SCOGEN: A SYSTEM FOR GENERATING ROTE SAS* CODE

Mark Branagh
Infant Health and Development Program
Stanford University School of Medicine

ABSTRACT

A SAS based data management system in a multisite clinical trial environment has benefited from the development of a SAS code generator implemented in the dBase III** programming language. A code generator is a program that generates other programs or portions of other programs. It can play an integral role in the automation of a production system where a series of similar programs is needed. System input is extracted from existing ASCII word-processing files and a menu driven data entry system. The output of the system consists of a series of ASCII files that include SAS dataset INPUT specifications, variable labels, macro variable assignments, and repetitive SAS macro invocations. Benefits of the system include greatly reduced programmer time spent keypunching rote SAS code, increased data and program integrity, and more easily maintained programs. The system has also facilitated efforts to work around a SAS compiler capacity limitation.

BACKGROUND

This paper describes a SAS code generator (SCOGEN) developed and employed by the Infant Health and Development Program (IHDP) at Stanford University. It was designed to work with an existing File Update System (FUS) to more fully automate SAS code development.

The Infant Health and Development Program (IHDP)*** is a multisite clinical trial. It is a longitudinal study, collecting data on more than 1000 low birth weight preterm infants. More than seventy different data forms will be collected and keypunched at scheduled intervals over a three year period.

The data for each form must checked for validity and consistency (cleaned), and descriptive statistics reports produced. The programs that perform the data cleaning and produce the descriptive statistics for each form contain the same DATA STEPs and PROCs, differing only in variable specific details.

Four program templates are run to perform the desired function on any form in the system. %INCLUDE statements are used to bring in the form specific sections of code. Each of the over seventy forms has six %INCLUDE files associated with it (see Figure 1).

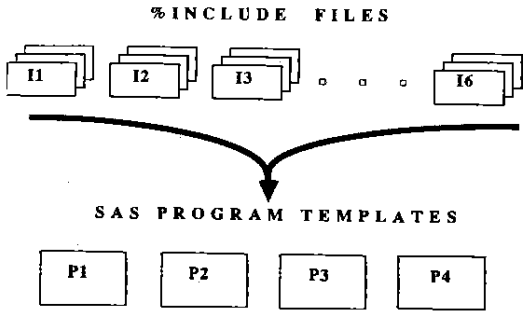

Figure 1

THE PROBLEM

A large proportion of programmer time was spent generating the rote SAS code located in the %INCLUDE files associated with each form. (Larger forms required over 6000 lines of code in the %INCLUDE files.) With an ever increasing number of forms requiring processing, it became desirable to reduce the programmer time necessary for this task.

Some of the larger forms were incurring a SAS Error 344. On the IBM 3090 under OS, this error results from a SAS internal address space overflow within a DATA STEP. Several factors contribute to this condition, particularly, heavy use of arrays, macro and DO loop invocations. (We often used an array and a DO loop to repeatedly invoke a macro with varying parameters.) To avoid Error 344 the use of these constructs had to be minimized, adding to the profusion of rote SAS code.

THE SOLUTION

Our solution to these problems was to develop a SAS code generator. (SCOGEN) It was not our goal to eliminate programmer involvement. Rather, we sought a tool that would remove the bulk of the drudgery inherent in producing the SAS code required by our system. Once the code has been generated, it is turned over to SAS programmers for fine tuning.

CHOOSING A PROGRAMMING PACKAGE

SCOGEN consists of a series of programs developed using dBase III. DBase III is a fourth generation database management system that runs on IBM PC, or equivalent, computers. In choosing a programming package in which to implement SCOGEN the following features were sought:

- Rapid program development time

- Full screen data entry tools

- Flexible string manipulation functions

- Capacity to read and create ASCII files

- Database sort and search capabilities

The system could have been implemented using a number of programming packages including interactive SAS with SAS FSP or PC SAS with FSP. Since we are limited to batch processing and, at the time, PC SAS with FSP was unavailable to us, dBase III was an excellent alternative.

SYSTEM REQUIREMENTS

A total of six %INCLUDE files are required for each of the over seventy forms. These contain the following SAS code:

1) Data step INPUT and FORMAT statements

2) Data step LABEL statements

3a) Variable range checking macro invocations

3b) Other data checking macro invocations

4) PROC FORMAT statements

5) DATA step FORMAT statements associated with the PROC FORMAT

6) %LET macro variable assignments

SYSTEM INPUT AND OUTPUT

Input for SCOGEN is derived from two sources. The first source is existing ASCII word processing files. Secretarial staff had already been producing and maintaining text files that contained information about the variables in each form, as shown in Figure 2.

Variable	Label	Line:Column	Values
IHDP	IHDP Number	1: 1-4	1000-8999
FORMVER	Form Version	1: 9	0-9
SITE	Site Name	1: 10-12	CHAR
BDATE	Infant's Birthdate	1: 13-18	11/05/84-09/07/85
TIMEBGN	Time Interview Began	1: 19-22	0600-2200
V3	Mother's Perception of Her Health Now	1: 27	1-3
V3	Mother's Perception, Her Health Overall	1: 28	1-5
V4	Mother in Hospital	1: 29	1-2

Figure 2

These files are parsed to produce an internal variables database which contains basic information about the form and its variables. The variable names, labels, position, ranges, and data type are all extracted. This database is then used to produce the following SAS code which is located in the %INCLUDE files 1) through 3a) above.

1) Data step INPUT and FORMAT statements

```
INPUT
    ihdp                    1-4
    formnum                 5-7
    formver                   8
    site $                  9-11
    @12 birdate mmddyy6.
    @20 disdate mmddyy6.
    @26 f7date mmddyy6.
    time                  32-35
    v2                      40
    v2a                   41-42
    ;

FORMAT birdate disdate f7date MMDDYY8.;
```

2) Data step Label statements

```
LABEL
IHDP     = "IHDP number"
FORMNUM  = "Form Number"
FORMVER  = "Form Version"
SITE     = "Site Name"
BIRDATE  = "Infant's Birthdate"
GA       = "Infant's Gestational Age"
DISDATE  = "Date of Discharge"
F7DATE   = "Date of 40-Week Interview"
TIME     = "Time Interview Began"
V2       = "Clinic Visits"
;
```

3a) Variable range checking macro invocations

```
%rangeck(vX=birdate,
    lowerlm="05NOV84"D,
    upperlm="07SEP85"D)
```

The second source of input to the system is direct data entry. The variables database, created by parsing the ASCII file, is an integral part of the data entry system. Besides containing information to directly produce several pieces of SAS code, it is used by other modules of the system that require variable lists or variable data type. For example, the following data entry screen is used to gather the information to produce invocations of a series of macros known as skip checks.

```
ROUTING VARIABLE: [v4          ]
ROUTING CONDITION: [EQ 2                    ]
      SKIPVAR(S): [v5 v6              ]
```

Figure 3

From this screen the following macro invocations are produced.

```
%SKIPCK(routvar= v4,
    routcond= EQ 2,
    skipvar= v5)

%SKIPCKC(routvar= v4,
    routcond= EQ 2,
    skipvar= v6)
```

Note that two different macro calls have been produced. The reason for this is that V5 is a numeric variable requiring a call to the numeric version (SKIPCK) of the macro while V6 is a character variable requiring the character version (SKIPCCK) of the skip check macro. By performing a look up of the variables V5 and V6 in the variables database, their data type is determined and the appropriate macro invocations are produced.

PROC FORMAT STATEMENTS

The variables database is also used in conjunction with data entry screens to produce PROC FORMAT statements and their associated DATA step FORMAT statements.

Not every variable will have a format associated with it. For those variables the screen is left blank. Often, two or more variables will share a common format. In this case, the screen is filled in for the first variable sharing the common format. For each subsequent variable having that format, only the 'Same As:' field is entered with the name of the first variable. The following examples illustrate the process.

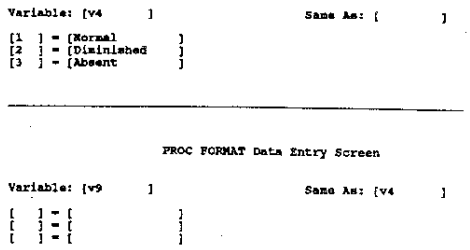

Figure 4

From the data entry screens shown above the SAS code in the %INCLUDE files 4) and 5) is produced as shown below.

```
FORMAT
   v4 v9 v4fmt.;


VALIE v4fmt   1='Normal'
              2='Diminished'
              3='Absent'
```

Figure 5

The descriptive statistics programs require lists of the variables categorized into the following groups:

Continuous - (Numeric) Example: Weight (grams)

Categorical - (Numeric) Example: Sex 1 = Male 2 = Female

Date - (Numeric) Example: Birthdate

ICD-9 - (Character) A special 5 character variable used to to code medical conditions.

These lists are passed to the SAS program via %LET statements contained in %INCLUDE file 6) from above. For example:

```
%LET con_vars= wght hght;

%LET cat_vars= v1 v3 v3a1 v3a2 v3a3
    v3a4 v3a5 v3b;

%LET datevars= bdate testdate
    mombdate;

%LET icd9vars= v16 v18 v33;
```

The information used to produce these statements is obtained from two sources. The internal variables database has already determined which variables are of type 'Date' and 'ICD-9'. To differentiate between categorical and continuous variables (both classified as 'Numeric' in the variables database) the database containing the PROC FORMAT information is used. Each categorical variable will have data associated with it for use in the PROC FORMAT, while continuous variables will not. Thus the final distinction is made and the %LET statements are produced.

CONCLUSIONS

A code generator can help free SAS programmers from the yoke of keypunching large volumes of routine SAS code. Information may be extracted from several sources including simple data entry screens and existing database or ASCII files. Once the information is extracted, complete or partial SAS programs can be produced.

The primary benefit of SCOGEN to the IHDP has been the significant reduction of programmer time required to produce the large volume of rote SAS code needed by our system. An estimated 75% reduction in overall programmer involvement has been achieved.

Other advantages of using a code generator include:

1) Greater program accuracy. A program is unlikely to misspell keywords or misplace punctuation when producing code.

2) Increased uniformity of programming style. In a situation where more than one programmer will be working with the code, there are advantages to having a standardized style.

3) Simpler code that is less likely to incur SAS Error 344.

For us, the return on investment has been tremendous. The SCOGEN system was implemented in two weeks, with further enhancements and refinements requiring an additional week.

Copies of the dBase III programs and documentation are available upon request from the author at:

Infant Health and Development Program
Building 460, Stanford University
Stanford, California 94305-2135

The Infant Health and Development Program (IHDP) is supported by grants to the Department of Pediatrics, Stanford University, the Frank Porter Graham Child Development Center, University of North Carolina, and the eight participating universities by the Robert Wood Johnson Foundation, with additional support to the Department of Pediatrics, Stanford University, from the Division of Maternal and Child Health of the U.S.P.H.S., the National Institute of Child Health and Human Development, the Pew Trusts, and Stanford University Center for the Study of Families, Children and Youth.

NOTES

*dBase III is a registered trademark of Ashton-Tate

**SAS is a registered trademark of SAS Institute Inc., Cary, North Carolina, USA

***The Infant Health and Development is a collaborative effort. The following are the senior staff involved in the study:

National Study Office:
Ruth T. Gross, M.D., Director; Donna Spiker, Ph.D., Deputy Director; Norman A. Constantine, Ph.D., Director of Data Analysis; Wendy L. Kreitman, Director of Field Operations (The Stanford University Center for the Study of Families, Children and Youth.)

Program Development Office:
Craig T. Ramey, Ph.D., Director; Donna Bryant, Ph.D., Associate Director; Joseph Sparling, Ph.D. and Barbara Wasik, Ph.D., Co-Directors of Curriculum Development; Isabelle Lewis and Claudia Lyons, Curriculum Development Specialists; Kaye Pandt, M.S.P.H., Director of Data Management and Statistical Computing (The Frank Porter Graham Child Development Center, University of North Carolina at Chapel Hill).

Participating Universities:
University of Arkansas for Medical Sciences (Arkansas); Albert Einstein College of Medicine (Einstein); Harvard Medical School (Harverd); University of Miami School of Medicine (Miami); University of Pennsylvania School of Medicine (Pennsylvania); University of Texas Health Science Center at Dallas (Texas); University of Washington School of Medicine (Washington); Yale University School of Medicine (Yale).

Site Directors:
Patrick H. Casey, M.D., Arkansas; Cecelia M. McCarton, M.D., Einstein; Michael W. Yogman, M.D., Harvard; Charles R. Bauer, M.D. and Keith G. Scott, Ph.D., Miami; Judith Bernbaum, M.D., Pennsylvania; Jon E. Tyson, M.D. and Mark Swanson, M.D., Texas; Clifford J. Salls, M.D. and Forrest C. Bennett, M.D., Washington; David T. Scott, Ph.D., Yale.

Education Directors:
Joan Rorex, M.Ed., Arkansas; Katy Lutzius, Einstein; Marcia Hartley, M.S., Harvard; Mimi Graham, Ph.D., Miami; Joanne Crooms, M.Ed., Pennsylvania; Beverly A. Mulvihill, M.Ed., Texas; Rebecca R. Fewell, Ph.D., Washington; Sandra E. Malmquist, M.A., Yale.

Bettye Caldwell, Ph.D., Educational Consultant, Arkansas; Ruth Turner, Ed.D., Dallas Independent School District Liaison, Texas.

Research Steering Committee:
Helena C. Kraemer, Ph.D., Chair (Stanford University); Charles R. Bauer, M.D.; J. Brooks-Gunn, Ph.D. (Educational Testing Service); Marie C. McCormick, M.D. (Brigham and Women's Hospital); Craig T. Ramey, Ph.D.; David T. Scott, Ph.D.; Sam Shapiro (The Johns Hopkins University School of Hygiene and Public Health); Donna Spiker, Ph.D.

ACKNOWLEDGMENTS

Sincere thanks to my colleagues at the IHDP, whose teamwork and dedication never ceases to amaze me. To Norm Constantine, Henk Pechler, Michael Shing, and Dana LeTendre for their consistant willingness to find a better way. To Nancy Greenwood, Mike Wrona, Cindy Duenas, and Pam Tro whose quality work has kept data flowing smoothly through the system.