PharmaSUG 2017 - Paper DV03

Multipage Adverse Event Reports Using PROC SGPLOT

Warren F. Kuhfeld, SAS Institute Inc., Cary, NC
Mary Beth Herring, Rho Inc., Chapel Hill, NC

ABSTRACT

Researchers and safety monitoring committees need to track adverse events (AEs) among clinical trial participants. Adverse events are commonly reported as a tabular summary by system organ class and preferred term for each treatment group. But it can be difficult to identify trends or potential safety warnings simply by reviewing numbers in a summary table. Trends are much easier to identify when the frequency of each AE is displayed in a graph.

The SGPLOT procedure can use axis tables to display tabular information next to scatter plots, but multipage printed reports require additional programming. This paper shows how to use PROC SGPLOT to create elegant multipage adverse event reports. Most of the work involves preparing the data so that page breaks occur in reasonable places and so that groups of adverse events and continuations onto a new page are both properly labeled. PROC SGPLOT along with a BY statement produces the final report. Groups of adverse events can be separated by blank lines or by reference lines. This paper explains all the steps that are needed for you to prepare and display multipage adverse event reports that are easier to interpret than summary tables.

INTRODUCTION

Researchers are responsible for tracking adverse events (AEs) among participants in clinical trials. For each treatment group, AEs are reported by listing the frequency of each preferred term grouped by system organ class. The report contains columns of numbers, but you cannot always spot trends or potential safety warnings simply by reviewing numbers. It is much easier to identify trends when you display the frequency of each AE in a graph. Hence, the reports graphically display each row in the table. See Figure 5, Figure 6, and Figure 7 to see the reports that are generated by the code in this paper.

This work was inspired by the Rho Inc. AE reporting function AEPLOT (`https://github.com/RhoInc/aeplot`). This paper uses the SAS® 9.4M4 release. The code also works in SAS 9.4M3.

BACKGROUND EXAMPLES

There are a number of techniques that you need to understand before you can create an AE report by using the example code presented in later sections. This section explains some of those techniques in the context of simple artificial data.

AXIS TABLES

Axis tables consist of rows or columns of text or formatted numbers that are displayed along with graphs. All axis tables in this paper are Y-axis tables that consist of columns of text or numbers that appear to the left or right of a graph. The following step displays a graph that consists of multiple axis tables and graphs:

```
ods graphics on;
proc reg data=sashelp.class;
   model weight = height / r;
quit;
```

The results, shown in Figure 1, have three axis tables (observation number, studentized residual, and Cook's D) and two horizontal bar charts (studentized residual and Cook's D). Subsequent examples show how to create Y-axis tables by using PROC SGPLOT and the YAXISTABLE statement.

AXIS TABLES, DISCRETE AND LINEAR AXES, GROUP SEPARATORS, AND NONBREAKING SPACES

The following step creates a data set that has 43 observations, including four groups of size 10 and three missing values:

```
data x(keep=n line);
   do g = 1 to 4;
      do i = 1 to 10;
         k + 1;
         n = k;
         line = put(k, words30.);
         output;
      end;
      * Use unique sequence of nonbreaking spaces. This will not work: line = ' ';
      line = repeat('A0'x, g - 1);
      n = .;
      if g lt 4 then output;
   end;
run;
```

The numeric variable **n** contains the values 1 through 40 along with three missing values. The character variable **Line** contains the values `'one'` through `'forty'` along with three missing values. The missing values occur after **n**=10, **n**=20, and **n**=30. The goal is to create a graph that consists of a Y-axis table and a scatter plot and uses blank lines to separate groups of 10 rows. The following step creates the plot in Figure 2:

```
proc sgplot data=x noborder;
   yaxistable line / position=left;
   scatter y=line x=n;
   yaxis reverse display=none;
   xaxis display=none;
   label line='00'x;
run;
```

Figure 1 Y-Axis Tables and Bar Charts

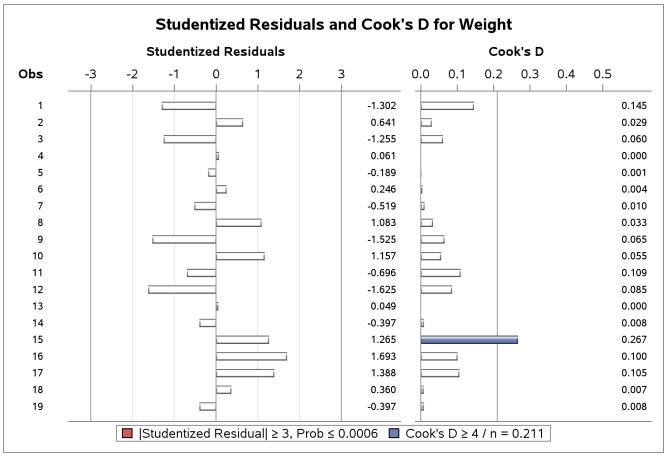
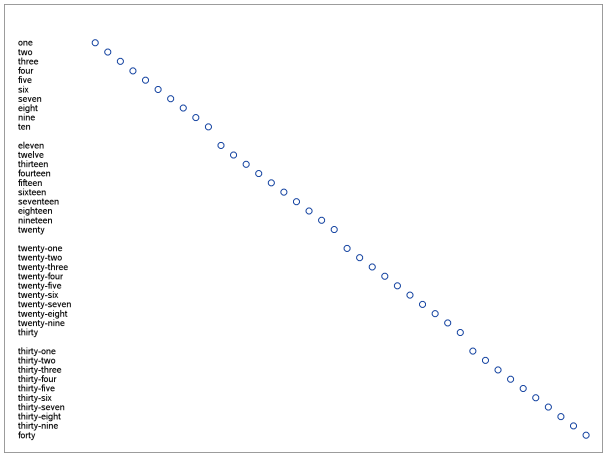


Figure 2 Group Separation



The YAXISTABLE statement is equivalent to the following statement:

```
yaxistable line / y=line position=left;
```

The implicit Y=**Line** option in the YAXISTABLE statement comes from the Y= option in the primary (first graph) statement, the SCATTER statement. When you use a character variable in the primary Y= option, the Y axis is discrete (TYPE=DISCRETE). When the Y axis is discrete and the AXISTABLE statement variable matches the primary Y variable, only unique values are displayed in the axis table. You can display blank lines by constructing unique sequences of nonbreaking spaces (NBSPs). NBSPs are specified using the hexadecimal constant notation in SAS. For example, '**A0**'**x** specifies a single NBSP and '**A0A0A0**'**x** specifies three NBSPs. The function REPEAT('A0'X, 0) creates one NBSP, the function REPEAT('A0'X, 1) creates two NBSPs, and so on. You can submit the following step

to see that the first blank line consists of one NBSP followed by 29 trailing blanks, the second blank line consists of two NBSPs followed by 28 trailing blanks, and so on:

```
proc print data=x(where=(n = .));
   format line $hex60.;
run;
```

An ordinary blank has the hexadecimal representation '**20**'**x**. PROC SGPLOT does not display blank lines in the axis table, nor does it display leading blanks; it does display the lines that consist of unique sequences of NBSPs. The following attempt to offset the formatted numbers fails because leading blanks are ignored:

```
line = repeat(' ', i) || put(k, words30.);
```

The following statement offsets the formatted numbers by using NBSPs:

```
line = repeat('A0'x, i) || put(k, words30.);
```

If the DATA step had created only one pattern of NBSPs, then only one blank line would have been displayed. All subsequent examples show how to make a linear Y axis (TYPE=LINEAR), which uses a scaled row number as the Y variable and does not require unique text.

ATTRIBUTE MAPS AND SECTION HEADERS

This example shows how to use attribute maps along with an augmented data set to display headers for each group of observations. The following steps create and display a data set that has three types of observations: header, indented rows, and blank.

```
data x(keep=n line value);
   do g = 1 to 4;
      Value = 'Head ';
      Line = catx(' ', 'Greater Than', 10  * (g - 1));
      output;
      value = 'IRow';
      do i = 1 to 10;
         k + 1;
         n = k;
         line = 'A0A0A0'x || put(k, words30.);
         output;
      end;
      value = 'Blank';
      line  = ' ';
      n = .;
      if g lt 4 then output;
   end;
run;

data x2;
   set x nobs=nobs;
   ObsID = 100 * (_n_ - 1) / (nobs - 1);
run;

proc print data=x2(obs=14);
   id obsid;
run;
```

The first 14 observations are displayed in Figure 3. Header values begin in column one. Values in the body of the table begin in column four after three NBSPs.

The following step creates an attribute map:

```
* Attribute map: bold section headers, normal table entries, blank lines;
data attrmap;
   id='aemap';    textcolor='Black';
   value='Head '; textweight='bold  '; output;
   value='IRow '; textweight='normal'; output;
   value='Blank';                      output;
run;
```

Attribute maps are SAS data sets that contain information about how certain aspects of graphs are displayed.[1] In this case, the attribute map controls how rows of text are displayed. Each graph has three types of observations: headers, the body of the table (indented rows), and blank lines that separate sections. Headers are bold, and body lines use a normal font. This attribute map is used in this and later examples.

The following step displays the data and uses the information in the attribute map to control the fonts:

```
proc sgplot data=x2 noborder dattrmap=attrmap;
   yaxistable line / position=left textgroup=value textgroupid=aemap;
   scatter y=obsid x=n / x2axis;
   yaxis reverse display=none;
   xaxis display=none;
   x2axis grid display=(noticks nolabel);
   label line='00'x;
run;
```

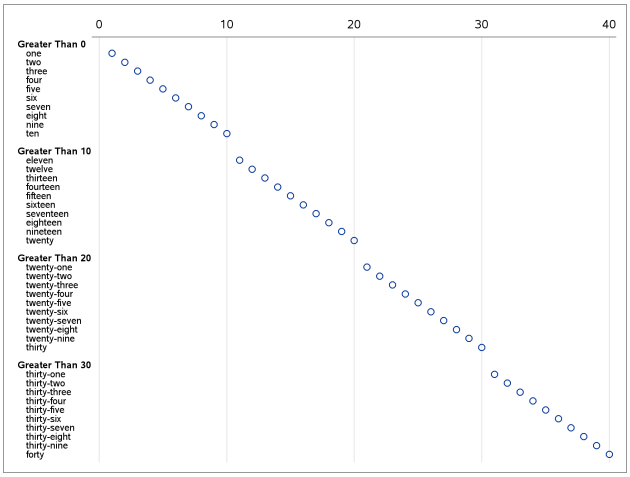
The results are displayed in Figure 4. There are four group headers, and each is displayed using a bold font. The remaining nonblank lines are displayed using a normal font. The X2AXIS statement provides an axis at the top of the scatter plot. The options that are used in this example are explained in detail in the AE example. The SCATTER statement is the primary statement, so the Y axis is based on the Y= variable from that statement. The Y= variable is **ObsID**, which is numeric and creates a linear Y axis. The variable **ObsID** is scaled so that its values range from 0 to 100, which precisely correspond to the tick values that PROC SGPLOT uses for this Y variable. In contrast, the row numbers (0 to 47) do not correspond to the default ticks (0 to 50), so the last line is not at the bottom of the graph if raw row numbers are used.

Most of the techniques that you need to make an AE report are illustrated in the preceding PROC SGPLOT step and Figure 4. Most of the work in making an AE report involves getting the data into the right form. In particular, you must be careful to properly create and label page breaks and continuations.

Figure 3 The First 14 Observations

ObsID	Value	Line	n
0.0000	Head	Greater Than 0	.
2.1739	IRow	one	1
4.3478	IRow	two	2
6.5217	IRow	three	3
8.6957	IRow	four	4
10.8696	IRow	five	5
13.0435	IRow	six	6
15.2174	IRow	seven	7
17.3913	IRow	eight	8
19.5652	IRow	nine	9
21.7391	IRow	ten	10
23.9130	Blank		.
26.0870	Head	Greater Than 10	.
28.2609	IRow	eleven	11

Figure 4 Section Headers and Indention



ADVERSE EVENT REPORTS

This section presents a series of multistep examples that show various ways to make adverse event reports.

CREATING AN ADVERSE EVENT DATA SET AND MACRO VARIABLES

The following steps read two data sets:

```
data adsl; /* Subject-level data used to get ns */
   input trtp $ @@;
   datalines;
```

[1] The name of the attribute map data set is specified in the DATTRMAP= option in PROC SGPLOT.

```
A A B B B A B B A B A A B B B B B A A B A B B A B A B B A A A A A B B A A A
A B A B A B A A A A A A B B B B B A B A B A A B A A A B A A B A B B A B B
;

data ae; /* Adverse events */
   length aebodsys aedecod $ 80 ap bp $ 12;
   input aebodsys $80. / aedecod $80. / acount bcount ap bp sort_bodysys;
   datalines;
Metabolic and nutritional
Peripheral edema
32 23 (82.1%) (63.9%) 1
Metabolic and nutritional
Hyperlipemia
18 17 (46.2%) (47.2%) 1
Metabolic and nutritional
Hyperkalemia

   ... more lines ...

Special senses
Retinal disorder
0 1 (0.0%) (2.8%) 13
;
```

The data set **ADSL** provides the treatment arms and the counts of subjects in each treatment arm. Duplicate AEs per ID are not counted.

The data set **AE** provides body systems, the preferred terms for each AE, and the number and percentage of subjects in treatment groups A and B who experience each AE. The preferred terms within each body system are sorted in descending order by the number of subjects who experience each AE.

The following steps create two macro variables:

```
proc freq data=adsl noprint; /* Get ns for both groups */
   tables trtp / out=trt;
run;

data _null_; /* Store ns in macro variables */
   set trt;
   call symputx(cats('n', trtp), cats('(n=', count, ')'));
run;
```

These macro variables contain the number of subjects in the two treatment groups. The macro variables are displayed in the headers of each page of the report.

The typical AE report has multiple pages. Each page contains a scatter plot, and each scatter plot is constructed so that the X axis ranges from 0 to a common maximum. This maximum is determined in the following step:

```
%let x2max = 0; /* Maximum count for X2 axis */
data _null_;
   set ae;
   call symputx('x2max', max(input(symget('x2max'), best16.), acount, bcount));
run;
```

This step sets the macro variable **X2Max** to the maximum of the variables **aCount** and **bCount** across all observations.

PREPARING ADVERSE EVENT DATA

The data-preparation code begins by specifying the number of lines per page:

```
%let maxperpage = 60; /* Maximum rows per page in a table */
```

This value is approximate. The actual number of rows on each page can vary. For example, the code presented in this paper does not begin a new body system in the last one or two lines on a page. Instead, it skips to a new page.

The next step begins the multistep process of adding headers, blank lines, page boundaries, and continuations across pages:

```
* Add section headers and spaces between sections. Section headers come from aebodsys.
  Its value is stored along with value='Head' in the RowLab variable.;
data plot1a;
   set ae;
   by sort_bodysys;
   length RowLab $ 80;
   if first.sort_bodysys then do;
      value  = 'Blank';
      call missing(rowlab, acount, bcount, ap, bp);
      if _n_ ne 1 then output;
      value  = 'Head';
      rowlab = aebodsys;         /* Section header                        */
      output;
      set ae point=_n_;          /* Retrieve data after setting up header */
   end;
   value  = 'IRow';
   rowlab = 'A0A0A0'x || aedecod;/* Indent by using 3 NBSPs               */
   output;
run;
```

The BY variable **Sort_BodySys** identifies body systems (groups of AEs). The header for each body system comes from the **AEBodSys** variable, and the statement ROWLAB = AEBODSYS creates the header for each body system. The statement SET AE POINT=_N_ restores all the data in the current observation, including those values that were changed to create header rows. Each AE comes from the **AEDECOD** variable. **RowLab** is first set to blank, then to the body system, and then to the AE. Simultaneously, the **Value** variable is set to `'Blank'`, `'Head'`, and then `'IRow'` (indented row). The attribute map uses the value of the **Value** variable to distinguish the types of observations. The value is also used in subsequent processing. Each AE is indented three spaces below the body system header by using NBSPs. Each value (except for the initial blank line) is written to a SAS data set.

The next step starts assigning rows to pages:

```
* A nondisplayed BY variable, BG, makes the different pages. The number
  of lines in each page is not constant to ensure nice header positions.;
data plot2a(drop=sort_bodysys aebodsys i page ningrp);
   set plot1a nobs=nobs;    by sort_bodysys;
   ningrp + 1; /* Number of lines in this BY group so far */

   * Start a new page when there are just a few lines left on the page;
   if ningrp ge &maxperpage - 3 and value eq 'Blank' then do; bg + 1; ningrp = 0; return; end;
   output;
   page = (ningrp ge &maxperpage and value eq 'IRow');

   if page then do; bg + 1; ningrp = 0; end;
   * If a group is continued, add a new header line(s) for the next page;
   if page and not last.sort_bodysys then do;
      call missing(acount, bcount, ap, bp);
      value  = 'Head';  rowlab = catx(' ', aebodsys, '(cont.)');  ningrp + 1;
      output;
   end;

   if _n_ eq nobs and ningrp then do; /* Add blank lines to end of last page */
      value = 'Blank';
      call missing(rowlab, acount, bcount, ap, bp);
      do i = ningrp to &maxperpage;    output;    end;
   end;
run;
```

The **BG** variable contains the BY group or page number. When a body system continues to a new page, the code adds a continuation header line that is designated by `'(cont.)'`. If the last page is shorter than the page size, extra blank lines are added to the end.

The next step finds the maximum header length and stores it in the macro variable **MaxLen**:

```
%let maxlen = 0; /* Maximum length of headers */
data _null_;
   set plot2a;
   if value eq 'Head' then
      call symputx('maxlen', max(input(symget('maxlen'), best16.), length(rowlab)));
run;
```

The next steps finish processing the data:

```
* Don't begin the page with a blank line;
data plot3a(drop=len);
   set plot2a nobs=nobs;   by bg;
   if value eq 'Head' then do;   /* Pad headers with NBSPs                */
      len = length(rowlab);      /* Less-than-perfect way to get a more   */
      if &maxlen - len gt 0 then /* uniform axis table width across pages.*/
         rowlab = substr(rowlab, 1, len) || repeat('A0'x, 2 * (&maxlen - len));
   end;
   if not ((first.bg or last.bg) and value eq 'Blank') then output;
run;

data plot4a(drop=first ningrp);
   set plot3a(keep=bg);    by bg;
   retain first ningrp 0;
   ningrp = ifn(first.bg, 0, ningrp + 1);
   if first.bg then first = _n_;
   if last.bg then do i = first to _n_;
      set plot3a point=i;
      obsid = 100 * (i - first) / ningrp; /* Scale row number from 0 to 100 */
      output;
   end;
run;
```

Headers are padded by adding NBSPs. This makes their length *approximately* uniform so that the graphs are approximately the same width across pages. This heuristic is not precise because the exact length of text rendered using proportional fonts cannot in general be ascertained.[2] This step also ensures that pages do not begin with a blank line. It ensures that continuation headers do not appear at the top of the next page when a body system ends on the previous page. Furthermore, a blank line at the end of a page is dropped. In the last step, **ObsID** is set to the row number (scaled from 0 to 100 so that the maximum **ObsID** corresponds to the maximum tick mark, to ensure that all available space is used).

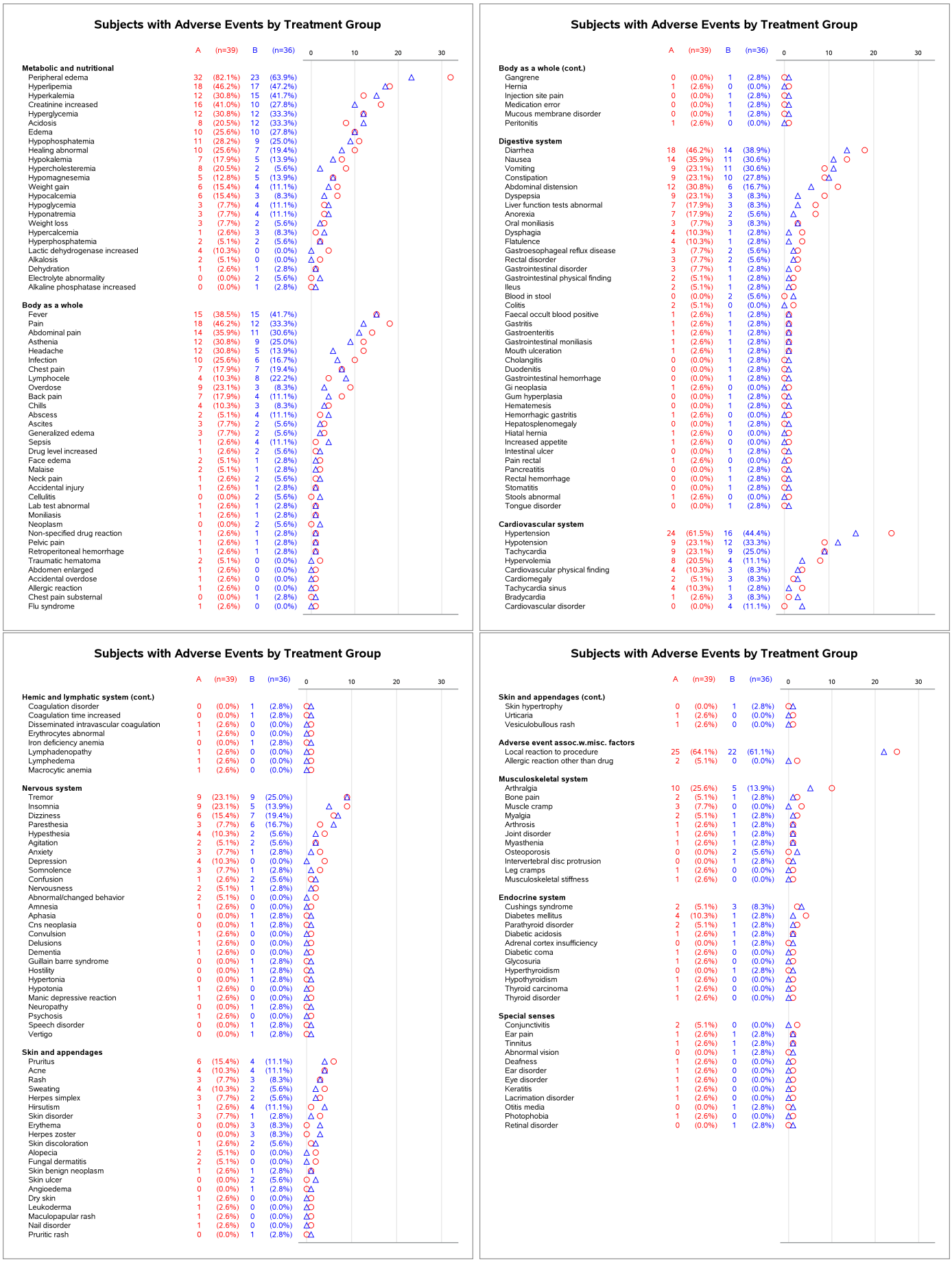
CREATING AN ADVERSE EVENT REPORT WITH BLANK LINES BETWEEN BODY SYSTEMS

PROC SGPLOT along with a BY statement and the attribute map from section "ATTRIBUTE MAPS AND SECTION HEADERS" on page 3 generates the multipage report. This step creates Figure 5, which shows four of the six pages of the report:

```
title 'Subjects with Adverse Events by Treatment Group';
* No BY lines. BY group is only for paging. Display missings as blanks.;
options nodate nonumber nobyline missing=' ';
ods listing close;
ods rtf file='ae.rtf' style=pearl image_dpi=300;
ods graphics on / height=10in width=7.5in border=off;
proc sgplot data=plot4a noautolegend noborder dattrmap=attrmap tmplout='temp.temp';
   by bg;
   yaxistable rowlab    / position=left textgroup=value textgroupid=aemap
                          pad=(right=.2in);
   yaxistable acount ap / position=left labelattrs=(size=10px) valuejustify=right
                          valueattrs=(color=red) labelattrs=(color=red);
   yaxistable bcount bp / position=left labelattrs=(size=10px) valuejustify=right
                          valueattrs=(color=blue) labelattrs=(color=blue);
   scatter x=acount y=obsid / x2axis markerattrs=(symbol=circle   color=red  size=10);
```

[2] For uniform column widths, you must use the Graph Template Language. For more information, see the section "MAKING COLUMN WIDTHS UNIFORM" on page 10.

Figure 5 Adverse Event Report (First Two and Last Two Pages)



```
   scatter x=bcount y=obsid / x2axis markerattrs=(symbol=triangle color=blue size=10);
   * Get bottom axis line;
   scatter x=acount y=obsid / markerattrs=(size=0);
   x2axis min=0 max=&x2max grid display=(noticks nolabel) valueattrs=(size=9px);
   xaxis display=(noticks nolabel novalues);
   yaxis display=none reverse;
   label rowlab = '00'x acount = "A" ap="&na" bcount = "B" bp="&nb";
run;
ods rtf close;
ods listing;
options byline missing='.';
```

Each page displays five axis tables and two scatter plots. A third scatter plot is invisible; it adds an axis line to the bottom of each page. The X2AXIS statement displays the axis at the top of each page along with the column headers. The system option NOBYLINE suppresses the BY line; the BY variable is used to make separate pages, but it is not displayed. The Pearl style is used; it is similar to the HTMLBlue style, which is the default for the HTML destination. The Pearl style has a white page background and is well suited to the RTF, PDF, and PRINTER destinations. The PROC SGPLOT statement option TMPLOUT='*temp.temp*' writes the template to a file named *temp.temp*. This file is used in the section "MAKING COLUMN WIDTHS UNIFORM" on page 10.

The first YAXISTABLE statement displays the AEs. The TEXTGROUP= option names the **Value** variable as the variable whose values match the values of the **Value** variable in the attribute maps data set. The name **Value** is required in the attribute map. The TEXTGROUPID= option requests that PROC SGPLOT use the **ID**='`aemap`' observations in the attribute map. Attribute maps can have many IDs, each identifying observations that affect different parts of the graph. The second two YAXISTABLE statements display the counts and percentages. Two statements are used so that they can have different value and label attributes. The two SCATTER statements display the two count variables by using colors that match the colors of the counts in the axis tables. The first two SCATTER statements use the X2 axis, which is at the top of the plot. The final SCATTER statement uses the X axis; it sets the marker size to 0 to make an invisible plot. This creates the X axis at the bottom of each page. By default, values along the Y axis start at the bottom and continue up. The REVERSE option displays values that start at the top and continue down. The LABEL statement provides the column headers for the axis tables. The first axis table column has no header, because the **RowLab** variable has a null label ('**00**'**x**).

CREATING AN ADVERSE EVENT REPORT WITH REFERENCE LINES BETWEEN BODY SYSTEMS

The next steps create the same report but use reference lines to separate the groups:

```
data plot5a(drop=nextval ningrp first);
   set plot3a nobs=nobs;    by bg;
   retain ningrp first 0;
   ningrp = ifn(first.bg, 0, ningrp + 1);
   if first.bg then first = _n_;
   if last.bg then do i = first to _n_;
      set plot3a point=i;
      obsid = 100 * (i - first) / ningrp; /* Scale row number from 0 to 100 */
      k = i + 1;   nextval = 'Blank';
      if k le _n_ then set plot3a(keep=value rename=(value=nextval)) point=k;
      ref = ifn(value eq 'Blank' and nextval ne 'Blank', obsid, .);
      output;
   end;
run;

options nobyline missing=' ';
ods listing close;
ods rtf file='ae2.rtf' style=pearl image_dpi=300;
ods graphics on / height=10in width=7.5in;
proc sgplot data=plot5a noautolegend noborder dattrmap=attrmap;
   by bg;
   refline ref;
   yaxistable rowlab    / position=left textgroup=value textgroupid=aemap
                          pad=(right=.2in) location=inside;
   yaxistable acount ap / position=left labelattrs=(size=10px) location=inside
```

```
                          valueattrs=(color=red) labelattrs=(color=red) valuejustify=right;
   yaxistable bcount bp / position=left labelattrs=(size=10px) location=inside
                          valueattrs=(color=blue) labelattrs=(color=blue) valuejustify=right;
   scatter x=acount y=obsid / x2axis markerattrs=(symbol=circle   color=red  size=10);
   scatter x=bcount y=obsid / x2axis markerattrs=(symbol=triangle color=blue size=10);
   * Get bottom axis line;
   scatter x=acount y=obsid / markerattrs=(size=0);
   x2axis min=0 max=&x2max grid display=(noticks nolabel) valueattrs=(size=9px);
   xaxis display=(noticks nolabel novalues);
   yaxis display=none reverse colorbands=odd colorbandsattrs=(transparency=1);
   label rowlab = '00'x acount = "A" ap="&na" bcount = "B" bp="&nb";
run;
ods rtf close;
ods listing;
options byline missing='.';
```

Ref is set to **ObsID** in the rows where reference lines are to appear. The results are displayed in Figure 6.

There are several differences in the PROC SGPLOT code. Now, the LOCATION=INSIDE option places each axis table inside the graph. The COLORBANDS=ODD and COLORBANDSATTRS=(TRANSPARENCY=1) options in the YAXIS statement enable the reference lines to extend into the axis tables.

MAKING COLUMN WIDTHS UNIFORM

The widths of the columns in Figure 5 and Figure 6 vary from page to page. You can precisely control the widths and create uniform column widths throughout your report by using the Graph Template Language (GTL). PROC SGPLOT writes and uses a graph template for every graph that it creates. You can write that template to a file, modify it, and save it. This is often easier than writing graph templates from scratch. The PROC SGPLOT statement option TMPLOUT='*temp.temp*' in the section "CREATING AN ADVERSE EVENT REPORT WITH BLANK LINES BETWEEN BODY SYSTEMS" on page 7 writes the template that this example uses.

You can use an editor to change the template name from `'sgplot'` to `'aeplot'`, change the column widths, and incorporate the macro variable **X2Max** (the maximum for the X2 axis). Or you can use a DATA step to make the same edits:

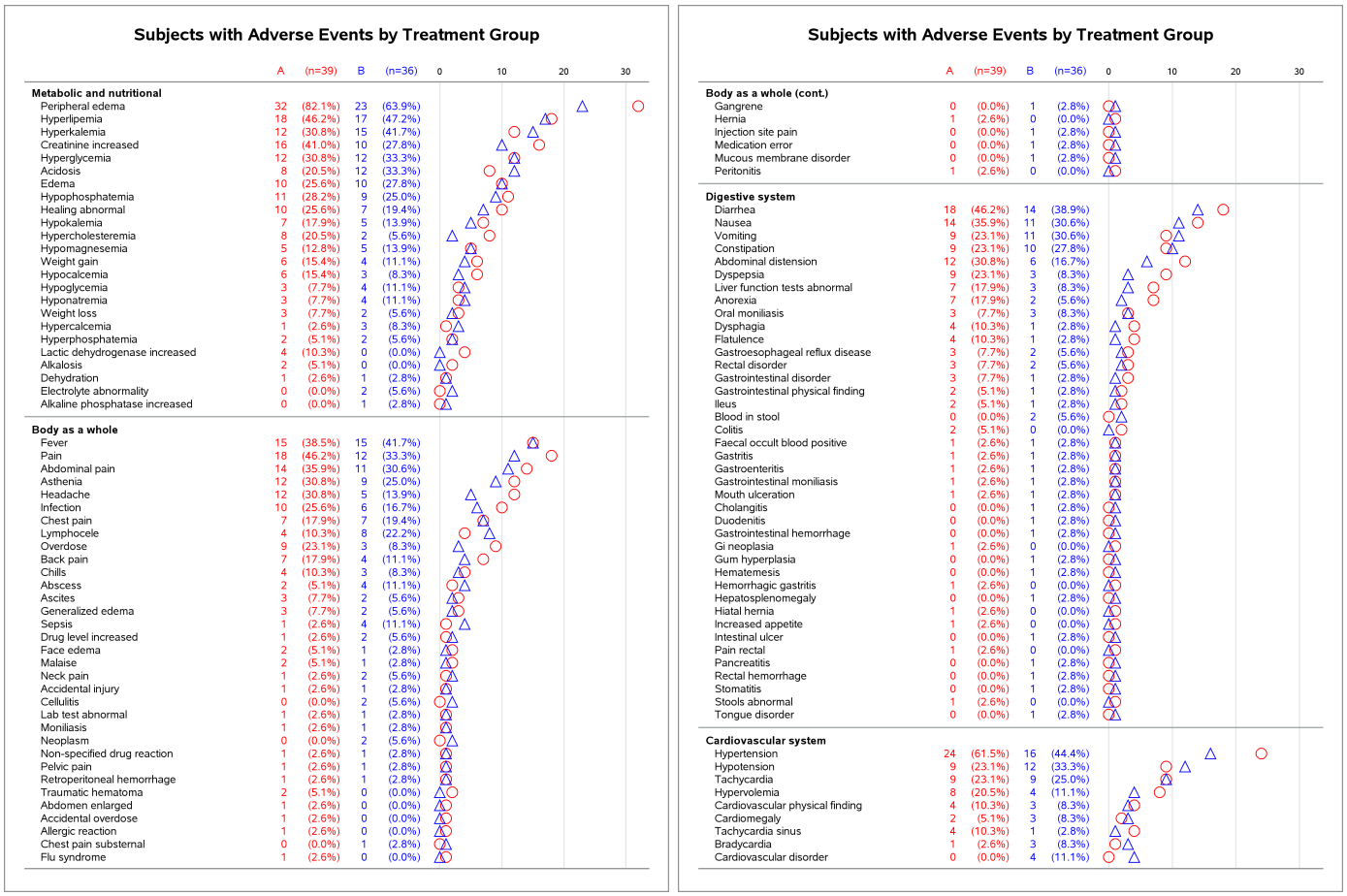
```
data _null_;
   infile 'temp.temp';
   input;
   * Change the template name. This is not required.;
   _infile_ = tranwrd(_infile_, 'statgraph sgplot', 'statgraph aeplot');
   * Change the column weights (proportion of vertical width for each column);
   _infile_ = tranwrd(_infile_, 'columnweights=preferred',
                      'columnweights=(0.42 0.04 0.1 0.04 0.1 0.3)');
   * Add an NMVAR statement that names the X2 axis maximum;
   if _infile_ =: 'dynamic' then call execute('nmvar x2max;');
   * Specify the maximum;
   i = index(_infile_, 'viewmax=');
   if i then _infile_ = tranwrd(_infile_, substr(_infile_, i,
                                find(_infile_, ')', i) - i), 'viewmax=x2max');
   * Submit the modified template;
   call execute(_infile_);
run;
```

The DATA step enables you to make the edits in a reproducible way. The CALL EXECUTE statements execute each of the original, modified, and new statements to create the AEPlot template.

By default, PROC SGPLOT uses the option COLUMNWEIGHTS=PREFERRED, which sets the widths based on the column contents. You can change the widths to a series of proportions by specifying a vector of weights: COLUMNWEIGHTS=(0.42 0.04 0.1 0.04 0.1 0.3).

The following step creates the report by using the modified template and column widths and the SGRENDER procedure:

Figure 6 Adverse Event Report Using Reference Lines (First Two Pages)




```
title;
options nobyline missing=' ';
ods listing close;
ods rtf file='ae3.rtf' style=pearl image_dpi=300;
ods graphics on / height=10in width=7.5in border=off;
proc sgrender data=plot4a template=aeplot;
   by bg;
   label rowlab = '00'x acount = "A" ap="&na" bcount = "B" bp="&nb";
run;
ods rtf close;
ods listing;
options byline missing='.';
```

The results of this step are not displayed. This approach enables you to skip the section of the data-preparation program that pads the headers by adding NBSPs.

CREATING A GENERAL-PURPOSE AE REPORT INCLUDING SPLIT-CHARACTER SUPPORT

The AE report example in the section "CREATING AN ADVERSE EVENT REPORT WITH BLANK LINES BETWEEN BODY SYSTEMS" on page 7 uses PROC SGPLOT, because PROC SGPLOT code is easier to write than a graph template. This example uses the GTL, but PROC SGPLOT wrote the first version of the template. Because the graph template has explicitly specified column weights, column widths are identical from page to page. This example also supports split characters. Some row labels are long and need to be split to avoid truncation. This example programmatically adds a split character (a tilde, `'~'`) between words when a new word starts after the 12th character. This makes the lines much shorter than they need to be, but it enables you to see a variety of splits. (For real examples, set the variable **s** to a value larger than 12, and adjust the column weights so that the first axis table is wider and the graph is narrower.) The data-preparation code must avoid splitting pages in the middle of an AE and must correctly continue split headers.

The following step inserts split characters into the **AEBodSys** and **AEDECOD** variables in the **AE** data set:

```
%let maxperpage = 60; /* Maximum rows per page in a table */

data aesplit(drop=i s); /* Adverse events */
   set ae;
   s = 12; /* Split once between words after column 12 */
   i = find(trim(aebodsys), ' ', s+1);
   if i then substr(aebodsys, i, 1) = '~';
   i = find(trim(aedecod), ' ', s+1);
   if i then substr(aedecod, i, 1) = '~';
run;
```

This and subsequent steps rely on the data and preliminary processing found in the section "CREATING AN ADVERSE EVENT DATA SET AND MACRO VARIABLES" on page 4. This example does not use the attribute map data set; that information is provided in the graph template. However, it does use the **Value** variable to distinguish the various types of rows in the table:

Value='Head'	a one-line header
Value='Head1'	the first line of a two-line header
Value='Head2'	the second line of a two-line header
Value='IRow'	a one-line AE
Value='IRow1'	the first line of a two-line AE
Value='IRow2'	the second line of a two-line AE
Value='Blank'	a blank line

A later step in the data preparation removes the numerals from the **Value** variable so that its values correspond to those in the attribute map. The next step begins the multistep process of adding headers, blank lines, page boundaries, and continuations across pages:

```
* Add section headers and spaces between sections;
data plot1b(drop=aedecod);
   set aesplit;
   by sort_bodysys;
   length RowLab $ 80;
   if first.sort_bodysys then do;
      value  = 'Blank';
      call missing(rowlab, acount, bcount, ap, bp);
      if _n_ ne 1 then output;
      value  = 'Head';
      rowlab = aebodsys;          /* Section header                        */
      output;
      set ae point=_n_;           /* Retrieve data after setting up header */
   end;
   value  = 'IRow';
   rowlab = 'A0A0A0'x || aedecod;/* Indent using 3 NBSPs                   */
   output;
run;
```

This step matches the **Plot1a** DATA step in the previous example. The next step splits lines that have split characters:

```
data plot1bsp(drop=t1 t2); /* Handle split characters */
   set plot1b;
   by sort_bodysys;
   length head1 head2 $ 80;
   retain head1 head2 ' ';
   if first.sort_bodysys then do; head1 = ' '; head2 = ' '; end;
   i = index(rowlab, '~');
   if i then do;
      substr(value, 5, 1) = '1';
      t1 = substr(rowlab, 1, i - 1);
      t2 = substr(rowlab, i + 1);
      rowlab = t1;
```

```
      if value =: 'Head' then do; head1 = t1; head2 = t2; end;
      output;
      substr(value, 5, 1) = '2';
      rowlab = 'A0A0'x || t2;
      if value =: 'IRow' then do;
         call missing(acount, bcount, ap, bp);
         rowlab = 'A0A0A0A0A0'x || rowlab;
      end;
   end;
   output;
run;
```

Split headers are temporarily stored in the **Head1** and **Head2** variables for use in other steps. The second part of a split line is indented by using NBSPs. The next step assigns lines to pages:

```
* Use a nondisplayed BY variable BG to make the different pages. The number
  of lines in each page is not constant to ensure nice header positions.;
data plot2b(drop=sort_bodysys head1 head2 aebodsys ningrp i page);
   set plot1bsp nobs=nobs;    by sort_bodysys;
   ningrp + 1; /* Number of lines in this BY group so far */

   * Start a new page when there are only a few lines left on the page;
   if ningrp ge &maxperpage - 5 and value eq 'Blank' then do; bg + 1; ningrp = 0; return; end;
   page = (ningrp ge &maxperpage - 1 and value in ('IRow', 'IRow2'));
   output;

   if page then do; bg + 1; ningrp = 0; end;
   * If a group is continued, add a new header line(s) for the next page;
   if page and not last.sort_bodysys then do;
      call missing(acount, bcount, ap, bp);
      ningrp + 1;
      if head1 eq ' ' then do;
         value  = 'Head';  rowlab = catx(' ', aebodsys, '(cont.)');
         output;
      end;
      else do;
         value  = 'Head1'; rowlab = head1;   ningrp + 1;                    output;
         value  = 'Head2'; rowlab = 'A0A0'x || catx(' ', head2, '(cont.)'); output;
      end;
   end;

   if _n_ eq nobs and ningrp then do; /* Add blank lines to end of last page */
      value = 'Blank';
      call missing(rowlab, acount, bcount, ap, bp);
      do i = ningrp to &maxperpage;    output;    end;
   end;
run;
```

This step is more complicated than in the earlier example because of the split lines. Care is taken not to split an AE across pages or start a page with a blank line. The previously created **Head1** and **Head2** variables are used to construct continuations of split headers. The next steps finish processing the data:

```
* Don't begin the page with a blank line;
data plot3b;
   set plot2b; by bg;
   if not ((first.bg or last.bg) and value eq 'Blank') then output;
run;

data plot4b(drop=first ningrp);
   set plot3b(keep=bg);    by bg;
   retain first ningrp 0;
   ningrp = ifn(first.bg, 0, ningrp + 1);
   if first.bg then first = _n_;
   if last.bg then do i = first to _n_;
```

```
      set plot3b point=i;
      value = compress(value, '12');      /* No longer need numeral split flags */
      obsid = 100 * (i - first) / ningrp; /* Scale row number from 0 to 100     */
      output;
   end;
run;
```

The **Plot4b** DATA step adds an observation ID variable and restores the **Value** variable to its original form.

The following step provides the graph template:

```
proc template;
   define statgraph aeplot;
      begingraph / collation=binary;
         nmvar x2max;
         DiscreteAttrVar attrvar=AEMAP_VALUE var=VALUE attrmap="__ATTRMAP__AEMAP";
         DiscreteAttrMap name="__ATTRMAP__AEMAP";
            Value "Head"  / textattrs=(color=CX000000 weight=bold);
            Value "IRow"  / textattrs=(color=CX000000 weight=normal);
            Value "Blank" / textattrs=(color=CX000000 weight=normal);
         EndDiscreteAttrMap;
         EntryTitle "Subjects with Adverse Events by Treatment Group";
         layout lattice / columnweights=(0.26 0.05 0.11 0.05 0.11 0.42)
                          columndatarange=union rowdatarange=union columns=6;
            %let opts = display=none type=linear;
            %let opts = yaxisopts=(reverse=true &opts) walldisplay=none y2axisopts=(&opts);
            Layout Overlay / &opts;
               AxisTable Value=rowlab Y=ObsID / labelPosition=min pad=(right=0.2in)
                                                textGroup=AEMAP_VALUE Display=(Label);
            endlayout;
            %macro col(v,c);
               Layout Overlay / &opts;
                  AxisTable Value=&v Y=ObsID / Display=(Label) valuejustify=right
                     labelPosition=min LabelAttrs=(Color=&c Size=9px) ValueAttrs=(Color=&c);
               endlayout;
            %mend;
            %col(acount, red)
            %col(ap    , red)
            %col(bcount, blue)
            %col(bp    , blue)
            layout overlay / walldisplay=(fill) xaxisopts=(display=(line))
                             yaxisopts=(reverse=true display=none type=linear)
                             x2axisopts=(display=(tickvalues line) TickValueAttrs=(Size=9px)
                             linearopts=(viewmin=0 viewmax=x2max) griddisplay=on);
               ScatterPlot X=acount Y=ObsID / xaxis=x2
                           Markerattrs=(Color=red  Symbol=circle   Size=10);
               ScatterPlot X=bcount Y=ObsID / xaxis=x2
                           Markerattrs=(Color=blue Symbol=triangle Size=10);
               ScatterPlot X=acount Y=ObsID / Markerattrs=(Size=0);
            endlayout;
         endlayout;
      endgraph;
   end;
run;
```

This template is an edited and simplified version of the graph template that was written by the PROC SGPLOT TMPLOUT= option in the section “CREATING AN ADVERSE EVENT REPORT WITH BLANK LINES BETWEEN BODY SYSTEMS” on page 7. The lattice contains six overlays—one for each of the five axis tables and one for the scatter plots. Notice that you specify row numbers in the Y= option in the AXISTABLE statement and row labels in the VALUE= option. Column weights are specified that work well for the short row labels in this example.

The following step creates the report in Figure 7:

```
options nobyline missing=' ';
ods listing close;
ods rtf file='ae4.rtf' style=pearl image_dpi=300;
ods graphics on / height=10in width=7.5in border=off;
proc sgrender data=plot4b template=aeplot;
   by bg;
   label rowlab = '00'x acount = "A" ap="&na" bcount = "B" bp="&nb";
run;
ods rtf close;
ods listing;
options byline missing='.';
```

The results show split headers (such as “Cardiovascular system”), a continuation of a split header, and numerous split AEs, none of which is split across a page. Column widths are uniform throughout.

If you need to make AE reports, you could easily create a macro that does all the work. You could set the maximum lines per page, column weights, and so on, as macro parameters.

In the interest of conserving space, this example is not expanded to show reference lines. To see how to do it, add the TMPLOUT= option to the PROC SGPLOT in the section “CREATING AN ADVERSE EVENT REPORT WITH REFERENCE LINES BETWEEN BODY SYSTEMS” on page 9 and modify the generated template. That template specifies the AXISTABLE statements in INNERMARGIN blocks rather than LAYOUT OVERLAY blocks.

CREATING THE SAME GRAPHS USING PROC SGPLOT AND TEMPLATE MODIFICATION

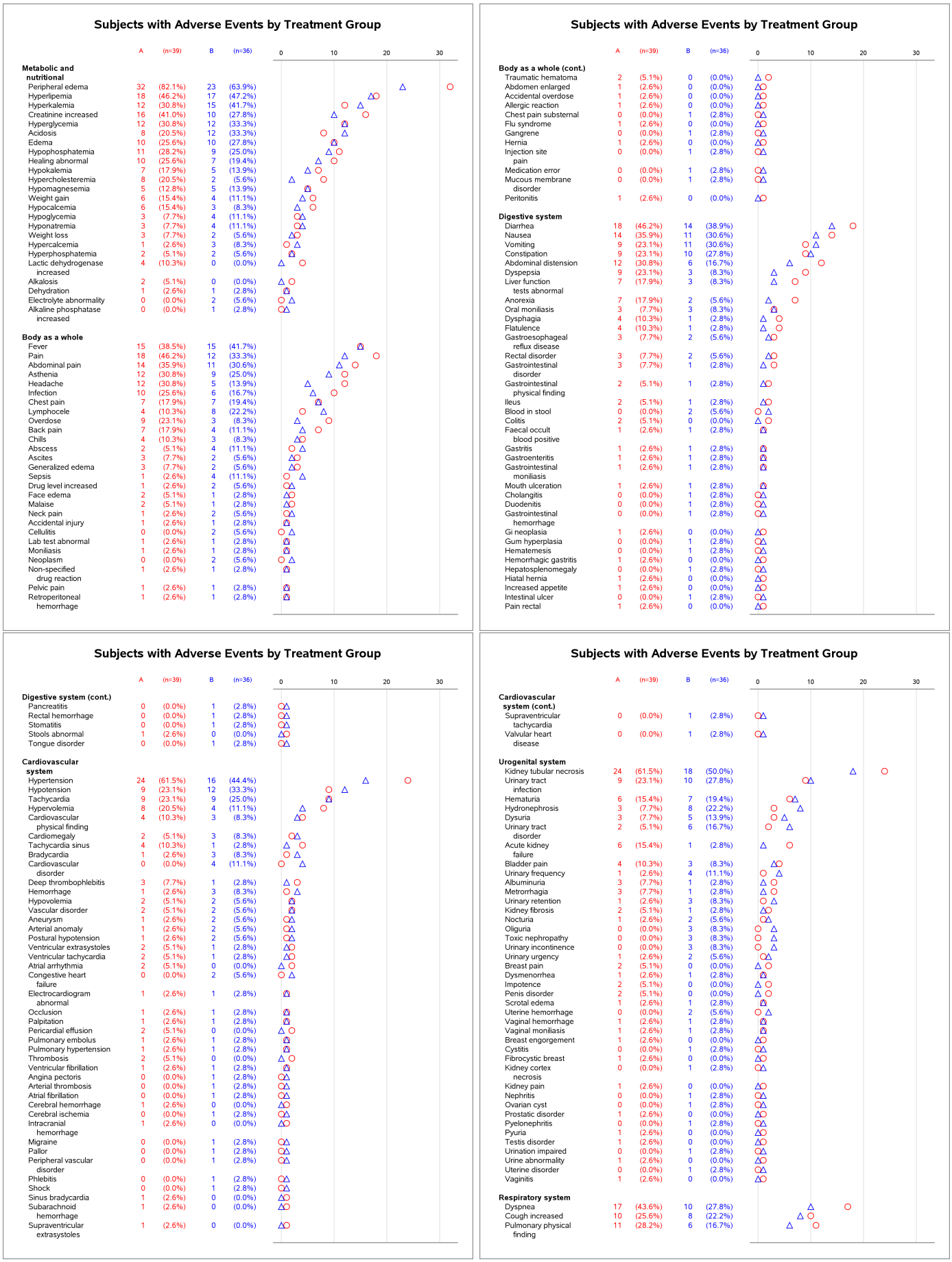
You can create the same graphs that are found in Figure 7 by using PROC SGPLOT, the attribute map from section “ATTRIBUTE MAPS AND SECTION HEADERS” on page 3, and template modifications:

```
title 'Subjects with Adverse Events by Treatment Group';
proc sgplot data=plot4b noautolegend noborder dattrmap=attrmap tmplout='temp2.temp';
   by bg;
   yaxistable rowlab    / position=left textgroup=value textgroupid=aemap;
   yaxistable acount ap / position=left labelattrs=(size=10px) valuejustify=right
                          valueattrs=(color=red) labelattrs=(color=red);
   yaxistable bcount bp / position=left labelattrs=(size=10px) valuejustify=right
                          valueattrs=(color=blue) labelattrs=(color=blue);
   scatter x=acount y=obsid / x2axis markerattrs=(symbol=circle   color=red  size=10);
   scatter x=bcount y=obsid / x2axis markerattrs=(symbol=triangle color=blue size=10);
   scatter x=acount y=obsid / markerattrs=(size=0);
   x2axis min=0 max=&x2max grid display=(noticks nolabel) valueattrs=(size=9px);
   xaxis display=(noticks nolabel novalues);
   yaxis display=none reverse;
   label rowlab = '00'x acount = "A" ap="&na" bcount = "B" bp="&nb";
run;

data _null_;
   infile 'temp2.temp';
   input;
   _infile_ = tranwrd(_infile_, 'statgraph sgplot', 'statgraph aeplot');
   _infile_ = tranwrd(_infile_, 'columnweights=preferred',
                      'columnweights=(0.26 0.05 0.11 0.05 0.11 0.42)');
   if _infile_ =: 'dynamic' then call execute('nmvar x2max;');
   i = index(_infile_, 'viewmax=');
   if i then _infile_ = tranwrd(_infile_, substr(_infile_, i,
                                find(_infile_, ')', i) - i), 'viewmax=x2max');
  call execute(_infile_);
run;

title;
options nobyline missing=' ';
ods listing close;
ods rtf file='ae5.rtf' style=pearl image_dpi=300;
ods graphics on / height=10in width=7.5in border=off;
```

Figure 7 Adverse Event Report with Splitting (First Four Pages)



```
proc sgrender data=plot4b template=aeplot;
   by bg;
   label rowlab = '00'x acount = "A" ap="&na" bcount = "B" bp="&nb";
run;
ods rtf close;
ods listing;
options byline missing='.';
```

The results of these steps (not shown) match the results displayed in Figure 7.

CONCLUSION

You are probably familiar with using PROC SGPLOT to create a single graph. By combining careful data preparation, axis tables, and BY-group processing, you can create multipage reports that combine tabular and graphical information. These techniques enable you to create multipage reports of adverse events and other multipage clinical reports. You can then use template modification techniques to enhance the final graphs.

ACKNOWLEDGMENTS

The authors are grateful to their editor, Ed Huddleston, for his helpful comments, to Sanjay Matange and Dan Heath for their help using the axis table, and to Karen Boyle for her valuable feedback.

RECOMMENDED READING

For complete documentation about ODS and ODS Graphics, see *SAS Output Delivery System: User's Guide*, *SAS Graph Template Language: Reference*, *SAS Viya ODS Graphics: Procedures Guide*, and *SAS Graph Template Language: User's Guide*. For more examples and more documentation, see Chapter 20, "Using the Output Delivery System"; Chapter 21, "Statistical Graphics Using ODS"; Chapter 22, "ODS Graphics Template Modification"; and Chapter 23, "Customizing the Kaplan-Meier Survival Plot" in the *SAS/STAT User's Guide*.

For a gentle and parallel introduction to PROC SGPLOT and the GTL, see the free web book *Basic ODS Graphics Examples* (http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsbasicg.pdf).

For more advanced topics, including an introduction to SG annotation, axis tables, and template modification using the CALL EXECUTE statement, see the free web book *Advanced ODS Graphics Examples* (http://support.sas.com/documentation/prod-p/grstat/9.4/en/PDF/odsadvg.pdf).

For tips, tricks, and the latest developments in ODS Graphics, see Sanjay Matange's blog *Graphically Speaking* (http://blogs.sas.com/content/graphicallyspeaking/) and his books (http://support.sas.com/publishing/authors/matange.html).

EXAMPLE CODE

Double-Click Here for All Example Code (The link is supported in Adobe Reader but not in many browsers.)

CONTACT INFORMATION

Warren F. Kuhfeld
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 531-7922
Warren.Kuhfeld@sas.com

Mary Beth Herring
Rho Inc.
6330 Quadrangle Dr.
Chapel Hill, NC 27517
(919) 595-6590
MaryBeth_Herring@rhoworld.com