Foundations of Data Science – Data Visualization: Making Interactive Charts

Before we Begin…

Before we begin this lesson, you will have to install the bokeh module. This can be accomplished by opening a command prompt and typing either:

```
conda install bokeh
```

Or

```
pip install bokeh
```

Introducing Bokeh… Like Matplotlib but Interactive

This lesson is going to introduce you Bokeh which is a data visualization library similar to the ones we already learn except that the goal is to produce interactive web based visualizations as opposed to static charts and graphs. What is neat about Bokeh is that it can produce HTML files or can embed output into an iPython notebook.

From their website:

> Bokeh is a Python interactive visualization library that targets modern web browsers for presentation providing elegant, concise construction of novel graphics with high-performance interactivity over very large or streaming datasets in quick and easy way. Offering both powerful and flexible features to enable very advanced customizations in one hand and simplicity on the other Bokeh exposes different interface levels to the users:
>
> - a low Level (and more flexible) glyph interface
> - an intermediate level interface called plotting
> - a high level interface that can be used to build complex plot in a simple way.

Bokeh can output an html page containing all the JavaScript to make it interactive, or can output directly to an iPython notebook. If you want your charts to appear directly in an iPython notebook, you'll need to include the following code in your notebook before you make any Bokeh calls.

```
from bokeh.plotting import output_notebook
output_notebook()
```

So let's dive into Bokeh.

Getting Started:

Bokeh allows you to do low level "drawing" but also has higher level wrappers which will enable to you draw standard charts with a simple method call. At the time of writing, Bokeh supports the following chart types:

- Area (Overlapped and Stacked)
- Bar (Grouped and Stacked)
- BoxPlot
- Donut (Ick!)
- Dot
- HeatMap
- Histogram
- Line
- Scatter
- Step
- Timeseries

To create one of these charts, you first need to import the appropriate chart from `bokeh.charts`.

For instance:

```
from bokeh.charts import Bar
```

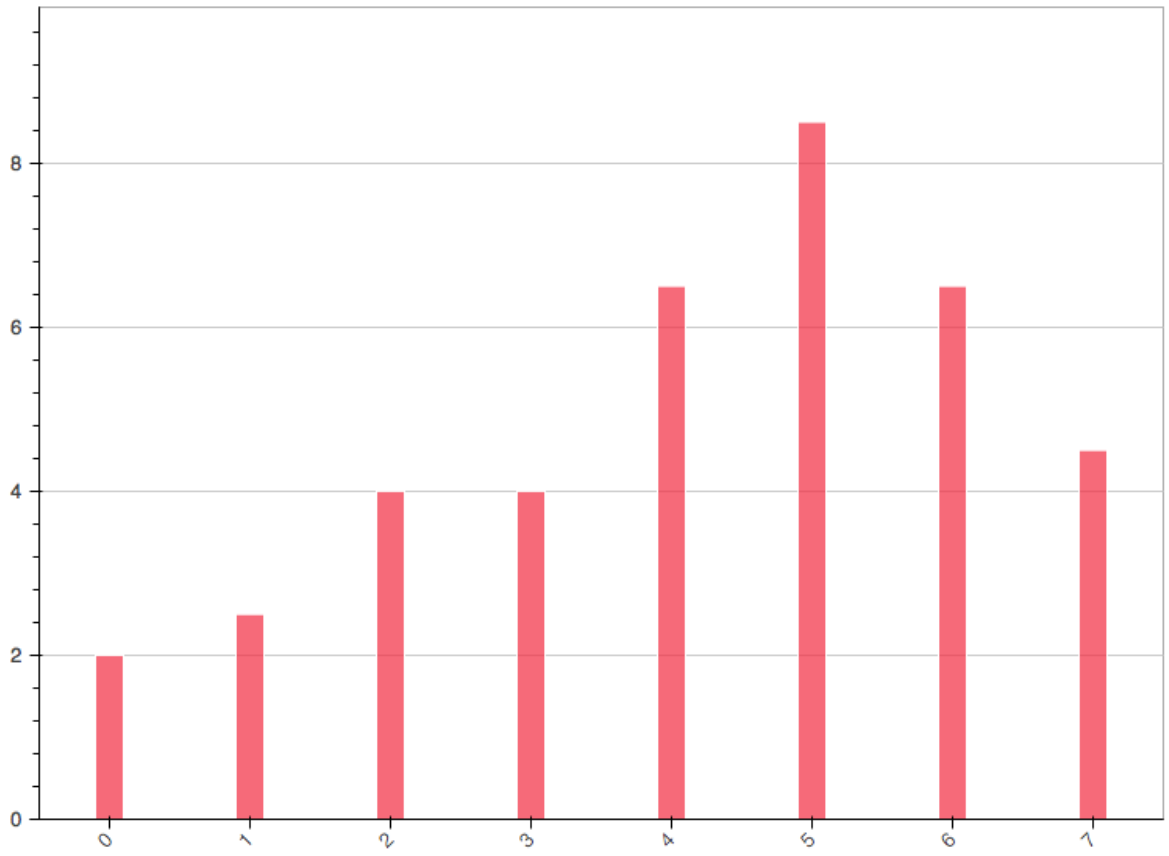
Once you've imported the correct chart interface, the next step is to create the chart by calling the method you just imported, then you configure the chart according to your specifications, and finally call the .show() method to actually display the chart to the screen. Let's look at a basic Bar chart.

All the chart arguments require a data source. You can use a simple list, dictionary, numpy array, or pandas DataFrame as a data source for the chart. For any of these charts, the data source is the first argument in the function call and in most cases, is the only required argument.

```
from bokeh.charts import Bar
data = [2,3,4,4,7,9,7,5]

barchart = Bar( data, notebook=True )
barchart.show()
```

This gets you a basic bar chart.



If you execute this in the web browser, you will also notice a toolbar at the top of the screen which allows you to manipulate this chart. Since the entire chart fits on the screen, the only option in the chart is to save it, but you will notice in other charts, that different options will appear in the tool bar.

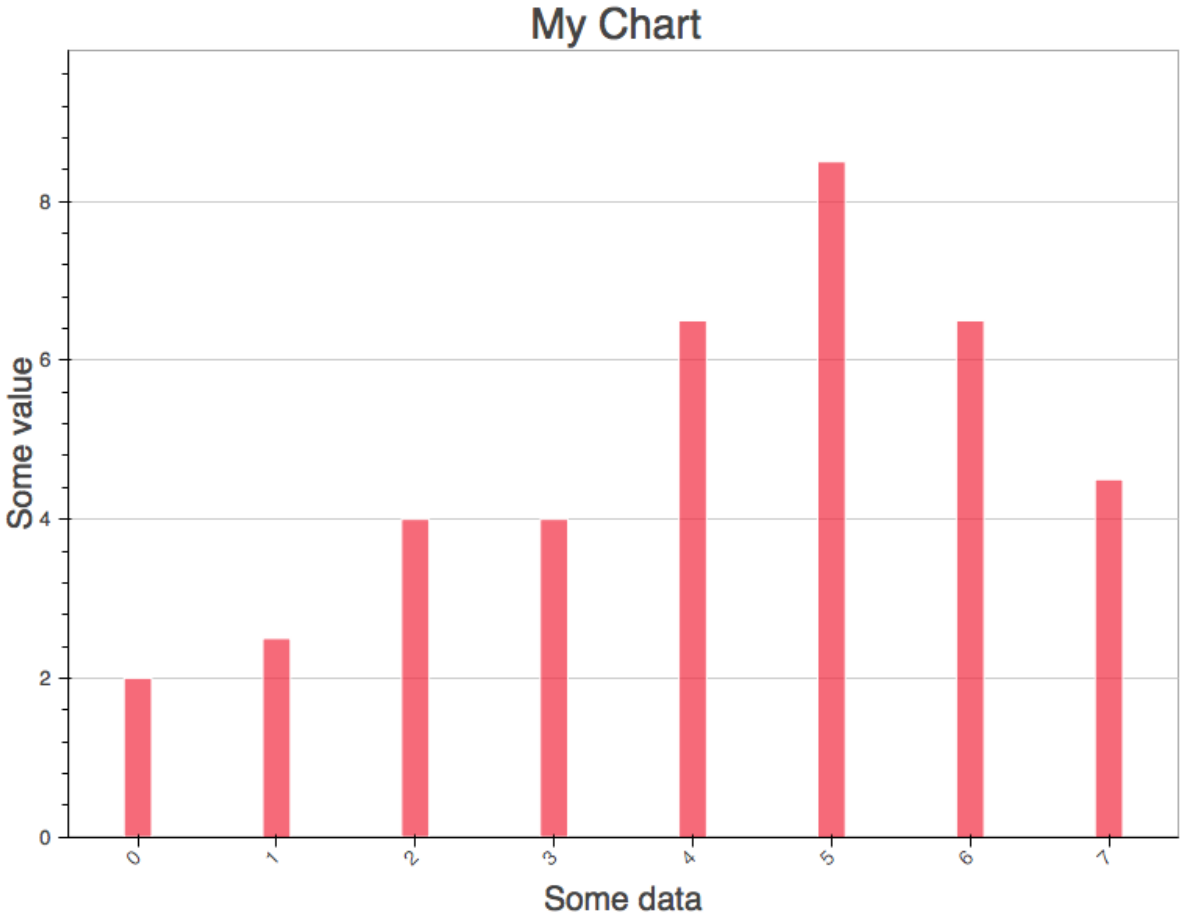
Chaining Methods

Bokeh allows you to specify additional configuration options using the following methods:

- title (str): the title of your plot.
- xlabel (str): the x-axis label of your plot.
- ylabel (str): the y-axis label of your plot.
- legend (str, bool): the legend of your plot.
- xscale (str): the x-axis type scale of your plot.
- yscale (str): the y-axis type scale of your plot.
- width (int): the width of your plot in pixels.
- height (int): the height of you plot in pixels.
- tools (bool): to enable or disable the tools in your plot.
- filename (str or bool): the name of the file where your plot will be written.
- server (str or bool): the name of your plot in the server.
- notebook (bool):if you want to output (or not) your plot into the IPython notebook.

These options are available for every type of chart. All these fields can be specified in the constructor or they can be chained. Let's look at both examples:

```
barchart2 = Bar( data, title="My Chart", xlabel="Some data", ylabel="Some
value", notebook=True )
barchart2.show()
```



Alternatively, you can chain these methods after the constructor:

```
barchart3 = Bar(data).title( "My Chained Chart" ).notebook( True ).legend(
True )
barchart3.show()
```

The only method which cannot be chained is the .show() method.

Chart Specific Options

Argument	Area	Bar	BoxPlot	HeatMap	Donut	Dot	Histogram	Line	Scatter	Step	TimeSeries
values	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	*Yes*	Yes	Yes
index	Yes	No	No	No	No	No	No	Yes	No	Yes	Yes
cat	No	Yes	No	Yes	No	Yes	No	No	No	No	No
facet	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
stacked	Yes	Yes	No	No	No	No	No	No	No	No	No
pallette	No	No	No	Yes	No	No	No	No	No	No	No
bins	No	No	No	No	No	No	Yes	No	No	No	No
mu	No	No	No	No	No	No	Yes	No	No	No	No
sigma	No	No	No	No	No	No	Yes	No	No	No	No