Business 4720 - Class 8

Data Visualization with Python

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
jevermann@mun.ca

This Class

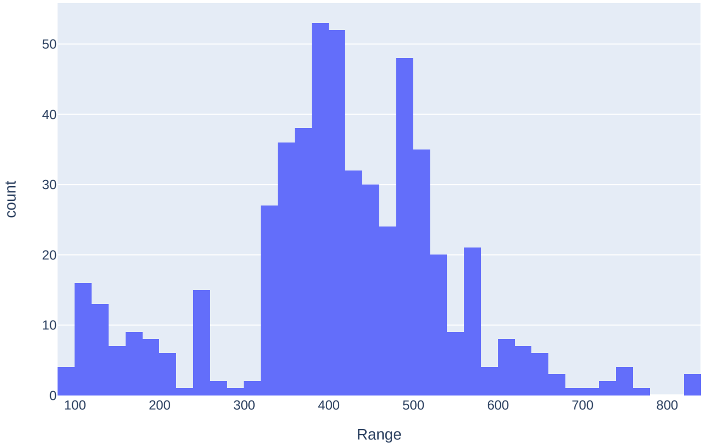
What You Will Learn:

- Visualizing data with Python using the Plotly Express library
- Interactive data dashboards with Plotly Dash

Histogram

```python
import pandas as pd
import plotly.express as px
import plotly.io as pio
pio.kaleido.scope.mathjax = None

# Read data
fuel = pd.read_csv('fuel.csv')

# Create histogram
fig = px.histogram(fuel, x='Range', nbins=50)

# Show histogram, by default show
# in interactive way in browser
fig.show()

# Save figure to image
fig.write_image("px.histogram.pdf",
       height=500, width=750)
```

Histogram



Histogram with Summary Information

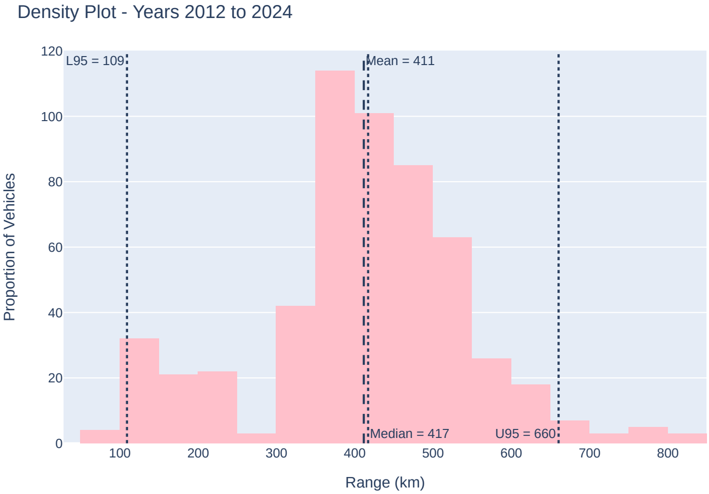
Prepare some summary statistics:

```
# Calculating summary statistics
mean_v = fuel['Range'].mean()
median_v = fuel['Range'].median()
lower95 = fuel['Range'].quantile(0.025)
upper95 = fuel['Range'].quantile(0.975)

# Creating the density plot
fig = px.histogram(fuel, x='Range',
         color_discrete_sequence=['pink'])
```

Histogram with Summary Information [cont'd]

```python
# Adding vertical lines and annotations
fig.add_vline(x=mean_v, line_dash='dash',
      annotation_text=f'Mean = {round(mean_v)}',
      annotation_position='top right')
fig.add_vline(x=median_v, line_dash='dot',
      annotation_text=f'Median = {round(median_v)}',
      annotation_position='bottom right')
fig.add_vline(x=lower95, line_dash='dot',
      annotation_text=f'L95 = {round(lower95)}',
      annotation_position='top left')
fig.add_vline(x=upper95, line_dash='dot',
      annotation_text=f'U95 = {round(upper95)}',
      annotation_position='bottom left')

fig.update_layout(
    title='Density Plot - Years 2012 to 2024',
    xaxis_title='Range (km)',
    yaxis_title='Proportion of Vehicles')
```

Histogram with Summary Information



Column Chart

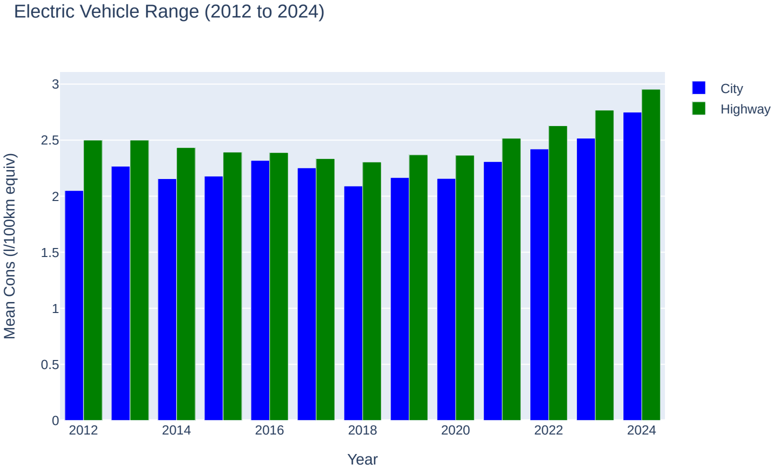
```
fuel_grouped = fuel.groupby('Year').agg(
       meanCity=pd.NamedAgg('City', 'mean'),
       meanHwy=pd.NamedAgg('Hwy', 'mean')) \
         .reset_index()

fuel_long = pd.melt(fuel_grouped,
                id_vars=['Year'],
                value_vars=['meanCity', 'meanHwy'],
                var_name='metric',
                value_name='consumption')

fuel_long['metric'] = fuel_long['metric'] \
       .map({'meanCity': 'City',
             'meanHwy': 'Highway'})
```

Column Chart [cont'd]

Continued from previous slide ...

```
fig = px.bar(fuel_long, x='Year', y='consumption',
   color='metric', barmode='group',
   labels={'consumption': 'Mean Cons\n(l/100km equiv)',
           'metric': ''},
   title='Electric Vehicle Range (2012 to 2024)',
   color_discrete_map={'City': 'blue',
                       'Highway': 'green'})

fig.update_layout(
    xaxis_title='Year',
    yaxis_title='Mean Cons\n(l/100km equiv)')
```

Column Chart

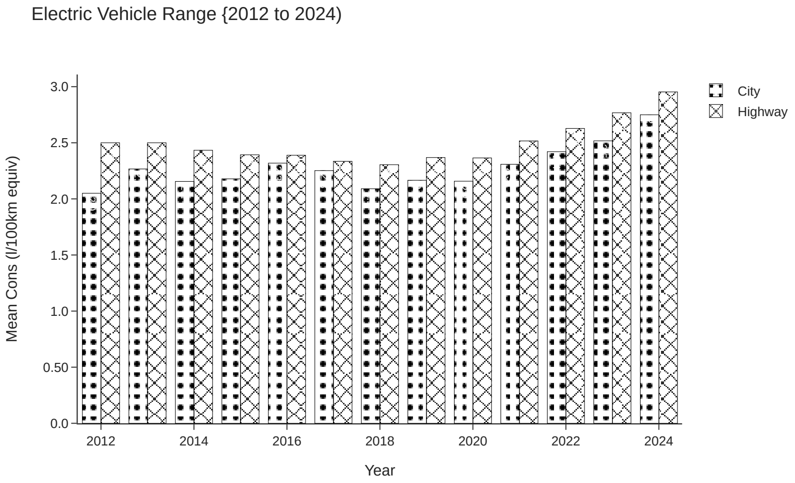


Column Chart (with Patterns)

Prepare data

```
fig = px.bar(fuel_long, x='Year', y='consumption',
   pattern_shape = 'metric', barmode='group',
   pattern_shape_sequence \
       = ['.', 'x', '+', '|', '-', '/'],
   title = 'Electric Vehicle Range {2012 to 2024)',
   text_auto=True,
   template="simple_white",
   labels={'consumption': 'Mean Cons\n(l/100km equiv)',
           'metric': ''})

fig.update_yaxes(tickformat=',.2r')
fig.update_traces(
      marker=dict(color='black', line_color='black',
                  pattern_fillmode='replace'))
```

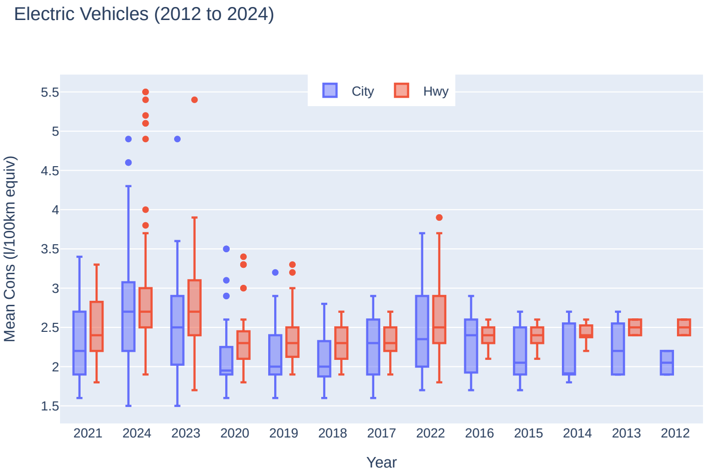
Column Chart (with Patterns)



Box Plot

```
fuel_long = pd.melt(fuel,
    id_vars=['Year'], value_vars=['City', 'Hwy'],
    var_name='metric', value_name='consumption')

fig = px.box(fuel_long,
         x=fuel_long['Year'].astype(str),
         y='consumption', color='metric',
         labels={'consumption': 'Mean Cons\n(l/100km)',
                 'metric': ''},
         title='Electric Vehicles (2012 to 2024)')

fig.update_layout(
         xaxis_title='Year',
         yaxis_title='Mean Cons\n(l/100km equiv)',
         legend_title_text='',
         legend=dict(orientation="h",
                     yanchor="top", y=1,
                     xanchor="center", x=0.5))
```

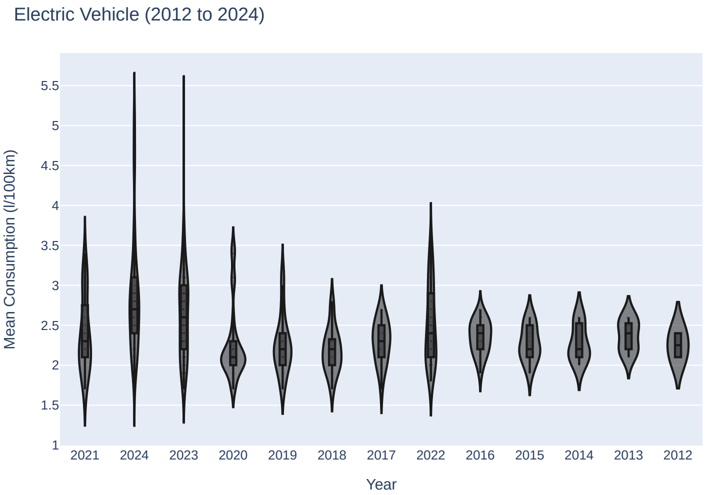
Box Plot


Electric Vehicles (2012 to 2024)
City
Hwy
Mean Cons (l/100km equiv)
5.5
5
4.5
4
3.5
3
2.5
2
1.5
2021
2024
2023
2020
2019
2018
2017
2022
2016
2015
2014
2013
2012
Year

Violin Plot

```
fig = px.violin(fuel,
        x=fuel['Year'].astype(str),
        y='Comb', box=True,
        points='all')

fig.update_traces(jitter=0.15, pointpos=0,
        marker=dict(color='black', size=1, opacity=0.5))

fig.update_layout(xaxis_title='Year',
        yaxis_title='Mean Consumption\n(l/100km)',
        title='Electric Vehicle (2012 to 2024)',
        legend_title_text='')
```

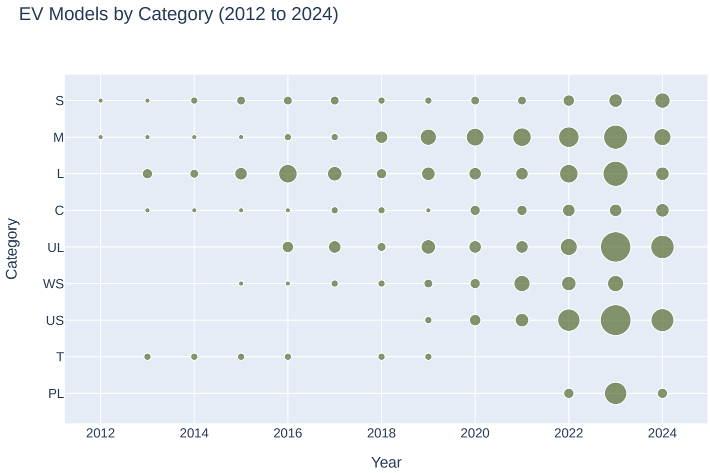
Violin Plot


Electric Vehicle (2012 to 2024)
Mean Consumption (l/100km)
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
2021
2024
2023
2020
2019
2018
2017
2022
2016
2015
2014
2013
2012
Year

Count Plot

```
count_df = fuel.groupby(['Year', 'Category']) \
   .size().reset_index(name='counts')

fig = px.scatter(count_df,
   x='Year', y='Category', size='counts',
   color_discrete_sequence=['darkolivegreen'],
   labels={'Category': '',
           'Year': 'Year',
           'counts': 'Count'},
   title='EV Models by Category (2012 to 2024)')
```

Count Plot

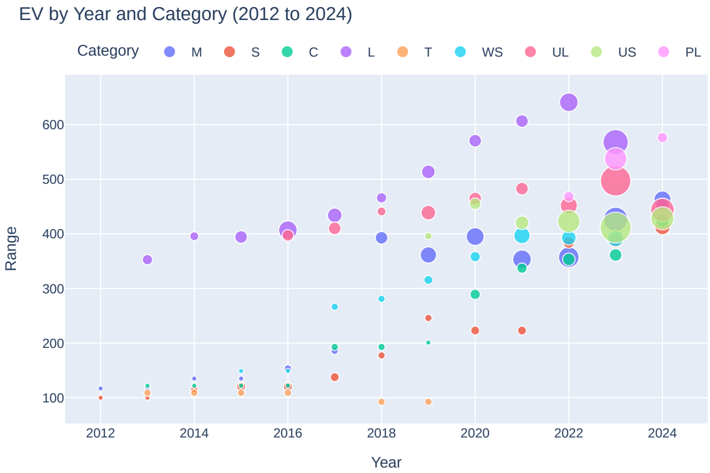


EV Models by Category (2012 to 2024)
Category
S
M
L
C
UL
WS
US
T
PL
Year
2012
2014
2016
2018
2020
2022
2024

Points Plot

```
grouped_fuel = fuel.groupby(['Year', 'Category']).agg(
    totalcount=pd.NamedAgg('Range', 'size'),
    meanRange=pd.NamedAgg('Range', 'mean')
).reset_index()

fig = px.scatter(grouped_fuel,
    x='Year', y='meanRange', size='totalcount',
    color='Category', hover_name='Category',
    labels={'meanRange': 'Range',
            'totalcount': 'Number of Models'},
    title='EV by Year and Category (2012 to 2024)',
    size_max=20, opacity=0.8)

fig.update_layout(
    xaxis_title='Year',
    yaxis_title='Range',
    legend_title_text='Category',
    legend=dict(orientation="h", yanchor="bottom",
                y=1.02, xanchor="right", x=1))
```

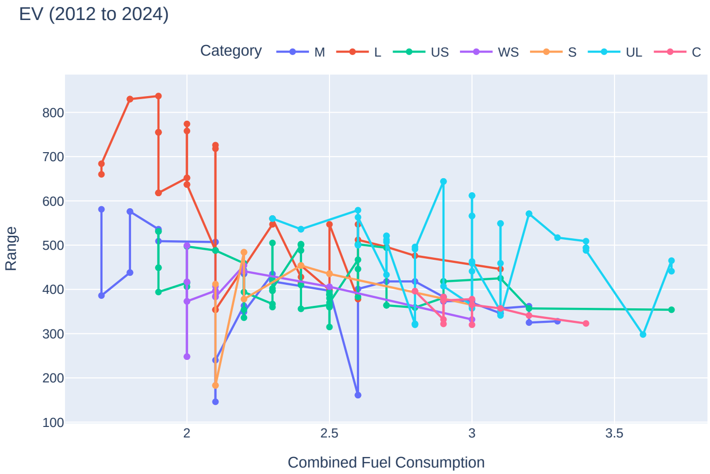
Points Plot


EV by Year and Category (2012 to 2024)
Category
M
S
C
L
T
WS
UL
US
PL
Range
600
500
400
300
200
100
2012
2014
2016
2018
2020
2022
2024
Year

Lines and Points Plot

```
filtered_fuel = \
    fuel[(fuel['Year'] >= 2022) &
         (fuel['Year'] <= 2023)]
filtered_fuel = \
    filtered_fuel[filtered_fuel['Comb'] <= 4]
filtered_fuel = \
    filtered_fuel[~filtered_fuel['Category'].isin(['PL', 'T'])]

fig = px.line(filtered_fuel,
    x='Comb', y='Range', color='Category',
    line_group='Category', markers=True,
    labels={'Range': 'Range', 'Comb': 'Combined Fuel Consumption'},
    title='EV (2012 to 2024)')

fig.update_layout(
    xaxis_title='Combined Fuel Consumption',
    yaxis_title='Range',
    legend_title_text='Category',
    legend=dict(orientation="h", yanchor="bottom",
                y=1.02, xanchor="right", x=1))
```

Lines and Points Plot



Pie Chart

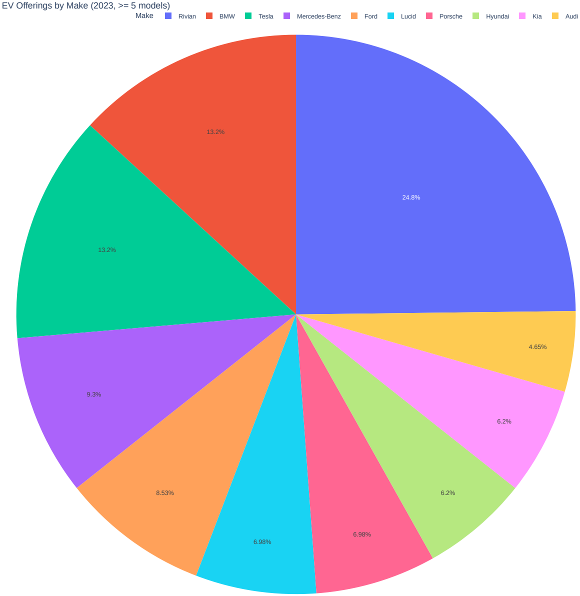
```
fuel_2023 = \
    fuel[fuel['Year'] == 2023]
fuel_grouped = \
    fuel_2023.groupby('Make').size() \
    .reset_index(name='totalcount')
fuel_grouped = \
    fuel_grouped[fuel_grouped['totalcount'] >= 5]

fig = px.pie(fuel_grouped,
    names='Make', values='totalcount', hole=0,
    title='EV Offerings by Make (2023, >= 5 models)',
    labels={'totalcount': 'Number of Models'})
```

Pie Chart [cont'd]

Continued from previous slide . . .

```python
for i, row in fuel_grouped.iterrows():
    fig.add_annotation(text=str(row['totalcount']),
            x=row['Make'], y=row['totalcount'],
            showarrow=False, font_color='lightgrey')

fig.update_layout(legend=dict(orientation="h", yanchor="bottom",
    y=1.02, xanchor="right", x=1),
    showlegend=True, legend_title_text='Make')
```

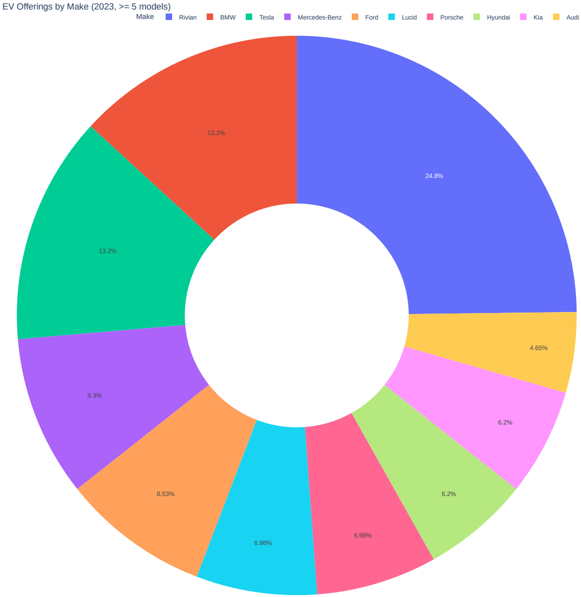
Pie Chart


EV Offerings by Make (2023, >= 5 models)
Make
Rivian
BMW
Tesla
Mercedes-Benz
Ford
Lucid
Porsche
Hyundai
Kia
Audi
24.8%
13.2%
13.2%
9.3%
8.53%
6.98%
6.98%
6.2%
6.2%
4.65%

Donut Chart

```
fig = px.pie(fuel_grouped,
    names='Make', values='totalcount', hole=0.4,
    title='EV Offerings by Make (2023, >= 5 models)',
    labels={'totalcount': 'Number of Models'})
```

Pie Chart


EV Offerings by Make (2023, >= 5 models)
Make
Rivian
BMW
Tesla
Mercedes-Benz
Ford
Lucid
Porsche
Hyundai
Kia
Audi
24.8%
13.2%
13.2%
9.3%
8.53%
6.98%
6.98%
6.2%
6.2%
4.65%

Radar Plot

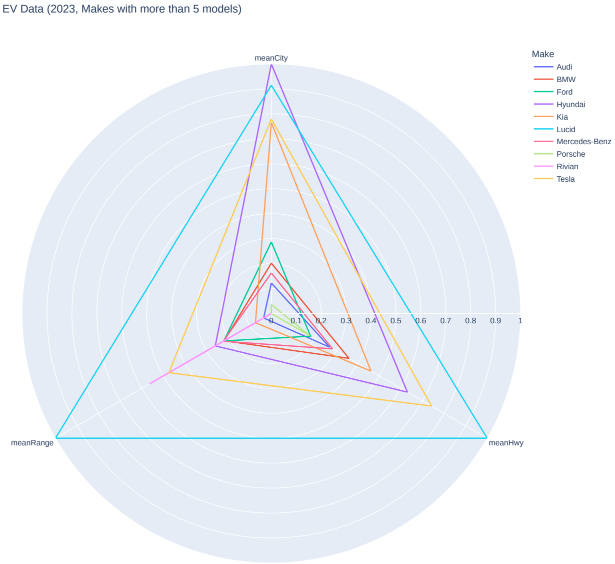
```python
from sklearn.preprocessing import MinMaxScaler

fuel_2023 = fuel[fuel['Year'] == 2023]
grouped = fuel_2023.groupby('Make').agg(
    meanCity=pd.NamedAgg('City',lambda x: 1/x.mean()),
    meanHwy=pd.NamedAgg('Hwy',lambda x: 1/x.mean()),
    meanRange=pd.NamedAgg('Range',lambda x: x.mean()/100),
    nModels=pd.NamedAgg('Make','size')
)
grouped = grouped[grouped['nModels'] >= 5]

grouped[['meanCity', 'meanHwy', 'meanRange']] = \
   MinMaxScaler().fit_transform(
      grouped[['meanCity', 'meanHwy', 'meanRange']])

melted = grouped.reset_index().melt(
    id_vars='Make',
     value_vars=['meanCity', 'meanHwy', 'meanRange'])
```

Radar Plot [cont'd]

Continued from previous slide . . .

```
fig = px.line_polar(melted,
      r='value',
      theta='variable',
      color='Make',
      line_close=True,
      labels={'variable': '',
              'value': '',
              'Make': 'Make'},
      title='EV Data (Makes with more than 5 models)')
```

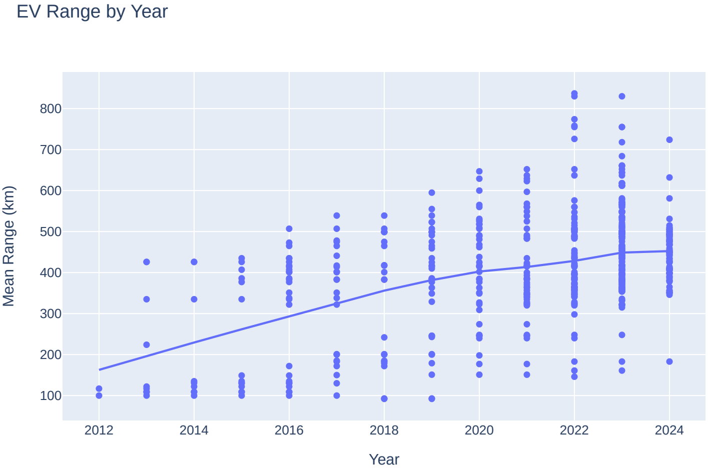
Radar Plot


EV Data (2023, Makes with more than 5 models)
meanCity
meanHwy
meanRange
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Make
Audi
BMW
Ford
Hyundai
Kia
Lucid
Mercedes-Benz
Porsche
Rivian
Tesla

Local Regression Smoothing Plot

```python
fig = px.scatter(fuel,
    x='Year', y='Range', trendline='lowess',
    labels={'Range': 'Mean Range (km)'},
    title='EV Range by Year')

fig.update_layout(xaxis_title='Year',
                  yaxis_title='Mean Range (km)')
```

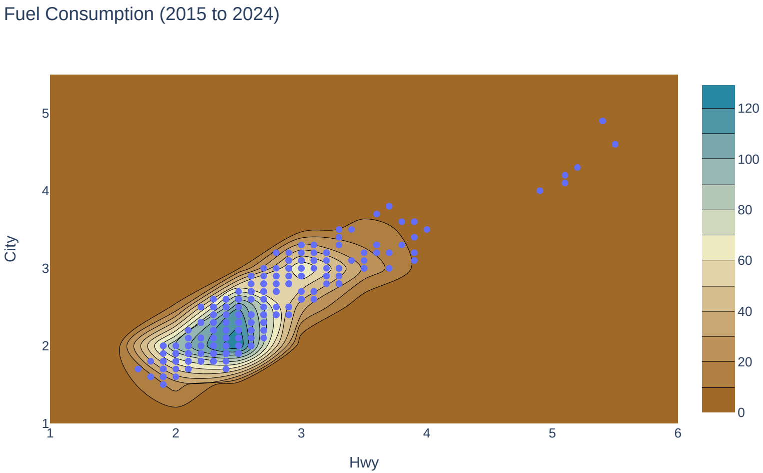
Local Regression Smoothing Plot



2D Density Plot

```python
import plotly.graph_objects as go

fig = px.scatter(fuel,
    x='Hwy', y='City',
    title='Fuel Consumption (2012 to 2024)')

fig.add_trace(go.Histogram2dContour(
    x=fuel['Hwy'], y=fuel['City'],
    colorscale='Earth'))

fig.update_layout(legend=dict(orientation="h", yanchor="bottom",
                  y=1.02, xanchor="right", x=1),
                  showlegend=False, plot_bgcolor='white')
```

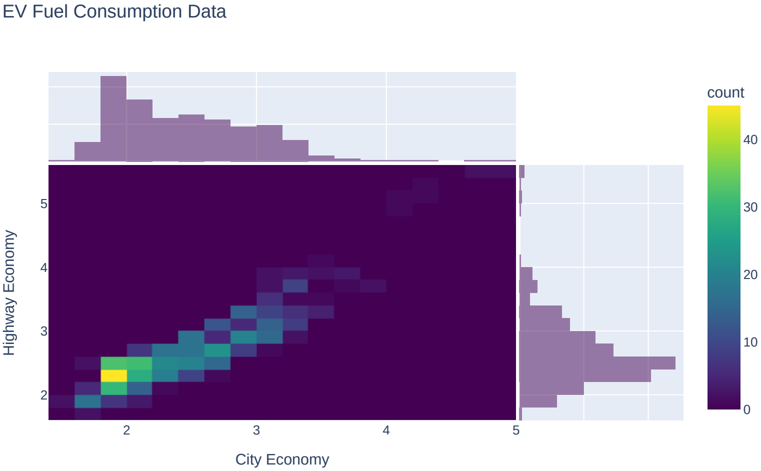
2D Density Plot



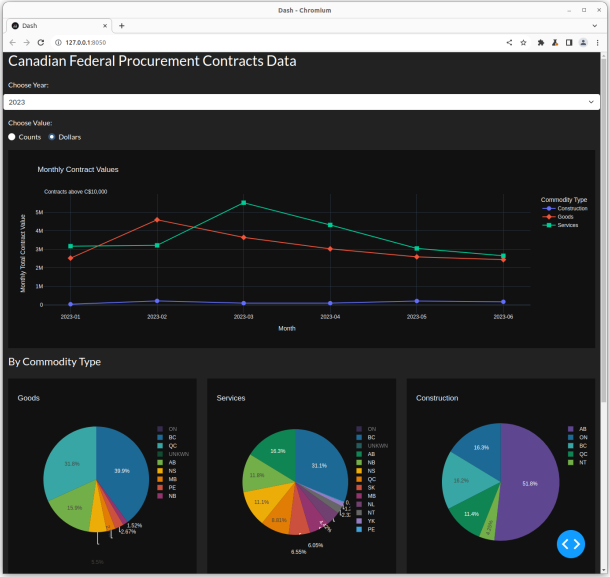
Heatmap with Marginals

```
fig = px.density_heatmap(fuel,
    x = 'City', y = 'Hwy',
    nbinsx=20, nbinsy=20,
    color_continuous_scale=px.colors.sequential.Viridis,
    marginal_x="histogram",
    marginal_y="histogram",
    title='EV Fuel Consumption Data',
    labels={"range" : "Range",
           "Hwy": "Highway Economy",
            "City": "City Economy"})
```

Heatmap with Marginals



Dashboards – Live Demo



Canadian Federal Procurement Contracts Data

Choose Year:

2023

Choose Value:

○ Counts ● Dollars

By Commodity Type

Hands-On Exercises

Using the Pagila database data from `https://evermann.ca/busi4720/rentals.csv`, create

1. A histogram and/or density chart of film length by film category
2. A column chart of the mean rental payments for films by film category
   - Add error bars to this chart
3. A scatter plot of total rental payments by year and week
   - Add a local regression line to this plot
4. A pie or donut chart of rental counts by film rating

Tips:

- The Pandas `read_csv()` function can read from a URL
- The data is de-normalized, use the Pandas `drop_duplicates()` function to get accurate film counts for exercise 1
- Use `.dt.strftime('%Y-%W')` to extract the year and week from a datetime column in Pandas