PharmaSUG 2016 - Paper QT14

SAS® and R - stop choosing, start combining and get benefits!

Diana Bulaienko, Experis Clinical, Kharkiv, Ukraine

ABSTRACT

The R software is powerful but it takes a long time to learn to use it well. However, you can keep using your current software to access and manage data, then call R for just the things your current software doesn't do. Learning to use a data analysis tool well takes significant effort and quite a lot of time, so this paper will be valuable to experienced SAS users who don't want totally switch to R, but want to use advantages of both SAS and R software.

This presentation introduces a minimal amount of R commands you will need to know to work this way. It will also describe how to call R routines from SAS to create beautiful graphs.

INTRODUCTION

In our days more and more pharmaceutical companies use R for their needs in data management and analysis. Currently R and SAS are often used side by side for similar tasks, and it's becoming more and more likely that you as a SAS programmer will be asked to provide data to someone using R or receive data from an R user to be integrated back into the SAS environment.

Both SAS and R can perform data management and create subsets and both of them have their own advantages and disadvantages. Data manipulation is much easier in SAS while generation of graphics can be easier in R. This paper will assume the data management is done in SAS. Both SAS and R can run basic statistical procedures and create graphs. You could argue that one language is better than the other for this, or at least that specific procedures are better on the SAS or R side. But we know for sure that some people prefer to run graphs in R. For purposes of illustration, in this paper I will first create the graph entirely in SAS and then I will create the same graph with the help of R statements within the same SAS program.

Interaction between SAS and R is not new. Users have done this before, but the solution was not trivial and required exporting SAS data to a delimited text file. With the SAS/IML interface, the process of exchanging data has become much simpler and the execution of code is more interactive. The SAS/IML interface to R allows users to take advantage of R within the SAS environment. Using SUBMIT/ENDSUBMIT statements in IML procedure and a few CALL routines, you can create R data frames from SAS data sets and execute R statements within your SAS programs and take advantage of both SAS and R softwares.

COUPLE OF WORDS ABOUT R

The R environment with its R language is a freely available open source software tool which focuses on statistical computing and graphics. You can download it from http://www.r-project.org. Written by Robert Gentleman and Ross Ihaka, R is based upon the S language developed by John Chambers and others at Bell Labs in the 1970's.

Before reading any of the example programs below, you'll need to know a few things about R. This is not a paper on how to program in R, but there are some basic knowledge needed in order to understand what is going on.

R has multiple object types that store data including data frames, matrices, and lists. The R data frame is most similar to the SAS data set, and in this paper we'll see how to create graph based on data frame. R uses the terms 'columns' and 'rows' instead of 'variables' and 'observations', but this paper I will use the SAS terminology. Case matters in R: for variable names, function names, and just about everything else.

The one of the most attractive features of R – is the R package system, which includes a voluminous collection of packages developed by the user community. Actually, the R environment consists of a small core program with the R interpreter and runtime environment and a set of preinstalled packages for all higher programming functions, statistical algorithms and graphics. In addition to being a convenient way to create collections of tools, R packages provide a framework for documentation that makes the tools easier to use. Most packages tend to have a very narrow scope, but may contain a large number of tools to perform a specific analysis or accomplish a specific task.

THE EASIEST WAY OF RUNNING R FROM SAS

As a SAS user, the easiest way to run R is through SAS/IML interface with the help of procedure PROC IML and its features to integrate the R language.

The SAS Interactive Matrix Language (IML) is a programming language for explorative data analysis. It implements a wide range of functions for most standard matrix operations. In addition SAS IML implements the interface to R which can be accessed from SAS via PROC IML. This paper focuses on the SAS procedure PROC IML and its features to integrate the R language. Special features of SAS/IML as well as the SAS/IML Studio which provides a dynamic and interactive interface to SAS/IML is not taken into account.

The SAS/IML interface enables you to embed tabular output from R into SAS reports and to transfer matrices and data frames from R into SAS. You can display R graphics in the native R graphics window or tell R to write its graphics as an image file with the help of PNG() function. SAS data sets can be easily send to R and back via the so called built-in subroutines – ExportDataSetToR, ExportMatrixToR, ImportDataSetFromR and ImportMatrixFromR, which is implemented within IML. The SUBMIT and ENDSUBMIT statements define sections of code to be executed outside PROC IML. The area between these two statements is called a SUBMIT block. Any code within a SUBMIT block will be executed in R. The end result is a process half-way between line-by-line execution and batch execution. The R workspace stays open until you execute the QUIT statement and the results of multiple submit blocks are cumulative within an IML session.

The following is an example. The # symbol is used to begin a comment, the + symbol appears for the continuation of the command.

```
proc iml;
  run ExportDatasetToR(“mydata”);     # this command sends your data set to R
  submit/r;                           # start executing R code
  attach(mydata)                      # the first R statement
  install.packages(“ggplot2”)         # package installation. do this one time
  library(ggplot2)                    # package loading. do this every time
  ggplot(students, aes(x = Level))
  + geom_bar()                        # the last R statement
  endsubmit;                          # stop executing R code
quit;                                 # executing the QUIT statement would terminate
                                        the IML and R sessions and close the plot
                                        window.
```

STEPS YOU NEED TO DO TO MAKE SAS CO-WORK WITH R

Fortunately, the setup is pretty easy.

First of all you need to install R. R is available as a download from the Comprehensive R Archive Network, or CRAN, at http://www.r-project.org. According to http://blogs.sas.com/content/iml/2013/09/16/what-versions-of-r-are-supported-by-sas/ with SAS 9.3 you can only use R 2.9.1 – 2.15.3 and if you upgrade to SAS 9.4, you can use up to R 3.0.1. We will run R-3.0.1 using SAS 9.4 (Foundation).

By default SAS forbids to call R functionalities. This is specified by the systems option NORLANG. You should check if your system was set up to read the R language (code below):

```
proc options option=rlang;
run;
```

If you get the following printed to the SAS log:

```
RLANG        Support access to R language interfaces
```

then everything is fine, and you could execute R commands from SAS.

But if you get in the SAS log:

```
NORLANG      Do not support access to R language interfaces
```

this means you need to add the -RLANG option to the SAS config file. Below is an example of the config file (C:\Program Files\SASHome\SASFoundation\9.4\sasv9.cfg):

```
-RLANG
-config "C:\Program Files\SASHome\SASFoundation\9.4\nls\en\sasv9.cfg"
(NOTE: the -RLANG had to be above the config reference for this to be recognized
properly.)
```

After changing the SAS config file you should re-open SAS and re-run the proc options code above and ensure that now SAS supports access to R language interfaces.

It might be that SAS may also require some additional stuff in the SAS config file to allow R to run and recognize where it is on your computer.

For example, you try running the following code:

```
proc iml;
 submit/r;
 install.packages(“ggplot2”)
 library (ggplot2)
endsubmit;
quit;
```

and you get the following error in the log:

```
ERROR: SAS could not initialize the R language interface.
```

Then from SAS submit following code:

```
%put %sysget(R_HOME);
```

and check if the correct path is returned? If not, you'll need to modify SAS config file to define where R_HOME environment variable is on your computer. Below is an example of the code which you should add to the config file “C:\Program Files\SASHome\SASFoundation\9.4\sasv9.cfg”:

```
-SET R_HOME "C:\Program Files\R\R-2.15.0"
```

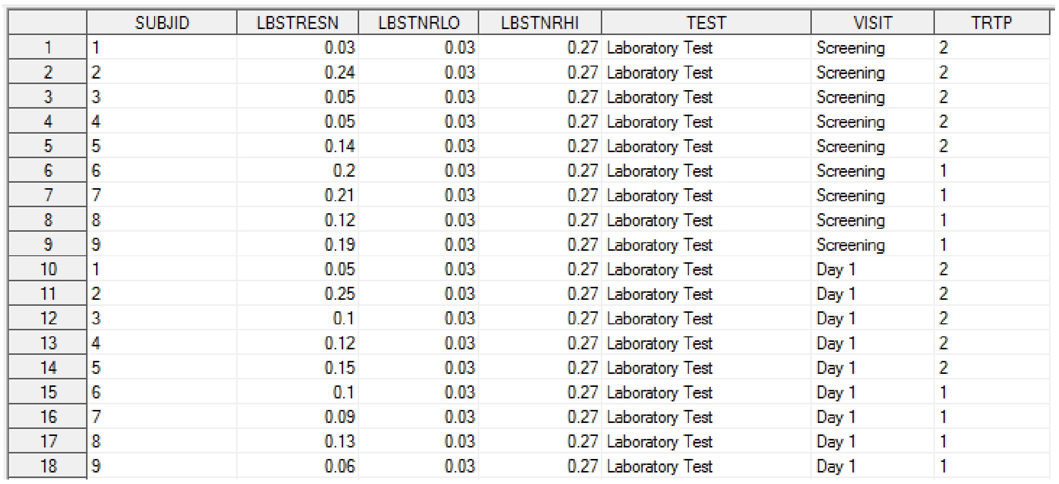
After all these manipulations with updating SAS config file you will allow SAS to read the R language and execute R commands directly from SAS and this is the easiest way of working with SAS and R together.

SAS AND R CO-WORK ON THE EXAMPLE OF CREATING GRAPH

I’ll start with drawing a plot with SAS/Graph and then I’ll draw a plot with the help of R on the SAS database.

Both R and SAS offer many different graphic output formats. In our example I have chosen to write to a png file format. The PNG format is lossless and is best for line diagrams and blocks of colour.

For my examples I have chosen to use a SAS data set named “plot_data”. The data set contains made-up data with one record per patient, per lab.test, per visit and is intended to simulate a clinical patient study where the outcome of interest is the level of some laboratory test result during the study. The “plot_data” data set contains 81 records and the first 18th are shown below in Display 1.



	SUBJID	LBSTRESN	LBSTNRLO	LBSTNRHI	TEST	VISIT	TRTP
1	1	0.03	0.03	0.27	Laboratory Test	Screening	2
2	2	0.24	0.03	0.27	Laboratory Test	Screening	2
3	3	0.05	0.03	0.27	Laboratory Test	Screening	2
4	4	0.05	0.03	0.27	Laboratory Test	Screening	2
5	5	0.14	0.03	0.27	Laboratory Test	Screening	2
6	6	0.2	0.03	0.27	Laboratory Test	Screening	1
7	7	0.21	0.03	0.27	Laboratory Test	Screening	1
8	8	0.12	0.03	0.27	Laboratory Test	Screening	1
9	9	0.19	0.03	0.27	Laboratory Test	Screening	1
10	1	0.05	0.03	0.27	Laboratory Test	Day 1	2
11	2	0.25	0.03	0.27	Laboratory Test	Day 1	2
12	3	0.1	0.03	0.27	Laboratory Test	Day 1	2
13	4	0.12	0.03	0.27	Laboratory Test	Day 1	2
14	5	0.15	0.03	0.27	Laboratory Test	Day 1	2
15	6	0.1	0.03	0.27	Laboratory Test	Day 1	1
16	7	0.09	0.03	0.27	Laboratory Test	Day 1	1
17	8	0.13	0.03	0.27	Laboratory Test	Day 1	1
18	9	0.06	0.03	0.27	Laboratory Test	Day 1	1

Display 1. Analysis data set “plot_data”

Below is SAS graph program which creates Individual Laboratory Test Results plot with high/low reference range limit lines and annotated outliers :

```
data anno;
   length label $20 x1space y1space $11 anchor $8;
   retain y1space 'datavalue' x1space 'datavalue' textcolor 'red' width 25;

   set plot_data;

   if LBSTRESN>LBSTNRHI  then do;
     function='text';
     x1=visitnum;
     y1=LBSTRESN+.001;
     anchor='right';
     label=trim(left(put(LBSTRESN,6.2)));
     output;
    end;

    if LBSTRESN<LBSTNRLO then do;
     function='text';
     x1=visitnum;
     y1=LBSTRESN-0.01;
     anchor='right';
     label=trim(left(put(LBSTRESN,6.2)));
     output;
    end;

     function='text';
     x1=2;
     y1=LBSTNRHI+.01;
     anchor='right';
     label="Upper Limit";
     output;

     function='text';
     x1=2;
     y1=LBSTNRLO-.01;
     anchor='right';
     label="Lower Limit";
     output;

run;

ods graphics / reset=index imagename='SAS_example' imagefmt=png border=off;
ods listing gpath="e:/…/";

proc sgplot data=plot_data sganno=anno;

  series x = visitnum y = LBSTRESN / name="s0"  group = SUBJID  markers
                                  lineattrs = (pattern = solid);
  refline LBSTNRHI/axis=y lineattrs=(color=red pattern = dash) label="Upper Limit";
  refline LBSTNRLO/axis=y lineattrs=(color=red pattern = dash) label="Lower Limit";
  yaxis label = "Individual Laboratory Test Results" values=(1 to 0.4 by 0.05)
grid;
  xaxis label = "Visit"  values=(1 to 9 by 1) grid  TICKVALUEFORMAT= vis.;
  keylegend "s0" / title = "Subject Number:"  noborder;
  title1 j=c "Individual Laboratory Test Results";

run;

ods graphics off;
ods listing close;
```

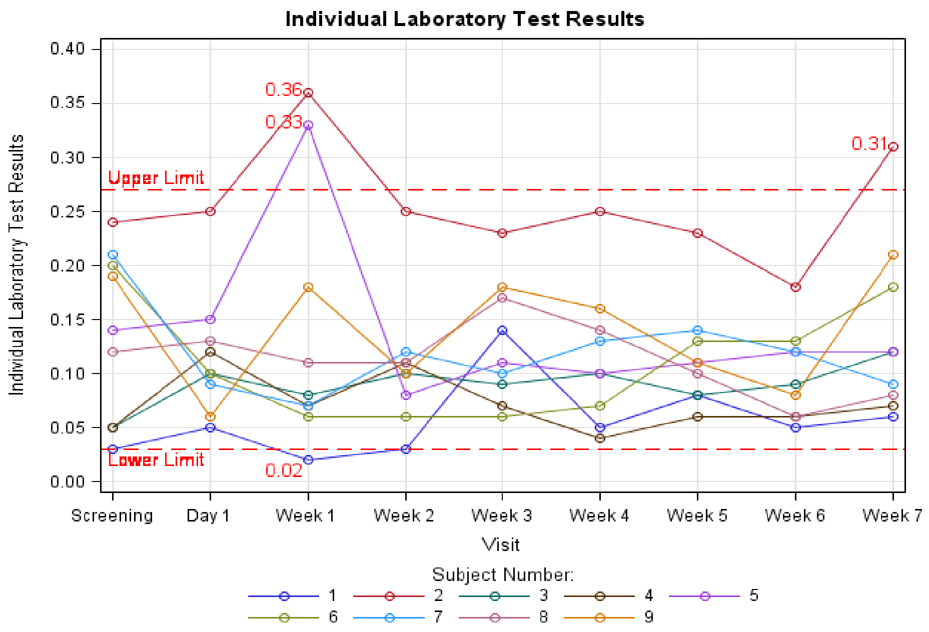


Figure 1. Subjects Individual Laboratory Test Results made with SAS

Now let's do the same plot on R. One of the things that make R such a great tool is its data visualizations capabilities. For performing visualizations in R, ggplot2 is probably the most well known package and a must learn for beginners. You can find all relevant information to get you started with ggplot2 on http://ggplot2.org/. To make it easy to get started, the ggplot2 package offers one of main functions: ggplot().

Package ggplot2 doesn't not come with the basic R installation and it needs to be installed. Please note, that packages only need to be installed once, but in order to be used, a package must be loaded each time a new R session is opened. Run the following commands to load the necessary package. The # symbol is used to begin a comment. When a command does not fit on a single line, a + sign appears for the continuation of the command.

```
proc iml;
  submit/r;                     # start executing R code
  install.packages(“ggplot2”)   # package “ggplot2” installation. do this one time
  endsubmit;                    # stop executing R code
quit;
```

Below is R graph program, which creates the same graph as above:

```
proc iml;
 run ExportDataSetToR("plot_data", "rds"); # sends SAS data set “plot_data” to R
                                             data frame “rds”.
 submit / R;
 attach(rds)
 library(ggplot2)
 png(file="e:/…/R_example.png", bg="transparent", width=700, height=500)

 ggplot(rds,aes(x=visitnum, y = LBSTRESN, group = SUBJID ))+

 geom_line(aes(colour = SUBJID,linetype=factor(TRTP, labels = c("Treatment 1",
          "Treatment 2"))), size=0.7)+
 geom_point(aes(colour = SUBJID, shape=factor(TRTP,labels = c("Treatment 1",
        "Treatment 2"))), size=2)+ scale_shape_manual(values = c(17,22))+

 geom_hline(aes(yintercept = LBSTNRHI), color="red", linetype="dashed", size=0.8) +
 geom_hline(aes(yintercept = LBSTNRLO), color="red", linetype="dashed", size=0.8) +

 labs(color = "Subjects number:",shape = "Treatment groups:", linetype = "Treatment
     groups:")+
 annotate("text", x = 1.1, y = LBSTNRHI+0.01, label = "Upper Limit",colour = "red",
          size=3)+
 annotate("text", x = 1.1, y = LBSTNRLO-0.01, label = "Lower Limit",colour = "red",
          size=3)+
 scale_x_discrete(name="Visit",limits=c("Screening", "Day 1", "Week 1", "Week 2",
                 "Week 3" ,"Week 4", "Week 5", "Week 6", "Week 7")) +
 scale_y_continuous(name="Laboratory Test Results",breaks=seq(0, 0.4, 0.05))+

 geom_text(aes(label=ifelse((LBSTRESN>LBSTNRHI|LBSTRESN<LBSTNRLO),LBSTRESN,"")),
               hjust=1.2, size=4, color="red")+
 ggtitle("Individual Laboratory Test Results")+

 theme(plot.title = element_text(face="bold", size = 12, vjust=2),
       legend.position="bottom",
       axis.text.x = element_text(angle=45, vjust=0.5,size=10,colour="black"),
       axis.title.y=element_text(vjust=1.5),
       axis.text.y = element_text( colour="black",size=10 ),
       panel.background = element_rect(fill = "white", colour = NA))
       panel.grid.major = element_line(colour = "grey90"),
       panel.border =  element_rect(fill = NA, colour="grey80"),
 dev.off()
 endsubmit;
quit;
```

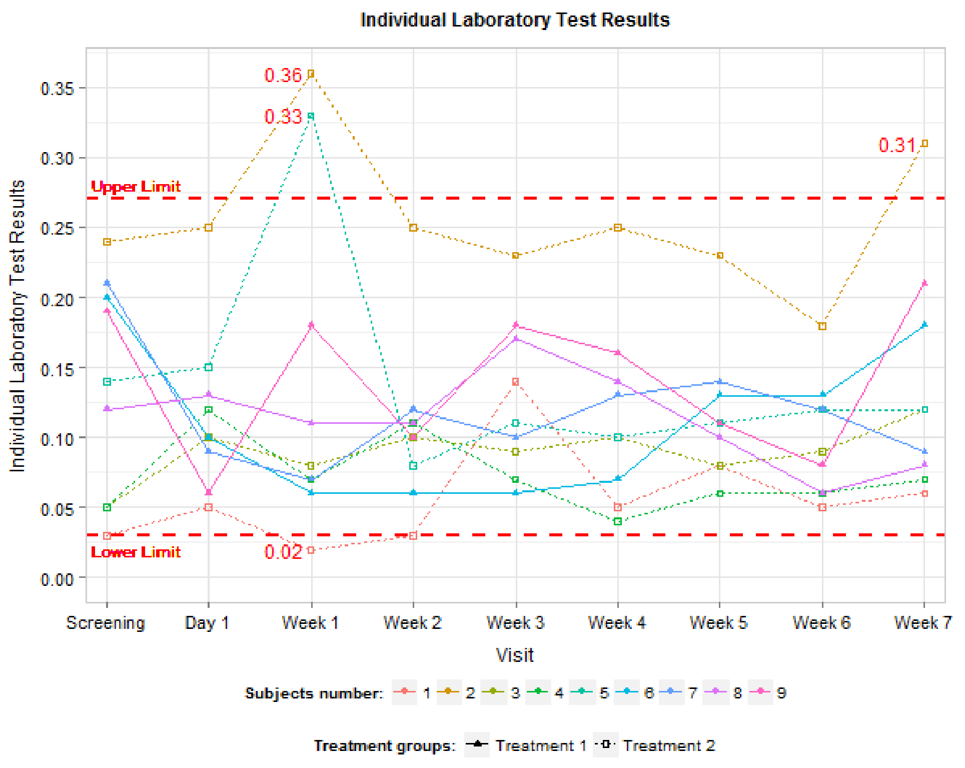


Figure 2. Subjects Individual Laboratory Test Results made with R

I found that it takes more efforts and more code lines to create this type of graph using traditional SAS/graph model. However, it is much easier to do this type of graph in R and as you can see I have even complicated the graph, making line type different for each treatment group, and it takes me less time. The R-blogger website (http://www.r-bloggers.com/) and website http://r4stats.com/ contain news, tutorials and a lot of R code examples contributed by a large number of users, and these website really help to find answers on your questions concerning R programming. The R program offers a greater degree of flexibility to manipulate individual graph elements similar to SAS/Graph and is a useful alternative to the present SAS strategies.

CONCLUSION

Even though SAS is a powerful statistical analysis software and a lot of great improvements have been made to match industry need, it is not realistic to depend only on SAS to solve all the challenges. Thus, some new features in SAS might not be available in our daily work. There is great need to look beyond SAS and investigate tools that could be used by SAS solve this problem. R is a good choice to be used by SAS for several reasons. First, R is a language and free software environment for statistical computing and graphics. Second, R provides a wide of variety of statistical and graphical techniques, including some very powerful graphics packages such as grid, ggplot2 and etc.

Given that you already have a statistics package at your disposal, the amount of R you need to learn to use an occasional R function is fairly minimal. The return on this investment of effort can be great: access to many thousands of additional analytical and graphical methods. Once you see what R offers, perhaps you will become interested in learning more about it. That's why using both technologies SAS and R to improve your data analysis and visualizations capabilities seems like the winning solution for all.

So, stop choosing, start combining and get benefits!

REFERENCES

The R Project for Statistical Computing, URL: http://www.r-project.org

Rick Wicklin. "What versions of R are supported by SAS?". URL: http://blogs.sas.com/content/iml/2013/09/16/what-versions-of-r-are-supported-by-sas/

Clarke Thacher, "Make Your SAS® Code Environmentally Aware". URL: http://support.sas.com/resources/papers/proceedings10/090-2010.pdf

Bob Muenchen,"R FOR SAS AND SPSS USERS". URL: https://science.nature.nps.gov/im/datamgmt/statistics/r/documents/r_for_sas_spss_users.pdf

For details and syntax of ggplot2, URL: http://ggplot2.org

A guide to getting started in R for SAS programmers, URL: http://r4stats.com

Examples of tasks replicated in SAS and R, URL: http://sas-and-r.blogspot.com/p/graphics-examples.html

RECOMMENDED READING

- SAS and SPSS users can learn more about R by reading the free version of R for SAS and SPSS Users, by Robert A. Muenchen, available at http://r4stats.com/books/free-version. That document focuses on data management and basic statistics, providing much of its explanation in the form of program comments.
- SAS users will also enjoy the book, "R and SAS", by Ken Kleinman and Nicholas J. Horton. That book offers brief descriptions of how to use R for a wide range of tasks.
- "R Language Definition", URL: http://cran.r-project.org/doc/manuals/r-release/R-lang.pdf

CONTACT INFORMATION

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

Name: Bulaienko Diana
Enterprise: Experis Clinical
Address: 43/2 Gagarin av.
City, State ZIP: Ukraine, Kharkiv, 61001
Work Phone: +38 057 755 7020 ext. 2410
Fax: +38 057 728 30 35
E-mail: diana.bulaienko@roche.com