Paper SAS3149-2019

Create Multi-Arm CONSORT Diagrams the Easy Way with SAS®

Prashant Hebbar, SAS Institute Inc., and Sanjay Matange

ABSTRACT

In clinical trials, Consolidated Standards of Reporting Trials (CONSORT) flow diagrams are an important part of the randomized trials report. These diagrams present a bird's eye view of the flow of patients through the different stages of the trial. The SG procedures do not support a statement for drawing these diagrams. But by completing some data processing steps, you can draw commonly used CONSORT diagrams by using the SGPLOT procedure. You can generate the output in all the formats that are supported by the Output Delivery System (ODS)—and not just RTF! In this paper, we show you how to harness the power of the SGPLOT procedure to create these diagrams.

INTRODUCTION

A CONSORT diagram presents the flow of subjects at each stage in a clinical trial. A typical flow progression is enrollment, allocation to treatment, follow-up or disposition status, and analysis. This diagram is considered an important tool for assessing a trial. (Hopewell et al. 2011).

There are many previously published works for creating CONSORT diagrams with SAS®. Some of them used Rich Text Format (RTF) templates (Carpenter and Fisher 2012; Mallavarapu and Shults 2016).

A program to create CONSORT diagrams using the SGPLOT procedure without using RTF templates was previously published in the Graphically Speaking blog post titled "Outside-the-box: CONSORT diagram" (Matange 2016). After that, further improvements were presented in the paper "CONSORT Diagrams with SG Procedures" (Hebbar and Matange 2018) and the paper "Multi-Arm CONSORT Diagrams with SAS" (Matange and Hebbar 2018). Yet another variation is "CONSORT Diagrams in SGPLOT: Adding Efficiencies" (Rosanbalm 2018).

This paper is the next iteration of the paper "Multi-Arm CONSORT Diagrams with SAS" (Matange and Hebbar 2018), with some differences. We have restricted the input data to just the stage and arm labels and their counts. You do not need to provide row and column indices or place holder (dummy) observations. The diagram layout code has been refactored out of the input DATA step. We have also made some minor modifications to the program code for clarity. The program was run with SAS® 9.4M6. However, the program can be easily modified to use the features provided with earlier SAS® 9.4 releases.

The program code for this paper is available at https://github.com/sascommunities/sas-global-forum-2019/tree/master/3149-2019-Hebbar.

DIAGRAM DATA AND STRUCTURE

The process used in this paper targets the commonly used structure and layout for CONSORT diagrams in the industry. Examination of such diagrams on the web shows the use of a standard flow. These typically start with the Enrollment stage, which includes Assessment, Exclusion, and Randomization, followed by the Allocation, Follow-Up, and Analysis stages. The last three stages can have multiple columns, one for each arm. While the details of the diagrams might vary, the basic steps are essentially the same.

An example using the standard flow for a four-arm study, based on Mallavarapu and Shults (2016), is shown in Figure 1.

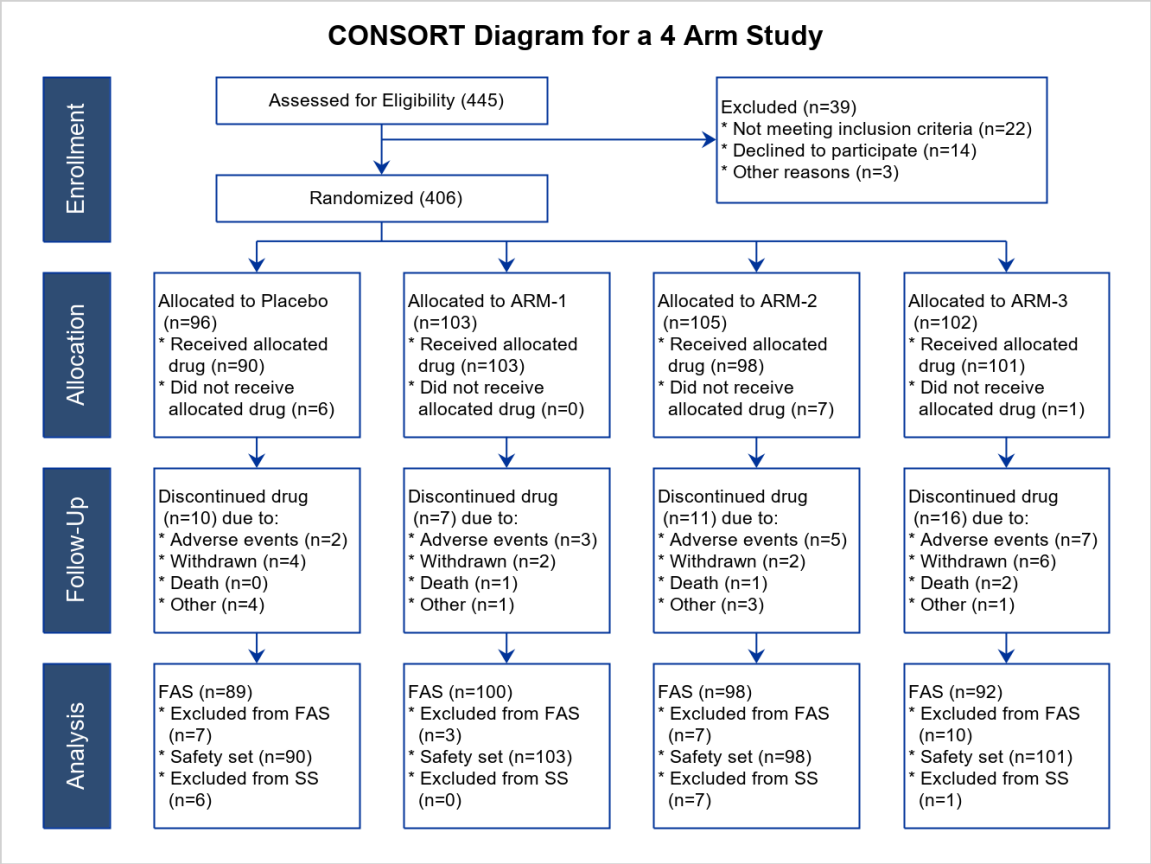

Figure 1. A Four-Arm CONSORT Diagram

As you can see, the CONSORT diagram is essentially composed of boxes linked by directed lines. The text in these boxes can be horizontal or vertical, either center- or left-aligned.

PLOT STATEMENTS

To implement these elements in PROC SGPLOT, you can use the following plot statements:

- POLYGON statement to draw the boxes (some with background fill)
- SERIES statement to draw the links and arrows
- TEXT statement to render the various text elements

FOUR-ARM CONSORT DIAGRAM

To describe how the program and macros work, it is useful to start with the end in mind. Figure 1 is a four-arm CONSORT diagram. To make the production of these diagrams easier, we are going to restrict ourselves to this standard diagram flow. Note the features of the diagram:

- There are four stages: Enrollment, Allocation, Follow-Up, and Analysis.
- Equal vertical space is provided for each stage, with a header on the left showing the stage name.
- The first stage, Enrollment, has three boxes (nodes) with custom layout and links.

- The bottom three stages, Allocation, Follow-Up, and Analysis, all have evenly spaced columns, one for each arm.
- Between the Enrollment and Allocation stages, the diagram has branched links to the multiple arms.

DATA PREPARATION

To prepare the data for the diagram, we assume a drawing area of [0,1] horizontal (X axis) space and [0, 1] vertical (Y axis) space, with the [0, 0] origin at the top left. Data sets for the link and box elements are created in this space. The layout of the diagram is set as a 5x5 grid, top to bottom and left to right, as shown in Figure 2, with columns labeled with the letter C and rows labeled with the letter R.

Vertically, each stage of the diagram is allocated 25% of the height, starting from the top. Figure 2 shows the positioning of the boxes for Enrollment, Allocation, Follow-Up, and Analysis, from top to bottom. The centerline for each of these stages is shown on the right in the figure.

Note that there is an extra Dummy stage between the top Enrollment stage and the Allocation stage. The centerline for this stage has the label shown in red. This is a placeholder for drawing the horizontal line connecting all four arms to the Randomized node and for dropping down the arrows to the boxes in the Allocation stage.

The Enrollment stage has three boxes for Assessed, Excluded, and Randomized that use customized placement. The input data part of the program uses macro variables to define the vertical positions of the three nodes as a fraction of the full diagram height. The horizontal locations of these nodes are generally positioned between the lower sets of columns. The header is positioned on the left, aligned with other headers.

Each of the lower three stages is located at equal vertical spacing of 0.25 of the diagram′s height, as shown in Figure 2. The centerline of each stage is shown along with a label on the right. A macro variable is used to set the spacing. Horizontally, a header for each stage is displayed at the left. The rest of the space is equally allocated to four arms.

Note that for the last four stages (Dummy, Allocation, Follow-Up, and Analysis), the columns are all on the grid shown in Figure 2, Their centerlines are located by Header, Placebo, Arm-1, Arm-2, and Arm-3. We do not create any boxes for the Dummy stage in the program.

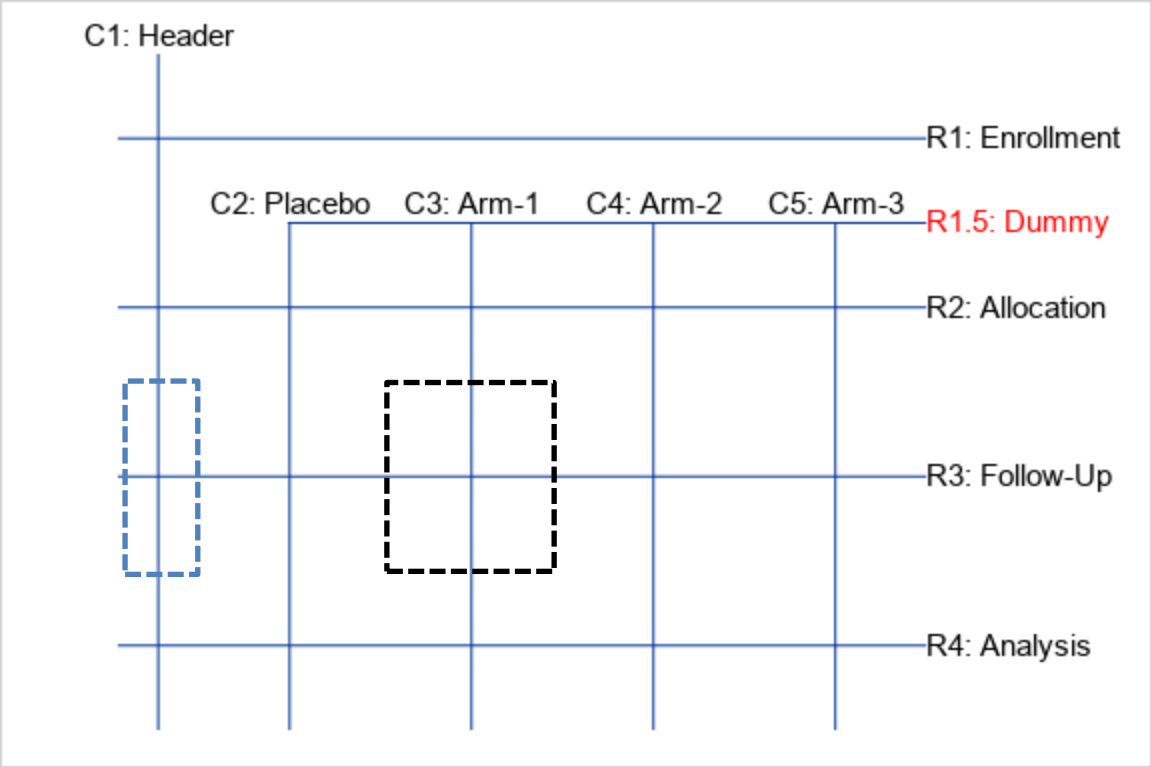

Figure 2. Layout Grid for a Four-Arm Diagram

PROGRAM STEPS

The program has three parts that are used in sequence:

1. Data input
2. Macro for computing the diagram data
3. Macro to render the diagram using the SGPLOT procedure

DATA INPUT

The user provides an input data set with the stage and arm labels, along with the counts to be shown in each box (*n*1 to *n*5). A snippet of input data for a four-arm study is shown in the following code:

```
input Stage $ Arm $ n1-n5;
datalines;     /* Your  n1   n2   n3  n4   n5 counts */
Enrollment  Assessed    445    .    .   .    .
Enrollment  Excluded     39   22   14   3    .
Enrollment  Randomized  406    .    .   .    .
Allocation  Placebo      96   90    6   .    .
Allocation  ARM-1       103  103    0   .    .
Allocation  ARM-2       105   98    7   .    .
Allocation  ARM-3       102  101    1   .    .
Follow-Up   Placebo      10    2    4   0    4
Follow-Up   ARM-1         7    3    2   1    1
Follow-Up   ARM-2        11    5    2   1    3
. . .
```

These observations must be ordered by stage and arm. The program keys off these standard label values. If your labels are different, you can modify the program to match them up. No other information is required in the input data set, unlike previous versions of this paper.

DIAGRAM DATA GENERATION

The %consortData (inData=, nColumns=, arms=, outData=) macro is used to compute the data for the diagram. This macro computes the diagram data from the input data set in multiple steps with row and column layout information, followed by node information and link information. These intermediate data sets are then combined to generate the final output data set. It has the following steps:

- A DATA step to augment your input data set with Row, Column variables. This step also adds a Header observation for every change in Stage value.
- A DATA step to calculate the width and height of all the boxes. This step also adds Dummy stage observations which are used in the link computations later.
- A DATA step to compute the center of the nodes for all stages of the diagram based on the previously generated Row and Column indices.
- A DATA step to compute the rectangle vertices for each node. These are for use by POLYGON statements to draw the boxes. Separate polygons are created for the filled and unfilled nodes. Columns (XF, YF) are for the filled polygon vertices and columns (X, Y) are the unfilled ones.

  This step also creates the text for each node using the n1-n5 counts in the input data set using some utility macros. These are for the labels and text inside the boxes, to be rendered using TEXT statements. Two columns are created: one for the vertical label of the headers (VLABEL), and another for the horizontal text (HLABEL).
- A DATA step to compute the points for the directed links between the nodes (XLINK, YLINK). This step also computes the points for the horizontal link that connects all the arms (XLINK2, YLINK2). These links will be drawn by SERIES statements.
- The node and link data are then combined into one data set.

A judicious set of macro variables is used here to specify drawing parameters such as row spacing, header widths and padding. A macro variable widthFix used for adjusting the Excluded box is also computed. We discuss this issue later in the paper.

The width and height calculations are shown in the following code snippet.

```
...
%let rowSpacing=0.25;
%let pad=0.02;
%let headerWidth=0.1;
%let nColumns=4;
%let yAssess=%sysevalf(0.2 * &rowSpacing);
%let yRandom=%sysevalf(0.7 * &rowSpacing);
%let yExclude=%sysevalf(0.4 * &rowSpacing);
%let yHeader=%sysevalf(0.5 * &rowSpacing);
    /*--Pad to increase the width of the Exclude box when columns=3--*/
%let widthFix = %sysevalf(0.05 * (4 - &nColumns));
    /* width for each arm column - excluding header column */
%let dx = %sysevalf((1 - &headerWidth)/&nColumns);

/*--Diagram Layout--*/
```

```
data layout;
  set &indata._aug end=last;
  /* Input counts data after adding row & col indices and header rows */

  h=&rowSpacing; /* Height */

  select (Arm); /* w = Width */
    when ('Header')
      w=&headerWidth;
    when ('Randomized', 'Assessed')
      do; w=(&nColumns-1)*dx/2; h=0.1; end;
    when ('Excluded')
      do; w=widthFix+(&nColumns-1)*dx/2; h=0.2; end;
    otherwise
      w=dx;
  end;
  output;

  /*--Generate the Row=Dummy obs between Enrollment and Allocation--*/
  if last then do;
     call missing (Arm, n1, n2, n3, n4, n5);
     Row=1.5; Stage='Dummy';  h = 2*&pad;
     do Col = 1 to 5; if (&arms > (Col - 2)) then output; end;
   end;
run;
```

The rest of data generation code has been omitted for brevity. The final step is to call the macro to render the diagram with the diagram data set we created.

RENDERING THE DIAGRAM

The %consortDiagram (diagData=, fillColor=) macro is used to render the diagram. The macro uses the diagram data set created by the %consortData() macro as its input and allows you to specify a fill color for the header boxes. The code for the macro uses the SGPLOT procedure step and is shown in the following code.

```
%macro consortDiagram(diagData= , fillColor=stgb);
   proc sgPlot data=&diagData noBorder noAutoLegend;
   /*--Filled boxes--*/
   polygon id=pid x=xf y=yf / fill outline
          fillAttrs=(color=&fillColor) lineAttrs=graphDataDefault;
   /*--vertical text--*/
   text x=xl y=yl text=vLabel / rotate=90 textAttrs=(size=9 color=white);
   /*--Empty boxes--*/
   polygon id=pid x=x y=y / lineAttrs=graphDataDefault;
   /*--horizontal text, both left and center aligned--*/
   text x=xl y=yl text=hLabel / splitChar='.' splitPolicy=splitAlways
          position=position;
   /*--Links with arrow heads--*/
   series x=xlink y=ylink / group=lid lineAttrs=graphDataDefault
          arrowHeadPos=end arrowHeadShape=barbed arrowHeadScale=0.4;
   /*--Links without arrow heads--*/
   series x=xlink2 y=ylink2 / group=lid2 lineAttrs=graphDataDefault;
   xAxis display=none min=0 max=1 offsetMin=0 offsetMax=0;
   yAxis display=none min=0 max=1 offsetMin=0 offsetMax=0 reverse;
   run;
```

```
%mend consortDiagram;
```

The details of the SGPLOT program are very straightforward, as seen in the code above. Recall that the diagram data coordinates we computed earlier are in normalized [0, 1] space in both dimensions, with the origin at the top left. To make this work correctly, you need to set OFFSETMIN= and OFFSETMAX= for the axes and set the REVERSE flag in the YAXIS statement.

VARIATIONS ON THE FOUR-ARM LAYOUT

Commonly used CONSORT diagrams have two, three, or four arms. Our program uses the nColumns macro parameter to set the number of columns to be used for the layout of the arms in the lower part of the diagram. The macro parameter arms= must match the number of arms (including placebo) that exist in your input data set.

Using these parameters, we can create CONSORT diagrams for the different combinations of arms and layout columns using the same program. Some of the possible combinations are as follows:

- arms=4, nColumns=4 (see Figure 1)
- arms=3, nColumns=4 (see Figure 3)
- arms=3, nColumns=3 (see Figure 4)
- arms=2, nColumns=4 (see Figure 5)

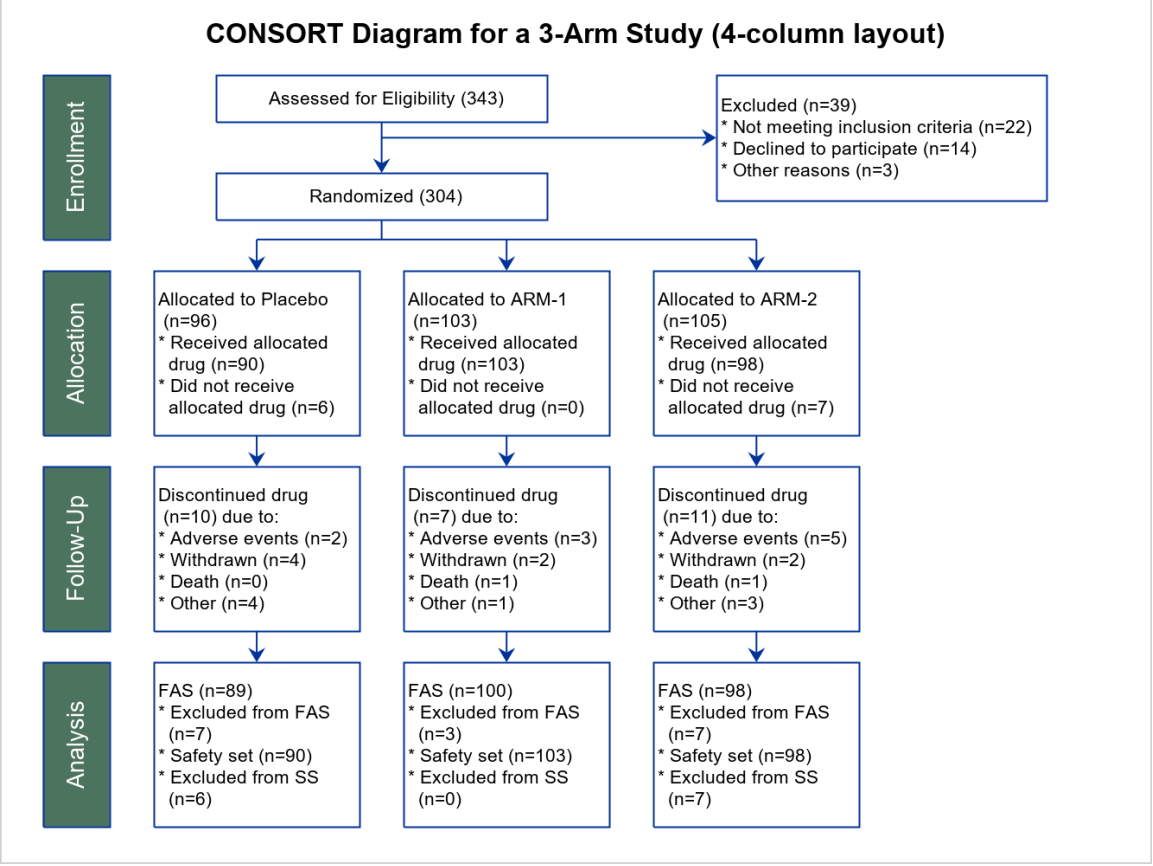

Figure 3. Three Arms, Four Columns (with Empty Space on the Right)

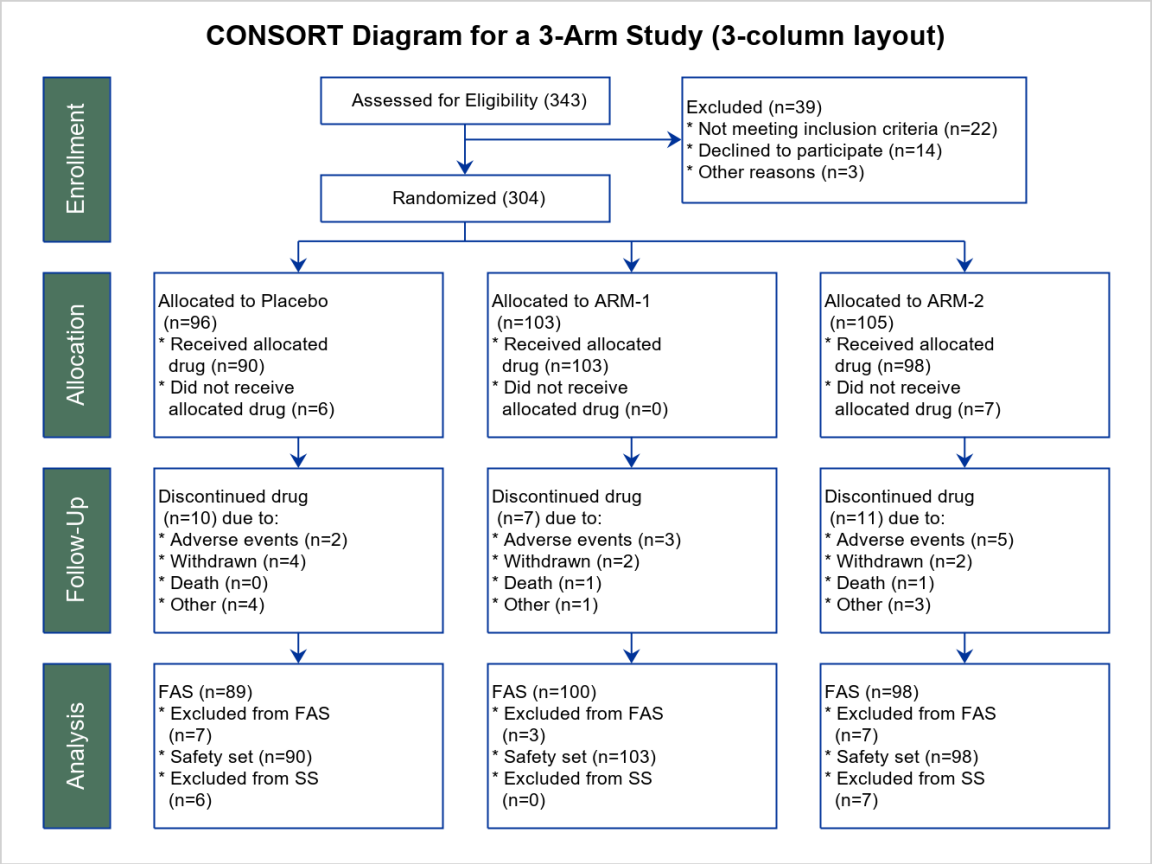

Figure 4. Three Arms, Three Columns (No Empty Space on the Right)

Note: The four-column layout positions the Enrollment stage nodes in the center of the lower pair of columns. The node widths are set to 1.5 times the width of the lower nodes. However, for a three-column layout, these values do not work well. In this case, we use the widthFix macro variable to adjust the nodes' positions and widths.

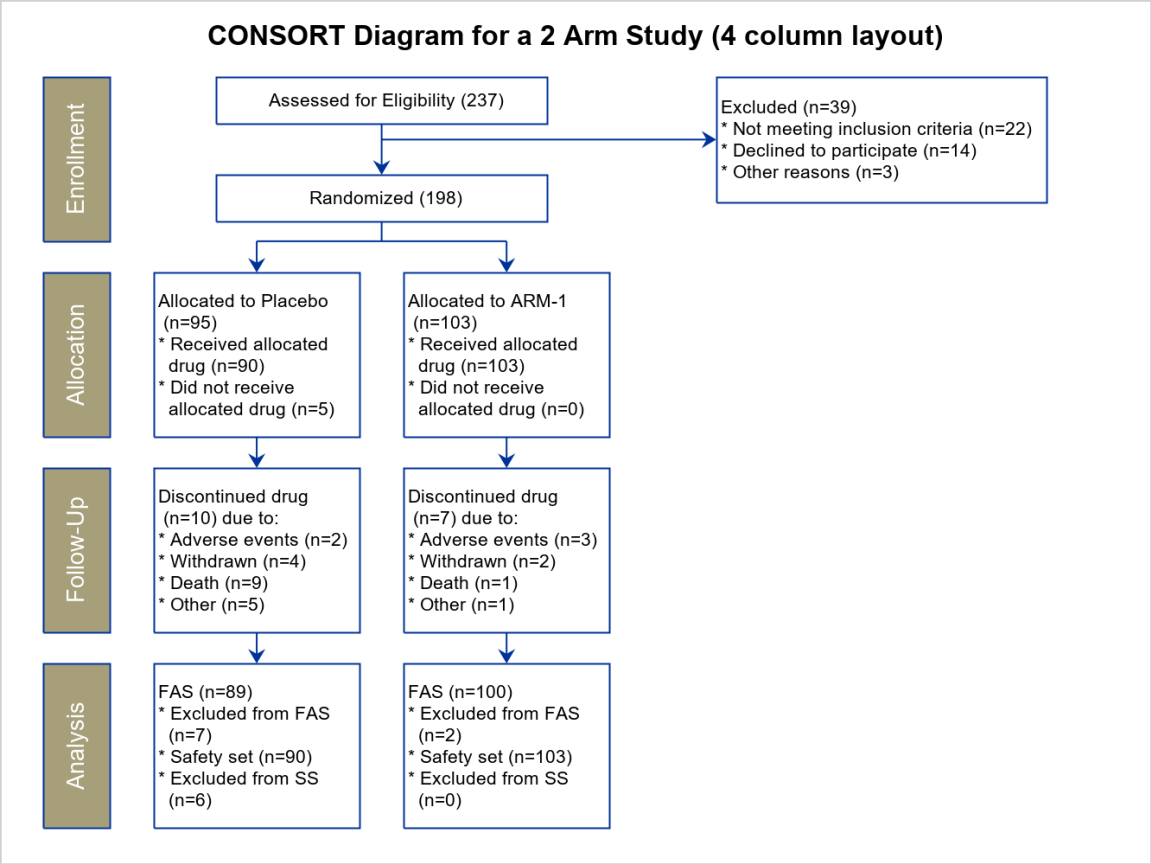

Figure 5. Two Arms, Four Columns (Lots of Empty Space on the Right)

CAVEATS

The program does not work well for the two-arm study data laid out in two columns. Also, if you reduce the number of columns (a three-column layout for a two-arm study, for example), the text rendering inside the boxes does not make use of the increased box widths, so there could be extra white space inside the boxes.

CONCLUSION

CONSORT diagrams can be created using only SAS code, thus reducing the complexity of using additional applications or template files. The output can be generated in any format supported by ODS Graphics. The computation of the data for the diagram has been separated from the rendering. As a result, once you compute the diagram data, you can render the diagram multiple times (and to multiple ODS destinations) without regenerating the data.

The diagrams can use layouts of three or four columns. You can render a three-arm diagram in a three-column layout to fill the space, or you can go for a larger column layout to create more space on the right side—the choice is yours! Also note that since PROC SGPLOT is based on the Graph Template Language (GTL), this graph can be similarly created using a GTL template and the SGRENDER procedure.

REFERENCES

Carpenter, A., and D. Fisher. 2012. "Reading and Writing RTF Documents as Data: Automatic Completion of CONSORT Flow Diagrams." *PharmaSUG 2012 Conference*

Proceedings. Philadelphia, PA: PharmaSUG. Available https://www.pharmasug.org/proceedings/2012/TF/PharmaSUG-2012-TF16.pdf.

Hebbar, P., and S. Matange. 2018. "CONSORT Diagrams with SG Procedures." *PharmaSUG 2018 Conference Proceedings*. Seattle, WA: PharmaSUG. Available https://www.pharmasug.org/proceedings/2018/DV/PharmaSUG-2018-DV24.pdf.

Hopewell, S., et al. 2011. "Reporting of participant flow diagrams in published reports of randomized trials." *Trials* 12:253.

Mallavarapu, A., and D. Shults. 2016. "CONSORT Diagram: Doing it with SAS." *Proceedings of PhUSE Connect Conference 2016*. Barcelona, Spain: PhUSE. Available http://www.phusewiki.org/docs/Conference%202016%20PP%20Papers/PP03.pdf.

Matange, Sanjay. "Graphically Speaking." Available https://blogs.sas.com/content/graphicallyspeaking/2016/10/20/outside-box-consort-diagram/. Last modified October 20, 2016. Accessed on March 6, 2019.

Matange, S., and D. Heath. 2011. *Statistical Graphics Procedures by Example: Effective Graphs Using SAS®*. Cary, NC: SAS Institute Inc.

Matange, S., and P. Hebbar. 2018. "Multi-Arm CONSORT Diagrams with SAS." *PharmaSUG China 2018 Conference Proceedings*. Beijing, China: PharmaSUG China. Available http://www.pharmasug.org/proceedings/china2018/DV/Pharmasug-China-2018-DV52.pdf.

Rosanbalm, S. 2018. "CONSORT Diagrams in SGPLOT: Adding Efficiencies." *SAS Conference Proceedings: SouthEast SAS Users Group 2018*. Available https://www.lexjansen.com/sesug/2018/SESUG2018_Paper-271_Final_PDF.pdf.

ACKNOWLEDGMENTS

Many thanks to Sanjay Matange for the original idea and being the co-author on this paper. Thanks to Jason Secosky for an elegant solution to a DATA step issue in the program code.

RECOMMENDED READING

- *SAS® 9.4 ODS Graphics: Procedures Guide*

CONTACT INFORMATION

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

Prashant Hebbar
SAS Institute, Inc.
prashant.hebbar@sas.com

Sanjay Matange
sanjaymatange@gmail.com