WUSS 2019 - Paper Number 86

Modern Data Visualization and Presentation: for SAS users

Jay Ahn, PhD, Washington State DOC, Tumwater, 7345 Linderson Way SW

ABSTRACT

Today's business environment has become increasingly unpredictable, characterized by Volatility, Uncertainty, Complexity and Ambiguousness (VUCA), especially after the financial crisis in 2009, due to accelerating, continuous innovations in our environment. As results, we often misread trends that have a modest beginning but have an exponential growth path in the long run. Fortunately, we have more data than ever to help us in making better decisions in the new VUCA world. In fact, data is like coming through a fire hose every day and we need to be able to find meanings in this sea of information quickly and communicate them using a powerful data visualization that is easy to understand and leaves a lasting impact on viewers. Therefore, in this paper, I will show a modern data visualization and presentation, utilizing HTML5 and SAS. Visualizing data meaning into something people can see will be most effective way of communicating and sharing the meaning. I will show how to prepare and deliver SAS data for dynamic HTML-based visualization. Since we are leveraging modern browsers for front-end of data visualization, the visualization results can be communicated to target audience anytime anywhere on demand. Also, users do not need any servers or admin rights to produce and deliver these dynamic data visualizations. Users will need only a web browser to consume the data visualization.

INTRODUCTION

At the beginning of Internet evolution, HTML enabled us to create web pages and share the XML data over Internet. In combination with back-end servers connected over Internet, HTML has become ever more powerful in sharing and communicating information. Later, JSON (JavaScript Object Notation) data format became the most popular data format on the web. I will use the HTML5 Canvas and RGraph, a JavaScript Library, to draw dynamic and interactive charts using the JSON data files generated by SAS. Then I will present those charts using Reveal.JS, another JavaScript library, to navigate those charts.

First, I will start with a demo of data visualization sample using HTML and SAS.

Second, I will show you how to convert SAS datasets to JSON objects using SAS PROC JSON and SAS MACRO. I will show you how to reference these JSON files in a HTML file so that those JSON objects are automatically loaded so that the data is available for JavaScript to render the charts. Although I used the SAS for generating the JSON files here, any tools, even a simple text editor like Note, could have been used to generate the JSON file. However, if you want to automate the updates of hundreds of data visualization, you may want use a tool like SAS or Python. By scheduling the execution of the SAS code or Python code right after the refresh or update of data warehouse or analysis-ready databases, the entire data visualization and its update can be automated and no manual maintenance will be necessary.

Third, I will show you to edit the HTML page so that users can control specifics of data visualizations on the web. Using some HTML attributes, users can control types of chart (pie, bar, rose, line, and etc.), styles of chart (stacked, grouped, nested and etc.), titles of chart, types of data (counts vs percentage), and self-narration based on text to speech.

Fourth, I will show some interactive features of HTML-based presentation that allows effective interactions with audience vs drawing or highlighting or any media (audios, videos, and image).

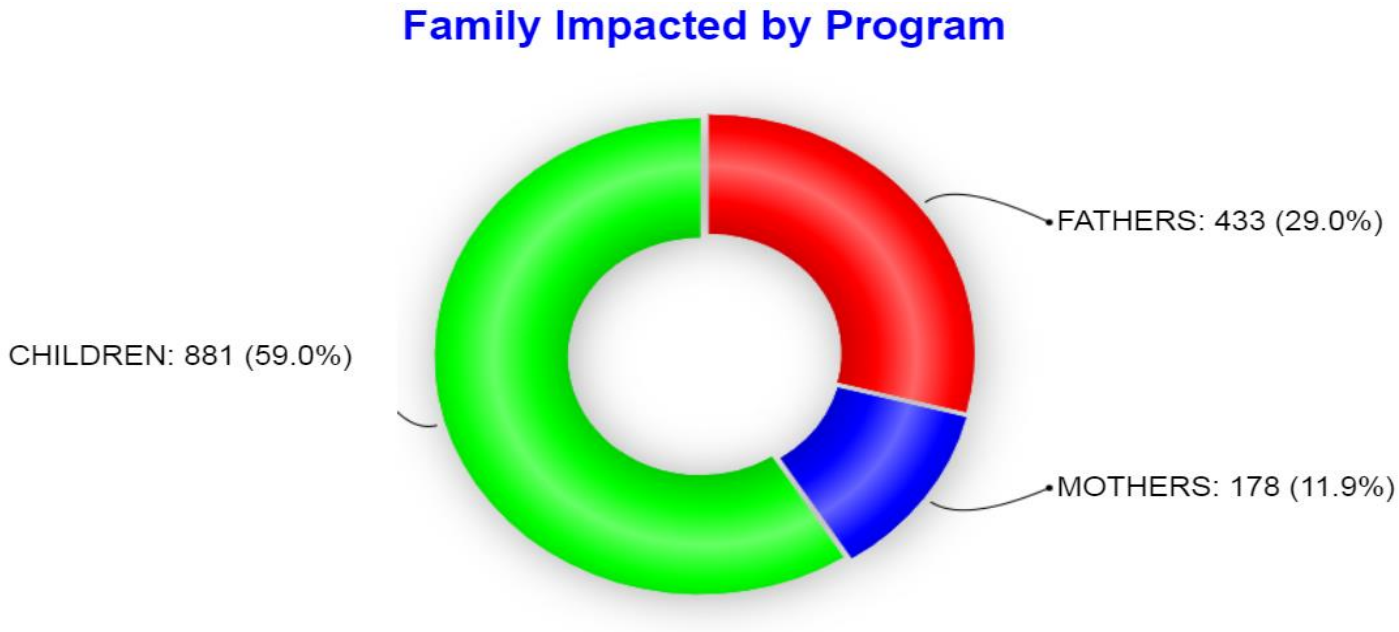
In summary, this paper discuss how to use SAS to prepare a JSON data file and create a powerful data visualization using the HTML 5 canvas. Also I will show how to combine these powerful charts with a HTML based presentation library such as Reveal.JS. By the end of the course, users will know how to move beyond the static power point slide or excel spreadsheets, and visually communicate your data story to the world without any boundary.

PREVIEW OF SOME DATA VIUSUALIZATION OUTCOME

Here are some previews of sample outcomes of data visualization. These charts are for the demonstration purpose only.

Sample Data Visualization with Donut Chart

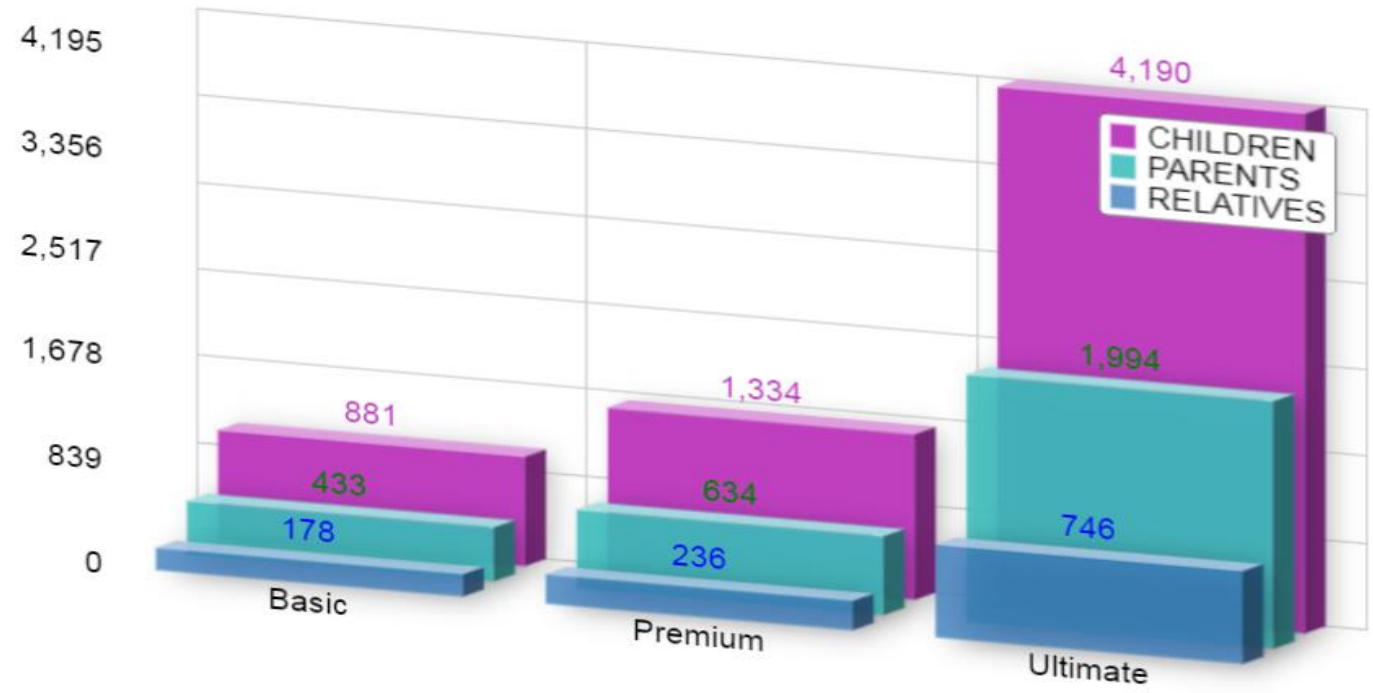
The following sample chart shows the relative compositions of people served by a fictitious sample program.



Display 1. People Impacted by Program

The following sample chart shows the relative composition of people served by three different programs.

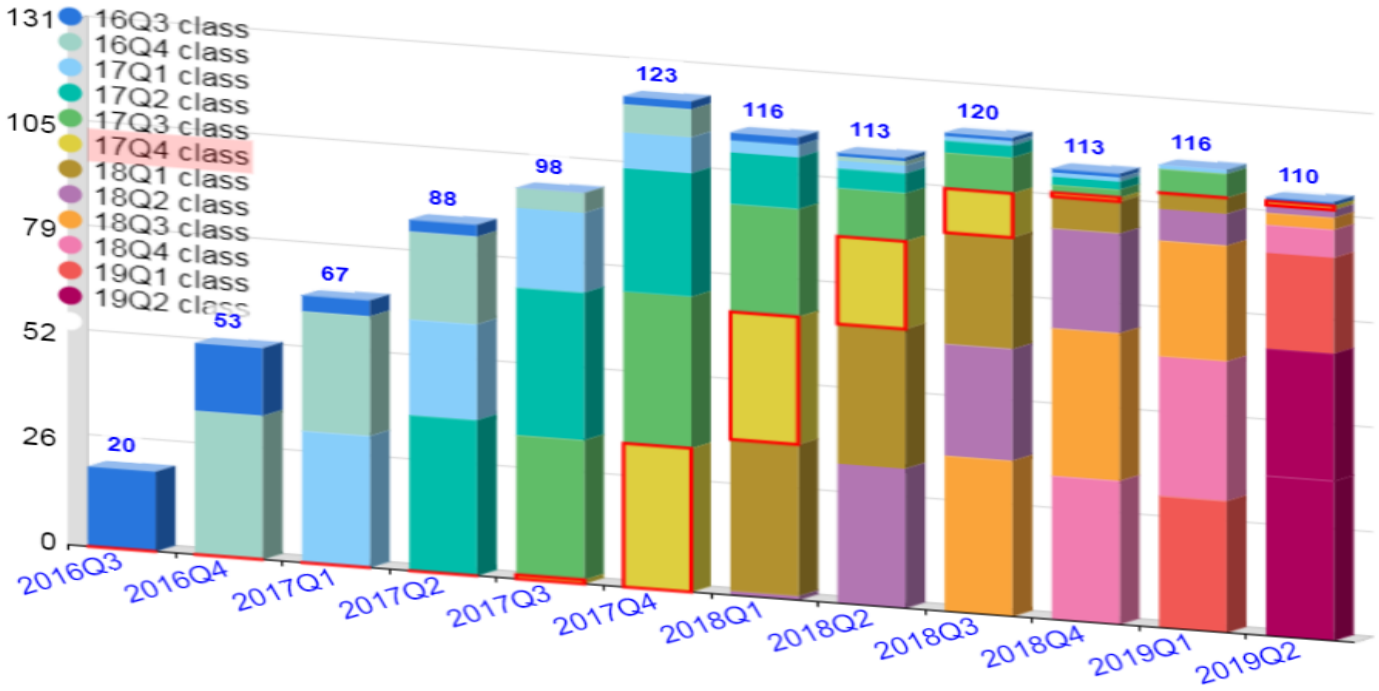
Sample Data Visualization with Layered Bar Chart



Display 2. People impacted across three comparative programs

The following sample chart shows the gradual exits of class cohorts. A new class cohort started in each quarter. The HTML-based data visualization allows a rich user interactions with the chart. By pressing a cohort label, you can visualize the gradual class attendance of the selected cohort over time.

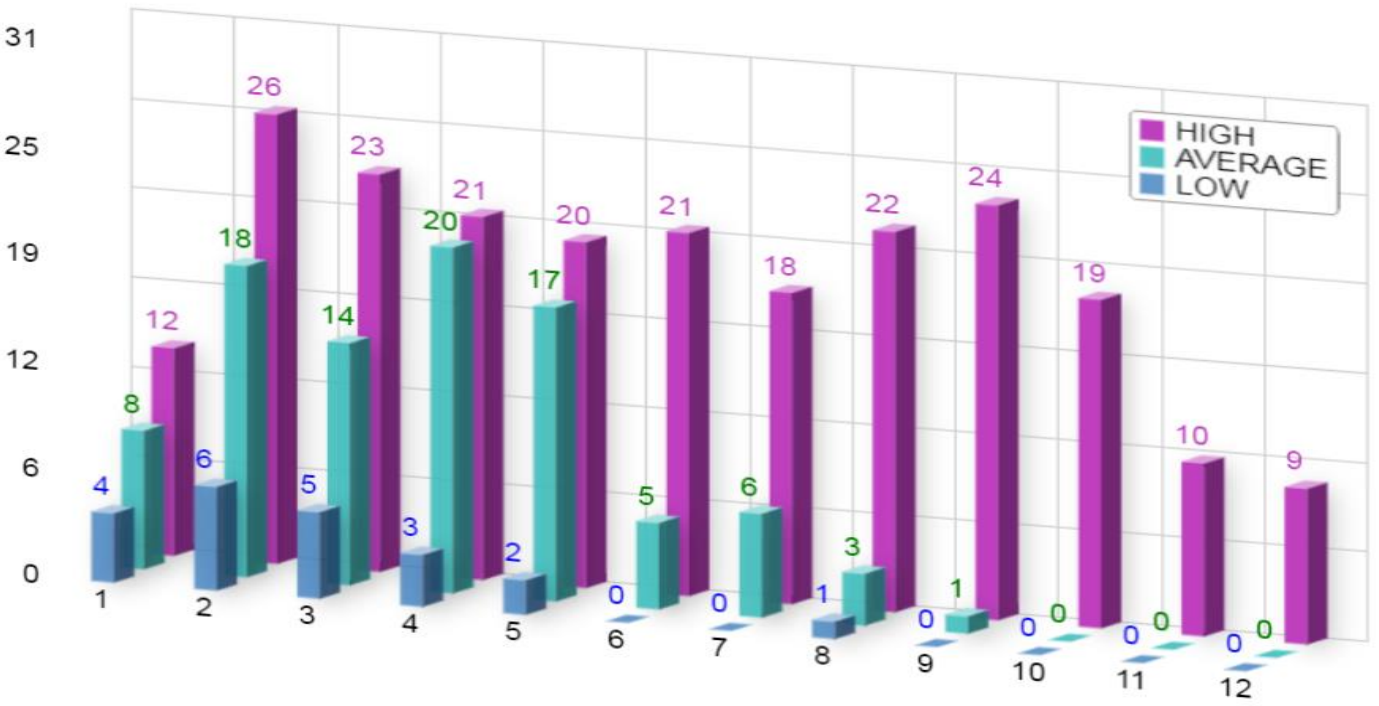
Sample Data Visualization with Stacked Bar Chat



Display 3. Class Attendance Dynamics for Cohort 1 through 12

The following sample chart shows three types of participants in a workshop.

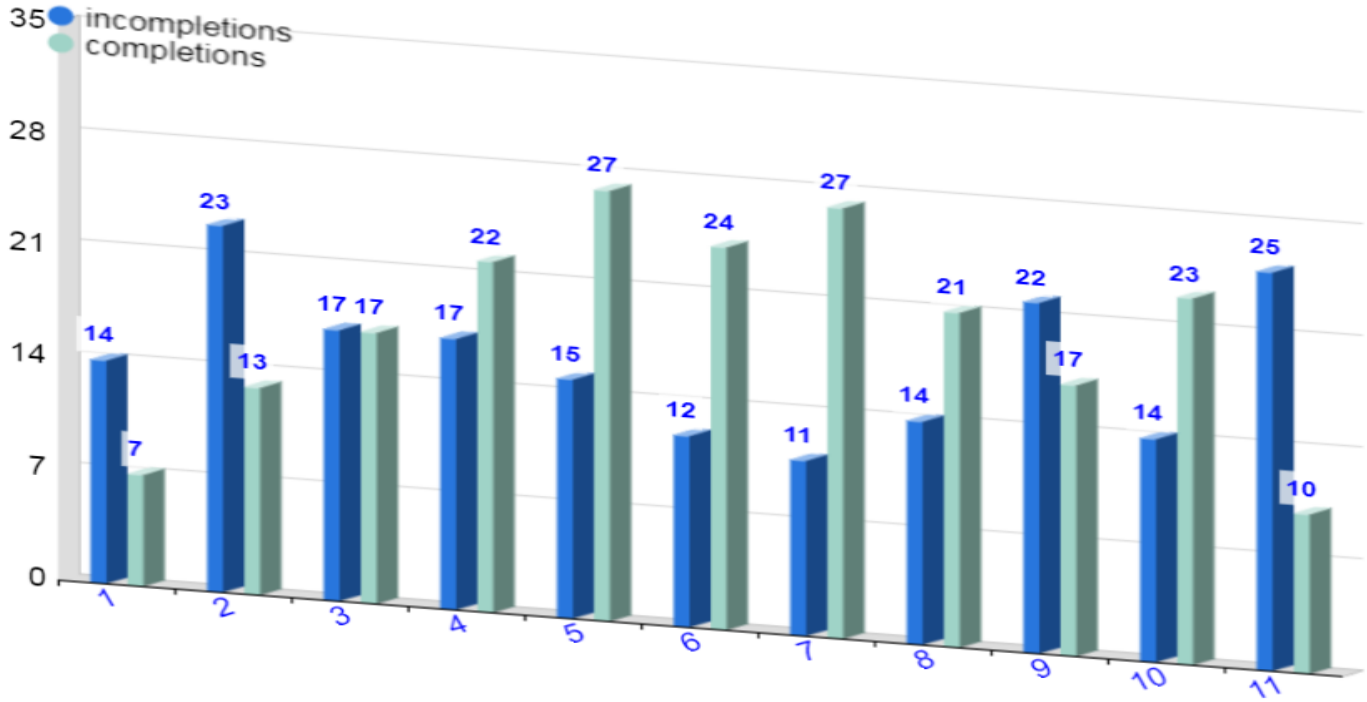
Sample Data Visualization with Comparative Bar Chart



Display 4. Performance Across Cohort 1 through 12

Sample Data Visualization with Contrasting Bar Charts

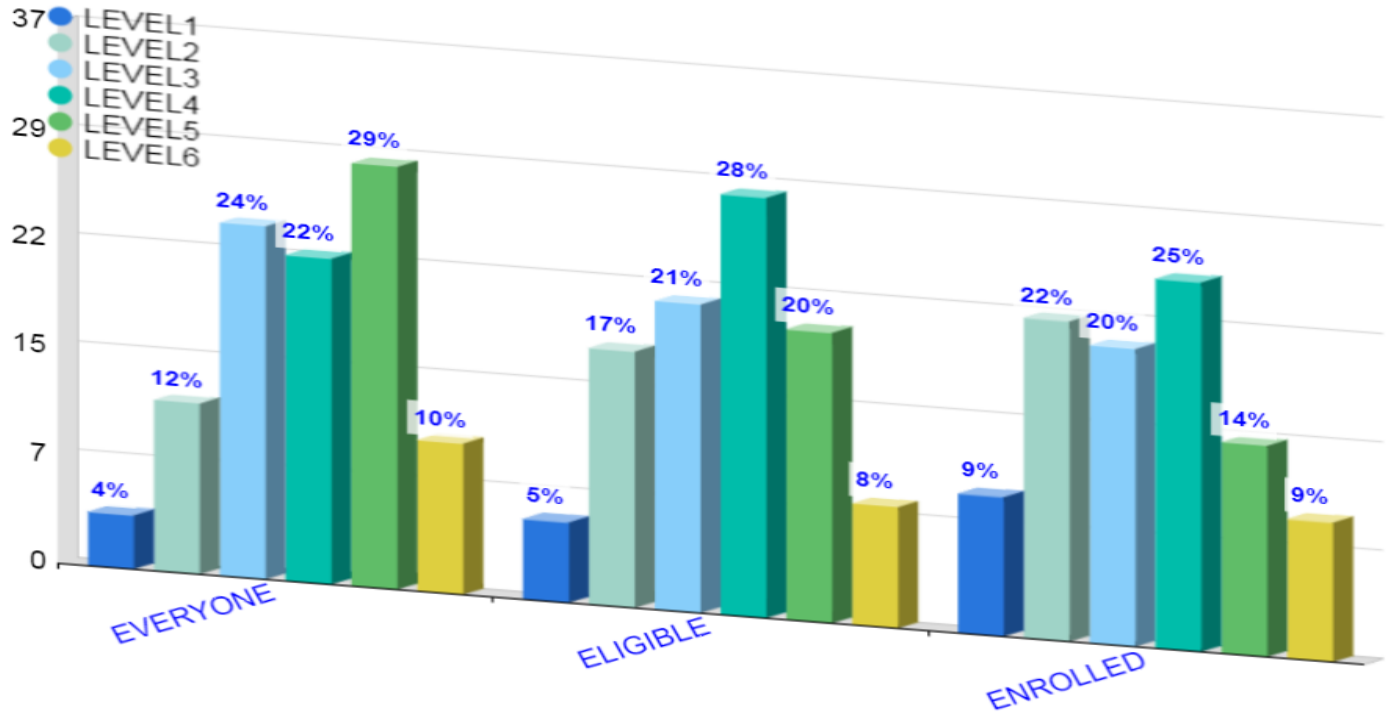
This is a contrasting visualization of data between two categories.



Display 5. Survey Completions by Month

Sample Data Visualization with Grouped Bar

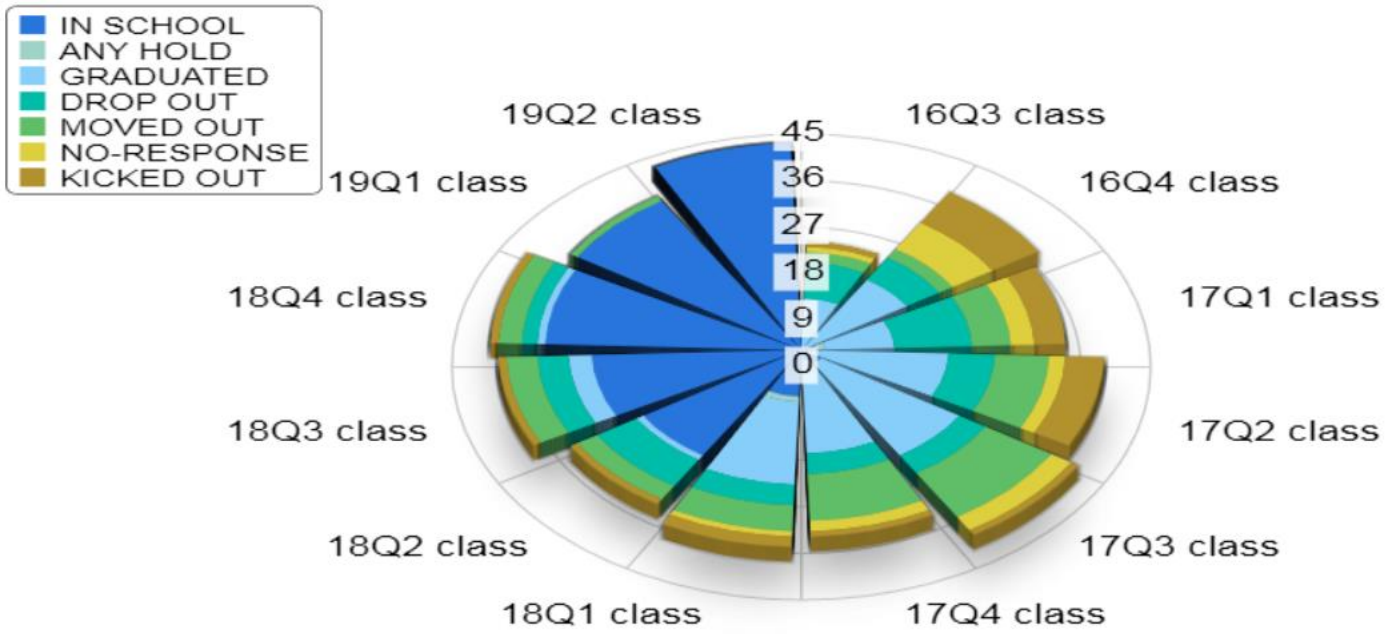
This is a visualization example for multiple groups for multiple sample populations. It also shows an interactive key for highlighting a specific group dynamically while chart is presented.



Display 7. Characteristics of Clients by Alternative Population

Sample Data Visualization with Rose Chart

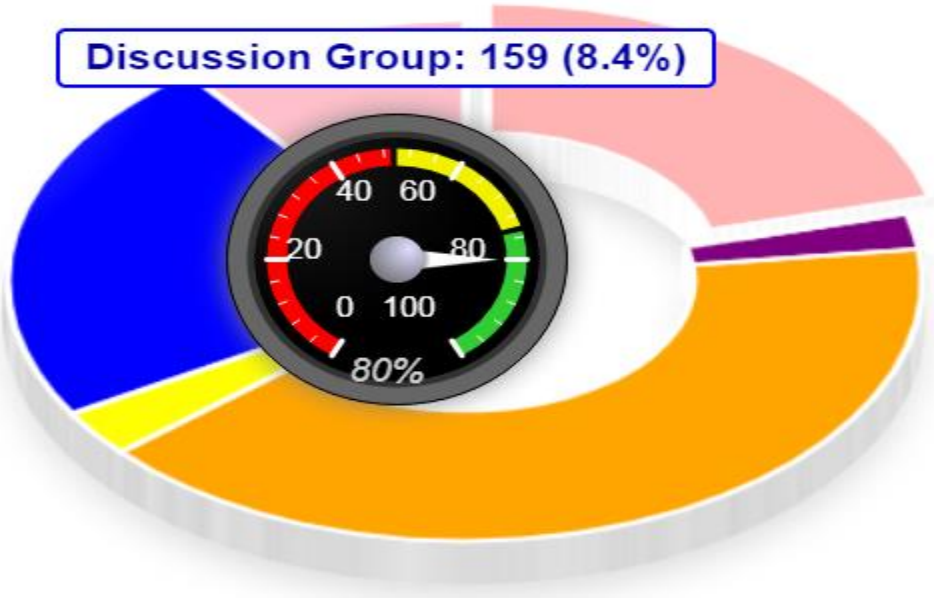
This is a data visualization example of many categories in a limited space. Although it is similar to a multiple stacked bar chart, it can present more data categories succinctly.



Display 8. Student Status by Quarterly Class Cohorts

The following is a nested gauge in each piece of a pie. The nested gauge showing metrics is displayed when each piece of pie is clicked.

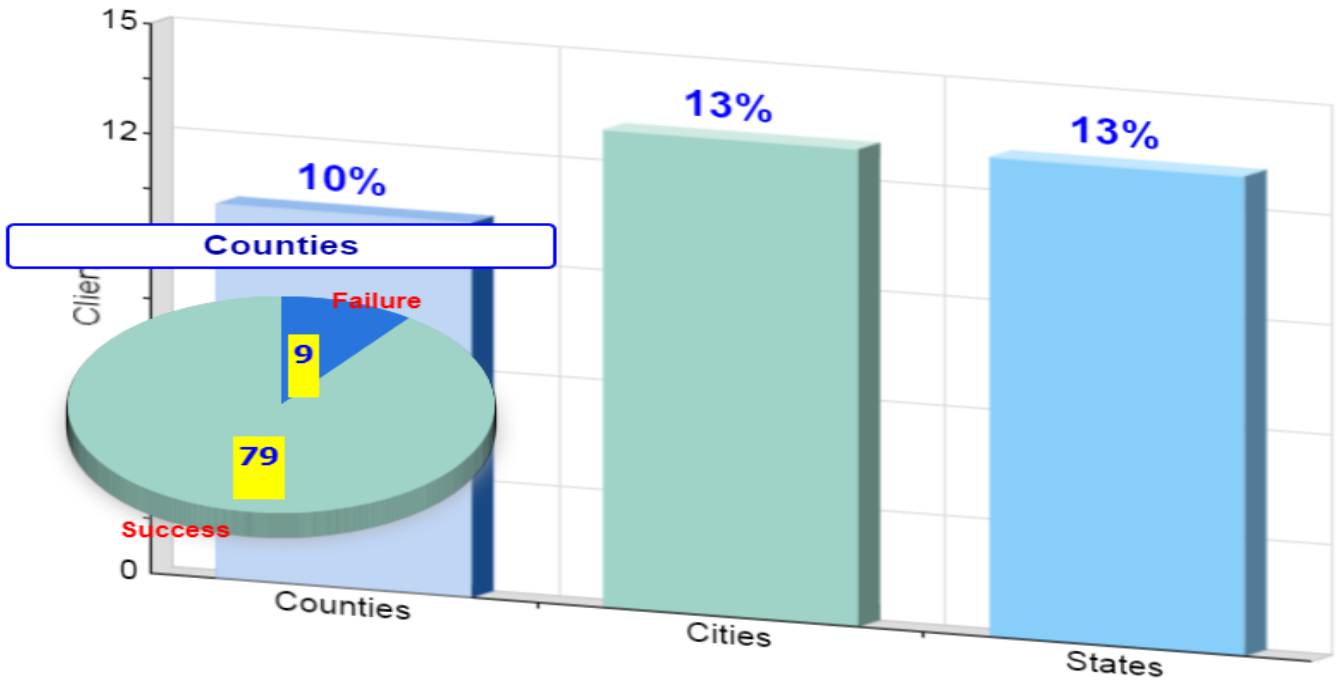
Sample Data Visualization with nested Gauge



Display 9. Performance measure of each group

Here is another example of visualization with a bar chart with a nested pie chart. When each bar is clicked, the nested pie chart is displayed as a tooltip. The nested visualization helps display two types of data such as counts and percentage in the same chart.

Sample Data Visualization of the bar chart with nested pie chart as interactive tooltip



Display 10. Recidivism comparison of different groups.

PREPARE AND CONVERT SAS DATASETS TO JSON OBJECTS

Analysts need to decide a desired data structure for visualizing the data. For example, determine how many categorical variables are needed for the type of chart that you want to draw? Then, create the desired SAS datasets using the SAS code. Typically this is done by data steps and SAS PROC SQL statements. Once you have the SAS dataset, the next step is to convert the datasets to the JSON files. Following is a few lines of SAS code to do the conversion.

```
/*Set your output library and folder then call SAS macro*/

libname my_html_folder 'path_to_folder_of_html_file';
libname myJSONoutputfolder 'path_to_folder_of_html_file \JSON\';
%let loc = %sysfunc(pathname(my_html_folder));
%INCLUDE "D:\folder_of_sas_macrofile\GetJSObject.txt';
%GETJSOBJECT(sas_dataset_name,json_object_name);
```

Users have to provide two parameters: 1) SAS dataset name to convert to JSON object and 2) JSON object name to create. This new JSON object name should be referenced in a script tag inside a HTML file. Then, when users open the HTML file, it will load the JSON object automatically. Then JavaScript will render the charts specified in the section tag of the HTML file.

You will have to add the include statement for a SAS macro text file (GetJSObject.txt) that I have attached in the appendix. If desired, you could develop your own macro file. Basically the macro takes care of reading the input SAS dataset and write the output JSON file according to the JSON object format and save the resulting JSON object file to the target destination where its reference is included in a script tag inside the HTML file. So that a charting script can use the JSON object variable to access the data values inside the JSON data structure.

It should be noted that this JSON object file is different from a normal JSON file in that the JSON data has to be assigned to a JavaScript variable inside the file. This is an important distinction because the normal JSON file would require a Web Server to load it through AJAX XMLHTTPREQUEST. However, the JSON object file that was created by the enclosed SAS Macro has the object assignment to JavaScript variable and thus will be loaded by a local file system without any server side functionality.

Lastly, I used the SAS to generate the JSON object files here. However, you can use any tools that you are familiar with to generate the JSON object files. Also you can schedule to run the SAS code right after refreshing your backend data warehouse or data mart. In this way, all the data visualization is updated automatically without human intervention.

CREATE AND EDIT A HTML FILE FOR DATA VISUALIZATION

Second task is to create an HTML file. You may use any text editor to create a HTM file. You can even use SAS if you want to automate the HTML file creation. The HTML file need to reference to the required external JavaScript libraries: JQuery, Reveal.js, and RGraph. Please download and unzip the files in the reveal.js folder next to the html file resides. You may change my folder structure as you desire. Below is the main HTML markups for these external scripts and css files in the head tag inside the <html></html> tag. Lastly, you need to include the JayDataVisualizationScripts.min.js to make all the scripts and html codes work together to generate the data visualization.

```
<link rel="stylesheet" href="reveal.js/css/reveal.css">
<link rel="stylesheet" href="reveal.js-plugins/menu/font-awesome-4.3.0/css/font-awesome.min.css">
<script src="reveal.js/lib/js/head.min.js"></script> -- head.js is for side menu and is optional --
<script src="reveal.js/js/reveal.js"></script>
<script src="reveal.js/lib/js/jquery-2.1.3.min.js"></script>
<script src="reveal.js/lib/rgraph/rgraph_module_you_need.js”></script>
    ….
    ….
<script type="text/javascript" src="js/JayDataVisualizationScripts.min.js"></script>
```

You need to include all the RGraph modules for all the chart types you need.

It is not recommended to change any of these script files unless you are a JavaScript expert and can troubleshoot these delicate scripts. Simply download them and reference them in your html file.

Then you can add a <section> tag inside <body><body> tag that reside inside the <html></html> tag as follows:

```
<section data-chart-title="your_chart_title"
        data-json-object="your_json_object_name"
        data-chart-type="your_chart_type"  (examples include pie, bar, rose, line, and etc.)
        data-dimension="xy"
        data-type=”y” (optional-- “y” for percentage data and “n” for counting data)
        data-chart-style="your_chart_style"  (optional--e.g., stacked, grouped, and etc.)
        data-audio-text="text for text to speech feature">
    <br />
    <canvas id="your_canvas_id" width="650" height="400">  </canvas>
    <ul>
       <li style="font-size: 26px">Foot note with data source.</li>
       <li style="font-size: 26px">foot note 2 --optional</li>
    </ul>
</section>
```

Each section tag represent a data visualization slide. Add as many section tags as you need.

You may adjust the canvas size by changing width and height attribute of the canvas tag, Also you can adjust color or font size of text and background.

You can control the chart title, chart type, chart style, data type, data dimension, text for text-to-speech feature. The text-to-speech feature is available only when it is deployed in a website.

You need to specify the data-dimension by the two digit numbers (xy). The first digit (x) represents the number of data categories and the second digit (y) represents the number of numeric variables. For example, the 23 means that there are two categorical variables and three numeric variable.

At the bottom of the body tag, you need to include all the references to the JSON object files you want to visualize. For example,

```
<script type="text/javascript" src="JSON/your_JSON_object_name1.json"></script>
   . . .
<script type="text/javascript" src="JSON/ your_JSON_object_name1.json"></script>
```

If you want, you may style your visualization further styling leveraging many great external JavaScript and CSS libraries.

MODERN PRESENTATION FEATURE

The visualization can have more impact if it is presented in more interactive manner. For example, a whiteboard drawing can be handy tool for aiding audience to understand easier. A highlighting pen can guide more attention to a critical implication of the visualization or make a point on an important finding.

The data visualization can be presented in the HTML slides leveraging the Reveal.js library. It provides many interesting navigations, highlighting, and whiteboard drawing, and other media features. However, more advanced feature of Reveal.js requires that the html file to be hosted by a web server.

CONCLUSION

This approach of using HTML for data visualization allows us to share and communicate data visualization instantly and universally without any requirement of purchasing a propriety software. The HTML approach allows users interact with a dynamic visualization. These characteristics of the HTML approach may be a key advantage to other alternative data visualization approaches. Furthermore, JSON is a universally accepted data format nowadays and most data tools provide a functionality to work with JSON data. Analysts/researchers can use any tools that they are familiar with to generate the JSON object files, although I demonstrated how to do it using the SAS in this paper.

Lastly, most organizations have a website or SharePoint site where you can host these HTML-based data visualization. Deploying the HTML-based data visualization in a website opens up even more possibilities such as self-talking and real-time data visualization. In some future, an advancement of Machine Learning algorithm will allow AI devices to completely automate searching and finding meanings out of overwhelming amount of data, finally delivering powerful, easy-to-understand, up-to-date data visualization anytime anywhere on demand.

REFERENCES

RGraph technical details from the link at https://www.rgraph.net/canvas/docs/api.html. RGraph is totally free to use, being available under the MIT license. See license details at https://www.rgraph.net/about.html.

Reveal.js JavaScript library at https://revealjs.com/. Reveal.js also is MIT licensed.

SAS Institute Inc. 2019. SAS®: User's Guide. Cary, NC: SAS Institute Inc. Available at support.sas.com/documentation/onlinedoc/forecast/14.1/fsug.pdf.

ACKNOWLEDGMENTS

Reveal.js and RGraph are utilized for data visualization. Many thanks to author of these wonderful scripts.

Thank you for Darin Goff, Karl Jones at the Washington State Department of Correction. They provided me a meaningful feedback on the data visualization.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jay Ahn, Ph.D.
State of Washington
jay.ahn@live.com

APPENDIX

1. SAS Macro for JSON Conversion:

```
%macro GETJSOBJECT(inputdsn, outfilename);
PROC JSON OUT="&loc\JSON\&outfilename..json"
PRETTY NOSASTAGS; EXPORT work.&inputdsn;
RUN;

DATA _JSON0; LENGTH jsoncode $ 1500;
INFILE "&loc\JSON\&outfilename..json"
DLM = "€" TRUNCOVER LRECL=1500;
INPUT jsoncode; ID=_N_;
RUN;

DATA _JSONCODE1;
SET _JSON0 END=mylast; ofn='var &outfilename=['; x=resolve(ofn);
%let endsemicolon = %str(];); myend ='&endsemicolon'; y=resolve(myend);
last=mylast; IF ID=1 then jsoncode= x || jsoncode;
IF last=1 then jsoncode = catx(jsoncode,y);
DATA _null_; SET _JSONCODE1;
FILE "&loc\JSON\&outfilename..json"; varlen=LENGTH(jsoncode);
PUT @1 jsoncode $varying200. varlen;
RUN;
%mend GETJSOBJECT;
```