SESUG Paper 271-2018

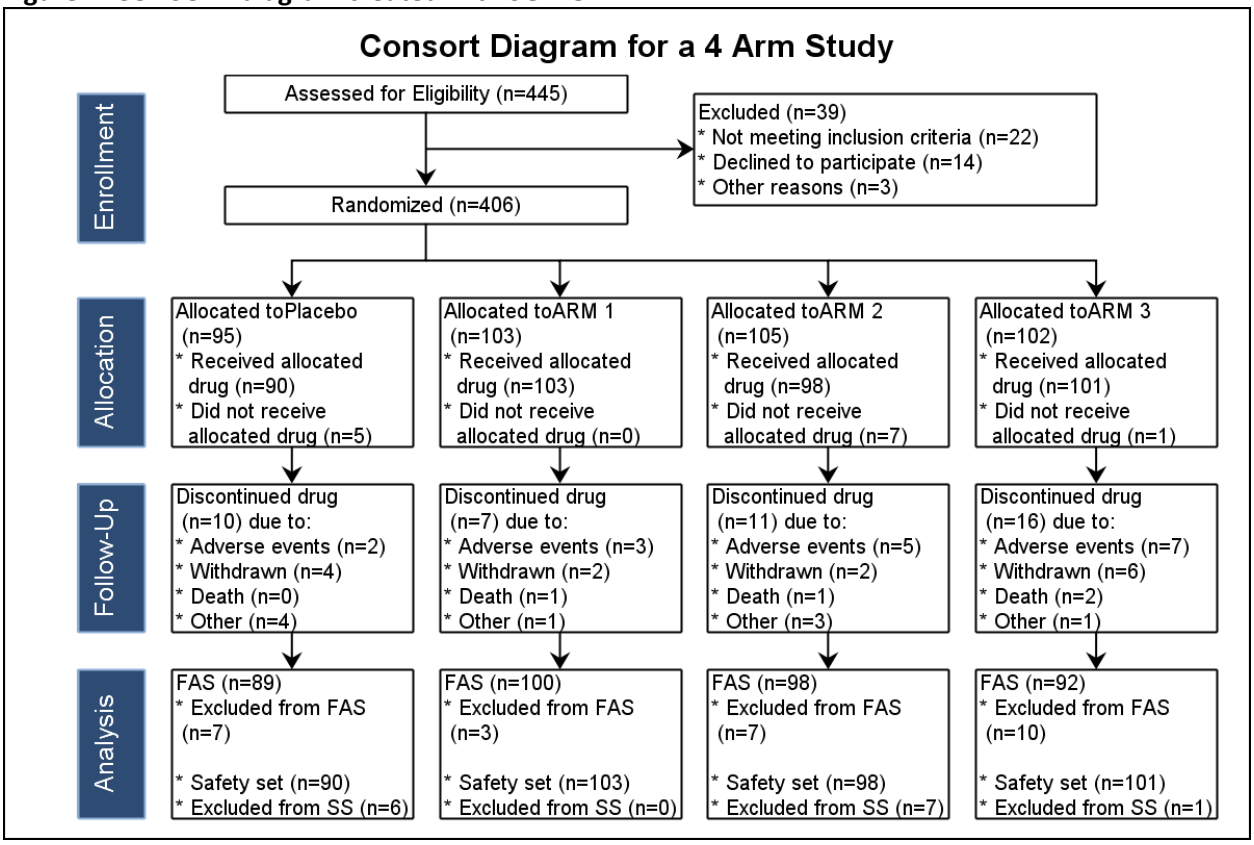
CONSORT Diagrams in SGPLOT: Adding Efficiencies

Shane Rosanbalm, Rho, Inc.

ABSTRACT

The PharmaSUG 2018 paper "CONSORT Diagrams with SG Procedures" was a major step forward in simplifying the process of creating CONSORT diagrams in SAS®. The SGPLOT-based approach superseded the previous RTF approach, which was adequate but tedious and quirky. Unfortunately, the SGPLOT approach requires the user to specify a lot of metadata. In order to create a diagram with 15 rectangles the user has to specify 404 numeric values over 223 lines of code. In this paper we explore options for reducing this specification burden. The proposed modifications result in the specification of 142 numeric values over 123 line of code.

Figure 1. CONSORT diagram created with SGPLOT



BACKGROUND

CONSORT diagrams are graphical representations of a disposition table. Several papers have been published about these diagrams in recent years.

- 2013: MWSUG paper proposes using an RTF template and TRANSTRN post-processing of the RTF to replace placeholder text with manually-entered subject counts.
- 2016: PhUSE paper proposes an efficiency to the RTF approach, adding some macro calls to make deriving the counts easier.
- 2016: Graphically Speaking blog provides a brief outline of an SGPLOT-based approach which removes the need for an RTF template.
- 2018: PharmaSUG features two separate paper presentations that fill in the gaps from the previous blog post.

The RTF approach was always a little bit quirky.

- Boxes and links do not snap together like in MS Visio or MS PowerPoint, meaning that there was always a lot of fiddling to get all of the shapes aligned and spaced properly.
- RTF files are edited in MS Word, meaning that sometimes when you edit the file, the text you see on the screen does not match the underlying text in the RTF file. When this happens it causes the TRANSTRN function to fail. This issue can be quite difficult to troubleshoot.
- If you want a different file format than RTF you have to perform some post-processing. This is particularly inconvenient if you want to do monthly/weekly/daily reruns for an online dashboard.

The SGPLOT approach solves the second and third issues. You can easily save the resulting diagram in a variety of file formats and you don't have to worry about strange MS Word latencies messing up your TRANSTRN functions.

INEFFICIENCIES

As for the fiddling to get the diagram to look right, the SGPLOT approach may actually be a small step backwards. In the RTF approach you simply drop a box on the page and drag the sides and corners until it looks right. In SGPLOT you have to specify the coordinates for the corners of each box in the diagram. Each box requires 8 numbers: an x and y value for each of the 4 corners. If you later decide that a box needs to move a little, you may end up having to modify all 8 numbers.

Adding to the inefficiency is the fact that the coordinates for the links (origins, insertions, elbows) are specified absolutely. Meaning if you move a box, you will have to update the coordinates for the corresponding links. Similarly, the coordinates for the text are specified absolutely. Again, if you move a box, you will have to update the coordinates of that box's text.

EFFICIENCIES

In order to get around some of these SGPLOT inefficiencies, we propose the following alternatives to how the boxes, links, and text are specified.

- Specify boxes with a 4-number summary: one coordinate, a width, and a height.
- Specify links by naming the box IDs to connect.
- Specify text location using a box ID.

We then use a set of macros to convert these alternative specifications back to the original formats called for in the SGPLOT approach.

BOXES

Using an 8-number summary for a box is sufficient but not necessary. We propose to instead specify where the `center-top` of the box should be located (one coordinate pair), along with a `width` and `height`. We then let the `%makeboxes` macro calculate where the corners of the boxes would be located. The macro's output dataset is a recreation of the required `emptyBoxes` dataset.

Figure 2. Switching from an 8-number summary to a 4-number summary

```
/**
 * Empty Box Data
 */
data emptyBoxes;
   input epId xEp yEp;
   datalines;
 1  15  200
 1  45  200
 1  45  190
 1  15  190
 2  15  170
 2  45  170
 2  45  160
 2  15  160
 3  50  195
 3  80  195
 3  80  165
 3  50  165
 4  11  140
 4  29  140
 4  29  100
 4  11  100
 5  31  140
 5  49  140
 5  49  100
 5  31  100
 6  51  140
 6  69  140
```

```
*-------------------------------------------;
*---------- empty boxes creation ----------;
*-------------------------------------------;

data emptyInfo;
   infile datalines dsd;
   input boxId center top width height;
   datalines;
 1, 30,200, 30,10
 2, 30,170, 30,10
 3, 65,195, 30,30
11, 20,140, 18,40
12, 40,140, 18,40
13, 60,140, 18,40
14, 80,140, 18,40
21, 20, 90, 18,40
22, 40, 90, 18,40
23, 60, 90, 18,40
24, 80, 90, 18,40
31, 20, 40, 18,40
32, 40, 40, 18,40
33, 60, 40, 18,40
34, 80, 40, 18,40
;
run;

%makeboxes(data=emptyInfo,type=empty);
```

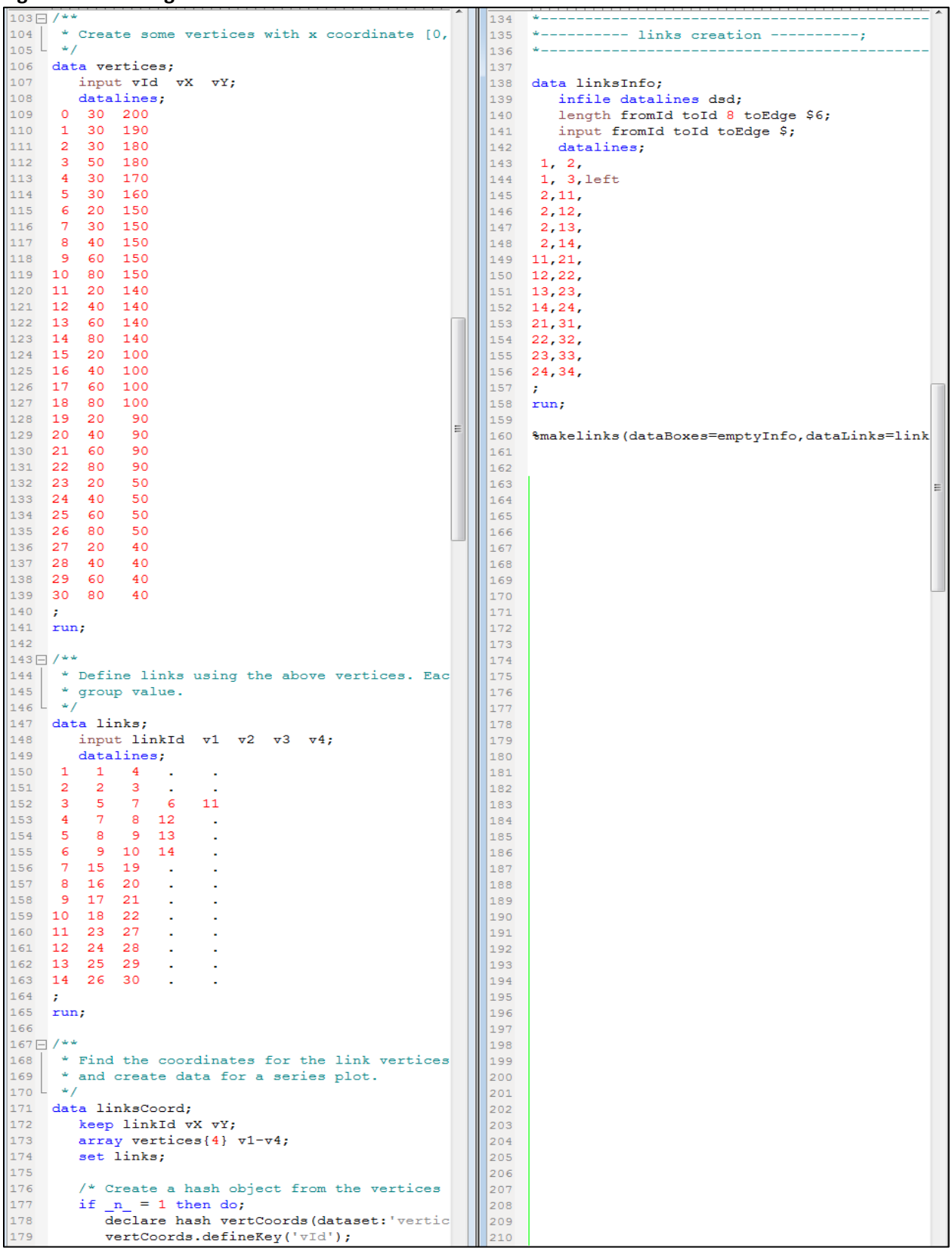
By switching from an 8-number summary to a 4-number summary we only have to specify half as much information.

That's nice, but it's not the best part of the 4-number summary approach. Imagine wanting to shift a box a little to the left. In the 8-number summary you will have to edit the x coordinate of all 4 corners. In the 4-number summary you will only have to edit the value of the variable `center`. Changing either the size or location of a box is always going to be easier using a 4-number summary.

LINKS

The SGPLOT code for specifying links is very clever. It creates a suite of vertices and then uses hash objects to reference the vertices and build a links coordinates dataset. We propose to abandon coordinates altogether when specifying links and instead simply name the boxes that are to be linked (e.g., to connect boxes 1 and 2 we would specify: `fromID`=1, `toID`=2). We then let the macro `%makelinks` calculate where the vertices should be located. The macro's output dataset is a recreation of the required `linksCoord` dataset.

Figure 3. Switching from vertices to boxID values



```
/**
 * Create some vertices with x coordinate [0,
 */
data vertices;
   input vId  vX  vY;
   datalines;
 0  30  200
 1  30  190
 2  30  180
 3  50  180
 4  30  170
 5  30  160
 6  20  150
 7  30  150
 8  40  150
 9  60  150
10  80  150
11  20  140
12  40  140
13  60  140
14  80  140
15  20  100
16  40  100
17  60  100
18  80  100
19  20   90
20  40   90
21  60   90
22  80   90
23  20   50
24  40   50
25  60   50
26  80   50
27  20   40
28  40   40
29  60   40
30  80   40
;
run;

/**
 * Define links using the above vertices. Eac
 * group value.
 */
data links;
   input linkId  v1  v2  v3  v4;
   datalines;
 1   1   4   .   .
 2   2   3   .   .
 3   5   7   6  11
 4   7   8  12   .
 5   8   9  13   .
 6   9  10  14   .
 7  15  19   .   .
 8  16  20   .   .
 9  17  21   .   .
10  18  22   .   .
11  23  27   .   .
12  24  28   .   .
13  25  29   .   .
14  26  30   .   .
;
run;

/**
 * Find the coordinates for the link vertices
 * and create data for a series plot.
 */
data linksCoord;
   keep linkId vX vY;
   array vertices{4} v1-v4;
   set links;

   /* Create a hash object from the vertices
   if _n_ = 1 then do;
      declare hash vertCoords(dataset:'vertic
      vertCoords.defineKey('vId');
```

```
*-----------------------------------------;
*---------- links creation ----------;
*-----------------------------------------;

data linksInfo;
   infile datalines dsd;
   length fromId toId 8 toEdge $6;
   input fromId toId toEdge $;
   datalines;
 1, 2,
 1, 3,left
 2,11,
 2,12,
 2,13,
 2,14,
11,21,
12,22,
13,23,
14,24,
21,31,
22,32,
23,33,
24,34,
;
run;

%makelinks(dataBoxes=emptyInfo,dataLinks=link
```

The front end benefit of this approach is that the number of lines of code needed is reduced from 74 to 22.

But that pales in comparison to the back end benefit. Because the links are defined relative to the boxes, if you were ever to move a box, the `%makelinks` macro would automatically update where the corresponding links were drawn.

TEXT

The SGPLOT code for specifying the text requires that you supply coordinates. We propose to instead name the box that the text goes with (e.g., `boxID`=1). We then let the macro `%positiontext` calculate where the text should be located. The macro's output dataset is a recreation of the required `vText` dataset.

Figure 4. Switching from coordinates to boxID values

```
/**
  * Vertical text for stage labels
  */
data vText;
   input xVt yVt vtext $10-75;
   datalines;
6  175   Enrollment
6  120   Allocation
6   70   Follow-Up
6   20   Analysis
;
run;
```

```
*-------------------------------------------
*---------- position vertical text ----------
*-------------------------------------------

data vTextInfo;
   input boxId vtext $10-75;
   datalines;
00       Enrollment
10       Allocation
20       Follow-Up
30       Analysis
;
run;

%positiontext
   (dataText=vTextInfo
   ,dataBoxes=filledInfo
   ,out=vText
   ,rotate=90
   );
```

There is no clear front end benefit here. While we do benefit ever-so-slightly from only having to specify one number for each text string (a single `boxID` instead of a coordinate pair), we pay a penalty by having to add the `%positiontext` macro call.

The real benefit here is going to be on the back end. Because the text position is defined relative to a `boxID`, if you were ever to move the box, the `%positiontext` macro would automatically recalculate where the text should be written.

MISCELLANEOUS

There is one subtle change in the efficient code that bears mentioning. In the original paper the various box, link, and text datasets are combined using a MERGE statement. In the efficient code version the MERGE has been changed to a SET.

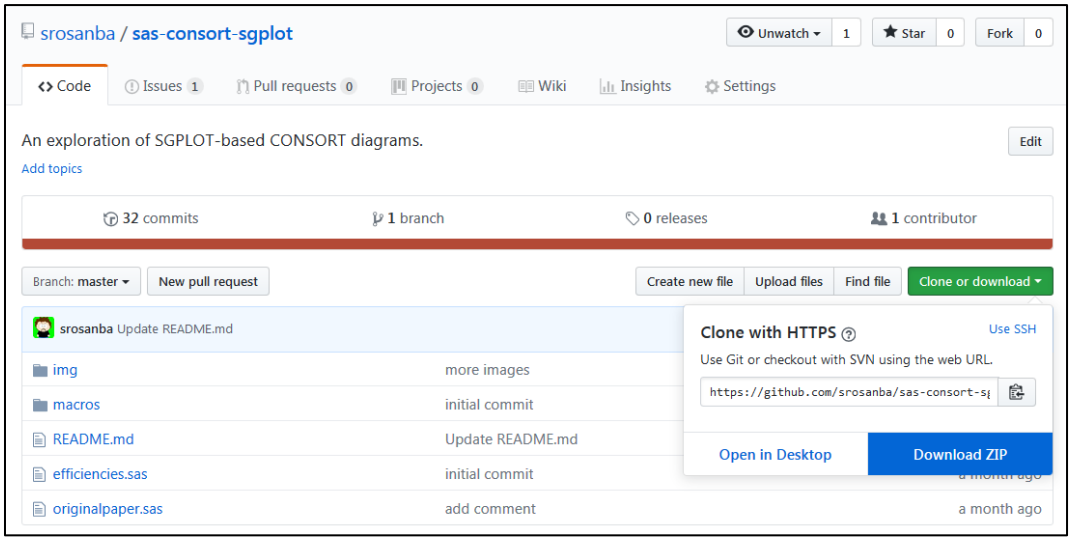
This change was necessary to support a prototyping feature that was added to the `%makeboxes` macro. Curious parties can go to the SGPLOT code to comment out the three original TEXT statements and comment in the TEXT statement that follows the comment `/* layout preview */`. This modification allows for rapid initial prototyping of the diagram layout prior to any text having been added. This may or may not be a useful feature.

SOURCE CODE

The source code for this presentation is available online at GitHub.

https://github.com/srosanba/sas-consort-sgplot

Figure 5. Screen shot of GitHub Repository



Under the green dropdown for [Clone or Download] select [Download ZIP]. Once the ZIP file has been extracted to your local drive, open the program `efficiencies.sas` and edit the first line to correspond to where the program has been extracted. From there you simply submit the program and watch the magic happen. ☺

CONCLUSION

Creating CONSORT diagrams with SGPLOT has a couple of key advantages over the RTF process. The only mild shortcoming of the previously published SGPLOT code what that it suffered from the inefficiency of having to specify a lot of coordinates, many of which were redundant. Using a suite of 3 macros allows for boxes to be specified with a simpler 4-number summary and for links and text to be positioned based on boxID values instead of coordinates. The result is less time spent setting up your initial diagram and less time spent updating your program down the road.

REFERENCES

Carpenter A. and Fisher D. 2013. "Reading and Writing RTF Documents as Data: Automatic Completion of CONSORT Flow Diagrams" *Proceedings of MWSUG 2018*. Available at https://www.mwsug.org/proceedings/2013/RX/MWSUG-2013-RX05.pdf

Mallavarapu A. and Shults D. 2016. "CONSORT Diagram: Doing it with SAS" *Proceedings of PhUSE 2016.* Available at https://www.lexjansen.com/phuse/2016/pp/PP03_ppt.pdf

Hebbar P. and Matange S. 2018. "CONSORT Diagrams with SG Procedures" *Proceedings of PharmaSUG 2018*. Available at https://www.lexjansen.com/pharmasug/2018/DV/PharmaSUG-2018-DV24.pdf

He H. 2018. "An Exploratory Way to Draw a Flow Diagram for the Subject Disposition of a Two-Arm Randomized Trial Using SAS® 9.4 M3" *Proceedings of PharmaSUG 2018*. Available at https://www.lexjansen.com/pharmasug/2018/AD/PharmaSUG-2018-AD28.pdf

CONTACT INFORMATION

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

Shane Rosanbalm
Rho, Inc.
srosanba@gmail.com