PharmaSUG 2023 - Paper QT-046

Repetitive Analyses in SAS® – Use of Macros Versus Data Inflation and BY Group Processing

Bradford J. Danner, IQVIA, Durham, NC
Indrani Sarkar, IQVIA, Durham, NC

ABSTRACT

While preparing clinical reports, we are commonly tasked to produce multiple outputs of the same analysis, using a different endpoint of interest, or slightly different populations of interest, or according to a suite of categorical subgroups. Naturally, we can accomplish such repetitive tasks efficiently using SAS® with MACRO processing. Alternatively, “data inflation”, an approach that does not employ MACRO processing, with careful use of OUTPUT statements in the SAS data step, we ‘inflate’ the source data, so that all variations of the multiple analyses are in one dataset, which can then pass-through analysis procedures once with BY group processing. The objective of this article is to demonstrate these two approaches, either of which can be used for the purpose of analysis and review. Outputs from both approaches can be consolidated and exported into one source which will make the review process less time-consuming. Time-to-event analyses (Kaplan-Meier and Cox regression) will be used to demonstrate both techniques and will be discussed and compared.

INTRODUCTION

Statistical programmers are often required to do the same analysis multiple times for a clinical summary report – different subgroups may be of interest and/or multiple analysis populations are defined. Since the results of the repetitive analyses are usually presented in individual reports in tabular or graphical form, programmers tend to employ a macro, calling it as many times as needed. All that is required is to pass in parameters to control what the macro produces. While this allows containment of much of the analysis code in one place, it is still up to the programmer to pass in the combinations of parameters required to produce the various outputs.

As statisticians we are often asked to check or reproduce the complete analysis using raw and/or submission level datasets. While we could develop an independent macro processing approach to produce comparative sources to check against the multiple outputs, one could assume that our approach would produce an alternative set of outputs to compare against the report outputs. Alternatively, we have found that it may be more efficient for us as reviewers to replicate the analyses and collate all the results in one source file which can be used to compare all the report outputs. In other words, we find it easier to complete a one versus many reviews, as opposed to a many versus many reviews.

To accomplish this, we ‘inflate’ the dataset by adding multiple OUTPUT commands to a SAS DATA step to create one singular larger dataset that contains the endpoints and populations of interest. Endpoint and population combinations are denoted in a temporary indicator variable of which we add to the data. We then pass the inflated dataset into the analysis procedures once, taking care to include our temporary indicator variable in all subsequent BY statements. Once the analysis procedures are run, and the output is consigned to ODS objects, we can then put them back together into one source, merely through our temporary indicator variable included in the BY statement.

In this paper, we will demonstrate this concept by performing a time to event analysis of efficacy endpoints in a Phase 3 oncology study, using both macro and data inflation approaches. Kaplan-Meier and Cox regression analysis will be included, for multiple endpoints and populations of interest. All output required to perform the review will be written to one file with which we can perform our review.

METHOD AND RESULTS

In large scale Phase 3 oncology studies, we will often have multiple or co-primary efficacy endpoints which need to be summarized, for a randomized placebo or standard course of treatment-controlled study. For example, both overall survival (OS) and progression-free survival (PFS) may be of interest.

As is often the case, studies will have multiple populations of interest, for example the full analysis set (FAS) or intent-to-treat (ITT), primary analysis set (PAS), secondary analysis set (SS), or even important subgroups determined by demographic characteristics of individual patients. For this exercise, we will illustrate with three populations of interest – FAS, PAS, and SS.

Therefore, the nature of our analysis will be two endpoints (OS and PFS) in three populations (FAS, PAS, and SS). This will result in 6 combinations of outputs that we must review. As is often the case, time-to-event analyses in clinical summary reports will be present in both summary table and a KM plot, so in this example, theoretically 12 outputs would be under scrutiny.

DATA DESCRIPTION

A small subset of the data we plan to summarize (ADTTE) may be illustrated as follows in **Table 1**.

USUBJID	ARMCD	PARAMCD	AVAL	CNSR	FASFL	PASFL	SSFL
1001	B	OS	23.3	1	Y	Y	N
1001	B	PFS	23.3	1	Y	Y	N
1002	A	OS	4.4	1	Y	Y	Y
1002	A	PFS	3.1	1	Y	Y	Y

Table 1. Sample print of ADTTE dataset.

FASFL, PASFL, and SSFL are population flags, where Y indicates the patient is included, and N indicates the patient is not included. The CNSR variable in this example indicates an event occurred when CNSR=0, or that the patient was censored when CNSR=1.

MACRO PROCESSING

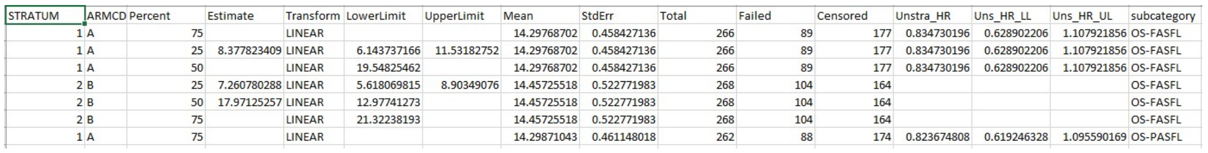
The macro we wrote for this purpose includes both PROC LIFETEST and PROC PHREG. ODS OUTPUT statements are used to extract needed statistics. Output datasets are then merged into one dataset using ARMCD as key variable. There will be one dataset for each of 6 combinations of 2 endpoints and 3 population flags. These 6 output datasets are then combined into one dataset in a DATA step.

Below is the complete code of the macro mentioned in this paper. Macro variables used for this are: “dsin”, “where”, “subset”, “out”, and “label”. “dsin” is used for input dataset. The macro variable “subset” indicates the combination used to subset the input dataset. The macro variable “out” defines the output dataset after combining all the summary datasets from two procedures. The variable “label” is used to identify the subset condition. This will be useful to map back to the condition used for analysis, when all the datasets are combined in one single dataset.

```
%macro tte(dsin=, where= , subset= , out= , label=);
ods output Quartiles = Quart means=mean censoredsummary=cens;
proc lifetest data=&dsin conftype=linear;
     where &where;
     strata armcd;
     time aval*cnsr(1);
run;
ods output ParameterEstimates = uns_est(keep=classval0 HazardRatio
HRLowerCL HRUpperCL rename=(HazardRatio=Unstra_HR HRLowerCL=Uns_HR_LL
HRUpperCL=Uns_HR_UL));
proc phreg data=&dsin; /*unstratified*/
     where &where;
     class armcd (ref="B");
```

```
      model aval*cnsr(1)=armcd/rl alpha=0.05;
  run;

  proc sql;
   create table ds_&out as select a.*, b.mean, b.stderr, c.total, c.failed,
   c.censored, d.Unstra_HR,d.Uns_HR_LL, d.Uns_HR_UL,
   &label as subcategory format= $15. length=15
           from (quart as a left join mean as b on a.armcd = b.armcd) left
           join cens as c on a.armcd=c.armcd left join uns_est as d on
           a.armcd=d.classval0
   order by a.armcd;
  quit;
  %mend;

  %tte(dsin=adtte, where=%str(paramcd = 'OS' and
  fasfl='Y'),subset=%str('OS'), out=OS_FAS, label = %str('OS-FASFL'));
  %tte(dsin=adtte, where=%str(paramcd = 'OS' and
  pasfl='Y'),subset=%str('OS'), out=OS_PAS, label = %str('OS-PASFL'));
  %tte(dsin=adtte, where=%str(paramcd = 'OS' and ssfl='Y'),subset=%str('OS'),
  out=OS_SAS, label = %str('OS-SSFL'));
  %tte(dsin=adtte, where=%str(paramcd = 'PFS' and
  fasfl='Y'),subset=%str('PFS'), out=PF_FAS, label = %str('PFS-FASFL'));
  %tte(dsin=adtte, where=%str(paramcd = 'PFS' and
  pasfl='Y'),subset=%str('PFS'), out=PF_PAS, label = %str('PFS-PASFL'));
  %tte(dsin=adtte, where=%str(paramcd = 'PFS' and
  ssfl='Y'),subset=%str('PFS'), out=PF_SAS, label = %str('PFS-SSFL'));

  data final;
      set ds:;
  run;
```

The dataset 'final' was exported to Microsoft Excel using a PROC EXPORT. A screenshot sample of the final file which is used to review all the outputs is illustrated in **Figure 1**.



STRATUM	ARMCD	Percent	Estimate	Transform	LowerLimit	UpperLimit	Mean	StdErr	Total	Failed	Censored	Unstra_HR	Uns_HR_LL	Uns_HR_UL	subcategory
1	A	75		LINEAR			14.29768702	0.458427136	266	89	177	0.834730196	0.628902206	1.107921856	OS-FASFL
1	A	25	8.377823409	LINEAR	6.143737166	11.53182752	14.29768702	0.458427136	266	89	177	0.834730196	0.628902206	1.107921856	OS-FASFL
1	A	50		LINEAR	19.54825462		14.29768702	0.458427136	266	89	177	0.834730196	0.628902206	1.107921856	OS-FASFL
2	B	25	7.260780288	LINEAR	5.618069815	8.90349076	14.45725518	0.522771983	268	104	164				OS-FASFL
2	B	50	17.97125257	LINEAR	12.97741273		14.45725518	0.522771983	268	104	164				OS-FASFL
2	B	75		LINEAR	21.32238193		14.45725518	0.522771983	268	104	164				OS-FASFL
1	A	75		LINEAR			14.29871043	0.461148018	262	88	174	0.823674808	0.619246328	1.095590169	OS-PASFL

Figure 1. Sample print of summaries created using a custom MACRO.

DATA INFLATION

An alternative to using a macro-based approach, which is called 6 times, involves inflating our ADTTE dataset, within a SAS DATA step, to include all our parameters and populations of interest. The following illustrates two examples by which this may be accomplished, by using multiple OUTPUT statements. The first example is probably the most straightforward. We include a second example, which utilizes an ARRAY and use of the VNAME function, which we have found makes the syntax more generalizable to subgroup analysis, when multiple demographic or baseline characteristics are of interest. Both approaches result in the same dataset for analysis:

```
data approach2; length type $40.;
  set ADTTE (where=(paramcd in ('OS','PFS')));
```

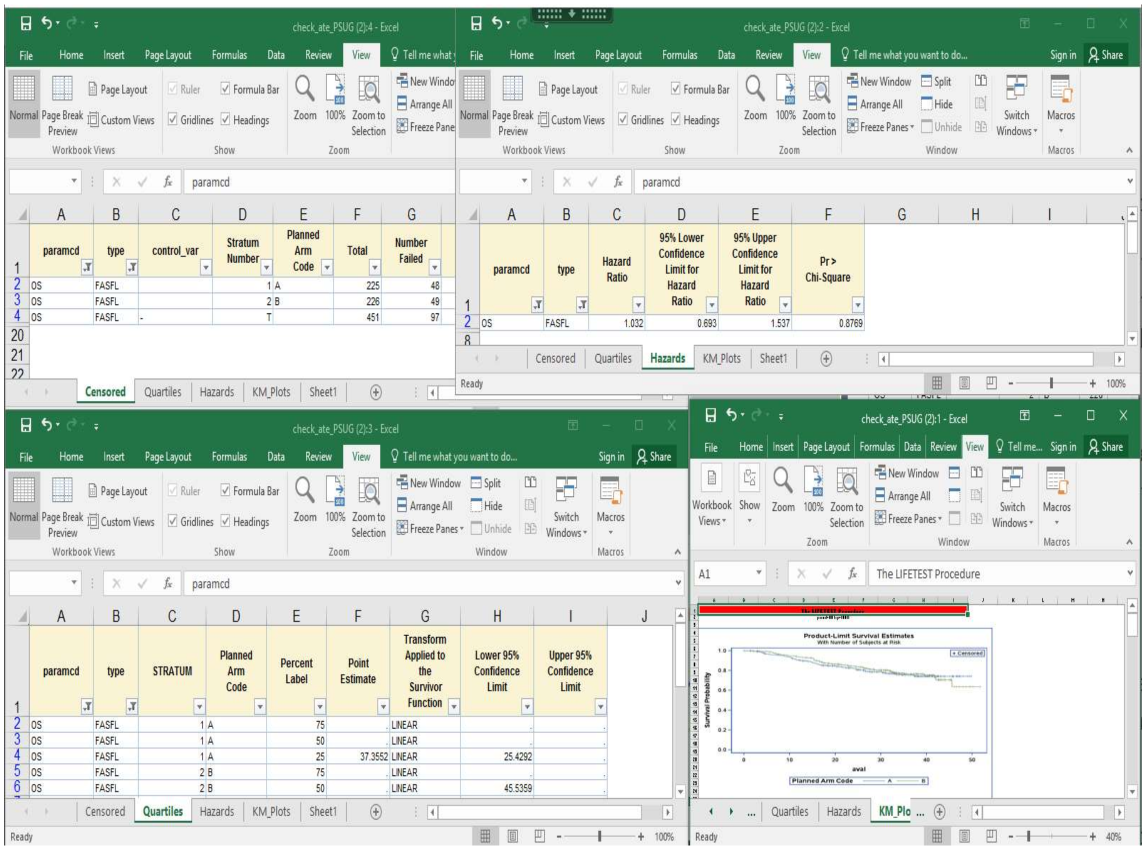
```
   if FASFL='Y'  then do;
      type='FASFL'; output;
   end;
   if pasfl='Y'  then do;
      type='PASFL'; output;
   end;
   if SSFL='Y'  then do;
      type='SSFL'; output;
   end;
run;

data approach2;  length TYPE $40.;
  set ADTTE (where=(paramcd in ('OS','PFS')));
  array POPS  FASFL PASFL SSFL;
  do i=1 to dim(POPS);
    if POPS(i)='Y' then do; TYPE=strip(vname(POPS(i))); output; end;
  end;
run;
```

We then pass our inflated dataset through LIFETEST and PHREG procedures, once. To keep things organized, we include in the BY statement the parameter and TYPE variable, which we added during the inflation step. This will be important later, depending on how we wish to arrange the results for comparative purposes. One caveat of this approach is the data must be sorted by the PARAMCD and TYPE variables:

```
proc sort data=approach2; by paramcd type; run;

ods output censoredsummary = cs Quartiles = quarts;
proc lifetest data = approach2
  method = km alpha = 0.05 alphaqt = 0.05 conftype = linear;
  time aval * cnsr (1);
  by paramcd type;
  strata armcd;
run ;

ods output  parameterestimates=unstrat_ab (keep=paramcd type hazardratio
hrlowercl hruppercl) GlobalTests=unstratp_ab (keep=paramcd type test
probchisq where=(test='Score'));
proc phreg data=approach2;
   class  armcd (ref='B');
   model aval*cnsr(1)=armcd /  ties=discrete  risklimits=both
   alpha=0.05 ;
   by paramcd type;
run;

*** to present the hazard ratio, associated confidence intervals, and p-
value, we merge as such ***;
data hrs;
  merge unstrat_ab unstratp_ab;
  by paramcd type ;
run;
```

To collate the output using the data inflation approach, we have chosen to output all comparative summary output and KM plots into one singular XLSX file, with multiple worksheets. The following code will produce the spreadsheet illustrated in **Figure 2**:

```
ods excel options( sheet_name="Censored");
proc print label data=work.cs noobs; run;
ods excel options( sheet_name="Quartiles");
proc print label data=work.quarts noobs; run;
ods excel options( sheet_name="Hazards");
proc print label data=work.hrs noobs;
var paramcd type hazardratio hrlowercl hruppercl probchisq;
run;

*** call LIFETEST again, to generate and present KM plots ***;
ods excel options( sheet_interval='NONE' sheet_name="KM_Plots");
ods select survivalplot;
proc lifetest data = approach2
  plots=survival(atrisk(outside(0.15))) method = km conftype = linear;
  time aval * cnsr (1);
  by paramcd type;
  strata armcd;
run ;
ods excel close;
```



check_ate_PSUG (2):4 - Excel
check_ate_PSUG (2):2 - Excel
check_ate_PSUG (2):3 - Excel
check_ate_PSUG (2):1 - Excel
Censored
Quartiles
Hazards
KM_Plots
Sheet1
The LIFETEST Procedure
Product-Limit Survival Estimates
With Number of Subjects at Risk
Survival Probability
aval
Planned Arm Code
Ready

paramcd	type	control_var	Stratum Number	Planned Arm Code	Total	Number Failed
OS	FASFL		1	A	225	48
OS	FASFL		2	B	226	49
OS	FASFL	-	T		451	97

paramcd	type	Hazard Ratio	95% Lower Confidence Limit for Hazard Ratio	95% Upper Confidence Limit for Hazard Ratio	Pr > Chi-Square
OS	FASFL	1.032	0.693	1.537	0.8769

paramcd	type	STRATUM	Planned Arm Code	Percent Label	Point Estimate	Transform Applied to the Survivor Function	Lower 95% Confidence Limit	Upper 95% Confidence Limit
OS	FASFL	1	A	75	.	LINEAR	.	.
OS	FASFL	1	A	50	.	LINEAR	.	.
OS	FASFL	1	A	25	37.3552	LINEAR	25.4292	.
OS	FASFL	2	B	75	.	LINEAR	.	.
OS	FASFL	2	B	50	.	LINEAR	45.5359	.

Figure 2. Sample print of summaries created using the data inflation approach.

Looking at the code used to inflate the data and perform the analysis, only the portion where we inflate the data to include what is of interest would require scrutiny and adjustment to be deployed for subsequent reviews on this, or other similarly designed studies.

CONCLUSION

As statisticians our goal is to conduct an efficient validation process while minimizing errors. Generating multiple outputs and using them to validate report summaries can be 1) time-consuming and 2) can lead to errors. Localizing our comparative results, either through use of a macro or data inflation, is more efficient for both aspects.

We feel both methods are relatively straightforward and can be modified as needed, regardless of a programmer or statistician's skill level. We asked a programming colleague to comment on both versions. The same results were obtained without difficulties, on a similarly designed study. She did mention she found it easier, or perhaps less tedious, to adjust the 'data inflation' approach, and that this approach may perhaps be more efficient to share with colleagues for other studies, and to generalize for similar analyses, from a validation perspective.

We have investigated these approaches applied to subgroup analysis based on baseline or demographic characteristics. This short paper is the result of a training exercise for our team's members. Nearly every time we have done this, a macro is written, which makes sense. However, difficulty is usually encountered when trying to consolidate the results. This suggests that perhaps more 'pre-planning' may be advantageous for colleagues that prefer to handle repetitive analyses with a macro. In closing, we leave it to users to decide which approach will be more suitable for their requirements.

ACKNOWLEDGMENTS

We would like to thank our colleagues in the Biostatistics and Statistical Programming groups we work with daily, and that provided real-world opportunities which inspired the topic itself. We would especially like to thank our colleague Meena Desai, who volunteered to run the code and offer suggestions. Their insights and guidance are appreciated.

CONTACT INFORMATION

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

Bradford J. Danner
bradford.danner@iqvia.com

Indrani Sarkar
Indrani.sarkar@iqvia.com