* Creating Dashboards with Plotly and Dash
  + Why
    - RT visual
    - Understand moving parts
    - Visually Track, Analyse, and display key performance indicators (KPI)
    - Take informed decisions and improve performance
    - Reduce hours of analysing.
  + Tools
    - Source
      * https://pyviz.org/dashboarding/
    - Dash from Plotly
    - Panel
    - Voila
    - Streamlit
    - Bokeh
    - Ipywidgets
    - Matplotlib
    - Bowtie
    - Flash
  + Intro to Plotly
    - Interactive, opensource library
    - Charts for statistical, financial, maps, scientific, and 3-dimensional data.
    - Plotly Graph Object: low- level interface to figures, traces, and layout
      * plotly.graph\_object.Figure
    - Ploty Express: High-level wrapper
    - Example (using plotly graph object)
      * **import plotly.graph\_objects as go**
      * **import plotly.express as px**
      * **import numpy as np**
      * #creating a random see
        + **np.random.seed(10)**
        + **x=np.arange(12)**
        + **y=np.random.randint(50,500,size=12)**
      * # plotly contains a JSON object which has a structure of dict .
      * # in this case ‘go’ is the Plotly JSON object.
      * **fig = go.Figure(data=go.Scatter(x=x,y=y))**
      * **Chart, line chart

        Description automatically generatedfig.update\_layout(title=’Simple Line Plot’, xaxis\_title=’Month’, yaxis\_title=’Sales’)**
      * **fig.show()**
    - Example (Using plotly.express) (All combined to a single line)
      * **import plotly.graph\_objects as go**
      * **import plotly.express as px**
      * **import numpy as np**
      * #creating a random see
        + **np.random.seed(10)**
        + **x=np.arange(12)**
        + **y=np.random.randint(50,500,size=12)**
      * fig = px.line(x=x, y=y, title=’Simple Line Plot’, labels = dict(x=’Month’, y = ‘Sales’))
      * fig.show()
      * #This creates the same graph.
    - Resources for plotly
      * <https://plotly.com/python/getting-started/>
      * <https://plotly.com/python/graph-objects/>
      * <https://plotly.com/python/plotly-express/>
      * <https://plotly.com/python-api-reference/>
      * <https://images.plot.ly/plotly-documentation/images/plotly_js_cheat_sheet.pdf>
      * <https://community.plotly.com/c/api/5>
      * <https://plotlygraphs.medium.com/>
      * <https://developer.ibm.com/exchanges/data/>
    - Lab Example
      * + **import pandas as pd**
        + **import plotly.express as px**
        + **import plotly.graph\_objects as go**
      * This can be used for randomly sampling 500 data, random\_state = 42 predetermines the random set.
        + **data = airline\_data.sample(n=500, random\_state=42)**

1. **Scatter Plot (Graph Object)**
   1. Idea: How departure time changes with respect to airport distance
   2. Code
      1. # First we create a figure using go.Figure and adding trace to it through go.scatter
      2. **fig = go.Figure(data=go.Scatter(x=data['Distance'], y=data['DepTime'], mode='markers', marker=dict(color='red')))**
      3. # Updating layout through `update\_layout`. Here we are adding title to the plot and providing title to x and y axis.
      4. **fig.update\_layout(title='Distance vs Departure Time', xaxis\_title='Distance', yaxis\_title='DepTime')**
      5. # Display the figure
      6. **fig.show()**
2. **Line Plot (Graph Object)**
   1. Idea: Extract average monthly arrival delay time and see how it changes over the year.
   2. Code
      1. # Group the data by Month and compute average over arrival delay time.
      2. **line\_data = data.groupby('Month')['ArrDelay'].mean().reset\_index()**
      3. # Display the data
      4. **line\_data**
      5. Create a line plot with x-axis being the month and y-axis being computed average delay time. Update plot title,
      6. xaxis, and yaxis title.
      7. Hint: Scatter and line plot vary by updating mode parameter.
      8. **fig = go.Figure(data=go.Scatter(x=line\_data['Month'], y=line\_data['ArrDelay'], mode='lines', marker=dict(color='red')))**
      9. **fig.show()**
3. **Bar Chart**
   1. Idea: Extract number of flights from a specific airline that goes to a destination
   2. Code
      1. # Group the data by destination state and reporting airline. Compute total number of flights in each combination
      2. **bar\_data = data.groupby(['DestState'])['Flights'].sum().reset\_index()**
      3. # Display the data
      4. **bar\_data**
      5. # Use plotly express bar chart function px.bar. Provide input data, x and y axis variable, and title of the chart.
      6. # This will give total number of flights to the destination state.
      7. **fig = px.bar(bar\_data, x="DestState", y="Flights", title='Total number of flights to the destination state split by reporting airline')**
      8. **fig.show()**
4. **Bubble Chart**
   1. # Group the data by reporting airline and get number of flights
   2. **bub\_data = data.groupby(['Reporting\_Airline'])['Flights'].sum().reset\_index()**
   3. # Note that size and hover name is referred to the data columns.
   4. **fig = px.scatter(bub\_data, x = 'Reporting\_Airline', y = 'Flights', title = 'Reporting Airlines vs Flights', size = 'Flights', hover\_name = 'Reporting\_Airline')**
   5. **fig.show()**
5. **Histogram**
   1. Idea: Get distribution of arrival delay
   2. Code:
      1. # Set missing values to 0
      2. **data['ArrDelay'] = data['ArrDelay'].fillna(0)**
      3. Use px.histogram and pass the dataset.
      4. Pass ArrDelay to x parameter.
      5. # Create histogram here
      6. **fig = px.histogram(bub\_data, x = data['ArrDelay'], title = 'Distribution of Arrival Delay', nbins = 20)**
      7. **fig.show()**
6. **Pie Chat**
   1. Idea: Proportion of distance group by month (month indicated by numbers)
   2. Code:
      1. # Use px.pie function to create the chart. Input dataset.
      2. # Values parameter will set values associated to the sector. 'Month' feature is passed to it.
      3. # labels for the sector are passed to the `names` parameter.
      4. **fig = px.pie(data, values='Month', names='DistanceGroup', title='Distance group proportion by month')**
      5. **fig.show()**
7. **Sunburst Charts**
   1. Idea: Hierarchical view in othe order of month and destination state holding value of number of flights
   2. Code:
      1. # Create sunburst chart here
      2. **fig = px.sunburst(data, path = ['Month', 'DestStateName'], values = 'Flights', title='Hierarchical view in othe order of month and destination state holding value of number of flights')**
      3. **fig.show()**

* **Dash Library**
  + **Import dash**
  + Open-source user interface python library from plotly.
  + Easy to build GUI
  + Declarative and Reactive
  + Rendered in web browser and can be deployed to servers
  + Inherently cross-platform and mobile ready.
  + Components
    - Core
      * **Import dash\_core\_components as dcc**
      * Higher level components that are interactive and are generated with JavaScript, HTML, and CSS through the React.js library.
      * Example: creating a slider, input area, check items, datepicker and so on.
    - HTML
      * **Import dash\_html\_components as html**
      * Keyword arguments describe the HTML attributes like style, className, and id
  + Lab
    - Create a new file name.py
    - Open new terminal
* **pip3** install pandas dash
  + - Once the code is completed use to run the application.
      * **python3 Airline\_Dashboard.py**
    - Now note the port in ####, and click launce application. Insert port and it should run.
  + Resources
    - [**https://dash.plotly.com/**](https://dash.plotly.com/)
    - [**https://dash.plotly.com/dash-core-components**](https://dash.plotly.com/dash-core-components)
    - [**https://dash.plotly.com/dash-html-components**](https://dash.plotly.com/dash-html-components)
    - [**https://community.plotly.com/c/dash/16**](https://community.plotly.com/c/dash/16)
    - [**https://medium.com/plotly/tagged/dash**](https://medium.com/plotly/tagged/dash)
  + Call back
    - Packages
      * **Import pandas as pd**
      * **Import plotly.express as px**
      * **Import dash**
      * **Import dash\_html\_components as html**
      * **Import dash\_core\_components as dcc**
      * **From dash.dependencies import Input, Output**
    - Call back function is a python function that are automatically called by dash whenever an input component’s property changes.
    - Decorator
      * **@app.callback**
    - Example
      * **@app\_callback(Output, Input)**
      * **Def callback\_function:**
      * **-**
      * **-**
      * **return some\_result**
    - **Graphical user interface, text

      Description automatically generated with medium confidenceText

      Description automatically generated**First read the dateFrame with **pd.read\_csv(‘link’, encoding = ‘type’, dtype={‘column’ : dtype}**

**Text

Description automatically generated with low confidenceText

Description automatically generated with medium confidenceGraphical user interface, application, Teams

Description automatically generated**

**Resources**

* [**https://realpython.com/primer-on-python-decorators/**](https://realpython.com/primer-on-python-decorators/)
* [**https://www.python.org/dev/peps/pep-0318/#current-syntax**](https://www.python.org/dev/peps/pep-0318/#current-syntax)
* [**https://dash.plotly.com/basic-callbacks**](https://dash.plotly.com/basic-callbacks)
* [**https://dash-gallery.plotly.host/Portal/**](https://dash-gallery.plotly.host/Portal/)
* [**https://plotly.com/dash-community-components/**](https://plotly.com/dash-community-components/)