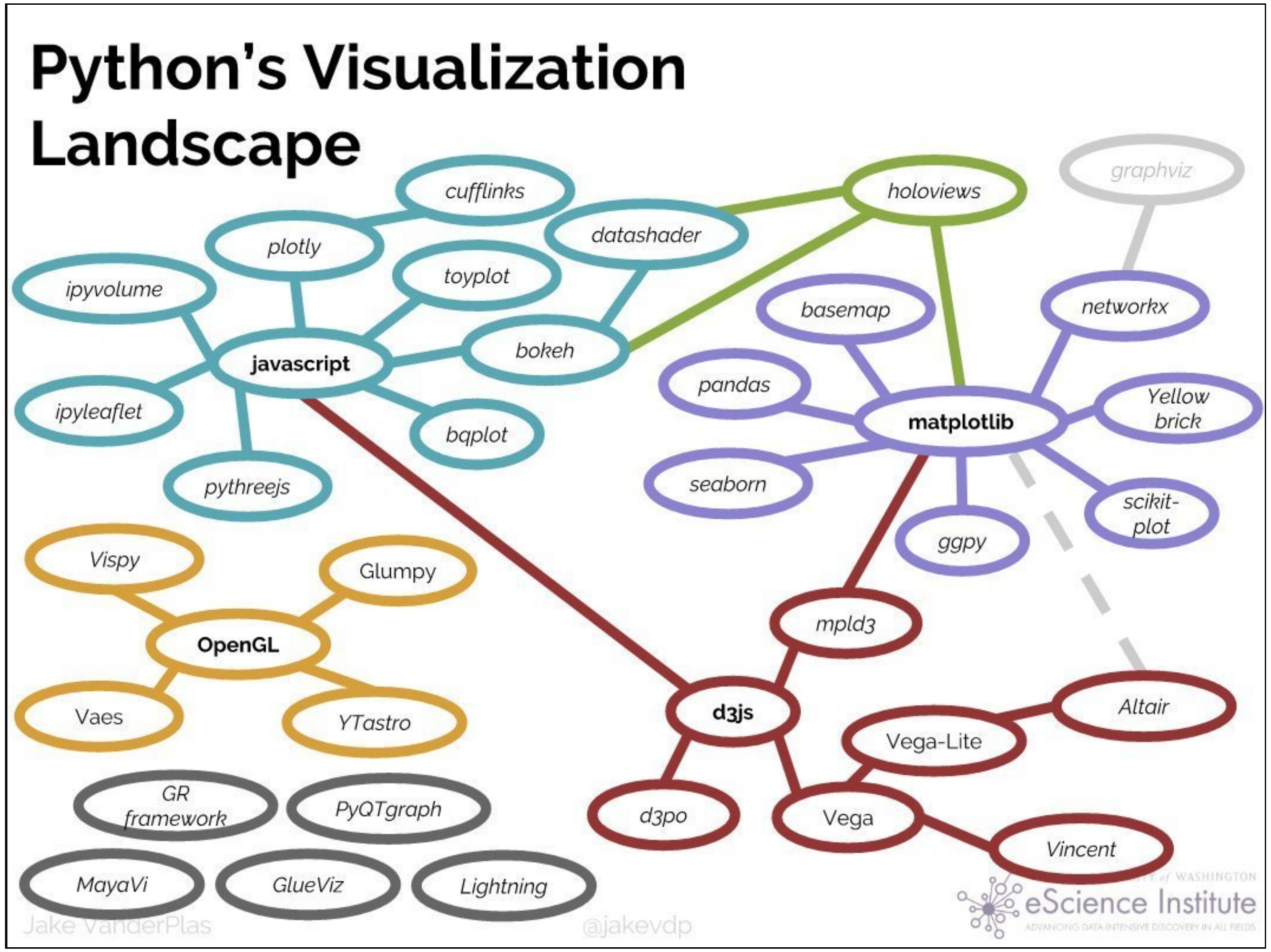
plotly
Python

Interactive Data Visualisations Built with Python

Jesús Martínez Blanco
Data Scientist

github.com/**chumo**
linkedin.com/in/**chumo**
twitter.com/**jmb_jesus**



Jake Vanderplas (PyCon 2017):
https://www.youtube.com/watch?v=FytuB8nFHPQ

Plotly's open source libraries for Data Science

Apart from their paid products, they have open sourced their plotting libraries:

- Plotly.js: JavaScript library for front-end graphs and dashboards (example here).
- Plotly for R: the Javascript code is generated from R code.
- Plotly for Python: the Javascript code is generated from Python code.
- Dash: Python framework for building analytical web applications (including server side).

They are free to use and are fully functional also OFFLINE
(no need to use their servers).

It is possible to convert your matplotlib graphics to Plotly plots!

What is Plotly

Website: https://plot.ly

A Canadian company building products around data analytics and visualisation tools:

- Charts: Web **UI** for building plots online.
- Dashboards: Online dashboards with **D3.js** Plotly charts.
- Slide Decks: Powerpoint-like slide decks online that have interactive Plotly charts.
- Database connectors: Connect Plotly charts to SQL.

They make money hosting your plots privately (***Plotly Professional***) and managing *on premises* installations (***Plotly Enterprise***).

Disclaimer: I do not hold any professional or commercial relationship with Plotly

Plotly for Python

We will focus on the Python binding of Plotly. Write Python code and get interactive plots rendered in the browser.

You'll need:

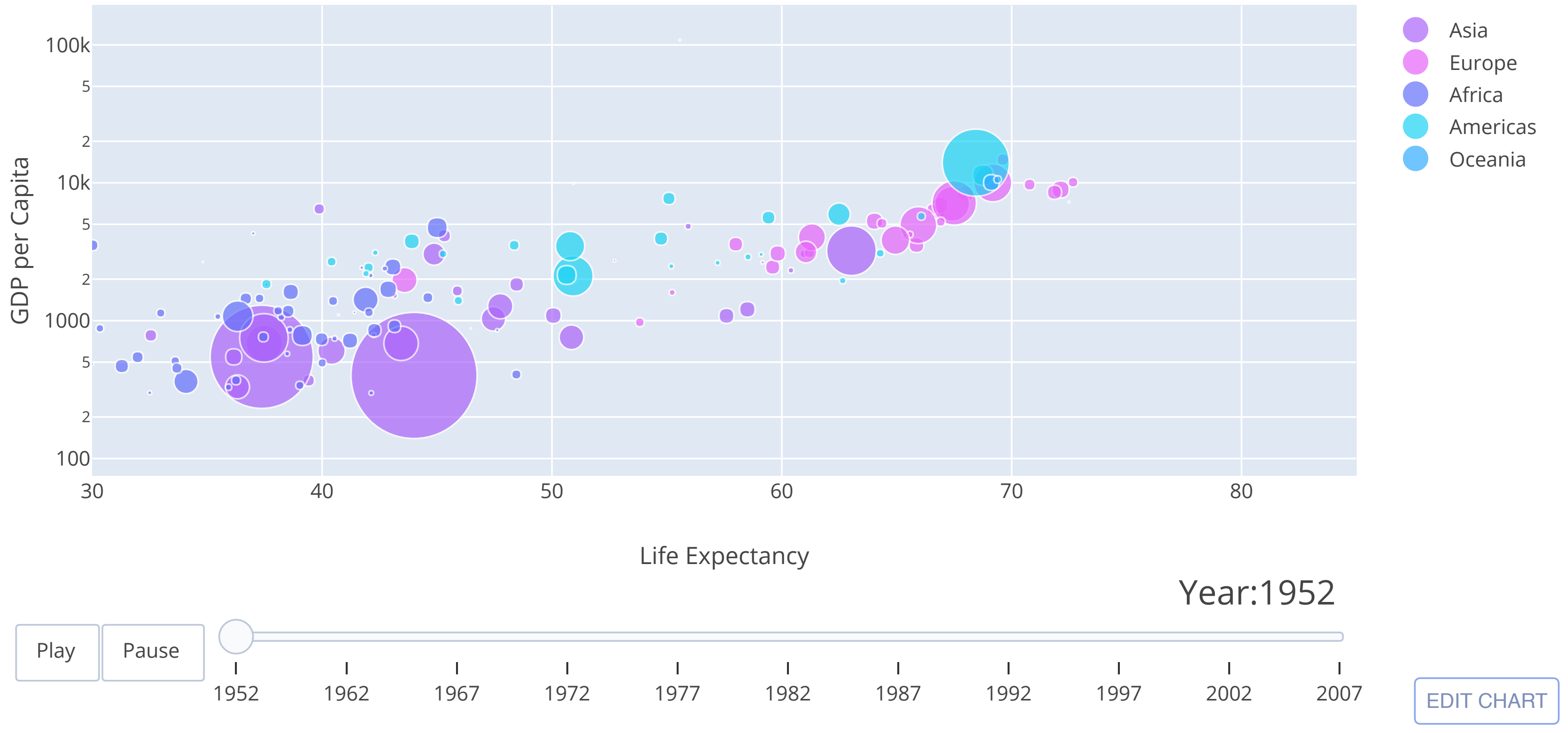
- Python > 3.5 installation (for example Anaconda distribution) and the Jupyter notebook.
- `pip install plotly`

Links to the docs:

- Plotly Python
- Plotly figure reference

A famous example

Video: How Does Income Relate to Life Expectancy? (The Gapminder Foundation)



Code at https://plot.ly/python/gapminder-example/

5.2

Let's get started

```
# Import the library
import plotly as py

# and enable the offline mode in the notebook
py.offline.init_notebook_mode(connected=True)
```

if False, the entire library plotly.js will be loaded into the notebook

Building blocks

A Plotly figure is built upon objects from **plotly.graph_objs** declared with Python ***dictionaries*** and ***lists***.

```
# Import the Plotly building blocks
import plotly.graph_objs as go
```

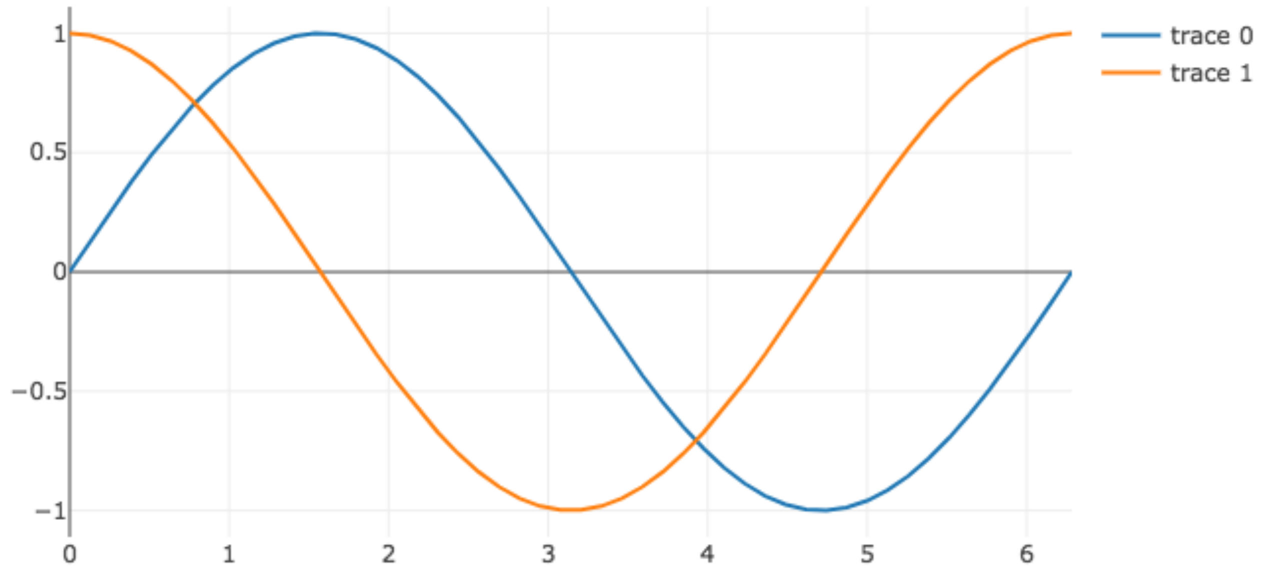
Examples of such objects are:

```
go.Scatter(), go.Bar(), go.Histogram()
go.Layout(), go.Legend()
go.Figure()
```

Constructing a figure

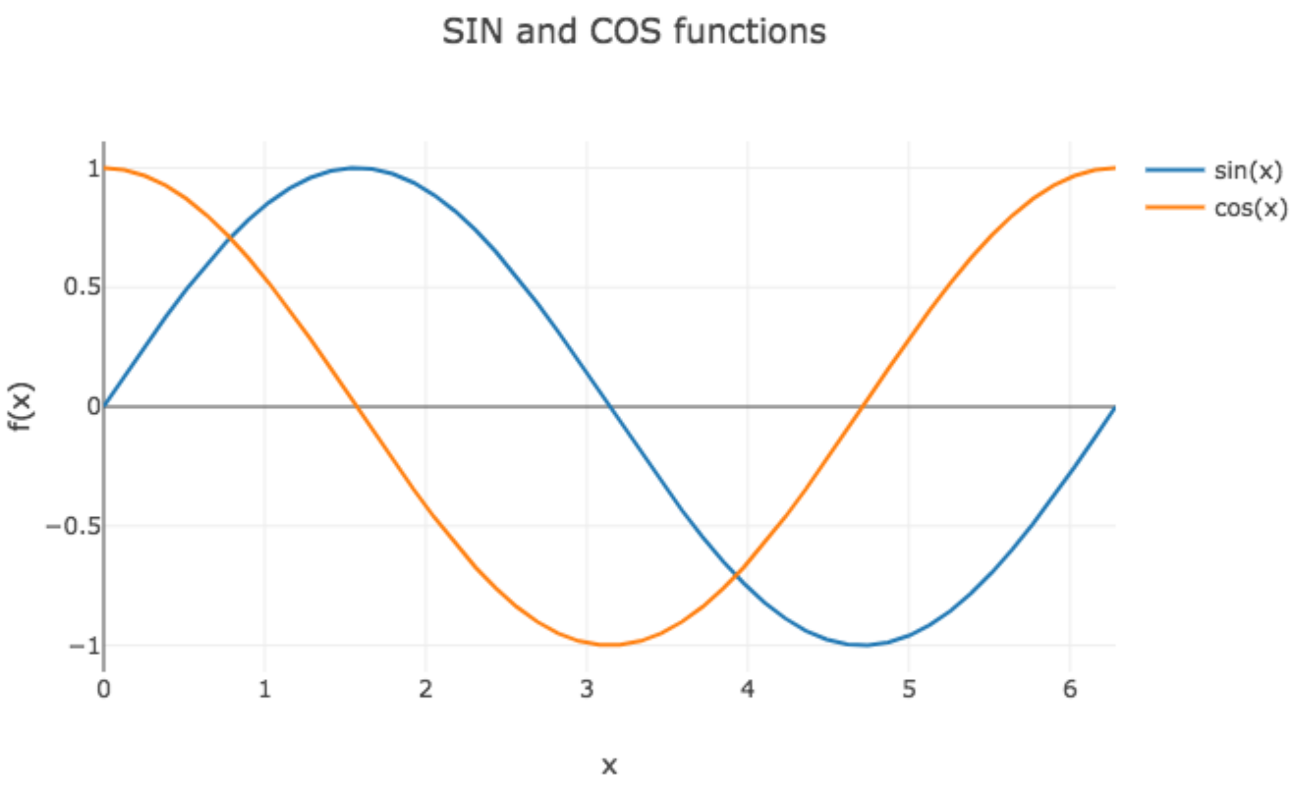
```
import numpy as np
x = np.linspace(0, 2*np.pi)

# Traces
trace0 = go.Scatter(x=x, y=np.sin(x))
trace1 = go.Scatter(x=x, y=np.cos(x))



# Figure
fig = go.Figure(data=[trace0, trace1])



# Display the result in the notebook with...
py.offline.iplot(fig)
```

6.3


trace 0
trace 1
1
0.5
0
−0.5
−1
0
1
2
3
4
5
6

and now with a *Layout*

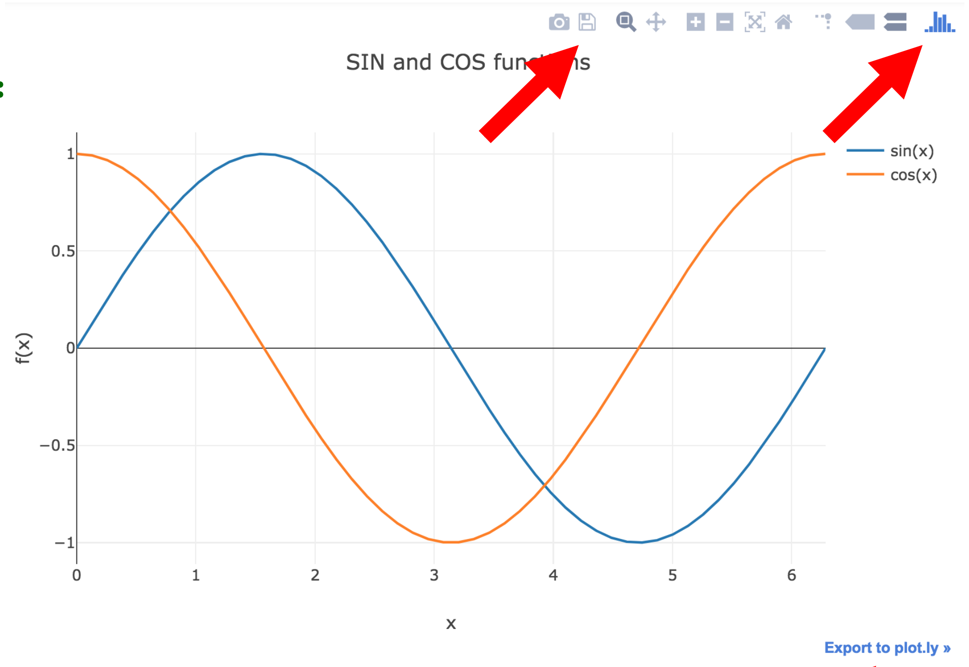
```
import numpy as np
x = np.linspace(0, 2*np.pi)

# Traces
trace0 = go.Scatter(x=x, y=np.sin(x), name='sin(x)')
trace1 = go.Scatter(x=x, y=np.cos(x), name='cos(x)')

# Layout
layout = go.Layout(title='SIN and COS functions',
                   xaxis=dict(title='x'),
                   yaxis=dict(title='f(x)'))

# Figure
fig = go.Figure(data=[trace0, trace1], layout=layout)
```



6.6

cleaner Layout: no Plotly links

```
# Default, with Plotly links:
py.offline.iplot(fig)
```



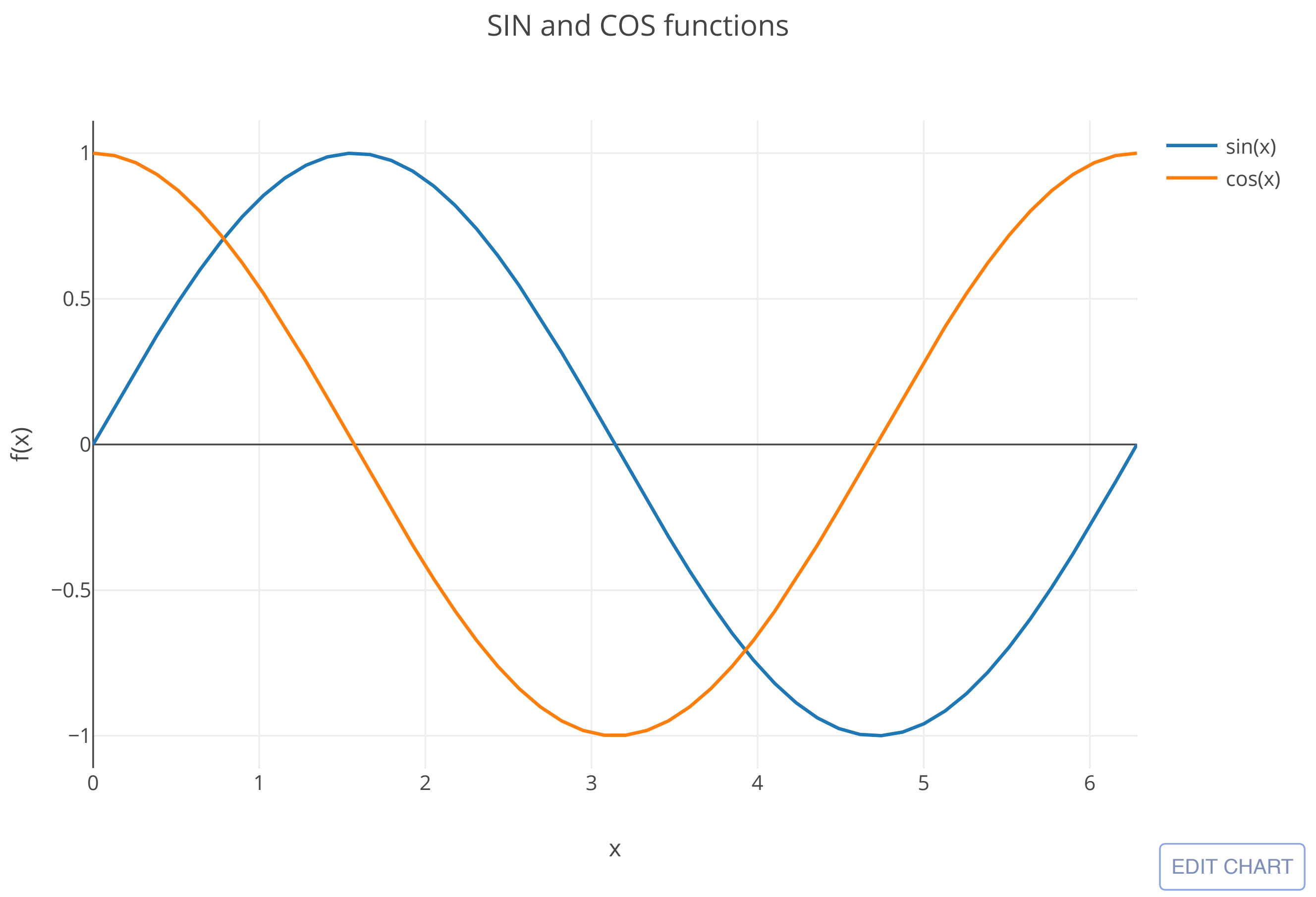
```
# Without Plotly links:
config = dict(modeBarButtonsToRemove=['sendDataToCloud'],
              displaylogo=False)

py.offline.iplot(fig,
                 show_link=False,
                 config=config)
```

Things you can do with the Figure object

```
# Display it in the notebook
py.offline.iplot(fig)

# Create a stand-alone html file
py.offline.plot(fig, filename='sin_cos.html')

# or just a <div> element with the plot to embed in your web page
div_str = py.offline.plot(fig, output_type='div', include_plotlyjs=False)

# Export it as static image
py.plotly.image.save_as(fig, filename='sin_cos.png')

# Share it via the Plotly cloud
py.plotly.plot(fig, filename='sin_cos', sharing='public')
```

Require a Plotly account (possible for free) and set credentials (API key) as explained here.

The figure, hosted in Plotly

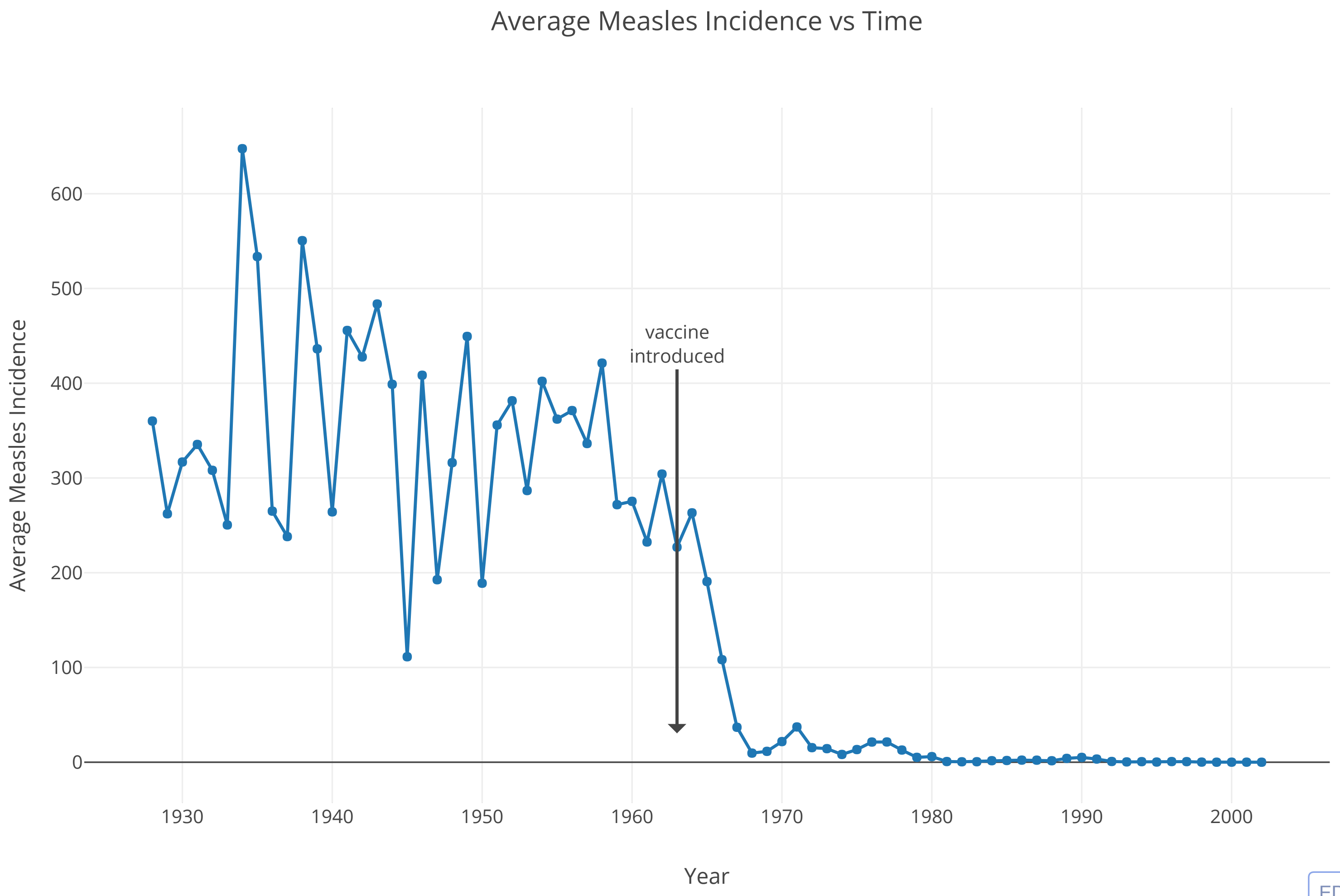


7.2

Exercise
with the data at...

https://github.com/chumo/VIZ_course/data/measles_incidence_per_year.csv

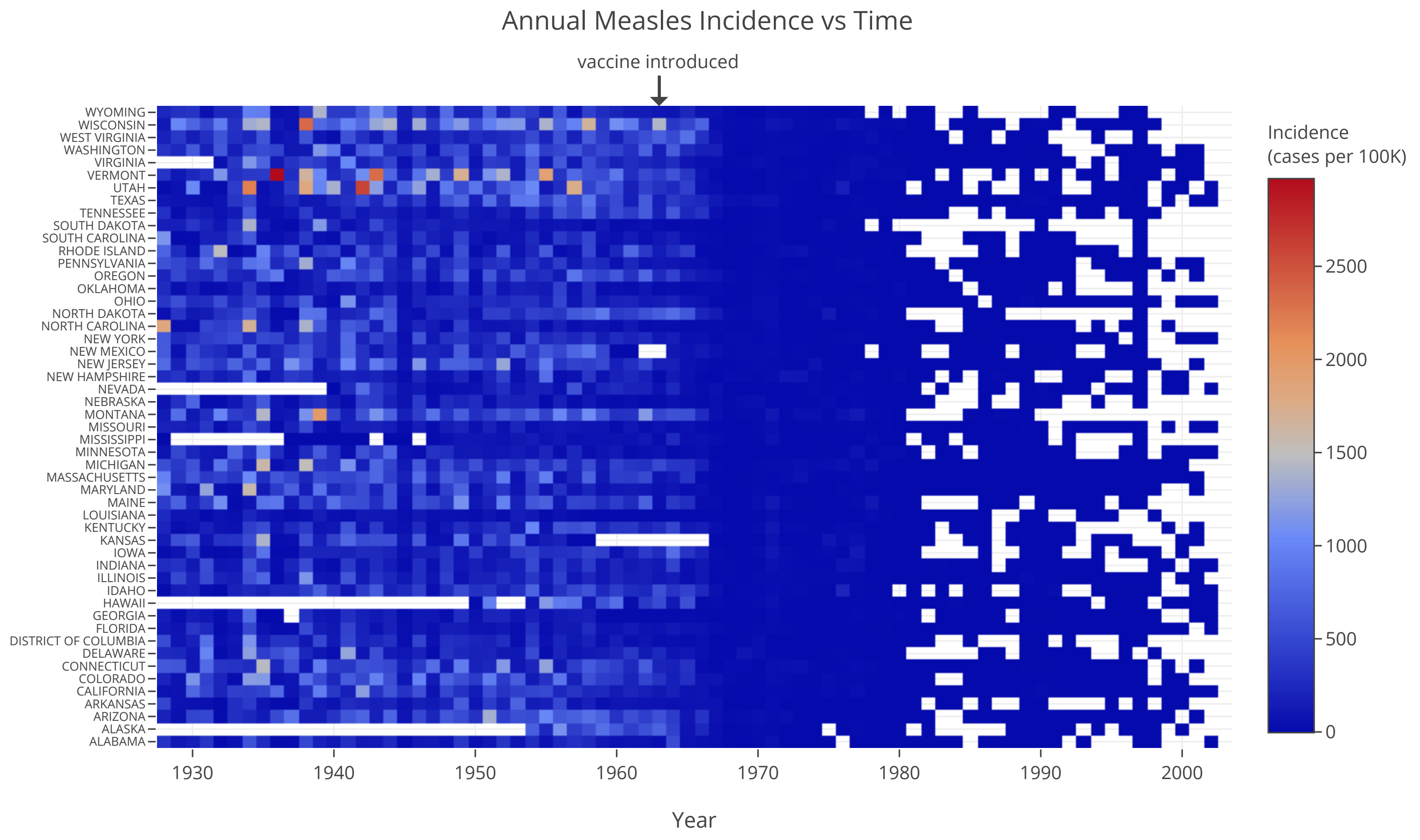
Exercise: Reproduce this!



EDIT CHART

8.2

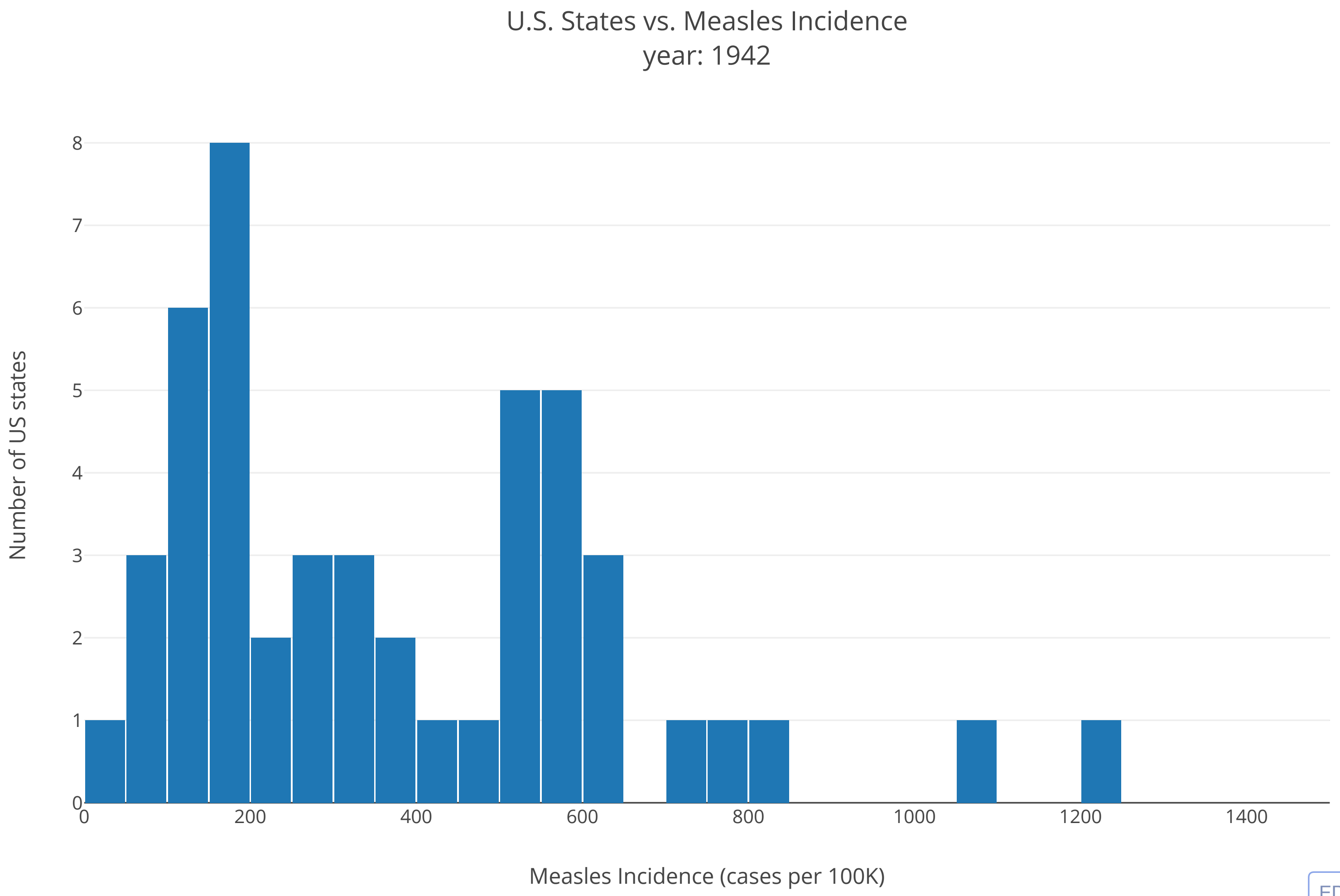
Exercise: Reproduce this!



EDIT CHART

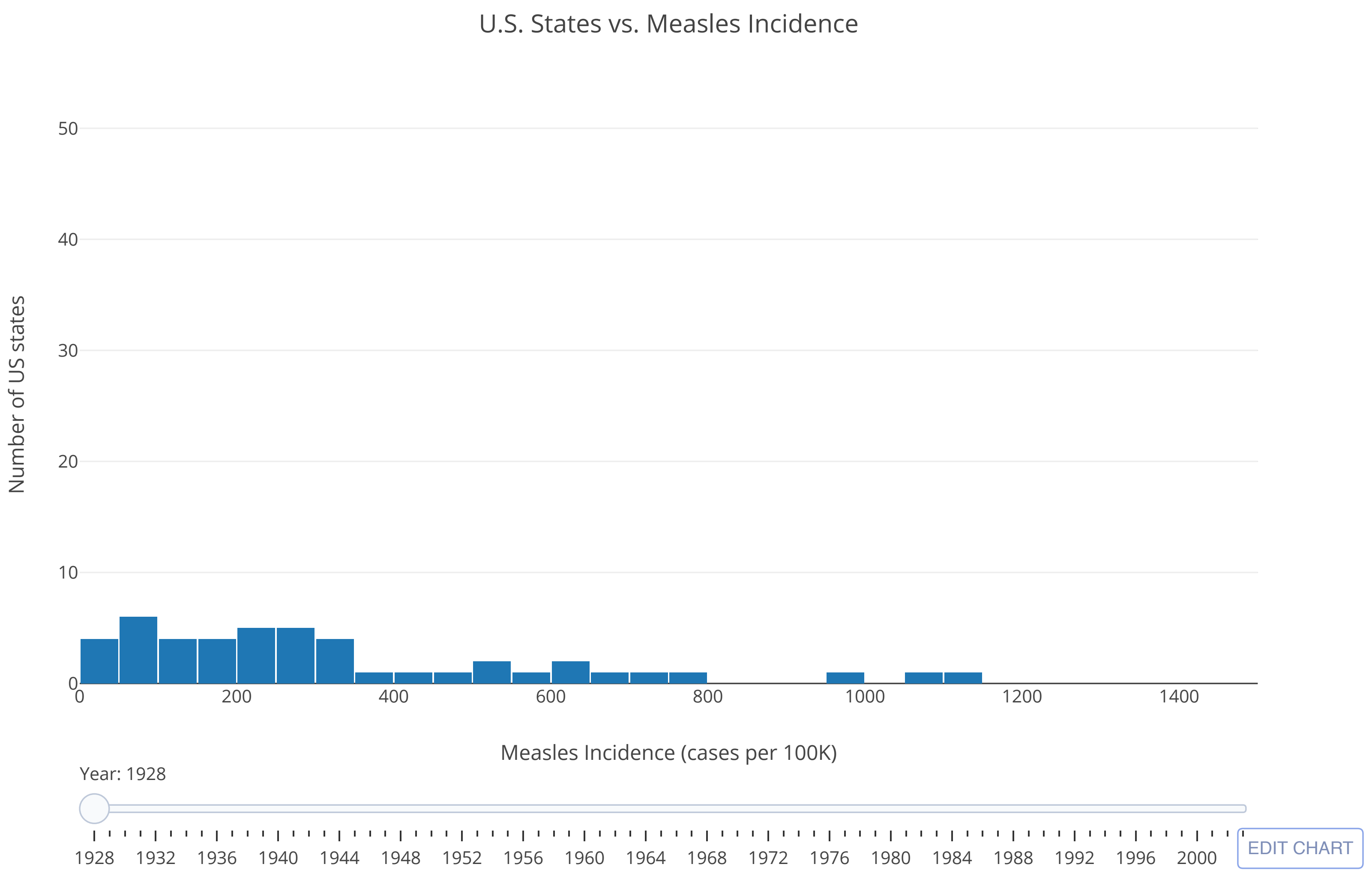
8.3

Exercise: Reproduce this!



EDIT CHART

Bonus: Reproduce this!



8.5

Building Dashboards

A few techniques that you can consider:

1. Bare metal with HTML, CSS and Javascript (Plotly.js).
2. Use the GUI (Plotly Dashboards).
3. Web based & server assisted (Plotly Dash).
4. Plotly + ipywidgets in Jupyter Dashboards (video).
5. Precompute plots in Python offline and load them dynamically on a static web page.

Building Dashboards

What if your plot changes but your dashboard template not?

1. Precompute plots with Plotly Python as <div> elements.
2. Upload the <div> to a static hosting service like GitHub or S3.
3. Write HTML template for the Dashboard with placeholders.

Every time your plots change, upload them again to the hosting service. The new version of your plots will show up in the Dashboard upon page refresh.

Building Dashboards

Embedding Plotly figures in static web pages

```
# Generate the HTML code of the plot in a <div> element
div_str = py.offline.plot(fig,
                          output_type='div',
                          include_plotlyjs=False)

# Insert the code in the page template
html_str = '''<!DOCTYPE html>
                <html>

                <head>
                  <script src="https://d3js.org/d3.v3.min.js"></script>
                  <script src='https://cdn.plot.ly/plotly-latest.min.js'></script>
                </head>


                <body>
                  <h1>Simple Dashboard</h1>
                  <p>The following plot is static and interactive at the same time ;)</p>

                  ''' + div_str + '''

                </body>

                </html>
                '''

# The resulting string can be saved in a file
with open('simple_dashboard.html', 'w') as f:
    f.write(html_str)
```

Building Dashboards

1. Precompute plots with Plotly Python as <div> elements

```
# Generate the HTML code of the plot in a <div> element
div_str = py.offline.plot(fig,
                          output_type='div',
                          include_plotlyjs=False)


# Save the resulting string in a file
with open('myplot.html', 'w') as f:
    f.write(div_str)
```

Building Dashboards

2. Upload the <div> to a static hosting service like GitHub or S3

```
git push ...

aws s3 cp ...
```

Building Dashboards

3. Write HTML template for the Dashboard with placeholders

```
<!DOCTYPE html>
<html>

<head>
  <script src="https://d3js.org/d3.v3.min.js"></script>
  <script src='https://cdn.plot.ly/plotly-latest.min.js'></script>
</head>


<body>
  <h1>Simple Dashboard</h1>
  <p>The following plots are loaded from outside this page...</p>

  <div id="plot_1"></div>
  <div id="plot_2"></div>

  <script>
    d3.html('https://raw.githubusercontent.com/chumo/Data2Serve/master/myplot1.html',
             function(fragment){d3.select('#plot_1').node().append(fragment);});
    d3.html('https://raw.githubusercontent.com/chumo/Data2Serve/master/myplot2.html',
             function(fragment){d3.select('#plot_2').node().append(fragment);});
  </script>

</body>

</html>
```

Cufflinks

Plotly plots directly from Pandas Dataframes

Requires the installation of another free library from Plotly:

```
pip install cufflinks
```

After importing, a new method (**iplot**) is available to generate Plotly plots from Pandas dataframes with a one liner:

```
import cufflinks as cf
import pandas as pd

# Configure it to work offline:
cf.go_offline(connected=True)
```

Cufflinks

Plotly plots directly from Pandas Dataframes

```
# A random dataframe as example
df = pd.DataFrame({'Column A':[2,5,3,4,1],
                   'Column B':[6,2,4,1,8]})
```

Two possible ways to generate the figure:

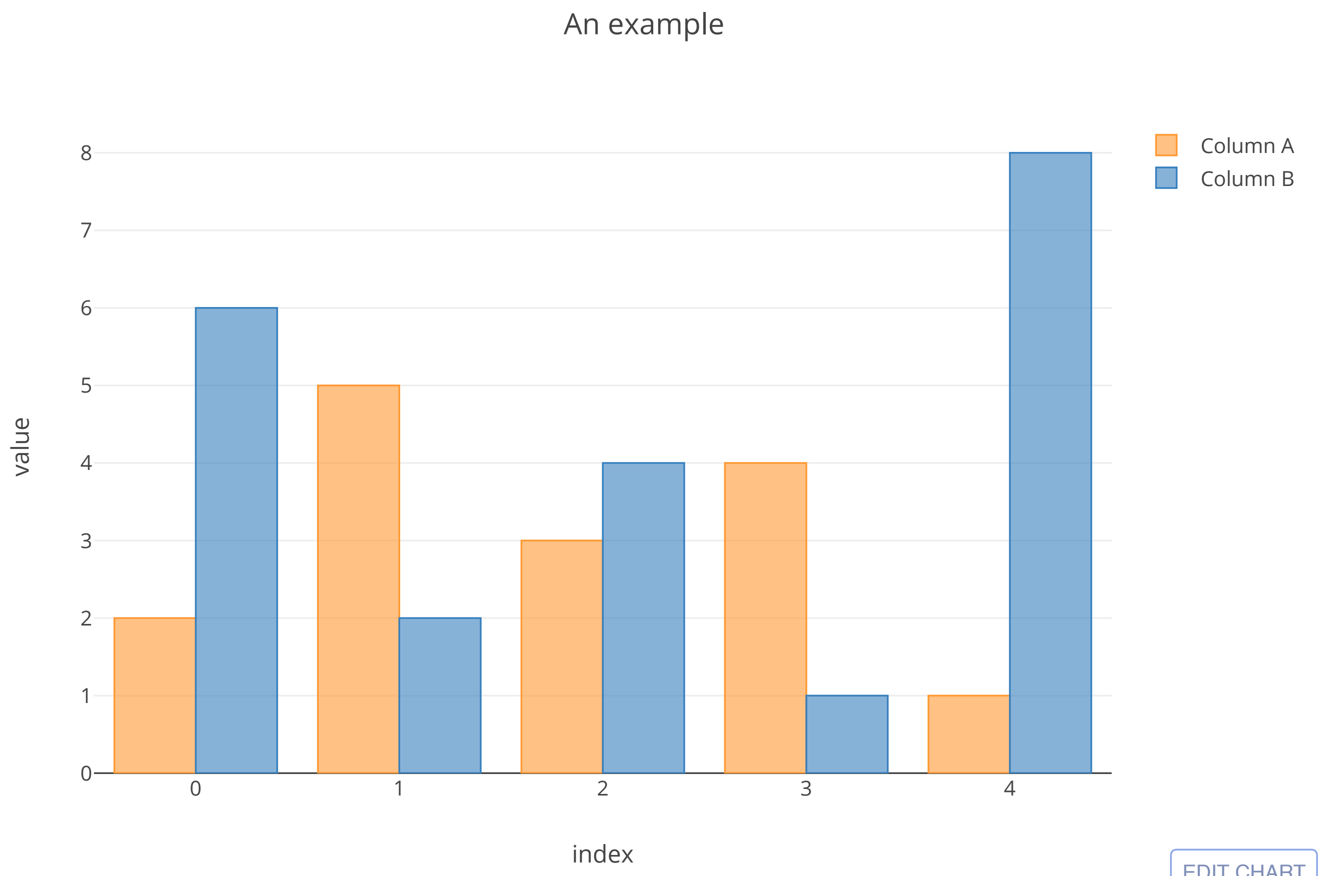
```
# Set layout assets individually:
df.iplot(kind='bar',
         title='An example',
         xTitle='index',
         yTitle='value')
```

```
# Set layout as a whole:
layout = dict(title='An example',
              xaxis={title='index'},
              yaxis={title='value'})

df.iplot(kind='bar',
         layout=layout)
```

- the data is taken from the dataframe
- the parameter **kind=** determines the trace type
- the layout assets can be specified as a whole or individually

Cufflinks

Plotly plots directly from Pandas Dataframes



EDIT CHART

Cufflinks

Plotly plots directly from Pandas Dataframes

```
# Use asFigure=True to return a Plotly figure...
fig = df.iplot(kind='bar',
               layout=layout,
               asFigure=True)



# ... that can be manipulated as such:
div_str = py.offline.plot(fig,
                          output_type='div',
                          include_plotlyjs=False)
```

Alternatives

- bokeh: from Anaconda (f.k.a. Continuum Analytics).
- pandas-highcharts: from Pandas dataframes to Highcharts viz.
- bqplot: Plotting library for IPython/Jupyter Notebooks.
- MPLD3: generate D3 visualisations from matplotlib graphics.