Custom buttons

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Alex Scriven
Data Scientist

datacamp

What can custom buttons do?

Custom buttons can:

- Update the data or layout elements of a plot
  - All of our `update_layout()` customizations could be in a button!
- Assist with animations (beyond the scope of this course)

Custom buttons in Plotly

Buttons are added via an `updatemenus` argument (a list of dictionaries) with important arguments:

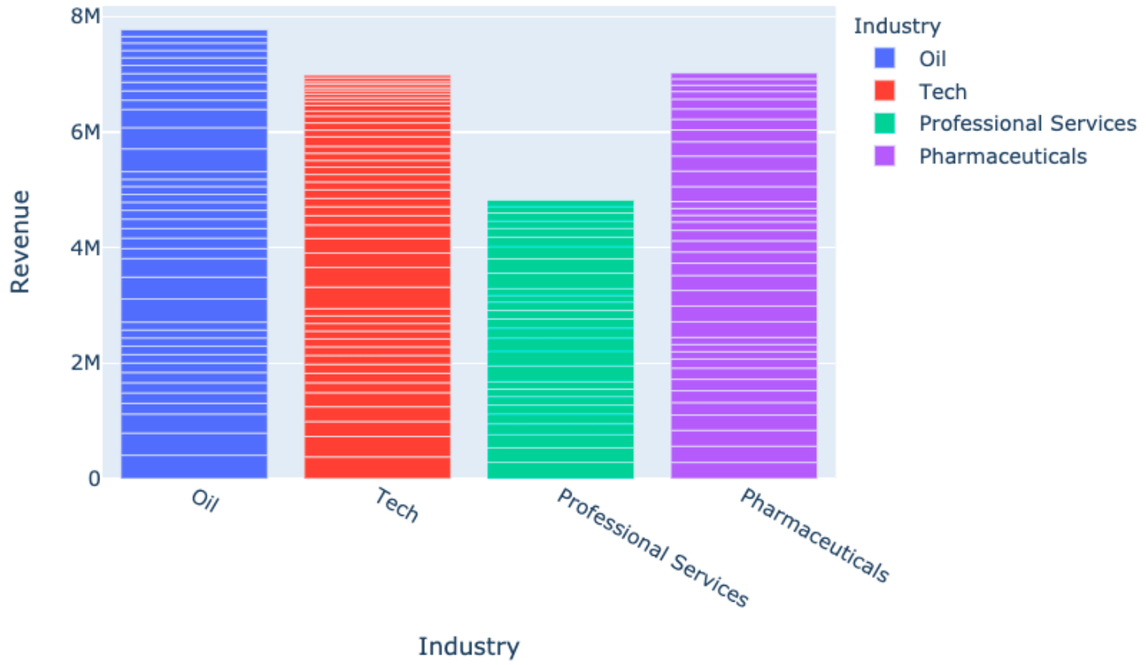
- **`type`** : `buttons` or `dropdown`
  - We will cover dropdowns later!
- **`direction`** : Button orientation
  - Buttons can be beside (`left`) or on top of (`down`) each other
- **`x` / `y`** : Floats to set the button positions as you have done before
- **`showactive`** : `True` / `False` to show the `active` (index of button) as pressed or not.
  - The active button is the currently selected one.
- **`buttons`** : A list of `button` objects

Plot type with buttons

Let's first set up a bar chart:

```
fig = px.bar(
      data_frame=revenues,
    x='Industry', y='Revenue',
    color='Industry')
fig.show()
```

Our simple bar chart:



Button set up

Create the buttons to switch plot type:

```
my_buttons = [
{'label': "Bar plot",
 'method': "update",
 'args': [{"type": "bar"}]},
{'label': "scatterplot",
 'method': "update",
 'args': [{"type": "scatter", 'mode': 'markers'}]}
]
```

The args argument

One of the most confusing arguments in Plotly!

- Its structure is:

```
[{dictionary to send to data},
{dictionary to send to layout}]
```

- See what happens when we use Python's `dir` on our figure object to see the internal structure
  - There are some familiar faces! (much more is printed)

```
data
for_each_annotation
for_each_coloraxis
for_each_geo
for_each_layout_image
for_each_mapbox
for_each_polar
for_each_scene
for_each_shape
for_each_ternary
for_each_trace
for_each_xaxis
for_each_yaxis
frames
get_subplot
layout
plotly_relayout
plotly_restyle
plotly_update
pop
print_grid
select_annotations
select_coloraxes
select_geos
select_layout_images
select_mapboxes
select_polars
select_scenes
select_shapes
select_ternaries
select_traces
select_xaxes
select_yaxes
show
```

Using args for layout updates

Let's see what is inside the figure's `layout` element:

```
dir(fig.layout)
```

```
['activeshape', 'angularaxis', 'annotationdefaults', 'annotations', 'autosize', 'bargap', 'bargroupga
p', 'barmode', 'barnorm', 'boxgap', 'boxgroupgap', 'boxmode', 'calendar', 'clickmode', 'coloraxis', 'co
lorscale', 'colorway', 'datarevision', 'direction', 'dragmode', 'editrevision', 'extendfunnelareacolor
s', 'extendpiecolors', 'extendsunburstcolors', 'extendtreemapcolors', 'figure', 'font', 'funnelareacolo
rway', 'funnelgap', 'funnelgroupgap', 'funnelmode', 'geo', 'grid', 'height', 'hiddenlabels', 'hiddenlab
elssrc', 'hidesources', 'hoverdistance', 'hoverlabel', 'hovermode', 'imagedefaults', 'images', 'legen
d', 'mapbox', 'margin', 'meta', 'metasrc', 'modebar', 'newshape', 'on_change', 'orientation', 'paper_bg
color', 'parent', 'piecolorway', 'plot_bgcolor', 'plotly_name', 'polar', 'pop', 'radialaxis', 're', 'sc
ene', 'selectdirection', 'selectionrevision', 'separators', 'shapedefaults', 'shapes', 'showlegend', 's
liderdefaults', 'sliders', 'spikedistance', 'sunburstcolorway', 'template', 'ternary', 'title', 'titlef
ont', 'to_plotly_json', 'transition', 'treemapcolorway', 'uirevision', 'uniformtext', 'update', 'update
menudefaults', 'updatemenus', 'violingap', 'violingroupgap', 'violinmode', 'waterfallgap', 'waterfallgr
oupgap', 'waterfallmode', 'width', 'xaxis', 'yaxis']
```

Phew! There are many, but some should be familiar.

Using args for data updates

Let's also what is inside the figure's `data` element (of the first trace):

```
dir(fig.data[0])
```

```
['alignmentgroup', 'base', 'basesrc', 'cliponaxis', 'constraintext', 'customdata', 'customdatasrc', 'd
x', 'dy', 'error_x', 'error_y', 'figure', 'hoverinfo', 'hoverinfosrc', 'hoverlabel', 'hovertemplate',
'hovertemplatesrc', 'hovertext', 'hovertextsrc', 'ids', 'idssrc', 'insidetextanchor', 'insidetextfont',
'legendgroup', 'marker', 'meta', 'metasrc', 'name', 'offset', 'offsetgroup', 'offsetsrc', 'on_change',
'on_click', 'on_deselect', 'on_hover', 'on_selection', 'on_unhover', 'opacity', 'orientation', 'outside
textfont', 'parent', 'plotly_name', 'pop', 'r', 'rsrc', 'selected', 'selectedpoints', 'showlegend', 'st
ream', 't', 'text', 'textangle', 'textfont', 'textposition', 'textpositionsrc', 'textsrc', 'texttemplat
e', 'texttemplatesrc', 'to_plotly_json', 'tsrc', 'type', 'uid', 'uirevision', 'unselected', 'update',
'visible', 'width', 'widthsrc', 'x', 'x0', 'xaxis', 'xcalendar', 'xsrc', 'y', 'y0', 'yaxis', 'ycalenda
r', 'ysrc']
```

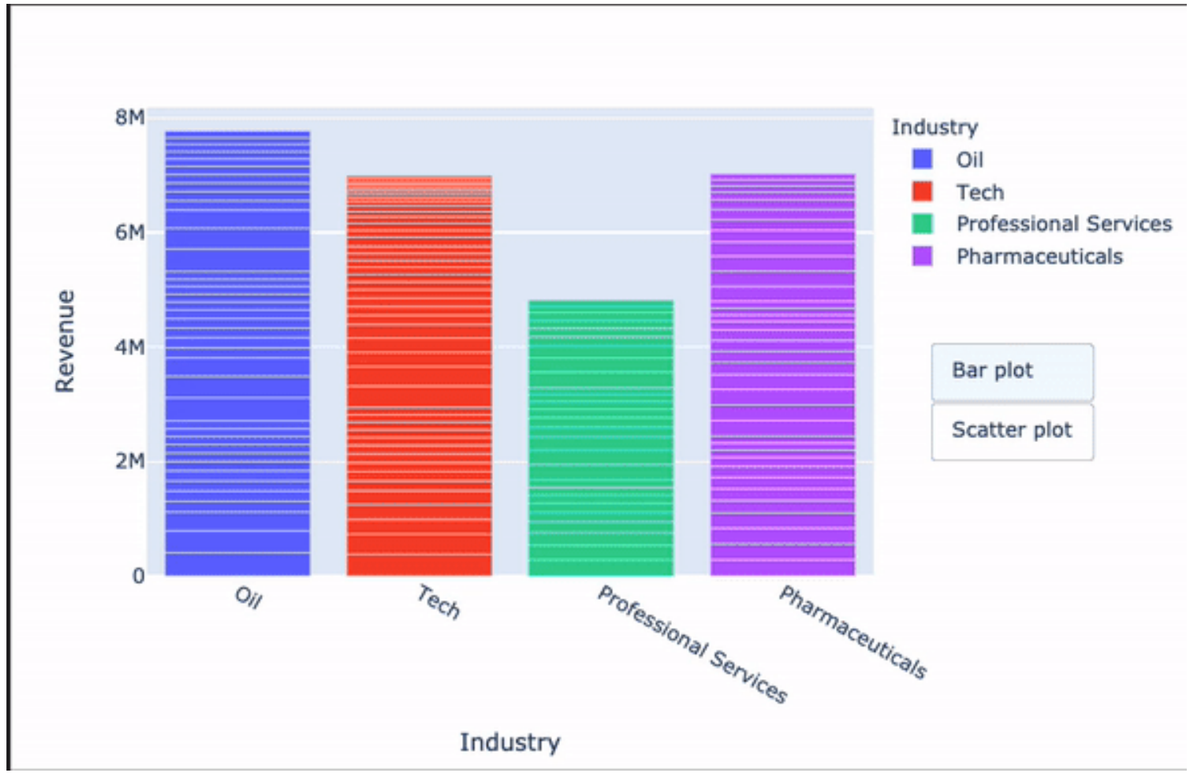
Some are familiar and some are worth noting for later!

Button interactivity

Set the button placement, stacking, and focus:

```
fig.update_layout({
  'updatemenus': [{'type': "buttons",
                   'direction': 'down',
                   'x': 1.3, 'y': 0.5,
                   'showactive': True,
                   'active': 0,
                   'buttons': my_buttons}]
  })
fig.show()
```

Our buttons at work!



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Dropdowns

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Alex Scriven
Data Scientist

datacamp

What is a dropdown?

- Allows user to select from a set of options
- These options will alter the plot in various ways

Dropdowns in Plotly

Dropdowns are created very similarly to buttons.

Create a figure and loop through DataFrames to add traces:

```
fig = go.Figure()
for suburb in ['Ashfield', 'Lidcombe', 'Bondi Junction']:
    df = syd_houses[syd_houses.Suburb == suburb]
    fig.add_trace(go.Bar(x=df['Year'], y=df['Median House Price'], name=suburb))
```

Why so many traces? Our dropdown is going to show/hide different ones!

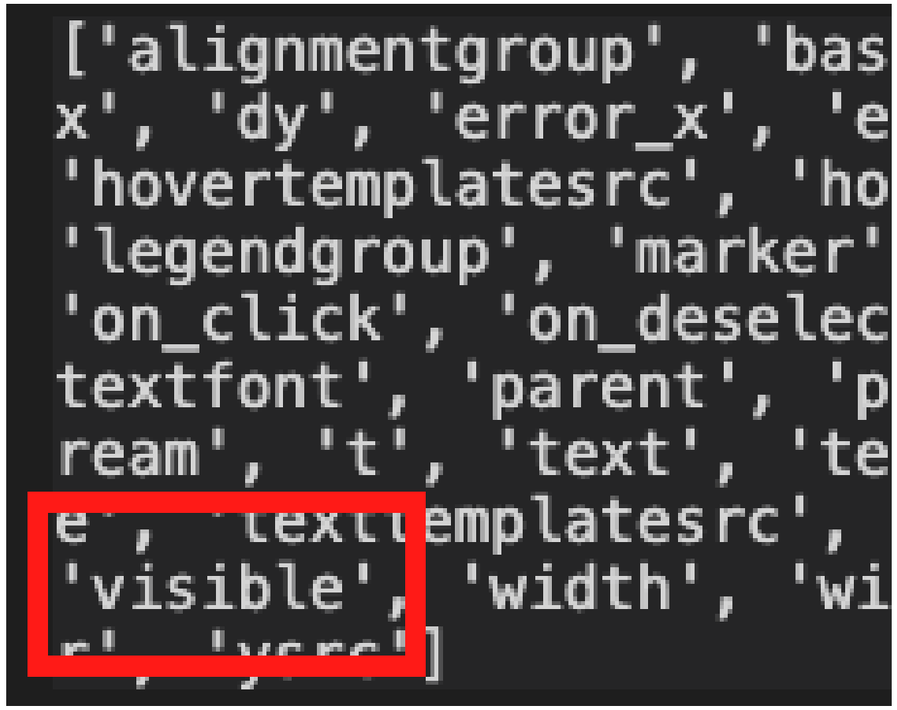
Hiding a trace

Recall what we can update in a figure's `data` element?

- The `visible` argument determines whether traces are visible (`True`) or not (`False`)
- We could use `args` to update the `visible` argument of different traces

```
args:[{'visible': [True, False, False]}]
```

- We can use a list for the `args` value to update all three traces



The dropdown object

The dropdown object, like the button object, is also a list with the same arguments.

```
# Create the dropdown
dropdown_buttons = [
  {'label': 'Ashfield', 'method': 'update',
   'args': [{'visible': [True, False, False]},
            {'title': 'Ashfield'}]},
  {'label': 'Lidcombe', 'method': 'update',
  'args': [{'visible': [False, True, False]},
           {'title': 'Lidcombe'}]},
  {'label': "Bondi Junction", 'method': "update",
  'args': [{"visible": [False, False, True]},
           {'title': 'Bondi Junction'}]}
]
```

Adding the dropdown

Adding the dropdown is also very similar:

```
fig.update_layout({
  'updatemenus':[{
    'type': "dropdown",
    'x': 1.3,
    'y': 0.5,
    'showactive': True,
    'active': 0,
    'buttons': dropdown_buttons}]
})
fig.show()
```

Our dropdown:

Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Sliders

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

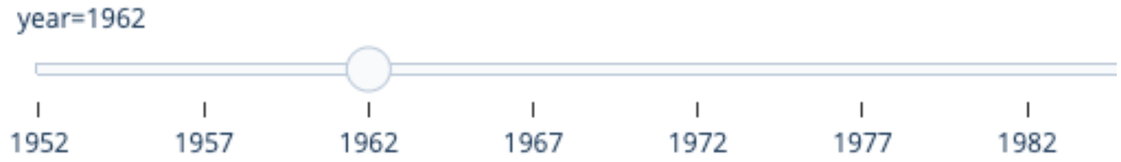
Alex Scriven
Data Scientist

datacamp

What are sliders?

- An interactive element to toggle between values and update your plot
- Often used for viewing data over time, such as data from different years
- Can be used for any group, such as penguin islands
- Ensure it makes sense in your plot

A year slider:



A penguin island slider:

Sliders in plotly.express

`plotly.express` allows sliders via the `animation_frame` and `animation_group` arguments

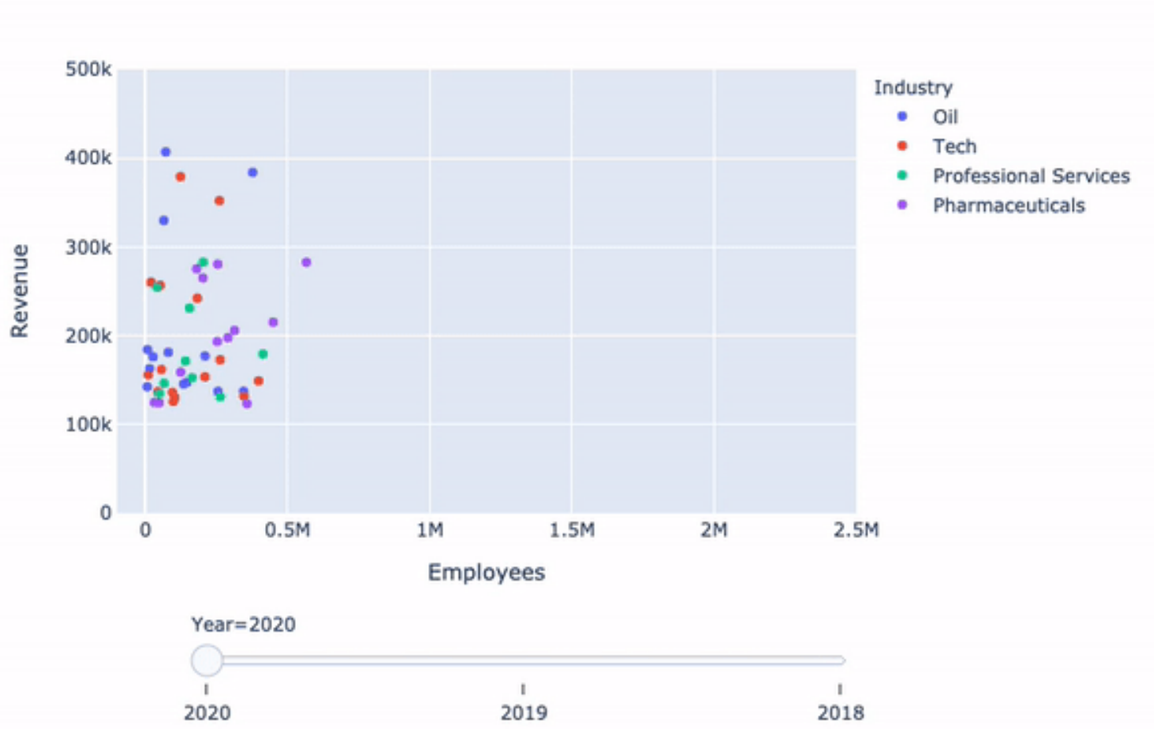
- `animation_frame` : What will be on the slider (`Year` or `Island` on previous slide)
- `animation_group` : How to tell Plotly it is the same object over time

Revenue vs. Employees with slider

```
fig = px.scatter(
  data_frame=revenues,
  y='Revenue',
  x='Employees',
  color='Industry',
  animation_frame='Year',
  animation_group='Company')

fig.update_layout({
  'yaxis': {'range': [0, 500000]},
  'xaxis': {'range': [-100000, 2500000]}
})

fig['layout'].pop('updatemenus')
fig.show()
```



plotly.express limitation: animate method

`plotly.express` sliders have a key limitation - the `animation` slider method

In the `Figure` object

```
fig['layout']['sliders'][0].steps[0]['method']
```

```
animate
```

- With `plotly.express` , you can't update data or layout — only animate the **same data point** over different 'frames'.
- To solve this, we need to use `graph_objects` to create the slider

Sliders with graph_objects

To use `graph_objects`, we need to:

1. Create a figure object with necessary traces
2. Create a sliders object to show/hide traces
3. Update the layout to add the slider to the figure

Creating the figure

Let's create the figure and add traces

```
fig = go.Figure()
for island in ['Torgersen', 'Biscoe', 'Dream']:
    df = penguins[penguins.Island == island]
    fig.add_trace(go.Scatter(
            x=df["Culmen Length (mm)"],
            y=df["Culmen Depth (mm)"], mode='markers', name=island))
```

Creating the slider

Let's create the slider object:

```
sliders = [
  {'steps':[
    {'method': 'update', 'label': 'Torgersen',
    'args': [{'visible': [True, False, False]}]},
    {'method': 'update', 'label': 'Bisco',
    'args': [{'visible': [False, True , False]}]},
    {'method': 'update', 'label': 'Dream',
    'args': [{'visible': [False, False, True]}]}
  ]}
]
```

More formatting options available in the **docs**!

Adding the slider

Now we can add the slider to our figure:

```
fig.update_layout({'sliders': sliders})
fig.show()
```

The first screen was a bit funny huh? Let's fix that!

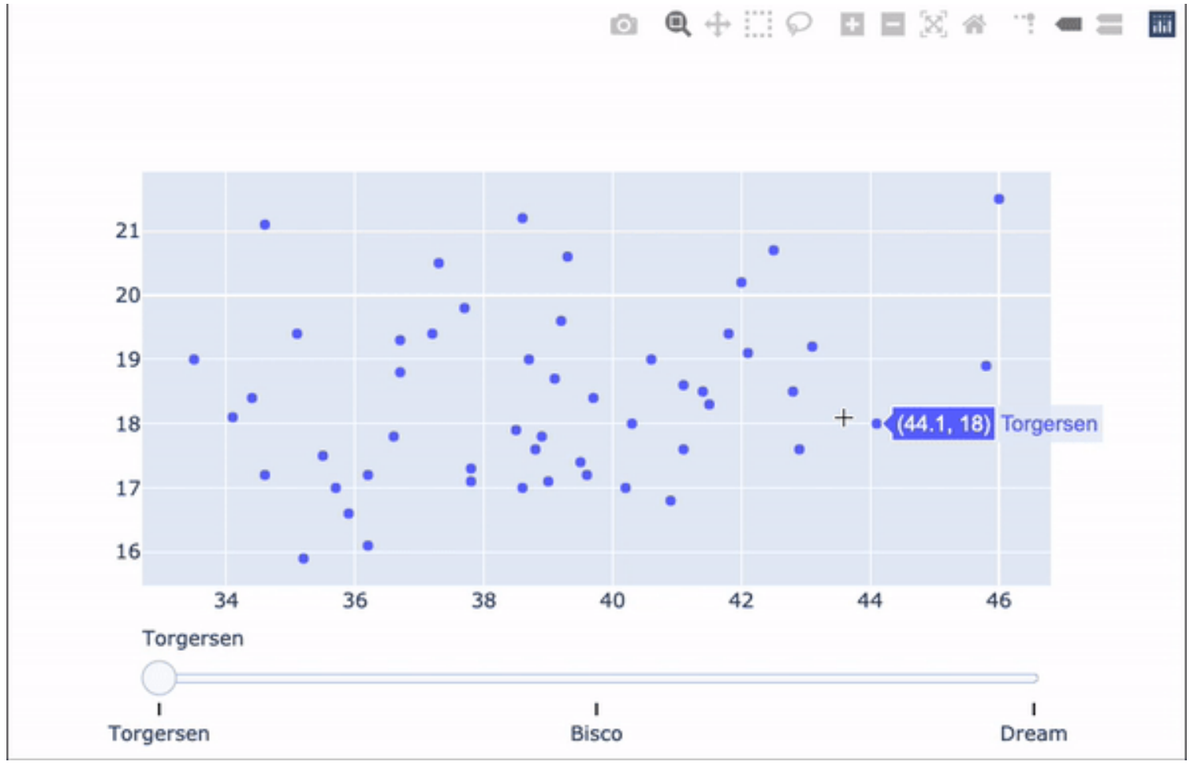
Fixing the initial display

We can fix the initial display by setting only the relevant traces to show.

```
# Make traces invisible
fig.data[1].visible=False
fig.data[2].visible=False

fig.update_layout({'sliders': sliders})
fig.show()
```

Much better!



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

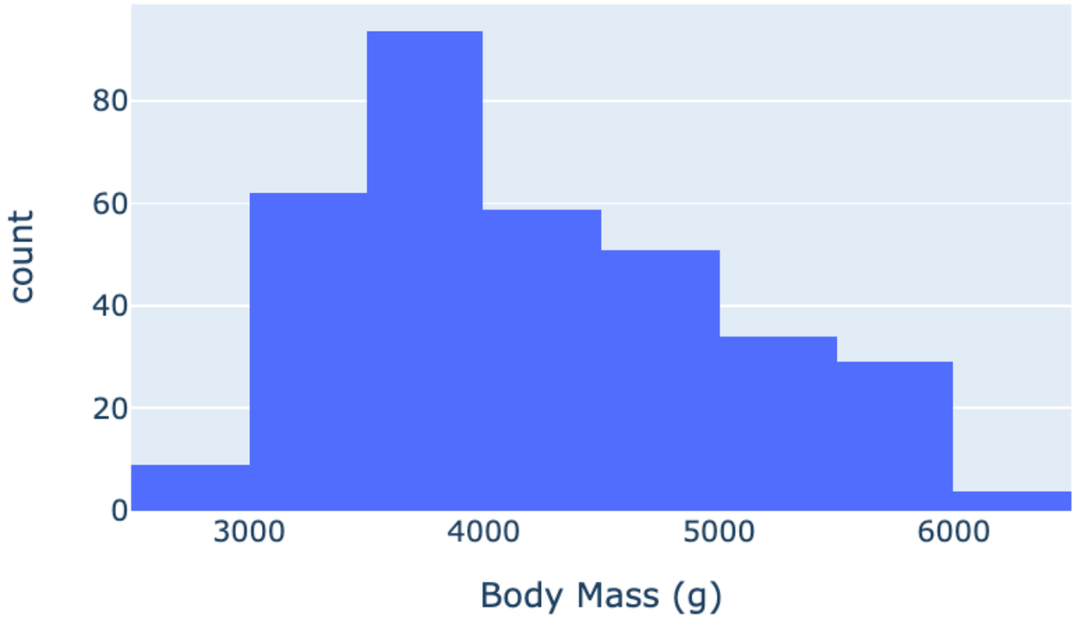
What you learned

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON

Alex Scriven
Data Scientist

Chapter 1

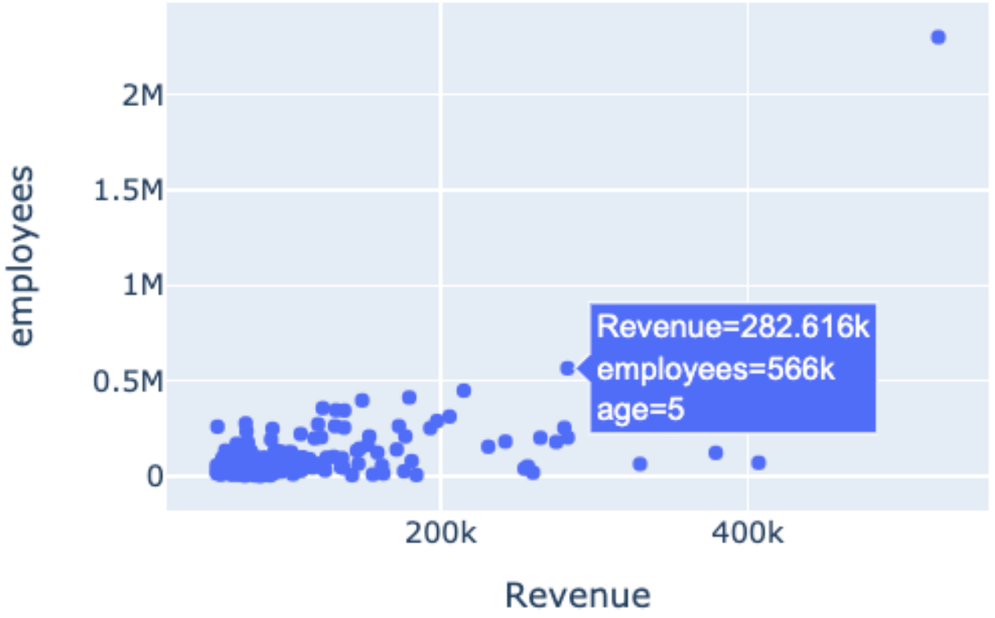
- The Plotly figure
- Univariate plots such as box plots and histograms
- Styled plots using color



Chapter 2

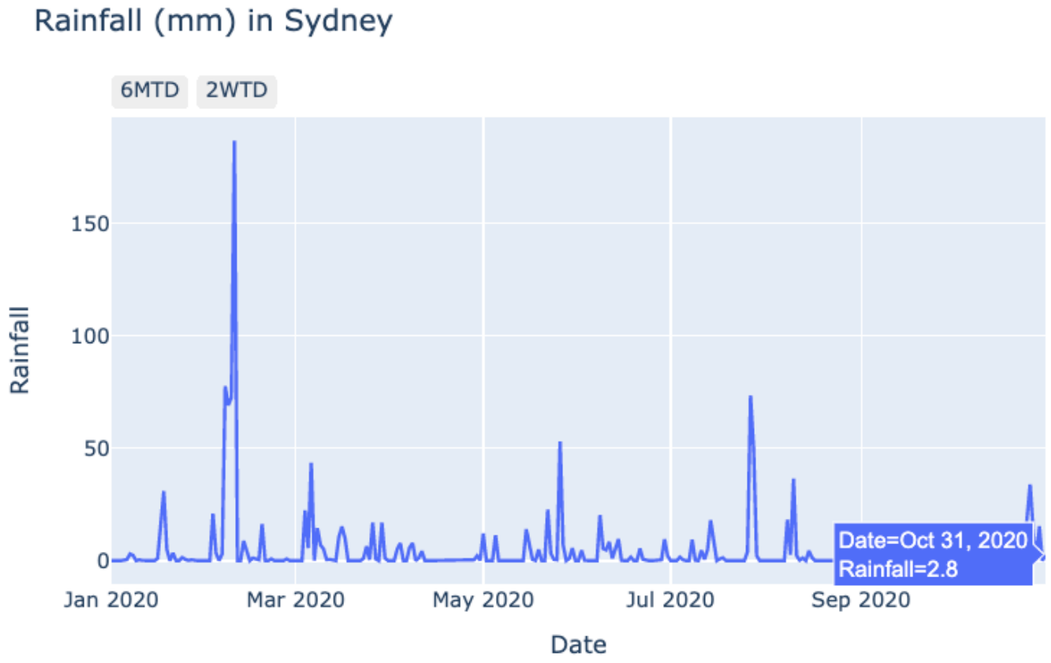
- Bivariate visualizations such as scatterplots and bar plots
- Customized your plots further with:
  - Hover information and legends
  - Annotations
  - Custom plot axes

Recall seeing company `age` (another variable) in the hover!



Chapter 3

- Advanced customization
  - Subplots of same or different types
  - Layering multiple plots on the same chart
  - An introduction to time buttons



Chapter 4

Using interactive elements:

- Buttons
- Dropdowns
- Sliders

Your houses dropdown:

Thank you!

INTRODUCTION TO DATA VISUALIZATION WITH PLOTLY IN PYTHON