bokeh_2

March 22, 2023

[ ]:
```
from bokeh.plotting import figure
from bokeh.io import output_notebook, show
```

[ ]:
```
# tell bokeh to display plots directly into the notebook
output_notebook()
```

Scatter Plots

[ ]:
```
# create a new plot with default tools using figure
p = figure (width = 300, height = 300)

# add a circle renderer
p.circle (
    [1, 2, 3, 4, 5],
    [9, 6, 8, 7, 5],
    size = 10,
    line_color = 'navy',
    fill_color = 'orange',
    fill_alpha = 0.5
)

# show the results
show (p)
```

Line Plots

[ ]:
```
# create a new plot
p = figure (width = 300, height = 300, title = 'Line Plot')

# add a line renderer
p.line ([1, 2, 3, 4, 5],
        [5, 6, 9, 7, 8],
        line_width = 2
       )

# show the result
show (p)
```

Column Data Source

[ ]:
```
from bokeh.models import ColumnDataSource

source = ColumnDataSource (data = {
    'x' : [1, 2, 3, 4, 5],
    'y' : [3, 1, 4, 6, 5],
})
```

[ ]:
```
p = figure (width = 300, height = 300)
p.circle ('x', 'y', size = 10, source = source)
show (p)
```

[ ]:
```
import numpy as np
x = np.linspace (-6, 6, 100)
y = np.cos(x)
```

[ ]:
```
# create a plot p
p = figure (width = 500, height = 500)
p.circle (x, y, size = 7, color = 'firebrick', alpha = 0.5)
show (p)
```

Bar Plot Example

[ ]:
```
from bokeh.sampledata.autompg import autompg

grouped = autompg.groupby ('yr')

mpg = grouped.mpg
avg, std = mpg.mean(), mpg.std()
years = list (grouped.groups)
american = autompg [autompg['origin'] == 1]
japanese = autompg [autompg['origin'] == 3]
```

[ ]:
```
p = figure (title = 'MPG by Year (Japan and US)')

p.vbar (x = years, bottom = avg - std, top = avg + std,
       width = 0.8, fill_alpha = 0.2, line_color = None,
       legend_label = 'MPG 1 std dev')

p.circle (x = japanese['yr'], y = japanese['mpg'], size = 10,
          alpha = 0.5, color = 'red',
          legend_label = 'Japanese')

p.triangle (x = american['yr'], y = american['mpg'], size = 10,
            alpha = 0.3, color = 'blue',
            legend_label = 'American')

p.legend.location = 'top_left'
```

```
show (p)
```

Linked Brushing

[ ]:
```
from bokeh.models import ColumnDataSource
from bokeh.layouts import gridplot

source = ColumnDataSource (autompg)

options = dict (width = 300, height = 300,
                tools = 'pan, wheel_zoom, box_zoom, box_select, lasso_select')

p1 = figure (title = 'MPG by Year', **options)
p1.circle ('yr', 'mpg', color = 'blue', source = source)

p2 = figure (title = 'HP vs Displacement', **options)
p2.circle ('hp', 'displ', color = 'green', source = source)

p3 = figure (title = 'MPG vs Displacement', **options)
p3.circle ('mpg', 'displ', size = 'cyl',
           line_color = 'red', fill_color = None,
           source = source)

p = gridplot ([[p1, p2, p3]], toolbar_location = 'right')

show (p)
```

Linked Panning

Multiple plots have ranges that stay in sync

[ ]:
```
from bokeh.layouts import gridplot

x = list(range(11))
y0, y1, y2 = x, [10 - i for i in x], [abs(i - 5) for i in x]

plot_options = dict (width = 250, height = 250, tools = 'pan,wheel_zoom')

# create a new plot
s1 = figure (**plot_options)
s1.circle (x, y0, size = 10, color = 'navy')

# create a new plot and share both ranges
s2 = figure (x_range = s1.x_range, y_range = s1.y_range, **plot_options)
s2.triangle (x, y1, size = 10, color = 'firebrick')

# create a new plot and share only one range
s3 = figure (x_range = s1.x_range, **plot_options)
```

```
s3.square (x, y2, size = 10, color = 'olive')

p = gridplot([[s1, s2, s3]])

show(p)
```

Linked Brushing

In databases, brushing and linking is the connection of two or more views of the same data, such that a change to the representation in one view affects the representation in the other.

[ ]:
```
from bokeh.models import ColumnDataSource

x = list (range (-20, 21))
y0, y1 = [abs(xx) for xx in x], [xx**2 for xx in x]

# create a column data source for the plots to share
source = ColumnDataSource (data = dict(x=x, y0=y0, y1=y1))

TOOLS = 'box_select,lasso_select,help'

# create a new plot and add a renderer
left = figure (tools = TOOLS, width = 300, height = 300)
left.circle ('x', 'y0', source = source)

# create another new plot and add a renderer
right = figure (tools = TOOLS, width = 300, height = 300)
right.circle ('x', 'y1', source = source)

p = gridplot ([[left, right]])

show(p)
```

Widgets

[ ]:
```
from bokeh.models import Slider

slider = Slider (start = 0, end = 10, value = 1, step = 1, title = 'foo')

show (slider)
```

[ ]:
```
from bokeh.layouts import column

x = [x*0.005 for x in range (0, 201)]

source = ColumnDataSource (data = dict (x = x, y = x))

plot = figure (width = 300, height = 300)
```

```
plot.line ('x', 'y', source = source, line_width = 3, line_alpha = 0.6)

slider = Slider (start = 0.1, end = 6, value = 1, step = 0.1, title = 'power')

update_curve = CustomJS (args = dict (source = source, slider = slider),
                         code = """
     const f = cb_obj.value
     const x = source.data.x
     const y = Array.from (x, (x) => Math.pow(x, f))
     source.data = {x, y}
""")
slider.js_on_change ('value', update_curve)

show (column (slider, plot))
```

Hover Tools

[ ]:
```
from bokeh.models import HoverTool

source = ColumnDataSource (
         data = dict (
                x = [1, 2, 3, 4, 5],
                y = [2, 5, 6, 2, 7],
                desc = ['A', 'b', 'C', 'd', 'E']
         )
)

hover = HoverTool (
        tooltips = [
            ('index', '$index'),
            ('(x,y)', '($x,$y)'),
            ('desc', '@desc')
        ]
)

p = figure (width = 300, height = 300,
            tools = [hover],
            title = 'Mouse over the dots')

p.circle ('x', 'y', size = 20, source = source)

show (p)
```

CustomJS Callbacks

[ ]:
```
from bokeh.models import TapTool, CustomJS

callback = CustomJS (code = "alert ('you tapped a circle')")
```

```
tap = TapTool (callback = callback)

p = figure (width = 600, height = 300, tools = [tap])

p.circle (x = [1, 2, 3, 4, 5], y = [2, 5, 3, 9, 7], size = 10)

show (p)
```