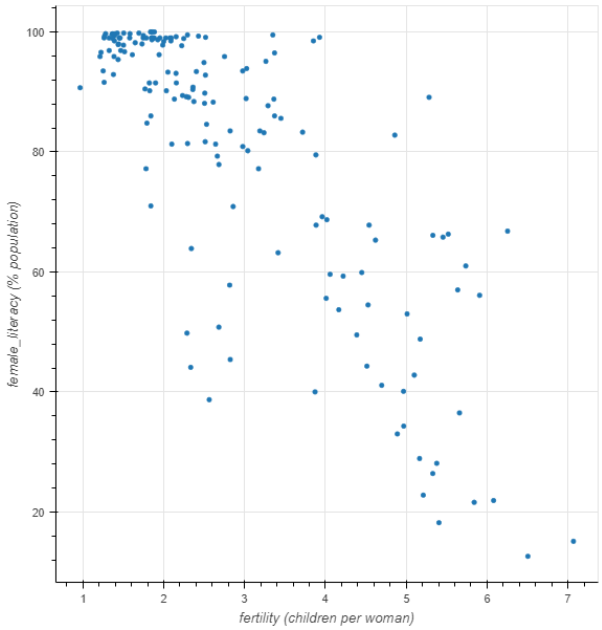
Interactive Data Visualization with Bokeh

1). Basic plotting with Bokeh

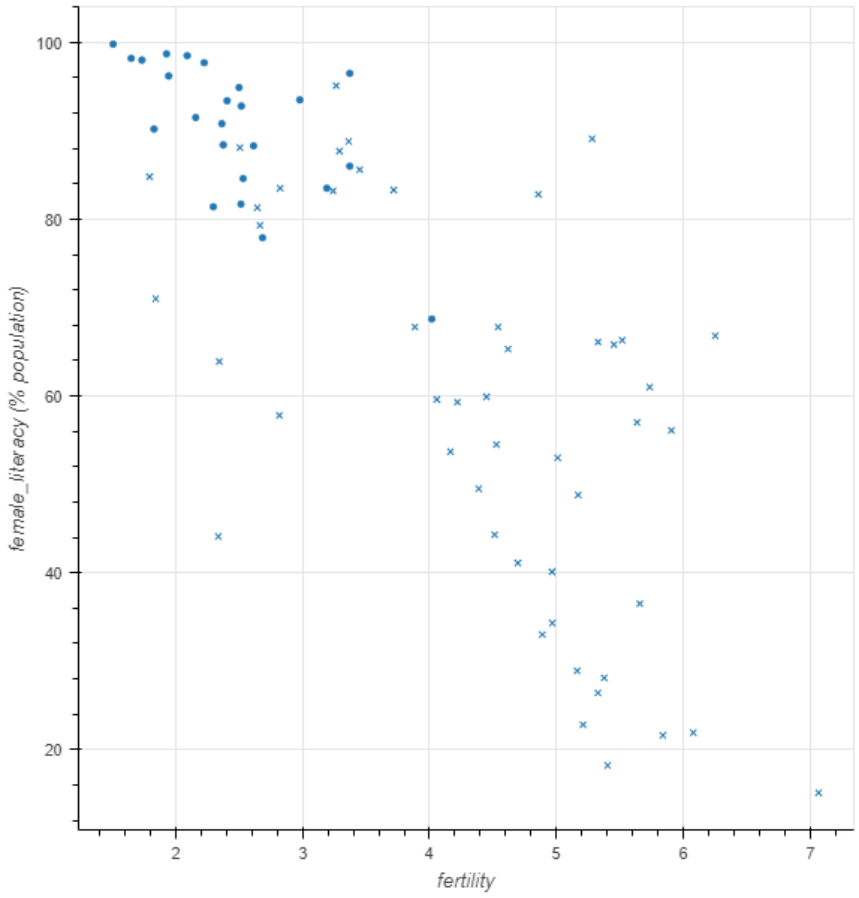
a). A Simple scatter plot

```
# Import figure from bokeh.plotting
from bokeh.plotting import figure

# Import output_file and show from bokeh.io
from bokeh.io import output_file, show

# Create the figure: p
p = figure(x_axis_label='fertility (children per woman)', y_axis_label='female_literacy (% population)')
# Add a circle glyph to the figure p
p.circle(fertility, female_literacy)
# Call the output_file() function and specify the name of the file
output_file('fert_lit.html')
# Display the plot
show(p)
```



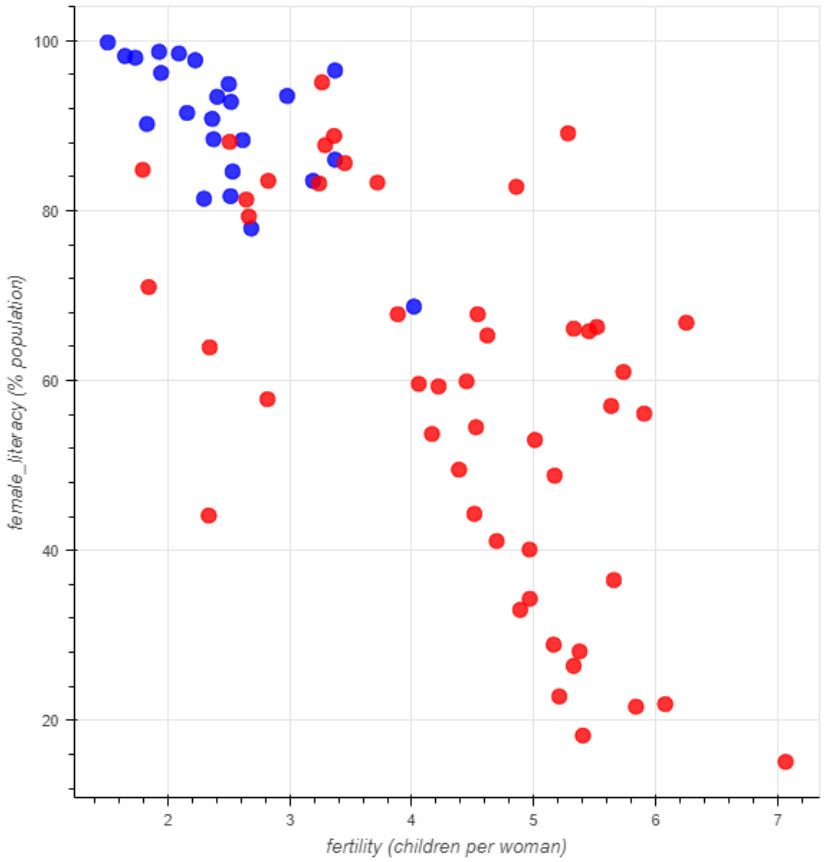
b). A scatter plot with different shapes

```
# Create the figure: p
p = figure(x_axis_label='fertility', y_axis_label='female_literacy (% population)')

# Add a circle glyph to the figure p
p.circle(fertility_latinamerica,female_literacy_latimamerica)

# Add an x glyph to the figure p
p.x(fertility_africa,female_literacy_africa)

# Specify the name of the file
output_file('fert_lit_separate.html')

# Display the plot
show(p)
```



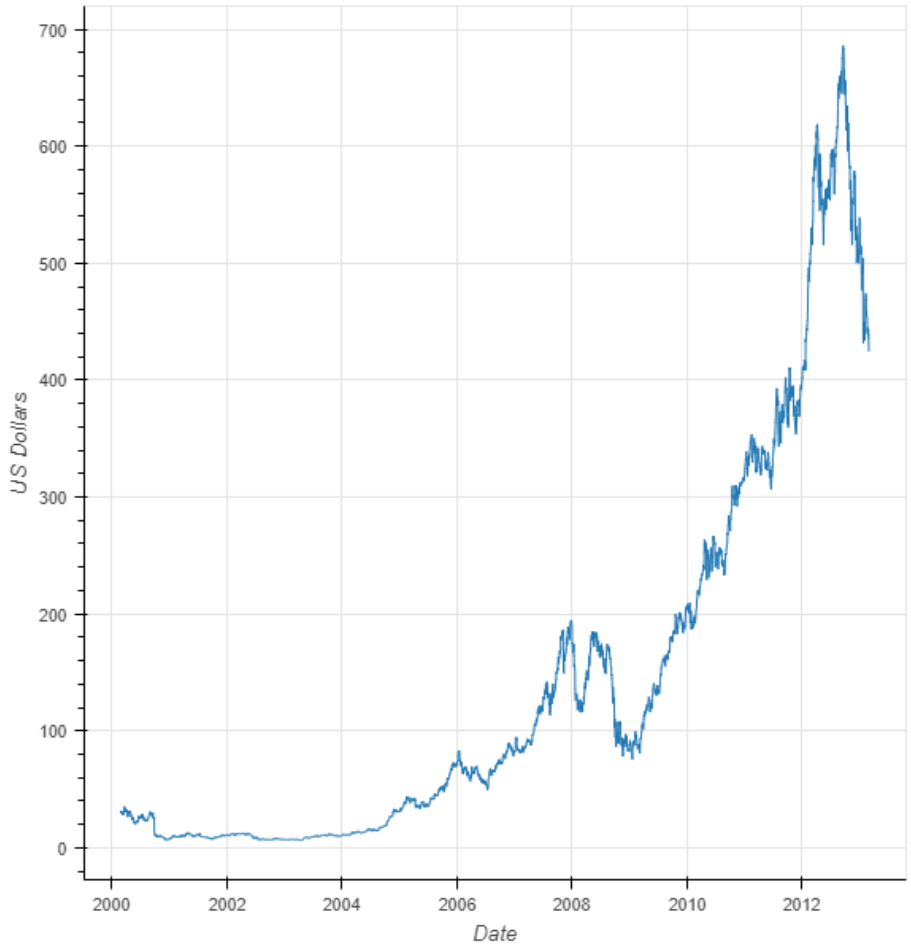
c). Customizing your scatter plots

```
# Create the figure: p
p = figure(x_axis_label='fertility (children per woman)', y_axis_label='female_literacy (% population)')

# Add a blue circle glyph to the figure p
p.circle(fertility_latinamerica, female_literacy_latinamerica, color='blue', size=10, alpha=0.8)

# Add a red circle glyph to the figure p
p.circle(fertility_africa,female_literacy_africa,color='red',size=10,alpha=0.8)

# Specify the name of the file
output_file('fert_lit_separate_colors.html')

# Display the plot
show(p)
```



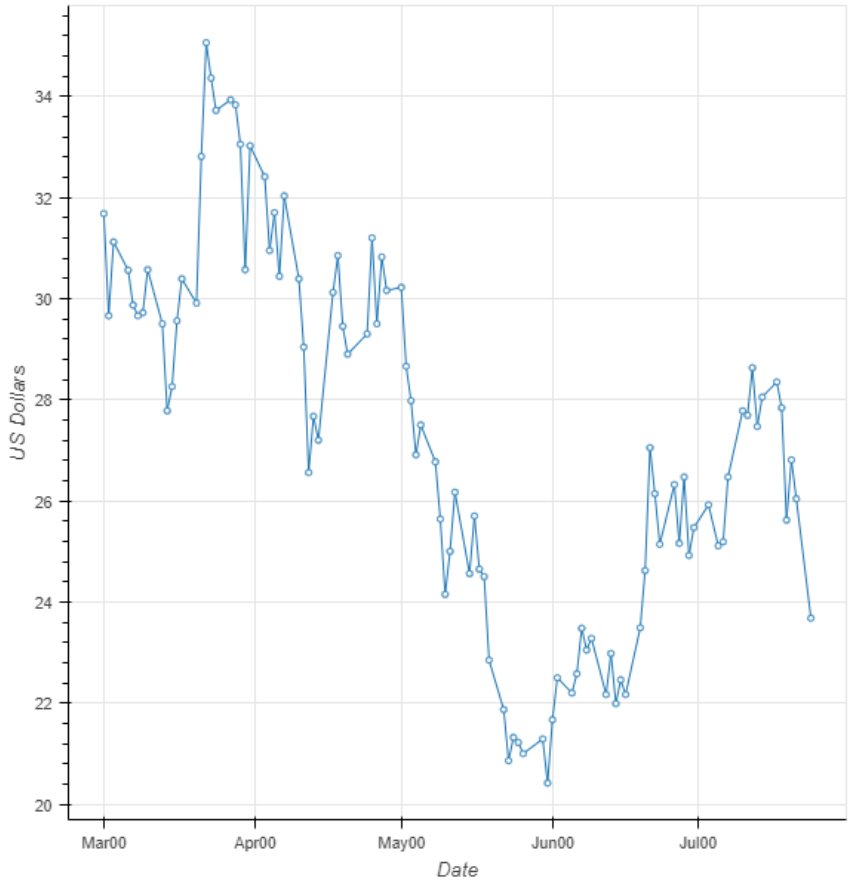
d). Lines

```
# Import figure from bokeh.plotting
from bokeh.plotting import figure

# Create a figure with x_axis_type="datetime": p
p = figure(x_axis_type='datetime', x_axis_label='Date', y_axis_label='US Dollars')

# Plot date along the x axis and price along the y axis
p.line(date,price,)

# Specify the name of the output file and show the result
output_file('line.html')
show(p)
```



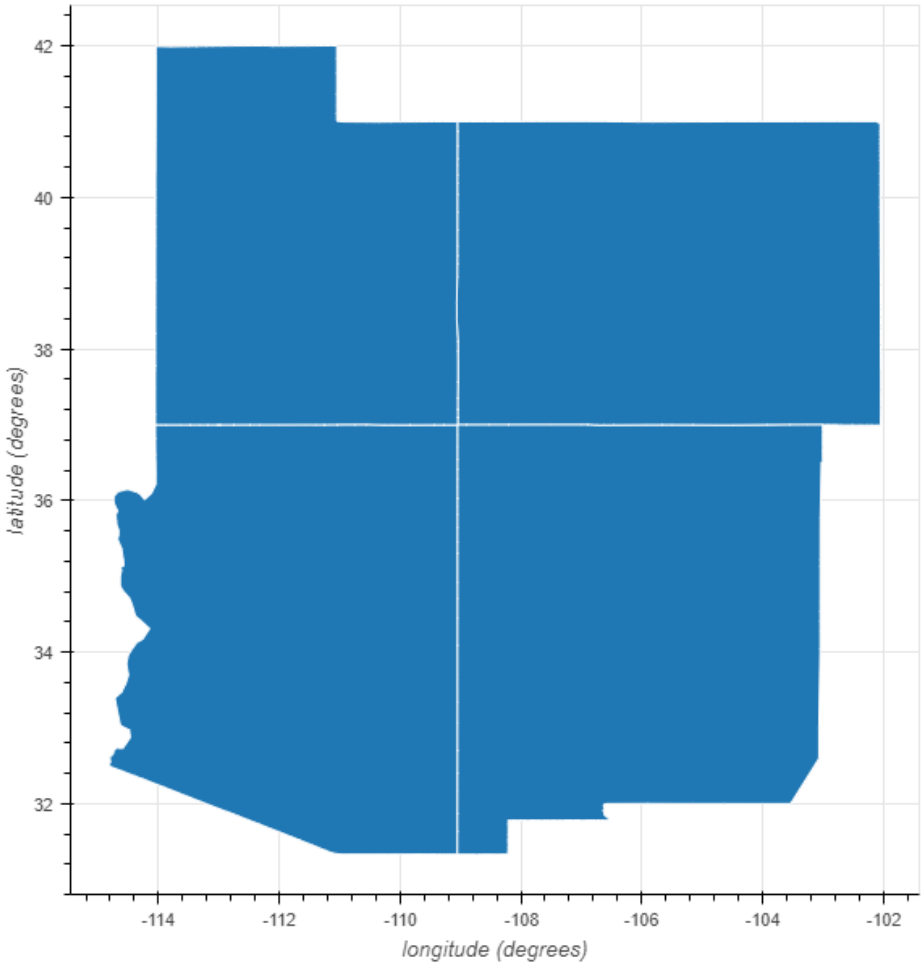
e). Lines and markers

```
# Import figure from bokeh.plotting
from bokeh.plotting import figure

# Create a figure with x_axis_type='datetime': p
p = figure(x_axis_type='datetime', x_axis_label='Date', y_axis_label='US Dollars')

# Plot date along the x-axis and price along the y-axis
p.line(date,price)

# With date on the x-axis and price on the y-axis, add a white circle glyph of size 4
p.circle(date, price, fill_color='white', size=4)
# Specify the name of the output file and show the result
output_file('line.html')
show(p)
```



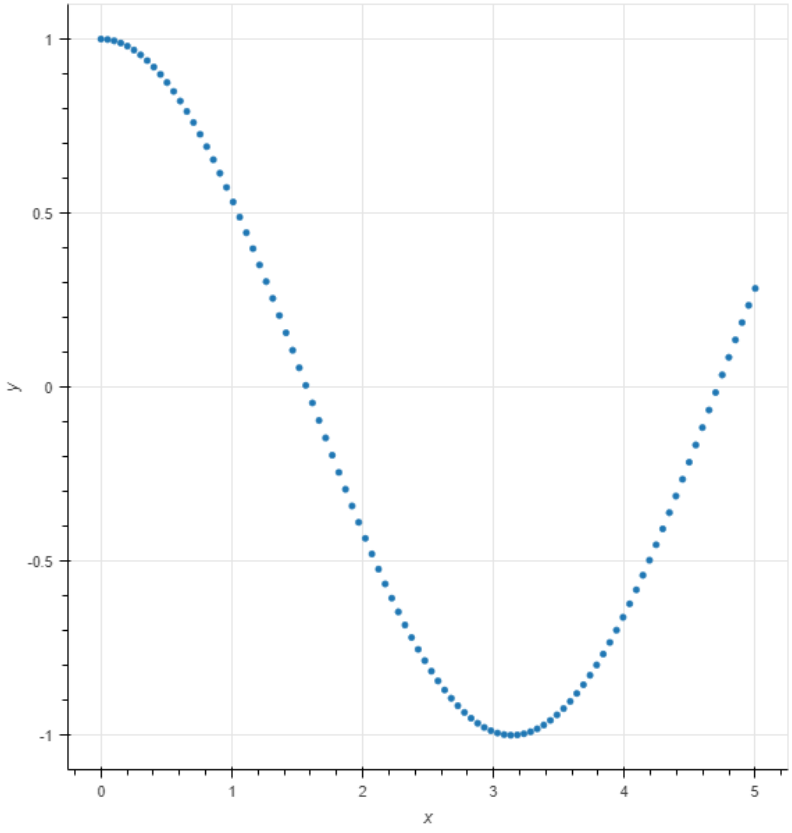
f). Patches

```
# Create a list of az_lons, co_lons, nm_lons and ut_lons: x
x = [az_lons, co_lons, nm_lons, ut_lons]

# Create a list of az_lats, co_lats, nm_lats and ut_lats: y
y = [az_lats, co_lats, nm_lats, ut_lats]

# Add patches to figure p with line_color=white for x and y
p.patches(x,y,line_color='white')

# Specify the name of the output file and show the result
output_file('four_corners.html')
show(p)
```



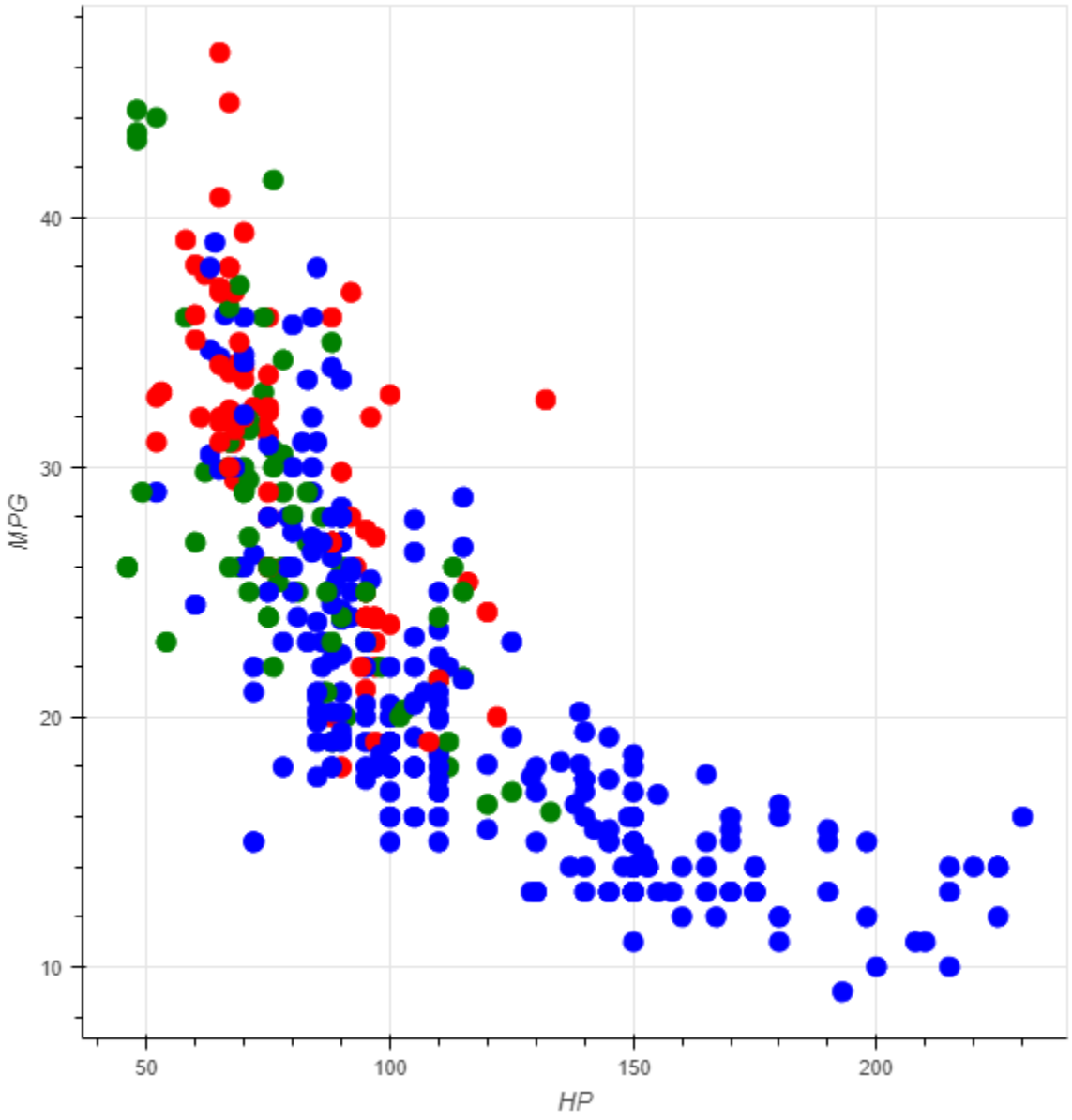
g). Plotting data from numpy arrays

```
# Import numpy as np
import numpy as np

# Create array using np.linspace: x
x = np.linspace(0,5,100)

# Create array using np.cos: y
y = np.cos(x)

# Add circles at x and y
p.circle(x,y)

# Specify the name of the output file and show the result
output_file('numpy.html')
show(p)
```



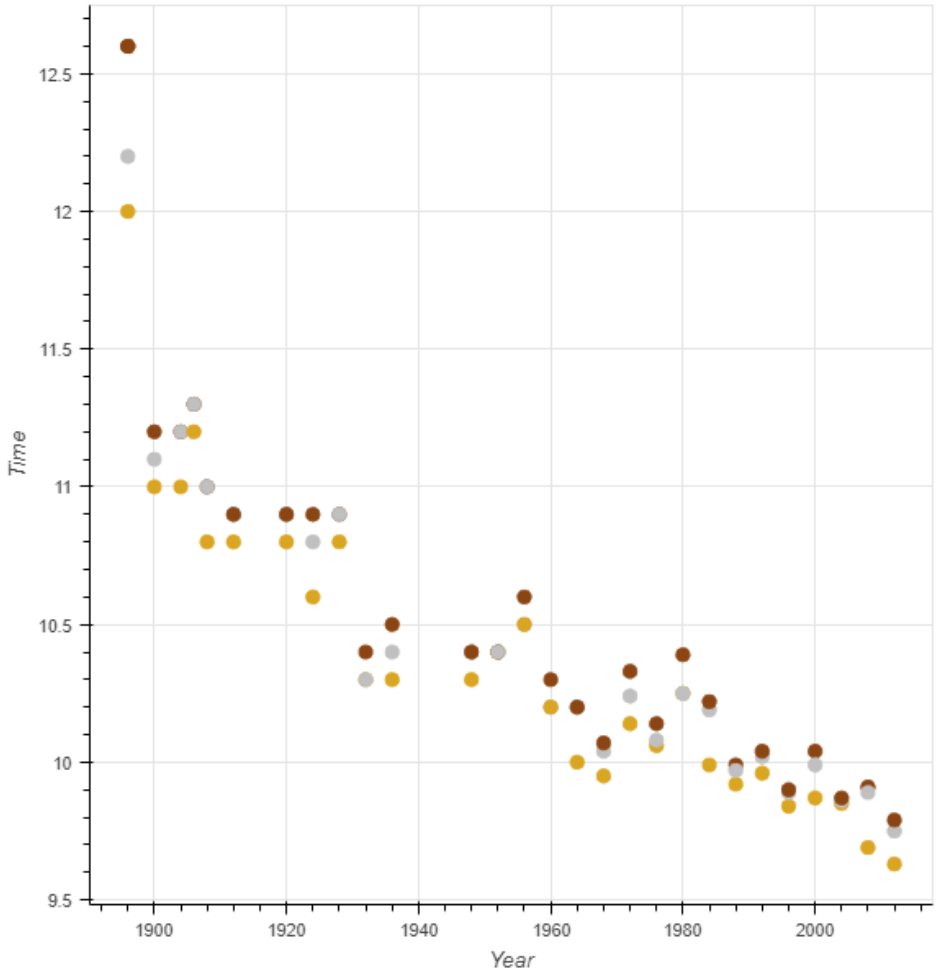
h). Plotting data from Pandas DataFrames

```
# Import pandas as pd
import pandas as pd

# Read in the CSV file: df
df = pd.read_csv('auto.csv')

# Import figure from bokeh.plotting
from bokeh.plotting import figure

# Create the figure: p
p = figure(x_axis_label='HP', y_axis_label='MPG')

# Plot mpg vs hp by color
p.circle(df['hp'],df['mpg'],color=df['color'],size=10)

# Specify the name of the output file and show the result
output_file('auto-df.html')
show(p)
```


40
30
20
10
MPG
50
100
150
200
HP

i). The Bokeh ColumnDataSource (continued)

```
# Import the ColumnDataSource class from bokeh.plotting
from bokeh.plotting import ColumnDataSource

# Create a ColumnDataSource from df: source
source = ColumnDataSource(df)

# Add circle glyphs to the figure p
p.circle('Year','Time',size=8,source=source,color='color')

# Specify the name of the output file and show the result
output_file('sprint.html')
show(p)
```

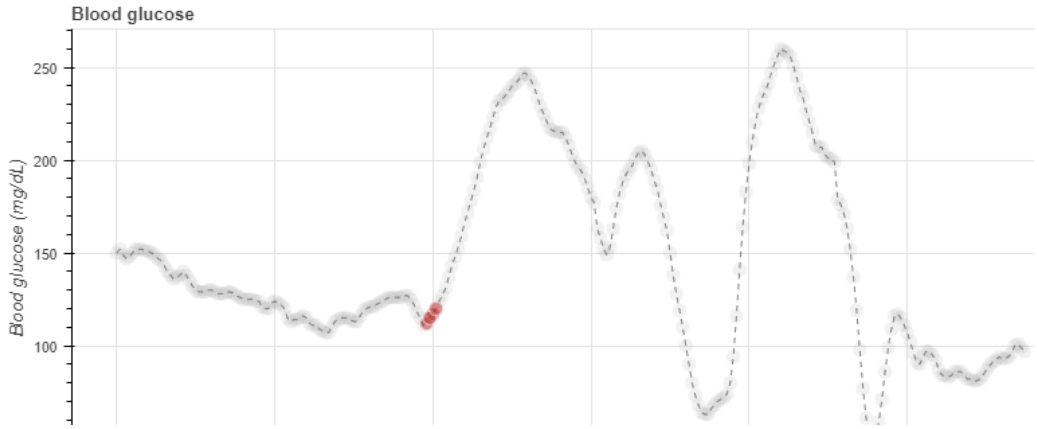


j). Selection & non selection glyphs

```
# Create a figure with the "box_select" tool: p
p = figure(x_axis_label='Year',y_axis_label='Time',tools='box_select')

# Add circle glyphs to the figure p with the selected and non-selected properties
p.circle('Year','Time',selection_color='red',nonselection_alpha=0.1,source=source)

# Specify the name of the output file and show the result
output_file('selection_glyph.html')
show(p)
```

k). Hover glyphs

```
# import the HoverTool
from bokeh.models import HoverTool

# Add circle glyphs to figure p
p.circle(x, y, size=10,
        fill_color='grey', alpha=0.1, line_color=None,
        hover_fill_color='firebrick', hover_alpha=0.5,
        hover_line_color='white')

# Create a HoverTool: hover
hover = HoverTool(tooltips=None,mode='vline')

# Add the hover tool to the figure p
p.add_tools(hover)

# Specify the name of the output file and show the result
output_file('hover_glyph.html')
show(p)
```



l).Colormapping

```
#Import CategoricalColorMapper from bokeh.models
from bokeh.models import CategoricalColorMapper

# Convert df to a ColumnDataSource: source
source = ColumnDataSource(df)

# Make a CategoricalColorMapper object: color_mapper
color_mapper = CategoricalColorMapper(factors=['Europe', 'Asia', 'US'],
                                      palette=['red', 'green', 'blue'])

# Add a circle glyph to the figure p
p.circle('weight', 'mpg', source=source,
         color=dict(field='origin', transform=color_mapper),
         legend='origin')

# Specify the name of the output file and show the result
output_file('colormap.html')
show(p)
```