# Data Analytics Course - Lesson 05

**Ths. Vu Duy Khuong**

## Agenda

- ❑I. Introduction to Data Visualization
- ❑II. Types of Chart

## I. Introduction to Data Visualization

**1. Definition**

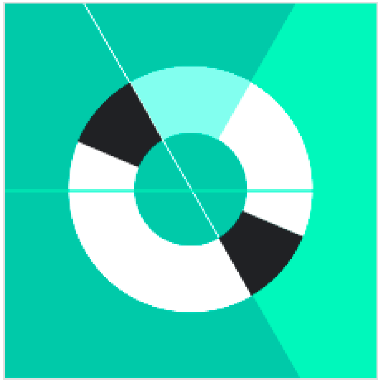
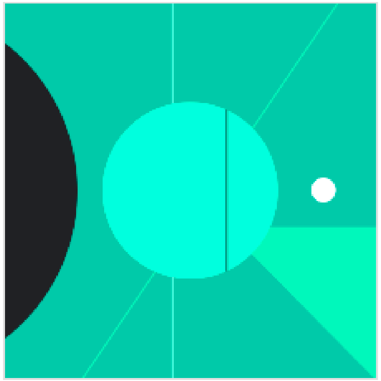
- Data visualization is a form of communication that portrays dense and complex information in graphical form.

- The resulting visuals are designed to make it easy to compare data and use it to tell a story – both of which can help users in decision making.

- Data visualization can express data of varying types and sizes: from a few data points to large multivariate datasets.

# I. Introduction to Data Visualization

## 1. Definition

**Accurate**
Prioritize data accuracy, clarity, and integrity, presenting information in a way that doesn't distort it.

**Helpful**
Help users navigate data with context and affordances that emphasize exploration and comparison.

**Scalable**
Adapt visualizations for different device sizes, while anticipating user needs on data depth, complexity, and modality.

## I. Introduction to Data Visualization

### 2. Classification

- Data visualization can be expressed in different forms. Charts are a common way of expressing data, as they depict different data varieties and allow data comparison.

- The type of chart you use depends primarily on two things: the data you want to communicate, and what you want to convey about that data.

## I. Introduction to Data Visualization

### 2. Classification

| Category | Purpose | Example | Charts |
|---|---|---|---|
| **Change over time** | Show data over a period of time | Stock price performance, Health statistics, Chronologies | Line charts, Bar charts. Stacked bar charts, Candlestick charts, Area charts, Timelines, Horizon charts, Waterfall charts |
| **Category comparison** | Compare data between multiple distinct categories. | Income across different countries, Popular venue times, Team allocations | Bar charts, Grouped bar charts,. Bubble charts, Multi-line charts, Parallel coordinate charts, Bullet charts |
| **Ranking** | Show an item's position in an ordered list. | Election results, Performance statistics | Ordered bar charts, Ordered column charts, Parallel coordinate chart |

## I. Introduction to Data Visualization

### 2. Classification

| Category | Purpose | Example | Charts |
|---|---|---|---|
| **Part-to-whole** | Show how partial elements add up to a total. | Consolidated revenue of product categories, Budgets | Stacked bar charts, Pie charts, Donut charts, Stacked area charts, Treemap charts, Sunburst charts |
| **Correlation** | Show correlation between two or more variables. | Income and life expectancy | Scatterplot charts, Bubble charts, Column and line charts, Heatmap charts |
| **Distribution** | Show how often each values occur in a dataset. | Population distribution, Income distribution | Histogram charts, Box plot charts, Violin charts, Density charts |

# I. Introduction to Data Visualization

## 2. Classification

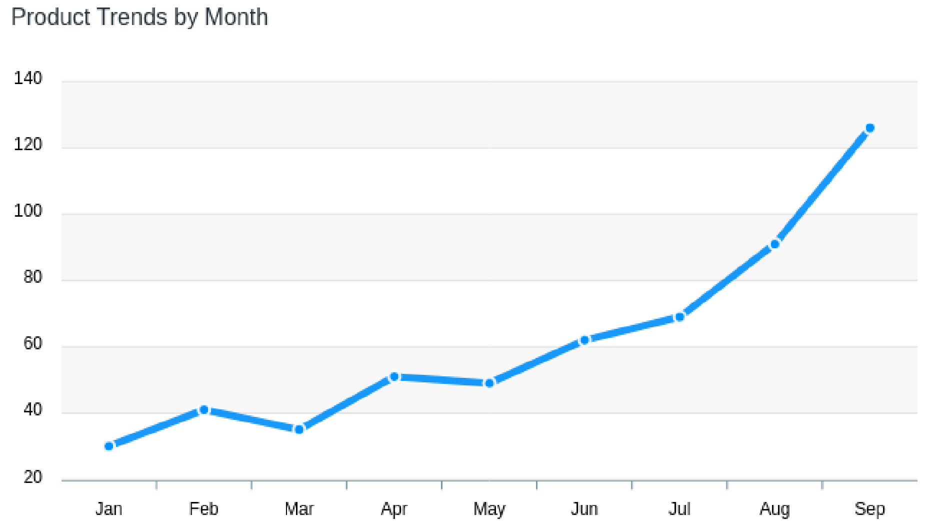
| Category | Purpose | Example | Charts |
|---|---|---|---|
| **Flow** | Show movement of data between multiple states. | Fund transfers, Vote counts and election results | Sankey charts, Gantt charts, Chord charts, Network charts |
| **Relationship** | Show how multiple items relate to one other. | Social networks, Word charts | Network charts, Venn diagrams, Chord charts, Sunburst charts |

# II. Types of Chart

## 1. Line Chart

### a. Definition

- A line chart is a graphical representation of an asset's historical price action that connects a series of data points with a continuous line.
- The horizontal axis is usually a time scale; while the vertical axis would be values corresponding with that time.



# II. Types of Chart

## 1. Line Chart

### b. Line Plots with plotly.express

```
import plotly.express as px

df = px.data.gapminder().query("country=='Canada'")
df.head()
```

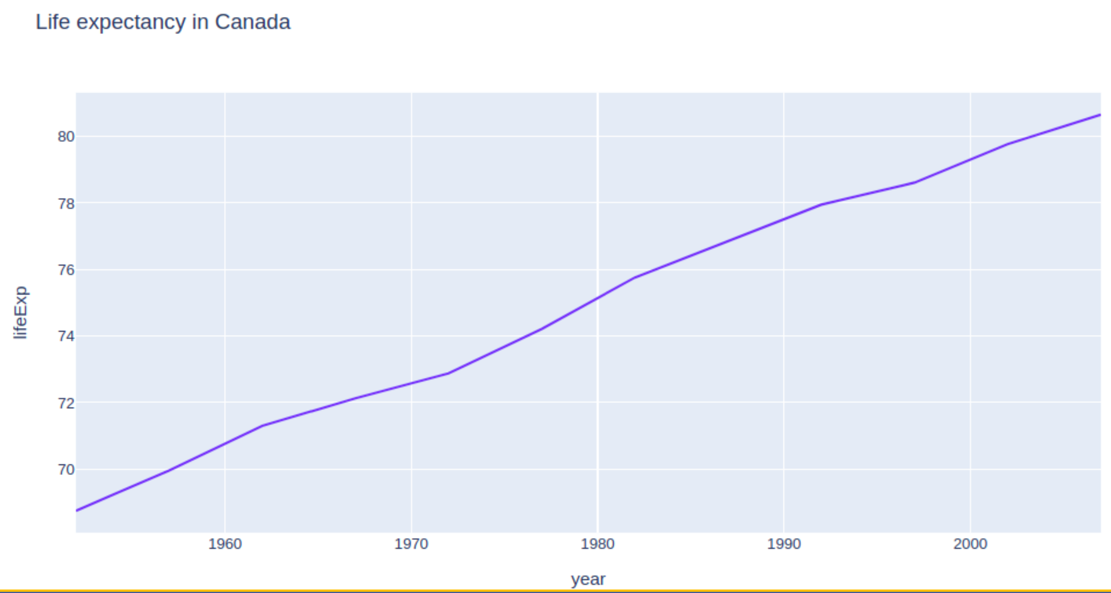
| | country | continent | year | lifeExp | pop | gdpPercap | iso_alpha | iso_num |
|---|---|---|---|---|---|---|---|---|
| **240** | Canada | Americas | 1952 | 68.75 | 14785584 | 11367.16112 | CAN | 124 |
| **241** | Canada | Americas | 1957 | 69.96 | 17010154 | 12489.95006 | CAN | 124 |
| **242** | Canada | Americas | 1962 | 71.30 | 18985849 | 13462.48555 | CAN | 124 |
| **243** | Canada | Americas | 1967 | 72.13 | 20819767 | 16076.58803 | CAN | 124 |
| **244** | Canada | Americas | 1972 | 72.88 | 22284500 | 18970.57086 | CAN | 124 |

# II. Types of Chart

## 1. Line Chart

### c. Line Plots with plotly.express

```
fig = px.line(df, x="year", y="lifeExp", title='Life expectancy in Canada')
fig.show()
```



# II. Types of Chart

## 1. Line Chart

### d. Line Plots with column encoding color

```python
df = px.data.gapminder().query("continent=='Oceania'")
df.head()
```

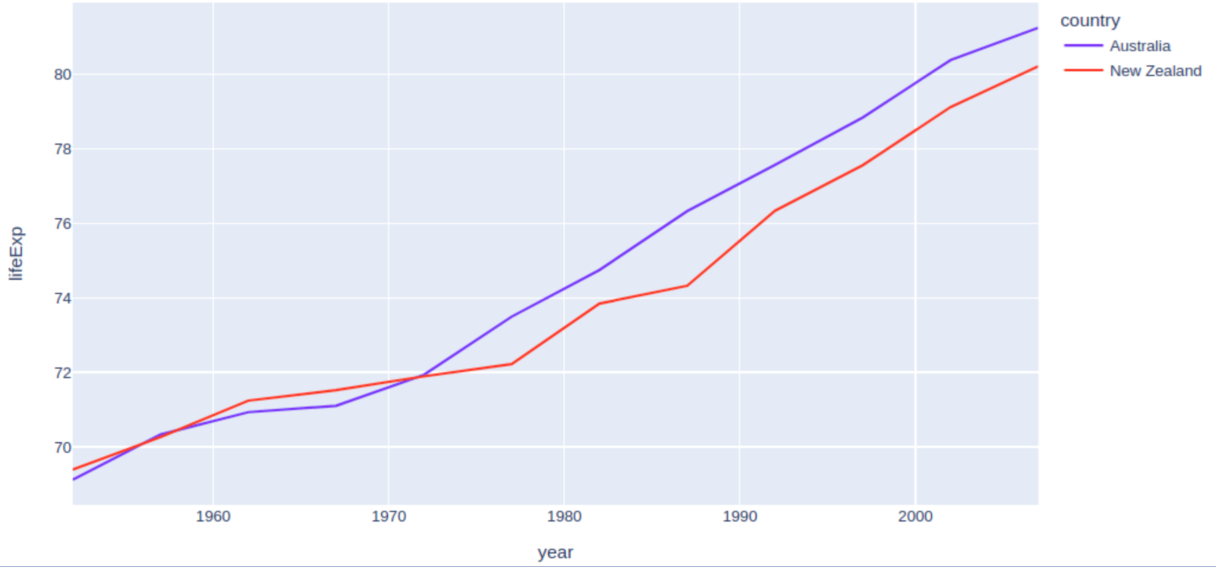
|  | country | continent | year | lifeExp | pop | gdpPercap | iso_alpha | iso_num |
|---|---|---|---|---|---|---|---|---|
| **60** | Australia | Oceania | 1952 | 69.12 | 8691212 | 10039.59564 | AUS | 36 |
| **61** | Australia | Oceania | 1957 | 70.33 | 9712569 | 10949.64959 | AUS | 36 |
| **62** | Australia | Oceania | 1962 | 70.93 | 10794968 | 12217.22686 | AUS | 36 |
| **63** | Australia | Oceania | 1967 | 71.10 | 11872264 | 14526.12465 | AUS | 36 |
| **64** | Australia | Oceania | 1972 | 71.93 | 13177000 | 16788.62948 | AUS | 36 |

# II. Types of Chart

## 1. Line Chart

### e. Line Plots with column encoding color

```
fig = px.line(df, x="year", y="lifeExp", color='country')
fig.show()
```

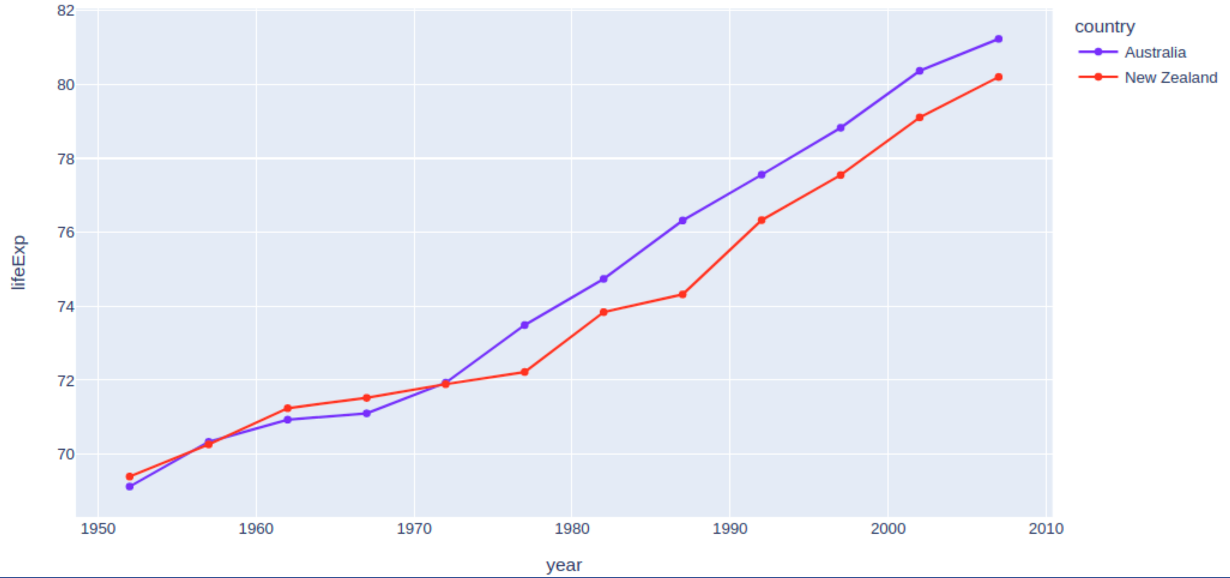


# II. Types of Chart

## 1. Line Chart

### f. Line charts with markers

```
fig = px.line(df, x='year', y='lifeExp', color='country', markers=True)
fig.show()
```

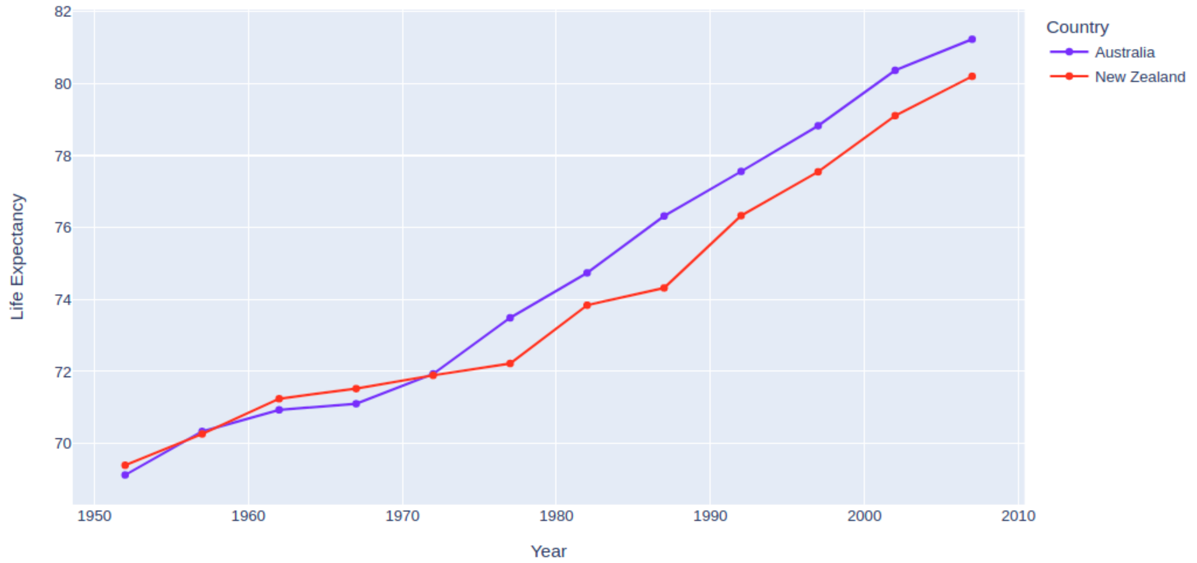


# II. Types of Chart

## 1. Line Chart

### h. Line charts with markers

```
fig = px.line(df, x='year', y='lifeExp', color='country', markers=True,
              labels={'lifeExp':'Life Expectancy', 'year':'Year', 'country':'Country'})
fig.show()
```

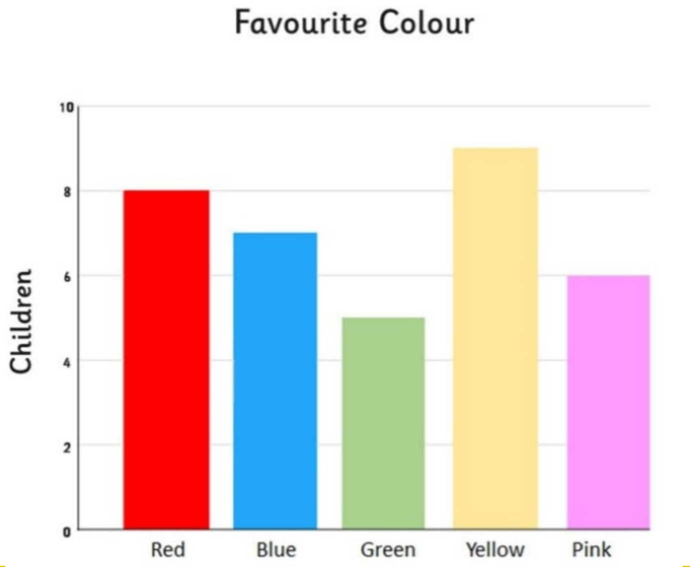


## II. Types of Chart

### 2. Bar Chart

#### a. Definition

- A bar graph is a chart that plots data using rectangular bars or columns (called bins) that represent the total amount of observations in the data for that category.
- Bar charts can be displayed with vertical columns, horizontal bars, comparative bars (multiple bars to show a comparison between values), or stacked bars (bars containing multiple types of information).



# II. Types of Chart

## 2. Bar Chart

### b. Bar chart with Plotly Express

```python
import plotly.express as px
df = px.data.gapminder().query("country == 'Canada'")
df.head()
```

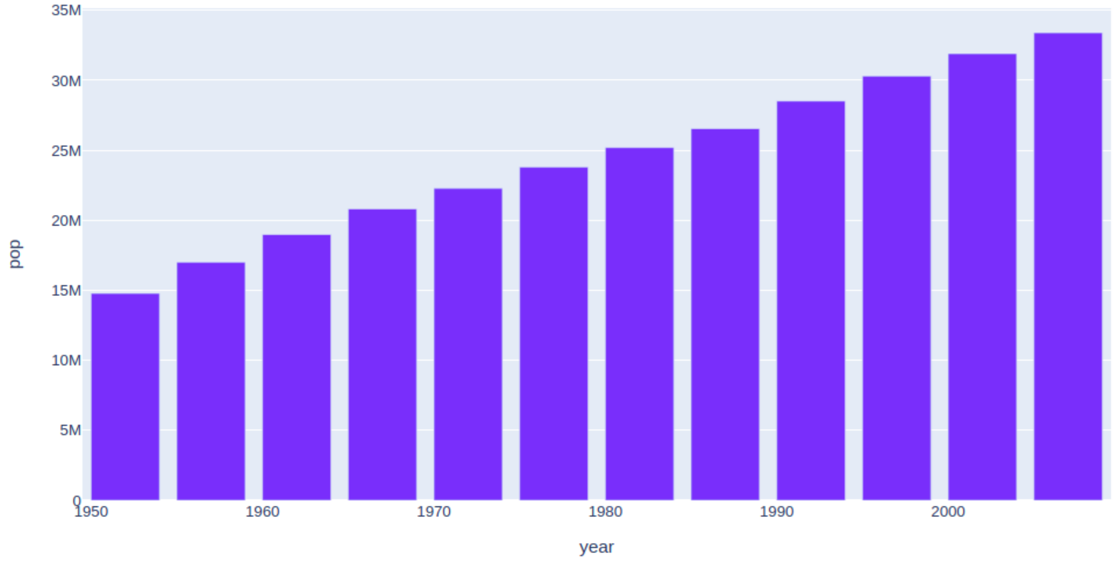
| | country | continent | year | lifeExp | pop | gdpPercap | iso_alpha | iso_num |
|---|---|---|---|---|---|---|---|---|
| **240** | Canada | Americas | 1952 | 68.75 | 14785584 | 11367.16112 | CAN | 124 |
| **241** | Canada | Americas | 1957 | 69.96 | 17010154 | 12489.95006 | CAN | 124 |
| **242** | Canada | Americas | 1962 | 71.30 | 18985849 | 13462.48555 | CAN | 124 |
| **243** | Canada | Americas | 1967 | 72.13 | 20819767 | 16076.58803 | CAN | 124 |
| **244** | Canada | Americas | 1972 | 72.88 | 22284500 | 18970.57086 | CAN | 124 |

# II. Types of Chart

## 2. Bar Chart

### c. Bar chart with Plotly Express

```
fig = px.bar(df, x='year', y='pop')
fig.show()
```

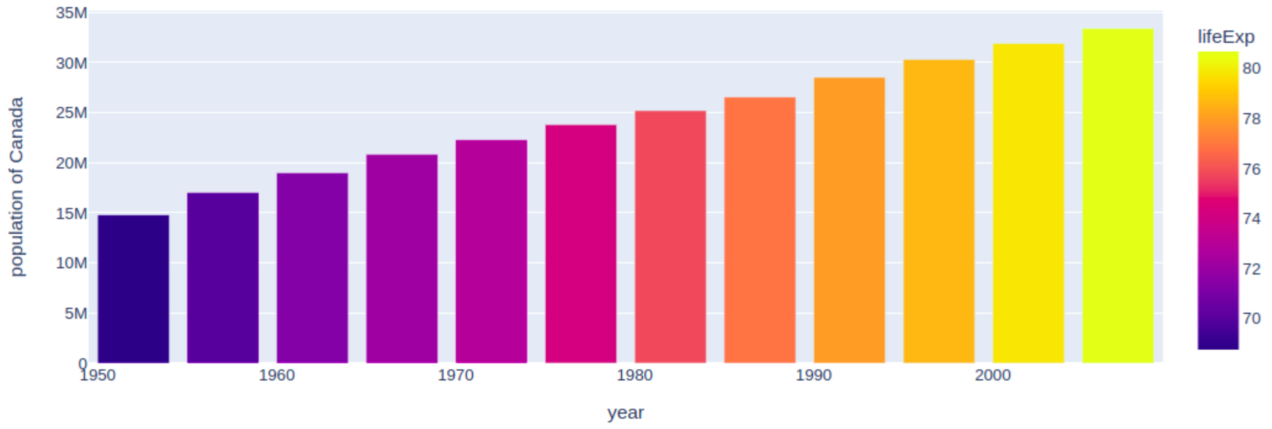


# II. Types of Chart

## 2. Bar Chart

### d. Customize bar charts with Plotly Express

```
import plotly.express as px
data = px.data.gapminder()

data_canada = data[data.country == 'Canada']
fig = px.bar(data_canada, x='year', y='pop',
             hover_data=['lifeExp', 'gdpPercap'], color='lifeExp',
             labels={'pop':'population of Canada'}, height=400)
fig.show()
```



# II. Types of Chart

## 2. Bar Chart

### e. Grouped Bar Chart

```python
import plotly.express as px

df = px.data.medals_long()
df
```

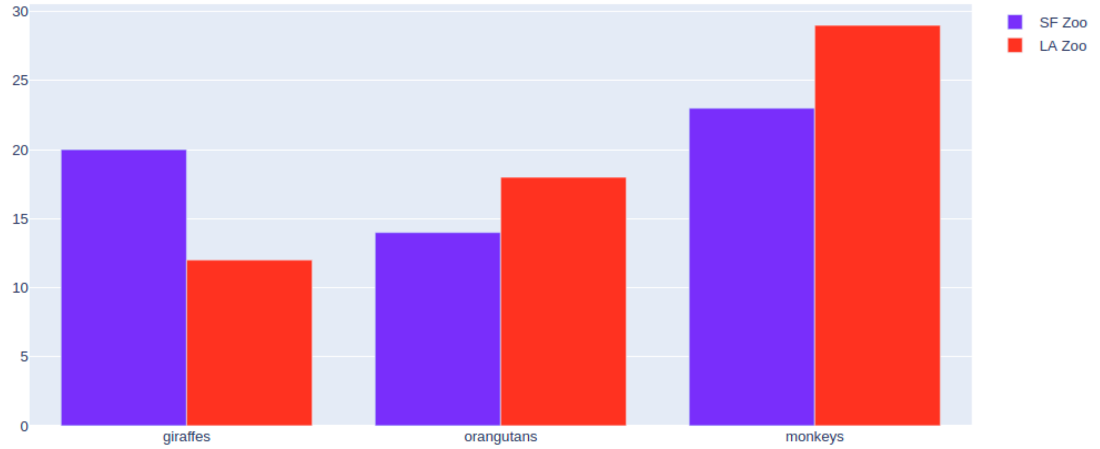
|   | nation | medal | count |
|---|---|---|---|
| **0** | South Korea | gold | 24 |
| **1** | China | gold | 10 |
| **2** | Canada | gold | 9 |
| **3** | South Korea | silver | 13 |
| **4** | China | silver | 15 |
| **5** | Canada | silver | 12 |
| **6** | South Korea | bronze | 11 |
| **7** | China | bronze | 8 |
| **8** | Canada | bronze | 12 |

## II. Types of Chart

### 2. Bar Chart

e. Grouped Bar Chart

```
import plotly.graph_objects as go
animals=['giraffes', 'orangutans', 'monkeys']

fig = go.Figure(data=[
    go.Bar(name='SF Zoo', x=animals, y=[20, 14, 23]),
    go.Bar(name='LA Zoo', x=animals, y=[12, 18, 29])
])
# Change the bar mode
fig.update_layout(barmode='group')
fig.show()
```

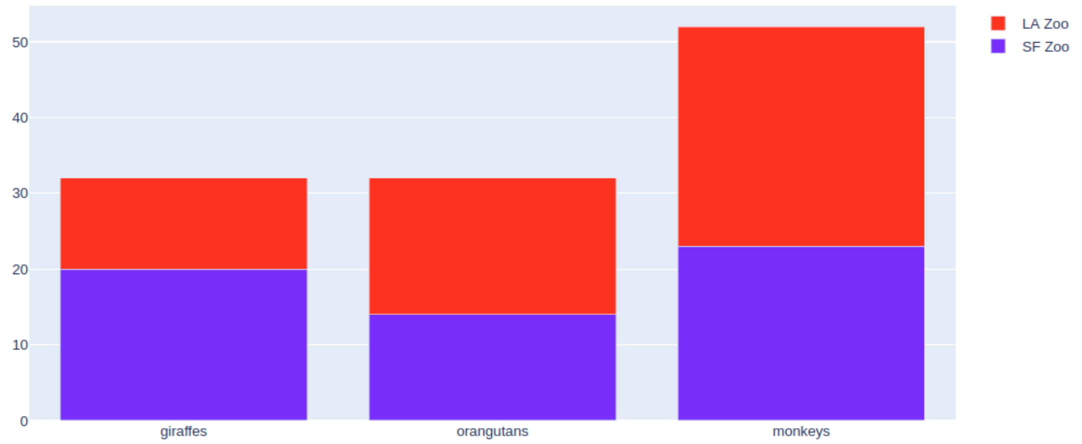


# II. Types of Chart

## 2. Bar Chart

### f. Stack Bar chart

```python
import plotly.graph_objects as go
animals=['giraffes', 'orangutans', 'monkeys']

fig = go.Figure(data=[
    go.Bar(name='SF Zoo', x=animals, y=[20, 14, 23]),
    go.Bar(name='LA Zoo', x=animals, y=[12, 18, 29])
])
# Change the bar mode
fig.update_layout(barmode='stack')
fig.show()
```

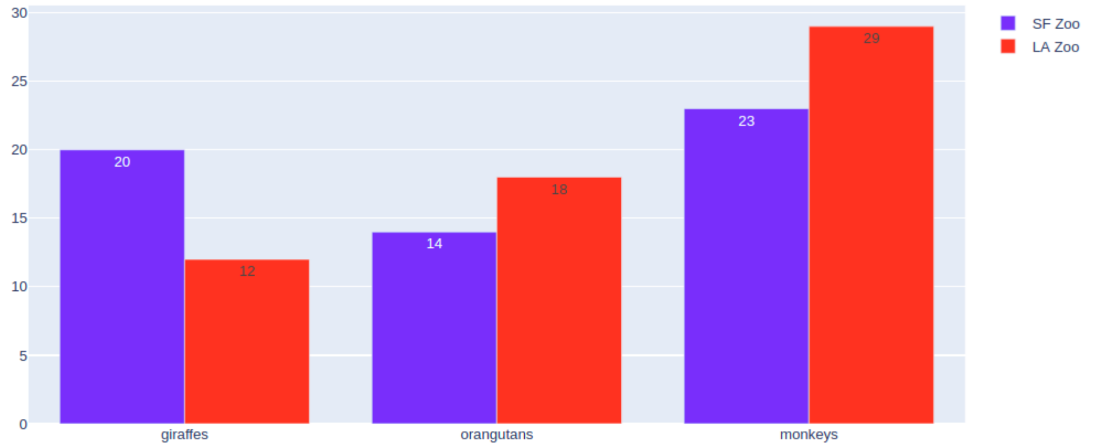


# II. Types of Chart

## 2. Bar Chart

### g. Bar Chart with Direct Labels

```python
import plotly.graph_objects as go
animals=['giraffes', 'orangutans', 'monkeys']

fig = go.Figure(data=[
    go.Bar(name='SF Zoo', x=animals, y=[20, 14, 23], text=[20, 14, 23], textposition='auto'),
    go.Bar(name='LA Zoo', x=animals, y=[12, 18, 29], text=[12, 18, 29], textposition='auto')
])
# Change the bar mode
fig.update_layout(barmode='group')
fig.show()
```

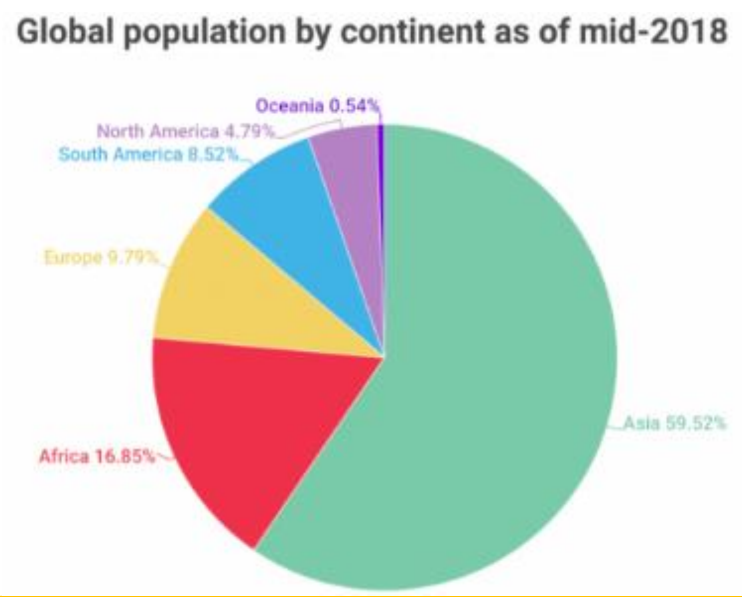


# II. Types of Chart

## 3. Pie Chart

### a. Definition

- A Pie Chart is a type of graph that displays data in a circular graph.
- The pieces of the graph are proportional to the fraction of the whole in each category. In other words, each slice of the pie is relative to the size of that category in the group as a whole.
- The entire “pie” represents 100 percent of a whole, while the pi “slices” represent portions of the whole.



# II. Types of Chart

## 3. Pie Chart

### b. Pie chart with plotly express

```python
import plotly.express as px
df = px.data.gapminder().query("year == 2007").query("continent == 'Europe'")
df.loc[df['pop'] < 2.e6, 'country'] = 'Other countries' # Represent only large countries
df.head()
```

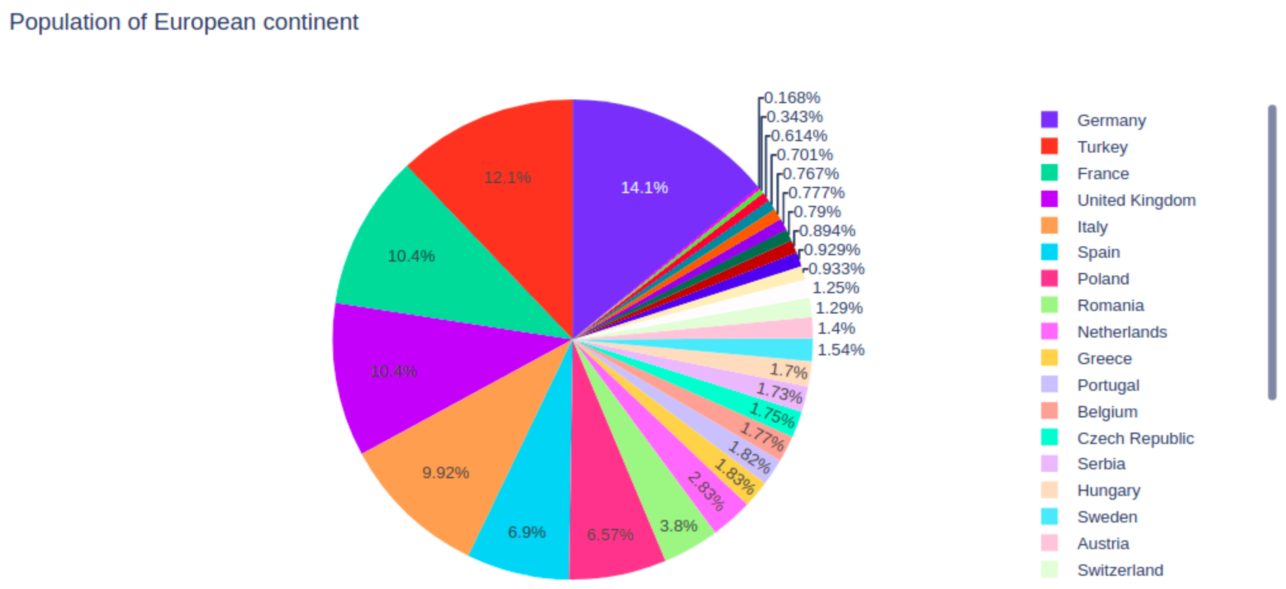
| | country | continent | year | lifeExp | pop | gdpPercap | iso_alpha | iso_num |
|---|---|---|---|---|---|---|---|---|
| **23** | Albania | Europe | 2007 | 76.423 | 3600523 | 5937.029526 | ALB | 8 |
| **83** | Austria | Europe | 2007 | 79.829 | 8199783 | 36126.492700 | AUT | 40 |
| **119** | Belgium | Europe | 2007 | 79.441 | 10392226 | 33692.605080 | BEL | 56 |
| **155** | Bosnia and Herzegovina | Europe | 2007 | 74.852 | 4552198 | 7446.298803 | BIH | 70 |
| **191** | Bulgaria | Europe | 2007 | 73.005 | 7322858 | 10680.792820 | BGR | 100 |

# II. Types of Chart

## 3. Pie Chart

### b. Pie chart with plotly express

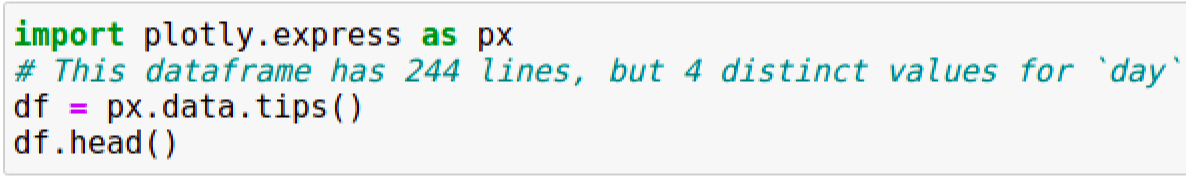
```
fig = px.pie(df, values='pop', names='country', title='Population of European continent')
fig.show()
```



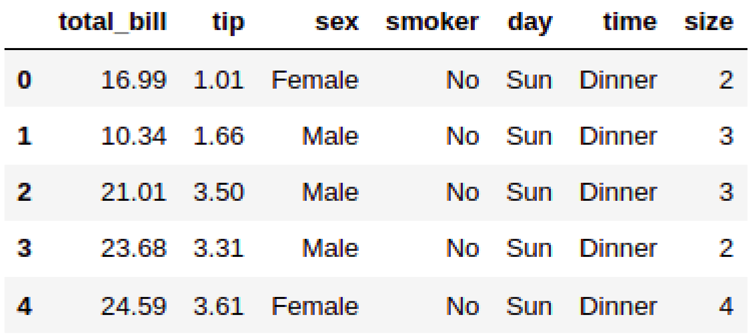
# II. Types of Chart

## 3. Pie Chart

### c. Pie chart with repeated labels



```python
import plotly.express as px
# This dataframe has 244 lines, but 4 distinct values for `day`
df = px.data.tips()
df.head()
```



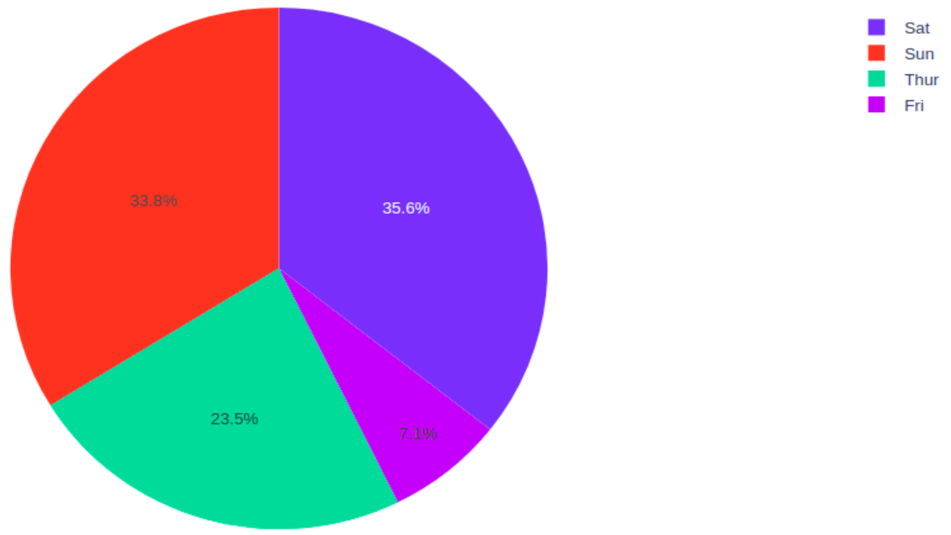
| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| **0** | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| **1** | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| **2** | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| **3** | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| **4** | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

## II. Types of Chart

### 3. Pie Chart

c. Pie chart with repeated labels

```
fig = px.pie(df, values='tip', names='day')
fig.show()
```

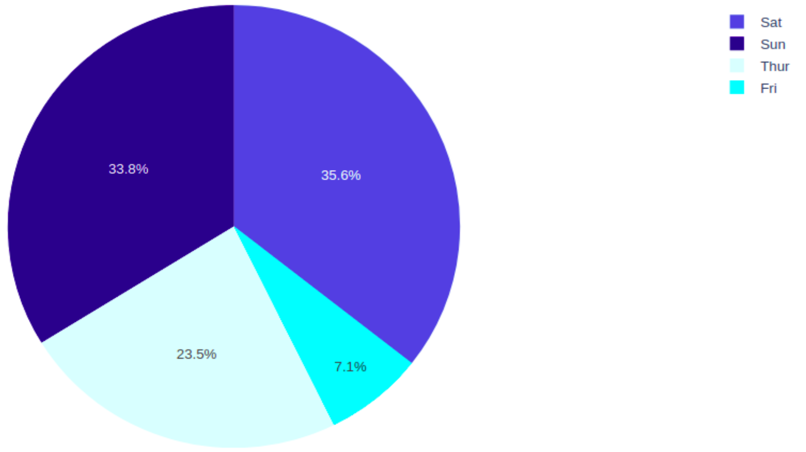


# II. Types of Chart

## 3. Pie Chart

### d. Using an explicit mapping for discrete colors

```
import plotly.express as px
df = px.data.tips()
fig = px.pie(df, values='tip', names='day', color='day',
             color_discrete_map={'Thur':'lightcyan',
                                 'Fri':'cyan',
                                 'Sat':'royalblue',
                                 'Sun':'darkblue'})
fig.show()
```

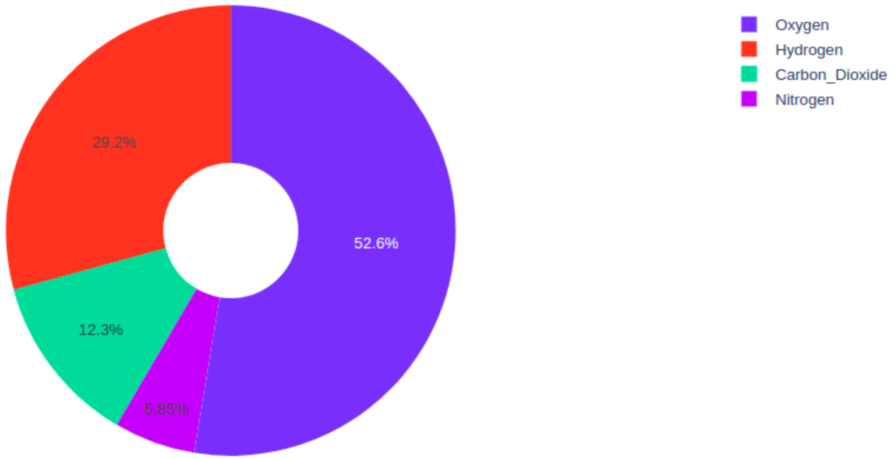


# II. Types of Chart

## 3. Pie Chart

### e. Donut Chart

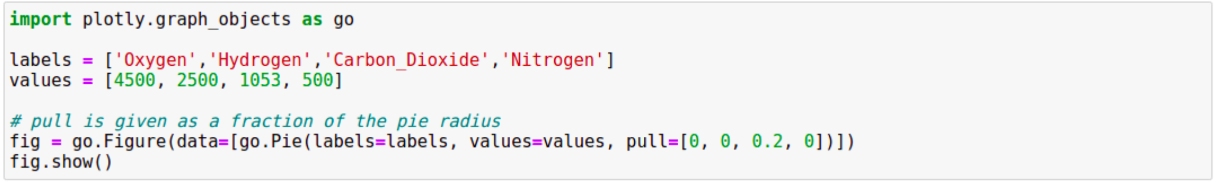
```
import plotly.graph_objects as go

labels = ['Oxygen','Hydrogen','Carbon_Dioxide','Nitrogen']
values = [4500, 2500, 1053, 500]

# Use `hole` to create a donut-like pie chart
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.show()
```



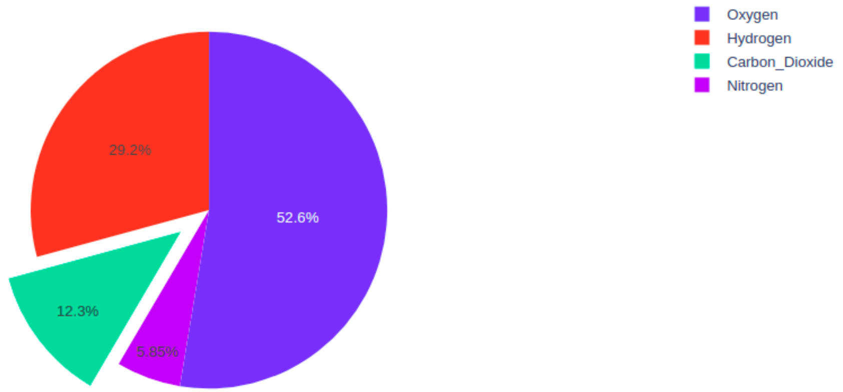
## II. Types of Chart

### 3. Pie Chart

f. Pulling sectors out from the center



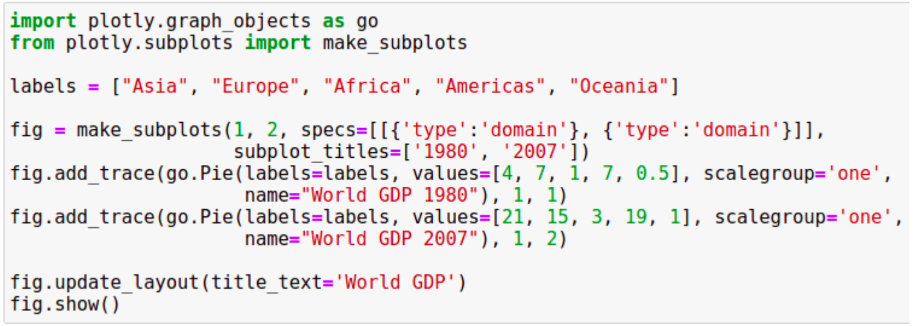
```
import plotly.graph_objects as go

labels = ['Oxygen','Hydrogen','Carbon_Dioxide','Nitrogen']
values = [4500, 2500, 1053, 500]

# pull is given as a fraction of the pie radius
fig = go.Figure(data=[go.Pie(labels=labels, values=values, pull=[0, 0, 0.2, 0])])
fig.show()
```



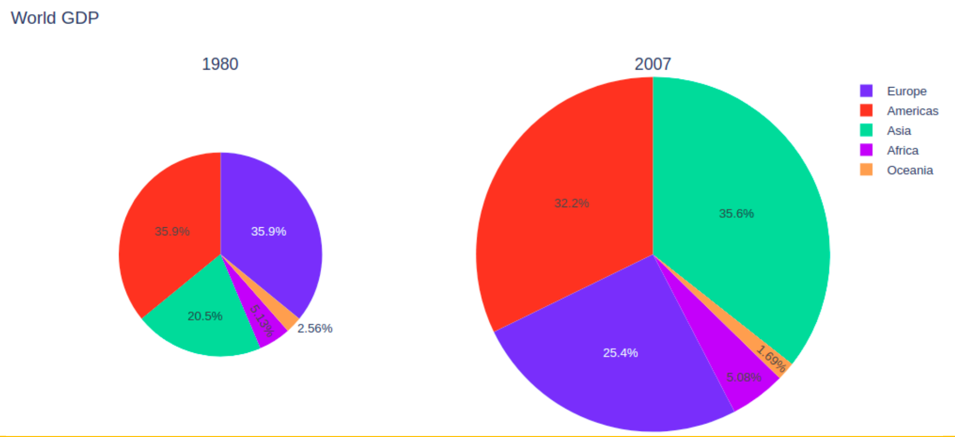
# II. Types of Chart

## 3. Pie Chart

### g. Multiple Pie Charts



```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

labels = ["Asia", "Europe", "Africa", "Americas", "Oceania"]

fig = make_subplots(1, 2, specs=[[{'type':'domain'}, {'type':'domain'}]],
                    subplot_titles=['1980', '2007'])
fig.add_trace(go.Pie(labels=labels, values=[4, 7, 1, 7, 0.5], scalegroup='one',
                     name="World GDP 1980"), 1, 1)
fig.add_trace(go.Pie(labels=labels, values=[21, 15, 3, 19, 1], scalegroup='one',
                     name="World GDP 2007"), 1, 2)

fig.update_layout(title_text='World GDP')
fig.show()
```

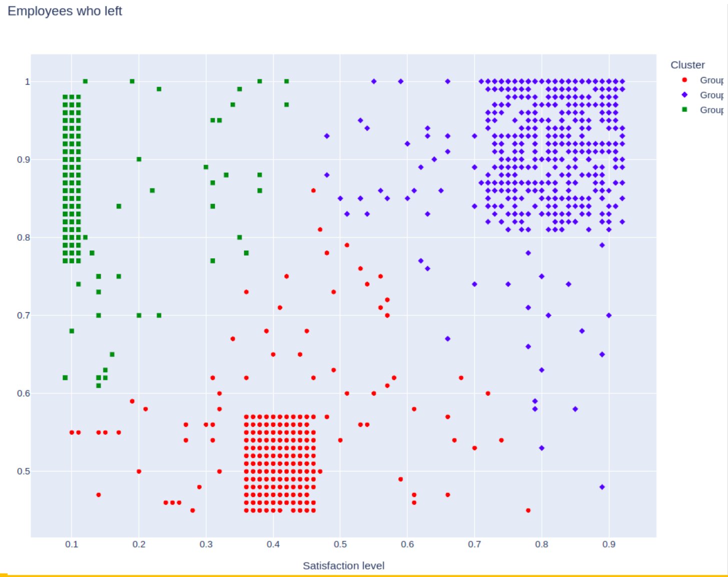


# II. Types of Chart

## 4. Scatter Chart

### a. Definition

- A scatter plot uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point.
- Scatter plots are also used to describe relationships between variables.



# II. Types of Chart

## 4. Scatter Chart

### b. Scatter plots with Plotly Express

```python
# x and y given as DataFrame columns
import plotly.express as px
df = px.data.iris() # iris is a pandas DataFrame
df.head()
```

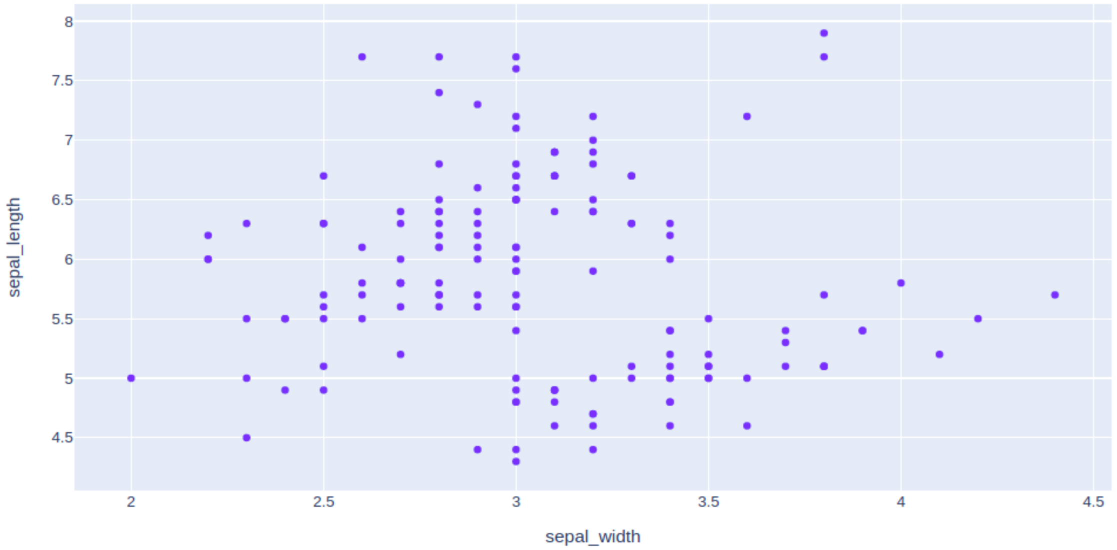
|   | sepal_length | sepal_width | petal_length | petal_width | species | species_id |
|---|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa | 1 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa | 1 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa | 1 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa | 1 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa | 1 |

## II. Types of Chart

### 4. Scratter Chart

b. Scatter plots with Plotly Express

```
fig = px.scatter(df, x="sepal_width", y="sepal_length")
fig.show()
```

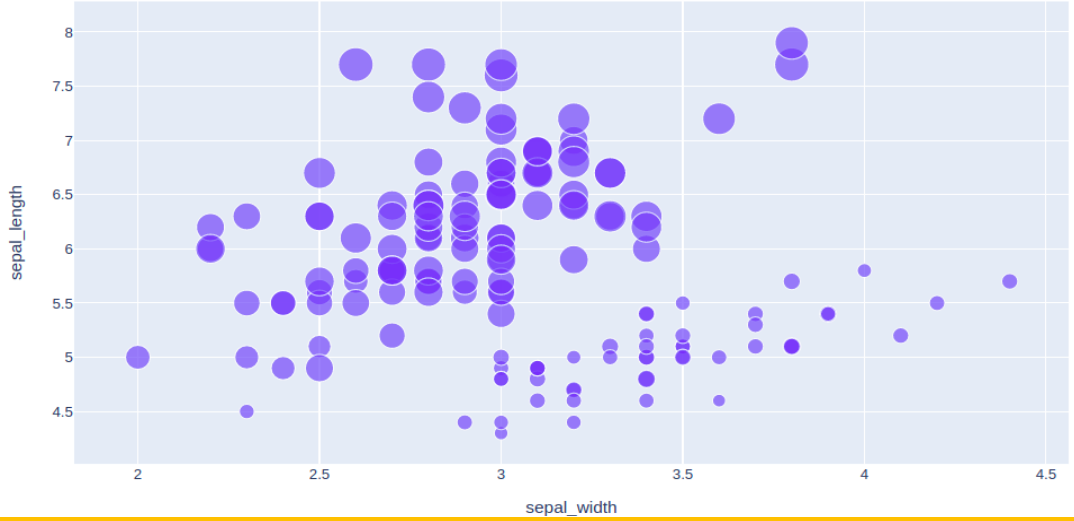


# II. Types of Chart

## 4. Scratter Chart

### c. Setting size with column names (bubble chart)

```python
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length",
                 size='petal_length', hover_data=['petal_width'])
fig.show()
```

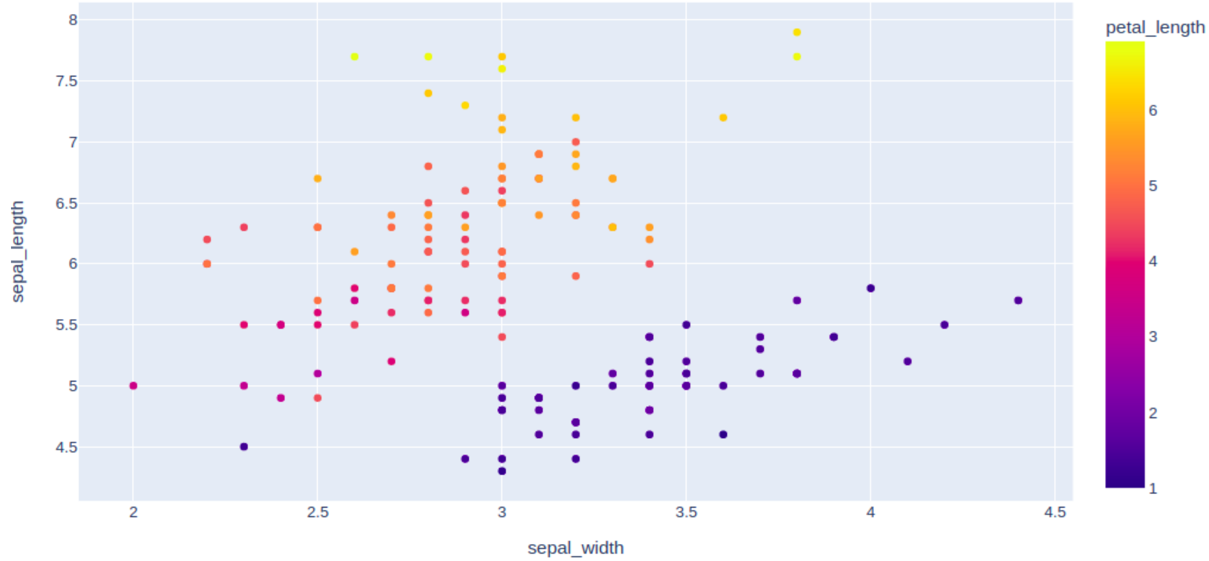


# II. Types of Chart

## 4. Scratter Chart

### d. Setting color with column names

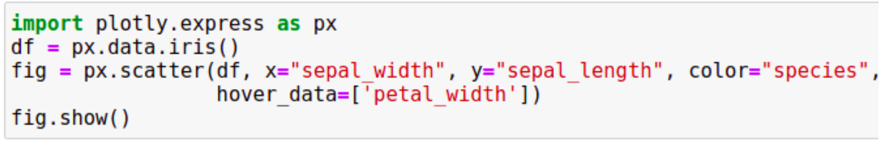
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color='petal_length')
fig.show()
```



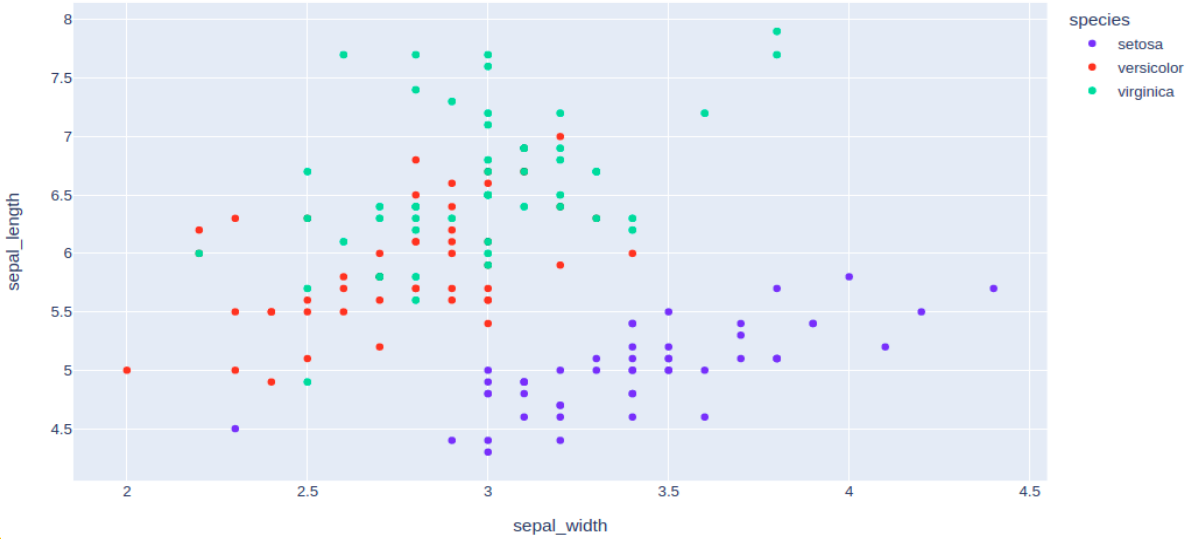
## II. Types of Chart

### 4. Scratter Chart

e. Setting color with column names



```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species",
                 hover_data=['petal_width'])
fig.show()
```

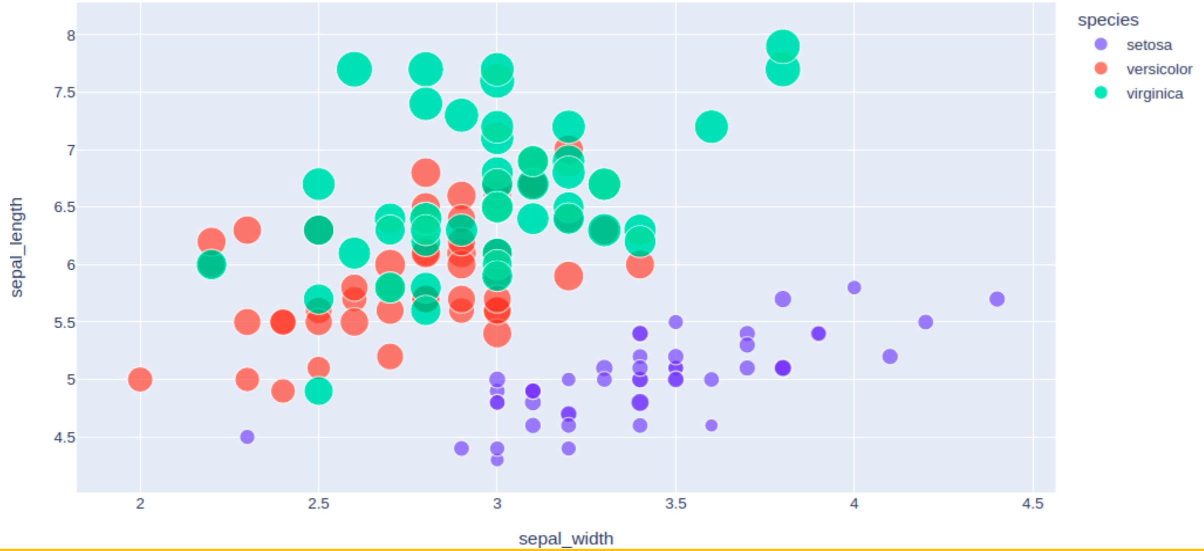


# II. Types of Chart

## 4. Scratter Chart

### f. Setting size and color with column names

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species",
                 size='petal_length', hover_data=['petal_width'])
fig.show()
```

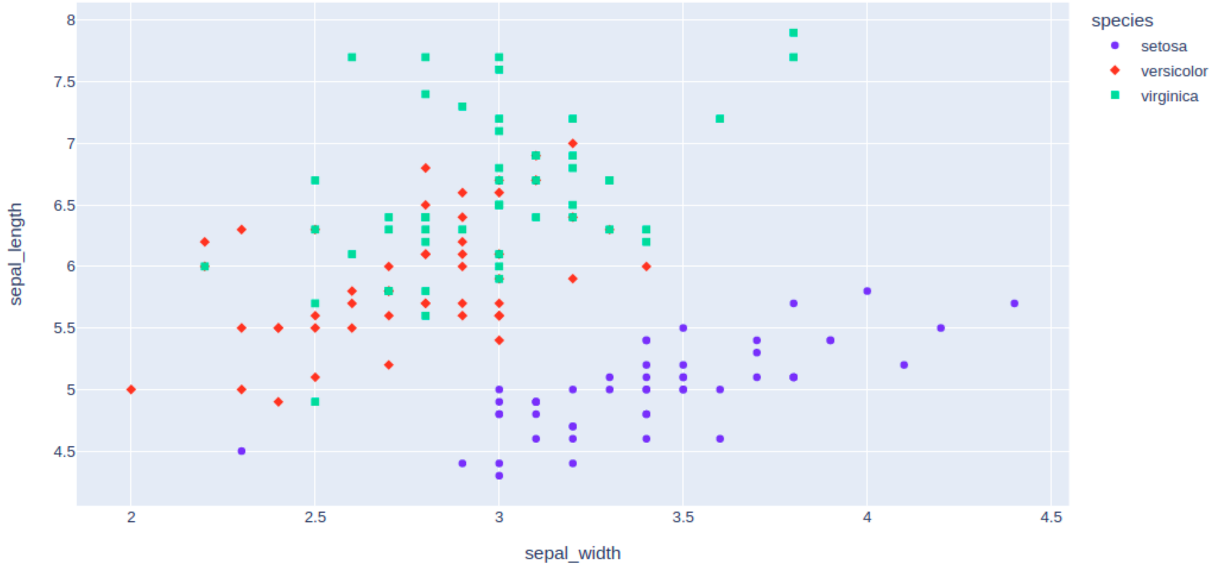


# II. Types of Chart

## 4. Scratter Chart

### g. Setting symbol with column names

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species", symbol="species")
fig.show()
```

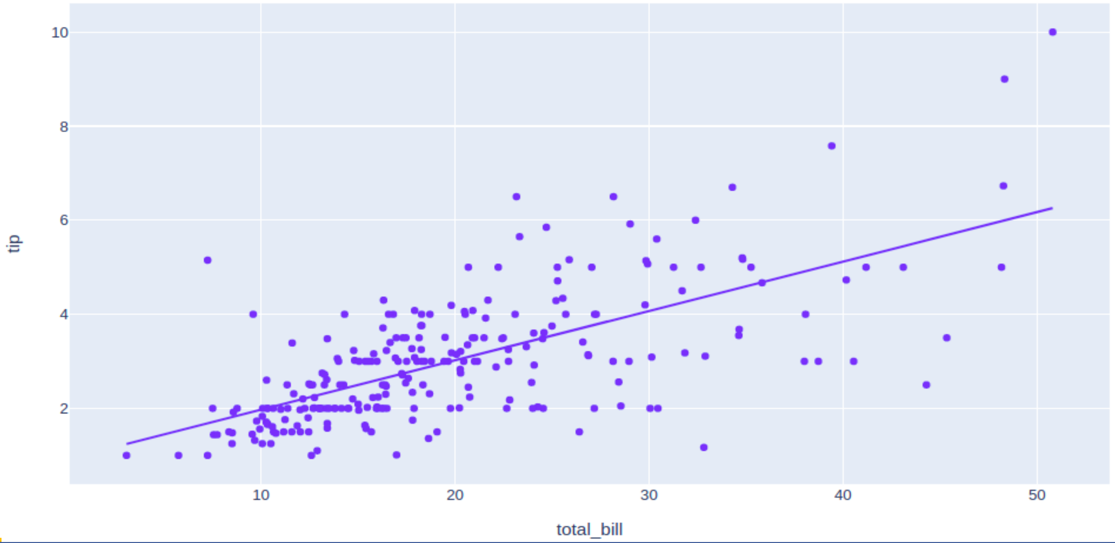


# II. Types of Chart

## 4. Scratter Chart

### h. Linear Regression and Other Trendlines

```
import plotly.express as px

df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", trendline="ols")
fig.show()
```

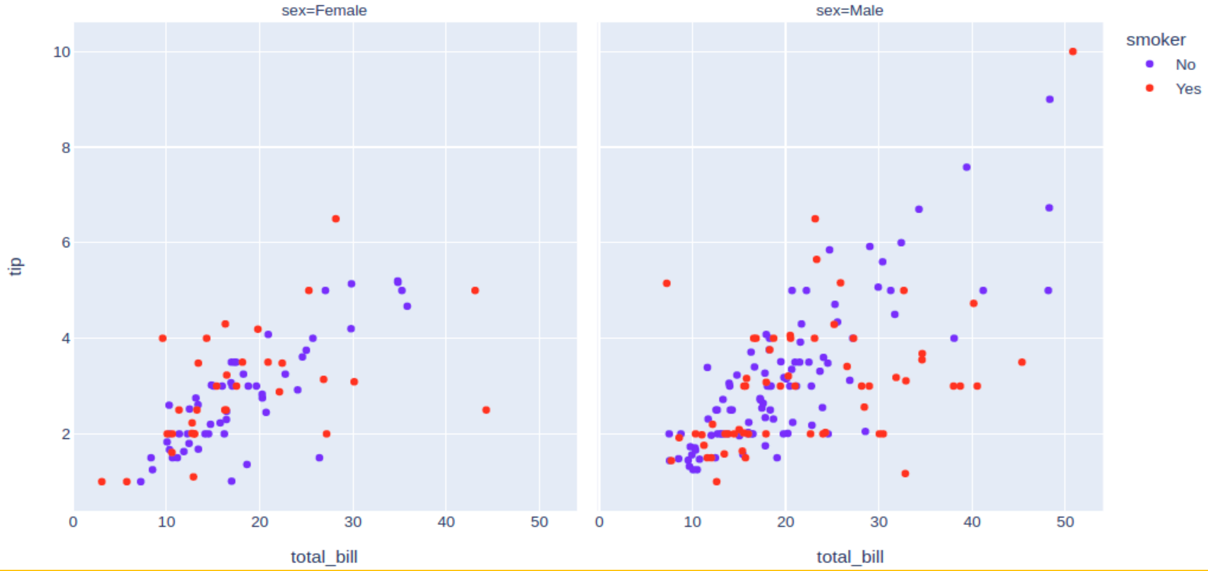


# II. Types of Chart

## 4. Scatter Chart

### i. Multiple Scatter Chart

```
import plotly.express as px
df = px.data.tips()
fig = px.scatter(df, x="total_bill", y="tip", color="smoker", facet_col="sex")
fig.show()
```



# II. Types of Chart

## 5. Violin Chart

### a. Definition

- Violin plots are a method of plotting numeric data and can be considered a combination of the box plot with a kernel density plot.

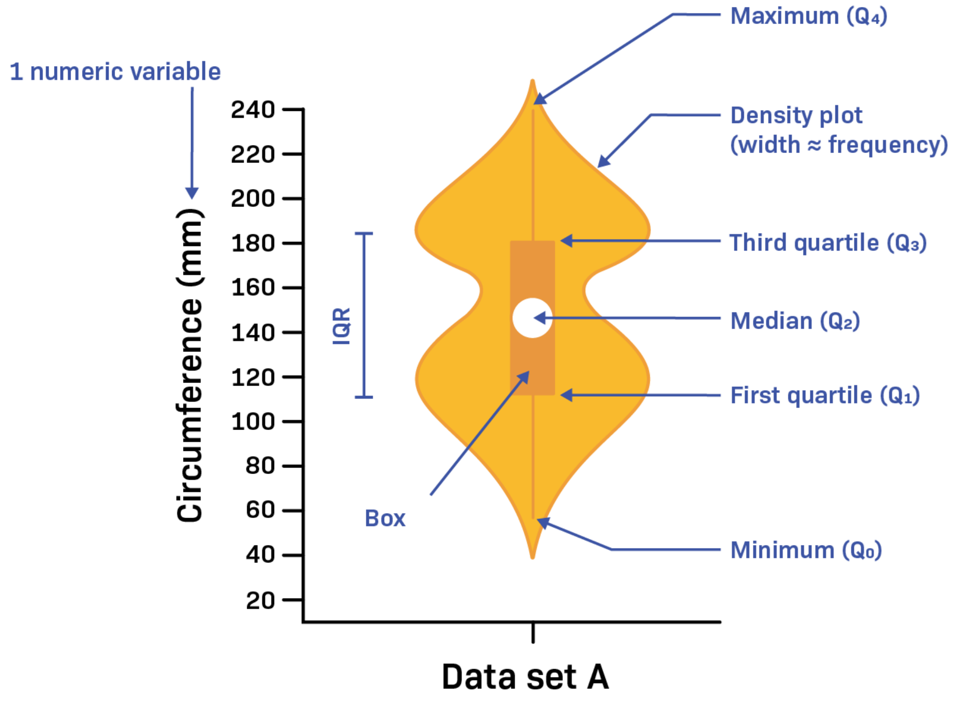
- In the violin plot, we can find the information such as:

- Distribution of data
- Median (a white dot on the violin plot)
- Interquartile range
- The lower/upper adjacent values

# II. Types of Chart

## 5. Violin Chart

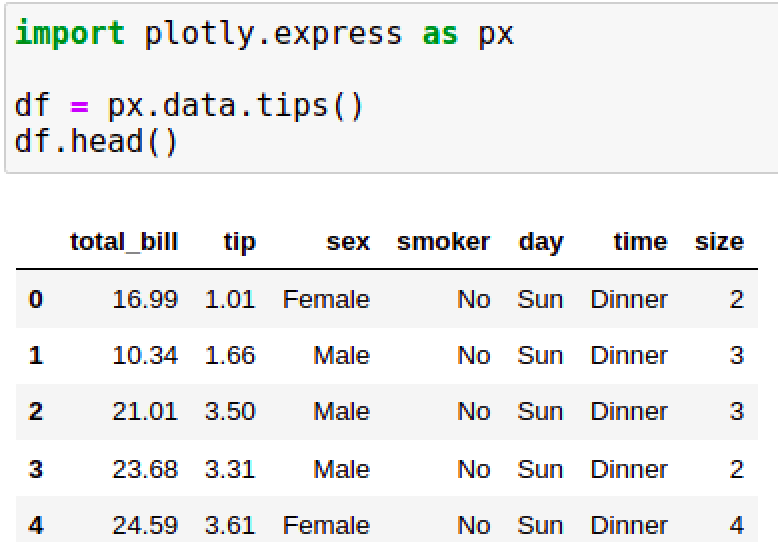
### a. Definition



# II. Types of Chart

## 5. Violin Chart

### b. Basic Violin Plot with Plotly Express



```python
import plotly.express as px

df = px.data.tips()
df.head()
```

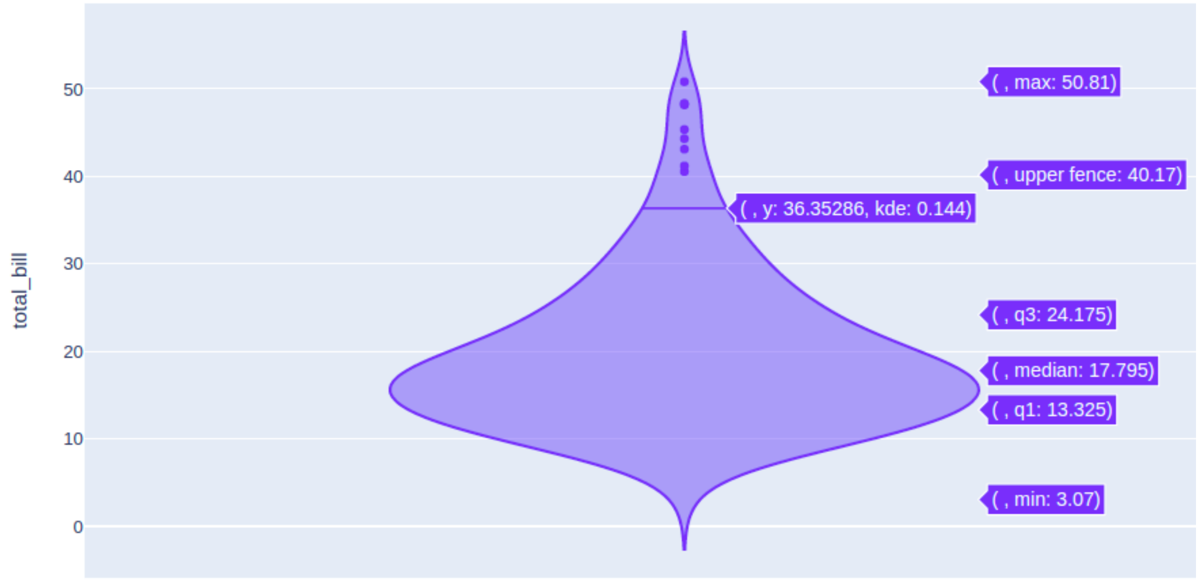
```
   total_bill   tip     sex  smoker  day    time  size
0       16.99  1.01  Female      No  Sun  Dinner     2
1       10.34  1.66    Male      No  Sun  Dinner     3
2       21.01  3.50    Male      No  Sun  Dinner     3
3       23.68  3.31    Male      No  Sun  Dinner     2
4       24.59  3.61  Female      No  Sun  Dinner     4
```

# II. Types of Chart

## 5. Violin Chart

### b. Basic Violin Plot with Plotly Express

```
fig = px.violin(df, y="total_bill")
fig.show()
```

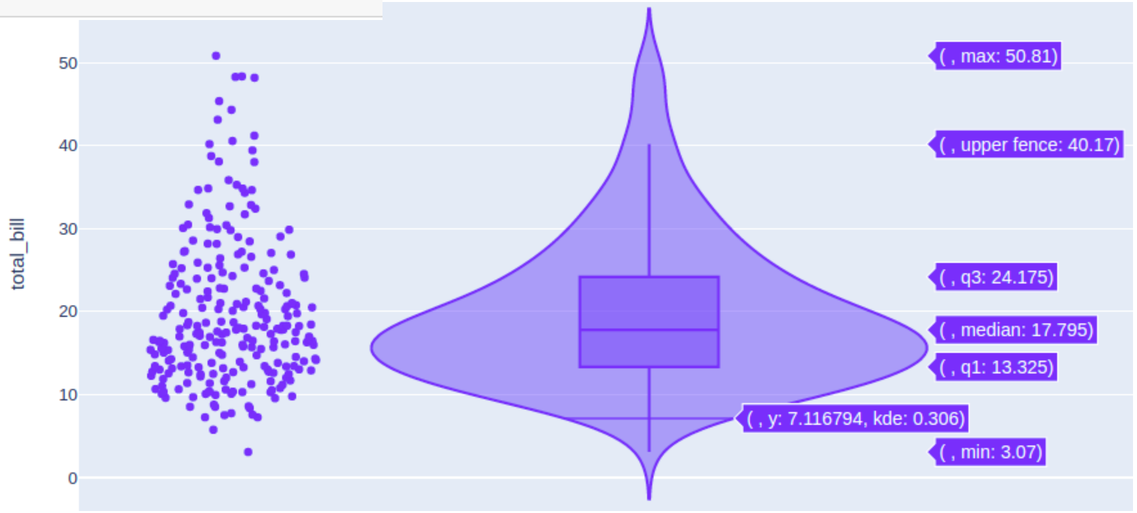


## II. Types of Chart

### 5. Violin Chart

c. Violin plot with box and data points

```
import plotly.express as px

df = px.data.tips()
fig = px.violin(df, y="total_bill", box=True, # draw box plot inside the violin
                points='all', # can be 'outliers', or False
               )
fig.show()
```

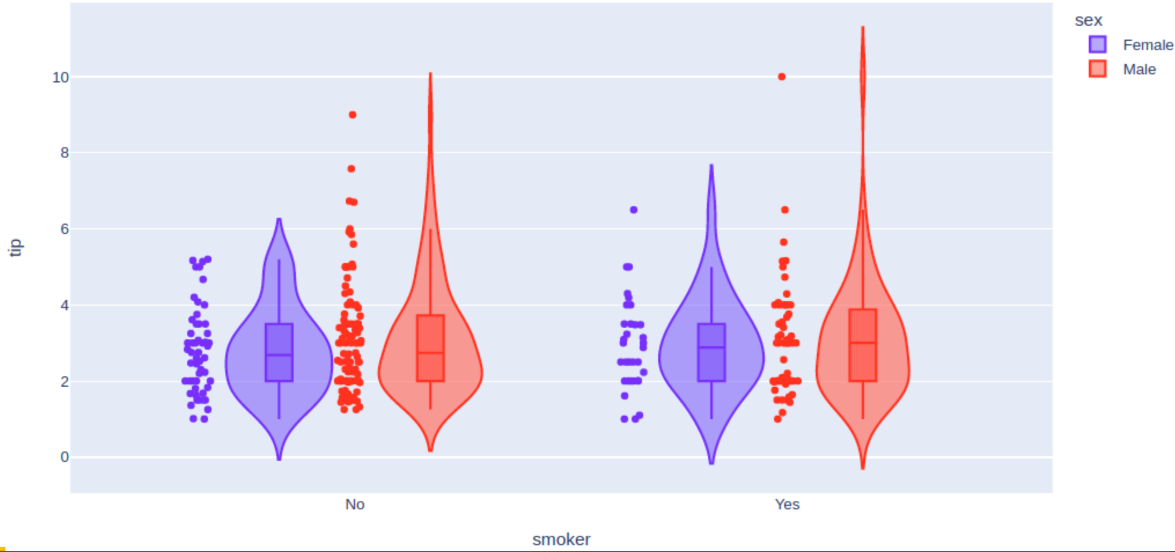


# II. Types of Chart

## 5. Violin Chart

### d. Multiple Violin Plots

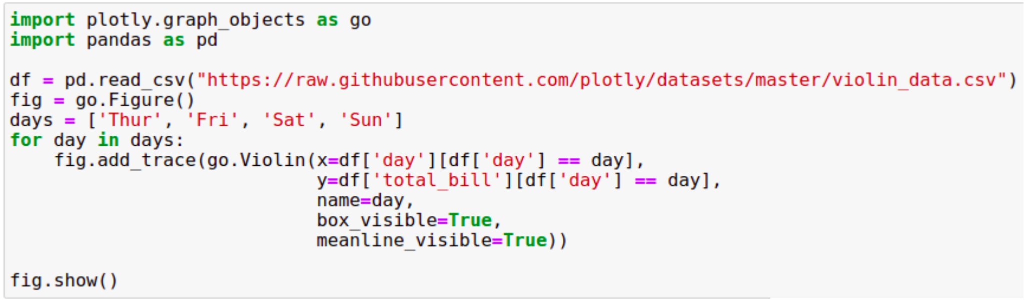
```
import plotly.express as px

df = px.data.tips()
fig = px.violin(df, y="tip", x="smoker", color="sex", box=True, points="all",
          hover_data=df.columns)
fig.show()
```



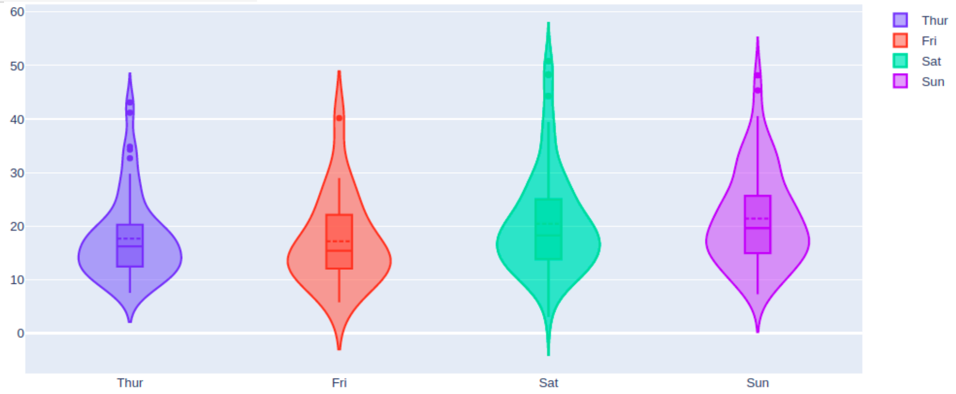
# II. Types of Chart

## 5. Violin Chart

### e. Multiple Violin Plots



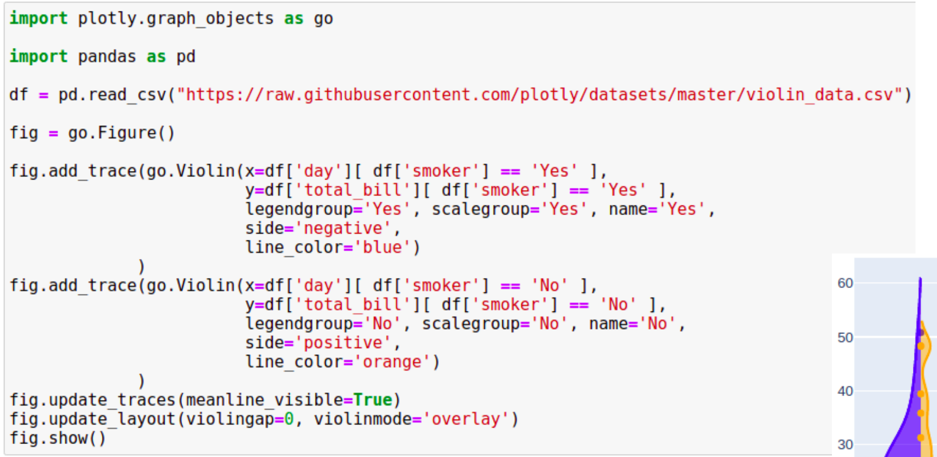
```
import plotly.graph_objects as go
import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/violin_data.csv")
fig = go.Figure()
days = ['Thur', 'Fri', 'Sat', 'Sun']
for day in days:
    fig.add_trace(go.Violin(x=df['day'][df['day'] == day],
                            y=df['total_bill'][df['day'] == day],
                            name=day,
                            box_visible=True,
                            meanline_visible=True))

fig.show()
```



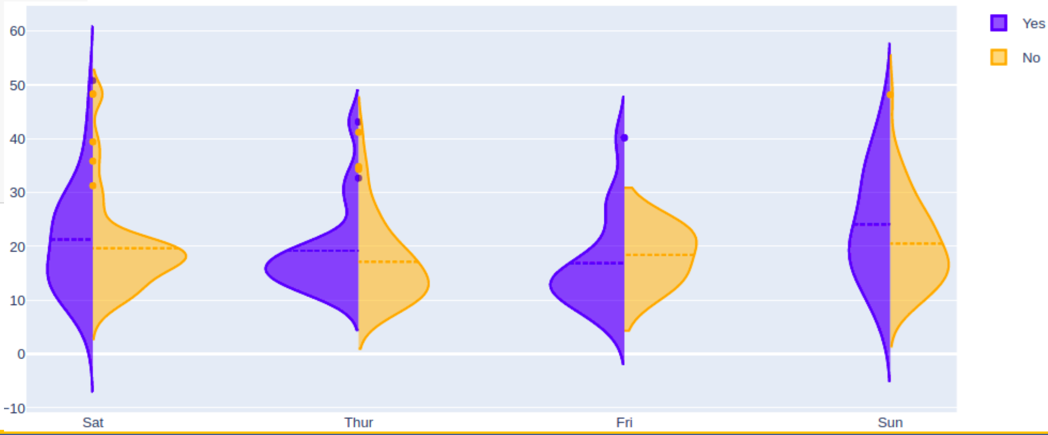
# II. Types of Chart

## 5. Violin Chart

### f. Split Violin Plot



```
import plotly.graph_objects as go

import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/violin_data.csv")

fig = go.Figure()

fig.add_trace(go.Violin(x=df['day'][ df['smoker'] == 'Yes' ],
                        y=df['total_bill'][ df['smoker'] == 'Yes' ],
                        legendgroup='Yes', scalegroup='Yes', name='Yes',
                        side='negative',
                        line_color='blue')
             )
fig.add_trace(go.Violin(x=df['day'][ df['smoker'] == 'No' ],
                        y=df['total_bill'][ df['smoker'] == 'No' ],
                        legendgroup='No', scalegroup='No', name='No',
                        side='positive',
                        line_color='orange')
             )
fig.update_traces(meanline_visible=True)
fig.update_layout(violingap=0, violinmode='overlay')
fig.show()
```

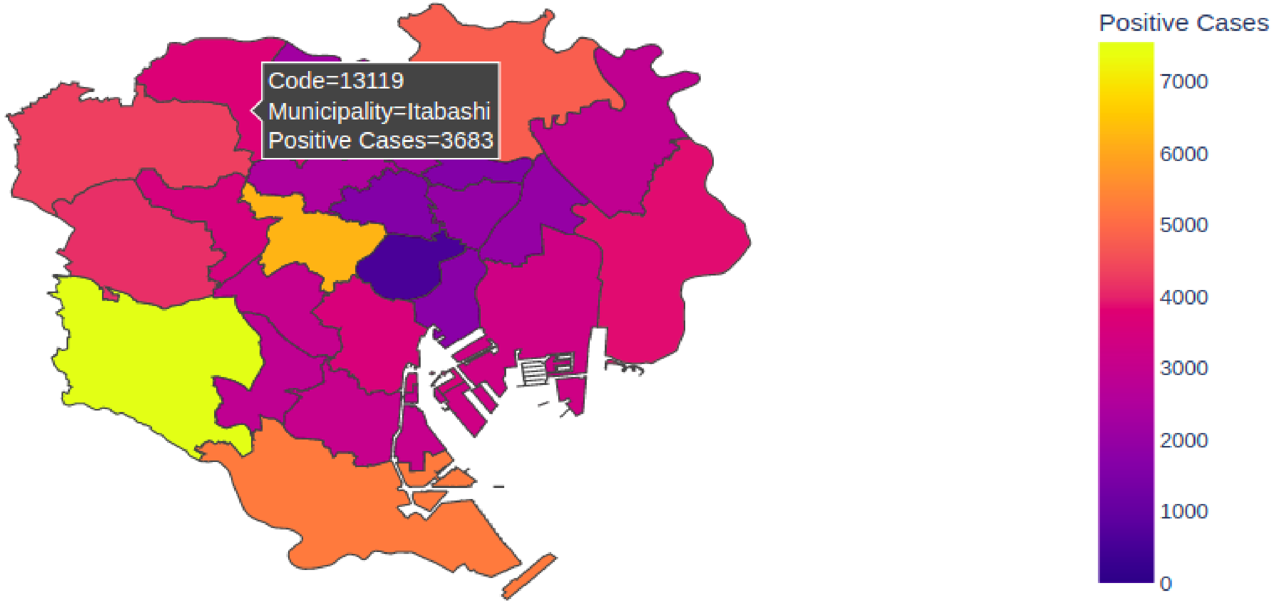


# II. Types of Chart

## 6. Map Chart

### a. Definition

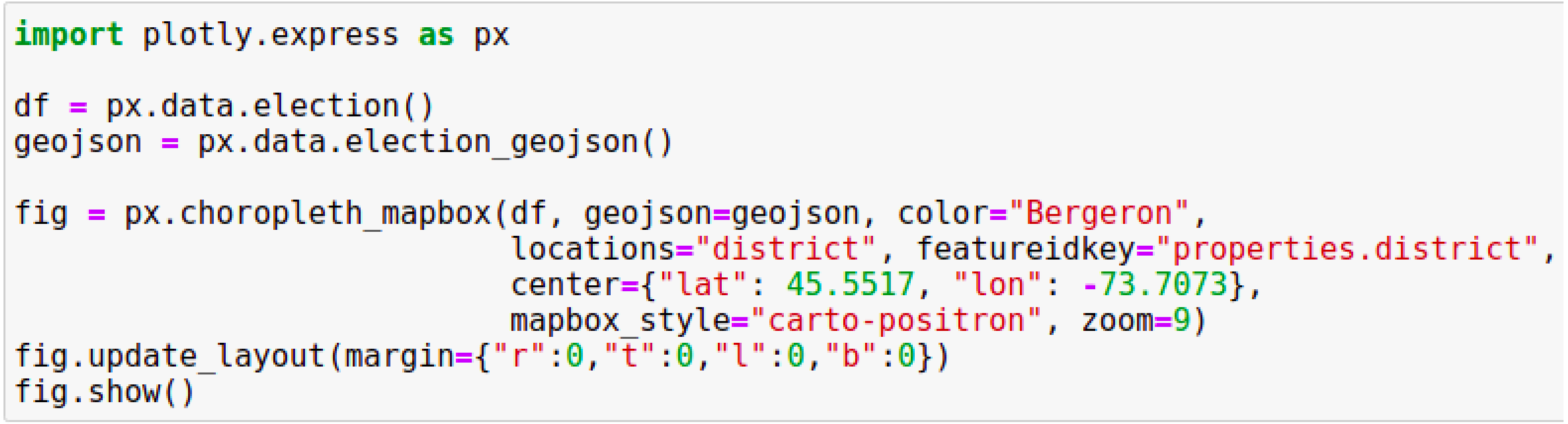
A map chart provides a visualization of a geographic region, which contains data from the underlying source data.



## II. Types of Chart

### 6. Map Chart

b. Map chart with continuous color

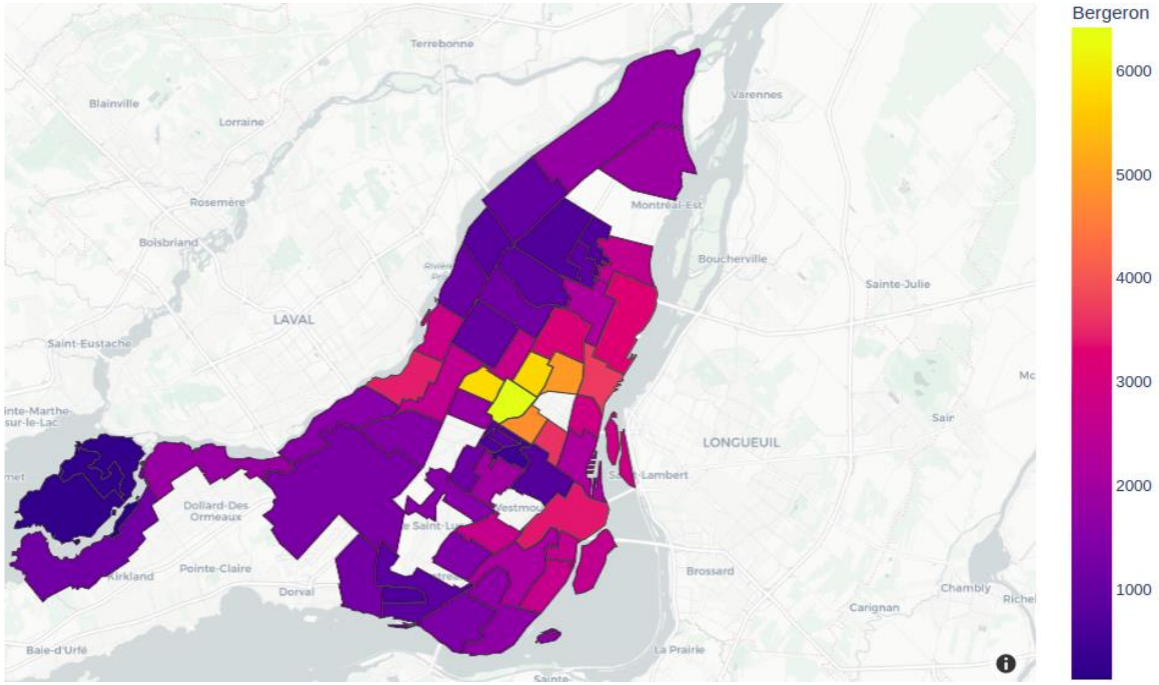


```python
import plotly.express as px

df = px.data.election()
geojson = px.data.election_geojson()

fig = px.choropleth_mapbox(df, geojson=geojson, color="Bergeron",
                           locations="district", featureidkey="properties.district",
                           center={"lat": 45.5517, "lon": -73.7073},
                           mapbox_style="carto-positron", zoom=9)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

# II. Types of Chart

## 6. Map Chart

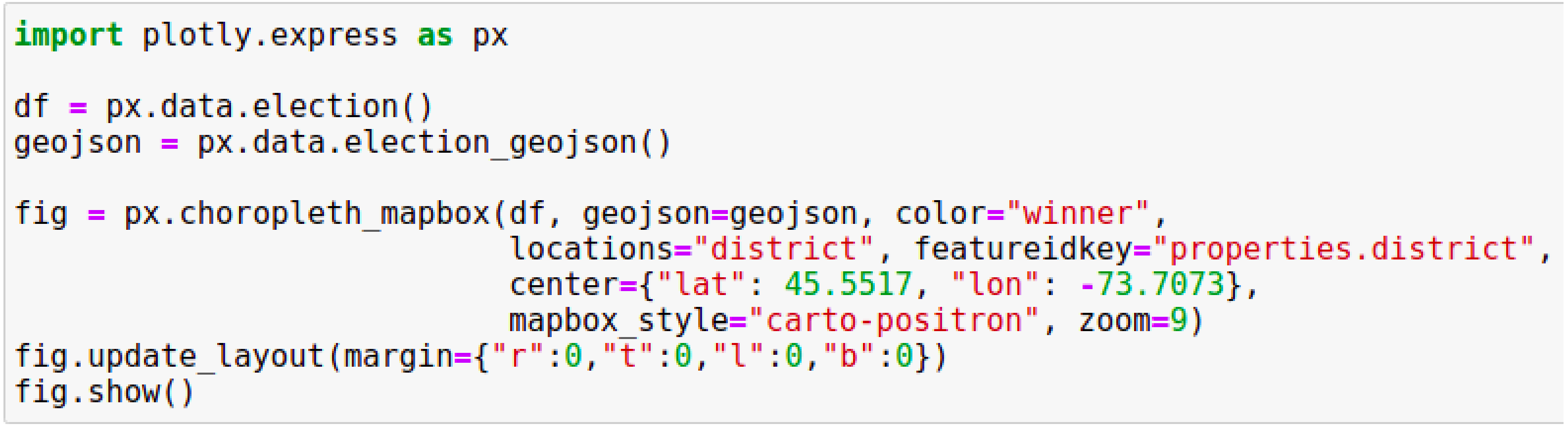
b. Map chart with continuous color



## II. Types of Chart

### 6. Map Chart

c. Map chart with Discrete color

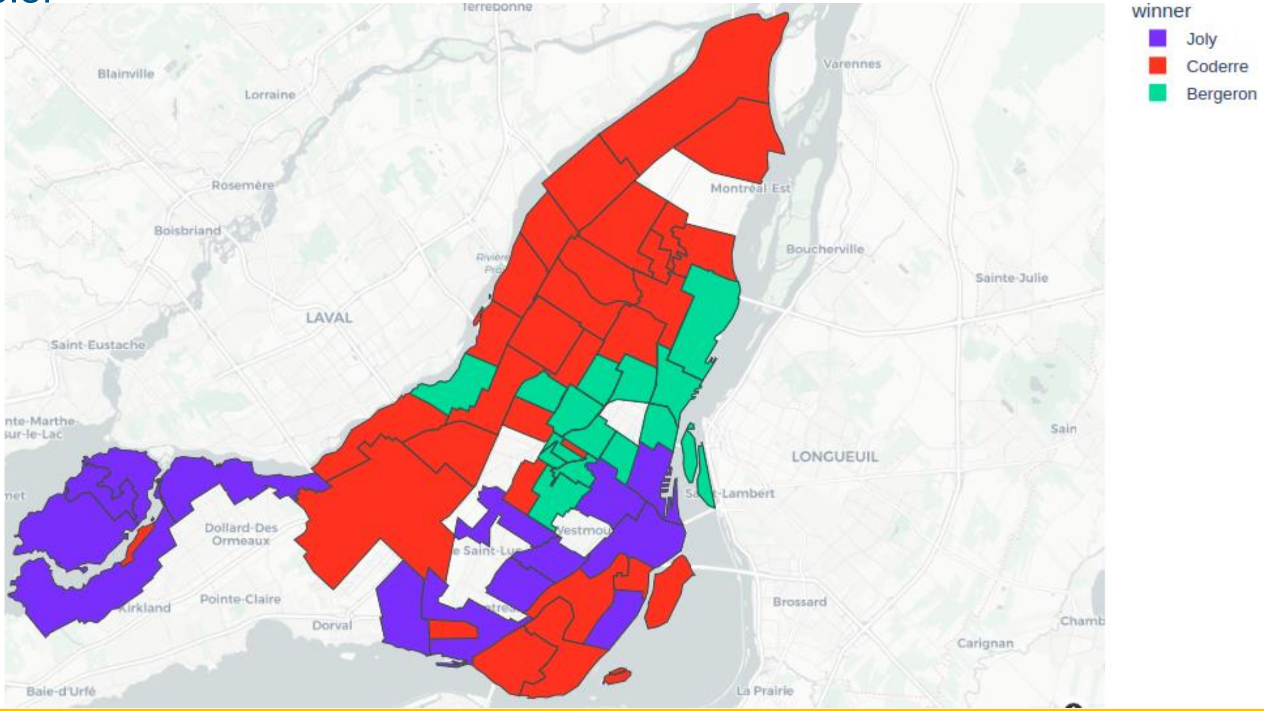


```
import plotly.express as px

df = px.data.election()
geojson = px.data.election_geojson()

fig = px.choropleth_mapbox(df, geojson=geojson, color="winner",
                           locations="district", featureidkey="properties.district",
                           center={"lat": 45.5517, "lon": -73.7073},
                           mapbox_style="carto-positron", zoom=9)
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

# II. Types of Chart

## 6. Map Chart

c. Map chart with Discrete color



## III. Reference

**Book:**

Storytelling with data, chapter 2, 6

# Q & A