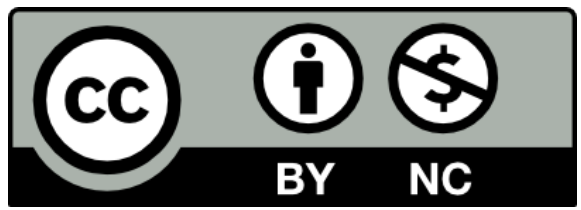
Business 4720 - Class 8

Data Visualization with Python

Joerg Evermann

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



MEMORIAL UNIVERSITY

This Class

What You Will Learn:

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

MEMORIAL UNIVERSITY

Example Dataset

- Government of Canada, Open Government Portal
- Fuel Consumption Ratings – Battery-electric vehicles – 2012–2023; last updated Oct 10, 2023
- https://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64

Column	Data Type
Make	Discrete
Model	Discrete
Year	Numeric
Category	Discrete
City	Numeric[1]
Hwy	Numeric
Comb	Numeric
Range	Numeric[2]

[1] Fuel consumption in l/100km equivalent

[2] Range in km

MEMORIAL UNIVERSITY

Data Preparation

Import required packages:

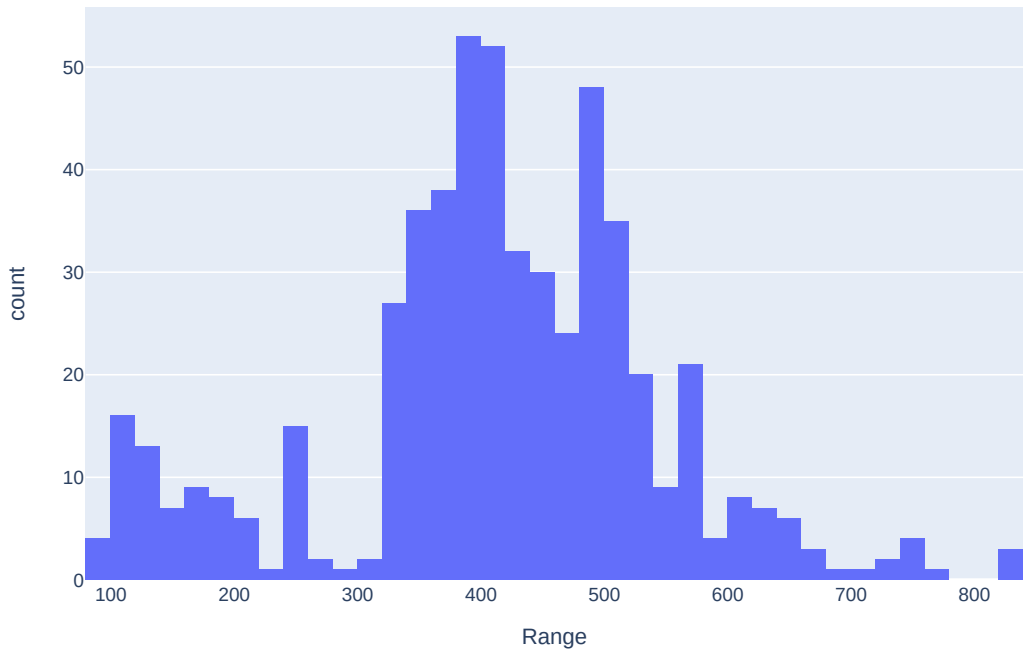
```
import pandas as pd
import plotly.express as px
import plotly.io as pio
pio.kaleido.scope.mathjax = None
```

```
# Read data
data = pd.read_csv('https://evermann.ca/busi4720/fuel.csv')
```

MEMORIAL UNIVERSITY

Histogram

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

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

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



MEMORIAL UNIVERSITY

Column Chart

Prepare data in "long" format using `pd.melt()`:

```
data_grouped = \
data.groupby('Year') \
  .agg(
     meanCity=('City','mean'),
     meanHwy=('Hwy','mean')) \
  .reset_index()

data_long = \
  pd.melt(data_grouped,
      id_vars=['Year'],
      value_vars=['meanCity',
                  'meanHwy'],
      var_name='metric',
      value_name='consumption')
```

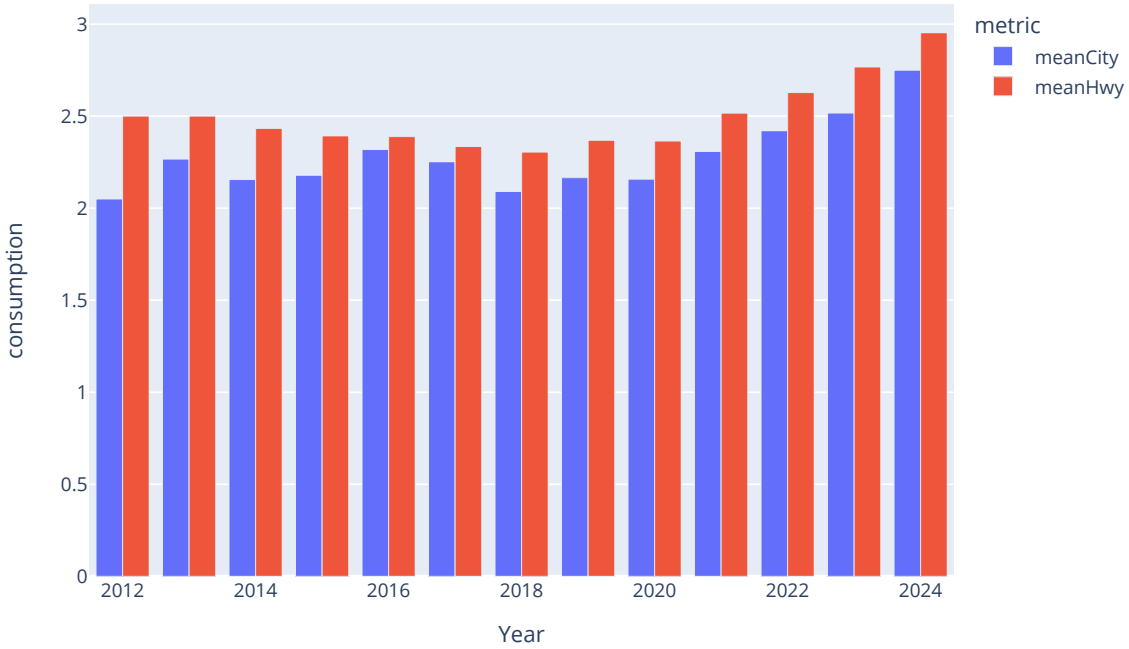
```
>>> data_grouped
    Year  meanCity   meanHwy
0   2012  2.050000  2.500000
1   2013  2.266667  2.500000
2   2014  2.155556  2.433333
3   2015  2.178571  2.392857
4   2016  2.318519  2.388889
```

```
>>> data_long
    Year    metric  consumption
0   2012  meanCity     2.050000
1   2013  meanCity     2.266667
2   2014  meanCity     2.155556
3   2015  meanCity     2.178571
4   2016  meanCity     2.318519
```

MEMORIAL UNIVERSITY

Column Chart [cont'd]

```
fig = px.bar(data_long,
   x='Year', y='consumption', color='metric',
   barmode='group')
```

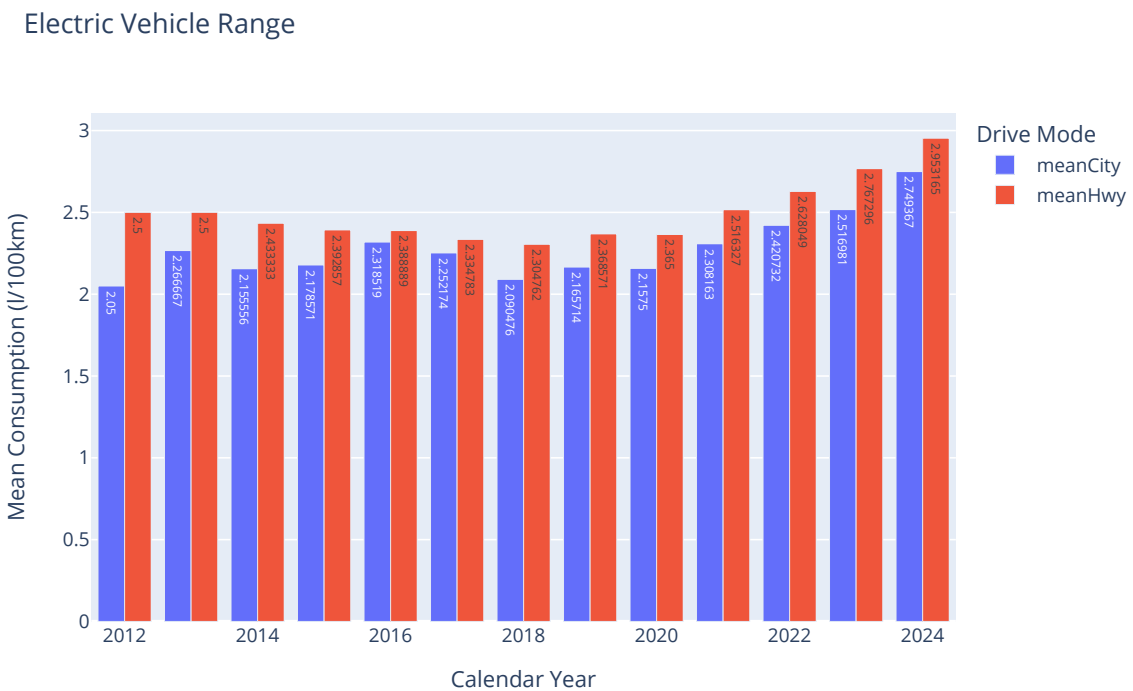


- The `barmode='group'` places bars of different categories next to each other (instead of stacking them)

MEMORIAL UNIVERSITY

Labelling

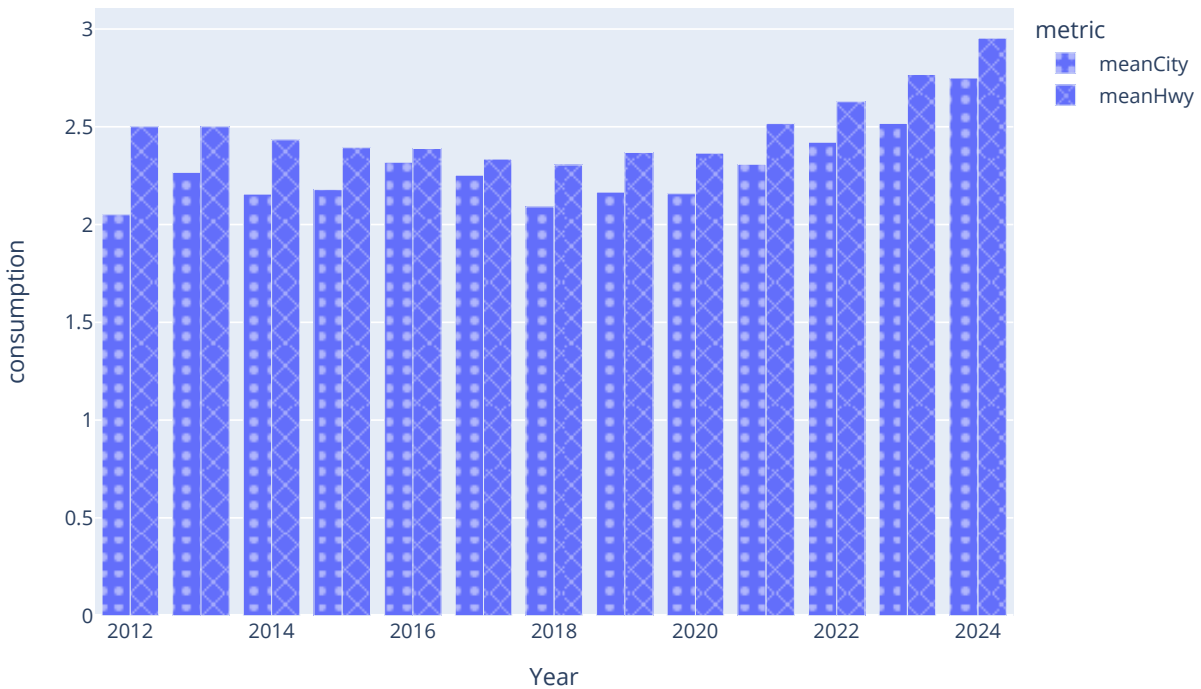
```
fig = px.bar(data_long,
  x='Year', y='consumption', color='metric', barmode='group',
  text_auto=True,
  title = 'Electric Vehicle Range',
  labels={'consumption': 'Mean Consumption (l/100km)',
          'Year': 'Calendar Year',
          'metric': 'Drive Mode'})
```



- No subtitles or captions
- Labels for each plot element
- Values with `text_auto=True` option

Column Chart (with Patterns)

```
fig = px.bar(data_long,
   x='Year', y='consumption',
   pattern_shape = 'metric',
   pattern_shape_sequence = ['.', 'x', '+', '|', '-', '/'],
   barmode='group')
```



- Note the different pattern shapes

Hands-On Exercises

1. Read the EV fuel efficiency data set into a Pandas data frame.
2. Create a histogram of highway fuel efficiency with 25 bins.
3. Add labels for the axes, and add a title.

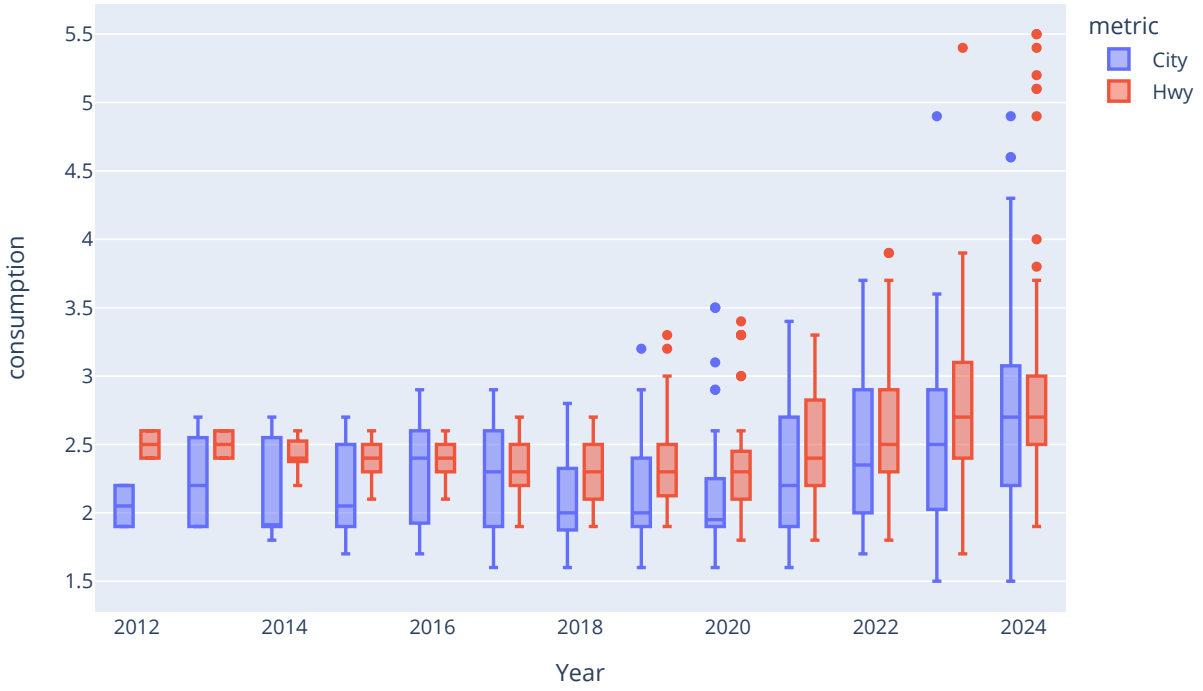
Tips

- Use the `pd.read_csv()` function from Pandas
- The column name is `Hwy`
- Use the `px.histogram()` function from Plotly Express
- Use the `title=...` option
- Use the `labels='...'` option

MEMORIAL UNIVERSITY

Box Plot

```
data_long = pd.melt(data,
    id_vars=['Year'], value_vars=['City', 'Hwy'],
    var_name='metric', value_name='consumption')
fig = px.box(data_long,
             x='Year', y='consumption', color='metric')
```

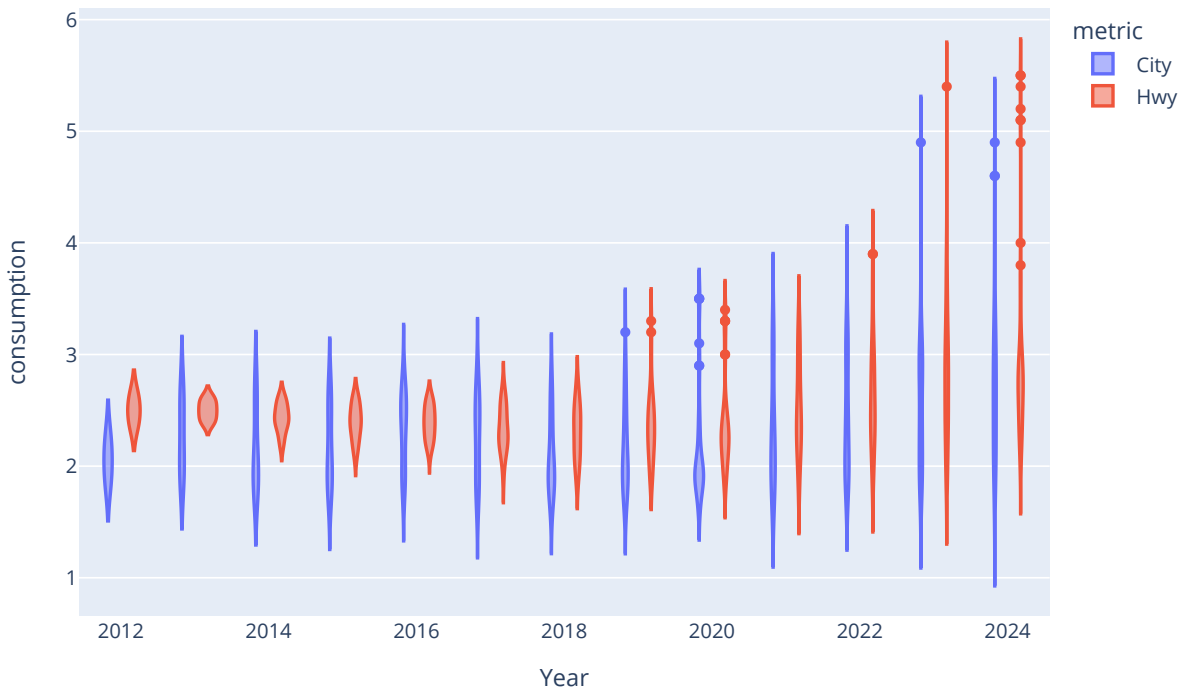


- Shows distribution
- Median
- 1st quartile Q_1
- 3rd quartile Q_3
- ”Inter-quartile range”
- $IQR = Q_3 - Q_1$
- ”Whiskers”
- $Q_3 + 1.5 \times IQR$
- $Q_1 - 1.5 \times IQR$
- ”Outliers”

MEMORIAL UNIVERSITY

Violin Plot

```
data_long = pd.melt(data,
    id_vars=['Year'], value_vars=['City', 'Hwy'],
    var_name='metric', value_name='consumption')
fig = px.violin(data_long,
                x='Year', y='consumption', color='metric')
```

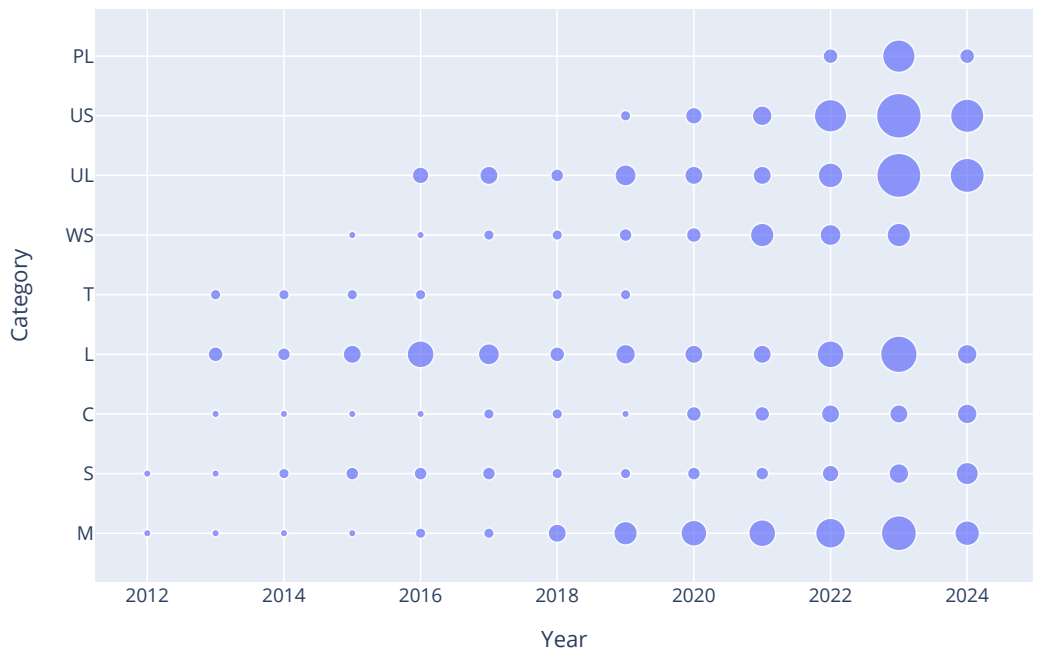


- Shows detailed density
- But no summary statistics

MEMORIAL UNIVERSITY

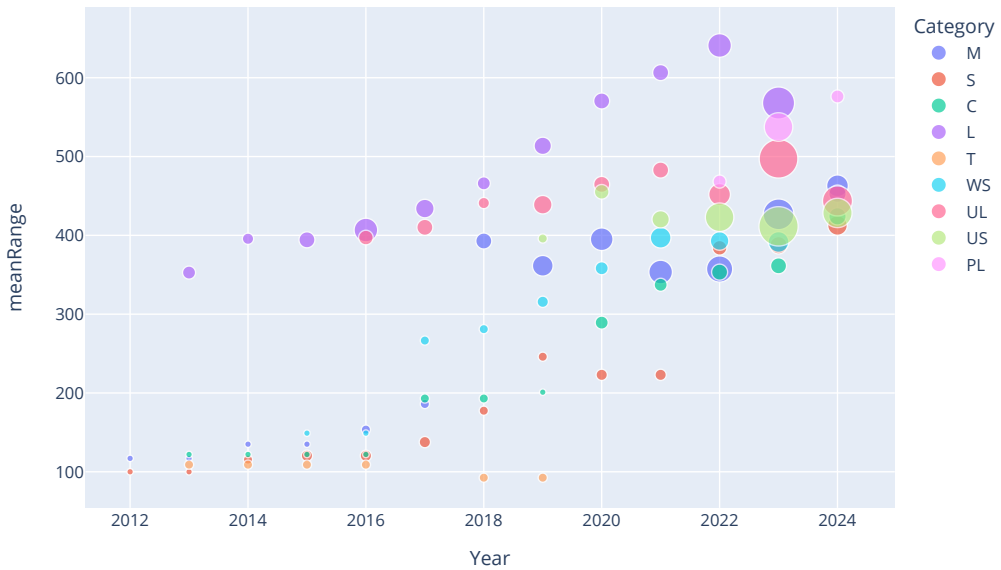
Count Plot

```
count_data = data \
    .groupby(['Year', 'Category']) \
    .agg(counts = ('Range', 'size')) \
    .reset_index()

fig = px.scatter(count_data, x='Year', y='Category', size='counts')
```



Points Plot

```
grouped_data = data \
    .groupby(['Year', 'Category']) \
    .agg(totalcount=('Range', 'size'),
         meanRange=('Range', 'mean')) \
    .reset_index()
fig = px.scatter(grouped_data,
            x='Year', y='meanRange',
            size='totalcount', color='Category')
```

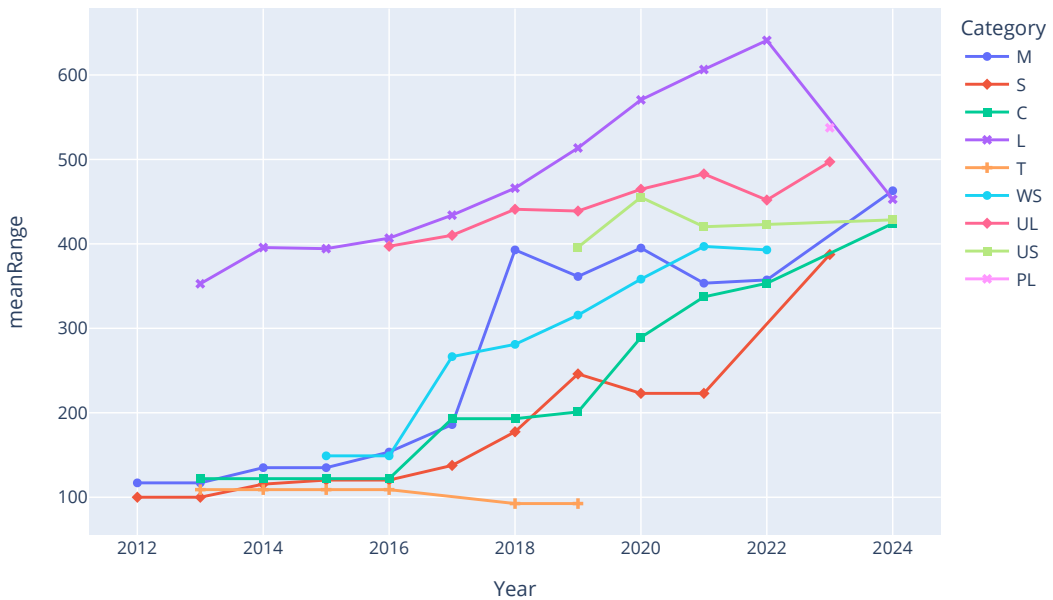


- Shows 4 variables

MEMORIAL UNIVERSITY

Lines and Points Plot

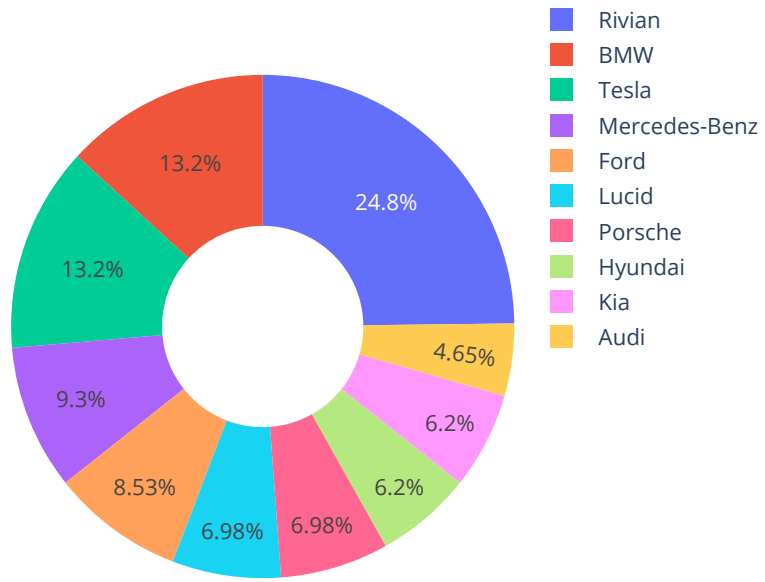
```
filtered_data =
    data.groupby(['Year', 'Category']) \
        .agg(meanRange=('Range', 'mean')) \
        .reset_index() \
        [data['Category'].isin(['C','L','M','S','US','UL'])]
fig = px.line(filtered_data,
    x='Year', y='meanRange', color='Category',
    symbol='Category', markers=True)
```



- Category mapped to two plot elements

Pie/Donut Chart

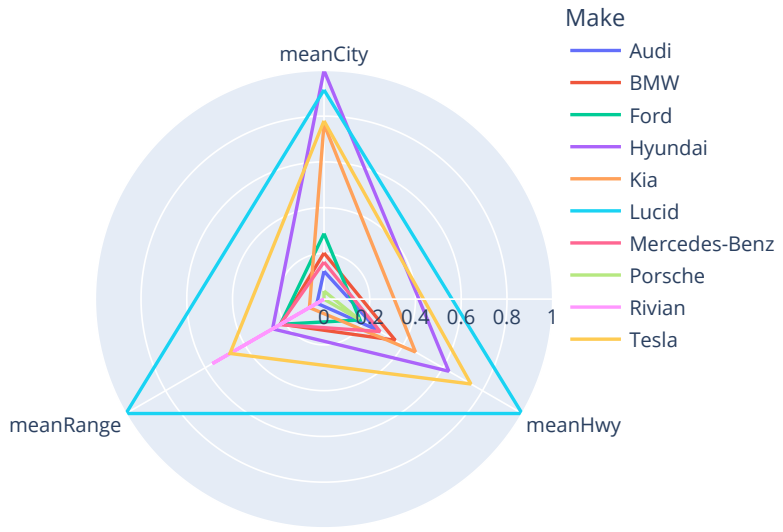
```
data_pie = \
  data[data['Year'] == 2023] \
    .groupby('Make') \
    .agg(totalcount=('Model', 'size')) \
    .reset_index() \
    .query('totalcount >= 5')

fig = px.pie(data_pie,
    names='Make',
    values='totalcount',
    hole=0.4)
```



MEMORIAL UNIVERSITY

Radar Plot

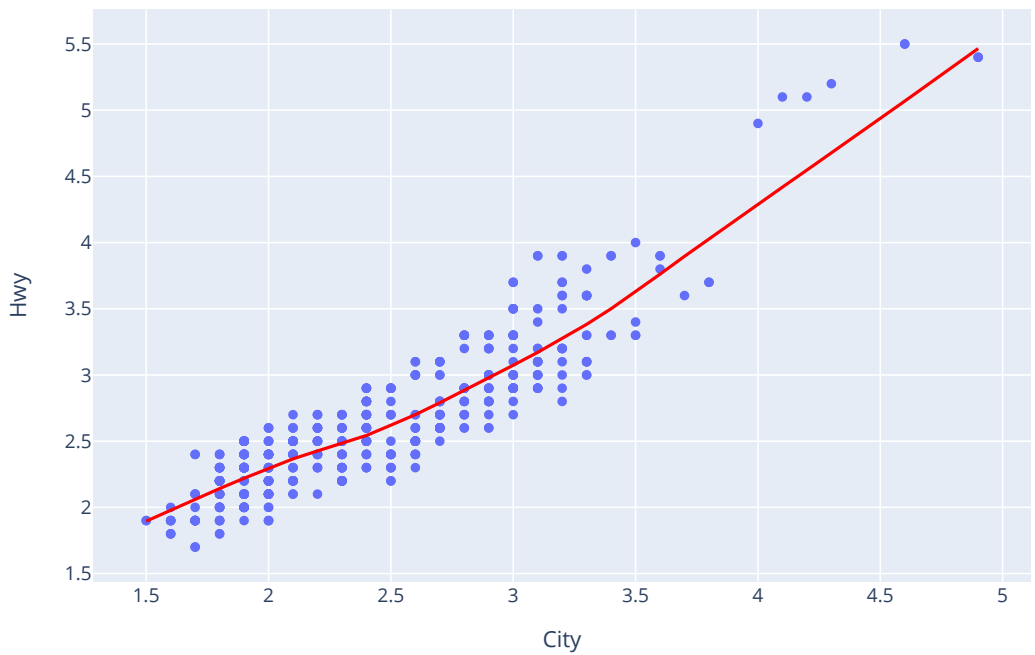
```
from sklearn.preprocessing \
  import MinMaxScaler
grouped = data \
  .query('Year == 2023') \
  .groupby('Make') \
  .agg(
    Cty=('City',lambda x: 1/x.mean()),
    Hwy=('Hwy',lambda x: 1/x.mean()),
    Rng=('Range',lambda x: x.mean()/100),
    nModels=('Make','size')) \
  .query('nModels >= 5')
grouped[['Cty', 'Hwy', 'Rng']] = \
  MinMaxScaler().fit_transform(
    grouped[['Cty', 'Hwy', 'Rng']])
melted = grouped \
  .reset_index() \
  .melt(id_vars='Make',
    value_vars=['Cty', 'Hwy', 'Rng'])
fig = px.line_polar(melted,
     r='value', theta='variable',
     color='Make',
     line_close=True)
```



MEMORIAL UNIVERSITY

Local Regression Smoothing Plot

```
fig = px.scatter(data,
    x='City', y='Hwy',
    trendline='lowess',
    trendline_color_override='red')
```

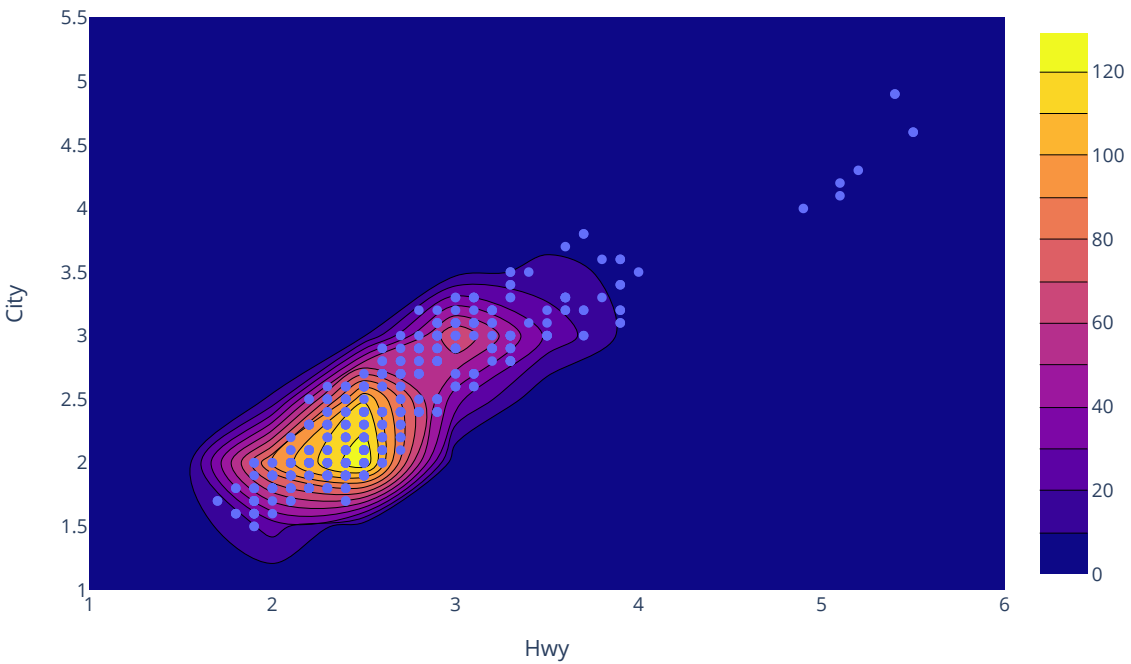


- Local regression line

MEMORIAL UNIVERSITY

2D Histogram Plot

```
import plotly.graph_objects as go

fig = px.scatter(data, x='Hwy', y='City')

fig.add_trace(
  go.Histogram2dContour(x=data['Hwy'], y=data['City']))
```

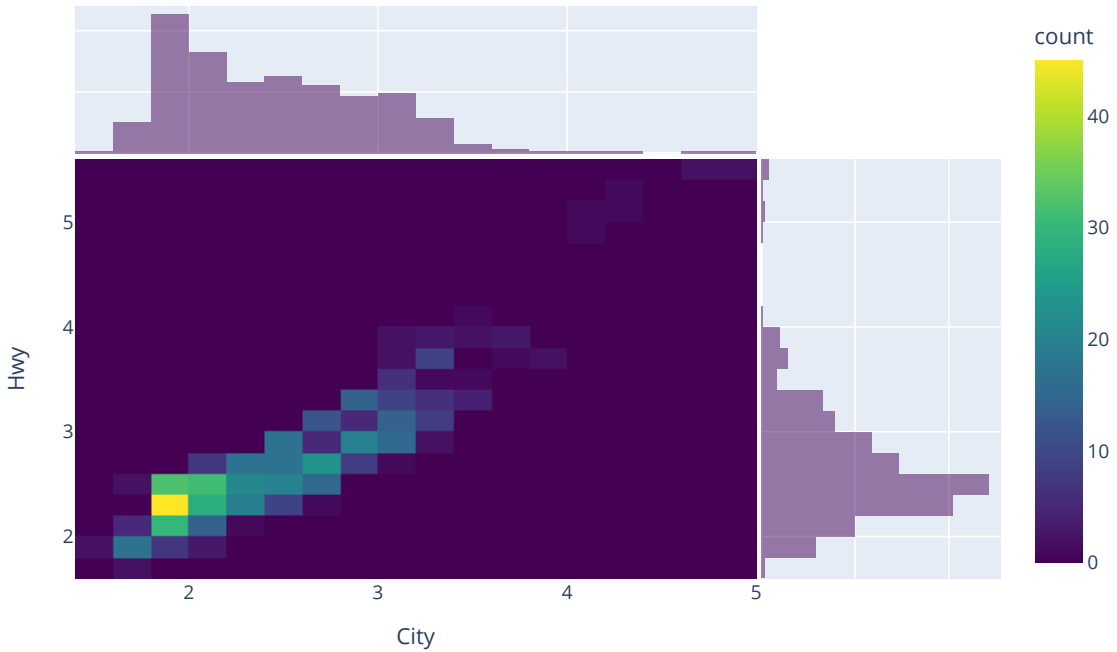


- Counts of observations/data points

MEMORIAL UNIVERSITY

Density Heatmap with Marginals

```
fig = px.density_heatmap(data,
  x = 'City', y = 'Hwy',
  nbinsx=20, nbinsy=20,
  marginal_x='histogram', marginal_y='histogram',
  color_continuous_scale = px.colors.sequential.Viridis)
```

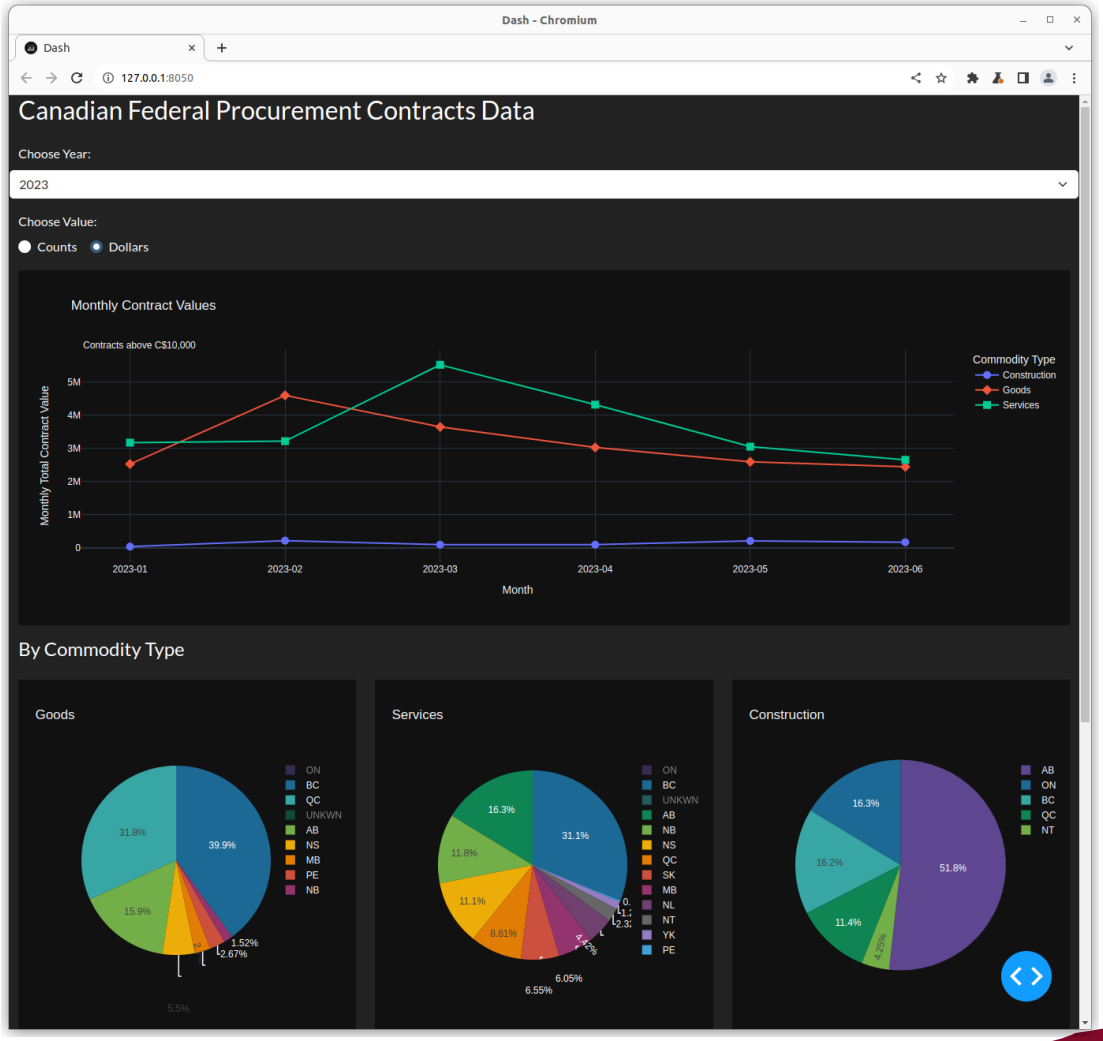


- Marginals can be applied to other plot types
- Other options for the marginal plots are `rug`, `box`, and `violin`.
- Custom colour palette/scale

MEMORIAL UNIVERSITY

Dashboards

- Set of related plots
- Interactive and customizable
- Automatically updated
- Simple and easy to understand
- Provides quick overview of key metrics



Step 1 – Import Packages

```
from dash import Dash, html, dcc, callback, Output, Input
import dash_bootstrap_components as dbc

import plotly.express as px
```

MEMORIAL UNIVERSITY

Step 2 – Visual Layout

- Container element with multiple Row elements
- Row elements contain Column elements, HTML elements, Label elements, Selection elements, Radio button elements, etc.

- First row is HTML heading
- Second row is Label element and Selection element, followed by another Label element and Radio button element

MEMORIAL UNIVERSITY

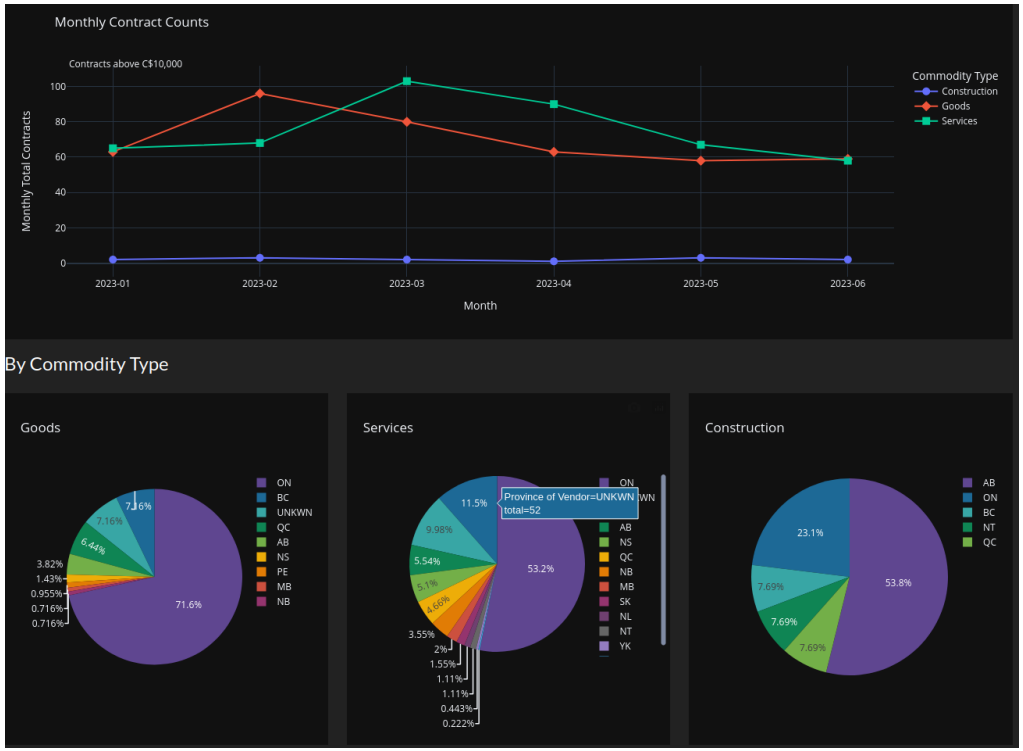
Step 2 – Visual Layout [cont'd]

```
app.layout = dbc.Container([
  dbc.Row([
    html.H3('Canadian Federal Procurement Contracts Data')]),
  dbc.Row([
    html.P(),
    dbc.Label('Choose Year: '),
    dbc.Select(
      id='year-selection',
      options=[{"label": x, "value": x} for x in years],
      value=years[-1]),
    html.P(),
    dbc.Label('Choose Value: '),
    dbc.RadioItems(
      options=[{"label": "Counts", "value": 0},
               {"label": "Dollars", "value": 1}],
      value=0, inline = True,
      id = 'value-selection'),
    html.P() ]),
```

- Every component has an ID!

MEMORIAL UNIVERSITY

Step 2 – Visual Layout [cont'd]



- First row contains single column with a Graph element that contains a figure (line chart)
- Second row is an HTML heading
- Third row contains three columns, each with one graph element that contain a figure (pie chart)

Step 2 – Visual Layout [cont'd]

```
# Continued from previous snippet
    dbc.Row([
        dbc.Col([
            dcc.Graph(figure={}, id='line')
        ], width=12),
    ]),
    dbc.Row([
        html.P(),
        html.H4('By Commodity Type'),
        html.P()
    ]),
    dbc.Row([
        dbc.Col([
            dcc.Graph(figure={}, id='pie1')
        ], width=4),
        dbc.Col([
            dcc.Graph(figure={}, id='pie2')
        ], width=4),
        dbc.Col([
            dcc.Graph(figure={}, id='pie3')
        ], width=4),
    ]),
```

Step 3 – Create the Figures

```
@callback(
    Output(component_id='line', component_property='figure'),
    [
        Input(component_id='year-selection',
              component_property='value'),
        Input(component_id='value-selection',
              component_property='value')
    ]
)
def update_line(year_chosen, value_chosen):
    # Figure creation code, for example
    # fig = px.histogram(...)
    return fig
```

- Each figure has an update function that creates and returns the figure
- Each update function has a callback specification
  - The `Output` specifies the Graph element for this figure
  - The `Input` elements specify which interactive elements provide values for creating or updating the figure

MEMORIAL UNIVERSITY

Step 4 – Run the App

```
app.run()
```

MEMORIAL UNIVERSITY

Hands-On Exercises

Use the Pagila film rentals data from
`https://evermann.ca/busi4720/rentals.csv`

1. Read the data into a Pandas data frame using `read_csv()`
2. Create a box plot of the rental payment amounts for films by rating
3. Create a violin plot of the rental payment amounts for films by rating

- Compare the information conveyed by a box plot and a violin plot. What are the commonalities and what are the differences?

MEMORIAL UNIVERSITY

Hands-On Exercises

Use the Pagila film rentals data from
`https://evermann.ca/busi4720/rentals.csv`

1. Read the data into a Pandas data frame using `read_csv()`
2. Use `drop_duplicates()` to drop duplicates of the film titles
   - Because each film may have been rented multiple times.
3. Produce a *histogram* of counts of films by rating

Tip:

- Use the `columns` and `shape` properties or the `describe()` method of a data frame to examine it.

MEMORIAL UNIVERSITY

Hands-On Exercises

Use the Pagila film rentals data from
`https://evermann.ca/busi4720/rentals.csv`

1. Read the data into a Pandas data frame using `read_csv()`
2. Create a data frame with the mean rental payments for films by rating
3. Generate a bar chart of the mean rental payments for films by rating
4. Generate a pie or donut chart of rental counts by film rating

Tips:

- Use the `groupby()` function to group the data
- Use the `mean()` and `std()` to find the mean and standard deviation of for a data frame column

MEMORIAL UNIVERSITY