Paper 018-31

Developing, Managing, and Evaluating a Standard Macro System

Albert Mo, Abgenix Inc., Fremont, CA

ABSTRACT

Standard Macro System (SMS) has become a very popular, effective, and critical tool at Biostatistics and Statistical Programming departments in many pharmaceutical, biotechnology, and medical device companies. It is a system designed to automate the generation of standard summary tables, listings, and graphs (TLGs). In this paper, both technical and non-technical aspects of developing a SMS will be discussed. The topics include definition, creation & evolution, structure, and benefits of SMS, along with recommendations for developing, managing, and evaluating SMS, Software Development Life Cycle (SDLC), and challenges/opportunities associated with SMS.

INTRODUCTION

DEFINITION OF SMS

SMS is a validated system consisting of fully integrated end-user and utility SAS macros used to automate the generation of TLGs. It is globally developed, centrally managed by an in-house SMS team. It is different from a Standard Macro Library, for the latter generally contains a set of independent, not fully integrated, SAS macros. It is also different from the project (therapeutic area) level SAS macros, for the latter are usually developed, maintained, and used individually by separate project teams.

CREATION AND EVOLUTION OF SMS

A typical Clinical Study Report (CSR) can involve hundreds, if not thousands, of TLGs. If every statistical programmer had to write individual TLG programs from scratch, there would be many challenges for members within a clinical study team. The challenge for statistical programmers is to produce TLGs with great speed and total accuracy. The challenge for TLG reviewers is to be assured with great confidence that all TLGs are produced consistently and accurately. And the challenge for managers is to ensure all parties are satisfied and the project is completed on time and within budget. This obviously created a huge amount of work and stress for all parties involved.

Company xxx Table XX (page x of y)
Protocol xxx Title of Table
(Population: xxx)

	Treatment A (N=XX)	Treatment B (N=XX)	P-value Treatment A vs. Treatment B
Variable N	XX	XX	X.XXX
n(%)	XX (XX.X%)	XX (XX.X%)	
Variable 2			
N	XX	XX	X.XXX
MEAN	XX.X	XX.X	
STD	XX.XX	XX.XX	
MEDIAN	XX.X	XX.X	
MIN-MAX	(XX-XX)	(XX-XX)	
95% CI	(XX, XX)	(XX, XX)	

Footnotes of Table:

Table 1. Sample Summary Table Shell

Let's use Table 1 as an example. Table 1 is a typical table shell for summary tables with statistics about different treatment groups. It would take much time and effort even for an experienced SAS programmer to create and validate the customized TLG program for Table 1. And it is most likely that other summary tables, similar to Table 1, will be requested with the following variations:

- Adding new columns of “Treatment C” and “Total”
- Adding new columns of “P-value Treatment A vs. Treatment C” and “P-value Treatment B vs. Treatment C”
- Adding Variable 3 (categorical variable) and Variable 4 (continuous variable)
- Adding a BY variable of Visit
- Sub-grouping the patient population to be patients with Age < 65
- Using a different statistical method to generate the p-values

To tailor for these variations, statistical programmers would be very busy searching for similar customized TLG programs, copying and creating new customized programs, and continue to spend valuable time and effort.

Over time, Biostatistics and Statistical Programming departments have realized these kinds of practices carried too much risk, wasted too many resources, and added extra burdens on people involved. Thus, they have re-thought their strategies and practices, centralized their development effort, and generalized those customized TLG programs into a Standard Macro System with the following two goals:

1. **The application programs can simply call SMS to automatically generate TLGs.**

2. **SMS can be used and re-used by all users, for all clinical studies, and under all therapeutic areas.**

STRUCTURE OF SMS

A SMS typically consists of three levels of standard macros stored in the SAS macro libraries:

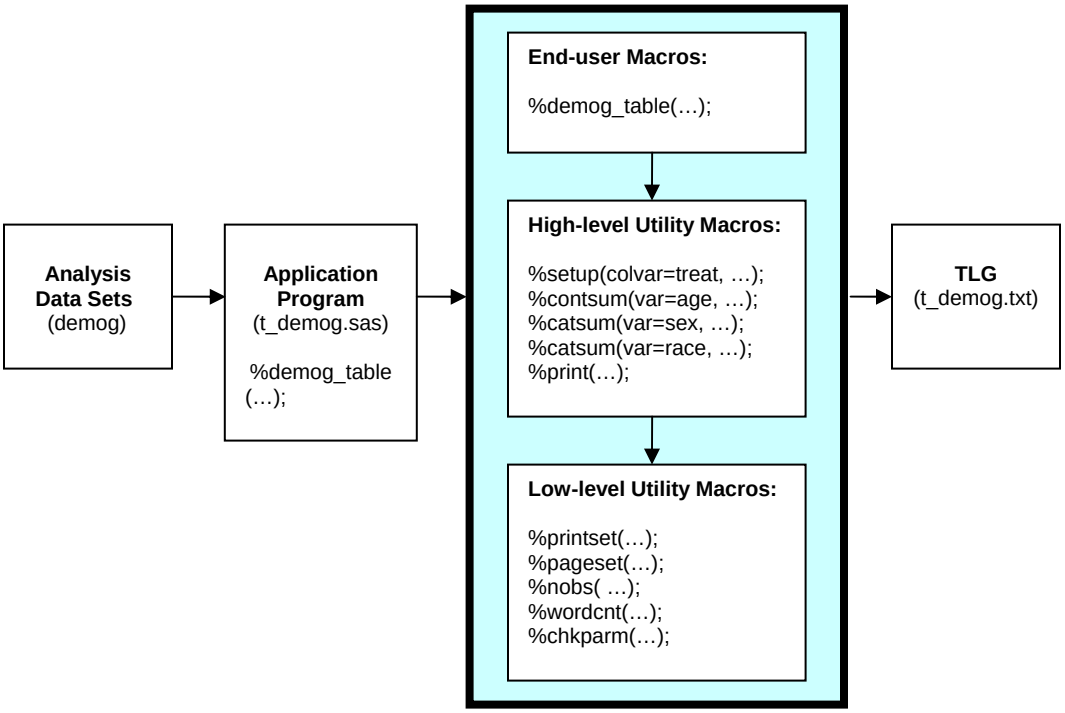


Figure 1. SMS System Flowchart

1. End-user macros:

These SAS macros are called directly by the application programs. They have common, consistent, and friendly interfaces with the users. To use them, users simply supply (pass) a list of keyword parameters. Most of the keyword parameters have default values so as to make the job of an end-user easier.

Examples of end-user macros are:

- For summary tables:
  %ae_table: To produce frequency statistics for Adverse Events

%demog_table: To produce summary statistics for demographic and baseline characteristics
%conmed_table: To produce frequency statistics for concomitant medications
%disp_table: To produce frequency statistics about patient disposition (exit status)

- For listings:
  %ae_list: To produce listings for Adverse Events
  %demog_list: To produce listings for demographic and baseline characteristics
  %conmed_list: To produce listings for concomitant meditations

- For graphs:
  %scatterplot: To produce the scatter plots
  %kmplot: To produce the Kaplan-Meier plots for survival analyses
  %longplot: To produce the longitudinal plots

2. High-level utility macros:

These SAS macros are called and shared by the end-user macros. They are the major building blocks of SMS. They generally have bigger scopes and perform larger tasks.

Examples of high-level utility macros are:

- %setup: To initialize macro variables, subset input data, and set up format/structure of output
- %catsum: To generate summary statistics for categorical variables
- %contsum: To generate summary statistics for continuous variables
- %aesum: To generate summary statistics for Adverse Events
- %print: To create the final output table or listing

3. Low-level utility macros:

These SAS macros are called by the high-level utility macros. They are auxiliary macros that have smaller scopes and perform more focused tasks.

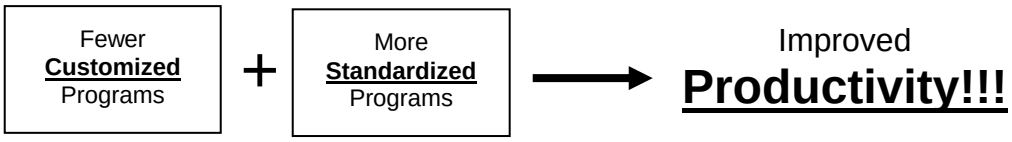
Examples of low-level utility macros are:

- %printset To establish titles and footnotes
- %pageset To set up “page x of y” in the TLG header and perform pagination
- %nobs To retrieve the number of observations in a data set
- %wordcnt To count words in a line and split the line if needed
- %chkparm To check for the validity of parameters

Note: Not all SMS’s have end-user macros. Some SMS’s require the application programmers to assemble a set of high-level utility macros to generate the desired TLGs.

BENEFITS OF SMS

1. Improved productivity:



This is the most important and prominent benefit of SMS. Because SMS minimizes the need to reinvent the wheel and greatly reduces the number of customized TLG programs, it in turn increases the productivity at Biostatistics and Statistical Programming departments in terms of speed, accuracy, consistency, re-usability, manageability, and of course, quality. Generally speaking, it can take an experienced statistical programmer 1-3 days to create his/her customized program and 1-3 days to validate it; with SMS, he/she may only need 1-3 hours to complete the tasks. With SMS in place, instead of writing and validating tens or hundreds lines of SAS code, creating a TLG can be just one standard macro call away.

2. Savings from centralized effort:

SMS is developed and validated by a centralized effort with pooled resources within Biostatistics and Statistical Programming departments. The economy of scale results in significant savings of overall resources involved in development, validation, and management of TLG programs.

3. Company-wide standardization:

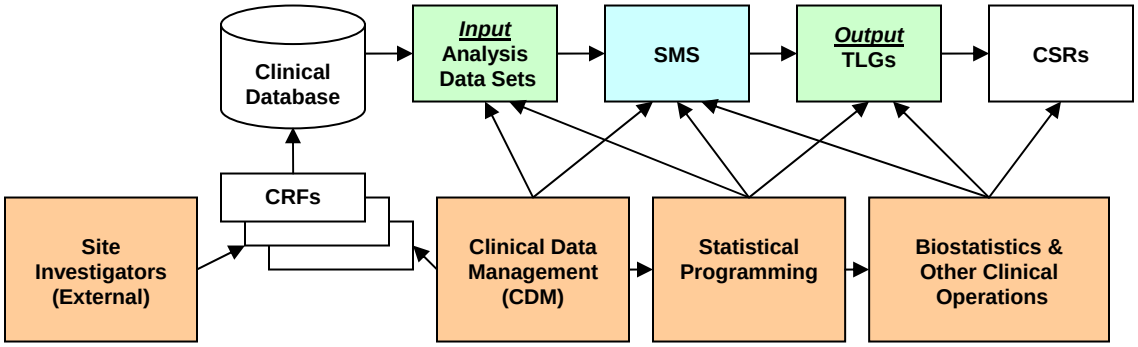


Figure 2. SMS Stakeholders and their Roles with SMS

Clinical trial data are complex and there are many players involved in collecting, processing, reporting, and reviewing them: site investigators, clinical data managers, statistical programmers, biostatisticians, drug safety, medical-writing, and other clinical operations personnel (See Figure 2).

To make SMS implementation possible, we need standardizations in both input and output environments. The major components of the I/O environments are:

1. Standardized Case Report Forms (CRFs) with standardized modules and questions to standardize how we collect and clean clinical data.
2. Standardized clinical databases with standardized table definitions and structures to standardize how we store clinical data.
3. Standardized analysis data sets with standardized data set names and structures, variable names, definitions, types, labels, lengths, formats, algorithms for derived variables, etc., to standardize how we organize clinical data.
4. Standardized TLGs with standardized layouts, titles/footnotes, formats, value precisions, etc., to standardize how we report and review clinical data.

If implemented effectively, SMS can serve as a catalyst in a company-wide standardization process to streamline operations and promote collaboration among multiple departments.

DEVELOPING SMS

The following are recommendations for developing a SMS:

1. Assemble a SMS team with skilled SAS programmers whose primary responsibilities are to support SMS development and implementation. Understand that SMS is a complex system and it requires full-time attention. Also keep in mind that creating a TLG customized program is very different from developing a standard macro and a good application programmer may not be a good SMS developer.

2. Establish coding standards (program header, programming style, structure, naming convention for macro variables and intermediate data sets, comments, documentation, etc.) to ensure that all standard macros look, structure, and function similarly and consistently. This is critical, since SMS maintenance is often performed by multiple developers.

3. Think globally covering all TLGs, yet work locally to build one end-user macro at a time, starting out with the most common and standardized TLGs such as Adverse Event (AE) and Demographics tables.

4. Follow Structured Programming principles in designing the infrastructure of SMS early in the process. Understand that SMS can become very complex thus modularization is critical. Keep in mind that maintaining (repairing) a complex system is much more expensive, so take the time and effort to design SMS right from the start. Use KISS (Keep It Simple & Structured) approach in design and coding. Minimize redundant code in SMS.

5. Perform parameter validation, error-handling, and data checking all along the way. Display meaningful warning/error messages so users can easily identify the exact causes and know what actions to take. Format these warning/error messages differently so users can differentiate them from those issued by SAS.

6. Respect users' working environment. Make sure macro variables and intermediate data sets used in SMS do not collide with those defined by users. Use %GLOBAL and %LOCAL statements strategically to clearly define the scopes of macro variables. Save users' SAS options before SMS and restore them after SMS.

7. Guarantee backward-compatibility by ensuring that the new version of SMS produces the same TLGs as those by previous versions.

8. Test SMS thoroughly and rigorously with sufficient number of test cases. Assign another developer as the reviewer of test plans and test results. Use systematic testing by using a test value database.

9. Improve SMS continuously in coding, documentation, and overall efficiency.

10. Ensure file security (read, write, execute access) in SMS' developmental and operational environment.

11. Incorporate new technologies, features, and upgrades from SAS or other internal/external sources.

12. Apply Software Development Life Cycle (SDLC) methodology faithfully and stringently. Document all SDLC processes thoroughly and advance the projects through SDLC methodically.

Software Development Life Cycle (SDLC) Methodology:

SDLC has been known and used by most software developers for some time; however, it may be a novelty for the statistical programming community.

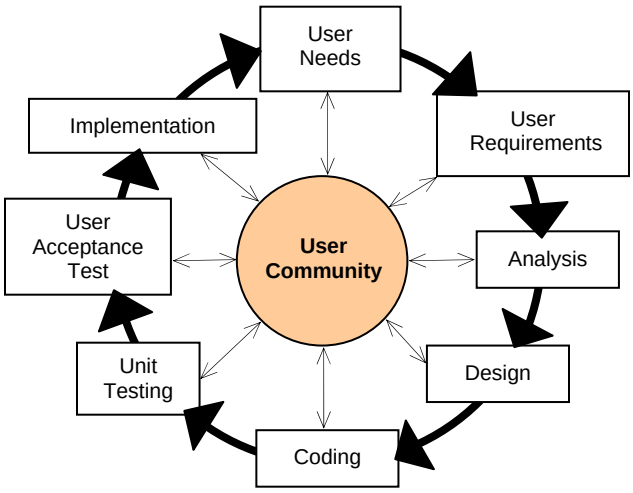


Figure 3. Software Development Life Cycle (SDLC)

SDLC begins with a need expressed by a future user of the system. If the project is initiated, SDLC then advances it through the developmental processes, one at a time, in the cycle formally and methodically. SDLC ultimately ends with a satisfied need for the user. Although the arrows in Figure 3 are unidirectional, users/developers are encouraged to return to previous processes for further clarification and additional communication. It is important to note that all processes are centered around the User Community (statistical programmers, biostatisticians, clinicians, drug safety, and medical writing personnel) and all parties have two-way communication in all processes. It is also important to be reminded that the Food and Drug Administration has mandated that SDLC be used to ensure the quality of software used to report results of clinical trials.[2]

The typical SDLC processes, deliverables, and major players are:

Process	***Deliverables/Documents***	***Major Players***
User Needs	New macro request, Enhancement request	User
User Requirement Collection	User requirements	User, Developer
Project Initiation	Need analysis, Cost-benefit analysis, Feasibility study	User, Developer
Analysis	Functional specifications, Prototyping	Developer
Design	Programming specifications	Developer
Coding	SAS macros, Enhancements	Developer
Unit Testing (UT)	Unit test plan	Tester, UT Reviewer
User Acceptance Test (UAT)	UAT plan	User, UAT Reviewer
Implementation	User's guides, eHandbook, Upgrade news, Training/education, Change control	Developer

SMS Developing & Operating Libraries:

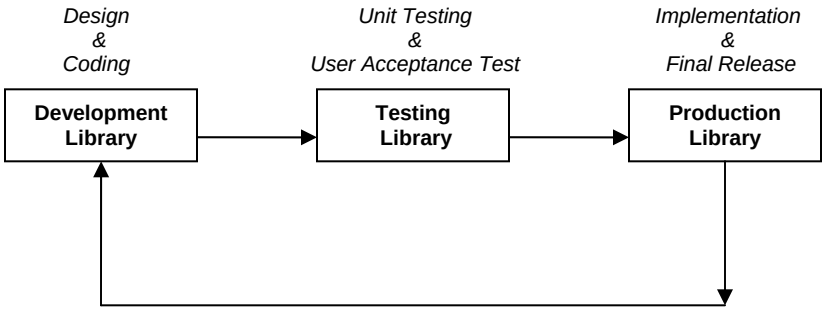


Figure 4. SMS Developing & Operating Libraries

It is also recommended to set up a physical structure (See Figure 4) where SMS is moved to the Development Library for *Design/Coding*, and then moved to the Testing Library for *UT/UAT*. And finally it is moved to the Production Library for *Implementation.* This setup also facilitates the process of version control.

Guided by these formal SDLC processes and disciplines, we are confident that SMS can become a much better software system.

MANAGING SMS

The following are recommendations for managing a SMS:

1. Secure upper management's support and commitment.

2. Provide proper leadership to the SMS team. An ideal team leader should be an experienced user and developer. He/she should believe in SMS and its benefits and must be a good project manager, listener, consensus builder, innovator, and marketer.

3. Establish a steering committee with representatives from all stakeholders to facilitate the communication and decision-making process. Use this committee to promote standardizations across multiple departments.

4. Utilize the centralized effort. Make sure that project (therapeutic area) level macros are limited and seek ways to incorporate them into SMS.

5. Build up and maintain user confidence. Attend to users' problems immediately and seriously.

6. Formalize the user communication channel; for example, users can submit enhancement requests and keep track of their progress on-line.

7. Conduct user satisfaction surveys to seek feedback so as to ensure the SMS team and SMS both satisfy the users' needs.

8. Embrace the customer-oriented attitude of "We give what our customers want!" instead of the product-oriented attitude of "We give our customers what we think they want!"

9. Establish 24/7 technical support, for SMS is often mission-critical.

10. Conduct regular training/education sessions, such as new user training, new macro/feature presentations, and consultations.

11. Create self-learning tools on the company intranet with items, such as SMS eHandbook, upgrade news, and "How to do…"/"Good to know…" articles. Provide users with ample sample programs imbedded with self-generated test data so the users can copy them into their own working area, run the sample programs, and learn about SMS at their own pace.

12. Organize a SMS User Group (SMSUG).

EVALUATING SMS

CRITICAL SUCCESS FACTORS OF SMS

The following Critical Success Factors (CSFs) are presented by using Success and Doomed statements expressed from the perspectives of SMS customers:

1. Management support:

This CSF should be ranked as the most important CSF, for a SMS is a long-term strategic investment that requires significant resources, commitment, and priorities.

- Success statement: *"Our VP strongly supports the development of SMS and treats it as a long-term investment…"*
- Success statement: *"If I need to deviate from SMS, I will need our VP's approval…"*
- Success statement: *"If I cannot use SMS and decide to write my own programs, I must provide my managers with legitimate reasons…"*
- Doomed statement: "*My manager said I could write my own TLG programs anytime I wanted to…"*

2. **Error-free confidence and reliability:**

- Success statement: *"Our SMS has gone through rigorous unit testing with 200+ test cases…"*
- Success statement: *"I have full confidence in our SMS because it is fully validated…"*
- Doomed statement: *"Our SMS from time to time creates some unknown errors..."*
- Doomed statement: *"I don't trust our SMS…"*

3. **Robustness and flexibility:**

- Success statement: *"Our SMS is so flexible; it can meet all of my needs…"*
- Success statement: *"Our SMS can be used for different therapeutic areas…"*
- Doomed statement: *"Our SMS does not let me change the table number…"*
- Doomed statement: *"Our SMS does not work for this kind of input data..."*
- Doomed Statement*: "Our SMS only does Safety tables..."*

4. **Quick responses:**

- Success statement: *"Our SMS team always delivers new features in a timely fashion…"*
- Success statement: *"Our SMS team fixed my problems using SMS so I could meet my deadlines…"*
- Doomed statement: *"Our SMS team has not responded fast enough so I had to write my own program…"*

5. **Continuous improvement:**

- Success statement: *"Our SMS team constantly listens to our needs and tries hard to make our lives easier…"*
- Success statement: *"Our SMS team always incorporates new technologies/features into our SMS…"*
- Doomed statement: *"Our SMS is too outdated…"*
- Doomed statement: *"Our SMS does not work with the new version of SAS…"*

6. **Continuous education/training:**

- Success statement: "*Our SMS team makes sure I know how to use SMS in the most effective way…"*
- Success statement: *"Our SMS eHandbook is great; it greatly reduced my learning curve…"*
- Doomed statement: *"I spent months learning SMS and there is no documentation and training available…"*

7. **Continuous promotion:**

- Success statement: *"Our SMS team makes sure I know about the new macros/features…"*
- Doomed statement: *"What new macros/features?!"*

Once again, the SMS team and the upper management should be aware of these CSFs and evaluate them on a regular basis to ensure the success of SMS.

CHALLENGES

COMPLEXITY STEMMED FROM HAVING MANY STAKEHOLDERS

SMS can grow into a very complex system, for there are many stakeholders:

- All input data (empty data sets, existence of data sets/variables, missing values, zero divisions, large sizes, etc.)
- All customers (statistical programmers, biostatisticians, clinicians, clinical data managers, drug safety, medical writing, and other clinical operations personnel, etc.)
- All statistical methods (PROC FREQ, PROC NPAR1WAY, PROC GLM, PROC LIFETEST, PROC LIFEREG, etc.)

- All tables (missing columns, missing cells, empty columns/cells, etc.)

- All output (styles, margins, formats, titles/footnotes, “No Report Generated”, etc.)

Good planning, prioritization, modularization, project management, documentation, and communication can make it a manageable environment.

CONCERNS ABOUT JOB SECURITY

Like many new technologies, there will be fears in introducing a new system: “Now that we have SMS, I will no longer be needed or have a job…” We must calm these fears in the statistical programming community by assuring everyone that there will always be plenty of work. In fact, SMS implementation will allow us to focus our attention on the following areas:

- Creation of analysis data sets
- Validation of application programs
- Project planning/management
- Participation of SMS development and testing
- Development of other selective customized TLG programs: There is an 80-20 rule: 80% of TLGs can be generated by SMS, while 20% of TLGs still need to be created with customized programs.

OPPORTUNITIES

EMERGENT TREND OF COMPANY-WIDE STANDARDIZATION

More and more companies are recognizing the benefits of company-wide standardizations in collecting, processing, and reporting clinical data, and they are making significant strides in the following areas:

- Standard CRF modules
- Standard Analysis Data Set (ADS) specifications
- Standard Statistical Analysis Plan (SAP) templates
- Standard CSR templates
- SOPs and other Operational Guides

EMERGENT TREND OF INDUSTRY-WIDE STANDARDIZATION

More and more concerted efforts in the industry have been made to standardize the way we collect, clean, report, and distribute clinical trial data, such as the effort led by Clinical Data Interchange Standards Consortium (CDISC).

Both trends will help create a smoother developmental and operational environment for SMS. Our ultimate goal of “A new TLG is just another standard macro call away!” is closer to becoming a reality.

CONCLUSION

Because clinical trial data are complex and involve many people, building an in-house SMS to process and report these data is not an easy job. Many companies have long since begun their journeys in building standard macro systems, while others are just starting. Although their paths may be very different, one thing is the same: They all have made the correct decision and are on the right track to reaping many benefits brought about by SMS. I sincerely hope that this paper provides sufficient points for consideration so as to make these journeys a bit smoother. Finally, I pose a question which I often have in my many years of personal and professional experience with SMS: “Are we ready for a comprehensive, industry-wide, off-the-shelf, SMS package?” This might be an interesting and important question for our conference sponsor, the SAS Institute, to ponder.

ACKNOWLEDGEMENTS

I would like to thank Gisela Schwab, Saling Huang, Jason Pineda, Kristine Flores, Lee Smith, Amy Schneider, Mahesh Patel, Paulynn Dorotheo-Hein, Doug Brown, Adam Sharp, and James Liu for reviewing this document and sharing their experiences and perspectives.

REFERENCES

1. Howard, Neil. 2003. *"Beyond Debugging: Program Validation." Proceedings of the 28th Annual SAS Users Group International Conference.*

2. Mitchell, Brian. 2005. *"Using Extreme Programming Processes in a SAS Environment." Proceedings of the 30th Annual SAS Users Group International Conference.*

3. Wilson, Steven A. 2002. *"Getting Started with SAS/AF Software." Proceedings of the 27th Annual SAS Users Group International Conference.*

4. Ward, Vivienne, and Haske, Carl R. 1996. *"A Statistical Analysis Macro Library in SAS Software." Proceedings of the 21st Annual SAS Users Group International Conference.*

5. Zuo, Jun, and Haske, Carl R. 2000. *"Creating Clinical Trial Summary Tables Containing P-Values: A Practical Approach Using Standard SAS Macros." Proceedings of the 25th Annual SAS Users Group International Conference.*

6. Zhang, Julia, and Chen, David. 2004. *"Data Management in Analyzing Clinical Trial Data – Metadata Application." Proceedings of the 29th Annual SAS Users Group International Conference.*

AUTHOR

Albert Mo is a Senior Manager with Abgenix, Inc., Fremont, CA. Albert has been developing applications and systems in SAS for nearly 18 years working at or consulting for pharmaceutical/biotechnology companies. He has extensive experience in Statistical Programming in Biostatistics and Clinical Data Management.

CONTACT INFORMATION

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

Albert Mo
Abgenix, Inc.
6701 Kaiser Drive, MS 42
Fremont, CA 94555
(510) 284-6979
albert.mo@abgenix.com or albertmo@verizon.net