PhUSE 2012

Paper CC02

Superior gRaphics in Statistical Reports

Sameer Bamnote, Cytel Statistical Software and Services Pvt. Ltd., Pune, India

ABSTRACT

This paper is aimed at users who know both SAS® and R, and are interested in exploiting best of both the environments.

Complex graphics are sometimes easier done in R than in SAS e.g. multiple graphs in a single plot frame, annotation, jittering etc. An R user has control over every aspect of the graph. R graphics are superior and visually appealing. Since we can call R through SAS, a good strategy is to create the graphs using R from SAS. Final statistical reports can be a seamless combination of R and SAS outputs generated through a sole SAS call. This is illustrated with some examples in the paper.

INTRODUCTION

Graphics play an important role in clinical trials not only in the reporting but also in exploratory analysis. Statistical graphics play a vital role in the interpretation of the data which further helps in drawing conclusions. SAS is widely used to generate tables, listings and figures to analyze clinical trial data. This paper is aimed at users who know SAS and are aware of R to a certain extent, and who are interested to exploit best of two worlds. The objective of the paper is to use both the environments (SAS and R) in tandem to prepare statistical reports.

Since now we can invoke R through SAS, the proposal discussed in this paper is –

1. To use SAS for generating tables or listings
2. Use R for generating figures, because some of the complex parts of the graphics can be easily handled in R

So the final statistical report is the combination of tables and listings generated in SAS and figures in R. This paper mainly focuses on generating graphs using R through SAS which is illustrated with the help of several examples.

After going through the paper, readers will have a better grasp of –

1. How R can be invoked from SAS
2. How to insert figures generated in rtf file in R
3. Various capabilities of the R graphics

SOFTWARE ENVIRONMENT

1. Examples described in this paper use Windows 7 [but any platform compatible with SAS and R can work]
2. Base SAS version 9.1
3. R 2.15.1 with Base package and some non-standard libraries (Hmisc, sciplot, gplots, plotrix, ggplot2, psych, rtf, grid). The rtf package in R is recently developed which helps in creating .rtf or .doc files.

CALLING R FROM SAS

SAS users are usually more comfortable in the SAS environment rather than working in any other new environment. So it is good idea to invoke R from SAS. But is this possible? The answer is yes. There are some ways through which R can be called in SAS:

1. Through “Proc IML” R can be called in SAS 9.22 or later and SAS/IML Studio 3.2 which requires separate license.

2. SAS2R2SAS a paper by Philip R Holland[1] describes a method where R code is executed by calling the R system inline-command mode using the SAS X statement. The methods can be used in any version of Base SAS from version 7 onwards.
3. The method which I am discussing is posted by Liang Xie[2] in his article 'Conduct R analysis within SAS'; this seems to be a good way of calling R from SAS. This includes defining two macros in SAS as-

```
%macro RScript(Rscript);
     data _null_;
     file "&Rscript";
     infile cards;
     input;
     put _infile_;
%mend;

%macro CallR(Rscript, Rlog);
        systask command "C:\Progra~1\R\R-2.15.1\bin\R.exe CMD BATCH --vanilla --quiet
                    &Rscript  &Rlog "
         taskname=rjob1  wait  status=rjobstatus1;
%mend;
```

The first macro RScript – is to write the R code using cards statement in SAS. The second macro CallR – invokes the R from SAS and stores its log. While invoking R from SAS we need to provide the path where the R executable file (.exe) is stored on the computer. The R runs in the backhand so user will not be able to see R console.

Now let us see how to write code within this RScript macro and how to look into R console.

```
%RScript(c:\rscript.r)
cards4;

  <write R code here>

;;;;
run;

%CallR(c:/rscript.r, c:/rlog1.txt);

/*Printing R log in the sas log window*/

data _null_;
     infile "c:\rlog1.txt";
     input;
     put _infile_;
run;
```

We will write the R codes in the yellow highlighted region above in the Rscript macro and specify the path where this R code needs to be save. Then we will specify the path where R code is stored and where R log needs to saved to the CallR macro. To check whether R code is error free we print the R log in the SAS log window by creating a null dataset.
If we are dealing with SAS dataset then we can either export the SAS dataset as CSV file and then read it in R or we can use the 'sas7bdat' package to directly read the SAS dataset in R.

GENERATING RTF FILES IN R

Statistical reports are generally prepared in word documents either as .doc file or .rtf file. The R2Wd package in R helps in creating reports in word format. It also creates headers, sub headers and other functionalities. But this package relies on rcom. It is a wrapper that uses the statconnDCOM server to communicate with MS-Word via the COM interface.

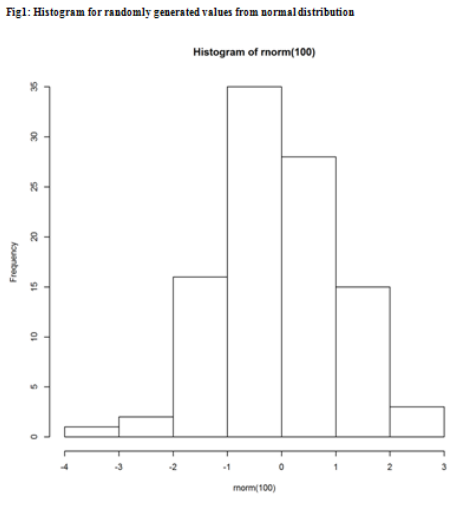
As per the recent development there is package called "rtf", which can be used to output Rich Text Format (RTF) files with high resolution tables and graphics that may be edited with standard word processors.

Let us see one illustration where an rtf file is generated with two plots being added to it. For simplicity both are the histograms of random numbers generated from normal distribution. The first one is generated and added to rtf using "addPlot" function from 'rtf' package of R. Second one is generated and saved as png file. Later this png file is added to rtf using "addPng" function present in the 'rtf' package.

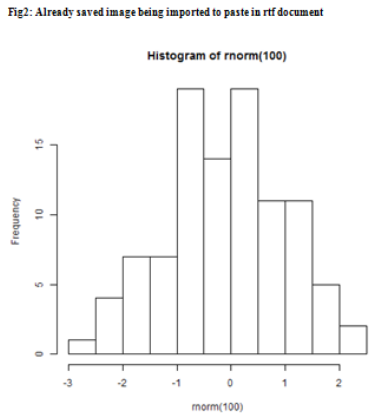
R CODE:

```
#Generating histogram using random numbers
png(file = "D:/myplot1.png", bg = "transparent")
hist(rnorm(100))
dev.off()

#Creating rtf file
library(rtf)
rtf<-RTF("D:/test_addPlot.doc",width=8.5,height=11,font.size=10,omi=c(1,1,1,1))
addHeader (rtf,title="Fig1: Histogram for randomly generated values from normal
distribution")
addPlot(rtf,plot.fun=hist,width=6,height=6,res=300,rnorm(100))

#Adding png plot to the rtf file
addPageBreak(rtf, width=8.5, height=11, omi=c(1, 1, 1, 1))
addHeader (rtf,title="Fig2: Already saved image being imported to paste in rtf
document")
addPng(rtf,"D:/myplot1.png", width =5, height = 5)
done(rtf)
```

Output:



PAGE 1



PAGE 2

As we can see from the code and above output that we can adjust the dimensions of the region where the figures can be placed in the rtf file using this package. We can add text, header, PNG files. We can also add table in the rtf file with this package, let's see an illustration where one figure and some observations from the existing R data frame (IRIS data) are listed in the form of the table.

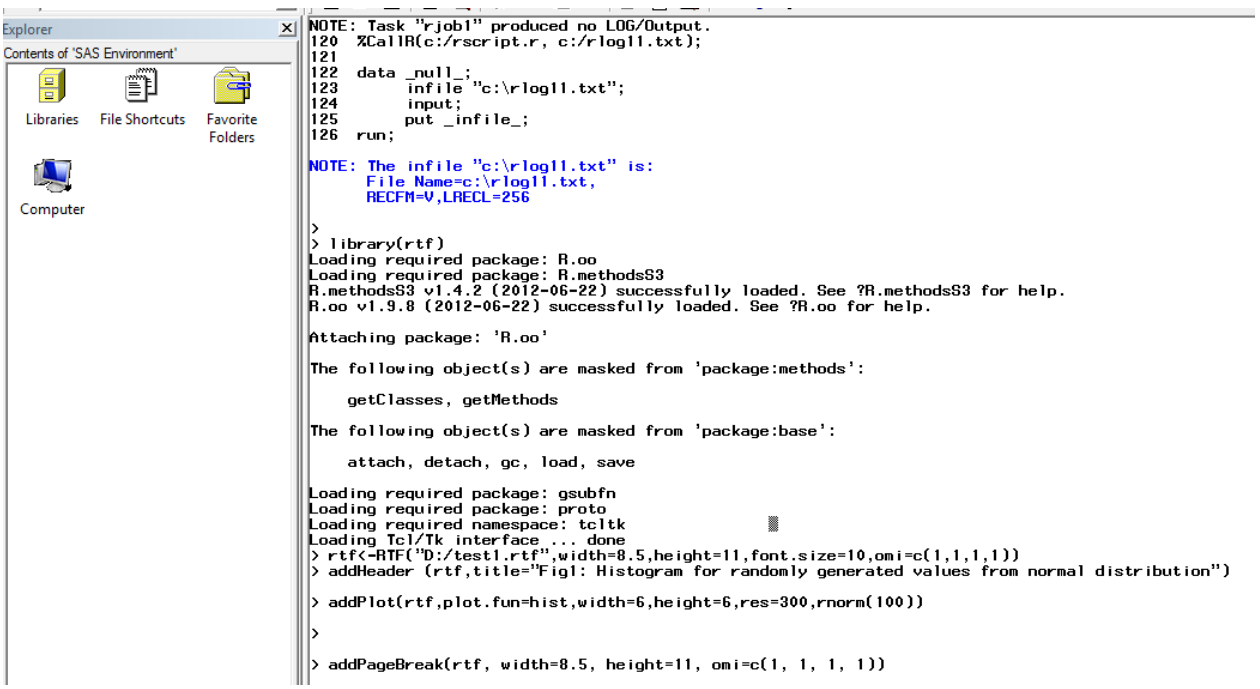
R CODE:

```
%macro RScript(Rscript);
data _null_;
     file "&Rscript";
```

```
     infile cards;
     input;
     put _infile_;
%mend;

%macro CallR(Rscript, Rlog);
systask command "C:\Progra~1\R\R-2.15.1\bin\R.exe CMD BATCH --vanilla --quiet
                  &Rscript  &Rlog "
       taskname=rjob1  wait  status=rjobstatus1;
%mend;

%RScript(c:\rscript.r)
cards4;

library(rtf)
rtf<-RTF("D:/test1.rtf",width=8.5,height=11,font.size=10,omi=c(1,1,1,1))
addHeader (rtf,title="Fig1: Histogram for randomly generated values from normal
distribution")
addPlot(rtf,plot.fun=hist,width=6,height=6,res=300,rnorm(100))

addPageBreak(rtf, width=8.5, height=11, omi=c(1, 1, 1, 1))
addHeader (rtf,title="Table1: First 10 observations of IRIS data present in R")
addTable(rtf, dat=iris[1:10,])
done(rtf)
;;;;
run;

%CallR(c:/rscript.r, c:/rlog11.txt);

data _null_;
     infile "c:\rlog11.txt";
     input;
     put _infile_;
run;
```

Let's have a look how R log (R Console) looks like in SAS log window.

R Log:



Output:

Fig1: Histogram for randomly generated values from normal distribution

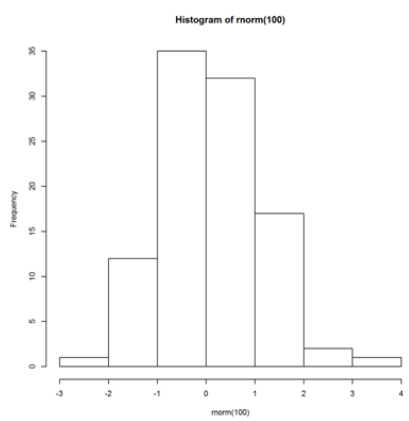


Table1: First 10 observations of IRIS data present in R

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

VARIOUS FEATURES OF R GRAPHICS

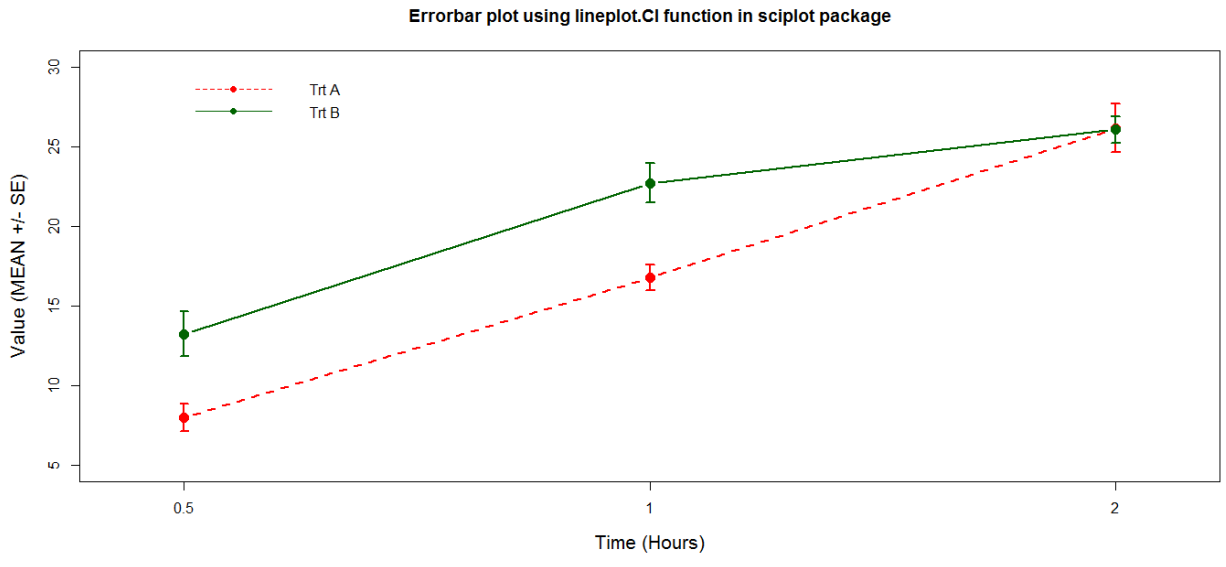
SAS users are well aware of how many efforts they spend on coding for graphics especially jittering (when values are overlapping) and annotation e.g. error bar plots, individual values plot. Now we will discuss couple of straight forward examples where we will see how R can handle both these aspects with minimal coding.

ERROR BAR PLOT

The data used to illustrate error bar plot is the inbuilt data in R named 'ToothGrowth' whose variables are just renamed (as rep, trt and time) in clinical trials terms. There are several functions available across the various packages of R with the help of which error bar plots can be created.

Ex. 1 Error bar plot using 'lineplot.CI' function from 'sciplot' package

Output

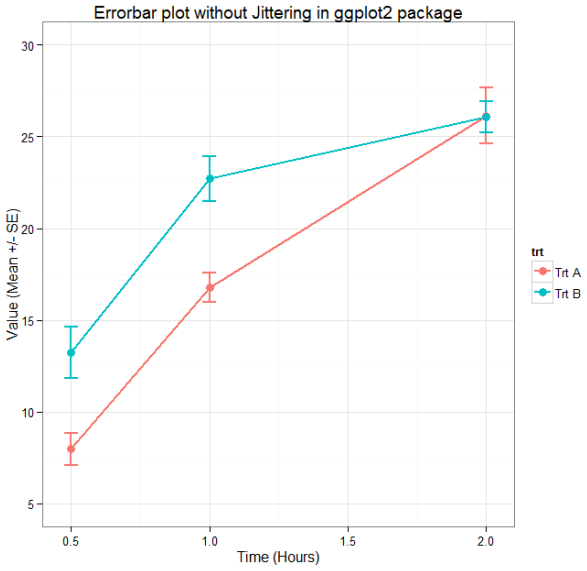


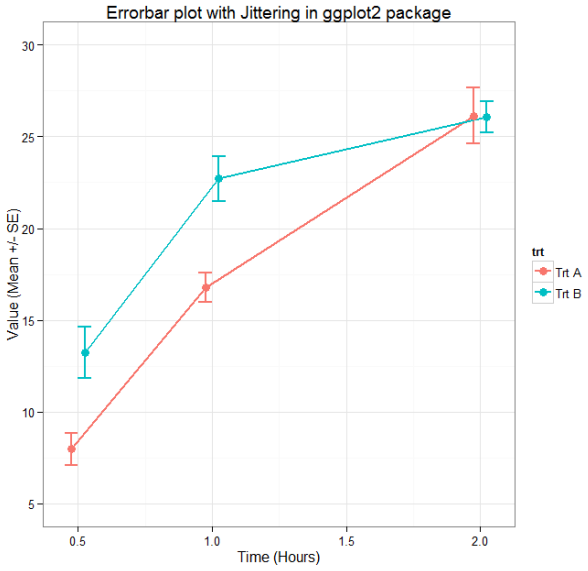
R Code

```
lineplot.CI(time, rep, group = trt, data = tg, cex = 1.2,
            xlab = "Time (Hours)", ylab = "Value (MEAN +/- SE)", cex.lab = 1.3, x.leg
            = 1,y.leg=30,col = c("red","dark green"), pch = c(16,16),ylim=c(5,30),
            err.width = 0.05, xaxt = 'n')
axis(1,at=c(1,2,3),labels=c(0.5, 1, 2))
```

In the example discussed above, we need to pass the data without computing summary statistics (e.g. mean, standard error etc.) to the function 'lineplot.CI'. In the displayed output above, we can see the means and error bars are overlapping for time 2 hours. To avoid this we need to jitter (separate or distinguish) overlapping values. Let us see the similar figures generated using 'ggplot2' package of R and we will also see how simple it is to do jittering in R. In 'ggplot2' we need to pass the summary dataset (where summary statistics like mean, standard error are already calculated).

Output:





R Code without Jittering

```
ggplot(summary, aes(x=time, y=rep,
colour=trt)) + geom_errorbar(aes(ymin=rep-
se,ymax=rep+se),width=.05) + geom_line() +
geom_point() + xlab ("Time(Hours)") + ylab
("Value (Mean +/- SE)")+ ggtitle("Errorbar
plot without Jittering in ggplot2
package") + theme_bw()+
scale_y_continuous(limits=c(5,30),
breaks=0:30*5)
```

R Code with Jittering

```
pd = position_dodge(.1) #Jittering
(adjusting position)

ggplot(summary, aes(x=time, y=rep,
colour=trt)) + geom_errorbar(aes(ymin=rep-
se,ymax=rep+se), width=.1,position=pd) +
geom_line(position=pd) +
geom_point(position=pd) + xlab ("Time
(Hours)") + ylab("Value (Mean +/- SE)")+
ggtitle("Errorbar plot with Jittering in
ggplot2 package")+ theme_bw()+
scale_y_continuous(limits=c(5,30),
breaks=0:30*5)
```

As we can observe addition of single line (yellow highlighted text in code above) to the existing code can do jittering in R. The beauty of R graphics is variation in graphical output across packages. In above example two figures are plotted in a single panel and this can be done with only few lines of code as shown below.

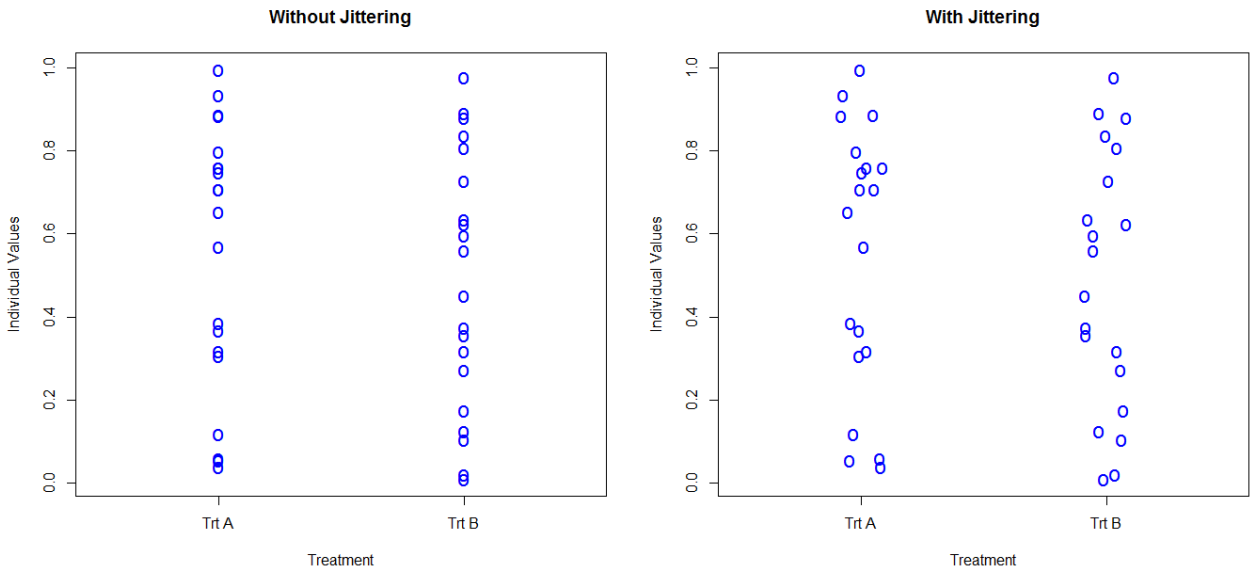
R Code For Plotting Two Plots in Single Panel

```
pagelayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)

grid.newpage()
pushViewport(viewport(layout = grid.layout(1, 2)))
print(plot1, vp = pagelayout(1, 1))
print(plot2, vp = pagelayout(1, 2))
```

INDIVIDUAL VALUES PLOT

The data used here in this example is randomly generated from uniform distribution (20 values for each treatment).

Ex. 2 Individual values plot



Individual values of a certain parameter are plotted for two treatments for 20 subjects. If we observe the first figure some of the values got overlapped and we can't even distinguish some subjects, whereas in second figure all values are clearly visible. This is what we are interested in the exact visualization of the data. To do jittering in this case, a function called 'jitter' in base package of R is used. SAS users are very well aware that how tedious is to write code for such kind of data to do jittering.

We have seen in all the three illustrations mentioned above that we can write codes with ease, which avoids writing numerous lines of code. Also we have option of using functions from various packages for the same task, creating multiple graphs in a single panel. Due to these capabilities and advance features, R can be considered as a revolution in terms of graphics when compared it with other software. The question might arise whether R is validated? Base package of R is validated one. Graphs can be validated by mere visual inspection. As we discussed earlier, idea is to create tables and listings in SAS which means the datasets required for graphics would be generated in SAS which would be a validated one.

SAVING GRAPHS IN R

While saving the graphical output generated in R, best choice is PDF format as the quality of graph is always retained. However the problem with PDF file is when we create report in word or power point, they are not able to handle embedded PDF graph reliably, it is painful to edit or review that documents. We can use PDF formats when we are sending only graphical output through emails and client needs the best quality as possible. JPEG format works fine for photograph-like images, but introduces blurry artifacts around lines and letters for the typical R graph. GIF format was the most popular format for many years, but it has several limitations (not least, graphs using many colors -- like image plots -- might not look correct in GIF format).

Hence I would recommend 'PNG' format with high resolution. When reports are prepared in word one can resize the graphic to an appropriate size, but the high resolution gives you the flexibility to choose a size while not compromising with the quality.

LIMITATIONS

1. This paper dose not talks about the capabilities of SAS 9.2, SAS GTL or SAS 9.3. The points which have been mentioned about SAS graphics like lengthy code, annotation and jittering are based on personal experience while creating graphics in SAS (v 9.1).

2. Help in R is more technical. It is not well organized as in other software. So first time users may find it difficult.

CONCLUSION

We have seen from the illustrations in the paper how graphs generated in R have an edge over SAS – in terms of appearance, time taken for coding and overall quality of the graph. So, R can be an ideal choice for creating graphs. Since we are calling R from SAS, we are able to create superior figures in SAS itself providing a sense of comfort to SAS users as well.

REFERENCES

1. SAS2R2SAS – Philip R Holland : http://www.hollandnumerics.co.uk/pdf/SAS2R2SAS_paper.pdf

2. Conduct R analysis within SAS – Liang Xie: http://www.sas-programming.com/2010_04_01_archive.html

3. R Project for Statistical Computing **:** www.r-project.org

4. R Graphic Manual**:** http://rgm2.lab.nig.ac.jp/RGM2/func.php?rd_id=psych:error.bars

ACKNOWLEDGMENTS

I would like to thank my managers, reviewers and colleagues at Cytel (India) for their support, co-ordination and valuable comments. Special thanks to Mr. Ajay Sathe for his guidance.

CONTACT INFORMATION

Author Name: Sameer Bamnote
Company: Cytel Statistical Software & Services Pvt. Ltd. (a subsidiary of Cytel Inc. USA)
Address: S.No. 150, Lohia-Jain IT Park, “A” Wing, 6th Floor, Paud Road, Kothrud
City / Postcode: Pune – 411038, (Maharashtra) India
Work Phone: +9120-67090133
Fax: +9120-67090120
Email: sameer.bamnote@cytel.com
Web: www.cytel.com