ABSTRACT

The capabilities and innovative techniques of two generic, real-world, SAS® Macro Language reporting applications used internationally by a major pharmaceutical firm are presented. Together, these applications can handle the majority of listings and tables found in a typical clinical submission. Users of these applications find their appeal to traditional paper plume reduced to a fraction of the time traditional methods used to take. The results they now achieve are more flexible, consistent, reproducible, aesthetically pleasing, and more accurate than results users accomplished using conventional SAS programming, or worse, hand-typing.

Generic reporting systems are emphasized, but the techniques presented here have a wide variety of uses. Example calls to GENLIST, which is a generic listing system, and GENTABLE, a generic table-producing system (written by this paper's author and Amy Frueh, also from DZS) are presented along with some macro utilities. Using our approach, users need only to access their data and describe the contents of their reports. Attractive presentations are derived from end-user specifications without unnecessary user-overhead, or forcing static report layouts on the clinical department. This paper's goal is not to teach the SAS Macro Language, but to show powerful techniques and possibilities using it.

INTRODUCTION

Since the introduction of the first comprehensive release of the SAS system in 1976, programming departments in larger businesses have tried to automate and supplement SAS-supplied reporting with regard toward presentation and production. This is especially true within the pharmaceutical industry, which places a high value on timely, accurate and aesthetically pleasing reports to submit to the FDA and foreign agencies. Delays due to the creation of individually prepared, time-consuming, faulty, or hard-to-read reports can be quite costly.

New releases of the SAS system have shown enhanced report creation capabilities, such as improvements in the PRINT and TABULATE procedures and the NEW REPORT procedure. However, the bulk of presentation and submission-ready reports are still not provided directly from SAS procedures. The personnel generating the reports often resort to tedious, complicated PUT statement programming or worse, to having the reports typed. The drawbacks to those methods include the excessive human intervention required and the increased risk of error. Many other drawbacks exist to those traditional methods, and have been discussed in papers such as "Simple Construction and Modification of Computer Generated Tables". Our solution to the problem of generic report production has been the use of the SAS Macro Language to create applications that marry the SAS Macro Language with SAS dataset and procedure output. GENLIST and GENTABLE are two such applications whose goals, benefits, and implementation will be discussed.

GOALS - GENERIC REPORTING APPLICATIONS

The basic goals of generic reporting applications such as GENLIST and GENTABLE include the following:

- Ease of Use - To create a system that is easy to use, the developers must have thorough knowledge of the background, needs, talents, and training of the target user community of the application. The resulting application should be one that the user community wants to use to expedite completion of their tasks.

- Accuracy - To achieve and maintain user confidence, the application must be written so that it is "bullet-proof" as possible. This includes diagnosing user input errors and providing descriptive messages for special conditions or potential errors. Extensive quality assurance and validation should be done before each release of the application.

- Flexibility - The application must be flexible enough to meet the user's present and future needs. For report applications, this means simple alteration of titles, footnotes, display and content. Flexibility also refers to operating system and environment under which the application runs. Ties to particular features in the current operating environment should be avoided to allow future application porting.

- Speed - Time from the conception of the needed report to its actual production should be shorter than the current methods used.

- Consistency - Reports should present findings in a logical, predictable manner regardless of which particular clinical group generates them. This also aids in international reporting where tables and listings may be submitted from disparate clinical groups to various governing agencies. The generic application would have a "standardizing" effect on the way results are presented.

Discussed below are two examples of powerful systems written with these goals in mind.

GENLIST, A GENERIC LISTING SYSTEM

The user requirements for GENLIST included the goals listed above and the following criteria:

- A system that would provide a powerful, capable alternative to the PRINT and REPORT procedures for detailed demographic, laboratory, and other listings.

- Ability to print the value of more than one variable in a commonly-labeled column. Dynamic variable value displays in the column labels were also required.

- The maximum amount of information available in the SAS environment should be extracted, letting the caller use familiar SAS FORMAT and LABEL statements to pass information to the application.

- The burdens of exact column placement of values, balancing the multi-line column labels, calculating the lowest column width needed for a variable, and other tedious tasks should be removed from the user.

- Handling out-of-range and change-from-baseline flagging based on user requests was required.

- Many other bells and whistles such as left-justified or centered titles and footnotes with dynamic variable value substitution, the placement of text that spans several columns, solid line placement for subsequent inclusion in WordPerfect® documents, page eject management based on user-defined groupings, and more would be necessary.
SAMPLE CALL

As the sample call to GENLIST in Appendix 1 shows, all the goals were attained. The input data, SAS program fragment, and output report are shown to illustrate the high degree of integration that can be achieved with the SAS system environment. The entire SAS job could not be shown due to limitations on the length of contributed SUGI papers. The section of code omitted simply merged demographic and vital signs data and assigned labels and formats. Please note that the solid lines on the report were not typed and not generated by the TABULATE procedure; they were applied by the macro (GENLIST does not use PROC TABULATE; it uses an offset datastep to have complete control of the presentation).

Please note that GENLIST presents the user with the following situation:

- The only “foreign” statements in a SAS job are the macro request (XGENLIST); and the actual macro call (Xgenlist(...)). Notice that there are not any XLETs and that intelligent parsing of the macro parameters avoids the tedious and limitations of repeated arguments (i.e. title1, title2, title3, etc.)

- To associate custom column labels and formats, SAS FORMAT and LABEL statements are used instead of having the user specify them in the macro call. Since most of the GENLIST macro parameters have defaults, the only required parameters are COLVARS= (the column variables to print) and PGHEADS= (the title lines). The rest of the parameters are only needed for special features and custom presentation for that report.

- Delimiters and display flags have been chosen that mimic, as closely as possible, those used in base SAS software. In the COLVARS= parameter, for instance, the dot (.) prefix on a column variable means “print it on front of the column”.

- Column variables, title lines, and footnotes can be easily moved about, deleted, and added. The resulting report will always be of high quality, integrating the user’s request for display variations as desired. The SOLIDLINES option can be used on listings that will be incorporated into WordPerfect documents, to facilitate correct centering and meet the desire for solid lines without human intervention.

The process and techniques used are as follows:

1. Parse and diagnose user input (parameters), sensing the presence of special feature requests (like normal range flagging) and options (like centering, solid lines). Extensive use of the BREAKUP macro is made (see its presentation later in this paper).

2. Execute a PROC CONTENTS with the OUT= specification and load needed information into macro variables for the user-referenced SAS variables. This is the moment that the macro finds out about user-associated formats and labels. Please see the presentation of FINDTYPE below, a macro routine similar to the one used in GENLIST for this purpose.

3. Using any available information from the PROC CONTENTS, execute a datastep that discovers the widest column space each column variable needs. See the presentation of FINDFMT below in this paper for the method used. This method relies on the use of decimal places in the width of the parsed SAS labels and the width of the data printed in the column. In Appendix 1, the formatted data width of VISIT was wider than its label “VISIT”. This causes the macro to use the data width for that column as the total column width. For “Diastolic BP” in the example, the label text was wider than the combined data widths of the BP value and High/Low flag, so use the label width in calculating the column layout.

7. If there are spanning headers (SPANHEAD=) or SOLIDLINES, make these adjustments to the column header routine. Note that information needed and stored during a GENLIST call is kept in macro variables. This prevents adding actual SAS variables to the report dataset that may clash with user variable names. These macro variables are kept local to particular routines whenever possible to avoid unnecessary filling the global macro variable table.

8. Construct the detail record SAS PUT statements. Since the column label and data widths are known, it is a straightforward matter to calculate the proper column placements.

9. Handle end-of-page control logic, integrating the observation count for sub-group data defined by the NOSPLIT= parameter (if one has been specified) and number of footnote lines. Allow constructed datastep to run.

While it is true that GENLIST is thousands of lines of macro code long, the actual building blocks were not that difficult to create. By breaking large system specifications into relatively small, manageable, well-defined pieces or modules, one can create powerful, reusable code. BREAKUP, FINDTYPE and FINDFMT presented later in this paper are good examples of these sorts of modules.

For now, I am trying to communicate the power that SAS offers to the applications developer and the level of integration the user should expect from a well-designed application.
GENTABLE, A GENERIC TABLE SYSTEM

GENTABLE is a more recently developed SAS macro application that builds on the goals set for GENLIST. Underlining the power of generic, modular programming, GENTABLE actually uses GENLIST as its display formatter and output generator. Using GENLIST within GENTABLE dramatically reduced the development time needed for GENTABLE while enabling it to quickly show attractively presented results.

The theory and goals for GENTABLE included all those for GENLIST plus the following:

- Move to a more "English-like" user-interface than GENLIST. For example, if a user wants two or more variables printed in the same column, the "&M" sign is used in GENLIST, the word "NEXTTO" is used in GENTABLE.

- Where GENLIST is excellent at producing detailed data listings, the purpose of GENTABLE would be to produce summary statistics tables based on user requests. The design of the system would allow the request of any combination of supported statistics for any number of variables printed to any precision. Any number of classification groupings would also be supported.

- Handle all necessary calls to SAS procedures, such as MEANS, FREQ, UNIVARIATE, GLM. Perform data transposing (NOT using the TRANSPOSE procedure) where needed to combine all the information retrieved from the statistical procedures into a single dataset.

- Never perform any SAS procedure output-stripping (using PROC PRINTTO and then a datastep to read the output) to acquire statistics. Instead, the output data combine all the information retrieved from the statistical procedures into a single dataset. This will dramatically improve the stability and dependability of the system.

- Continue the use of modular-programming techniques to facilitate easier quality assurance and the future inclusion of data from other statistical procedures.

- Provide the most efficient user interface possible. This includes implementing "short-hand" methods of asking for repeated sets of statistics, and continuing to gather as much information as possible from the SAS environment.

- Begin with a X-by-Y matrix style interface that would allow easy transposing of variables that lie across the page with those that run down the page.

Inspecting the user's calls to GENTABLE and subsequent output (Appendix 2) should show that these goals have been met. Even though GENTABLE development has spanned less than a year, impressive results are already being realized through its use. Very favorable comments have come from the user community, and the users are producing better looking and more accurate tables for submission faster than before. There are statistics needed that are not yet generated by GENTABLE, but due to its modular design, these will be fairly easy to integrate.

The basic process and techniques used in GENTABLE are as follows:

1. Parse user input parameters. Load requests for statistics (N, MEAN, MEDIAN, FREQ, etc.) for each variable into macro variables. Make a note (macro variable with a 0 or 1) of which SAS procedures will need to be run by GENTABLE based on the requested statistics.

   DISPLAY = Statistic keywords optional rounding factors <variables>,

   For example, if the user wanted to see the N, MEAN (to 2 decimal places), MINIMUM next to the MAXIMUM and the PVALUE (from GLM's ISMEANS) next to the SIGNificance of the SAS variables SYSBP and DIABP:

   DISPLAY = N MEAN .01 MIN NEXTTO MAX
   PVLSMEANS NEXTTO SIG &SYSBP &DIABP,

   would be specified. Currently, over fifty statistic keywords are implemented.

2. Construct BY and TABLE statements from SAS variables in the TITLE= (page title lines), the DOAM= (the "M" variables of the matrix), and the ACROSS= (the "X" variables of the matrix) parameters, if any.

3. Execute any necessary statistical procedures accumulating the results in a single dataset. For illustration I will show the basic process for the request DISPLAY = FREQ <DISC>. This is a request that GENTABLE look at the user's variable DISC in the user's dataset and print the counts of all values of DISC.

   I. Perform a PROC FREQ for the variable based on any DOAM and ACROSS variables.

   II. Using the FINDTYPE macro, find out if the user has associated a format with DISC. Load the discrete values for DISC from the PROC FREQ output dataset into macro variables. If there is a user format, execute a datastep using PUT(value, user-format.) to load the formatted values as potential column labels for the results. In the first GENTABLE example note the placement of the formatted values for the SAS variable "DISC" as column labels for the cell counts.

4. Based on the DOAM= and ACROSS= parameters and the STATSACROSS (request for statistics to run across the page) and VARSDOWN (request for variables to run down the page) options, perform dataset transposing if needed.

5. Round the displayed statistics to default precision, if the user has not specified their own precision.

6. If the statistics are to run down the page, create a column labeled "Statistic" to display the types of statistics. Also, convert the statistic results to character equivalents (carefully, using the PUT function AFTER rounding the values) to allow them to display in a common column in the GENLIST call.

7. Setup and call GENLIST to produce the report, considering all other user parameters.

A more detailed explanation of how GENLIST and GENTABLE operate can be given at a later date, if user-group interest warrants it. This discussion of GENTABLE would not be complete without reference to the SUGI 12 paper by Bernard Costa entitled "TABLETS - A TABLE GENERATOR". The concept behind Bernard Costa's excellent TABLES table macro has contributed to the theories used in GENTABLE.

A FEW UTILITY MACROS

Please see Appendix 3 for the source code and example calls to BREAKUP, FINDTYPE and FINDFMT. Macros similar to these are at the core of GENLIST and GENTABLE and should have equivalents in most major SAS macro applications. They comprise less than 200 lines of macro code, but illustrate the type of modular programming that lends itself to larger systems development. Their purposes are:

BREAKUP - Splits a macro variable value (BUTEXT) into as many macro variable pieces (NTEXT pieces will have names that begin with the &BROOT) as there are in the macro variable based on the specified delimiter (DELM). See the BESTFMTS macro in Appendix 3 for an example call.

FINDTYPE - Loads the SAS variable types (into macro names prefixed with &TROOT and suffixed with numbers
The payback for time and talents invested in well-designed, modular generic applications can be large. Making the best use of all the rudiments provided by the programming language and good end-user interaction is essential to the success of these applications. We approach our application design systems from the aspect of what would the users like the application to do for them, then work back to how we will accomplish the many small tasks that make up the goal.

By providing very functional, powerful, and flexible applications, end-users are finding they have more time to spend on tasks other than tedious manual report preparation. Now, users can quickly and easily try out different statistical report layouts, and their submissions are looking more consistent. By incorporating solid lines on the listings and tables, the outputs easily become centered WordPerfect sub-documents in clinical reports, and the human process of drawing lines on the output is eliminated.

CREDITS

Christian M. Forster from DZS Computer Solutions is the designer and developer of GENLIST.

Christian M. Forster and Amy Frueh from DZS Computer Solutions are the designers and developers of GENTABLE.

REFERENCES


2Costa, Bernard "TABLETS - A Table Generator", SUGI 12 Proceedings.

SAS is a registered trademark of the SAS Institute, Cary, NC, USA in the USA and other countries. * indicates USA registration.

WordPerfect is a registered trademark of WordPerfect Corporation, Orem, Utah, USA in the USA and other countries. * indicates USA registration.

CONTACTS

Doron Z. Steeper, President or Christian M. Forster, Medical Systems Manager/Analyst DZS Computer Solutions, Inc. (908) 356 - 6961 1661 Route 22 West Bound Brook, NJ 08805
APPENDIX 2
GENTABLE Sample Calls

```plaintext
proc format;
value disc = 'Adverse Reaction' 'Uncooperative' 'Completed';
value text = 'Drug 1' 'Drug 2';
data demo;
input week: 1 ; tllit=2 ; pat=1 ;eli $c:::3; .pCp: 130; d,aoc:92; output;
format text litt. also disc.
label litt = 'No of [Patient Status at Endpoint]';
diab = 'Diastolic BP';
syst = 'Systolic BP';
run;
```

**TABLE 1**

Patient Status at Endpoint

<table>
<thead>
<tr>
<th>Treatment Group</th>
<th>No of</th>
<th>Adverse Reaction</th>
<th>Uncooperative</th>
<th>Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drug 1</td>
<td>5</td>
<td>2 (40%)</td>
<td>0 (0%)</td>
<td>3 (60%)</td>
</tr>
<tr>
<td>Drug 2</td>
<td>4</td>
<td>0 (0%)</td>
<td>3 (75%)</td>
<td>1 (25%)</td>
</tr>
</tbody>
</table>

**TABLE 2**

Vital Signs - Blood Pressure Statistics

<table>
<thead>
<tr>
<th>Systolic BP</th>
<th>Mean</th>
<th>SD</th>
<th>Range</th>
<th>n</th>
<th>Disc</th>
</tr>
</thead>
</table>
| 150         | 132  | 7.9| 125-150| 96 | Diastolic BP
| 120         | 92   | 7.1| 80-130 | 96 | Systolic BP

**TABLE 3**

Sample Calls

```plaintext
proc format;
value disc = 'Adverse Reaction' 'Uncooperative' 'Completed';
value text = 'Drug 1' 'Drug 2';
data demo;
input week: 1 ; tllit=2 ; pat=1 ;eli $c:::3; .pCp: 130; d,aoc:92; output;
format text litt. also disc.
label litt = 'No of [Patient Status at Endpoint]';
diab = 'Diastolic BP';
syst = 'Systolic BP';
run;
```
Macro findfllt(varname, varroot, tltportion, type, fmt); 

/* FINDFLLT: COPYRIGHT 1999, DSS Computer Solutions, Inc. By C.Forster; */

/* Macro findfllt(varname, varroot, tltportion, type, fmt); */

* Generates a data null to get macro information of the dataset; * 
* data null; * set names even checks; * 
* retain _cft; _cfltest(_cft); _ctens(_cld) _cfl1(1); 
* array _ctvals{_n} S 132 _ctv1-_ctv{n}; X· last values •• ; 
* array _citens<{in} _cfl1:_cUt{n}; '; •• To hold tout lengths •• ; 
* retain all cH{n} ddi'n cfd{n} 0; 
* ; 
* Call BESTFNTS to find out the best formats for these variables */ 
* bestfllt(test.study sex); ** Call BESTFNTS to find out the best formats for these variables */ 
* test or reports or * 
* for variable std no format assoc., total length: 3, Use 3.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */ 
* for variable std no format assoc., total length: 1, Use 1.0 for output format. */