**Paper SP12**

# Concepts and Ideas for Developing and Maintaining a Utility Toolbox of SAS Statistical Procedures

David W. Carr, ICON Clinical Research, Redwood City, CA

## ABSTRACT

Most statistical analyses of clinical trials data require that one or more SAS statistical procedures be employed to determine significance of differences either between treatment groups or between points in time for a specific endpoint measurement. Oftentimes the shell SAS code necessary to implement such procedures is provided to the programmer either in the Statistical Analysis Plan (SAP) or in an analysis-related technical document. This is not the case however for all formal study analyses, and in these cases it can be helpful for the programmer if he/she has a library or toolbox of template statistical procedures at the ready. Of course, in order to possess such a utility the programmer must first develop and maintain the toolbox, as well as understand the procedures included therein. The purpose of this paper is to present some concepts for the development and maintenance of a SAS statistical procedures toolbox and to provide some examples of typical procedural code the SAS programmer might want to include. This presentation is based on SAS version 8.2 or above, is not limited to any particular operating system, and is intended for intermediate to advanced SAS programmers who have some familiarity with rudimentary SAS statistical procedures.

**KEYWORDS:** SAS, PROC FREQ, CHI-SQUARE, PROC MIXED, PROC NPAR1WAY, KRUSKAL-WALLIS, PROC LIFETEST, KAPLAN-MEIER, ANOVA

## INTRODUCTION

Although template SAS code for advanced statistical procedures is typically provided to the programmer in a study SAP or related technical document, there are occasions in clinical trials analysis where such code may not be provided. In these cases it would greatly benefit the programmer to have their own template code readily at hand so the loss of production time is mitigated. Additionally, it is important the programmer not only possesses such code but also understands both the relevant procedure itself and how to implement the procedure in order to achieve the desired results in the study analysis.

The following section will discuss basic concepts for developing and maintaining a statistical procedures toolbox. The next section will then present some examples of SAS statistical procedures template code that would likely be desired in such a toolbox. (NOTE: Clearly an exhaustive discussion - indeed, even just provision of sample code - of all the SAS statistical procedures used to analyze clinical trials data is beyond the scope of this paper. The sample code presented is to help demonstrate the concepts of developing a useful toolbox.) The final two sections will briefly describe some relative advantages of possessing and understanding the toolbox as well as other technical considerations.

## BASIC DEVELOPMENT AND MAINTENANCE CONCEPTS

The following are concepts that can be utilized in putting together an effective statistical procedures toolbox. These are not intended to be hard and fast rules but rather general guidelines for development and maintenance based on the author's personal SAS programming experience. It is understood that the recommendations provided allow for much individual flexibility and are open to discussion (and, as always, suggestions for improvement!).

### BEGIN DEVELOPMENT BY INCLUDING PROCEDURES THAT ARE USED FREQUENTLY

Obviously any toolbox should include those SAS statistical procedures the programmer tends to use frequently across projects. Common examples of this might be a PROC FREQ to produce a chi-square statistic or a PROC TTEST to produce a T-test statistic. Although the programmer may very well know the code necessary for procedures such as these, it is still beneficial to have template code in place (and in one place where it can reliably be found) so that a copy-and-paste of the code into the study account can be done quickly.

### INCLUDE EXAMPLES OF ACTUAL PROCEDURE USAGE AND OUTPUT

Having template code in place is a good start but including examples of where the procedure has actually been used in another project can help enhance and enforce the programmer's understanding of how to apply the procedure correctly. Ideally this would include code for the procedure itself as well as associated ODS OUTPUT statements and any DATA STEP processing that occurred afterwards to produce the statistics required for the program output.

#### TAKE THE TIME TO UNDERSTAND THE PROCEDURES AND WHAT THEY ARE REALLY DOING

Frequently SAS programmers in dealing with tight deliverable timelines do not have the chance to reflect on statistical code and what it is really accomplishing in terms of SAP requirements (particularly if it is based on template code provided in the SAP or technical document). Nevertheless, it is advisable for any programmer to take a few minutes to investigate the statistical procedures employed during project work as they are programming. This might entail studying the procedure syntax, the desired output (mock table) and the SAP or technical document for the study. In turn, the programmer should gain a better understanding of the various statements and components of the procedure as well as what the code is really intended to do. If including code from past projects, attempt to go back and read through the sections of the study SAP (or technical document) that are relevant to the statistical procedure in addition to examining the code and the resulant table output. A SAS/STAT® manual would be another valuable resource to which the programmer can refer for insight into the nuts and bolts of statistical procedures.

#### INCLUDE PLENTY OF COMMENTS TO SUPPLEMENT THE TEMPLATE TOOLBOX CODE

The inclusion of comments with template code is another way the programmer can ensure efficiency in usage of the toolbox. This is particularly true in instances when a specific statistical procedure is required for a study analysis but the programmer hasn't employed the procedure for quite a long period of time. Having relevant comments with the sample code can help the programmer recall specific details about what the procedure does and how it is to be implemented.

#### BE AWARE OF SAS VERSION CHANGES AND HOW THEY AFFECT PROCEDURAL OUTPUT

Version changes in SAS software are inevitable (as well as necessary) and sometimes result in modifications to procedural processing and output. The programmer should always attempt to identify and adjust for any changes that exist across differing SAS versions (e.g. procedural syntax, ODS OUTPUT dataset names). If marked differences for a particular procedure do exist from one version to another, it is advisable to keep a code template for each specific version where necessary.

## EXAMPLES OF TEMPLATE STATISTICAL PROCEDURE CODE

Sample template code for a number of SAS statistical procedures (as well as other relevant code) is presented in the following examples. As previously stated, this in no way constitutes what should explicitly be contained in the toolbox but rather serves to demonstrate common procedures that could be (and probably should be) included in the utility. Note that, in the example code provided, <*dataset*> refers to the input dataset name and <*TX variable*> refers to the study treatment variable (numeric).

#### PEARSON CHI-SQUARE TEST

Template code for deriving a p-value from a Pearson's chi-square test might resemble the following:

```
proc freq data=<dataset> (where=(  )) noprint;
   table <TX variable>*<categorical variable> / chisq;
   output out=chisq (keep=p_pchi) chisq;
run;
```

The p-value from this procedure is then obtained from the variable P_PCHI in the resulting output dataset. This is illustrated in the example below where the p-value for age group (AGEGROUP) by treatment arm (TXARM) is created as a macro variable.

```
proc freq data=demo (where=(txarm in (1 2))) noprint;
   table txarm*agegroup / chisq;
   output out=chisq (keep=p_pchi) chisq;
run;

data _null_;
   set chisq;
   call symput('pval',put(p_pchi,pvalue6.4);
run;
```

#### ANALYSIS OF VARIANCE (ANOVA)

The SAS code to determine a simple ANOVA p-value from the MIXED procedure could look something like this:

```
ods output Tests3=anova;
proc mixed data=<dataset> (where=(  ));
   class <TX variable>;
   model <continuous variable>=<TX variable>;
run;
```

The p-value would then be derived from the variable PROBF in the output dataset ANOVA as demonstrated below in an example for change in systolic blood pressure from baseline (VSSBPCHG). Note that although the very powerful MIXED procedure has been employed here, the resultant p-value could also be determined similarly by using the GLM procedure or the older ANOVA procedure.

```
ods output Tests3=anova;
proc mixed data=vitals (where=(vssbpchg>.));
   class txarm;
   model vssbpchg=txarm;
run;

data pvalue;
   length pval $6;
   set anova;
   pval=put(probf,pvalue6.4);
run;
```

**KRUSKAL-WALLIS TEST**

P-values are often obtained from the non-parametric Kruskal-Wallis test instead of an ANOVA when a normal distribution is not assumed. Sample code of the NPAR1WAY procedure to obtain a p-value might look like this:

```
proc npar1way data=<dataset> (where=(  )) wilcoxon;
   class <TX variable>;
   var <analysis variable>;
   output out=kwtest (keep=p_kw);
run;
```

The variable P_KW in the output dataset KWTEST carries the p-value and is used in this case to create a macro variable PVAL.

```
data _null_;
   set kwtest;
   call symput('pval',put(round(p_kw,.0001),6.4));
run;
```

**KAPLAN-MEIER SURVIVAL ESTIMATES**

Many oncology trials employ the Kaplan-Meier method to analyze survival endpoints. Sample PROC LIFETEST code for calculating totals, quartiles, the mean, standard error and product life estimates might resemble the following:

```
ods output quartiles=qrts means=mns censoredsummary=tot
           productlimitestimates=ple;
proc lifetest data=<dataset> (where=(<TX variable> in (  ))) method=km;
   time <time variable>*<censoring variable( )>;
   strata <TX variable>;
   id ptid;
run;
```

NOTE: The LIFETEST procedure requires a duration of time variable (<*time variable*>) as well as a binary censoring variable (<*censoring variable*>).

If the p-value from a log-rank test is also required, the PROC LIFETEST sample code could look something like this:

```
ods output logunichisq=pval;
proc lifetest data=<dataset> (where=(<TX variable> in (  ))) method=km;
   time <time variable>*<censoring variable()>;
   test <TX variable>;
   strata <stratification variable(s)>;              /* IF ANY */
   id ptid;
run;
```

Consider now the example table on the following page:

Table 1
Summary of Time to Disease Progression
Efficacy Population

| | ARM I | ARM II | P-VALUE |
|---|---|---|---|
| Time to Progression (Months) | | | |
| N | 24 | 19 | |
| Median | 20.5 | 11.6 | |
| 75% Time | 28.1 | NC | |
| 95% CI of Median | 12.2 – 28.1 | 9.0 - NC | |
| Mean | 19.96 | 13.86 | |
| Std Error Mean | 2.184 | 1.274 | |
| Log-Rank Test | | | 0.4706 |

For such a table, the LIFETEST procedure would be applied for time to disease progression by treatment arm. The statistics can be derived and put together in a DATA STEP in the fashion shown below (this data would then require transposing before display in the table). Note that for this example, T_PROG is the duration of time variable and C_PROG is the censoring variable.

```
ods output quartiles=qrts means=mns censoredsummary=tot
           productlimitestimates=ple;
proc lifetest data=efficacy (where=(txarm in (1 2))) method=km;
   time t_prog*c_prog(0)>;
   strata txarm;
   id ptid;
run;

ods output logunichisq=pval;
proc lifetest data=efficacy (where=(txarm in (1 2))) method=km;
   time t_prog*c_prog(0)>;
   test txarm;
   id ptid;
run;

data stats;
   length stat $16;
   set tot (in=in_to  where=(txarm in (1 2)))          /* N                */
       qrts (in=in_md  where=(percent=50))             /* MEDIAN           */
       qrts (in=in_75  where=(percent=75))             /* 75% TIME         */
       qrts (in=in_ci  where=(percent=50))             /* 95% CI           */
       mns (in=in_mn)                                  /* MEAN             */
       mns (in=in_se)                                  /* SE               */
       pval (in=in_pv)                                 /* LOG-RANK P-VALUE */
       ;

   if in_to then
      do;
         ord=1;
         stat=put(total,3.);
      end;
   else if in_md then
      do;
         ord=2;
         if estimate=. then stat=' NC';
         else stat=put(estimate,5.1);
      end;
```

```
   else if in_75 then
      do;
         ord=3;
         if estimate=. then stat=' NC';
         else stat=put(estimate,5.1);
      end;
   else if in_ci then
      do;
         ord=4;
         if lowerlimit=. and upperlimit=. then stat=' NC - NC';
         else if lowerlimit=. then stat=' NC - '||
            compress(put(upperlimit,5.1));
         else if upperlimit=. then stat=put(lowerlimit,5.1)||' - NC';
         else stat=put(lowerlimit,5.1)||' - '||
            compress(put(upperlimit,5.1));
      end;
   else if in_mn then
      do;
         ord=5;
         if mean=. then stat=' NC';
         else if txarm=1 then stat=put(mean,6.2);
         else if txarm=2 then stat=put(mean,6.2);
      end;
   else if in_se then
      do;
         ord=6;
         if stderr=. then stat=' NC';
         else if txarm=1 then stat=put(stderr,7.3);
         else if txarm=2 then stat=put(stderr,7.3);
      end;
   else if in_pv then
      do;
         txarm=3;
         ord=7;
         stat=put(probchisq,pvalue6.4);
      end;
run;
```

## ADVANTAGES OF POSSESSING A STATISTICAL PROCEDURES TOOLBOX

Development and maintenance of a statistical procedures toolbox is advantageous to the clinical trials SAS programmer for a number of reasons. These include but are not limited to the following:

- As mentioned previously, possession of such a utility can help to decrease the amount of production time that might be lost in the event explicit sample code is not provided in the SAP or technical document (or in cases where ad hoc tables need to be produced very quickly).
- The toolbox allows the programmer access to a large amount of template code that can be accessed quickly since it is housed in one central location (as opposed to possibly scouring other project accounts trying to find relevant code).
- The programmer should be more knowledgable about the statistical procedures contained in his/her toolbox and thus should be able to implement them more accurately and efficiently in new projects.
- The knowledge and understanding of the various statistical procedures gained by the programmer during development and maintenance can result in more credibility among colleagues, including project statisticians, study sponsors, and fellow SAS programmers.

## OTHER SUGGESTIONS/CONSIDERATIONS

Programmers wishing to create their own toolbox of statistical procedures might also take into account these additional considerations.

- Although the text and examples provided previously imply development of statistical template code housed in a single file, another option would be to create a library with various programs containing related items. For example, a library might contain a program with code to generate Kaplan-Meier statistics, a program with code to generate ANOVA results (PROC ANOVA, PROC GLM), a program with code to generate ANCOVA results (e.g. PROC MIXED), etc.

- After becoming proficient in the implementation of various statistical procedures, the programmer may want to develop a macro library that generates multiple statistics and can easily be called into programs (resulting in, of course, less program code!).
- Consult or even collaborate with other programmers or statisticians known to be accomplished in the use of SAS statistical procedures. This could potentially lead to a more comprehensive toolbox or library that can be shared by an entire programming group, the result of which should be gains in programming speed and efficiency company-wide.

## CONCLUSION

In summary, most SAS programmers in the pharmaceutical research industry would be well-served to develop a toolbox or library of template statistical procedures commonly used in clinical trials analyses. Ownership of such a utility could result in enhanced programming speed and efficiency as well as facilitating programmer knowledge of the various procedures and the results they are designed to produce. Specific procedures to be included in the toolbox could vary greatly depending on the individual programmer, the company for which they work, and the types of analyses (or clinical trials) in which the programmer is typically engaged.

## ACKNOWLEDGEMENTS

I would like to extend a special thanks to the statistical programming management team at ICON for their ongoing support of progress, innovation and professional development, and to my wife Laurey, without whose unwavering support none of this would be possible.

## CONTACT INFORMATION

The author can be contacted at:

David W. Carr
ICON Clinical Research
555 Twin Dolphin Drive, Suite 400
Redwood City, CA 94065
Work Phone: 215.616.4966
E-mail: david.carr@iconplc.com