Analysis and research on computer visualization in data science with bokeh and JavaScript

Tianyu Zhang*, Long Mei
North China Electric Power Research Institute, Beijing 100032, China

*Corresponding author Email: zzzttttyy456@126.com

Abstract. With the continuous development of data science, the data manifests an explosive growth in volume and dimension. In this historic background, the demand for data visualization is becoming increasingly urgent, because of which, many powerful instruments have been invented for handling the issues. Among them, the bokeh is a burgeoning, JavaScript-powered, open source python library which draws investigators, data scientists and developers. Present review mainly casts light on the application in field of data visualization with bokeh, including glossary, figures’ type and widgets.

Keywords: visualization, data science, bokeh.

1. Introduction
Bokeh is a Python library for creating interactive visualizations for modern web browsers [1]. It helps you build beautiful graphics, ranging from simple plots to complex dashboards with streaming datasets. With Bokeh, you can create JavaScript-powered visualizations without writing any JavaScript yourself.

As python is becoming more and more popular in data mining and artificial intelligence [2,3], despite the self-contained, long-historied and well-maintained advantages of matplotlib, it still exposes some weakness in the terms of interactive design, default setting and novice friendliness. bokeh supplies beneficial supplement for python environment.

2. Glossary of terms
It is a glossary which defines the most important concepts and terms that Bokeh’s environment consists of. Limited by the length of present review, only the common-used terms are demonstrated.
Table 1. Common items in bokeh

| Index | Terms | Usages |
|-------|-------|--------|
| 1.    | plot  | Containers that hold all the various objects (such as renderers, glyphs, or annotations) of a visualization. The bokeh.plotting interface provides the figure() function to help with assembling all the necessary objects. |
| 2.    | model | The lowest-level objects that Bokeh visualizations consist of. |
| 3.    | glyph | API objects that draw vectorized graphics to represent data. Glyphs are the basic visual building blocks of Bokeh plots. This includes elements such as lines, rectangles, squares, wedges, or the circles of a scatter plot. |
| 4.    | layout | A collection of Bokeh objects. This can be several plots and widgets, arranged in nested rows and columns. |
| 5.    | annotation | Visual aids that make reading the plot easier. This includes titles, legends, labels, or bands, for example. |
| 6.    | renderer | General term for any method or function that draws elements of the plot. Examples of elements that are generated by renderers are glyphs or annotations. |
| 7.    | widget | User interface elements that are not directly part of a Bokeh plot, such as sliders, drop-down menus, or buttons. You can use events and data from widgets in your Python code, or you can use input from widgets to update your Bokeh plot itself. |

3. Demonstration of figures in bokeh

With just a few lines of Python code, Bokeh enables you to create interactive, JavaScript-powered visualizations displayable in a web browser. The basic idea of Bokeh is a two-step process: First, you select from Bokeh’s building blocks to create your visualization with a python-based library. Second, you customize these building blocks to fit your needs with a JavaScript library which underlies in the background. This chapter introduces several types of figures and codes itself.

3.1. Frequently used figures’ type

It first begins with plotting with basic glyphs that are most common used, such as scatter markers, line glyphs and bars. The simplest demo figures are shown below.

As shown above, a whole figure consists of main elements: axes, glyph and data. Each main element also includes required and optional arguments to adjust the figure to fit certain demand. For example, The type of axis can be selected as datetime and category by stating `x_axis_type = ‘datetime’ and ‘x_range=FactorRange()’, the size and shape of the scatter can be modified though
size in figure statement and diamond outside figure() function, the bars’ width and color can be adjusted as well to satisfy various visualization needs.

Limited by the space of present paper, three basic figure types are shown. Flexible and suitable use of these basic plots can result in complex and elegant plotting, as shown in Fig. 2. The more complicated figure fuses more elements, such as legend, multi-data and coloring, which belongs to the type of glyph and renderers. Its novice-friendly character makes it easy to learn and use.

3.2. Making interactions in figure
Unlike the popular libraries Matplotlib and Seaborn in the field of Python visualization, bokeh takes advantages of HTML and JavaScript to render its graphics, which makes it an ideal candidate in building web-based applications. It is very helpful for exploring and understanding data, or creating beautiful project reports and custom charts with the method of interacting with the figure itself.

Main interactions include linking behavior, interactive legend, and widgets.

3.2.1. Linking behavior. It’s often useful to link plots to add connected interactivity between plots. The behavior in present section comprises linked panning, brushing and properties.

Linked panning
It’s often desired to link pan or zooming actions across many plots. All that is needed to enable this feature is to share range objects between figure() calls by stating \( x_{\text{range}}=s1.x_{\text{range}} \) and \( y_{\text{range}}=s1.y_{\text{range}} \) where \( s1 \) is the plot object’s name, then sum them up by gridplot to achieve the coordinating in ranges between different plots. As shown below, the middle figure shares the same range with the first one, while the third figure only shares the x-axis, having fixed y-axis range.

Fig. 2 Complex figures from simple ones

Fig. 3 Linked Panning Demo
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.

![Fig 4. Linked brush](image)

![Fig 5. Linked properties (radius in scatter)](image)

Linked properties

It is also possible to link values of Bokeh model properties together so that they remain synchronized, using the `js_link` method. The example shown in Fig 5. links a circle glyph radius to the value of a slider widget.

3.2.2. Interactive legend. Legends added to Bokeh plots can be made interactive so that clicking or tapping on the legend entries will hide or mute the corresponding glyph in a plot. These modes are activated by setting the `click_policy` property on a Legend to either "hide" or "mute". Hiding means making the line disappear entirely, while muting means making the line leaving a shadow by stating `mute_alpha` and `mute_color`.

![Fig. 6 Hiding and muting of legend (Note the difference in AAPL line)](image)

3.2.3. Adding widgets. Widgets are interactive controls that can be added to Bokeh applications to provide a front-end user interface to a visualization. They can drive new computations, update plots, and connect to other programmatic functionality. When used with the Bokeh server, widgets can run
arbitrary Python code, enabling complex applications. Widgets can also be used without the Bokeh server in standalone HTML documents through the browser’s JavaScript runtime.

Various widgets make the figure more comprehensive and easier to read than static picture. Common used widgets include button, checkbox group, data table, dropdown list, slide and spinner. To use widgets, you must add them to your document and define their callbacks. Widgets can be added directly to the document root or nested inside a layout. All widgets take an attribute name and one or more event handlers as parameters, through which, the widgets are linked with plots and glyph. Due the limited paper, further description can be referred in relative literature [4].

4. Conclusion
Compared to data analysis, AI and quantitative investigation, data visualization in python is, to some extent, lagging behind. The most classical library – matplotlib, which animates the MATLAB plotting library, is increasingly difficult to meet the demand of modern aesthetic. The coming out of bokeh makes a complement with its D3.js based library and great compatible ability with Web. Present paper only takes a glimpse on the visualization by bokeh from limited aspects, and the real powerful instrument for plotting interactive and modern-viewed figure still needs to be explored.

References
[1] Sarah Bird, Luke Canavan. Bokeh User Manual [M]. NumFOCUS, 2020.
[2] Liu Yanling, Yao Jiansheng. Application of Python in data visualization [J]. Fujian computer, 2020 (3): 68-70.
[3] Luo Bowei. Data visualization based on Python [J]. Information recording materials, 2019, 020 (012): 72-74.
[4] Qu X. Python Data Visualization: Dynamic Visualization Based on Bokeh [M], China Machine Press, 2020.