},
                        ),
                        className="svg-container",
                        style={"height": 150},
                    ),
                ],
                className="six columns pretty_container",
                id="histo-div",
```

Suggestion: as you build your Dash app, build the Plotly plots you want to use in Jupyter Notebooks, and then add them to the app.py after



exploring-v1.ipynb is an introduction to pandas dataframes and plotly plots

exploring-v2.ipynb has more examples, including examples with timeseries data

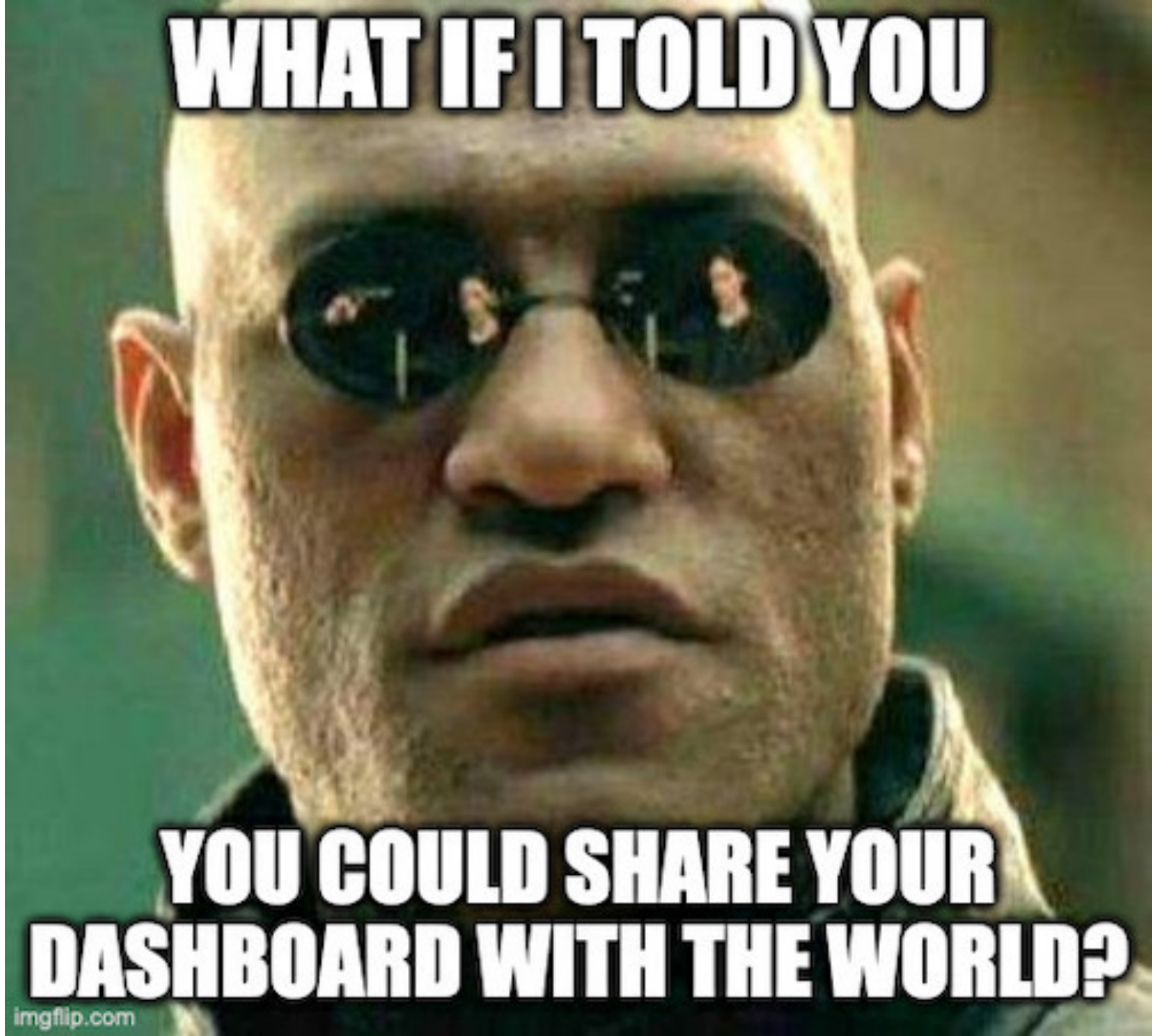
Time to explore

Let yourself become an expert!

The best way I learned through Plotly was to imagine my perfect chart, and make it happen. There's so much Plotly can do that I haven't covered yet, like regional maps and animation for time series data.

- Find data you like
- Explore all the reference libraries
- Don't be afraid to hack your own solution
- Manipulate the data as needed

You got this!


WHAT IF I TOLD YOU
YOU COULD SHARE YOUR DASHBOARD WITH THE WORLD?
imgflip.com

Your domain is available!

yourmoneydash.com

~~$19.99~~ $0.01 for the first year with a 2 year registration

Make it yours

Settings

Any changes you make to custom domains will apply immediately

- Navigation
- Brand images
- Viewer tools
- Custom domains
- Analytics
- Announcement banner

Connected domains

www.kristenhallas.com

Add

SPIE. SPIE @ UTRGV

Home Seminars

Shameless Plug – SPIE @ UTRGV

- Portfolio Site Workshop
- Member Giveaways
- $25 gift cards
- Join free
  - Use QR Code

Join SPIE @ UTRGV

The University of Texas Rio Grande Valley (UTRGV) is one of the most rapidly growing HSI (Hispanic serving institutions) in the nation. The UTRGV Chapter of SPIE (Society of Photographic Instrumentation Engineers) was officially established in May 2014 to connect UTRGV students with the world's foremost nonlinear optics with interdisciplinary applications and photonics technical society. It has since grown to become the largest interdisciplinary research organization at the UTRGV campus,

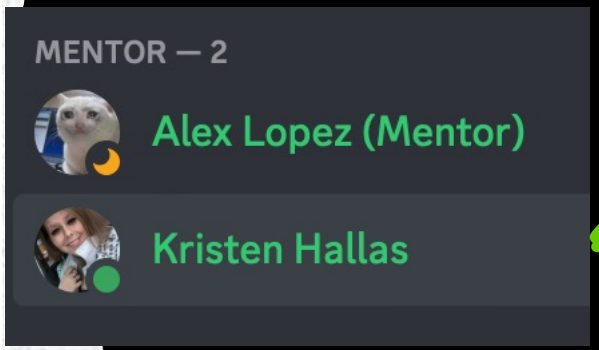
Happy Hacking ☺

Yes you can do hard things!!!
Time to become the expert and hack yourself something new for your online portfolio

kristen.hallas01@utrgv.edu

Yes, I am on the Discord!

ASK ME ANYTHING



Acknowledgements

This template was repurposed with love by Kristen Hallas, for the purpose of the inaugural Frontera Hacks Hackathon. Frontera Hacks A Dashboard would not be possible without the following sources.

- Dashboard written in Python using the Plotly Dash web framework.
- This dashboard was sourced heavily from World Cell Towers found here.
- For huge data, parallel and distributed calculations can be implemented using the Dask Python library.
- Icons provided by Font Awesome and used under the *Font Awesome Free License*.
- Primary dataset is from Catherine Rasgaitis on Kaggle, Music & Mental Health Survey Results.
- Secondary dataset was collated/cleaned by Kristen Hallas, sourced from the World Happiness Report on Kaggle.
- Last and certainly not least, Oak Ridge National Lab (ORNL) deserves a big thank you - in Summer 2022, through the U.S. Department of Energy's Omni Technology Alliance Internship Program, I was appointed to work on real-world challenges related to cybersecurity and information technology. To this end, I developed a novel, dynamically colored map using a similar interactive visualization framework.