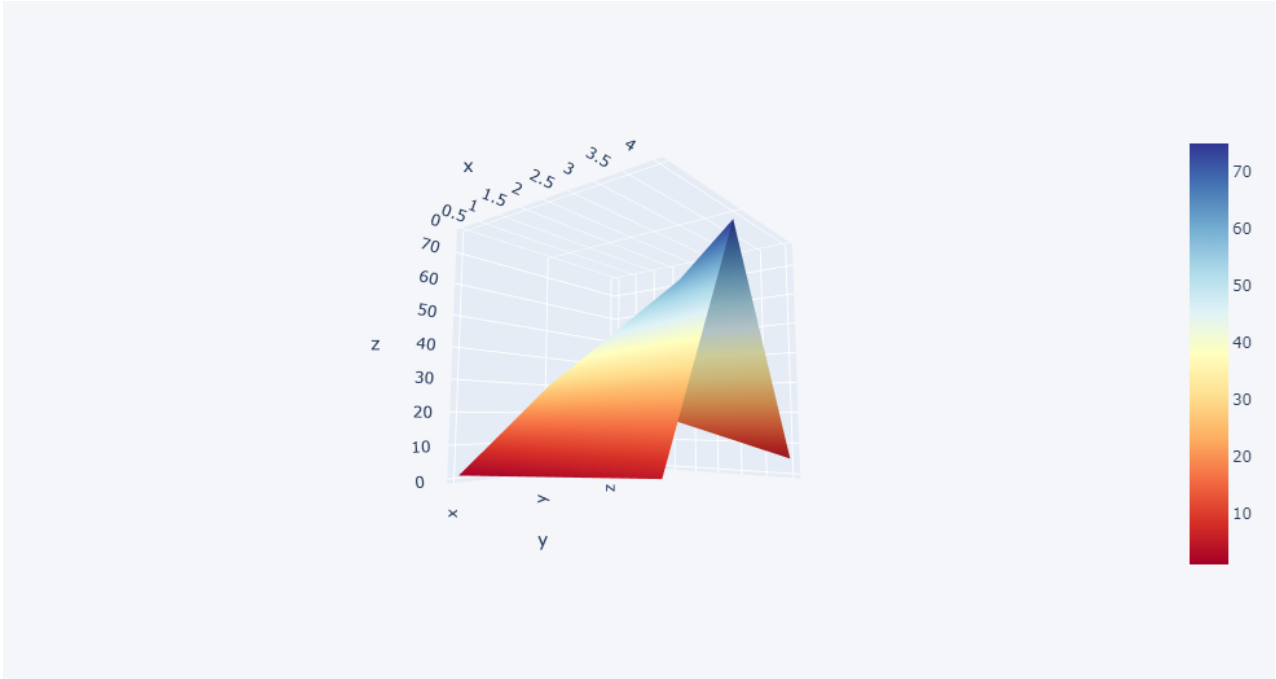
A Comprehensive Guide to Data Visualization with Python for Complete Beginners – Plotly and Cufflinks

helloml.org/a-comprehensive-guide-to-data-visualization-with-python-for-complete-beginners-plotly-and-cufflinks/

October 21, 2021

This article is a part of the 'Comprehensive guide to data visualization with Python for complete beginners' series, with the previous articles covering an introduction to data visualization and hands-on data visualization with two of the most popular data visualization tools for Python: Matplotlib and Seaborn. The links to the tutorials can be found here and here respectively for Matplotlib and Seaborn. We also covered visualization with Pandas built-in data visualization, which can come in handy if you want to graph plots directly off Pandas data frames in no time. That article can be found over here.

As we had seen previously, Seaborn was a massive improvement over Matplotlib in terms of ease of writing code (shorter syntax) and had stunning default themes as compared to using Matplotlib, which in turn was essentially like being given a clean slate and choosing to draw whatever we want however we want. We also saw that Matplotlib was the most fundamental data visualization library in Python; Seaborn just provided a high-level interface over Matplotlib. This time, we will have a look at Plotly, which takes the level of visualizations further up a notch.



A snapshot of an interactive 3-D surface plot graphed using Plotly

Brief Introduction to Plotly and Cufflinks

Plotly is an open-source online (browser-based) graphing library that allows us to create interactive plots that we can use in dashboards or websites (We can save them as HTML files or static images). Plotly provides support for many programming languages including

Python. Plotly has beautiful default themes and has options for interacting with our plots. It also allows us to create advanced plots such as 3-D graphs with just a few lines of code.

Cufflinks is a library that helps us to use Plotly with our data. Cufflinks binds the powerful features of Plotly with the flexibility of Pandas so that we can call plots directly off of a Pandas dataframe (like we did with Pandas built-in data visualizations) and create powerful, stunning visualizations with ease.

Installing Plotly and Cufflinks

Cufflinks is not directly available through **conda** but is available through **pip**. To install Plotly through conda install, type in the following command in the anaconda prompt or your regular terminal.

```
conda install -c plotly
```

To install both Plotly and Cufflinks directly through pip, enter the following commands in the command line.

```
pip install plotly cufflinks
```

Next, we require some other basic libraries for our tasks. These can again be installed through anaconda as shown below.

```
conda install numpy pandas matplotlib
```

Or with pip as shown below.

```
pip install numpy pandas matplotlib
```

Next up is importing the required libraries and setting up the commands so that we can start plotting our graphs straight away.

```
import pandas as pd
import numpy as np
%matplotlib inline
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

print(__version__)

4.14.3
```

Note that in the last line, we are printing the version of Plotly which we have downloaded. This is just to check whether we are having the correct version of the library for our use. For our purposes, we need our Plotly version to be 1.9.0 or higher. In my case, I had version 4.14.3 installed. So, we are good to go and set up cufflinks for offline usage of Plotly from right within our Jupyter notebook, which we will do so in the following few lines of code.

```
import cufflinks as cf
init_notebook_mode(connected=True)
cf.go_offline()
```

Alright! This one took a bit of setup, but we are now ready to graph some plots! But, before that, we need data. Instead of using a library built-in dataset or external dataset, we will create our own fake data:

```
df = pd.DataFrame(np.random.randn(200,4),columns='w x y z'.split())
df.head()
```

```
          w         x         y         z
0  0.922164  0.850043  1.207454 -0.642254
1  0.006018 -0.127617  1.344215 -0.754415
2  0.408488 -0.911879 -0.705910 -1.559610
3 -0.941959 -2.053259  0.190097  0.903762
4 -1.010822 -1.793520 -0.635108  0.905367
```

We will also create another dataframe, to plot some categorical plots:

```
df2 = pd.DataFrame(data=[17, 41, 58, 73, 97 ], index='A B C D E'.split(), columns=
['Data'])
df2.head()
```

```
   Data
A    17
B    41
C    58
D    73
E    97
```

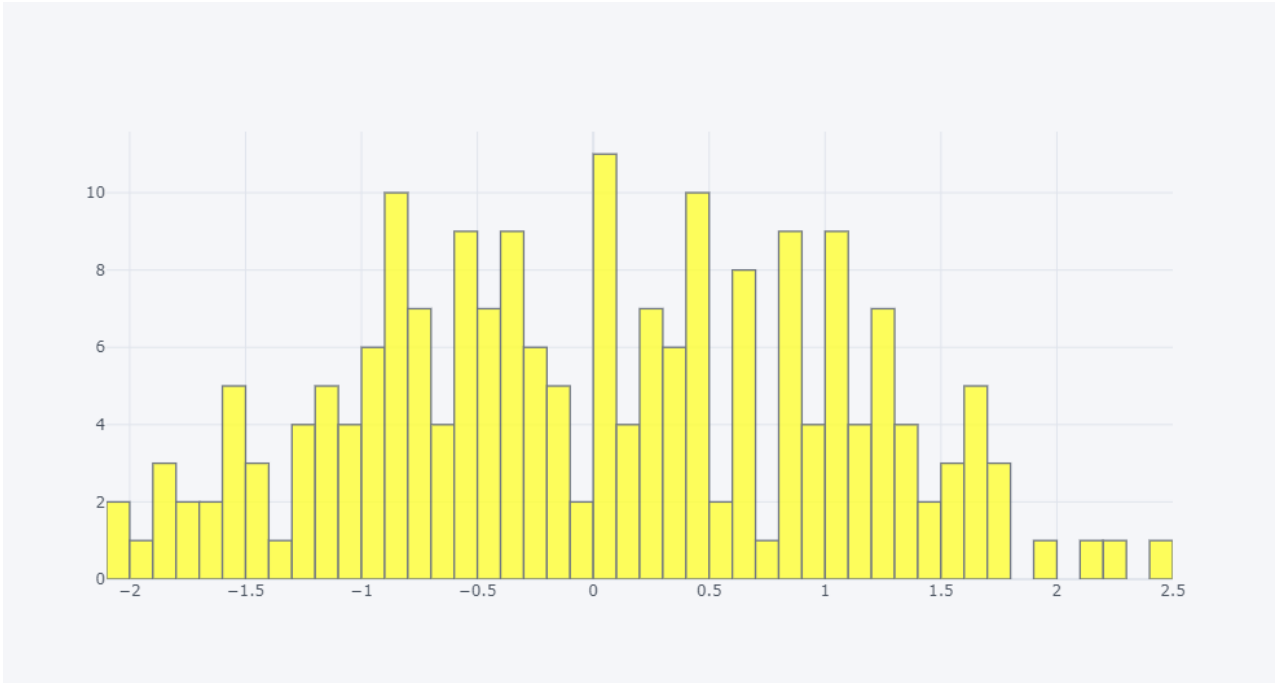
Now, once we have generated fake data, we can start by calling iplot() directly off the data frames, similar to what we did while visualizing data with Pandas' built-in visualization with **plot()**. This is possible due to Cufflinks, which binds Pandas and Plotly together.

We will go through the below-listed types of plots (most of which we have already seen). If you would like to read more about them, please check out the previous articles linked at the top of the article.

1. Histogram
2. Scatter Plot
3. Bar Plot
4. Box Plot
5. Spread Plot
6. Surface Plot
7. Bubble Plot

Histogram

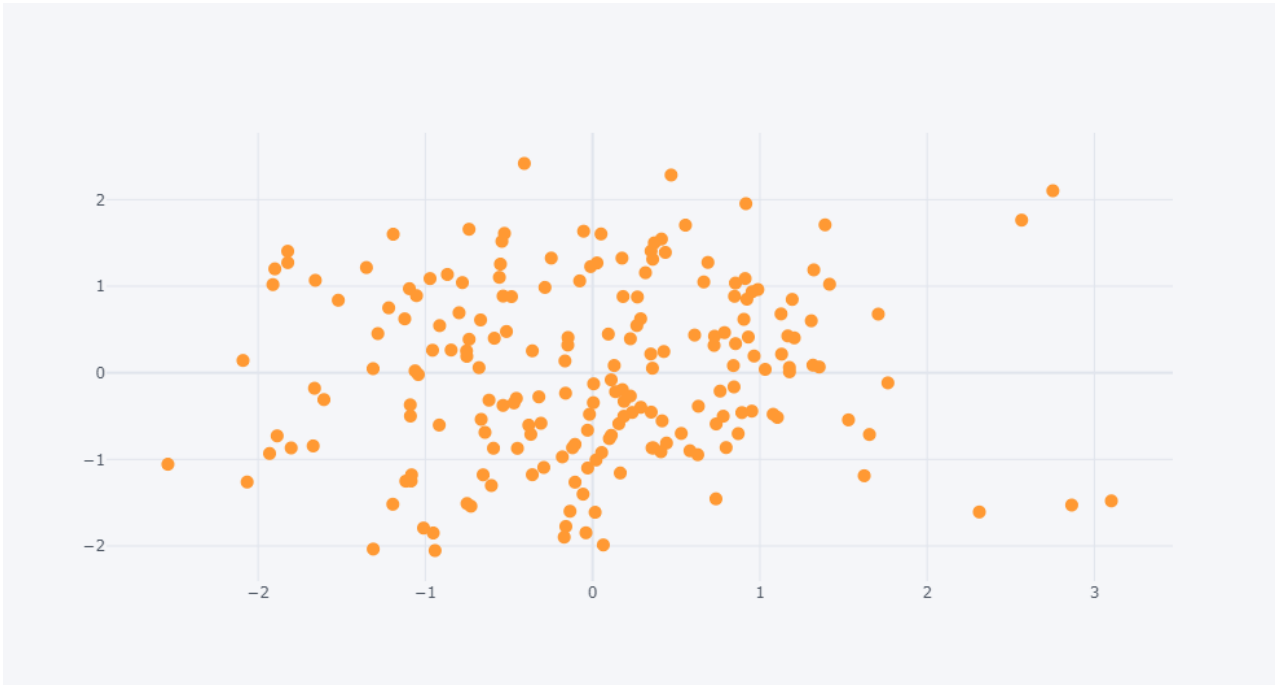
```
df['x'].iplot(kind='hist',bins=50, color='yellow')
```



The output of the above code.

Scatter Plot

```
df.iplot(kind='scatter',x='w',y='x',mode='markers',size=10)
```



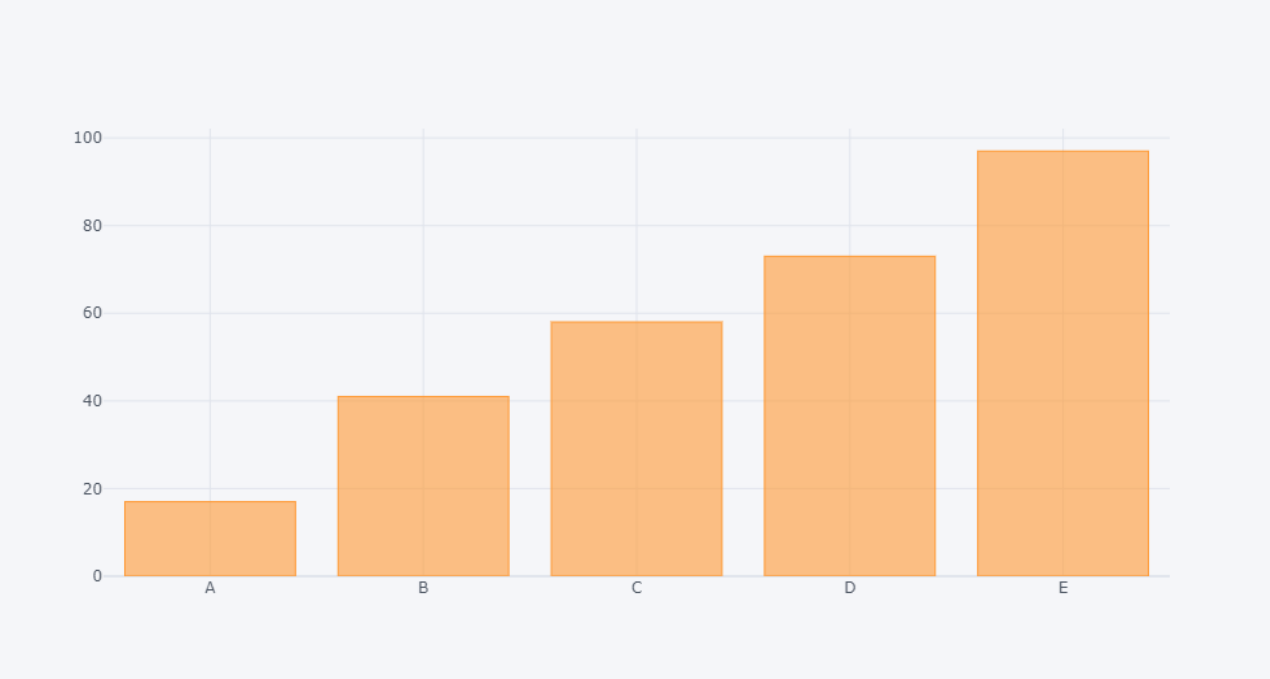
Output

Bar Plot

A bar plot is an example of a categorical plot. Categorical plots are the ones in which we plot a categorical column vs a numerical column or another categorical column.

A bar plot presents categorical data with rectangular bars with lengths proportional to the values that they represent. It is useful for showing comparisons.

```
df2.iplot(kind='bar', y='Data')
```

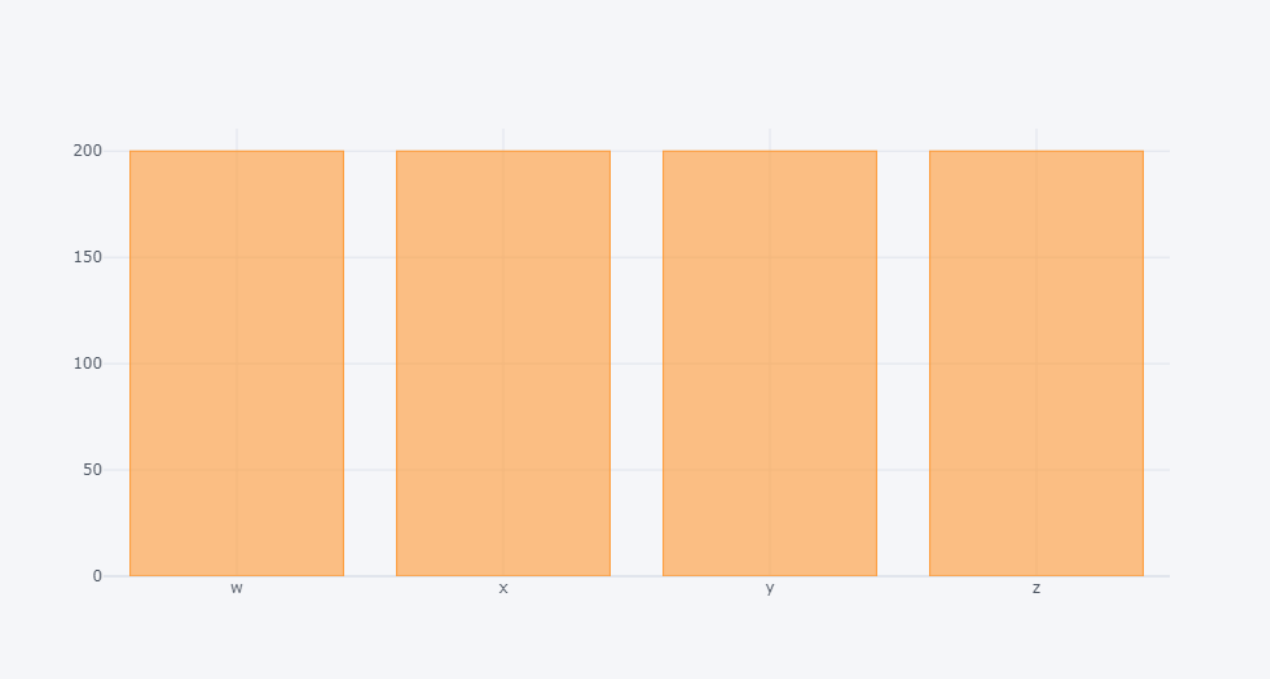


Output

Count Plot

A count plot is essentially the same as the previously discussed bar plot, except that here the y-axis variable is fixed and represents the count values for the column entered.

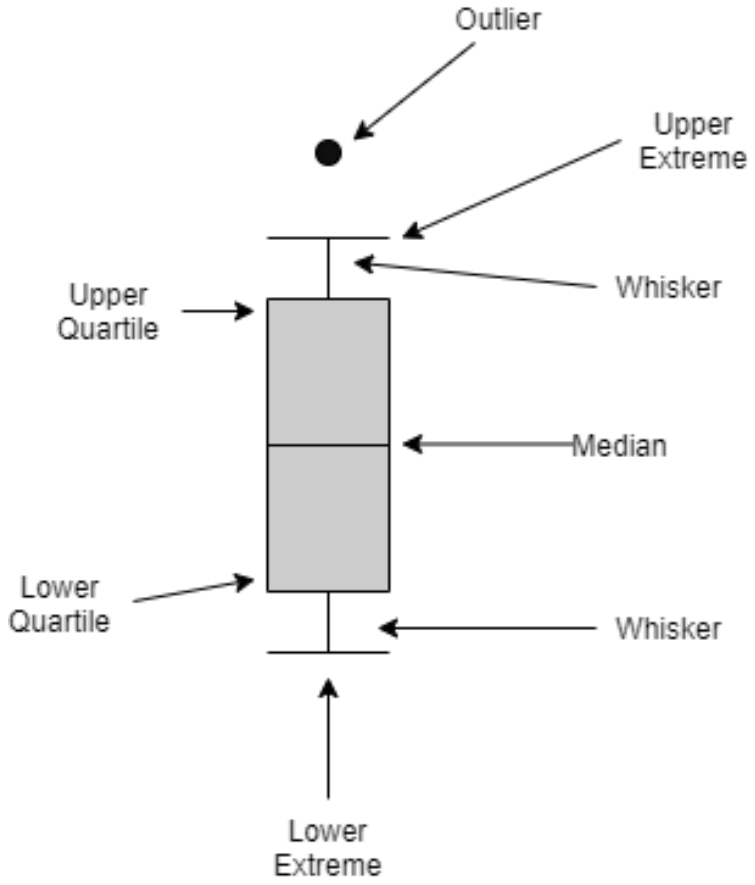
```
df.count().iplot(kind='bar')
```



Output

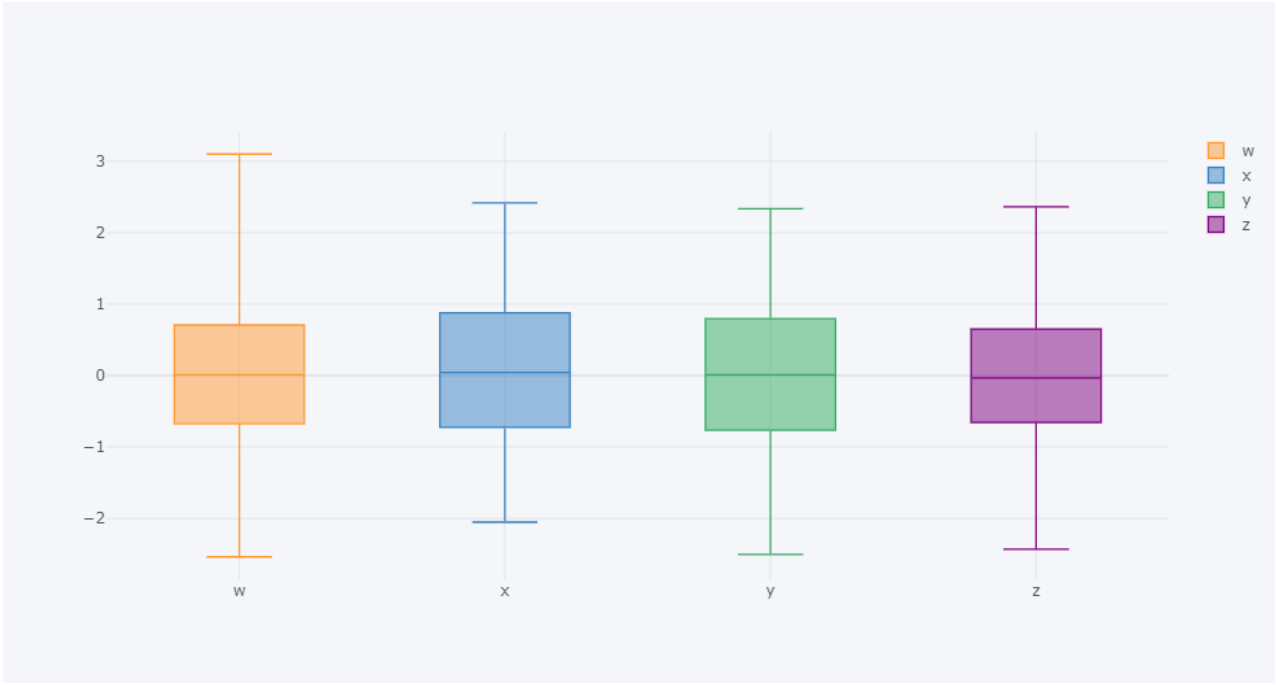
Box Plot

A box plot displays the five-number summary of a set of data. The five-number summary is the minimum, first quartile, median, third quartile, and maximum. The minimum and maximum points are represented by the dash in the horizontal axis at the edge of each plot. The lines extending parallel (inline) from the boxes which reach up to the extremes (upper and lower) are known as the “whiskers”, which indicate variability outside the upper and lower quartiles. For some plots, dots follow the maximum dash. Those points are the outliers. The outliers are in the same line as the whiskers. For more clarity, refer to the below figure:



Explanation of a Bar plot

```
df.iplot(kind='box')
```

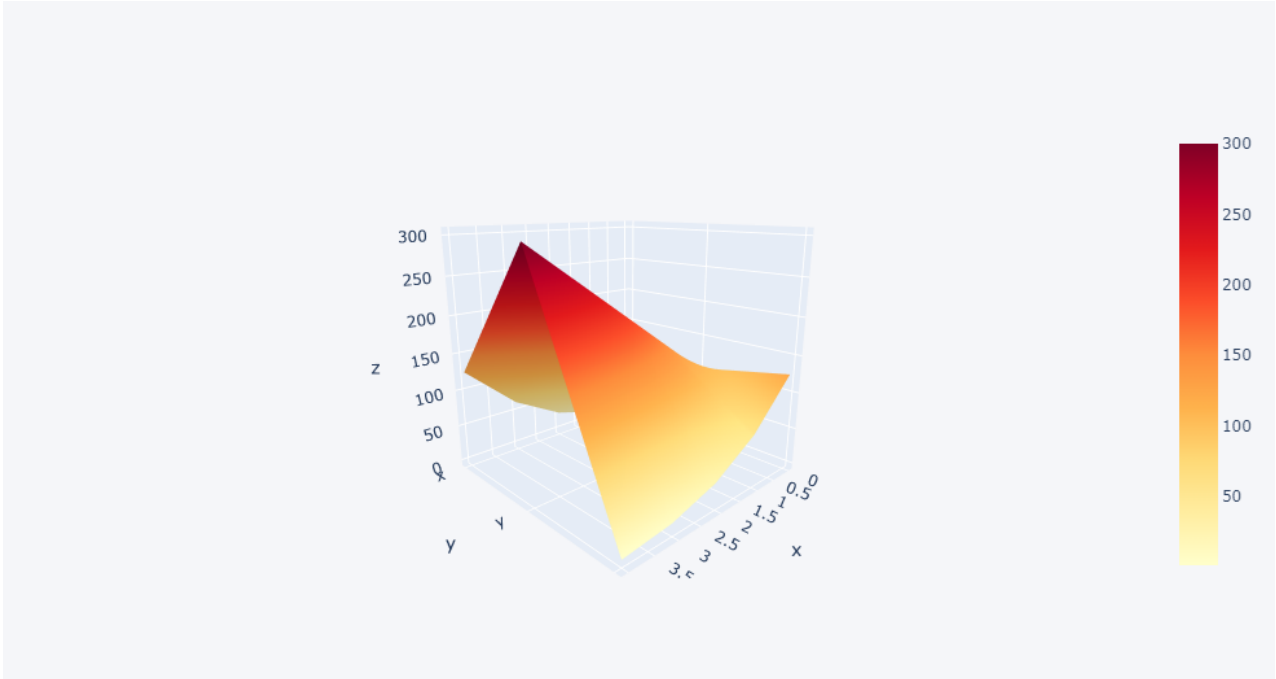


Output

3-D Surface Plot

We have not seen this kind of plot yet. However, as the name suggests, it graphs out in 3 directions, so we will need to provide values for our contour in the x, y, and z directions.

```
df3 = pd.DataFrame({'x':[1, 8, 27, 64, 125],'y':[100, 150, 200, 250, 300] ,'z':
[125, 64, 27, 8, 1]})
df3.iplot(kind='surface',colorscale='ylorrd')
```

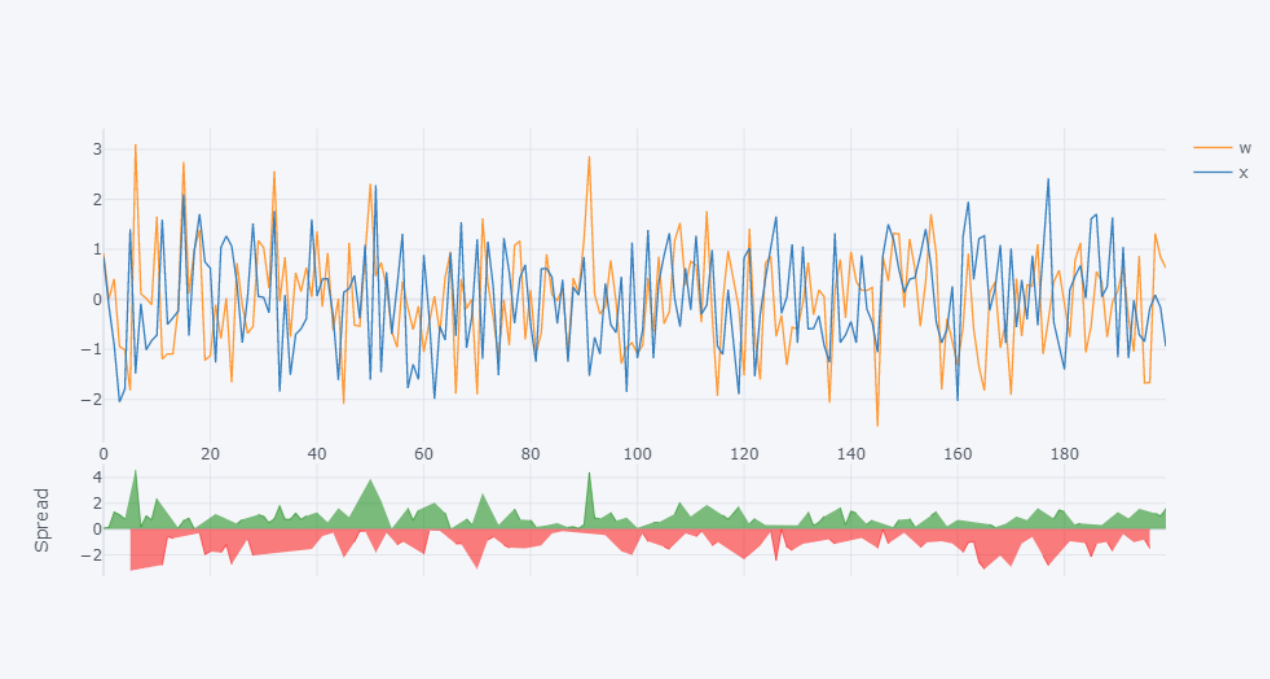


Output

Spread Plot

Spread plots are a useful way to show the spread of data across groups.

```
df[['w','x']].iplot(kind='spread')
```

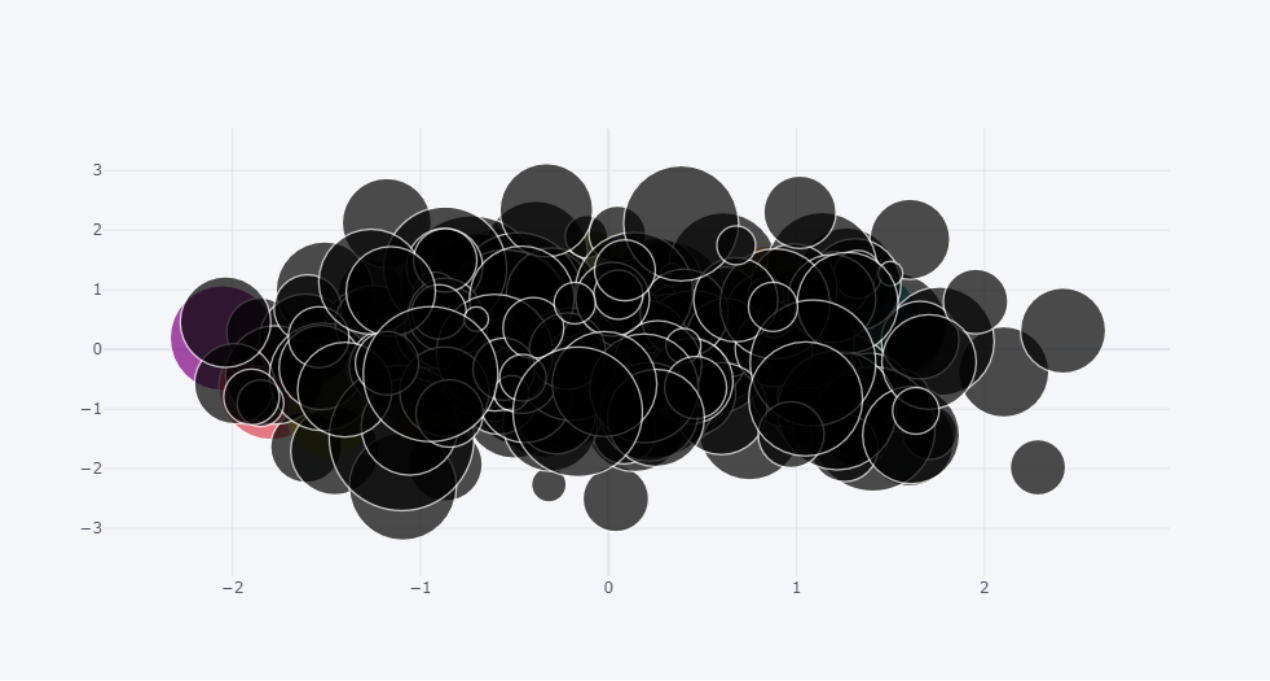


Output

Bubble plot

A bubble plot is the same as a scatter plot, with just an added dimension; the value of the added variable is represented through the size of the scatter points (Thus forming the so-called bubbles). Bubble plots help us to look at relationships between three numeric variables.

```
df.iplot(kind='bubble',x='x',y='y',size='z')
```

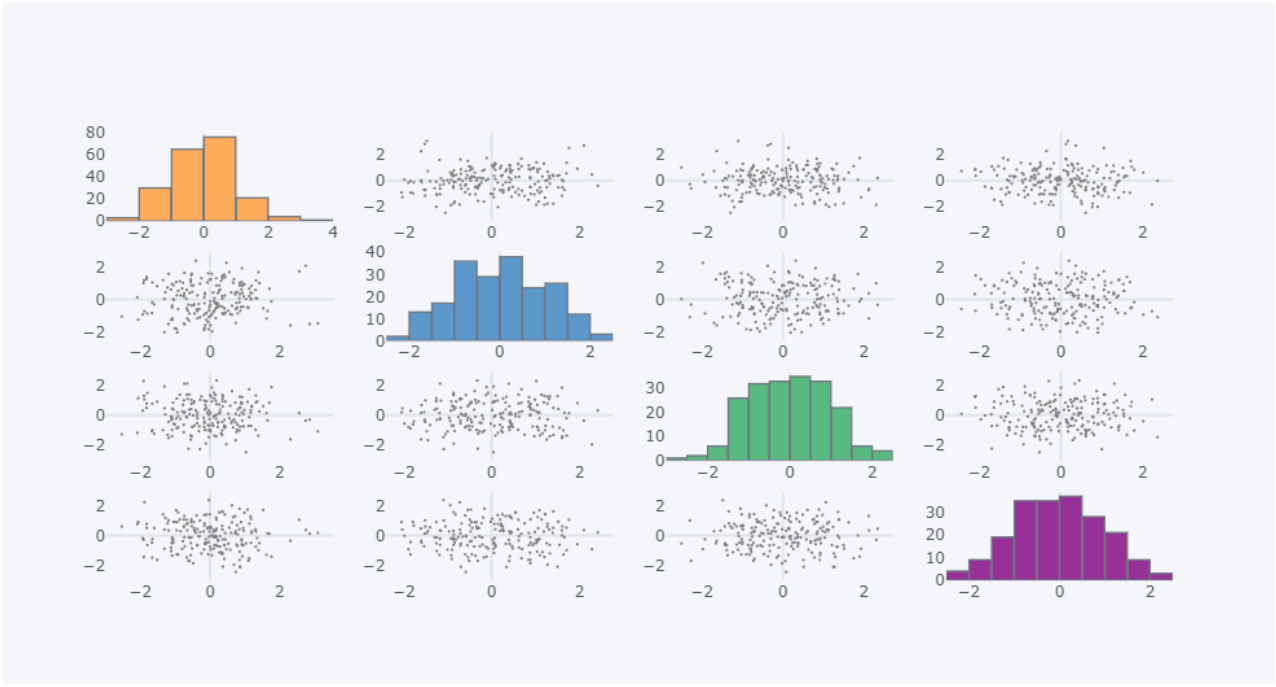


Output

Scatter Matrix

A scatter matrix is the same as Seaborn's pairplot; As we had seen in the article involving Seaborn, a pairplot allows us to plot pairwise relationships among all the columns present in the dataset.

```
df.scatter_matrix()
```



Output

That's it for this article! If you've come this far, you know how to use Plotly locally, calling iplot straight off Pandas' dataframes. Plotly provides us with an interactive window; So pan, zoom, rotate, and play around with your plots!