FGV EMAp

Tutoriais | Intro

python

Bokeh

---

08 / 10 (Hoje)

Bokeh

0.13.0 ▾ INSTALLATION USER GUIDE GALLERY TUTORIAL REFERENCE RELEASES DEVELOPER GUIDE

https://bokeh.pydata.org/

Welcome to Bokeh ¶

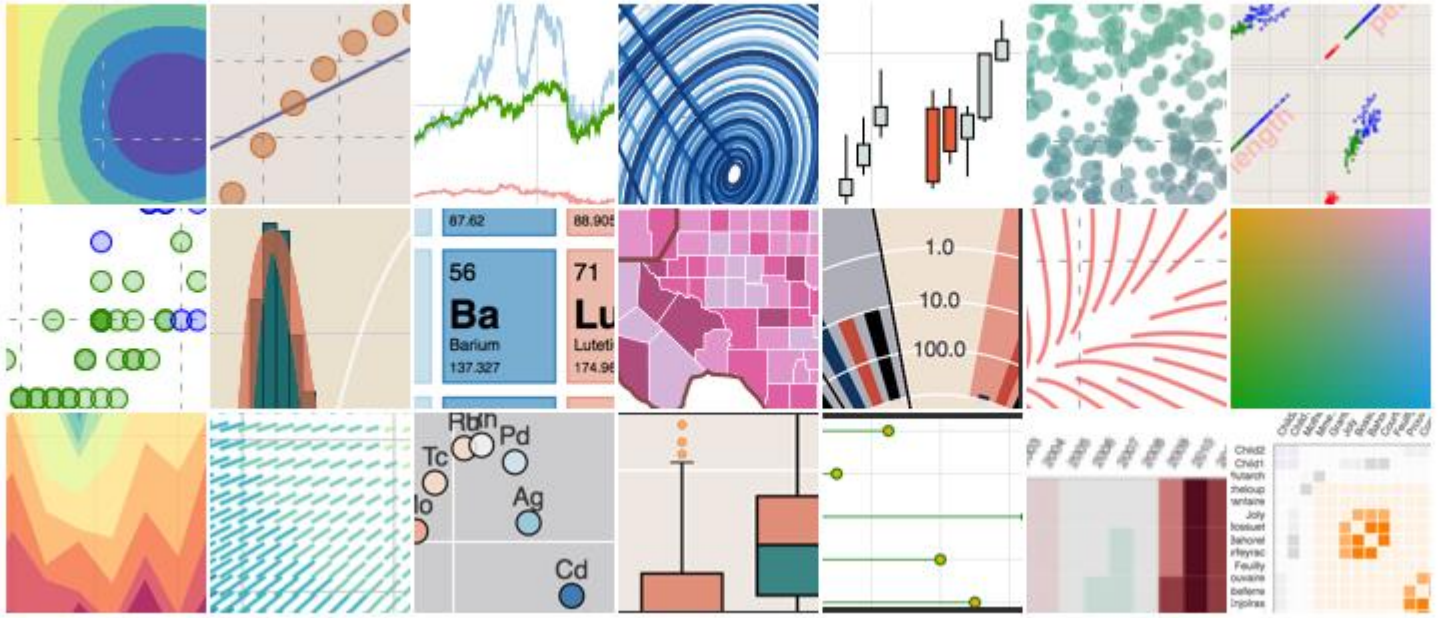
Bokeh is an interactive visualization library that targets modern web browsers for presentation. Its goal is to provide elegant, concise construction of versatile graphics, and to extend this capability with high-performance interactivity over very large or streaming datasets. Bokeh can help anyone who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

FGV EMAp

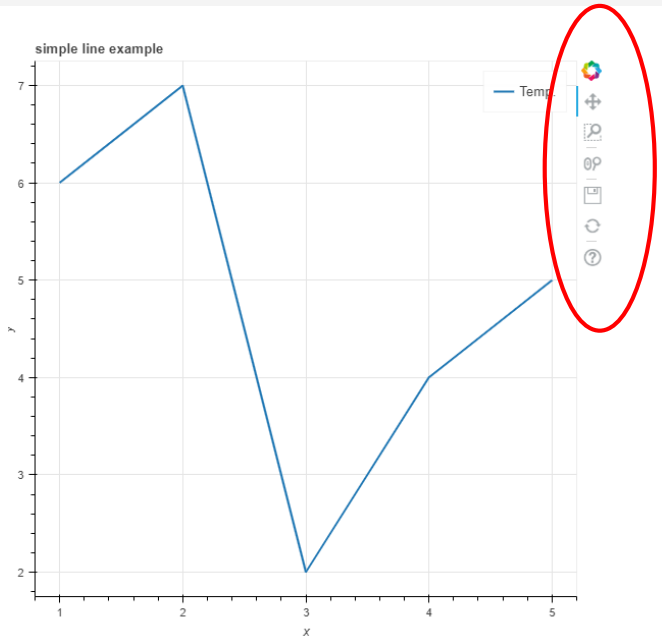
Key Concepts

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.

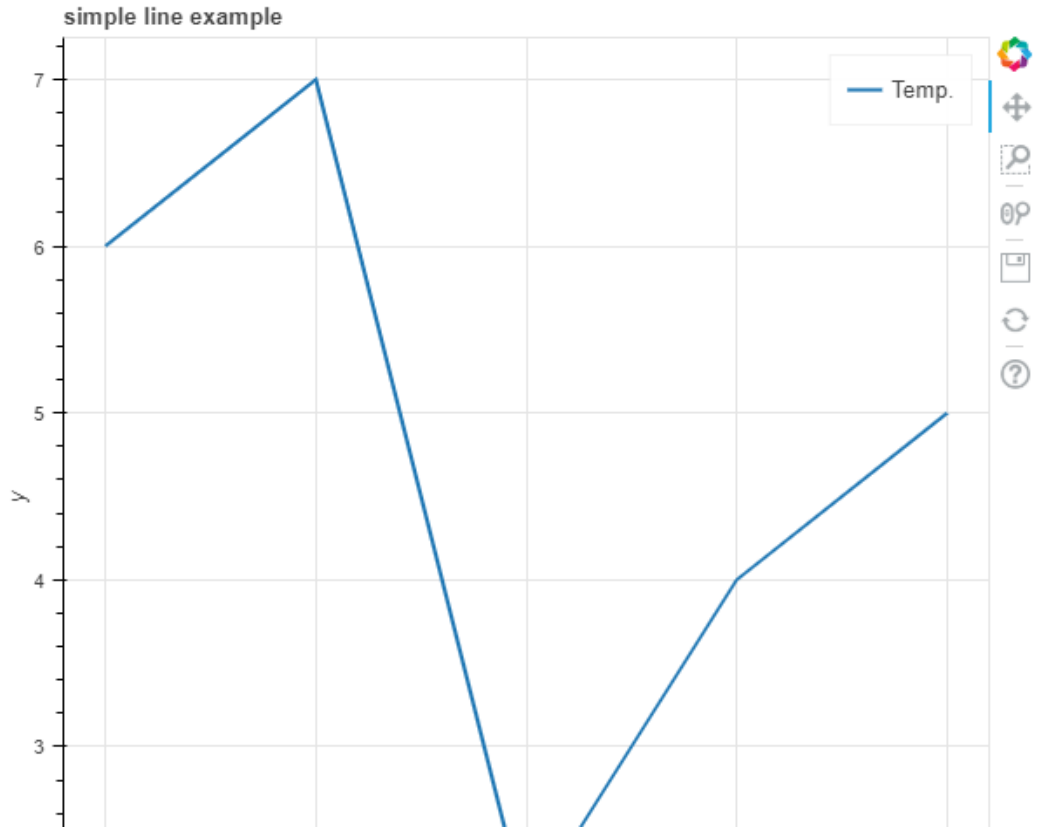
Ask Bokeh to `show()` or `save()` the results.

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

FGV EMAp

rbokeh - R Interface for Bokeh

Docs | Package Ref | Github

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].

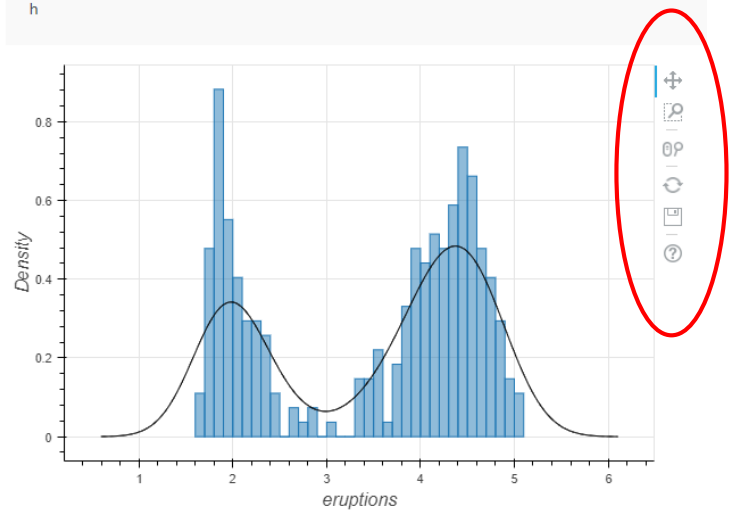
Installation

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
```



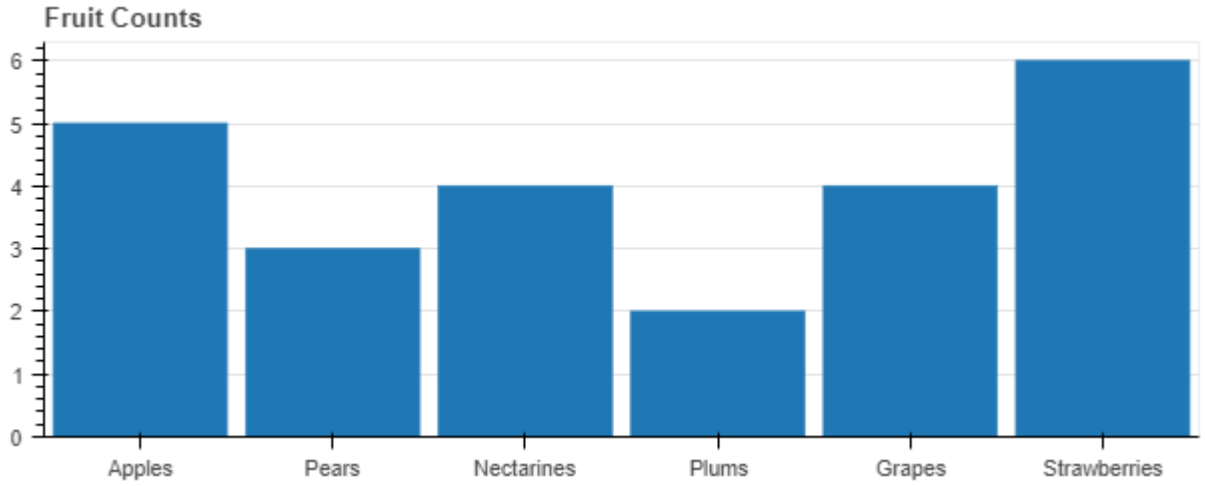
Handling Categorical Data

```
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

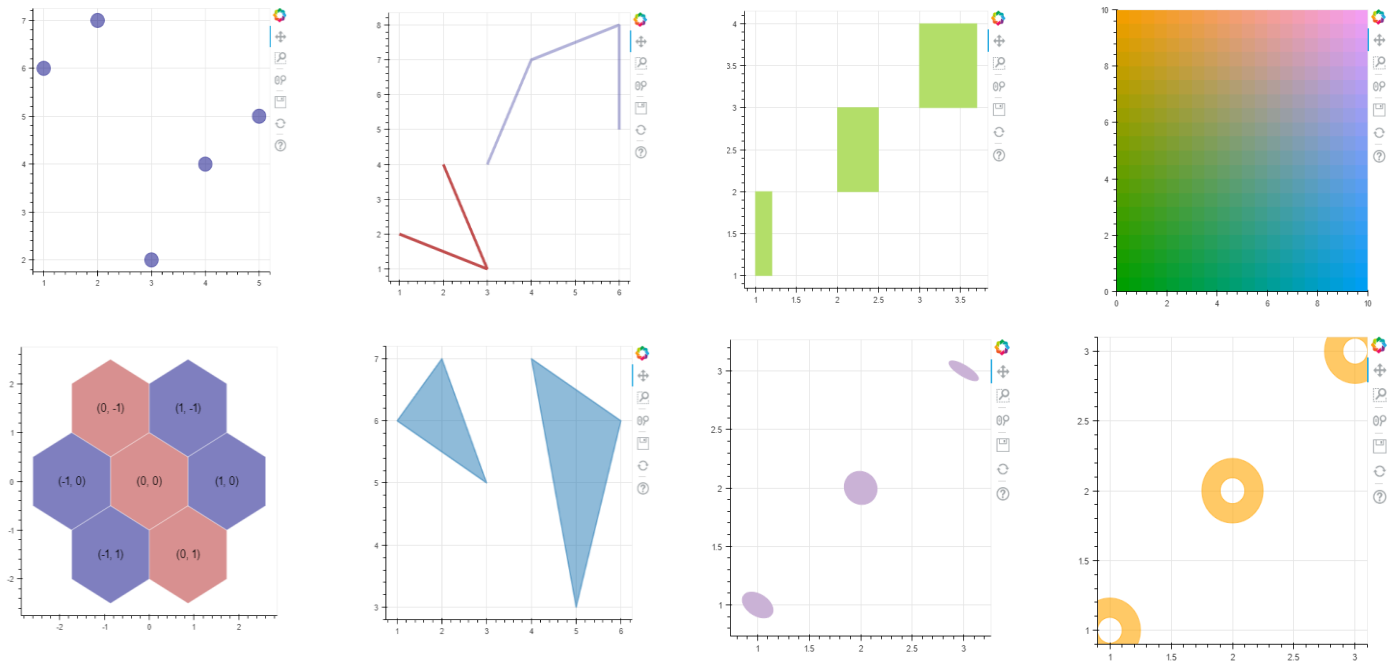
Ovals and Ellipses

Images

Segments and Rays

Wedges and Arcs

Specialized Curves

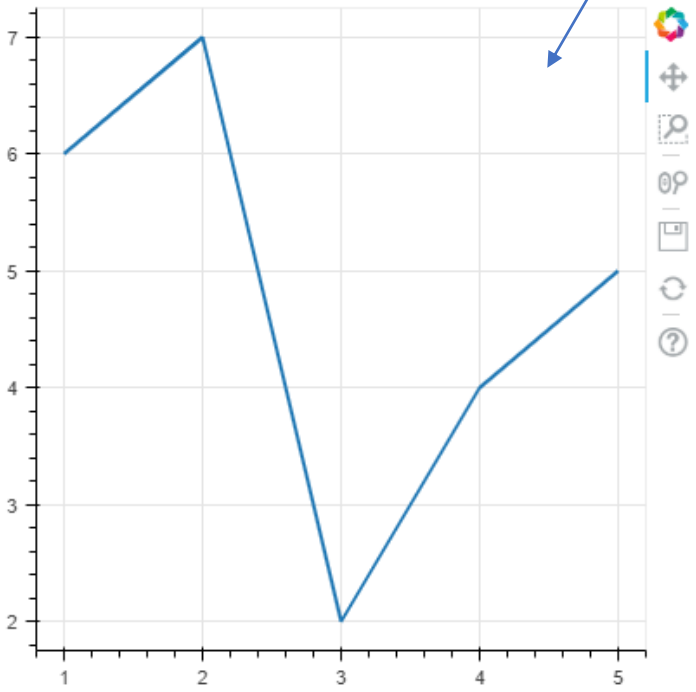


Line Glyphs

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:

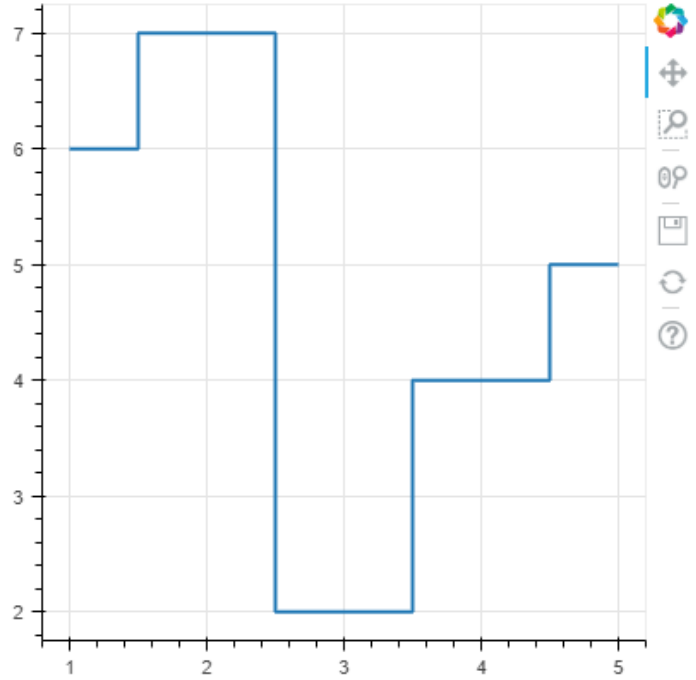
```
from bokeh.plotting import figure, output_file, show

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)
```



vis3.py

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

vis4.py

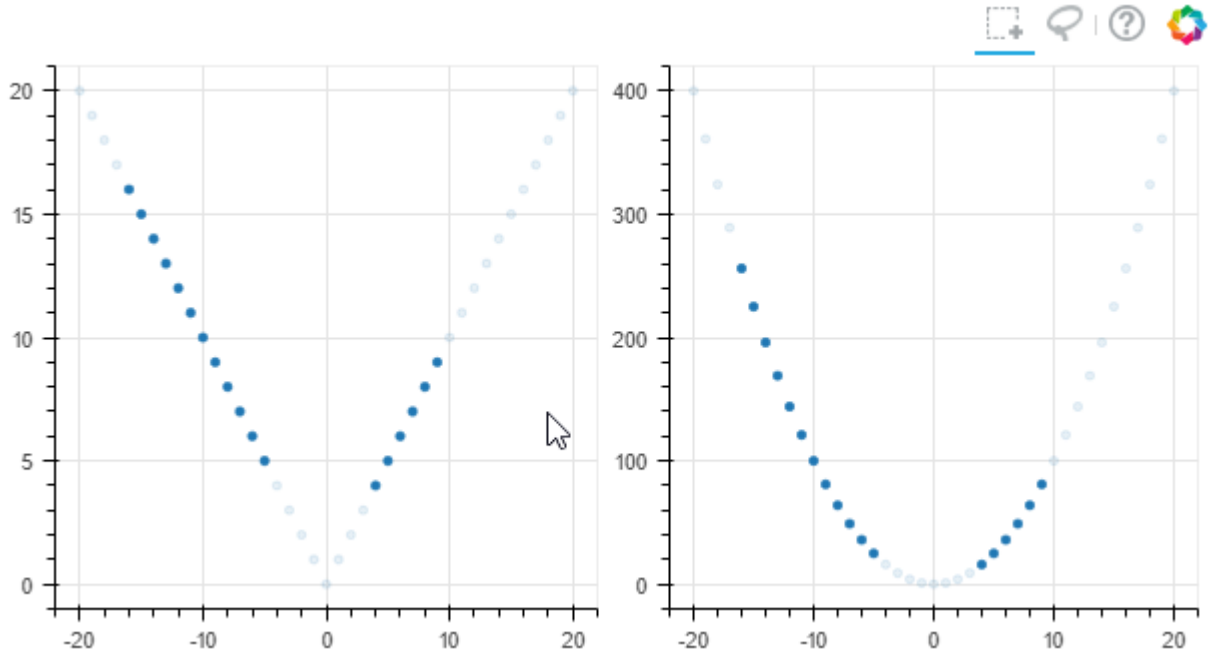


https://bokeh.pydata.org/en/latest/docs/user_guide/interaction/linking.html

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.

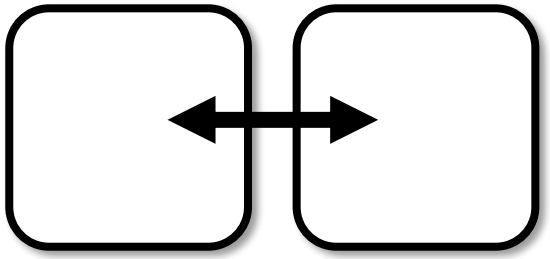
```
from bokeh.io import output_file, show
from bokeh.layouts import gridplot
from bokeh.models import ColumnDataSource
from bokeh.plotting import figure

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

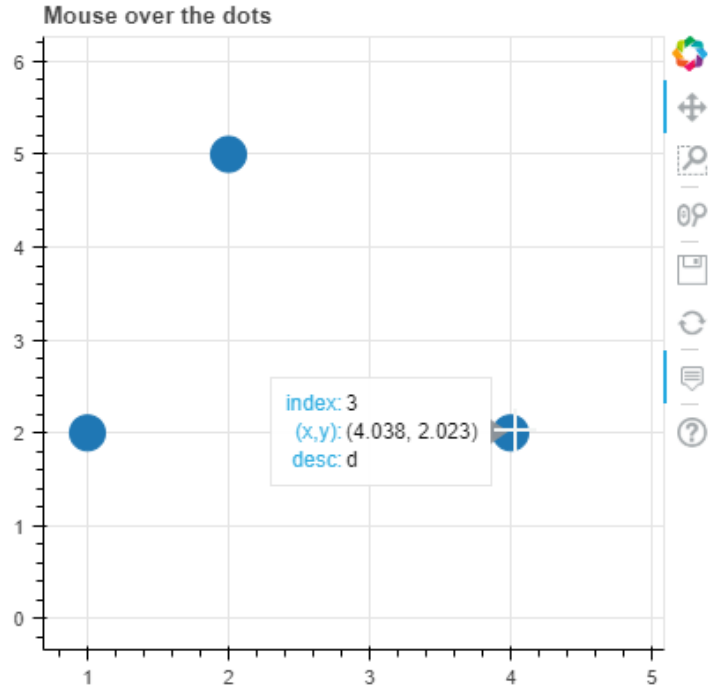
https://bokeh.pydata.org/en/latest/docs/user_guide/tools.html#basic-tooltips

```
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)
```

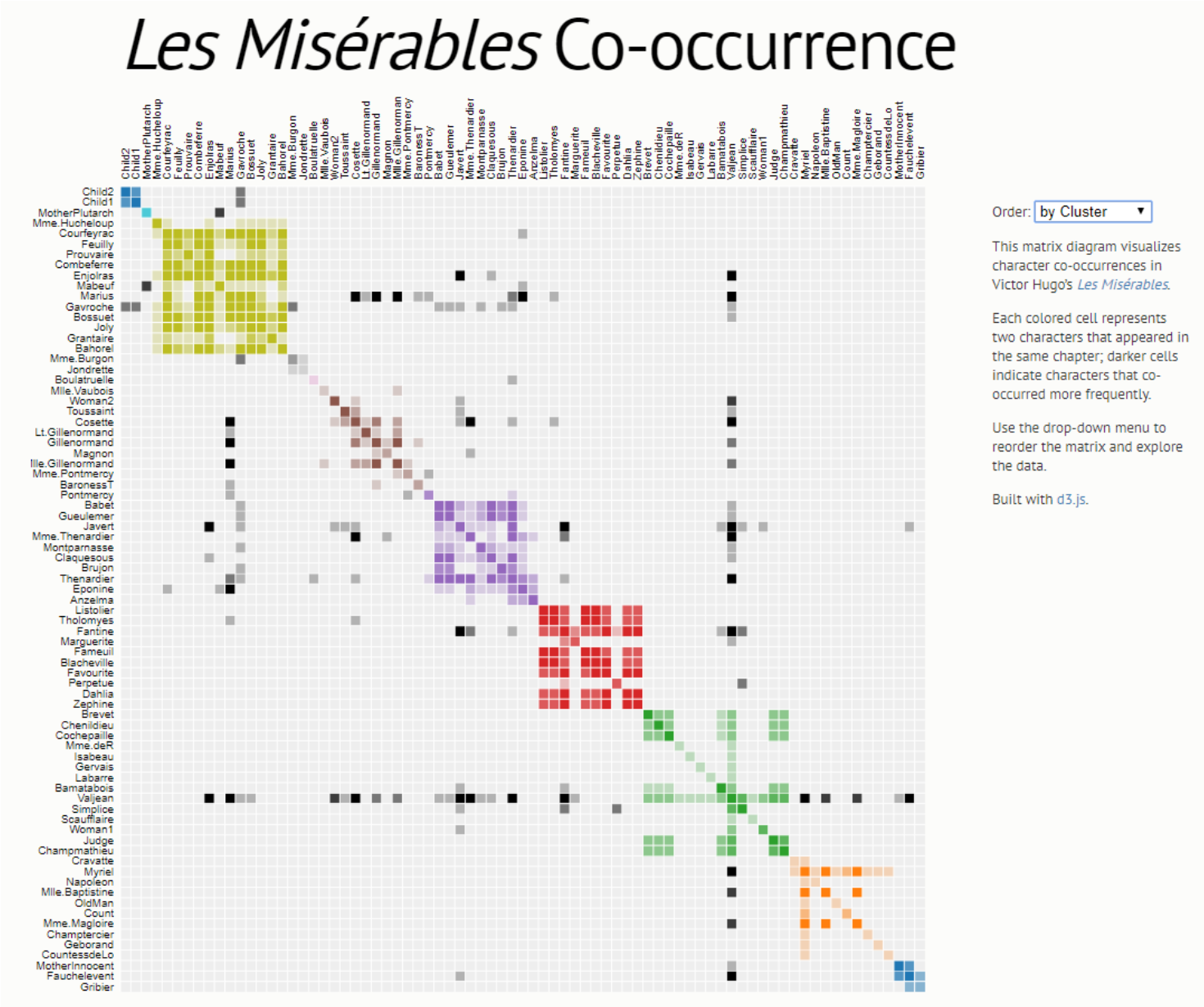
tooltip.py

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/



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

FGV EMAp

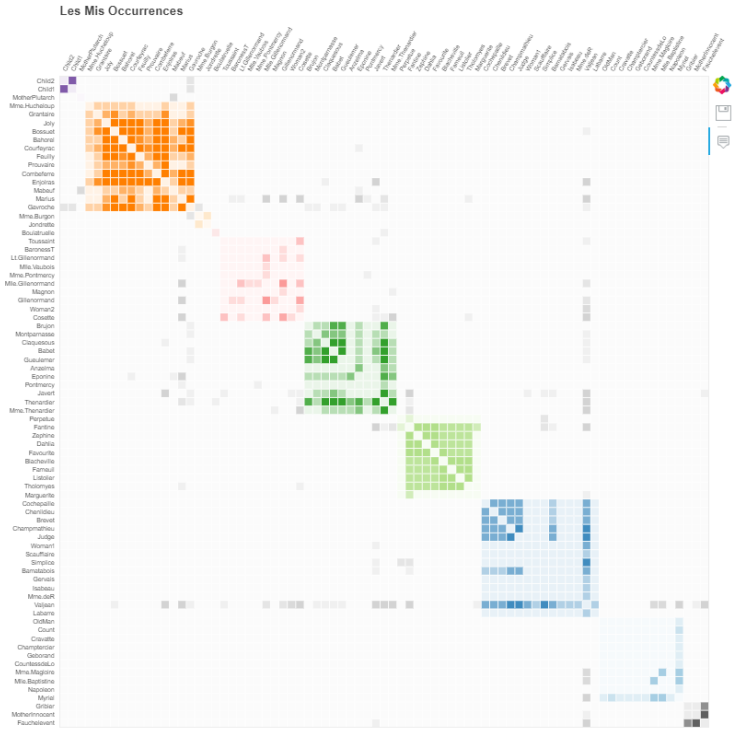
```python
import numpy as np

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(),
)
```

Preparação dos dados

```python
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
```

Plot

les_mis.py ¶



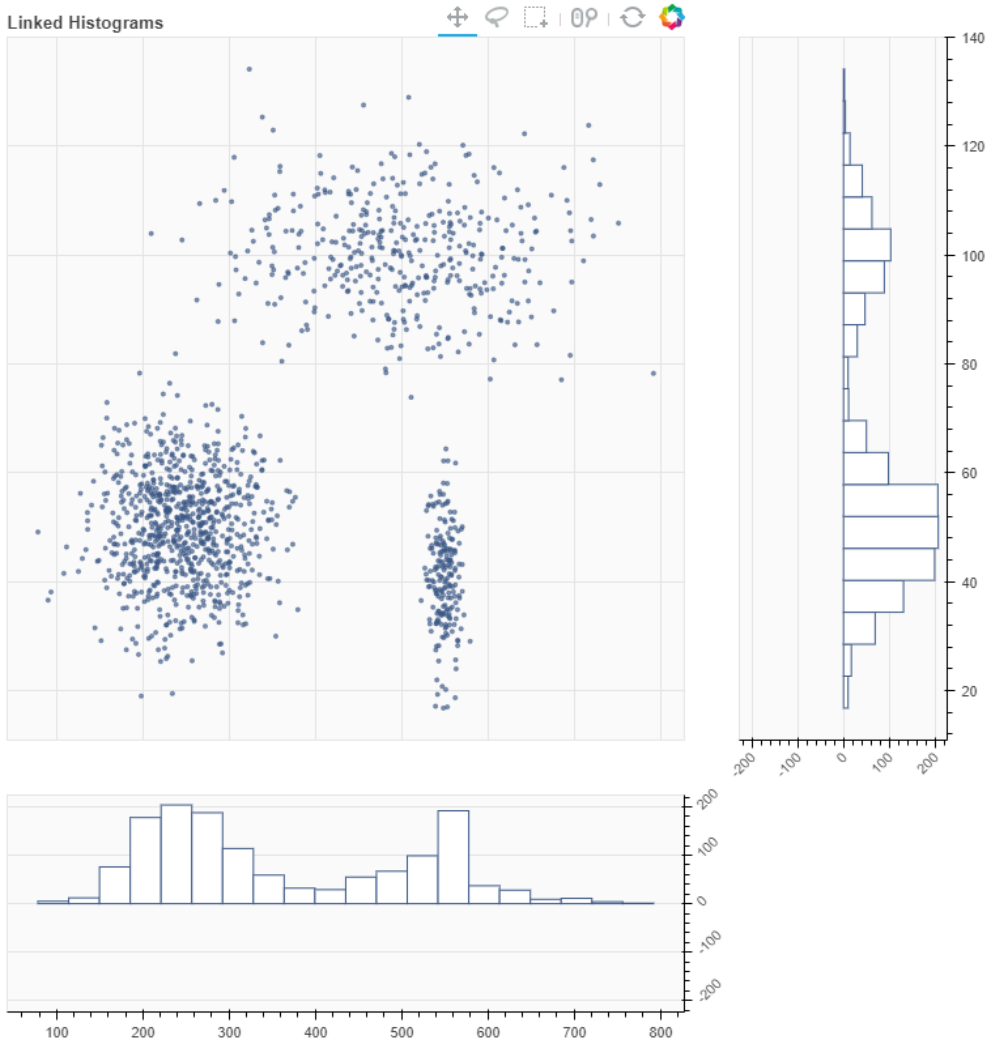
https://demo.bokehplots.com/

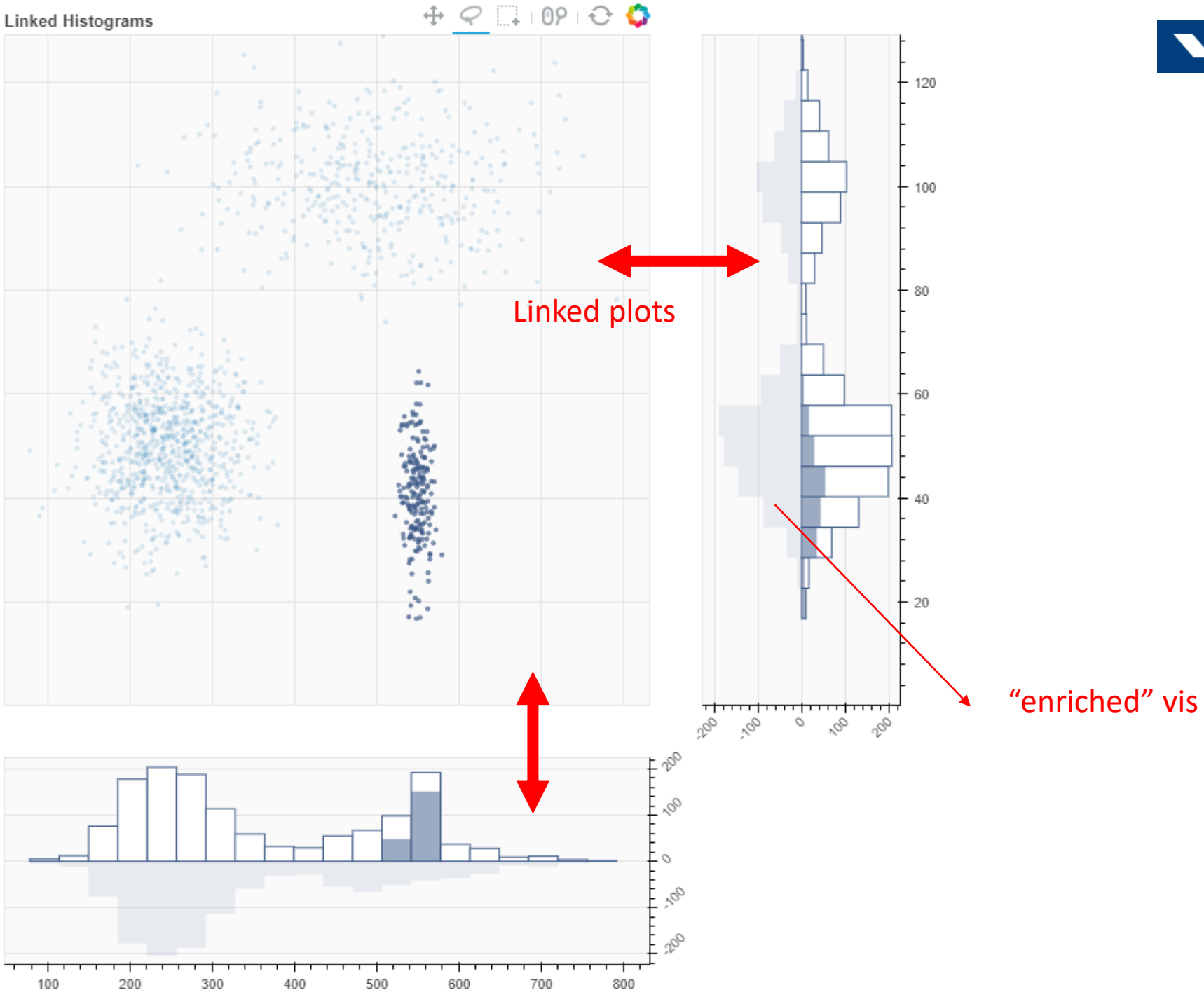
FGV EMAp

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



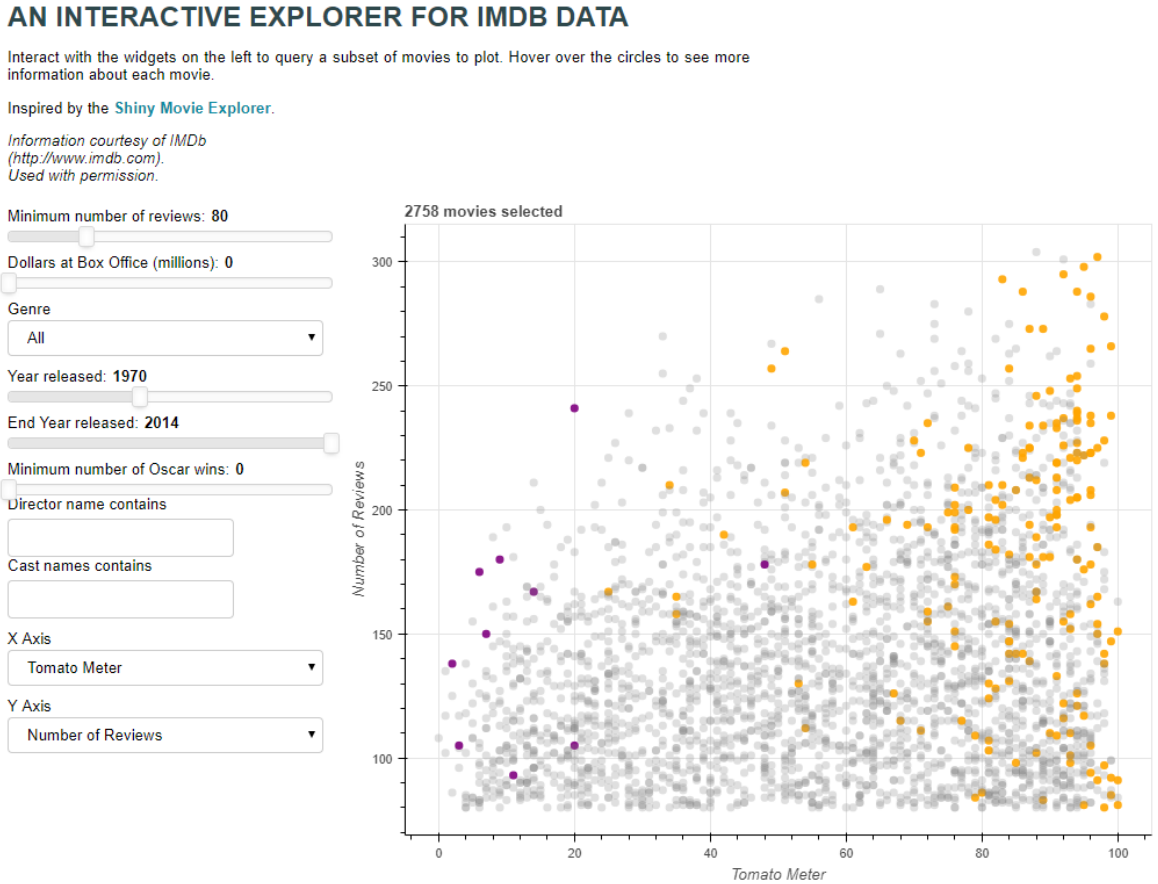

Linked Histograms
Linked plots
"enriched" vis
120
100
80
60
40
20
-200
-100
0
100
200
100
200
300
400
500
600
700
800

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.

Multiple filters

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

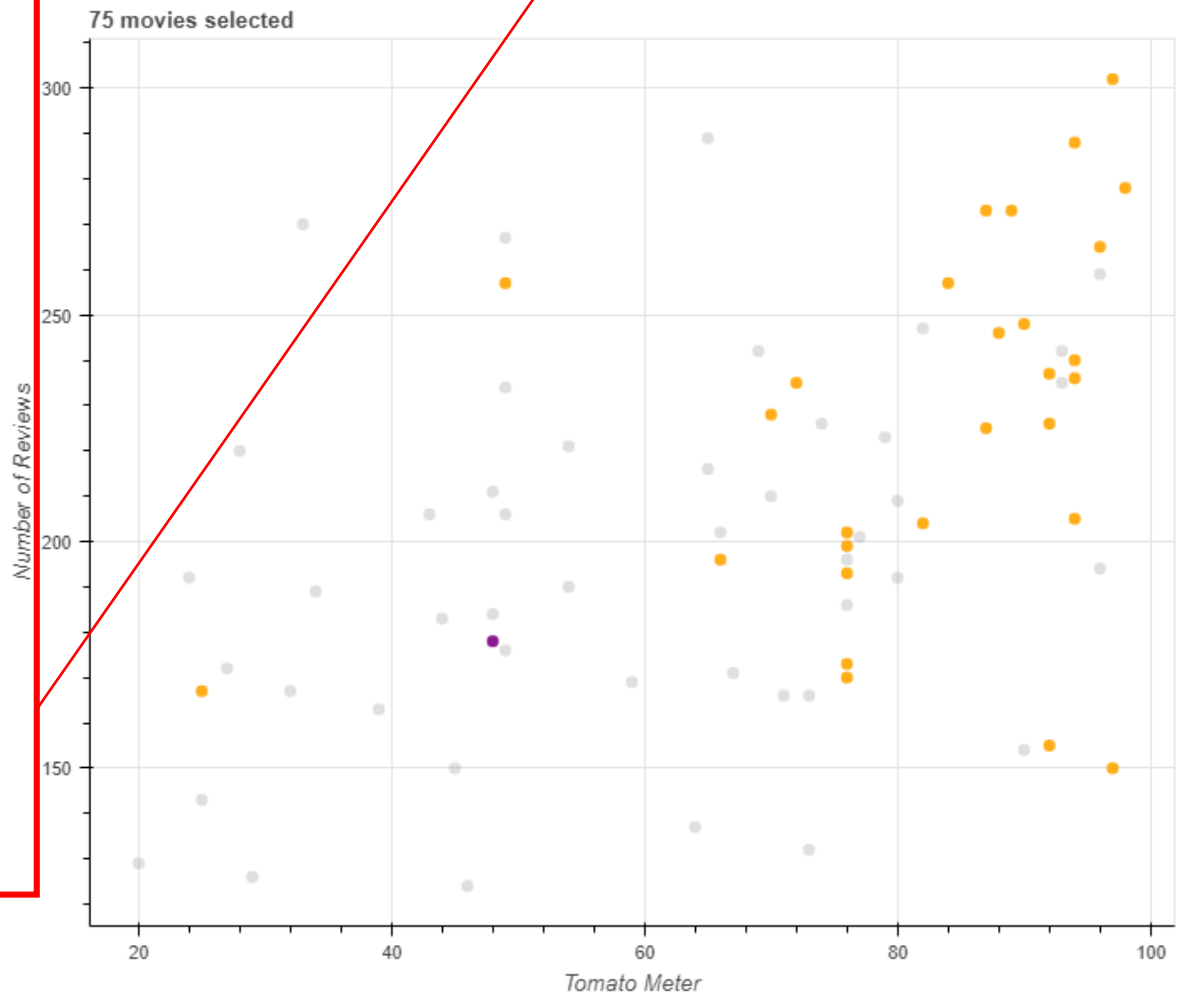
Cast names contains

X Axis

Tomato Meter

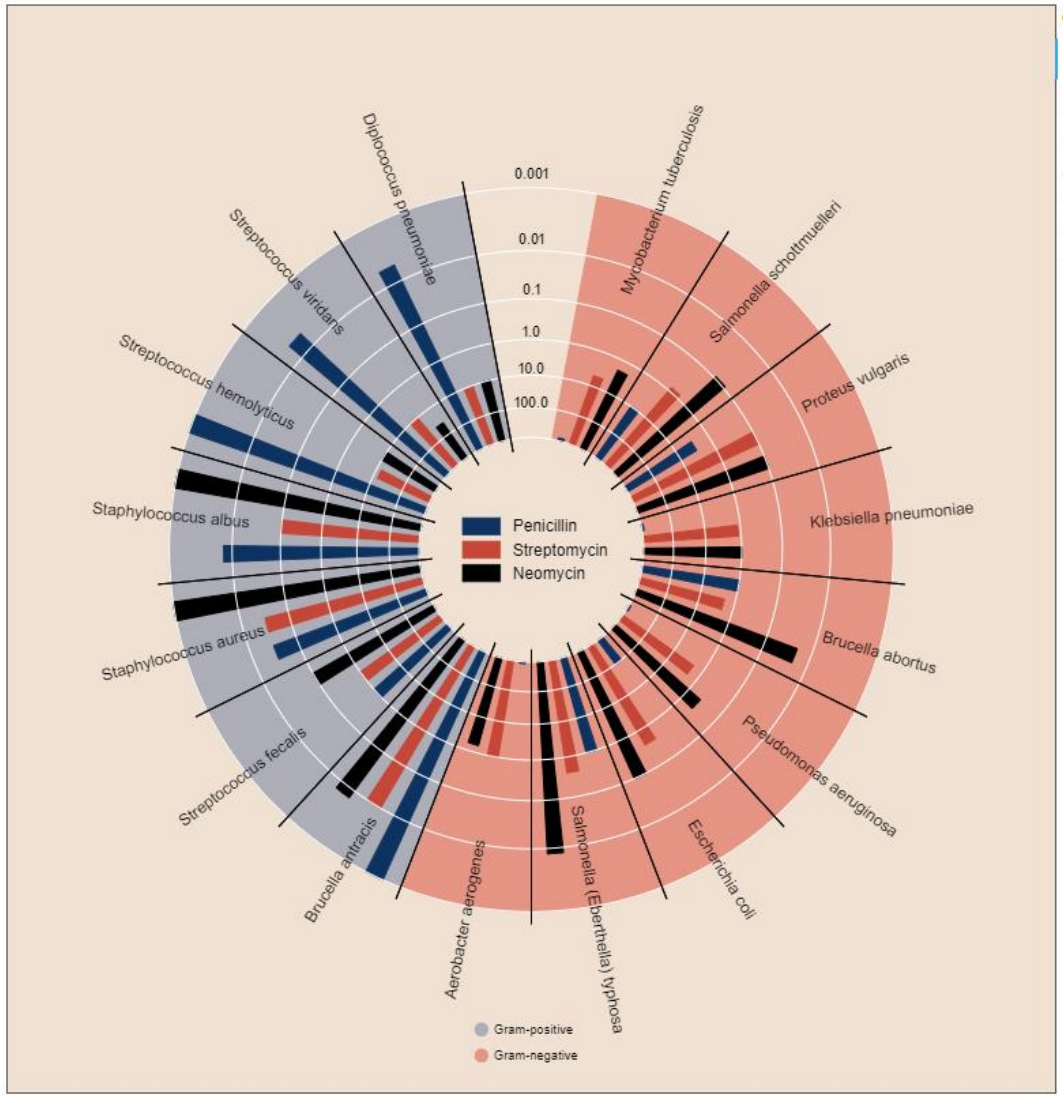
Y Axis

Number of Reviews



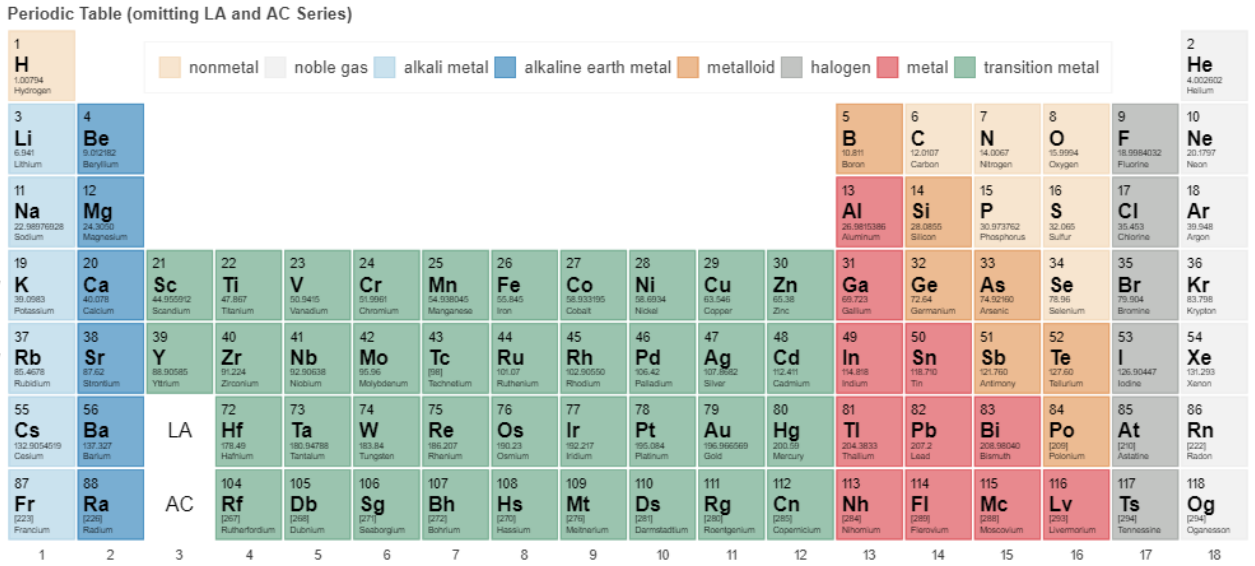
https://bokeh.pydata.org/en/latest/docs/gallery/burtin.html

burtin.py ¶



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

periodic.py ¶

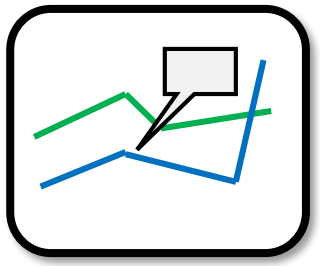


Mão na massa

```python
# -*- 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

Line chart + tooltip



Compartilhados comigo > Visualização_Turma2018 > Bokeh_Python tutorial .py .csv files

Nome

- brushing.py
- exercicio.py
- gapminder_fertility.csv
- gapminder_life_expectancy.csv
- tooltip.py
- vis1_notebook.py
- vis1.py
- vis2.py
- vis3.py
- vis4.py