Custom buttons

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT 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)

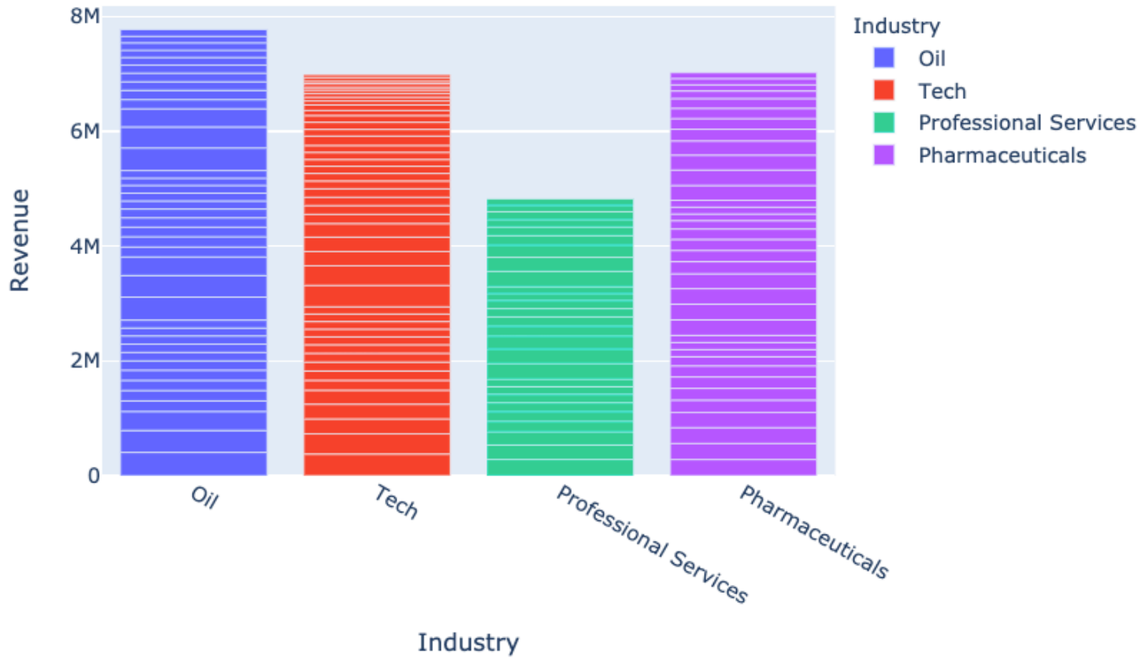
Custom buttons in Plotly

Added via an `updatemenus` argument with 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
- **`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

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



Button set up

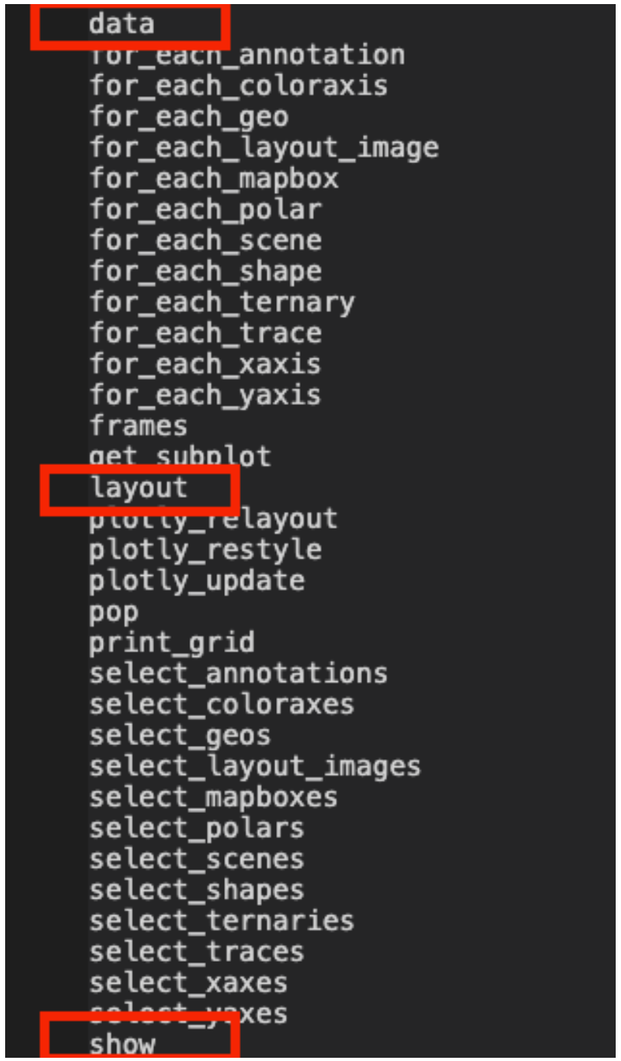
```
my_buttons = [
{'label': "Bar plot",
 'method': "update",
 'args': [{"type": "bar"}]},
{'label': "scatterplot",
 'method': "update",
 'args': [{"type": "scatter", 'mode': 'markers'}]}
]
```

The args argument

One of the most confusing parts of Plotly!

- Its structure is:

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



Using args for layout updates

```
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']
```

- Update any of the arguments with `'args'`

Using args for data updates

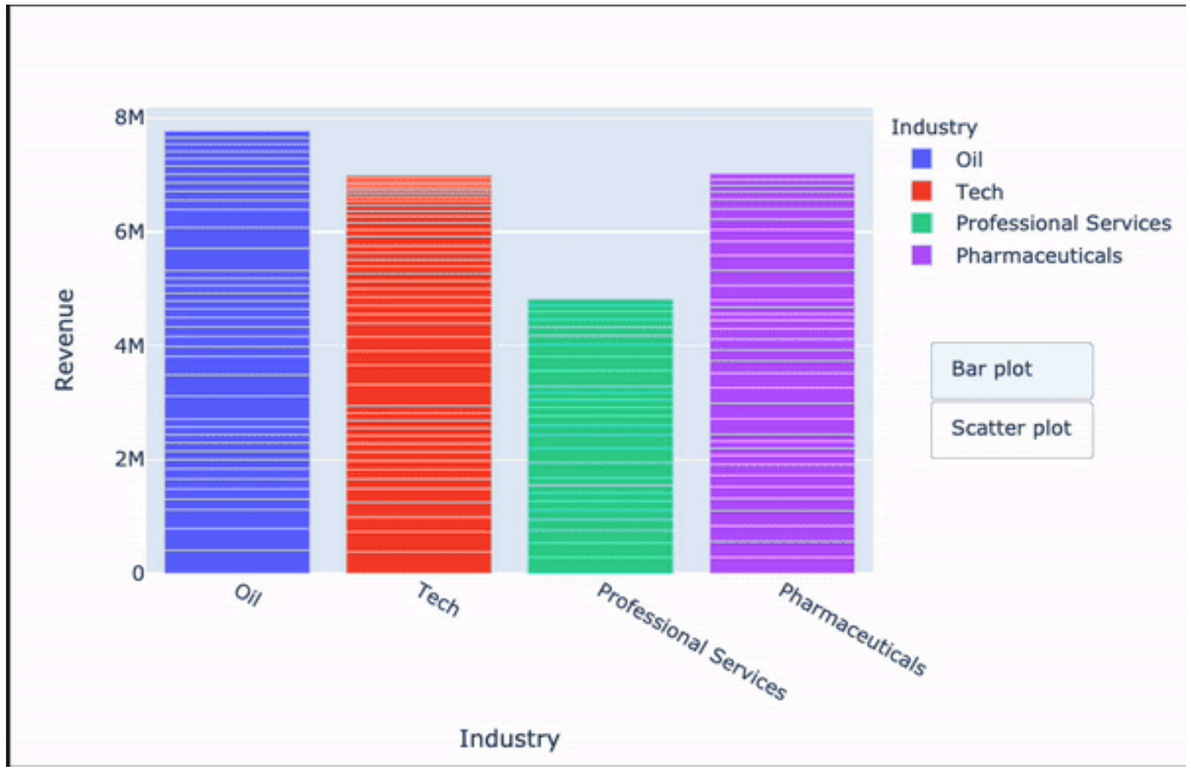
```
dir(fig.data[0])
```

```
['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 will be helpful later

Button interactivity

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



Let's practice!

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT PLOTLY IN PYTHON

Dropdowns

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT PLOTLY IN PYTHON



Alex Scriven
Data Scientist

datacamp

What is a dropdown?

- Allows the user to select from a set of options
- Updates data or layout elements



Dropdowns in Plotly

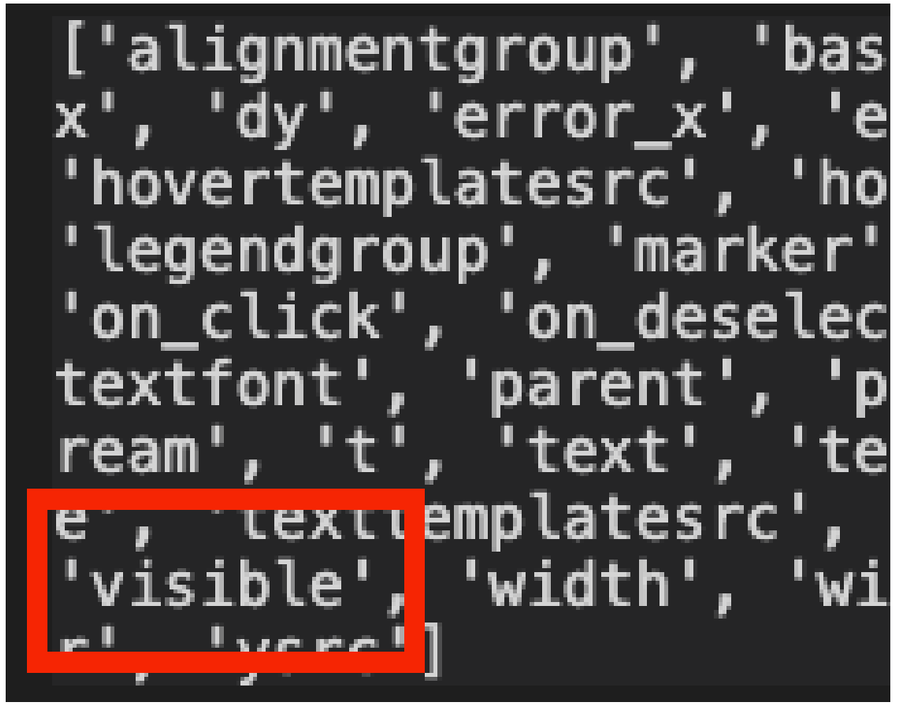
```
fig = go.Figure()
for suburb in ['Ashfield', 'Lidcombe', 'Bondi Junction']:
    df = syd_houses[syd_houses.Suburb == suburb]
    fig.add_trace(
    px.bar(df, x='Year', y='Median House Price').data[0])
```

- Dropdowns work by showing and hiding specific traces

Hiding a trace

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

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



The dropdown object

```
# 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

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



Let's practice!

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT PLOTLY IN PYTHON

Sliders

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT PLOTLY IN PYTHON



Alex Scriven
Data Scientist

datacamp

What are sliders?

- An interactive element to toggle between values and update a plot
- Used for viewing data over time
- Can be used for any group

A year slider:



A penguin island slider:



- 🤔 Ensure it makes sense in your plot

Sliders in plotly.express

- `animation_frame` What will be on the slider (`Year` or `Island` on the previous slide)

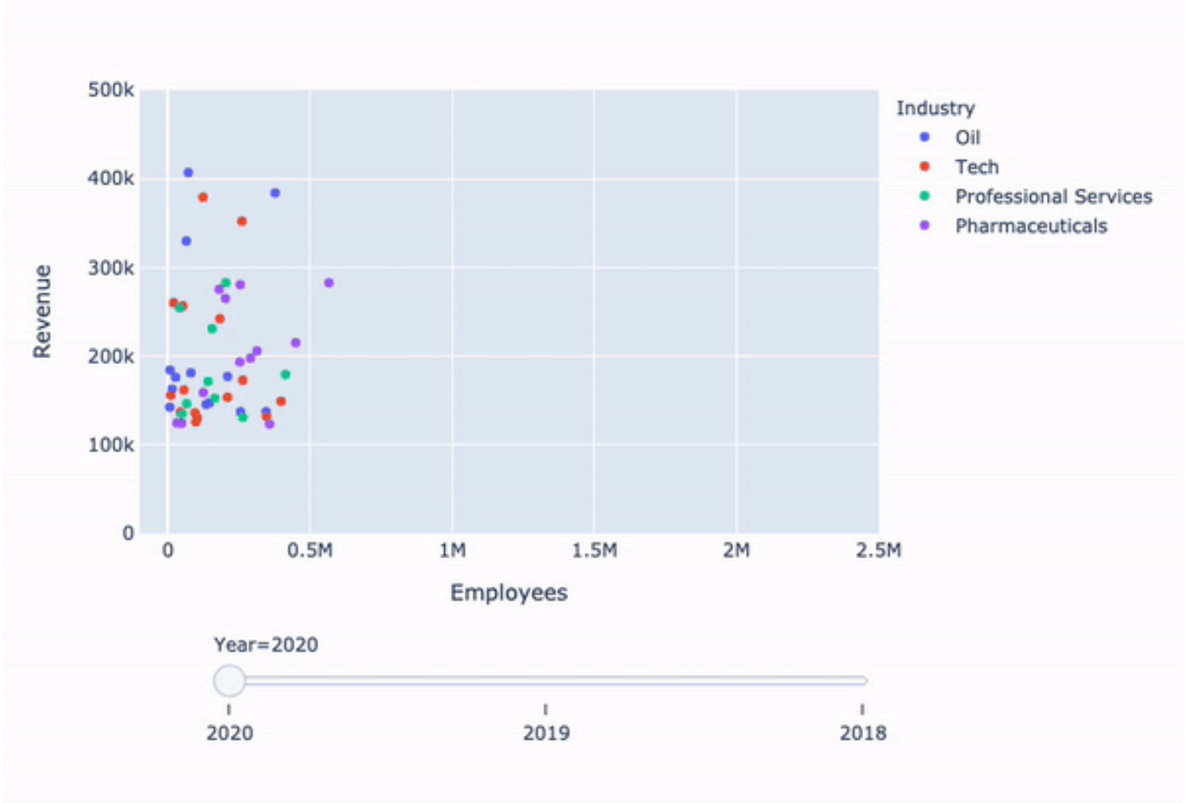
- `animation_group` : Identifies which samples stay consistent across frames

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()
```



Limitation: animate method

`plotly.express` implements sliders using `animate` method

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

```
animate
```

- Only animates the **same data point** over time
- Build the slider using traces

Our plan

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

Creating the figure

```
fig = go.Figure()
for island in ['Torgersen', 'Biscoe', 'Dream']:
    df = penguins[penguins.Island == island]
    temp_trace = px.scatter(df, x="Culmen Length (mm)", y="Culmen Depth (mm)")
    fig.add_trace(temp_trace.data[0])
```

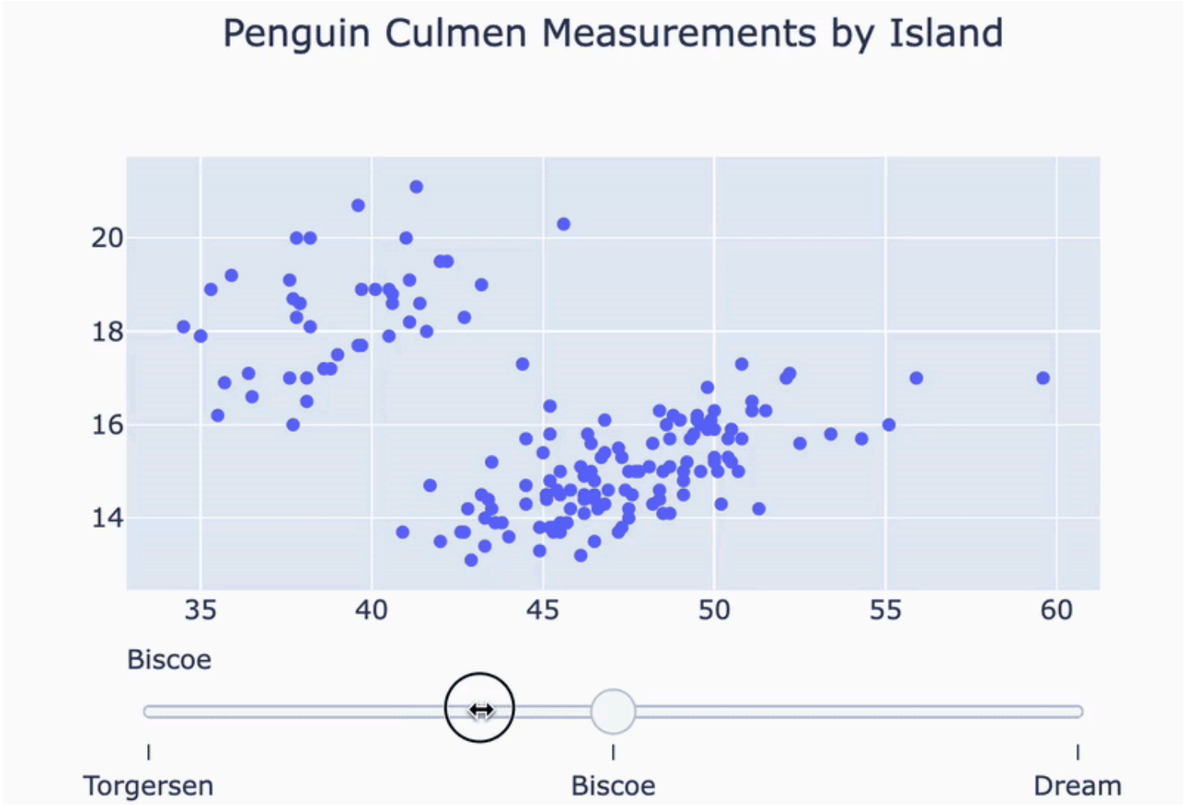
Creating the slider

```
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 **documentation**

Adding the slider

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



Let's practice!

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT PLOTLY IN PYTHON

What you learned

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT PLOTLY IN PYTHON

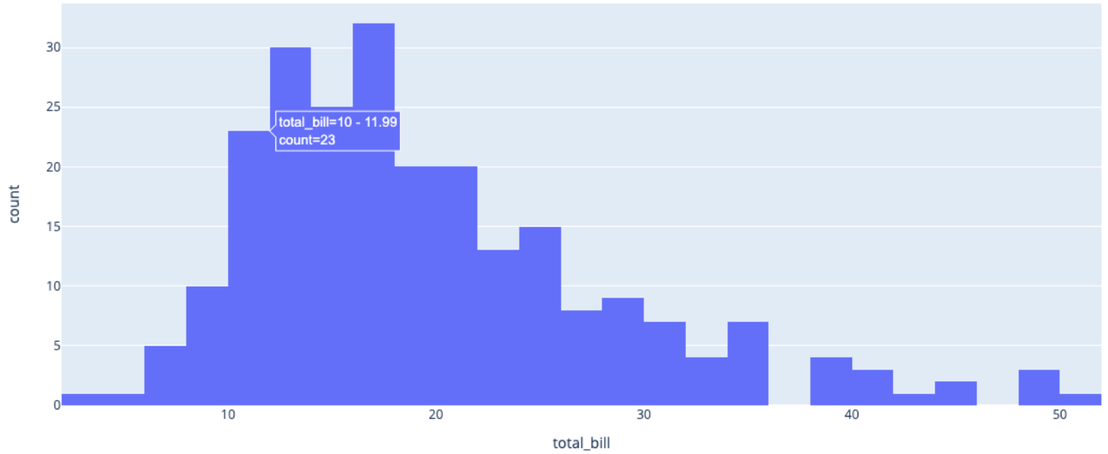


Alex Scriven
Data Scientist

datacamp

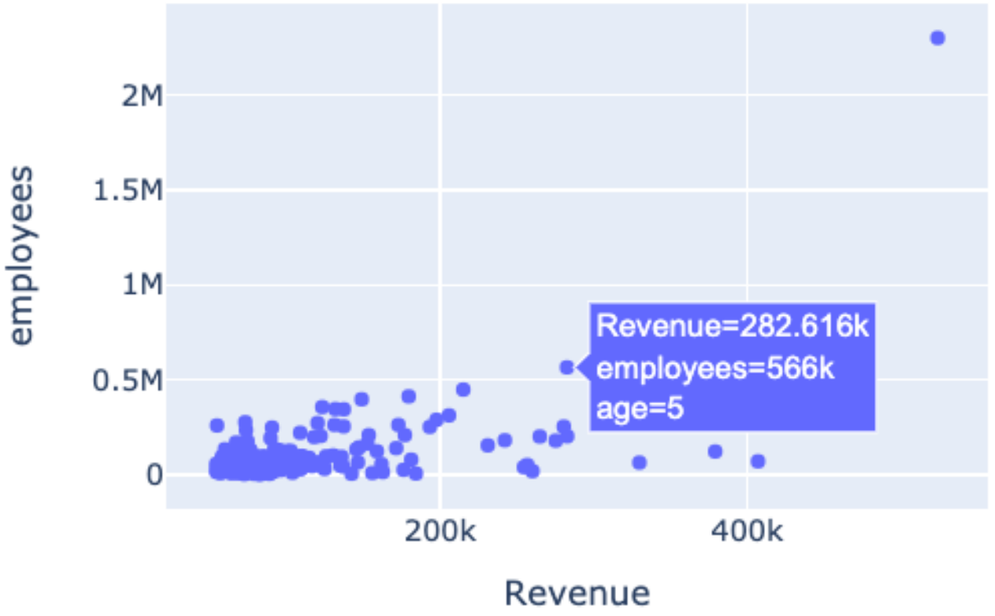
Chapter 1

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



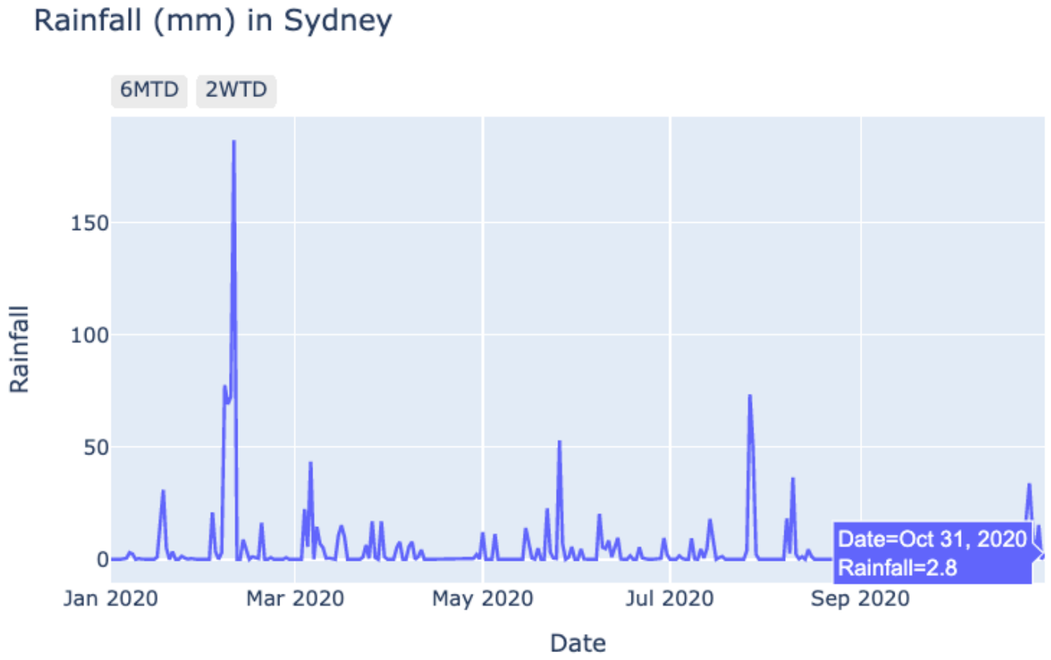
Chapter 2

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



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



What next?



Building Dashboards with Dash and Plotly

Thank you!

EINFÜHRUNG IN DIE DATENVISUALISIERUNG MIT PLOTLY IN PYTHON