A SAS * SOLUTION: BUILDING A FLEXIBLE/STANDARD SYSTEM TO REPORT THE ANALYSIS OF EFFICACY DATA FOR CLINICAL TRIALS

Judy A. Sullivan, Pfizer, Inc.
Naitee Ting, Pfizer, Inc.

Introduction

Clinical trial data can be divided into two broad categories, Safety data and Efficacy data. A pharmaceutical company is responsible for proving to the Food and Drug Administration that a drug is effective against a particular indicated disease and that it is also safe for patients to take. In order to accomplish this task clinical trials are carried out to collect data concerning a drug's safety and efficacy. Data collected through clinical trials are analyzed and reported in statistical tables that become part of the new drug application (NDA). A NDA is composed of many separate studies setup to research a drug candidate. A special data processing need of a NDA is the necessity to repetitively produce data tables that have the same or nearly the same format.

An example is presented which illustrates the use of a hierarchical directory structure, a data-lookup table and SAS Macro code to create a system that provides standard output with study-specific routines necessary for data analysis of efficacy data and provides a way to repeatedly produce large numbers of tables automatically using the same piece of code among studies.

The Problem

Analysis of efficacy data is specific to the disease being studied and also specific to the individual study within a clinical trial. Hundreds of tables are produced representing various analyses based on different subsets of patients or different statistical designs. The conventional approach to this problem is to run an individual SAS program to each table or group of tables within a particular study. For succeeding studies, this code was simply copied and modified where necessary and run again, producing a great many tables but along with it a great many SAS files. This method posed difficult problems in terms of software maintenance and validation.

Our approach is to write generalized programs that need never be altered regardless of the study or drug project which uses the software. Instead of maintaining separate software for each table within each trial, this system is designed to produce many tables through the use of very generic SAS macros that receive specific information from a data lookup table. By using a structured, top-down programming approach, the software to produce a group of tables is modularized into separate macro routines based upon their applicability to other drug projects.

The challenge was to devise a way of accommodating the need for specificity within a system designed to optimize standardization. We have prototyped such a process within the established inhouse reporting system, DOF/TSR (Data Output Flow/Trial Submission Reporting).

An example of how this system produces a report is described below.

System Design

DOF/TSR makes extensive use of the SAS system's macro processing capabilities. All SAS programming for report generation is done using SAS macros. SAS/AF" is used as a front end to control the program flow, set up external files and to obtain key information from the user via a SAS dataset which acts as a data lookup table.

The user interface is written in SAS SCL that provides the user with a series of menus similar to those in SAS/ASSIST*. When a user executes a program from one of the menus, information is read in from the lookup table and passed as global macro variables to a SAS file to be read in later by a %include statement from one of the generic,
external SAS macro programs (fig. 1). The SAS
macros software is written in modules with low-level
modules nested in higher level modules and the
level pertains to how general or specific the macro
is.

For instance:

% macro table24
% analyze
%
glm
% report
%mend table24

illustrates three modules nested within a main
program. A directory tree structure is used to
physically place program modules in levels
according to how specific and/or general they are to
drug projects and studies. Program modules are
placed in “Macro” subdirectories, the top-level
programs are the most general, the low-level ones
are specific to a protocol or study. This directory
structure is organized so that different drugs and
studies within drugs are located below the highest
level (the “Company” level) (fig. 2).

SAS modules that contain generic subroutines are
placed in one of the Macro subdirectories and can
reside in any one of the three levels (Company,
Drug or Study) depending how generic it is. The
highest level modules can be used by all drug
projects within the company, this level would
contain macros like standard safety tables for
instance. Where the format is basically the same
for any drug study (in the example above \%table24
is a company level module). The next level down,
the Drug level, also contains a Macro subdirectory
that holds modules that are less generic than the
Company level but still generic enough to be used
by all studies within that particular Drug project.
The lowest level, Study, contains modules used
only by that Study.

The Sas Mautosource System Option is used to
enable the SAS macro autocall facility to create a
search path starting at the lowest, most specific
level and traversing upwards to the highest most
general level (NOTE: the search begins at the most
specific level first).

Options mautosource sasautos=
"macs.&study","macs.&drug", "macs.&company",sasautos);

where macs&* resolve to librefs that point to the
Macro subdirectories at the Study, Drug and
Company levels. The advantage is that program
modules don't require renaming whenever a
modification is made thus eliminating the need to
modify the top level module that makes the call.

The example in this paper illustrates how the
system makes use of these hierarchical directory
levels and data driven programs to create a
standard and flexible process for mass production
of efficacy reports.

The Efficacy Analysis and Data

This paper uses as an example a statistical report
based on a clinical study of a drug used to treat
arthritis sufferers (fig. 3).

The effectiveness of the drug is assessed using
twenty eight measurements, defined by the medical
community and are referred to as efficacy
parameters. This example focuses on a report
using the first five and are commonly referred to as
the “primary” efficacy parameters.

Clinical trial data is collected to assess a drug’s
potential effectiveness for the disease being
studied. A statistical analysis is performed
comparing different doses of the drug. The
example here illustrates one part of the analysis, a
comparison of disease activity taken before the
patients start taking the active drug (referred to as
Baseline) compared with values obtained after the
drug has been taken for a certain prespecified
length of time (referred to as Final) or when a
patient discontinues treatment (referred to as
Endpoint).

The input data is processed beforehand into six
arrays each with 28 elements representing the 28
efficacy parameters. The array names are: BA and
BB for baseline values (Final and Endpoint
respectively); F for Final values; E, Endpoint
values; DA, difference between Baseline and Final
and DB, difference between Baseline and Endpoint.

These parameters have been mapped to text using
Proc Format so that their numeric representation as
array subscripts can be translated to text on the written report:

```sas
proc format;
value linetext
1 = 'Physicians Assessment of Disease Activity'
2 = 'Patient Assessment of Disease Activity'
3 = 'Number of Painful Joints'
4 = 'Number of Swollen Joints'
5 = 'Visual Analog Pain Scale'
28 = 'Household Activity';
```

**SAS Macros and Program Flow**

There are many variations of this table that could be produced utilizing the same process simply by changing some of the user-defined macro variables in the data lookup table. An example of some of the Macro variables defined are:

```sas
%let frstparm = 1;
%let lastparm= 5;
%let depvar1 = compdrug;
%let depvar2 = center;
%let point = endpoint;
%let trtvar = digdose;
```

where &depvar1 and &depvar2 are used in the analysis of variance part of the program; &point decides which array of data to use Final or Endpoint; and &frstparm and &lastparm indicate the numeric values of the first parameter and the last parameter to use in the table, e.g. use parameters 1 - 5.

The report can be carved into three basic sections; 1) A section that is data-derived, 2) a section generated using user-defined macro variables and 3) a section that is "hardwired" or remains standard from one report to another. For this particular report, only the column headings are hard-coded; the rest of the report is data driven or derived from macro variable definitions. PROC PRINT is used instead of PUT statements so the number of treatment groups or number of efficacy parameters appearing on the report can vary according to what the data contains without having to modify code. The second section is exemplified by titles, footnotes and the type of report it is, Baseline vs. Endpoint or Baseline vs. Final. The table type is determined using the value of &point which then sets up the proper variables to use within the first program:

```sas
%if &point = final %then
  %do;
  %let x1=ba; /*final baseline */
  %let x2 = f; /* final value */
  %let x3 = da; /* difference */
  %end;
%else
  %if &point = endpoint %then
    %do;
    %let x1=bb; /* end baseline */
    %let x2 = e; /* end value */
    %let x3 = db; /* difference */
    %end;
```

The parameters (first column) are derived from &frstparm and &lastparm which get translated using the Proc Format statements described above by:

```sas
%do
  counter = &frstparm %to &lastparm
  txt=put (&counter,line); 
%end;
```

Treatment, titles and footnotes are also defined through macro variables and can therefore be modified without touching the SAS program. Their values are simply passed into the program as &title, &footnote and &trtvar.

The remainder of the table is the result of the actually analysis of the data that are generated from the SAS macros described below.

To do the right analysis the correct set of arrays and the appropriate parameters for the table must be brought in. The following code is used to capture the right variables in a keep list, again pulling information from the lookup table, dataeff is the preprocessed data as described above where the arrays are defined:

```sas
data work.eft;
  set &lib.&study.dtaeff;
  keep &trtvar &depvar1 &depvar2
  %do i = &frstparm %to &lastparm;
  %do;
```
%macro efttab;
%do i = &frstpart %to &lastparm;
  %dfiles (bsline=&x1.&i,
          last = &x2.&i,
          diff = &x3.&i
          dsnout= work.d&i);
%end;
proc print .... etc.
%mend efttab;

The macro Dfiles then calls a protocol specific module that sets up the appropriate orthogonal contrast coefficients in %contrast. Each study within the drug project requires its own specific set of comparisons that are unique to the study design. Without modifying the %efttab code we can capture this specific information to produce the same table for many different studies.

Conclusion

Having a directory structure that allows for both specific and general SAS modules for analysis and reporting, the SAS Mauto source facility is used to automatically select the appropriate SAS Macro. User-derived macro variables are set up to "feed" the macros at any level so as to be able to use the same general purpose macro by just giving the appropriate values for variables. The analysis using PROC GLM can be specific to individual trials by modularizing the SAS programs placing the &contrast macro at the [...STUDY..MACRO] level.

Notice

*SAS, SAS/ASSIST and SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.
Figure 2. Directory Tree showing Levels of Macro Subdirectories

Figure 1. How DOF/TSR system works
Table XXX

Comparison of endpoint-baseline changes (intent to treat) between treatments for primary efficacy parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TREATMENT GROUP</th>
<th>N</th>
<th>BASELINE MEAN</th>
<th>ENDPOINT MEAN</th>
<th>END-SSL CHANGE</th>
<th>STD ERR</th>
<th>OVERALL P-VALUES</th>
<th>EA. DOSE VS. PCBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician assessment of disease activity</td>
<td>LOW DOSE MEDIUM DOSE HIGH DOSE PLACEBO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient assessment of disease activity</td>
<td>LOW DOSE MEDIUM DOSE HIGH DOSE PLACEBO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of painful joints</td>
<td>LOW DOSE MEDIUM DOSE HIGH DOSE PLACEBO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of swollen joints</td>
<td>LOW DOSE MEDIUM DOSE HIGH DOSE PLACEBO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visual analog pain scale</td>
<td>LOW DOSE MEDIUM DOSE HIGH DOSE PLACEBO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Example of a statistical report

Figure 4. Program flow