**SCSUG 2018**

# Animate your Data!

Kriss Harris, SAS Specialists Limited; Richann Watson, DataRich Consulting

## ABSTRACT

When reporting your safety data, do you ever feel sorry for the person who has to read all the laboratory listings and summaries? Or have you ever wondered if there is a better way to visualize safety data? Let's use animation to help the reviewer and to reveal patterns in your safety data, or in any data!

This hands-on workshop demonstrates how you can use animation in SAS® 9.4 to report your safety data, using techniques such as visualizing a patient's laboratory results, vital sign results, and electrocardiogram results and seeing how those safety results change over time. In addition, you will learn how to animate adverse events over time, and how to show the relationships between adverse events and laboratory results using animation. You will also learn how to use the EXPAND procedure to ensure that your animations are smooth. Animating your data will bring your data to life and help improve lives!

## INTRODUCTION

By animating your data, whether it is over time, by subject or some other key component it can be easier to understand and interpret your data. Such as, you can quickly get an idea of the number of subjects that had abnormal laboratory results. You are also able to easily determine the adverse events each subject had and the duration of the adverse event. Using SAS 9.4 you can create animations with the new animation OPTIONS in conjunction with the PRINTER destination.

## ANIMATION AND GTL

You can animate figures that you have produced using Graph Template Language (GTL) or the SGPLOT, SGPANEL and SGSCATTER procedures by encapsulating those procedures within the animation OPTIONS and the ODS PRINTER statements. In all the examples within this paper, the focus will be on producing figures using GTL. Sample Code 1 below details how you can produce animations.

```
proc template;
 define statgraph <template-name>;
 begingraph / <options>;
 <GTL statements>;
 endgraph;
 end;
run;
```

1

```
options printerpath=gif animation=start animduration=0.5 animloop=yes
noanimoverlay;
ods printer file="C:\animation.gif";

  proc sgrender data=<data-set-name> template=<template-name>;
   <other optional statements>;
  run;

options printerpath=gif animation=stop;
ods printer close;
```

3

2

**Sample Code 1: Example Animation Program**

1 The structure of the graph in the form of the STATGRAPH template using GTL is defined here. So you can create a template here that produces a boxplot, scatterplot, line plot or any other plot of your choice.

For more details on using GTL please see (Harris, Hands-On Graph Template Language (GTL): Part A, 2017) or (Harris & Watson, Great Time to Learn GTL, 2018).

2 Prior to specifying the SGRENDER procedure, we need to specify the necessary animation options as well as the ODS printer statement which points to where the GIF file will be stored. To specify the animiation options, in the OPTIONS statement, we utilize the PRINTERPATH option which generates a GIF image, the ANIMATION option which starts the creation of the animation file, and the ANIMDURATION option which specifies the duration that each frame is held. In our example, we want each frame to be held for 0.5 seconds. In addition, the ANIMLOOP option specifies to continuously repeat the animation loop. The NOANIMOVERLAY option specifies that each frame is played sequentially. Further on, after Proc SGRENDER, the ANIMATION options stops creating animation files and the printer is closed. The default values for the animation options are:

- ANIMATION = STOP
- ANIMDURATION = MIN (based on device driver's default)
- ANIMLOOP = YES
- ANIMOVERLAY

3 Here, you associate the data with the template using the SGRENDER procedure to create the graph. To animate figures, SGRENDER will need to produce more than one image. Therefore you will either need to use the BY statement in the SGRENDER procedure or use multiple SGRENDER statements or create a macro that will loop through each combination.

## ANIMATING LABORATORY RESULTS

Typically, when assessing safety within laboratory results a summary table of the abnormal results for each laboratory parameter is produced, and a listing of every observation for each parameter is produced. These tables and listings can become very large – especially the listings, however the outputs are currently required by the regulatory authorities. Occasionally boxplots are produced of the laboratory parameters, and the boxplots show the distribution of the laboratory measurements by treatment and/or any other variables of interest. From the boxplots you can interpret if any of the measurements are above or below the standard normal limits.

Generally, these boxplots only show one visit per page, and there are usually a lot of visits, and so it is currently difficult to assess how the measurements are changing over time. Usually this assessment is done by scrolling down many pages (and then having to scroll back up and down again). Using animation will make this process easier because the screen will be fixed, and the visits will be incremented periodically, and you will be able to easily see how the data is changing over time.

The data set used in this paper is from the CDISC SDTM / ADAM Pilot Project and this is obtained from the CDISC website (CDISC, 2013). The data set has been further filtered down to only contain the Creatinine laboratory parameter and only male subjects. The reason the data set has been filtered to only contain males is to make the examples easier to follow. The counts of the total number of subjects with observations for each treatment and visit have been added on to the data set too. This enables the number of subjects for each treatment group at each visit to be displayed on the animation.

When producing animations, it is important to know what each image is showing. For example, if your graph is for laboratory data over time, you would want to know the laboratory parameter and the visit that each image within the animation is displaying. The DYNAMIC statement along with the ENTRYTITLE statement in GTL can help you to place identifiers on the images, in particular when you are using the BY statement in SGRENDER. In the example below the DYNAMIC, ENTRYTITLE and BY statement allows us to know the laboratory parameter, gender and visit that is being plotted. Figure 1, shows an animation of the boxplots of the laboratory results by treatment, and it is animated over time. Click on the figure to play the animation. Program 1 created the Figure 1 output.

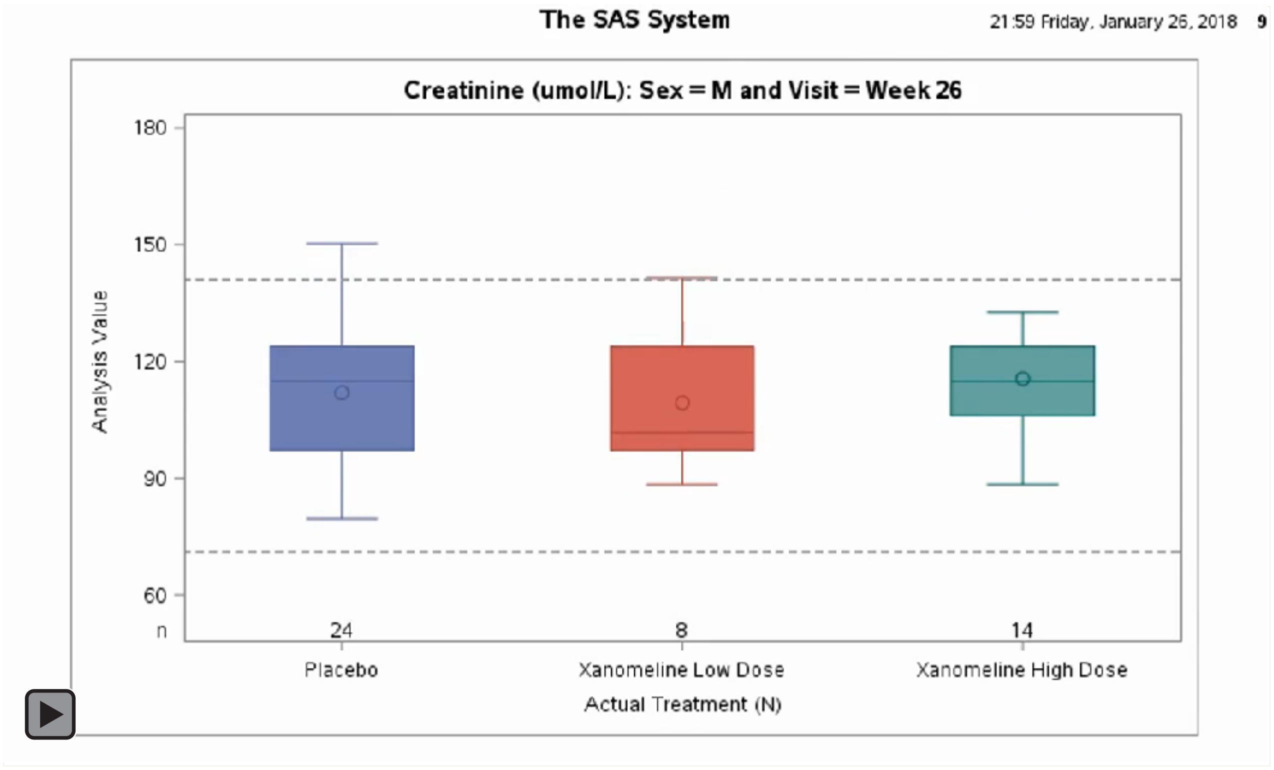


**Figure 1: Animated Boxplot of Laboratory Results**

```
 * Creating template;
 proc template;
   define statgraph boxplot_template_sex;
     dynamic _byval_ _byval2_ _byval4_ upperlim lowerlim;
     begingraph;
         entrytitle halign = center _byval_ ":" " Sex = " _byval2_ " and
           Visit = " _byval4_;
         layout overlay / yaxisopts=(linearopts=(viewmin=60 viewmax=180
             tickvaluesequence=(start=60 end=180 increment=30)));
           boxplot x = trtan y = aval / group = trtan groupdisplay=cluster;   1
           referenceline y=lowerlim / lineattrs=(pattern=2);
           referenceline y=upperlim / lineattrs=(pattern=2);
           innermargin;
             axistable x = trtan value = nobs_subject_visit / stat=mean
             label= "n" valueattrs=(size=9);
           endinnermargin;
         endlayout ;
     endgraph;
   end;
 run;
```

2

```
* Counting distinct subjects numbers in each treatment group at each
timepoint;
   * ADLBC - Source data set;
   data adlbc_all;
     set source.adlbc;
     where param in ("Creatinine (umol/L)") and ablfl ne "Y";
     avisit=strip(avisit);
   run;
```

3

```
   * Adding on n numbers;
   proc sql;
     create table adlbc_all_n_gen as
     select *, count(distinct usubjid) as nobs_subject_visit
     from adlbc_all
     group by trtan, avisitn, sex;
   quit;

   proc sort data = adlbc_all_n_gen;
     by param sex avisitn avisit trtan;
   run;
```

4

```
   * Producing animation;
   options nobyline;

   options papersize=('8 in', '4.8 in') printerpath=gif animation=start
   animduration=0.5 animloop=yes noanimoverlay;
   ods printer file="&outpath\boxplotM1.gif";

   ods graphics / width=8in height=4.8in imagefmt=gif;

   proc sgrender data = adlbc_all_n_gen template = boxplot_template_sex;
     where avisitn ne . and sex = "M";
     by param sex avisitn avisit;
     dynamic upperlim = "a1hi" lowerlim = "a1lo";
     format trtan trtfmt.;
   run;

   options printerpath=gif animation=stop;
   ods printer close;
```

**Program 1: Animating Boxplot of Laboratory Results**

1 The template used to produce the boxplot is defined here. The dynamic variables _BYVAL_, _BYVAL2_, and _BYVAL4_ are used to display the appropriate laboratory parameter, gender, and visit respectively, during the animation. There is no _BYVAL3_ being used because in this example, _BYVAL3_ would display the visit number, and only showing the visit is sufficient, and the visit is obtained from _BYVAL4_.

The BOXPLOT statement produces the boxplots, the REFERENCELINE statements draw the standard lower and upper limits for the parameter and the AXISTABLE is used to display the number of patients in each treatment group.

2 Filtering the data set to only contain the post-dose “Creatinine” lab results.

3 Calculating the number of subjects for each treatment group, visit and gender. In the ADLBC_ALL_N_GEN data set, the number of subject is captured on every record, and the same number of subjects is shown for each combination of treatment, visit and gender. This is why in the AXISTABLE statement in Section 1 of Program 1, the STAT=MEAN option takes the mean of the records (which is actually the same as actual number) for each treatment, visit and gender and then displays this number on the graph, to show the number of subjects at each treatment group.

4 Creating the animated file.

## SCATTERPLOT OF LABORATORY RESULTS

Scatterplots can be useful at explaining laboratory results, especially when the number of observations in each treatment group is quite small and/or you want to see the amount of results that fall outside the standard normal limits. Program 2 below was used to produce an animation of the scatterplots as seen in Figure 2, which at each visit shows the Creatinine measurements by the treatment and also shows the number of subjects whose results fell outside the standard normal limits.

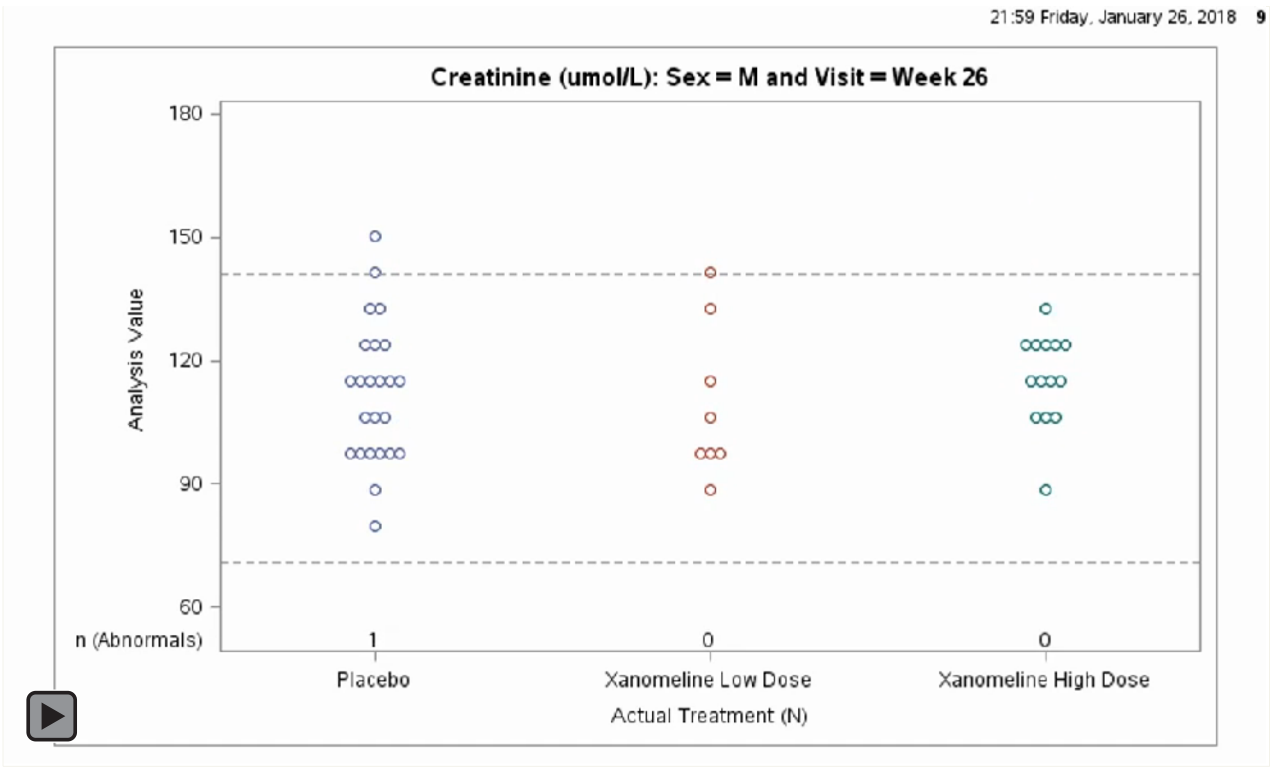


**Figure 2: Animated Scatterplot of Laboratory Results**

```
* For Males;
proc template;
  define statgraph scatterplot_template_sex;
    dynamic _byval_ _byval2_ _byval4_ upperlim lowerlim;
    begingraph;
        entrytitle halign = center _byval_ ":" " Sex = " _byval2_ " and
             Visit = " _byval4_;
        layout overlay / yaxisopts=(linearopts=(viewmin=60 viewmax=180
            tickvaluesequence=(start=60 end=180 increment=30)));
          scatterplot x = trtan y = aval / group = trtan groupdisplay =
            cluster jitter=auto jitteropts=(axis=x);
          referenceline y=lowerlim / lineattrs=(pattern=2);
          referenceline y=upperlim / lineattrs=(pattern=2);
          innermargin;
            axistable x = trtan value = nobs_abn_subject_visit / stat=mean
            label= "n" valueattrs=(size=9);
          endinnermargin;
        endlayout;
     endgraph;
```

1

```
  end;
run;
```

```
  * Counting distinct subjects numbers with abnormal observations in each
  treatment group at each timepoint;
  proc sql;
    create table abnormal_n as
    select distinct param, trtan, avisitn, avisit, sex,
      count(distinct usubjid)
    as nobs_abn_subject_visit
    from adlbc_all (where=(LBNRIND ne "NORMAL"))
    group by param, trtan, avisitn, sex;
  quit;
```
2

```
  * Merging abnormal n numbers onto adlbc_all data set, for plotting;
  proc sql;
    create table adlbc_all_abnormal_n_gen as
    select a.*, b.nobs_abn_subject_visit
    from adlbc_all as a left join abnormal_n as b
    on a.param = b.param and a.sex = b.sex and a.avisitn = b.avisitn and
    a.avisit = b.avisit and a.trtan = b.trtan;
  quit;

  proc sort data = adlbc_all_abnormal_n_gen;
    by param sex avisitn avisit trtan;
  run;
```
3

```
  options nobyline;

  goptions reset=all;
  options papersize=('8 in', '4.8 in') printerpath=gif animation=start
  animduration=0.5 animloop=yes noanimoverlay;
  ods printer file="&outpath\scatterplotM1.gif";

  ods graphics / width=8in height=4.8in imagefmt=gif;

  proc sgrender data = adlbc_all_abnormal_n_gen template =
   scatterplot_template_sex;
    where avisitn ne . and sex = "M";
    by param sex avisitn avisit;
    dynamic upperlim = "a1hi" lowerlim = "a1lo";
    format trtan trtfmt.;
  run;

  options printerpath=gif animation=stop;
  ods printer close;
```
4

**Program 2: Animating Scatterplot of Laboratory Results**

1 This is very similar to Section 1 of Program 1. The only differences here are that the template produces a scatterplot instead of a boxplot because the SCATTERPLOT statement is used, and the number of abnormal subjects are displayed instead of the total number of subjects.

2 Calculating the number of abnormal subjects for each laboratory parameter, treatment group, visit and gender. In this example only the Creatinine laboratory parameter is used.

3 Merging the abnormal results onto the data set, so that the final data set is ready to be used.

4 In order to produce the animated scatterplot, we need to reference the new template and the new data set. Again this is very similar to Section 4 of Program 1, with the only differences being the output filename, and the input data set and template on Proc SGRENDER statement.

## SCATTERPLOT OF LABORATORY RESULTS WITH SMOOTH ANIMATION

In Figure 2, you may have noticed that the animation shows an image of a scatterplot by treatment at each time-point. However, it may be more beneficial to see how a subject's results change over time (i.e., from visit to visit). This can be accomplished with the aid of interpolation.

As mentioned in (McCarthy, 2017), interpolation is necessary in order to create enough frames to make a smooth animation. However, it is important that the interpolation method does not misrepresent the underlying data. Although linear interpolation is the simplest option, splines make the animations smoother and more natural. Both interpolation methods can be done using the EXPAND procedure, and in the following example the linear interpolation method will be used as it is the simplest option and does not misrepresent the underlying data. Figure 3, shows the scatterplot of laboratory results using linear interpolation.

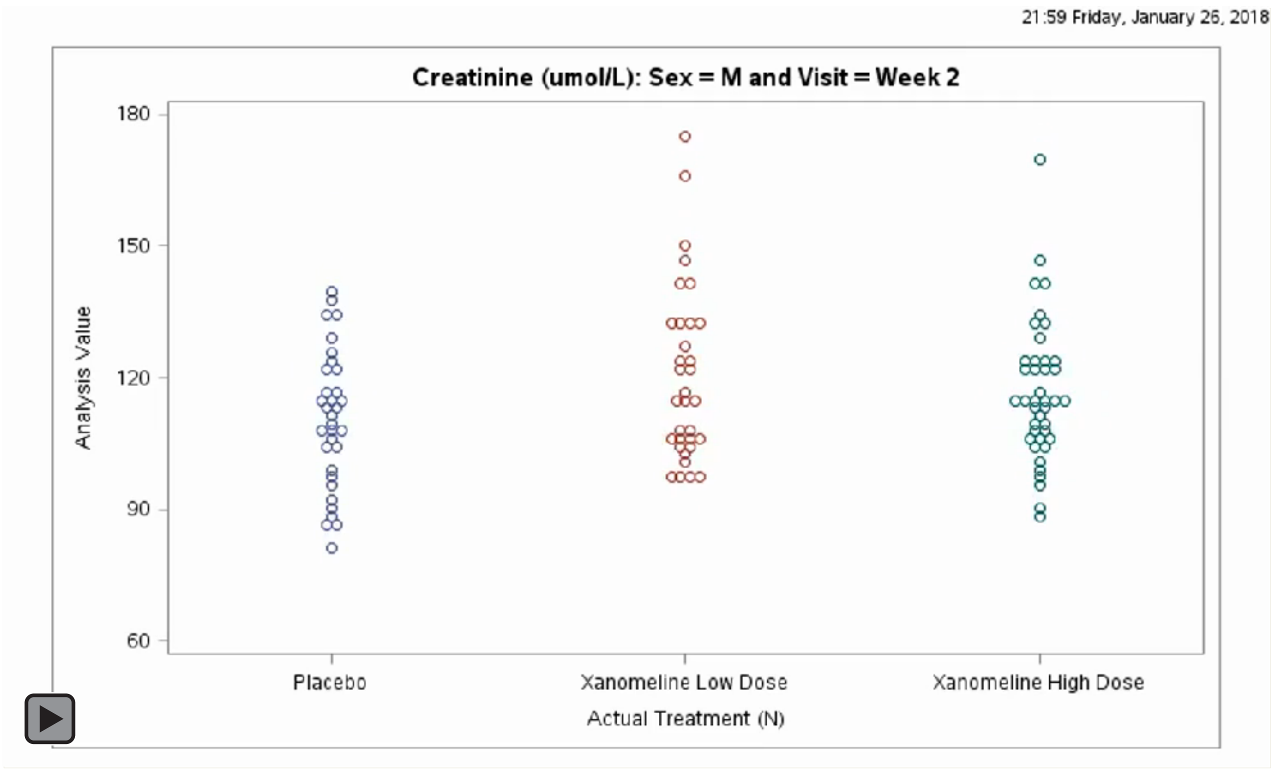
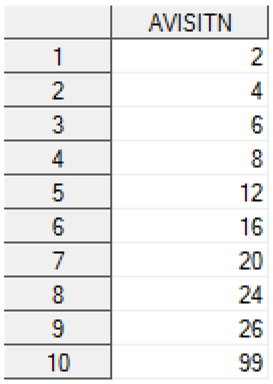


**Figure 3: Animated Scatterplot of Laboratory Results using Linear Interpolation**

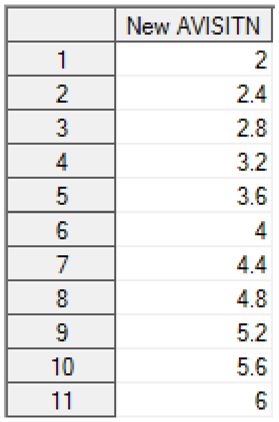
The first thing to do when considering interpolation is to assess the parameter that will be used to drive your animation. In this case the parameter is the visit number: AVISITN. In this example, as seen in Output 1, there are 10 visits. The visits 2 to 26 are on-treatment visits and visit 99 is the follow-up visit.



| | AVISITN |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 8 |
| 5 | 12 |
| 6 | 16 |
| 7 | 20 |
| 8 | 24 |
| 9 | 26 |
| 10 | 99 |

**Output 1: Values of AVISITN**

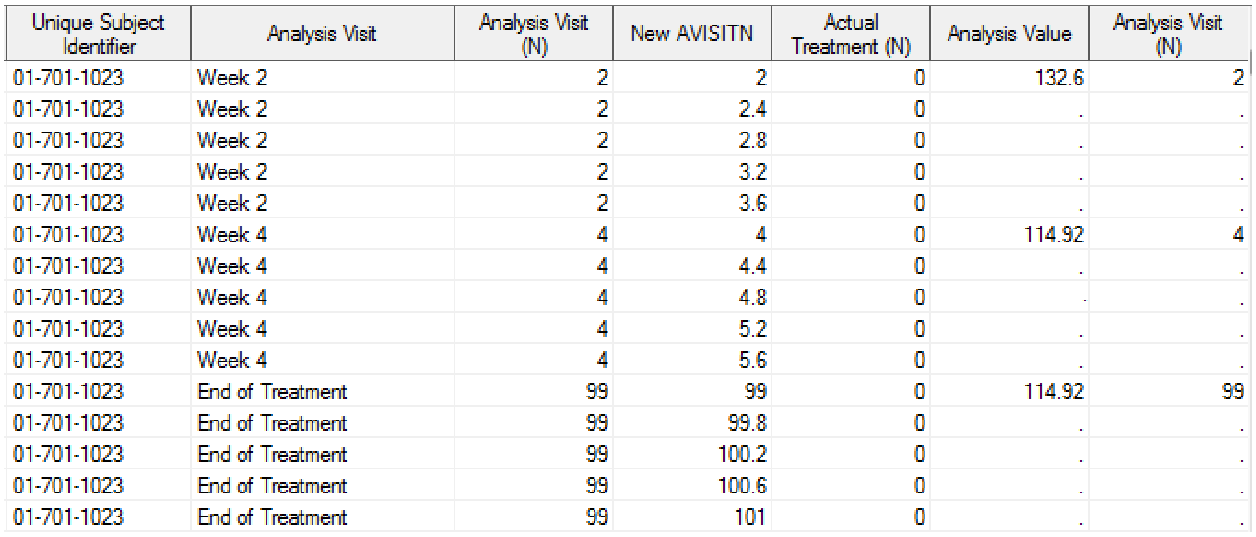
To make the animation smoother between the visits, the decision was made to add 4 new visits between each of the existing visits, and so instead of AVISITN being 2 and 4, it would now be 2, 2.4, 3.2, 3.6 and 4.Output 2, illustrates how the new visits will look. It is also useful to add 4 visits after the last value, which in this case is visit 99, and the benefit of adding those 4 visits is so the animation does not abruptly stop when it gets to visit 99.



| | New AVISITN |
|---|---|
| 1 | 2 |
| 2 | 2.4 |
| 3 | 2.8 |
| 4 | 3.2 |
| 5 | 3.6 |
| 6 | 4 |
| 7 | 4.4 |
| 8 | 4.8 |
| 9 | 5.2 |
| 10 | 5.6 |
| 11 | 6 |

**Output 2: Samples Values of New AVISITN**

The next step is to merge the data together in a manner so that the new visit records are between the existing visit records as illustrated in Output 3. This is will allow the interpolation to done on the **Analysis Value** column.



| Unique Subject Identifier | Analysis Visit | Analysis Visit (N) | New AVISITN | Actual Treatment (N) | Analysis Value | Analysis Visit (N) |
|---|---|---|---|---|---|---|
| 01-701-1023 | Week 2 | 2 | 2 | 0 | 132.6 | 2 |
| 01-701-1023 | Week 2 | 2 | 2.4 | 0 | . | . |
| 01-701-1023 | Week 2 | 2 | 2.8 | 0 | . | . |
| 01-701-1023 | Week 2 | 2 | 3.2 | 0 | . | . |
| 01-701-1023 | Week 2 | 2 | 3.6 | 0 | . | . |
| 01-701-1023 | Week 4 | 4 | 4 | 0 | 114.92 | 4 |
| 01-701-1023 | Week 4 | 4 | 4.4 | 0 | . | . |
| 01-701-1023 | Week 4 | 4 | 4.8 | 0 | . | . |
| 01-701-1023 | Week 4 | 4 | 5.2 | 0 | . | . |
| 01-701-1023 | Week 4 | 4 | 5.6 | 0 | . | . |
| 01-701-1023 | End of Treatment | 99 | 99 | 0 | 114.92 | 99 |
| 01-701-1023 | End of Treatment | 99 | 99.8 | 0 | . | . |
| 01-701-1023 | End of Treatment | 99 | 100.2 | 0 | . | . |
| 01-701-1023 | End of Treatment | 99 | 100.6 | 0 | . | . |
| 01-701-1023 | End of Treatment | 99 | 101 | 0 | . | . |

**Output 3: Actual Data Merged with New Visits**

Now the data is in the format that is needed for Proc EXPAND.  Program 3 illustrates how to interpolate the results between the visits.

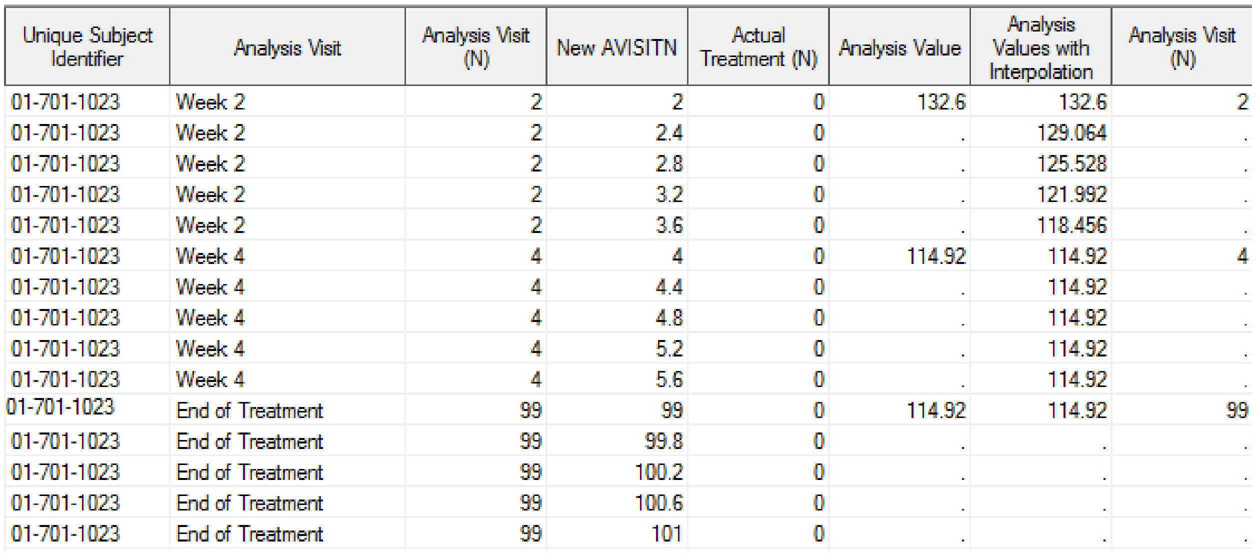
```
  proc expand data=data_for_proc_expand2 out=LinInterp_scatter;
    by usubjid param sex trtan;
    convert aval=linear / method=join;
    id id;
  run;
```

**Program 3: Proc EXPAND Code**

In the Proc EXPAND code, the OUT option creates the data set with the interpolation. The BY statement indicates that the interpolation should be done separately for each combination of Subject, Parameter, Sex and Treatment. CONVERT AVAL=LINEAR statement creates the new variable LINEAR from the existing variable AVAL, and the new variable LINEAR has the interpolated values. The METHOD=JOIN option specifies that linear interpolation will be used. The other options are: SPLINE, STEP, AGGREGATE, and NONE, and the default option is METHOD=SPLINE. The EXPAND procedure is currently unavailable on the SAS UNIVERSITY EDITION.

Output 4, shows the Interpolated results which can be seen in the **Analysis Values with Interpolation** column. Thus, instead of the animation between Visit 2 and Visit 4 dropping dramatically from 132.6 to 114.92 in 1 frame, the values will have a smoother decline from 132.6 to 129.06, to 125.53, to 121.92, to 118.46 and finally to 114.92.

However, this is not the final data set used to create the animations.  As mentioned previously, it is useful to have 4 visits after visit 99 to stop the animation from ending abruptly. Without these additional “visits”, when looping is being used, the animation will only be on visit 99 for a split second. To avoid this abrupt ending, we arbitrarily choose visit values of 99.8, 100.2, 100.6 and 101.  The values after the last visit could be any value as long as they occur after the final visit. In Output 4, the visits 99.8, 100.2, 100.6 and 101 have not been interpolated because in order to interpolate you need to have an actual value before and after the interpolated value. Since there are no actual values after visit 99, therefore, there are no actual values after visit 101. However, there is no need for the values to be interpolated using Proc Expand because to stop the animation from ending abruptly all that is needed is to set the measurements after visit 99 to be equal to visit 99.  Thus for this example, after Proc EXPAND is completed, an additional data step is needed so that all the values after visit 99 can be set to equal 114.92 (i.e., the value at visit 99).



| Unique Subject Identifier | Analysis Visit | Analysis Visit (N) | New AVISITN | Actual Treatment (N) | Analysis Value | Analysis Values with Interpolation | Analysis Visit (N) |
|---|---|---|---|---|---|---|---|
| 01-701-1023 | Week 2 | 2 | 2 | 0 | 132.6 | 132.6 | 2 |
| 01-701-1023 | Week 2 | 2 | 2.4 | 0 | . | 129.064 | . |
| 01-701-1023 | Week 2 | 2 | 2.8 | 0 | . | 125.528 | . |
| 01-701-1023 | Week 2 | 2 | 3.2 | 0 | . | 121.992 | . |
| 01-701-1023 | Week 2 | 2 | 3.6 | 0 | . | 118.456 | . |
| 01-701-1023 | Week 4 | 4 | 4 | 0 | 114.92 | 114.92 | 4 |
| 01-701-1023 | Week 4 | 4 | 4.4 | 0 | . | 114.92 | . |
| 01-701-1023 | Week 4 | 4 | 4.8 | 0 | . | 114.92 | . |
| 01-701-1023 | Week 4 | 4 | 5.2 | 0 | . | 114.92 | . |
| 01-701-1023 | Week 4 | 4 | 5.6 | 0 | . | 114.92 | . |
| 01-701-1023 | End of Treatment | 99 | 99 | 0 | 114.92 | 114.92 | 99 |
| 01-701-1023 | End of Treatment | 99 | 99.8 | 0 | . | . | . |
| 01-701-1023 | End of Treatment | 99 | 100.2 | 0 | . | . | . |
| 01-701-1023 | End of Treatment | 99 | 100.6 | 0 | . | . | . |
| 01-701-1023 | End of Treatment | 99 | 101 | 0 | . | . | . |

**Output 4: Interpolated Results**

## SERIES PLOT OVER TIME BY TREATMENT

Series plots are great when you want to see the change of data over time. But what if you want to compare these changes over time across the different treatments. You could incorporate all the treatments on the same graph, but this can become difficult to read especially when the data starts to overlay. If you separate the series plot for each treatment onto separate pages, then you would need to page through each page to see if there is a difference and sometimes it may be hard to see those subtle differences. However, with the use of animation, it is easier to see where a treatment may have a significant change compared to another treatment. Program 4 illustrates how to build the template so that Figure 4 can be created. Figure 4 illustrates the series plot for change from baseline for both diastolic and systolic blood pressure (BP) over time for each treatment group.

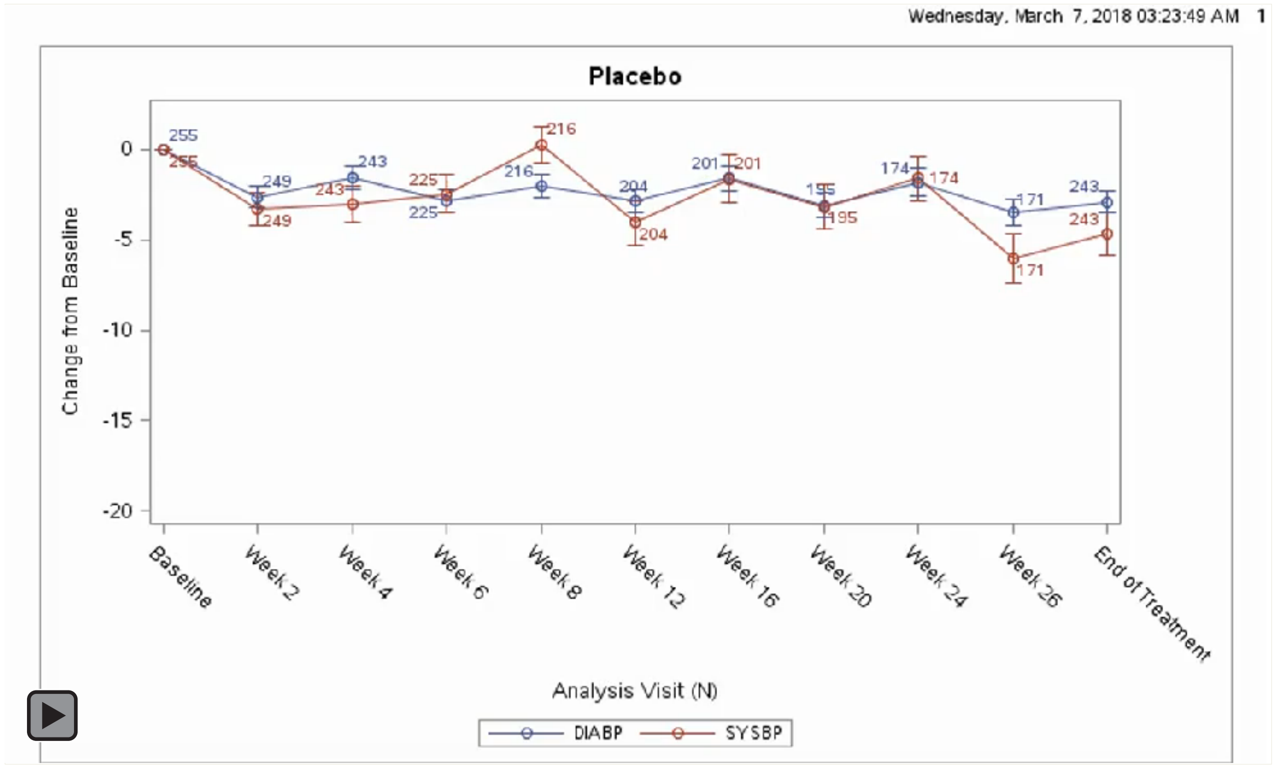


**Figure 4: Animated Series Plot of Change from Baseline by Treatment**

```
/* define template used to produce change from baseline over time */
proc template;
  define statgraph series_chg;
    dynamic _byval_;
      begingraph;
         entrytitle halign = center _byval_;
         layout overlay / xaxisopts = (type = discrete
                                 discreteopts = (tickvaluelist =
                                                ('0' '2' '4' '6'
                                                 '8' '12' '16'
                                                 '20' '24'                1
                                                 '26' '99')))
                          yaxisopts = (linearopts = (viewmin = -20
                                                     viewmax = 2));

          /* produce the series plot of the mean values */
```

```
           seriesplot x = AVISITN y = mean / display = all
                                             group = PARAMCD
                                             datalabel = n
                                             name = "prm";

           /* produce the one standard error from mean whiskers */
           scatterplot x = AVISITN y = mean / group = PARAMCD
                                              yerrorupper = eval(mean+se)
                                              yerrorlower = eval(mean-se);

           discretelegend "prm" / across = 2 location = outside;
         endlayout;
     endgraph;
  end;
run;
```

```
/* retrieve all the blood pressure data for each analysis visit */
proc sort data = source.advs
           out = advs;
   where not missing(CHG) and PARAMCD ? 'BP' and not missing(AVISIT);
   by TRTAN AVISITN AVISIT PARAMCD;
run;

/* calculated change from baseline for each treatment */
proc means data = advs noprint;
    var CHG;
    by TRTAN AVISITN AVISIT PARAMCD;
    output out = sumstats n = n mean = mean stderr = se;
run;
```
2

```
options nobyline;
ods pdf;
options papersize = ('8 in', '4.8 in') printerpath = gif animation = start
animduration = 1.0 animloop = yes noanimoverlay;
ods printer file = "&outpath/&pgmname..gif";

ods graphics / width = 8in height = 4.8in imagefmt = gif;

proc sgrender data = sumstats template = series_chg;
  by TRTAN;
  format trtan trtfmt. AVISITN visfmt.;
run;

options printerpath = gif animation = stop;
ods printer close;
```
3

**Program 4: Animated Series Plot of Change from Baseline by Treatment**

1 As with the other examples, the template will have the standard structure. The axes options are adjusted to allow for a discrete axis. Note that within this design, both a series plot and a scatterplot are needed. The series plot will show the mean change over time. The scatterplot is used to draw the one standard deviation from the mean. In addition, we added the discretelegend so that we can tell the difference between the change from baseline for diastolic BP and systolic BP.

2 The ADVS data set from the CDISC SDTM/ADaM Pilot Project is subsetted for BP data that was captured at an analysis visit (i.e., visits that did not fall within the analysis visit window are excluded). Since the

graph is to display mean change from baseline over time, the means for each treatment, analysis visit, and parameter combination need to be calculated.

To produce the desired graph the subsetted data needs to be associated with the new template (series_chg). In addition, formats were applied so that the treatment labels and the analysis visits labels
3 would be displayed instead of their numeric counterparts. You may also notice that the duration was set to 1.0 because we wanted the images to appear just a bit longer on the screen before the next treatment graph appeared.

## ADVERSE EVENTS OVER TIME BY BASELINE LABORATORY RESULTS

Looking at a scatterplot of cumulative adverse events (AEs) for each subject over time by baseline laboratory results is a great way to assess the AE counts and visualize the relationship between the AE counts and the baseline laboratory results, as seen in Figure 5. This plot is often referred to as a Grass Plot because of the way it grows. In Figure 5, it is apparent that the baseline Creatinine results do not affect the treatment emergent AE (TEAE) counts, and you can also see that there are more TEAE counts in the High Dose group. It is also quite noticeable that the longer that patients are in the study the more TEAEs they have and the rates of the TEAEs are fairly constant over time for each treatment.

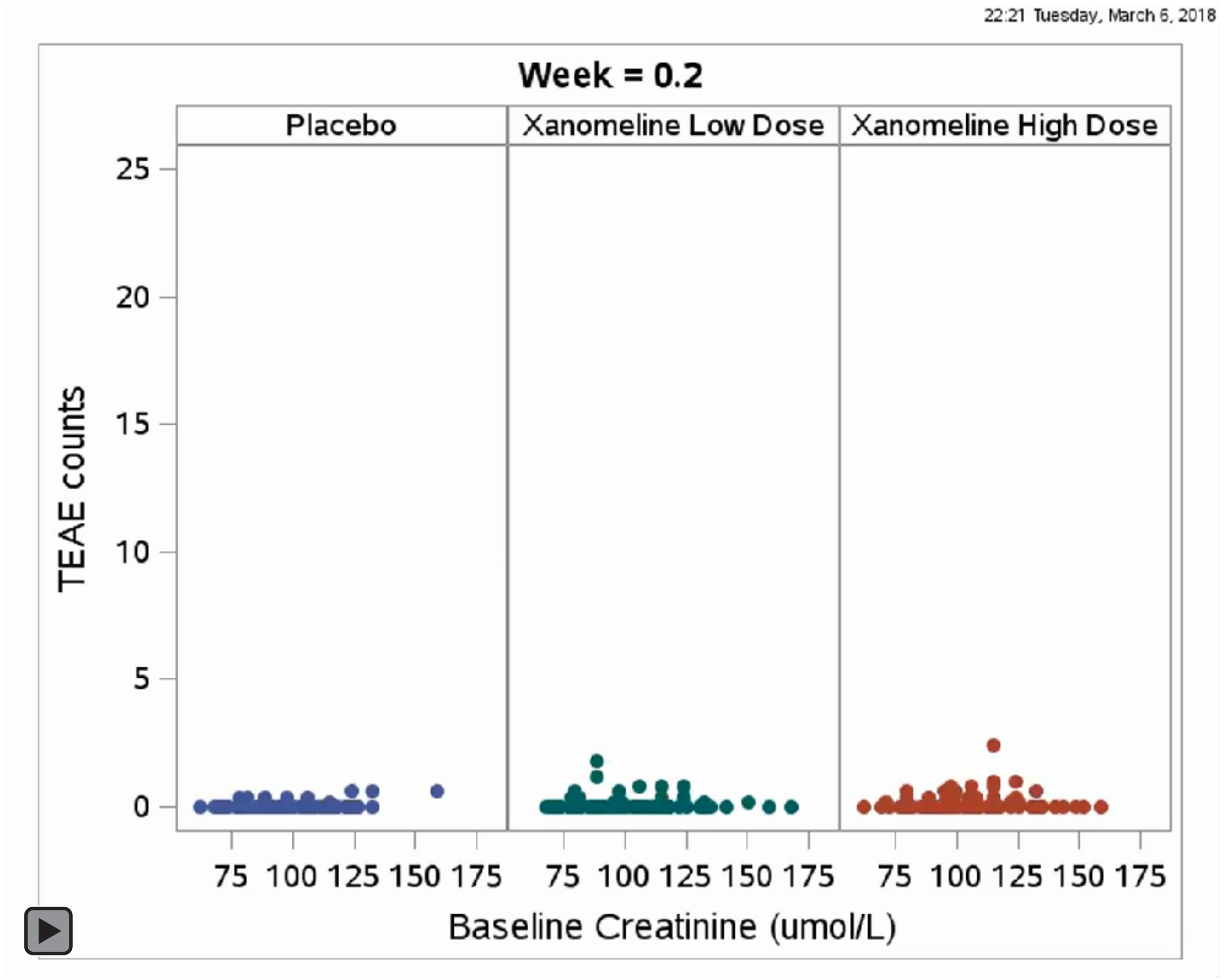
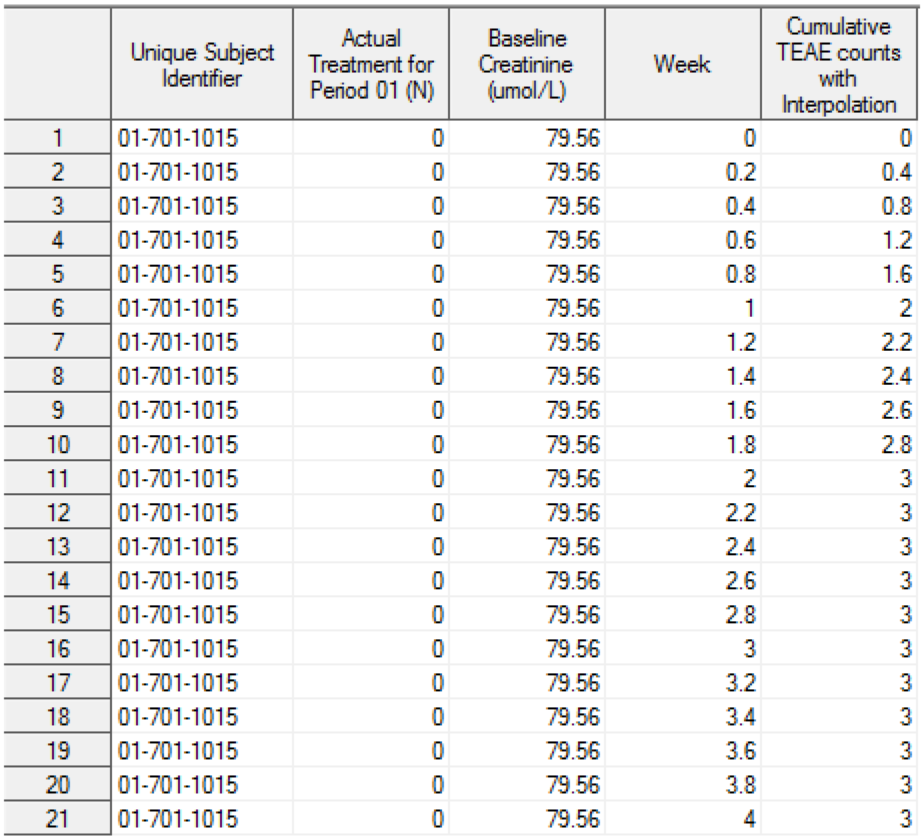


**Figure 5: Treatment Emergent Adverse Events (TEAEs) Over Time by Creatinine at Baseline**

The following steps were used to get the data set in the right format for plotting the animation.

- The ADAE data set from the CDISC SDTM/ADaM Pilot Project was filtered to only contain the TEAEs.
- The number of TEAEs were calculated at each week, up to the final week of Week 28 for each subject, and the cumulative number of TEAEs for each subject were also calculated.
- The baseline Creatinine results were merged onto the data with the TEAE cumulative counts.
- Similar to the data set used for Figure 3, between each visit, 4 visits were imputed as preparation for Proc EXPAND to do the interpolation on the cumulative counts, and hence help to produce a smoother animation.
- Proc EXPAND was used to interpolate the cumulative counts, and a few manipulations were done afterwards to ensure that the data set was in the right order, and that the format of the data set will allow the ending of the animation to finish gradually.

Output 5, is a preview of the data set used in Figure 5.



| | Unique Subject Identifier | Actual Treatment for Period 01 (N) | Baseline Creatinine (umol/L) | Week | Cumulative TEAE counts with Interpolation |
|---|---|---|---|---|---|
| 1 | 01-701-1015 | 0 | 79.56 | 0 | 0 |
| 2 | 01-701-1015 | 0 | 79.56 | 0.2 | 0.4 |
| 3 | 01-701-1015 | 0 | 79.56 | 0.4 | 0.8 |
| 4 | 01-701-1015 | 0 | 79.56 | 0.6 | 1.2 |
| 5 | 01-701-1015 | 0 | 79.56 | 0.8 | 1.6 |
| 6 | 01-701-1015 | 0 | 79.56 | 1 | 2 |
| 7 | 01-701-1015 | 0 | 79.56 | 1.2 | 2.2 |
| 8 | 01-701-1015 | 0 | 79.56 | 1.4 | 2.4 |
| 9 | 01-701-1015 | 0 | 79.56 | 1.6 | 2.6 |
| 10 | 01-701-1015 | 0 | 79.56 | 1.8 | 2.8 |
| 11 | 01-701-1015 | 0 | 79.56 | 2 | 3 |
| 12 | 01-701-1015 | 0 | 79.56 | 2.2 | 3 |
| 13 | 01-701-1015 | 0 | 79.56 | 2.4 | 3 |
| 14 | 01-701-1015 | 0 | 79.56 | 2.6 | 3 |
| 15 | 01-701-1015 | 0 | 79.56 | 2.8 | 3 |
| 16 | 01-701-1015 | 0 | 79.56 | 3 | 3 |
| 17 | 01-701-1015 | 0 | 79.56 | 3.2 | 3 |
| 18 | 01-701-1015 | 0 | 79.56 | 3.4 | 3 |
| 19 | 01-701-1015 | 0 | 79.56 | 3.6 | 3 |
| 20 | 01-701-1015 | 0 | 79.56 | 3.8 | 3 |
| 21 | 01-701-1015 | 0 | 79.56 | 4 | 3 |

**Output 5: Data set used for TEAE Over Time by Creatinine at Baseline**

```
* Creating template;
proc template;
  define statgraph aedecod_anim_scat;
    dynamic _byval_;
    begingraph / designwidth = 1200px designheight = 960px;
        entrytitle textattrs = (size = 18pt) halign = center "Week = "
          _byval_;

        layout datapanel classvars= (trtan) / headerlabeldisplay=value
           headerlabelattrs=(size = 15pt) columns = 3 rows = 1
          rowaxisopts=(label = "TEAE counts" labelattrs=(size = 18pt)
            tickvalueattrs=(size = 16pt) linearopts=(viewmin=0 viewmax=25
            tickvaluesequence=(start=0 end=25 increment=5)))
          columnaxisopts = (label = "Baseline Creatinine (umol/L)"
            labelattrs=(size = 18pt) tickvalueattrs=(size = 16pt));
          layout prototype;
            scatterplot x = baseline_creatinine y = aval / group = trtan
              markerattrs = (size = 15px symbol=circlefilled);
          endlayout;
        endlayout;
      endgraph;
  end;
run;
```

1

```
* Producing animation;
options nobyline;
goptions reset = all;
options papersize=('8 in', '6.4 in') printerpath=gif animation=start
animduration=0.13 animloop=yes noanimoverlay nonumber;
ods printer file="&outpath\ae_anim1_scat.gif";

ods graphics / reset = all width=8in height=6.4in imagefmt=png;

options nobyline;

proc sgrender data = final_scatterplot template= aedecod_anim_scat;
  by avisitn;
  format trtan trtfmt.;
run;

ods graphics / reset = all;

options printerpath=gif animation=stop;
ods printer close;
```

2

**Program 5: TEAE Over Time by Creatinine at Baseline**

1 This template is slightly different from the other templates and uses the DATAPANEL layout to display the treatments side by side in the figure. This approach is used because the x-axis is now numeric, as it contains the baseline Creatinine values. This program creates a figure with a larger height compared to the previous program and also defines the sizes for a lot of the text values to display chosen font sizes.

2 The animation options in this program have made the height of the graph taller compared to the previous graphs. The ANIMDURATION was set to 0.13 because there are a lot of visits in the data set and this allows the animation to get through all the visits faster. Also the IMAGEFMT=PNG option was used to

produce static PNG images at each week. These static PNG plots are useful when writing reports that will not allow for an animated figure.

## ELECTROCARDIOGRAM (ECG) HEART RATE OVER TIME

We have all seen a heart rate monitor that displays how someone's heart rate is beating over time when that person is hooked up to a machine, but how do we display that data based on our clinical data. You can do a static graph as illustrated in Figure 6 that contains all data which can be hard to see the time of any anomalies since the data is captured in milliseconds. Or, you can break the data up into minute intervals and display each minute individually and then page through each minute for each patient to find where there may be a concern in the heart rate. But, with the use of animation, the display of the ECG heart rate over time in milliseconds can be easily viewed in minute intervals. Program 6 illustrates how to build the template so that Figure 7 can be created. Figure 7 illustrates the series plot for the heart rate over 15 minutes broken into 1-minute intervals.

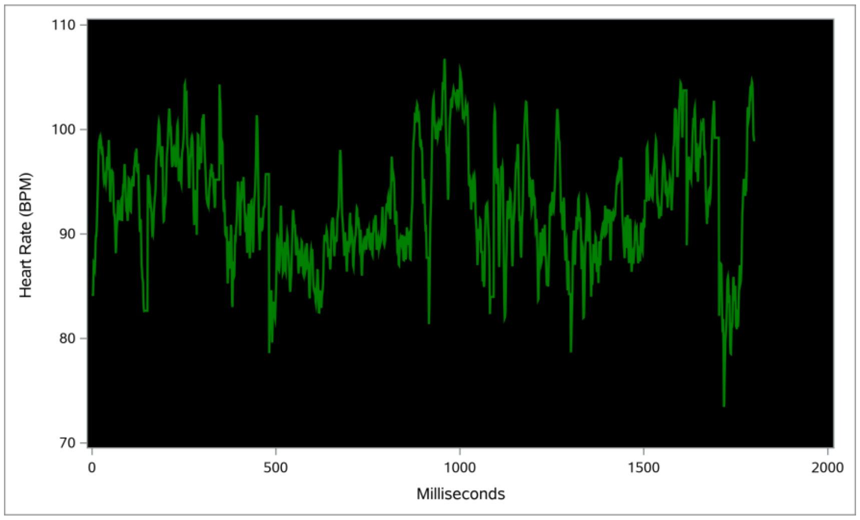
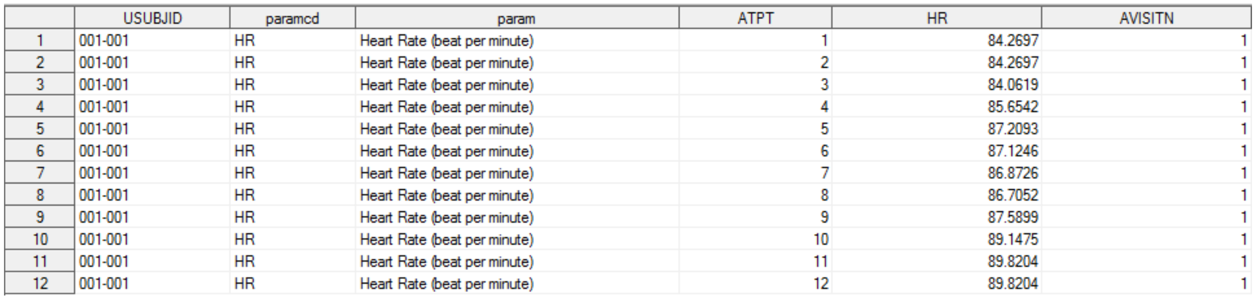


**Figure 6: ECG Heart Rate (BPM) over Time Measured (msec)**

Prior to the creation of the animated ECG plot the data needs to be expanded so that there is a 'drawing' effect of the graph. To create this 'drawing' effect, within each minute which is in the variable AVISITN, we need to create sequential records so that the series plot could be drawn with animation. Before the data can be expanded, it is broken into a set of animation series with each animation series containing 3 seconds of data. Therefore, for each minute interval there are 20 animation series. Once the data is broken into animation series, additional records for each series are generated for the prior animation series so that as the subsequent series are displayed, it is overlayed while the previous series is still visible. Without these additional records, only the records associated with the current series will be displayed. Output 6 is a sample of what the data looks like prior to processing. After expanding the data, we have multiple records for each timepoint.

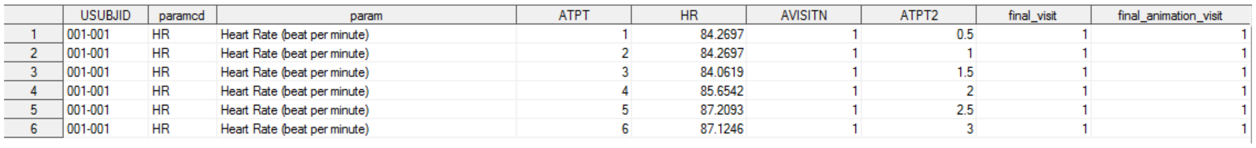
Output 7 illustrates the records for the first animation. These records are the same as the original data with two additional variables added to indicate which animation frame the records are associated with. In order to produce the animation and the 'drawing effect', each animation frame, contains the data of the previous animation frame. Therefore, for the second animation, the records from the first animation (rows 1 - 6 in Output 7) are used along with the rows 7 – 12 in Output 8, which are actually the original rows for the second animation frame. When the records from the previous animation is being used the value in the final_animation_visit changes to reflect the animation frame, as shown in rows 1 - 6 in Output 8. This process is repeated for each subsequent animation frame.

Now that the data has been expanded we can utilize standard GTL statements (Program 6) to generate the ECG Heart Rate over Time Broken Down by Minute as displayed in Figure 7.



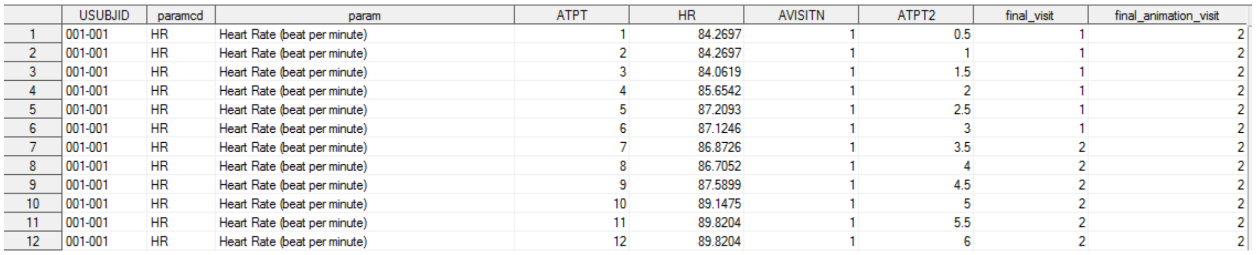
| | USUBJID | paramcd | param | ATPT | HR | AVISITN |
|---|---|---|---|---|---|---|
| 1 | 001-001 | HR | Heart Rate (beat per minute) | 1 | 84.2697 | 1 |
| 2 | 001-001 | HR | Heart Rate (beat per minute) | 2 | 84.2697 | 1 |
| 3 | 001-001 | HR | Heart Rate (beat per minute) | 3 | 84.0619 | 1 |
| 4 | 001-001 | HR | Heart Rate (beat per minute) | 4 | 85.6542 | 1 |
| 5 | 001-001 | HR | Heart Rate (beat per minute) | 5 | 87.2093 | 1 |
| 6 | 001-001 | HR | Heart Rate (beat per minute) | 6 | 87.1246 | 1 |
| 7 | 001-001 | HR | Heart Rate (beat per minute) | 7 | 86.8726 | 1 |
| 8 | 001-001 | HR | Heart Rate (beat per minute) | 8 | 86.7052 | 1 |
| 9 | 001-001 | HR | Heart Rate (beat per minute) | 9 | 87.5899 | 1 |
| 10 | 001-001 | HR | Heart Rate (beat per minute) | 10 | 89.1475 | 1 |
| 11 | 001-001 | HR | Heart Rate (beat per minute) | 11 | 89.8204 | 1 |
| 12 | 001-001 | HR | Heart Rate (beat per minute) | 12 | 89.8204 | 1 |

**Output 6: ECG Heart Rate (BMP) Data (source: http://ecg.mit.edu/time-series/ - series 1)**



| | USUBJID | paramcd | param | ATPT | HR | AVISITN | ATPT2 | final_visit | final_animation_visit |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 001-001 | HR | Heart Rate (beat per minute) | 1 | 84.2697 | 1 | 0.5 | 1 | 1 |
| 2 | 001-001 | HR | Heart Rate (beat per minute) | 2 | 84.2697 | 1 | 1 | 1 | 1 |
| 3 | 001-001 | HR | Heart Rate (beat per minute) | 3 | 84.0619 | 1 | 1.5 | 1 | 1 |
| 4 | 001-001 | HR | Heart Rate (beat per minute) | 4 | 85.6542 | 1 | 2 | 1 | 1 |
| 5 | 001-001 | HR | Heart Rate (beat per minute) | 5 | 87.2093 | 1 | 2.5 | 1 | 1 |
| 6 | 001-001 | HR | Heart Rate (beat per minute) | 6 | 87.1246 | 1 | 3 | 1 | 1 |

**Output 7: ECG Heart Rate (BMP) with Additional Records for First Animation**



| | USUBJID | paramcd | param | ATPT | HR | AVISITN | ATPT2 | final_visit | final_animation_visit |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 001-001 | HR | Heart Rate (beat per minute) | 1 | 84.2697 | 1 | 0.5 | 1 | 2 |
| 2 | 001-001 | HR | Heart Rate (beat per minute) | 2 | 84.2697 | 1 | 1 | 1 | 2 |
| 3 | 001-001 | HR | Heart Rate (beat per minute) | 3 | 84.0619 | 1 | 1.5 | 1 | 2 |
| 4 | 001-001 | HR | Heart Rate (beat per minute) | 4 | 85.6542 | 1 | 2 | 1 | 2 |
| 5 | 001-001 | HR | Heart Rate (beat per minute) | 5 | 87.2093 | 1 | 2.5 | 1 | 2 |
| 6 | 001-001 | HR | Heart Rate (beat per minute) | 6 | 87.1246 | 1 | 3 | 1 | 2 |
| 7 | 001-001 | HR | Heart Rate (beat per minute) | 7 | 86.8726 | 1 | 3.5 | 2 | 2 |
| 8 | 001-001 | HR | Heart Rate (beat per minute) | 8 | 86.7052 | 1 | 4 | 2 | 2 |
| 9 | 001-001 | HR | Heart Rate (beat per minute) | 9 | 87.5899 | 1 | 4.5 | 2 | 2 |
| 10 | 001-001 | HR | Heart Rate (beat per minute) | 10 | 89.1475 | 1 | 5 | 2 | 2 |
| 11 | 001-001 | HR | Heart Rate (beat per minute) | 11 | 89.8204 | 1 | 5.5 | 2 | 2 |
| 12 | 001-001 | HR | Heart Rate (beat per minute) | 12 | 89.8204 | 1 | 6 | 2 | 2 |

**Output 8: ECG Heart Rate (BMP) with Additional Records for Second Animation**

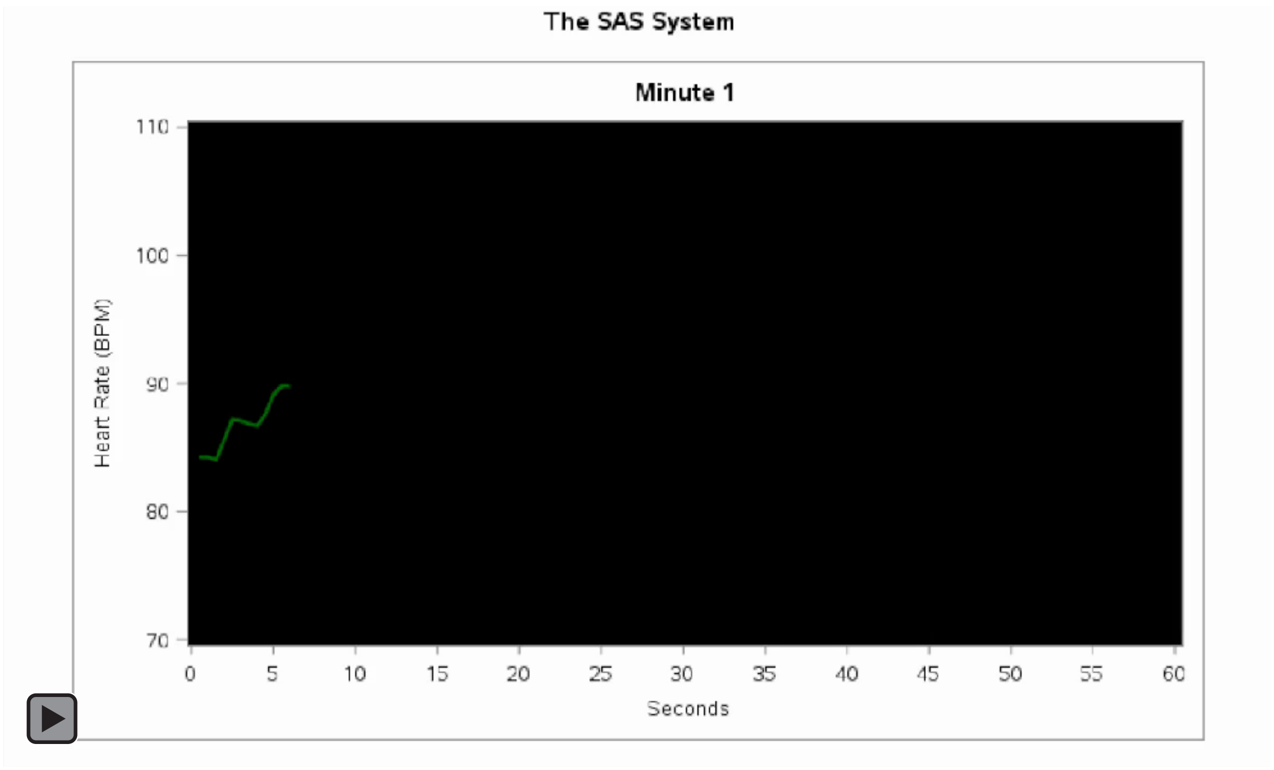


**Figure 7: Animated ECG Heart Rate (BPM) Over Time Broken Down by Minute**

```
proc template;
  define statgraph ecg_template;
      dynamic _byval_;
      begingraph;
        entrytitle "Minute " _byval_;
        layout overlay / wallcolor=black
                         yaxisopts=(label= "Heart Rate (BPM)"
                              linearopts=(viewmin=70 viewmax=110
                                 tickvaluesequence=(start=70 end=110
                                                    increment=10)))
                         xaxisopts=(label= "Seconds"
                              linearopts=(viewmin=0 viewmax=60
                                 tickvaluesequence=(start=0 end=60
                                                    increment=5)));
          seriesplot x = atpt2 y = hr / group=usubjid
                                        lineattrs = (color=green
                                                     thickness=2);
        endlayout;
      endgraph;
  end;
run;

proc sgrender data = ecg_expanded template = ecg_template;
  by avisitn final_animation_visit;
run;
```
1

**Program 6: Animated ECG Heart Rate (BPM) Over Time Broken Down by Minute**

1 Similar to the other examples, the template will have the standard structure. The x-axis will display the time in seconds (ATPT2) with each set of the timepoints displayed by minute (AVISITN) and animation series (FINAL_ANIMATION_VISIT). There was nothing new in this example. Much of the work was done in prepping the data set so that it can be utilized to produce the heart rate over time broken down my minute.

## CONCLUSION

Animations can be done quite easily with SAS 9.4. All that is needed is to wrap the appropriate OPTIONS and the ODS PRINTER statements around the plots.

Smoother animations can be created by using Proc EXPAND to interpolate results in between the visits, and then producing the animation on the data set which contains interpolated results.

## REFERENCES

CDISC. (2013). CDISC. Retrieved from SDTM/ADaM Pilot Project: http://www.cdisc.org/system/files/members/article/application/zip/updated_pilot_submission_package.zip

Harris, K. (2016). Animate Your Safety Data. *PharmaSUG China.* Beijing: PharmaSUG.

Harris, K. (2017). Hands-On Graph Template Language (GTL): Part A. *SAS Global Forum.* Denver: SAS Global Forum.

Harris, K., & Watson, R. (2018). Great Time to Learn GTL. *PharmaSUG.* Seattle: PharmaSUG.

Matange, S. (n.d.). *Animation using SGPLOT*. Retrieved from Graphically Speaking: https://blogs.sas.com/content/graphicallyspeaking/2013/05/23/animation-using-sgplot/

McCarthy, A. (2017). Using Animated Graphics to Show PKPD Relationships in SAS 9.4. *PharmaSUG* (pp. 4-6). Baltimore: PharmaSUG.

## ACKNOWLEDGEMENTS

We will like to acknowledge Tao Shu, Lixiang Larry Liu, and Jianfei Jiang for introducing us to the Cumulative TEAE plot.

## CONTACT INFORMATION

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

Kriss Harris
SAS Specialists Limited
italjet125@yahoo.com
http://www.krissharris.co.uk

Richann Watson
DataRich Consulting
richann.watson@datarichconsulting.com
http://www.datarichconsulting.com