plotly

Show Sidebar

Fork on Github

Cufflinks in Python

An overview of cufflinks, a library for easy interactive Pandas charting with Plotly.

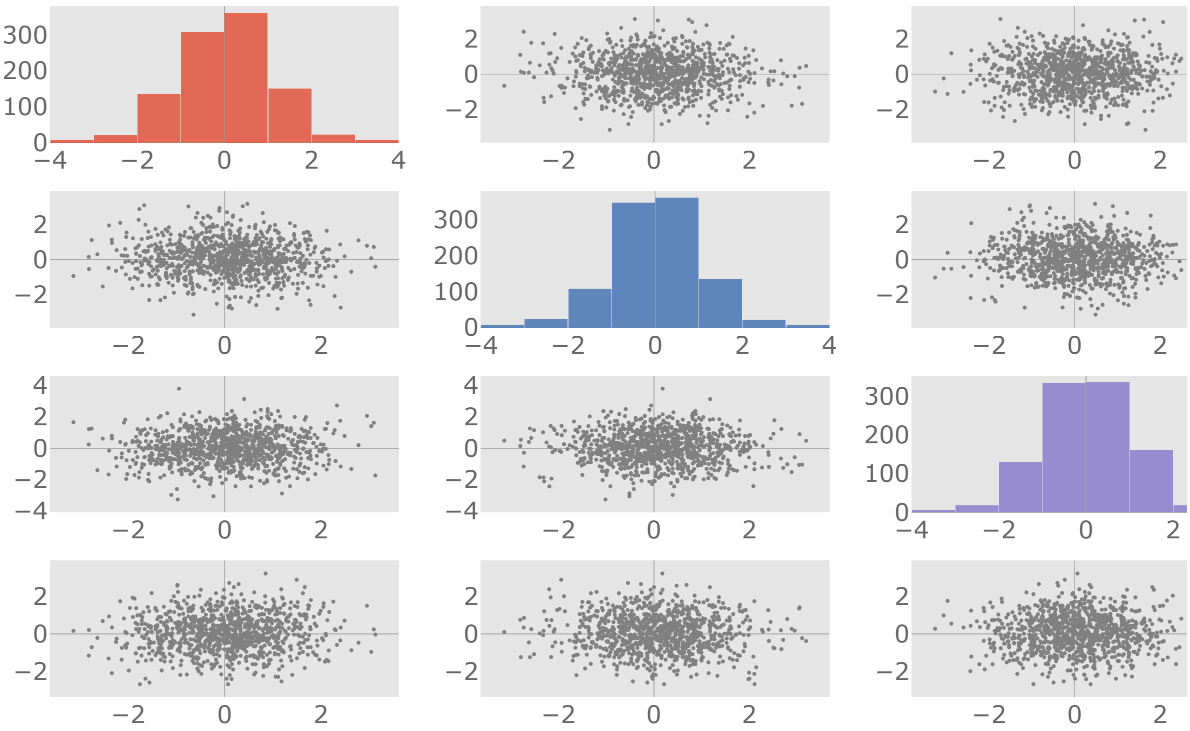
Cufflinks binds Plotly directly to pandas dataframes.

plotly

Show Sidebar

Fork on Github

Out[1]:



EDIT CHART

Packages

Run `! pip install cufflinks --upgrade` to install Cufflinks. In addition to Plotly, pandas and Cufflinks, this tutorial will also use NumPy.

```
import numpy as np
print cf.__version__
```

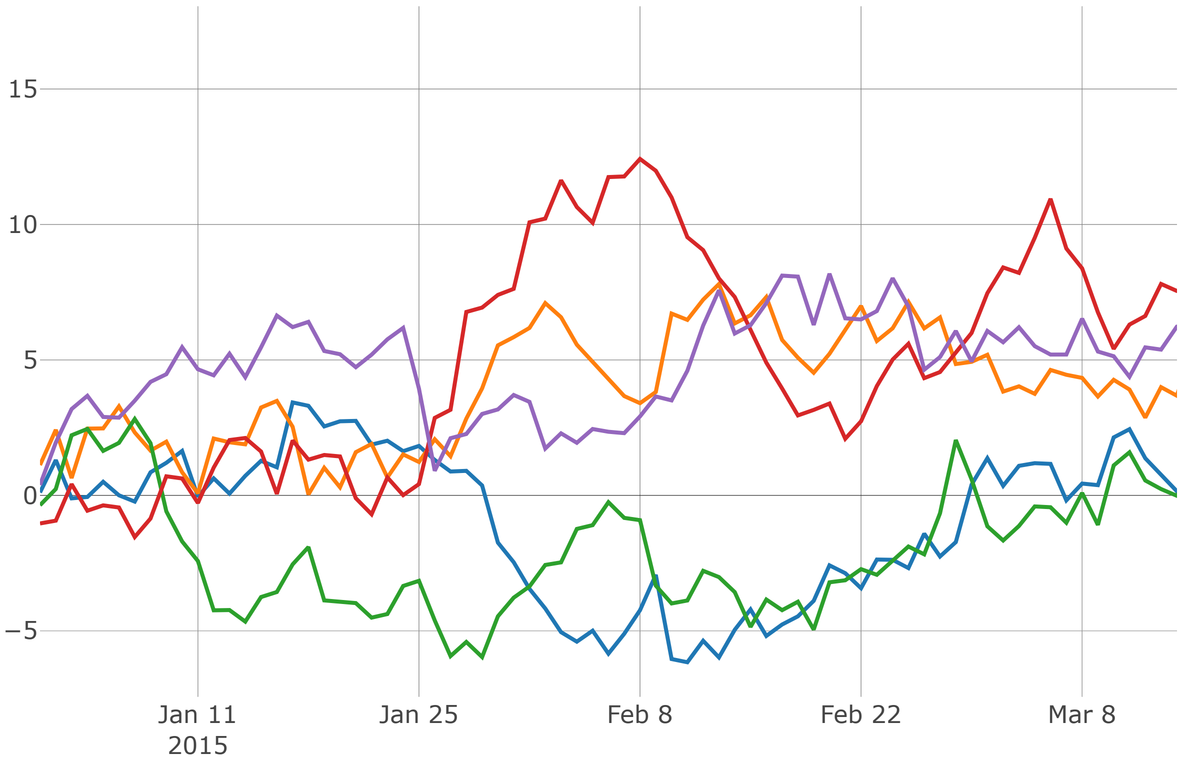
```
0.8.2
```

Dataframes

With Plotly's Python library, you can describe figures with DataFrame's series and index's

```
        'x': df.index,
        'y': df[col],
        'name': col
    }  for col in df.columns], filename='cufflinks/simple-line')
```

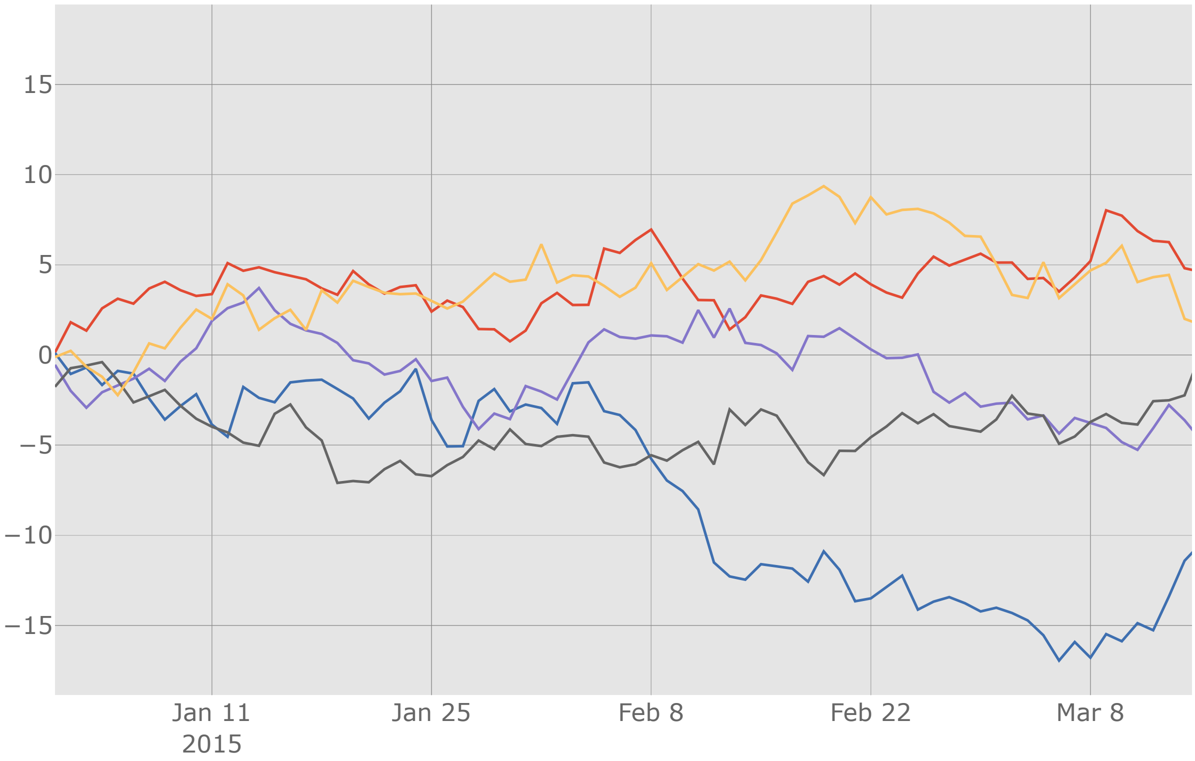
Out[3]:



EDIT CHART

But with `cufflinks`, you can plot them directly

Out[5]:



EDIT CHART

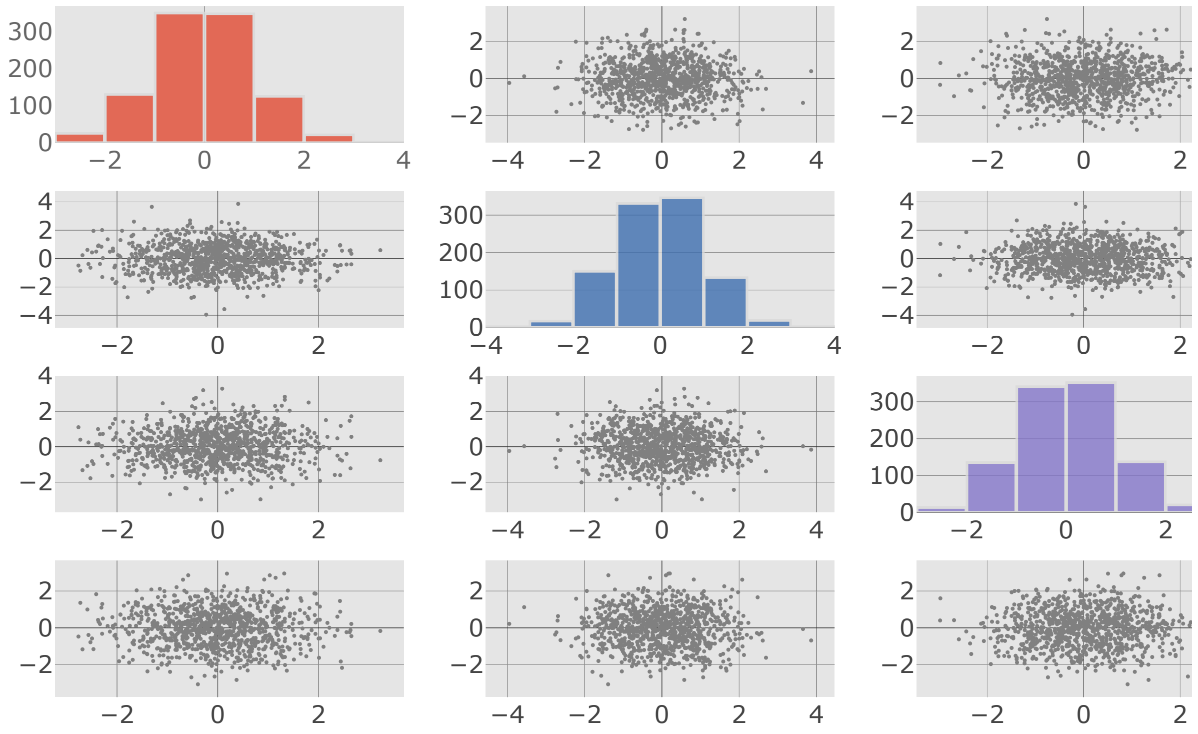
Almost every chart that you make in `cufflinks` will be created with just one line of code.

plotly

Show Sidebar

Fork on Github

Out[6]:



EDIT CHART

Charts created with `cufflinks` are synced with your online Plotly account. You'll need to configure your credentials to get started. `cufflinks` can also be configured to work offline in IPython notebooks with Plotly Offline. To get started with Plotly Offline, download a trial library and run `cf.go_offline()`.

In [14]:

```
cf.go_online() # switch back to online mode, where graphs are saved on your
 online plotly account
```

In [15]:

```
df.a.iplot(kind='histogram', world_readable=False)
```

Out[15]:

4 4

Plot twist!

There's nothing here...

You might need to sign in to see this plot.
Chart Studio is free to view plots that have been shared with you.

Only *you* (the creator) will be able to see this chart, or change the global, default settings with `cf.set_config_file`

In [16]:

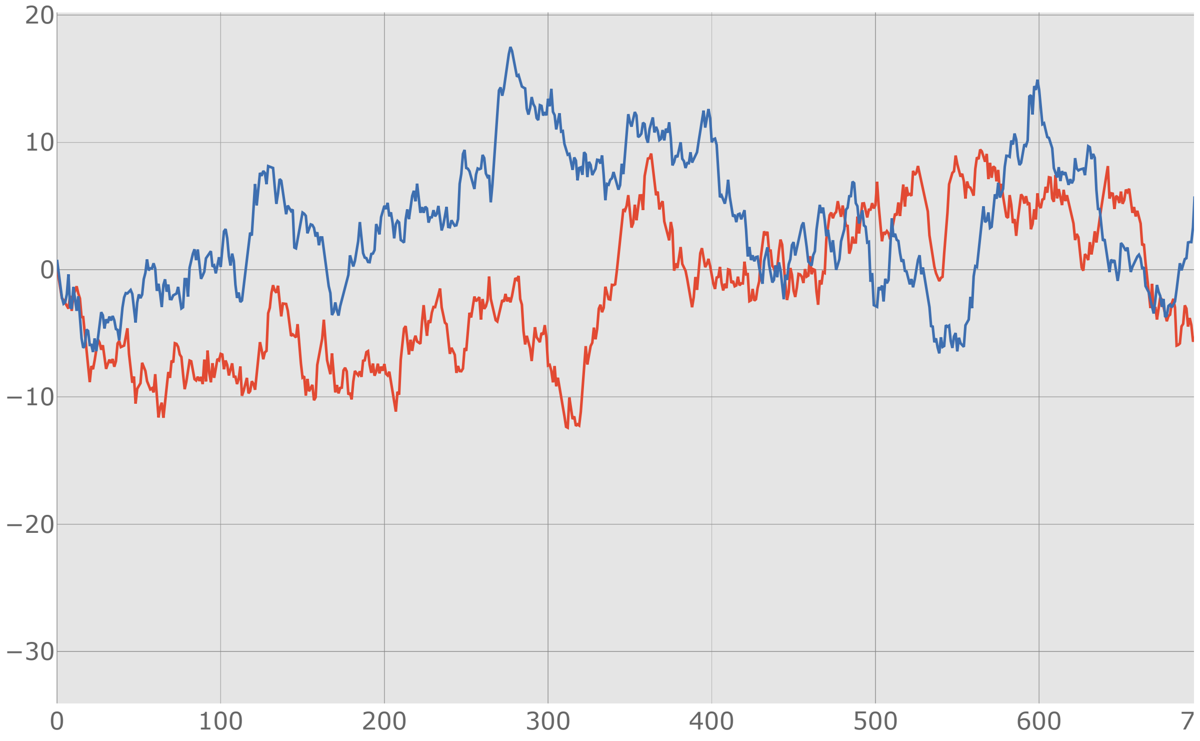
```
cf.set_config_file(offline=False, world_readable=True, theme='ggplot')
```

Line Charts

In [17]:

```
df = pd.DataFrame(np.random.randn(1000, 2), columns=['A', 'B']).cumsum()
df.iplot(filename='cufflinks/line-example')
```

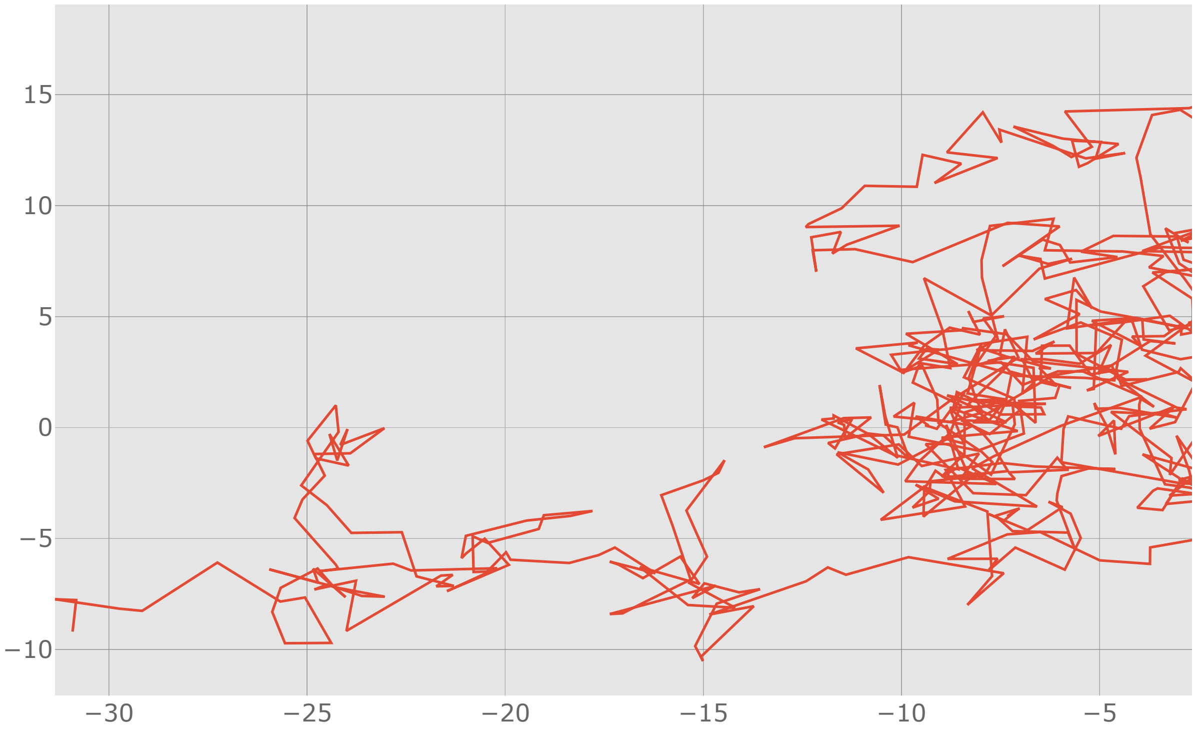
Out[17]:



EDIT CHART

Plot one column vs another with x and y keywords

Out[15]:



EDIT CHART

Bar Charts

Download some civic data. A time series log of the 311 complaints in NYC.

In [ ]:

```
df = pd.read_csv('https://raw.githubusercontent.com/plotly/widgets/master/ip
ython-examples/311_150k.csv', parse_dates=True, index_col=1)
df.head(3)
```

plotly

Show Sidebar

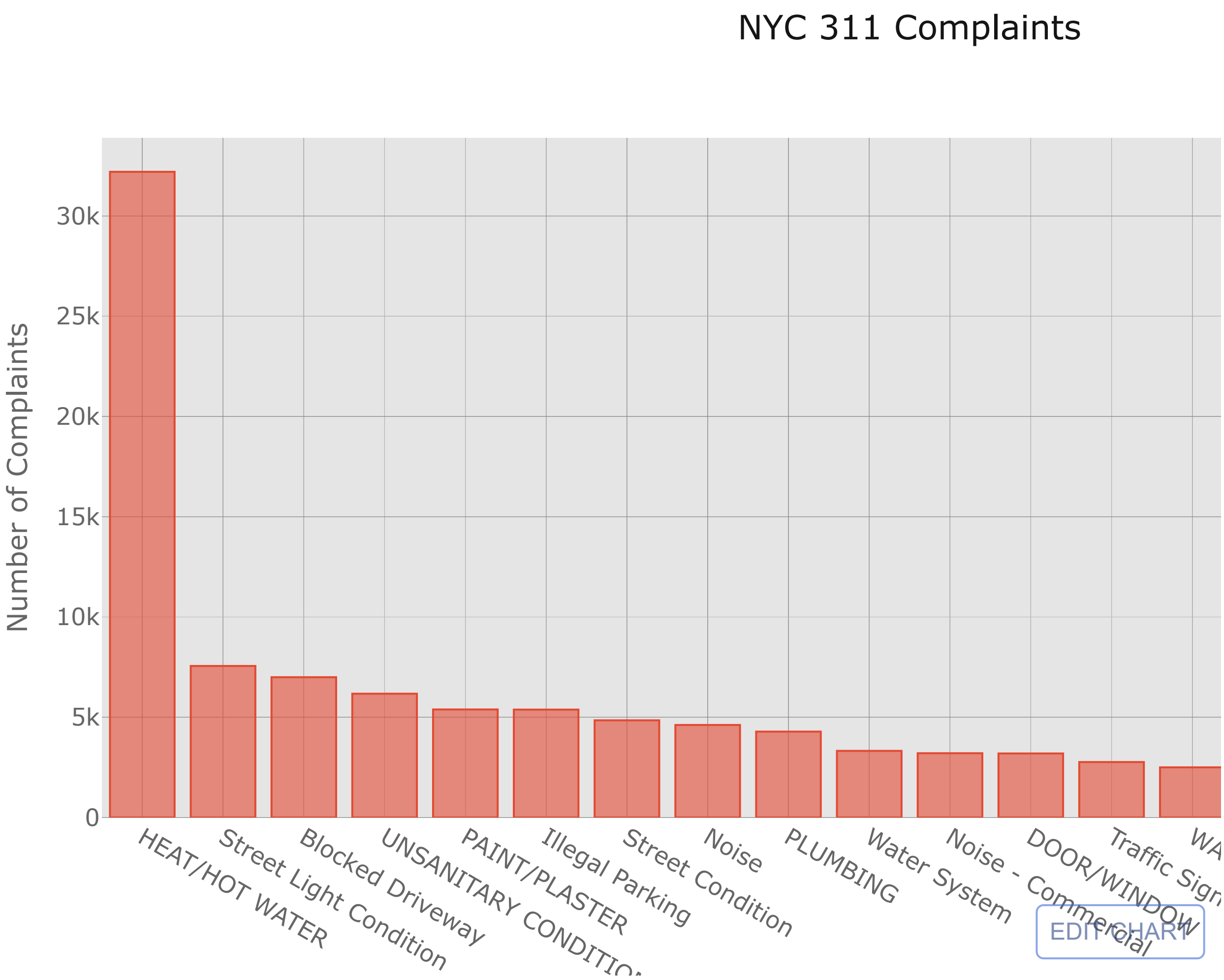
Fork on Github

Plot a series directly

In [18]:

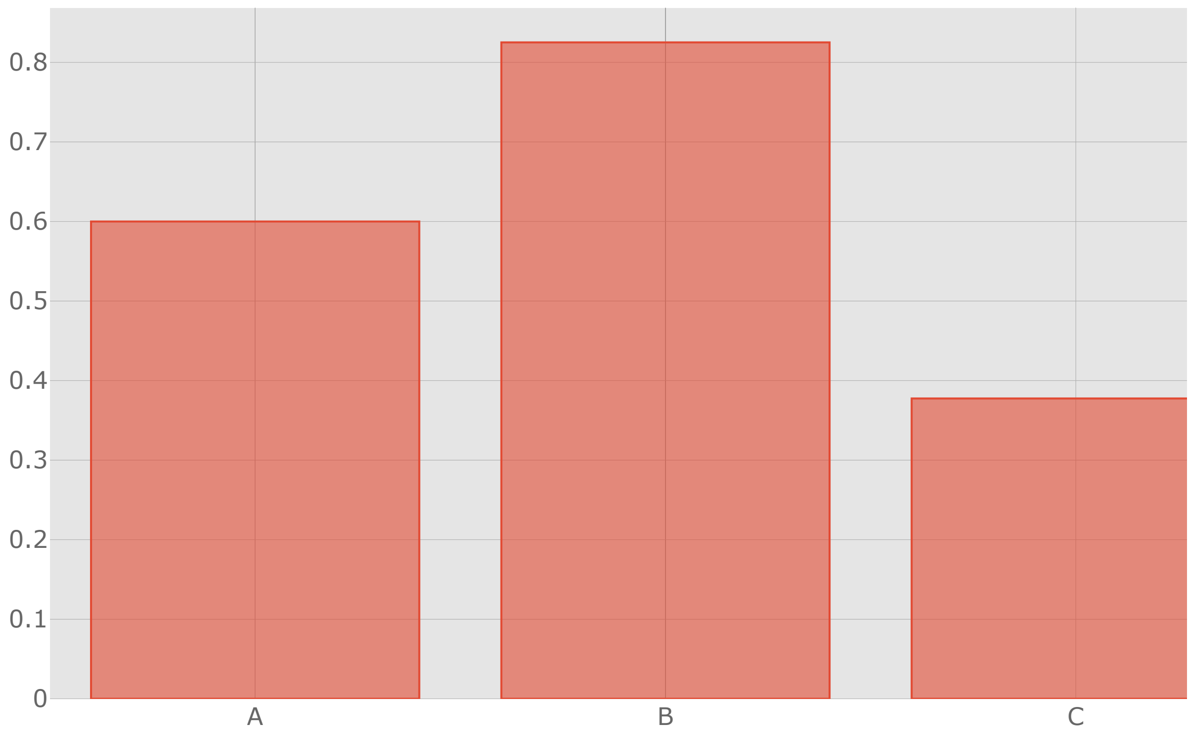
```
series.iplot(kind='bar', yTitle='Number of Complaints', title='NYC 311 Complaints',
             filename='cufflinks/categorical-bar-chart')
```

Out[18]:



```
df = pd.DataFrame(np.random.rand(10, 4), columns=['A', 'B', 'C', 'D'])
row = df.ix[5]
row.iplot(kind='bar', filename='cufflinks/bar-chart-row')
```

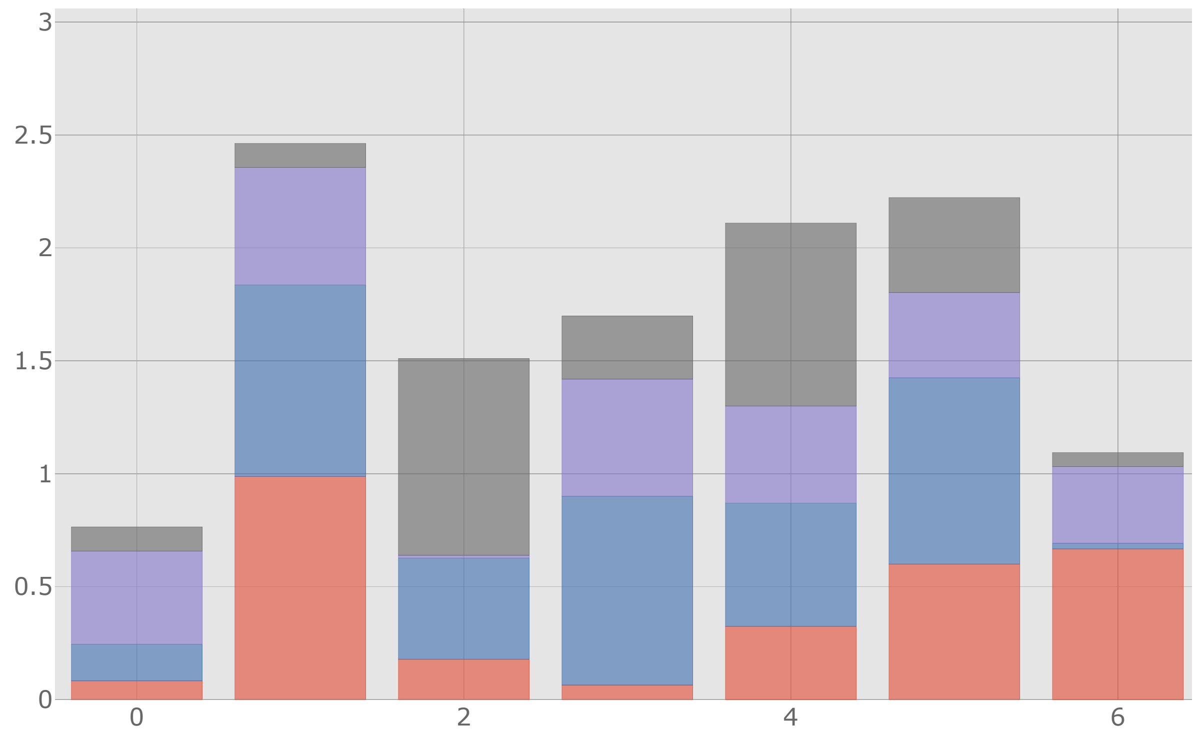
Out[19]:



EDIT CHART

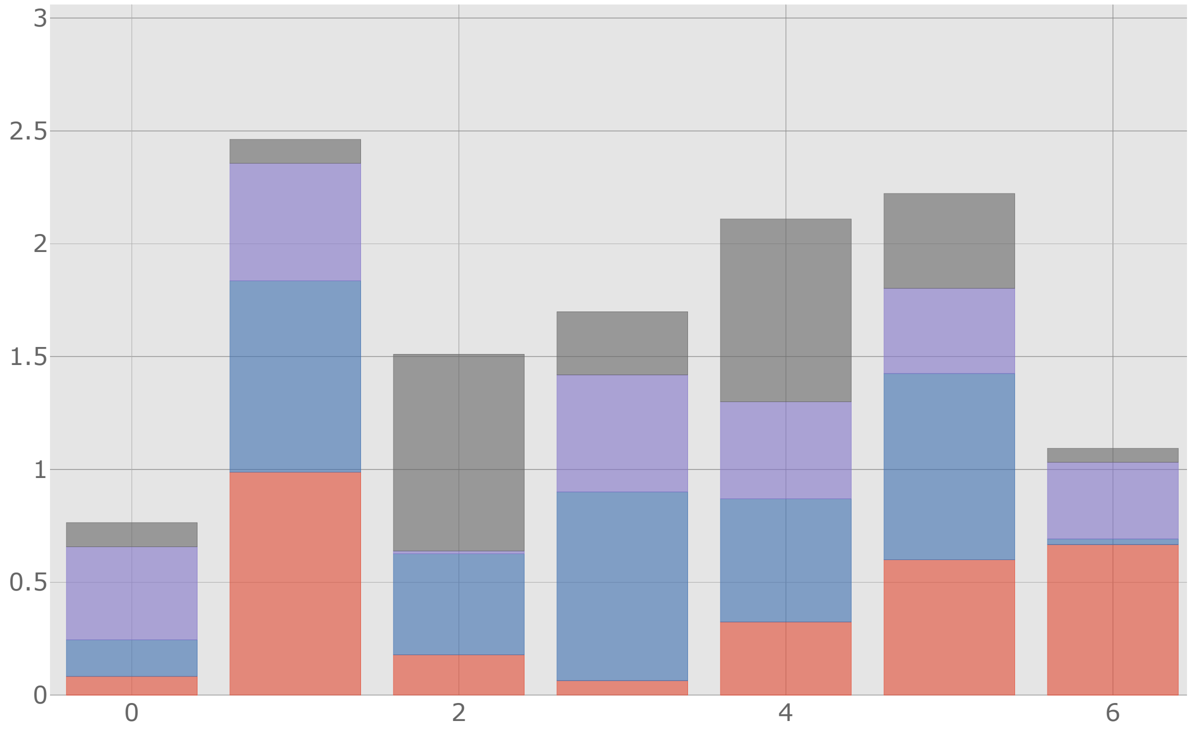
Call `iplot(kind='bar')` on a dataframe to produce a grouped bar chart

Out[[illegible]]:

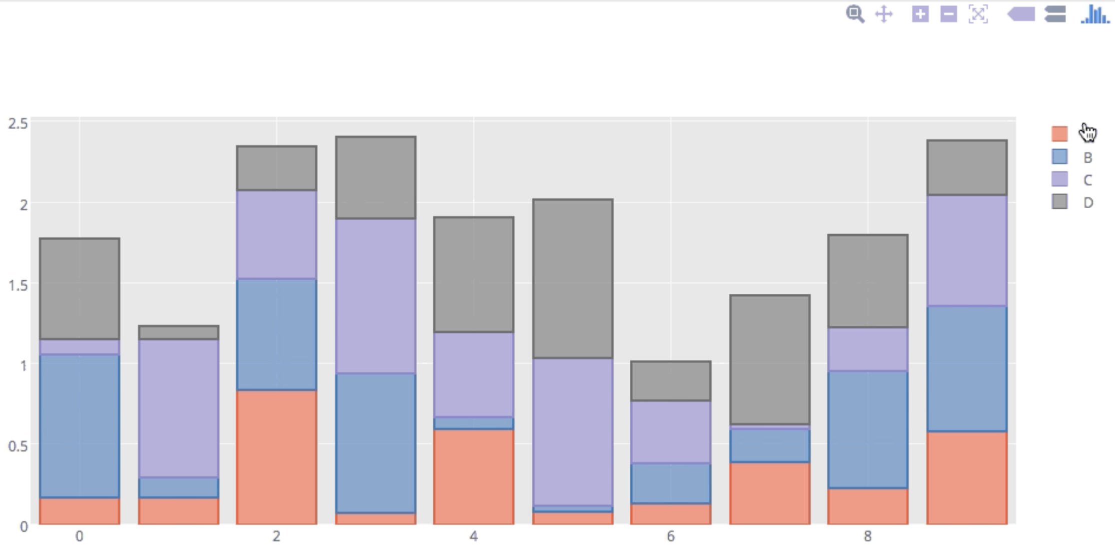


EDIT CHART

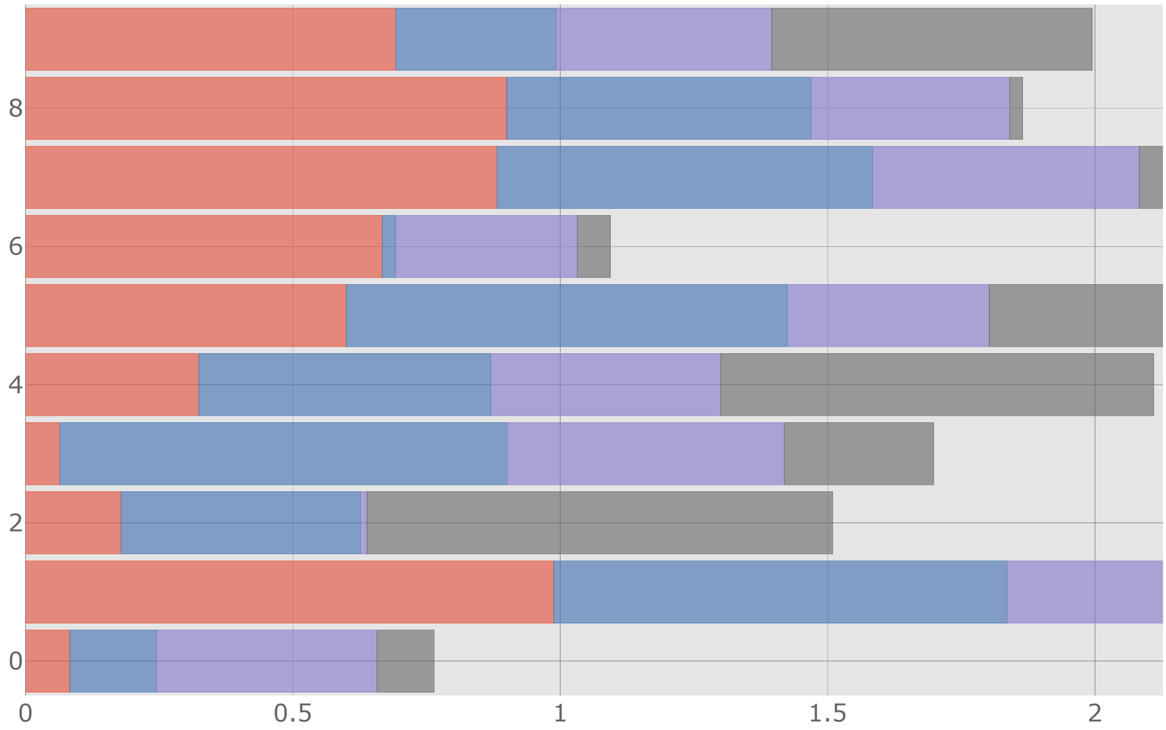
Out[21]:



EDIT CHART



Make your bar charts horizontal with `kind='barh'`



EDIT CHART

Themes

cufflinks ships with a few themes. View available themes with `cf.getThemes`, apply them with `cf.set_config_file`

```
['pearl', 'white', 'ggplot', 'solar', 'space']
```

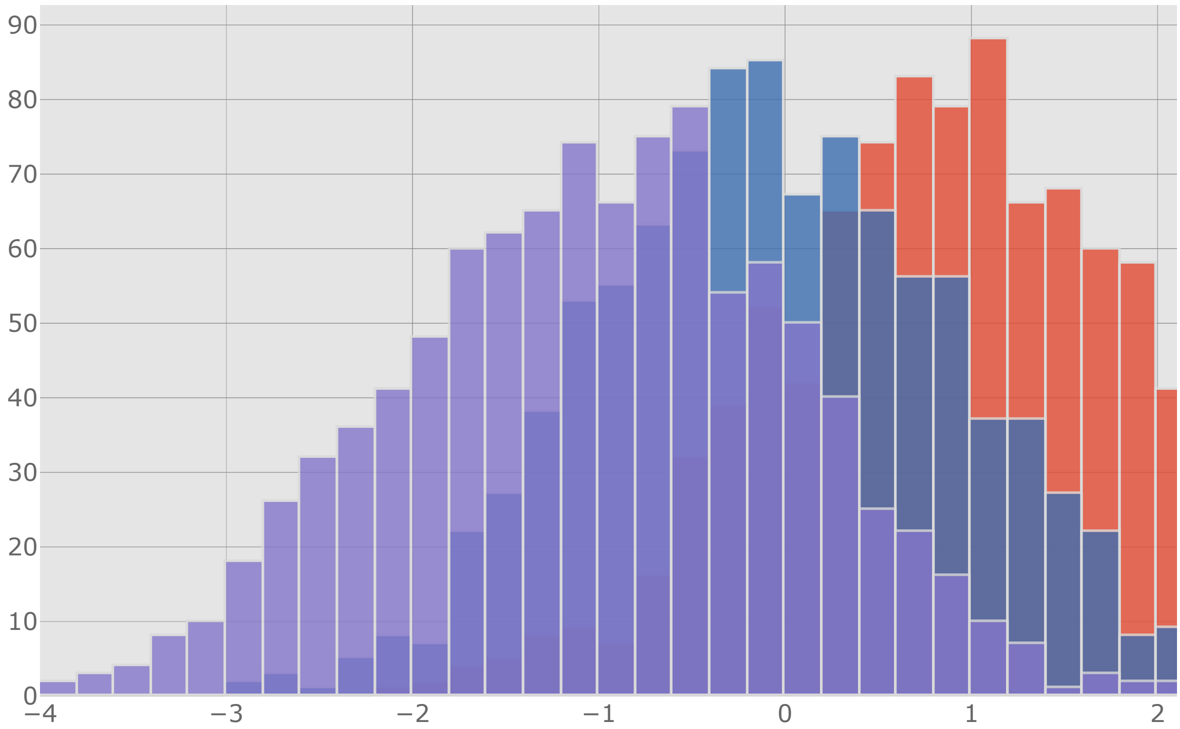
In [24]:

```
cf.set_config_file(theme='pearl')
```

Histograms

```
df.iplot(kind='histogram', filename='cufflinks/basic-histogram')
```

Out[4]:



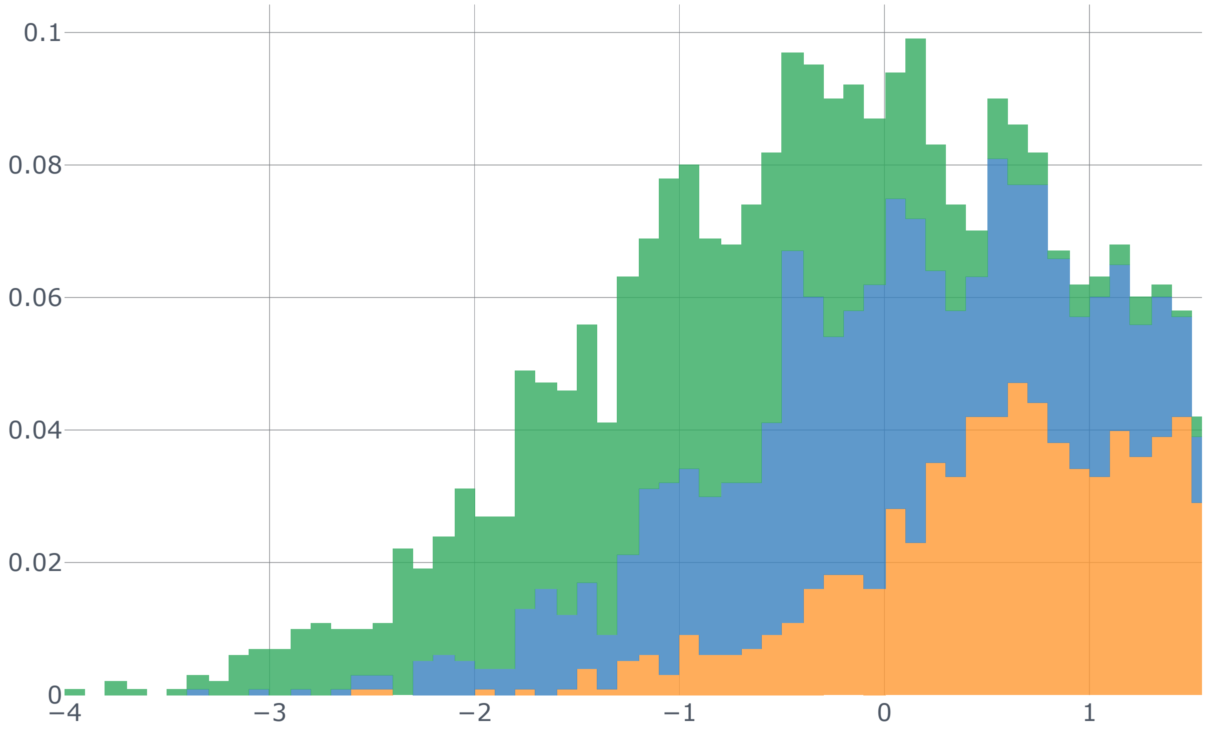
EDIT CHART

- `bins` (`int`)
- `histnorm` (`''` | `'percent'` | `'probability'` | `'density'` | `'probability density'`)
- `histfunc` (`'count'` | `'sum'` | `'avg'` | `'min'` | `'max'`)

In [27]:

```
df.iplot(kind='histogram', barmode='stack', bins=100, histnorm='probability'
, filename='cufflinks/customized-histogram')
```

Out[27]:

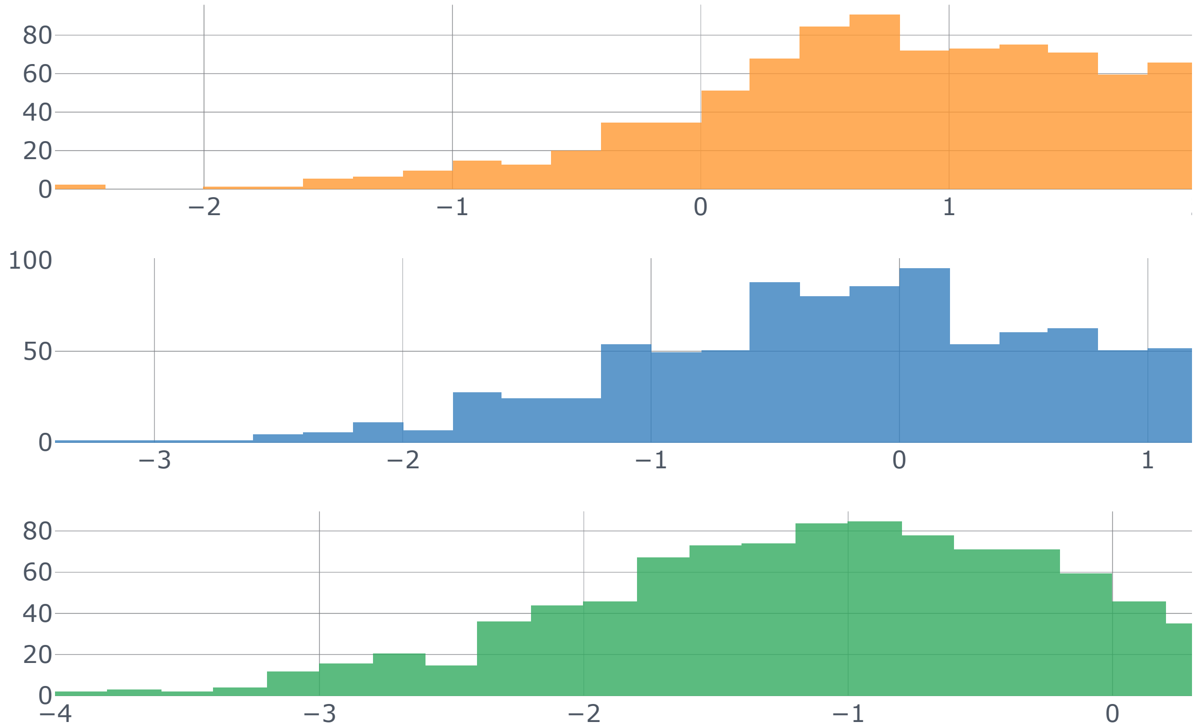


EDIT CHART

In [28]:

```
df.iplot(kind='histogram', subplots=True, shape=(3, 1), filename='cufflinks/
histogram-subplots')
```

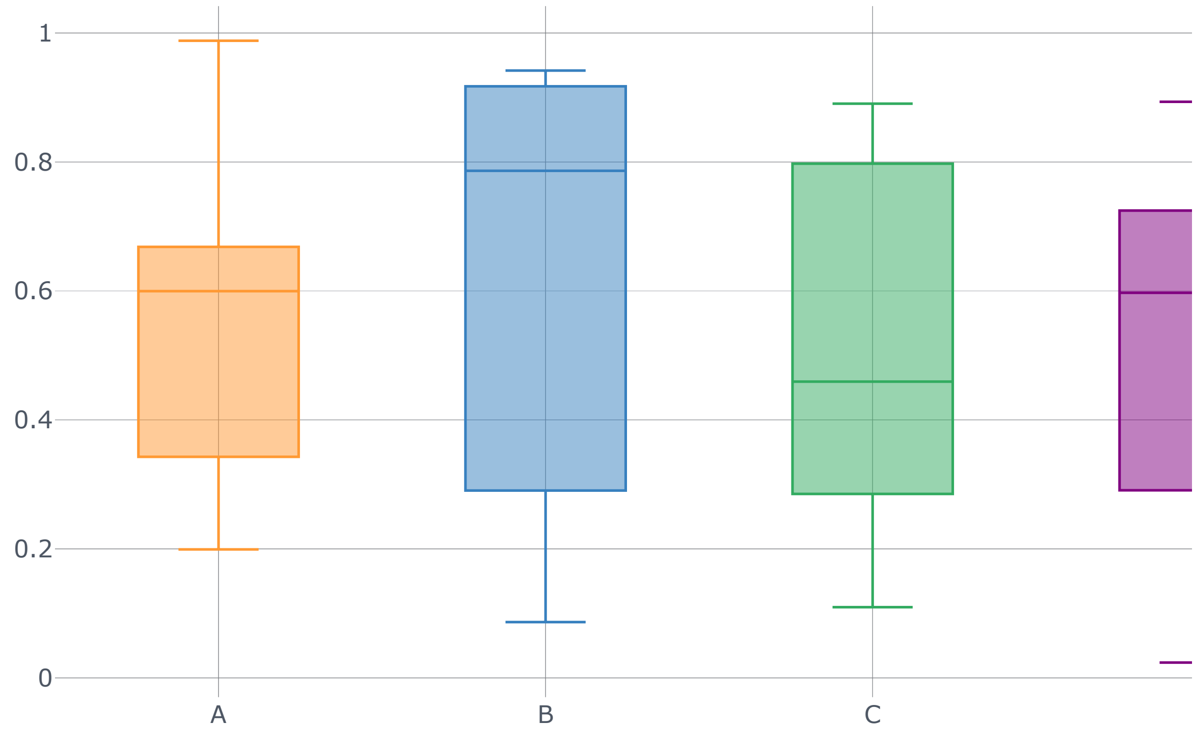
Out[28]:



EDIT CHART

Box Plots

```
Out[29]:
```



EDIT CHART

Area Charts

To produce stacked area plot, each column must be either all positive or all negative values.

When input data contains `NaN`, it will be automatically filled by 0. If you want to drop or fill by different values, use `dataframe.dropna()` or `dataframe.fillna()` before calling plot.

plotly

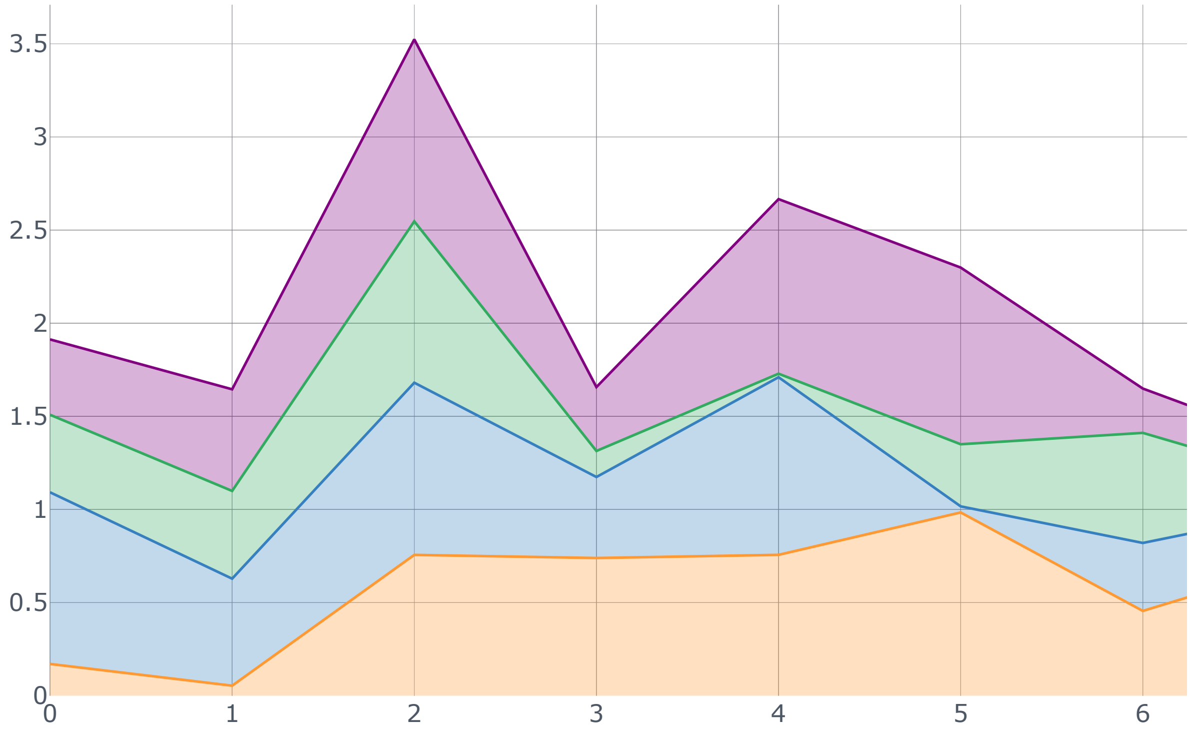
Show Sidebar

Fork on Github

In [31]:

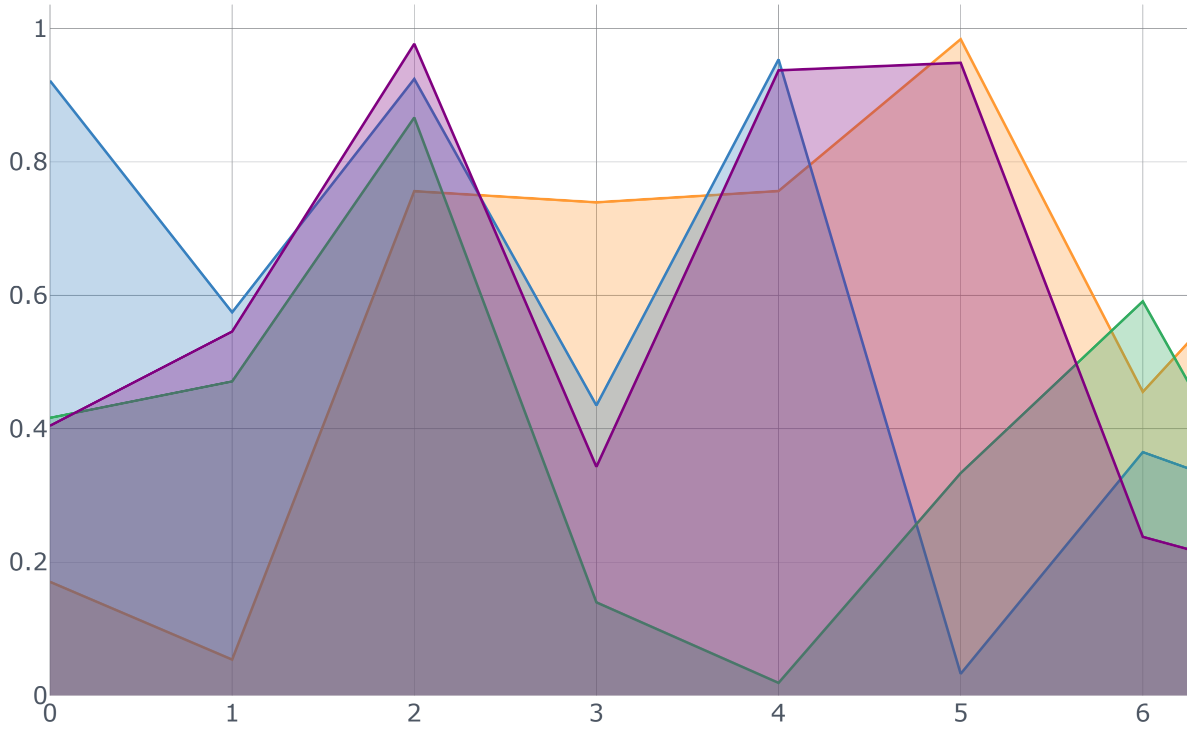
```
df.iplot(kind='area', fill=True, filename='cuflinks/stacked-area')
```

Out[31]:



EDIT CHART

For non-stacked area charts, set `kind=scatter` with `fill=True`



EDIT CHART

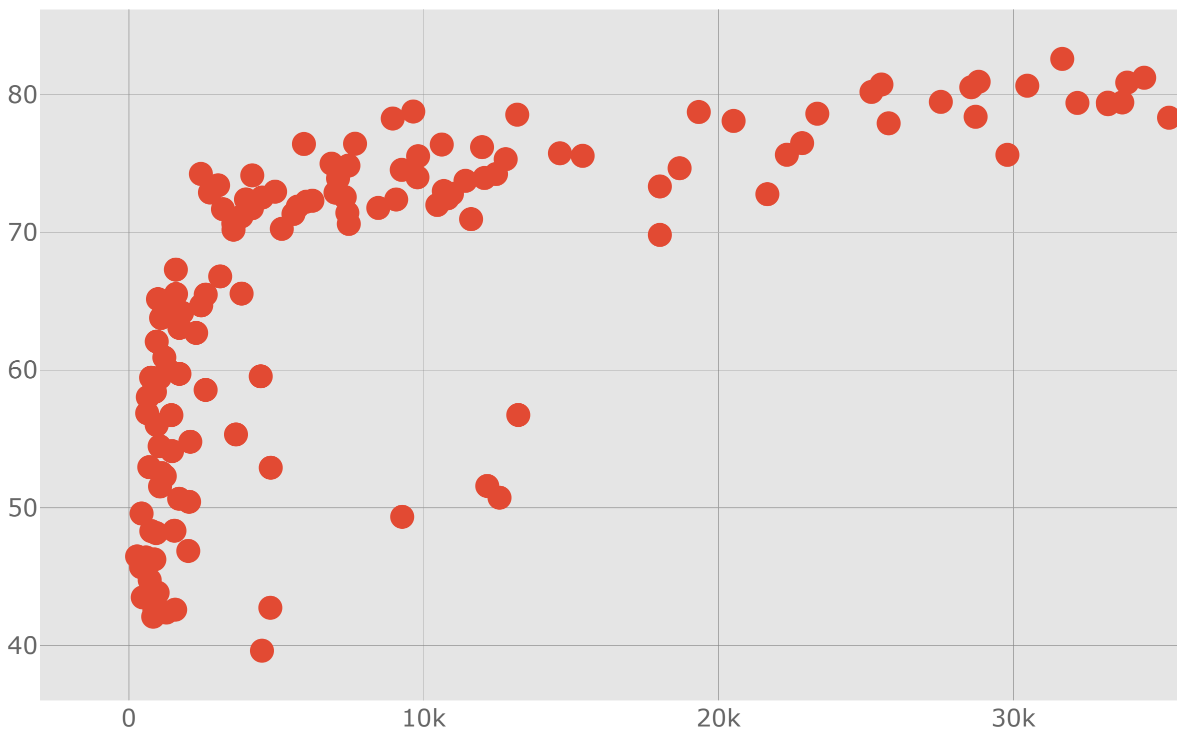
Scatter Plot

Set x and y as column names. If x isn't supplied, `df.index` will be used.

```
df2007 = df[df.year==2007]
df1952 = df[df.year==1952]

df2007.iplot(kind='scatter', mode='markers', x='gdpPercap', y='lifeExp', filename='cufflinks/simple-scatter')
```

Out[5]:

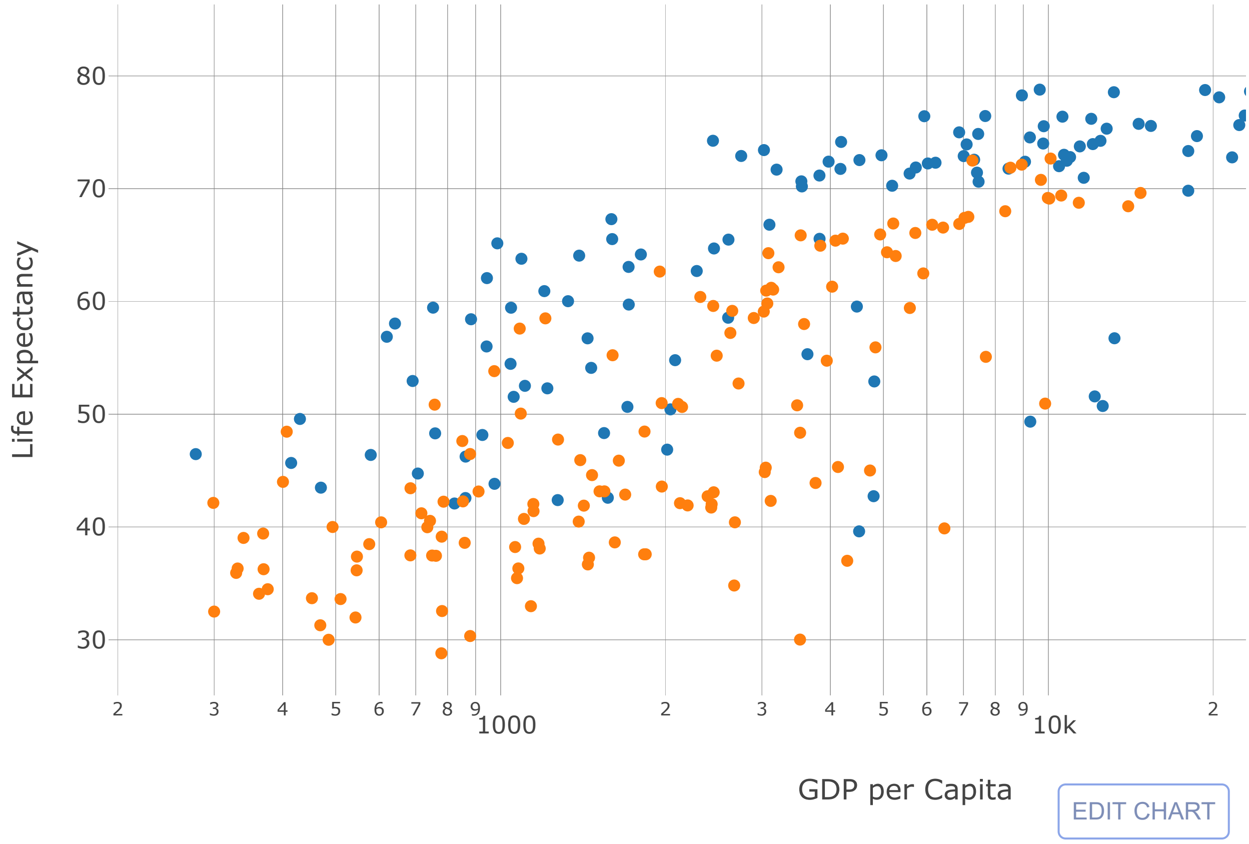


EDIT CHART

Plotting multiple column scatter plots isn't as easy with cufflinks. Here is an example with Plotly's native syntax

```
 'mode': 'markers', 'name': '2007'},
        {'x': df1952.gdpPercap, 'y': df1952.lifeExp, 'text': df1952.country,
 'mode': 'markers', 'name': '1952'}
    ],
    'layout': {
        'xaxis': {'title': 'GDP per Capita', 'type': 'log'},
        'yaxis': {'title': "Life Expectancy"}
    }
}
py.iplot(fig, filename='cufflinks/multiple-scatter')
```

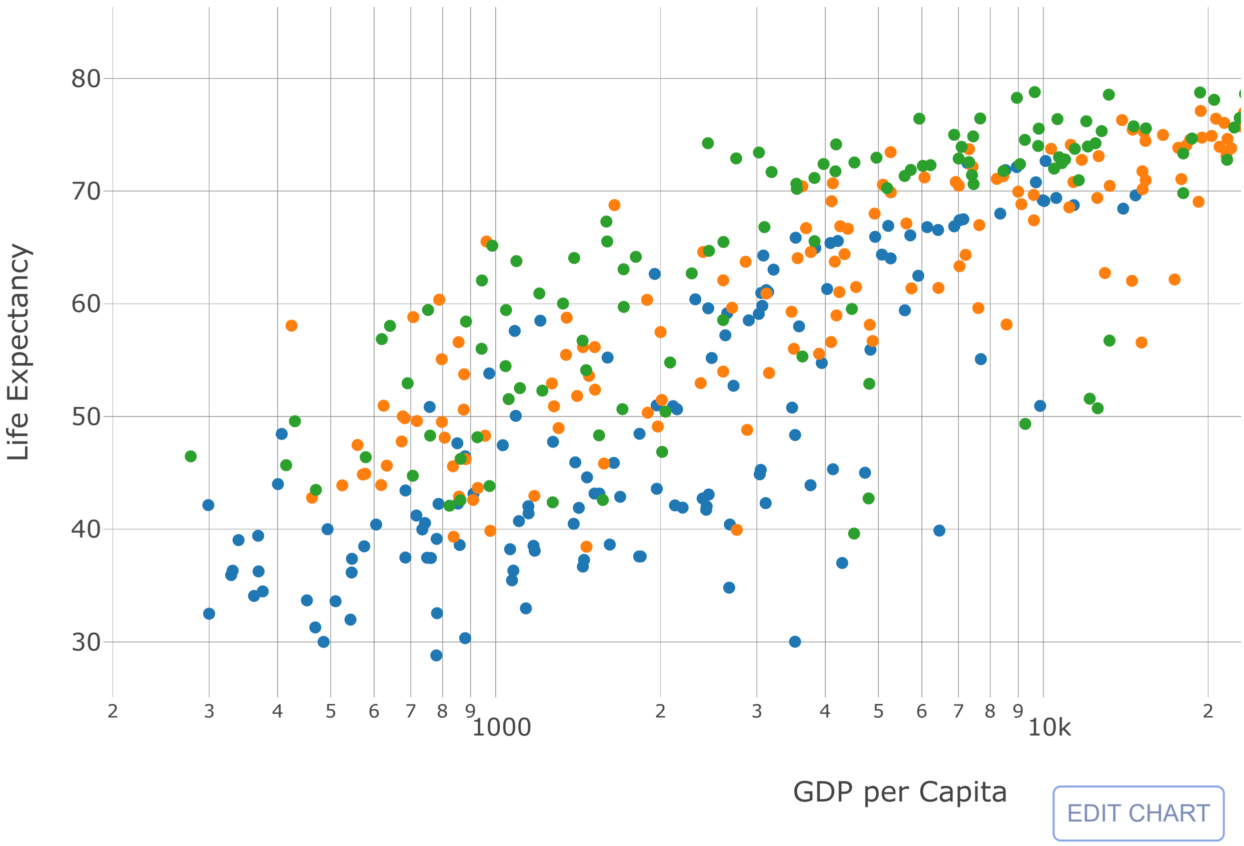
Out[35]:



Grouping isn't as easy either. But, with Plotly's native syntax:

```
            {
                'x': df[df['year']==year]['gdpPercap'],
                'y': df[df['year']==year]['lifeExp'],
                'name': year, 'mode': 'markers',
            } for year in [1952, 1982, 2007]
        ],
        'layout': {
            'xaxis': {'title': 'GDP per Capita', 'type': 'log'},
            'yaxis': {'title': "Life Expectancy"}
        }
    }, filename='cufflinks/scatter-group-by')
```

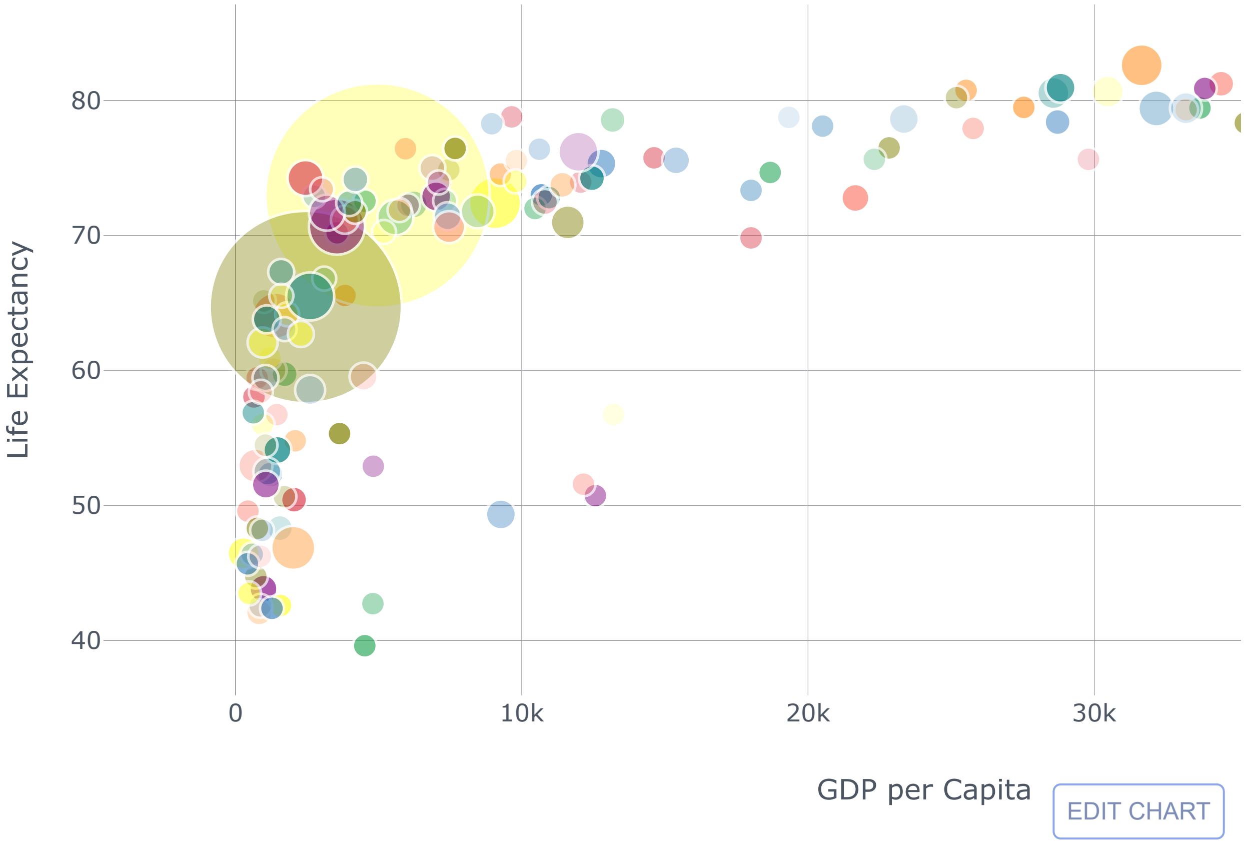
Out[36]:



In [37]:

```
df2007.iplot(kind='bubble', x='gdpPercap', y='lifeExp', size='pop', text='country',
             xTitle='GDP per Capita', yTitle='Life Expectancy',
             filename='cufflinks/simple-bubble-chart')
```

Out[37]:

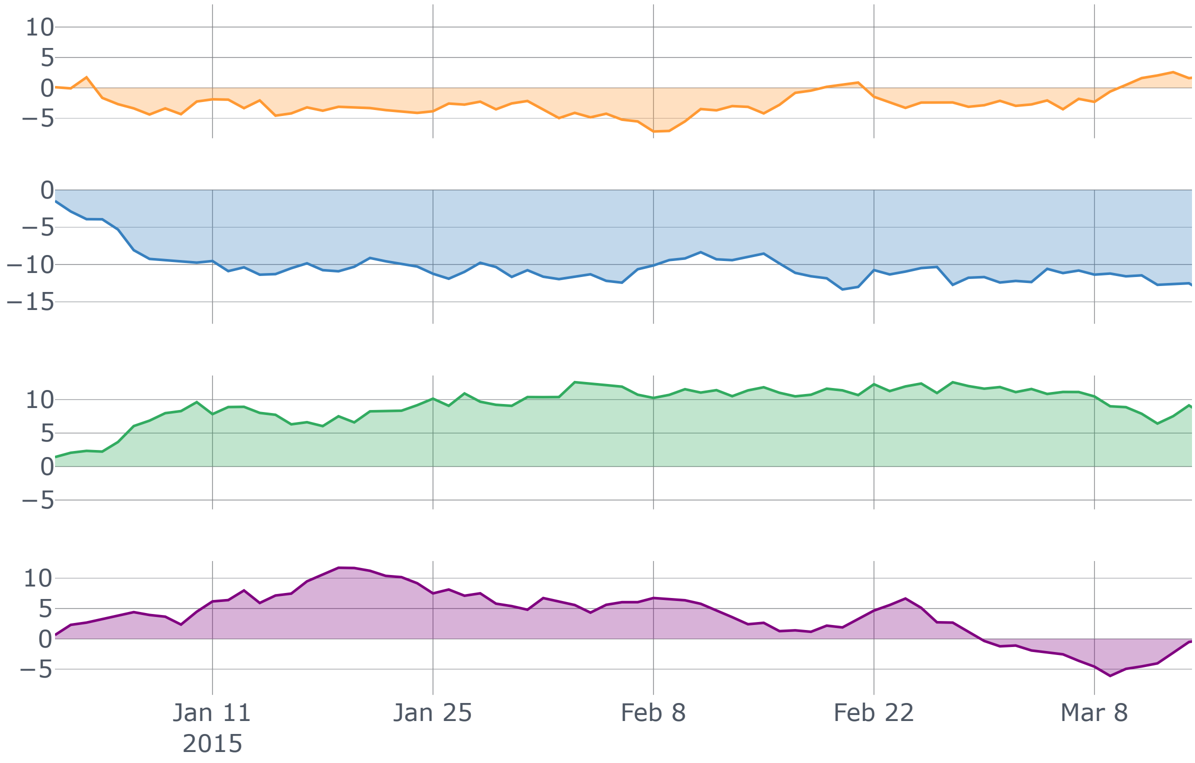


Subplots

In [38]:

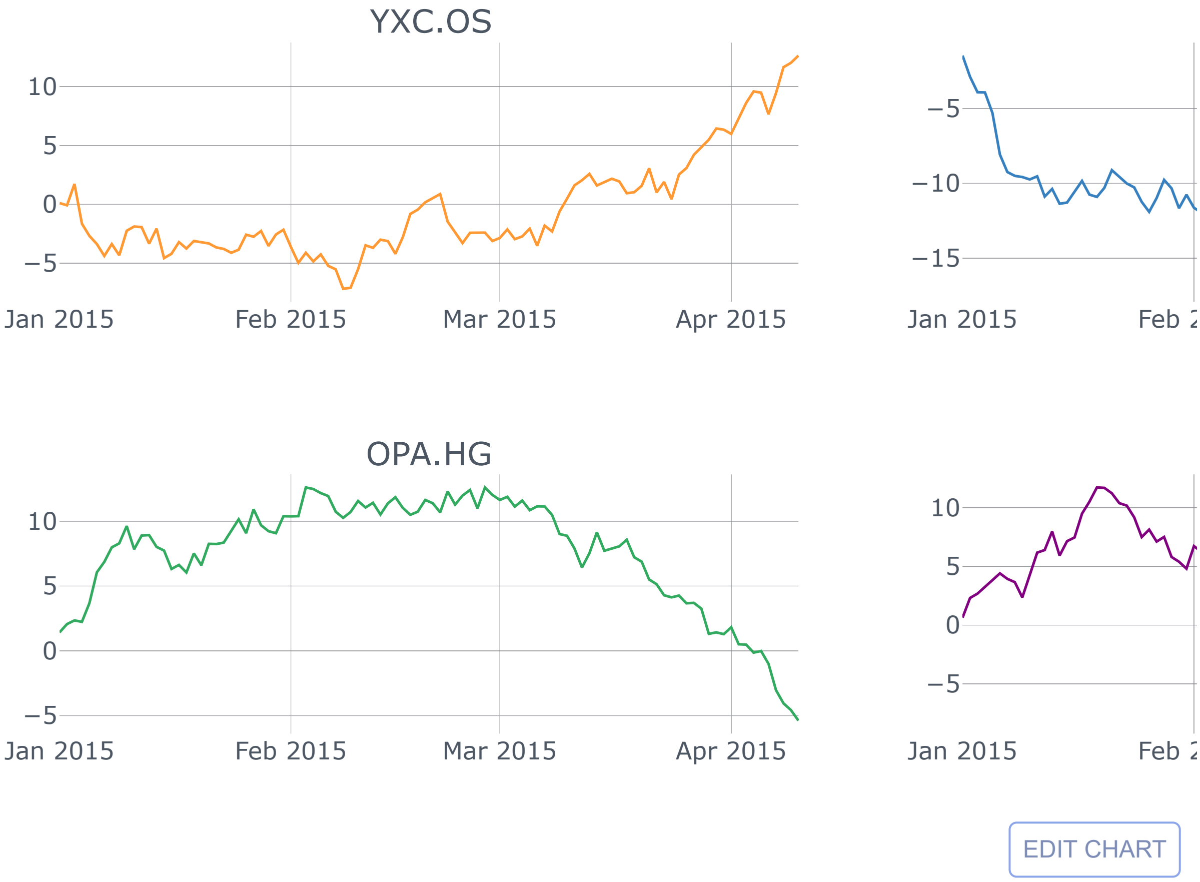
```
df=cf.datagen.lines(4)
df.iplot(subplots=True, shape=(4,1), shared_xaxes=True, fill=True, filename=
'cufflinks/simple-subplots')
```

Out[38]:



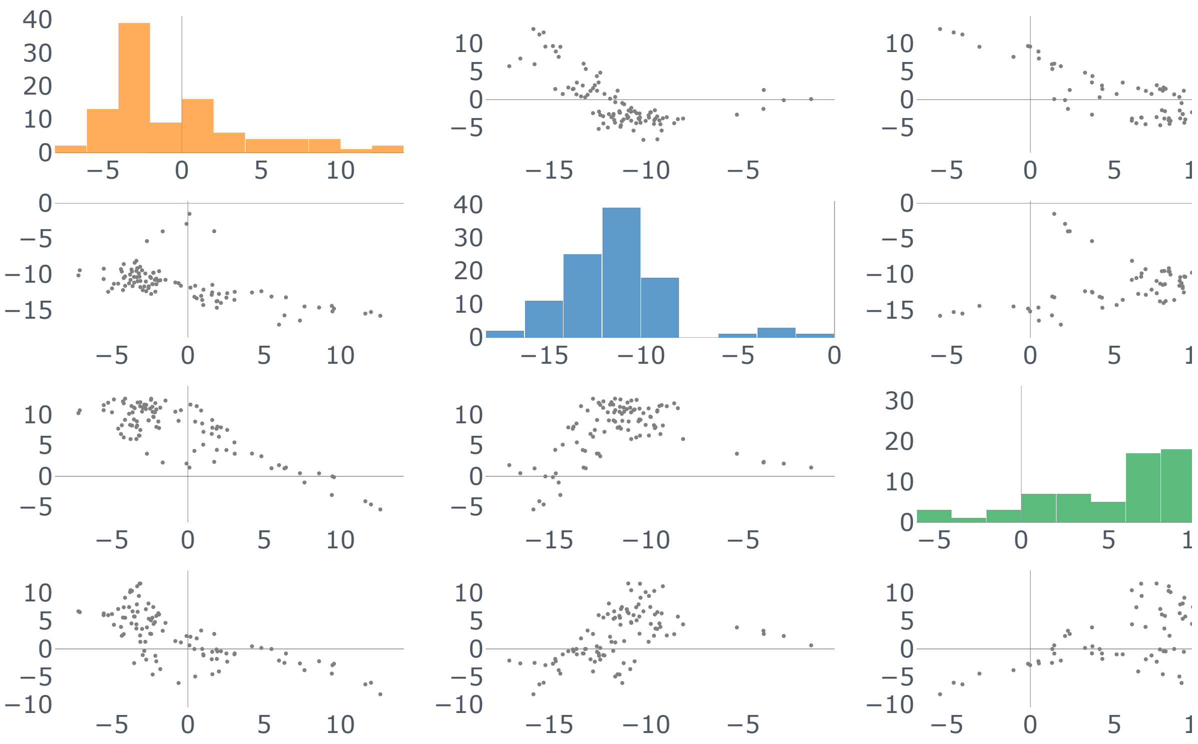
EDIT CHART

Add subplot titles with `subplot_titles` as a list of titles or `True` to use column names.



Scatter matrix

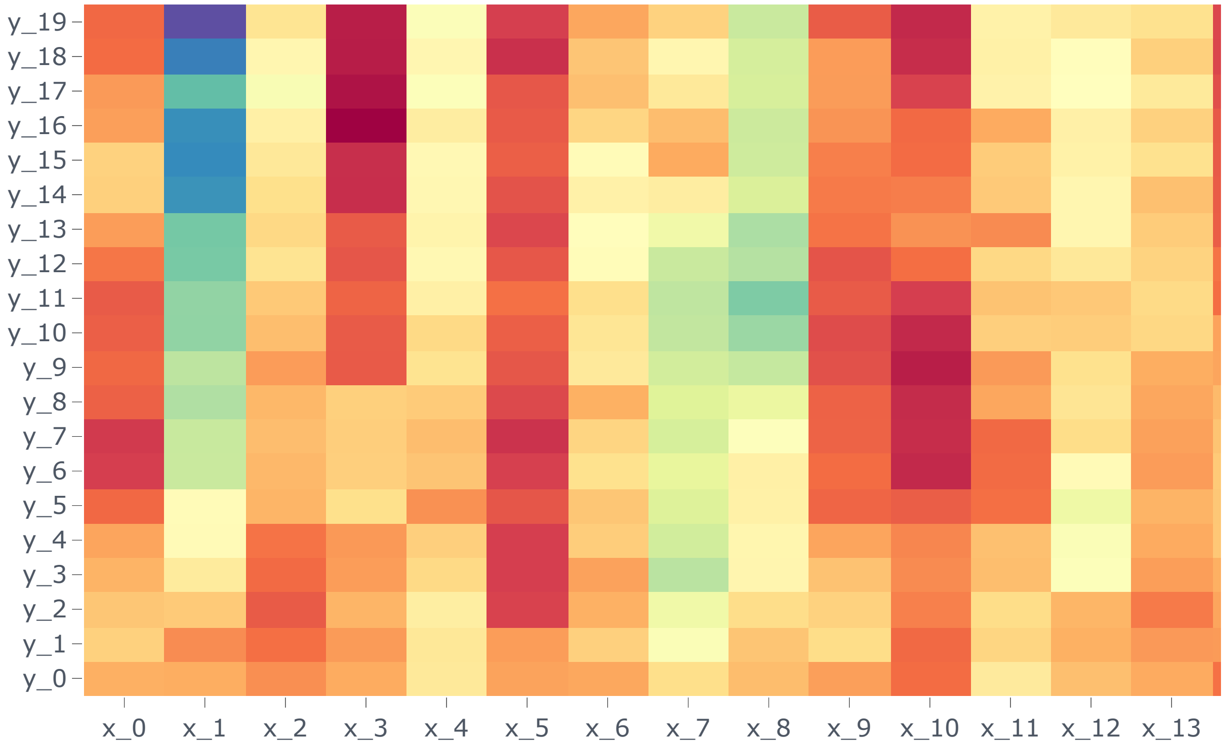
Out[40]:



EDIT CHART

Heatmaps

Out[41]:



EDIT CHART

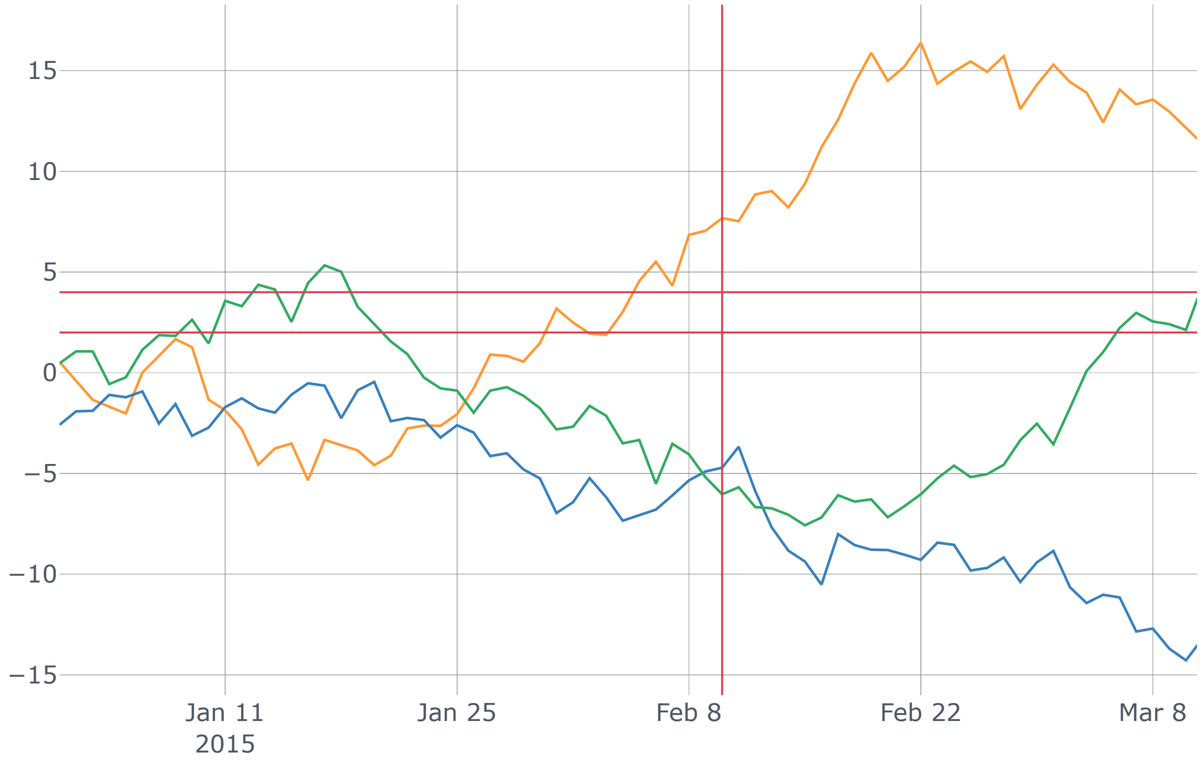
Lines and Shaded Areas

Use `hline` and `vline` for horizontal and vertical lines.

In [42]:

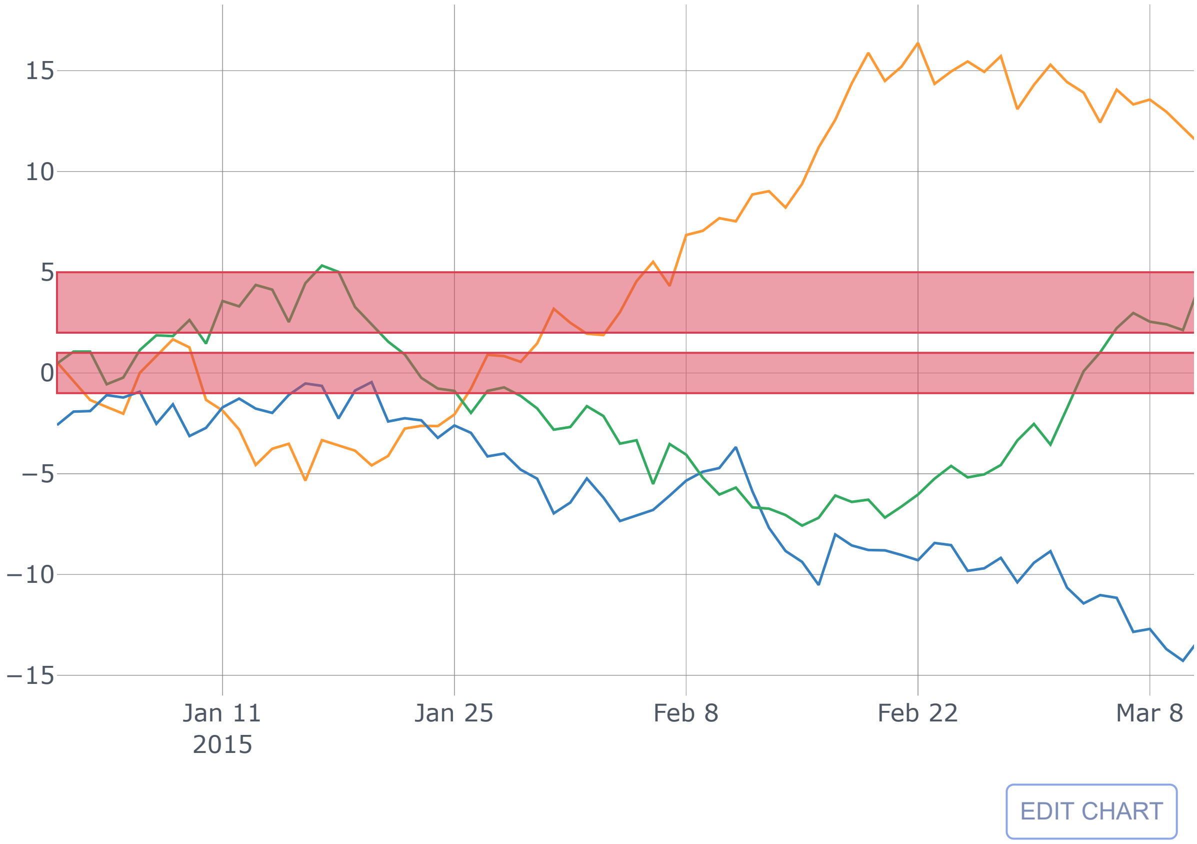
```
df=cf.datagen.lines(3,columns=['a','b','c'])
```

Out[45]:



EDIT CHART

Draw shaded regions with hspan



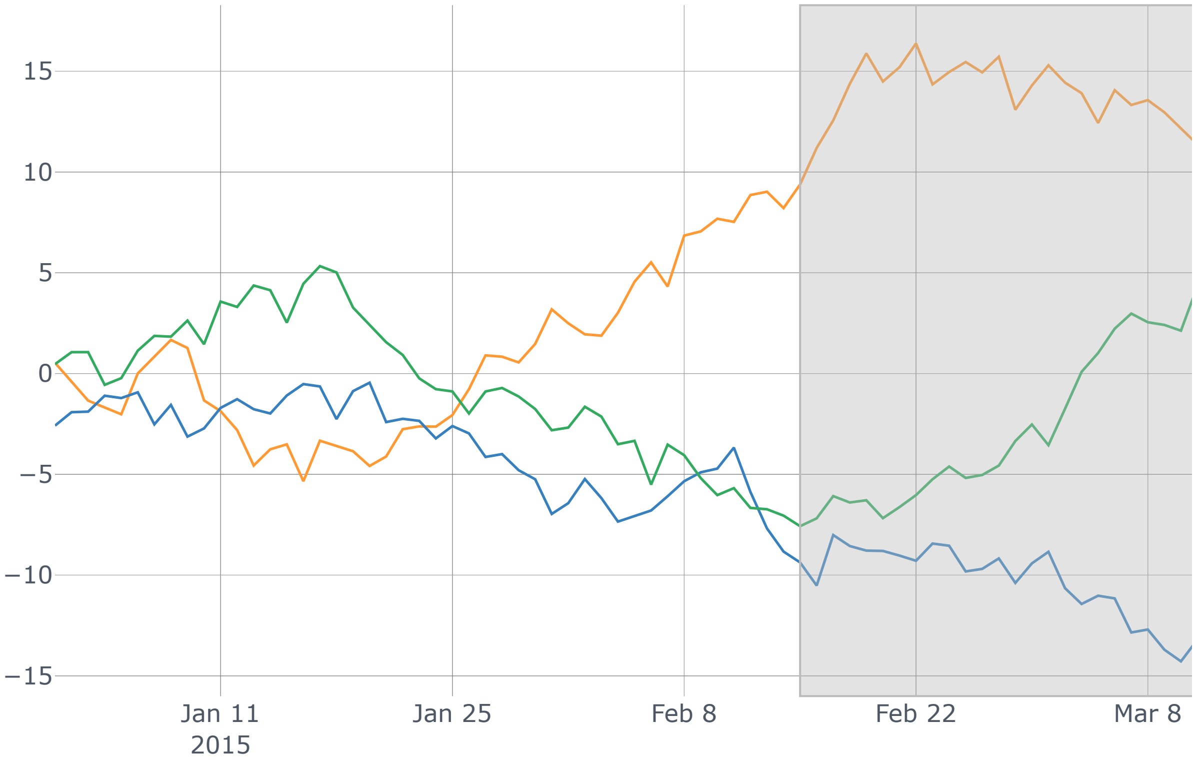
Extra parameters can be passed in the form of dictionaries, width, fill, color, fillcolor, opacity

plotly

Show Sidebar

Fork on Github

Out[45]:



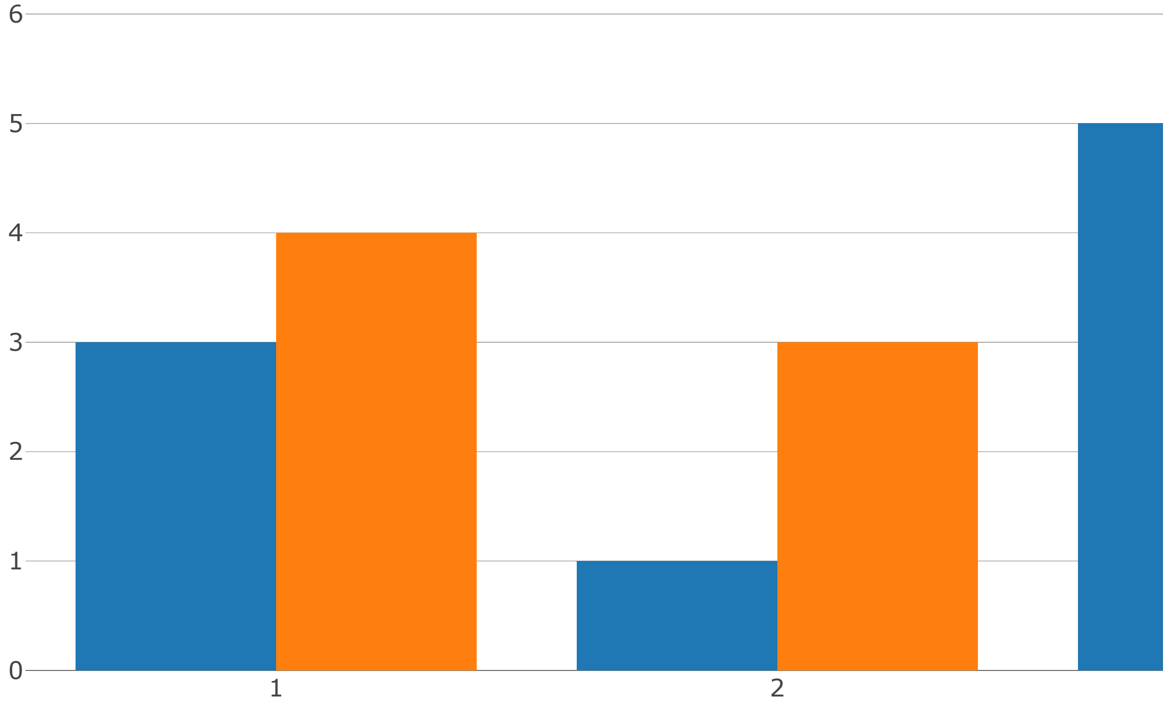
EDIT CHART

cufflinks is designed for simple one-line charting with Pandas and Plotly. All of the Plotly chart attributes are not directly assignable in the `df.iplot` call signature.

To update attributes of a `cufflinks` chart that aren't available, first convert it to a figure (`asFigure=True`), then tweak it, then plot it with `plotly.plotly.iplot`.

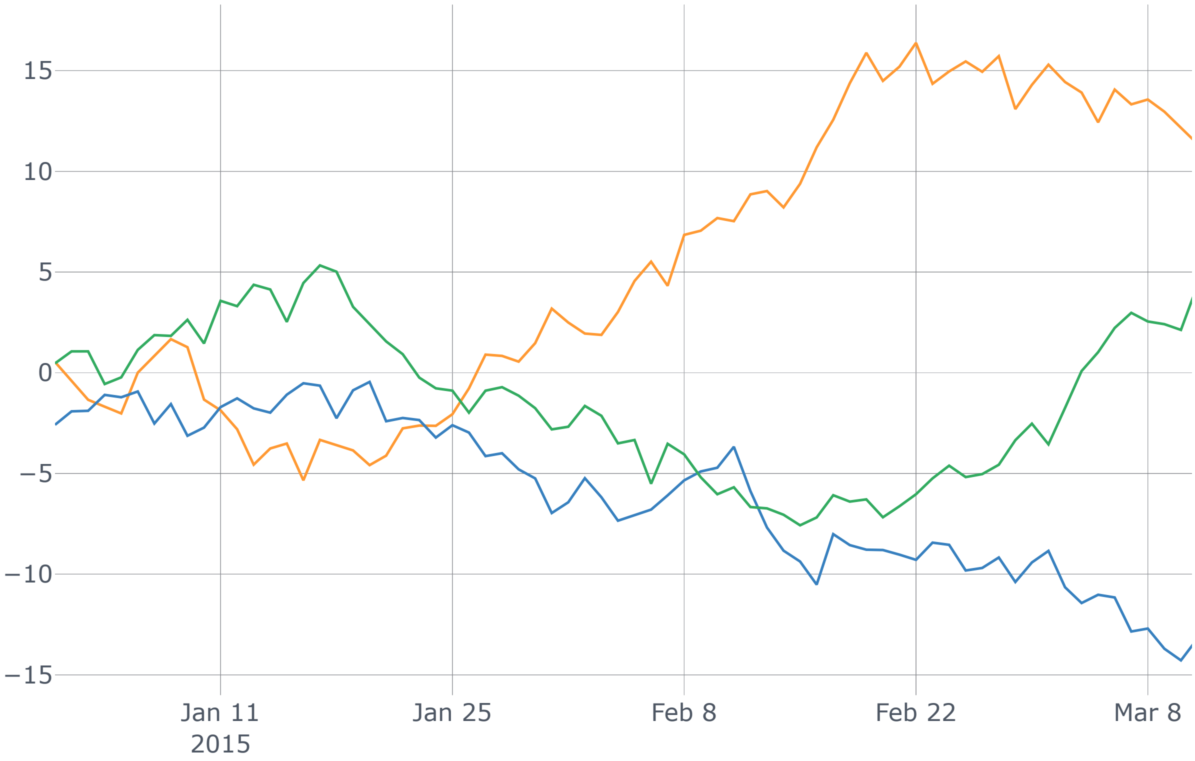
Here is an example of a simple plotly figure. You can find more examples in our online python documentation.

```
            Bar(**{
                'x': [1, 2, 3],
                'y': [3, 1, 5],
                'name': 'first trace',
                'type': 'bar'
            }),
            Bar(**{
                'x': [1, 2, 3],
                'y': [4, 3, 6],
                'name': 'second trace',
                'type': 'bar'
            })
        ],
        'layout': Layout(**{
            'title': 'simple example'
        })
    }, filename='cufflinks/simple-plotly-example')
```



EDIT CHART

`cufflinks` generates these figure's that describe plotly graphs. For example, this graph:



EDIT CHART

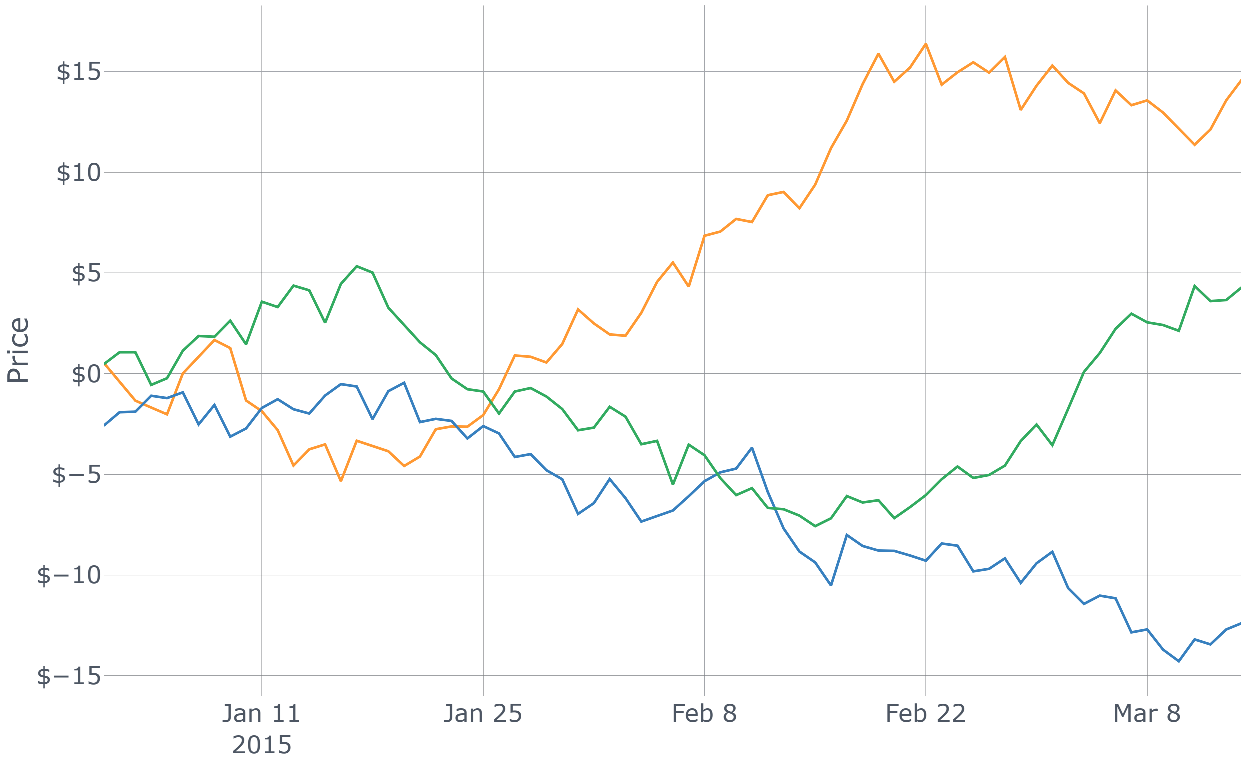
has this description:

```
Figure(
    data=Data([
        Scatter(
            x=['2015-01-01', '2015-01-02', '2015-01-03', '2015-01-04', '..'
],
            y=array([  5.35393544e-01,  -3.51020567e-01,  -1.34207933e+00,
 ..,
            mode='lines',
            name='a',
            line=Line(
                color='rgba(255, 153, 51, 1.0)',
                width='1.3'
            )
```

So, if you want to edit any attribute of a Plotly graph from cufflinks, first convert it to a figure and then edit the figure objects. Let's add a yaxis title, tick suffixes, and new legend names to this example:

```
py.iplot(figure, filename='cufflinks/customized-chart')
```

Out[50]:



EDIT CHART

See more examples of Plotly graphs or view the entire reference of valid attributes

Cufflinks Reference

Cufflinks is open source on github!

```
Help on method _iplot in module cufflinks.plotlytools:

_iplot(self, data=None, layout=None, filename='', world_readable=None, kind
='scatter', title='', xTitle='', yTitle='', zTitle='', theme=None, colors=No
ne, colorscale=None, fill=False, width=None, mode='lines', symbol='dot', siz
e=12, barmode='', sortbars=False, bargap=None, bargroupgap=None, bins=None,
histnorm='', histfunc='count', orientation='v', boxpoints=False, annotations
=None, keys=False, bestfit=False, bestfit_colors=None, categories='', x='',
y='', z='', text='', gridcolor=None, zerolinecolor=None, margin=None, subplo
ts=False, shape=None, asFrame=False, asDates=False, asFigure=False, asImage=
False, dimensions=(1116, 587), asPlot=False, asUrl=False, online=None, **kwa
rgs) method of pandas.core.frame.DataFrame instance
       Returns a plotly chart either as inline chart, image of Figure object
```

Still need help?

Contact Us

community.plot.ly

support.plot.ly

github.com/plotly

For guaranteed 24 hour response turnarounds, upgrade to a Developer Support Plan.

Dash DAQ
Dash Deployment Server

Customer Contact
#plotlylife
Twitter
GitHub

Community Support
Documentation

GRAPHING LIBRARIES

Dash
Plotly.js
Plotly.py
Plotly.R

EMBEDDED BI/OEM

Chart Studio
Dashboards

JOIN THE DASH CLUB

Dash Club is a no-fluff, twice-a-month email with links and notes on the latest Dash developments and community happenings.

Your Email Address

Subscribe

Terms of Service Privacy Policy

Show Sidebar

Fork on Github

plotly

Show Sidebar

Fork on Github

plotly

Show Sidebar

Fork on Github