Paper AD-93

Automating Simulation Studies with Base SAS® Macros

C. Vincent Hunter, Georgia State University

Abstract

Simulations are common in methodological studies in the social sciences. Even the most dedicated researchers have difficulty processing more than 100-200 repetitions, especially where different analysis programs must be processed in sequence. However, studies requiring hundreds or thousands of processing repetitions of data under the same set of study conditions are necessary for robust results. Where different study conditions are to be compared, the number of repetitions becomes even larger making the processing of an adequate number of iterations all but impossible when done one at a time.

Base SAS offers two tools for automating the processing of simulations: (1) Macros which divide the task into distinct jobs that may be easily modified for different conditions and number of iterations; (2) the ability to invoke non-SAS analysis programs (e.g., Mplus, R, and BILOG). Using these tools a researcher can create and process an appropriate amount of simulated data to obtain adequate power and control of Type I errors.

A recent simulation performed by the author is used as an example.

Introduction

Simulation studies are common in quantitative literature in the social sciences, especially in situations where statistical theory is weak or non-existent, where the distribution of a statistic is theoretically unknown, or where the assumptions for theory have been violated (Fan, Felsölvályi, Sivo, & Keenan, 2001; Harwell, Stone, Hsu & Kirisci, 1996). Such situations commonly occur both for studying different conditions within a method and for comparing different methods (Fidalgo, Hashimoto, Bartram & Muñiz, 2007; Meade, Lautenschlager, and Johnson, 2007).

Simulations study the distribution of statistics by drawing many samples of a statistic of interest from a population. Thus, simulations are statistical experiments that need to follow the same criteria that are appropriate for all empirical studies (Fan et al. 2001; Harwell et al., 1996)) These criteria include an appropriate sample size.

An appropriate number of iterations of the study will depend on the purpose of the study (Driels & Shin, 2004) as well as the area of investigation. In the area of military weapons, Driels and Shin recommend several thousand iterations. In psychometrics Harwell and colleagues (1996) recommend between 25 and 500. However many iterations are chosen, the process of creating data and running one or more analyses requires significant time and effort, especially when specialized software is required for parts of the analysis. Doing this process manually may be feasible for very small numbers of tasks and of iterations, but as the number of tasks and of iterations increases, the work load of manually creating data sets, saving them, and entering them into each subsequent process quickly becomes unmanageable (Gagne, Furlow, & Ross, 2009).

Base SAS offers two tools for automating the processing of simulations: (1) Macros which divide the task into distinct jobs that may be easily modified for different conditions and number of iterations; and (2) the ability to invoke non-SAS analysis programs (e.g., Mplus, R, and BILOG). Using these tools a researcher can automate the simulation process, creating and processing sufficient simulated data to obtain adequate power and control of Type I errors.

Overall Structure

A simulation process can be divided into four basic tasks: Housekeeping, Data Generation, Data Management, and Data Analysis. Housekeeping functions are done in base SAS, while the other three tasks are wrapped into macros. The most important task in Housekeeping is to set a global macro variable for the number of iterations to process in each macro.

```
%LET reps = 100 ;
```

Placing it here makes it available to all macros, and allows it to be changed in only one place after testing is finished before beginning the actual simulation, or if the researchers decide that a different number of iterations is more appropriate.

When processing large simulations on a micro-computer, the large volume of data written to the log and the output files may quickly exceed the available memory allocated to them. This can cause either a job failure or a warning dialog box which says that capacity has been exceeded, and requires user intervention before processing can continue. Clearing the log and output files will permit processing to resume. However, this destroys data that may be useful later on to debug the programs, to analyze data or to prepare reports from it. In any case, both the time savings for the researcher and the efficient processing of the simulation have been seriously and negatively affected.

These problems can be avoided easily by directing the log and the print output to data files.

```
PROC PRINTTO Log=   "C:\MySim\SIMLOG.LOG" ;
PROC PRINTTO Print= "C:\MySim\SIMPRINT.txt" ;
```

This keeps the log and print data available, and prevents interrupting the simulation processing.

Macros

Data generation, management, and analysis can be done within a single macro, or within a separate macro for each discrete task. A single macro has obvious efficiencies of processing time. Separate macros for each task have two potential advantages: (1) coding for each task can overlap processing; and (2) the separate tasks to be done are made obvious.

For example, a data generation macro that has been coded and tested, can be run, while the researcher is coding the next task. Branches in the process where one set of data sets goes to one analysis path, and another set goes to a second analysis path clearly track the original design. And the place in the series of steps that is being processed is obvious, so that the researcher has a better idea of what has been completed and what is yet to happen.

Macros That Incorporate Macros, and Macros That Use Unique Code

Macros can consist of unique code written specifically for that macro. They can also invoke other SAS macros, SAS programs, and programs written in other languages. Here is a sequence of three macros. The first generates two sets of data, one for each of two groups of observations, using an invoked macro. The second combines the two matched data sets generated by the first macro, separately identifying the different groups for an external analysis program. The third macro reformats the SAS data sets as .dat text

files without combining them for a different analysis program. These three macros could have been combined, especially the second and third. However, by keeping them separate the data generation macro was able to run while the other two were being written and tested. The outputs from the second and third macros were read into different analysis programs with different formatting requirements. Keeping them separate allowed the researcher to maintain a visual separation of processing flows within the structure of the SAS program. The decision of how to combine or separate tasks must be made based on the particular needs of the research and the researchers.

The following macro (GenData) calls in Macro IRTGEN (Whittaker, Fitzpatrick, Williams, & Dodd, 2003) to generate test data based on pre-existing parameters read from the input files. Two things to note about this code are the %INCLUDE statement and the %IRTGEN statement. Macro IRTGEN is stored in file path C:\MySim\ as data set IRTGENcase.SAS. The %INCLUDE statement invokes the named macro found in that file path. The %IRTGEN statement is the standard macro invocation with the key word parameters specified by macro IRTGEN. In this case the key word parameters are assigned from the key word parameters of the calling macro (GenData). The Do loop processes the macro the specified number of iterations. The iteration number is included as a part of the output data set name in the macro call to keep all generated data.

```
%MACRO GenData (SEED=, DataIn= , DataOut= , Items= , Examinees= ) ;

%DO r = 1 %TO &reps  ;
      %INCLUDE 'C:\MySim\IRTGENcase.SAS'  ;
      %IRTGEN(model=L3, data=&DataIn, out=&DataOut , NI=&Items ,
            NE=&Examinees )
%END;
%MEND GenData;

%GenData (SEED=1265133, DataIn=prmtrs.parmuf20102,
            DataOut=results.uf20102k33&r., Items=20, Examinees=3000);
```

The following macro is written with unique code to combine two data sets for two different groups of data into a single data set, while assigning a variable to distinguish the data observations by input group. The output data set is then written as a text file (.dat) for processing into an external program that requires combined data.

```
%MACRO CombineData (DSNref= , DSNfoc= , DataOut=output, Items= ) ;
%DO r = 1 %TO &reps  ;
DATA combined.&DataOut.&r ;
      SET results.&DSNfoc.&r (in=focref)
          results.&DSNref.&r            ;
             input
      IF focref THEN group = 1;
                ELSE group = 0;
      KEEP group case  r1-r&Items ;

DATA _NULL_ ;
      SET  "C:\MySim\Combined\&DataOut.&r" ;
      FILE "C:\MySim\Combdat\&DataOut.&r..dat" ;
      PUT      @1 group    @3  case  @11 r1-r&Items;
%END;
%MEND CombineData ;
```

This third macro is written with unique code to convert SAS data sets into a text file (.dat) for processing into an external program that requires separate data sets for the two input groups. This keeps a logical processing flow structurally separate from the other flows, but at the cost of slower processing.

```
%MACRO MakeDatFiles (Datain= ,Items= ) ;
%DO r = 1 %TO &reps ;

DATA _NULL_ ;
      SET  results.&DataIn&r ;
      FILE "C:\MySim\DATFILES\&DataIn&r..dat" ;
      PUT     @1 case
```

```
                @5 theta
                @17  (r1-r&Items) (1.) ;
RUN ;

%END;
%MEND MakeDatFiles ;
```

Macros That Incorporate SAS Programs

Entire SAS programs can be invoked by a macro using the %include statement:

```
%INCLUDE "C:\MySim\Programs\Scoring.sas" ;
```

In this case, all information required by the program must be available in the same format as it is coded in the program. This may not be convenient when running a simulation, because data set names will be likely to vary, and forcing them to be a single value could cause a loss of needed information about which data is being processed. A possible work-around would be to modify the invoked program to run as a macro.

An example of this would be the DIFCUT program (Nanda, Oshima, & Gagne, 2006). DIFCUT requires that the user to insert four input data set names into the code. (Because this is an IRT analysis program, these files are the focal and reference covariance parameter files, the focal score file, and the linking file.) By placing a %MACRO and a %MEND instruction at the beginning and end of the program and using key word parameters in place of hard coded data set names, the program is converted into a macro which can be included as seen previously:

```
%INCLUDE 'C:\MySim\macroDIFCUT.sas'  ;
 %macroDIFCUT(foc= &focal , ref= &reference , lnk= &link  );
```

The pertinent code in DIFCUT follows:

```
%MACRO MACRODIFCUT (foc=  , ref=  , lnk=  );
      DATA cov;
      /*In parentheses below, user must enter the name of their focal
      group file with the .cov extension*/
            INFILE IOB (&foc&r..cov) missover firstobs=3;

      DATA sco;
      /*In parentheses below, user must enter the name of their focal
      group file with the .sco extension*/
            INFILE IOB (&foc&r..sco) missover firstobs=3;

      DATA covref;
      /*In parentheses below, user must enter the name of their
      reference group file with the .cov extension*/
            INFILE IOB (&ref&r..cov) missover firstobs=3;
      DATA iplink;
      /*In parentheses below, user must enter the name of their PLINK
      file with the .csv extension*/
      INFILE IOL (L&lnk&r..csv) DSD missover;
%MEND macroDIFCUT;
```

Macros That Incorporate DOS Programs

Gagne, Furlow, and Ross (2009) provide guidance on invoking DOS programs from base SAS, and on creating a batch file to run selected IRT analysis programs. As an example the code to run BILOG-MG3 (Zimowski, Muraki, Mislevy, & Bock, 2003) is

```
DM " X 'C:\MySim\runBILOG'" ;
```

which submits a text file containing executable commands for the invoked program. For BILOG this file contains these instructions:

```
CD "C:\program files\bilogmg"
BLM1 MySimbatch
BLM2 MySimbatch
BLM3 MySimbatch
EXIT
```

The first line changes the working directory to where BILOG-MG3 is installed. The next three lines extract information from a batch file containing the instructions that BILOG-MG3 needs to execute. The last line causes the program to exit from the DOS environment where BILOG is running and return to SAS (Hurley, 2006). However, before BILOG-MG3 can be invoked, a batch (.blm) file with instructions for BILOG to execute must be created and saved.

In the following macro, the data step creates the batch file with the instructions for BILOG. The PUT statement writes 12 lines to the .blm file. Several of these lines have a semi-colon within the bracketing double quotes. These semi-colons are part of the BILOG instruction, not of the PUT statement. The final line has the only semi-colon that is outside of the double quotes. This semi-colon ends the PUT statement.

Because the lines written to the .blm file, have information that varies according to the varying characteristics of the simulation, each line of the put statement is preceded by a %STR function to mask the required special characters in the BILOG code during macro compilation. Several of the lines contain an embedded file path surrounded by required single or double quotes. Additionally each line of code is surrounded by double quotes. This quotes within quotes caused job failure. As a solution, the inner-pair of quotes was changed to a macro variable, &tic, which was assigned as a global variable in housekeeping:

```
%let tic=%STR(%') ;
```

The following macro shows the coding of the BILOG batch file, followed by the invocation call.

```
%MACRO runBILOG (Root= , items= ) ;

      %DO r = 1 %TO &reps  ;

      *****   Following section modified from Jas. Algina –
             Personal communication *********
      DATA _null_  ;
            file 'C:\Program Files\BILOGMG\MySimbatch.blm';
      PUT
      %STR(">TITLE    logistic simulated data &root&r " )   /
      "        two title lines required        "      /
      %STR(">GLOBAL DFName=&tic.C:\MySim\DATFILES\&root&r..dat&tic,
      NPArm=3, SAVE; ")/
      %STR(">SAVE SCORE=&tic.C:\MySim\BILOGfiles\&root&r..sco&tic ,
            " )/
      %STR("covariance=&tic.C:\MySim\BILOGfiles\&root&r..cov&tic , ") /
      %STR("PARM=&tic.C:\MySim\BILOGfiles\&root&r..par&tic ; ") /
      %STR(">LENGTH nitems = &Items; " ) /
      %STR(">INPUT NTOtal = &Items, NALt = 5, NIDch = 4 ; " ) /
      %STR(">Items INUMBERS=(1(1)&Items), INAMES=(I1(1)I&Items);"  ) /
      %STR(">TEST1 TNAme = itms-&r ; " ) /
      %STR("(4A1,12x,&Items.A1)  ") /
      %STR(">CALIB cycles =100; " ) /
      %STR(">score method=2;  ") ;

      *****     End Instructions
      *********

      *****                    Call Bilog
      *********

      DM " X 'C:\MySim\runBILOG'" ;
      RUN ;
      %END;
```

```
%MEND  runBILOG ;
```

Macros That Incorporate R Programs

Similarly to DOS programs, R (R Core Team, 2014) can be invoked from base SAS without using PROC IML. The basic structure is the same as for DOS programs: create a file containing the R code, then invoke R. The code in the following macro creates an R program to run R package plink (Weeks, 2010). R is then invoked using the SAS X command.

```
%MACRO runPLINK (ref= , foc= , items= ) ;

%DO r = 1 %TO &reps  ;

DATA _null_ ;
      file 'C:\Program Files\R\R-2.14.0\bin\runlink.r';
 PUT " library(lattice)"                                  ;
 PUT " library(plink)"                                                ;
 PUT " setwd(C:/MySim/BILOGfiles')"          ;
 PUT " gf.pars <- read.bilog('&ref&r..par')"     ;
 PUT " gr.pars <- read.bilog('&foc&r..par')"    ;
 PUT " common <- matrix (c(1:&items,1:&items), &items, 2)"     ;
 PUT " pars <- combine.pars(list(gr.pars, gf.pars), common)"       ;
 PUT " out <- plink(pars, rescale = 'SL', method = 'SL',   "            ;
 PUT "         weights.t = as.weight(30, normal.wt = TRUE) )"       ;
 PUT " conname <- link.con(out)"                                  ;
 PUT " setwd(C:/MySim/LinkFiles') "                          ;
 PUT " write.csv(conname,'L&foc&r..csv') "              ;
 PUT "q() "                                                       ;
 PUT "n"                                                          ;
*****     End create R program                *********
OPTIONS XWAIT XSYNC ;
X ' "c:\Program Files\R\R-2.14.0\bin\r.exe" CMD BATCH --vanilla --slave
    "c:\Program Files\R\R-2.14.0\bin\runlink.r"  ';
 RUN ;
%END;

%MEND  runPLINK ;
```

In the above code, the XSYNC cause SAS to wait until R has finished before continuing, and the XWAIT option waits until EXIT is entered to the command line (SAS Institute, n.d.). The two R instructions “q()”, which closes R, and “n”, which does not save the R work space, return control to just as EXIT would from DOS.

The X command first sets the active directory to where R is stored. CMD BATCH runs R in batch mode. The vanilla option reduces the number of default options used by R (UMBC, 2013), and the slave option suppresses the R initialization lines as well as prompts, and commands (Smith, 2009).

Conclusion

SAS is an excellent tool for automating the simulation process. Not only can SAS create simulated data and perform statistical analysis, but it has the ability to invoke and run other SAS programs, as well as programs written in other programming languages, such as BILOG or R, using the X command combined with simple DOS calls (Gagne et al., 2009). Automating the simulation using SAS macros will free time for researchers to be productive in other areas, without having to check on the status of simulation processing more than occasionally.

References

Driels, M. R., & Shin, Y. S. (2004). Determining the number of iterations for Monte Carlo simulations of weapon effectiveness. Monterey, CA: Naval Post Graduate School. Downloaded on 12 July 2015 from http://www.dtic.mil/dtic/tr/fulltext/u2/a423541.pdf

Fan, X., Felsölvályi, A., Sivo, S., & Keenan, S. (2001). *SAS for Monte Carlo Studies: A guide for quantitative researchers.* Cary, NC: SAS Institute.

Fidalgo, A., Hashimoto, K., Bartram, D., & Muñiz, J. (20007). Empirical Bayes versus standard Mantel-Haenszel statistics for detecting differential item functioning under small sample conditions. *The Journal of Experimental Education, 75*(4), 293-314.

Gagne, P., Furlow, C., & Ross, T. (2009). Increasing the number of replications in item response theory simulations: Automation through SAS and disk operating system. *Educational and Psychological Measurement,* 69(1), 79-84.

Harwell, M., Stone, C. A., Hsu, T-C., & Kirisci, L. (1996). Monte Carlo studies in item response theory. *Applied Psychological Measurement, 20*(2), 101-125.

Hurley, G. J. (2006, March) *Xamining the X Statement (and Some Other Xciting Code).* Paper presented at the annual meeting of the SAS Users Group International, San Francisco, CA.

Meade, A., Lautenschlager, G., & Johnson, E. (2007). A Monte Carlo examination of the sensitivity of the differential functioning of items and tests framework for tests of measurement invariance with Likert data. *Applied Psychological Measurement, 31*(5), 430-455.

Nanda, A., Oshima, T., & Gagne, P. (2006). DIFCUT: A SAS/IML program for conducting significance tests for Differential Functioning of Items and Tests (DFIT). *Applied Psychological Measurement, 30*(2), 150-151.

R Core Team. (2014). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org

SAS Institute Inc. (n.d.). *Running Windows or MS-DOS Commands from within SAS.* Retrieved from http://support.sas.com/documentation/cdl/en/hostwin/63047/HTML/default/viewer.htm#n14rxpt7iu6iksn17r895o6kcpgi.htm

Smith, D. (2009 June 24). *Running scripts with R CMD BATCH* [web log post] Retrieved from http://blog.revolutionanalytics.com/2009/06/running-scripts-with-r-cmd-batch.html

University of Missouri Bioinformatics Consortium (UMBC). (2013). *How to Run R.* Retrieved from http://umbc.rnet.missouri.edu/resources/How2RunR.html

Weeks, J. P. (2010). plink: An R Package for Linking Mixed-Format Tests Using IRT-Based Methods. *Journal of Statistical Software, 35*(12), 1-33.

Whittaker, T., Fitzpatrick, S., Williams, N., & Dodd, B. (2003). IRTGEN: A SAS Macro Program to Generate Known Trait Scores and Item Responses for Commonly Used Item Response Theory Models. *Applied Psychological Measurement, 27*(4), 299-300.

Zimowski, M., Muraki, E., Mislevy, R., & Bock, R. D. (2003). BILOG-MG (Version 3) [computer software]. Lincolnwood, IL: Scientific Software International, Inc.

ACKNOWLEDGMENTS

Acknowledgements go to Dr. Jihye Kim, Kennesaw State University, for proofreading a draft of this paper and for her valuable comments.

CONTACT INFORMATION

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

C. Vincent Hunter
Georgia State University
College of Education & Human Development
Department of Educational Policy Studies
30 Pryor Street
Suite 450
Atlanta, GA 45252
Phone: 404-277-9841
E-mail: chunter1@gsu.edu