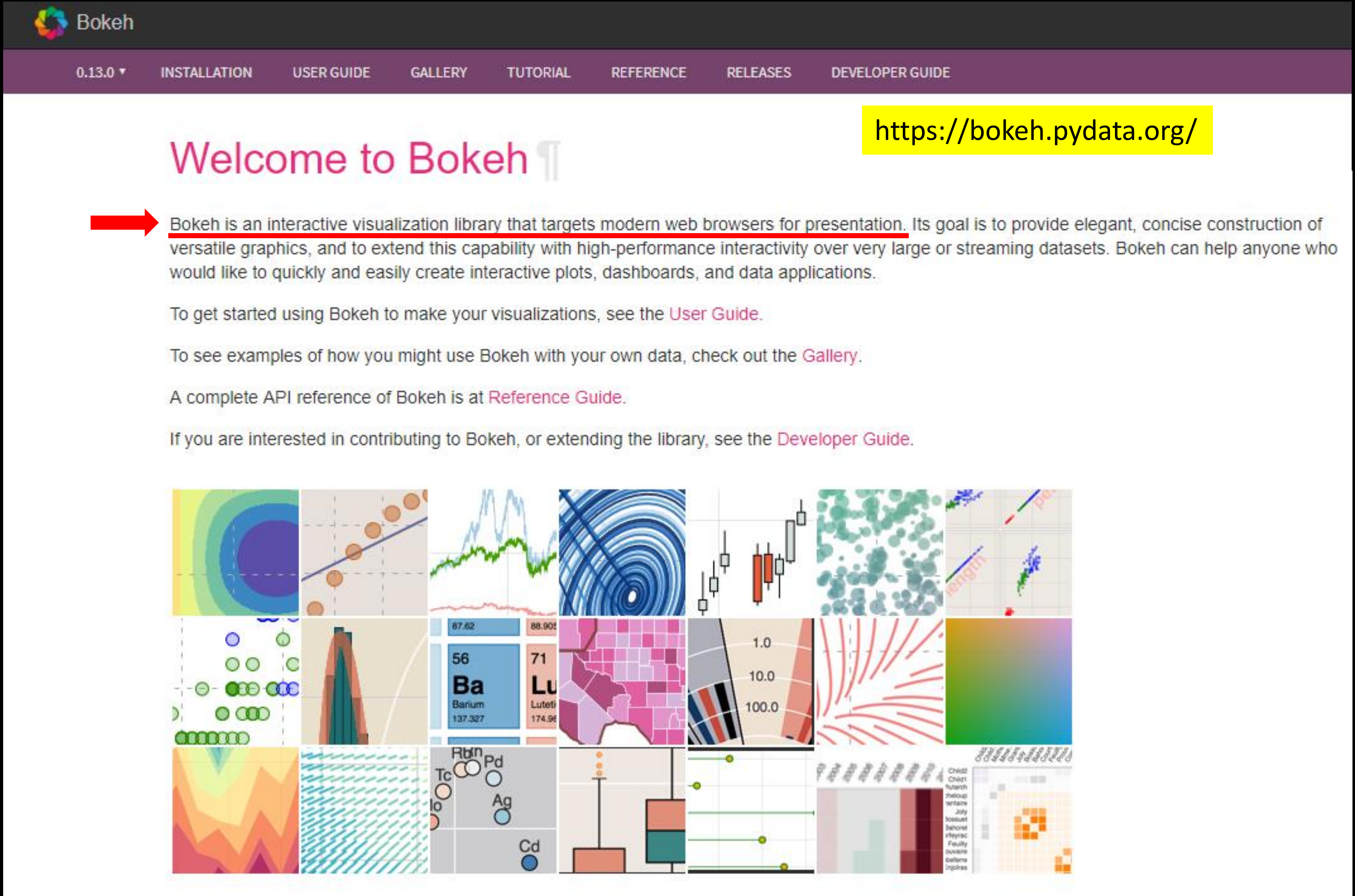
Tutoriais | Intro

Bokeh

python

08 / 10 (Hoje)



https://bokeh.pydata.org/

Welcome to Bokeh ¶

Bokeh is an interactive visualization library that targets modern web browsers for presentation. Its goal is to provide elegant, concise construction of versatile graphics, and to extend this capability with high-performance interactivity over very large or streaming datasets. Bokeh can help anyone who would like to quickly and easily create interactive plots, dashboards, and data applications.

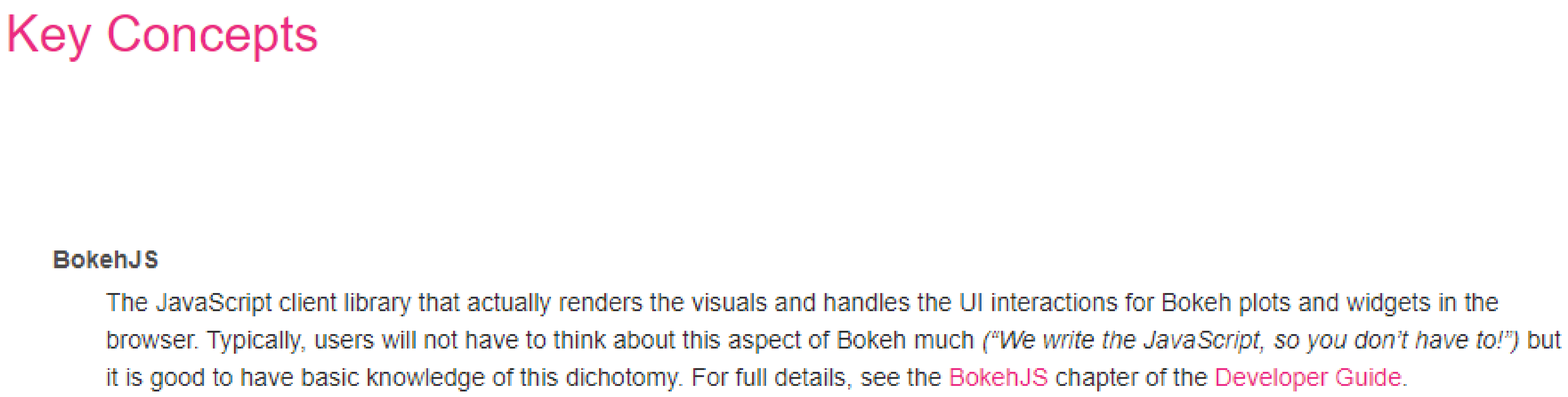
To get started using Bokeh to make your visualizations, see the User Guide.

To see examples of how you might use Bokeh with your own data, check out the Gallery.

A complete API reference of Bokeh is at Reference Guide.

If you are interested in contributing to Bokeh, or extending the library, see the Developer Guide.

https://bokeh.pydata.org/en/latest/docs/user_guide/concepts.html#

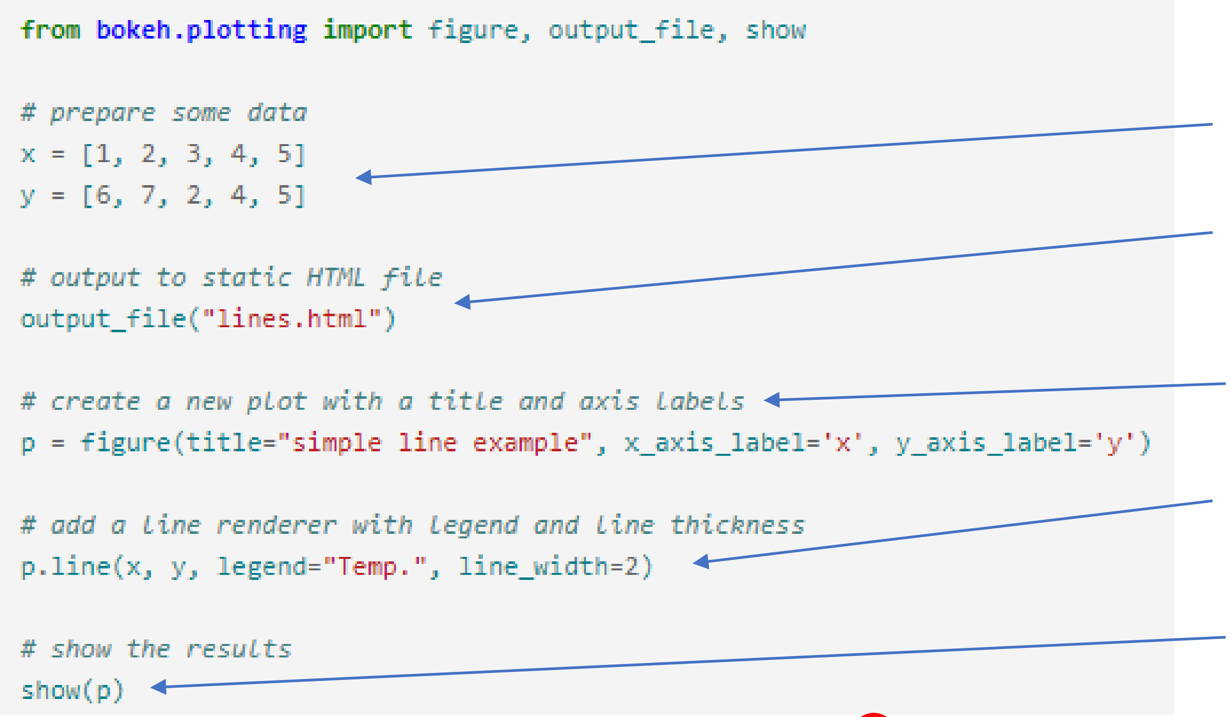


Key Concepts

BokehJS

The JavaScript client library that actually renders the visuals and handles the UI interactions for Bokeh plots and widgets in the browser. Typically, users will not have to think about this aspect of Bokeh much *(“We write the JavaScript, so you don’t have to!”)* but it is good to have basic knowledge of this dichotomy. For full details, see the BokehJS chapter of the Developer Guide.

“We write the JavaScript, so you don’t have to!” 😉



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

# prepare some data
x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]

# output to static HTML file
output_file("lines.html")

# create a new plot with a title and axis labels
p = figure(title="simple line example", x_axis_label='x', y_axis_label='y')

# add a line renderer with legend and line thickness
p.line(x, y, legend="Temp.", line_width=2)

# show the results
show(p)
```

The basic steps to creating plots with the bokeh.plotting interface are:

Prepare some data

In this case plain python lists, but could also be NumPy arrays or Pandas series.

Tell Bokeh where to generate output

In this case using `output_file()`, with the filename `"lines.html"`. Another option is `output_notebook()` for use in Jupyter notebooks.

Call `figure()`

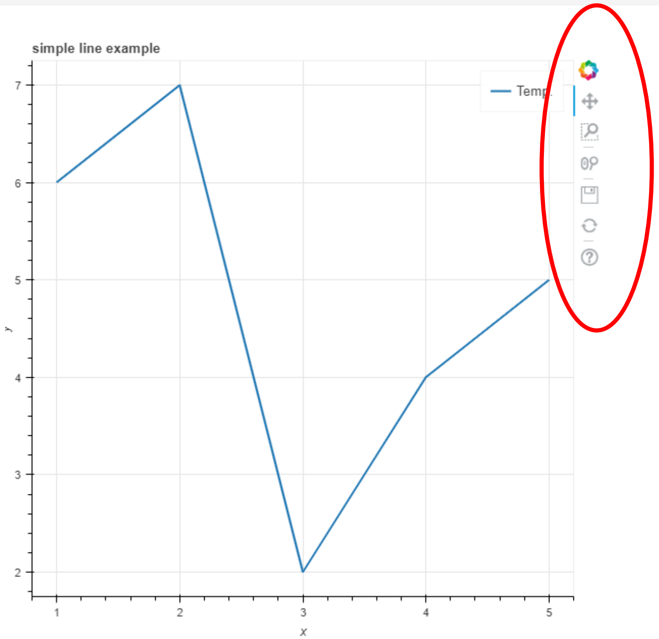
This creates a plot with typical default options and easy customization of title, tools, and axes labels.

Add renderers

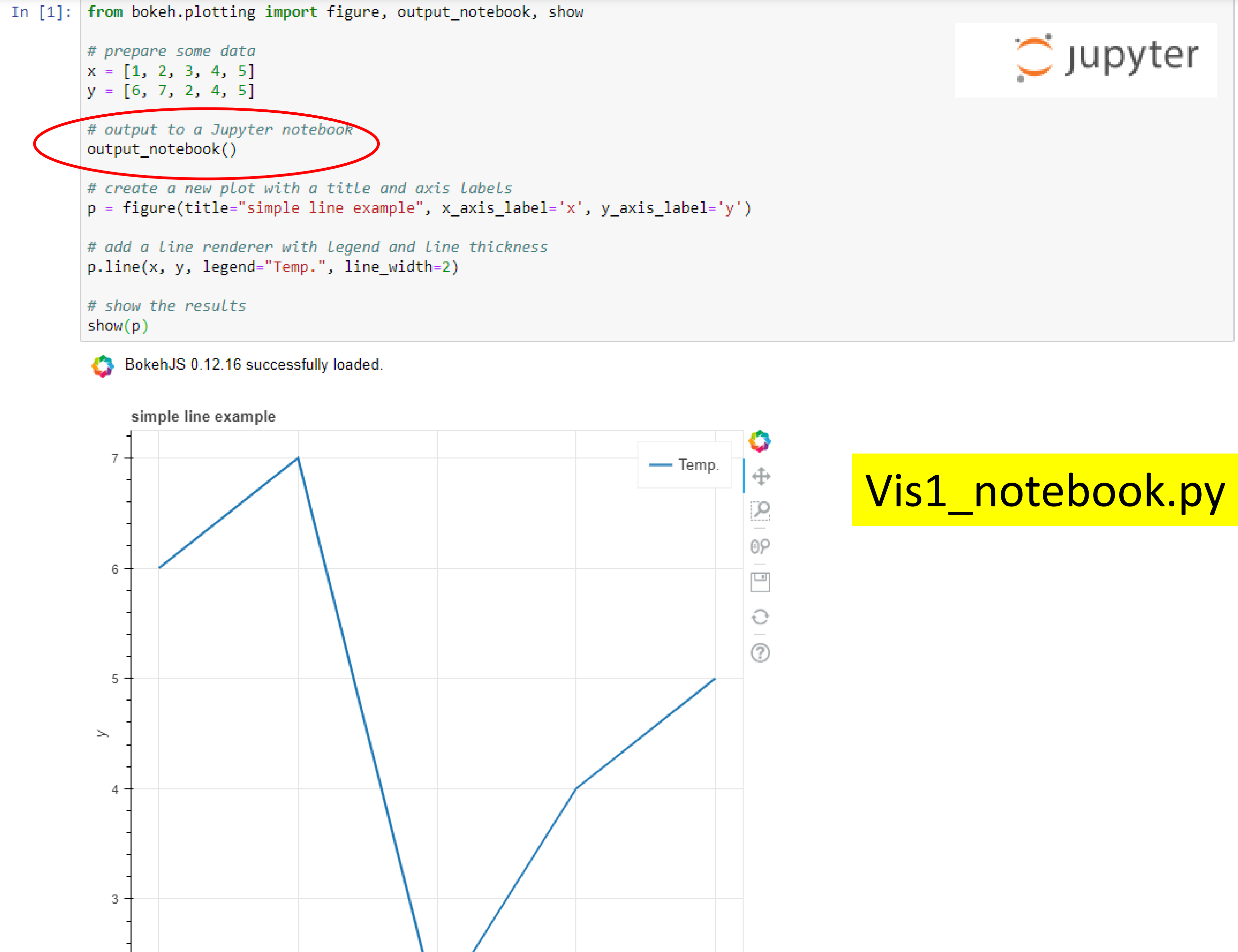
In this case, we use `line()` for our data, specifying visual customizations like colors, legends and widths.

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

These functions save the plot to an HTML file and optionally display it in a browser.



vis1.py

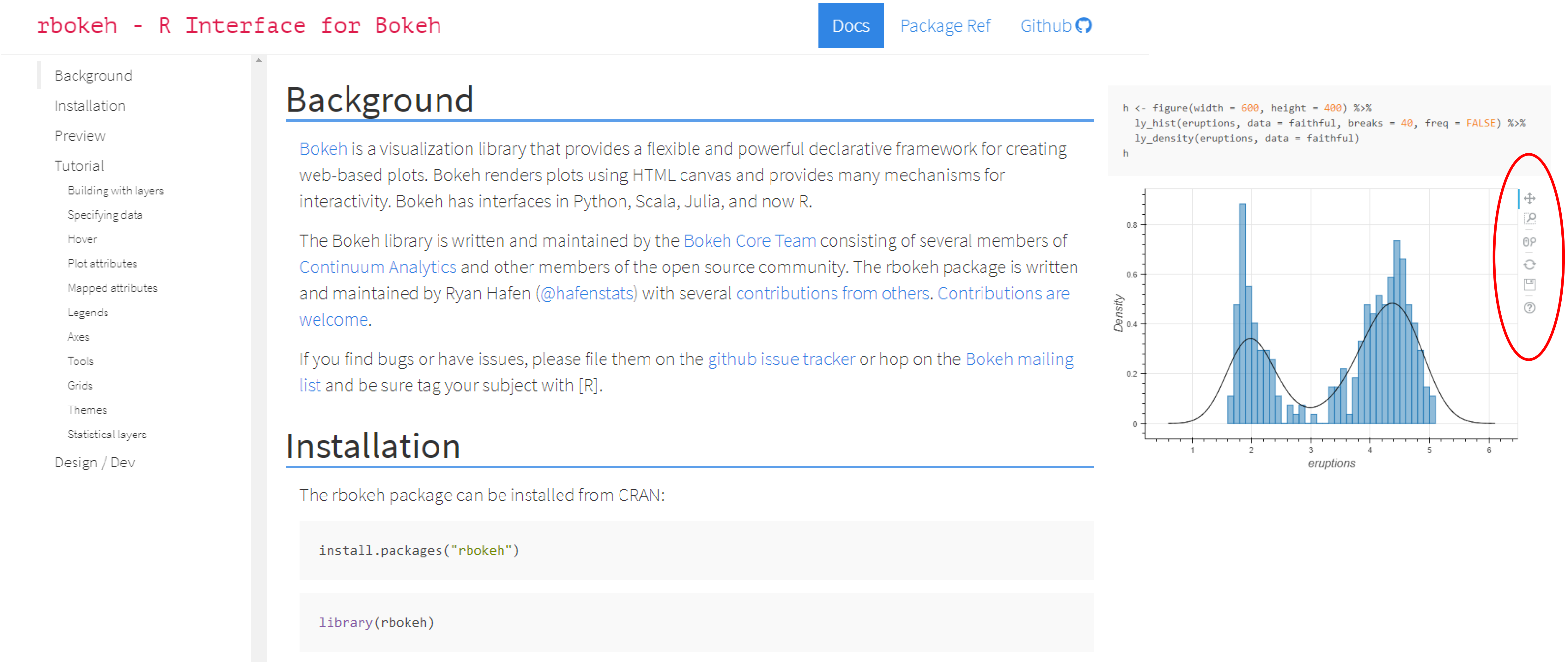


```
In [1]: from bokeh.plotting import figure, output_notebook, show

        # prepare some data
        x = [1, 2, 3, 4, 5]
        y = [6, 7, 2, 4, 5]

        # output to a Jupyter notebook
        output_notebook()

        # create a new plot with a title and axis labels
        p = figure(title="simple line example", x_axis_label='x', y_axis_label='y')

        # add a line renderer with legend and line thickness
        p.line(x, y, legend="Temp.", line_width=2)

        # show the results
        show(p)
```

```
BokehJS 0.12.16 successfully loaded.
```

Vis1_notebook.py

http://hafen.github.io/rbokeh/index.html



rbokeh - R Interface for Bokeh

Docs | Package Ref | Github

- Background
- Installation
- Preview
- Tutorial
  - Building with layers
  - Specifying data
  - Hover
  - Plot attributes
  - Mapped attributes
  - Legends
  - Axes
  - Tools
  - Grids
  - Themes
  - Statistical layers
- Design / Dev

Background

Bokeh is a visualization library that provides a flexible and powerful declarative framework for creating web-based plots. Bokeh renders plots using HTML canvas and provides many mechanisms for interactivity. Bokeh has interfaces in Python, Scala, Julia, and now R.

The Bokeh library is written and maintained by the Bokeh Core Team consisting of several members of Continuum Analytics and other members of the open source community. The rbokeh package is written and maintained by Ryan Hafen (@hafenstats) with several contributions from others. Contributions are welcome.

If you find bugs or have issues, please file them on the github issue tracker or hop on the Bokeh mailing list and be sure tag your subject with [R].

Installation

The rbokeh package can be installed from CRAN:

```
install.packages("rbokeh")
```

```
library(rbokeh)
```

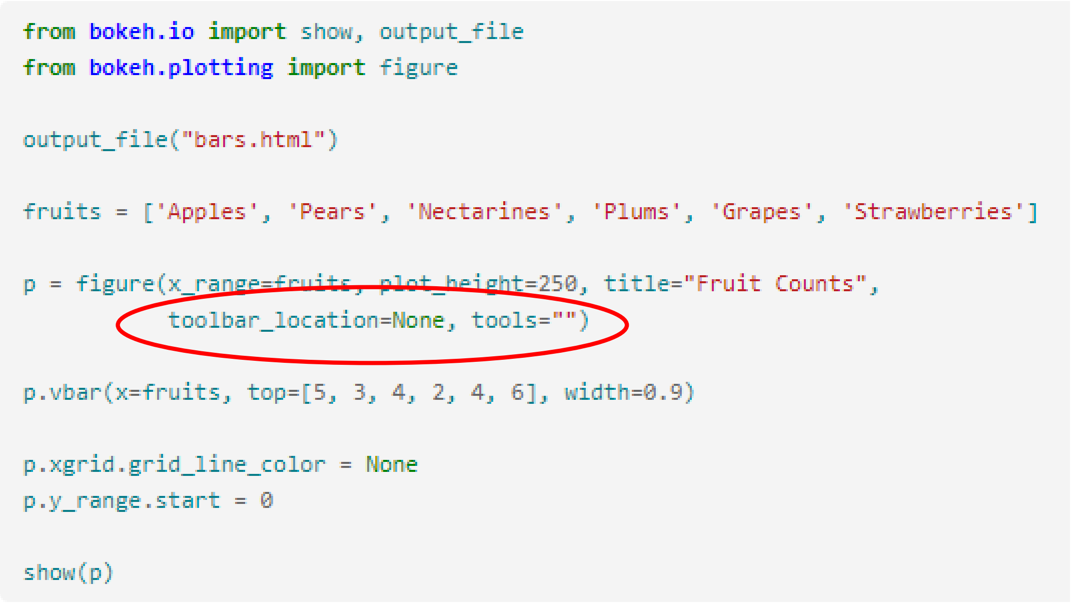
```
h <- figure(width = 600, height = 400) %>%
  ly_hist(eruptions, data = faithful, breaks = 40, freq = FALSE) %>%
  ly_density(eruptions, data = faithful)
h
```

Handling Categorical Data

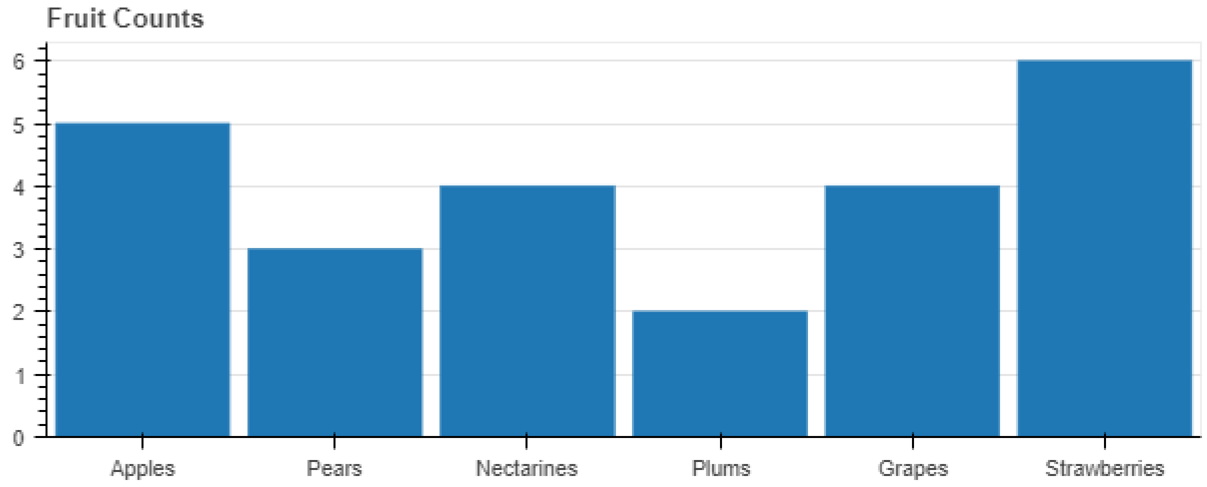
```
fruits = ['Apples', 'Pears', 'Nectarines', 'Plums', 'Grapes', 'Strawberries']
```

To inform Bokeh that the x-axis is categorical, we pass this list of factors as the `x_range` argument

```
p = figure(x_range=fruits, ... )
```

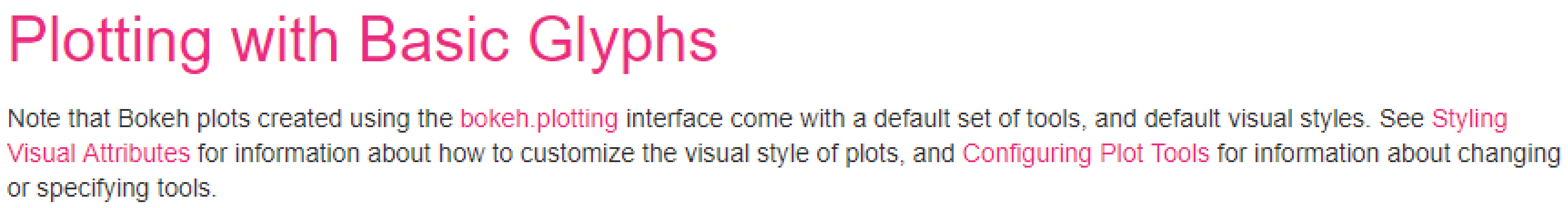


```
from bokeh.io import show, output_file
from bokeh.plotting import figure

output_file("bars.html")

fruits = ['Apples', 'Pears', 'Nectarines', 'Plums', 'Grapes', 'Strawberries']

p = figure(x_range=fruits, plot_height=250, title="Fruit Counts",
           toolbar_location=None, tools="")

p.vbar(x=fruits, top=[5, 3, 4, 2, 4, 6], width=0.9)

p.xgrid.grid_line_color = None
p.y_range.start = 0

show(p)
```



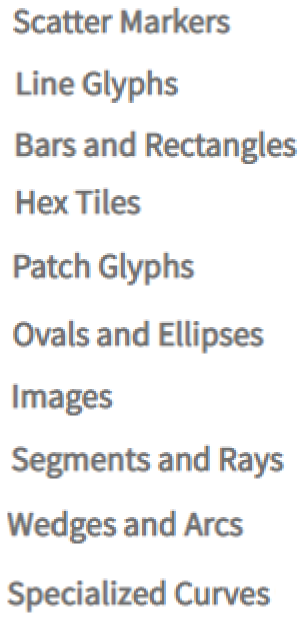
vis2.py

https://bokeh.pydata.org/en/latest/docs/user_guide/plotting.html#userguide-plotting

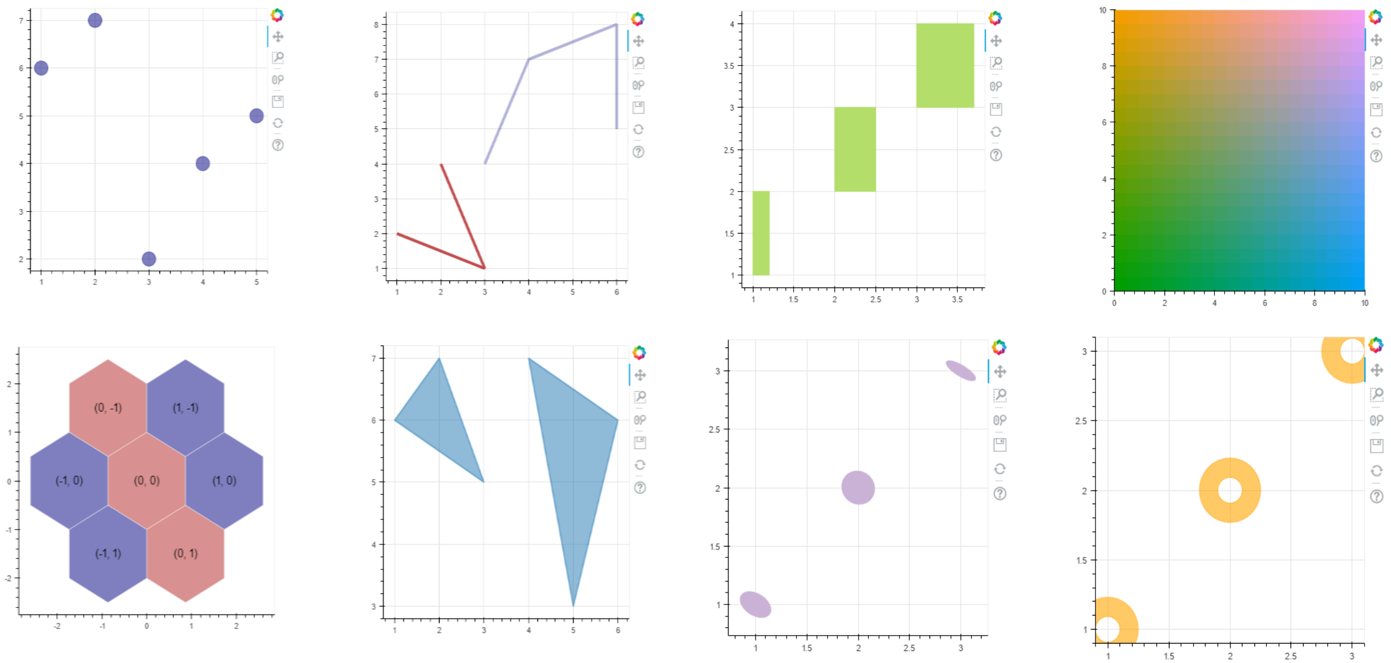


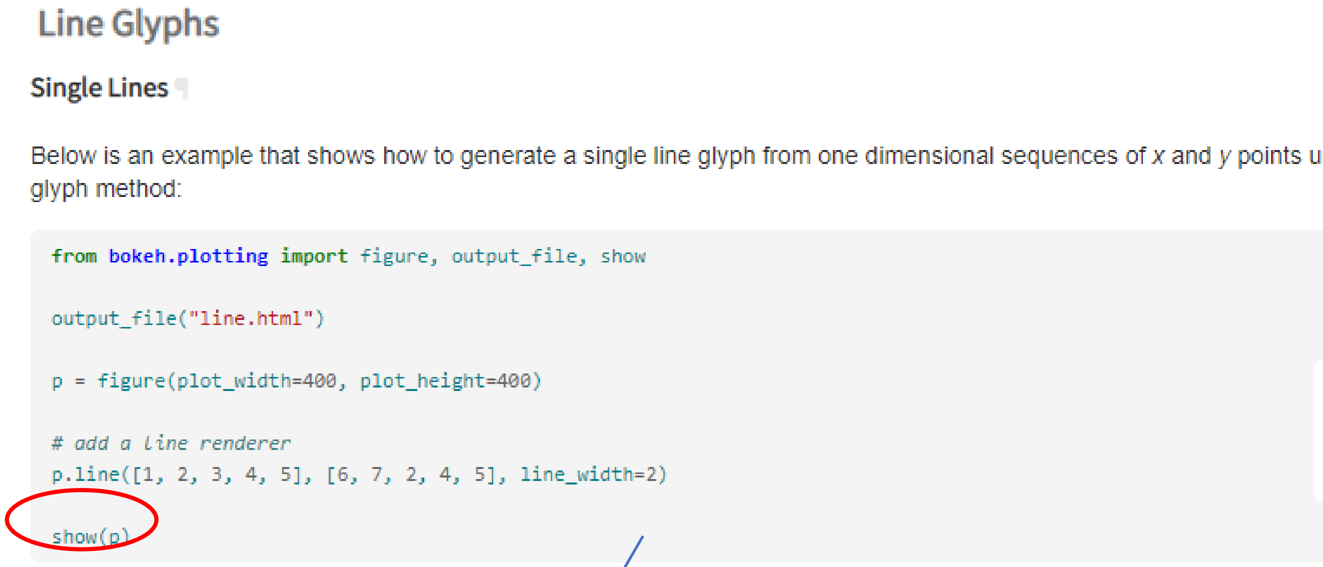
Plotting with Basic Glyphs

Note that Bokeh plots created using the bokeh.plotting interface come with a default set of tools, and default visual styles. See Styling Visual Attributes for information about how to customize the visual style of plots, and Configuring Plot Tools for information about changing or specifying tools.



- Scatter Markers
- Line Glyphs
- Bars and Rectangles
- Hex Tiles
- Patch Glyphs
- Ovals and Ellipses
- Images
- Segments and Rays
- Wedges and Arcs
- Specialized Curves



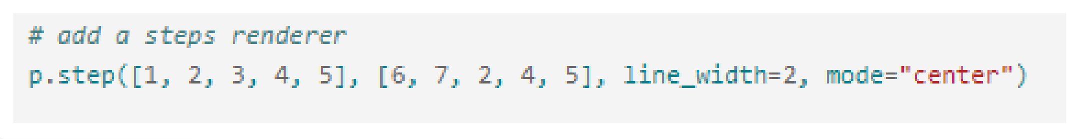


Line Glyphs

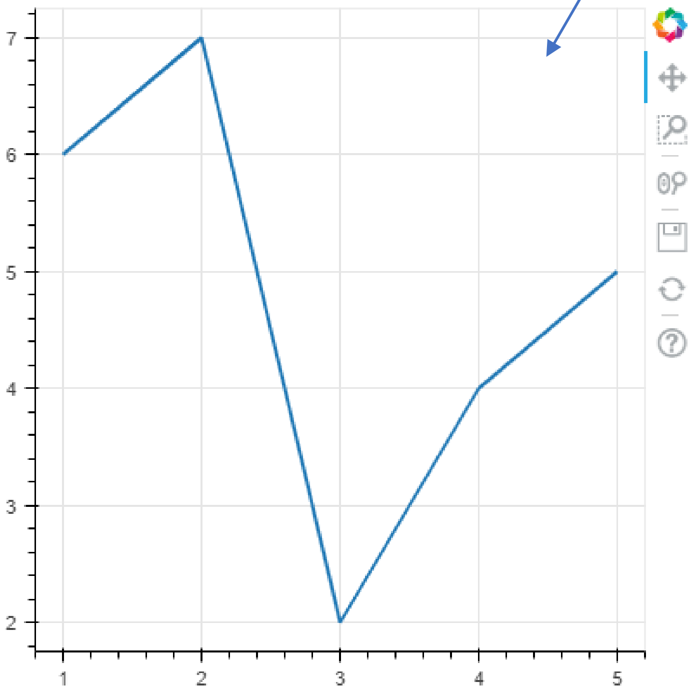
Single Lines

Below is an example that shows how to generate a single line glyph from one dimensional sequences of *x* and *y* points using the `line()` glyph method:

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

output_file("line.html")

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

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

show(p)
```

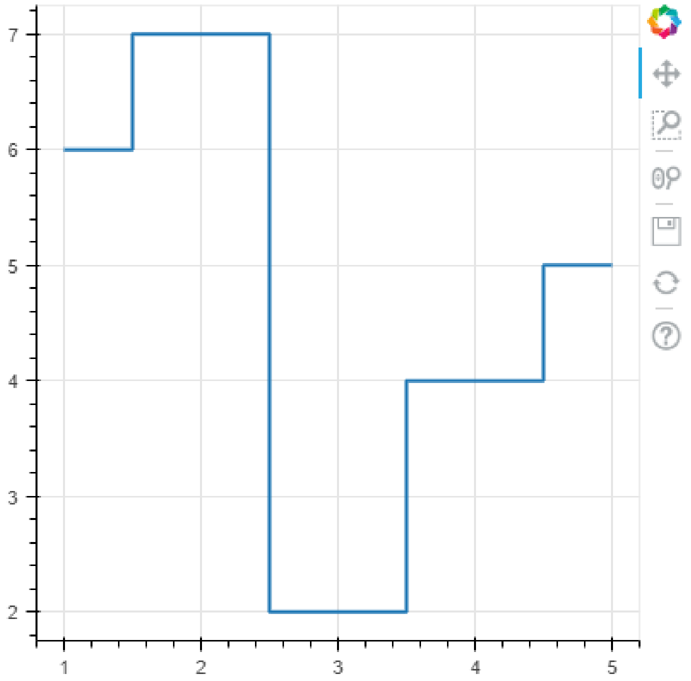


```
# add a steps renderer
p.step([1, 2, 3, 4, 5], [6, 7, 2, 4, 5], line_width=2, mode="center")
```



vis3.py

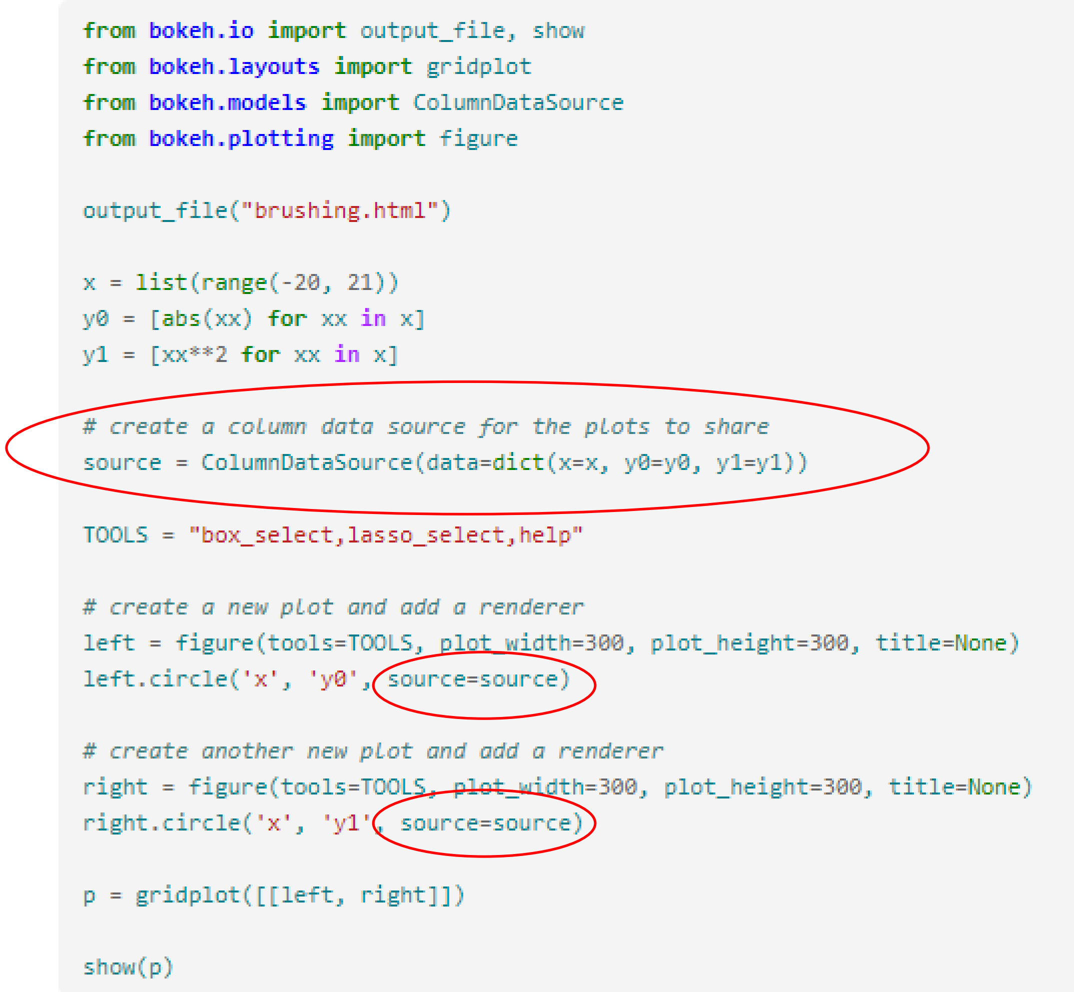
vis4.py



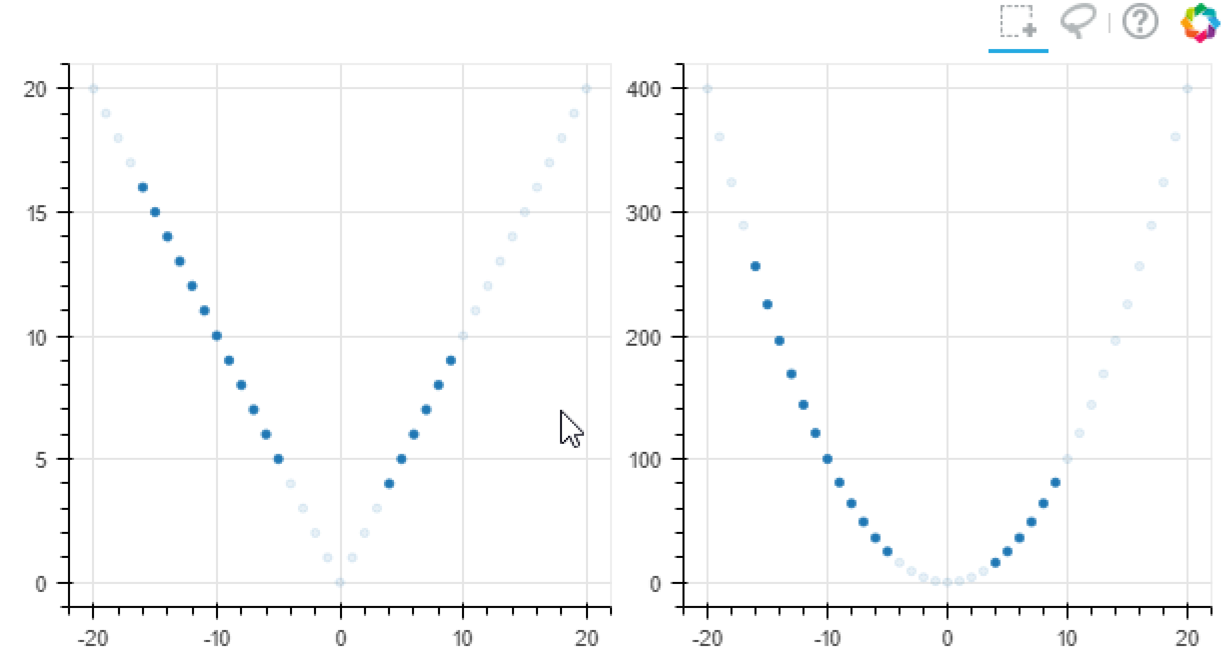
https://bokeh.pydata.org/en/latest/docs/user_guide/interaction/linking.html

Linked Brushing

Linked brushing in Bokeh is expressed by sharing data sources between glyph renderers. This is all Bokeh needs to understand that selections acted on one glyph must pass to all other glyphs that share that same source.



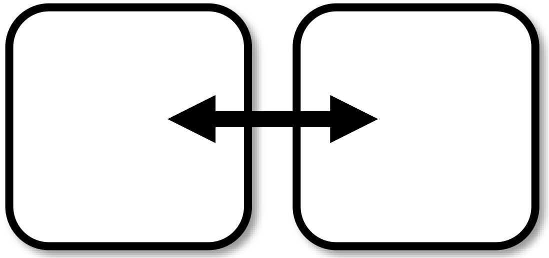
```
from bokeh.io import output_file, show
from bokeh.layouts import gridplot
from bokeh.models import ColumnDataSource
from bokeh.plotting import figure

output_file("brushing.html")

x = list(range(-20, 21))
y0 = [abs(xx) for xx in x]
y1 = [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, plot_width=300, plot_height=300, title=None)
left.circle('x', 'y0', source=source)

# create another new plot and add a renderer
right = figure(tools=TOOLS, plot_width=300, plot_height=300, title=None)
right.circle('x', 'y1', source=source)

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

show(p)
```

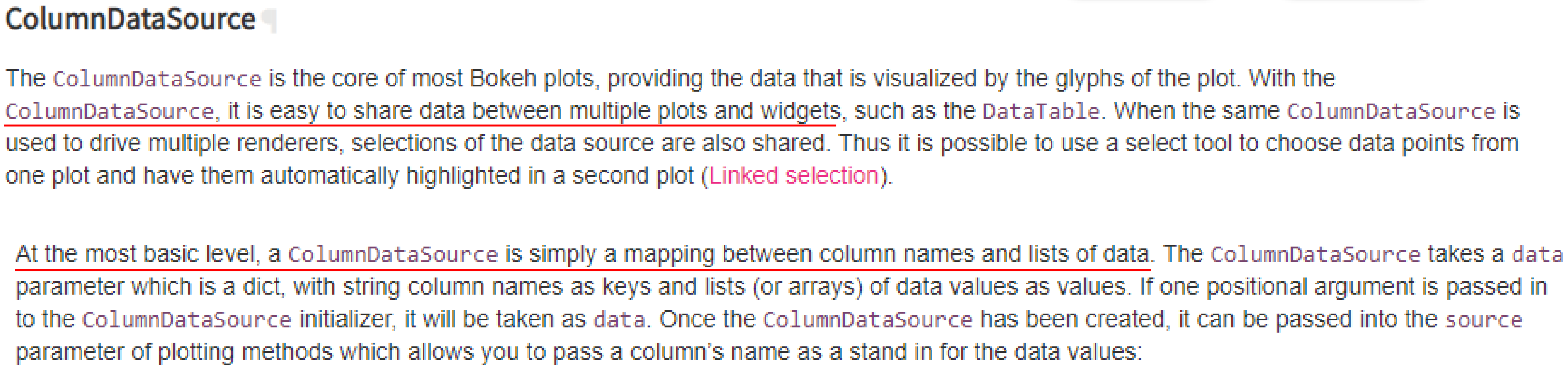


brushing.py

https://bokeh.pydata.org/en/latest/docs/user_guide/data.html

Linked visualizations





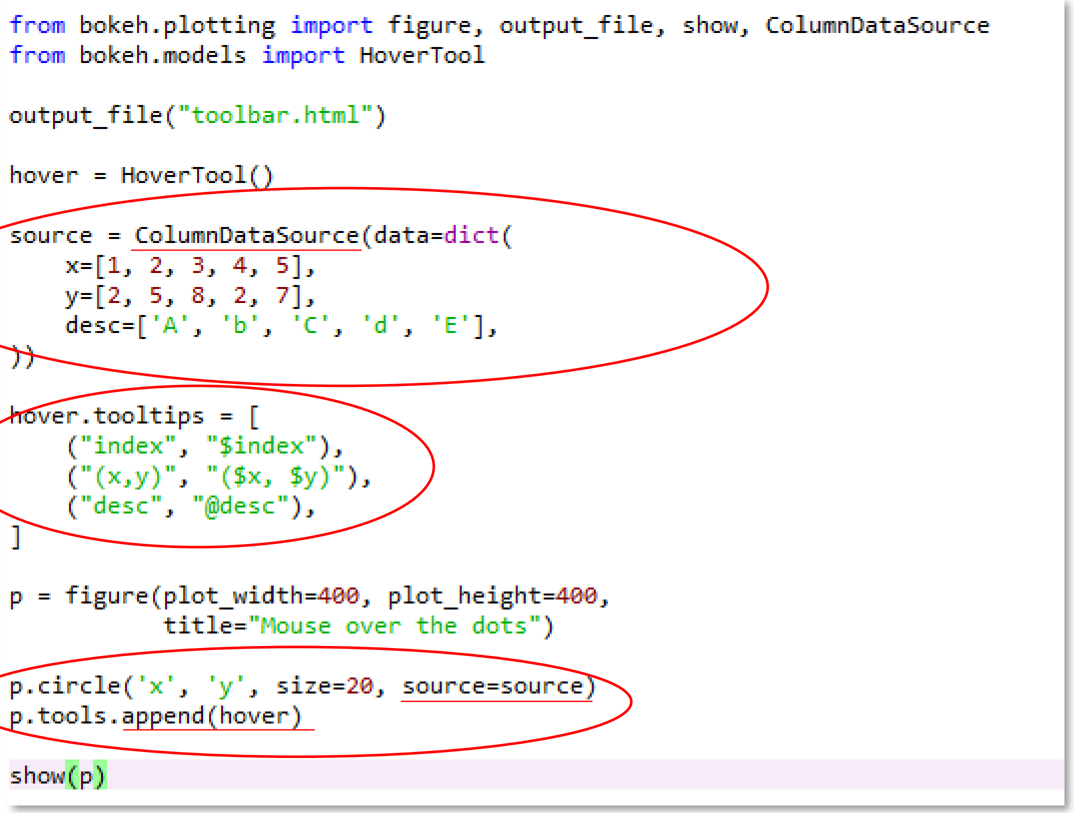
ColumnDataSource¶

The `ColumnDataSource` is the core of most Bokeh plots, providing the data that is visualized by the glyphs of the plot. With the `ColumnDataSource`, it is easy to share data between multiple plots and widgets, such as the `DataTable`. When the same `ColumnDataSource` is used to drive multiple renderers, selections of the data source are also shared. Thus it is possible to use a select tool to choose data points from one plot and have them automatically highlighted in a second plot (Linked selection).

At the most basic level, a `ColumnDataSource` is simply a mapping between column names and lists of data. The `ColumnDataSource` takes a `data` parameter which is a dict, with string column names as keys and lists (or arrays) of data values as values. If one positional argument is passed in to the `ColumnDataSource` initializer, it will be taken as `data`. Once the `ColumnDataSource` has been created, it can be passed into the `source` parameter of plotting methods which allows you to pass a column's name as a stand in for the data values:

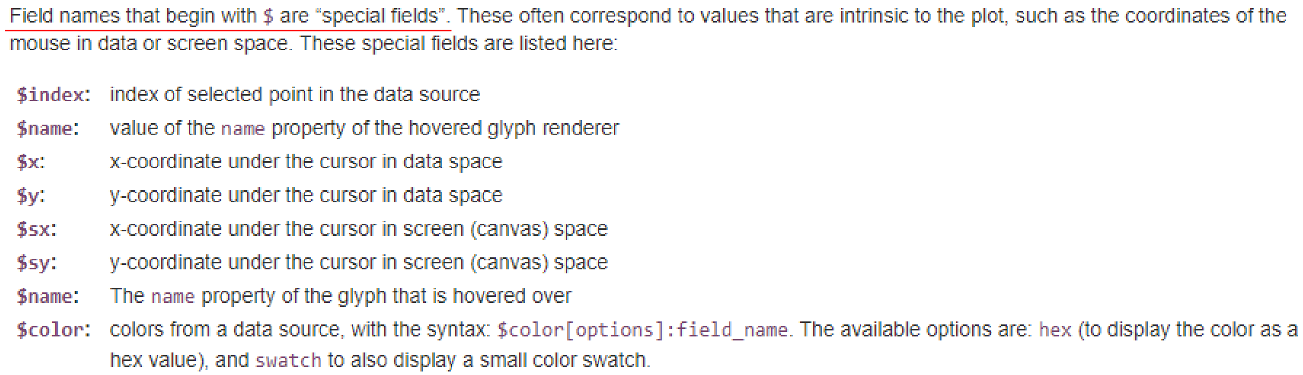
Basic Tooltips

https://bokeh.pydata.org/en/latest/docs/user_guide/tools.html#basic-tooltips



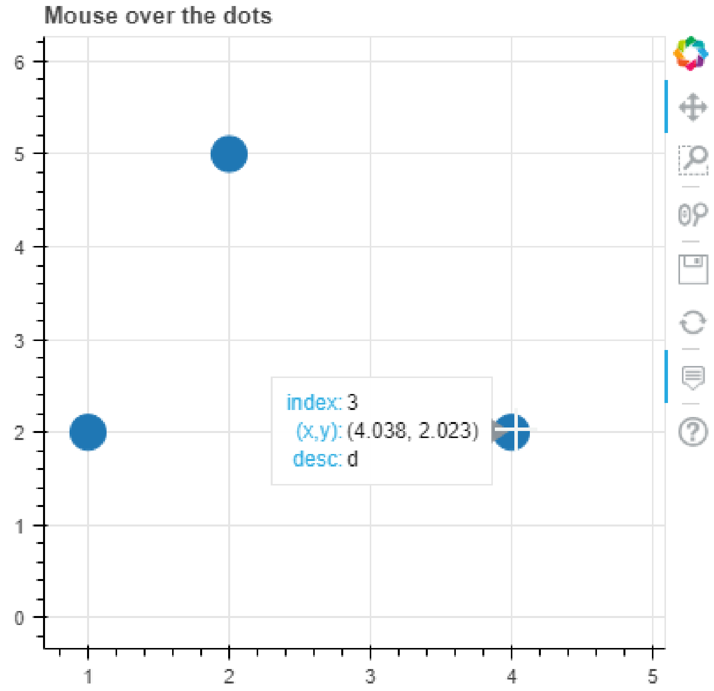
```
from bokeh.plotting import figure, output_file, show, ColumnDataSource
from bokeh.models import HoverTool

output_file("toolbar.html")

hover = HoverTool()

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

hover.tooltips = [
    ("index", "$index"),
    ("(x,y)", "($x, $y)"),
    ("desc", "@desc"),
]

p = figure(plot_width=400, plot_height=400,
           title="Mouse over the dots")

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

show(p)
```

tooltip.py

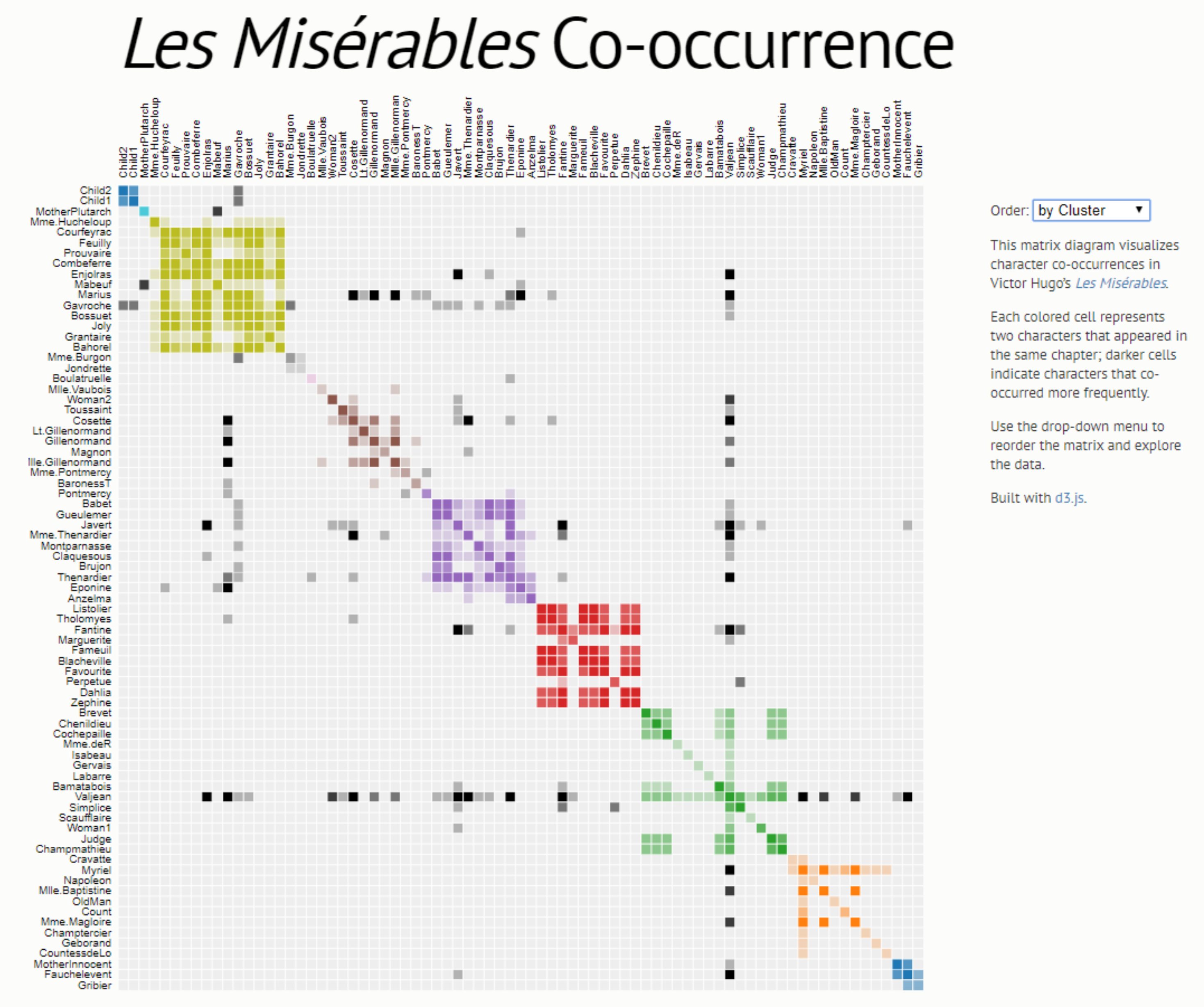


Field names that begin with $ are "special fields". These often correspond to values that are intrinsic to the plot, such as the coordinates of the mouse in data or screen space. These special fields are listed here:

`$index`:	index of selected point in the data source
`$name`:	value of the `name` property of the hovered glyph renderer
`$x`:	x-coordinate under the cursor in data space
`$y`:	y-coordinate under the cursor in data space
`$sx`:	x-coordinate under the cursor in screen (canvas) space
`$sy`:	y-coordinate under the cursor in screen (canvas) space
`$name`:	The `name` property of the glyph that is hovered over
`$color`:	colors from a data source, with the syntax: `$color[options]:field_name`. The available options are: `hex` (to display the color as a hex value), and `swatch` to also display a small color swatch.



https://bost.ocks.org/mike/miserables/



Les Misérables Co-occurrence

Order: by Cluster ▾

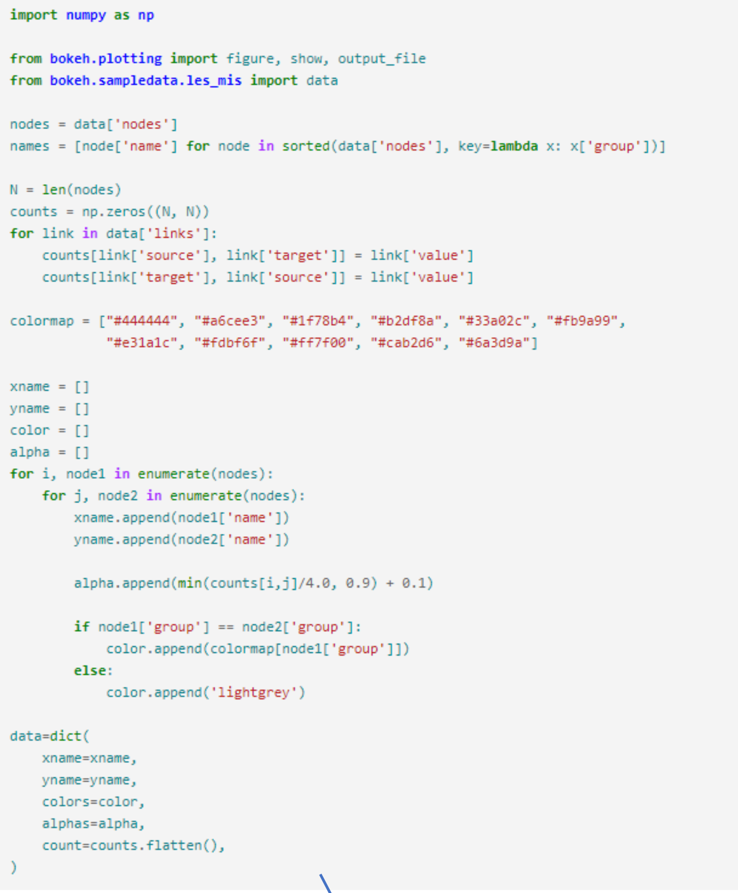
This matrix diagram visualizes character co-occurrences in Victor Hugo's *Les Misérables*.

Each colored cell represents two characters that appeared in the same chapter; darker cells indicate characters that co-occurred more frequently.

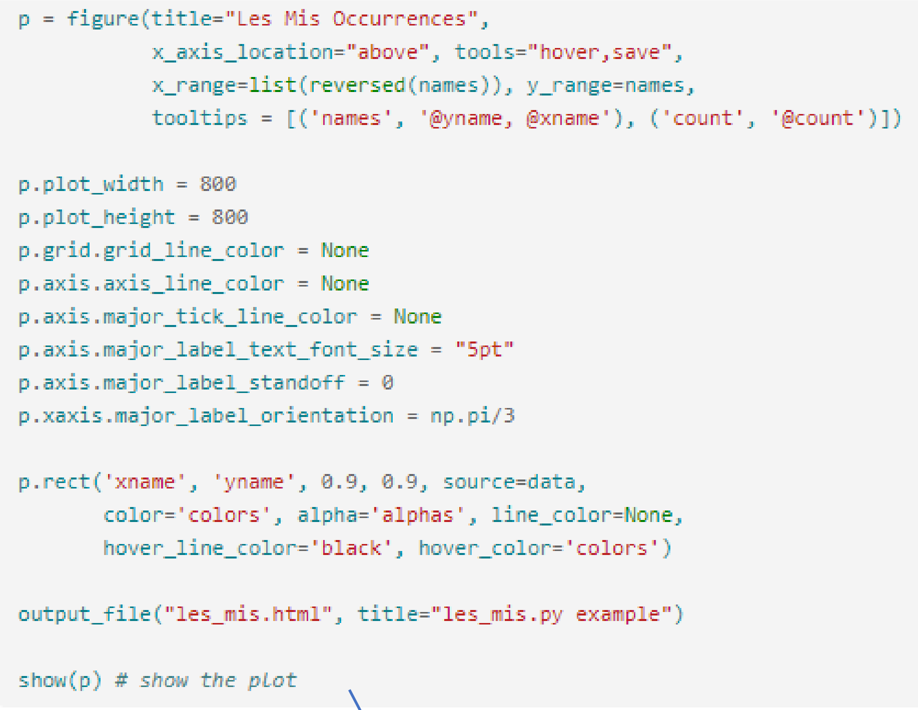
Use the drop-down menu to reorder the matrix and explore the data.

Built with d3.js.

https://bokeh.pydata.org/en/latest/docs/gallery/les_mis.html

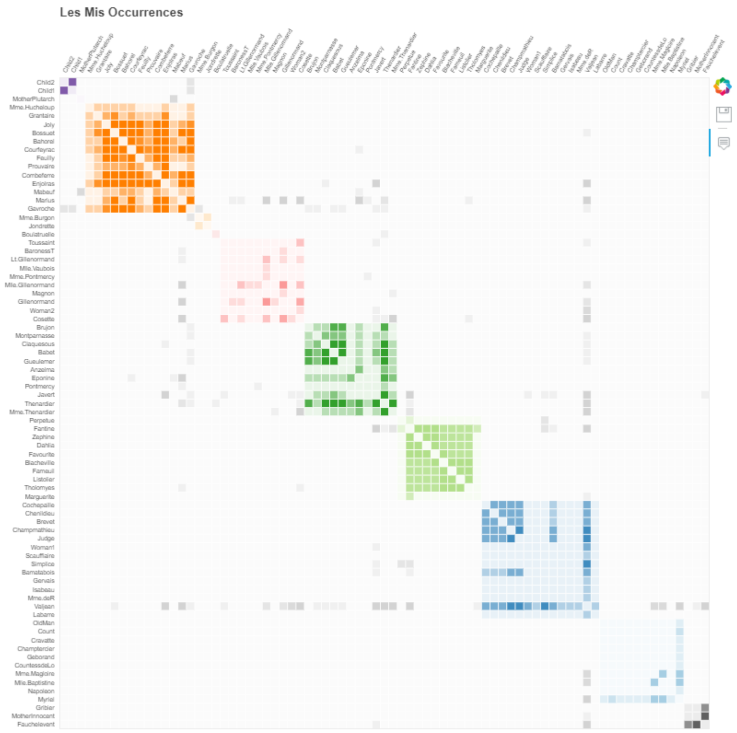


```
import numpy as np

from bokeh.plotting import figure, show, output_file
from bokeh.sampledata.les_mis import data

nodes = data['nodes']
names = [node['name'] for node in sorted(data['nodes'], key=lambda x: x['group'])]

N = len(nodes)
counts = np.zeros((N, N))
for link in data['links']:
    counts[link['source'], link['target']] = link['value']
    counts[link['target'], link['source']] = link['value']

colormap = ["#444444", "#a6cee3", "#1f78b4", "#b2df8a", "#33a02c", "#fb9a99",
            "#e31a1c", "#fdbf6f", "#ff7f00", "#cab2d6", "#6a3d9a"]

xname = []
yname = []
color = []
alpha = []
for i, node1 in enumerate(nodes):
    for j, node2 in enumerate(nodes):
        xname.append(node1['name'])
        yname.append(node2['name'])

        alpha.append(min(counts[i,j]/4.0, 0.9) + 0.1)

        if node1['group'] == node2['group']:
            color.append(colormap[node1['group']])
        else:
            color.append('lightgrey')

data=dict(
    xname=xname,
    yname=yname,
    colors=color,
    alphas=alpha,
    count=counts.flatten(),
)
```



```
p = figure(title="Les Mis Occurrences",
           x_axis_location="above", tools="hover,save",
           x_range=list(reversed(names)), y_range=names,
           tooltips = [('names', '@yname, @xname'), ('count', '@count')])

p.plot_width = 800
p.plot_height = 800
p.grid.grid_line_color = None
p.axis.axis_line_color = None
p.axis.major_tick_line_color = None
p.axis.major_label_text_font_size = "5pt"
p.axis.major_label_standoff = 0
p.xaxis.major_label_orientation = np.pi/3

p.rect('xname', 'yname', 0.9, 0.9, source=data,
       color='colors', alpha='alphas', line_color=None,
       hover_line_color='black', hover_color='colors')

output_file("les_mis.html", title="les_mis.py example")

show(p) # show the plot
```

les_mis.py¶



Plot

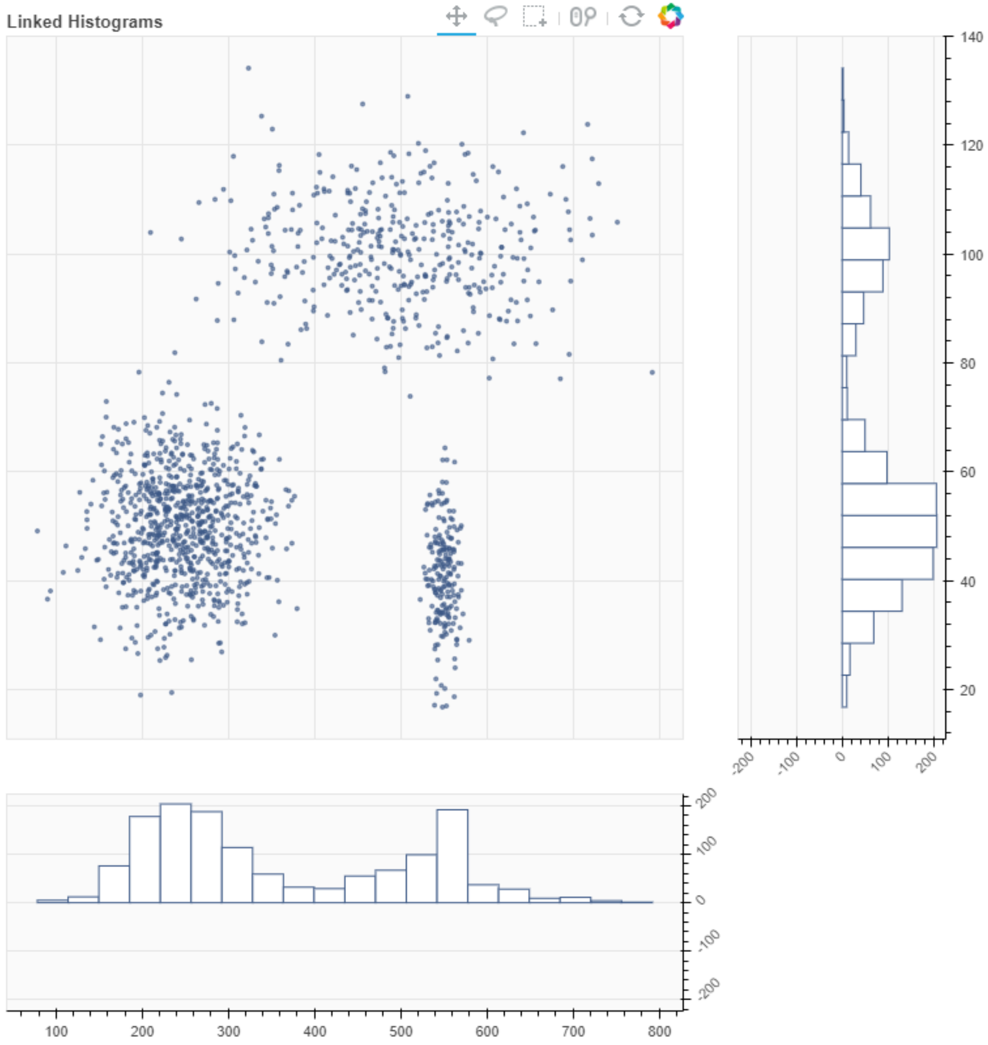
Preparação dos dados

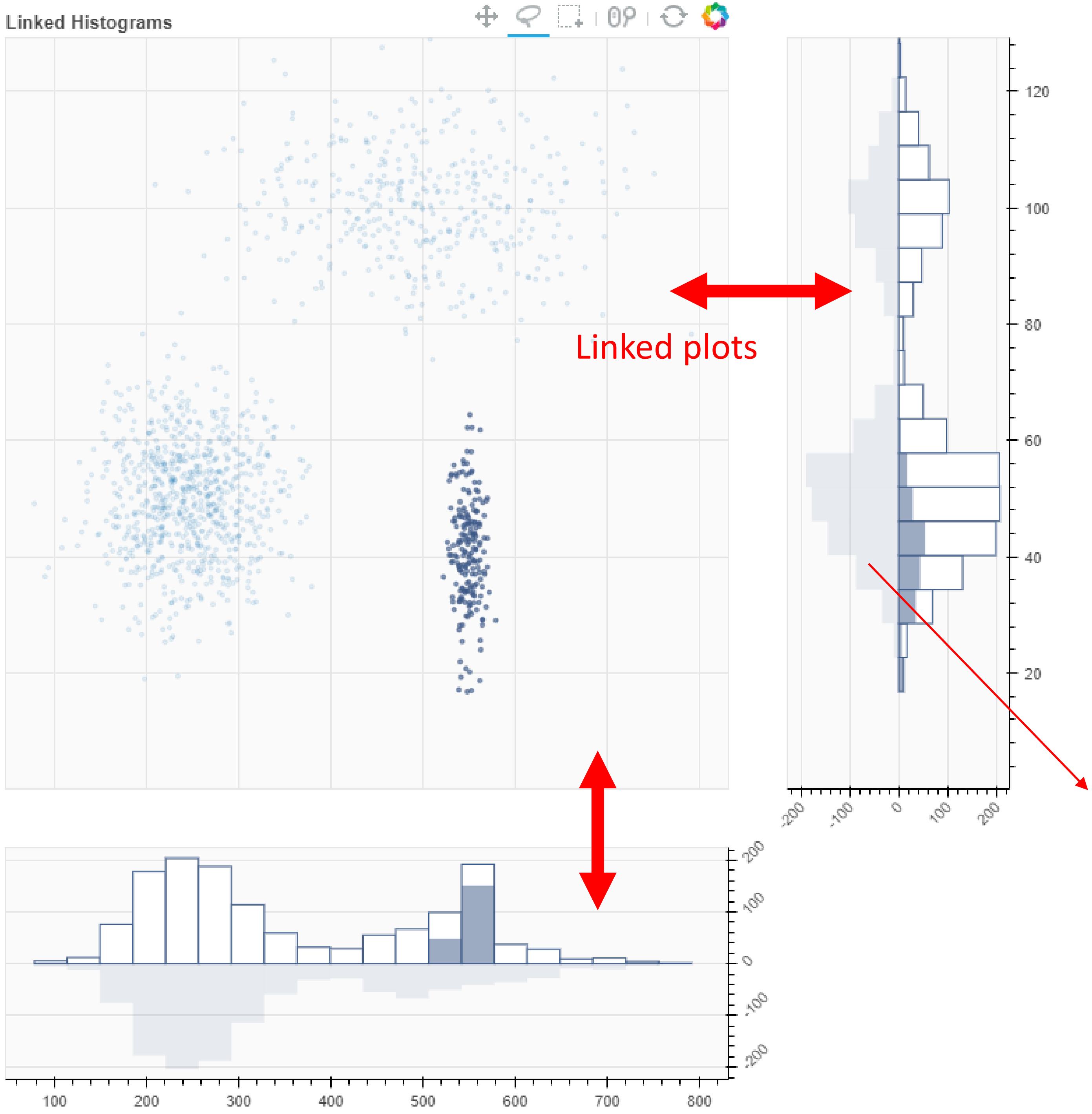
https://demo.bokehplots.com/

Bokeh Applications

This site hosts examples of applications built using Bokeh, a library for building data visualizations and applications in the browser from Python (and other languages), without writing JavaScript.

Exemplo 1
https://demo.bokehplots.com/apps/selection_histogram





Linked plots

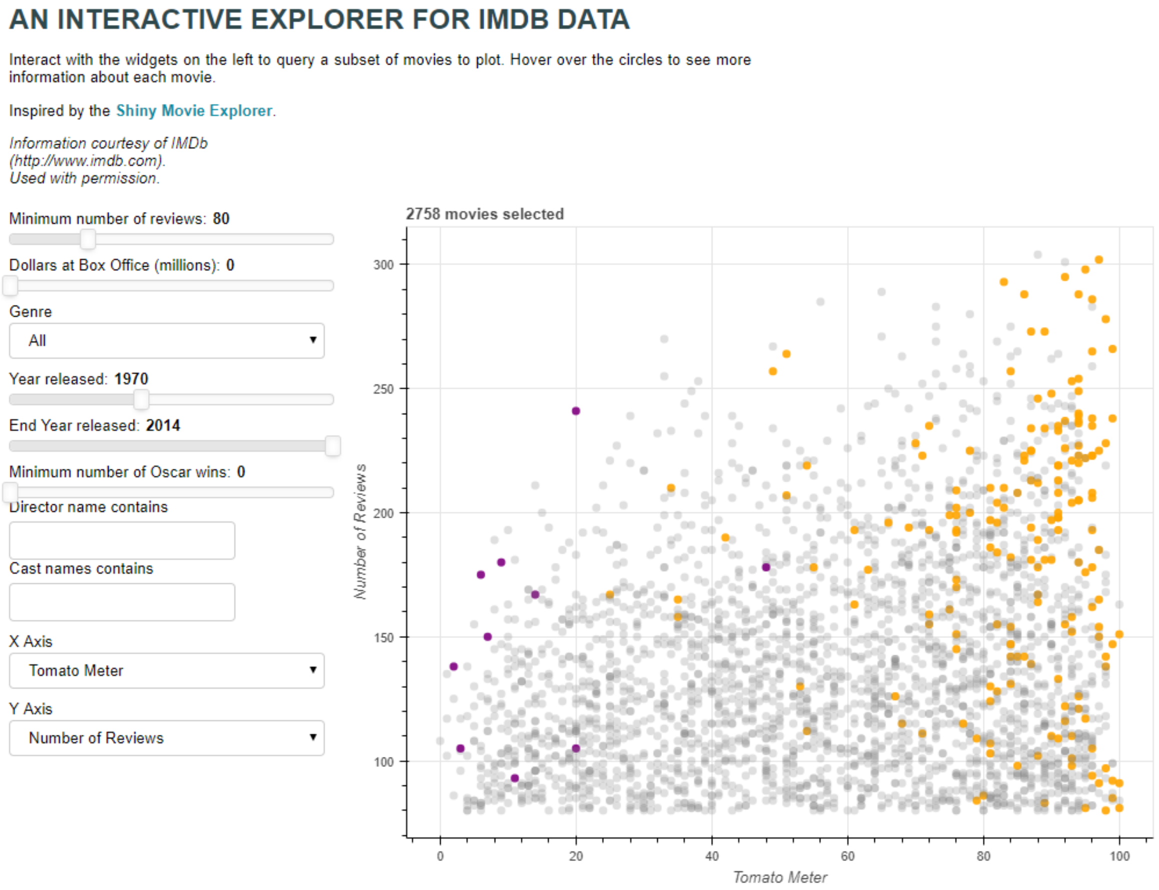
“enriched” vis

https://demo.bokehplots.com/

Bokeh Applications

This site hosts examples of applications built using Bokeh, a library for building data visualizations and applications in the browser from Python (and other languages), without writing JavaScript.

Exemplo 2
https://demo.bokehplots.com/apps/movies



AN INTERACTIVE EXPLORER FOR IMDB DATA

Interact with the widgets on the left to query a subset of movies to plot. Hover over the circles to see more information about each movie.

Inspired by the Shiny Movie Explorer.

Information courtesy of IMDb (http://www.imdb.com). Used with permission.

Minimum number of reviews: 80

Dollars at Box Office (millions): 0

Genre: All

Year released: 1970

End Year released: 2014

Minimum number of Oscar wins: 0

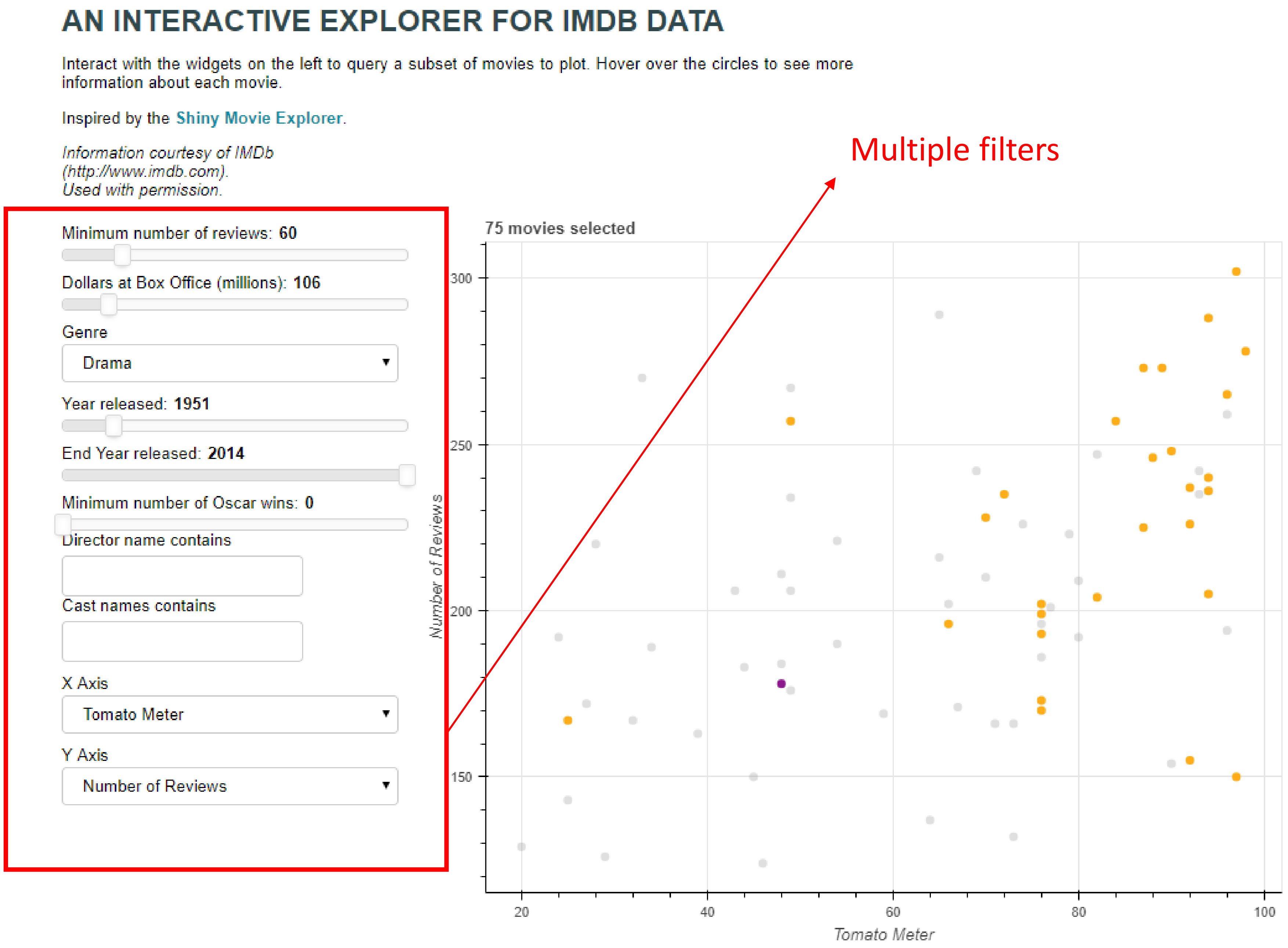
Director name contains

Cast names contains

X Axis: Tomato Meter

Y Axis: Number of Reviews

2758 movies selected



AN INTERACTIVE EXPLORER FOR IMDB DATA

Interact with the widgets on the left to query a subset of movies to plot. Hover over the circles to see more information about each movie.

Inspired by the **Shiny Movie Explorer**.

Information courtesy of IMDb (http://www.imdb.com). Used with permission.

Minimum number of reviews: **60**

Dollars at Box Office (millions): **106**

Genre: Drama

Year released: **1951**

End Year released: **2014**

Minimum number of Oscar wins: **0**

Director name contains

Cast names contains

X Axis: Tomato Meter

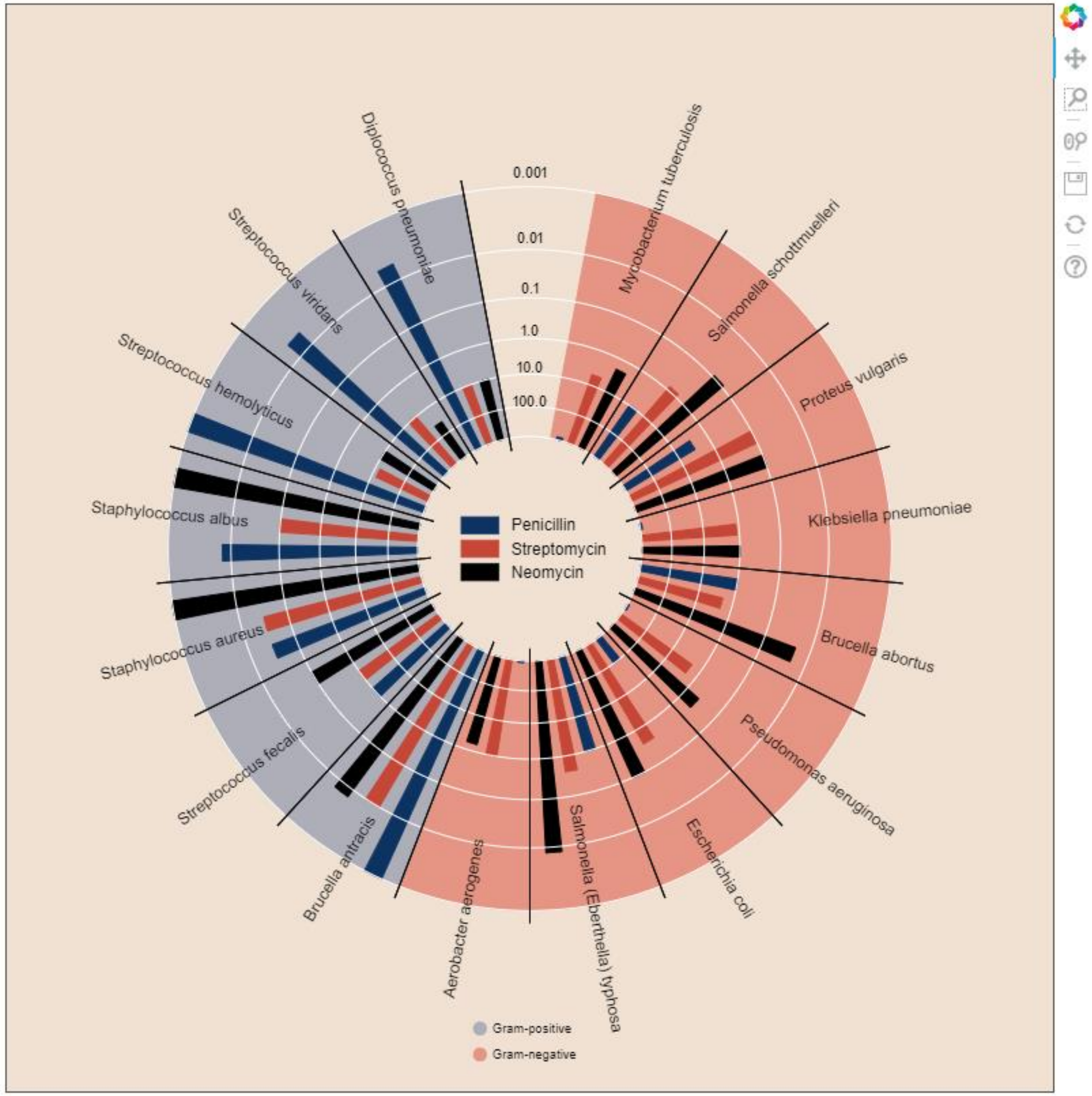
Y Axis: Number of Reviews

75 movies selected

Multiple filters

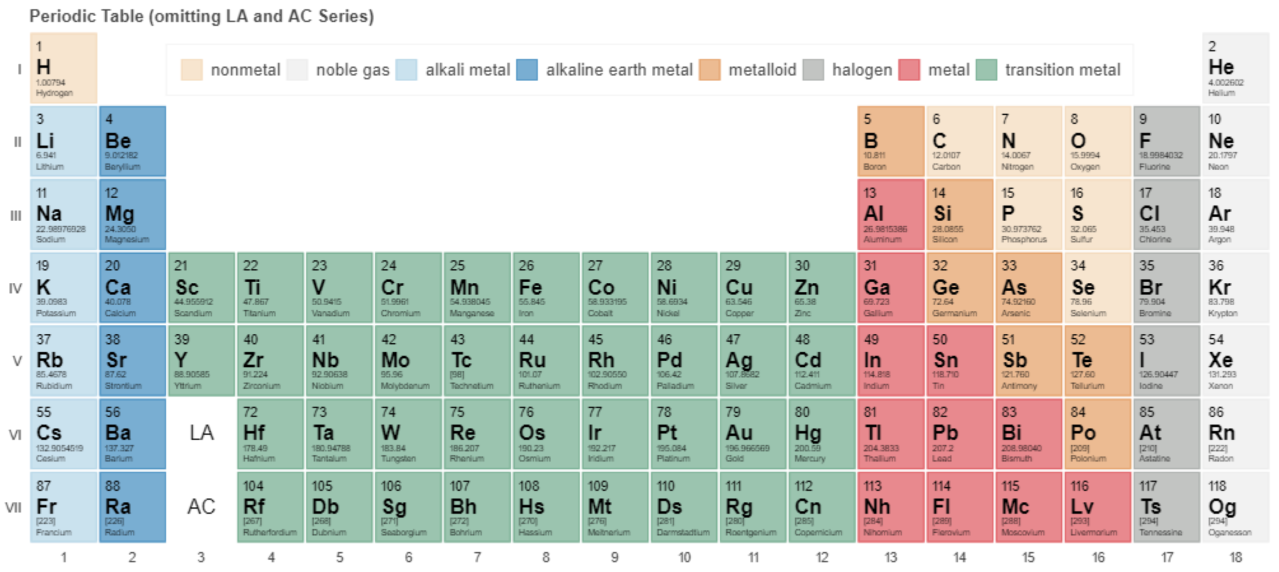
https://bokeh.pydata.org/en/latest/docs/gallery/burtin.html

burtin.py ¶



https://bokeh.pydata.org/en/latest/docs/gallery/periodic.html

periodic.py ¶

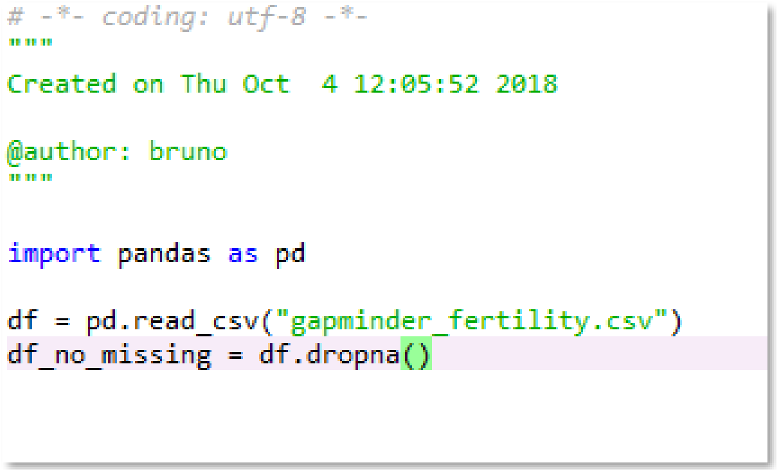


Periodic Table (omitting LA and AC Series)

nonmetal noble gas alkali metal alkaline earth metal metalloid halogen metal transition metal

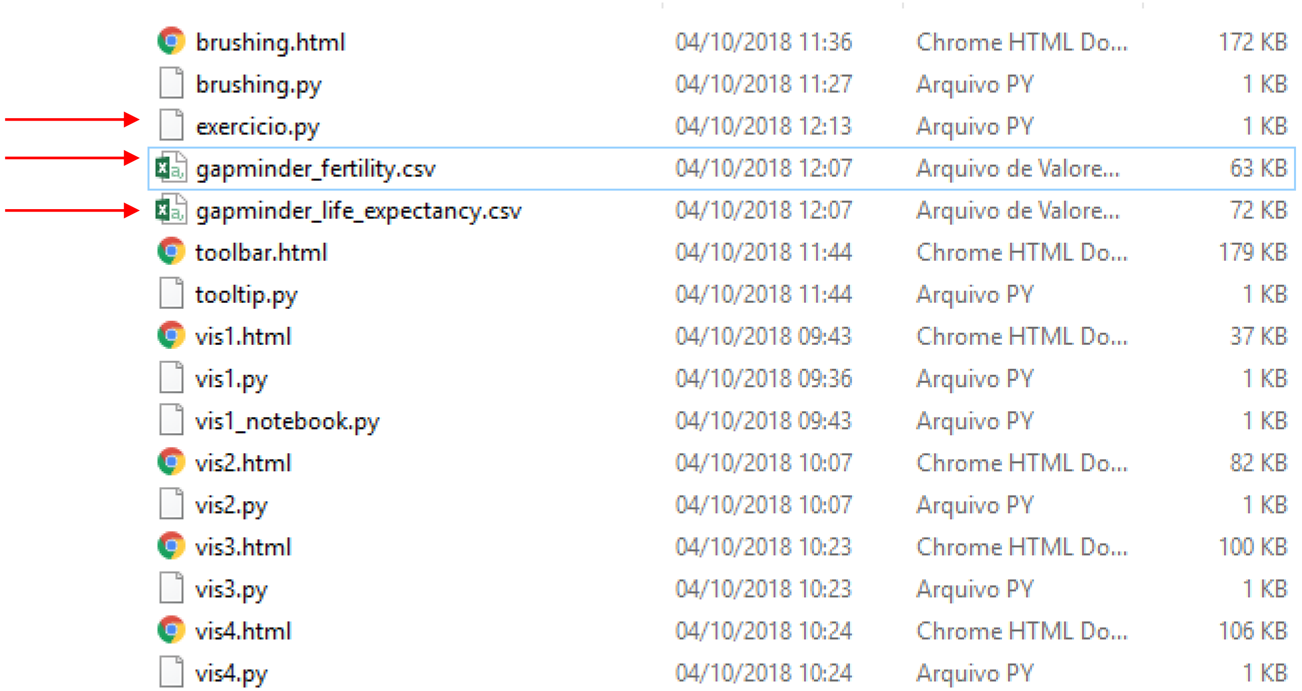
I	1 H 1.00794 Hydrogen																	2 He 4.002602 Helium
II	3 Li 6.941 Lithium	4 Be 9.012182 Beryllium											5 B 10.811 Boron	6 C 12.0107 Carbon	7 N 14.0067 Nitrogen	8 O 15.9994 Oxygen	9 F 18.9984032 Fluorine	10 Ne 20.1797 Neon
III	11 Na 22.98976928 Sodium	12 Mg 24.3050 Magnesium											13 Al 26.9815386 Aluminum	14 Si 28.0855 Silicon	15 P 30.973762 Phosphorus	16 S 32.065 Sulfur	17 Cl 35.453 Chlorine	18 Ar 39.948 Argon
IV	19 K 39.0983 Potassium	20 Ca 40.078 Calcium	21 Sc 44.955912 Scandium	22 Ti 47.867 Titanium	23 V 50.9415 Vanadium	24 Cr 51.9961 Chromium	25 Mn 54.938045 Manganese	26 Fe 55.845 Iron	27 Co 58.933195 Cobalt	28 Ni 58.6934 Nickel	29 Cu 63.546 Copper	30 Zn 65.38 Zinc	31 Ga 69.723 Gallium	32 Ge 72.64 Germanium	33 As 74.92160 Arsenic	34 Se 78.96 Selenium	35 Br 79.904 Bromine	36 Kr 83.798 Krypton
V	37 Rb 85.4678 Rubidium	38 Sr 87.62 Strontium	39 Y 88.90585 Yttrium	40 Zr 91.224 Zirconium	41 Nb 92.90638 Niobium	42 Mo 95.96 Molybdenum	43 Tc [98] Technetium	44 Ru 101.07 Ruthenium	45 Rh 102.90550 Rhodium	46 Pd 106.42 Palladium	47 Ag 107.8682 Silver	48 Cd 112.411 Cadmium	49 In 114.818 Indium	50 Sn 118.710 Tin	51 Sb 121.760 Antimony	52 Te 127.60 Tellurium	53 I 126.90447 Iodine	54 Xe 131.293 Xenon
VI	55 Cs 132.9054519 Cesium	56 Ba 137.327 Barium	LA	72 Hf 178.49 Hafnium	73 Ta 180.94788 Tantalum	74 W 183.84 Tungsten	75 Re 186.207 Rhenium	76 Os 190.23 Osmium	77 Ir 192.217 Iridium	78 Pt 195.084 Platinum	79 Au 196.966569 Gold	80 Hg 200.59 Mercury	81 Tl 204.3833 Thallium	82 Pb 207.2 Lead	83 Bi 208.98040 Bismuth	84 Po [209] Polonium	85 At [210] Astatine	86 Rn [222] Radon
VII	87 Fr [223] Francium	88 Ra [226] Radium	AC	104 Rf [267] Rutherfordium	105 Db [268] Dubnium	106 Sg [271] Seaborgium	107 Bh [272] Bohrium	108 Hs [270] Hassium	109 Mt [276] Meitnerium	110 Ds [281] Darmstadtium	111 Rg [280] Roentgenium	112 Cn [285] Copernicium	113 Nh [284] Nihonium	114 Fl [289] Flerovium	115 Mc [288] Moscovium	116 Lv [293] Livermorium	117 Ts [294] Tennessine	118 Og [294] Oganesson
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Mão na massa



```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct  4 12:05:52 2018

@author: bruno
"""

import pandas as pd

df = pd.read_csv("gapminder_fertility.csv")
df_no_missing = df.dropna()
```

exercicio.py



brushing.html	04/10/2018 11:36	Chrome HTML Do...	172 KB
brushing.py	04/10/2018 11:27	Arquivo PY	1 KB
exercicio.py	04/10/2018 12:13	Arquivo PY	1 KB
gapminder_fertility.csv	04/10/2018 12:07	Arquivo de Valore...	63 KB
gapminder_life_expectancy.csv	04/10/2018 12:07	Arquivo de Valore...	72 KB
toolbar.html	04/10/2018 11:44	Chrome HTML Do...	179 KB
tooltip.py	04/10/2018 11:44	Arquivo PY	1 KB
vis1.html	04/10/2018 09:43	Chrome HTML Do...	37 KB
vis1.py	04/10/2018 09:36	Arquivo PY	1 KB
vis1_notebook.py	04/10/2018 09:43	Arquivo PY	1 KB
vis2.html	04/10/2018 10:07	Chrome HTML Do...	82 KB
vis2.py	04/10/2018 10:07	Arquivo PY	1 KB
vis3.html	04/10/2018 10:23	Chrome HTML Do...	100 KB
vis3.py	04/10/2018 10:23	Arquivo PY	1 KB
vis4.html	04/10/2018 10:24	Chrome HTML Do...	106 KB
vis4.py	04/10/2018 10:24	Arquivo PY	1 KB