PharmaSUG 2018 - Paper DV-24

CONSORT Diagrams with SG Procedures

Prashant Hebbar and Sanjay Matange, SAS Institute Inc., Cary, NC

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 with some data processing steps and tricks, we can draw CONSORT diagrams using SG Procedures. In this paper, we show how to harness power of SG Procedures to create these diagrams.

INTRODUCTION

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

Previously published work for drawing these diagrams with SAS primarily used Rich Text Format (RTF) templates (Carpenter and Fisher, 2012; Mallavarapu and Shults, 2016). Here we show how to use an SG procedure, namely the SGPLOT procedure, to draw a CONSORT diagram. We assume that the requisite counts used in the diagram have already been computed. Some DATA step pre-processing is needed to create the data set for the graph.

The SG procedures debuted with ODS Graphics in SAS® 9.2. They offer a simple interface to using the Graph Template Language (GTL) that underlies the ODS Graphics system. The SGPLOT procedure lets you create single-celled scatter plots, series plots, box plots and more in a quick and simple manner.

The program in this paper is based on a previously published *Graphically Speaking* blog post titled "Outside-the-box: CONSORT diagram" at https://blogs.sas.com/content/graphicallyspeaking/2016/10/20/outside-box-consort-diagram/. It was tested with SAS 9.4M3 and newer releases.

CONSORT DIAGRAM DATA AND STRUCTURE

We are going to recreate the diagram shown in Mallavarapu and Shults (2016). This figure is for a 4 arm study. At its essence, the CONSORT diagram is composed of boxes, with or without colored background that may be linked by directed lines. The text in these boxes may be horizontal or vertical with center or left alignment.

PLOT STATEMENTS

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

- POLYGON statement to draw the boxes (with background fill and without).
- SERIES statement to draw the links and arrows.
- TEXT statement to render the various text elements.

DATA PREPARATION

The data for the diagram is based on assuming a data area of [0,100] horizontal (X axis) space and [0, 200] vertical (Y axis) space. Data sets for the links, boxes and text elements are created in this coordinate space.

Links Data

Vertices for the links are defined in the vertices data set as per the connections needed between the boxes in the diagram. This data set has 3 variables: a vertex identifier vId and its coordinates (vX, vY).

Next, a links data set is created by defining each link as a multi-segment line in terms of the previously defined vertex id vId in the vertices data set. This data set has 5 variables: linkId, v1-v4. For the diagram at hand, a maximum of 4 vertices is sufficient, but more can be added if needed.

We then create another links data set (linksCoord) that contains the (vX, vY) coordinates for the corresponding vertices from the vertices data set. A data set hash object populated from the vertices data set is used to look up the coordinates by the vertex id (vId). These variables are used for a SERIES plot statement with an arrow head at the end of the series.

Note that the two-step data generation is simply for convenience in diagramming the links. We can also create the linksCoord data set by directly specifying the link vertex coordinates.

Data for the Boxes

Two separate data sets are defined for the empty rectangles and the filled rectangles. You can then use two POLYGON statements: one for the empty boxes and another with FILL and FILLATTRS= options for boxes with background color for the phase labels on the left. The variables have been defined directly, but they could easily be based on vertex ids and then resolved to their coordinates via the hash object, like we did for the link data.

Text Data

Similarly, text is defined in three data sets, one for the rotated text in the phase labels, one for the center-aligned text and one for the left-aligned text. While the phase labels are simple to define, the horizontal text have varying counts embedded in the same text for a given phase row. You can use the CATS() function to generate these values by combining the text with count variables. This allows for easy reuse with a different set of counts for other studies. Each text observation also has x and y coordinate variables.

These variables are used in three separate TEXT statements to populate the text inside the boxes. The phase labels are rotated via the ROTATE=90 option. Note that we have embedded the '.' character judiciously in the text at positions where we want to split the text. The split is achieved by using the FITPOLICY=SPLITALWAYS with the SPLITCHAR="." options on the TEXT statements.

Once again, the coordinates for the text data have been specified directly, but they could have been defined as vertex ids and resolved by the hash object for coordinates.

Consolidated Graph Data Set

The last step in the data preparation is to combine all the data sets generated so far. Since we have used distinct variables for each plot, we can merge all the data sets to create the final consort data set. This combined data set is then used with PROC SGPLOT.

PROC SGPLOT

The code for PROC SGPLOT is fairly simple and shown below:

```
proc sgplot data=consort noborder noautolegend;

          /* lines connecting boxes, with arrows */
  series x=vX y=vY / group=linkid lineattrs=graphdatadefault
          arrowheadpos=end arrowheadshape=barbed arrowheadscale=0.4;

  polygon id=epid x=xEp y=yEp;               /* Empty boxes */
  polygon id=fpid x=xFp y=yFp / fill outline /* Filled boxes */
          fillattrs=(color=STGB) lineattrs=(color=VLIGB);
          /* horizontal text, centered */
  text x=xHtc y=yHtc text=hTextC / splitchar='.' splitpolicy=splitalways;
          /* horizontal text, left aligned */
  text x=xHtl y=yHtl text=hTextl / splitchar='.' splitpolicy=splitalways
                                   position=right;
          /* vertical text */
  text x=xVt y=yVt text=vtext / rotate=90 textattrs=(size=9 color=white);

  xaxis display=none min=0 max=90 offsetmin=0 offsetmax=0;  /* suppress */
  yaxis display=none min=0 max=200 offsetmin=0 offsetmax=0; /* suppress */
run;
```

Note that we have hidden the X and Y axes since they are not needed for the diagram.

SGPLOT OUTPUT

The listing destination output from the above program (at 200 DPI) is shown in Figure 1 below.

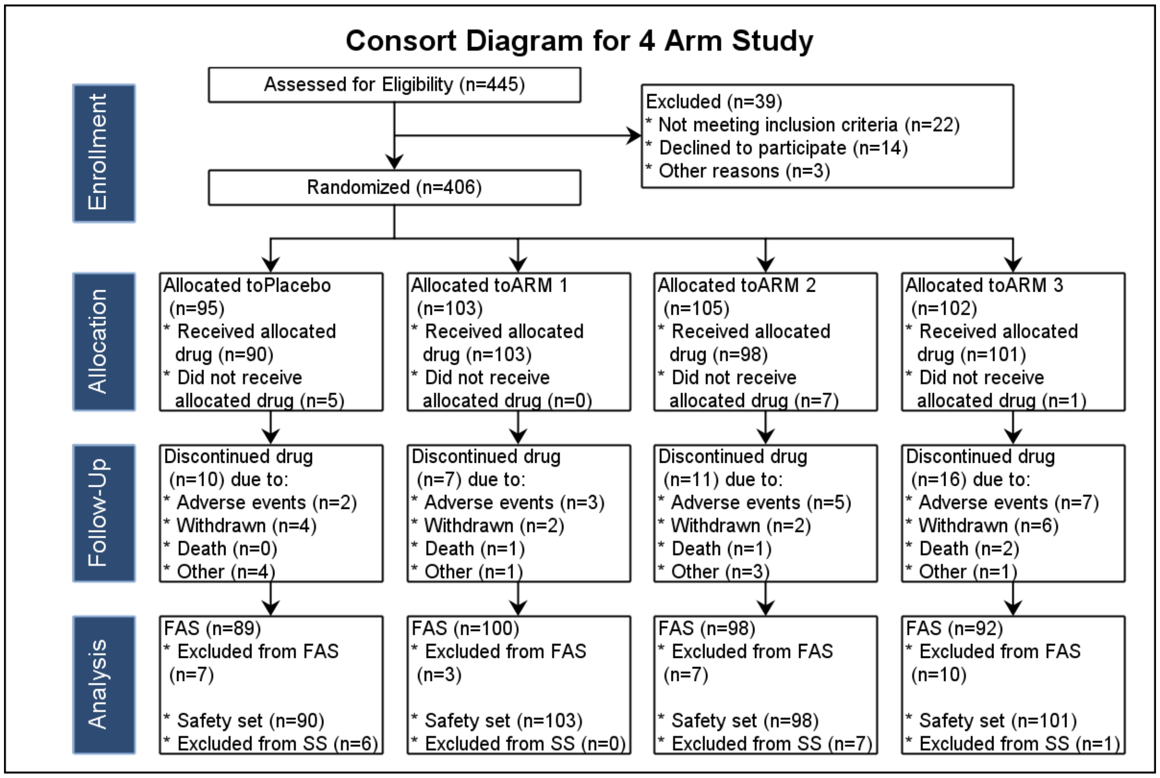
Consort Diagram for 4 Arm Study
Enrollment
Allocation
Follow-Up
Analysis
Assessed for Eligibility (n=445)
Excluded (n=39)
* Not meeting inclusion criteria (n=22)
* Declined to participate (n=14)
* Other reasons (n=3)
Randomized (n=406)
Allocated toPlacebo (n=95) * Received allocated drug (n=90) * Did not receive allocated drug (n=5)
Allocated toARM 1 (n=103) * Received allocated drug (n=103) * Did not receive allocated drug (n=0)
Allocated toARM 2 (n=105) * Received allocated drug (n=98) * Did not receive allocated drug (n=7)
Allocated toARM 3 (n=102) * Received allocated drug (n=101) * Did not receive allocated drug (n=1)
Discontinued drug (n=10) due to: * Adverse events (n=2) * Withdrawn (n=4) * Death (n=0) * Other (n=4)
Discontinued drug (n=7) due to: * Adverse events (n=3) * Withdrawn (n=2) * Death (n=1) * Other (n=1)
Discontinued drug (n=11) due to: * Adverse events (n=5) * Withdrawn (n=2) * Death (n=1) * Other (n=3)
Discontinued drug (n=16) due to: * Adverse events (n=7) * Withdrawn (n=6) * Death (n=2) * Other (n=1)
FAS (n=89) * Excluded from FAS (n=7) * Safety set (n=90) * Excluded from SS (n=6)
FAS (n=100) * Excluded from FAS (n=3) * Safety set (n=103) * Excluded from SS (n=0)
FAS (n=98) * Excluded from FAS (n=7) * Safety set (n=98) * Excluded from SS (n=7)
FAS (n=92) * Excluded from FAS (n=10) * Safety set (n=101) * Excluded from SS (n=1)

Figure 1. Consort Diagram Output from the SGPLOT Procedure.

CONCLUSION

We have demonstrated that a CONSORT diagram can be created using only SAS, reducing the complexity of using additional applications or template files. The output can be generated in a variety of formats supported by ODS Graphics.

Also note that since PROC SGPLOT is based on GTL, this graph can be similarly created using a GTL template and the SGRENDER procedure.

REFERENCES

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

Carpenter, A. and Fisher D. G. 2012. “Reading and Writing RTF Documents as Data: Automatic Completion of CONSORT Flow Diagrams.” *Proceedings of PharmaSUG 2012*, TF16. Available at https://www.pharmasug.org/proceedings/2012/TF/PharmaSUG-2012-TF16.pdf

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

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

Matange, Sanjay. “Graphically Speaking.” Available at http://blogs.sas.com/content/graphicallyspeaking . Accessed on March 12, 2018.

ACKNOWLEDGMENTS

Thanks to Warren Kuhfeld and Rick Langston for their help with the data step 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
SAS Institute, Inc.
sanjay.matange@sas.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.

APPENDIX – FULL SOURCE CODE

```
/**
 * Create some vertices with x coordinate [0,100], and y [0,200].
 */
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. Each link is assigned a separate
 * 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 from the vertices data set
 * 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 data set */
  if _n_ = 1 then do;
```

```
      declare hash vertCoords(dataset:'vertices');
      vertCoords.defineKey('vId');
      vertCoords.defineData('vX', 'vY');
      vertCoords.defineDone();
      call missing(vX, vY); /* avoid NOTE about uninitialized vars */
    end;

    /* Set vertex coordinates */
    do idx = 1 to dim(vertices); /* iterate over vertices{} */
      if vertices{idx} ne . then do;
        vId = vertices{idx};
        if vertCoords.find() eq 0 then
          output;
      end;
    end;
  run;

  /**
   * 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
   6  69  100
   6  51  100
   7  71  140
   7  89  140
   7  89  100
   7  71  100
   8  11   90
   8  29   90
   8  29   50
   8  11   50
   9  31   90
   9  49   90
```

```
 9  49   50
 9  31   50
10  51   90
10  69   90
10  69   50
10  51   50
11  71   90
11  89   90
11  89   50
11  71   50
12  11   40
12  29   40
12  29    0
12  11    0
13  31   40
13  49   40
13  49    0
13  31    0
14  51   40
14  69   40
14  69    0
14  51    0
15  71   40
15  89   40
15  89    0
15  71    0
;
run;

/**
 * Filled Box Data
 */
data filledBoxes;
  input fpId xFp yFp;
  datalines;
1   4  195
1   9  195
1   9  155
1   4  155
2   4  140
2   9  140
2   9  100
2   4  100
3   4   90
3   9   90
3   9   50
3   4   50
4   4   40
4   9   40
4   9    0
4   4    0
;
run;

/**
* Horizontal text, center aligned.
*/
```

```
data hTextC;
  input  xHtc   yHtc  htextc $10-75;
  datalines;
30  195   Assessed for Eligibility (n=445)
30  165   Randomized (n=406)
;
run;

/**
 * Horizontal text, left aligned. With help from Warren Kuhfeld.
 */
data hTextL(drop=type n1-n5 arm);
  length type $12 hTextL $125;
  input xHtl yHtl type $ arm $ 20-27 n1-n5;
  infile datalines missover;
  select (type);
    when ('Enrollment')
      hTextL = cats('Excluded (n=', n1,
                    ').* Not meeting inclusion criteria (n=', n2,
                    ').* Declined to participate (n=', n3,
                    ').* Other reasons (n=', n4, ')');
    when ('Allocation')
      hTextL = cats('Allocated to ', arm, '. (n=', n1,
                    ').* Received allocated.  drug (n=', n2,
                    ').* Did not receive.  allocated drug (n=', n3, ')');
    when ('Follow-Up')
      hTextL = cats('Discontinued drug. (n=', n1,
                    ') due to:.* Adverse events (n=', n2,
                    ').* Withdrawn (n=', n3,
                    ').* Death (n=', n4, ').* Other (n=', n5, ')');
    when ('Analysis')
      hTextL = cats('FAS (n=', n1,
                    ').* Excluded from FAS. (n=', n2,
                    '). .* Safety set (n=', n3,
                    ').* Excluded from SS (n=', n4, ')');
    otherwise;
  end;

  datalines;
50 180 Enrollment          39 22 14 3
11 120 Allocation  Placebo 95 90 5
31 120 Allocation  ARM 1   103 103 0
51 120 Allocation  ARM 2   105 98 7
71 120 Allocation  ARM 3   102 101 1
11  70 Follow-Up           10 2 4 0 4
31  70 Follow-Up           7 3 2 1 1
51  70 Follow-Up           11 5 2 1 3
71  70 Follow-Up           16 7 6 2 1
11  20 Analysis            89 7 90 6
31  20 Analysis            100 3 103 0
51  20 Analysis            98 7 98 7
71  20 Analysis            92 10 101 1
;
run;
```

```
/**
 * 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;

/**
 * Combine all graph data
 */
data consort;
  merge vertices linksCoord emptyBoxes filledBoxes hTextC hTextL vText;
run;

%let dpi=200;
ods listing image_dpi=&dpi;

/**
 * Draw the Consort Diagram
 */
ods graphics / reset width=6in height=4in imagename='Consort';
title 'Consort Diagram for a 4 Arm Study';

proc sgplot data=consort noborder noautolegend;
    /* lines connecting boxes, including arrows */
  series x=vX y=vY / group=linkid lineattrs=graphdatadefault
          arrowheadpos=end arrowheadshape=barbed arrowheadscale=0.4;
    /* Empty boxes */
  polygon id=epid x=xEp y=yEp;
    /* Filled boxes */
  polygon id=fpid x=xFp y=yFp / fill outline
          fillattrs=(color=STGB) lineattrs=(color=VLIGB);
    /* horizontal text, centered */
  text x=xHtc y=yHtc text=hTextC / splitchar='.' splitpolicy=splitalways;
    /* horizontal text, left aligned */
  text x=xHtl y=yHtl text=hTextl / splitchar='.' splitpolicy=splitalways
                                                     position=right;
    /* vertical text */
  text x=xVt y=yVt text=vtext / rotate=90 textattrs=(size=9 color=white);

  xaxis display=none min=0 max=90 offsetmin=0 offsetmax=0;
  yaxis display=none min=0 max=200 offsetmin=0 offsetmax=0;
run;

ods _all_ close;

/*** End program ***/
```