LabSession_AdvancedViz

September 26, 2024

1 Submission

Put the ipynb file and html file in the github branch you created in the last assignment and submit the link to the commit in brightspace

[1]:
```
from plotly.offline import init_notebook_mode
import plotly.io as pio
import plotly.express as px

init_notebook_mode(connected=True)
pio.renderers.default = "plotly_mimetype+notebook"
```

[2]:
```
#load data
df = px.data.gapminder()
df.head()
```

[2]:

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	iso_num
0	Afghanistan	Asia	1952	28.801	8425333	779.445314	AFG	4
1	Afghanistan	Asia	1957	30.332	9240934	820.853030	AFG	4
2	Afghanistan	Asia	1962	31.997	10267083	853.100710	AFG	4
3	Afghanistan	Asia	1967	34.020	11537966	836.197138	AFG	4
4	Afghanistan	Asia	1972	36.088	13079460	739.981106	AFG	4

1.1 Question 1:

Recreate the barplot below that shows the population of different continents for the year 2007.

Hints:

- Extract the 2007 year data from the dataframe. You have to process the data accordingly
- use plotly bar
- Add different colors for different continents

- Sort the order of the continent for the visualisation. Use axis layout setting
- Add text to each bar that represents the population

[3]:
```
df_2007 = df[df['year'] == 2007]
continent_pop = df_2007.groupby('continent', as_index=False)['pop'].sum()
continent_pop = continent_pop.sort_values(by='continent', ascending=True)
fig = px.bar(continent_pop,
             x='pop',
             y='continent',
             color='continent')
fig.update_layout(
    yaxis_title='continent',
    xaxis_title='pop',
    showlegend=False)
fig.show()
```

1.2 Question 2:

Sort the order of the continent for the visualisation

Hint: Use axis layout setting

[4]:
```
df_2007 = df[df['year'] == 2007]
continent_pop = df_2007.groupby('continent', as_index=False)['pop'].sum()
continent_pop = continent_pop.sort_values(by='pop', ascending=False)
fig = px.bar(continent_pop,
             x='pop',
             y='continent',
             color='continent')
fig.update_layout(
    yaxis_title='continent',
    xaxis_title='pop',
    showlegend=False)
fig.show()
```

1.3 Question 3:

Add text to each bar that represents the population

[5]:
```
df_2007 = df[df['year'] == 2007]
continent_pop = df_2007.groupby('continent', as_index=False)['pop'].sum()
continent_pop = continent_pop.sort_values(by='pop', ascending=False)
continent_pop['pop_text'] = continent_pop['pop'].apply(
    lambda x: f'{x / 1e9:.1f}B' if x >= 1e9 else f'{x / 1e6:.0f}M')
fig = px.bar(continent_pop,
             x='pop',
             y='continent',
             color='continent',
             text='pop_text')
```

```
fig.update_layout(
    yaxis_title='continent',
    xaxis_title='pop',
    showlegend=False)
fig.update_traces(textposition='outside')
fig.show()
```

1.4 Question 4:

Thus far we looked at data from one year (2007). Lets create an animation to see the population growth of the continents through the years

[12]:
```
df_year = df.groupby(['year', 'continent'])['pop'].sum().reset_index()
df_year = df_year.sort_values(by=['year', 'pop'], ascending=[True, False])
fig = px.bar(data_frame=df_year,
             x='pop',
             y='continent',
             animation_frame='year',
             animation_group='continent',
             color='continent')
fig.update_layout(
    yaxis_title='continent',
    xaxis_title='pop',
    showlegend=False,
    yaxis={'categoryorder': 'total ascending'})
fig.show()
```

1.5 Question 5:

Instead of the continents, lets look at individual countries. Create an animation that shows the population growth of the countries through the years

[15]:
```
df_year_country = df.groupby(['year', 'country'])['pop'].sum().reset_index()
df_year_country = df_year_country.sort_values(by=['year', 'pop'],␣
 ↪ascending=[True, False])
fig = px.bar(data_frame=df_year_country,
             x='pop',
             y='country',
             animation_frame='year',
             animation_group='country',
             color='country')
fig.update_layout(
    yaxis_title='country',
    xaxis_title='pop',
    showlegend=False,
    yaxis={'categoryorder': 'total ascending'})
fig.show()
```

1.6 Question 6:

Clean up the country animation. Set the height size of the figure to 1000 to have a better view of the animation

[16]:
```
df_year_country = df.groupby(['year', 'country'])['pop'].sum().reset_index()
df_year_country = df_year_country.sort_values(by=['year', 'pop'],␣
 ↪ascending=[True, False])
fig = px.bar(data_frame=df_year_country,
             x='pop',
             y='country',
             animation_frame='year',
             animation_group='country',
             color='country')
fig.update_layout(
    yaxis_title='country',
    xaxis_title='pop',
    showlegend=False,
    yaxis={'categoryorder': 'total ascending'},
    height=1000)
fig.show()
```

1.7 Question 7:

Show only the top 10 countries in the animation

Hint: Use the axis limit to set this.

[20]:
```
df_year_country = df.groupby(['year', 'country'])['pop'].sum().reset_index()
df_year_country = df_year_country.sort_values(by=['year', 'pop'],␣
 ↪ascending=[True, False])
df_top10 = df_year_country.groupby('year').apply(lambda x: x.nlargest(10,␣
 ↪'pop')).reset_index(drop=True)
fig = px.bar(data_frame=df_top10,
             x='pop',
             y='country',
             animation_frame='year',
             animation_group='country',
             color='country')
fig.update_layout(
    yaxis_title='country',
    xaxis_title='pop',
    showlegend=False,
    yaxis={'categoryorder': 'total ascending'})
fig.show()
```

```
C:\Users\olede\AppData\Local\Temp\ipykernel_19976\3849235328.py:3:
DeprecationWarning:
```

```
DataFrameGroupBy.apply operated on the grouping columns. This behavior is
deprecated, and in a future version of pandas the grouping columns will be
excluded from the operation. Either pass `include_groups=False` to exclude the
groupings or explicitly select the grouping columns after groupby to silence
this warning.
```