PharmaSUG 2018 - Paper DV-16

Graphic Visualization for Treatment Switch Pattern

Anni Weng and Huanxue Zhou, KMK Consulting, Inc.

ABSTRACT

Visual report is a commonly used tool to present complex data by drafting understandable and digestible graphs. A well-designed visual report not only extracts key business insights from vast amounts of information, but also provides instant visual feedback to real-time changes. In this paper, we will introduce two elegant and easily-generated visual reports for treatment switch pattern. First is the web-based interactive tree diagram, which uses circles to represent treatments and length of links between two circles to represent time length of each treatment switch. The biggest advantage of this diagram is that it offers users complete control over displayed information as each circle can be collapsed or expanded. The second visual report is the web-based Sankey diagram, which adopts nodes as treatments and width of flows between two nodes as quantities switched from one treatment to another. This diagram enables users to easily perceive the transition trend over time. To implement these two graphs in SAS, we will briefly introduce syntax of PROC STREAM and how it embeds JavaScript framework to generate the output. The purpose of this paper is to empower users to explore more possibilities of visual reports by using existed features in SAS.

INTRODUCTION

Treatment switch pattern can be used in a real world setting to help researchers understand how patients use medications over time. It contains a wealth of information in hierarchical structure, including treatment sequences, medication adherence, patient switch portion, and the change of product market share. Due to its depth and complexity of information, treatment switch pattern also requires an effective communication tool in order to integrate all data together and present it in an intuitive way.

In this paper, we will introduce two interactive, web-based visual report tools -- tree diagram and Sankey diagram -- along with JavaScript framework to illustrate treatment switch pattern. Both diagrams provide comprehensive information in an easy-to-understand format to help users quickly identify potential treatment sequences and main switch patterns, as well as detailed information during each transition.

The intention of this paper is not to provide instruction on HTML or JavaScript, but how to use the JSON procedure to feed the data and STREAM procedure to generate HTML through SAS. Note: For additional information on using HTML or JavaScript, please check D3 (Data-Driven Documents) or other websites to find well-established templates that can readily be used. To explore more real world applications of PROC STREAM, please refer to Don Henderson's publication "PROC STREAM and SAS® Server Pages: Generating Custom HTML Reports".

INTERACTIVE TREE DIAGRAM

INTRODUCTION

Tree diagram is well known for displaying hierarchical data, which is also one of the biggest characteristics of treatment switch pattern. In this example, we will introduce an upgraded version of tree diagram to give users control of displayed content based on their needs and interests.

To implement interactive tree diagram in SAS, three main steps will be involved - organize data in JSON format, prepare external files for web content, and use PROC STREAM in SAS to pass all files into the web page to generate the diagram.

DATA PREPARATION USING PROC JSON

In order to produce a web-based tree diagram, we choose to use JSON data, as it is a common data format used for browser-server communication. Like the example shown below, each observation listed in

JSON stands for a node in diagram, and paired data in the observation indicates different characteristics of that node.

```
[
  {
    "desc": "Drug B",
    "name": "Drug B",
    "parent": "null",
    "value": 933,
    "timeFromOrg": 0,
    "timeFromParent": 0,
    "color": "#c0392b",
    "size": 100,
    "pos": 5
  },
  {
    "desc": "Drug A",
    "name": "Drug B->Drug A",
    "parent": "Drug B",
    "value": 206,
    "timeFromOrg": 104,
    "timeFromParent": 104,
    "color": "#b7950b",
    "size": 26,
    "pos": 34
  }
]
```

Below are some highlights of JSON file:

- Variable “desc” indicates the current drug for the node.
- Variable “name” describes the path to the current node and it is the unique identifier of the node.
- Variable “parent” indicates the name of parent node.
- Variable “value” contains number of patients in this node.
- Variable “timeFromOrg” provides average number of days from initial to current node.
- Variable “timeFromParent” provides average number of days from parent node to current node.

The rest of the variables are used to specify attributes of node in the tree diagram, such as node color, node size and relative horizontal position comparing with parent. The size of node is scaled based on “value” and the horizontal position of node is scaled based on variable “timeFromParent”.

Starting from version 9.4, SAS provides the JSON procedure to read data from SAS format into JSON reference. Below is an example of source data in SAS format.

desc	name	parent	value	timeFromOrg	timeFromParent	color	size	pos
Drug B	Drug B	null	933	0	0	#E91E63	100	5
Drug A	Drug B->Drug A	Drug B	206	104	104	#9C27B0	26	34

Table 1. Tree Diagram SAS Data

SAS data is passed into PROC JSON to produce JSON file for interactive tree diagram.

```
PROC JSON OUT = 'H:\data\HEOR\Develop\D3\Test\Tree\switchTree_tree.json'
PRETTY NOSASTAGS;
```

```
    WRITE OPEN ARRAY;
          EXPORT dat.tree(DROP = cur pre);
    WRITE CLOSE;
RUN;
```

Here are some key features of PROC JSON in this example:

- PRETTY option – create a more readable format for better illustration of JSON data structure.
- NOSASTAGS option – compress SAS metadata.
- WRITE OPEN ARRAY statement – explicitly opens an array (which contains the list of observations.)It must be closed with WRITE CLOSE statement.

For more information, see Base SAS® 9.4 Procedures Guide: JSON Procedure.

OUTPUT GENERATION

After data preparation, we also need to get external files ready so that they could be successfully passed into PROC STREAM. In this example, external files include a HTML file to display output, a CSS file to customize the web page, a JavaScript file to generate tree diagram, and a JSON file to input data. Details of those files will not be further explained in this paper, as it involves non-SAS techniques.

```
%LET Dir = H:\switch tree;
LIBNAME dat "&dir\";
FILENAME htmlout "&Dir\switchTree_tree.html";
FILENAME cssfile "&Dir\switchTree_tree.css";
FILENAME jsfile "&Dir\switchTree_tree.js";
FILENAME jsonfile "&Dir\switchTree_tree.json";
```

PROC STREAM is a new procedure introduced in Base SAS® 9.3. It aims to process external files and pass their resolved results along with the remaining arbitrary text to the destination. In this example, above external files (i.e. “cssfile”) will be resolved, and their values will be streamed back into the input stack and sent to the external file “htmlout”.

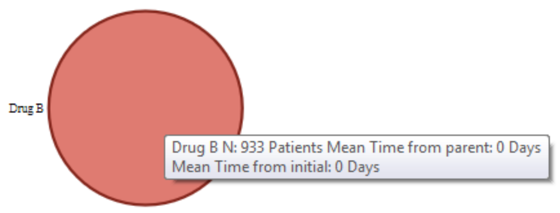
```
%LET _chartName = treeChart;
PROC STREAM OUTFILE = htmlout RESETDELIM = 'goto';
BEGIN
<!DOCTYPE html>
&streamDelim NEWLINE; <html lang="en">
&streamDelim NEWLINE;       <head>
&streamDelim NEWLINE;       <meta charset="utf-8">
&streamDelim NEWLINE;             <title> Switch Tree </title>
&streamDelim NEWLINE;             <style>
&streamDelim NEWLINE;             &streamDelim READFILE cssfile;
&streamDelim NEWLINE;       </style>
&streamDelim NEWLINE;       </head>
&streamDelim NEWLINE;       <body>
&streamDelim NEWLINE;             <div id="&_chartName"></div>
&streamDelim NEWLINE;             <script
                                  src='http://d3js.org/d3.v3.min.js'></scri
                                  pt>
&streamDelim NEWLINE;             <script>
&streamDelim NEWLINE;             &streamDelim READFILE jsfile;
&streamDelim NEWLINE;             </script>
&streamDelim NEWLINE;             <script>
&streamDelim NEWLINE;                   VAR graphData=&streamDelim READFILE
                                        jsonfile;;
&streamDelim NEWLINE;                   treeGraph(&_chartName, graphData)
```

```
&streamDelim NEWLINE;             </script>
&streamDelim NEWLINE;       </body>
&streamDelim NEWLINE; </html>
;;;;
```

Here are some highlights of PROC STREAM:

- RESETDELIM option: sets values for &STREAMDELIM
- &STREAMDELIM statement: recognizes and reads special statement, macro, or external files
- NEWLINE statement: starts a new line to make format more readable
- READFILE: passes content of external files into PROC STREAM

INTERACTIVE TREE DIAGRAM OUTPUT


Drug B
Drug B N: 933 Patients Mean Time from parent: 0 Days
Mean Time from initial: 0 Days

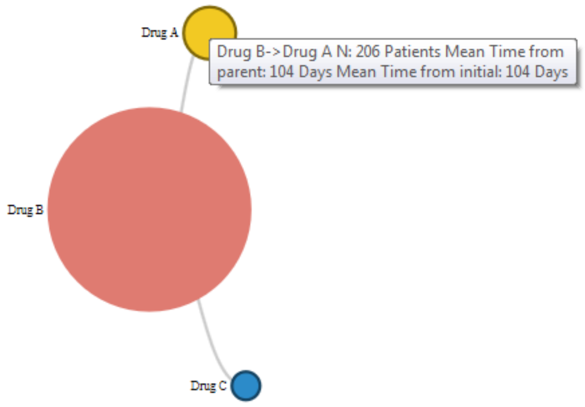

Drug A
Drug B->Drug A N: 206 Patients Mean Time from
parent: 104 Days Mean Time from initial: 104 Days
Drug B
Drug C

Figure 1. Interactive Tree Diagram - Initial Status

Figure 2. Interactive Tree Diagram - Intermediate Status

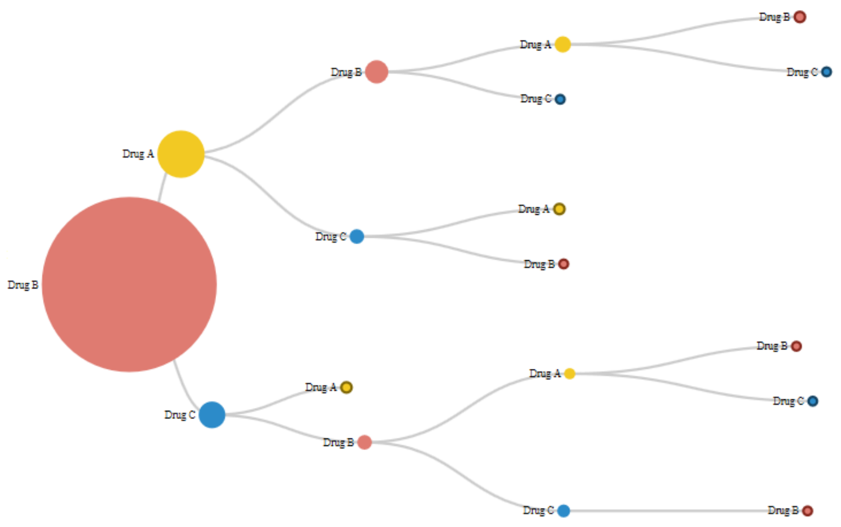


Figure 3. Interactive Tree Diagram - Final Status

The above figures represent three statuses of our tree diagram: initial, middle, and final. It starts with one treatment (Figure 1) and then collapses the next layer of details (Figure 2), and it ends with a fully expanded tree diagram showing all possible treatment switch pathways. (Figure 3)

Any node in this diagram can be clicked on to collapse the next layer. Details like patient amount and switch time length can be directly referred to node size and link length and both of them are designed proportionally to the real data. To retrieve more specific insights, such as exact number of patients, simply hover the mouse over a node, and all information will then pop up.

SANKEY DIAGRAM

INTRODUCTION

Sankey diagram is a type of flow diagram that connects a series of events over time. Nodes in Sankey are usually called events, and a sequence of nodes make up a path. Since treatment switch pattern is multiple sequences of treatments over time, it can be visualized by Sankey diagram in nature.

DATA PREPARATION USING PROC JSON

JSON data for this example will be slightly more complicated than that for the tree diagram. Below is a brief overview of how it looks.

```
{
    "links": [
        {"source":"Patients in cohort X","target":"Initial Drug
        A","value":1574,"color":"#9E9E9E"},
        {"source":"Patients in cohort X","target":"Initial Drug
        B","value":933,"color":"#9E9E9E"},
        {"source":"Initial Drug A","target":"1st switch to Drug
        B","value":428,"color":"#C0392B"},
        {"source":"Initial Drug B","target":"1st switch to Drug
        A","value":206,"color":"#B7950B"}
    ],
    "nodes": [
        {"name":"Patients in cohort X","color":"#9E9E9E"},
        {"name":"Initial Drug A","color":"#C0392B"},
        {"name":"Initial Drug B","color":"#B7950B"},
        {"name":"1st switch to Drug A","color":"#C0392B"}
    ]
};
```

- Source: source treatment in a switch
- Target: target treatment in a switch
- Value: number of patients in a switch
- Name: display the event name for each node
- Color: color of links or nodes

As shown above, JSON data in this example has more than one array - “node” and “links.” Similar to the one in tree diagram, each array is made up of a list of name/value paired data, and the purpose of having two arrays here is to make the data structure clearer since they indicate different parts of Sankey. Please also note that if there are multiple arrays in a JSON file, use a square bracket to embed the content and use a comma to separate each two arrays.

Here is a SAS raw data sample for Sankey diagram.

Link data set:

source	target	value	color
Patients in cohort X	Initial Drug A	1574	#9E9E9E
Patients in cohort X	Initial Drug B	933	#9E9E9E
Patients in cohort X	Initial Drug C	313	#9E9E9E
Initial Drug C	1st switch to Drug A	42	#FF9800
Initial Drug C	1st switch to Drug B	76	#FF9800

Table 2. SAS Link Data

Node data set:

name	color
Patients in cohort X	#9E9E9E
Initial Drug A	#9C27B0
Initial Drug B	#E91E63
Initial Drug C	#FF9800
1st switch to Drug A	#9C27B0

Table 3. SAS Node Data

SAS data is transformed into JSON reference to produce Sankey diagram.

```
PROC JSON OUT = 'H:\sankey\switchTree_sankey2.json' PRETTY NOSASTAGS;
    WRITE OPEN OBJECT;
          WRITE VALUE "links";
          WRITE OPEN ARRAY;
                EXPORT links;
          WRITE CLOSE;

          WRITE VALUE "nodes";
          WRITE OPEN ARRAY;
                EXPORT nodes;
          WRITE CLOSE;
    WRITE CLOSE;
RUN;
```

As it involves more than one array, we add two new options in the JSON procedure.

- WRITE OPEN OBJECT statement: explicitly opens an object (which contains a list of arrays.) It must be closed with a WRITE CLOSE statement.
- WRITE VALUE statement: specify the name of array under quotation mark.

OUTPUT GENERATION

External file required for producing Sankey diagram.

```
%LET Dir = H:\sankey;

LIBNAME dat "&dir\";

FILENAME htmlout "&Dir\switchTree_sankey.html";

FILENAME cssfile "&Dir\switchTree_sankey.css";

FILENAME jsfile "&Dir\switchTree_sankey.js";

FILENAME jsonfile "&Dir\switchTree_sankey.json";
```

Read external files in PROC STREAM to produce the Sankey diagram.

```
%LET _chartName = sankeyChart;
PROC STREAM OUTFILE = htmlout RESETDELIM = 'goto';
BEGIN
<!DOCTYPE html>
&streamDelim NEWLINE; <html lang="en">
&streamDelim NEWLINE;   <head>
&streamDelim NEWLINE;         <meta charset="utf-8">
&streamDelim NEWLINE;         <title> Switch Tree </title>
&streamDelim NEWLINE;         <style>
&streamDelim NEWLINE;               &streamDelim READFILE cssfile;
&streamDelim NEWLINE;         </style>
&streamDelim NEWLINE;   </head>
&streamDelim NEWLINE;   <body>
&streamDelim NEWLINE;         <div id="&_chartName"></div>
&streamDelim NEWLINE;         <script
                              src='http://d3js.org/d3.v3.min.js'></scri
                              pt>
&streamDelim NEWLINE;         <script>
&streamDelim NEWLINE;               &streamDelim READFILE jsfile;
&streamDelim NEWLINE;         </script>
&streamDelim NEWLINE;         <script>
&streamDelim NEWLINE;               VAR graphData=&streamDelim READFILE
                                    jsonfile;;
&streamDelim NEWLINE;               sankeyGraph(&_chartName,
                                    graphData);
&streamDelim NEWLINE;         </script>
&streamDelim NEWLINE;   </body>
&streamDelim NEWLINE; </html>
;;;;
```

As one can observe, programs for tree and Sankey are quite similar. Users can also use this template to generate other web-based diagrams by slightly changing the provided code.

SANKEY DIAGRAM OUTPUT

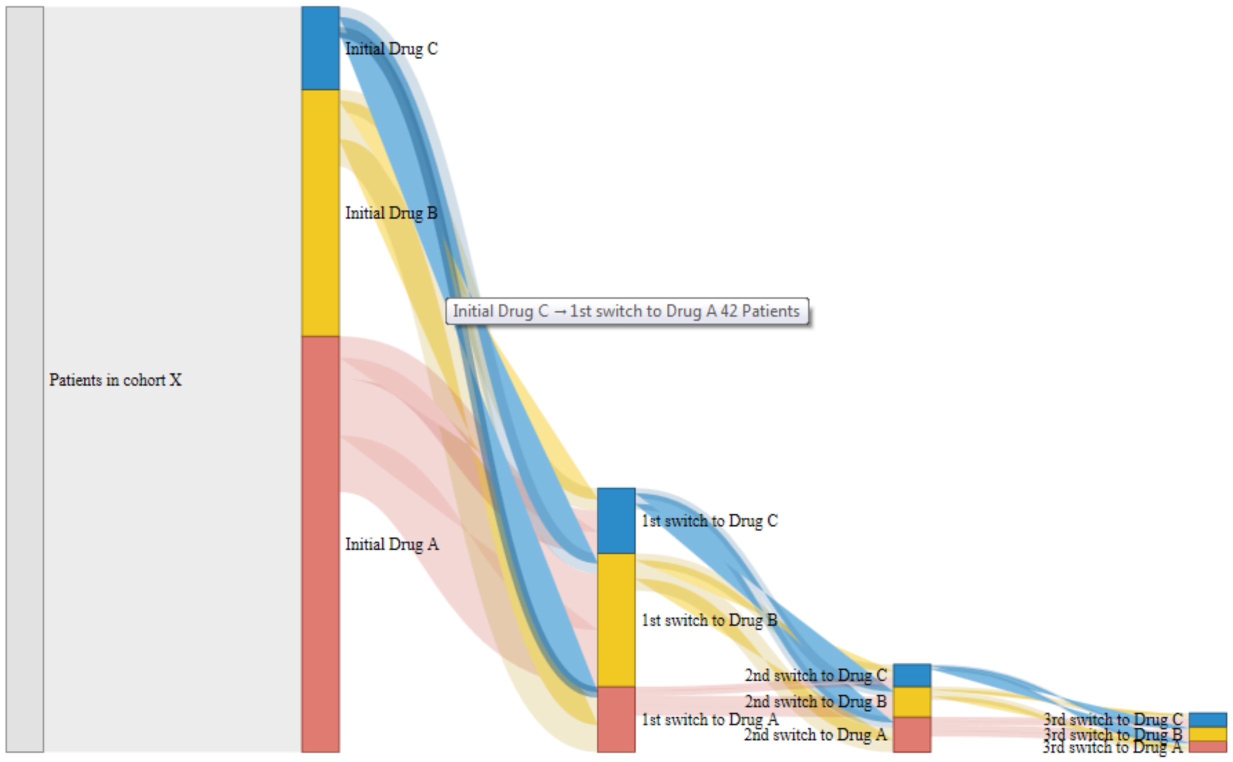


Figure 4. Sankey Diagram Output

In the above diagram, the width of each node indicates total patients for each treatment, and the width of each link represents patients transferred from one treatment to another.

Specific number of patients can be obtained by hovering the mouse over a node. If multiple flows are crossed together, you can move the nodes to any position to better perceive the trend.

SANKEY VS. INTERACTIVE TREE DIAGRAM

Both Sankey and tree diagram provide high-quality visual reports for treatment switch pattern, but the emphasis of the two diagrams will be slightly different. In order to help users choose the diagram that best meets their needs, we summarize the main characteristics of two diagrams in the following table.

Characteristics	Sankey Diagram	Interactive Tree Diagram
Content	Overall Picture	Drill Down Analysis
Dominant Trend	Obvious	Not Obvious
Time Length	Not Provided	Provided
Interactive	Not Interactive	Interactive

Table 4. Comparison Table between Sankey Diagram and Interactive Tree Diagram

Sankey diagram aims to display a whole picture of event changes while tree diagram focuses more on drill down analysis. For example, you can easily track all changes that drive into one treatment in Sankey, yet in tree diagram, it is more likely to find a specific pathway from one treatment to another. In Sankey diagram, you can quickly observe the dominant trend by comparing link width between each two nodes, while in tree diagram, you check the link length to see how long each switch takes. Moreover, one big advantage of thetree diagram is that it creates an interactive user interface, which can be used to hide or reveal the information based on the user's needs.

CONCLUSION

Interactive tree and Sankey diagram are two powerful communication tools to visualize treatment switch pattern, and they can be easily generated by JSON and STREAM procedure. To explore more compelling visualizations, the SAS template provided in this paper is for readers to use and combine with pre-built JavaScript frameworks online.

REFERENCES

Henderson, Don. 2014. “PROC STREAM and SAS® Server Pages: Generating Custom HTML Reports”. *Proceedings of the SAS Global 2014 Conference.* Cary, NC: SAS Institute. Available online at http://support.sas.com/resources/papers/proceedings14/1738-2014.pdf.

Hinson, Joseph. 2015. “Proc STREAM: The Perfect Tool For Creating Patient Narratives”. *Proceedings of the SAS Global 2015 Conference.* Dallas, TX: SAS Institute. Available online at https://www.pharmasug.org/proceedings/2015/AD/PharmaSUG-2015-AD03.pdf.

Maclean, Malcolm. “Sankey Diagrams.” D3 Tips & Tricks: Interactive Data Visualization in a Web Browser, Lean Publishing, 2014. Accessed December 5, 2017. https://leanpub.com/D3-Tips-and-Tricks/read#leanpub-auto-sankey-diagrams

Maclean, Malcolm. “Tree Diagrams.” D3 Tips & Tricks: Interactive Data Visualization in a Web Browser, Lean Publishing, 2014. Accessed Dec 5, 2017. https://leanpub.com/D3-Tips-and-Tricks/read#leanpub-auto-tree-diagrams

CONTACT INFORMATION

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

Anni Weng
KMK Consulting,Inc.
23 Headquarters Plaza
Morristwon, NJ 07960
anni.weng@kmkconsultinginc.com

Huanxue Zhou
KMK Consulting,Inc.
23 Headquarters Plaza
Morristwon, NJ 07960
huanxue.zhou@kmkconsultinginc.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.