Paper 128

From SAS® Data to Interactive Web Graphics Built Through PROC JSON

Robert Seffrin, National Agricultural Statistics Service

ABSTRACT

The National Agricultural Statistics Service (NASS) publishes extensive data covering the breadth of agriculture in the United States. To make this data more accessible to the public, NASS is exploring new and dynamic visualizations through the web. JavaScript has become a standard for displaying and interacting with this type of data. Developing charts from scratch has a steep learning curve requiring skill in JavaScript, HTML, and cascading style sheets. Many JavaScript visualization libraries assist with various aspects of charting, but a library called Vega greatly reduces the need for programming by defining chart parameters through a declarative grammar formatted as JSON (JavaScript Object Notation). While this eliminates most, if not all of, the JavaScript programming the JSON declarations can be complex with multiple nested levels. The new PROC JSON accessed through the SAS® University Edition greatly simplifies the creation of a JSON file to create an interactive scatterplot matrix where a selection in one subplot will appear in all other subplots. Charting parameters will be stored in an easy to edit Excel file which SAS will read and use to build a JSON file with data set specific variable names. Creating interactive web charts from SAS data is as simple as updating some parameters and building the JSON file.

INTRODUCTION

Interactive web graphics offer some exciting opportunities for NASS's internal and external customers. NASS already produces numerous data graphics such as for field crops, mostly static charts, some dynamic, and some with interactive features such as the census web maps. The crop progress charts, found at this link, are an example of static charts produced from the crop progress report using the SAS software graphics template language. The development in the past few years of numerous JavaScript charting libraries opens the door to creating a variety of new and interactive graphics from NASS's published data to increase the data's usefulness and appeal to the public.

While the charting libraries greatly simplify the visualization of data, most require an understanding of JavaScript at a low level. One exception to this is the Vega JavaScript library. With Vega all of the chart layout parameters come from a predefined grammar and are encoded in the JSON (JavaScript Object Notation) format. This removes the need to learn JavaScript but introduces its own complexities with multiple nested definitions, up to eight layers deep with the scatterplot matrix example used for this paper.

This project takes the scatterplot matrix example as seen in Figure 1 from the Vega website and generalizes the JSON parameter file so that other SAS data sets with NASS specific data may be plotted. The JSON file was flattened from the complex multi-bracketed format into three variables as an Excel file so that updates could be easily made. The focus was taken away from the JSON structure and given to the "string":value pairs which need to be reviewed and edited. Some macro variables were substituted for data set specific field names and some special character quoting needs were addressed. PROC JSON was then used to greatly simplify the recreation of the JSON parameter file for Vega to plot. Once the JSON parameter file is customized other SAS data sets are easily plotted.

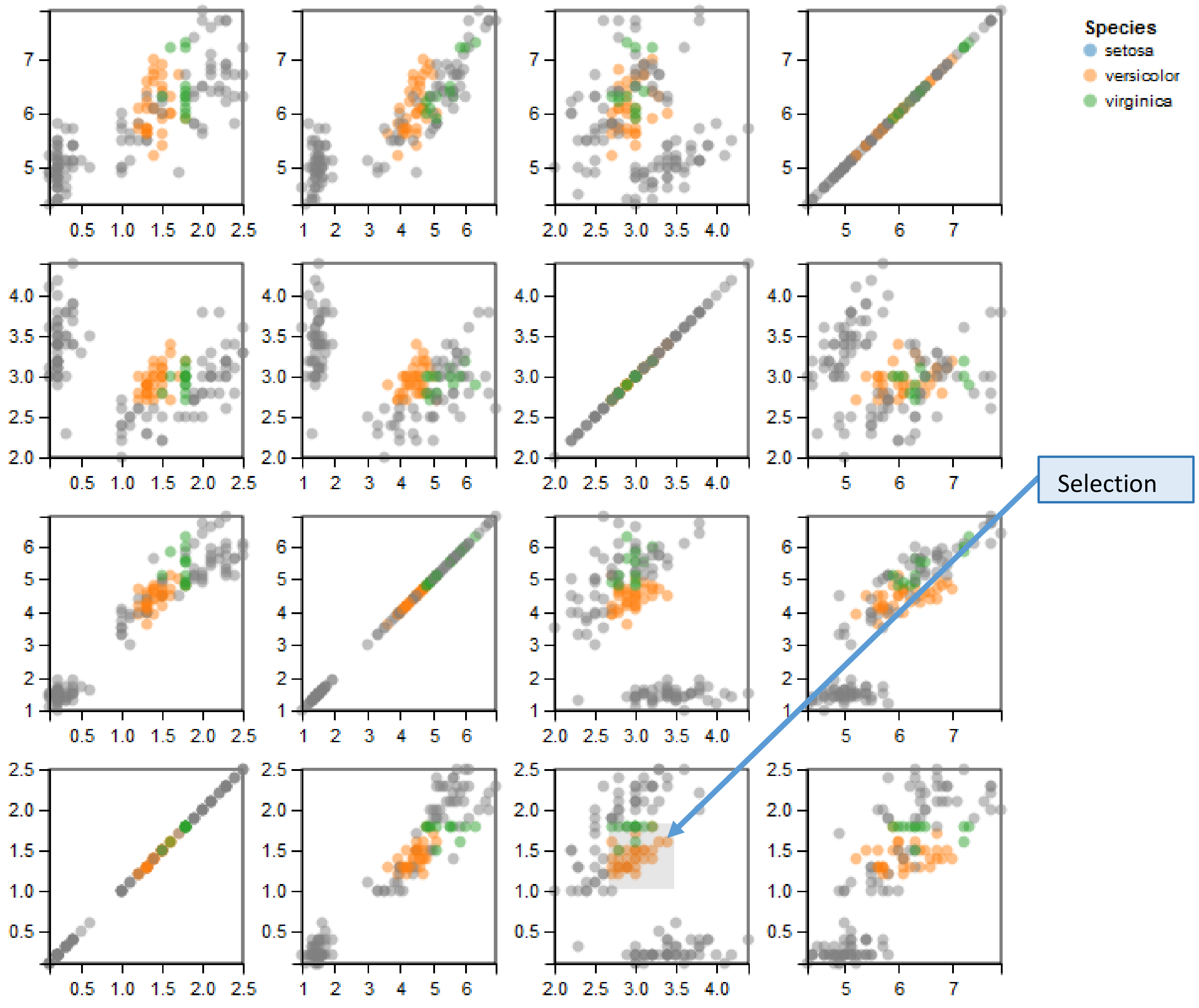

Figure 1: Web page display of scatterplot matrix where data selected by a mouse drag in one plot is selected in all plots.

GENERALIZING JSON INPUT TO VEGA

The components of a JSON file are relatively simple. For the purposes of this project the components are either a pair of values separated by a colon (“string”: value), objects inside of curly brackets {}, or arrays inside of square brackets []. The JSON code in Figure 2 is the “data” declaration section of the JSON file to render the plot. The complete JSON parameter file to define the chart is in the appendix.

```
{
  "width": 600,
  "height": 600,
  "data": [
    {
      "name": "iris",
      "url": "data/iris.json"
    },
    {
      "name": "fields",
      "values": ["petalWidth", "petalLength", "sepalWidth", "sepalLength"]
    }
  ],
```

Figure 2. Portion of original JSON specification for a scatterplot matrix.

Since SAS software does not yet have a JSON import procedure the JSON file was manually flattened in Excel into the three variables of: String, Value, and Bracket. The Member column is only there to help locate sections of the JSON file. The "data" section in Figure 2 is represented in Table 1 matching the syntax of the above JSON snippet in the Excel file is shown below.

member	String	Value	Bracket
width	width	600	
height	height	600	
data	data		[{
data	name	_data_	
data	url	"**data/&Data_json**"	}{
data	name	fields	
data	values		[
data	**&VarList /noscan**		**]}]**

Table 1. The "data" section in Excel with modifications.

Most of the JSON file defines the chart and layout parameters and does not need to be modified. To generalize this process for other input SAS data sets some substitutions were made as seen in Table 1. The value of 'iris' was replaced with '_data_' and the list of variables was replaced with a macro variable generated from a PROC SQL :INTO statement from Dictionary.columns. The /noscan option after the &VarList keeps PROC JSON from parsing the spaces between the field names.

There are some other special situations that required updates to the Excel file as seen in Table 2. The value "species" is data set dependent and was replaced in multiple places with the macro variable &Category. Although PROC JSON will quote text values, if they contain special characters such as a dot, they need to be quoted in the Excel file to avoid PROC JSON errors. Finally, to prevent macro interpretation of the value "&&", it was replaced with %NRSTR("&&").

member	String	Value	Bracket
predicates	signal	"start.x"	}{
predicates	type	%NRSTR("&&")	
marks	field	&Category	}{

Table 2. Some values requiring special coding in Excel.

Generating the JSON files

Once the Excel file has been customized it is imported.

```
PROC IMPORT
  DATAFILE="&Path.Vega_matrix_spec.xlsx"
  OUT=work.Vega_matrix_spec
  DBMS=xlsx REPLACE;
  RANGE="Sheet1$a1:d220"n;
  RUN;
```

The JSON parameter file is created directly from the imported Excel file using CALL EXECUTE() commands based on the contents of the String and Bracket variables.

Input found	PROC JSON output command
String	‘WRITE VALUES ’ String Value ,
[	‘WRITE OPEN ARRAY’,
{	‘WRITE OPEN OBJECT’,
] or }	‘WRITE CLOSE’

The **`PRETTY`** option creates a more readable output with indentation and line feeds.

```
DATA _NULL_;
  SET Work.Vega_matrix_spec END=Last;
  IF _N_ = 1 THEN
    CALL EXECUTE ('PROC JSON OUT="/folders/myfolders/Code_JSON.json" PRETTY;');
  IF String NE '' THEN
    String_Val = 'WRITE VALUES '||TRIM(String)||' '||Value||';';
    CALL EXECUTE( String_Val );
  IF Bracket NE '' THEN DO i=1 TO LENGTH(Bracket);
    iBracket = CHAR(Bracket, i);
    IF iBracket = '[' THEN CALL EXECUTE ('WRITE OPEN ARRAY;');
    IF iBracket = '{' THEN CALL EXECUTE ('WRITE OPEN OBJECT;');
    IF iBracket IN(']', '}') THEN DO;
      CALL EXECUTE ('WRITE CLOSE;');
      END;
    END;
  IF Last THEN CALL EXECUTE('RUN;');
  RUN;
```

The Iris data set which SAS supplies in the SASHELP library is also output to JSON format with PROC JSON:

```
PROC JSON
    OUT="/folders/myfolders/&Data_json"
    PRETTY;
  EXPORT Sashelp.iris /NOSASTAGS;
  RUN;
```

RECREATING THE SCATTERPLOT MATRIX

The HTML file to view the graphic is relatively simple since all of the JavaScript parameters are now in the JSON file. The only dependencies are to D3.js and Vega.js. Since the data is built into the JSON structure the graphic may be displayed directly in a web browser without a web server.

```
<html>
  <head>
    <title>Iris_interact_scatter</title>
<script src="../d3.v3.min.js"></script>
<script src="vega.js"></script>
  </head>
  <body>
    <div id="vis"></div>
  </body>
  <script type="text/javascript">
  function parse(spec) {
    vg.parse.spec(spec, function(chart) {
      var view = chart({ el:"#vis" });
      view.viewport(null)
          .renderer("svg")
          .update();
    });
  }
  parse("Code_json.json");
  </script>
  </html>
```

CONCLUSION

A SAS software user can generate interactive web graphics with minimal JavaScript programming by taking advantage of the Vega JavaScript library and PROC JSON in SAS. Although the chart specifications can be complex, once created in Excel they can be easily tweaked and reused. In this example only the data set and fields in the JSON parameter file were generalized with macro variables. Several additional parameters could be macroized and controlled from SAS to extend a look and feel across multiple chart types. To simplify the process a data step could be written to import the original JSON file directly into the flat format that was created manually here. PROC JSON now available in SAS 9.4 greatly simplifies generating the structured JSON output.

REFERENCES

Intel Science and Technology Center for Big Data. "Vega." July 23, 2015. Available at http://vega.github.io/vega/

National Agricultural Statistics Service. http://www.nass.usda.gov/Charts_and_Maps/Crop_Progress_&_Condition/index.asp

SAS Institute Inc. 2015. Base SAS® 9.4 Procedures Guide, Fourth Edition. Cary, NC: SAS Institute Inc.

Contact Information

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

Name:	Robert Seffrin
Enterprise:	USDA, National Agricultural Statistics Service
Address:	3251 Old Lee Hwy
City, State ZIP:	Fairfax, VA 22030
Work Phone:	703-877-8000 ext. 155
E-mail:	Robert_seffrin@nass.usda.gov
Web:	www.nass.usda.gov

APPENDIX

Complete original JSON specification for the scatterplot matrix example from the Vega website.

```
{
  "width": 600,
  "height": 600,
  "data": [
    {
      "name": "iris",
      "url": "data/iris.json"
    },
    {
      "name": "fields",
      "values": ["petalWidth", "petalLength", "sepalWidth", "sepalLength"]
    }
  ],

  "signals": [
    {
      "name": "cell",
      "init": {},
      "streams": [
        {"type": ".cell:mousedown", "expr": "i"},
        {"type": ".symbol:mousedown", "expr": "i.mark.group"}
      ]
    },
    {
      "name": "brush_start",
      "init": {"x": 0, "y": 0},
      "streams": [{"type": ".cell:mousedown, .symbol:mousedown", "expr": "p"}]
    },
    {
      "name": "brush_end",
      "init": {"x": 0, "y": 0},
      "streams": [
        {"type": ".cell:mousedown, .symbol:mousedown, .cell:mouseup, .symbol:mouseup",
"expr": "p"},
        {"type": "[(.cell:mousedown, .symbol:mousedown), mouseup] > mousemove",
"expr": "p"}
      ]
    },
    {
      "name": "start_x",
      "init": 0,
      "streams": [{
        "type": "brush_start",
        "expr": "brush_start.x - cell.x",
        "item": {"signal": "cell"},
        "scale": "x",
        "invert": true
      }]
    },
    {
      "name": "start_y",
      "init": 0,
      "streams": [{
        "type": "brush_start",
        "expr": "brush_start.y - cell.y",
        "item": {"signal": "cell"},
        "scale": "y",
        "invert": true
      }]
    },
    {
      "name": "end_x",
      "init": 0,
      "streams": [{
        "type": "brush_end",
        "expr": "brush_end.x - cell.x",
        "item": {"signal": "cell"},
        "scale": "x",
        "invert": true
      }]
    },
    {
      "name": "end_y",
      "init": 0,
      "streams": [{
        "type": "brush_end",
        "expr": "brush_end.y - cell.y",
        "item": {"signal": "cell"},
        "scale": "y",
        "invert": true
      }]
    }
  ],

  "predicates": [
    {
      "name": "xRange",
      "type": "in",
      "item": {"arg": "x"},
      "range": [{"signal": "start_x"}, {"signal": "end_x"}]
    },

    {
      "name": "yRange",
      "type": "in",
      "item": {"arg": "y"},
      "range": [{"signal": "start_y"}, {"signal": "end_y"}]
    },

    {
      "name": "inRange",
      "type": "&&",
      "operands": [
        {"predicate": "xRange"},
        {"predicate": "yRange"}
      ]
    }
  ],

  "scales": [
    {
      "name": "gx",
      "type": "ordinal",
      "range": "width",
      "round": true,
      "domain": {"data": "fields", "field": "data"}
    },
    {
      "name": "gy",
      "type": "ordinal",
      "range": "height",
      "round": true,
      "reverse": true,
      "domain": {"data": "fields", "field": "data"}
    },
    {
      "name": "c",
      "type": "ordinal",
      "domain": {"data": "iris", "field": "species"},
      "range": "category10"
    }
  ],
  "legends": [
    {
      "fill": "c",
      "title": "Species",
      "offset": 10,
      "properties": {
        "symbols": {
          "fillOpacity": {"value": 0.5},
          "stroke": {"value": "transparent"}
        }
      }
    }
  ],
  "marks": [
    {
      "type": "group",
      "from": {
        "data": "fields",
        "transform": [{"type": "cross"}]
      },
      "properties": {
        "enter": {
          "class": {"value": "cell"},
          "a": {"field": "a.data"},
          "b": {"field": "b.data"},
          "x": {"scale": "gx", "field": "a.data"},
          "y": {"scale": "gy", "field": "b.data"},
          "width": {"scale": "gx", "band": true, "offset":-35},
          "height": {"scale": "gy", "band": true, "offset":-35},
          "fill": {"value": "#fff"},
          "stroke": {"value": "#ddd"}
        }
      },
      "scales": [
        {
          "name": "x",
          "range": "width",
          "zero": false,
          "round": true,
          "domain": {"data": "iris", "field": {"group": "a.data"}}
        },
        {
          "name": "y",
          "range": "height",
          "zero": false,
          "round": true,
          "domain": {"data": "iris", "field": {"group": "b.data"}}
        }
      ],
      "axes": [
        {"type": "x", "scale": "x", "ticks": 5},
        {"type": "y", "scale": "y", "ticks": 5}
      ],
      "marks": [
        {
          "type": "symbol",
          "from": {"data": "iris"},
          "properties": {
            "enter": {
              "class": {"value": "symbol"},
              "x": {"scale": "x", "field": {"group": "a.data"}},
              "y": {"scale": "y", "field": {"group": "b.data"}},
              "fill": {"scale": "c", "field": "species"},
              "fillOpacity": {"value": 0.5},
              "size": {"value": 36}
            },
            "update": {
              "fill": {
                "rule": [
                  {
                    "predicate": "inRange",
                    "input": {
                      "x": {"field": {"signal": "cell.a"}},
                      "y": {"field": {"signal": "cell.b"}}
                    },

                    "scale": "c",
                    "field": "species"
                  },
                  {"value": "grey"}
                ]
              }
            }
          }
        }
      ]
    },

    {
      "type": "rect",
      "properties": {
        "enter": {
          "fill": {"value": "grey"},
          "fillOpacity": {"value": 0.2}
        },
        "update": {
          "x": {"signal": "brush_start.x"},
          "x2": {"signal": "brush_end.x"},
          "y": {"signal": "brush_start.y"},
          "y2": {"signal": "brush_end.y"}
        }
      }
    }
  ]
}
```