International Journal of Computer Applications (0975 - 8887)
Volume 121 - No. 1, July 2015

Usability of Visualization Libraries for Web Browsers for Use in Scientific Analysis

Luke Barnard
Technical Student
CERN, Route de Meyrin 385
1217 Meyrin, Switzerland

Matej Mertik
Scientific Associate
CERN, Route de Meyrin 385
1217 Meyrin, Switzerland

ABSTRACT

Data visualization is incredibly important in order to make informed decisions about the relationships between different data in a dataset. When datasets are stored in relational databases, they can become complex and it is necessary to build tools that can condense the vast number of possible relationships to a few that are of interest. Navigating a dataset is one challenge; SQL queries are necessary to access the data and can be confusing to those with a non-technical background. Once data is acquired, it should be simple to visually test the relationships between data. This is the common workflow of an ELQA (ELectrical Quality Assurance) engineer at CERN, and a tool is described here that shortens this process based on the experience of the engineers. The currently available software libraries for data access, analysis and visualization were used to create such a tool. The focus of this paper is on data visualization and the libraries that can be used to provide an interactive, web-based solution for multiple members of a data collection team.

General Terms:

Electrical Quality Assurance, Data Mining, Cluster Analysis, Data Visualization, Web-Based Solutions, Object-Relational Mapping

Keywords

Python, Bokeh, Mpld3, Django, Oracle

1. INTRODUCTION

Python is attracting a lot of attention from the scientific and research computing communities [1]. There are many reasons for using Python. It is readable by nature allowing for easy code maintenance and future development. Compilation is not needed because of it's dynamically-typed nature and reuse of code written in other languages is possible. For these reasons and also due to the abundance of well written, well documented, scientific Python libraries, Python has become very popular in scientific communities [2, 3].

The TE-MPE-EE section (Electrical Engineering section of the Machine Protection and Electrical Integrity group of the Technology department) of CERN is responsible for electrical quality assurance during the hardware commissioning phase of the LHC (Large Hadron Collider). The hardware commissioning of the LHC consists of carrying out various tests on the superconducting circuits and are of vital importance for the successful running of the LHC. In order to interpret and verify measurements of tests, different tools are developed at CERN in environments such as LabView and specially developed web interfaces are used to access the data. These tools expose the data to the engineers in way a useful for the job at hand. However, none of them are capable of observing the data set as a whole in order to make better decisions as to where potential trends are occurring between confounding and measured variables during test campaigns. This requires a solution that is fundamentally different to the current, rather simple data acquisition applications. This solution should integrate appropriate data analysis methods from the domains of statistics, machine learning and data mining in order to explore the data for better insights and search for potentially related patterns.

Presented in this paper is a study of the selection of the appropriate components for developing an analytical tool designed for the specific analysis of the LHC hardware commissioning data. The selection of the data visualization component is focused on here. There are many different solutions currently being developed by different open source projects. A methodology for selecting appropriate programming libraries for this task is presented and some of their strengths are compared. Finally the architecture of the tool that was developed through an iterative prototype development process whilst searching for optimal solutions to develop the tool is presented.

The main contribution of this paper is to outline and detail the process of component selection for creating a bespoke software solution within the confines of a highly developed scientific environment.

The components support integration of efficient access to the data, different algorithms for mining and exploring the data and efficient and dynamic visualization of the results in a web environment within interactive plots.

Performance, security, interoperability, modifiability and architecture are the aspects that were used in order to select an appropriate library for data visualization. Each candidate library varies across all of these and it is necessary to consider them all in relation to the tool discussed here.

2. BACKGROUND AND OVERVIEW

Sophisticated, modern, web-based, collaboration environments today require an interactive computational environment with support for various output formats such as HTML, PDF and others. Used in the scientific community, IPython notebooks are JSON documents containing an ordered list of input/output cells which can contain code, text, mathematics, plots and rich media [4]. IPython is capable of sharing various data analyses, however it is a framework for sharing various users' data and analyses within a community and it is not a framework on which sophisticated applications for data analysis can be straightforwardly developed. Also, it is necessary to have Python programming skills in order to use it, which some ELQA team members do not possess.

Because IPython is not appropriate for the tasks of ELQA, and because there is no other suitable tool available, a combination of existing software components was required. ORM (Object-Relational Mapping) for Oracle databases was used with a selection of appropriate python frameworks to allow for SQL-free algorithms to acquire and analyze the data. Web-oriented libraries for plotting were then tested, some of which extending the functionalities of the Matplotlib library (see 5.2). Figure 1 shows the layered architecture of the tool with visualization and representation layers shown.

The following are the libraries that were considered for being used to generate web-based plots in Python.

3. JAVASCRIPT LIBRARIES (CLIENT-SIDE ONLY)

For the specific function of rendering interactive graphs onto a web page, there are few client-side libraries available. These are libraries that can be used independently of a server, which allows for total flexibility when considering the data source.

3.1 D3 - a JavaScript DOM Library

D3 provides a series of functions for modifying the HTML DOM model of a web page [5]. This can be used to edit SVG objects to display data by using plots such as scatter diagrams [6]. However, the library does not actually have any functions specific to plotting [7]. Although there are many examples available, specifying plots from scratch for specific purposes is preferred by developers over modifying examples.

This requires a lot of time because it is necessary to implement basic features as opposed to using the basic features already present, such as displaying a plot axis.

3.2 Dygraphs - a JavaScript Library for Time-Series Plotting

Dygraphs is a library that was used previously by the ELQA team to create a tool for data visualization [8]. It claims to be flexible. However:

—it lacks built-in functionality so a lot of simple and desirable features have to be implemented such as having a zoom history, displaying a crosshair and displaying hovering data values;

—the data must be in a CSV-like format, so plotting multiple series requires additional data reconfiguration.

4. PYTHON LIBRARIES

4.1 Matplotlib

Matplotlib is based on the plotting capabilities of MATLAB, the scientific programming, analysis and plotting environment and programming language [9, 10]. It does not provide a way to generate interactive plots that can be viewed in the web browser. The closest it gets to this is using basic GUI elements to affect the plots drawn within the confines of a windowing toolkit such as Tkinter [11]. This is not suitable for choosing data sets to be analyzed or exploring the data by using filters. The MPL slider demo shows what these simple GUIs look like [12]. There is sparse documentation related to using Matplotlib as part of a python web server. Although the documentation covers exposing image data in a variety of ways, none of them provide interaction with the rendered plots [13].

5. PYTHON LIBRARIES THAT GENERATE HTML

The data flow here is conceptually different to the JavaScript-only libraries described above. Because these libraries' plots must be generated by a python program, the data must be obtained using python and fed into python script that can create them. These libraries are then capable of either serving the plot on a web server that they create, outputting the HTML to the standard output or simply generating the HTML as a string. This is a powerful property because, using this method, HTML plots can be generated from any data stored in a python variable (provided it is in a format that can be represented in JavaScript).

5.1 Bokeh - a Python Interactive Visualization Library

Bokeh includes a python library that can be used to generate plots using a small set of functions. It is not as powerful as, for example, Matplotlib, which has an extensive set of functions that can be used to create any conceivable plot [9]. However, it does allow for the rendering of the most important types of graph and they are all rendered in line with the modern HTML 5 web standard. This has speed advantages over using the SVG approach (as with Mpld3, below) but some browsers do not support the HTML 5 standard, such as older versions of Internet Explorer.

5.2 Mpld3 - bringing Matplotlib and D3.js together

Mpld3 is very similar to Bokeh in that it too generates HTML from graph-creating functions. There are a few big differences that must be considered when deciding whether to use it instead of Bokeh. The main one being Mpld3 can convert any Matplotlib plot into HTML that can be rendered onto a web page using D3.js, a DOM (Document Object Model) manipulator library, mentioned above [5]. It creates a layer between Matplotlib and D3, acting as the go-between and removing any need for programming using the D3 functions. This is a welcomed advantage because Matplotlib is often used in conjunction with these popular scientific analysis libraries to visualize results [14]. This might not be in favor of its use. However, there are some other significant differences between these libraries that require consideration.

5.3 Methodology

The selection of an appropriate library was highly dependent on the programming language, database layer and the client-server-based architecture. The following features were considered carefully when looking through each of the mentioned libraries: flexibility

of gathering input data and their formats, ability to produce a HTML output, the interactivity of the plots produced, the activity of any open source communities surrounding the library, the documentation and the available support. Below, Table 1 shows a comparison of these features between the presented libraries.

The features were selected based on the study of different cases and simple examples and the requirements of the data tool. They were selected by developing multiple prototypes of the tool. The two best candidates are presented with several comparisons between them.

5.4 Bokeh vs. Mpld3

The use of one of these two python-based libraries satisfied the requirements of the ELQA analytical tool when comparing all solutions. Each of these libraries provide ways of creating plots with a dedicated interface surrounding the plots with a variety of functionality. They can be integrated with the other components of the architecture (including the data analysis libraries SciPy, NumPy and Scikit-learn). The following evaluations were carried out on both libraries in order to select the most appropriate one:

5.4.1 Loading Time. The loading times of the libraries are very important as loading time is one of the largest bottle-necks of web applications. Web sites may be visited frequently and for small amounts of time, so plots could be generated frequently and if necessary, data will be analyzed as frequently. A test was carried out to determine how the amount of time taken to transfer the HTML from a (local) web server to a browser increases with the number of points plotted. For these tests, the Python web framework Django was used to display a plot using each method showing n randomly-sized and -colored, transparent circles scattered randomly in a 1-by-1 square. By using a Python program to measure the time taken to transfer the HTML over the network loop-back interface, data was gathered for a range between 0 and 90000 random circles. The reason this was not done over a network was to reduce interference from network traffic.

The results are shown in Figure 2, which clearly shows Bokeh as being faster than Mpld3. It also shows that both of the libraries exhibit a linear increase of load time with an increase of points rendered.

5.4.2 Rendering Time. The rendering time of anything within a browser depends on many factors, including the browser, the browser's JavaScript Engine, the browser's internal model of the web page, etc. Therefore, it is not easy to say which library truly renders at a faster rate in general. However, what can be compared are the parts of the HTML standard that are used by each library. Namely, whilst Bokeh uses the modern 'canvas' tag, Mpld3 uses the 'svg' tag to render onto a web page.

The canvas tag and the associated JavaScript standard provide a way to draw immediate mode graphics onto a web page and the result is an image [15]. On the other hand, the 'svg' tag uses SVG (Scalable Vector Graphics) to maintain a graph scene of shapes with various styles that determine their exact appearance. Both approaches are appropriate for developing the tool, but the following observations were in favor of selecting Bokeh as the data visualization component of the analysis tool.

5.4.3 Evaluated Features. Both Mpld3 and Bokeh represent very appropriate choices for developing the visualization component of the analytical tool. Both are developed and supported by open source communities (see table 1). Both support interaction within the plots and the rendering of HTML code onto a web interface. Mpld3 is capable of making Matplotlib plots ready to display on a website. Bokeh does not fully support this feature; there is the suggestion of Bokeh interacting with MPL better in the future [16]. Bokeh coding does not differ significantly from Matplotlib. The interactive tool API provided by Bokeh is better than that which is provided by Mpld3 due to its limited nature and lack of support (as it requires customized JavaScript).

It is also worth mentioning that from a user's point of view, Bokeh offers better data interaction options. It supports the selection of individual points or groups of points in a variety of ways, allowing for further mining by updating other plots in response to interaction. It also supports "Actions" which allow tools on a plot to cause the browser to open a new tab or run customized JavaScript within the user's browser.

Comparing the two open source projects surrounding these two libraries, it is clear that Bokeh is more actively developed than Mpld3. This is thanks to the company currently developing it (Continuum Analytics) and a group of contributors from the open source community who fix bugs and implement features in their spare time [17]. As noted in May 2015,the Github page for Mpld3 had 15 less contributors and also 34 less merge requests in the last month (Mpld3 had one of each) [18].

Last but not least, the documentation of Bokeh is not evolved enough such as to be able to use all functionality without understanding the source-code in-depth. In contrast, Mpld3 documentation was not detailed enough surrounding it's ability to interface with customized JavaScript, necessary for interactive plots. However, Mpld3 consists mostly of a wrapper architecture around MPL, which has extensive documentation. The examples provided in the Bokeh documentation were numerous. However the more sophisticated examples lacked detailed documentation, so it was necessary to study the source-code in order to achieve the plots required. In terms of rendering there is no significant differences between the libraries, however the ability to develop an application within a Bokeh server enables more convenient development of prototypes than when programming using Mpld3. This is also one of the main reasons that the Bokeh library was selected over Mpld3.

6. RESULTS AND DISCUSSION

Presented in this article are some of the issues experienced when integrating various parts of an open source software stack for building a scientific tool for data analysis in the TE-MPE-EE section of CERN. General requirements for the tool have been presented and the selection of the suitable components for visualization of the analyzed data has been focused on.

The evaluation process has been described for selecting the visualization component for an analysis tool. Bokeh has been shown to be the best choice out of all of the available options. This is due to a number of factors that have been covered and experienced whilst searching for an appropriate solution. One of the main advantages discovered during development was the ability of Bokeh to act as a web server in order to expose a fully-fledged data visualization and analysis tool to multiple users. This was a feature that was discovered and was efficiently used for fast prototyping.

What's more is that Bokeh has a growing community of developers with vested interests in its success. This has lead to a steady stream of contributions to the open source project that is managed by its creating company, ContinuumIO [17]. In fact, during the creation

of the analysis tool, some contributions were made by the authors of this paper which were accepted by the Bokeh team. These were critically needed in order to create the tool itself.

This however does not mean that Mpld3 should be underestimated as a visualization library. For the requirements of the TE-MPE-EE section, Bokeh's server-client architecture and data interaction concept offers the most appropriate solution.

Through researching the capabilities of the presented libraries and developed prototypes, an architecture of the analytical tool has been defined in detail.

This architecture is well suited for the requirements of the analysis tool and satisfaction of future use-cases defined by the ELQA group at CERN.

7. ACKNOWLEDGEMENT

The analytical tool is still a prototype being used to analyze test data taken by the ELQA team. The objective is to compare measurements between the two hardware commissioning periods of the LHC that occurred before the first and the second runs. The authors would like to thank the entire ELQA team at CERN for their persistent help throughout the ongoing development process.

8. REFERENCES

[1] Guido van Rossum. Python reference manual. Report CS-R9525, April 1995.

[2] K. Jarrod Millman and Michael Aivazis. Python for scientists and engineers. *Computing in Science Engineering*, 13(2):9–12, March 2011.

[3] Pierre de Buyl and Nelle Varoquaux. Proceedings of the 7th european conference on python in science (euroscipy 2014). *CoRR*, abs/1412.7030, 2014.

[4] Fernando Prez and Brian E. Granger. Ipython: A system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29, 2007.

[5] Michael Bostock. D3. js. *Data Driven Documents*, 2012.

[6] Enamul Hoque and Giuseppe Carenini. Convisit: Interactive topic modeling for exploring asynchronous online conversations. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 169–180, New York, NY, USA, 2015. ACM.

[7] Eugene Wu, Leilani Battle, and Samuel R. Madden. The case for data visualization management systems: Vision paper. *Proc. VLDB Endow.*, 7(10):903–906, June 2014.

[8] Dan Vanderkam. Dygraphs javascript charting library, 2006.

[9] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, May-Jun 2007.

[10] MATLAB Users Guide. The mathworks. *Inc., Natick, MA*, 5:333, 1998.

[11] Phil Hughes. Python and tkinter programming. *Linux J.*, 2000(77es), September 2000.

[12] Matplotlib.org. "screenshots matplotlib 1.4.2 documentation". "http://matplotlib.org/users/screenshots.html#slider-demo".

[13] Matplotlib.org. "how-to matplotlib 1.4.2 documentation". "http://matplotlib.org/faq/howto_faq.html#matplotlib-with-django".

[14] Wes McKinney. *"Python for data analysis: Data wrangling with Pandas, NumPy, and IPython"*. " O'Reilly Media, Inc.", 2012.

[15] "Microsoft". "svg vs. canvas: how to choose (windows)". https://msdn.microsoft.com/library/gg193983(v=vs.85) .aspx#pixel_manipultation.

[16] "Continuum Analytics". "compatibility layers bokeh 0.8.2 documentation". http://bokeh.pydata.org/en/latest/docs/reference/compat.html.

[17] "Continuum Analytics". bokeh/bokeh. https://github.com/bokeh/bokeh.

[18] "Jake Vanderplas". jakevdp/mpld3. https://github.com/jakevdp/mpld3.

9. APPENDIX

Table 1. A comparison of the available visualization libraries

Package	Produces HTML	Graphing interface	Flexible input data	Active open source community	Documentation
Bokeh	Yes	Yes	Yes	Yes	Not extensive
Mpld3	Yes	Yes	Yes	No	Lacking JavaScript documentation
Dygraphs	Yes	Yes	No	-	Good
D3	Yes	No	Yes	-	Excellent
Mpl	No	Yes	No	-	Excellent

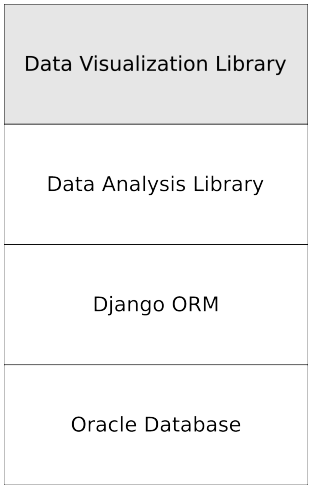

Fig. 1. The layered architecture of the ELQA data analysis tool

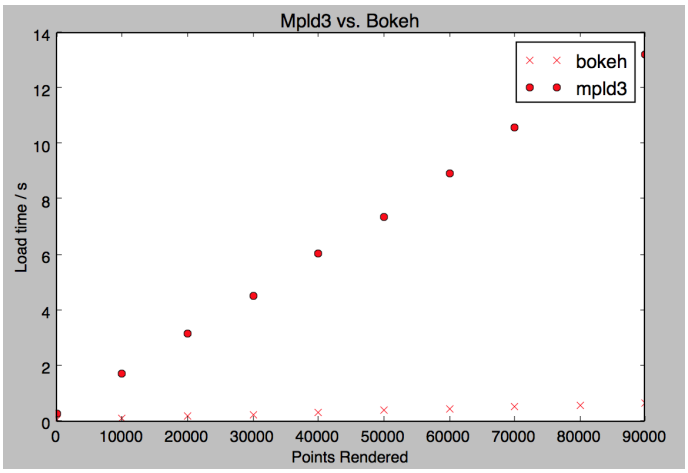

Fig. 2. Both libraries have a linear increase of time taken over an increase of points rendered.