**Paper CT05**

# Generating TFLs in R - Challenges and Successes compared to SAS

Amol Waykar, d-wise, Manchester, UK
Kevin Kramer, Experis, Portage, Michigan, USA
Kalyani Komarasetti, Experis, Portage, Michigan, USA
Andrew Miskell, Eli Lilly and Company, Indianapolis, Indiana, USA

## ABSTRACT

Companies have been creating TFLs using SAS for decades. More recently, R usage has increased for TFLs. As this trend continues, knowing R is only a first step. There are many challenges both in coding and infrastructure which must be foreseen to replicate the results produced by SAS. We will present examples of issues which arose as our department started using R for some TFLs. Examples include sorting and rounding differences, derivation of p-values and LS Means, producing submission quality reports, and handling version updates. Some resolutions are based on knowing the different defaults SAS uses rather than R. Others are based on creating R functions to work around different assumptions that R and SAS make. Still others depend on users being aware of the different ways the languages work. We will present one use-case where we still have not figured out how to replicate the values in R.

## INTRODUCTION

Tables, Figures, and Listings (TFLs) are produced by every pharmaceutical company in preparation for submissions, for internal analyses, for publications, and for many other reasons. Traditionally SAS has been used to create TFLs. The methods, defaults, and assumptions are generally well-known. As R gains more prevalence across the industry, this may lead to building complex TFLs using R language as well.

There are different defaults and assumptions which R makes as compared with SAS. It is important to note that neither SAS nor R can be considered precise in comparison with each other – they are simply different. It is essential that authors and reviewers understand the differences between the two languages. If the coder is attempting an exact match between SAS output and R output, this will come into play. If there is not an exact match, the reviewer must understand the different assumptions that were used to fully understand the analysis.

This paper will document many of the challenges and processes our group went through while creating TFLs in R for the first time. In some cases, the challenges are structural in nature (system and philosophical issues partially unrelated to coding). In other cases, the challenges are based on different defaults which R has as compared with SAS. In yet other cases, coding workarounds were written to ensure that R mirrors SAS behavior.

The paper will start out discussing the system and foundational issues in which R differs from SAS and how that affects TFL coding. The next sections will focus on results and output styles between the two languages and how to workaround the differences. The last section will focus on one area which still has not been resolved and some of the steps we have taken.

## VERSION AND CONSISTENCY ISSUES

SAS is a company-owned and supported tool whereas R is opensource. There are benefits and drawbacks to each of these which affect TFL creation.

The SAS language is governed by SAS Institute and thereby has a support structure in place. With the support structure, SAS tends to be slower to adapt to new functionalities. However, when updates are rolled out, they are more likely to be backward compatible and integrated into the rest of the language. SAS versions are updated when the IT group tests and implements them – typically in a time period of months or years.

R can quickly adapt and update in a changing environment since it is opensource code which is not supported by a single entity. If new features are needed – whether they be for statistical or graphing analyses or purely output and file type-related – the R user base can update functions. The drawback is since the R user base has ‘day jobs’, the priority to update these functions may be lower on their list, may actually take longer, and may not align with the needs of the users desiring to use the functions. Also, without a single entity overseeing these updates, the robustness of the validation varies and functions may not be as backward-compatible as they otherwise could be. R functions may be updated at any time and the Base R language is in a constant state of evolution that many companies update it multiple times per year.

Our company typically updates the base R version on a rolling six month cycle. When this occurs, we do some random checking and occasionally a bug appears. If a bug does appear, we either fix it via a coding update or we request a different version of the appropriate package.

## SORTING DIFFERENCES

When creating TFLs, it is important to understand how the data will be sorted. Two inherent reasons come to mind:
1) How the data will be presented in the output.
2) Which data will be selected if choosing based on position in a list (e.g. selecting first position or selecting last position).

SAS and R handle missing values differently in the sorting process.

In SAS, while using a PROC SORT, missing values will sort before populated values whether they come from a character variable or a numeric variable.

In R, while using the order function, missing values will sort before populated values when ordering on a character column. Missing values will sort after populated values when ordering on a numeric column.

Let us assume there is a dataset (data frame) called anl_sort with a numeric variable (numvar) and a character variable (charvar).

**Numeric Variable Sorting**
SAS: proc sort data=anl_sort; by numvar; run;
R: anl_sort <- anl_sort[order(anl_sort$numvar), ]

| SAS | | R |
|---|---|---|
| **numvar** | | |
| . | | 1 |
| 1 | | 2 |
| 2 | | 3 |
| 3 | | 4 |
| 4 | | 5 |
| 5 | | NA |

**Character Variable Sorting**
SAS: proc sort data=anl_sort; by charvar; run;
R: anl_sort <- anl_sort[order(anl_sort$charvar), ]

| SAS | | R |
|---|---|---|
| **charvar** | | |
| . | | NA |
| 1 | | 1 |
| 2 | | 2 |
| 3 | | 3 |
| 4 | | 4 |
| 5 | | 5 |

The user must be aware of the differences if trying to reproduce the same results in SAS and R. Additionally, the differences may produce different results if any selection is done based on the position of a value in the data.

## ROUNDING DIFFERENCES

SAS and R have different algorithms for rounding data which could produce different results.

Within SAS, if the value is exactly ‘5’, it will round away from 0 (if >0, then round up; if <0 then round down). This method is what everyone is taught from elementary school onwards and is presumably what a customer would presume is followed. However, it could be construed as biased as ‘5’ is exactly halfway between the rounding destinations and it always moves away from 0.

R uses the IEC 60559 method which states if the value falls exactly on ‘5’, it will round to the even number. 1.5 will round to 2; 2.5 will also round to 2. Different operating systems may produce different results based on how the number is stored in the system. This method tries to address the bias issues encountered in the normative rounding rules but could be construed as non-transparent unless it is clearly documented to the user which method was used.

This method is documented here: https://www.rdocumentation.org/packages/base/versions/3.5.2/topics/Round

Rounding to integers:
SAS: sas_roundvar=round(unroundedvar,1);
R: dataframe$r_roundvar <- round(dataframe$unroundedvar,digits=0)

Rounding to tenths:
SAS: roundvar=round(unroundedvar,.1);
R: dataframe$r_roundvar <- round(dataframe$unroundedvar,digits=1)

| Raw Value (unroundedvar) | Rounding Precision | SAS Rounded Value (PROC SORT) | R Rounded Value (round function) |
|---|---|---|---|
| -2.5 | integer | -3 | -2 |
| -1.5 | integer | -2 | -2 |
| -.5 | integer | -1 | 0 |
| .5 | integer | 1 | 0 |
| 1.5 | integer | 2 | 2 |
| 2.5 | integer | 3 | 2 |
| 8.75 | tenths | 8.8 | 8.8 |
| 8.85 | tenths | 8.9 | 8.8 |
| 8.95 | tenths | 9.0 | 9.0 |
| 9.05 | tenths | 9.1 | 9.0 |
| 9.15 | tenths | 9.2 | 9.2 |
| 9.25 | tenths | 9.3 | 9.2 |

The ramifications are many. From a cosmetic point of view, the output will differ. If there is rounding during multiple steps of a calculation, the difference could compound.

Our group wrote an R function to round away from 0 which will match the SAS method.

```
ut_round <- function(x, n=0)
{
  # x is the value to be rounded
  # n is the precision of the rounding
   scale <- 10^n
   y <- trunc(x * scale + sign(x) * 0.5) / scale
   # Return the rounded number
   return(y)
}
# If rounding to integer
dataframe$ut_roundvar <- ut_round(dataframe$unroundedvar,n=0)
# If rounding to tenths
dataframe$ut_roundvar <- ut_round(dataframe$unroundedvar,n=1)
```

| Raw Value (unroundedvar) | Rounding Precision | SAS Rounded Value (PROC SORT) | R Rounded Value (round function) | R Rounded Value (ut_round function) |
|---|---|---|---|---|
| -2.5 | integer | -3 | -2 | -3 |
| -1.5 | integer | -2 | -2 | -2 |
| -.5 | integer | -1 | 0 | -1 |
| .5 | integer | 1 | 0 | 1 |
| 1.5 | integer | 2 | 2 | 2 |
| 2.5 | integer | 3 | 2 | 3 |
| 8.75 | tenths | 8.8 | 8.8 | 8.8 |
| 8.85 | tenths | 8.9 | 8.8 | 8.9 |
| 8.95 | tenths | 9.0 | 9.0 | 9.0 |
| 9.05 | tenths | 9.1 | 9.0 | 9.1 |
| 9.15 | tenths | 9.2 | 9.2 | 9.2 |
| 9.25 | tenths | 9.3 | 9.2 | 9.3 |

Please note that if other functions have the round function embedded within them, results may still vary from SAS.
If the ut_round function above is renamed instead to round, then that could overwrite the round function in base which

is called by other functions. Our group was unsure how broad the ramifications of this option would be so we did not test on a large scale basis.

## FISHER'S EXACT AND PEARSON CHI-SQUARED TESTS

Fisher's Exact test is a statistical significance test used in the analysis of contingency tables. It is named after its inventor, Ronald Fisher. The test uses the following formula when determining pvalues:

p= ( ( a + b ) ! ( c + d ) ! ( a + c ) ! ( b + d ) ! ) / a ! b ! c ! d ! N !

Both SAS and R provide procedures/functions used to retrieve the p-values for this test. In SAS these values can be attained by passing a 2 X 2 contingency table into a PROC FREQ. In R the same 2 X 2 contingency table can be passed into the Fisher's Exact.test function from the stats library.

While doing comparison testing between the 2 functions it was noted that when the inputs to the 2 X 2 contingency table both contained 0 values the output p-value from SAS and R would produce different results. These results can be seen in the 2 sample programs below.

Here is a SAS program that uses PROC FREQ for a Fisher's Exact test. The data contains 2 treatment groups with weighted counts both set to 0.

```
DATA test_case;
  INPUT treatment $ count indicator;
  DATALINES;
Trt_A 0 1
Trt_B 0 1
Trt_A 627 0
Trt B 300 0
;

PROC FREQ DATA=test_case;
  WEIGHT count;
  TABLES treatment*indicator;
  Exact fisher / maxtime = 10;
  ods output FishersExact = Fishers;
RUN;
```

NOTE: No statistics are computed for treatment * indicator because treatment has fewer than 2 nonmissing levels.
WARNING: Output 'FishersExact' was not created. Make sure that the output object name, label, or path is spelled correctly. Also, verify that the appropriate procedure options are used to produce the requested output object. For example, verify that the NOPRINT option is not used.

Here is the equivalent program in R that uses the same input data the PROC FREQ used above.

```
test_case = matrix(c(0, 0, 627, 300),
                   ncol = 2
                  )
pval = fisher.test(test_case)
pval

Fisher's Exact Test for Count Data

data: test_case
p-value = 1
```

As you can see from the warning produced by the SAS program, PROC FREQ does not return any Fisher's Exact output when both the treatments contain 0 counts, but in R the output returned by the function Fisher's Exact.test is equal to 1. Based on the formula for calculating the Fisher's Exact, one would conclude that 1 is the correct answer. It should be noted that comparisons run on other combinations of 2 X 2 contingency tables returned matching results.

Pearson's Chi-squared test is a statistical test applied to sets of categorical data to evaluate how likely it is that any observed difference between the sets arose by chance. It tests a null hypothesis stating that the frequency distribution of certain events observed in a sample is consistent with a particular theoretical distribution. Like Fisher's Exact test, both SAS and R provide procedures/functions that can produce pvalues for such a test.

Here is a SAS program that uses PROC FREQ for a Chi-squared test. The data contains 2 treatment groups with weighted counts.

```
DATA test_case;
  INPUT treatment $ count indicator;
  DATALINES;
Trt_A 0 1
Trt_B 0 1
Trt_A 200 0
Trt B 250 0
;

PROC FREQ DATA=test_case;
  WEIGHT count;
  TABLES treatment*indicator/ chisq;
  ods output ChiSq=chisq;
run;
```

Here is the equivalent program in R that uses the same input data that SAS used when run through the PROC FREQ above.

```
test_case = matrix(c(1,1,200,250),
                   ncol = 2
                  )

pval_chisq = suppressWarnings(chisq.test(test_case, correct = FALSE))
pval_chisq

Pearson's Chi-squared test
data: test_case
X-squared = 0.024887, df = 1, p-value = 0.8746
```

It should be noted that while comparing the Chi-squared results between both SAS and R using the methods described above, the p-values returned were identical regardless of the values provided in the 2 x 2 table.

## FORMATTED OUTPUT REPORTING DIFFERENCES

We would like to share some of our experience in the TFL report output creation area. Our in-house requirement was to create the reporting output file type to be in DOCX and the plots generated to be vector graphs, so we will focus our discussion in that context.

Challenges and Workaround - We have listed some of the areas in which we faced challenges and the work around we used to overcome them.

1) PLOTS and VECTOR GRAPHS

Previously, when figures are needed our group has created what is known as a static (raster) graph which is basically just a screen shot. In the past couple years, the need to make our graphs in vector format has become apparent. Let us quickly explain the difference between a static graph and vector graph. A static graph would be an image such as a JPG or PNG file where the resolution remains the same regardless of size and is non-editable in software. A vector graph uses mathematical formulas to draw lines and curves that can be combined to create an image from geometric objects such as circles and polygons. Vector images may be edited by manipulating the lines and curves that make up the image outline after they are generated.

SAS uses various procs for creating different figures (i.e. boxplot, scattered plot etc.) although we currently do not generate the figures as vector graphs. SAS can produce vector graphs in other destinations such as PDF but due to medical writing requirements, we are limited to DOCX vector graphs in our current environment.

We are currently on SAS 9.4 Maintenance Release 6 which has some experimental functionalities for creating vectors graphs in DOCX format. There are still some challenges to be worked through which we have not completed yet although it does seem feasible in the very near future.

R does have some robust packages for building vector graphs such using the ggplot2, gtable, grid etc. However, outputting the plot as a vector graph into DOCX output format required some specific packages and additional work arounds as mentioned below. Please see R script directly below.

```
# This section shows how to create an EMF Vector Graph embedded in a DOCX file
# Load needed libraries
library(officer)
library(flextable)
library(devEMF)

#Create a DOCX object -mydoc using the officer package ready to be used below.
 mydoc  <- officer::read_docx(path = template) %>% cursor_begin() %>% body_remove()

#We can use any plot/graph (let say plot – p1) object which is created using ggplot2, gridtable, grid.
#This ggplot object now needs to be inserted into an emf file ready to be placed into the DOCX file.

# So, now let's create a temporary emf file
 emf_ggplot_file <- tempfile(fileext = ".emf")

#Modify the emf with the correct width and height required
#Note make sure to set the options for emfPlus = TRUE, emfPlusFont =TRUE and emfPlusRaster= TRUE, failure to
do so will not render the graph correctly in DOCX file.

        emf(file = emf_ggplot_file, width = 8.9, height = graph_height
            ,emfPlus=TRUE, emfPlusFont=TRUE, emfPlusRaster =TRUE
            )

#Place the plot now inside the emf file.
        grid.draw(p1)
        dev.off()

 #Now add it to a new Word document:
        mydoc <- officer::body_add_img(x = mydoc
                                       ,src = emf_ggplot_file
                                       ,width = 8.9
                                       ,height = graph_height
                                      )
```

2) CUSTOMIZED TABLES and LISTINGS

A large amount of time and energy is spent in the development of tables and listings. As developers it seems we spend most of that time making the output aesthetically pleasing to the outside world.

When it comes to SAS, we tend to gravitate towards PROC REPORT as our main reporting engine. With it we can control the appearance of just about any piece of the output from the size of a column to the color of a cell.

With R, producing the same quality of reports is not as straight forward and simple as one would hope. There are multiple packages that can generate output but only a few were found that could produce the quality, specificity, and complexity that is required for submission level reports. With much investigation we chose to go with two packages (officer, flextable) that provided the functionality that best fit our needs. These two packages combined provided the ability to produce the specific styles, widths, merging of column headers and rows, and more that you would find in PROC REPORT. Even with all the functionality that officer and flextable contained we still were faced with several challenges that we had to overcome.

The first challenge we had to overcome was placing titles and footnotes on every page of the report. PROC REPORT does this automatically when creating its pages but with R there is no simple way to tell flextable or officer to force the footnotes and titles to appear on every page. In order to provide this functionality, we had to first determine the amount of space each piece (titles, footnotes, body) of the report would take up per page. Once we were able to figure out the size of each page then it was just a matter of breaking up the input data into its appropriate pages and printing it to the report. The steps we took to handle this task follows:

1. Created a utility function (ut_pagesize) that linked each row in the data to a page. The utility function would take as input the titles, footnotes, report data, and report layout. From the inputs it could determine the number of records per page based on the size of the column and the amount of wrapping that would occur within each cell of the report.

2. Once the rows of the data were linked to a page on the report all that's left is looping through the data and printing out the title, rows associated with the page, and the footnotes. This is done for each page in the report.

The second challenge we encountered was due to performance within the flextable package. As the number of pages increase the amount of time to produce the reports went up drastically. Our challenge was to find a workaround to the limitation of the package when producing large number of pages. The solution we found was to produce each page as a separate DOCX file and then bring all the pages together into one main DOCX file at the end. This limitation has been noted by the author of the package and set to be resolved in a future release.

## ANCOVA (WITHOUT REPEATED MEASURES) AND REPEATED MEASURES

Least Squares Mean (LSM) change estimate, LSM change Standard Error (SE) and p-values (Probt Values) for the box plots (also called box-and-whisker plots or box-whisker plots) in R can be calculated in numerous ways. One of the main challenges the team ran into was to get the results from SAS and R to match 100%, especially when different covariance structures are used.

The results of the LSM and p-values were matching 100% between SAS (using PROC MIXED) and R (using lm and emmeans function) when Repeated Measures were not included in the analysis.

For any analysis with Repeated Measures, the match is not always 100% in every instance with SAS. Only around 2-4 decimal places were matching depending on the Covariate Matrices/Structure. AR(1) is one example of Covariate Matrix where the LSM values matched 100%. Other Covariate Matrices match the LSM values up to 4 decimal places. All p-values with Repeated Measures Analysis are matching to about 2 or 3 decimal places (but no more) regardless of the Covariate Matrix.

We will be demonstrating the above clearly using the test data which we created and tested in both SAS and R. Here is a snapshot of the input data structure, along with the output generated from SAS and R.

```
> glimpse(sas_ds1)
Observations: 300
Variables: 14
$ USUBJID   <dbl> 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7…
$ SEX       <chr> "F", "F", "F", "M", "M", "M", "F", "F", "F", "M", "M", …
$ AGE       <dbl> 53, 53, 53, 73, 73, 73, 71, 71, 71, 57, 57, 57, 55, 55,…
$ COUNTRY   <chr> "FRA", "FRA", "FRA", "FRA", "FRA", "FRA", "FRA", "FRA",…
$ BASE      <dbl> 96.65055, 96.65055, 96.65055, 104.30261, 104.30261, 104…
$ VISITCN   <dbl> 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1…
$ VISITC    <chr> "WEEK_1", "WEEK_2", "WEEK_3", "WEEK_1", "WEEK_2", "WEEK…
$ TRT       <chr> "Drug A 100 mg", "Drug A 100 mg", "Drug A 100 mg", "Dru…
$ TRTN      <dbl> 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, …
$ CHG       <dbl> 23.20724, 18.22174, 26.96845, 32.02932, 10.34195, 26.62…
$ PARAMCD   <chr> "DIABP", "DIABP", "DIABP", "DIABP", "DIABP", "DIABP", "…
$ PARAM     <chr> "DIABETIC EFFICACY MEASURE", "DIABETIC EFFICACY MEASURE…
$ TRTN_F    <fct> 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, …
$ VISITCN_F <fct> 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1…
```

**CASE-1: No Repeated Measure (Calculating LSM_CHG, LSM_CHG_SE and P_VALUE)**

SAS:

```
title "USING NO REPEATED MEASURE ";

ods output lsmeans=outloc.anl_lsm_norep(keep=paramcd param effect trtn visitcn estimate stderr) ;
ods output diffs=outloc.anl_pvals_norep(keep=paramcd param trtn _trtn probt visitcn _visitcn
                                        where=(visitcn=_visitcn and not missing(visitcn)));
proc mixed data=outloc.box_mmr_test;
  class trtn(ref="100") visitcn  usubjid;
  by paramcd param;
  model chg=base visitcn trtn trtn*visitcn   ;
  lsmeans trtn/diff ;
  lsmeans trtn*visitcn/diff slice=visitcn;
run;
ods _all_ close;
```

R:

```
# USING NO REPEATED MEASURE - LM FUNCTION
model_chg_m = lm(CHG ~ BASE + VISITCN_F + TRTN_F  + TRTN_F*VISITCN_F, data = sas_ds1)
```

lsm_calc_lst_chg_m <- emmeans(model_chg_m, revpairwise ~ TRTN_F*VISITCN_F, adjust="none")

R

USING NO REPEATED MEASURE

| TRTN_F | VISITCN_F | TRTN_F_ | VISITCN_F_ | P_VALUE |
|---|---|---|---|---|
| 200 | 1 | 100 | 1 | 0.00000000 |
| 200 | 2 | 100 | 2 | 0.00000000 |
| 200 | 3 | 100 | 3 | 0.00000000 |
| 300 | 1 | 100 | 1 | 0.11510889 |
| 300 | 2 | 100 | 2 | 0.78961278 |
| 300 | 3 | 100 | 3 | 0.27338961 |
| 400 | 1 | 100 | 1 | 0.03594974 |
| 400 | 2 | 100 | 2 | 0.01002154 |
| 400 | 3 | 100 | 3 | 0.08186870 |

| TRTN_F | VISITCN_F | LSM_CHG | LSM_CHG_SE |
|---|---|---|---|
| 100 | 1 | 23.095927 | 1.553357 |
| 100 | 2 | 23.607709 | 1.553357 |
| 100 | 3 | 22.003855 | 1.553357 |
| 200 | 1 | 6.049322 | 1.530213 |
| 200 | 2 | 4.725101 | 1.530213 |
| 200 | 3 | 6.830730 | 1.530213 |
| 300 | 1 | 26.938881 | 1.870359 |
| 300 | 2 | 24.257093 | 1.870359 |
| 300 | 3 | 24.672254 | 1.870359 |
| 400 | 1 | 28.150635 | 1.829541 |
| 400 | 2 | 29.825481 | 1.829541 |
| 400 | 3 | 26.191883 | 1.829541 |

SAS

USING NO REPEATED MEASURE

| trtn | _visitcn | _trtn | visitcn | Probt |
|---|---|---|---|---|
| 200 | 1 | 100 | 1 | 0.00000000 |
| 200 | 2 | 100 | 2 | 0.00000000 |
| 200 | 3 | 100 | 3 | 0.00000000 |
| 300 | 1 | 100 | 1 | 0.11510889 |
| 300 | 2 | 100 | 2 | 0.78961278 |
| 300 | 3 | 100 | 3 | 0.27338961 |
| 400 | 1 | 100 | 1 | 0.03594974 |
| 400 | 2 | 100 | 2 | 0.01002154 |
| 400 | 3 | 100 | 3 | 0.08186870 |

| trtn | visitcn | Estimate | StdErr |
|---|---|---|---|
| 100 | 1 | 23.095927 | 1.5533569 |
| 100 | 2 | 23.607709 | 1.5533569 |
| 100 | 3 | 22.003855 | 1.5533569 |
| 200 | 1 | 6.0493218 | 1.5302131 |
| 200 | 2 | 4.7251005 | 1.5302131 |
| 200 | 3 | 6.8307301 | 1.5302131 |
| 300 | 1 | 26.938881 | 1.8703592 |
| 300 | 2 | 24.257093 | 1.8703592 |
| 300 | 3 | 24.672254 | 1.8703592 |
| 400 | 1 | 28.150635 | 1.8295411 |
| 400 | 2 | 29.825481 | 1.8295411 |
| 400 | 3 | 26.191883 | 1.8295411 |

**CASE-2: With Repeated Measures (Calculating LSM_CHG, LSM_CHG_SE and P_VALUE)**

SAS:

```
title "USING REPEATED MEASURES - AR1 ";
ods output lsmeans=outloc.anl_lsm_rep(keep=paramcd param effect trtn visitcn estimate stderr);
ods output diffs=outloc.anl_pvals_rep(keep=paramcd param trtn _trtn probt visitcn _visitcn
                                      where=(visitcn=_visitcn and not missing(visitcn)));
proc mixed data=outloc.box_mmr_test;
  class trtn(ref="100") visitcn usubjid;
  by paramcd param;
  model chg=base visitcn trtn trtn*visitcn;
  repeated visitcn/subject=usubjid type=ar(1);
  lsmeans trtn/diff ;
  lsmeans trtn*visitcn/diff slice=visitcn;
run;
ods _all_ close;
```

R:

```
# USING REPEATED MEASURES – lme function for AR1
model_chg_m = lme(CHG ~ BASE + VISITCN_F + TRTN_F  + TRTN_F*VISITCN_F, random = ~1|USUBJID,
correlation = corAR1(0.8,form = ~VISITCN | USUBJID), data = sas_ds1)

lsm_calc_lst_chg_m <- emmeans(model_chg_m, revpairwise ~ TRTN_F*VISITCN_F, adjust="none")

# USING REPEATED MEASURES - gls FUNCTION for AR1
library(nlme)
model_chg_m =gls(CHG ~ BASE + VISITCN_F + TRTN_F  + TRTN_F*VISITCN_F, correlation = corAR1(0.8,form =
~VISITCN | USUBJID), data = sas_ds1)

lsm_calc_lst_chg_m <- emmeans(model_chg_m, revpairwise ~ TRTN_F*VISITCN_F, adjust="none")
```

R (*USING - lme function*)

USING REPEATED MEASURE

| TRTN_F | VISITCN_F | TRTN_F_ | VISITCN_F_ | P_VALUE |
|---|---|---|---|---|
| 200 | 1 | 100 | 1 | 0.00000000 |
| 200 | 2 | 100 | 2 | 0.00000000 |
| 200 | 3 | 100 | 3 | 0.00000000 |
| 300 | 1 | 100 | 1 | 0.11732492 |
| 300 | 2 | 100 | 2 | 0.78996190 |
| 300 | 3 | 100 | 3 | 0.27522788 |
| 400 | 1 | 100 | 1 | 0.03772905 |
| 400 | 2 | 100 | 2 | 0.01104292 |
| 400 | 3 | 100 | 3 | 0.08405548 |

| TRTN_F | VISITCN_F | LSM_CHG | LSM_CHG_SE |
|---|---|---|---|
| 100 | 1 | 23.095858 | 1.553381 |
| 100 | 2 | 23.607640 | 1.553381 |
| 100 | 3 | 22.003786 | 1.553381 |
| 200 | 1 | 6.049564 | 1.530260 |
| 200 | 2 | 4.725343 | 1.530260 |
| 200 | 3 | 6.830973 | 1.530260 |
| 300 | 1 | 26.938943 | 1.870387 |
| 300 | 2 | 24.257155 | 1.870387 |
| 300 | 3 | 24.672317 | 1.870387 |
| 400 | 1 | 28.150324 | 1.829602 |
| 400 | 2 | 29.825170 | 1.829602 |
| 400 | 3 | 26.191572 | 1.829602 |

SAS

USING REPEATED MEASURE - AR1

| trtn | _visitcn | _trtn | visitcn | Probt |
|---|---|---|---|---|
| 200 | 1 | 100 | 1 | 0.00000000 |
| 200 | 2 | 100 | 2 | 0.00000000 |
| 200 | 3 | 100 | 3 | 0.00000000 |
| 300 | 1 | 100 | 1 | 0.11564699 |
| 300 | 2 | 100 | 2 | 0.78966959 |
| 300 | 3 | 100 | 3 | 0.27382843 |
| 400 | 1 | 100 | 1 | 0.03639332 |
| 400 | 2 | 100 | 2 | 0.01027126 |
| 400 | 3 | 100 | 3 | 0.08242219 |

| trtn | visitcn | Estimate | StdErr |
|---|---|---|---|
| 100 | 1 | 23.095858 | 1.5533810 |
| 100 | 2 | 23.607640 | 1.5533810 |
| 100 | 3 | 22.003786 | 1.5533810 |
| 200 | 1 | 6.0495642 | 1.5302604 |
| 200 | 2 | 4.7253429 | 1.5302604 |
| 200 | 3 | 6.8309725 | 1.5302604 |
| 300 | 1 | 26.938943 | 1.8703873 |
| 300 | 2 | 24.257155 | 1.8703873 |
| 300 | 3 | 24.672317 | 1.8703873 |
| 400 | 1 | 28.150324 | 1.8296024 |
| 400 | 2 | 29.825170 | 1.8296024 |
| 400 | 3 | 26.191572 | 1.8296024 |

R (*USING - gls function*)

USING REPEATED MEASURE

| TRTN_F | VISITCN_F | TRTN_F_ | VISITCN_F_ | P_VALUE |
|---|---|---|---|---|
| 200 | 1 | 100 | 1 | 0.00000000 |
| 200 | 2 | 100 | 2 | 0.00000000 |
| 200 | 3 | 100 | 3 | 0.00000000 |
| 300 | 1 | 100 | 1 | 0.11510259 |
| 300 | 2 | 100 | 2 | 0.78957476 |
| 300 | 3 | 100 | 3 | 0.27337402 |
| 400 | 1 | 100 | 1 | 0.03596261 |
| 400 | 2 | 100 | 2 | 0.01002606 |
| 400 | 3 | 100 | 3 | 0.08189290 |

| TRTN_F | VISITCN_F | LSM_CHG | LSM_CHG_SE |
|---|---|---|---|
| 100 | 1 | 23.095858 | 1.553381 |
| 100 | 2 | 23.607640 | 1.553381 |
| 100 | 3 | 22.003786 | 1.553381 |
| 200 | 1 | 6.049564 | 1.530260 |
| 200 | 2 | 4.725343 | 1.530260 |
| 200 | 3 | 6.830973 | 1.530260 |
| 300 | 1 | 26.938943 | 1.870387 |
| 300 | 2 | 24.257155 | 1.870387 |
| 300 | 3 | 24.672317 | 1.870387 |
| 400 | 1 | 28.150324 | 1.829602 |
| 400 | 2 | 29.825170 | 1.829602 |
| 400 | 3 | 26.191572 | 1.829602 |

SAS

USING REPEATED MEASURE - AR1

| trtn | _visitcn | _trtn | visitcn | Probt |
|---|---|---|---|---|
| 200 | 1 | 100 | 1 | 0.00000000 |
| 200 | 2 | 100 | 2 | 0.00000000 |
| 200 | 3 | 100 | 3 | 0.00000000 |
| 300 | 1 | 100 | 1 | 0.11564699 |
| 300 | 2 | 100 | 2 | 0.78966959 |
| 300 | 3 | 100 | 3 | 0.27382843 |
| 400 | 1 | 100 | 1 | 0.03639332 |
| 400 | 2 | 100 | 2 | 0.01027126 |
| 400 | 3 | 100 | 3 | 0.08242219 |

| trtn | visitcn | Estimate | StdErr |
|---|---|---|---|
| 100 | 1 | 23.095858 | 1.5533810 |
| 100 | 2 | 23.607640 | 1.5533810 |
| 100 | 3 | 22.003786 | 1.5533810 |
| 200 | 1 | 6.0495642 | 1.5302604 |
| 200 | 2 | 4.7253429 | 1.5302604 |
| 200 | 3 | 6.8309725 | 1.5302604 |
| 300 | 1 | 26.938943 | 1.8703873 |
| 300 | 2 | 24.257155 | 1.8703873 |
| 300 | 3 | 24.672317 | 1.8703873 |
| 400 | 1 | 28.150324 | 1.8296024 |
| 400 | 2 | 29.825170 | 1.8296024 |
| 400 | 3 | 26.191572 | 1.8296024 |

**CONCLUSION**

Creating TFLs via R can be accomplished but knowing the differences between R and SAS is essential. As more companies and groups utilize R to create TFLs, many differences can be expected. Both R and SAS are capable of creating TFLs which can be used for regulatory submissions, internal analyses, and publications. Hopefully this

paper presents the differences between the two languages to help groups identify the best way to build a TFL as per their business needs, as well as transparently highlight any differences which could arise and proactively mitigate version and system issues.

## CONTACT INFORMATION

Amol Waykar
d-wise
Suite 7A, Manchester One, 53 Portland St.
Manchester, United Kingdom, M1 3LD
Amol.Waykar@d-wise.com

Kevin Kramer
Experis
5220 Lovers Lane
Portage, Michigan, USA 49002
kevin.kramer@experis.com

Kalyani Komarasetti
Experis
5220 Lovers Lane
Portage, Michigan, USA 49002
kalyani.komarasetti@experis.com

Andrew Miskell
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, Indiana, USA 46285
(317) 651-7903
andrew.miskell@lilly.com