PharmaSUG 2018 - Paper AD-28

An Exploratory Way to Draw a Flow Diagram for the Subject Disposition of a Two-Arm Randomized Trial Using SAS® 9.4 M3

Huijuan (Tracy) He, Gilead Sciences, Inc

ABSTRACT

Many clinical trials are designed as two-arm randomized trials. Statistical programmers and biostatisticians work together to provide a flow diagram describing the disposition of subjects in the study. From the diagram, people may obtain information about the progress and status of the study visually, such as the number of subjects who were screened, randomized, dosed, discontinued or were ongoing in each arm. The CONSORT (Consolidated Standards of Reporting Trials) statement 2010 provides a flow diagram template to be filled out, available in PDF and DOC format. Users can download the template, modify as needed and manually input the numbers. Alternatively, the diagram can also be prepared using software such as Microsoft Visio® although manual entry is still required. These manual approaches can lead to data entry errors. While other figures prepared by programmers are typically generated by SAS® software, flow diagrams have rarely been produced in SAS®, let alone using basic DATA steps.

In this paper, using example data, we provide an exploratory way to draw the disposition of a two-arm trial as a flow diagram using SAS® 9.4 M3 with basic DATA steps and the SGPLOT procedure, where the counts in each box are determined programmatically via macro variables.

INTRODUCTION

Many pharmaceutical companies carry out similarly designed two-arm randomized trials for various drugs. In this paper, a dummy phase 3 two-arm randomized clinical trial was designed and conducted. Subjects were randomized to receive either DRUG A or DRUG B in two arms, respectively.

A flow diagram of disposition is used to present a visual description of progress in this trial. One option to create such a diagram is to obtain an editable template from the CONSORT statement 2010 website. One drawback of this approach is that the template is not very flexible. It takes a long time to manually retype and reorder in the template to meet the specific needs of a given study. Carpenter and Fisher (2012) presented a paper describing a SAS® macro that was used to read an RTF and rewrote the table as a completed CONSORT flow diagram. Mallavarapu and Shults (2016) from Cytel presented an automated way to create the diagram using an RTF template. Both approaches required an RTF template and solid programming knowledge.

Another option is using software such as Microsoft Visio®. Boxes/links and non-italic can be stored as a static template in Microsoft Visio® for usage of different trials but similar design, as shown in Figure 1. However, the user is still required to manually enter the drug names, counts and reasons, displayed as *italic* font here. In real life, the counts and reasons will change for each milestone of the study. For example, at week 24 there are likely more subjects who prematurely discontinued from treatment *DRUG A* than there were at week 12. *N11, N12,* or the list of *Reasons* will need to be updated. Additionally, for blinded studies, after unbinding, much of the content may change if a draft had been prepared using dummy treatment codes. Therefore, such tedious manual labor not only requires extra work, but is also prone to mistakes.

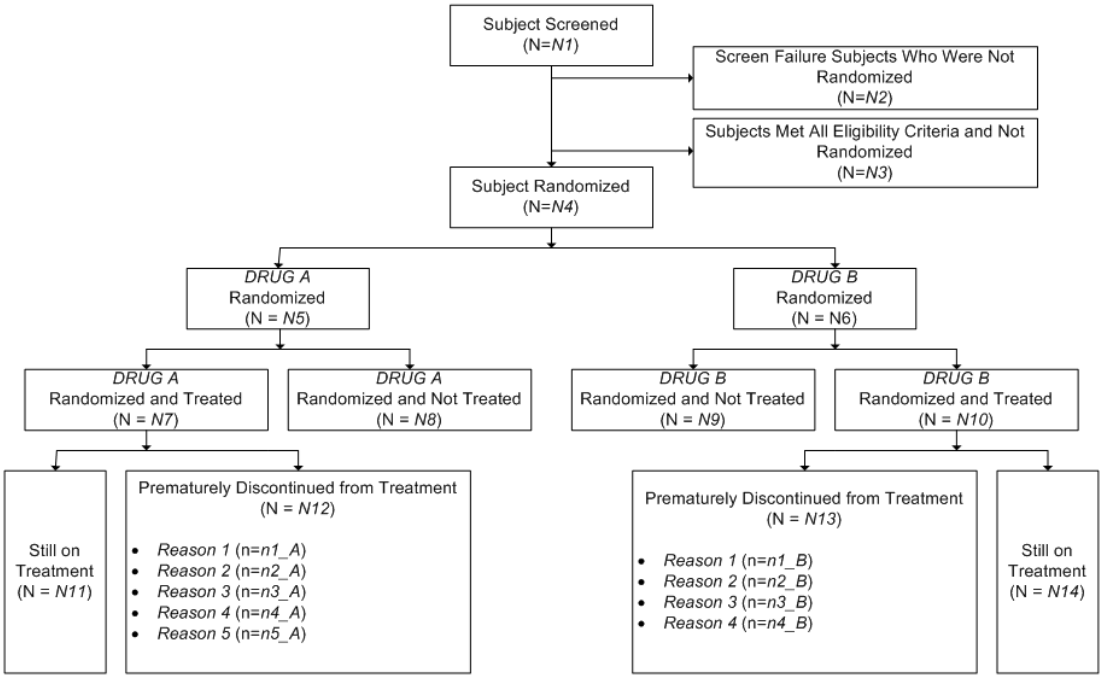

Figure 1. A flow diagram generated by Microsoft Visio®

People may need to manually "group" all items: text boxes and links with extra steps in order to drag the figure as a whole. Additionally, the figure's resolution could be negatively impacted if converting between formats (e.g. RTF/DOC to PDF). Therefore, the author of this paper brought up a challenging topic: Can SAS code be used exclusively to generate a flow diagram like the one shown in Figure 1?

Sanjay Matange (2016) posted an article "Outside-the-box: CONSORT diagram" in his blog at Graphically Speaking, to create the CONSORT diagram but fully with SAS 9.4 M3. The article gave a hint for this paper. In his article, a Hash Object is created to hold the node ids and their x and y coordinates to handle links and rectangles.

Many programmers may be unfamiliar with Hash Object. This paper presents an alternative construction that uses basic DATA steps and the SGPLOT procedure to create a flow diagram as Figure 1, in PDF or RTF format with high resolution. The structure is unaffected by zoom-in and out. For the italic drug names, counts and reasons, they can be held as separate macro variables and placed into the corresponding text box in the end.

OVERALL STRATEGY

Here is the overall strategy to draw the disposition flow diagram for two-arm trial using SAS®.

1) Understand the requirements and determine figure's layout. Draw a blueprint of the figure.
2) Point out xb and yb coordinates the boxes. Prepare BOX dataset by DATA steps in SAS®.
3) Point out xl and yl coordinates the links. Prepare LINK dataset by DATA steps in SAS®.
4) Generate the macro variables to be placed. Prepare CTEXT dataset for centered text, LTEXT dataset for left-aligned text by DATA steps in SAS®.
5) Combine BOX, LINK, CTEXT and LTEXT datasets and run the SGPLOT procedure.
6) Adjust the alignment (if needed) after checking layout.
7) Output the figure using Output Delivery System (ODS) in RTF, PDF format or both. Add title or footnote as requested.

STEP-BY-STEP PROCESS

1. LAYOUT AND BLUEPRINT

As a good work practice manner and also to avoid extra adjustment work afterwards, it is critical to determine the layout of the figure by checking study documents such as study's protocol, statistical analysis plan (SAP) and speaking with key stakeholders, such as study biostatistician. Some questions to consider are:

- How many categories as text boxes need to be shown?
- How to link the categories?
- What is the recommended contents and wording in each category?
- How should the figure be displayed (e.g. mirrored-symmetric or clonal-symmetric)?

The dummy example in this paper, like Figure 1, contains 14 boxes with pre-defined words in each category, using a mirrored-symmetric structure.

After determining the layout, we can draw a blueprint. This process may require a few iterations to reach a final/semi-final version of the figure. In the dummy example in this paper, such a blueprint appears in Figure 2. The highlighted parts are the "dynamic" macro variables such as drug names, counts and reasons of discontinuation. The standard title and footnote are also added in the draft if possible so that stakeholders can get a better idea about the visual layout of the final figure.

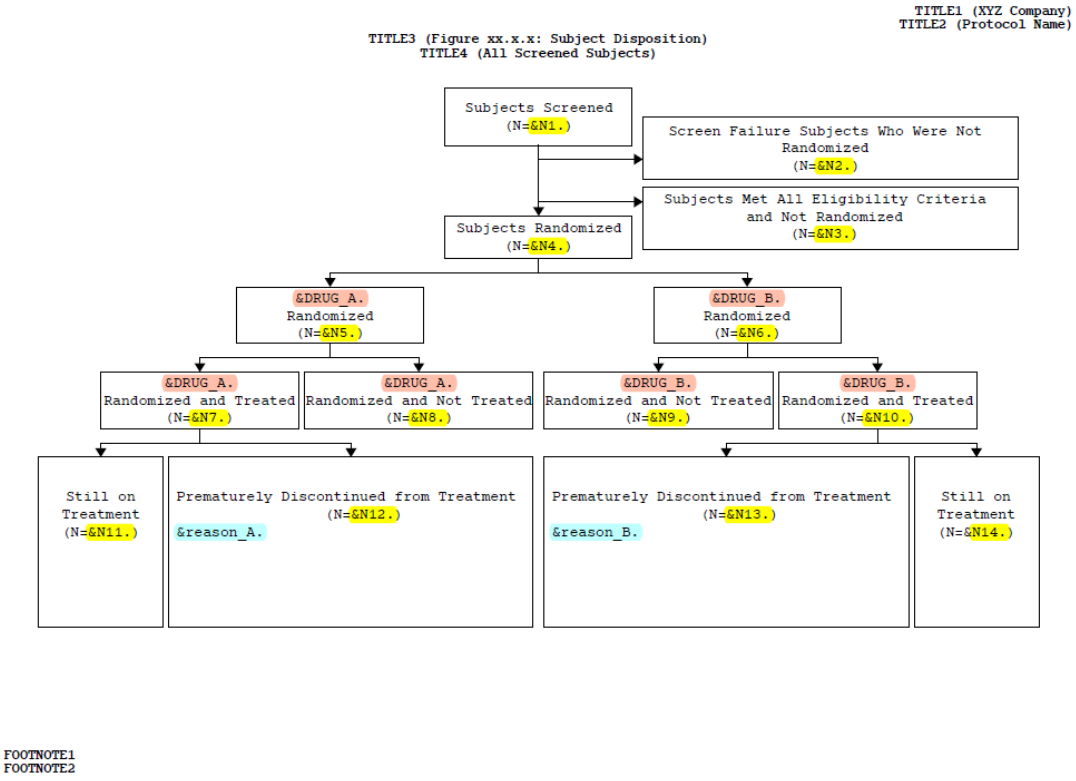

Figure 2. Blueprint before programming (on paper or using software)

2. BOX

In the dummy example in this paper, categories indicating different status of disposition are presented as rectangular boxes linked together. There are a total of 14 boxes displayed. We need to prepare a dataset containing three numeric variables: boxid, xb and yb that will be used in the POLYGON statement in the SGPLOT procedure. Coordinates (xb,yb) need to be identified for all 4 points of the rectangle. The width

(length) and height are going with the text to be fit properly in the box. Variable xb indicates the width's position and variable yb indicates the height's position. For Box 1 where variable boxid=1 as an example, "Subjects Screened" and "(N=&N1.)" are in two lines. Since the flow diagram is in a 0 to 200 vertical and 0 to100 horizontal space, in order to fit the phrase in Box1, the coordinates are assigned as (41,200), (59,200), (59,180) and (41, 180) in a clock-wise order to build a rectangle with a value of 18 as width and a value of 20 as height. Full coordinates for all boxes are pre-defined as Figure 4.

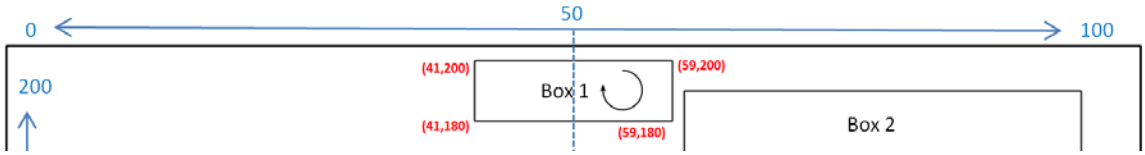

Figure 3. Assignment of Coordinates (xb, yb) using Box 1 as an example

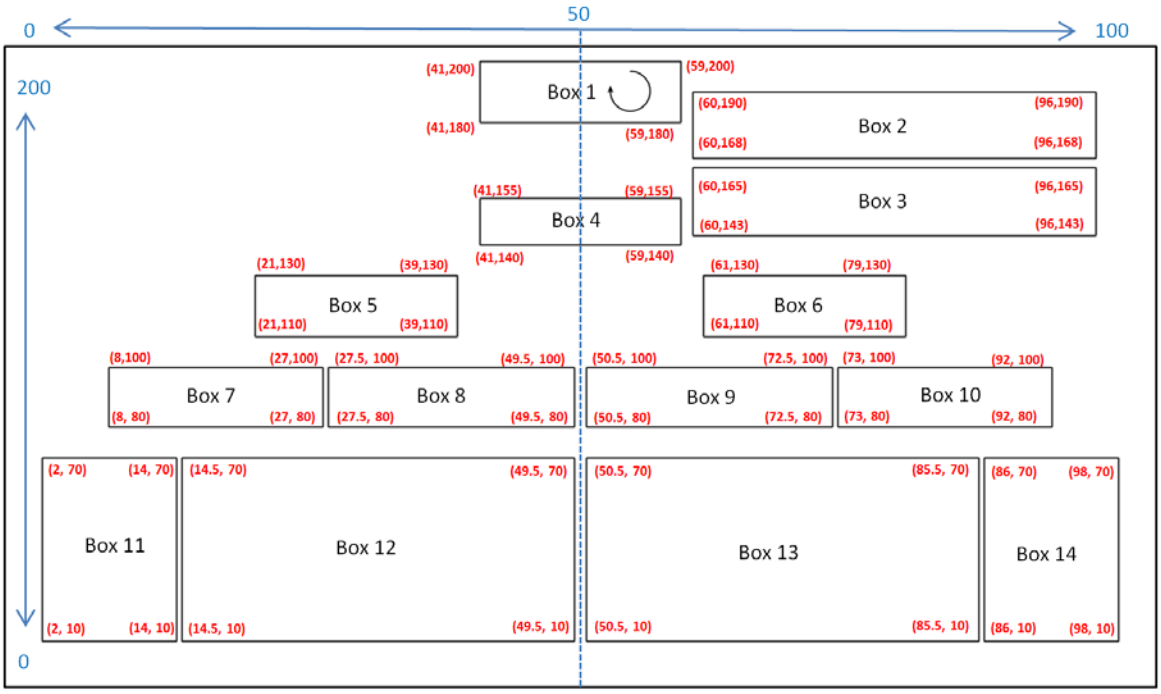

Figure 4. Full Boxes' Coordinates (xb, yb)

There are multiple ways to generate the BOX dataset in DATA steps. Here we use INPUT and DATALINES statement as below:

```
/*boxid xb    yb   boxid xb    yb  boxid xb    yb  boxid xb    yb*/
   data BOX;
   input boxid xb yb @@;
   datalines;
   1    41    200   1    59    200  1    59    180   1    41    180
   2    60    190   2    96    190  2    96    168   2    60    168
   3    60    165   3    96    165  3    96    143   3    60    143
   4    41    155   4    59    155  4    59    140   4    41    140
   5    21    130   5    39    130  5    39    110   5    21    110
   6    61    130   6    79    130  6    79    110   6    61    110
   7    8     100   7    27    100  7    27    80    7    8     80
   8    27.5  100   8    49.5  100  8    49.5  80    8    27.5  80
   9    50.5  100   9    72.5  100  9    72.5  80    9    50.5  80
   10   73    100   10   92    100  10   92    80    10   73    80
   11   2     70    11   14    70   11   14    10    11   2     10
   12   14.5  70    12   49.5  70   12   49.5  10    12   14.5  10
   13   50.5  70    13   85.5  70   13   85.5  10    13   50.5  10
   14   86    70    14   98    70   14   98    10    14   86    10
;
   run;
```

3. LINK

Similar to the BOX dataset, a LINK dataset containing three numeric variables: linkid, xl and yl is needed for linking the boxes in the figure. For the example in this paper, there are two types of links.

One is an arrowed link having one start point and one arrowed end point, such as the link between Box 1 and Box 4 as Figure 5. In this case, we need to identify the two coordinates (xl,yl) with variable linkid=1. The coordinates are assigned as (50,180) for start point, and (50,155) for end point.

The other type is a more complicated link having one start point but two end points, with elbow connectors, such as the link from Box 4 to both Box 5 and Box 6 as Figure 6. The approach to assign the coordinates is to split variable linkid into two values: 4 and 5. In this case, we need to point out all coordinates for each node. When linkid=4, there are four nodes, (50,140), (50, 135), (30, 135) and (30, 130). When linkid=5, there are three nodes, (50, 135), (70, 135), (70, 130). Coordinate (50,135) shows twice for a joint connector in the very middle. Full coordinates for all links are pre-defined as Figure 7.

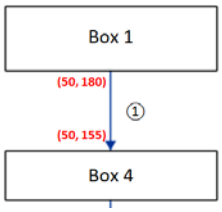

Figure 5. Link with One Start Point and One End Point

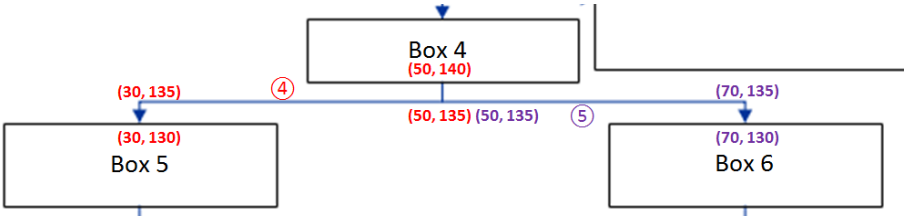

Figure 6. Link with One start Point and Two End Points with Elbow Connectors

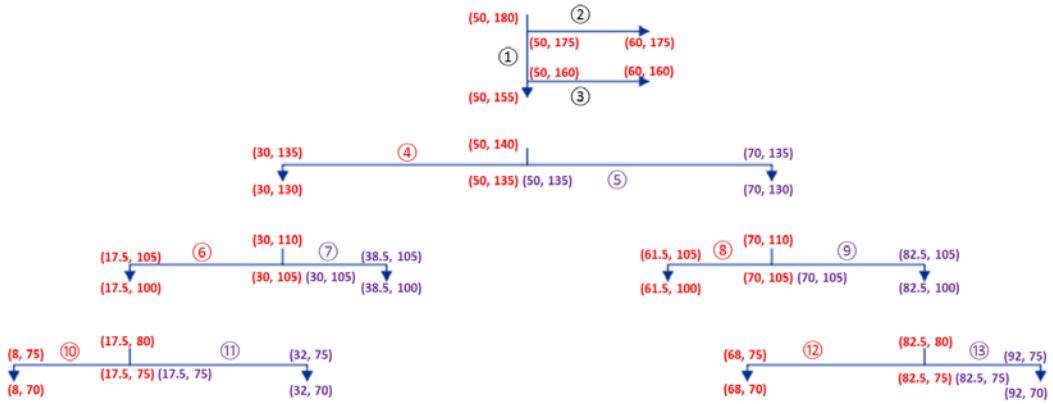

Figure 7. Full Links' Coordinates (xl, yl)

Similar as BOX dataset, to generate LINK dataset, we use INPUT and DATALINES statement as below:

```
/*linkid xl  yl   boxid xl    yl  linkid xl   yl  linkid xl    yl*/
   data LINK;
    input linkid xl yl @@;
    datalines;
  1    50    180   1    50    155
  2    50    175   2    60    175
  3    50    160   3    60    160
  4    50    140   4    50    135   4   30    135    4   30    130
  5    50    135   5    70    135   5   70    130
  6    30    110   6    30    105   6   17.5  105    6   17.5  100
  7    30    105   7    38.5  105   7   38.5  100
  8    70    110   8    70    105   8   61.5  105    8   61.5  100
  9    70    105   9    82.5  105   9   82.5  100
  10   17.5  80    10   17.5  75    10  8     75     10  8     70
  11   17.5  75    11   32    75    11  32    70
  12   82.5  80    12   82.5  75    12  68    75     12  68    70
  13   82.5  75    13   92    75    13  92    70
;
   run;
```

4. MACRO VARIABLES, CTEXT AND LTEXT

After creating the BOX and LINK datasets, the basic layout of the figure is complete, as shown in Figure 8. The next step is to fill out the content for each text box.

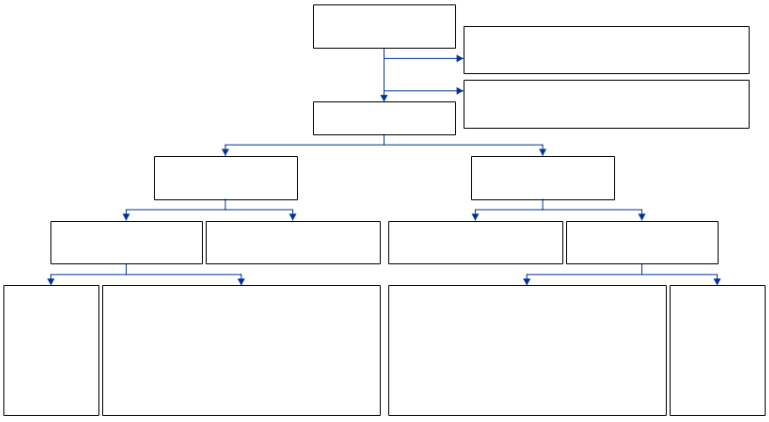

Figure 8. Layout of Empty Boxes and Links of Example Figure

There are a total of 14 text boxes to be populated. For box 12 and box 13, there are discontinuation reasons to be displayed as well. All counts and reasons are dynamic throughout the study so we need to store these counts in macro variables. Typically, we will have an ADaM Subject-Level Dataset (ADSL) that contains such information. Consider the following example ADSL:

```
   data ADSL;
     infile datalines missover;
     input usubjid $ trt01p $6. trt01pn scrnfl $ scrnffl $ randfl $saffl $ eotstt $12. dctreas $;
     datalines;
     001 DRUG A 1 Y N Y Y ONGOING
     002         . Y N N N
     003 DRUG A 1 Y N Y Y DISCONTINUED Reason3
     004 DRUG A 1 Y N Y Y DISCONTINUED Reason1
     005 DRUG B 2 Y N Y Y DISCONTINUED Reason1
     006 DRUG B 2 Y N Y Y DISCONTINUED Reason2
     007 DRUG B 2 Y N Y Y ONGOING
     008 DRUG A 1 Y N Y Y ONGOING
     009 DRUG A 1 Y N Y Y DISCONTINUED Reason5
     010 DRUG B 2 Y N Y Y DISCONTINUED Reason2
     011 DRUG A 1 Y N Y N
     012 DRUG A 1 Y N Y Y ONGOING
     013 DRUG B 2 Y N Y Y ONGOING
     014 DRUG B 2 Y N Y Y ONGOING
     015 DRUG A 1 Y N Y Y DISCONTINUED Reason2
     016 DRUG A 1 Y N Y Y DISCONTINUED Reason3
     017 DRUG B 2 Y N Y Y DISCONTINUED Reason4
     018 DRUG A 1 Y N Y Y ONGOING
     019 DRUG B 2 Y N Y Y ONGOING
     020 DRUG A 1 Y N Y Y ONGOING
     021 DRUG B 2 Y N Y Y ONGOING
     022 DRUG A 1 Y N Y Y DISCONTINUED Reason2
     023 DRUG B 2 Y N Y Y DISCONTINUED Reason2
     024 DRUG A 1 Y N Y Y ONGOING
     025 DRUG B 2 Y N Y Y ONGOING
     026 DRUG B 2 Y N Y Y ONGOING
     027 DRUG B 2 Y N Y Y ONGOING
     028 DRUG A 1 Y N Y Y DISCONTINUED Reason1
     029 DRUG A 1 Y N Y Y DISCONTINUED Reason2
     030 DRUG B 2 Y N Y Y ONGOING
     031         . Y Y N N
     032         . Y N N N
     033 DRUG B 2 Y N Y Y ONGOING
;
   run;
```

To generate the macro variables for drug name and counts of subjects in each text box, the SQL procedure is implemented as below using multiple SELECT INTO: statements in different conditions based on population indicator variables such as Screened Population Flag (SCRNFL), Screen Failure Flag (SCRNFFL), Randomized Population Flag (RANDFL), Safety Population Flag (SAFFL), treatment variables such as Planned Treatment for Period 01 (TRT01P) and Planned Treatment for Period 01 (N) (TRT01PN), and trial experience variables such as End of Treatment Status (EOTSTT), in dummy ADSL dataset .

```
proc sql noprint;
 select distinct(trt01p) into: DRUG_A from adsl where trt01pn=1;
 select distinct(trt01p) into: DRUG_B from adsl where trt01pn=2;
 select count(usubjid) into: N1  from adsl where SCRNFL="Y";
 select count(usubjid) into: N2  from adsl where SCRNFFL="Y" and RANDFL="N";
 select count(usubjid) into: N3  from adsl where SCRNFFL="N" and RANDFL="N";
 select count(usubjid) into: N4  from adsl where RANDFL="Y";
 select count(usubjid) into: N5  from adsl where RANDFL="Y" and trt01pn=1;
 select count(usubjid) into: N6  from adsl where RANDFL="Y" and trt01pn=2;
 select count(usubjid) into: N7  from adsl where SAFFL="Y" and trt01pn=1;
 select count(usubjid) into: N8  from adsl where SAFFL="N" and trt01pn=1;
 select count(usubjid) into: N9  from adsl where SAFFL="N" and trt01pn=2;
 select count(usubjid) into: N10 from adsl where SAFFL="Y" and trt01pn=2;
 select count(usubjid) into: N11 from adsl where SAFFL="Y" and trt01pn=1 and EOTSTT="ONGOING";
 select count(usubjid) into: N12 from adsl where SAFFL="Y" and trt01pn=1 and
                                                 EOTSTT="DISCONTINUED";
 select count(usubjid) into: N13 from adsl where SAFFL="Y" and trt01pn=2 and
                                                 EOTSTT="DISCONTINUED";
 select count(usubjid) into: N14 from adsl where SAFFL="Y" and trt01pn=2 and EOTSTT="ONGOING";
quit;
```

The ADSL dataset contains detailed information about the discontinuation reasons for each subject who discontinued treatment. Suppose the case report form (CRF) has only 6 possible reasons listed as “Reason1” to “Reason6”. Also suppose in the current analysis that no subject discontinued treatment due to “Reason6”. To generate the macro variables for the list of reasons for each arm, the FREQ and TRANSPOSE procedure are used as below to create list of reasons, concatenated by a short dash “-” as text characters. The CALL SYMPUT statement is used to generate macros “&REASON_A.” and “&REASON_B” for the reasons. “$” is the split character to be used later into the option SPLITCHAR= in the TEXT statement in order to wrap up the text lines in SGPLOT procedure.

The INDEX function in the codes is to find the key word in the term of reasons and ORD variable is created to order the list of reasons. In the example, for the literal “Reason1” to “Reason6” we use “1” to “6” as key words to order. The codes presented are only applicable when users want to display the existed reasons for each arm, which means “Reason6” is not showing as a discontinuation reason in both arms. If users want to display all reasons including missing ones, for example “Reason6 (n=0)”, we have to modify the codes, which is not presented in this paper.

```
proc freq data=adsl noprint;
  where EOTSTT="DISCONTINUED" and SAFFL="Y";
  table TRT01PN*trt01p*DCTREAS/missing out=dctreas;
run;

options missing="";
data dctreas;
  set dctreas;
  text=strip(DCTREAS)||" (n="||strip(put(COUNT,best.))||")";
  if index(upcase(DCTREAS),"1") then ord=1;
  if index(upcase(DCTREAS),"2") then ord=2;
  if index(upcase(DCTREAS),"3") then ord=3;
  if index(upcase(DCTREAS),"4") then ord=4;
  if index(upcase(DCTREAS),"5") then ord=5;
  if index(upcase(DCTREAS),"6") then ord=6;
run;
proc sort data=dctreas; by trt01pn ord; run;
proc transpose data=dctreas out=xdctreas;
  by trt01pn trt01p;
  id ord;
  var text;
run;
```

```
data xdctreas;
  length text $200;
  set xdctreas;
  text="$- "||catx("$- ", of _1 _2 _3 _4 _5 _6);
  call symput("reason_"||compress(trt01p,"DRUG "),strip(text));
run;
```

In Figure9, the texts in all boxes are displayed center-justified, except for boxes 12 and 13 that are displayed left-justified. Since later in the SGPLOT procedure, two TEXT statements are written for centered and left-aligned texts, two datasets CTEXT (for centered text) and LTEXT (for left-aligned text) are created using the DATA steps. The idea is to define coordinates (xt, yt) for the text in each box. Coordinates (xt,yt) of the texts are associated with pre-defined coordinates of boxes as shown in Figure 4. Full coordinates (xt, yt) in both CTEXT and LTEXT are as shown in Figure 9.

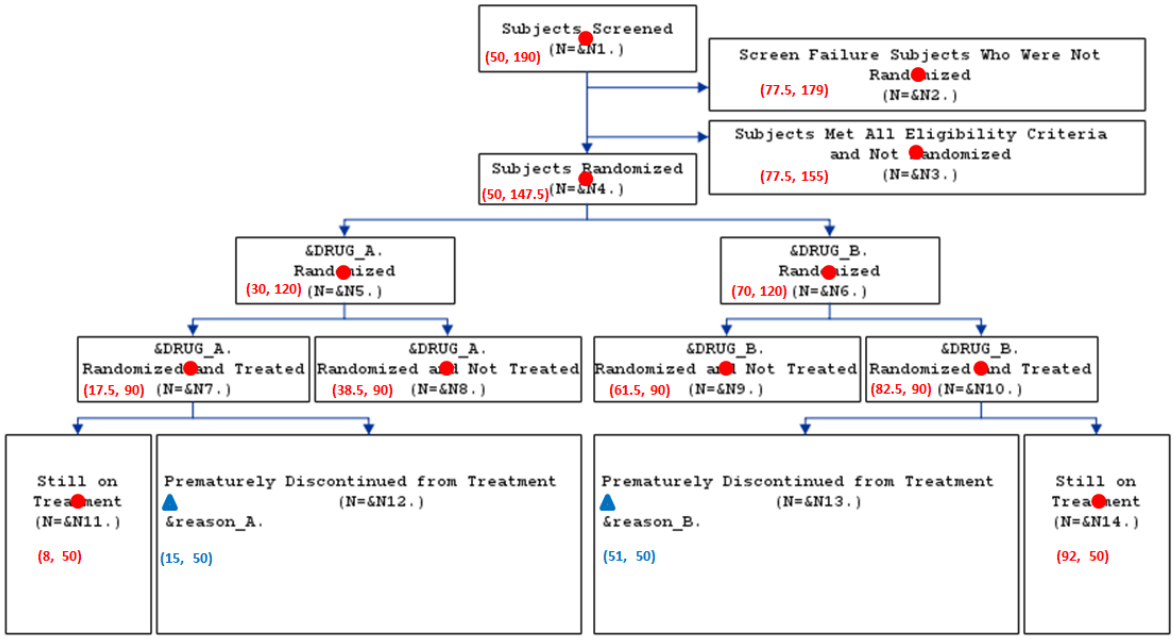

Figure 9. Full Texts' Coordinates (circles for CTEXT; triangles for LTEXT)

```
data CTEXT;
  length ctext $200;
  xt=50;  yt=190; ctext="Subjects Screened$(N=%cmpres(&N1.))"; output;
  xt=77.5;yt=179; ctext="Screen Failure Subjects Who Were
Not$Randomized$(N=%cmpres(&N2.))";output;
  xt=77.5;yt=155; ctext="Subjects Met All Eligibility Criteria$and Not
Randomized$(N=%cmpres(&N3.))";output;
  xt=50;  yt=147.5; ctext="Subjects Randomized$(N=%cmpres(&N4.))"; output;
  xt=30;  yt=120; ctext="&DRUG_A.$Randomized$(N=%cmpres(&N5.))"; output;
  xt=70;  yt=120; ctext="&DRUG_B.$Randomized$(N=%cmpres(&N6.))"; output;
  xt=17.5;yt=90;ctext="&DRUG_A.$Randomized and Treated$(N=%cmpres(&N7.))"; output;
  xt=38.5;yt=90;ctext="&DRUG_A.$Randomized and Not Treated$(N=%cmpres(&N8.))"; output;
  xt=61.5;yt=90;ctext="&DRUG_B.$Randomized and Not Treated$(N=%cmpres(&N9.))"; output;
  xt=82.5;yt=90;ctext="&DRUG_B.$Randomized and Treated$(N=%cmpres(&N10.))"; output;
  xt=8;   yt=50;ctext="Still on$Treatment$(N=%cmpres(&N11.))"; output;
  xt=92;  yt=50;ctext="Still on$Treatment$(N=%cmpres(&N14.))"; output;
run;
data LTEXT;
  length ltext $200;
  xt=15; yt=50; ltext="Prematurely Discontinued from Treatment$
(N=%cmpres(&N12.))$&reason_A."; output;
  xt=51; yt=50; ltext="Prematurely Discontinued from Treatment$
(N=%cmpres(&N13.))$&reason_B."; output;
run;
```

5. COMBINATION AND PROC SGPLOT

Now, we stack all the datasets: BOX, LINK, CTEXT and LTEXT into a FINAL dataset that will be used in PROC SGPLOT.

```
/*--Combine data--*/
   data FINAL;
     set BOX LINK CTEXT LTEXT;
   run;
```

After FINAL dataset is created, we run the SGPLOT procedure below. NOBORDER and NOAUTOLEGEND are options used to remove the border and to disable automatic legends.

Five core statements in PROC SGPLOT are listed here.

- POLYGON creates the rectangles, the coordinates (xb, yb) of which are specified in X= and Y= arguments. ID= argument identifies one rectangular box. In this example, boxid ranges from 1 to 14 indicating each of the14 boxes. Each boxid is supported by 4 sets of coordinates.
- SERIES creates the links among boxes, the coordinates (xl, yl) of which are specified in X= and Y= arguments. GROUP= argument is used to group the data as one line. Since arrow head is needed in the given example, LINEATTRS= argument specifies the appearance of the lines, which is set to the default of “graphdatadefault” in the example. Arguments of ARROWHEADPOS= and ARROWHEADSHAPE= are used to draw the arrow ends.
- TEXT creates the texts in the boxes, the coordinates (xt, yt) of which are specified in X= and Y= arguments. We need two TEXT statements in this example because box 12 and box 13 are left-aligned, while the others are centered. SPLITCHAR= and SPLITPOLICY= arguments split the text at the specified character and are not displayed (“$” is used in the example). TEXTATTRS= argument specifies visual attributes of the texts. For left-aligned text, one more optional argument is needed, POSITION=, which specifies the position of the text with respect to the location of the data point. For left-aligned text, we want to display text to the right of the data point, so we use “position=right”. Note: “position=center” is a defaulted option so it’s unnecessary for centered text.
- XAXIS specifies the range and display of the x-axis of the figure. In this paper, we have (MIN=0, MAX=100) as the horizontal space, with no axis displayed and no area to be reserved at the minimum or maximum end of the axis (OFFSETMIN=0 and OFFSETMAX=0).
- YAXIS specifies the range and display of the y-axis of the figure. In this paper, we have (MIN=0, MAX=200) as the vertical space, with no axis displayed and no area to be reserved at the minimum or maximum end of the axis.

```
ods graphics / reset width=9in height=5in noborder;

proc sgplot data=final noborder noautolegend;
  polygon id=boxid x=xb y=yb;
  series x=xl y=yl / group=linkid lineattrs=graphdatadefault
                     arrowheadpos=end arrowheadshape=filled;
  text x=xt y=yt text=ctext / splitchar='$' splitpolicy=splitalways
                              textattrs=(size=8 family="Cumberland AMT");
  text x=xt y=yt text=ltext / splitchar='$' splitpolicy=splitalways
                              textattrs=(size=8 family="Cumberland AMT") position=right;
  xaxis display=none min=0 max=100 offsetmin=0 offsetmax=0;
  yaxis display=none min=0 max=200 offsetmin=0 offsetmax=0;
run;
```

6. ADJUST THE ALIGNMENT (IF NEEDED)

After generating the figure, further adjustments may be required. Most elements of the figure are static but a few (such as the discontinuation reasons in boxes 12 and 13) are not. If the length of the text is too long, the contents of the text will appear outside of the box. For example, suppose the text for “Reason2” is very long. One way to fix the issue is to manually put a split character (“$”) in the middle of the text where a new line is needed. It may also be necessary to change the coordinates (xt, yt) in the LTEXT dataset in order to fit the contents in the box.

7. TITLE/FOOTNOTE AND ODS OUTPUTS

After finishing the main body part of PROC SGPLOT, we can add titles or footnote as requested and output to desired formats (e.g. RTF or PDF). Below are the examples codes for titles, footnotes, both PDF and RTF format outputs using options to ouput the figure in high resolution. The PDF (name as "AD_28.pdf") layout is as Figure 10.

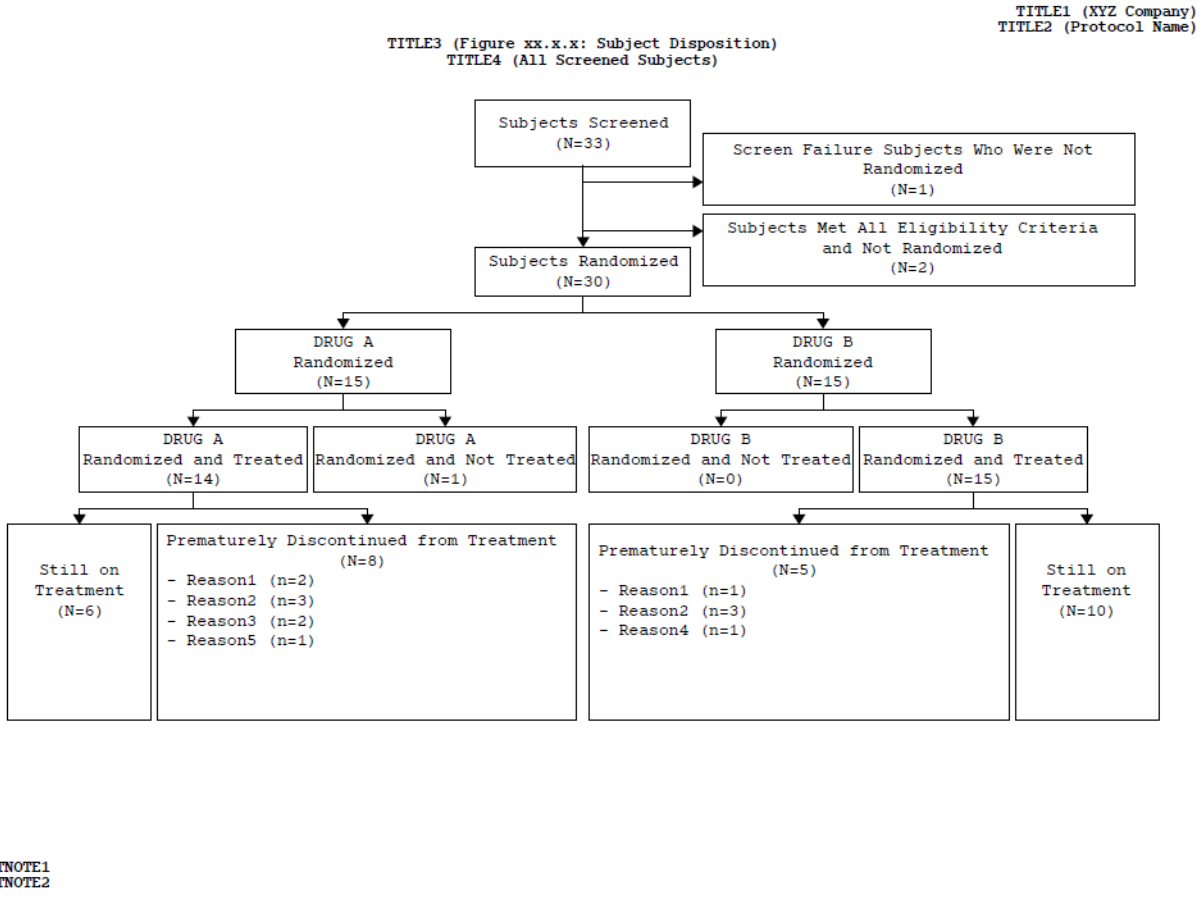

Figure 10. Final Display of Flow Diagram

```
Options nodate nonumber center nobyline PAPERSIZE=Letter ORIENTATION=LANDSCAPE
topmargin = "1 in" bottommargin = "1 in" leftmargin = "1 in" rightmargin = "1 in";

/*--Titles and Footnotes--*/
title1 j=r font="Courier New" Bold height=1.1 "TITLE1 (XYZ Company)";
title2 j=r font="Courier New" Bold height=1.1 "TITLE2 (Protocol Name)";
title3 j=c font="Courier New" Bold height=1.1 "TITLE3 (Figure xx.x.x: Subject Disposition)";
title4 j=c font="Courier New" Bold height=1.1 "TITLE4 (All Screened Subjects)";

footnote1 j=l font="Courier New" Bold height=1.1 "FOOTNOTE1";
footnote2 j=l font="Courier New" Bold height=1.1 "FOOTNOTE2";

ods graphics / reset width=9in height=5in noborder;
/*--PDF--*/
GOPTIONS device=emf;
ods pdf file="AD_28.pdf" nogtitle nogfootnote nobookmarkgen style=journal;
ods layout gridded;

/*--Set up a macro for SGPLOT procedure, proc sgplot is same as step 5--*/
%macro plot();
proc sgplot data=final noborder noautolegend;
  polygon id=boxid x=xb y=yb;
```

```
  series x=xl y=yl / group=linkid lineattrs=graphdatadefault
                     arrowheadpos=end arrowheadshape=filled;
  text x=xt y=yt text=ctext / splitchar='$' splitpolicy=splitalways
                              textattrs=(size=8 family="Cumberland AMT");
  text x=xt y=yt text=ltext / splitchar='$' splitpolicy=splitalways
                              textattrs=(size=8 family="Cumberland AMT") position=right;
  xaxis display=none min=0 max=100 offsetmin=0 offsetmax=0;
  yaxis display=none min=0 max=200 offsetmin=0 offsetmax=0;
run;
%mend;
%plot;

ods layout end;
ods pdf close;

ods graphics / reset width=9in height=5in outputfmt=png noborder;
/*--RTF--*/
ods rtf file="AD_28.rtf" nogtitle nogfootnote style=journal image_dpi=300;
%plot;
ods rtf close;
```

CONCLUSION

Drawing a flow diagram exclusively with SAS® has proven to be an unconventional exploration, as SGPLOT is not designed to create diagrams. To address the issue of transcription errors, the need for reordering, and potentially low resolution when using downloaded template or other software for flow diagrams, this paper presents a feasible way to draw a flow diagram showing the disposition of subjects in two-arm randomized trial using SAS® 9.4 M3. With DATA steps to set up boxes, links and texts to be displayed, combined with the SGPLOT procedure, it is possible for SAS® to draw such a figure. In application we can draw set of boxes/links/fixed texts using different layouts as templates to satisfy different study designs as needed.

However, there are still some questions and ideas from the author as follows.

- The work environment of SAS® is limited to 9.4 M3 or higher version for the approach in this paper. For lower version, is there any alternative approach to draw the same figure by SAS®?
- Setting up each individual set of coordinates (x,y) in the DATA steps is time-consuming and labor-intensive. Since boxes, links and texts are highly associated, there can be more efficient way to associate the coordinates of these items so that if one of the coordinates changed, the other items could automatically shift in a proportional manner.
- There is still some difference in format between the output made by Microsoft Visio® and SAS®. For instance, in box 12 and box 13 in Figure 1, list of “discontinuation reasons” was originally displayed as bulleted (“•”) list, which is now replaced by short dash (“-”) in Figure 10 because special Unicode characters could not be processed properly after the author attempted to do. So, is there any way to make it work?

REFERENCES

CONSORT Statement 2010. http://www.consort-statement.org/consort-statement/flow-diagram.

Carpenter, Art and Fisher, Dennis. 2012. “Reading and Writing RTF documents as Data: Automatic Completion of CONSORT Flow Diagrams.” *Proceedings of 2012 Pharmaceutical SAS Users Group (PharmaSUG) Conference*. http://www.pharmasug.org/proceedings/2012/TF/PharmaSUG-2012-TF16.pdf

Mallavarapu, Anusha and Shults, Dean. 2016. “CONSORT Diagram: Doing it With SAS.” *Proceedings of 2016 Pharmaceutical Users Software Exchange (PhUSE) conference.* https://www.lexjansen.com/phuse/2016/pp/PP03.pdf

Matange, Sanjay. 2016. “Outside-the-box: CONSORT diagram” *posted October 20, 2016.* https://blogs.sas.com/content/graphicallyspeaking/2016/10/20/outside-box-consort-diagram/

ACKNOWLEDGMENTS

I would like to thank all my colleagues at Gilead Sciences, Inc. who shared their experiences and gave me the constructive feedback. Special thanks for continuous encouragement and technical support from Will Garner, Jay Huang and my manager, Hui Liu.

CONTACT INFORMATION

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

Name: Huijuan (Tracy) He
Enterprise: Gilead Sciences, Inc
Address: 333 Lakeside Drive
City, State, ZIP: Foster City, CA, 94404
Work Phone: (650) 524-0897
Email: tracy.he@gilead.com
Web: http://www.gilead.com/