Customizing and Automating your Graphs using the SAS SG Procedures

Jesse A. Canchola, Roche Molecular Systems, Pleasanton, CA USA

Shiva Narra, ex-Roche Molecular Systems, Pleasanton, CA USA

Disclaimer

The views and opinions expressed in this presentation are solely those of the authors and do not necessarily reflect the official policy or position of

Roche Molecular Systems, Inc.

Today at a Glance

- The SAS/Graph Journey
- Introduction to SG Procedures
- Creating One or a Few Graphs

Today at a Glance

- The SAS/Graph Journey
- Introduction to SG Procedures
- Creating One or a Few Graphs

Side Dishes (Asides)

Today at a Glance

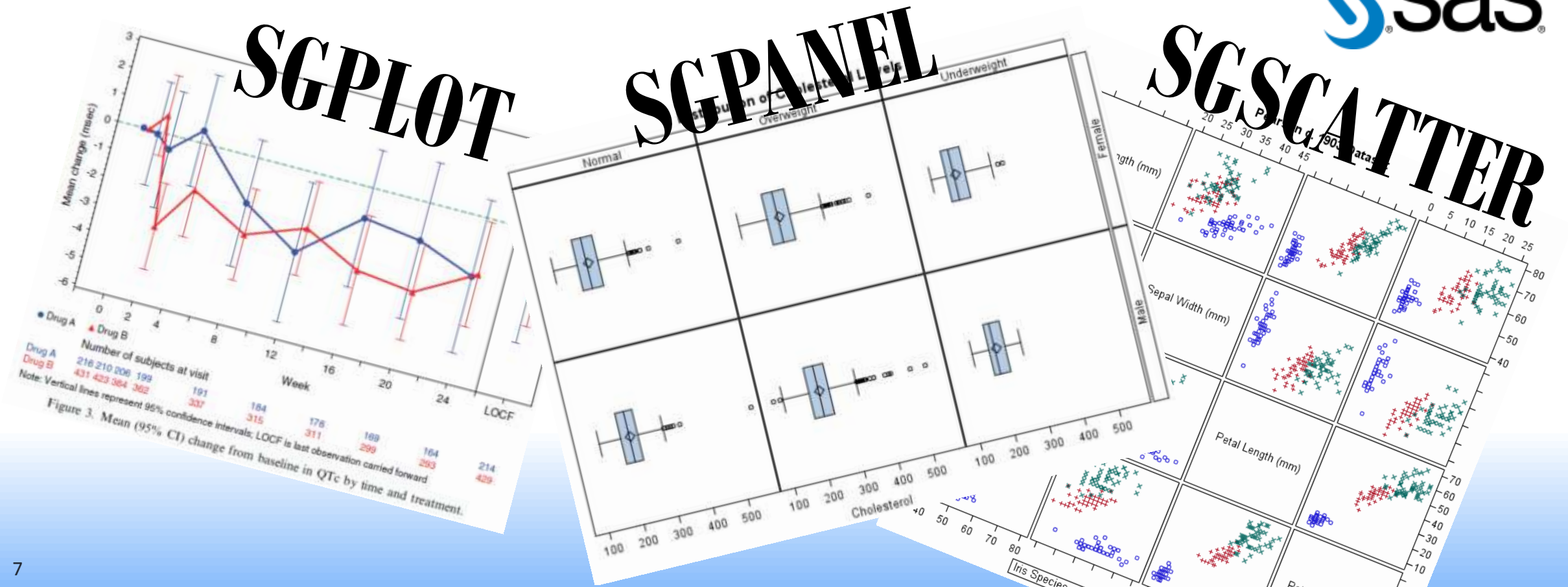
- The SAS/Graph Journey
- Introduction to SG Procedures
- Creating One or a Few Graphs
- Automation
- Customization
- Macrotization of your Customized Automation
- Some Conclusions
- Contact Information
- References

The SAS/Graph Journey

The SAS/Graph Journey

Introduction

- The SAS SG procedures arguably are the best tools in SAS for creating and customizing your graphs.



The SAS/Graph Journey

Introduction

- In the past, automating and customizing the SG procedures may have been complex and tedious.

Bitmap Graphics

Post-Production Graphics Editing

ADOBE PHOTOSHOP

Vector Graphics

ADOBE ILLUSTRATOR

INKSCAPE Draw Freely.

GRAVIT DESIGNER

Graphic Types

SAS/GRAPH®

Annotate

The SAS/Graph Journey

Introduction

- However, many new options have been implemented for the SG procedures beginning with SAS Version 9.4 that promise to make customization and automation even easier.

The SAS/Graph Journey

Introduction

- We will take you through examples that show the improvements and provide you with a road map for successfully leveraging the power of the SAS SG procedures.

Introduction to SG Procedures

Introduction to SG Procedures

SG Procedures

- Whether you are
  - generating graphics for a one-off or for
  - generating similar but repeated figures (e.g., one for each subject)

Introduction to SG Procedures

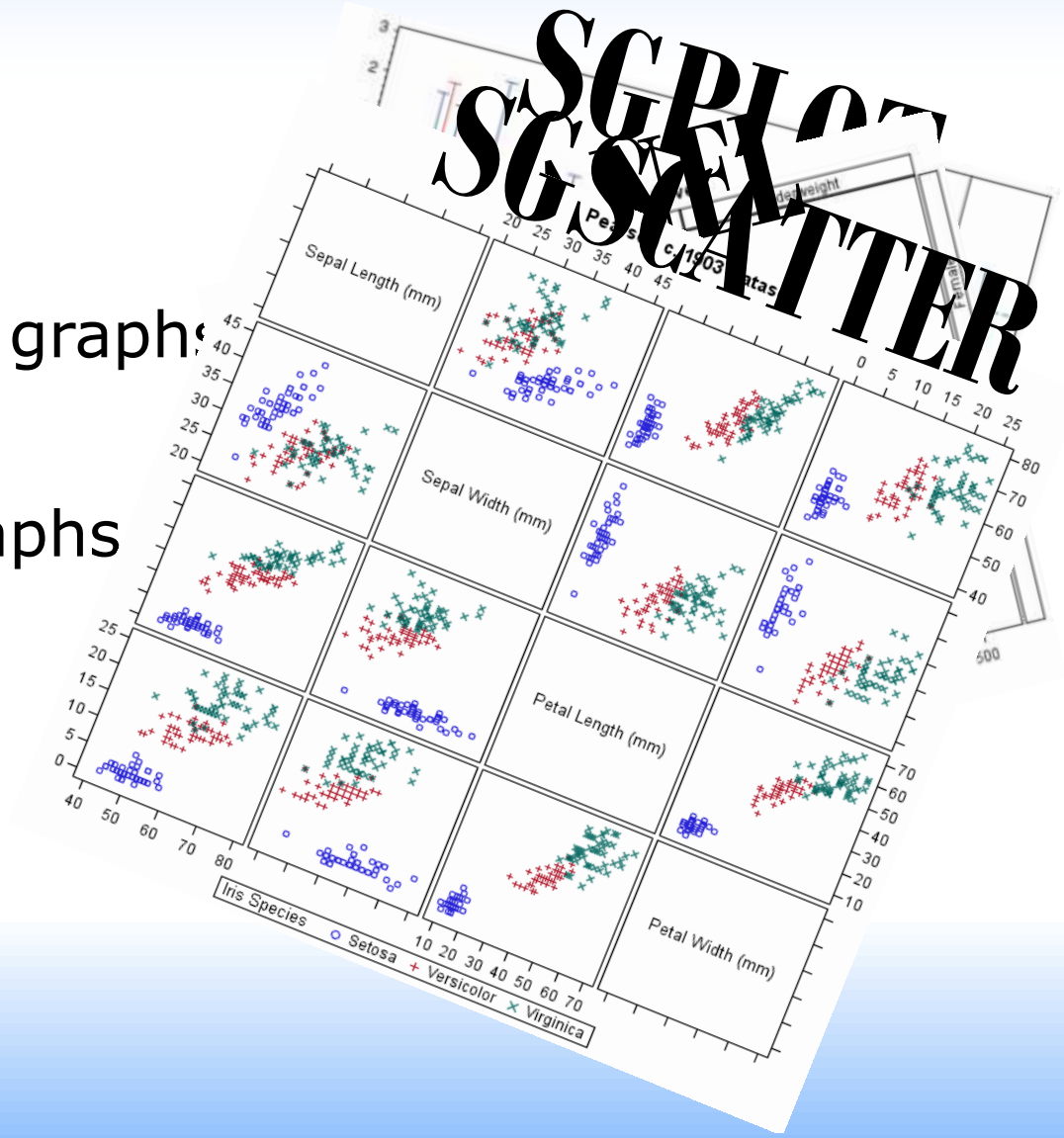
SG Procedures

- Whether you are
  - generating graphics for a one-off or for
  - generating similar but repeated figures (e.g., one for each subject)
- The SAS Institute (Cary, NC) provides for a rich and flexible toolbox with their SAS SG procedures.

Introduction to SG Procedures

SG Procedures

- The SG procedures are as follows:
  - **SGPLOT**: Produces a single panel of graphs
  - **SGPANEL**: Produces multi-panel graphs with common axes
  - **SGSCATTER**: Produces multi-panel graphs with a different axes capability

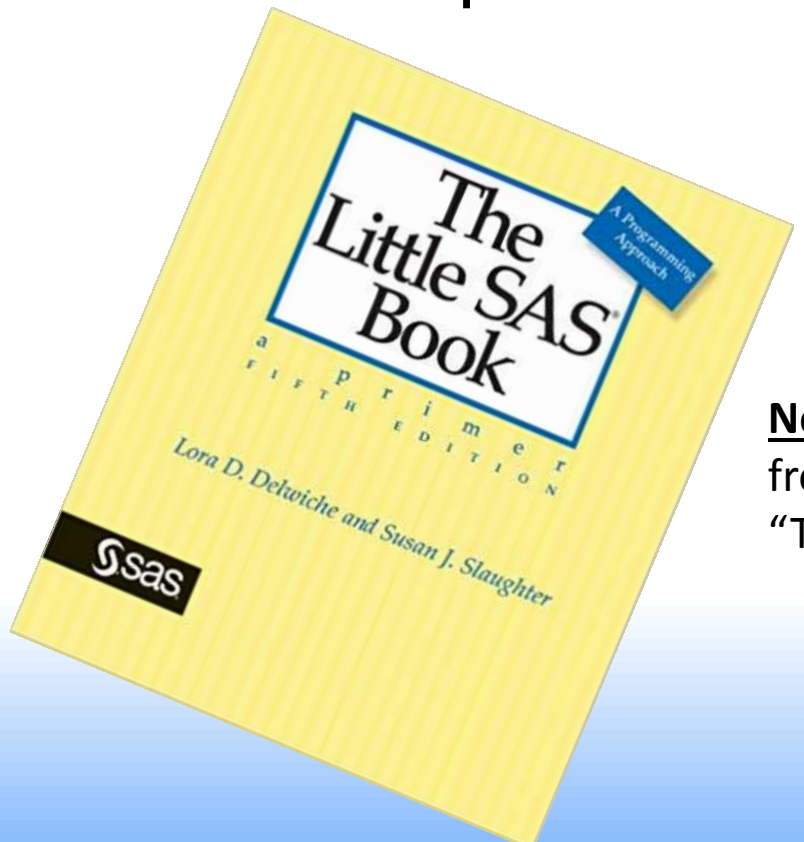


Introduction to SG Procedures

SG Procedures

- Lora Delwiche and Susan Slaughter (of *The Little SAS Book* fame) in their 2012 proceedings paper give a comprehensive overview of the SG procedures.

L-R: Lora Delwiche, Jesse Canchola, Susan Slaughter.
SAS Global Forum 2018, Denver, Colorado. Mile High Stadium.

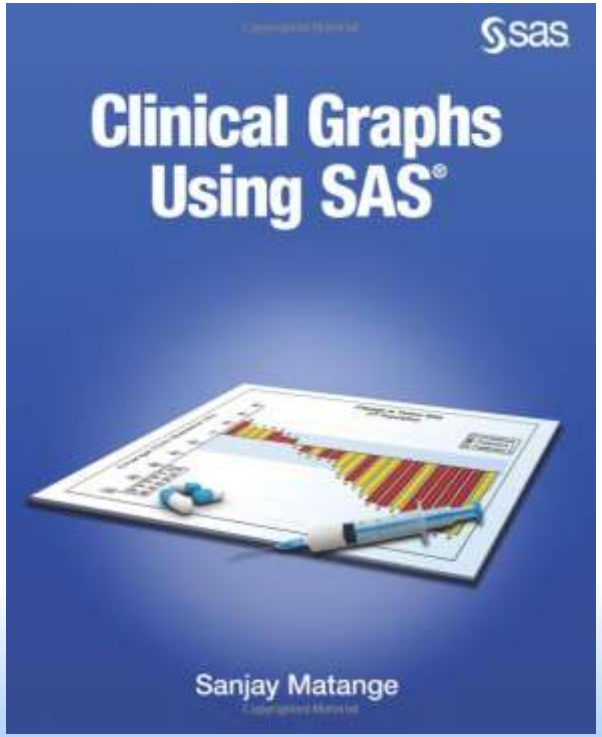


Note: the paper noted above is from a proceedings, not from "The Little SAS Book".

Introduction to SG Procedures

SG Procedures

- Sangjay Matange (2016) wrote a nice book called, "Clinical Graphs Using SAS®" that, somewhat comprehensively, covers the SG procedures.

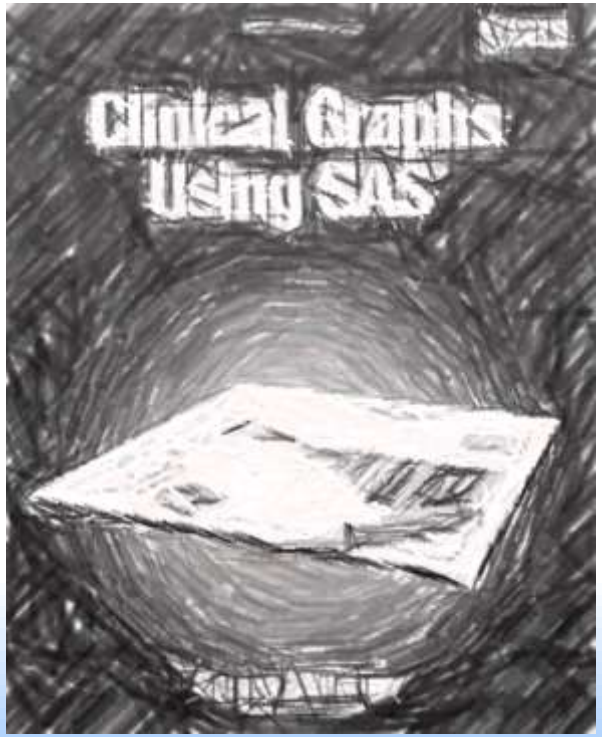


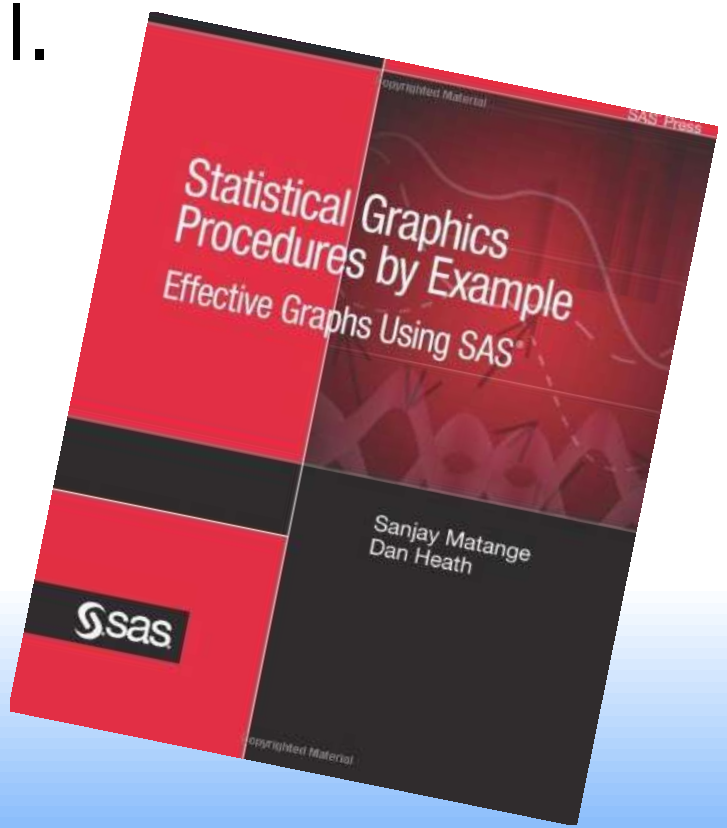
Introduction to SG Procedures

SG Procedures

- Matange (2016) wrote a nice book called, "Clinical Graphs Using SAS®" that, somewhat comprehensively, covers the SG procedures.

… and one in 2011 (with Dan Heath) that is still quite useful.





Introduction to SG Procedures

SG Procedures

- Delwiche and Slaughter (2012) give a comprehensive overview of the SG procedures.

- The current paper augments the SG procedures capabilities with the advanced topics of automation and customization using:
  - SAS Macro language,
  - SAS IML (Interactive Matrix Language)
  - SAS SQL
  - SAS Annotate facility (to a limited extent).

SG Procedures

- We begin with a short review of the current capabilities and

- continue with automation of your graphs and end with how to successfully deal with repetitive figures.

Creating One or A Few Graphs

Creating One or a Few Graphs

Standard SG

- In the past, SG stood for "Statistical Graphics".
- However, advancements and enhancements over the years have made the SAS SG procedures much more than that!
- The SG acronym may now be more correctly represented as
  - "Splendid Graphics" or simply,
  - "SAS Graphics".

Creating One or a Few Graphs

Standard SG

- Let us look at an example of one SG procedure with all patients/subjects together.

Creating One or a Few Graphs

Standard SG

- For this we use the simulated patient Hepatitis C Therapy Response data (HepCTR; Appendix A of paper).

- Briefly, the HepCTR study:
  - examines the patient Hepatitis C viral load response
  - with respect to treatment
  - over a period of 30 weeks.

Creating One or a Few Graphs

Standard SG

- For this we use the simulated patient Hepatitis C Therapy Response data (HepCTR; Appendix A of paper).

- Briefly, the HepCTR study:
  - examines the patient Hepatitis C viral load response
  - with respect to treatment
  - over a period of 30 weeks.

- Basically, how much virus does a patient have in their body over the course of treatment.

Creating One or a Few Graphs

Standard SG

- Two sets of data are simulated:
  - the "Responders" (i.e., patients responding to therapy)
    - whose viral load reduces to undetectable levels over time
  - "Relapsers" (i.e., patients not responding to therapy)
    - whose viral load comes back up (i.e., is in an upswing) at a point in time during the treatment regimen.

Creating One or a Few Graphs

Standard SG

- The user should proceed using the following steps for generating the Responder data set and plotting the results.

Creating One or a Few Graphs

Standard SG: Step 1

Step 1: Run the Appendix A simulated data code to use the HepCTR Responder data example.

Creating One or a Few Graphs

Standard SG: Step 1

Simulating Responder patients.

100 Patients
13 Timepoints
Up to 35 Weeks

```
* RESPONDERS ;
* generate 100 patients with 13 time points each ;
%LET SampleID   = 100 ; * generate 100 subjects ;
%LET NumSamples = 13  ; * with 13 time points each ;
data Xvalues (rename = (i = SampleID j = Time)) ;
   seed = 81638 ;
   do i = 1 to &SampleID ;
      do j = 1 to &NumSamples ;
         X = 35 * ranuni(seed) ; * generate numbers from 0 to 35 as time variable in
                                   weeks ;
            output ;
         end ;
   end ;
run ;
```

Creating One or a Few Graphs

Standard SG: Step 1

Program*

Save • Run • Stop | Selected Server: SASApp (Connected) • | Analyze Program • | Export • Send To • Create • | Changes Commit History | Properties

```
* RESPONDERS ;
* generate 100 patients with 13 time points each ;
%LET SampleID   = 100 ; * generate 100 subjects ;
%LET NumSamples = 13  ; * with 13 time points each ;
data Xvalues (rename = (i = SampleID j = Time)) ;
   seed = 81638 ;
   do i = 1 to &SampleID ;
      do j = 1 to &NumSamples ;
         X = 35 * ranuni(seed) ; * generate numbers from 0 to 35 as time
                                   variable in weeks ;
         output ;
      end ;
   end ;
run ;

* simulating responder patients ;
data Responders_TMP0 (drop=Seed S B0 B1 B2 B3) ;
   set Xvalues ;
   seed = 183609 ;

   b0 = 6.6 ; b1 = -4.5E-01 ; b2 = 1.2 ; b3 = 2.01E-01 ;
   * rational model ;
   Y = ( b0 + b1 * X ) / ( 1 + b2 * X + b3 * X**2 ) ;

   * jittering X values for each subject ;
   s = 1 ; * scale factor ;
   jY = Y + s * ranuni(seed) - 0.001 ;
   if jY < 0 then jY = 0 ;

   Test1 = Y  ;
   Test2 = jY ;
run ;
data Responders_Test1 (rename=(Test1=log10Observed)) ;
   set Responders_TMP0 (drop=Y jY Test2) ;
   Test = "Test1" ;
run ;
data Responders_Test2 (rename=(Test2=log10Observed)) ;
   set Responders_TMP0 (drop=Y jY Test1) ;
   Test = "Test2" ;
run ;
data Responders_TMP ;
set Responders_Test1 Responders_Test2 ;
run ;

data Responders_TMP1 ;
   set Responders_TMP ( rename=(X=Weeks) ) ;
```

Generate

Appendix A

Data

Creating One or a Few Graphs

Standard SG

```
* simulating responder patients ;
data Responders_TMP0 (drop=Seed S B0 B1 B2 B3) ;
  set Xvalues ;
  seed = 183609 ;

  b0 = 6.6 ; b1 = -4.5E-01 ; b2 = 1.2 ; b3 = 2.01E-01 ;
  * rational model ;
  Y = ( b0 + b1 * X ) / ( 1 + b2 * X + b3 * X**2 ) ;

  * jittering X values for each subject ;
  s = 1 ; * scale factor ;
  jY = Y + s * ranuni(seed) - 0.001 ;
  if jY < 0 then jY = 0 ;

  Test1 = Y  ;
  Test2 = jY ;
run ;
```

Simulating Responder patients (cont.)

Random Data are run through Rational Model

X values are "jittered" to create linear dependency with variation.

Creating One or a Few Graphs

Standard SG

Simulating Responder patients (cont.)

```
data Responders_Test1 (rename=(Test1=log10Observed))
  set Responders_TMP0 (drop=Y jY Test2) ;
  Test = "Test1" ;
run ;
data Responders_Test2 (rename=(Test2=log10Observed)) ;
  set Responders_TMP0 (drop=Y jY Test1) ;
  Test = "Test2" ;
run ;
data Responders_TMP ;
set Responders_Test1 Responders_Test2 ;
run ;
```

Creating long data set (i.e., stacked)

Creating One or a Few Graphs

Program* | Log | Output Data

Save | Run | Stop | Selected Server: SASApp (Connected) | Analyze Program | Export | Send To | Create | Changes | Commit | History | Properties

```
* RESPONDERS ;
* generate 100 patients with 13 time points each ;
%LET SampleID   = 100 ; * generate 100 subjects ;
%LET NumSamples = 13  ; * with 13 time points each ;
data Xvalues (rename = (i = SampleID j = Time)) ;
   seed = 81638 ;
   do i = 1 to &SampleID ;
      do j = 1 to &NumSamples ;
         X = 35 * ranuni(seed) ; * generate numbers from 0 to 35 as time
                                   variable in weeks ;
         output ;
      end ;
   end ;
run ;

* simulating responder patients ;
data Responders_TMP0 (drop=Seed S B0 B1 B2 B3) ;
   set Xvalues ;
   seed = 183609 ;

   b0 = 6.6 ; b1 = -4.5E-01 ; b2 = 1.2 ; b3 = 2.01E-01 ;
   * rational model ;
   Y = ( b0 + b1 * X ) / ( 1 + b2 * X + b3 * X**2 ) ;

   * jittering X values for each subject ;
   s = 1 ; * scale factor ;
   jY = Y + s * ranuni(seed) - 0.001 ;
   if jY < 0 then jY = 0 ;

   Test1 = Y  ;
   Test2 = jY ;
run ;
data Responders_Test1 (rename=(Test1=log10Observed)) ;
   set Responders_TMP0 (drop=Y jY Test2) ;
   Test = "Test1" ;
run ;
data Responders_Test2 (rename=(Test2=log10Observed)) ;
   set Responders_TMP0 (drop=Y jY Test1) ;
   Test = "Test2" ;
run ;
data Responders_TMP ;
set Responders_Test1 Responders_Test2 ;
```

Simulate

Responder

Patients

Log Summary

Errors (0) | Warnings (0) | Notes (3)

Description	Line	Affected Code	Log Line	Program Li...
NOTE: Writing TAGSETS.SASREPORT13(EGSR) Body file: EGSR	27	options(rolap="on")	22	

Creating One or a Few Graphs

Standard SG

Simulating Responder patients (cont.)

```
data Responders_TMP1 ;
  set Responders_TMP ( rename=(X=Weeks) ) ;

  Responder = 1 ;

   * Select time points at Weeks (+-3 days) ;
        if 0.0  <  Weeks < 1.0  then TP = "00: Baseline" ;
   else if 1.0  <= Weeks < 1.4  then TP = "01: Week 1"   ;
   else if 1.6  <= Weeks < 2.4  then TP = "02: Week 2"   ;
   else if 3.6  <= Weeks < 4.4  then TP = "03: Week 4"   ;
   else if 5.6  <= Weeks < 6.4  then TP = "04: Week 5"   ;
   else if 11.6 <= Weeks < 12.4 then TP = "05: Week 12"  ;
   else if 23.6 <= Weeks < 25.4 then TP = "06: Week 24"  ;
   else if 29.6 <= Weeks < 30.4 then TP = "07: Week 30"  ;
```

Select time points (TP) within 3 days on either side.

...

Creating One or a Few Graphs

Standard SG

Simulating Responder patients (cont.)

Trim the data down to a manageable pedagogical size.

```
...

* keep only the data for specific Time periods ;
   if substr(TP,1,2) in ("00","01","02","03","04","05","06","07") ;

* indicator variable of just 1s to sum up in Proc SQL below ;
   indicator = 1 ;

   TimeStem = substr(TP,1,2) ;
run ;
```

Creating One or a Few Graphs

Program* | Log | Output Data (4)

Save | Run | Stop | Selected Server: SASApp (Connected) | Analyze Program | Export | Send To | Create | Changes | Commit | History | Properties

```
* RESPONDERS ;
* generate 100 patients with 13 time points each ;
%LET SampleID   = 100 ; * generate 100 subjects ;
%LET NumSamples = 13  ; * with 13 time points each ;
data Xvalues (rename = (i = SampleID j = Time)) ;
   seed = 81638 ;
   do i = 1 to &SampleID ;
      do j = 1 to &NumSamples ;
         X = 35 * ranuni(seed) ; * generate numbers from 0 to 35 as time
                                   variable in weeks ;
         output ;
      end ;
   end ;
run ;

* simulating responder patients ;
data Responders_TMP0 (drop=Seed S B0 B1 B2 B3) ;
  set Xvalues ;
  seed = 183609 ;

  b0 = 6.6 ; b1 = -4.5E-01 ; b2 = 1.2 ; b3 = 2.01E-01 ;
  * rational model ;
  Y = ( b0 + b1 * X ) / ( 1 + b2 * X + b3 * X**2 ) ;

  * jittering X values for each subject ;
  s = 1 ; * scale factor ;
  jY = Y + s * ranuni(seed) - 0.001 ;
  if jY < 0 then jY = 0 ;

  Test1 = Y  ;
  Test2 = jY ;
run ;
data Responders_Test1 (rename=(Test1=log10Observed)) ;
  set Responders_TMP0 (drop=Y jY Test2) ;
  Test = "Test1" ;
run ;
data Responders_Test2 (rename=(Test2=log10Observed)) ;
  set Responders_TMP0 (drop=Y jY Test1) ;
  Test = "Test2" ;
run ;
data Responders_TMP ;
set Responders_Test1 Responders_Test2 ;
```

Log Summary

Errors (0) | Warnings (0) | Notes (14)

Description	Line	Affected Code	Log Line	Program Li...
NOTE: Writing TAGSETS.SASREPORT13(EGSR) Body file: EGSR	27	options(rolap="on")	22	

Definition of Time Point Categories

Creating One or a Few Graphs

Standard SG

Simulating Responder patients (cont.)

Take first date if multiple dates are generated by simulation.

```
* if multiple dates within subject generated by simulation
  then take the first date ;
proc sort data=Responders_TMP1 ; by SampleID TP TimeStem ;
run ;
data Responders_TMP2 ;
  set Responders_TMP1 ;
  by SampleID TP TimeStem ;
  if first.TimeStem ;
run ;
```

Creating One or a Few Graphs

Standard SG

Simulating Responder patients (cont.)

Set up data to keep observation if number of timepoints is 4 or more.

```
* Set up data to keep if number of
time points is 4 or more ;
proc sql ;
  create table Responders_GE4WKS as
  select
      SampleID
       ,sum(indicator) as Total
  from Responders_TMP2
  group by SampleID
  order by SampleID ;
quit ;
```

Sum up indicator variable created earlier and call it Total.

Creating One or a Few Graphs

Standard SG

Simulating Responder patients (cont.)

```
proc sort data=Responders_TMP2    ; by SampleID ; run ;
proc sort data=Responders_GE4WKS ; by SampleID ; run ;
data Responders (drop=indicator timestem total) ;
  merge Responders_TMP2 Responders_GE4WKS ;
  by SampleID ;
  if Total ge 4 ;
run ;
* end data set creation for Responders ;
* END OF SAS CODE ;
```

Merge data sets then take patients with 4 or more time points.

Creating One or a Few Graphs

Program* | Log | Output Data

Save | Run | Stop | Selected Server: SASApp (Connected) | Analyze Program | Export | Send To | Create | Changes | Commit | History | Properties

```
  Test = "Test2" ;
run ;
data Responders_TMP ;
set Responders_Test1 Responders_Test2 ;
run ;

data Responders_TMP1 ;
  set Responders_TMP ( rename=(X=Weeks) ) ;

  Responder = 1 ;

  * Select time points at Weeks (+-3 days) ;
      if 0.0  <  Weeks < 1.0  then TP = "00: Baseline" ;
  else if 1.0  <= Weeks < 1.4  then TP = "01: Week 1"   ;
  else if 1.6  <= Weeks < 2.4  then TP = "02: Week 2"   ;
  else if 3.6  <= Weeks < 4.4  then TP = "03: Week 4"   ;
  else if 5.6  <= Weeks < 6.4  then TP = "04: Week 5"   ;
  else if 11.6 <= Weeks < 12.4 then TP = "05: Week 12"  ;
  else if 23.6 <= Weeks < 25.4 then TP = "06: Week 24"  ;
  else if 29.6 <= Weeks < 30.4 then TP = "07: Week 30"  ;

  * keep only the data for specific Time periods ;
  if substr(TP,1,2) in ("00","01","02","03","04","05","06","07") ;

  * indicator variable of just 1s to sum up in Proc SQL below ;
  indicator = 1 ;

  TimeStem = substr(TP,1,2) ;
run ;

* if multiple dates within subject generated by simulation then take the first date ;
proc sort data=Responders_TMP1 ; by SampleID TP TimeStem ; run ;
data Responders_TMP2 ;
  set Responders_TMP1 ;
  by SampleID TP TimeStem ;
  if first.TimeStem ;
run ;

* keep if number of time points is 4 ;
proc sql ;
  create table Responders_GE4WKS as
  select
     SampleID
```

Take first of multiple time points.

Log Summary

Errors (0) | Warnings (0) | Notes (4)

Description	Line	Affected Code	Log Line	Program Li...
NOTE: Writing TAGSETS.SASREPORT13(EGSR) Body file: EGSR	27	options(rolap="on")	22	

Creating One or a Few Graphs

Standard SG: Step 2

Step 2: **All Patients One Graph**. Run the following code to produce a standard SGPLOT graph:

Creating One or a Few Graphs

Standard SG: SAS Code

Produce Standard SGPLOT

```
ods noproctitle ;
ods RTF file =
            "C:\WUSS2018_Responders_Overall_&sysdate..RTF"
                 style=MonoChromePrinter ;
  * above: sysdate appends current data to your file name and
    style gives white background ;
  title2 "" ;
  footnote1 "Responder Patients: Overall
            (Subjects combined in aggregate)" ;
  * Overall ;
  proc sort data=Responders; by TP ; run ;
  proc sgplot data=Responders ;
       scatter y=log10Observed x=TP / group=Test ;
       yaxis label = "Overall Assay Test (log10 units)"  ;
       xaxis label = "Weeks" ;
  run ;
ods rtf close ;
```

Creating One or a Few Graphs

Program* | Log | Output Data (4)

Save | Run | Stop | Selected Server: SASApp (Connected) | Analyze Program | Export | Send To | Create | Changes | Commit | History | Properties

```
    * indicator variable of just 1s to sum up in Proc SQL below ;
    indicator = 1 ;

    TimeStem = substr(TP,1,2) ;
  run ;

  * if multiple dates within subject generated by simulation then take the first date ;
proc sort data=Responders_TMP1 ; by SampleID TP TimeStem ; run ;
data Responders_TMP2 ;
  set Responders_TMP1 ;
  by SampleID TP TimeStem ;
  if first.TimeStem ;
run ;

* keep if number of time points is 4 ;
proc sql ;
  create table Responders_GE4WKS as
  select
      SampleID
      ,sum(indicator) as Total
  from Responders_TMP2
  group by SampleID
  order by SampleID ;
quit ;

proc sort data=Responders_TMP2   ; by SampleID ; run ;
proc sort data=Responders_GE4WKS ; by SampleID ; run ;
data Responders (drop=indicator timestem total) ;
  merge Responders_TMP2 Responders_GE4WKS ;
  by SampleID ;
run ;
* end data set creation for Responders ;

* Log-Log Data using SGPLOT: Customization ;
%LET SampleID   = 150 ; * generate 150 subjects ;
%LET NumSamples = 2   ; * with 2 replicate each but later only take first replicate ;
data Xvalues (rename = (i = SubjID j = Replicate)) ;
    seed = 603287 ;
    do i = 1 to &SampleID ;
       do j = 1 to &NumSamples ;
          X = ranuni(seed) ;
          output ;
       end ;
```

Graphing results of data for all patients.

Log Summary

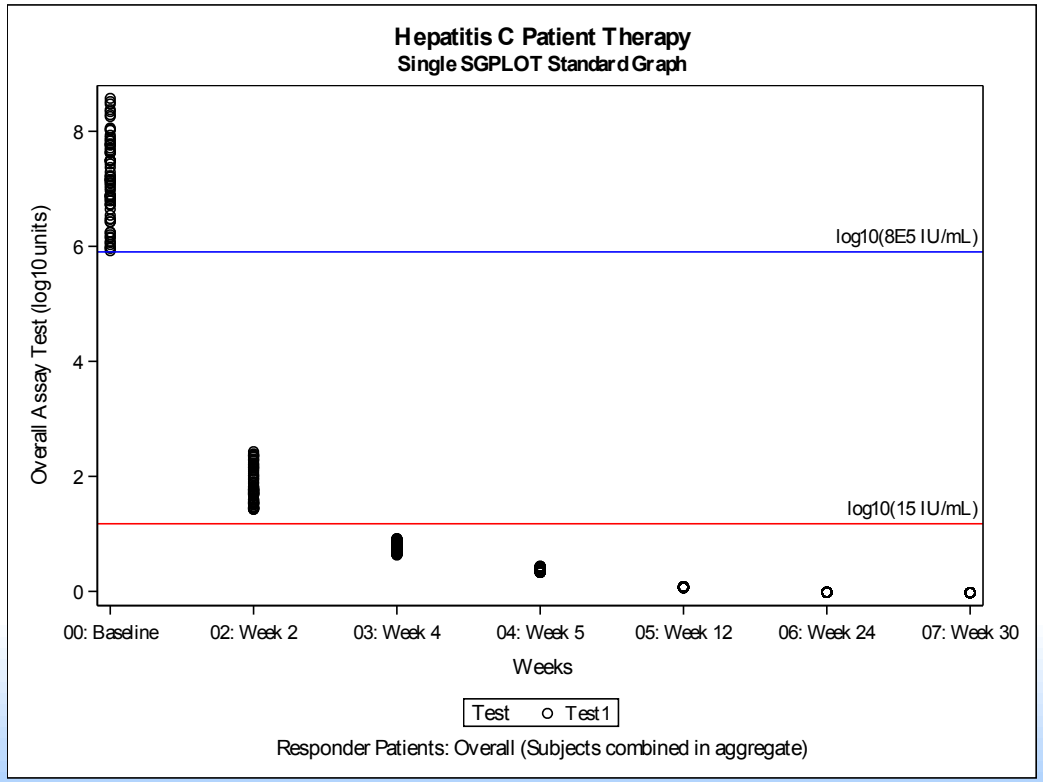
Errors (0) | Warnings (0) | Notes (18)

Description	Line	Affected Code	Log Line	Program Li...
NOTE: Writing TAGSETS.SASREPORT13(EGSR) Body file: EGSR	27	options(rolap="on")	22	

Creating One or a Few Graphs

Standard SG: Results!

- Graphical results of running the standard SAS code above for the Hepatitis C Therapy Response (HepCTR) Simulated Aggregate Data for Responders across 30 weeks on treatment.



Creating One or a Few Graphs

Standard SG

- Now, suppose pre-SAS v9.4, the user wishing to replicate an SGPLOT across many-many subjects (not just a few where quick repeated code would save the day), for example, would need to perform some fancy programming.

Automation

Automation

Automate It!

- Producing one to a just a few graphs normally is quite easy to do with a quick cut and paste or macrotizing your repeating code to use with a few macro calls.

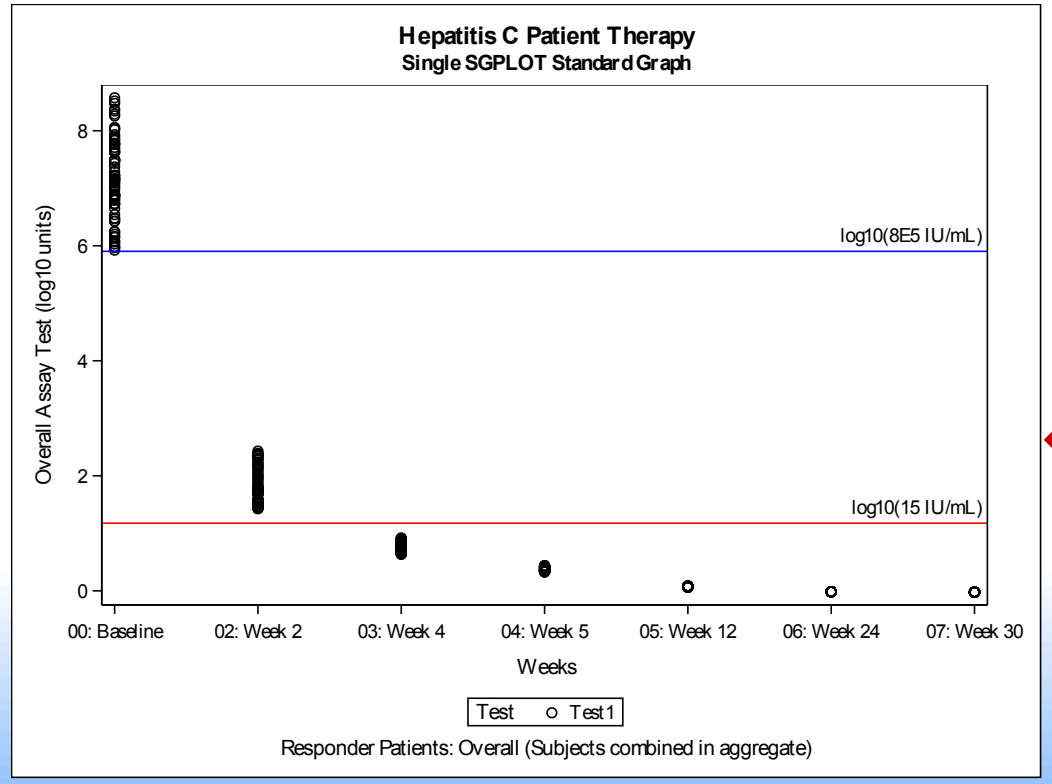
Automate It!

- However, at times, one may need to produce the same graph for more than just a few times (say, 50, 100, 200, etc.)

Automation

Automate It!

- For example, one may need to expand (or separate) an aggregate graph with a great many patients (or subjects) into individual patient graphs.



What if we need to create individual patient graphs from the <u>combined patients in our previous analysis</u>!?

Automation

Automate It!

- For this, the copy/paste and macrotizing of the simple code not work efficiently at all!

Automate It!

- For this, the copy/paste and macrotizing of the simple code not work efficiently at all!

- This necessitates an automation solution.

- We give an example using the simulated patient Hepatitis C Therapy Response data (HepCTR; Appendix A).

Automate It!

Step 3: **Many Patients in Separate Graphs with pre-SAS v9.4**. Run the following *fancy* code to produce separate SGPLOT graphs using SAS Macro code.

Automation

Automate It!

```
* Create macro variables with values and a total
count of distinct values to iterate the later macro
by SubjectID ;
   proc sql noprint ;
      select distinct SampleID into :varVal1- from
                                          Responders ;
      %let varCount = &SQLOBS. ;
   quit ;
```

Automation

Automate It!

An Aside…

```
* %modstyle is a SAS-provided style macro for ODS
  should run in your SAS installation ;
* It cycles through the indicated types (in example
  below, in this case, for up to 4 groups) in your SG
  graphs when "style=markers" is indicated on the ODS
  line ;
   %modstyle(        name = markers ,
                   parent = listing ,
                     type = CLM ,
               linestyles = solid dash shortdash dot ,
                   colors = green blue purple red    ,
                  markers = circle triangle square
                                          diamond ) ;
```

Automation

Automate It!

An Aside…

```
* this will re-direct to a non-server path –
  useful if working on a restricted server that
  if path not set will produce write errors;

ods listing style=markers gpath="c:\" ;
options orientation = landscape ; * setting page
                                    orientation to landscape ;
```

Later, we will see how not having this gpath redirect will produce a log error when it attempts to write to the read-only server.

Automation

Automate It!

Define your macro …

Comes from the PROC SQL code a few slides before…

```
%macro bySampleID ;
    %do index = 1 %to &varCount ;
        title1 "Hepatitis C Patient Therapy Example" ;
        title2 "Multiple SGPLOT Graphs by Patient using
                Proc SQL and SAS Macro" ;
```

Automation

Automate It!

Dynamic coding comes from the PROC SQL code a few slides before…

```
proc sgplot data = Responders
     (where=(SampleID=&&varVal&index.)) ;
    scatter y=log10Observed x=TP /
     markerattrs=(SIZE=8px) group=TEST name="TEST" ;
     yaxis values = (0 TO 8 BY 0.5)
     labelattrs=(color=Grey family=Imago)
     valueattrs=(color=Grey family=Imago) ;
```

Automation

Automate It!

```
refline 5.903089987 / axis=y label="log10(8E5 IU/mL)"
                      lineattrs=(color="blue")
                      labelloc=inside labelpos=max ;
refline 1.17609126  / axis=y label="log10(15 IU/mL)"
                      lineattrs=(color="red")
                      labelloc=inside labelpos=max ;
inset "Sample ID = &&varVal&index." / position =
                                          topright ;
label TP = "Time Point"
      log10Observed = "Result (log10 IU/mL)" ;
run ;
%end ;
%mend bySampleID ;
```

Dynamic coding comes from the PROC SQL code a few slides before…

Automate It!

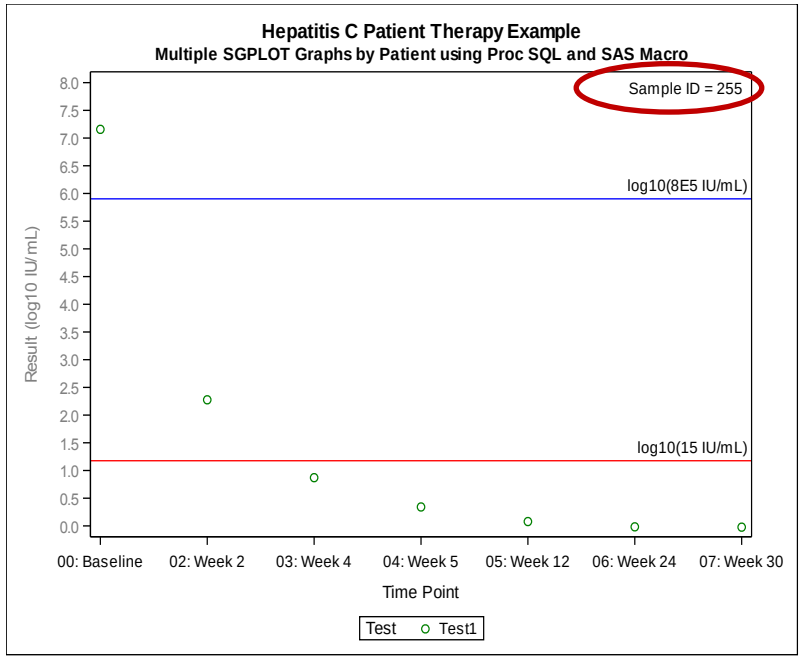
Now, that was the end of the macro definition.

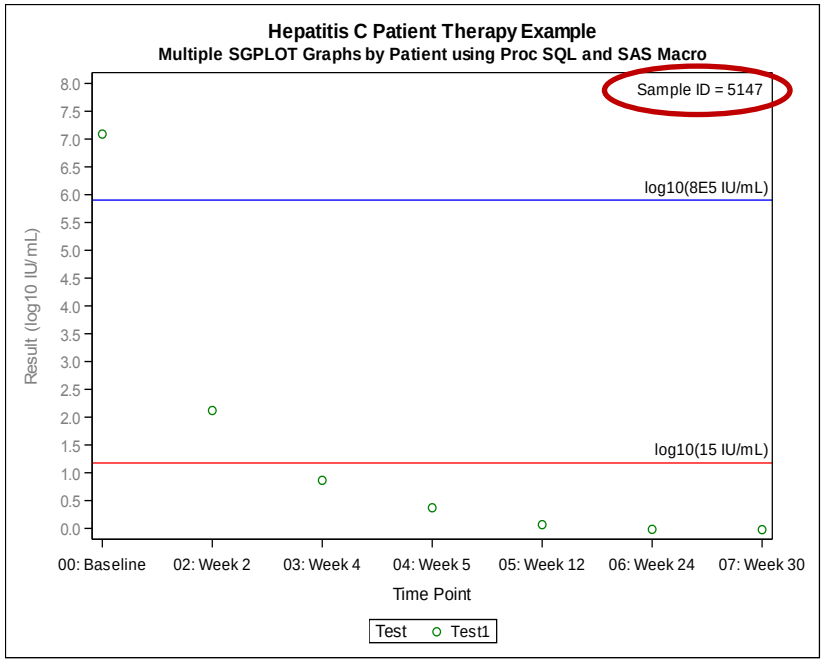
Now we need to call the macro to run it…

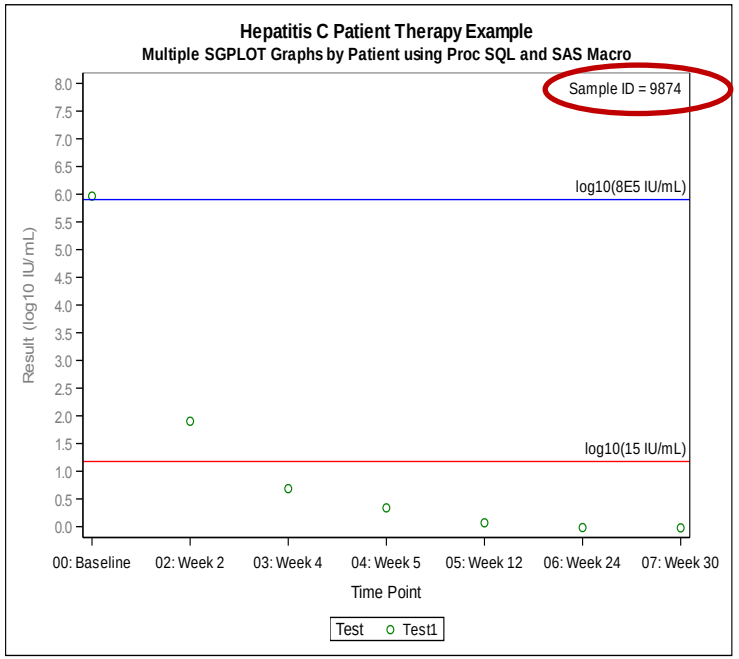
```
%bySampleID ;
ods rtf close ;
options orientation = portrait ;
 * resetting page orientation to portrait;
```

Automation

Automate It!







Three graphs of the 10000 results of running the SAS code above for the Hepatitis C Therapy Response (HepCTR) Simulated by-Subject Data using PROC SQL and SAS/MACRO code for Responders across 30 weeks on treatment.

Automate It!

Recent advancements in the SAS SG procedures have added the “by” statement to make it much easier than the macro coding found in **Step 3**, above.

Automate It!

Step 4: **Patients in Separate Graphs with SAS v9.4**.

Run the following code to produce separate SGPLOT graphs.

- Notice the "by" in the code is red, indicating it may not be allowed in the SGPLOT code.
- However, this is acceptable and will run in the meantime that SAS Institute adds this to the procedure-allowed code and turn it light blue (current SAS Enterprise Guide used is version 7.15 HF3 with SAS back-end implementation version 9.4 M5)

Automation

Automate It!

This code will run fine in SAS v9.4 M5, even if "by" is red.

TP=Time Point

```
ods noproctitle ;
ods RTF file = "C:\WUSS2018_Responders_bySubject_&sysdate..RTF"
                                                                  ePrinter ;
* above: sysdate appends current data to your file nam                white
    background ;
title2 "" ;
footnote1 "Responder Patients: Individual Subjects (separately)" ;
proc sort data=Responders ; by SampleID TP ; run ;
proc sgplot data=Responders ;
    by SampleID ;
     scatter y=log10Observed x=TP / group=Test ;
     YAXIS LABEL = "Overall Assay Test (log10 units)" ;
      XAXIS LABEL = "Weeks" ;
run ;
ods rtf close ;
title2 ; footnote1 ;
```

Creating One or a Few Graphs

Program* | Log | Results - SAS Report | Results - RTF

Save | Run | Stop | Selected Server: SASApp (Connected) | Analyze Program | Export | Send To | Create | Changes | Commit | History | Properties

```
data Responders (drop=indicator timestem total) ;
   merge Responders_Test1_TMP2 Responders_Test1_GE4WKS ;
   by SampleID ;
     * select only those subjects with complete data ;
   if Total >= 7 and Max_log10Observed > log10(8E5) ;

 run ;
 * end data set creation for Responders ;

 ods noproctitle ;
 ods RTF file = "\\rpbmssaspl\biometrics\Global\Presentations\WUSS\2018_Sacramento\Results\RTF\WUSS2018_Responders_bySubject_&sysdate..RTF" style=MonoChromePrinter ; * sysdate
 title1 "Hepatitis C Patient Therapy Example" ;
 title2 "Multiple SGPLOT Graphs by Patient using 'by' in SAS v9.4" ;
 footnote1 "Responder Patients: by Subject Individual Subjects (Separately)" ;
proc sort data=Responders ; by SampleID TP ; run ;
proc sgplot data=Responders ;
     by SampleID ;
     scatter y=log10Observed x=TP  / group=Test ;

     yaxis values = (0 TO 8 BY 0.5) labelattrs=(color=Grey family=Imago) valueattrs=(color=Grey family=Imago ) ;

     refline 5.903089987 / axis=y label="log10(8E5 IU/mL)" lineattrs=(color="blue") labelloc=inside labelpos=max ;
     refline 1.17609126  / axis=y label="log10(15 IU/mL)"  lineattrs=(color="red")  labelloc=inside labelpos=max ;

     yaxis label = "Overall Assay Test (log10 units)"  ;
     xaxis label = "Weeks" ;
run ;
ods rtf close ;
title1 ; title2 ; footnote1 ;
```

Using "by" statement in SGPLOT

Customization

Customization

Customize It!

- Sometimes, we are forced to use customization because the SG procedures may not have a user-requested option available as of yet.

  - For example, the user cannot currently add a "targeted" *regression equation* in an SGPLOT graph. Here is an example of this using the simulated log-log data from two tests (Log-Log; Appendix B of paper).

Customization

Customize It!

GENERAL STEPS

- **Step 5**: **Adding a Regression Equation and Other Items to Your Graph**. Using the following code steps:
- **Step 5a**: create a regression equation using PROC REG and save the results using ODS OUTPUT OUT, then
- **Step 5b**: Use PROC IML to extract bits and pieces of what you need, declaring them macro variables then
- **Step 5c**: Add them to your SGPLOT graph.

Customization

Customize It!

- **By the way**: This is an example that can be generalized to add any type of SAS procedure output to any SG graph that you may desire to customize.

Customization

Customize It!

SPECIFIC EXPANDED STEPS

- **Step 5a: Creating a Regression Equation and Saving Results.**

Customization

Customize It!

- After running the SAS code in **Appendix B** of the paper to create the Log-Log data,

… run the following code to produce a regression equation using PROC REG

… whilst saving the result using the ODS OUTPUT OUT option

Customization

Customize It!

- **NOTE:** In order to do this, you will need to
  - use the "**ods trace on ;**" at the beginning of your code that you want to extract parameter estimates from
  - and "**ods trace off ;**" at the end of that code in order to find the correct output for the "**ods output …**"

(See, for example, Oltsik, 2008 and **APPENDIX D**)

Customization

Customize It!

```
* bits and pieces ;
ods trace on ;
ods output ParameterEstimates=Parms1
           FitStatistics=FitStats1 ;
proc reg data= TwoTests_OneRep ;
 model Test2 = Test1 ;
 output out=OLSresids1 predicted=pred1
        residual=resids1 press=press1 ;
run ; quit ;
ods trace off ;
```

Bracket your PROC (whatever proc that may be) with these.

Note that these results are found in the SAS LOG window/tab *after* running your code.

Upon running this code, you will see the output in the **SAS LOG** window/tab as shown in the **Output 5a** SAS log output box below.

Output 5a. SAS Log Output using ODS TRACE for correct ODS OUTPUT parameter discovery and naming.

```
Output Added:
-------------
Name:        NObs
Label:       Number of Observations
Template:    Stat.Reg.NObs
Path:        Reg.MODEL1.Fit.Test2.NObs
-------------
Output Added:
-------------
Name:        ANOVA
Label:       Analysis of Variance
Template:    Stat.REG.ANOVA
Path:        Reg.MODEL1.Fit.Test2.ANOVA
-------------
Output Added:
-------------
Name:        FitStatistics
Label:       Fit Statistics
Template:    Stat.REG.FitStatistics
Path:        Reg.MODEL1.Fit.Test2.FitStatistics
-------------
Output Added:
-------------
Name:        ParameterEstimates
Label:       Parameter Estimates
Template:    Stat.REG.ParameterEstimates
Path:        Reg.MODEL1.Fit.Test2.ParameterEstimates
...
```

Selected PROC REG ODS output data set names (among 6 possible data set outputs: Nobs, ANOVA, FitStatistics, ParameterEstimates, OutputStatistics, ResidualStatistics) using the ODS TRACE ON and ODS TRACE OFF option bracketing your PROC code.

Note that these results are found in the SAS LOG window/tab ***after*** running your code (as shown in "**First Pass**" above).

Customization

Customize It!

- **Step 5b: Extract Information from ODS OUTPUT for IML Processing (see previous slide).**

- Make sure to familiarize yourself with the contents of the ODS OUTPUT temporary SAS data sets **(viz., open the data set up to take a look inside!)**

OLSplot1_WUSS2018_Sacramento_EXAMPLE1_07Aug2018A

Program | Log | Output Data (6) | Results - SAS Report | Results - HTML | Results - RTF

FITSTATS1

FITSTATS1

Filter and Sort | Query Builder | Where | Data | Describe | Graph | Analyze | Export | Send To

	Model	Dependent	Label1	cValue1	nValue1	Label2	cValue2	nValue2
1	MODEL1	Test2	Root MSE	0.43422	0.434215	R-Square	0.9136	0.913595
2	MODEL1	Test2	Dependent Mean	5.36763	5.367629	Adj R-Sq	0.9124	0.912378
3	MODEL1	Test2	Coeff Var	8.08951	8.089512			0

Customization

Customize It!

- Step 5b: Extract Information from ODS OUTPUT for IML Processing (see previous slide).

- Make sure to familiarize yourself with the contents of the ODS OUTPUT temporary SAS data sets (viz., open the data set up to take a look inside!)

OLSplot1_WUSS2018_Sacramento_EXAMPLE1_07Aug2018A

Program | Log | Output Data (6) | Results - SAS Report | Results - HTML | Results - RTF

PARMS1

PARMS1

Filter and Sort | Query Builder | Where | Data | Describe | Graph | Analyze | Export | Send To

	Model	Dependent	Variable	DF	Estimate	StdErr	tValue	Probt	LowerCL	UpperCL
1	MODEL1	Test2	Intercept	1	0.77433	0.17518	4.42	<.0001	0.42504	1.12363
2	MODEL1	Test2	Test1	1	0.99153	0.03619	27.40	<.0001	0.91937	1.06369

Customization

Customize It!

- Next, run the SAS code that uses the SAS IML procedure to extract the parameter estimates from the saved results in **Step 5a**.

- **What's the General Strategy in IML?**
  1. Use < Data 1 >
  2. Read Only Variables Interested in from < Data 1 >
  3. Close < Data 1 >
  4. Do something with < Data 1 >
  5. Create a macro variable using CALL SYMPUTX() to drop into customized SG Graph.

Customization

Customize It!

NOTE: Deeper dive coming up in a couple of slides…

```
proc iml ;
    use Parms1 ; * use the Parms1 data set from
                   above ;
    read all VAR{estimate lowercl uppercl} into X ;
                 * read only variables we need;
    close Parms1 ;

    use FitStats1 ;   * use the FitStats1 data set
                        from above ;
    read all VAR{NVALUE2} into Y ;
                      * read only variable we need;
    close Fitstats1 ;

    use OLSresids1 ; * use the OLSresids1 data set
                       from above ;
    read all VAR{SubjID} into Z ; * extract SubjID
                                    to count how many IDs ;
    close OLSresids1 ;
```

Customization

Program* | Log | Output Data

Save | Run | Stop | Selected Server: SASApp (Connected) | Analyze Program | Export | Send To | Create | Changes | Commit | History | Properties

```
      end ;
    end ;
run ;

* simulating responder patients ;
data TwoTests_OneRep (drop=Seed S B0 B1) ;
  set Xvalues ;
  seed = 284631 ;

  b0 = 1.24 ; b1 = 9.98 ;
   Y = b0 + b1 * X ; * linear model ;

  * jittering X values for each subject ;
  s = 1.5 ; * scale factor for spread ;
  jY = Y + s * ranuni(seed) - 0.01 ; * random uniform distribution to
                                       generate random observations ;
  if jY < 0 then jY = 0 ;

  * let Y be Test1 and the jittered Y be Test2 ;
  Test1 = Y  ;
  Test2 = jY ;

  * restrict observations from 2 to 8 using Test1 and Test2 ;
  if ( 2 <= Test1 <= 8 ) AND ( 2 <= Test2 <= 8) ;

  * select first of two replicates for pedagogical reasons ;
  if replicate = 1 ;
run ;

* END Log-Log data creation ;
********************************************************************** ;
********************************************************************** ;

ods _ALL_ close ;
* Use the MODSTYLE macro supplied by SAS to change the default colors and markers ;
 %modstyle(name = markers ,
           parent = listing,
           type = CLM ,
           markers = circle triangle square) ; *circle triangle square
                              circlefilled trianglefilled diamondfilled) ;

* The investigators want a square plot (no proc template necessary) ;
ods graphics / width = 550px height = 550px ;
```

PROC IML

derived

variables

Log Summary

Errors (0) | Warnings (0) | Notes (4)

Description	Line	Affected Code	Log Line	Program Li...
NOTE: Writing TAGSETS.SASREPORT13(EGSR) Body file: EGSR	27	options(rolap="on")	22	

Roche

Customization

Deeper dive

Customize It!

```
proc iml ;
    use Parms1 ; * use the Parms1 data set from
                   above ;
    read all VAR{estimate lowercl uppercl} into X ;
                 * read only variables we need;
    close Parms1 ;
```

PARMS1 Temporary SAS Data Set from ODS OUTPUT OUT Statement above.

Filter and Sort | Query Builder | Where | Data | Describe | Graph | Analyze | Export | Send To

	Model	Dependent	Variable	DF	Estimate	StdErr	tValue	Probt	LowerCL	UpperCL
1	MODEL1	Test2	Intercept	1	0.77433	0.17518	4.42	<.0001	0.42504	1.12363
2	MODEL1	Test2	Test1	1	0.99153	0.03619	27.40	<.0001	0.91937	1.06369

```
    use OLSresids1 ; * use the OLSresids1 data set
                       from above ;
    read all VAR{SubjID} into Z ; * extract SubjID
                                    to count how many IDs ;
    close OLSresids1 ;
```

Customization

Customize It!

Example of reading data read into PROC IML:
Xrc = X[row, column] => X[1,1] = X[row 1, column 1] = 0.77433

```
N_Tot_1 = nrow(Z[,1]) ;

ols1_b0 = X[1,1] ; * extract intercept in Y = b0 + b1*X equation ;
ols1_b1 = X[2,1] ; * extract slope ;

CI95_LL1_b0 = X[1,2] ; * extract lower confidence limit for intercept ;
CI95_UL1_b0 = X[1,3] ; * extract upper confidence limit for intercept ;

CI95_LL1_b1 = X[2,2] ; * extract lower confidence limit for slope ;
CI95_UL1_b1 = X[2,3] ; * extract upper confidence limit for slope ;

Rsquare1 = Y[1,1] ; * extract R-square value ;
```

Estimate	LowerCL	UpperCL
0.77433	0.42504	1.12363
0.99153	0.91937	1.06369

Customization

Example of reading data read into PROC IML:
Xrc = X[row, column]

An ASIDE…

GENERAL TIP: Regarding rows and columns, a mnemonic for remember which is which, my computer science professor at UC Berkeley told us, You "row across" … as in a row boat and "Call'em up" on the telephone. So the row goes across and column goes up and down. Interesting that I still remember that all of these years! -Jesse

Customization

Customize It!

```
 * assign a value to a macro variable for use in SG procedures ;
 call symputx("N_Tot_1",N_Tot_1) ;

 call symputx("ols1_
 call symputx("ols1_

 call symputx("CI95_
 call symputx("CI95_

 call symputx("CI95_LL1_b1",round(CI95_LL1_b1,0.001)) ;
 call symputx("CI95_UL1_b1",round(CI95_UL1_b1,0.001)) ;

 call symputx("Rsquare1",round(Rsquare1,0.01)) ;
run ;
```

Use CALL SYMPUTX() to assign a data step value to a macro variable for later use in your customized SAS SG graphs.

Customize It!

Customization

```
 * assign a val
 call symputx("

 call symputx("
 call symputx("

 call symputx("
 call symputx("

 call symputx("
 call symputx("

 call symputx("
run ;
```

ASIDE General TIP:

Question: Why use CALL SYMPUTX() rather than CALL SYMPUT()?

Answer: CALL SYMPUTX() was introduced in SAS v9 to enhance CALL SYMPUT() so it is more advanced.

1. CALL SYMPUTX will automatically convert any numeric variables into character variables before assigning them to macro variables. That is, there is NO need to manually convert variables (for example, using a PUT() statement) as when using CALL SYMPUT().

2. CALL SYMPUTX will remove any leading or trailing blanks from a variable. That is, there is NO need to use the STRIP() or LEFT() functions for removal of any leading spaces.

Customize It!

Step 5c: Adding the Macro Variable Results to SGPLOT.

Finally, run the next SAS code that uses the macro variable results in **Step 5b** to populate the SGPLOT graph as shown in the figure below.

Customization

Program* | Log | Output Data (3)

Save | Run | Stop | Selected Server: SASApp (Connected) | Analyze Program | Export | Send To | Create | Changes | Commit | History | Properties

```
CI95_LL1_b1 = X[2,2] ;
CI95_UL1_b1 = X[2,3] ;

Rsquare1 = Y[1,1] ;

call symputx("N_Tot_1",N_Tot_1) ;

call symputx("ols1_b0",round(ols1_b0,0.001)) ;
call symputx("ols1_b1",round(ols1_b1,0.001)) ;

call symputx("CI95_LL1_b0",round(CI95_LL1_b0,0.001)) ;
call symputx("CI95_UL1_b0",round(CI95_UL1_b0,0.001)) ;

call symputx("CI95_LL1_b1",round(CI95_LL1_b1,0.001)) ;
call symputx("CI95_UL1_b1",round(CI95_UL1_b1,0.001)) ;

call symputx("Rsquare1",round(Rsquare1,0.01)) ;
run ;

* Customize the SGPLOT using the result from above ;
ods rtf file = "C:\Results_MethodComparison_FullMonty_&sysdate..rtf" gtitle style=markers ;
proc sgplot data = TwoTests_OneRep noautolegend ;
  scatter x=Test1 y=Test2 / markerattrs=(size=5px) ;
  reg     x=Test1 y=Test2 / markerattrs=(size=5px)
          LINEATTRS = (THICKNESS=0.9 COLOR=Blue PATTERN=1) name="OLS" ;

          LINEPARM x=0 y=0 slope=1 / LEGENDLABEL = "Unity: Y=X"
                   LINEATTRS = (THICKNESS=0.2 COLOR=Black PATTERN=2)
                                                 name="Unity" ;

  XAXIS LABEL = "Test 1 (log10 cp/mL)" VALUES = (1 TO 9 BY 0.5) ;
  YAXIS LABEL = "Test 2 (log10 cp/mL)" VALUES = (1 TO 9 BY 0.5) ;

  KEYLEGEND "Unity" "OLS" / LOCATION=outside POSITION=bottom ;
  TITLE1 "Method Comparison Study" ;
  TITLE2 "Test 2 (log10 cp/mL) vs. Test 1 (log10 cp/mL)" ;

  FOOTNOTE1 ; FOOTNOTE2 ;
  INSET  "OLS Regression (N= &&N_Tot_1) "
         "Y = &&ols1_b0 + &&ols1_b1 X"
         "R-square= &&Rsquare1"
         "95% CI Intercept (&&CI95_LL1_b0, &&CI95_UL1_b0)"
```

Example of Customization using PROC IML

Log Summary

Errors (0) | Warnings (1) | Notes (11)

Description	Line	Affected Code	Log Line	Program Li...
NOTE: Writing TAGSETS.SASREPORT13(EGSR) Body file: EGSR	27	options(rolap="on")	22	

Customization

Customize It!

An Aside…

```
ods rtf file =
"C:\Results_MethodComparison_FullMonty_&sysdate..rtf"
  gtitle style=markers ;
```

```
* gtitle will add your titles to the
  top of your graph - NOTE: if your title still does not show
  up, perform an "ODS _ALL_ close ;" command right before your
  code as illustrated in APPENDIX B of the paper and re-run
  your code - we guarantee it will show up ;
```

Customization

Customize It!

Run the main SGPLOT code …

```
proc sgplot data = TwoTests_OneRep noautolegend ;
     scatter x=Test1 y=Test2 / markerattrs=(size=5px) ;
     reg     x=Test1 y=Test2 / markerattrs=(size=5px)
             LINEATTRS = (THICKNESS=0.9 COLOR=Blue PATTERN=1) name="OLS" ;

     LINEPARM x=0 y=0 slope=1 / LEGENDLABEL = "Unity: Y=X"
     LINEATTRS = (THICKNESS=0.2 COLOR=Black PATTERN=2) name="Unity" ;

     XAXIS LABEL = "Test 1 (log10 cp/mL)" VALUES = (1 TO 9 BY 0.5) ;
     YAXIS LABEL = "Test 2 (log10 cp/mL)" VALUES = (1 TO 9 BY 0.5) ;

     KEYLEGEND "Unity" "OLS" / LOCATION=outside POSITION=bottom ;
     TITLE1 "Method Comparison Study" ;
     TITLE2 "Test 2 (log10 cp/mL) vs. Test 1 (log10 cp/mL)" ;
```

Notice down to here it is standard code! Next piece will add your customization…

Customization

Customize It!

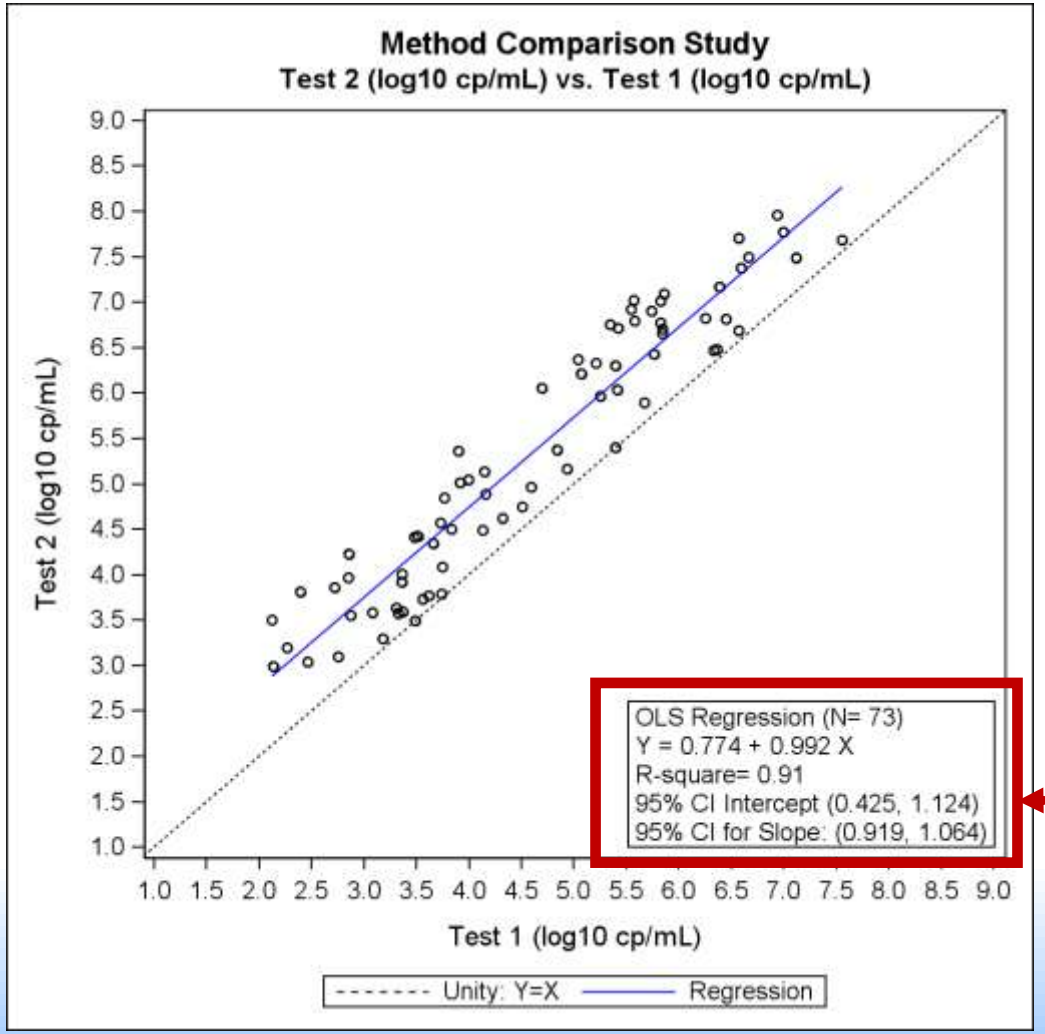
Run the main SGPLOT code (cont.)

```
FOOTNOTE1 ; FOOTNOTE2 ;
     INSET  "OLS Regression (N= &&N_Tot_1) "
            "Y = &&ols1_b0 + &&ols1_b1 X"
            "R-square= &&Rsquare1"
            "95% CI Intercept (&&CI95_LL1_b0, &&CI95_UL1_b0)"
            "95% CI for Slope: (&&CI95_LL1_b1, &&CI95_UL1_b1)"
            / POSITION = BOTTOMRIGHT BORDER;
run ;
ods rtf close ;
```

ASIDE: "It is not uncommon within the SAS Language for some special characters, such as; single quotes, periods and ampersands, to have dual meanings. Dual use is indicated in the code by placing two of the desired symbol. During the resolution of the macro variables the two 2 symbols are resolved to one." -Art Carpenter (SUGI22, 1997).

Customization

Customize It!



The graph shows a customized SGPLOT graph using PROC IML and the simulated Log-Log method comparison/correlation data from Appendix B in the paper following the above **Steps 5a** to **5c**.

Note the customization!

Macrotization

Macrotization

Macrotize It!

- The next natural step is to macrotize your customized code to use again and again.

- **The next slide** shows the OLSplot SAS MACRO that macrotizes the customization in the previous section using the Log-Log method comparison data.

- The macro is shown in its entirety in the paper, including the macro call at the end.

Macrotization

Macrotize It!

```
%macro OLSplot(
          dsin  = ,    /* temporary SAS data set                    */
          where = ,    /* any subsetting                            */
       analyte  = ,    /* can be HCV, HIV, HBV, CMV, etc.           */
          unit  = ,    /* units, can be cp/mL or IU/mL, etc.        */
         visit  = ,    /* visit variable, can be blank              */
       subject  = ,    /* subject or sample id variable             */
          xvar  = ,    /* x-axis variable                           */
          xlab  = ,    /* x-axis label                              */
          yvar  = ,    /* y-axis variable                           */
          ylab  = ,    /* y-axis label                              */
          grid  = ,    /* if graph grid is desired: Yes or No       */
        parmno  = ,    /* user supplied unique analysis number      */
      graphmin  = ,    /* x- and y-axis minimum graph values        */
      graphmax  = ,    /* x- and y-axis maximum graph values        */
graphincrement  = ,    /* x- and y-axis graph increment values      */
         title1 = ,
         title2 = ) ;
```

Macrotization

Macrotize It!

Remember this used to be "Parms1"!

```
ods output ParameterEstimates=Parms&parmno.
FitStatistics=FitStats&parmno. ;
proc reg data= &dsin ;
 model &yvar = &xvar / clm cli clb ;
 output out=OLSresids&parmno.
predicted=pred&parmno. residual=resids&parmno.
press=press&parmno. ;
  %if %upcase(&visit) >= 0  %then %do ;
      %str(where visitn = &visit ; )  ;
  %end ;
run ; quit ;
```

Macrotization

Macrotize It!

Remember these used to have a "1" at the end. Now they are generalized!

```
proc iml ;
 use Parms&parmno. ;
 read all VAR{estimate lowercl uppercl} into X ;
 close Parms&parmno. ;

 use FitStats&parmno. ;
 read all VAR{NVALUE2} into Y ;
 close Fitstats&parmno. ;

 use OLSresids&parmno. ;
 read all VAR{&subject.} into Z ;
 close OLSresids&parmno. ;
```

Macrotization

Macrotize It!

More generalizations …

```
N_Tot_&parmno. = nrow(Z[,1]) ;

ols&parmno._b0 = X[1,1] ;
ols&parmno._b1 = X[2,1] ;

CI95_LL&parmno._b0 = X[1,2] ;
CI95_UL&parmno._b0 = X[1,3] ;

CI95_LL&parmno._b1 = X[2,2] ;
CI95_UL&parmno._b1 = X[2,3] ;

Rsquare&parmno. = Y[1,1] ;
```

Macrotization

Macrotize It!

More generalizations (cont.) …

```
 call symputx("N_Tot_&parmno.",N_Tot_&parmno.) ;

 call symputx("ols&parmno._b0",round(ols&parmno._b0,0.001)) ;
 call symputx("ols&parmno._b1",round(ols&parmno._b1,0.001)) ;

 call symputx("CI95_LL&parmno._b0",round(CI95_LL&parmno._b0,0.001)) ;
 call symputx("CI95_UL&parmno._b0",round(CI95_UL&parmno._b0,0.001)) ;

 call symputx("CI95_LL&parmno._b1",round(CI95_LL&parmno._b1,0.001)) ;
 call symputx("CI95_UL&parmno._b1",round(CI95_UL&parmno._b1,0.001)) ;

 call symputx("Rsquare&parmno.",round(Rsquare&parmno.,0.01)) ;
run ;
```

Macrotization

Macrotize It!

More generalizations (cont.) …

```
proc sgplot data = &dsin noautolegend ;
  scatter x=&xvar y=&yvar / markerattrs=(size=5px) ;
  reg     x=&xvar y=&yvar / markerattrs=(size=5px)
          lineattrs = (thickness=0.9 color=blue pattern=1)
name="OLS" ;
  lineparm x=0 y=0 slope=1 / legendlabel = "Unity: Y=X"
           lineattrs = (thickness=0.2 COLOR=Black pattern=2)
name="Unity" ;
```

Macrotization

Macrotize It!

More generalizations (contd.) …

```
%if %upcase(&grid) = "YES" %then %do ;
        %str( XAXIS LABEL = &xlab VALUES = (&GraphMin TO
&GraphMax BY &GraphIncrement) GRID ; )  ;
      %end ;
      %if %upcase(&grid) ^= "YES" %then %do ;
        %str( XAXIS LABEL = &xlab VALUES = (&GraphMin TO
&GraphMax BY &GraphIncrement) ; )  ;
      %end ;

    %if %upcase(&grid) = "YES" %then %do ;
        %str( YAXIS LABEL = &ylab VALUES = (&GraphMin TO
&GraphMax BY &GraphIncrement) GRID ; )  ;
      %end ;
      %if %upcase(&grid) ^= "YES" %then %do ;
        %str( YAXIS LABEL = &ylab VALUES = (&GraphMin TO
&GraphMax BY &GraphIncrement) ; )  ;
      %end ;
```

Roche

Macrotization

Macrotize It!

More generalizations (cont.) …

```
KEYLEGEND "Unity" "OLS" / LOCATION=outside POSITION=bottom ;
 TITLE1 &title1 ;
 TITLE2 &title2 ;

 FOOTNOTE1 ; FOOTNOTE2 ;
 INSET  "OLS Regression (N= &&N_Tot_&parmno.) "
        "Y = &&ols&parmno._b0 + &&ols&parmno._b1 X"
        "R-square= &&Rsquare&parmno."
        "95% CI Intercept (&&CI95_LL&parmno._b0,
 &&CI95_UL&parmno._b0)"
        "95% CI for Slope: (&&CI95_LL&parmno._b1,
 &&CI95_UL&parmno._b1)" / POSITION = BOTTOMRIGHT BORDER;

  %if %upcase(&visit) >= 0  %then %do ;
      %str(where visitn = &visit; )  ;
  %end ;
 run ;
 %mend OLSplot ;
```

…and end the macro definition.

Macrotization

```
* ************************** ;
* MACRO CALL for OLSplot.sas ;
* ************************** ;
%include "C:\OLSplot.sas" ;
ods _ALL_ close ;
```

MACRO CALL for OLSplot.sas MACRO

Read SAS Macro in…
If not already defined in body of your SAS program.

```
* Use the MODSTYLE macro supplied by SAS to change the default colors and markers ;
%modstyle(name = markers ,
          parent = listing,
          type = CLM ,
          markers = circle triangle square) ;

* The investigators want a square plot (no proc template necessary) ;
ods graphics / width=550px height=550px imagename="C:\PNG" imagefmt=png ;

* For any macro troubleshooting ;
options mprint mlogic symbolgen ;
```

Macrotization

```
ods rtf file = "C:\Results_MethodComparison_&sysdate..rtf" gtitle style=markers ;
%OLSplot(
    dsin      = TwoTests_OneRep ,          /* required                          */
    where     = ,
    analyte   = CMV     ,                  /* required                          */
    unit      = cp/mL   ,                  /* required                          */
    visit     = ,                          /* visit variable, can be blank      */
    subject   = SubjID ,
    xvar      = Test1   ,                  /* required                          */
    xlab      = "Test 1 (log10 cp/mL)" ,   /* can be blank                      */
    yvar      = Test2   ,                  /* required                          */
    ylab      = "Test 2 (log10 cp/mL)" ,   /* can be blank                      */
    grid      = "NO"    ,                  /* required: Yes or No               */
    parmno    = 1       ,                  /* required: analysis number: >= 1 */
    graphmin  = 1       ,                  /* required                          */
    graphmax  = 9       ,                  /* required                          */
    graphincrement = 0.5 ,                 /* required                          */
    title1 = "Method Comparison Study", /* can be blank                      */
    title2 = "Test 2 (log10 cp/mL) vs. Test 1 (log10 cp/mL)") ; /* can be blank */
ods rtf close ;

* END OF SAS CODE ;
```

Aside: Appends date to your file: _DDMMMYY.rtf

e.g., _06SEP18.rtf

Macrotization

Run the SAS Macro code and you will get the same results as when running it non-macrotized but now you can do many calls of the macro.

Review and Conclusion

Review and Conclusion

What have we learned?

- The SAS/Graph Journey was a learning experience
- Introduced to Your mum's SG Procedures
- Created One or a Few Graphs
- Learned Automation for a Gazillion Graphs
- Learned Customization for any Graph
- Macrotized your Customized Automation

Review and Conclusion

What can we conclude?

- You now have the tools to be smarter in your occasional or daily SAS Graph life.

- You can apply these methods to other SAS procedures and become a SAS supa star!

Thank YOU for the privilege of your time!

Contact Information

Questions?

Jesse A. Canchola

Address:

Roche Molecular Systems, Inc.

4300 Hacienda Drive

Pleasanton, CA 94588 USA

Email:

Jesse.Canchola@Roche.Com

References

References

SAS References

- Heath D (2009). "Paper 324-2009: Secrets of the SG Procedures". *Proceedings of the SAS Global Forum 2009 Conference*. Washington, DC: The SAS Institute, Inc., Cary, NC. Available at https://support.sas.com/resources/papers/proceedings09/324-2009.pdf

- Matange S (2011). "Paper 281-2011: Tips and tricks for clinical graphs using ODS graphics". *Proceedings of the SAS Global Forum 2011 Conference*. Las Vegas, NV: The SAS Institute, Inc., Cary, NC. Available at https://support.sas.com/resources/papers/proceedings11/281-2011.pdf

- Matange S (2016). "PharmaSUG 2016 – Paper DG02: Clinical graphs using SAS". *Proceedings of the PharmaSUG 2016 Conference*. Denver, CO: The SAS Institute, Inc., Cary, NC. Available at https://support.sas.com/resources/papers/proceedings16/SAS4321-2016.pdf

- Oltsik M (2008). "ODS and Output Data Sets: What you need to know". *Proceedings of the SAS Global Forum 2008 Conference*. San Antonio, TX: The SAS Institute, Inc., Cary, NC. Available at http://www2.sas.com/proceedings/forum2008/086-2008.pdf

- Slaughter SJ, Delwiche LD (2012). "Paper 259-2012: Graphing made easy with SG Procedures". *Proceedings of the SAS Global Forum 2012 Conference.* Orlando, FL: The SAS Institute, Inc., Cary, NC. Available at http://support.sas.com/resources/papers/proceedings12/259-2012.pdf

- Slaughter SJ, Delwiche LD (2015). "Paper 2441-2015: Graphing made easy with SGPLOT and SGPanel Procedures". *Proceedings of the SAS Global Forum 2012 Conference.* Washington, DC: The SAS Institute, Inc., Cary, NC. Available at https://support.sas.com/resources/papers/proceedings15/2441-2015.pdf

Doing now what patients need next

Side Dishes

(aka Asides)

SAS and Graphics Types

Side Dishes

SAS and Graphics Types

As indicated early on in our presentation, SAS can produce **both** major graphic types:

- Bitmap (aka *raster or static graphics*; e.g., types include JPG/JPEG, PNG, APNG, GIF)
  - graphs made up of pixels;
  - editing software includes GIMP, Vectr, Adobe Photoshop;
  - preferred by graphics artists to highlight their creativity;
  - standard in SAS v9.3 and below.
- Vector (e.g., types include SVG, EPS, PS, PDF)
  - graphs made up of lines and curves that can be moved around;
  - editing software includes InkScape, Gravit Designer, Adobe Illustrator;
  - preferred by graphics designers and editors for ease of editing;
  - standard in SAS v9.4 and later (depending on output destination).

Side Dishes

SAS and Graphics Types

Beginning with SAS v9.4, SAS defaults to vector graphics if the right graphic output type is indicated. With SAS v9.3 or prior, you can specify a vector graphic type. [see: https://bit.ly/2LBkVoQ for more information]

CAVEAT: However, even if the output type indicated is a vector graphic type, it is no guarantee that you will get an actual vector graphic in the graphic "container" (e.g., pdf, svg, ps, eps, etc.) that you choose (even if it is a traditional vector graphic container type as noted above)!

Side Dishes

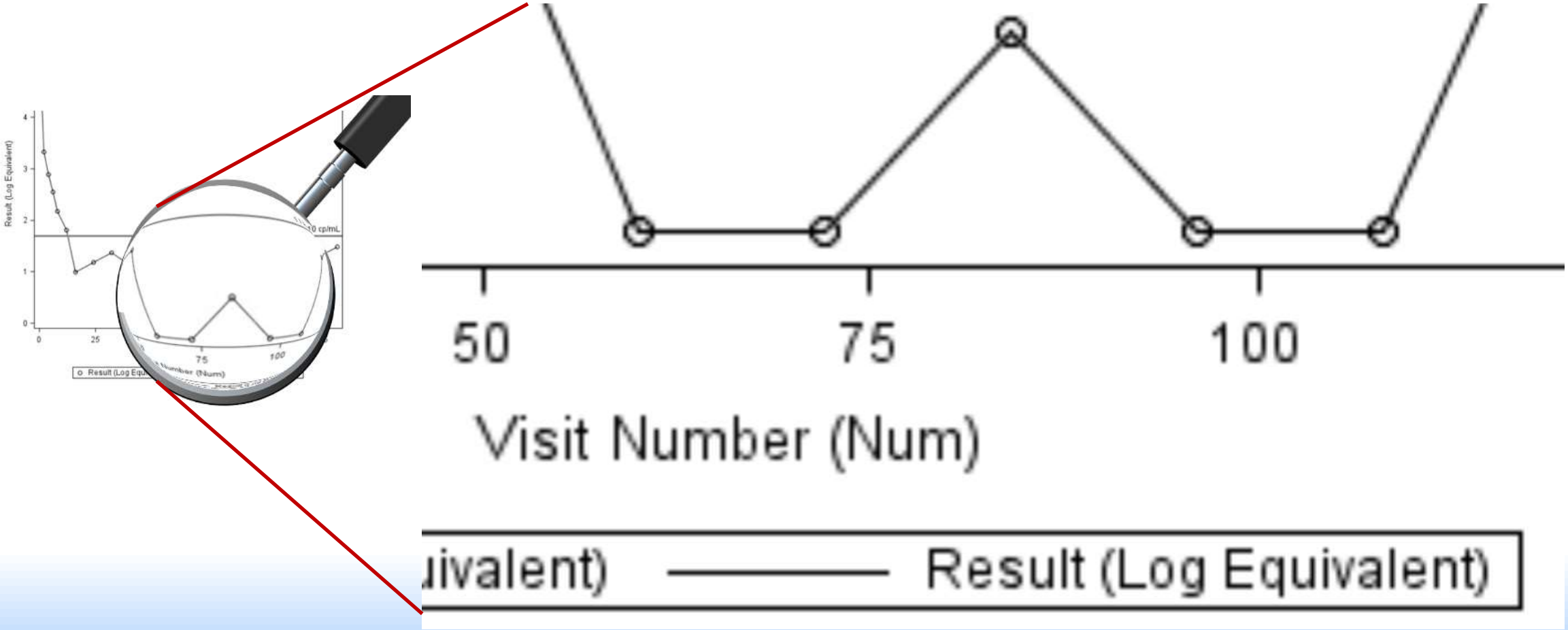
SAS and Graphics Types

TIP: The only real visual way to check if you have a vector graphic is to …

- **increase the magnification of your graphic in whatever application you are using to the highest level and**
  - **if there is pixellation (i.e., loss of resolution), *you have a bitmap*;**
  - **If you see no pixellation, and high resolution smooth lines and curves, *you have yourself a vector graphic*.**

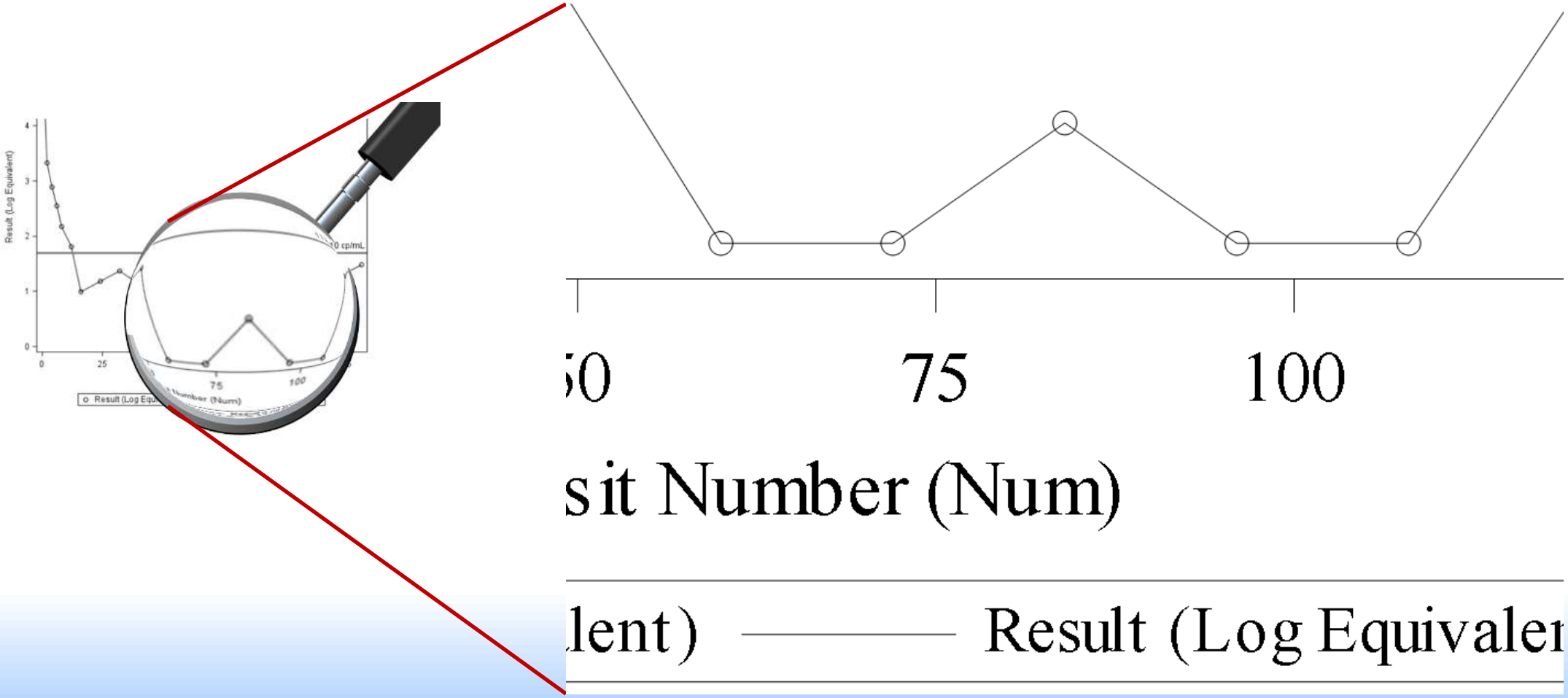
Side Dishes

SAS and Graphics Types: Bitmap



Side Dishes

SAS and Graphics Types: Vector



Back to Sanity…

Doing now what patients need next