Assignment 5: Plotly Express and Plotly Dash

The visualizations that I created in this notebook were certainly challenging. I, of course, stuck to the data that I used in Assignment 4 - penguins. I tried to follow basically the same principals that I used in the previous assignment. However, I was able to improve some things in this assignment that I had hoped to accomplish in the last one, so that was nice. Firstly, in Assignment 4, I had 3 separate count plots describing the populations on each island. I was happy that I was able to turn that into 1 interactive count plot here. Secondly, in my previous assignment, I tried to have different marker symbols to represent males and females in my scatter plots. While I failed in Assignment 4, I succeeded in Assignment 5. Finally, in Assignment 4, I had wanted to go into more detail about the differences in physical characteristics between the sexes in each of penguin species, but did not want to be so repetetive in my plots. Thankfully, in this assignment, the user may choose which feature to compare using only one plot. After this assignment, I feel much better equipped with Plotly Express and Plotly Dash.

To view my visualizations, run the whole notebook and then scroll down until you reach the third block of code. I've moved my plots closer to the top of this notebook for easier access, but I originally had them at the very bottom as I was working through everything. I've also chose to have my plots inline with the notebook, but if you would prefer to view them in a separate web browser window, simply change the third block of code from

> if **name** == "**main**":
>
> app.run_server(mode="inline")

to

> if **name** == "**main**":
>
> app.run_server(mode="external")

In [ ]:
```
import pandas as pd
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from dash import Dash, dcc, html
from dash.dependencies import Input, Output
from jupyter_dash import JupyterDash

app = Dash(__name__)
```

In [ ]:
```
app = JupyterDash()

app.layout = html.Div(
    children=[
        html.H1(
            children="Penguins of the Antarctic Palmer Archipelago",
            style={"text-align":"center"}
```

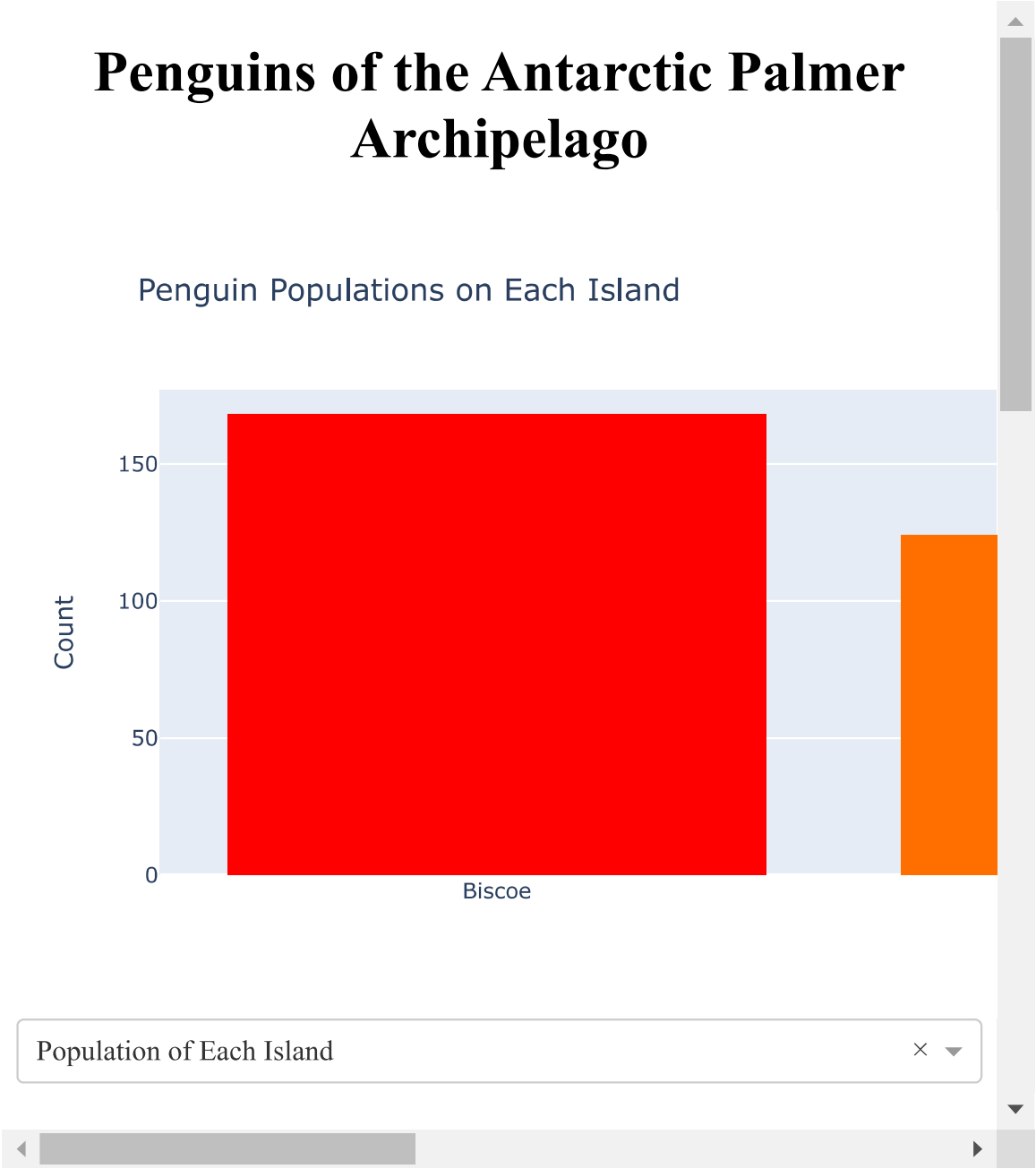
```
        ),
        dcc.Graph(
            id="island_graph"
        ),
        dcc.Dropdown(
            id="island_dropdown",
            options=[
                {"label":"Population of Each Island", "value":"Island"},
                {"label":"Population of Each Species on Each Island", "value":"Spe
                {"label":"Population of Each Sex on Each Island", "value":"Sex"}
            ],
            multi=False,
            value="Island"
        ),
        html.Br(),
        html.Br(),
        html.Br(),


        dcc.Graph(
            id="species_graph"
        ),
        dcc.Dropdown(
            id="species_dropdown_x",
            options=[
                {"label":"X: Bill Length (mm)", "value":"Bill Length (mm)"},
                {"label":"X: Bill Depth (mm)", "value":"Bill Depth (mm)"},
                {"label":"X: Flipper Length (mm)", "value":"Flipper Length (mm)"},
                {"label":"X: Body Mass (g)", "value":"Body Mass (g)"}
            ],
            multi=False,
            value="Bill Length (mm)"
        ),
        dcc.Dropdown(
            id="species_dropdown_y",
            options=[
                {"label":"Y: Bill Length (mm)", "value":"Bill Length (mm)"},
                {"label":"Y: Bill Depth (mm)", "value":"Bill Depth (mm)"},
                {"label":"Y: Flipper Length (mm)", "value":"Flipper Length (mm)"},
                {"label":"Y: Body Mass (g)", "value":"Body Mass (g)"}
            ],
            multi=False,
            value="Bill Length (mm)"
        ),
        html.Br(),
        html.Br(),
        html.Br(),


        dcc.Graph(
            id="sex_graph"
        ),
        dcc.Dropdown(
            id="sex_dropdown",
            options=[
                "Bill Length (mm)",
                "Bill Depth (mm)",
                "Flipper Length (mm)",
                "Body Mass (g)"
            ],
            multi=False,
            value="Bill Length (mm)"
        ),
        html.Br(),
```

```
        html.Br(),
        html.Br()
    ]
)
```

In [ ]:
```
if __name__ == "__main__":
    app.run_server(mode="inline")
```



In [ ]:
```
df = pd.DataFrame(sns.load_dataset("penguins"))

df.columns = ["Species", "Island", "Bill Length (mm)", "Bill Depth (mm)", "Flipper

df.sort_values(by=["Sex", "Species", "Island"], inplace=True)

df
```

Out[ ]:

	Species	Island	Bill Length (mm)	Bill Depth (mm)	Flipper Length (mm)	Body Mass (g)	Sex
20	Adelie	Biscoe	37.8	18.3	174.0	3400.0	Female
22	Adelie	Biscoe	35.9	19.2	189.0	3800.0	Female
25	Adelie	Biscoe	35.3	18.9	187.0	3800.0	Female
27	Adelie	Biscoe	40.5	17.9	187.0	3200.0	Female
28	Adelie	Biscoe	37.9	18.6	172.0	3150.0	Female
...	...	...	...	...	...	...	...
246	Gentoo	Biscoe	44.5	14.3	216.0	4100.0	NaN
286	Gentoo	Biscoe	46.2	14.4	214.0	4650.0	NaN
324	Gentoo	Biscoe	47.3	13.8	216.0	4725.0	NaN
336	Gentoo	Biscoe	44.5	15.7	217.0	4875.0	NaN
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN

344 rows × 7 columns

In [ ]:
```
colors = px.colors.sequential.Rainbow_r[:6] + px.colors.sequential.Rainbow_r[7:]
colors
```

Out[ ]:
```
['rgb(255,0,0)',
 'rgb(255,111,0)',
 'rgb(255,234,0)',
 'rgb(151,255,0)',
 'rgb(44,255,150)',
 'rgb(0,152,255)',
 'rgb(0,0,200)',
 'rgb(150,0,90)']
```

In [ ]:
```
@app.callback(
    Output(component_id="island_graph", component_property="figure"),
    Input(component_id="island_dropdown", component_property="value")
)
def update_island_graph(selected):

    # change colors to match variables
    if selected == "Island": x=0
    if selected == "Species": x=3
    if selected == "Sex": x=6

    fig = px.histogram(
        df,
        x="Island",
        color=selected,
        color_discrete_sequence=colors[x:], # insert changed color location
        title="Penguin Populations on Each Island"
    )

    fig.update_layout(
        yaxis=dict(
            title="Count"
        )
    )

    return fig
```

In [ ]:
```
@app.callback(
    Output(component_id="species_graph", component_property="figure"),
    [Input(component_id="species_dropdown_x", component_property="value"),
     Input(component_id="species_dropdown_y", component_property="value")]
)
def update_species_graph(selected_x, selected_y):
    fig = px.scatter(
        df,
        x=selected_x,
        y=selected_y,
        hover_data=df.columns,
        color="Species",
        symbol="Sex",
        color_discrete_sequence=colors[3:],
        title="Comparisons Between Penguin Body Measurements"
    )

    fig.update_traces(marker=dict(size=10,
                                  line=dict(width=1,
                                            color="black")))

    fig.update_layout(
        legend=dict(
            title=""
        )
    )

    return fig
```

In [ ]:
```
@app.callback(
    Output(component_id="sex_graph", component_property="figure"),
    Input(component_id="sex_dropdown", component_property="value")
)
def update_sex_graph(selected):
    fig = go.Figure()

    fig.add_trace(go.Violin(
        x=df["Species"][df["Sex"] == "Female"],
        y=df[selected][ df['Sex'] == 'Female' ],
        legendgroup='Female',
        scalegroup='Female',
        name='Female',
        side='negative',
        line_color=colors[6]
        )
    )

    fig.add_trace(go.Violin(
        x=df["Species"][df["Sex"] == "Male"],
        y=df[selected][ df['Sex'] == 'Male' ],
        legendgroup='Male',
        scalegroup='Male',
        name='Male',
        side='positive',
        line_color=colors[7]
        )
    )

    fig.update_layout(
        title="Comparison of Male and Female Penguin Physical Characteristics",
        xaxis_title="Species",
        yaxis_title=selected
    )
```

```
    return fig
```