**Python Visualization Dashboards with Plotly's Dash Library**

Contents

[**Resources:** 2](#_Toc508797165)

[**Curriculum:** 2](#_Toc508797166)

[**Udemy Course Management Page:** 2](#_Toc508797167)

[**Github Repository (private):** 2](#_Toc508797168)

[**Datasets:** 2](#_Toc508797169)

[**Choosing an IDE:** 2](#_Toc508797170)

[**JupyterLab** 2](#_Toc508797171)

[**Atom** 2](#_Toc508797172)

[**Online vs Offline:** 2](#_Toc508797173)

[**Lessons:** 3](#_Toc508797174)

[**Plotly Basics** 3](#_Toc508797175)

[**11.** **Plotly Basics Overview** Discuss the various plot types we will show how to do. 3](#_Toc508797176)

[**12.** **Scatter Plots** Show Scatter plot examples with basic plotly 3](#_Toc508797177)

[**13.** **Line Charts** 3](#_Toc508797178)

[**14.** **Bar Charts** Bar Charts with Plotly 3](#_Toc508797179)

[**15.** **Bubble Plots** Bubble Plots with Plotly 3](#_Toc508797180)

[**16.** **Box Plots** Box Plots with Plotly 3](#_Toc508797181)

[**17.** **Histograms** Histograms with Plotly 3](#_Toc508797182)

[**18.** **Distplots** Distribution Plots with Plotly 3](#_Toc508797183)

[**19.** **Heatmaps** Heatmaps with plotly 3](#_Toc508797184)

[**20.** **Exercise: Plotly Basics** Exercise Notebook with Plotly Questions 3](#_Toc508797185)

[**21.** **Plotly Basics Exercise Solution** Solutions to exercise questions 3](#_Toc508797186)

[**Code:** 4](#_Toc508797187)

[**Plotly Basics** 4](#_Toc508797188)

[**22.** **Plotly Basics Overview** basic1.py 4](#_Toc508797189)

[**23.** **Scatter Plots** scatter1.py 5](#_Toc508797190)

**Resources:**

**Curriculum:**

<https://drive.google.com/drive/folders/1yPINoaRdCj8DaidXQL3qDyEmdKs-sJLP>

<https://docs.google.com/spreadsheets/d/1I1aAMkKyraFA2ST89SizZwoQ1LKT1KL4ht9tENRJkEE/edit#gid=1589164222>

**Udemy Course Management Page:**

<https://www.udemy.com/course/1575562/manage/curriculum/>

**Github Repository (private):**

<https://github.com/Pierian-Data/Plotly-Dashboards-with-Dash>

**Datasets:**

mpg data: <https://gist.github.com/omarish/5687264>   
Around four hundred 1970's era cars with stats on miles per gallon, acceleration, number of cylinders, etc.

**Choosing an IDE:**

**JupyterLab**

– just released for users (see <https://blog.jupyter.org/jupyterlab-is-ready-for-users-5a6f039b8906>) JupyterLab offers an integrated development platform where multiple programming environments and visualizations appear under the same browser window.

**PROS:**

* Adds some nice navigation features to classic Jupyter.
* Notebooks are already familiar to many students, and are easy to markup.
* Has a *very* strong .csv table renderer that handles big data – but without many features.

**CONS:**

* poor terminal support for Windows. There is a bash terminal option with Jupyter Notebook 5.3, but it may confuse many students.
* Requires some gymnastics to run plotly & dash – installing node.js and a notebook plugin
* Local webserver may collide with Dash's server down the road (untested), as these build independent browser pages.

**Atom**

– more than just a text editor, Atom offers context highlighting, autocomplete, and (with a simple package install) a built-in terminal.

**PROS:**

* Students create scripts right away. Reinforces the habit of performing imports for every script – something that was lacking in Jupyter-based classes.

**CONS:**

* Markup is harder than with Jupyter, particularly with milestone walkthroughs.

**Online vs Offline:**

* I have *not* set up a plotly account, to ensure that I see any errors that pop up without one.
* In the code, I use import plotly.offline as pyo  
  "pyo" is my invention, to emphasize the offline version of plotly.

**Lessons:**

**Plotly Basics**

*Use this data: https://gist.github.com/omarish/5687264*

1. **Plotly Basics Overview**Discuss the various plot types we will show how to do.

* Compare plotly to a static matplotlib plot (with the same data), show the interactivity of plotly in the browser
* Plotly example shows an exotic list comprehension that creates a trace for each column in the DataFrame. Mention that this is covered in much more detail later.   
  I HAVEN'T FOUND A WAY TO DISPLAY A DATAFRAME PLOT AS SUCCINCTLY AS WITH CUFFLINKS
* Show how clicking on a "trace" (in the legend, a trace represents one of the displayed datasets) removes it from the rest, and double-clicking a trace isolates it. Double-click again to redisplay the other traces.
* Show how plotly created a *temp-plot.html* file in the PlotlyBasics folder, and that this is what's showing in the browser. Show that adding a **filename='something-else'** argument lets you change the name of the file (useful when working with multiple plots). Re-running a given script replaces earlier copies of the file.
* Show how a plotly graph can be saved to an image file from the toolbar that appears in the browser.

1. **Scatter Plots**Show Scatter plot examples with basic plotly

* First show a basic scatterplot (markers only). Because we use np.linespace to provide evenly-spaced   
  x-values, this *could* be shown as a lineplot.
* Next show three stacked plots (one of each type) on the same graph.
* Show how a list of traces can be passed in as data

1. **Line Charts**

For this exercise we import a dataset from the U.S. Census Bureau and do some pandas manipulations to extract the rows and columns we want.

* The key pandas operations are:
  + **Create a DataFrame from a US Census Bureau dataset:**  
    df = pd.read\_csv('https://www2.census.gov/programs-surveys/popest/datasets/2010-2017/national/totals/nst-est2017-alldata.csv')
  + **Grab just the six New England states:**df2 = df[df['DIVISION']=='1']
  + **Set the index to state name:**  
    df2.set\_index('NAME', inplace=True)
  + **Grab just the population columns:**  
    df2 = df2[[col for col in df2.columns if col.startswith('POP')]]
* Then, to display the plot in the browser:

pyo.plot([{  
 'x': df2.columns,  
 'y': df2.loc[name],  
 'name': name  
} for name in df2.index])

* One thing I like about this plot is that the populations of Maine and New Hampshire are nearly equivalent, and you don't see this until you hover over the red line.

1. **Bar Charts**Bar Charts with Plotly
2. **Bubble Plots**Bubble Plots with Plotly
3. **Box Plots**Box Plots with Plotly
4. **Histograms**Histograms with Plotly
5. **Distplots**Distribution Plots with Plotly
6. **Heatmaps**Heatmaps with plotly
7. **Exercise: Plotly Basics**Exercise Notebook with Plotly Questions
8. **Plotly Basics Exercise Solution**Solutions to exercise questions

**Code:**

**Plotly Basics**

1. **Plotly Basics Overview**

basic1.py

"""

This script creates a static matplotlib plot

"""

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

*# create fake data:*

df = pd.DataFrame(np.random.randn(100,4),columns='A B C D'.split())

df.plot()

plt.show()

"""

At the terminal run: python plot1.py

Close the plot window to close the script

"""

basic2.py

"""

This script creates the same type of plot as basic1.py,

but in plotly. Note that it creates an .html file!

"""

import numpy as np

import pandas as pd

import plotly.offline as pyo

import plotly.graph\_objs as go

*# create fake data:*

df = pd.DataFrame(np.random.randn(100,4),columns='A B C D'.split())

pyo.plot([{

'x': df.index,

'y': df[col],

'name': col

} for col in df.columns])

1. **Scatter Plots**

scatter1.py

"""

This plots 500 points from 0 to 1 against random

y-axis values that are normal about zero.

"""

import numpy as np

import pandas as pd

import plotly.offline as pyo

import plotly.graph\_objs as go

N = 500

x = np.linspace(0, 1, N)

y = np.random.randn(N)

df = pd.DataFrame({'x': x, 'y': y})

pyo.plot({

"data": [go.Scatter(x=df['x'], y=df['y'],mode='markers')],

"layout": go.Layout(title="simple scatter")

})

scatter2.py

import numpy as np

import pandas as pd

import plotly.offline as pyo

import plotly.graph\_objs as go

N = 100

random\_x = np.linspace(0, 1, N)

random\_y = np.random.randn(N)

*# Create traces*

trace0 = go.Scatter(

x = random\_x,

y = random\_y+5,

mode = 'markers',

name = 'markers'

)

trace1 = go.Scatter(

x = random\_x,

y = random\_y,

mode = 'lines+markers',

name = 'lines+markers'

)

trace2 = go.Scatter(

x = random\_x,

y = random\_y-5,

mode = 'lines',

name = 'lines'

)

data = [trace0, trace1, trace2]

pyo.plot(data, filename='scatter-mode')

1. **Line Charts**
2. **Bar Charts**Bar Charts with Plotly
3. **Bubble Plots**Bubble Plots with Plotly
4. **Box Plots**Box Plots with Plotly
5. **Histograms**Histograms with Plotly
6. **Distplots**Distribution Plots with Plotly
7. **Heatmaps**Heatmaps with plotly
8. **Exercise: Plotly Basics**Exercise Notebook with Plotly Questions
9. **Plotly Basics Exercise Solution**Solutions to exercise questions