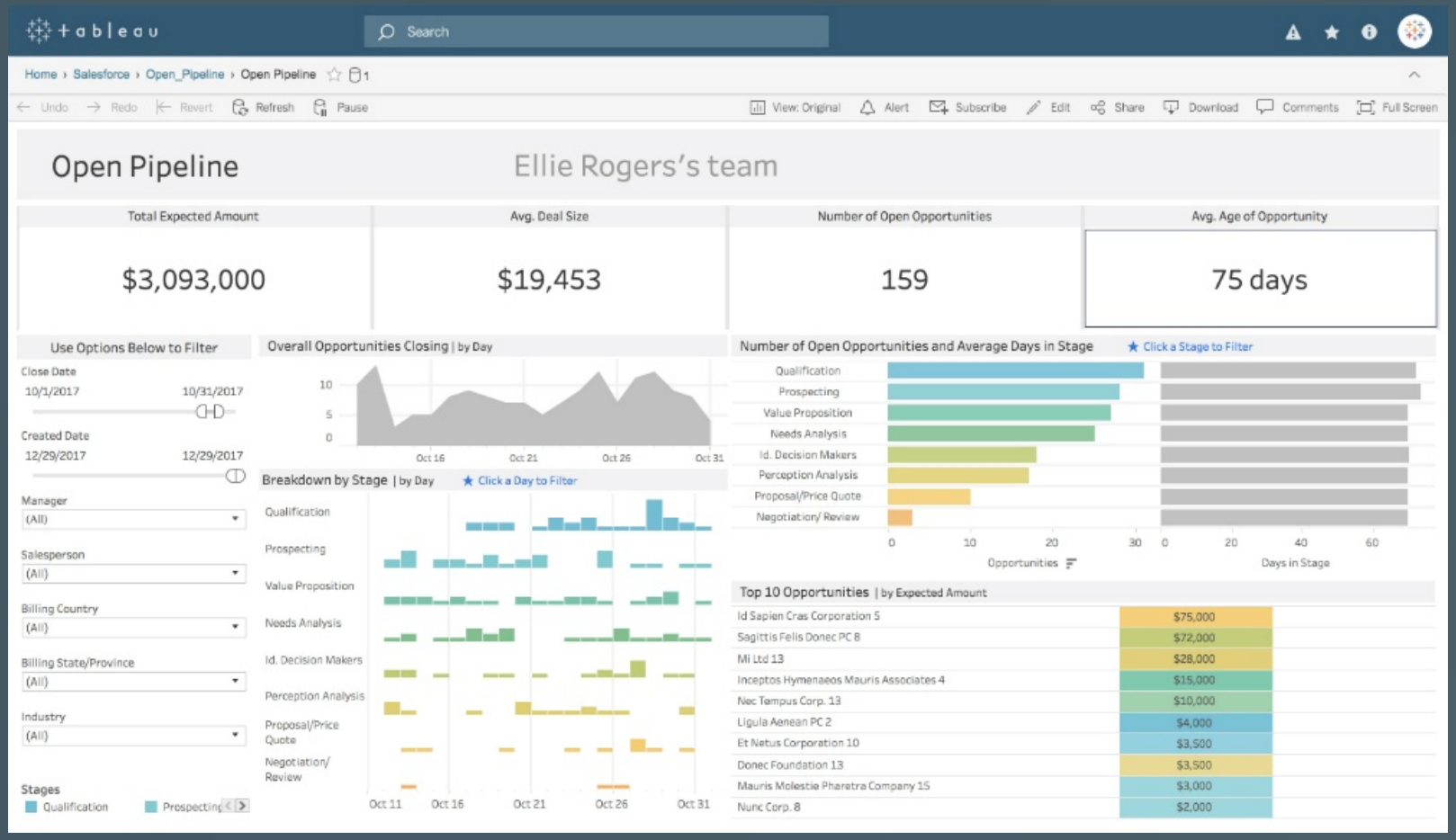
# Dashboards with Plotly/Dash

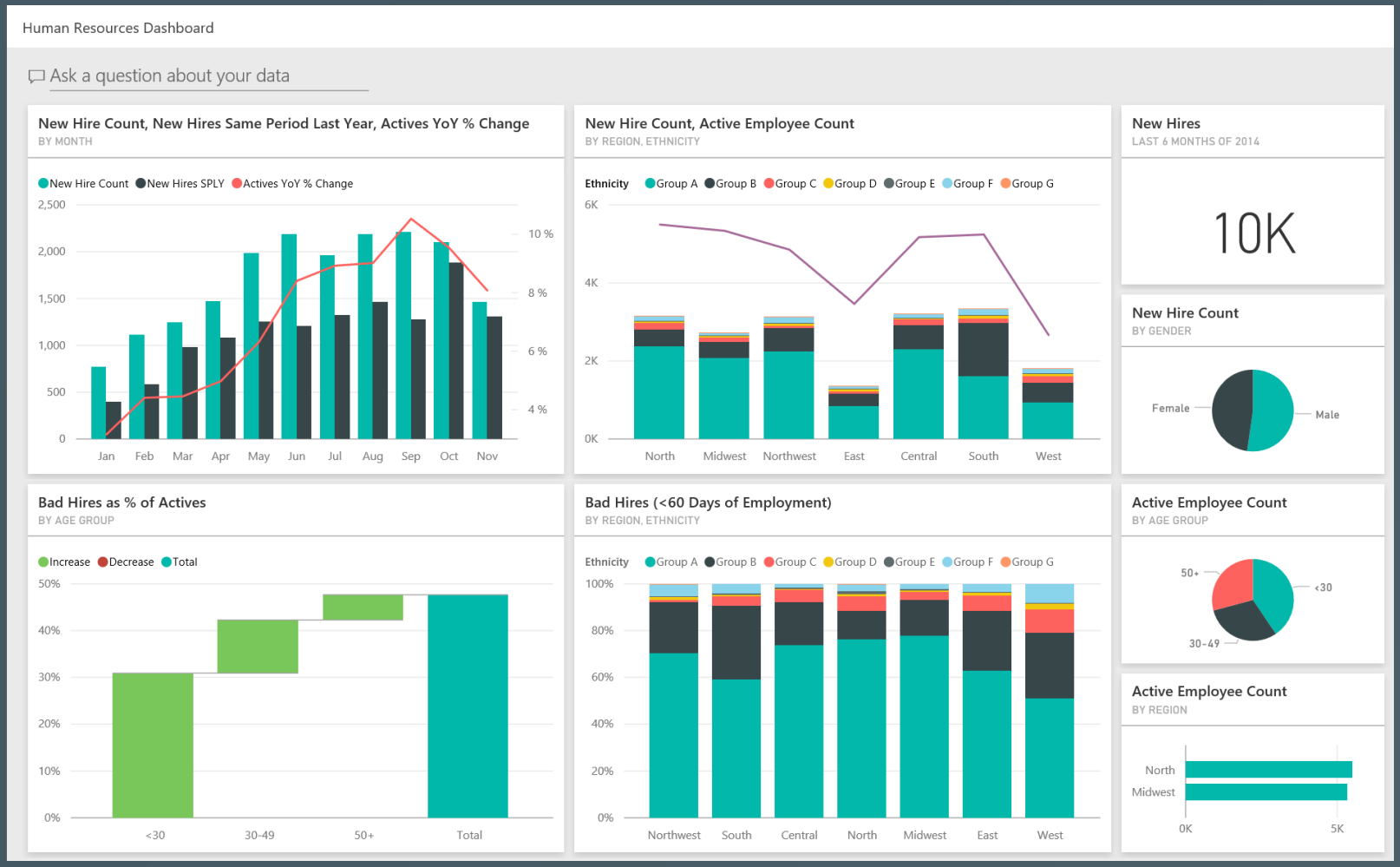
# What Are Dashboards?

- Information repositories
- Make information available to non-specialists
- Provide easy access to diverse information
- Allow the end-user to explore data
- Alternative to static reporting

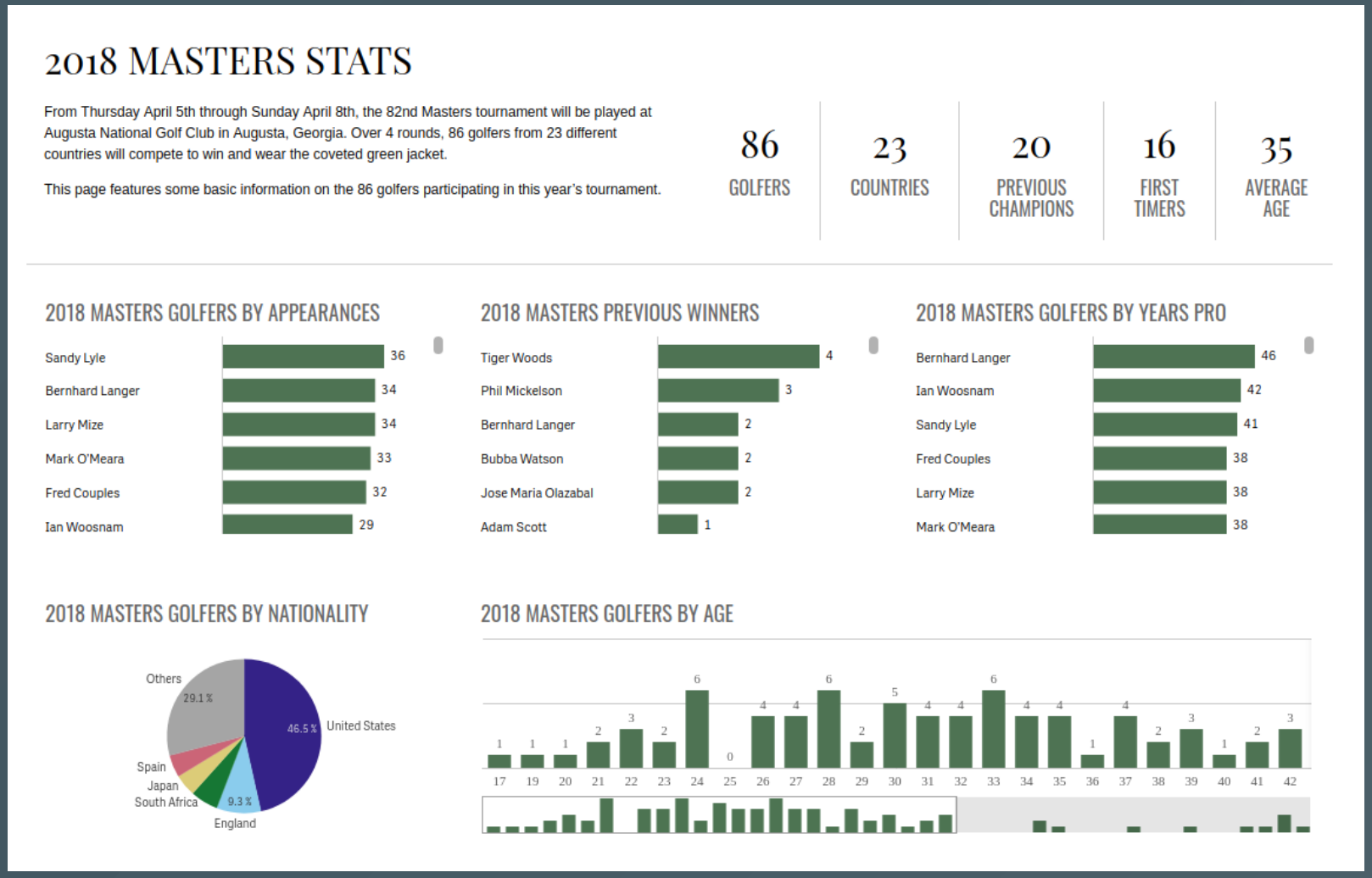
# Dashboard Options - Tableau



# Dashboard Options - PowerBI



# Dashboard Options - Qlik



# Dashboard Options - BYOD

We might also choose to design our own dashboard, based on needs that are unique to our business problem or data.

In Python, we can do this with Dash, made by the Plotly team. (In R, this can be done with Shiny)

This may not be the quickest way to make dashboards, but it is absolutely the most flexible.

# Dash - Introduction

Let's walk through some example code provided by Dash to get started.

First, we need our import statements

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import plotly.graph_objs as go
```

Because Dash will construct our website for us, we have a lot of imports to make.

# Dash - Introduction

```
app = dash.Dash()

df = pd.read_csv(
  'https://gist.githubusercontent.com/chriddyp/' +
  '5d1ea79569ed194d432e56108a04d188/raw/' +
  'a9f9e8076b837d541398e999dcbac2b2826a81f8/'+
  'gdp-life-exp-2007.csv')
```

Here, we are just initiating our dashboard application, and importing a csv reflecting life expectancy by country.

# Dash - Introduction

```
dashData = [
  go.Scatter(
    x=df[df['continent'] == i]['gdp per capita'],
    y=df[df['continent'] == i]['life expectancy'],
    text=df[df['continent'] == i]['country'],
    mode='markers',
    opacity=0.7,
    marker={
      'size': 15,
      'line': {'width': 0.5, 'color': 'white'}
    },
    name=i
    ) for i in df.continent.unique()
  ]
```

Our Data and Layout look just like Plotly here!

# Dash - Introduction

```
dashLayout = go.Layout(
  xaxis={'type': 'log', 'title': 'GDP Per Capita'},
  yaxis={'title': 'Life Expectancy'},
  margin={'l': 40, 'b': 40, 't': 10, 'r': 10},
  legend={'x': 0, 'y': 1},
  hovermode='closest'
  )
```

Our Data and Layout look just like Plotly here!

This is because Plotly is the underlying graphing library that we are using to generate the graphs.

# Dash - Introduction

```
app.layout = html.Div([
  dcc.Graph(
    id='life-exp-vs-gdp',
    figure={
       'data': dashData,
       'layout': dashLayout
      }
  )
])

if __name__ == '__main__':
  app.run_server()
```

We need to organize our plot inside of the HTML structure. We can start with an `html.Div` object.

# Using Object ID's

In the code on the last slide, we assign the `dcc.Graph` object an `id` attribute:

```
dcc.Graph(
    id='life-exp-vs-gdp',
    ...
```

This `id` allows us to manipulate the contents of this object as we design our visual for interactivity. We will be able to refer back to this `dcc.Graph` object using its `id` whenever we want to modify it.

# Quick Note: `html.Div`

What is a `Div`? This (newish) HTML tag is used to organize websites, and is particularly valuable when using CSS (cascading style-sheets) to format the website.

In Dash, they will be used as containers for different kinds of objects on our dashboard. We can even replace their contents based on input from the user if we so choose.

# Dash - Adding Controls

In order to add controls to our Dashboard (so that it isn't just a single Plotly visual), we need to include another import statement:

```
from dash.dependencies import Input, Output
```

We can now describe to Dash the way in which we want information to interact with our visual.

# Dash - Adding Controls

After our import statements, we can start a new Dashboard:

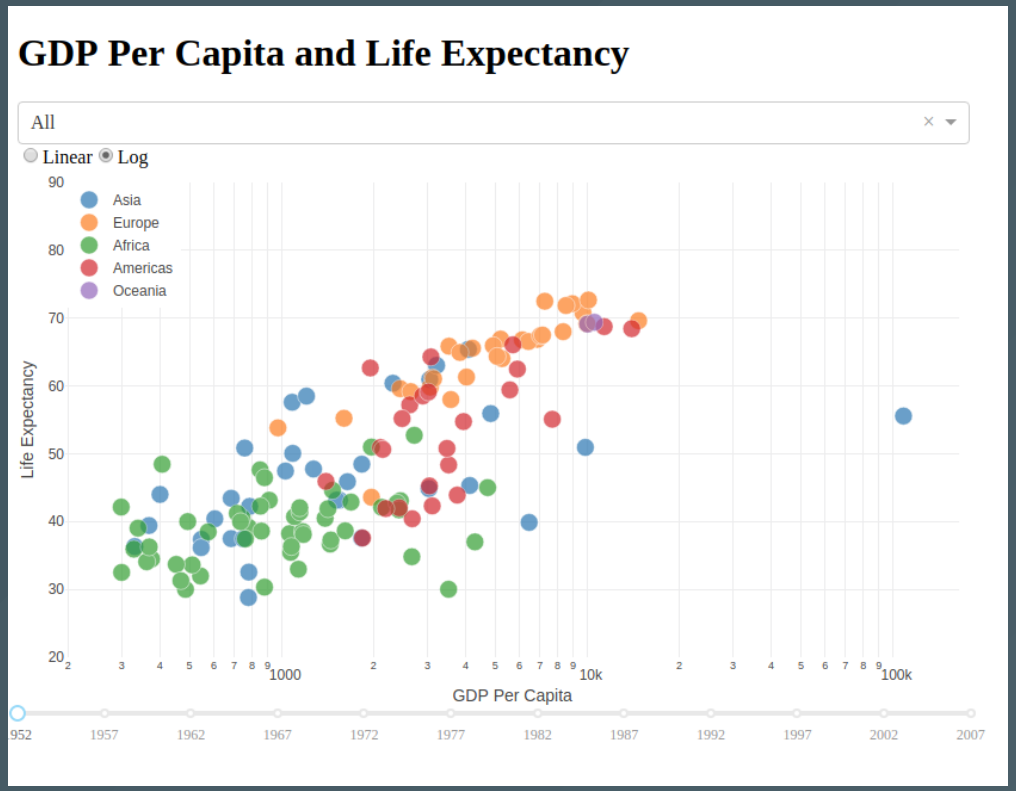
```
df = pd.read_csv(
    'https://raw.githubusercontent.com/plotly/'
    'datasets/master/gapminderDataFiveYear.csv')

app = dash.Dash()
```

This dataset will offer us more information on life expectancy, so that we can increase the amount of control that we have.

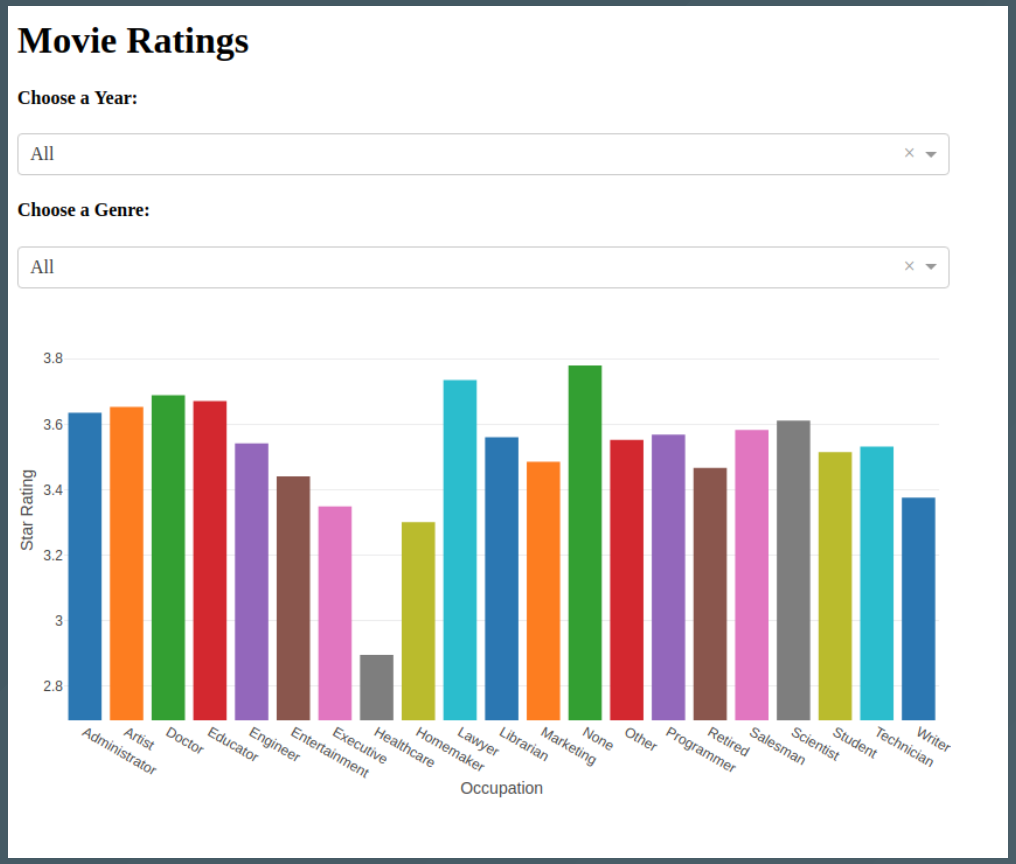
# Dash - Adding Controls

From here, we need to look at the script as a whole, since our code will all work together to generate our interactive plot.



# Dash - Another Example

Let's take a look at a different example using movie rating information:



# Dash - Adding Controls

- We can utilize as many inputs as we would like
  - BUT! We can only use one output per callback
- Our options for inputs include
  - Dropdown Menus
  - Sliders
  - Date Selection
  - Interactive Tables
  - Checkboxes and Radio Buttons

# Dash - Interactivity

We can also implement interactivity between plot elements using the following features

1. React to user hovering over a point
2. React to user selecting points
3. React to user zooming on points
4. React to user clicking on points

# [Optional] Lab Today

Pick your favorite dataset from the semester, and create a dashboard that allows an imaginary user (or just yourself!) to interactively explore the data in at least two dimensions. Try including the following:

1. One plot that responds to inputs
2. A slider
3. A dropdown menu
4. Radio Buttons