Plotly and the Plotly Figure

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Alex Scriven
Data Scientist

datacamp

What is Plotly?

- A JavaScript graphing library
  - Don't worry - no need to know JavaScript!
- Plotly has a Python wrapper

Why Plotly?

Plotly has a number of unique advantages:

- Fast and easy to use
- Low code/low effort options using `plotly.express`
- (If desired) Extremely customizable
- Built-in interactivity

Creating Plotly Figures

Plotly graphs can be created:

1. With `plotly.express` for quick plots (`px`)
2. With `plotly.graph_objects` (`go`) for more customization

 We will spend most of our time on `px`

The importance of documentation

Save the links to key documentation!

1. **Interactive documentation**
2. **Detailed reference page** for specific plots

For Scatter plots:

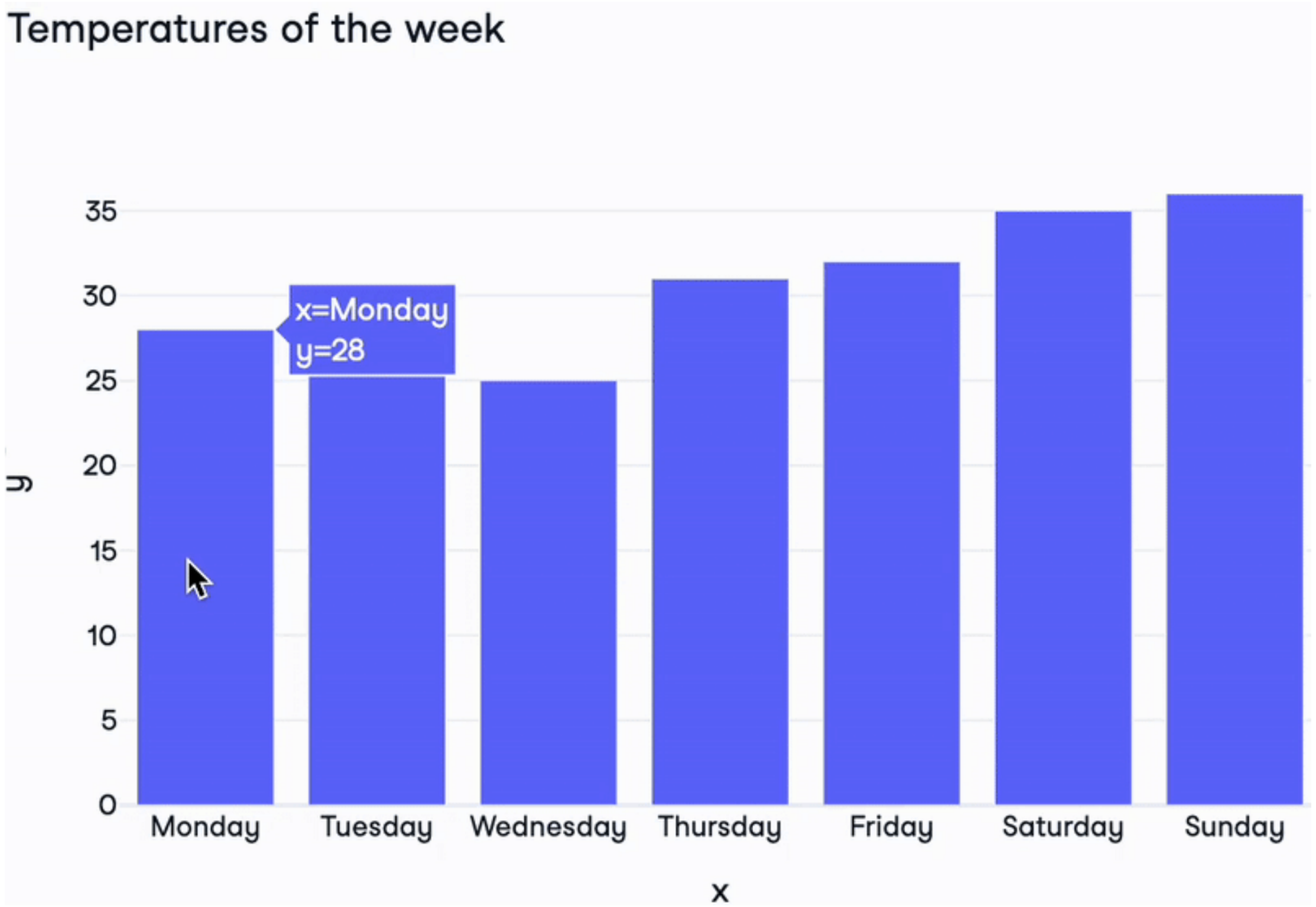
- `plotly.express` **page** with examples
- **Technical reference** for the underlying object

Creating our Figure

A basic plotly figure:

```
import plotly.express as px
days = ["Monday", "Tuesday", "Wednesday", "Thursday",
        "Friday", "Saturday", "Sunday"]
temperatures = [28, 27, 25, 31, 32, 35, 36]
fig = px.bar(
    x=days,
    y=temperatures,
    title="Temperatures of the week")
fig.show()
```

Our Figure revealed



The Plotly Figure

Plotly Figure components:

- `layout` : Dictionary controlling style of the figure
  - One `layout` per figure
- `data` : List of dictionaries setting graph type and data itself
  - Data + type = a `trace` . There are over 40 types!
  - Can have multiple traces per graph
- `frames` : For animated plots (beyond this course)

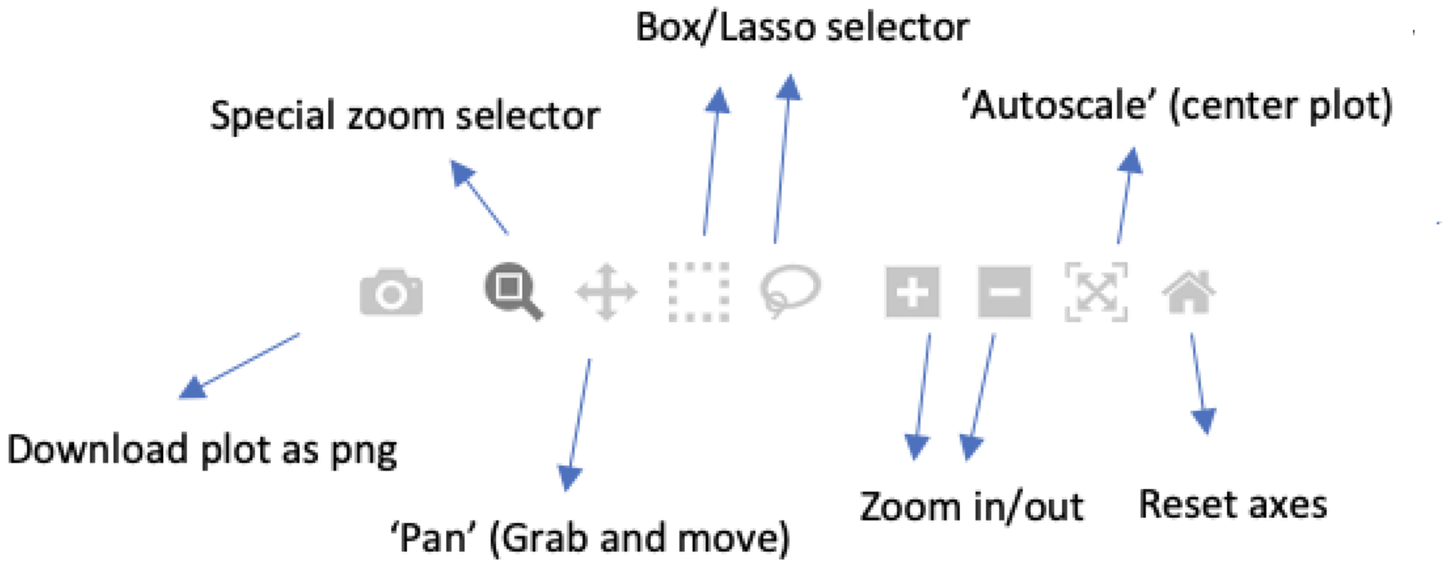
Inside a Plotly Figure

```
print(fig)
```

```
Figure({'data': [{'type': 'bar',
              'x': array(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'
              , 'Saturday','Sunday'], dtype=object),
              'y': {'bdata': 'HBsZHyAjJA==....'}],
    'layout': {'title': {'text': 'Temperatures of the week'}}})
```

Plotly's instant interactivity

- Hover over data points
- Extra interactive buttons



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Univariate visualizations

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Alex Scriven
Data Scientist

datacamp

What are univariate plots?

- Univariate plots display only one variable

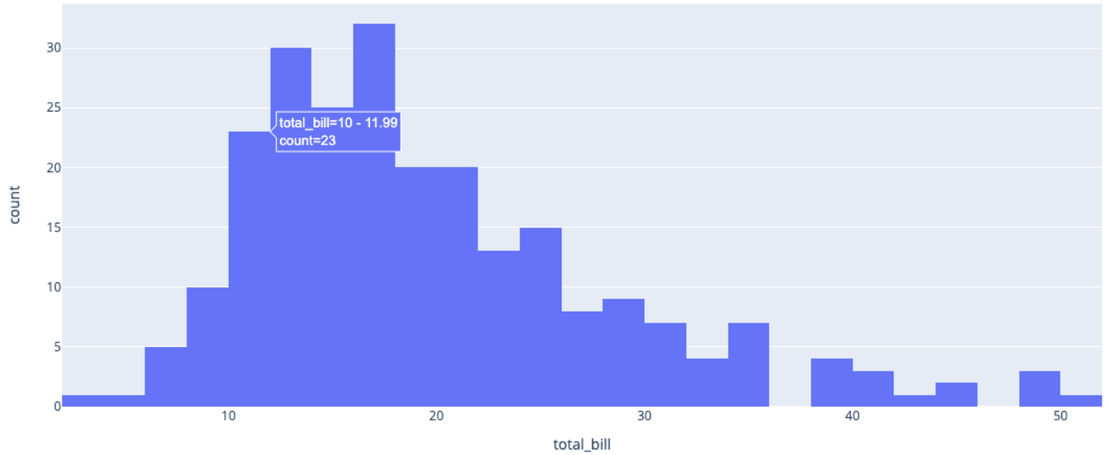
Common univariate plots:

- Bar chart
- Histogram
- Box plot
- Density plots

Histograms

Histograms have:

- Multiple columns (called "bins") representing a range of values
  - The height of each bar = count of samples within that bin range
- The number of bins can be manual or automatic



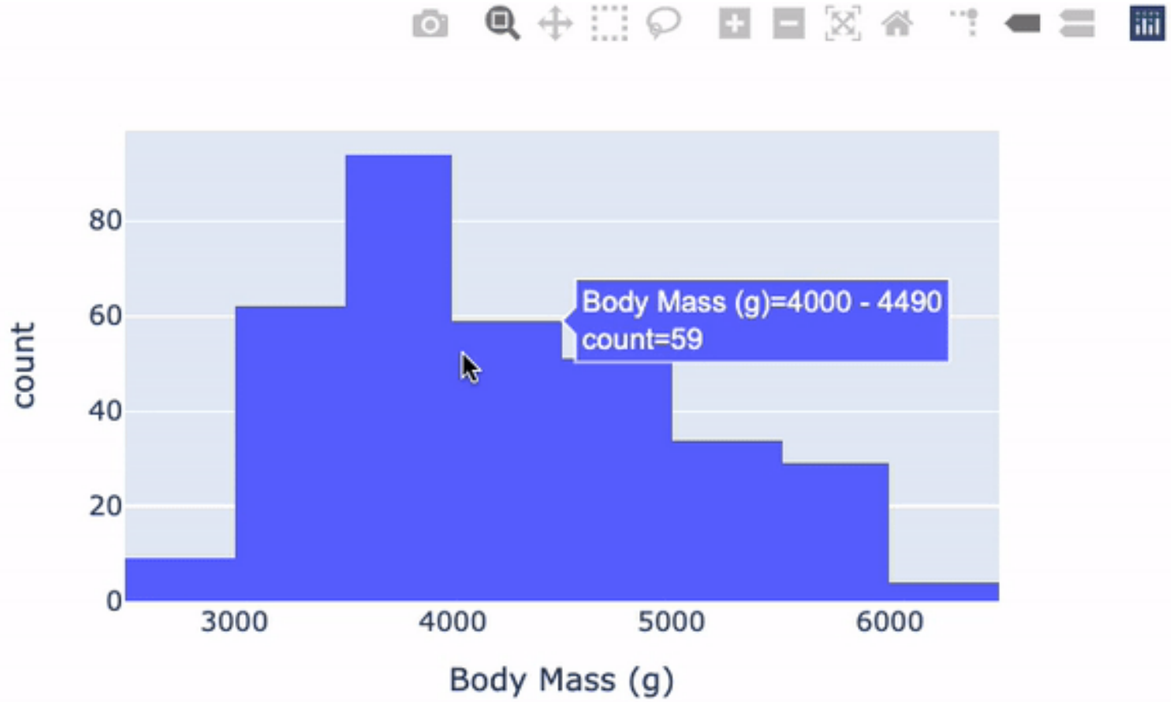
Our dataset

The dataset collected by scientific researchers on Penguins:

- Contains various body measurements like beak size, weight, etc.
- Contains different species, genders, and ages of penguins

Histograms with plotly.express

```
fig = px.histogram(
            data_frame=penguins,
            x="Body Mass (g)",
            nbins=10)
fig.show()
```



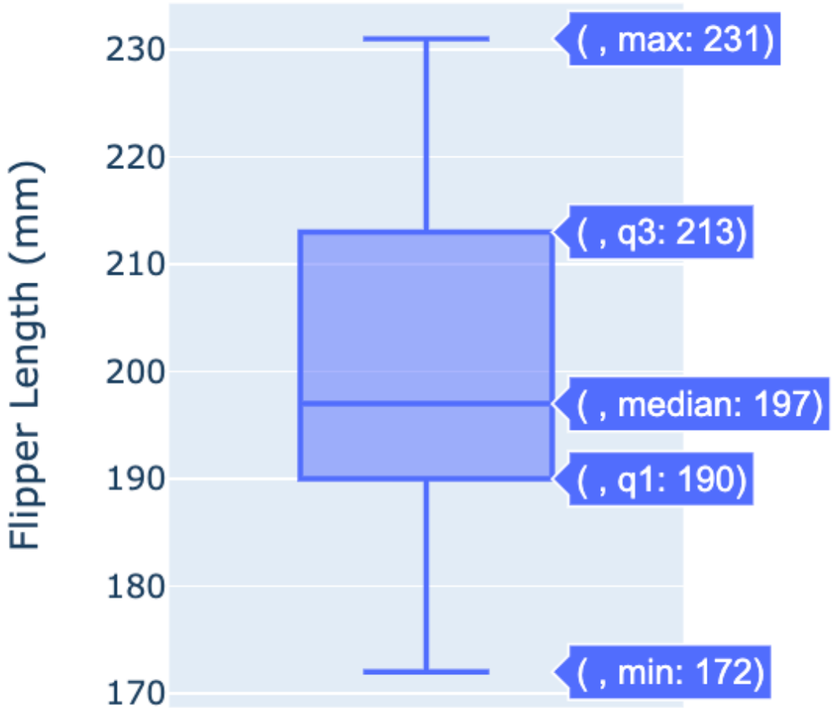
Useful histogram arguments

- `orientation` : To orient the plot vertically (`v`) or horizontally (`h`)
- `histfunc` : Set the bin aggregation (eg: average, min, max).

Check the **documentation** for more

Box (and whisker) plots

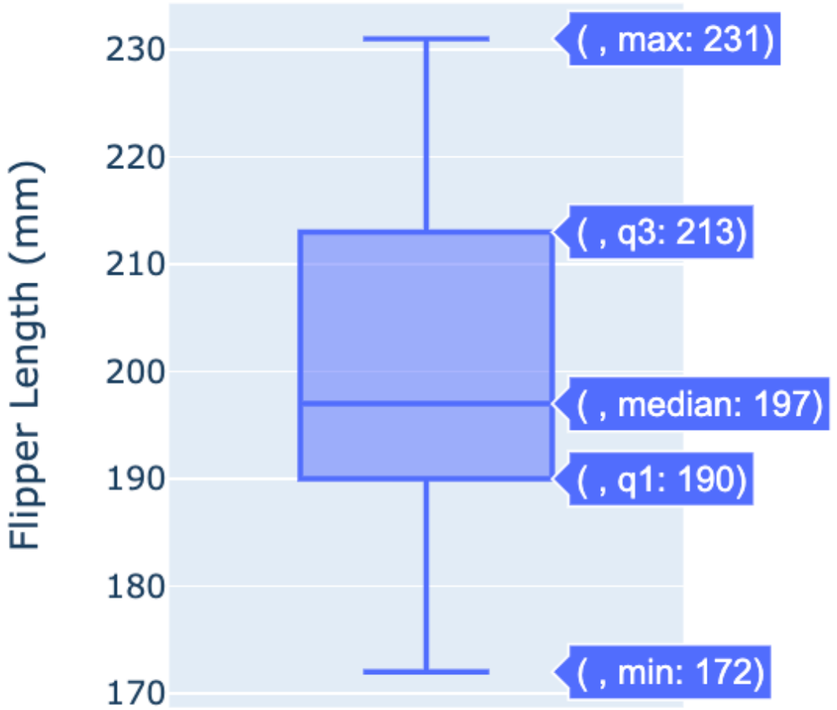
Summarizes a variable using quartile calculations



- Middle area represents *interquartile range*
  - Top line = 3rd quartile (75th percentile)
  - Middle line = median (50th percentile)
  - Bottom line = first quartile (25th percentile)
- Top/bottom bars = min/max, excluding outliers
- Outlying dots are outliers

Box plots with plotly.express

```
fig = px.box(data_frame=penguins,
             y="Flipper Length (mm)")
fig.show()
```



Useful box plot arguments

- `hover_data` : A list of column name(s) to display on hover
  - Useful to understand outliers
- `points` : Further specify how to show outliers

Check the **documentation** for more

Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Customizing color

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Alex Scriven
Data Scientist

datacamp

Customization in general

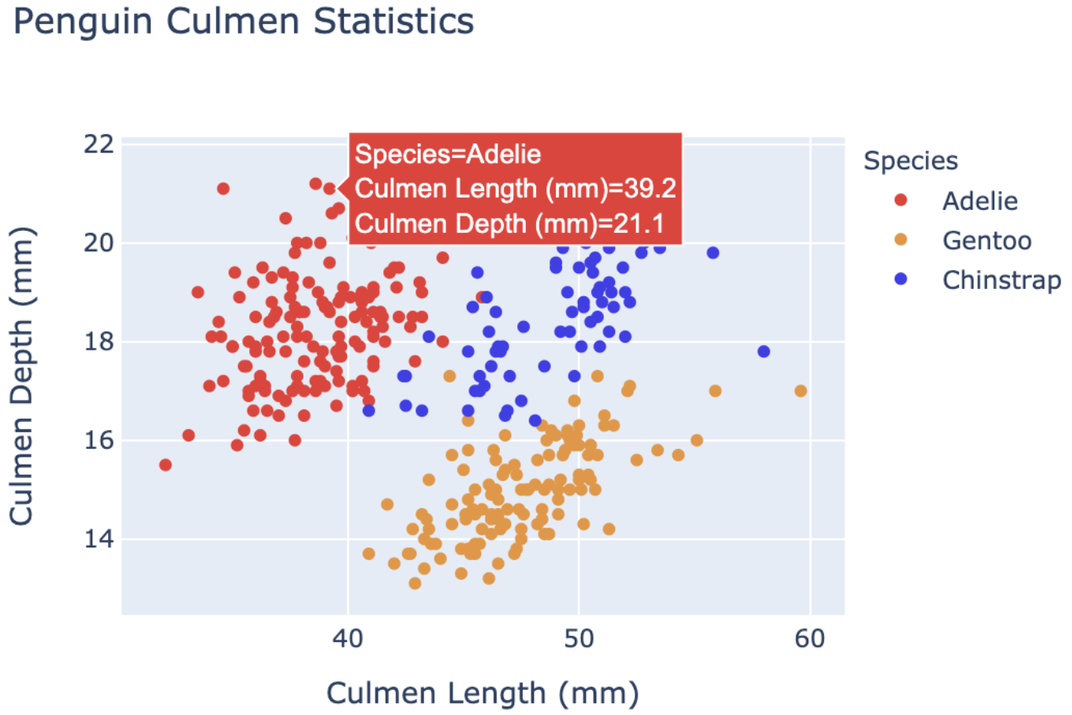
How to customize plots:

1. At plot creation, if an argument exists (like `color`)
2. After plot is created use `update_layout()`
   - Takes a dictionary
   - `fig.update_layout({"title":{"text":"A New Title"}})`

Why customize color?

Customizing color can help:

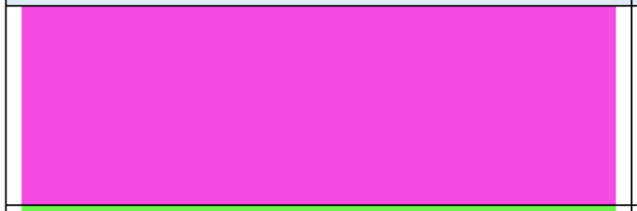
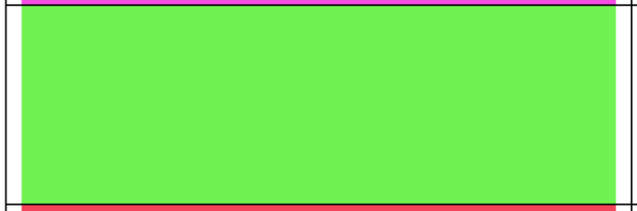
1. Make plots look awesome!
2. Convey analytical insights
   - Color in this scatterplot adds a 3rd dimension.



Some color theory

Computers use RGB encoding to specify colors:

- RGB = A 3-digit code (each 0-255) mixing **R**ed, **G**reen, **B**lue
  - (0,0,255) is blue and (255,255,0) is yellow

Color	RGB Code
	(245, 66, 230)
	(105, 245, 66)
	(245, 66, 87)
	(50, 47, 247)

See more in this **article**

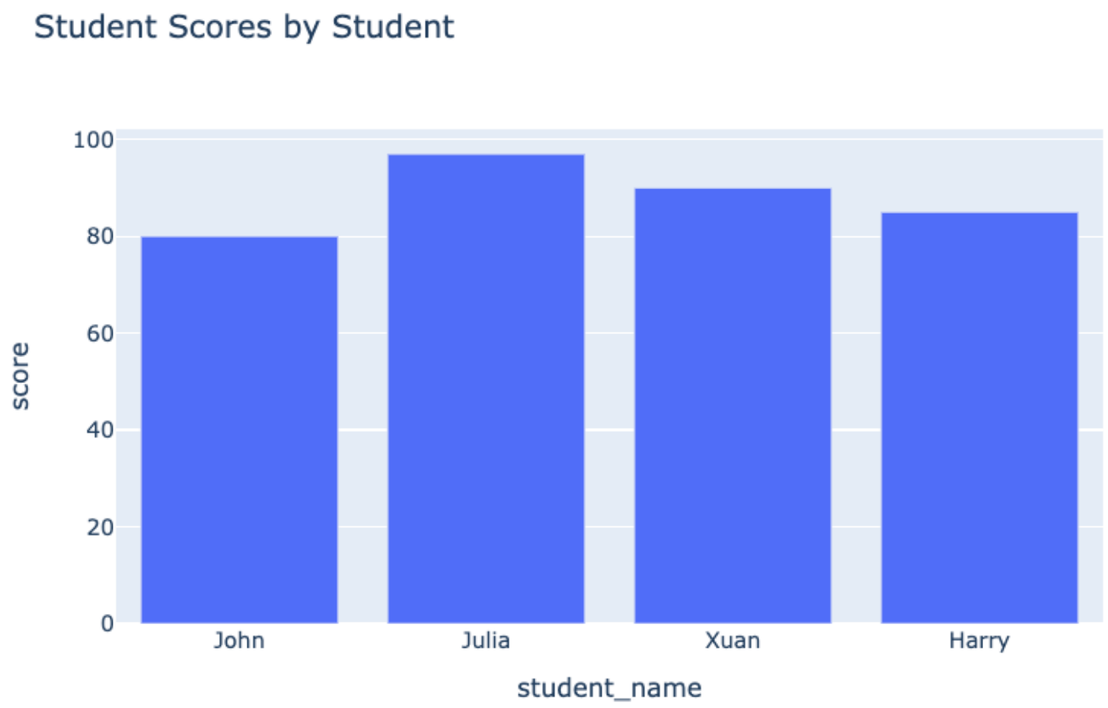
Specifying colors in plotly.express

- `color` argument (DataFrame column)
  - Each category gets a color (*automatically*)
  - A color scale is used for numerical columns

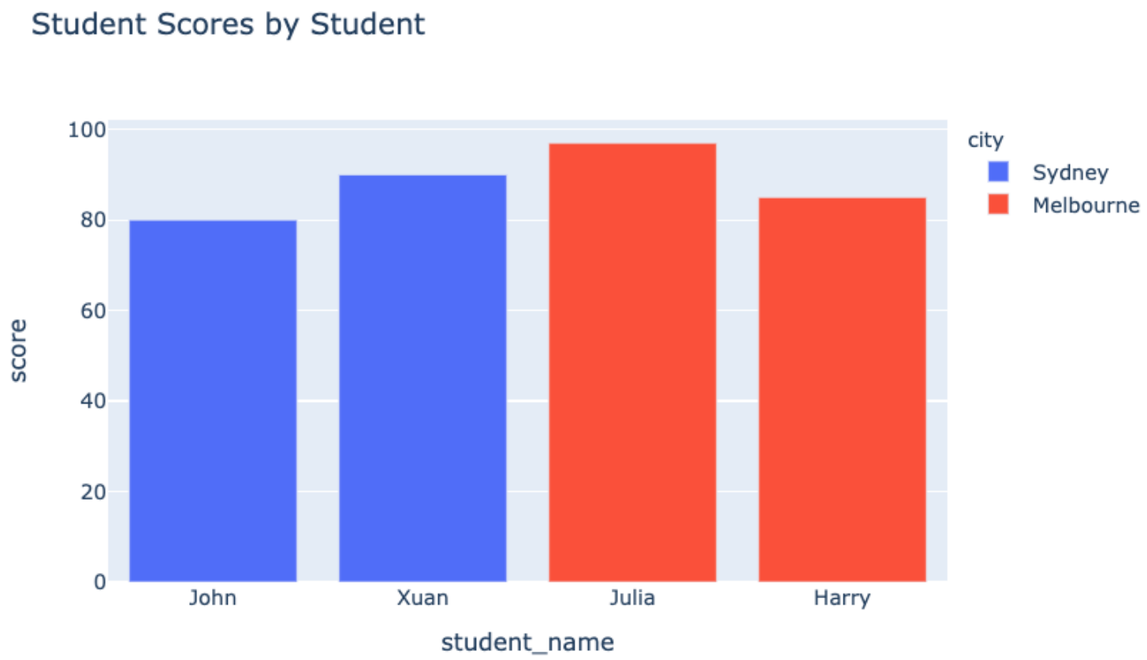
```
fig = px.bar(data_frame=student_scores,
       x="student_name",
       y="score",
       title="Student Scores by Student"
       color="city")
fig.show()
```

Our colors revealed

The plot before:



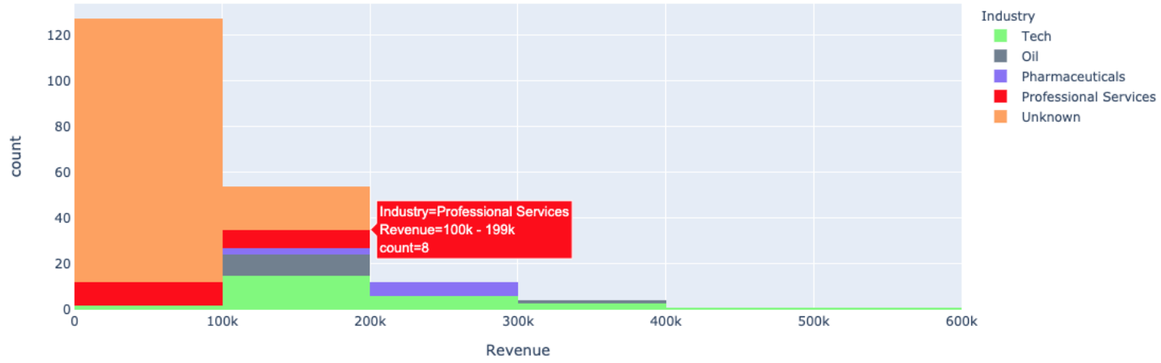
Our plot after:

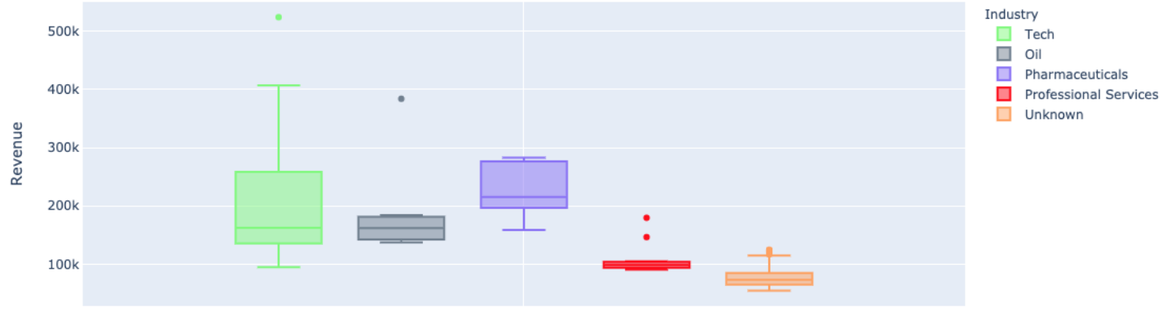


Color with univariate plots

Using `plotly.express` `color` argument with univariate (bar, histogram) plots:

- Histograms - stacked bars
- Box plots - produces multiple plots



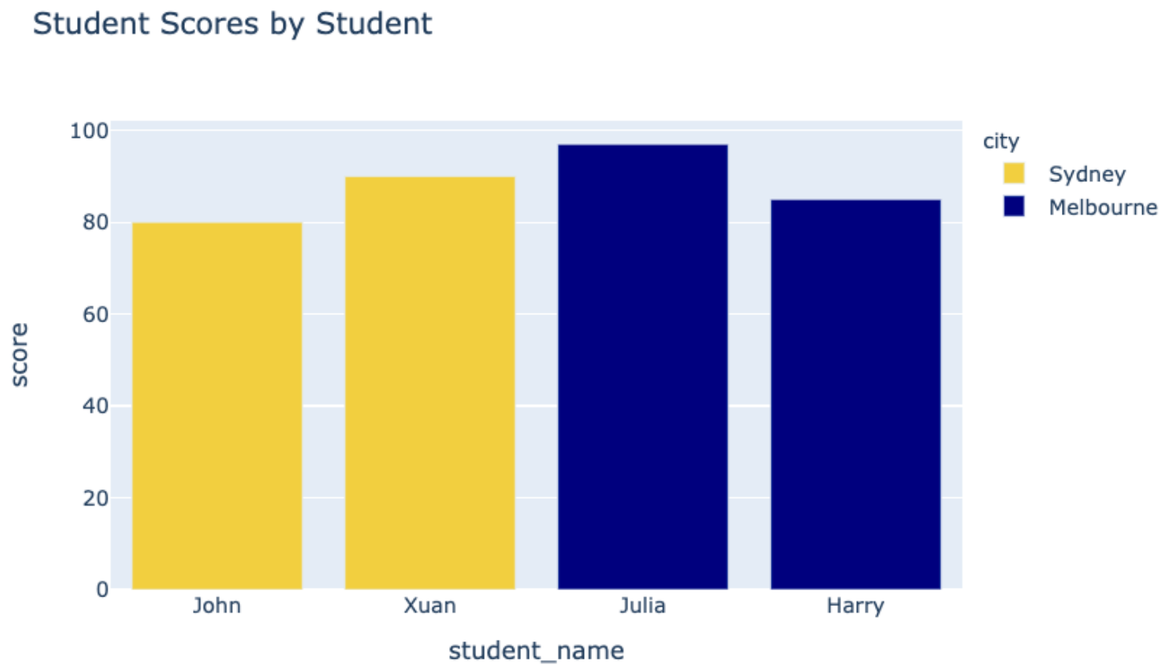


Specific colors in plotly.express

- `color_discrete_map` : A dictionary that maps categorical values to colors
- Can also express (basic) colors as strings such as `"red"` , `"green"` etc.

Our specific colors

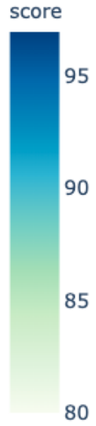
```
fig = px.bar(
    data_frame=student_scores,
    x="student_name", y="score",
    title="Student Scores by Student",
    color_discrete_map={
    "Melbourne": "rgb(0,0,128)",
    "Sydney": "rgb(235, 207, 52)"},
    color="city")
```



Color scales in plotly.express

- Single color scales (light to dark green)

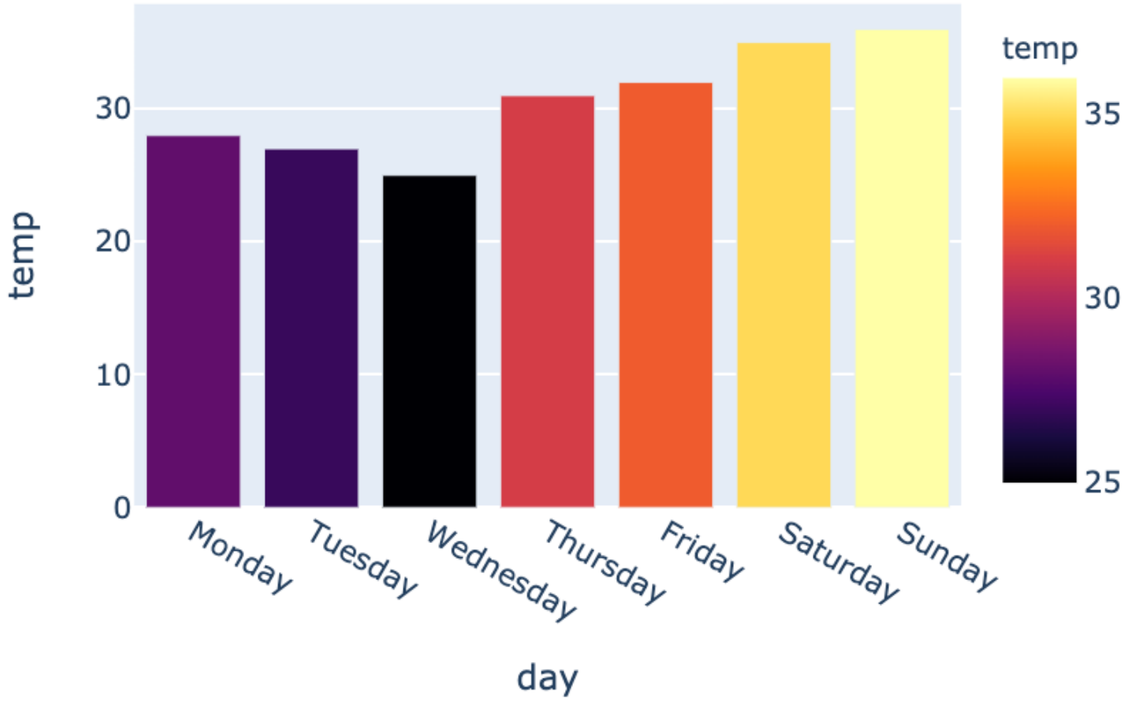
- Blended gradient (green into blue)



- `color_continuous_scale` argument for color scales.

Using built-in color scales

```
fig = px.bar(data_frame=weekly_temps,
    x="day", y="temp",
    color="temp",
    color_continuous_scale="inferno")
fig.show()
```

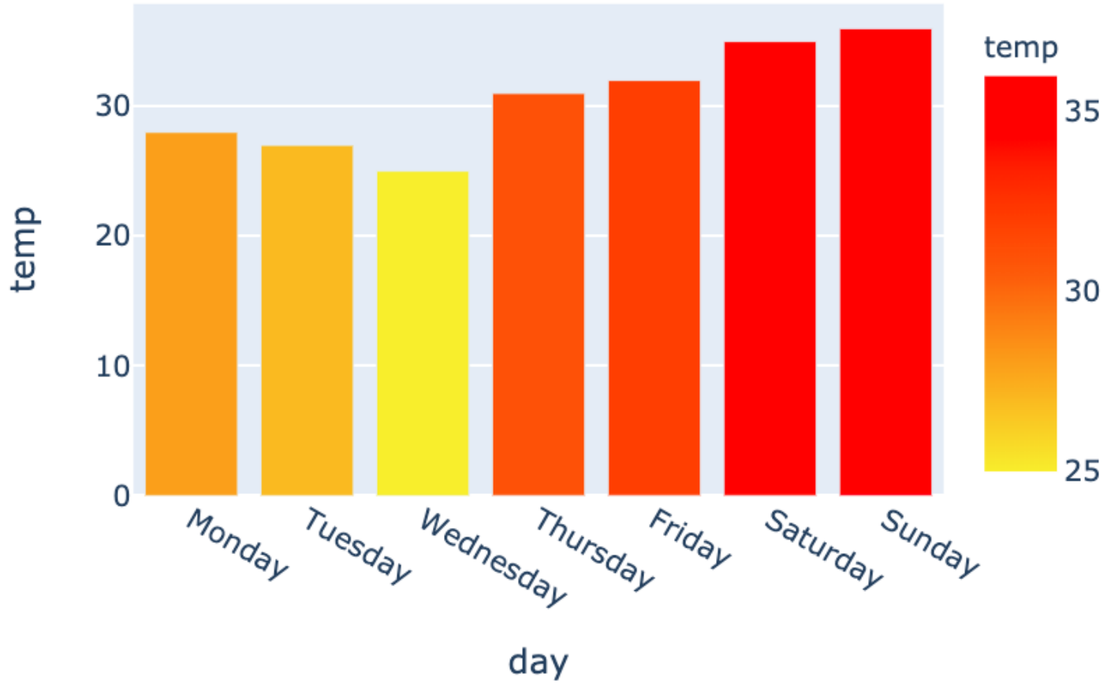


- Many **built-in scales** available

Constructing our own color range

- Custom color scale (yellow - orange - red)

```
my_scale=[("rgb(242, 238, 10)"),
          ("rgb(242, 95, 10)"),
          ("rgb(255,0,0)")]
fig = px.bar(data_frame=weekly_temps,
    x="day", y="temp",
    color_continuous_scale=my_scale,
    color="temp")
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON