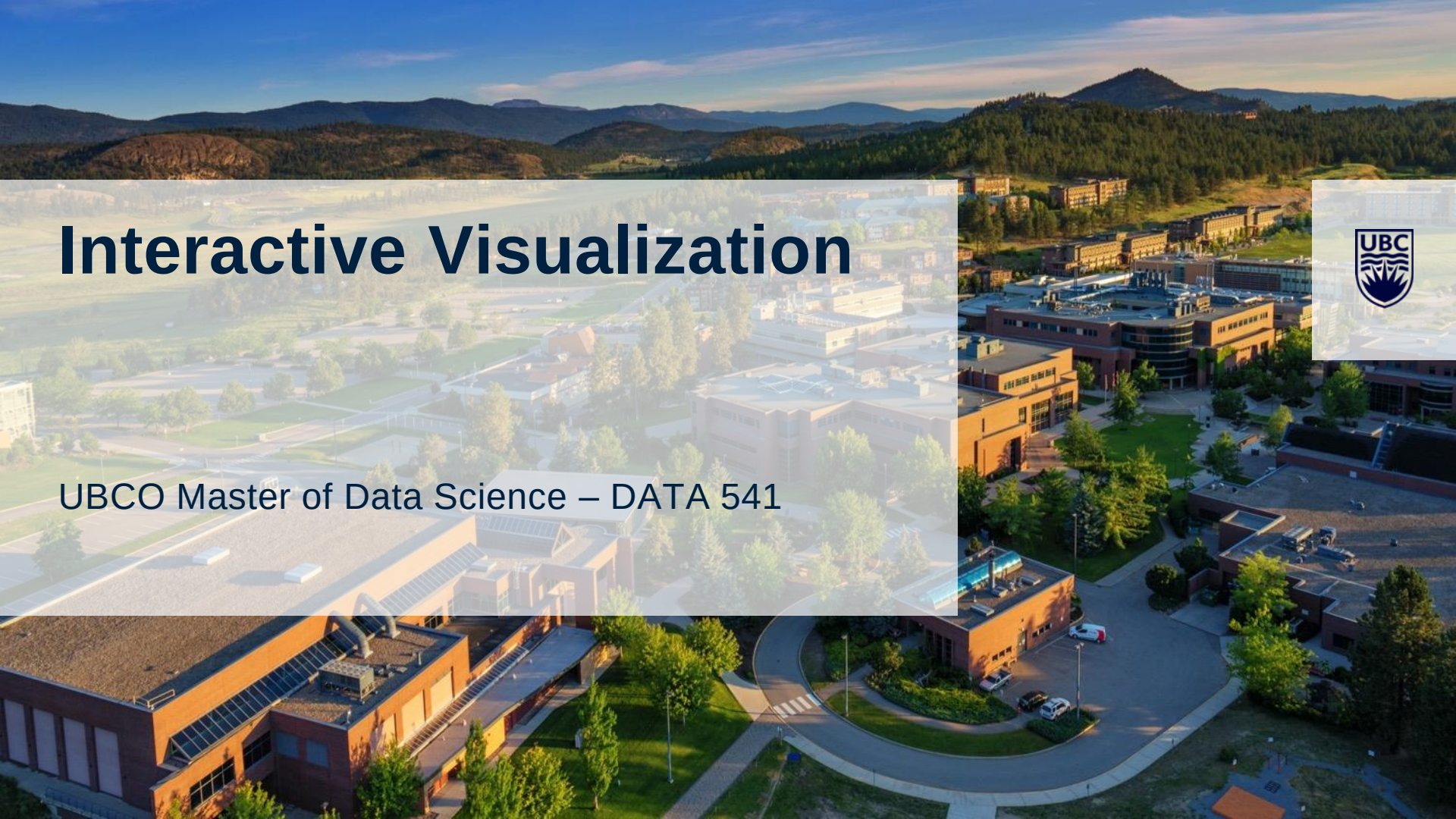


Interactive Visualization

UBCO Master of Data Science – DATA 541

Today's Class

Visualization

Interactive Visualization

Python Libraries for Interactive Visualization

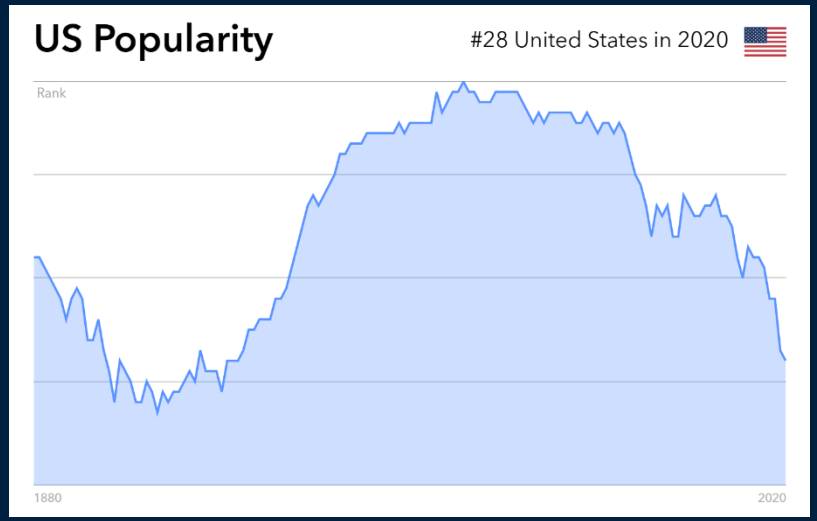
Pygal

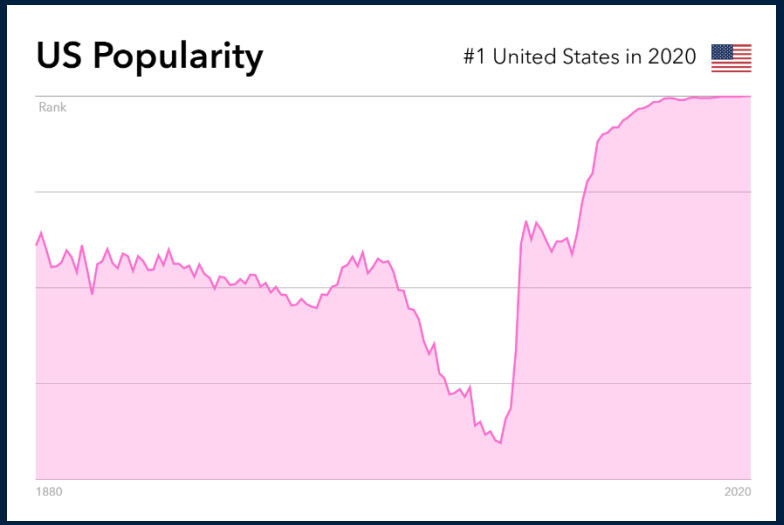
Bokeh

Visualization

Visual Representation of Data

Baby Name (https://mom.com/baby-names/)





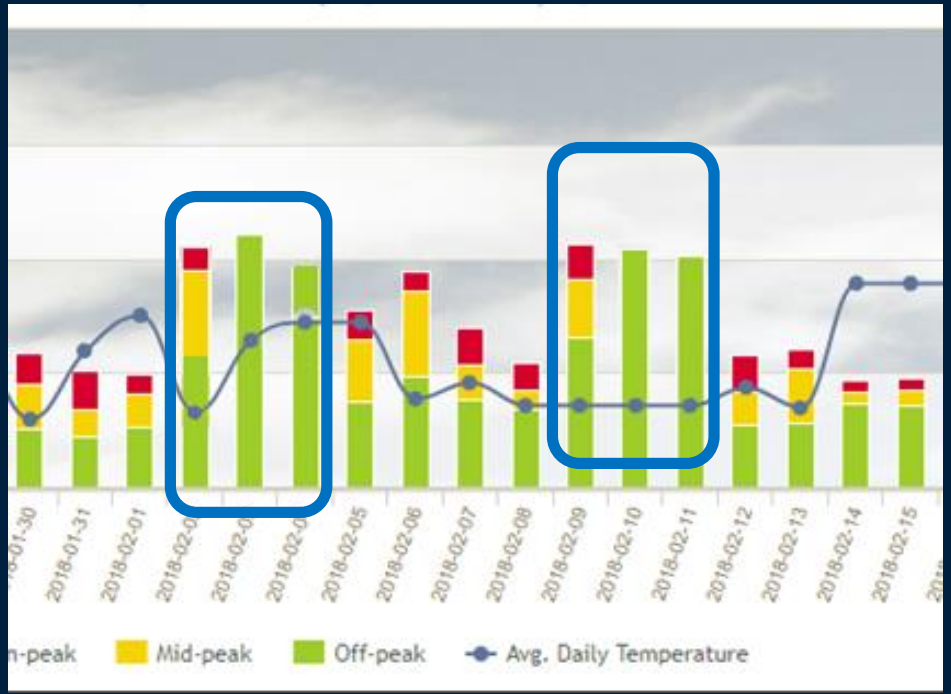
Visualization

Visualization is the process of graphically presenting data to reveal its patterns, trends, and meanings

Simplifies data values, promotes the understanding of data, and communicates important concepts and ideas

Example

Reading Date	Total On-Peak	Total Mid-Peak	Total Off-Peak	Total Consumption
2018-01-21	0	0	12.49	12.49
2018-01-22	0.96	1.41	6.56	8.93
2018-01-23	2.41	3.37	7.16	12.94
2018-01-24	5.9	1.78	7.89	15.57
2018-01-25	2.36	3.52	5.11	10.99
2018-01-26	3.47	3.3	5.94	12.71
2018-01-27	0	0	13.39	13.39
2018-01-28	0	0	11.24	11.24
2018-01-29	6.14	2.25	12.22	20.61
2018-01-30	2.58	3.92	5.16	11.66
2018-01-31	3.4	2.31	4.5	10.21
2018-02-01	1.69	2.9	5.2	9.79
2018-02-02	2.08	9.53	4.24	15.85
2018-02-03	0	0	22.1	22.1
2018-02-04	0	0	14.85	14.85
2018-02-05	2.63	5.36	7.5	15.49
2018-02-06	1.58	7.53	9.74	18.85
2018-02-07	3.26	3.06	7.69	14.01
2018-02-08	2.29	1.66	6.89	10.84
2018-02-09	3.02	5.1	13.17	21.29
2018-02-10	0	0	17.03	17.03
2018-02-11	0	0	20.28	20.28
2018-02-12	3.27	2.9	5.46	11.63
2018-02-13	1.6	4.81	5.64	12.05
2018-02-14	1.08	1	7.28	9.36
2018-02-15	1.02	1.46	7.08	9.56
2018-02-16	2.83	7.43	5.4	15.66
2018-02-17	0	0	12.2	12.2
2018-02-18	0	0	9.21	9.21



Static and Interactive Visualization

Static visualizations are

- commonly seen images posted on the web or printed as handouts.
- usually focused on a specific data story
- users can't go beyond a single view to explore additional stories beyond what's in front of them.

Interactive visualizations are

- commonly seen on the web only as applications.
- users can select specific data points to build a visualized story of their choosing.
- allow the user to be part of the data visualization process by building a story of their choosing.

Python Libraries for Interactive Visualization

Pygal

- http://www.pygal.org/

Bokeh

- https://bokeh.pydata.org/

Plotly

- https://plot.ly/python/

mpld3

- http://mpld3.github.io/

HoloViews

- http://holoviews.org/

Pygal

Pygal is a useful tool to create beautiful interactive charts with very few lines of code

Pygal specializes in allowing the user to create SVGs.

SVGs can be added to a web page with an embed tag or by inserting the code directly into the HTML.

Installation

The recommended way to install `pygal` is using the following command (from Anaconda Prompt):

```
$ pip install pygal
```

or

```
$ sudo pip install pygal
```

Bar Chart

```
import pygal
```

First import pygal

```
x = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Add some values

```
bar_chart = pygal.Bar()
```

Then create a bar graph object

```
bar_chart.add('Fibonacci',x)
```

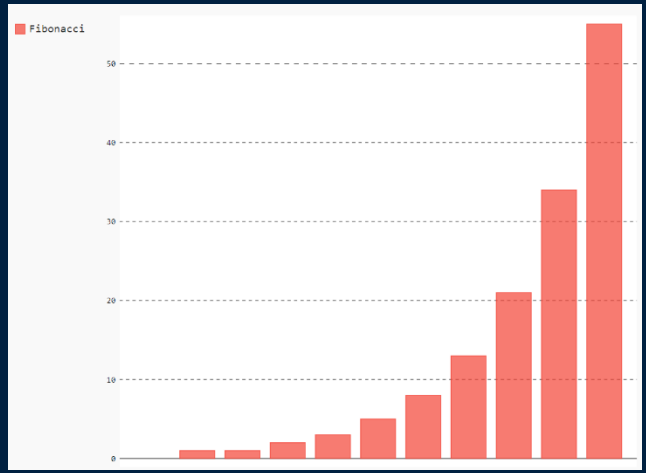
Add the values to the chart

```
bar_chart.render_to_file('bar_chart.svg')
```

Save the svg to a file

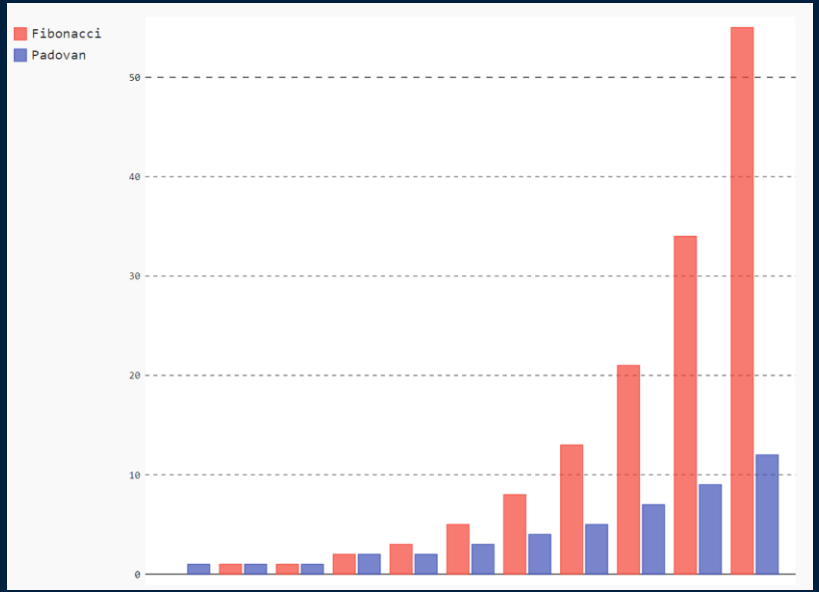
For Jupyter Notebook output:

```
from IPython.display import SVG, display

display(SVG(bar_chart.render
(disable_xml_declaration=True)))
```



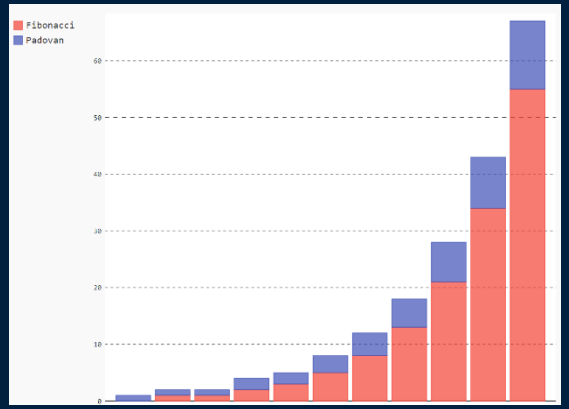
Multiple Series

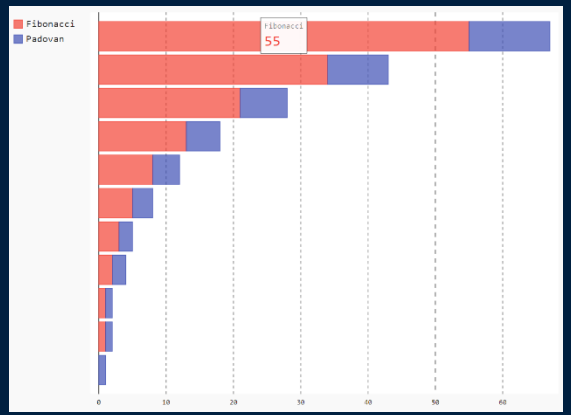
```
import pygal
x = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
y = [1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12]
bar_chart = pygal.Bar()
bar_chart.add('Fibonacci', x)
bar_chart.add('Padovan', y)
bar_chart.render_to_file('outputML.svg')
```



Stacked Bar

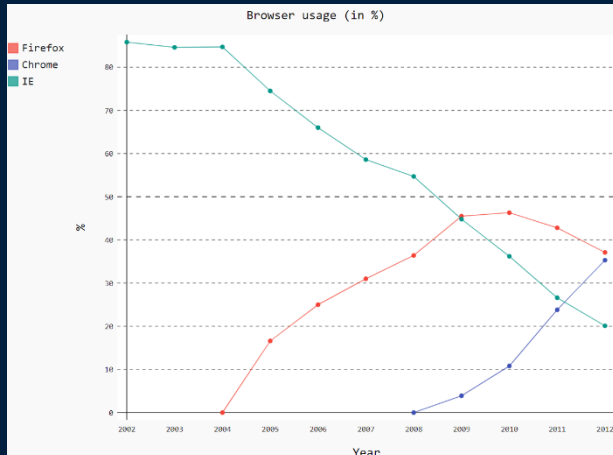
```
import pygal
x = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
y = [1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12]
bar_chart = pygal.StackedBar()
bar_chart.add('Fibonacci', x)
bar_chart.add('Padovan', y)
bar_chart.render_to_file('StackedBar.svg')


…
bar_chart = pygal.HorizontalStackedBar()
…
```





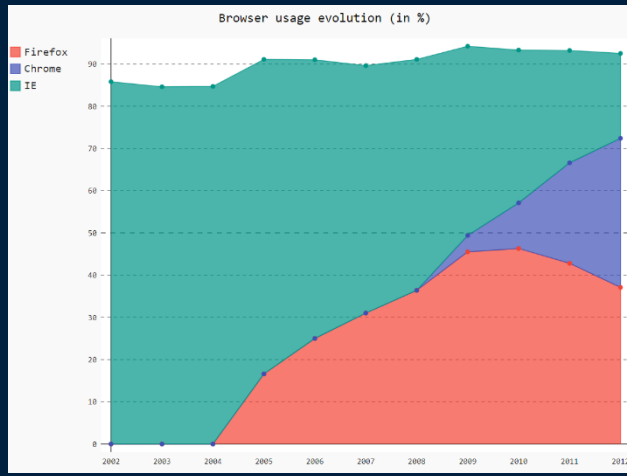
Line Graph

```
import pygal
line_chart = pygal.Line()
fx = [None, None,0, 16.6, 25, 31, 36.4, 45.5, 46.3, 42.8, 37.1]
ch = [None, None, None, None, None, None, 0, 3.9, 10.8, 23.8, 35.3]
ie = [85.8, 84.6, 84.7, 74.5, 66, 58.6, 54.7, 44.8, 36.2, 26.6, 20.1]
line_chart.title = 'Browser usage (in %)'
line_chart.x_title='Year'
line_chart.y_title='%'
line_chart.x_labels = map(str, range(2002, 2013))
line_chart.add('Firefox', fx)
line_chart.add('Chrome',  ch)
line_chart.add('IE', ie)
line_chart.render_to_file('line.svg')
```



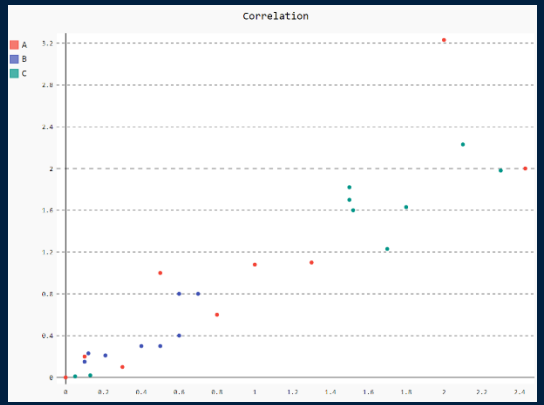
Stacked

```
import pygal
line_chart = pygal.StackedLine(fill=True)
fx = [None, None,0, 16.6, 25, 31, 36.4, 45.5, 46.3, 42.8, 37.1]
ch = [None, None, None, None, None, None, 0, 3.9, 10.8, 23.8, 35.3]
ie = [85.8, 84.6, 84.7, 74.5, 66, 58.6, 54.7, 44.8, 36.2, 26.6, 20.1]
line_chart.title = 'Browser usage (in %)'
line_chart.x_labels = map(str, range(2002, 2013))
line_chart.add('Firefox', fx)
line_chart.add('Chrome',  ch)
line_chart.add('IE', ie)
line_chart.render_to_file('stacked.svg')
```



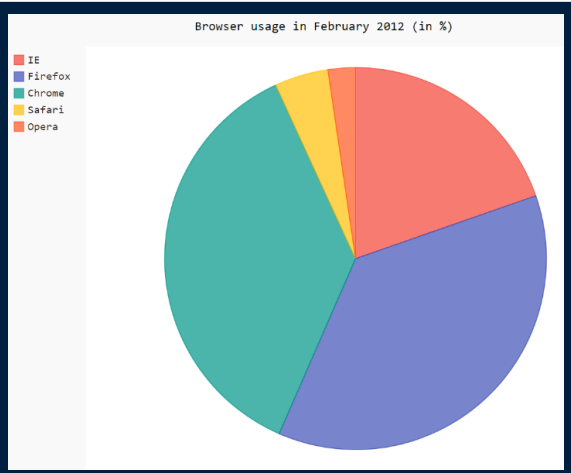
Scatter Plot

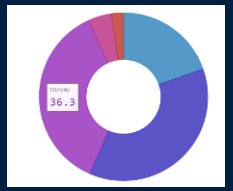
```
import pygal

xy_chart = pygal.XY(stroke=False)

set1 = [(0, 0), (.1, .2), (.3, .1), (.5, 1), (.8, .6), (1, 1.08), (1.3,
1.1), (2, 3.23), (2.43, 2)]

set2 = [(.1, .15), (.12, .23), (.4, .3), (.6, .4), (.21, .21), (.5,
.3), (.6, .8), (.7, .8)]

set3 = [(.05, .01), (.13, .02), (1.5, 1.7), (1.52, 1.6), (1.8, 1.63),
(1.5, 1.82), (1.7, 1.23), (2.1, 2.23), (2.3, 1.98)]

xy_chart.title = 'Correlation'

xy_chart.add('A', set1)

xy_chart.add('B', set2)

xy_chart.add('C', set3)

xy_chart.render_to_file('scatter.svg')
```



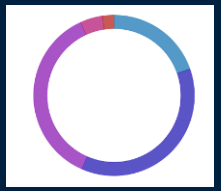
Pie

```
import pygal

pie_chart = pygal.Pie()

pie_chart.title = 'Browser usage in
February 2012 (in %)'

pie_chart.add('IE', 19.5)

pie_chart.add('Firefox', 36.6)

pie_chart.add('Chrome', 36.3)

pie_chart.add('Safari', 4.5)

pie_chart.add('Opera', 2.3)

pie_chart.render_to_file('pie.svg')
```

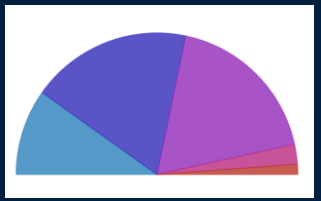




Donut



Ring



Half-Pie

Other types: https://www.pygal.org/en/stable/documentation/types/pie.html

Box

```
import pygal
box_plot = pygal.Box()
box_plot.title = 'V8 benchmark results'
box_plot.add('Chrome', [639, 821, 752, 721, 1246, 166, 212, 860])
box_plot.add('Firefox', [747, 809, 1170, 265, 636, 104, 379, 945])
box_plot.add('Opera', [347, 293, 420, 522, 580, 188, 903, 469])
box_plot.add('IE', [223, 211, 229, 349, 314, 416, 414,112])
box_plot.render_to_file('box.svg')
```

Other options:
https://www.pygal.org/en/stable/documentation/types/box.html

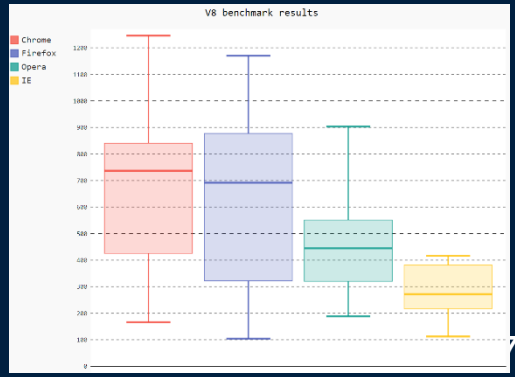


Chart Basic Question

Question: The following graph is an example of:

A) Horizontal bar graph

B) Stacked bar graph

C) Line graph

D) Scatter plot

E) Box plot

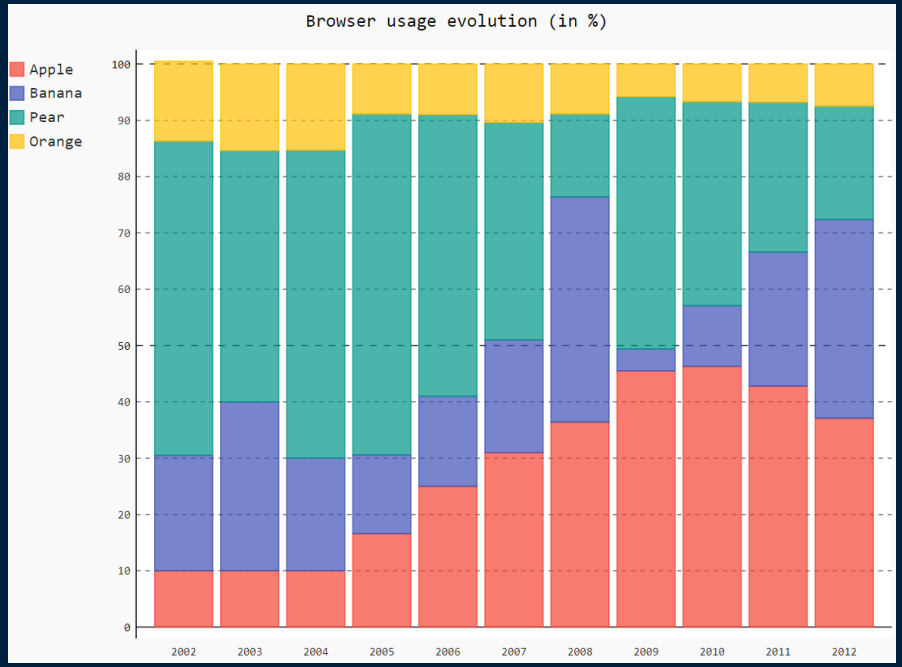
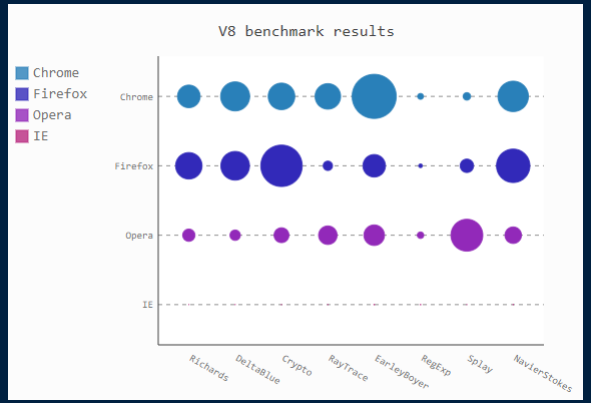
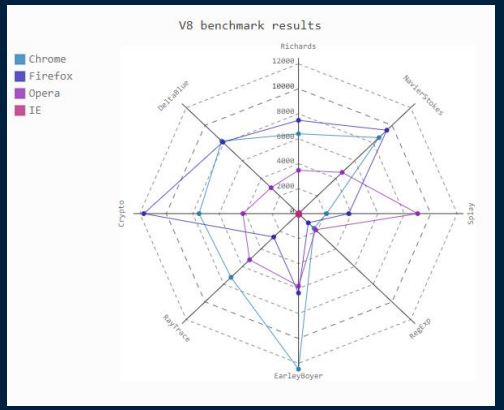


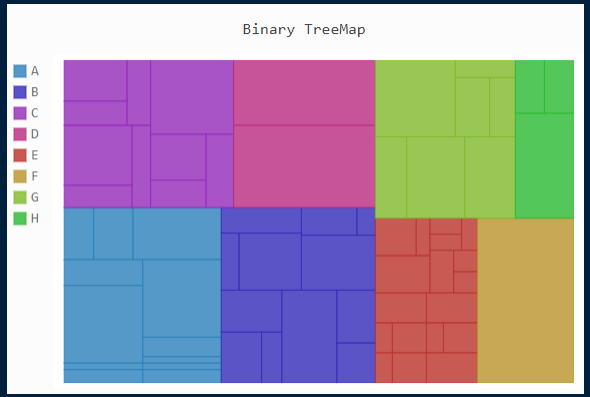
Chart Types



Dot



Radar



Treemap

Dot: http://www.pygal.org/en/stable/documentation/types/dot.html
Radar: http://www.pygal.org/en/stable/documentation/types/radar.html
Treemap: http://www.pygal.org/en/stable/documentation/types/treemap.html
Other: http://www.pygal.org/en/stable/documentation/types/index.html

Built-in Style

http://www.pygal.org/en/latest/documentation/builtin_styles.html

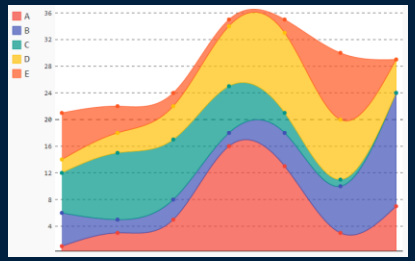
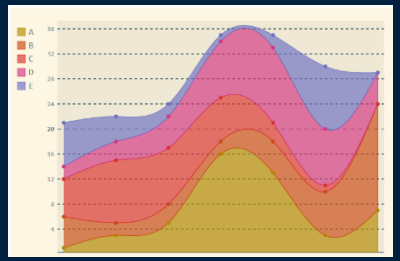
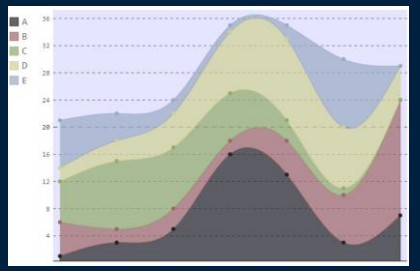
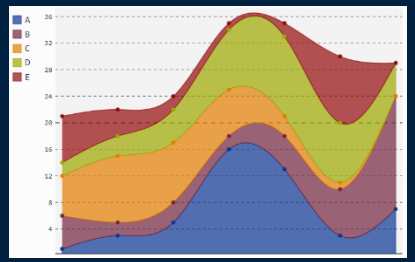
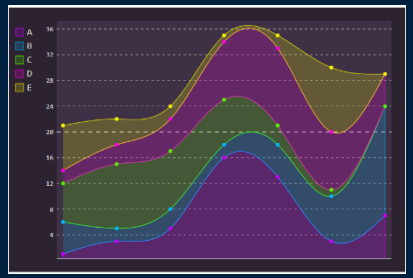
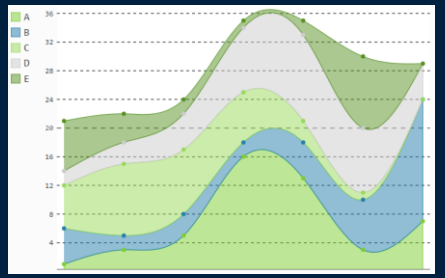







Chart Configuration

Sizing

- width
- height
- explicit_size
- spacing
- margin
- margin_top
- margin_right
- margin_bottom
- margin_left

Titles

- title
- x_title
- y_title

Legend

- show_Legend
- legend_at_bottom
- legend_box_size

- More options
  - http://www.pygal.org/en/latest/documentation/configuration/chart.html

Serie Configuration

More information:

http://www.pygal.org/en/latest/documentation/configuration/serie.html

stroke

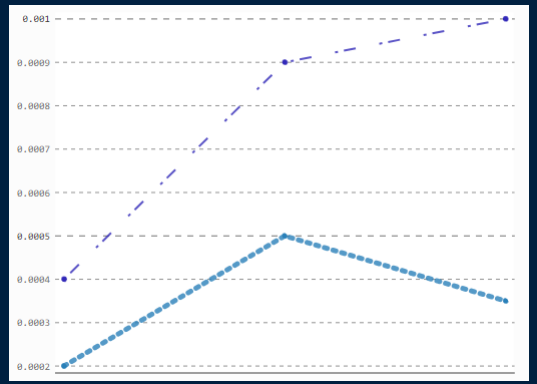
fill

show_dots

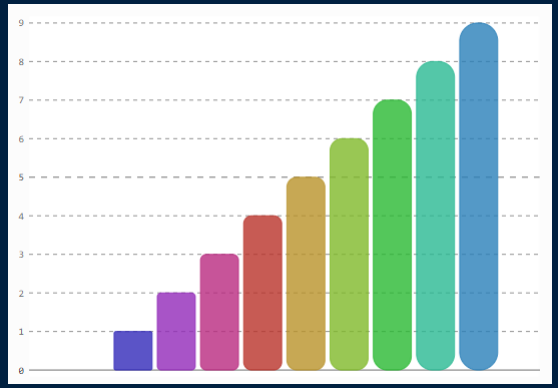
dots_size

stroke_style

rounded_bars



Stroke Style



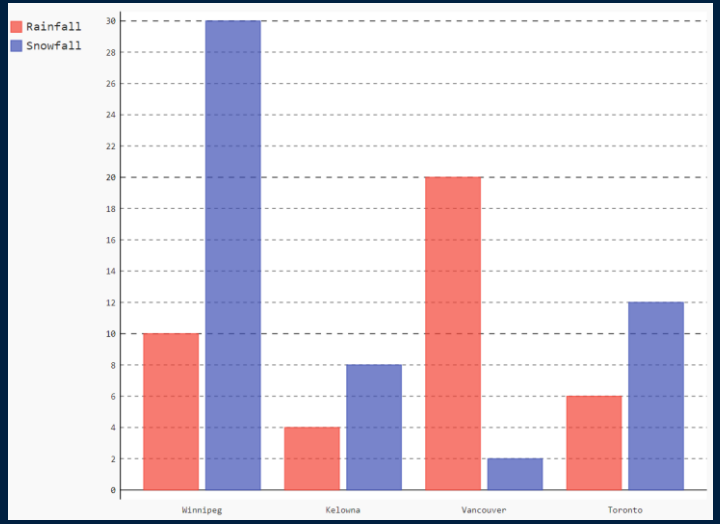
Rounded bar

Read from a CSV File

```
import pygal
import pandas as pd
data = pd.read_csv('c:\\data\\sample1.csv')
bar_chart = pygal.Bar()
for index, row in data.iterrows():
    bar_chart.add(row["City"], row["Rainfall"])
bar_chart.render_to_file('output.svg')
```

Read from a CSV File

```
import pygal
import pandas as pd

data = pd.read_csv('c:\\data\\sample1.csv')
city=data['City'].values[:].tolist()
rainfall=data['Rainfall'].values[:].tolist()
snowfall=data['Snowfall'].values[:].tolist()

bar_chart = pygal.Bar()
bar_chart.x_labels = city
bar_chart.add('Rainfall', rainfall)
bar_chart.add('Snowfall', snowfall)
bar_chart.render_to_file('output2.svg')
```



Try it: Pygal Multi-Line Graph

Show the following information in a multi-line graph:

City	Rainfall	Snowfall
Winnipeg	10	30
Kelowna	4	8
Vancouver	20	2
Toronto	6	12

Bokeh

Bokeh

- builds complex statistical plots quickly through simple commands
- provides output in various formats like HTML and notebook
- provides flexibility for applying interaction, layouts and different styling option to visualization

Key Concepts

Glyphs

- The basic visual building blocks of Bokeh plots, e.g. lines, rectangles, squares, wedges, patches, etc.
- The `bokeh.plotting` interface provides a convenient way to create plots centered around glyphs.

Widgets

- User interface elements outside of a Bokeh plot such as sliders, drop down menus, buttons, etc.
- Events and updates from widgets can inform additional computations, or cause Bokeh plots to update.

Example

```
from bokeh.plotting import figure, output_file, show

x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]

output_file("lines.html")

p = figure(title="simple line example", x_axis_label='x', y_axis_label='y')

p.line(x, y, legend_label="Temp.", line_width=2)

show(p)
```

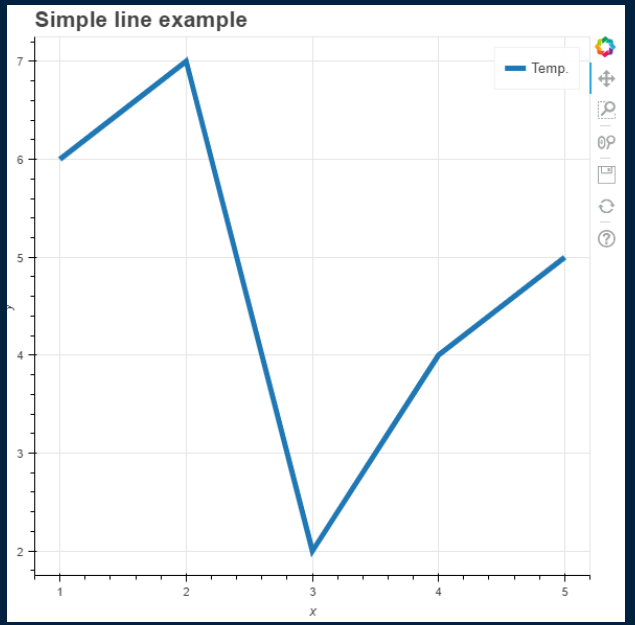
prepare data

output to HTML

new plot with a title and axis labels

add a line

show the results



Bokeh - Steps

Prepare some data

- Plain python lists, but could also be `NumPy` arrays or `Pandas` series.

Tell Bokeh where to generate output

- using `output_file()`, with a filename (e.g., “output.html")
- Another option is `output_notebook()` for use in Jupyter notebooks.

Call figure()

- Creates a plot with default options (e.g., title, tools, and axes labels).

Add renderers

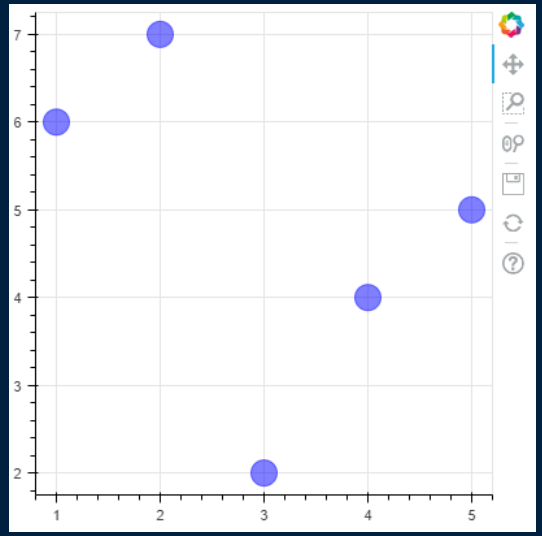
- Specifying visual customizations like colors, legends and widths.

Ask Bokeh to show() or save() the results.

- Save the plot to an HTML file and optionally display it in a browser.

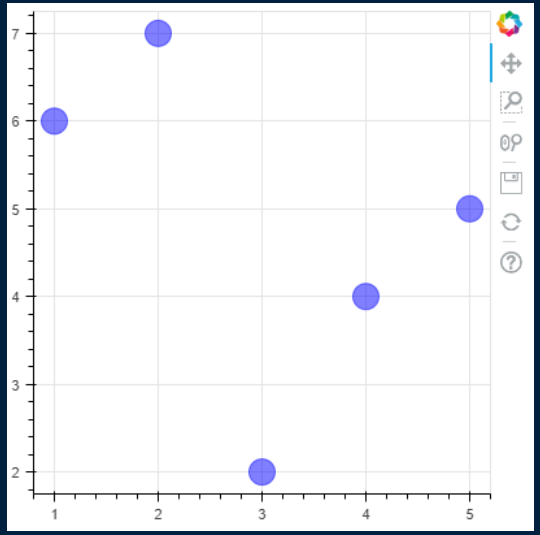
Scatter Markers

```
from bokeh.plotting import figure, output_file,
show


x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]


output_file("output.html")


p = figure(plot_width=400, plot_height=400)


p.circle(x, y, size=20, color="blue", alpha=0.5)


show(p)
```



ColumnDataSource

```
from bokeh.models import ColumnDataSource
source = ColumnDataSource(data={
    'x' : [1, 2, 3, 4, 5],
    'y' : [3, 7, 8, 5, 1],
})
p = figure(plot_width=400, plot_height=400)
p.circle('x', 'y', size=20, source=source)
show(p)
```

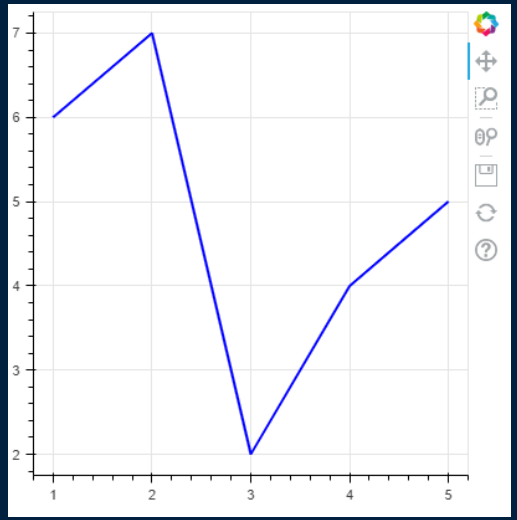


Other Options

```
square()
asterisk()
circle()
circle_cross()
cross()
diamond()
diamond_cross()
inverted_triangle()
square()
square_cross()
square_x()
triangle()
x()
```

Line Glyphs

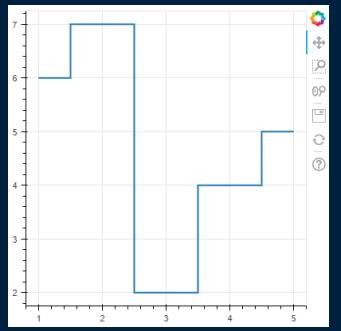
```
from bokeh.plotting import figure, output_file,
show

x = [1, 2, 3, 4, 5]

y = [6, 7, 2, 4, 5]


output_file("output.html")


p = figure(plot_width=400, plot_height=400)

p.line(x, y, line_width=2, color="blue")


show(p)
```



Other Options

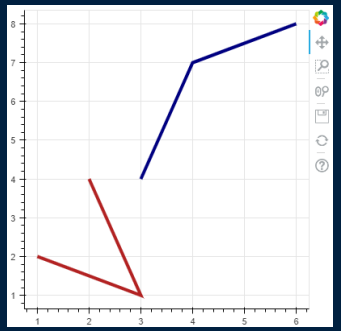
Step Lines

```
p.step(x, y, line_width=2, mode="center")
mode="before", mode="after"
```



Multiple Lines

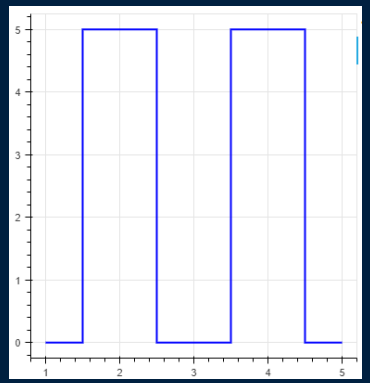
```
x1 = [1, 3, 2] y1 = [3, 4, 6]
x2 = [2, 1, 4] y2 = [4, 7, 8]
p.multi_line([x1, y1], [x2, y2],
color=["firebrick", "navy"], line_width=4)
```



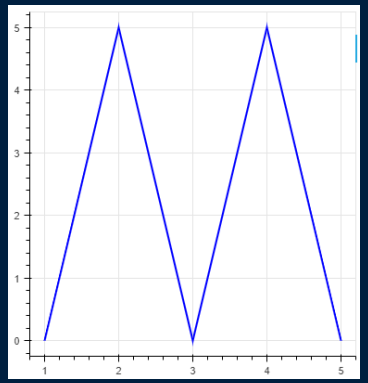
Line Glyphs Question

Question: The following code will produce:

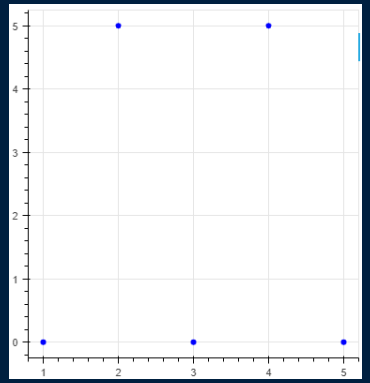
```
x = [1, 2, 3, 4, 5]
y = [0, 5, 0, 5, 0]
p = figure(plot_width=400, plot_height=400)
p.step(x, y, line_width=2, color="blue", mode="center")
show(p)
```



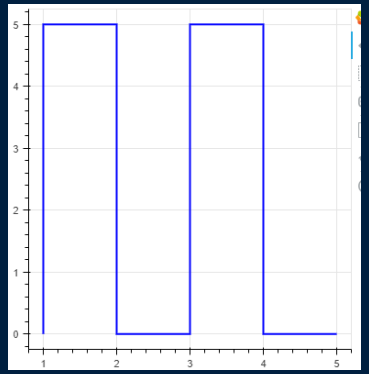
A)



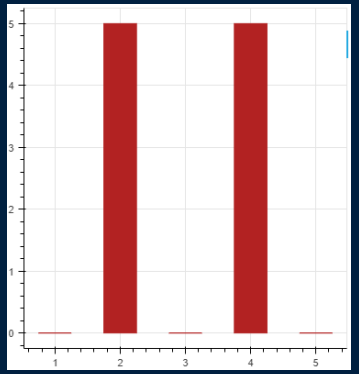
B)



C)



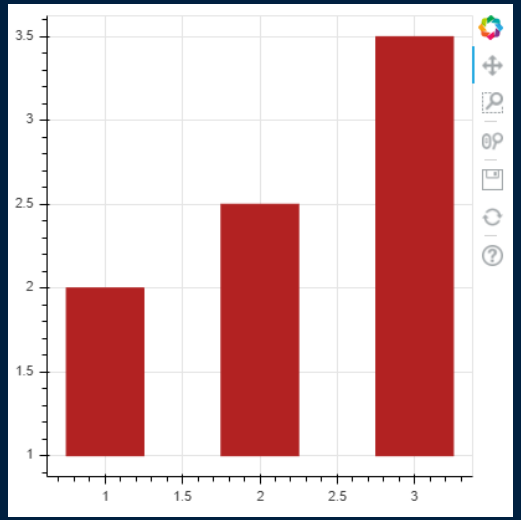
D)



E)

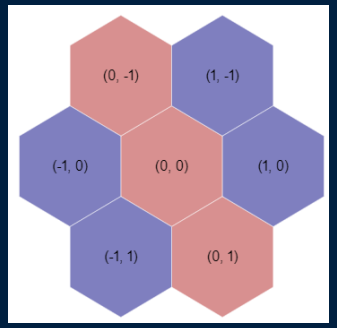
Bars

```
from bokeh.plotting import figure, show, output_file

output_file('output.html')

p = figure(plot_width=400, plot_height=400)
p.vbar(x=[1, 2, 3], width=0.5, bottom=1,
       top=[2, 2.5, 3.5], color="firebrick")

show(p)
```

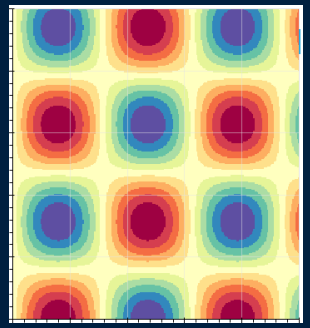


To draw horizontal bars: use the `hbar()` function

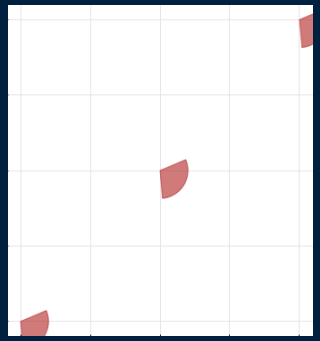
Other Glyphs



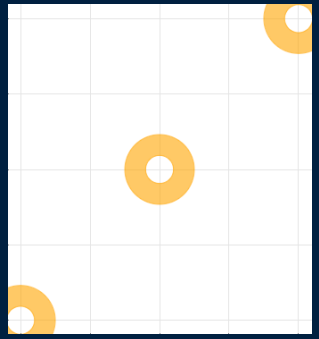
Hex Tiles



Images



Wedges

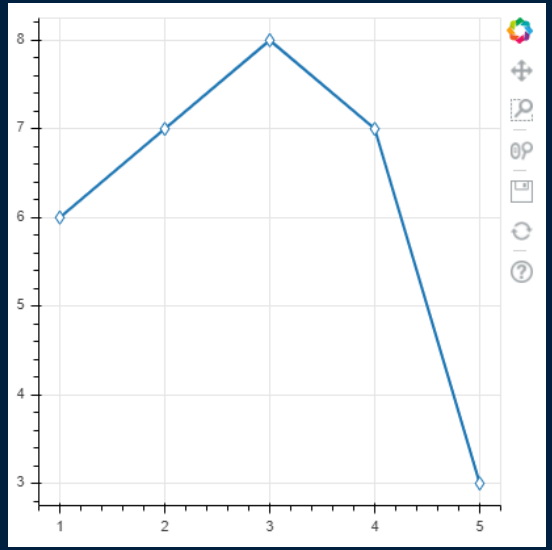


Annular

More information: https://docs.bokeh.org/en/latest/docs/user_guide/plotting.html

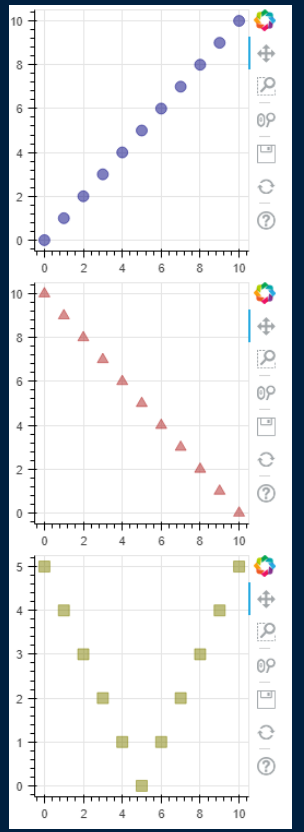
Combining Multiple Glyphs

```
from bokeh.plotting import figure, output_file, show

x = [1, 2, 3, 4, 5]
y = [6, 7, 8, 7, 3]

output_file("output.html")

p = figure(plot_width=400, plot_height=400)

p.line(x, y, line_width=2)
p.diamond(x, y, fill_color="white", size=10)

show(p)
```



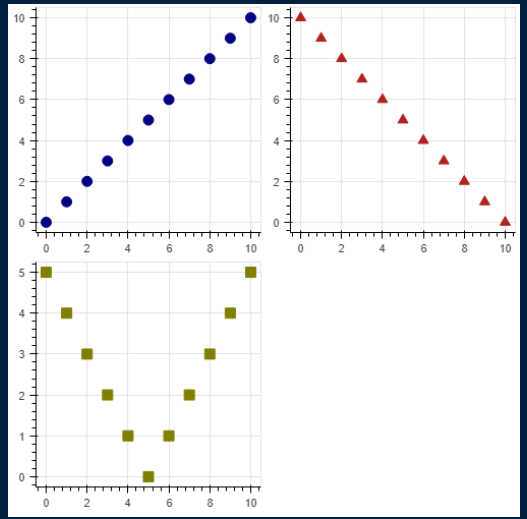
Rows and Columns

```
from bokeh.layouts import column
from bokeh.plotting import figure, output_file, show
x = list(range(11))
y0,y1,y2 = x,[10-i for i in x],[abs(i-5) for i in x]
s1 = figure(width=250, plot_height=250)
s1.circle(x, y0, size=10, color="navy")
s2 = figure(width=250, height=250)
s2.triangle(x, y1, size=10, color="firebrick")
s3 = figure(width=250, height=250)
s3.square(x, y2, size=10, color="olive")
show(column(s1, s2, s3))
```



Grid Plots

```
from bokeh.layouts import gridplot

s1 = figure(width=250, plot_height=250)
s1.circle(x, y0, size=10, color="navy")
s2 = figure(width=250, height=250)
s2.triangle(x, y1, size=10, color="firebrick")
s3 = figure(width=250, height=250)
s3.square(x, y2, size=10, color="olive")

p = gridplot([[s1, s2], [s3, None]], toolbar_location=None)
show(p)
```



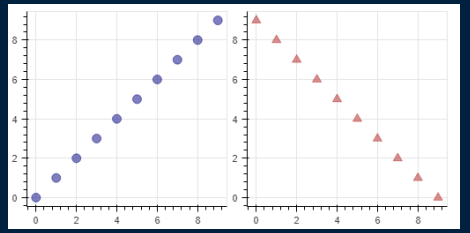
Try it: Bokeh Grid Plot

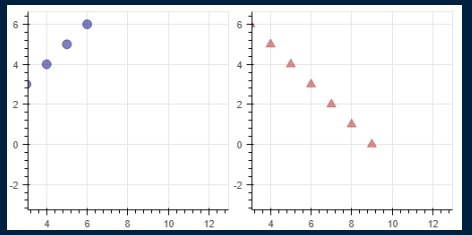
Create a grid plot to visualize the following information in the 2×2 grid:

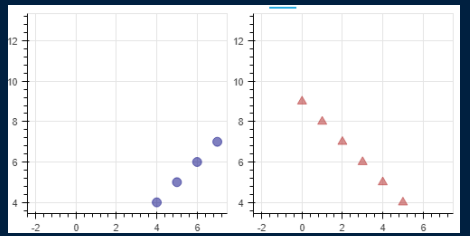
Products	2014	2015	2016	2017	2018
Round steak	16.5	17.5	17.9	18.25	18.58
Sirloin steak	22.14	22.53	22.81	23.52	23.92
Prime rib roast	30.02	30.13	30.23	30.55	30.94
Blade roast	15.51	16.1	16.25	16.76	16.96

Linking Plots

```
from bokeh.layouts import gridplot
from bokeh.plotting import figure, output_file, show
output_file("output.html")
x = list(range(10))
y0 = x
y1 = x[::-1]
s1 = figure(plot_width=250, plot_height=250)
s1.circle(x, y0, size=10, color="navy", alpha=0.5)
s2 = figure(plot_width=250, plot_height=250,
x_range=s1.x_range, y_range=s1.y_range)
s2.triangle(x, y1, size=10, color="firebrick",
alpha=0.5)
p = gridplot([[s1, s2]], toolbar_location=None)
show(p)
```

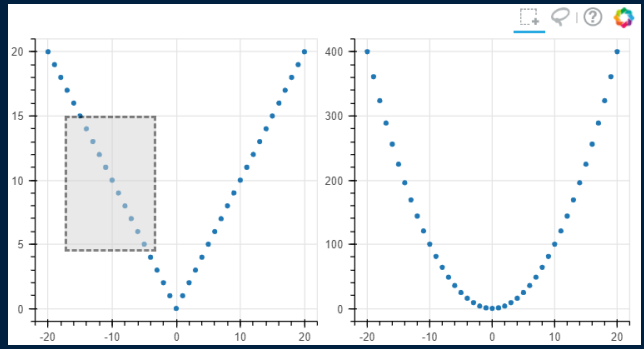


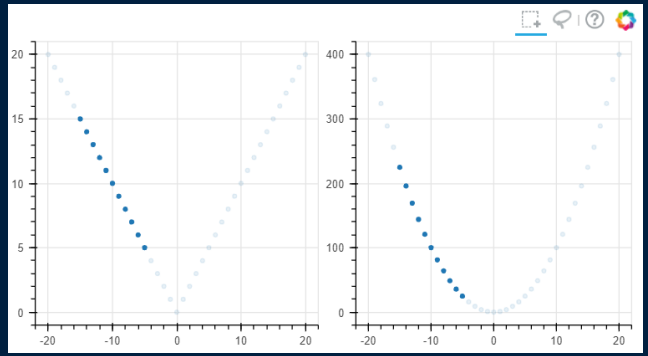




Brushing

```
from bokeh.layouts import gridplot
from bokeh.models import ColumnDataSource
from bokeh.plotting import output_file, show, figure
output_file("output.html")
x = list(range(-20, 21))
y0 = [abs(xx) for xx in x]
y1 = [xx**2 for xx in x]
source = ColumnDataSource(data=dict(x=x, y0=y0, y1=y1))
TOOLS = "box_select,lasso_select,help"
left = figure(tools=TOOLS, plot_width=300, plot_height=300)
left.circle('x', 'y0', source=source)
right = figure(tools=TOOLS, plot_width=300, plot_height=300)
right.circle('x', 'y1', source=source)
p = gridplot([[left, right]])
show(p)
```



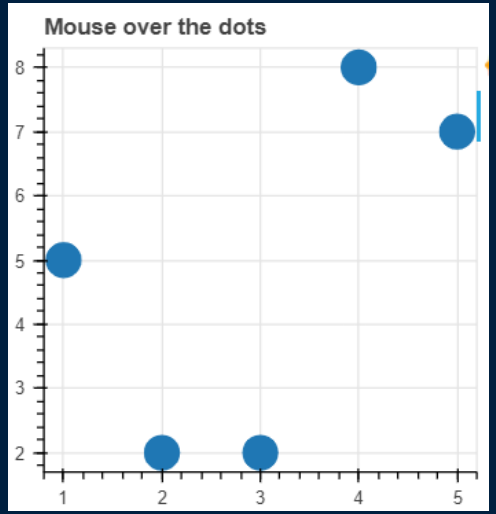


Hover

```
from bokeh.models import HoverTool

source = ColumnDataSource(
        data=dict(
            x=[2, 1, 4, 3, 5],
            y=[2, 5, 8, 2, 7],
            desc=['A', 'b', 'C', 'd', 'E'],
        )
    )
```

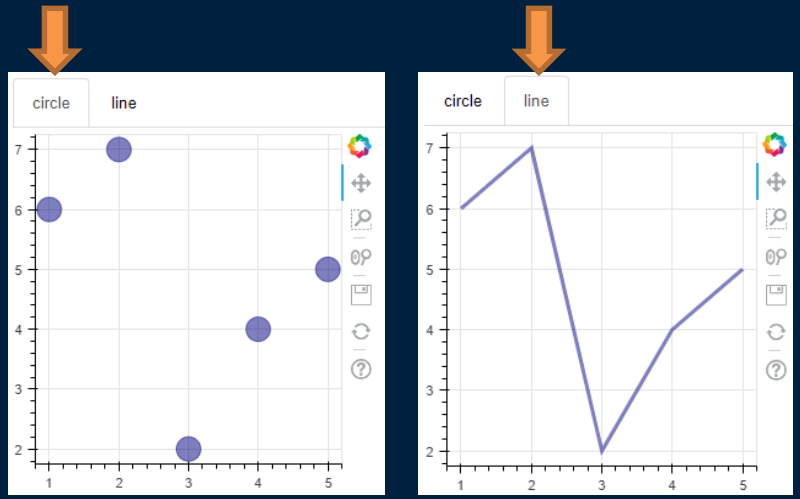
Hover

```
hover = HoverTool(
        tooltips=[
            ("index", "$index"),
            ("(x,y)", "($x, $y)"),
            ("desc", "@desc"),
        ]
    )
p = figure(plot_width=300, plot_height=300, tools=[hover],
title="Mouse over the dots")
p.circle('x', 'y', size=20, source=source)
show(p)
```



Widgets

```
from bokeh.models.widgets import Panel, Tabs
from bokeh.io import output_file, show
from bokeh.plotting import figure
x=[1, 2, 3, 4, 5]
y=[6, 7, 2, 4, 5]
output_file("output.html")
p1 = figure(plot_width=300, plot_height=300)
p1.circle(x, y, size=20, color="navy", alpha=0.5)
tab1 = Panel(child=p1, title="circle")
p2 = figure(plot_width=300, plot_height=300)
p2.line(x, y, line_width=3, color="navy", alpha=0.5)
tab2 = Panel(child=p2, title="line")
tabs = Tabs(tabs=[ tab1, tab2 ])
show(tabs)
```



Google Maps

Bokeh has support for working with Geographical data.

Bokeh can also plot glyphs over a Google Map using the `gmap()` function.

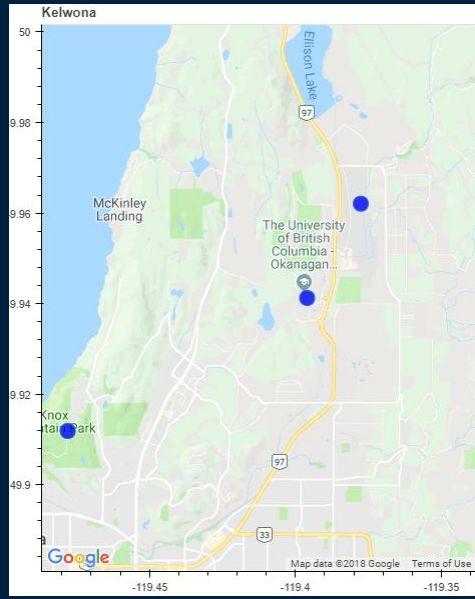
You must pass this function Google API Key in order for it to work, as well as any `GMapOptions` to configure the Google Map underlay.

To get an API key:

https://developers.google.com/maps/documentation/javascript/get-api-key

Google Maps

```
from bokeh.io import output_file, show
from bokeh.models import ColumnDataSource, GMapOptions
from bokeh.plotting import gmap
output_file("gmap.html")
map_options = GMapOptions(lat=49.941251, lng=-119.395981,
map_type="roadmap", zoom=11)
p = gmap("GOOGLE_API_KEY", map_options, title="Kelwona")
source = ColumnDataSource(
    data=dict(lat=[49.941251, 49.962085,  49.911859],
              lon=[-119.395981, -119.377597, -119.477992])
)
p.circle(x="lon", y="lat", size=15, fill_color="blue",
fill_alpha=0.8, source=source)
show(p)
```



Objectives

- Define data visualization and interactive visualization
- List some of the python libraries for interactive visualization
- Use pygal to create interactive visualization with a few lines of code
- Create interactive data visualization using Bokeh
- Learn how to use Bokeh for working with Geographical data (e.g., plotting data on Google map)


THE UNIVERSITY OF BRITISH COLUMBIA