2 - Linked Plots

December 15, 2015

1 Linked Plots

In Bokeh, it's easy to show multiple plots and link them, right from Python. Additionally, we can easily customize the kind of linkage straight from Python, without needing to fiddle around with arcane Javascript callback functions.

```
In [1]: from bokeh.plotting import figure
        from bokeh.io import output_notebook, show, gridplot
        output_notebook()
```

1.1 First, we'll load some data to plot, and compute some derived statistics.

```
In [4]: from numpy import asarray
        from bokeh.sampledata.autompg import autompg

        grouped = autompg.groupby("yr")
        mpg = grouped["mpg"]
        avg = mpg.mean()
        std = mpg.std()
        years = asarray(grouped.groups.keys())
```

```
In [5]: # Next, we'll look at the differences between American and Japanese cars.
        american = autompg[autompg["origin"]==1]
        japanese = autompg[autompg["origin"]==3]

        p = figure(title="Model Year vs MPG (USA/Jap)")
        p.quad(left=years-0.4, right=years+0.4, bottom=avg-std, top=avg+std,
             fill_alpha=0.4)
        p.circle(x=asarray(japanese["yr"]), y=asarray(japanese["mpg"]),
               size=8,
               alpha=0.4, line_color="red", fill_color=None, line_width=2)
        p.triangle(x=asarray(american["yr"]), y=asarray(american["mpg"]),
                 size=8, alpha=0.4, line_color="blue", fill_color=None,
                 line_width=2)
        show(p)
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-5-14bf37164867> in <module>()
  4
  5 p = figure(title="Model Year vs MPG (USA/Jap)")
```

```
----> 6 p.quad(left=years-0.4, right=years+0.4, bottom=avg-std, top=avg+std,
      7      fill_alpha=0.4)
      8 p.circle(x=asarray(japanese["yr"]), y=asarray(japanese["mpg"]),


TypeError: 'float' object is not iterable
```

1.2 Linked Scroll & Zoom

Next we'll create two plots that move together as we zoom in and out.

First we'll create the two plots.

```
In [6]: s1 = figure(title="MPG by Year", width=300, height=300)
        s1.circle(autompg["yr"], autompg["mpg"], size=10, color="navy", alpha=0.15)

        s2 = figure(title="HP by Year", width=300, height=300)
        s2.circle(autompg["yr"], autompg["hp"], size=10, color="red", alpha=0.15)

        p = gridplot([[s1, s2]])
        show(p)
```

To link them, we need to just set them to use the same "ranges":

```
In [8]: # Set s2 from the previous code block to share a range with s1:
        s3 = figure(title="MPG by Year", width=300, height=300)
        s3.circle(autompg["yr"], autompg["mpg"], size=10, color="navy", alpha=0.3)

        s4 = figure(title="HP by Year", width=300, height=300)
        s4.x_range = s3.x_range
        s4.circle(autompg["yr"], autompg["hp"], size=10, color="red", alpha=0.3)
        p = gridplot([[s3, s4]])
        show(p)
```

We can mix and match axes and ranges however we like. We can add a third plot that shares a vertical axes with plot `s3`:

```
In [9]: s5 = figure(title="MPG by HP", width=300, height=300)
        s5.triangle(autompg["hp"], autompg["mpg"], size=10, color="darkgreen", alpha=0.3)
        s5.x_range = s4.y_range
        s5.y_range = s3.y_range
        p = gridplot([[s3, s4, s5]])
        show(p)
```

1.3 Linking Data and Selection

The previous plots showed how we can link axes in very flexible ways, but Bokeh can go one step further. We can link data and selections in plots very easily, but using an object called a `DataSource`. In this example, we will use a `ColumnDataSource`, which you can think of as a simple data table or data frame.

```
In [10]: # We'll create the datasource from the autompg dataset:
         from bokeh.models import ColumnDataSource
         source = ColumnDataSource(autompg.to_dict("list"))
         source.add(autompg["yr"], name="yr")
```

```
Out[10]: 'yr'
```

In [11]:
```
# Now we can create some plots using the datasource.
# Note that rather than passing in arrays to the plotting functions,
# we're passing in names of columns to look up.

plot_config = dict(plot_width=300, plot_height=300,
                   tools="pan,wheel_zoom,reset,lasso_select,box_select")
c1 = figure(title="MPG by Year", **plot_config)
c1.circle("yr", "mpg", color="blue", source=source)

c2 = figure(title="HP vs. Displacement", **plot_config)
c2.circle("hp", "displ", color="green", source=source)

c3 = figure(title="MPG vs. Displacement", **plot_config)
c3.circle("mpg", "displ", size="cyl", line_color="red",
          fill_color=None, source=source)

show(gridplot([[c1, c2, c3]]))
```

In [ ]: