3%20-%20Plotly

April 12, 2023

Table of Contents

https://plotly.com/python-api-reference/plotly.express.html

https://plotly.com/python/

1. Plotly express
   - bar chart
   - line chart
   - scatterplot
   - exercise - pick two, create share
2. plotly graph objects (go)
   - figure structure - The structure of a figure - data, traces and layout explained
     - https://plotly.com/python/figure-structure/
     - tree of attributes
     - data (aka traces)
     - layout
     - frames (used in animated plots)
   - display figures
     - in a notebook or script… fig.show()
     - renderers png, jpeg, etc. fig.show(renderer="png", width=800, height=300)
     - export to html
     - static using Kaleido…..https://plotly.com/python/static-image-export/
   - bar charts
   - line charts

    - scatterplot
    - map
3. subplots
    - https://plotly.com/python/creating-and-updating-figures/
    - go down to subplot section

1 Getting the data

These next few steps read data from data.cdc.gov and do some clean-up and data prep.

[ ]:

```
import requests
import pandas as pd
import numpy as np
import plotly.express as px
```

[ ]:

```
# Get the data from CDC and look at it in json format

response = requests.get("https://data.cdc.gov/resource/saz5-9hgg.json")
jsonhold = response.json()
#jsonhold
```

[ ]:

```
# Put the data into a DataFrame

vaccines = pd.DataFrame(jsonhold)

# Create month and week columns

vaccines['month'] = pd.to_datetime(vaccines['week_of_allocations']).dt.month
vaccines['day'] =  pd.to_datetime(vaccines['week_of_allocations']).dt.day



# Changing the datatypes & column names

vaccines['month'] = vaccines.month.astype(str)
vaccines['day'] = vaccines.day.astype(str)
vaccines['_1st_dose_allocations'] = pd.
 ↪to_numeric(vaccines['_1st_dose_allocations']).astype(int)
vaccines['_2nd_dose_allocations'] = pd.
 ↪to_numeric(vaccines['_2nd_dose_allocations']).astype(int)
vaccines['_2nd_dose_allocations'] = vaccines._2nd_dose_allocations*1.2


short_names = {'_1st_dose_allocations':'first',
               '_2nd_dose_allocations':'second'}
vaccines.rename(columns=short_names, inplace=True)
vaccines = vaccines[vaccines.jurisdiction.isin(['Massachusetts','New␣
 ↪Hampshire', 'Rhode Island'])]
```

```python
vaccines.head()
```

[ ]:
```python
vaccines.shape
```

[ ]:
```python
vaccines = vaccines.sort_values(by='month')

fig = px.line(vaccines,
              x = 'month',
              y = 'first',
              color = 'jurisdiction',
              markers = True,
              symbol = 'jurisdiction')
fig.show()
```

[ ]:
```python
v_day = vaccines.groupby('day').sum().reset_index()
v_day.head()
```

[ ]:
```python
v_month = vaccines.groupby('month').sum().reset_index()
v_month.head()
```

[ ]:
```python
v_sm = vaccines.groupby(['jurisdiction','month']).sum().reset_index()
v_sm.head()
```

2 plotly express

https://plotly.com/python-api-reference/plotly.express.html

[ ]:
```python
fig = px.line(v_day, x = 'day', y = 'first')
fig.show()
```

[ ]:
```python
fig = px.line(v_sm,
              x = 'month',
              y = 'first',
              color = 'jurisdiction',
              markers = True,
              symbol = 'jurisdiction',
              text = 'first')
fig.show()
```

[ ]:
```python
fig = px.scatter(v_sm,
              x = 'first',
              y = 'second')
fig.show()
```

[ ]:

```
# Using aggregated data
fig = px.bar(v_sm,
              x = 'month',
              y = 'first')
fig.show()
```

[ ]:

```
fig = px.bar(v_sm,
             x = 'month',
             y = 'first',
             color = 'jurisdiction')
fig.show()
```

[ ]:

```
# Continuous color
fig = px.bar(v_month,
             x = 'month',
             y = 'first',
             color = 'second')
fig.show()
```

[ ]:

```
# Unaggregated data

fig = px.bar(vaccines, x = 'jurisdiction', y = 'first', color = 'month')
fig.show()
```

[ ]:

```
# A more dramatic example of same phenomena

df = px.data.tips()
fig = px.bar(df,
             x="sex",
             y="total_bill",
             color='time')
fig.show()
```

[ ]:

```
# Stacked unaggregated data

fig = px.bar(vaccines, x = 'jurisdiction', y = 'first', color = 'month')
fig.show()
```

[ ]:

```
# Side-by-side unaggregated data

fig = px.bar(vaccines,
             x = 'jurisdiction',
             y = 'first',
             color = 'month',
             barmode = 'group')
fig.show()
```

[ ]:

```
# Use histogram to aggregate

fig = px.histogram(vaccines,
             x = 'jurisdiction',
             y = 'first',
             color = 'month',
             barmode = 'group')
fig.show()
```

[ ]:

```
# faceted subplots   ##### Different dataset!

df = px.data.tips()
fig = px.bar(df,
             x="sex",
             y="total_bill",
             color="smoker",
             barmode="group",
             facet_row="time",
             facet_col="day",
             category_orders={"day": ["Thur", "Fri", "Sat", "Sun"],
                              "time": ["Lunch", "Dinner"]})
fig.show()
```

[ ]:

```

```

3 Exercise 1 - 10 minutes

[ ]:

```
# Exercise 1 plotly express 1 - pie chart
```

[ ]:

```
# Exercise 1 plotly express 2 - boxplot
```

4 plotly graph objects

4.1 Getting the data ready

[ ]:

```
import plotly.graph_objects as go
import pandas as pd
ob = pd.read_csv('https://raw.githubusercontent.com/jimcody2014/Python-Data/
 ↪main/outbreaks-dashboard.csv')
ob.head()
```

[ ]:

```
ob_month = ob.groupby('Month')[['Illnesses','Hospitalizations', 'Fatalities']].
 ↪sum().reset_index()
```

```
[ ]: oby = ob.groupby('Year')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
      ↪reset_index()
```

```
[ ]: obs = ob.groupby('State')[['Illnesses','Hospitalizations', 'Fatalities']].sum().
      ↪reset_index()
```

4.2 Bar Charts

```
[ ]: # Basic graph object
     fig = go.Figure(
         data=[go.Bar(x=['apples', 'oranges', 'bananas'], y=[1, 3, 2])],
         layout=go.Layout(
             title=go.layout.Title(text="A Figure Specified By A Graph Object")
         )
     )

     fig.show()
```

```
[ ]: print(fig)
```

```
[ ]: # Very minimal

     fig = go.Figure([go.Bar(x=['apples', 'oranges', 'bananas'], y=[1, 3, 2])])
     fig.show()
```

```
[ ]: # With dataframe data - version 1
     fig = go.Figure(go.Bar(x=ob['Month'], y = ob['Illnesses'],hovertemplate = "%{x}:
      ↪ <br>Illnesses: %{y} </br> %{y}"))
     fig.show()
```

```
[ ]: # With dataframe data - version 2 - just a different way of accessing the␣
      ↪variables

     fig = go.Figure(go.Bar(x=ob.Month, y = ob.Illnesses))
     fig.show()
```

```
[ ]: # With aggregated dataframe data
     fig = go.Figure(go.Bar(x=ob_month.Month, y = ob_month.Illnesses))
     fig.show()
```

```
[ ]: fig = go.Figure(go.Bar(x=ob_month.Month, y = ob_month.Illnesses))
     fig.update_layout(xaxis={'categoryorder':'array', 'categoryarray':
      ↪['January','February','March','April','May','June','July','August',
                               'September','October','November','December']})
     fig.show()
```

[ ]:

```
# Multiple traces
fig = go.Figure(
    data=[go.Bar(name = 'ill', x=ob_month.Month, y = ob_month.Illnesses),
         go.Bar(name = 'hosp', x=ob_month.Month, y = ob_month.
 ↪Hospitalizations)],
    layout=go.Layout(
        title=go.layout.Title(text="A Figure Specified By A Graph Object")
    )
)

fig.show()
```

[ ]:

```
# Layout update
fig = go.Figure(
    data=[go.Bar(name = 'ill', x=ob_month.Month, y = ob_month.Illnesses),
         go.Bar(name = 'hosp', x=ob_month.Month, y = ob_month.
 ↪Hospitalizations)],
    layout=go.Layout(
        title=go.layout.Title(text="A Figure Specified By A Graph Object")
    )
)
fig.update_layout(barmode='stack')
fig.show()
```

[ ]:

```
# From the documentation - Adding multiple 'traces'
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
          'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

fig = go.Figure()
fig.add_trace(go.Bar(
    x=months,
    y=[20, 14, 25, 16, 18, 22, 19, 15, 12, 16, 14, 17],
    name='Primary Product',
    marker_color='indianred'
))
fig.add_trace(go.Bar(
    x=months,
    y=[19, 14, 22, 14, 16, 19, 15, 14, 10, 12, 12, 16],
    name='Secondary Product',
    marker_color='lightsalmon'
))

# Here we modify the tickangle of the xaxis, resulting in rotated labels.
fig.update_layout(barmode='group', xaxis_tickangle=-45)
fig.show()
```

[ ]:
```
# Modifying the Hover text & traces update

fig = go.Figure(go.Bar(x=ob_month.Month, y = ob_month.Illnesses,
                      hovertext=['A lot', 'medium', 'Big']))

fig.update_traces(marker_color='rgb(158,202,225)',␣
 ↪marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5, opacity=0.6)
fig.update_layout(title_text='Outbreaks by Month')
fig.show()
```

[ ]:
```
# Modifying colors

# amts = [37,27,33,30,29,30,35,33,37,32,27,24]
colors = ['lightslategray',] * 12
colors[11] = 'crimson'

fig = go.Figure(go.Bar(x=ob_month.Month, y = ob_month.Illnesses,
                      hovertext=['A lot', 'medium', 'Big'],
                      text = ob_month.Illnesses,
                      textposition = 'auto',
                      marker_color = colors)
               )

fig.update_layout(title_text='Outbreaks by Month')

fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')

fig.show()
```

[ ]:
```
# Sorting as part ogf the layout

fig = go.Figure(
    data=[go.Bar(name = 'ill', x=ob_month.Month, y = ob_month.Illnesses),
         go.Bar(name = 'hosp', x=ob_month.Month, y = ob_month.
 ↪Hospitalizations)],
    layout=go.Layout(
        title=go.layout.Title(text="A Figure Specified By A Graph Object")
    )
)
fig.update_layout(barmode='stack', xaxis={'categoryorder':'total ascending'}) ␣
 ↪# descending
fig.show()
```

4.3 Scatterplot

Reminder: ob is outbreaks. ob_month is outbreak data aggregated to the month

[ ]:

```
fig = go.Figure(data=go.Scatter(x=ob_month.Illnesses, y=ob_month.Fatalities,␣
 ↪mode = 'markers'))
fig.show()
```

[ ]:

```
# Same figure as above
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=ob_month.Illnesses,
    y=ob_month.Fatalities,
    mode = 'markers',
    marker_color='indianred'
))
```

When using Plotly graphic objects, **Scatter** is also used to create line charts. The marker used chnges the style.

[ ]:

```
# From documentation

import numpy as np
np.random.seed(1)

N = 100
random_x = np.linspace(0, 1, N)
random_y0 = np.random.randn(N) + 5
random_y1 = np.random.randn(N)
random_y2 = np.random.randn(N) - 5

# Create traces
fig = go.Figure()
fig.add_trace(go.Scatter(x=random_x, y=random_y0, mode='lines', name='lines'))
fig.add_trace(go.Scatter(x=random_x, y=random_y1, mode='lines+markers',␣
 ↪name='lines+markers'))
fig.add_trace(go.Scatter(x=random_x, y=random_y2, mode='markers',␣
 ↪name='markers'))

fig.show()
```

[ ]:

```
# Change the marker size
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=ob_month.Illnesses,
    y=ob_month.Hospitalizations,
    mode = 'markers',
    marker_size=ob_month.Fatalities,
    marker_color='indianred'

# Below are different formatting options to try.

    #marker_color = ob_month.Fatalities
    #marker=dict(
    #    size=16,
    #    color=ob_month.Fatalities, #set color equal to a variable
    #    colorscale='inferno', # one of plotly colorscales
    #    showscale=True
    #)
))
#fig.update_traces(mode='markers', marker_line_width=2, marker_size=ob_month.
 ↪Fatalities)
# If multiple traces exist, the update will be applied to all traces.

#fig.update_layout(title='Sized Scatterplot')

# Update the x axes
#fig.update_xaxes(tickangle = 90,title_text = "Illnesses",title_font={"size":␣
 ↪20},title_standoff = 25)
#fig.update_xaxes(showline=True, linewidth=2, linecolor='black')
#fig.update_xaxes(showgrid=False)

# Update the x axes
#fig.update_yaxes(title_text = "Hospitalizations",title_standoff = 25)
#fig.update_yaxes(showline=True, linewidth=2, linecolor='black')
#fig.update_yaxes(title_font=dict(size=18, family='Courier', color='crimson'))
#fig.update_yaxes(ticklabelposition="inside top", title='Hospitalizations')



fig.show()

# https://plotly.com/python/builtin-colorscales/
```

[ ]:

```
# Using a large dataset - from documentation
N = 100000
fig = go.Figure(data=go.Scattergl(
    x = np.random.randn(N),
    y = np.random.randn(N),
    mode='markers',
    marker=dict(
        color=np.random.randn(N),
        colorscale='Viridis',
        line_width=1
    )
))
fig.show()
```

4.4 Line Charts

When using graph objects, line charts are scatter charts with connected marks.

[ ]:

```
# Line charts are Scatter charts with connected markers.
# The default scatter creates a line

fig = go.Figure(go.Scatter(x=oby.Year, y=oby.Illnesses))
fig.show()
```

[ ]:

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=oby.Year,
                          y=oby.Illnesses,
                          name = 'Illnesses'))

fig.add_trace(go.Scatter(x=oby.Year,
                        y=oby.Hospitalizations,
                        name = 'Hospitalizations',
                        line=dict(color='lightgrey', width=4, dash='dot')))
# dash options include 'dash', 'dot', and 'dashdot'

fig.add_trace(go.Scatter(x=oby.Year,
                          y=oby.Fatalities,
                          name = 'Fatalities'))

fig.update_layout(title='Illnesses by Year',
                   xaxis_title='Year',
                   yaxis_title='Number of Illnesses')


fig.show()
```

5 Exercise - 30 minutes

- Create a new notebook (don't forget the imports)
- Name the notebook **Diabetes Analysis Dashboard**
- read in the diabetes_for_plotly dataset
- group data as needed
- Use express or graph objects
- Create a scatter plot of any two measures. Use a third measure to adjust the size. Color by a categorical value. Add hover text to show the age group.
- Create a side-by-side bar chart showing number of lab procedures and number of non lab procedures by gender.

- Create a line chart showing number of number of medications by month.
- Create a line chart showing number of number of procedures by month.
- Create a fifth chart of your choice (NOT scatter, bar or line) using the documentation.

https://bitbucket.org/jimcody/sampledata/raw/b2aa6df015816ec35afc482b53df1b7ca7a31f80/diabetes_for_plotl