Paper AD-190

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

Alex Buck, Rho®

ABSTRACT

This 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 is intended as a follow-up to PharmaSUG 2016 Paper QT12. It will focus on the application of %BreakIt, as a SAS macro which sorts and analyzes data and returns RANGES or VALUES option text as appropriate. This tool will allow users to dynamically and confidently produce graphs based on the input data while minimizing the manual updates. The macro utilizes %AxisOrder which was presented at SESUG in 2015 as Paper RIV122 and is available online. The user is encouraged to review the %AxisOrder paper as well as SAS macro documentation before using this macro.

The purpose of this macro is to provide the user with a macro variable that can be called in PROC SGPLOT and will accommodate many different scenarios in the data. This can include no gaps being present and up to three gaps being present in the data. For the purposes of this paper, a gap is defined as a large space between values. See below for some quick examples:

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

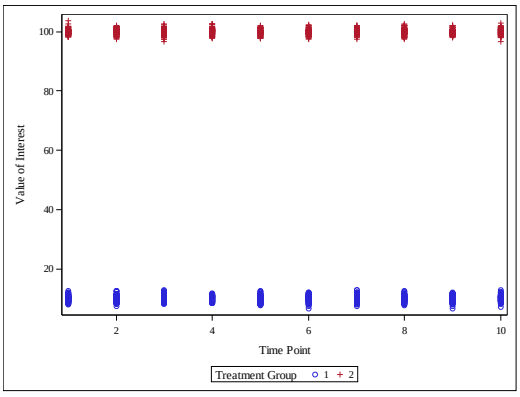

Figure1. Plot with 1 Gap

```
proc sgplot data=norm_trtdiff;
scatter x=time y=value/group=trtn ;
yaxis &FigText;
run;
```

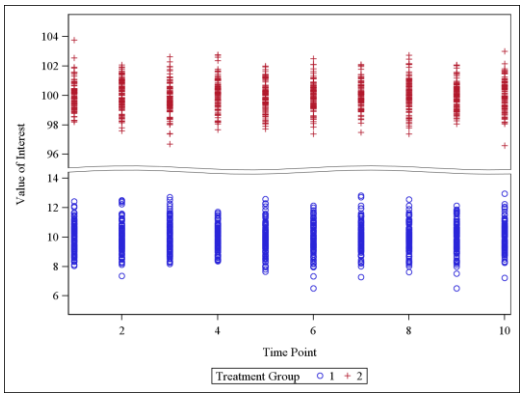

Figure 2. 1 Gap Plot Utilizing &FigText from %BreakIt

```
proc sgplot data=norm_trtdiff_3gap;
   scatter x=time y=value/group=trtn ;
run;
```

```
proc sgplot data=norm_trtdiff_3gap;
   scatter x=time y=value/group=trtn ;
   yaxis &FigText;
run;
```

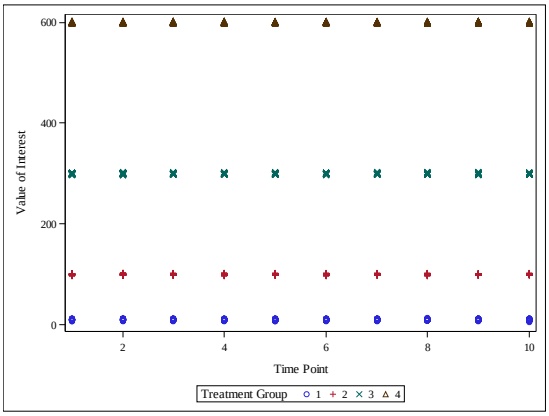

Figure 3. Plot with 3 Gaps

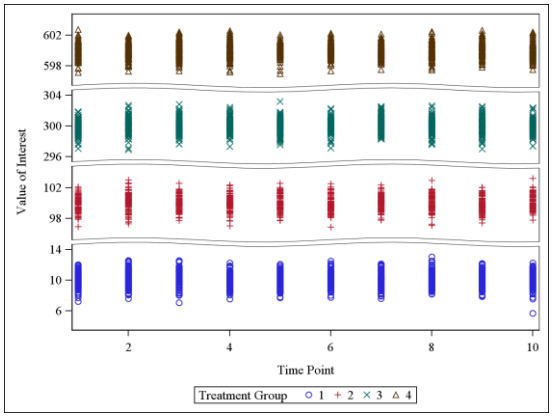

Figure 4. 3 Gap Plot Utilizing &FigText from %BreakIt

THE MACRO

The macro has five required input values

- Data: Input dataset. The same dataset will be used in the SGPLOT call.
- Var: The variable on the axis of interest. Will be used to derive VALUES/RANGES recommendation.
- ChkPct: The requested percentage of the overall range that a gap must meet to consider the use of RANGE appropriate. The default is 0.25 (25%).
- MarPct: The requested space, as a percentage of effective range, to be added above/below RANGES values. The default is 0.10 (10%).
- MaxGap: Limits the macro in the number of gaps to be used. The default is 3, the maximum allowed in the program.

The macro call is as follows:

```
%Breakit (data=,
          var=,
          chkpct= 0.25,
          marpct= 0.10,
          maxgap= 3);
```

The macro outputs will be determined entirely on the data and the user-defined percentage checks. If no gap meeting the criteria is found in the data, the macro will output the following variables:

- OvMin: The overall minimum value from the dataset.
- OvMax: The overall maximum value from the dataset.
- FigLow1: The lower limit of the graph values.
- FigUp1: The upper limit of the graph values.
- FigBy: The derived axis increment/by value.
- FigText: Text string for XAXIS/YXAXIS statement. Will take the value VALUES=(&FigLow1 to &FigUp1 by &FigBy).

If at least one gap meets the criteria, the macro will output the following variables:

- OvMin: The overall minimum value from the dataset.
- OvMax: The overall maximum value from the dataset.
- FigLow1: The derived lower value for the first range.
- FigUp1: The derived upper value for the first range.
- FigLow2: The derived lower value for the second range.
- FigUp2: The derived upper value for the second range.
- FigLow3, FigUp3, FigLow4, FigUp4 are created if 2 or 3 gaps meet criteria.
- FigText: Takes the value RANGES=(&FigLow1-&FigUp1 &FigLow2-&FigUp2) for data with one gap. &FigLow3-&FigUp3 and &FigLow4-&FigUp4 are included if 2 or 3 gaps meet criteria.

MACRO PROCESS

First and foremost, the macro confirms that the dataset and variable exist. If this is not the case, a note will be written to the log and the macro will abort. If both the dataset and variable exist, a new dataset is created and the variable is renamed to AVAR.

```
data _bi_indat;
      set &Data.;
      where not missing(&Var.);
      avar=&Var.;
run;
```

The overall minimum and overall maximum will be put into macro variables OVMIN and OVMAX. The dataset is then sorted by AVAR. Next AVAR_LAG is created using the LAG function. Subtracting AVAR_LAG from AVAR_GAP creates the data point gap value, AVAR_GAP. The overall range is derived from OVMAX-OVMIN. If AVAR_GAP is greater than or equal to the user-specified percentage of overall range (CHKPCT) then RANGES is considered appropriate and the macro will move forward. Otherwise, the macro will put a note in the log and call %AxisOrder.

```
%if &GapFl.^=Y %then %do;

    %put BREAKIT -> The largest gap is not greater than the user-specified cutoff
    (OvRange * [ &ChkPct. ]). RANGES option is not appropriate.;

    %axisorder(data=_bi_indat, var=avar);
    %let FigLow1=&_AxisStart.;
    %let FigUp1=&_AxisEnd.;
    %let FigBy=&_AxisBy.;
    %let FigText=%str(VALUES=(&FigLow1. to &FigUp1. by &Figby.));

/* List Final Macro Values */
    %put OvMin    = [ &OvMin. ];
    %put OvMax    = [ &OvMax. ];
    %put FigLow1  = [ &FigLow1. ];
    %put FigUp1   = [ &FigUp1. ];
    %put FigBy    = [ &FigBy. ];
    %put FigText  = [ &FigText. ];

%end;
```

The next check is to determine the number of gaps to be utilized. The data is sorted so that the largest AVAR_GAP value is first, then the second, and so on until the MaxGap value is reached. The data is then sorted by AVAR_LAG to set the gaps in order of data values. The data is then collapsed to include all relevant range values in one observation. These values will include the minimum and maximum as well as the AVAR and AVAR_LAG values meeting the gap criteria. These AVAR and AVAR_LAG values will be renamed RAVAR and RAVA_LAG with a suffix of 1, 2, or 3 depending on their order after the sort.

While the range values start as values from the input dataset, they are then updated by adding or subtracting a percentage of the effective range. This will ensure that the axis break line will not cut off the marker. The effective

range is defined as the sum of each range's lower limit subtracted from its upper limit, or the sum of the non-gap graph space. In the case of two gaps the data starts with three ranges: MIN-RAVAR_LAG1, RAVAR1-RAVAR_LAG2, RAVAR-MAX. These are then transformed using the MarPct and appropriate EFFRANGE to create three new ranges: LOW1-UP1, LOW2-UP2, LOW3-UP3.

```
if ngap=2 then do;
    effrange1  = (ravar_lag2-ravar1)+(ravar_lag1-min);
    effrange2  = (max-ravar2)+(ravar_lag2-ravar1);
    effrange12 = max(effrange1,effrange2);
    low1       = min - (effrange1*&MarPct.);
    up1        = ravar_lag1 + (effrange1*&MarPct.);
    low2       = ravar1 - (effrange12*&MarPct.);
    up2        = ravar_lag2 + (effrange12*&MarPct.);
    low3       = ravar2 - (effrange2*&MarPct.);
    up3        = max + (effrange2*&MarPct.);
end;
```

Note that EFFRANGE12 is used for the second range and is the maximum of EFFRANGE1 and EFFRANGE2. That is because the second range will share an effective range with both the first and third ranges. As a precaution, the largest effective range is used when finding the margins to add and subtract.

If the MarPct value is too large the upper limit of the first range may be greater than or equal to the lower limit of the second range, causing an overlap. If the issue is not addressed, SAS will merge the ranges where the upper limit of the previous range equals the lower limit of the current range or ignore the RANGES option if the upper limit is greater than the lower limit. The macro will look for any overlap between ranges. If they are present, the macro will put a note in the log advising the user to review their MarPct. The macro will then merge the ranges and update the number of gaps present to allow the macro to continue.

If we return to our example with two gaps, we see checks at each of the possible overlap points. The potential scenarios (all ranges overlap, only one overlaps) are tested. Range values are merged and NGAP is updated as appropriate. If NGAP is reset to 0, the %AxisOrder macro will be called and a VALUES statement will be produced. For example if the first gap overlaps with the second, the upper value of the second range becomes the upper value of the first range. The lower and upper values of the third range become the lower and upper values of the second range. The number of gaps is reset to one and the macro continues with only two ranges of interest.

```
if up1>=low2 then do;
    putlog "BREAKIT -> UP1 >= LOW2. Review MarPct and/or MaxGap. Ranges will be
            merged";
end;
if up2>=low3 then do;
    putlog "BREAKIT -> UP2 >= LOW3. Review MarPct and/or MaxGap. Ranges will be
            merged”;
end;
if up1>=low2 and up2>=low3 then do;
    call symputx('NGAP',0);
end;
else if up1>=low2 and up2<low3 then do;
    up1=up2;
    low2=low3;
    up2=up3;
    call symputx('NGAP',1);
end;
else if up2>=low3 and up1<low2 then do;
    up2=up3;
    call symputx('NGAP',1);
end;
```

Once the final number of gaps and the range values are set, the macro variables for the axis break are created. The macro values are printed to the log for user review.

```
   %if &NGap.=2 %then %do;

       proc sql;
              select low1 into:low1 from _bi_range;
              select up1  into:up1 from _bi_range;
              select low2 into:low2 from _bi_range;
              select up2  into:up2 from _bi_range;
              select low3 into:low3 from _bi_range;
              select up3  into:up3 from _bi_range;
       quit;

       %let FigLow1 = &low1.;
       %let FigUp1  = &up1.;
       %let FigLow2 = &low2.;
       %let FigUp2  = &up2.;
       %let FigLow3 = &low3.;
       %let FigUp3  = &up3.;
       %let FigText = %str(RANGES=(&FigLow1.-&FigUp1. &FigLow2.-&FigUp2.
                                   &FigLow3.-&FigUp3.));

       %put OvMin    = [ &OvMin. ];
       %put OvMax    = [ &OvMax. ];
       %put FigLow1  = [ &FigLow1. ];
       %put FigUp1   = [ &FigUp1. ];
       %put FigLow2  = [ &FigLow2. ];
       %put FigUp2   = [ &FigUp2. ];
       %put FigLow3  = [ &FigLow3. ];
       %put FigUp3   = [ &FigUp3. ];
       %put FigText  = [ &FigText. ];

    %end;
```

The user can then call &FigText within their axis statement to produce a plot with dynamic axis values.

```
proc sgplot data=norm_trtdiff_2gap;
   scatter x=time y=value/group=trtn ;
   yaxis &FigText;
run;
```

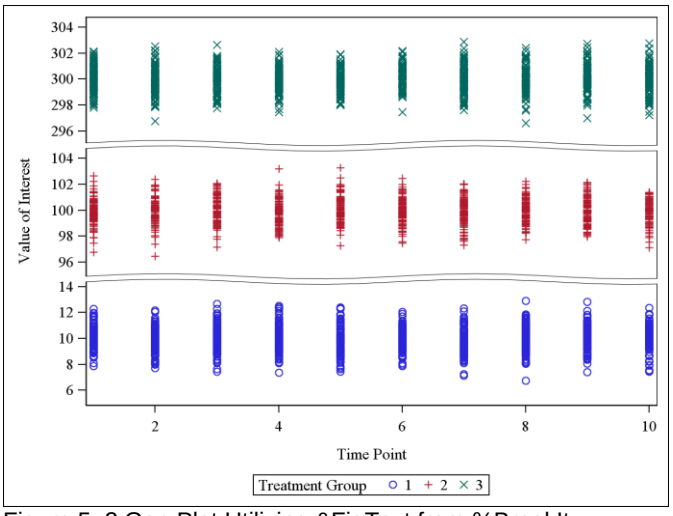

Figure 5. 2 Gap Plot Utilizing &FigText from %BreakIt

CONCLUSION

With the utilization of %BREAKIT, dynamic and clean graphs can be created quickly with less manual tinkering of the axis statement. The user retains control of the requirements for RANGES to be used and the maximum number of gaps they would like to see while allowing the macro to search for the most appropriate data-driven display values.

While this paper focused on the y-axis, these methods can be performed with x-axis values as well. It is the belief of this author that this macro will make graphing more efficient for users.

REFERENCES

Rosanbalm, Shane. 2016. "The Last Axis Macro You'll Ever Need." *Proceedings of the SESUG 2015 Conference*. Available at http://www.lexjansen.com/sesug/2015/122_Final_PDF.pdf

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 Dr
City, State ZIP: Chapel Hill, NC 27517
E-mail: Alexandra_Buck@Rhoworld.com

APPENDIX

Creating Data with One Gap

```
data norm_trtdiff;
   do i=1 to 2;
      do j=1 to 100;
         do k=1 to 10;
         trtn=i;
         subject=j;
         time=k;
         if trtn=1 then value=rand('NORMAL',10,1);
         else if trtn=2 then value=rand('NORMAL',100, 1);output;
         end;
      end;
   end;
   label trtn='Treatment Group' time='Time Point' value='Value of Interest';
run;
```

Creating Data with Two Gapsdata norm_trtdiff_2gap;

```
   do i=1 to 3;
      do j=1 to 100;
         do k=1 to 10;
         trtn=i;
         subject=j;
         time=k;
         if trtn=1 then value=rand('NORMAL',10,1);
         else if trtn=2 then value=rand('NORMAL',100, 1);
         else if trtn=3 then value=rand('NORMAL',300, 1); output;
         end;
      end;
   end;
   label trtn='Treatment Group' time='Time Point' value='Value of Interest';
run;
```

Creating Data with Three Gaps

```
data norm_trtdiff_3gap;
   do i=1 to 4;
      do j=1 to 100;
         do k=1 to 10;
         trtn=i;
         subject=j;
         time=k;
         if trtn=1 then value=rand('NORMAL',10,1);
         else if trtn=2 then value=rand('NORMAL',100, 1);
         else if trtn=3 then value=rand('NORMAL',300, 1);
         else if trtn=4 then value=rand('NORMAL',600, 1);output;
         end;
      end;
   end;
   label trtn='Treatment Group' time='Time Point' value='Value of Interest';
run;
```