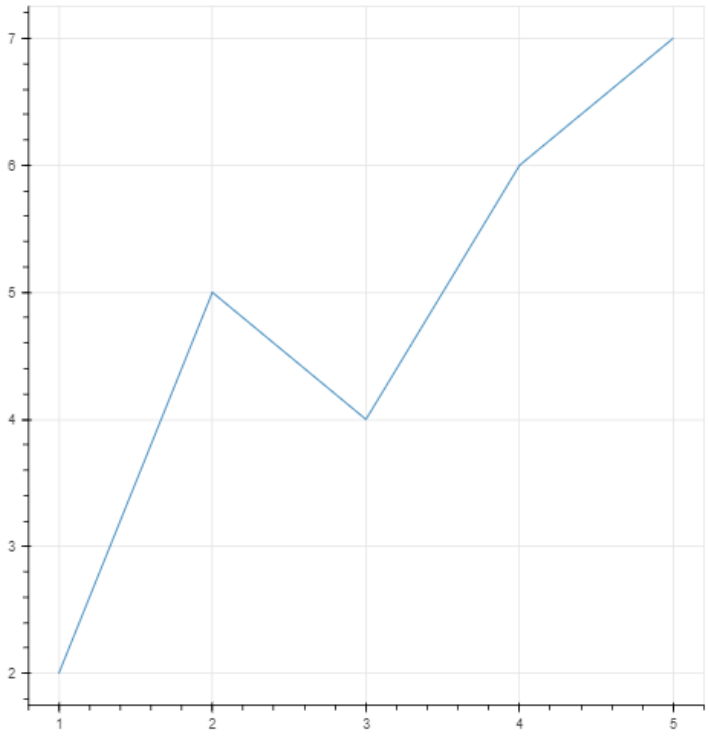
Interactive Data Visualization with bokeh

3). Layouts, Interactions, and Annotations

a). Using the current document

```
# Perform necessary imports
from bokeh.io import curdoc
from bokeh.plotting import figure

# Create a new plot: plot
plot = figure()

# Add a line to the plot
plot.line([1,2,3,4,5],[2,5,4,6,7])

# Add the plot to the current document
curdoc().add_root(plot)
```

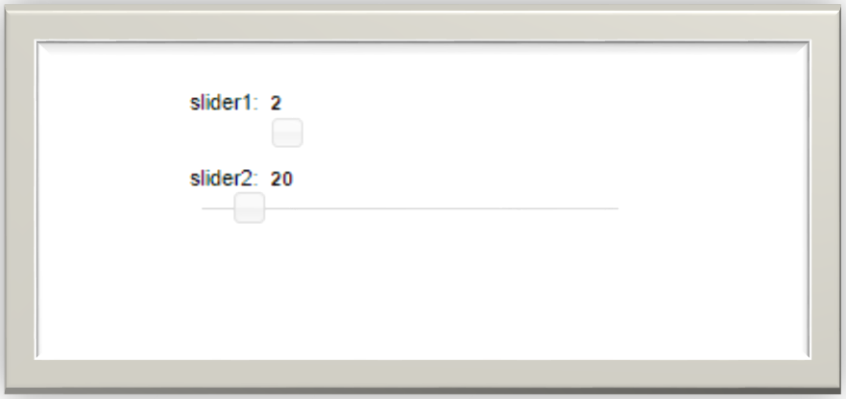


b). <u>Using the current document</u>

```
# Perform the necessary imports
from bokeh.io import curdoc
from bokeh.layouts import widgetbox
from bokeh.models import Slider

# Create a slider: slider
slider = Slider(title='my slider', start=0, end=10, step=0.1, value=2)

# Create a widgetbox layout: layout
layout = widgetbox(slider)

# Add the layout to the current document
curdoc().add_root(layout)
```

c). <u>Creating multiple Sliders in single document</u>

```
# Perform necessary imports
from bokeh.io import curdoc
from bokeh.layouts import widgetbox
from bokeh.models import Slider

# Create first slider: slider1
slider1 = Slider(title='slider1',start=0, end=10, step=0.1, value=2)

# Create second slider: slider2
slider2 = Slider(title='slider2',start=10, end=100, step=1, value=20)

# Add slider1 and slider2 to a widgetbox
layout = widgetbox(slider1, slider2)

# Add the layout to the current document
curdoc().add_root(layout)
```



d). <u>How to combine bokeh model into layouts</u>

#Create ColumnDataSource: source

source = ColumnDataSource(data={'x':x,'y':y})

 # Add a line to the plot

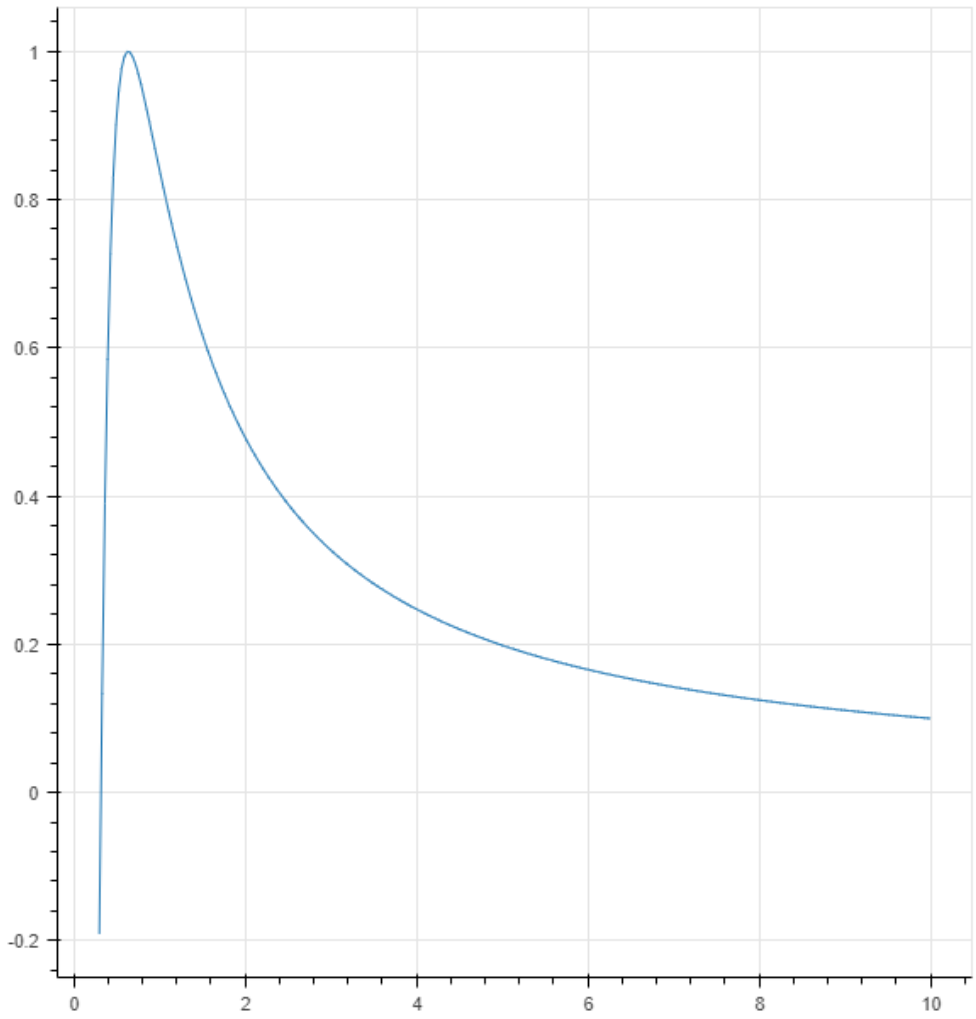
plot.line('x', 'y', source=source)

 # Create a column layout: layout

layout = column(widgetbox(slider), plot)

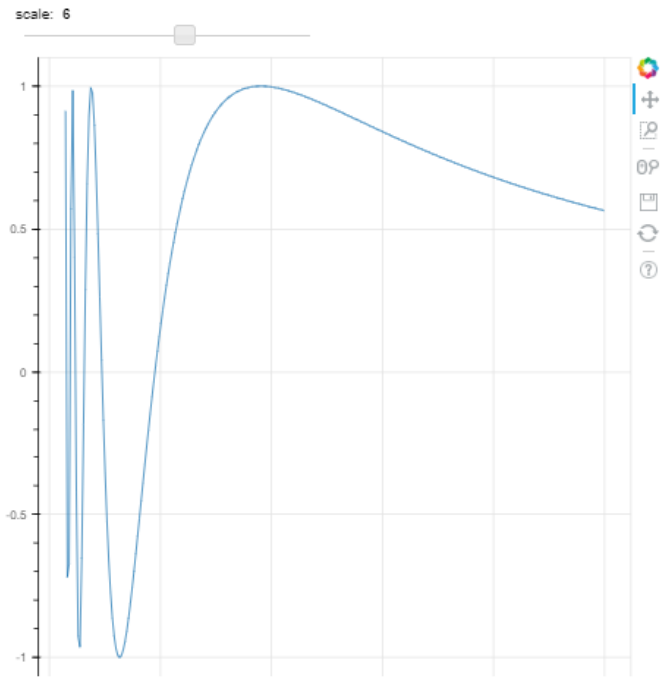
 # Add the layout to the current document

curdoc().add_root(layout)



e). <u>Learn about widget callback</u>

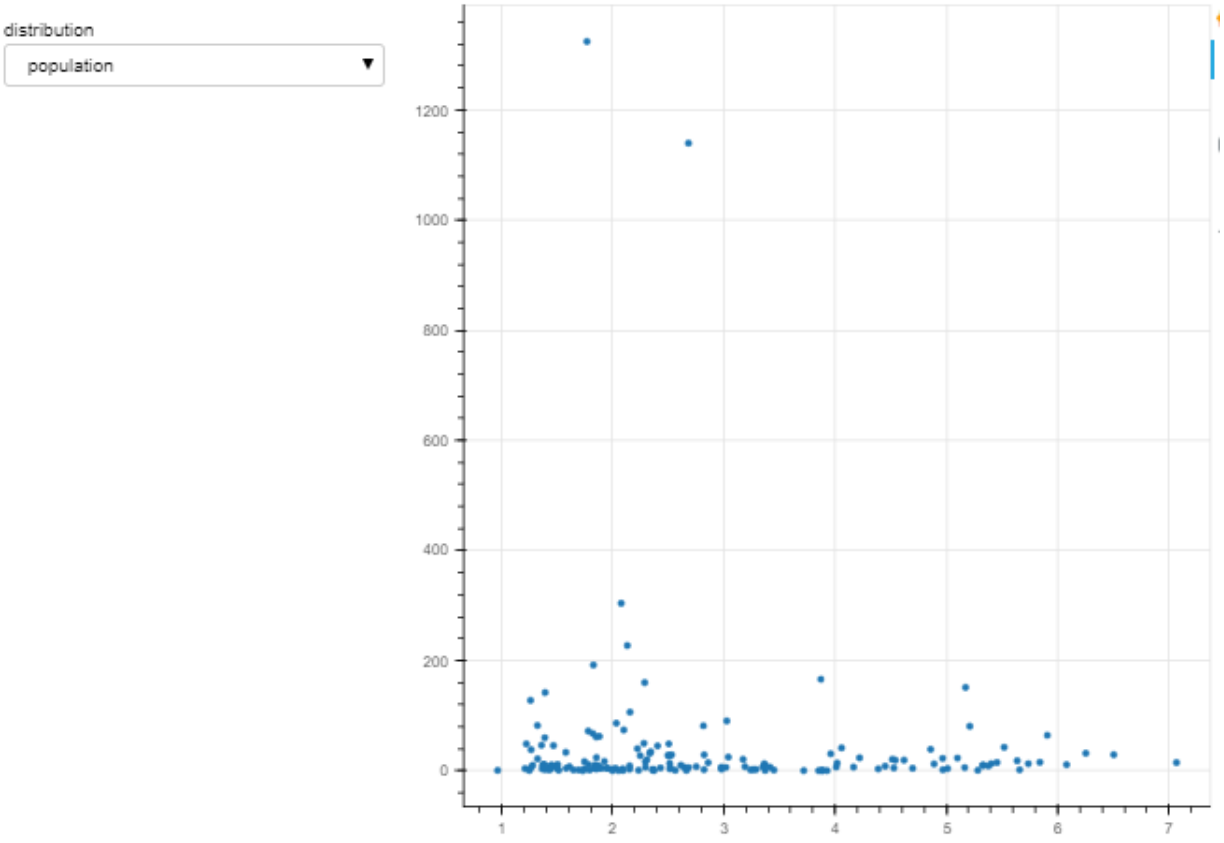
```
# Define a callback function: callback
def callback(attr, old, new):

    # Read the current value of the slider: scale
    scale = slider.value

    # Compute the updated y using np.sin(scale/x): new_y
    new_y = np.sin(scale/x)

    # Update source with the new data values
    source.data = {'x': x, 'y': new_y}

# Attach the callback to the 'value' property of slider
slider.on_change('value',callback)

# Create layout and add to current document
layout = column(widgetbox(slider), plot)
curdoc().add_root(layout)
```



f). <u>Updating data sources from dropdown callbacks</u>

```
# Perform necessary imports
from bokeh.models import ColumnDataSource, Select

# Create ColumnDataSource: source
source = ColumnDataSource(data={
    'x' : fertility,
    'y' : female_literacy
})

# Create a new plot: plot
plot = figure()

# Add circles to the plot
plot.circle('x', 'y', source=source)

# Define a callback function: update_plot
def update_plot(attr, old, new):
    # If the new Selection is 'female_literacy', update 'y' to female_literacy
    if new == 'female_literacy':
        source.data = {
            'x' : fertility,
            'y' : female_literacy
        }
    # Else, update 'y' to population
    else:
        source.data = {
            'x' : fertility,
            'y' : population
        }

# Create a dropdown Select widget: select
```

```
select = Select(title="distribution", options=['female_literacy', 'population'], value='female_literacy')

# Attach the update_plot callback to the 'value' property of select
select.on_change('value', update_plot)

# Create layout and add to current document
layout = row(select, plot)
curdoc().add_root(layout)
```

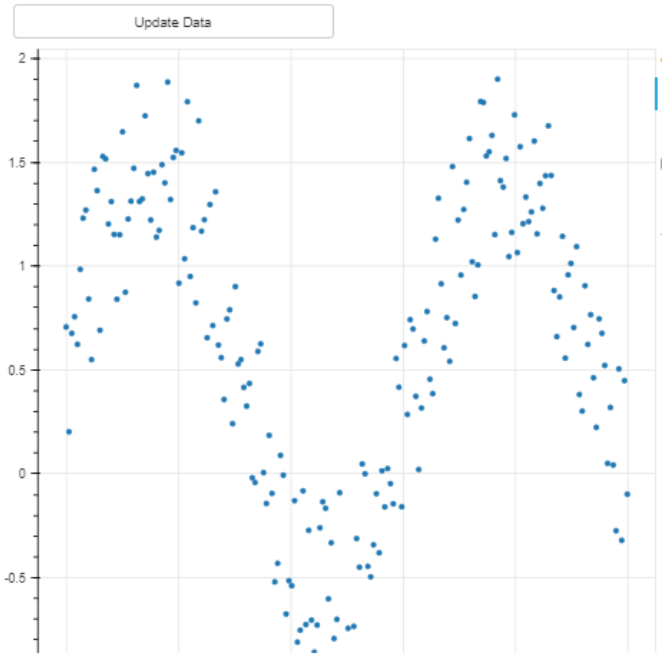


g). Synchronize two dropdowns

```
# Create two dropdown Select widgets: select1, select2
select1 = Select(title='First', options=['A', 'B'], value='A')
select2 = Select(title='Second', options=['1', '2', '3'], value='1')


# Define a callback function: callback
def callback(attr, old, new):
    # If select1 is 'A'
    if select1.value == 'A':
        # Set select2 options to ['1', '2', '3']
        select2.options = ['1','2','3']


        # Set select2 value to '1'
        select2.value = '1'
    else:
        # Set select2 options to ['100', '200', '300']
        select2.options = ['100','200','300']
        # Set select2 value to '100'
        select2.value = '100'
# Attach the callback to the 'value' property of select1
select1.on_change('value', callback)
# Create layout and add to current document
layout = widgetbox(select1, select2)
curdoc().add_root(layout)
```

h). <u>Button Widget</u>

```
# Create a Button with label 'Update Data'
button = Button(label='Update Data')

# Define an update callback with no arguments: update
def update():

    # Compute new y values: y
    y = np.sin(x) + np.random.random(N)

    # Update the ColumnDataSource data dictionary
    source.data = {'x':x, 'y':y}

# Add the update callback to the button
button.on_click(update)

# Create layout and add to current document
layout = column(widgetbox(button), plot)
curdoc().add_root(layout)
```



i). Button Styles

```
# Import CheckboxGroup, RadioGroup, Toggle from bokeh.models
from bokeh.models import CheckboxGroup, RadioGroup, Toggle

# Add a Toggle: toggle
toggle = Toggle(button_type='success', label='Toggle button')

# Add a CheckboxGroup: checkbox
checkbox = CheckboxGroup(labels=['Option 1','Option 2','Option 3'])

# Add a RadioGroup: radio
radio = RadioGroup(labels=['Option 1','Option 2','Option 3'])

# Add widgetbox(toggle, checkbox, radio) to the current document
curdoc().add_root(widgetbox(toggle, checkbox, radio))
```