Plotly and the Plotly Figure

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Alex Scriven
Data Scientist

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 implement simple plots
- Low code/low effort options using `plotly.express`
- (If desired) Extremely customizable
- Interactive plots by default

Creating Plotly Figures

Plotly graphs can be created:

1. With `plotly.express` for simple, quick plots (`px`)
2. With `plotly.graph_objects` (`go`) for more customization
3. With `plotly.figure_factory` for specific, advanced figures

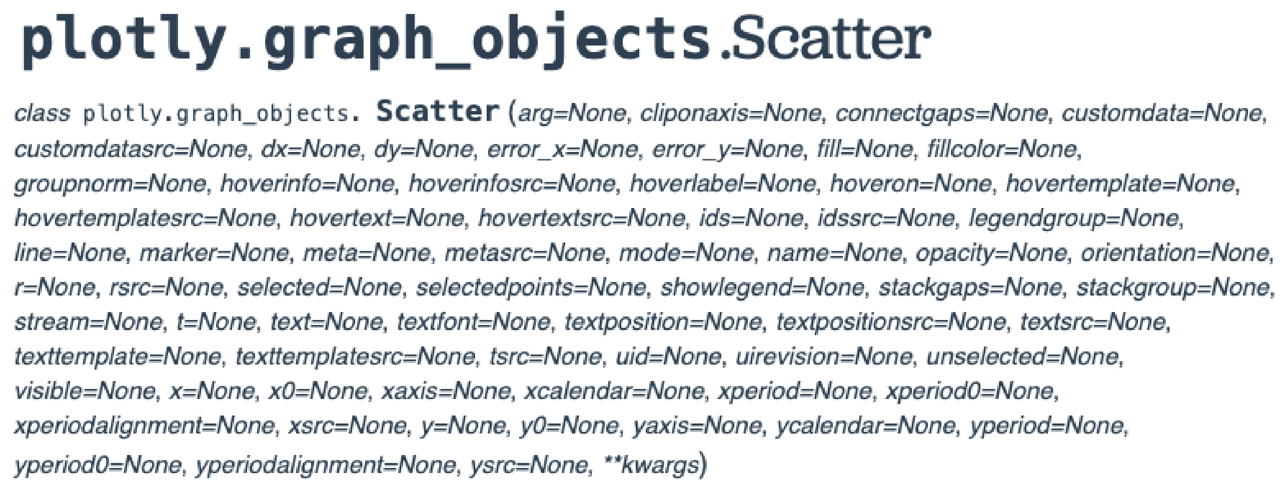
We will spend most of our time on **1** and **2**!

The importance of documentation

Save the links to key documentation!

1. Interactive, introductory docs (with many examples!)
   - **https://plotly.com/python**
2. Graph_objects pages for specific plots
   - Index **here**
   - For example, `go.scatter` **here**
3. The base `go.Figure` documentation linked **here**
   - Important when we cover `update_layout()` later!

The `go.scatter` documentation page:



plotly.graph_objects.Scatter

class `plotly.graph_objects.` **`Scatter`** *(arg=None, cliponaxis=None, connectgaps=None, customdata=None, customdatasrc=None, dx=None, dy=None, error_x=None, error_y=None, fill=None, fillcolor=None, groupnorm=None, hoverinfo=None, hoverinfosrc=None, hoverlabel=None, hoveron=None, hovertemplate=None, hovertemplatesrc=None, hovertext=None, hovertextsrc=None, ids=None, idssrc=None, legendgroup=None, line=None, marker=None, meta=None, metasrc=None, mode=None, name=None, opacity=None, orientation=None, r=None, rsrc=None, selected=None, selectedpoints=None, showlegend=None, stackgaps=None, stackgroup=None, stream=None, t=None, text=None, textfont=None, textposition=None, textpositionsrc=None, textsrc=None, texttemplate=None, texttemplatesrc=None, tsrc=None, uid=None, uirevision=None, unselected=None, visible=None, x=None, x0=None, xaxis=None, xcalendar=None, xperiod=None, xperiod0=None, xperiodalignment=None, xsrc=None, y=None, y0=None, yaxis=None, ycalendar=None, yperiod=None, yperiod0=None, yperiodalignment=None, ysrc=None, **kwargs)*

The Plotly Figure

A Plotly Figure has 3 main 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

Let's see inside an example Plotly `figure` object:

```
print(fig)
```

```
Figure({'data': [{'type': 'bar',
         'x': [Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday],
          'y': [28, 27, 25, 31, 32, 35, 36]}],
     'layout': {'template': '...',
                'title': {'font': {'color': 'red', 'size': 15},
                 'text': 'Temperatures of the week', 'x': 0.5}}})
```

- What do you think this graph will look like?

Inside our Figure

```
Figure({ 'data': [{'type': 'bar',
      'x': [Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday],
      'y': [28, 27, 25, 31, 32, 35, 36]}],
      'layout': {'template': '...','title': {'font': {'color': 'red', 'size': 15},
      'text': 'Temperatures of the week', 'x': 0.5}}})
```

- Type 'bar'
- An X and Y axis with data noted
- A title with some text around temperatures of the week

Guess: A bar chart of temperatures of the days of the week

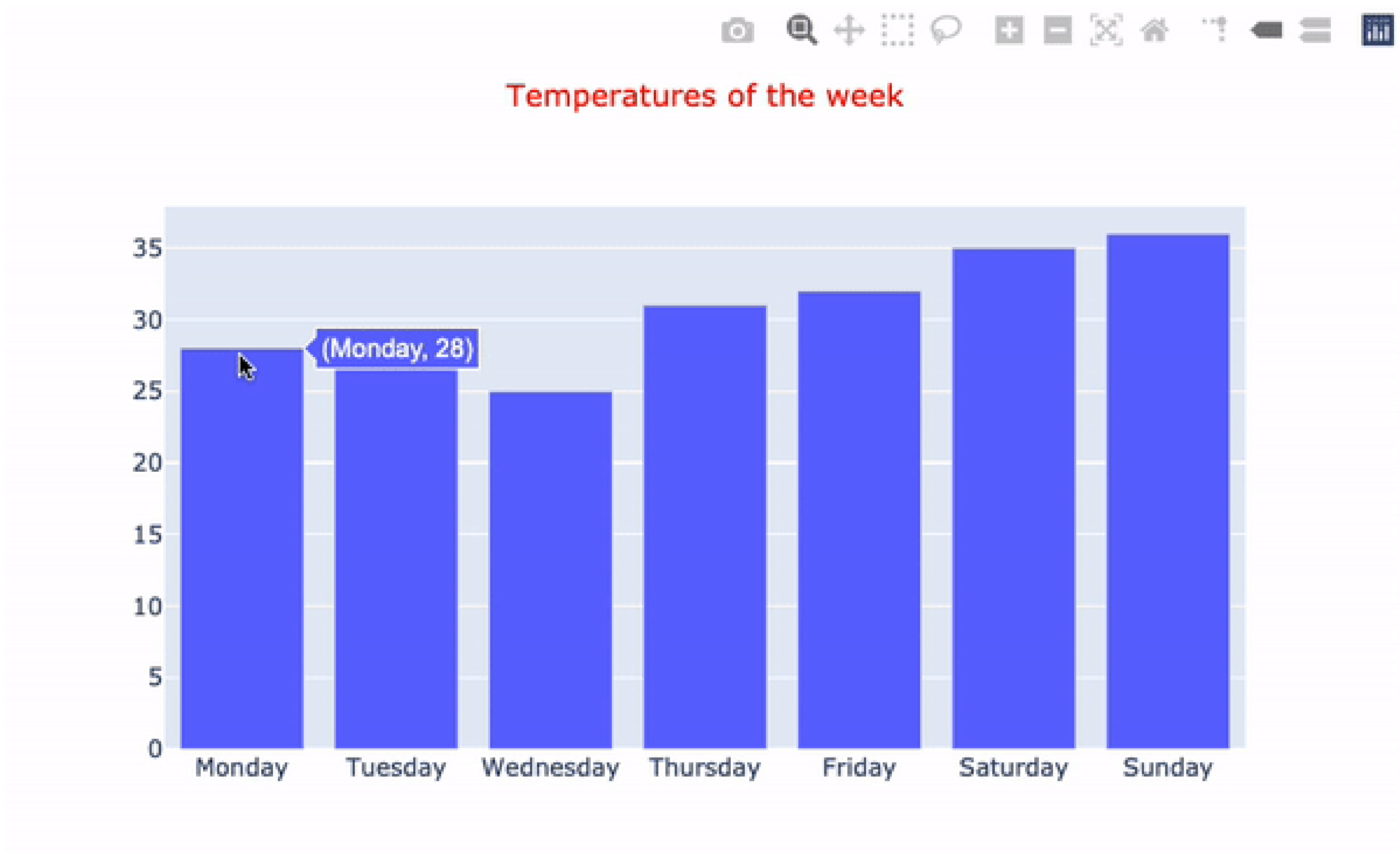
Creating our Figure

Constructing this figure from scratch (just this once!):

```
import plotly.graph_objects as go
figure_config = dict({ "data": [{"type": "bar",
              "x": ["Monday", "Tuesday", "Wednesday",
              "Thursday", "Friday", "Saturday", "Sunday"],
              "y": [28, 27, 25, 31, 32, 35, 36]}],
            "layout": {"title": {"text": "Temperatures of the week",
            "x": 0.5, "font": {'color': 'red', 'size': 15}}}})
fig = go.Figure(figure_config)
fig.show()
```

Our Figure revealed

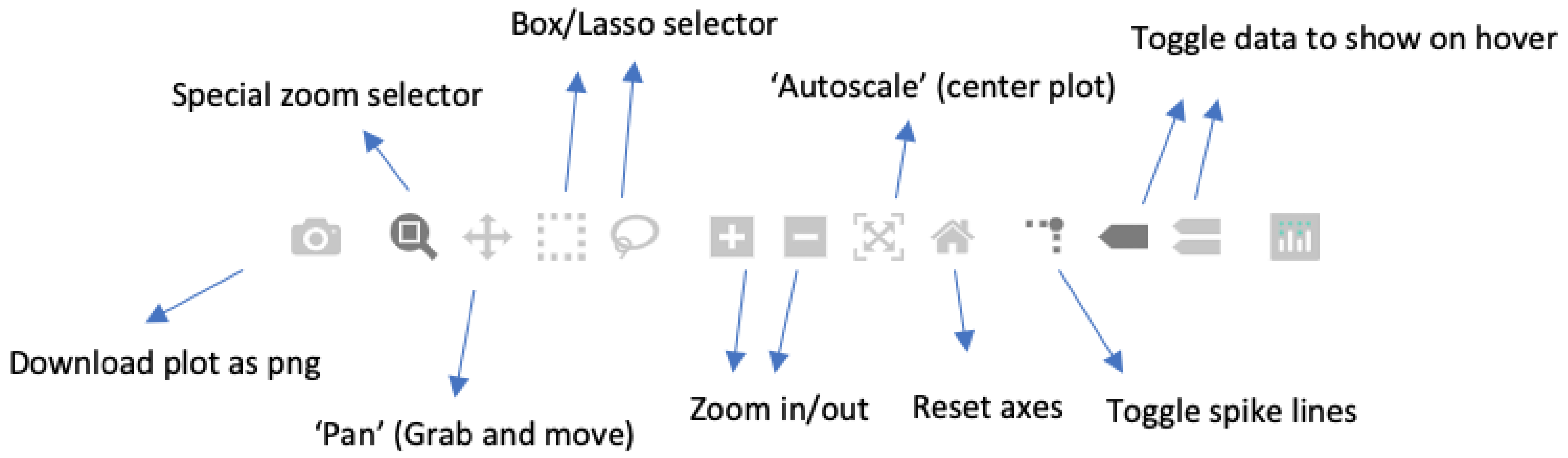
Let's see what is produced!



Plotly's instant interactivity

Plotly provides 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

Our approach

Plotly shortcut methods:

1. `plotly.express`
   - Specify a DataFrame and its columns as arguments
   - Quick, nice but less customization
2. `graph_objects` `go.X` methods (`go.Bar()` , `Go.Scatter()`) etc.
   - Many more customization options, but more code needed

What are univariate plots?

Univariate plots display only one variable

For analyzing the *distribution* of that variable

Common univariate plots:

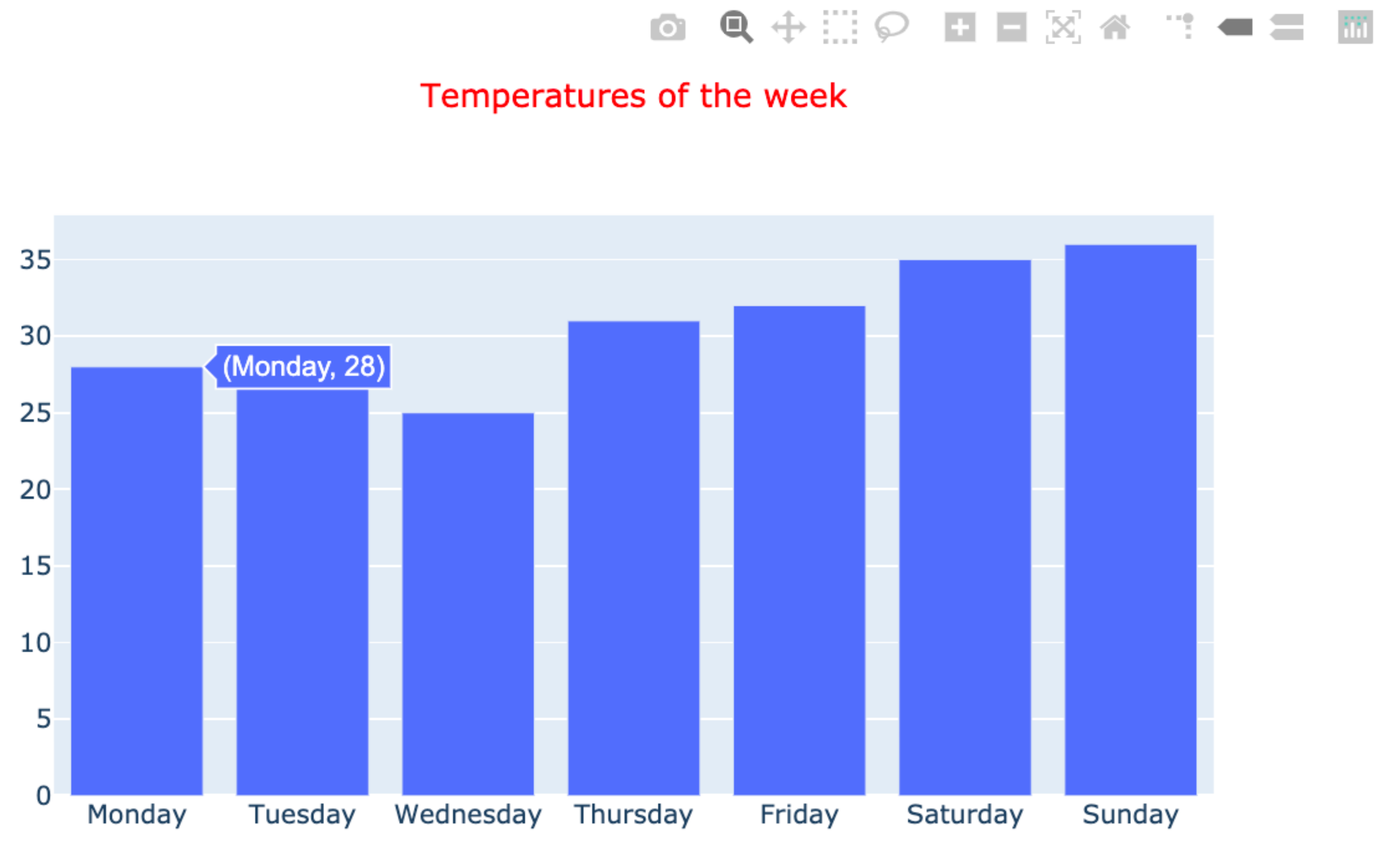
- Bar chart
- Histogram
- Box plot
- Density plots

Bar charts

Bar charts have:

- X-axis with a bar per group
  - One group = one bar! (Hence UNI-variate)
- The y-axis height represents the value of some variable

We built one in the last lesson!



Bar charts with plotly.express

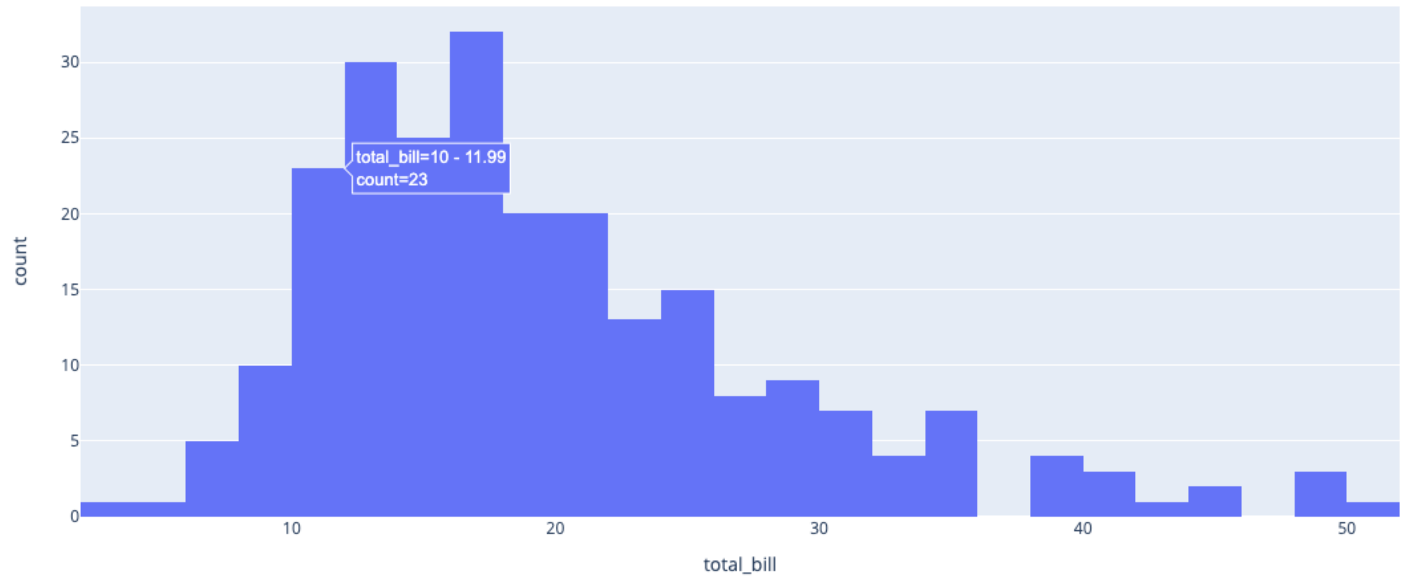
Let's rebuild with `plotly.express`

```python
import plotly.express as px
weekly_temps = pd.DataFrame({
    'day': ['Monday', 'Tuesday',
            'Wednesday', 'Thursday', 'Friday',
            'Saturday', 'Sunday'],
    'temp': [28, 27, 25, 31, 32, 35, 36]})
fig = px.bar(data_frame=weekly_temps, x='day', y='temp')
fig.show()
```

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

Dataset collected by scientific researchers on Penguins!

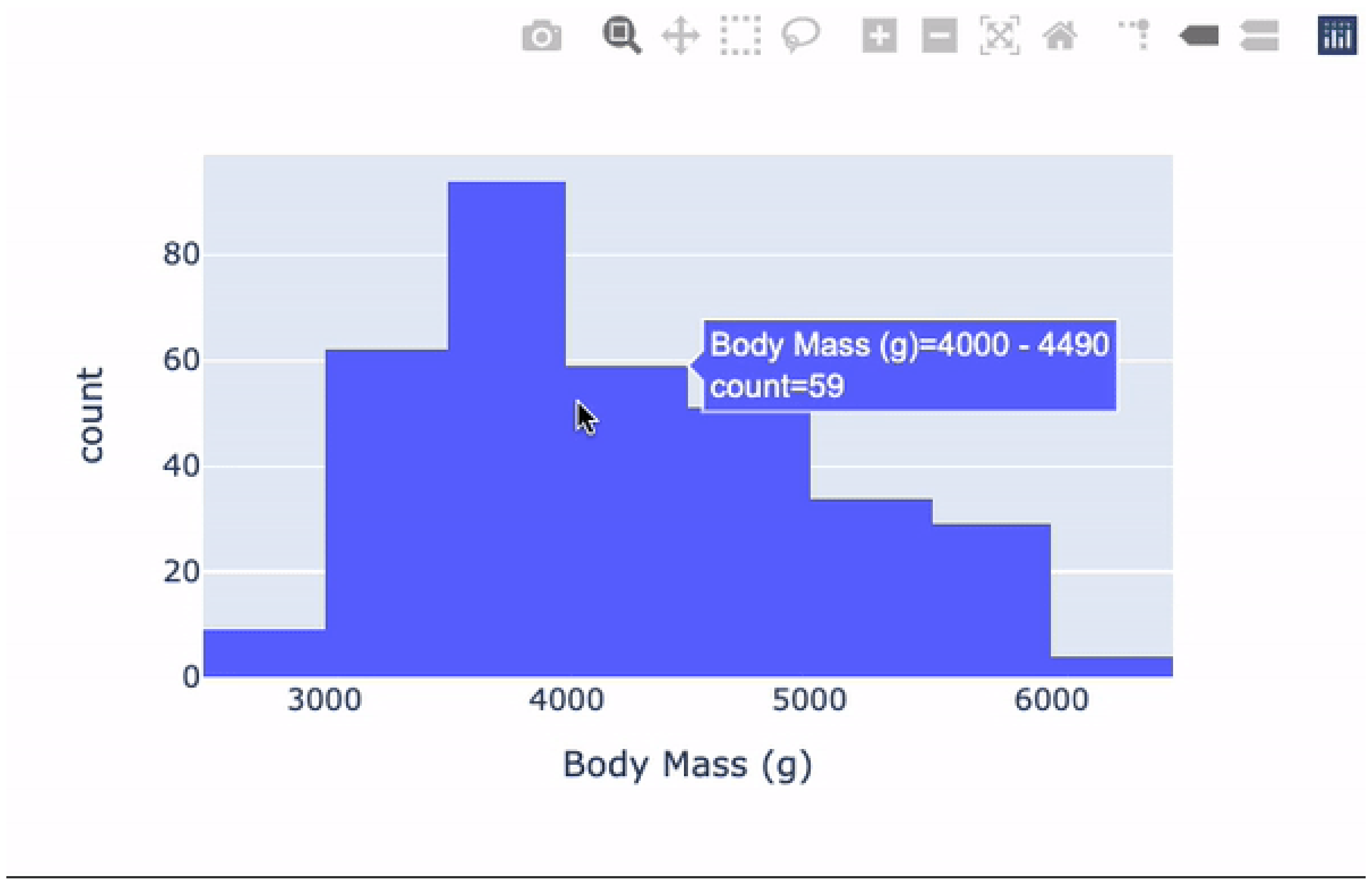
- Contains various body measurements like, beak size, weight, etc.
- Contains different species, genders, and ages of penguins

Histograms with plotly.express

We can create a simple histogram:

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

This is what is produced:



Useful histogram arguments

Other `px.histogram` arguments :

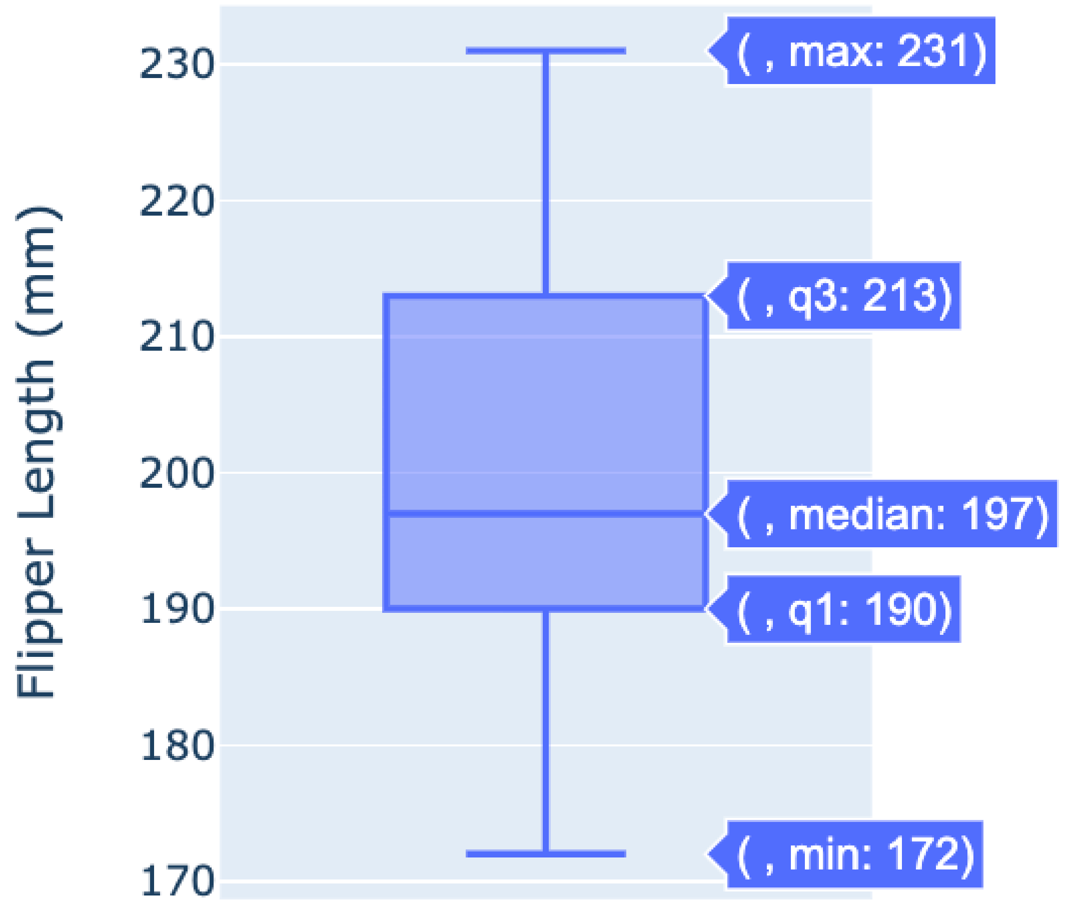
- `orientation` : To orient the plot vertically (`v`) or horizontally (`h`)
- `histfunc` : Set the bin aggregation (eg: average, min, max).

Check the **docs** for more!

Box (and whisker) plots

Summarizes a variable visually 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

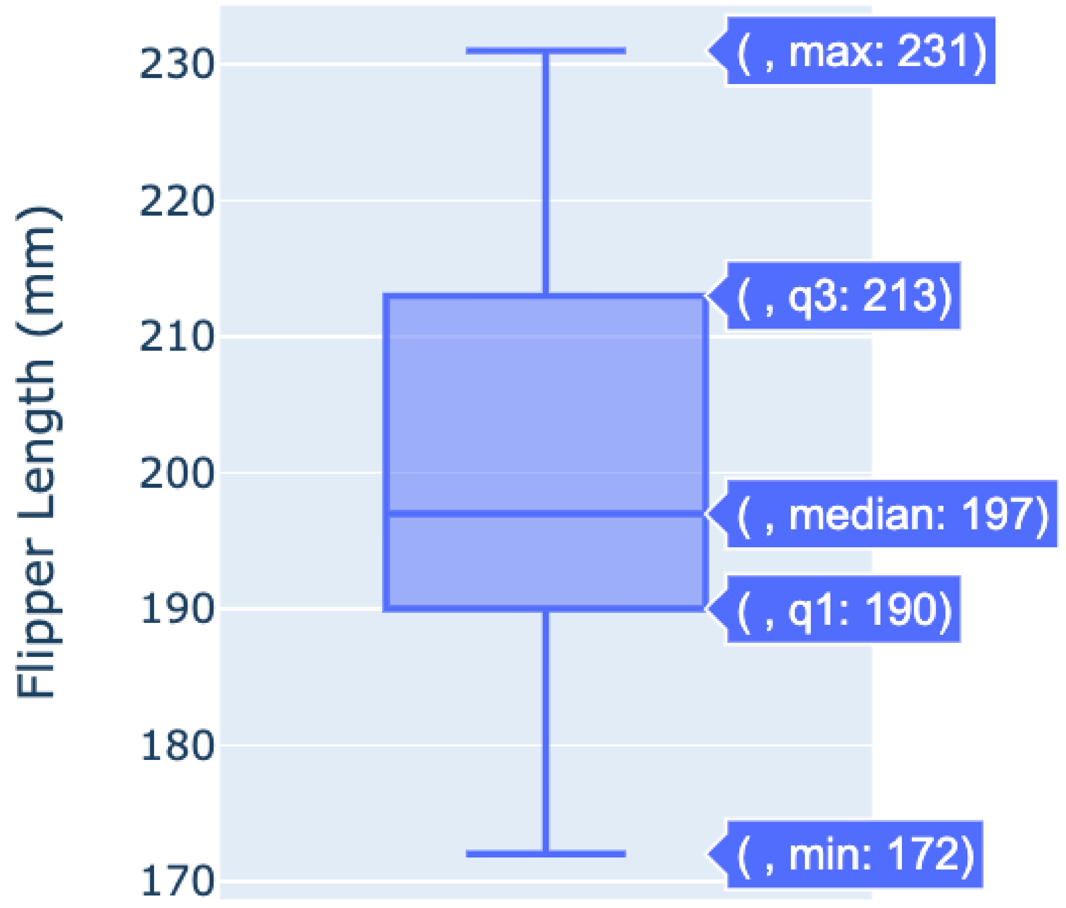


Box plots with plotly.express

Let's create a simple box plot:

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

This is what is produced:



Useful box plot arguments

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 **docs** 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

Customization in general

How to customize plots:

1. At figure creation if an argument exists (like `color` !)
2. Using an **important** function `update_layout()`
   - Takes a dictionary argument
   - E.g.: `fig.update_layout({'title':{text:'A New Title'}})`

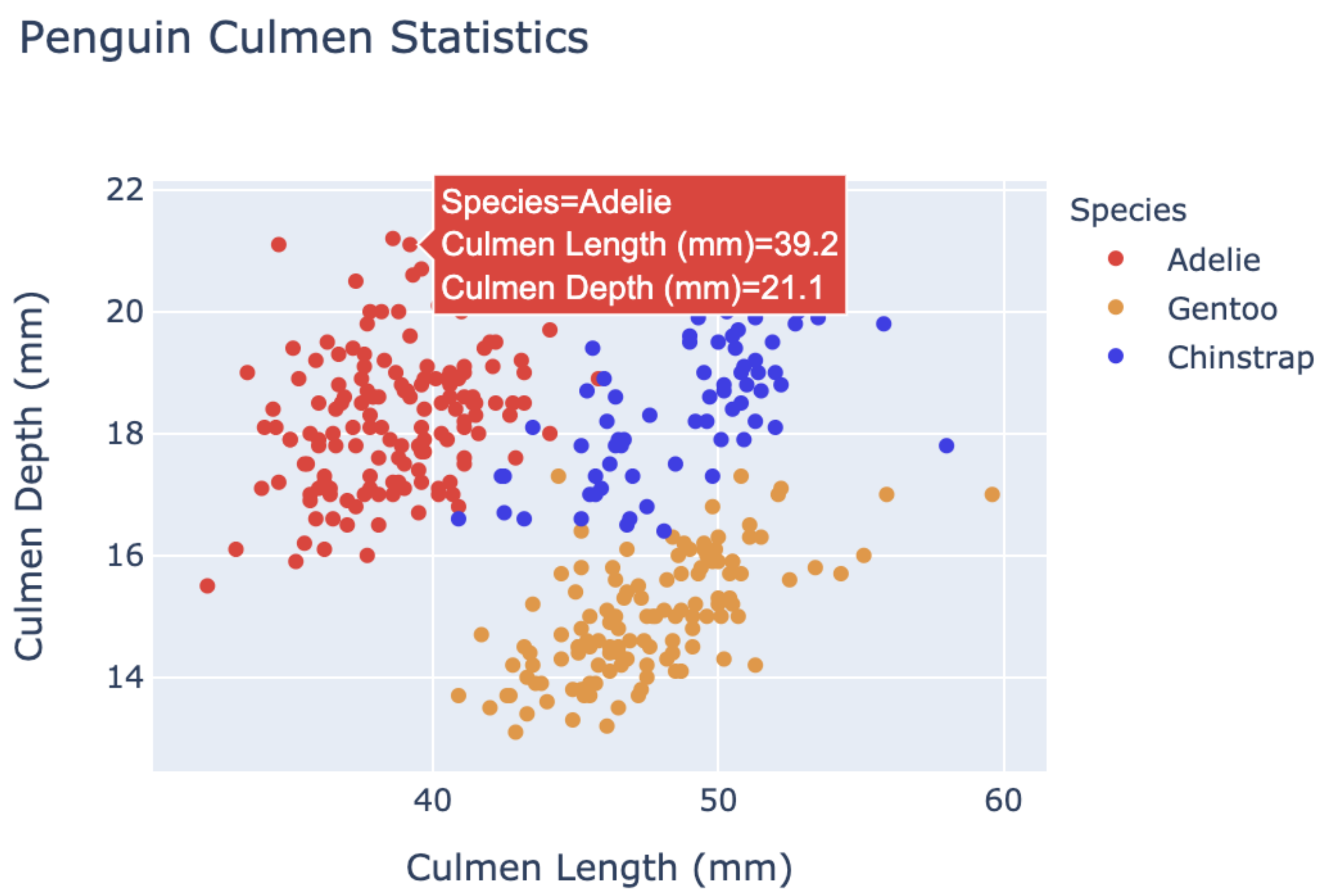
The method chosen depends on plot type how it was created.

MANY properties possible — See the **documentation**

Why customize color?

Customizing color can help you

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 together to make colors.
  - Imagine mixing Red, Green and Blue paint together!
  - (0,0,255) is totally blue and (255,255,0) is yellow

See more in **this article**

Some other examples of RGB colors:

Color	RGB Code
🟪	(245, 66, 230)
🟩	(105, 245, 66)
🟥	(245, 66, 87)
🟦	(50, 47, 247)

Specifying colors in plotly.express

In `plotly.express`:

- Often a `color` argument (DataFrame column)
  - A different (*automatic*) color given to each category in this column
  - A color scale/range is used if numerical column specified

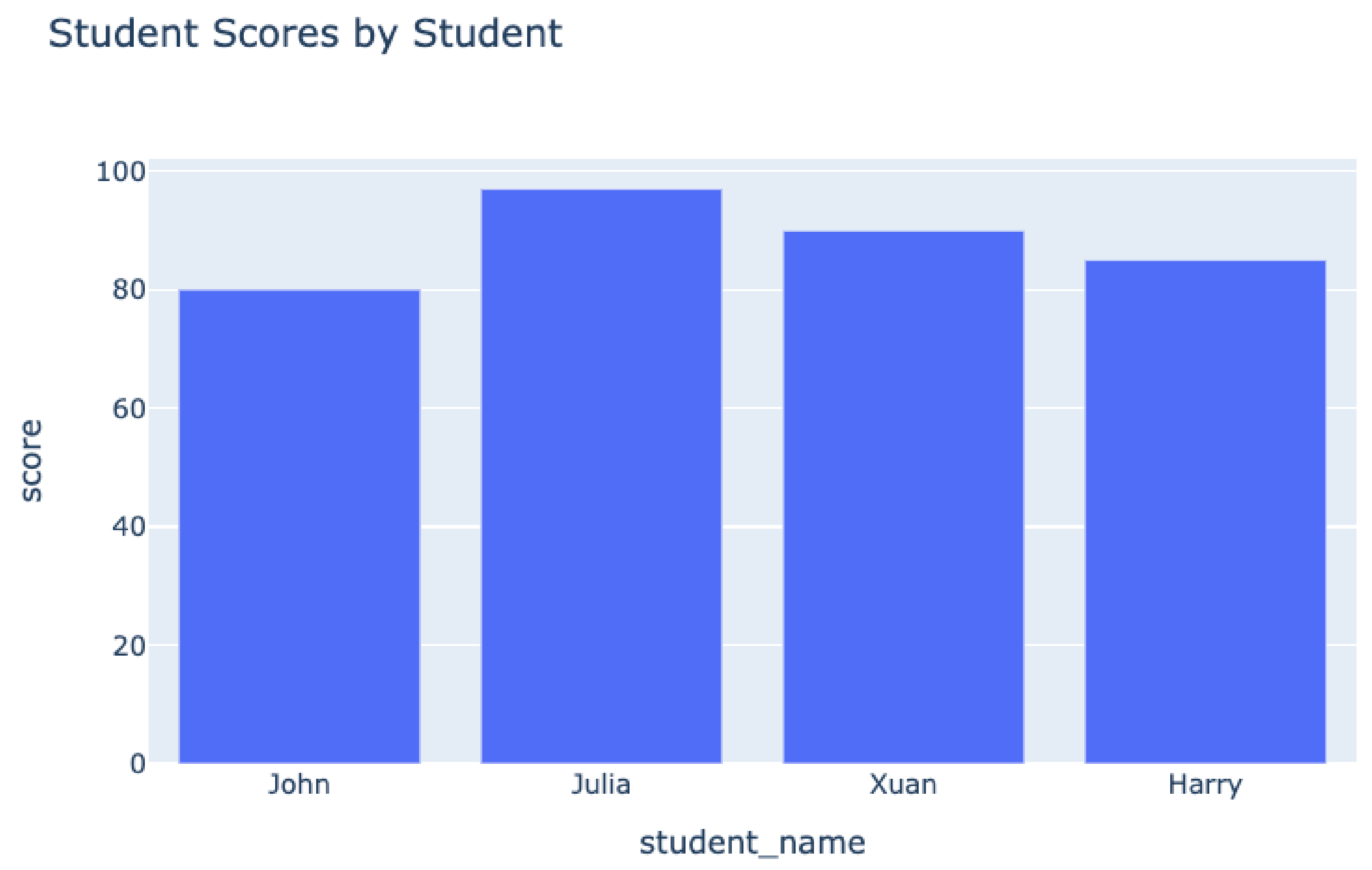
Our simple bar chart from a previous lesson (adding a `City` column)

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

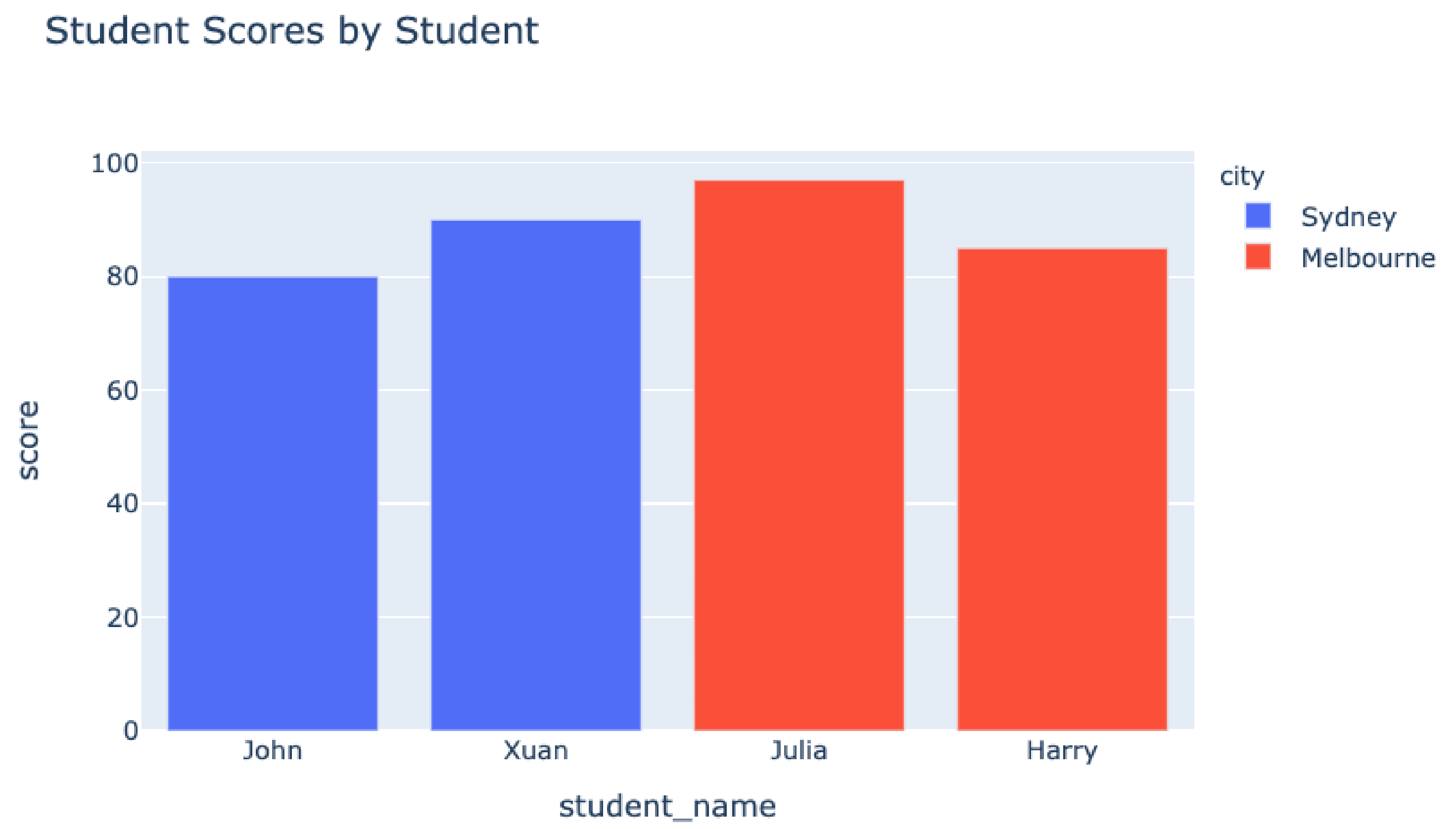
[1] Make sure to check the documentation for each figure.

Our colors revealed

The plot before:



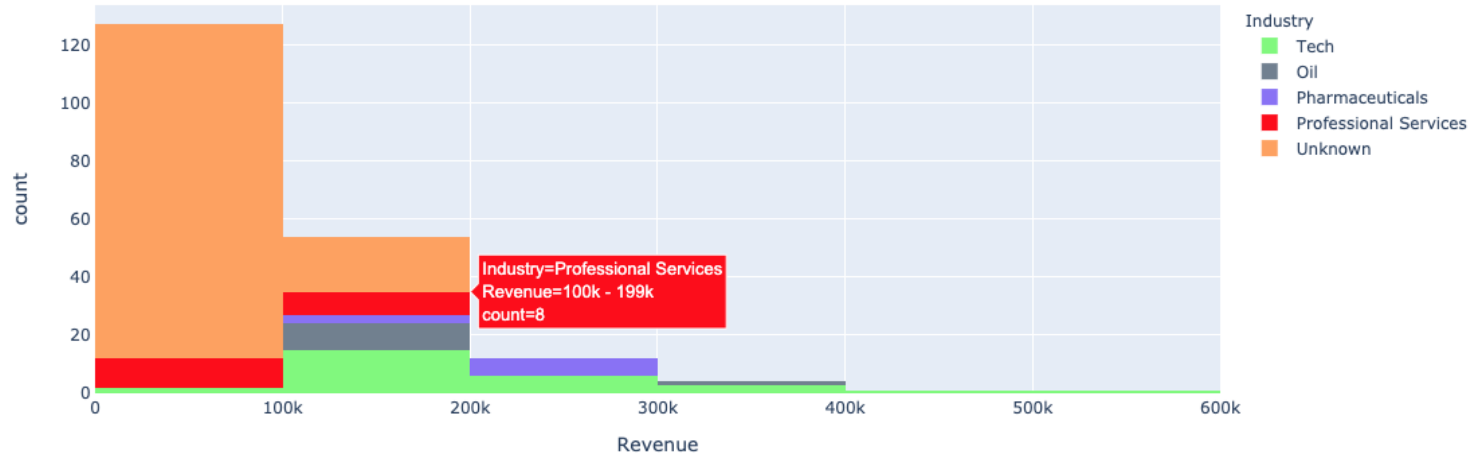
Our plot after:

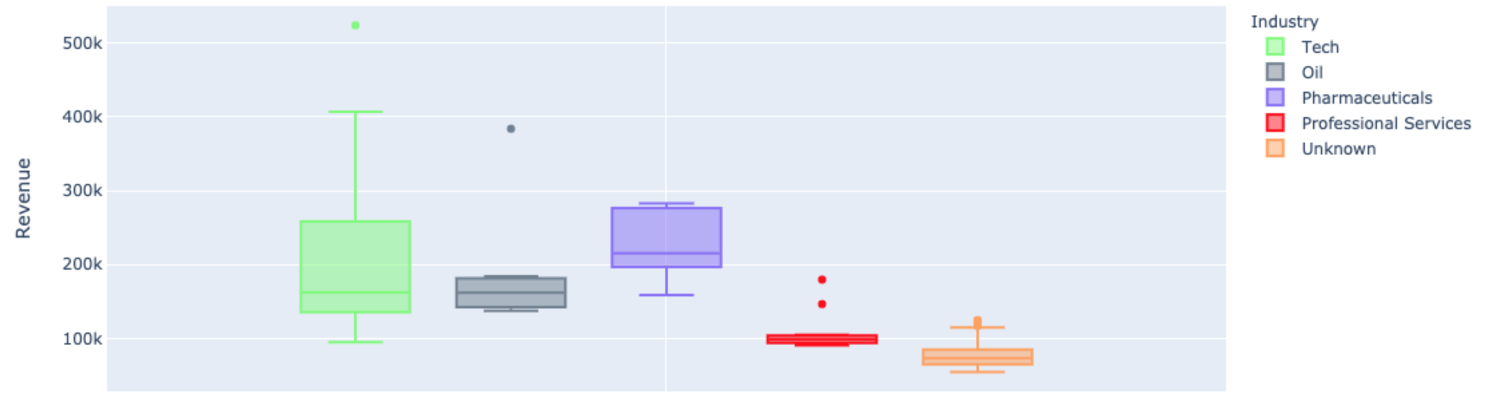


Color with univariate plots

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

- Histograms - stacked bars
- Box plots - produces multiple (one per category)





Specific colors in plotly.express

What if we don't like the automatic colors?

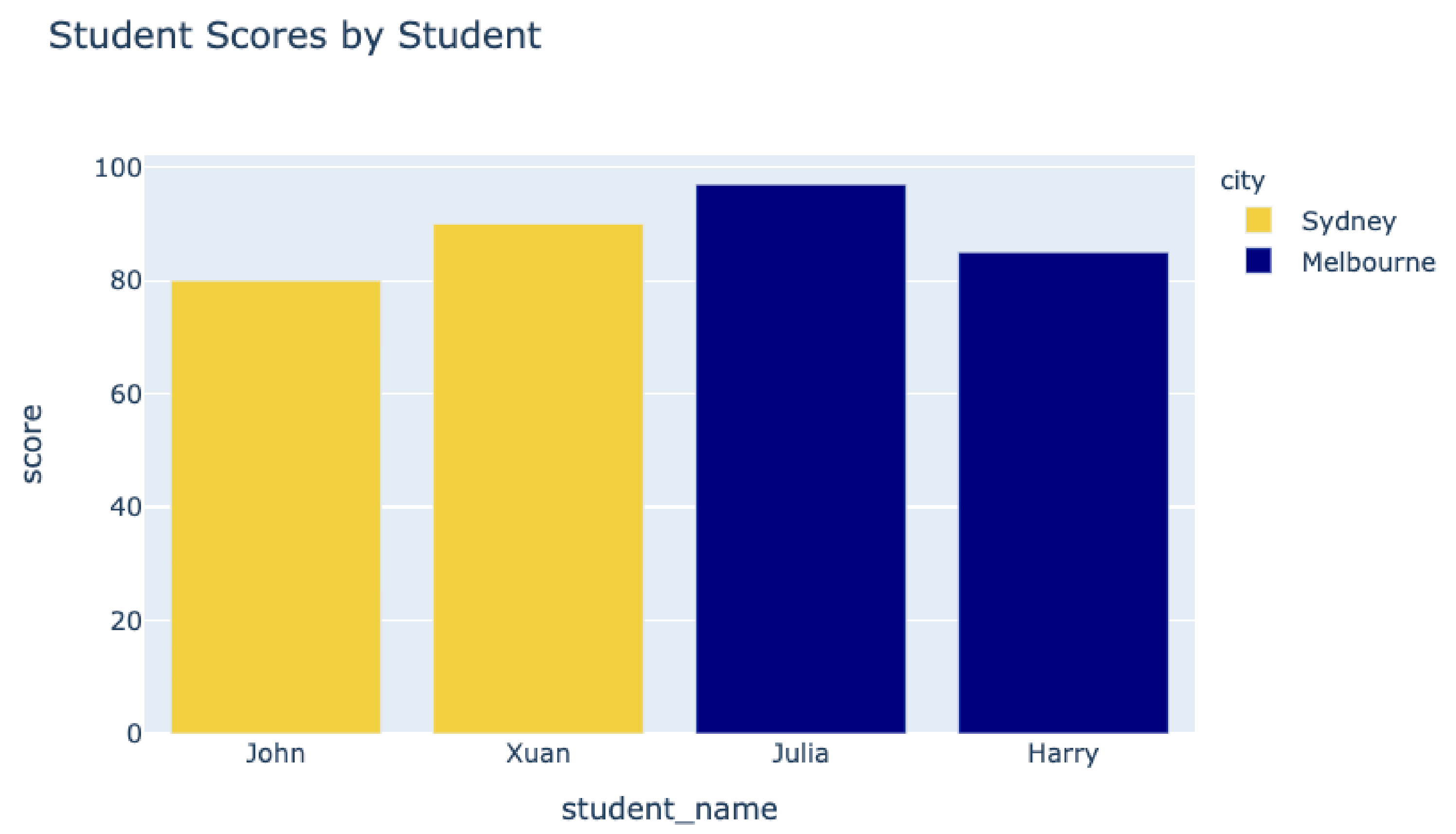
- `color_discrete_map` : A dictionary mapping specific categorical values to colors using a string RGB code specification — `'rgb(X,X,X)'`
- Can also express (basic) colors as strings such as `'red'` , `'green'` etc.

Our specific colors

Let's update our colors. Sandy yellow for 'Sydney' and navy blue for 'Melbourne'

```
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')
```

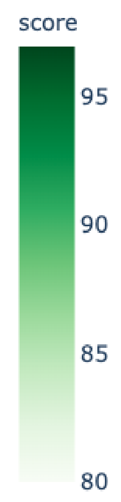
Produces:



Color scales in plotly.express

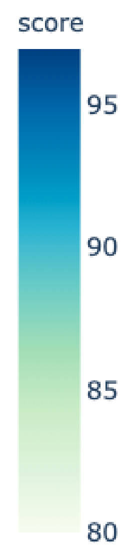
You can create color scales too.

- Single color scales. For example, light to dark green.



- Multiple colors to merge into each other. For example, green into blue.

`color_continuous_scale` allows us to do this with built-in or constructed color scales.



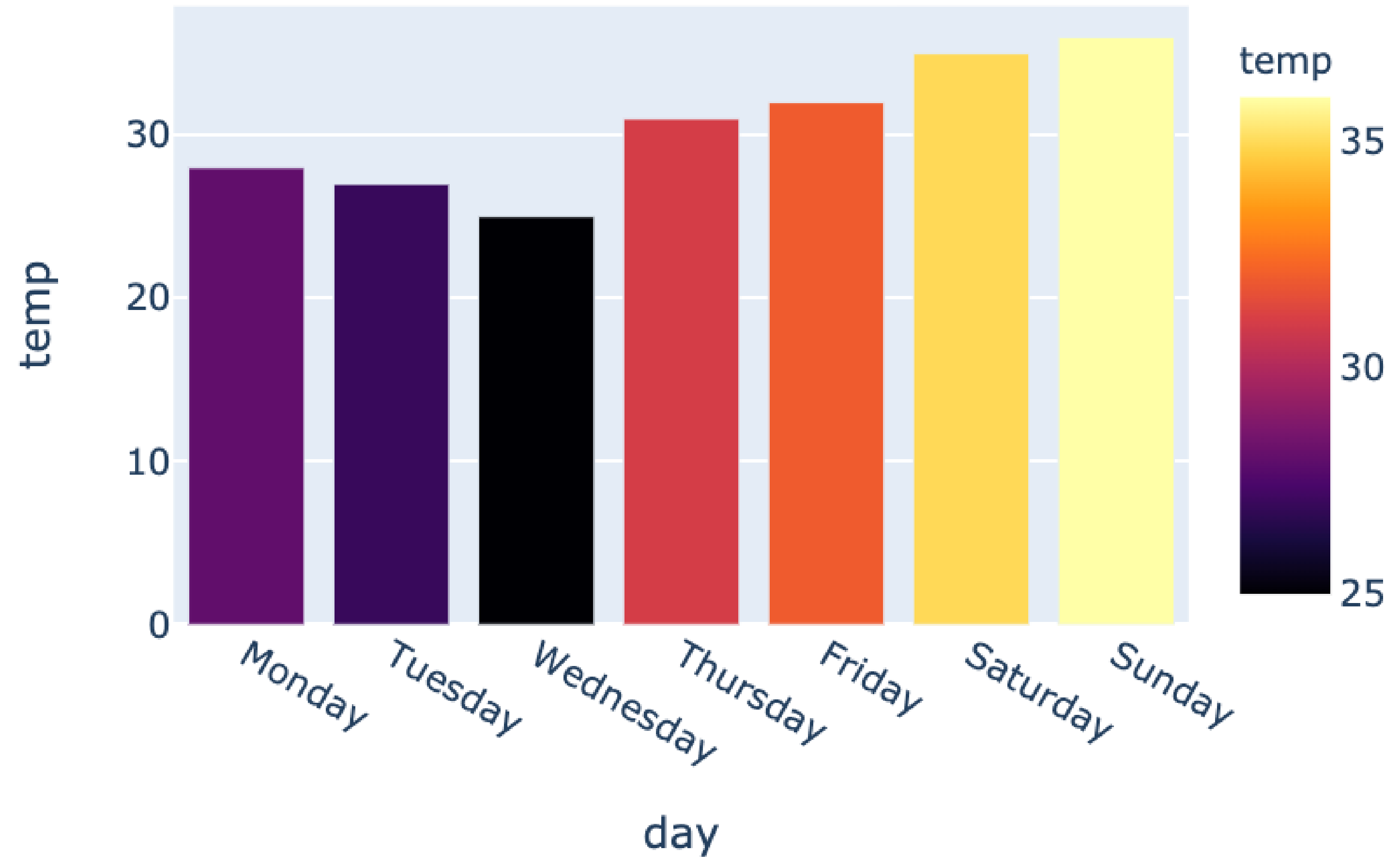
Using built-in color scales

Let's use a built-in color scale:

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

- Many **built-in scales** available

Our plot:

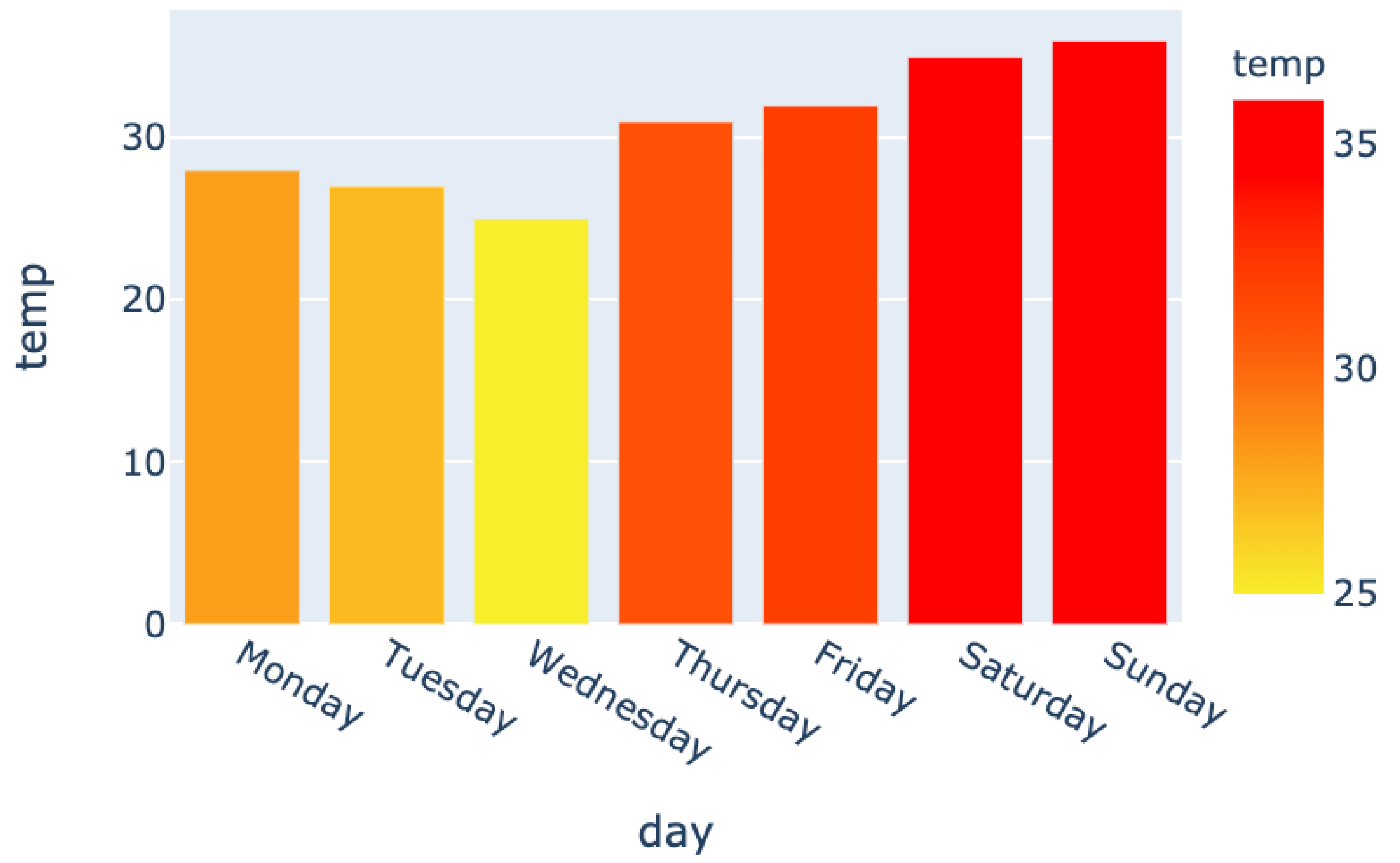


Constructing our own color range

Let's construct our own color scale - yellow through orange to 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')
```

Our plot:



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON