USING AUTOCALL MACROS AS A COMMON CODEBASE OF DATA CHECKS FOR CLINICAL TRIALS: EXAMPLES DRAWN FROM THE PHARMACEUTICAL INDUSTRY

Michael E. Tomb
Informatica ECS

Abstract

Referencing libraries of re-usable code by way of the SAS® Macro Autocall facility is an effective method of minimizing program editing and maintenance. This paper describes a system that FISONS Pharmaceuticals has devised for a common codebase of Clinical Trial data checks and listings. All variable names, arrays, formats, dataset names and pivotal variable values are referenced symbolically. The common code is applied using a two-step process. The user may override or accept default values, names and variable typing for all symbolic references. The user then selects from a list of code modules to run using current data. This system assures that validated code is always used even though data may originate from diverse entry systems.

Introduction

The analysts and programmers who build datasets for the analysis of Clinical Trials are often responsible for ensuring data quality by running a series of data checking programs. At times this has meant that programmers have simply copied existing programs from similar protocols and extensively edited the code for the study at hand. The SAS Macro Autocall facility offers an alternative way of sharing a codebase of such data checks that frees the programmer from recoding existing algorithms. In this manner, battle-tested code can be re-used across projects and satisfy the validation requirements of good clinical practice.

What Is An Autocall Library?

All of the SAS Code referenced in this paper relies on the use of the SAS Macro Autocall Facility. The Autocall Facility allows the creation of libraries of macros that can be referenced directly by macro name without having to include specific filenames in a SAS program. A macro source code file must be named the same as the macro it contains for the automatic lookup to succeed. The library itself is often a globally accessible directory or folder. The SASAUTOS option provides for the specification of several directories to be search sequentially for macro programs. Listing local directories of test and custom versions of macros first in the SASAUTOS option ensures that these will be searched prior to the production version libraries. At FISONS, production versions of macros are stored in a VMS library such as FISONSSCOMMON:[SAS_MACROS], while template programs illustrating use of the system might be stored in a directory similar to FISONSSCOMMON:[SAS_EDITS]

Clinical Trial Data Entry Systems

A diverse set of data entry systems may be used to create the data to which clinical programmers must apply data checks. These are illustrated in Figure 1 on the next page. Systems may be tailored to a specific therapeutic area or clinical phase. Such systems could include:

- Direct SAS Entry Systems
Programming tools as SAS/FSP®, SAS/AF® and SCL may be used to directly create SAS datasets.

- RDMS Systems
Oracle® based systems such as Clintrial® offer secure data entry and employ relational methods. Some of these utilize a data dictionary. Vendors may supply software that converts tables to SAS datasets or products such as SAS/ACCESS® can be used to bring the data into the SAS system.

- Flat File Entry Systems
‘Legacy Data’ from existing studies that require re-analysis may be in the form of simple flat files. It may be convenient to enter Clinical Pharmacology studies that contain small numbers of patients into flat files. Such studies might involve unique data structures that have
Figure 1. The Sources of Clinical Trial Data

- **Data Checks and Listings**

- **Oracle® or Other RDMS Based Data Entry Systems**
  - Custom Conversion to SAS®
  - SAS/ACCESS®

- **Direct Data Entry Systems**
  - Fax-Based OCR Data Entry Systems
  - On-site Remote Data Entry

- **Flat File Entry**
  - Clinical Pharmacology
  - Data from Other Sources
  - Legacy Studies

- **Direct Entry into SAS®**
  - SAS/FSP®
  - SAS/AF®

limited reusability. Patient laboratory data electronically acquired from the Clinical Laboratories that perform the tests may be distributed in the form of sequential ASCII files as well.

- **Direct Data Entry Systems**

Existing systems allow on-site clinicians to fax each page of the Case Report Form (CRF) into a waiting computer system. OCR technology is then used to populate the database directly from the image of the CRF and bypass the need to keypunch data. In addition, many pharmaceutical companies are currently exploring the use of Remote Data Entry Systems. A personal computer may be installed in the investigator's office and effectively replace the paper CRF with programs that use electronic forms. The data, in the form of flat files or PC style databases, may be sent to the Clinical Data Management department by modem or diskette.

More than one system may be in place within a single data management department; especially during the transition period after the department acquires new entry software. Even during the course of a single study, multiple entry systems may be used.

The result of this complex variety of systems can present a dilemma to the clinical programmer. Enforcing naming conventions and variable typing for all SAS variables eventually derived from the diverse systems may be impractical. For example: the limits of OCR may require that simple numeric codes be used in answer to certain questions, whereas similar questions entered by a data entry personnel had previously been standardized using alphanumeric encoding schemes. A robust codebase that allows either systems' datasets to be checked must not be variable type dependent in order to address this problem.

**Data Checking Within the Processing of Clinical Trials Data**

Figure 2 illustrates the role data checking plays in processing of Clinical Trial Data. The timely generation of data checks is necessary to allow the clinical sites ample opportunity to issue corrected copies of CRF pages and other clarification documentation. A system using a
Figure 2. The Processing of Clinical Trial Data

Unblinding, Analysis and Reporting

Data Checks
- On Going
- End of Study

Data Entry
- Entry Checks
- Verification

Clarifications
- Missing Data
- Corrected Copies

Case Report Forms
- Clinician Recorded
- Patient Diaries
- Laboratory Data

A reusable codebase can accelerate this processing by removing days of custom coding, testing and program verification. Note that within this paper, I am defining data checks as events that take place post entry. Data Entry Errors caught by the data verification or auditing process, and individual data field verification is not covered within the scope of this system. Often such checks are handled by programming native to the DBMS system used for entry.

Examples Of Standard Data Checks And Listings

The standardization of the data checks at FISONS is the responsibility of a team with representation from Data Management, Biostatistics, Clinical Research and Regulatory Affairs. Based on previous experience and professional expertise, all personnel involved in the gathering, entering and monitoring of Clinical data were given the opportunity to suggest rules for checking the standardized CRF pages. Working from a copy of this itemized list, a team of clinical programmers produced a series of macros specific to each section of a standard CRF. The source of many of the macros was existing code that was re-written for clarity and so as to be self-contained.

The following table is a sampling of standard data checks and listings that a clinical programmer might be requested to produce. This is a small abstract from an existing list of hundreds of such checks, many of which are specific to a therapeutic area.

<table>
<thead>
<tr>
<th>Case Report Form Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demography and Vital Signs</td>
<td>List Duplicate Combinations of Patient Initials and Dates of Birth</td>
</tr>
<tr>
<td></td>
<td>List Vital Signs Measurements Outside of Suggested Ranges</td>
</tr>
<tr>
<td></td>
<td>Cross Check Age related Eligibility Questions with Date of Birth</td>
</tr>
<tr>
<td>Adverse Events</td>
<td>Cross-check Sex and Age in Laboratory Data against Demography page</td>
</tr>
<tr>
<td>Disposition and Withdrawal</td>
<td>For Adverse Events listed as related to Study Medication, check that the Date of Onset is after the first dose of Study Medication</td>
</tr>
<tr>
<td></td>
<td>Check that an Adverse Event Page Exists for All Patients</td>
</tr>
<tr>
<td></td>
<td>List All Adverse Events of Patients who withdrew due to an Adverse Event</td>
</tr>
<tr>
<td></td>
<td>List All Adverse Events Sorted by Preferred Term Codes</td>
</tr>
<tr>
<td>Withdrawal</td>
<td>List Patients giving a reason for withdrawal without listing a withdrawal date</td>
</tr>
<tr>
<td></td>
<td>List Randomized Patients who failed an entry criteria</td>
</tr>
<tr>
<td></td>
<td>List Patients who withdrew for ‘other’ reasons uncommented</td>
</tr>
<tr>
<td></td>
<td>List a Clinic Wide Summary of Completed, Withdrawn and Randomized Patients</td>
</tr>
</tbody>
</table>
Included with the data checks are summary lists used by Clinical monitors, Biostatisticians and Data Analysts for a variety of descriptive and diagnostics purposes.

**Codebase Definition**

The term 'codebase' is used to refer to a library of macros that contains no direct references to permanent variables or datasets. This is, in practice database of routines with automatic lookup provided by the AUTOCALL facility. The code in this database is not necessarily complex. In fact, simple code is almost always preferred. The codebase is a good place to store any SAS code that needs to be re-used for different combinations of datasets and variables.

The value of a codebase can be understood when dealing with datasets of similar structures, but with differing naming conventions or even internal representations. In the codebase for data checks, two macros are associated with each set of data checks. The first macro is used to map all symbolic references to actual names of variables, formats, and dataset names. The second macro allows selection of groups of data checks to run against the datasets that represents pages of CRF data. *Figure 3* shows the flow of a typical data check program.

**Guidelines for Writing Data Check Codebase Modules.**

Here are a few of the coding practices used for converting open SAS code to the style of the macro driven codebase.

- Write well commented, indented code. The code is intended for use by more than one programmer.
- Supply all SORTS within the code; Don’t assume a sorting order.
- Do not share logic across data checks. Remember that each data check or listing can be run independently.
- Always use the option MPRINT option to mirror the macro-written code into the log. The option MLOGIC is also invaluable while debugging. Accurate Macro debugging information is one of SAS’s weaknesses.

---

**Figure 3 Use of the Data Check Codebase**

**SAS AUTOCALL MACRO LIBRARY**

- **CLININFO.SAS**
  - Create Standard LIBNAMEs and TITLEs
- **DP_SETUP.SAS**
  - Resolve references for Disposition Data Checks
- **DP_DATCK.SAS**
  - Run Selected Data Checks for Disposition/Withdrawals
- **Other Data Check Modules:**
  - Adverse Events
  - Diary Efficacy, Etc.

---

Sample Data Check SAS Program

```sas
%CLININFO( PN=1995, SN=3995 )
%DP_SETUP( PATNO = PATNO2)
%DP_DATCK( RUNALL = TRUE )
```

Reports, Listings and Other Output
Pharmaceutical Industry

- Use WHERE statements to define subsets of observation when ever possible. Use in combination with PROC PRINT or PROC REPORT for simple, effective code.

- Avoid direct references to variable names, dimension of arrays, formats and datasets and even, where possible, variable typing. Allow the user to override all the default ranges tested. For example, instead of IF AGE > 21 Use IF &AGE > &AGELIMIT.

- Use one format statement for every variable explicitly formatted. This way, users may override the format specification and even set variables to be unformatted.

- Use the DO OVER statement or DIM function to handle array dimensions. This avoids explicit reference to the number of array elements. Allow lists of the actual items in array to be passed in as macro parameters.

- Provide a Clinical Programmer’s cross-reference to the Codebase. Include lists and descriptions of all code modules and each macro’s parameters, sorted alphabetically and by a specific data check. For the system shown, a utility SAS program is used that reads SAS codebase modules and creates the cross references.

- Validate the codebase by constructing test databases. Keep the test data well organized and secure. The test sets will be re-used when codebase modules are updated.

Standard Libraries: Macro CLININFO

This is the first of three sample macros for which source code is supplied at the end of this paper. This macro sets up the use of standard dataset and format libraries related to the access of Clinical Trial data. Standard directory names are composed using the Project Number (PN) and Study Number (SN) for a protocol. In addition, a default title for the SAS job is created. As you can see from a sample SAS log showing the code generated from an invocation of this macro....

\%CLININFO( PN=1295,SN=3995 ,SDRUG=CureAll )

![Figure 4 SAS LOG AFTER INVOCATION OF CLININFO MACRO](image)

...the macro is not limited to use within the data checking codebase. Virtually any job accessing data for a protocol can gain access to study specific and global LIBNAMES with the use of this one line. If required the default directory names can be overridden with MACRO calling parameters. Source code for this macro is given at the end of this document in the source listing section. This macro also creates a few global macro variables used to assign default dataset names in code to come.

Macro DP_SETUP

This macro is essentially a list of all symbolic references used in the data check section of the code with default values supplied for each. The programmer uses this macro to customize the data checks for the dataset names and variables to be passed through the system. The macros simply creates a global macro variable that does the mapping from actual parameter to symbolic name.

Macro DP_DATCK

This macro has the actual code used for each disposition data check available within the codebase. A simple example is supplied in the listing section of this paper. Here two data checks are coded. Notice that even pivotal variable values are treated symbolically. The line: WHERE &COMPLETE = &YES could resolve to either a numeric or character comparison, depending on the
nature of the data. In actual practice, this macro provides dozens of distinct data checks. Automatic generation of summary listings of all patients' reasons for withdrawal are generated as well. Regardless of the entry system system used and naming practices of the output files produced, the macro gives the programmer standard SAS programs that can easily be applied from study to study.

**Enhancements And Other Applications Of AUTOCALL Libraries**

The techniques of building libraries of re-usable code is not limited within Clinical Trial programming to data checking. The same methodology described can be applied to libraries of programs for:
- Data listings within statistical appendices
- Summary tabulations used in the body of reports
- Standard Statistical Analyses
- Any repetitive use of similar SAS programs

Products such as SAS/ACCESS® can extend the use of data checking program libraries to views of existing tables in RDMS systems such as Oracle® without extracting to permanent SAS datasets.

**Acknowledgments**

I would like to thank Robert Agostinelli, Ken Bauer, Alicia Brocht, and Jeffrey Hart of FISONS Pharmaceuticals, all of whom were involved in the development of this system.

**Author Address**

Michael E. Tomb  
Informatica ECS  
183 Midtown Plaza # 128  
Rochester, NY 14604  
Tel:(716) 256-0401  
Email: Compuserve Id 70054,2465

**Source Listings**

**MACRO CLININFO.SAS**

*---------------------------------------------------------------*
MACRO CLININFO  Setup Clinical Trial Job Files and Titles  
For Edit Checks or General Work  
PROGRAMMER: Michael E. Tomb  DATE: 4/14/94  
MODIFIED: 7/95 Compressed Demonstration Version for NESUG  
*----------------------------------------------------------------*  
Example  
INPUT:  PN - Project Number  
SN - Study Number  
SDRUG - Drug Title  
STITLE - Study Title  
LIBDIR Optional,  
Specify Database Directory [TOMB_M...  
FMTLIB Optional,  
Specify Format Libraries [TOMB_M...  
EX: %CLININFO( PN = 1995 , SN = 3945 , SDRUG = CureAll )  
or %CLININFO( PN = 1995 , SN = 39452075 , STITLE = Another Great Drug Redux ,  
LIBDIR = OTHER$DRIVE:[S3945.REDUX.SASDATA]  
FMTLIB = OTHER$DRIVE:[S3945.REDUX.FORMATS] )  
*---------------------------------------------------------------*;  
%MACRO CLININFO( PN = NONE , SN =NOSTUDY ,  
SDRUG = NODRUG , STITLE = NONE ,  
LIBDIR = DEFAULT , FMTLIB = DEFAULT );  
*---------------------------------------------------------------*  
567
COPY LOCAL VARIABLES TO GLOBAL ONES FOR USE IN OTHER MACROS
SETUP Local Flags used in deciding which Options are Present

*/---------------------------------------------------------------*
%GLOBAL STUDYNO SPRE PROJNO; %LET STUDYNO =&SN; %LET PROJNO =&PN;
%LOCAL DO_LIB DO_FMT DO_TIT DO_DRG DO_PRE ;
%LET DO_LIB = %EVAL( $QUOTE(&LIBDIR) = DEFAULT ) ;
%LET DO_FMT = %EVAL( $QUOTE(&FMTLIB) = DEFAULT ) ;
%LET DO_TIT = %EVAL( $QUOTE(&STITLE) = NONE ) ;
%LET DO_DRG = %EVAL( NOT( $QUOTE(&SDRUG) = NODRUG ) ) ;
%LET DO_PRE = %EVAL( NOT( $QUOTE(&SN) = NOSTUDY ) ) ;
*/---------------------------------------------------------------*

CREATE FORMAT LIBRARY LIBREF AND FMTSEARCH STATEMENTS

*/---------------------------------------------------------------*
%IF &DO_FMT THEN DO ;
OPTIONS FMTSEARCH=( STUDYFOR COMMONAE ) ;
LIBNAME COMMONAE 'FISON$COMMON: [AE.60S]' ;
LIBNAME STUDYFOR "STUDY$DISK: [S&SN..FORMATS]" ;
%END;
%ELSE %DO ;
OPTIONS FMTSEARCH=( STUDYFOR COMMONAE ) ;
LIBNAME COMMONAE 'FISON$COMMON: [AE.60S]' ;
LIBNAME STUDYFOR "$FMTLIB" ;
%END;
*/---------------------------------------------------------------*

CREATE BASE LIBNAME ( LOCATION OF DATASETS )
CREATE PRIMARY TITLE

*/---------------------------------------------------------------*
%IF &DO_LIB THEN DO ;
LIBNAME BASE "STUDY$DISK: [S&SN..SASDATA]" ;
%END;
%ELSE %DO ;
LIBNAME BASE "$LIBDIR" ;
%END;
%IF &DO_TIT THEN DO ;
%IF &DO_DRG THEN DO ;
TITLE " STUDY NO. &PN-&SN -- &SDRUG " ;
%END;
%ELSE %DO ; TITLE " STUDY NO. &PN-&SN " ;%END;
%END;
%ELSE %DO ; TITLE "$STITLE" ;%END;
*/---------------------------------------------------------------*

CREATE DATASET PREFIX VARIABLES USED IN NAMING DEFAULT DATASETS
SUCH AS THE S1240 IN S1240DLB, USE THE FOUR DIGITS FROM STUDY #
(IF PROVIDED) OTHERWISE USE 'DAT' LIBNAME ( LOCATION OF DATASETS )

*/---------------------------------------------------------------*
%IF &DO_PRE THEN DO ;
%LET SPRE = S%SUBSTR( &SN , %LENGTH( &SN ) - 3 ) ;
%END;
%ELSE %DO ; %LET SPRE = DAT ;%END;
%MEND;

Macro DP_SETUP.SAS
MACRO DP_SETUP SAMPLE Setup Disposition information
For Data Checks or General Work
PROGRAMMER: Michael E. Tomb and Jeffrey B. Hart
DATE: 8/10/94
MODIFIED: 7/95 Sample (Very Compressed) Version for NESUG

INPUT PARAMETERS:

- **MPTNO** - Patient Number
- **MCLINIC** - Clinic
- **MBASE** - Libref for Datasets
- **MLASTDOS** - Date of Last Dose of Study Drug
- **MDATDP** - Disposition Dataset Name
- **MDATDLB** - Randomization Dataset Name
- **MCOMPLET** - Did Patient Complete Study?

EXAMPLE

- **PTNO**
- **CLINIC**
- **BASE**
- **LASTDOSE**
- **DATDP**
- **DATDLB**
- **COMPLETE**

NOTE: The following are the values used within the edit checks.
If your code list is different then pass in the appropriate values

- **MYES** Default is 'Y'.

FORMATS:

- **MCFMT** - Format for CLINIC Variable

EX: %DP_SETUP( MDATDP = S39952D );

%MACRO_DP_SETUP( MPTNO = PTO, 
MBASE = BASE, MDATDP = &SPRE.WD, 
MDATDLB = &SPRE.DLB, MCLINIC = CLINIC, 
MCOMPLET = COMPLETE, MLASTDOS = LASTDOSE, 
MYES = 'Y', MCFMT = $CLINIC. );

COPY LOCAL VARIABLES TO GLOBAL ONES FOR USE IN OTHER MACROS

%GLOBAL PTNO BASE DATDP DATDLB CLINIC YES CFMT COMPLETE LASTDOSE ;

%LET PTNO = &MPTNO;
%LET CLINIC = &MCLINIC;
%LET BASE = &MBASE;
%LET DATDP = &MDATDP;
%LET YES = &MYES;
%LET COMPLETE = &MCOMPLET;
%LET LASTDOSE = &MLASTDOS;
%LET DATDLB = &MDATDLB;
%LET CFMT = &MCFMT;
%MEND;

Macro DP_DATCK.SAS
MACRO DP_DATCK SAMPLE Disposition DATA Checks
PROGRAMMERS: Michael E. Tomb DATE: 8/2/94
MODIFIED: 7/95 Very Truncated Version for NESUG

Set Flags for Desired Action. Overall Flags Overide Individual Ones.

Upper Case all Passed Arguments

Flags set by Passing a value of T (True)

Overall setup of Disposition dataset for use throughout data checks

Sample Data Check # 1

If patient checked "yes" to completing study, Date of Last Study Drug dose Should be empty. If not, it will be reported.

Sample Data Check # 2

All patients must have a disposition / withdrawal page. DATDLB is a SAS data set containing a list of every study patient.
%IF &DO_ALL OR &DO_DATCK OR &DO_ED2 %THEN %DO;
TITLE2 "Disposition Form Data Check # 2";
%PUT NOTE: Data Check #2 Will Be Run;

PROC SORT DATA=&BASE..&DATDLB OUT=DLB;
   BY &CLINIC &PTNO;
DATA TEMP;
   MERGE DISP (IN=A) DLB (IN=B); BY &CLINIC &PTNO;
   IF B ^= A;
PROC PRINT; BY &CLINIC;
   VAR &PTNO;
   FORMAT &CLINIC &CFMT.;
TITLE3 'Patients Missing the Disposition Page';
RUN;
%END;
%MEND;