USING SAS® SOFTWARE FOR MONTE CARLO SIMULATION

Yi-Cheng Wu, The University of Alabama, Tuscaloosa, AL USA
James E. McLean, The University of Alabama, Tuscaloosa, AL USA

ABSTRACT

Most Monte Carlo simulation studies have used programming languages such as Fortran. SAS® software is often conceived of by statisticians as an analysis package rather than a major programming language. Because of this, SAS is rarely used for Monte Carlo simulation. The purpose of this paper is to propose a Monta Carlo simulation process using only SAS code. The methods of selecting seed numbers for SAS random functions and the problems of using the computer clock to generate seeds are also discussed.

A computer simulation system using only SAS code was developed for studying the statistical power of five ANOVA procedures. SAS programs for simulation are much shorter and more understandable than equivalent progrems written in primary programming languages. The greatest contribution of the system is its flaxibility. It can be used with minor modification for a multitude of studies. The primary disadvantage of using SAS rather than a programming language is that it uses more computer CPU time. However, this disadvantage can be overcome using a high speed computer such as the IBM® 3090/400E computer used in this study.

INTRODUCTION

Most Monte Carlo simulation studies use programming languages such as Fortran. SAS is often conceived of by statisticians as an analysis package rather than a major programming language. Because of this, SAS is rarely used for Monte Carlo simulation studies. The purpose of this paper is to propose a Monte Carlo simulation process using only SAS code. The methods of selecting seed numbers for SAS functions and the problems of using the computer clock to generate seeds are also discussed.

A computer simulation system consisting only of SAS code was developed for a research project comparing statistical powar of five analysis procedures under 48 sets of experimental conditions. The five analysis procedures were: the one-way analysis of variance; two-block, four-block, and eight-block block designs; and the analysis of covariance. The 48 sets of experimental conditions were the combinations of four levels of the number of treatments (2, 3, 4, and 5), three levels of the number of subjects per treatment (8, 40, and 72), and four levels of the correlation coefficient (.00, .28, .56, and .84). The results of this study indicated that different procedures should be used depending on the set of conditions. The correlation coefficient between the concomitant and the dependent variable was the critical factor that should influence the choice. The one-way ANOVA was the best choice when there was no correlation, block designs were preferred when the corralation was low, and the ANCOVA achieved the highest power when the correlation was high. With moderate correlation, block designs should be selected only when the number of treatmants and the number of subjects per treatment were large; otherwise, the ANCOVA should be used. Block designs and the ANCOVA became more powerful and the optimal number of blocks for a block design increased as tha correlation coefficient, the number of treatments, and the number of subjects per treatment increased.

THE COMPUTER SIMULATION SYSTEM

The generation and the analyses of the data were accomplished on the IBM 3090/400E computer at The University of Alabama. The computer codes for one set of the 48 sets of experimental conditions are listed in the Appendix. The computer codes for the other sets of experimental conditions are essentially the same. Bivariate correlated data were generated using SAS commands provided by Clark and Woodward (1992). Thesa commands generate random data from a bivariate normal distribution with a mean of 0, a variance of 1, and the user-specified correlation coefficient.

The computer simulation system included one executable file and two SAS programs (International Business Machines®, 1988a; International Business Machines®, 1988b; SAS Institute Inc., 1990a; SAS Institute Inc., 1990b). The computer code is listed in the Appendix. For each of the 48 sets of experimental conditions, the executable file ran the first SAS program 1,000 times, then ran the second SAS program. The first SAS program generated a set of data, analyzed that set of data with the five analysis procedures being compared, and output the results of the analyses to a data file. After the first SAS program had run for 1,000 times, the data file contained 1,000 records of the results of the analyses. The second SAS program calculated the empirical powers of the analysis procedures based on the 1,000 sets of data each. In order to obtain three observations per cell, the executable file was run three times for each of the 48 sets of experimental conditions. Totally, there were 144,000 (1,000 X 3 X 48) sets of data generated and 720,000 (5 X 144,000) analyses conducted. For complete information regarding the designs, procedures, and results from this study, please refer to Wu (1994).

SEEDS

A seed must be provided to generate a set of random data using SAS random functions. The values of seeds can be any integer ranged from 1 to $2^{31} - 2$ (i.e., 2,147,483,646). The same seed will always generate the same set of data; therefore, a Monte Carlo study usually requires a large number of seeds. Three methods of selecting seeds were tested: (1) arbitrarily selecting an integer from 1 to 2,147,483,644 as the seed for the UNIFORM or RANUNI function to generate the required number of seeds in a DO loop, (2) using 0 as the seed for the UNIFORM or RANUNI function to generate the raquired number of seeds in a DO loop, and (3) equally spacing the required number of seeds in the range of 1 to 2,147,483,646. If method 1 or 2 is chosen, it needs to be output to a SAS data set after the required number of seeds are generated. The SAS program that generates random data needs to access that SAS data set to acquire the seed for each run. One technique for accomplishing this is to delete the seed after each run so that a new seed will be accessed for each subsequent run. The computer code for this is as follows:

```
DATA TEMP;
  SET SDS.SEED;
DATA SDS.SEED;
  SET TEMP;
  IF _N_ = 1 THEN DELETE;
```

Other techniques are also feasible. For example, a flag could be set for each seed number as it is used and checked before the next run to avoid accessing a used seed. If method 3 is chosen, an initial seed needs to be coded in the executable file and an integer needs to be used as an increment to equally space the seeds over the range of possible seeds. Pilot studies indicated all three methods yielded satisfactory results. Methods 1 and 2 generate true random seeds while method 3 generates systematic random seeds. This study adopted the third method because it seemed to employ the seeds more systematically and representatively.

Users can let the computer clock generate the seed by specifying a seed value of 0. Using a computer clock to generate random data was tested and found to have three problems: (1) The computer clock may not increment enough to generate different data; (2) the computer clock may generate repeated or patterned data; and (3) the program may not be executed because the computer clock generates invalid seeds. Therefore, it is recommended that positive seeds be used instead of the computer clock. Using positive seeds also has the advantage of making the experiment replicable.

DISCUSSION AND IMPLICATIONS

Several pilot studies which examined the parameters and distributions of the mean, variance, and correlation coefficient were conducted before the experiment, and the resulting sampling distributions of the statistics were checked after the experiment. Furthermore, before the experiment, the powers of the one-way ANOVAs that were controlled et .50 were checked. The resulting empirical powers had a mean of .50 and a variance of .0001; also the within-cell mean square error was only .0001—all these provide evidence that the computer simulation system is able to generate data that meet specifications. It is recommended that future research adapt the system to examine relatad problems.

The computer simulation system developed for this study could be modified easily for a multitude of other studies. For example, it could be used to investigate other criteria such as Type I errors, examine other levels of the experimental conditions, or test blocking methods in addition to the post-hoc blocking used in this study. This study does not include the treatment-by-block interaction in the block designs since the interaction does not exist in the population. Future studies could also examine the effects of including the interaction using essentially the same computer codes, or, by varying the parameters of the population, examine the effects of including and excluding the interaction when the interaction does exist in the population. The optimal number of blocks for e block design could ba investigated by including other feasible blocking schemes such as 5-, 10-, 20-, and 40-blocks for the condition of 40 subjects per treatment.

Countless other studies could be done using this computer simulation system as a frame work. The SAS programs are much shorter and more understandable as compared to equivalent programs written in Fortran or other programming languages. The primary disadvantage of using SAS rather than a programming language is that SAS uses more computer CPU time. However, this disadvantage can be overcome by using a high speed computer such as the IBM 3090/400E computer used in this study.

REFERENCES

Clark, M. R., & Woodward, D. E. (1992). Generating random numbers with base SAS software. *Observations, 1*(4), 12-19.

International Business Machines. (1988a, March). *VM/XA System Product interpreter: Reference manual* (1st ed.). International Business Machines.

International Business Machines. (1988b, March). *VM/XA System Product interpreter: User's guide* (1st ed.). International Business Machines.

SAS Institute Inc. (1990a). SAS procedures guide, Version 6, Third Edition, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990b). *SAS/STAT user's guide: Volume 2,GLM-VARCOMP Version 6, Third Edition,)*. Cary, NC: Author.

Wu, Y. (1994). *To block or covary a concomitant variable: Which is more powerful?* Unpublished doctoral dissertation, The University of Alabama, Tuscaloosa, AL.

APPENDIX

COMPUTER CODES

Executable File

```
/*   */
ADDRESS COMMAND
"ERASE PVALUE DATA A"
NUMERIC DIGITS 10
TIME = 1
DO WHILE TIME < 1001
SEED = 2132560 + (TIME -1) * 2147483
"EXECIO 1 DISKW" NEWSEED DATA A "(STRING" SEED
"EXEC SAS T57284"
"ERASE NEWSEED DATA A"
TIME=TIME+1
END
"EXEC SAS T57284P"
```

First SAS Program (T57284 SAS A)

```
CMS FILEDEF INDATA DISK NEWSEED DATA A;
CMS FILEDEF PVALUE DISK PVALUE DATA A (LRECL 210
BLKSIZE 210 RECFM FBS;
CMS FILEDEF SASLIST DISK T57284 LISTING A;
DATA BIVNORM (DROP=I);
  INFILE INDATA;
    INPUT SEED;
  DO I=1 TO 72;
    GROUP=1;
    X=RANNOR(SEED);
    Y=.84*X+SQRT(1-.84**2)*RANNOR(SEED);
    OUTPUT;
  END;
  DO I=1 TO 72;
    GROUP=2;
    X=RANNOR(SEED);
    Y=.84*X+SQRT(1-.84**2)*RANNOR(SEED);
    Y=0.0951+Y;
    OUTPUT;
  END;
```

```
 DO I=1 TO 72;
   GROUP=3;
   X=RANNOR(SEED);
   Y=.84*X+SQRT(1-.84**2)*RANNOR(SEED);
   Y=0.1903+Y;
   OUTPUT;
 END;
 DO I=1 TO 72;
   GROUP=4;
   X=RANNOR(SEED);
   Y=.84*X+SQRT(1-.84**2)*RANNOR(SEED);
   Y=0.2854+Y;
   OUTPUT;
 END;
 DO I=1 TO 72;
   GROUP=5;
   X=RANNOR(SEED);
   Y=.84*X+SQRT(1-.84**2)*RANNOR(SEED);
   Y=0.3805+Y;
   OUTPUT;
 END;
PROC SORT;
  BY GROUP X;
DATA BIVNORM;
  SET BIVNORM;
  BN=MOD(_N_,72); IF BN=0 THEN BN=72;
  IF BN<=36 THEN B2=1; ELSE B2=2;
  IF BN<=18 THEN B4=1; ELSE IF BN<=36 THEN B4=2;
  ELSE IF BN<=54 THEN B4=3; ELSE B4=4;
  IF BN<=9 THEN B8=1; ELSE IF BN<=18 THEN B8=2;
  ELSE IF BN<=27 THEN B8=3;
  ELSE IF BN<=36 THEN B8=4;
  ELSE IF BN<=45 THEN B8=5;
  ELSE IF BN<=54 THEN B8=6;
  ELSE IF BN<=63 THEN B8=7;
  ELSE B8=8;
PROC PRINT;
PROC CORR DATA=BIVNORM;
  VAR X Y;
  BY GROUP;
PROC GLM;
  CLASS GROUP;
  MODEL Y=GROUP/SS3;
PROC GLM;
  CLASS GROUP B2;
  MODEL Y=GROUP B2/SS3;
PROC GLM;
  CLASS GROUP B4;
  MODEL Y=GROUP B4/SS3;
PROC GLM;
  CLASS GROUP B8;
  MODEL Y=GROUP B8/SS3;
PROC GLM;
  CLASS GROUP;
  MODEL Y=GROUP X/SS3;
DATA;
  INFILE SASLIST;
  INPUT WORD1 $ WORD2 $ @;
  FILE PVALUE MOD;
  IF WORD1 = 'X' AND WORD2 ='72' THEN DO;
    INPUT MEAN STDDEV;
    PUT MEAN 6.4 STDDEV 6.4 @;
    INPUT Y $ N MEAN STDDEV;
    PUT MEAN 6.4 STDDEV 6.4 @;
    END;
  ELSE IF WORD1="X" AND WORD2 = '1.00000' THEN DO;
    INPUT CORR;
    PUT CORR 6.4 @;
    END;
 ELSE IF WORD1="GROUP" AND WORD2 = '4' THEN DO;
    INPUT SS MS F PR;
    PUT PR 6.4 @;
    INPUT BLOCK $ DF SS MS F PR;
    PUT PR 6.4 @;
    END;
```

Second SAS Program (T57284P SAS A)

```
CMS FILEDEF INDATA DISK PVALUE DATA A;
DATA PVALUE;
INFILE INDATA;
INPUT (G1XMEAN G1XSD G1YMEAN G1YSD G1CORR
    G2XMEAN G2XSD G2YMEAN G2YSD G2CORR
    G3XMEAN G3XSD G3YMEAN G3YSD G3CORR
    G4XMEAN G4XSD G4YMEAN G4YSD G4CORR
    G5XMEAN G5XSD G5YMEAN G5YSD G5CORR
    GROUP1B BLOCK1B GROUP2B BLOCK2B GROUP4B
    BLOCK4B   GROUP8B BLOCK8B GROUPANC
    BLOCKANC) (35* 6.4);
   TOTAL=0;
   G1BSG=0;
   B1BSG=0;
   G2BSG=0;
   B2BSG=0;
   G4BSG=0;
   B4BSG=0;
   G8BSG=0;
   B8BSG=0;
   GANCSG=0;
   BANCSG=0;
   TOTAL=1;
   IF GROUP1B <= 0.05 THEN G1BSG=1;
   IF BLOCK1B <= 0.05 THEN B1BSG=1;
   IF GROUP2B <= 0.05 THEN G2BSG=1;
   IF BLOCK2B <= 0.05 THEN B2BSG=1;
   IF GROUP4B <= 0.05 THEN G4BSG=1;
   IF BLOCK4B <= 0.05 THEN B4BSG=1;
   IF GROUP8B <= 0.05 THEN G8BSG=1;
   IF BLOCK8B <= 0.05 THEN B8BSG=1;
   IF GROUPANC <= 0.05 THEN GANCSG=1;
   IF BLOCKANC <= 0.05 THEN BANCSG=1;
PROC FREQ;
   TABLE G1BSG -- BANCSG;
PROC SUMMARY DATA=PVALUE;
   VAR G1XMEAN -- BANCSG;
   OUTPUT OUT = DESCRIPT;
PROC PRINT DATA=DESCRIPT;
PROC UNIVARIATE DATA=PVALUE PLOT NORMAL;
   VAR G1XMEAN -- BLOCKANC;
```