Introduction to Plot.ly and Dash

Jason Leigh

Laboratory for Advanced Visualization & Applications
University of Hawai‘i at Mānoa

Dash

- Dash is a python-based API that lets you build data visualization web apps without having to code in JavaScript, HTML and CSS. Underlying charts are created with Plot.ly.
- Dash apps are typically run as python programs (which initiate a web server).
- You connect to the app by using your web browser.
- Alternatively you use Dash's cloud environment.
- In addition to supporting Python it also supports Jupyter notebooks.

Installation

- Install Python3 first.
- Make sure to set your PATH correctly to point to Python3 bin directory.
- Install pip (helps with installing Python libraries)
- https://pip.pypa.io/en/stable/installation/#get-pip-py
- Then install Dash and Pandas
- https://dash.plotly.com/installation
- Pandas is for data wrangling. You don’t absolutely need it but the Dash tutorials use it, and it is worth learning.
- https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html

Dash

- 2 components:
  - Layout and Interaction
- These are housed in 2 libraries:
- dash_html_components – has every HTML tag in it (e.g. html.H1(“Hello”)
- https://dash.plotly.com/dash-html-components

- dash_core_components – higher level components that are interactive (generated with JavaScript, HTML, CSS through React.js)
- One major core component is Graph which uses Plot.ly to make charts.
- https://dash.plotly.com/dash-core-components

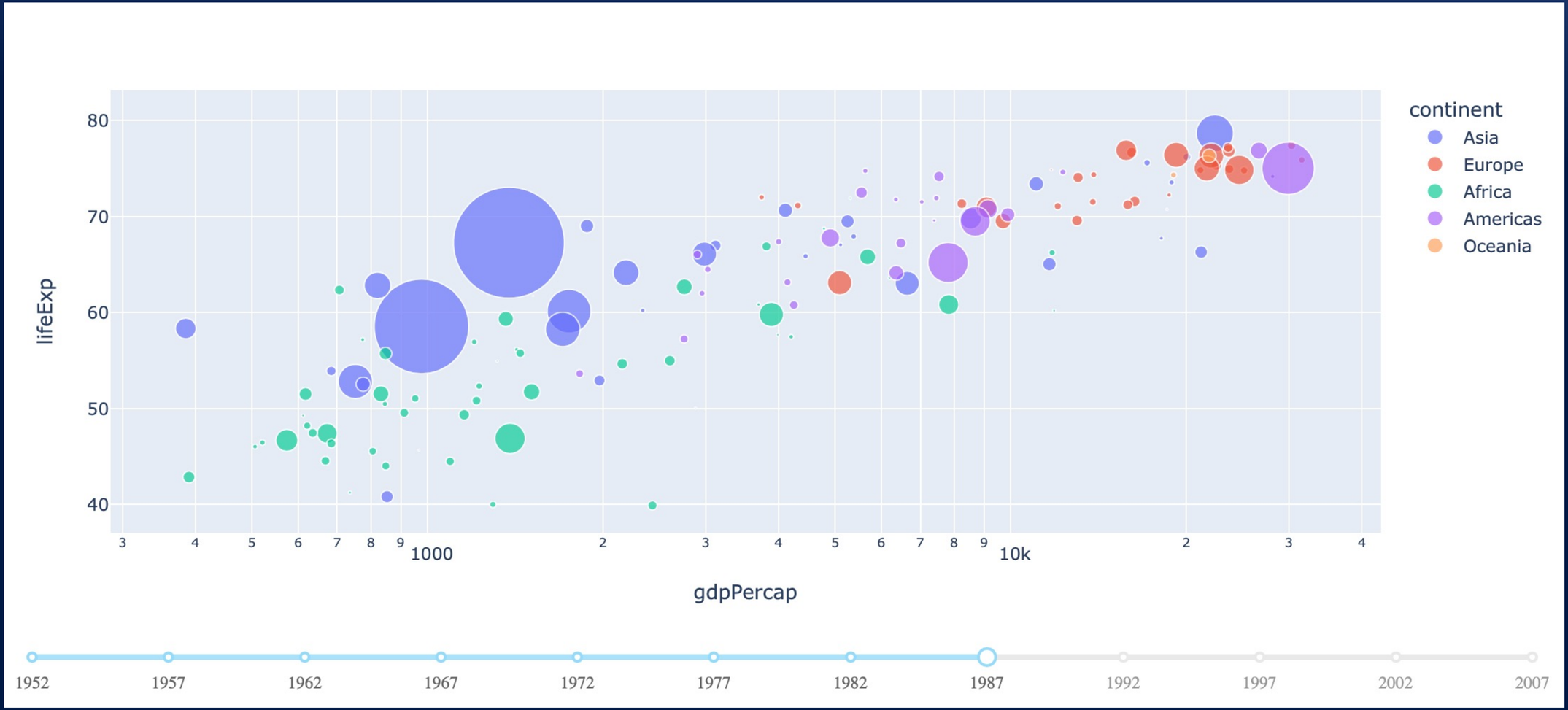
- Interaction between GUI elements is done by using Callbacks.

Now Go Learn Dash

- Installation
- https://dash.plotly.com/installation
- Layout
- https://dash.plotly.com/layout
- Callbacks
- https://dash.plotly.com/basic-callbacks

Simple Interactive Example

Second example from:
https://dash.plotly.com/basic-callbacks



```python
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd

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

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        id='year-slider',
        min=df['year'].min(),
        max=df['year'].max(),
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        step=None
    )
])
```

Imports libraries (Dash, html & core components, Plotly Express, Pandas)

Use Pandas to read in a CSV (Comma Separated File)

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

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        id='year-slider',
        min=df['year'].min(),
        max=df['year'].max(),
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        step=None
    )
])
```

Store data in df (in Pandas lingo this is called a data frame hence df)

'https://raw.githubusercontent.com/plotly/datasets/master/gapminderDataFiveYear.csv'

Data looks like this.
Notice the 1st line of
attribute names.

country, year, pop, continent, lifeExp, gdpPercap

```
country,year,pop,continent,lifeExp,gdpPercap
Afghanistan,1952,8425333,Asia,28.801,779.4453145
Afghanistan,1957,9240934,Asia,30.332,820.8530296
Afghanistan,1962,10267083,Asia,31.997,853.10071
Afghanistan,1967,11537966,Asia,34.02,836.1971382
Afghanistan,1972,13079460,Asia,36.088,739.9811058
Afghanistan,1977,14880372,Asia,38.438,786.11336
Afghanistan,1982,12881816,Asia,39.854,978.0114388
Afghanistan,1987,13867957,Asia,40.822,852.3959448
Afghanistan,1992,16317921,Asia,41.674,649.3413952
Afghanistan,1997,22227415,Asia,41.763,635.341351
Afghanistan,2002,25268405,Asia,42.129,726.7340548
Afghanistan,2007,31889923,Asia,43.828,974.5803384
Albania,1952,1282697,Europe,55.23,1601.056136
Albania,1957,1476505,Europe,59.28,1942.284244
Albania,1962,1728137,Europe,64.82,2312.888958
Albania,1967,1984060,Europe,66.22,2760.196931
Albania,1972,2263554,Europe,67.69,3313.422188
Albania,1977,2509048,Europe,68.93,3533.00391
Albania,1982,2780097,Europe,70.42,3630.880722
Albania,1987,3075321,Europe,72,3738.932735
Albania,1992,3326498,Europe,71.581,2497.437901
Albania,1997,3428038,Europe,72.95,3193.054604
Albania,2002,3508512,Europe,75.651,4604.211737
Albania,2007,3600523,Europe,76.423,5937.029526
Algeria,1952,9279525,Africa,43.077,2449.008185
Algeria,1957,10270856,Africa,45.685,3013.976023
Algeria,1962,11000948,Africa,48.303,2550.81688
Algeria,1967,12760499,Africa,51.407,3246.991771
Algeria,1972,14760787,Africa,54.518,4182.663766
Algeria,1977,17152804,Africa,58.014,4910.416756
Algeria,1982,20033753,Africa,61.368,5745.160213
Algeria,1987,23254956,Africa,65.799,5681.358539
Algeria,1992,26298373,Africa,67.744,5023.216647
Algeria,1997,29072015,Africa,69.152,4797.295051
Algeria,2002,31287142,Africa,70.994,5288.040382
Algeria,2007,33333216,Africa,72.301,6223.367465
Angola,1952,4232095,Africa,30.015,3520.610273
Angola,1957,4561361,Africa,31.999,3827.940465
Angola,1962,4826015,Africa,34,4269.276742
Angola,1967,5247469,Africa,35.985,5522.776375
Angola,1972,5894858,Africa,37.928,5473.288005
Angola,1977,6162675,Africa,39.483,3008.647355
Angola,1982,7016384,Africa,39.942,2756.953672
Angola,1987,7874230,Africa,39.906,2430.208311
Angola,1992,8735988,Africa,40.647,2627.845685
Angola,1997,9875024,Africa,40.963,2277.140884
Angola,2002,10866106,Africa,41.003,2773.287312
Angola,2007,12420476,Africa,42.731,4797.231267
Argentina,1952,17876956,Americas,62.485,5911.315053
Argentina,1957,19610538,Americas,64.399,6856.856212
Argentina,1962,21283783,Americas,65.142,7133.166023
Argentina,1967,22934225,Americas,65.634,8052.953021
Argentina,1972,24779799,Americas,67.065,9443.038526
Argentina,1977,26983828,Americas,68.481,10079.02674
Argentina,1982,29341374,Americas,69.942,8997.897412
Argentina,1987,31620918,Americas,70.774,9139.671389
Argentina,1992,33958947,Americas,71.868,9308.41871
Argentina,1997,36203463,Americas,73.275,10967.28195
Argentina,2002,38331121,Americas,74.34,8797.640716
```

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

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        id='year-slider',
        min=df['year'].min(),
        max=df['year'].max(),
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        step=None
    )
])
```

Start your Dash app

Layout contains a **Div** with an array of 2 things, a **Graph** and a **Slider**.

Set step to None so that using arrow keys jumps to next discrete Year value

{} is a Python dictionary. Label each entry in the dictionary using each unique year found in df. Use this to set the tick mark label in the slider.

Set min and max of slider to the data frame's min and max value based on the Year attribute

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

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        id='year-slider',
        min=df['year'].min(),
        max=df['year'].max(),
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        step=None
    )
])
```

Output to graph-with-slider is a Plot.ly figure

Callback declaration

```python
@app.callback(
    Output('graph-with-slider', 'figure'),
    Input('year-slider', 'value'))
def update_figure(selected_year):

    # Filter the data frame (df) on the Year where Year is selected_year
    filtered_df = df[df.year == selected_year]

    fig = px.scatter(filtered_df, x="gdpPercap", y="lifeExp",
                     size="pop", color="continent", hover_name="country",
                     log_x=True, size_max=55)

    fig.update_layout(transition_duration=500)

    return fig


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

Input is from year-slider slider

Dash calls this function (you can call it anything) whenever an input value changes in the year-slider

```python
@app.callback(
    Output('graph-with-slider', 'figure'),
    Input('year-slider', 'value'))
def update_figure(selected_year):

    # Filter the data frame (df) on the Year where Year is selected_year
    filtered_df = df[df.year == selected_year]

    fig = px.scatter(filtered_df, x="gdpPercap", y="lifeExp",
                     size="pop", color="continent", hover_name="country",
                     log_x=True, size_max=55)

    fig.update_layout(transition_duration=500)

    return fig


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

This filters the data frame on Year whenever Year is selected_year

Make a scatterplot using Plot.ly Express using the filtered year (filtered_df)

Plot gdpPercap on X axis.
Plot lifeExp on Y axis.
Make Size the Population
Make Color the Continent
If you hover the Country name shows up.
Set X axis to logarithmic.

Update the figure over a transition period of 500 milliseconds

This enables “Hot Reloading” which means whenever the code changes, your browser will reload the page automatically.

Need to return the figure because this is the output of the callback

Running Your App

- Generally:
- python3 demo.py
- Then open a browser and go to:
- http://127.0.0.1:8050/