Synthetic Chart Image Generator: An application for generating chart image datasets

Rafael Akiyama[1], Tiago Araújo[1], Paulo Chagas[1], Brunelli Miranda[1], Carlos Santos[1], Jefferson Morais[1], Bianchi Meiguins[1]

1Universidade Federal do Pará
Belém, Pará
{rafaeldaisuke2, tiagodavi70, paulo.robertojr100, brunelli.miranda, gustavo.cbcc, jeffersonmorais, bianchi.serique}@gmail.com

Abstract—**The scarcity of chart images public datasets and its generators makes difficult the comparative studies between works in areas such as information visualization, usability, computer vision, and machine learning. Therefore, this paper presents a Web tool for generating data chart images called Synthetic Chart Image Generator (SCIG). The tool uses VEGA declarative language, and the chart features are fully parameterizable for every eleven different classes, as well as image quantity and its resolution. Additionally, the charts are constructed from synthetic data, randomly generated by probability distributions functions, and rendered in PNG format. Finally, this paper presents a performance test of the chart images generation.**

Keywords-chart image generator, image dataset, synthetic data generator.

I. INTRODUCTION

Since the development of the Information Visualization (InfoVis) area and the need to understand a vast amount of data, several novel types of charts have appeared [1], such as Sunburst, Parallel Coordinates, and Treemap. The amount of digitally available data to be visualized foster new visualizations and analysis techniques every day [2].

A data visualization essentially needs data to be useful. However, there are specific domains or areas where collecting data may be difficult, due to conditions such as privacy, scarcity or lack of variability of data, which may hinder access to these datasets.

A solution adopted by several works is generating their datasets, mainly for comparison and evaluation of techniques in many areas, such as information visualization, usability tests, security, renewable resources, classification of 3D objects [3], sign language recognition [4], and agriculture [5] [6].

It may be more difficult to find these images publicly, considering only synthetic image datasets and the specific analysis domain, making a comparative analysis of previous work more difficult. The InfoVis area also has its demands for generating data chart images such as large-scale usability evaluation of charts, machine-learning solutions for chart image classification, and automatic data extraction from charts.

Also, there is no standard dataset of chart images, such as MNIST [7] for digit classification, or ImageNet [8] for classification of natural images. Another problem is the difficulty in collecting certain chart types from the internet. An example is the arc diagram, which has far fewer results in search of Google Images when compared to a more popular chart, such as the bar chart, which may imply in an unbalanced dataset.

Thus, this work focuses on an automatic and parameterizable generation of a large amount of data chart images of several different chart types. We present the Synthetic Chart Image Generator (SCIG) tool, which helps on generating chart image datasets of up to 11 chart classes: Arc Diagram, Area chart, Line chart, Parallel Coordinates, Pie chart, Reorderable Matrix, Scatter Plot, Scatter Plot Matrix, Sunburst, and Treemap. The charts are constructed from synthetic data randomly generated using probability distributions and rendered in PNG format images through the Vega visualization grammar [9].

Considering the importance of a diversified image dataset, this work proposes a parametrized tool, which the user can configure several visual elements (such as subtitles, titles, axis labels, item labels, and colors) and the underlying data of each chart that can vary at the user's decision. SCIG has a simple GUI (Graphical User Interface), allowing the choice of particular parameters for each type of chart, and the automatic generation of chart images in the amount specified by the user. There is no need for the user to have any programming skills for generating the chart images.

The remaining sections of this paper are organized as follows: Section II discusses related works, Section III presents the SCIG, its process of images generation, architecture and interface. Section IV shows the performance considerations about chart image generation and Section V concludes this work.

II. RELATED WORKS

There are many generators of data chart images on the Web, but they only allow the generation of a single chart. The selected related works are focused on tools, APIs, and grammars to be applied for parameterizable generation of chart images, emphasizing characteristics and requirements such as diversity of parameters, a variety of visualizations, and the necessity of programming by the user.

D3 [10] is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. D3 helps visualize data using HTML, SVG, and CSS. D3's emphasis on web standards gives the full capabilities to combining powerful visualization components and a data-driven approach to DOM manipulation.

Vega 3.0 [9] is a visualization grammar, a declarative language for creating, saving, and sharing interactive visualization designs. Compared to coding directly, Vega can be learned by non-coders quite easily. At the same time, Vega is rich enough to be expressive, and not just easy to use. We chose Vega as our chart generator, as it is robust to ease the creation of vast amounts of different types of charts.

Vega-lite 2.0 [11] is a high-level visualization grammar. It provides a concise JSON syntax for supporting rapid generation of visualizations to support analysis and is built on top of Vega. Its primary goal is to provide simple interfaces to user interactions within generated charts.

Polestar [12] is a web-based visualization specification interface. It is possible to associate attributes of an already loaded database with visual variables of specific charts such as scatter plot points, bars, and lines, using drag-and-drop interactions. Analysts can rapidly generate visualizations as part of the data exploration process.

Protovis [13] is a toolkit for building visualizations from the composition of simple and hierarchically organized graphical primitives. The main idea of their proposal is to allow the flexible and straightforward manipulation of the visual elements of the charts. Since the charts are described hierarchically, it is possible to control the construction of these specific visual elements for each chart type (such as lines, and bars) as well as more specific data of each element more internally in the hierarchy.

Flare [14] is an ActionScript library for creating visualizations that can be executed on Adobe Flash Player. It supports various types of data charts, from bar and pie charts to more complex and interactive visualizations. A remarkable feature is the use of animations to understand the relationship between the data. Some programming skills are required for visualization development. The library does not provide automated chart generation in large numbers since there is a need for a database as input.

Lyra [15] is an interactive environment for building customizable visualizations from drag-and-drop interactions. In this application, the user can associate the attributes of some dataset with specific visual elements of certain types of charts. The created visualizations are described using Vega. However, there is no option to generate multiple charts automatically by varying the configuration parameters.

Voyager [16] is a system that integrates manual and automatic chart generation. Voyager allows the user to choose a set of database attributes and relate them to specific visual features. In this way, as soon as a data feature is configured through drag-and-drop interactions, a set of charts is suggested. This application requires a database in such way that no programming skill is required since most of the generated charts are based on the configuration of axes, varying shape and size color. This approach has proven to be efficient in cases where the dataset is unknown. Thus, instead of the user having to choose attributes that a priori does not know about, the tool itself generates a set of charts as a suggestion for the user.

Figure 1, adapted from [17], displays the levels in a way that the top layer shows languages and tools that are built on D3. In the low level, the D3 is used as a visualization kernel for all applications. Then in the low-mid level is the Vega visualization grammar. A visualization grammar allows a visualization be expressed in a simplified model, as the drawing is executed separately. Upwards in the mid-level is the Vega-lite, which is a simplified version of Vega and serves as a visual analysis grammar. In the high level are Polestar and Voyager built on top of Vega-lite, where both applications perform data exploration with GUI. On this level, the user has control of the data and the charts generation, but one need to know all the parameters of one visualization to use the tool.

The Lyra is a full GUI design environment built on top of Vega. With this kind of environment, one can directly encode visual items to data, customizing each detail of the chart in a visual manner, without the need of a deep understanding of the charts. Our SCIG application is also built on top of Vega and has a GUI interface to generate visualizations without coding.

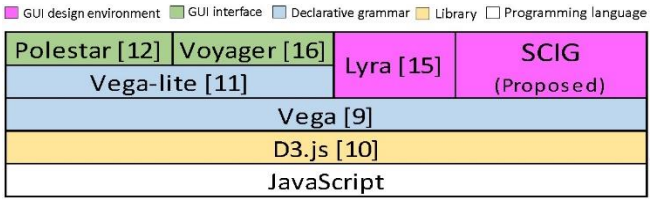

Figure 1. The levels of languages and tools built on D3. Adapted from [17]

Furthermore, it is important to highlight in these works the importance of using a graphical interface to configure visualization parameters, helping the user on creating their visualizations without the direct need of a programming language, using automatic generated JSONs for the charts data. The tool developed in this paper follows this approach as well.

III. SYNTHETIC CHART IMAGE GENERATOR (SCIG)

SCIG is a tool that generates datasets of chart images presenting a graphical interface that allows the user to configure several parameters for each chart type. SCIG currently can generate images of 11 classes of 2D charts: Arc Diagram, Area, Bar, Line, Parallel Coordinates, Pie, Reorderable Matrix, Scatter Plot, Scatter Plot Matrix, Sunburst, and Treemap [1] [6] (see Figure 2).

The proposed application was built on Vega visualization grammar [9], allowing highly parametrized charts. Vega uses a declarative JSON format and defines reusable chart components. SCIG visualizations are exported as Vega JSON files and PNG images. Since the charts are described in JSON, all the data and visualization specifications can be recovered for later analysis.

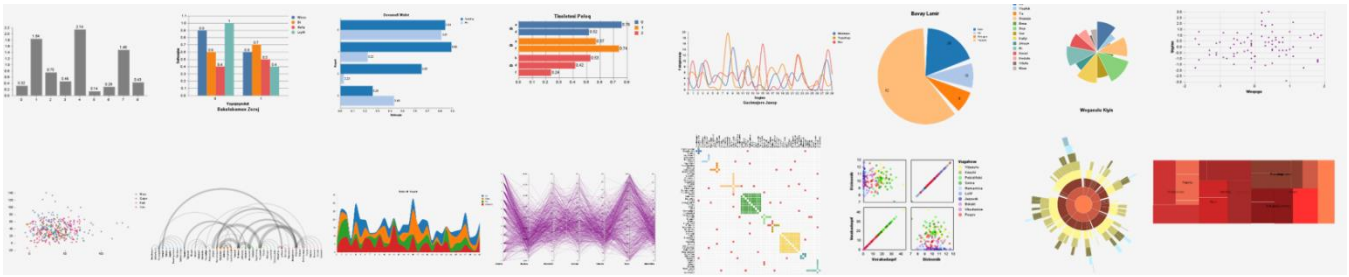

Figure 2. Charts generated by SCIG.

A. Chart Image Generation

Firstly, the chart image generation process does not need a prior dataset as input. SCIG uses probability distributions (such as Chi-Square, Normal, Uniform, and Poisson) defined by the user with the intervals and specific parameters for each probability function, for data generation of each chart type. The application can be divided into two generation modes, as it can be seen in Figure 3: Automatic and manual.

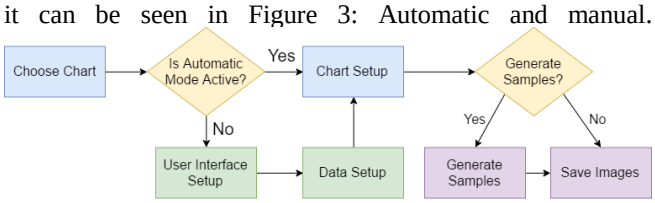

Figure 3. Automatic and Manual generation modes.

The first one is the automatic mode, which generates chart images randomly varying the specific features of each chart, such as specific visual elements (e.g., bars, lines, and points), color, subtitle position, title position, and the data generation parameters. In this way, there is no need for the user to set all the parameters for chart type and data generation.

In this mode, the data generation is done automatically by deciding how much data instances will be created, selecting randomly from a range of predefined values. Afterward, the probability distribution will be randomly chosen, and then, each data value will be generated following the selected probability function. The function parameters will be randomly selected within a range of values, such as for average and standard deviation on Normal distribution.

Pre-configured value ranges in automatic mode have been chosen empirically so that the generated charts resemble the graphics images on the internet accessed via Google Images.

In the manual generation mode, the user can define the parameters of the charts by hand, such as the amount and interval of data, type of probability distribution, and the respective parameters of that distribution. Regardless of the generation mode, the user can modify other chart parameters, which are different for each chart type, such as subtitle, title, axis labels, and item values. All texts for chart labels are randomly generated alternating syllables and vowels. The user decides the quantity and resolution of the graphics images, and all images have the PNG format.

B. Architecture Application

SCIG generates chart image datasets as a cloud service through the Software as a Service (SaaS) concept. In general, the user accesses a web client to setup the chart image generation, which later communicates with the application server. The server receives and interprets the parameters for generating the dataset. Finally, a Uniform Resource Identifier (URI) returns for the web client to download the dataset, as shown in Figure 4.

NodeJS was chosen for implementation in a way that the communication is performed by WebSockets, using JSON as the model for this communication. As additional features, the client side has a module for checking the parameters for each chart type, which are sent to the server. In case of erroneous or missing values, the application warns the user of the need for correction. There is also an intermediary step that is the generation of samples. These samples are generated and sent to the client side so the user can verify whether the chart models are suitable for generating the entire dataset.

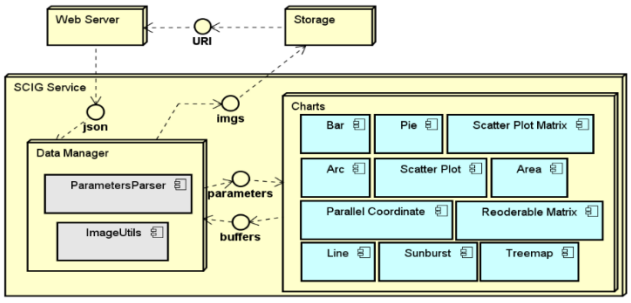

Figure 4. Architecture of SCIG.

C. The SCIG Interface

The SCIG interface is organized into 11 tabs, one for each chart type, as shown in Figure 5 – A. Each tab is composed of a set of parameterizable features for each chart type generation. Some chart types have variations, such as the bar chart, which has four types of variations: Vertical Bar, Vertical Grouped Bar, Horizontal Grouped Bar and Nested Bar. These variations can be selected in the chart design area (Fig 5 - B) where each design choice has its configuration

screen, which is displayed in the Custom Configuration area (Fig 5 - CUSTOM).

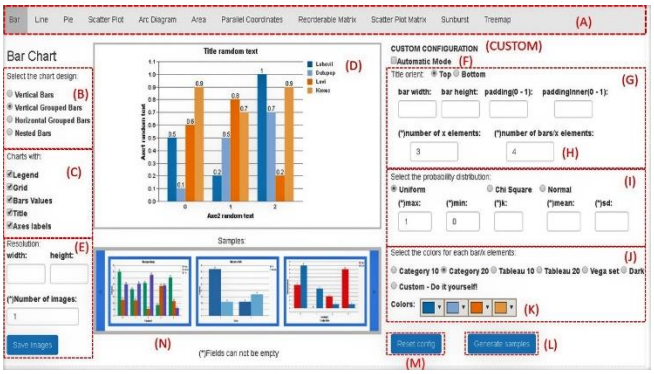

Figure 5. Vertical Grouped Bar chart Interface.

The user can choose which elements the bar chart types can have. For example, for bar charts, elements such as subtitle, grid, value labels in the bars, title and label of the axes (Fig 5 - C) can be configured. Figure 5 - D is the display area for the selected chart type, and is where the elements inserted in the graph are displayed. The elements that can be inserted in the chart can change depending on the chart type selected and are independent of the automatic mode (Fig 5 - F).

The tool allows the user to set the resolution of the images and the number of images to be generated. The "Save Images" button is responsible for generating and saving images in PNG format using all user-defined settings (Fig 5 - E).

The automatic mode (Fig 5 - F), as explained previously, disables all other items belonging to the Custom Configuration area (Fig 5 - CUSTOM), performing random selection for several parameters. Figure 5 - G illustrates the specific configurations of the chart types, which will change its layout depending on the chart type. The data values are generated by the probability distribution chosen by the user (Fig 5 - I).

The user can choose the colors that will be used in the chart, with some predefined color palettes available (Fig 5 - J). However, there is also the option to choose each color using a color picker (Fig 5 - K). The quantity of colors that can be defined depends on the number of elements for each chart type (Fig 5 - H). The application has a button (Fig 5 - M) that returns all settings from the Custom Configuration area (Fig 5 - CUSTOM) to the initial state, not changing the generation mode (automatic or not).

Finally, the "Generate Samples" button (Fig 5 - L) generate samples so that the user can have a preview of their settings for the chart images, always generating 18 images which are displayed on the carousel (Fig 5 - N) and the gallery just below on the interface. The user can click on these images that are in the carousel or gallery, which work as a link to view them in a larger size in the format of slides. It is important to note that the "Generate Samples" button also works when the automatic mode is activated.

Figure 6 shows the Configuration tab of the Scatter Plot Matrix chart to highlight the different interfaces that each chart requires.

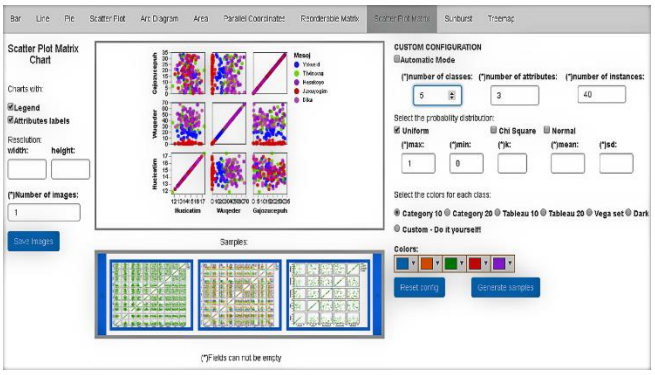

Figure 6. Scatter Plot Matrix chart Interface.

IV. PERFORMANCE CONSIDERATIONS

We present some data about the performance of the developed tool (SCIG) about the generation of chart images, observing the number of images, chart type and time.

We run a test to evaluate the time taken for generating a determined number of charts images by type. The test ran on a machine with the following configuration: Processor Intel I7 7ª Gen, SSD HD, 16 GB RAM and Video Card GTX 1060 GB.

This test used, for each chart type, four groups varying the number of images: 100, 500, 1000, and 2500 images of 1200x800 resolution. The disk space used for these groups vary between 70MB to 1.5GB. We took the average time in 20 runs for each group to measure the time. The results are in Figure 7.

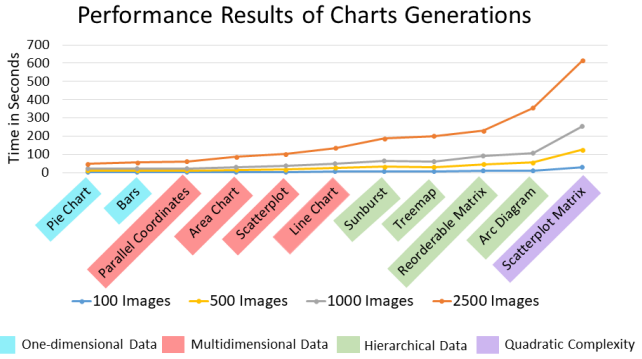

Figure 7. Performance Results of Images generation (Time x Chart Type).

The Scatter Plot Matrix is the chart type that demands more time to generate its images. It has a quadratic complexity based on the numbers of data attributes, as it creates one Scatter Plot for each possible pair of attributes, which explains why its time is higher than the others time.

Next, the charts that represent hierarchical data like Treemap and Arc Diagram (highlighted in green in Figure 7) show generation time bigger than traditional charts like Bar chart and Scatter Plot. For instance, the generation of 2500 Arc Diagrams took 352 seconds on average, on the other hand, for the same parameters, the Scatter Plot took 102 seconds on average.

Finally, the charts highlighted in red, such as parallel coordinates and line, showed longer execution time compared to charts highlighted in blue. Because they present greater data dimensionality, it needs more time to generate the synthetic data and then create the images.

V. Final Remarks and Future Works

In this paper, we present an application (SCIG) capable of generating synthetic chart images in great quantity and different resolutions. The main motivation was the scarcity of chart image datasets as well as tools for generating large amounts of chart images without the need of programming skills. The application has 11 classes of different charts, fully parameterizable through a web interface. The generation of the chart images follows the concept of Software as Service, where a server application is responsible for generating the images according to the parameters defined by the user.

Some considerations have been highlighted about the time to generate the images. Scatter Plot Matrix take a higher time to generate charts, as it generates a chart for each pair of attributes. The hierarchical data chart images take more time than multidimensional charts. Chart images of data with multidimensional graphical elements take more time than one-dimensional charts. Finally, in general, the application presents good performance.

As future work, it is intended to carry out statistical evaluations on the chart images generated, increase the number of chart types, and make the application publicly available on our research group site. Additionally, the profile creation of datasets will be saved so that these profiles can be replicated in other scenarios. Furthermore, image and data noise parameters will be included for specific application scenarios such as pattern recognition on images.

References

[1] Muzammil Khan, Sarwar Shah Khan. Data and information visualization methods, and interactive mechanisms: A survey. In International Journal of Computer Applications. 1-14. November 2011.

[2] Geral Franciscani Jr, et al. An annotation process for data visualization techniques. In Proceedings of International Conference on Data Analytics. 2014.

[3] Artem Rozantsev, Vincent Lepetit and Pascal Fua. On rendering synthetic images for training an object detector. In Computer Vision and Image Understanding, Volume 137, 24-37. August 2015.

[4] Feng Jiang, et al. Synthetic data generation technique in Signer-independent sign language recognition. In Pattern Recognition Letters, Volume 30, Issue 5, 513-524. April 2009.

[5] M. Di Cicco, et al. Automatic model based dataset generation for fast and accurate crop and weeds detection. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 5188-5195. Sept. 2017.

[6] R. Barth, et al. Synthetic bootstrapping of convolutional neural networks for semantic plant part segmentation. In Computers and Electronics in Agriculture. 2017.

[7] L. Deng, The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. In IEEE Signal Processing Magazine, vol. 29, no. 6, 141-142, Nov. 2012.

[8] Jia Deng, et al. Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2009. IEEE, 248-255. June 2009.

[9] Vega: Visualization Grammar. https://vega.github.io/vega/. August 2017.

[10] M. Bostock, V. Ogievetsky and J. Heer. D³ Data-Driven Documents. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, 2301-2309. Dec. 2011.

[11] A. Satyanarayan, D. Moritz, K. Wongsuphasawat and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, 341-350. Jan. 2017.

[12] Polestar. http://vega.github.io/polestar/. March 2018.

[13] M. Bostock and J. Heer. Protovis: A Graphical Toolkit for Visualization. In IEEE Transactions on Visualization and Computer Graphics, vol. 15, no. 6, 1121-1128, Nov.-Dec. 2009.

[14] Flare. https://github.com/prefuse/Flare. March 2018.

[15] Arvind Satyanarayan, Jeffrey Heer. Lyra: An Interactive Visualization Design Environment. Computer Graphics Forum (Proc. EuroVis). 2014.

[16] Kanit Wongsuphasawat, et al. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis). 2016.

[17] Éric Marty. The D3/Vega "Stack". https://blog.ericmarty.com/the-d3-/-vega-stack. March 2018.