PharmaSUG 2016 - Paper QT12

Breaking up (Axes) Isn't Hard to Do: A Macro for Choosing Axis Breaks

Alex Buck, Rho®, Chapel Hill, NC

ABSTRACT

SAS® 9.4 brought some wonderful new graphics options. One of the most exciting is the addition of the RANGE option for SGPLOT. As the name suggests, specifying ranges for a broken axis is controlled by the user. The only question left is where to set the breaks and if a break is actually needed. That is what this macro is designed to do. The macro analyzes the specified input parameter to create macro variables for the overall minimum and maximum, as well as macro variables specifying values prior to and following the largest difference that occurs between successive parameter values. The macro will also create variables for suggested break values to ensure graphic items such as markers are displayed in full. The user then utilizes these macro variables to determine if an axis break is needed and where to set those breaks. With the macro's dynamic nature it can be incorporated into larger graphics macro programs easily while making specific recommendations for each individual parameter. A complete and intuitive graph is produced with every macro call.

INTRODUCTION

This article focuses on the application of the %BreakIt, a SAS macro which sorts and analyzes data prior to plotting in order to fully utilize the RANGE option. This tool will allow users to dynamically and confidently set graph breaks based on the data inputted. Global macros, PROC SQL, PRXMATCH, and DATA options are used within the macro. The user should have a basic understanding of macro variables in order to call the macro variables produced in their proceeding graphic code.

The goal of the macro is to take the same data that is used to create the simple Figure 1 and use it to create Figure 2:

```
proc sgplot data=indat;
   scatter x=time y=value/group=trtn ;
run;
```

```
proc sgplot data=indat;
   scatter x=time y=value/group=trtn ;
   yaxis ranges=(&ovmin.-&lowbreak.
                 &upbreak.-&ovmax.);
   run;
```

Figure 1

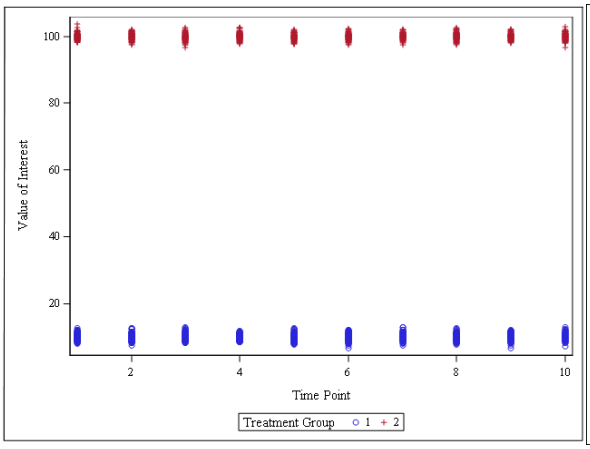

Figure 1. Simple Scatter Plot without Range Fix

Figure 2

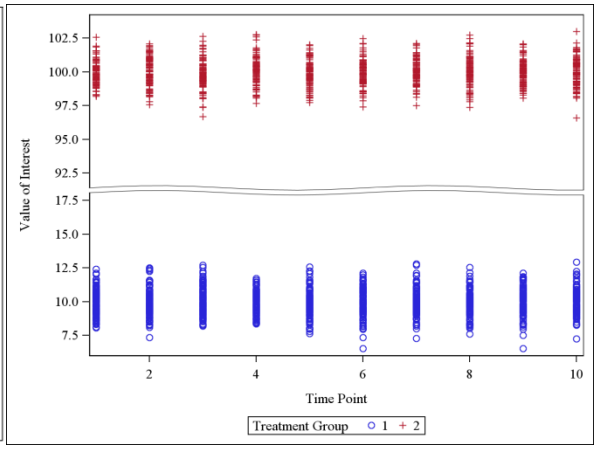

Figure 2. Simple Scatter Plot Utilizing %BREAKIT

THE MACRO

There are a few inputs required for the macro to analyze the data and output the macro variable values:

- INPUTDAT: The data to be graphed
- INVAL: The variable with the value graphed to the axis of interest
- GAPCHECK: The requested factor difference in input dataset values to consider the use of RANGE appropriate. Default is 2 times the lower value.
- MARG: The margin requested between the data values and the axis break. Default is 5.
- DIFFCHECK: The requested factor difference in macro variable values to consider the use of RANGE appropriate. Default is 2 times the lower value.
- OBSCHECK: The maximum number of iterations requested when determining macro values. Default is 100.

The macro call is as follows:

```
%BREAKIT (INPUTDAT  = ,
          INVAL     = ,
          GAPCHECK  = 2,
          MARG      = 5,
          DIFFCHECK = 2,
          OBSCHECK  = 100);
```

Once the macro is run the user will be provided global macro variables that can be called in their proceeding SGPLOT code. The output macro variables are as follows:

- &OVMIN: The overall minimum value from the dataset
- &OVMAX: The overall maximum value from the dataset
- &PRIOVAL: The dataset value prior to the largest value gap
- &POSTVAL: The dataset value following the largest value gap
- &GAPVAL: The size of the value gap defined as the difference between &PRIORVAL and &POSTVAL
- &LOWBREAK: The suggested value for the lower axis range
- &UPBREAK: The suggest value for the upper axis range
- &DIFFP: The factor difference between &UPBREAK and &LOWBREAK

CREATING MACRO VARIABLES FOR DATA VALUES

The user will specify the dataset input as well as the variable values analyzed as the axis of interest, referred to in the macro as AVAR. All missing values will be dropped from the input dataset:

```
data indat;
    set &inputdat;
    where not missing(&inval);
    avar=&inval;
run;
```

An SQL call on the data will create &OVMIN, the overall minimum, and &OVMAX, the overall maximum.

```
proc sql;
   select min(avar) into:ovmin from indat;
   select max(avar) into:ovmax from indat;
quit;
```

The dataset is sorted by AVAR. The first value in the new dataset will correspond to &OVMIN while the last value correspond to &OVMAX. Along with confirming the minimum and maximum values the sort keeps the closest values of AVAR next to each other. Within the sorted dataset a variable containing the previous ordered value for AVAR, called AVAR_LAG is created using LAG function. Subtracting AVAR_LAG from AVAR creates AVAR_GAP containing the data point gap value.

```
proc sort data=indat out=indat_asort;
   by avar;
run;

data avardat;
   set indat_asort;
   by avar;

   avar_lag=lag(avar);
   avar_gap=avar-avar_lag;
run;
```

Another sort by AVAR_GAP will reveal the largest difference in data point values in the dataset as the last observation. The END option of the SET statement creates the variable EOF as the end of dataset flag. Only the last observation is kept in the dataset. This observation is called in PROC SQL to create PRIORVAL as the value of AVAR prior to the largest gap, POSTVAL as the value of AVAR following the largest gap, and GAPVAL as the value of the gap defined as the difference between PRIORVAL and POSTVAL. If the GAPVAL is smaller than a set percentage of the PRIORVAL a message will be printed to the log. The percentage is determined by a user defined value or a default of 2 times the value of PRIORVAL.

```
proc sort data=avardat out=avardat_gsort;
   by avar_gap;
run;

data avardat_gsort_last;
   set avardat_gsort end=eof;
   by avar_gap;

   gapperc=avar_gap/abs(avar_lag);
   if eof;
run;

proc sql;
   select avar into:postval from avardat_gsort_last;
   select avar_lag into:priorval from avardat_gsort_last;
   select avar_gap into:gapval from avardat_gsort_last;
   select gapperc into:gapp from avardat_gsort_last;
quit;

%if %sysevalf(&gapp<&gapcheck) %then %do;
%put GAP IS < &gapcheck TIMES GREATER THAN PRIORVAL. RANGE MAY NOT BE APPROPRIATE
     FOR THIS DATA.;
%end;
```

CREATING THE RANGE VALUES

A graph using the strict data values will lead to cutoffs of the markers within the graph (see below). Notice how the line break goes through markers on the top and bottom. While setting the range allows the user to drop the extraneous information in the data gap, using exact data values can cause important information such as exact value points to be lost.

Figure 3

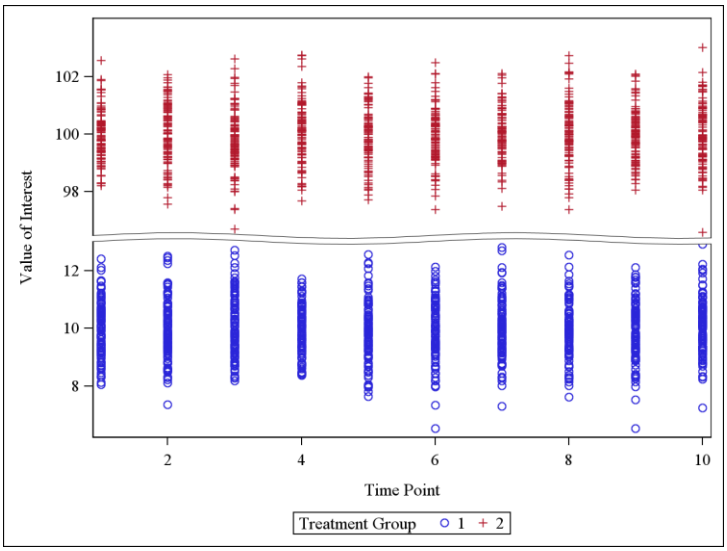

Figure 3. Scatter Plot with Ranges using Data Values

In order to avoid this issue, new macro variables are created based on the original prior/post values. The code analyzes the magnitude of the prior and post values and creates suggested lower and upper break values with a margin that can be set by the user or using the default of 5. The margin will require updates depending on graphic marker size and shape. The use of the margin will depend on the values in AVAR and AVAR_LAG. If both absolute values are greater than 1 than the margin is simply added to the AVAR_LAG and subtracted from the AVAR. In the event that one of the values is between -1 and 1 the first non-zero decimal place is determined in order to set the magnitude. The margin is then converted to the same magnitude and added or subtracted as appropriate.

The code below handles the case where AVAR_LAG requires a magnitude search. The first line within the do code creates a character version of the AVAR_LAG variable. If the decimal is small enough the character value will be of the form 'xxE-yy' where xx is the first two non-zero values after the decimal and yy is the index of the first non-zero number. If the character value is of the form 0.xxxxx where x can take values of zero and non-zero the character variable becomes a substring starting after the decimal place and a PRXMATCH call is performed to determine the index of the first non-zero number.

```
else if -1<avar_lag<1 and (avar>=1 or avar<=-1) then do;
      avar_lagc = strip(put(avar_lag, best.));
      if index(avar_lagc,'E-')>0 then
           lag_mag=input(substr(avar_lagc,(index(avar_lagc,'E-')+2)),8.);
      else lag_mag=prxmatch('/[^0]/', substr(avar_lagc,index(avar_lagc,'.') + 1));

      low=avar_lag+(&marg.*(0.1**lag_mag));
      up=avar-&marg.;
end;
```

Once the margin is added the macro values of UPBREAK and LOWBREAK are created. If the margin is too large the value of UPBREAK may be less than the value of LOWBREAK. A range check is performed. If this is the case the margin is decrease by 1 and the code is run again. If the margin is very large then the first iteration where UPBREAK>LOWBREAK may be a smaller difference in values than desired. Another check is done on the percentage difference between UPBREAK and LOWBREAK. If the percentage difference is lower than the user defined value or default of 0.1 the margin is again decreased by 1 and the code is run again. There is a stopping rule based on number of iterations. If the number of iterations exceeds the user defined number or default of 100 a note is placed in the log and the final value of UPBREAK and LOWBREAK stand. Note in the code below that the %FINDRANG macro is producing UPBREAK and LOWBREAK based on the logic in previous paragraphs.

```
%let ready=0;
%do %until (&ready=1);

   %findrang(&marg);
   %let stopobs=%sysevalf(&stopobs+1);

   %if %sysevalf(&upbreak<=&lowbreak) or %sysevalf(&diffcheck<0.1) %then %do;
      %let ready=0 ;
      %let marg=%sysevalf(&marg-1);
   %end;

   %else %if %sysevalf(&upbreak>&lowbreak) and %sysevalf(&diffcheck >=0.1) %then
%do;
      %let ready=1;
   %end;

   %if %sysevalf(&stopobs>&obscheck) %then %do;
      %put MAXIMUM NUMBER OF ITERATIONS REACHED. PLEASE REVIEW MARGIN AND
            PERCENTAGE DIFFERENCE INPUTS;
      %let ready=1;
   %end;

%end;
```

After all checks are performed the user will have the values needed to create an acceptable graphic with a broken axis.

```
proc sgplot data=indat;
   scatter x=time y=value/group=trtn ;
   yaxis ranges=(&ovmin.-&lowbreak. &upbreak.-&ovmax.);
run;
```

Figure 4

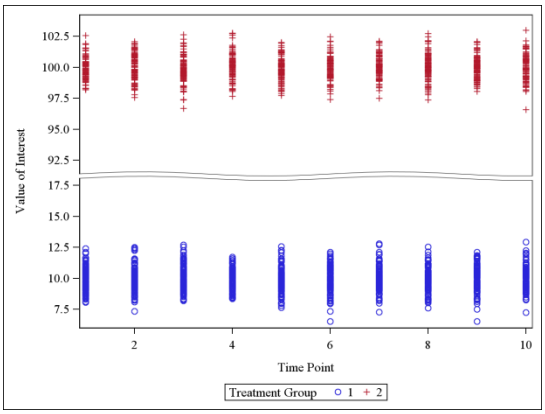

Figure 4. Scatter Plot with Ranges using %BREAKIT Macro Variables

CONCLUSION

With the utilization of %BREAKIT dynamic and clean graphs can be created quickly with less manual tinkering of the axis break. While this paper focused on the y-axis these methods can be performed with x-axis values as well. It is the belief of the author that this macro could reduce the amount of time taken for large runs of similar graphs such as laboratory parameters.

RECOMMENDED READING

- http://www.regular-expressions.info/
- SAS® 9.4 ODS Graphics: Procedures Guide, Fifth Edition.

CONTACT INFORMATION

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

Name: Alex Buck
Enterprise: Rho
Address: 6330 Quadrangle Drive
City, State ZIP: Chapel Hill, NC 27517
E-mail: alexandra_buck@rhoworld.com