An Overview of the SAS® System Random Number Generators

Bart Killam

SAS Institute Inc.

Cary, NC

Abstract

The SAS USER'S GUIDE: BASICS, Version 5 Edition, lists nine random number generators (not counting the UNIFORM and NORMAL, which are older implementations of generators for the uniform and normal distributions respectively). Of these, five are correspond to common continuous distributions in statistics. The importance of the RANUNI generator lies in the fact that all the other continuous generators use RANUNI and transform methods to generate their respective streams. The RANUNI generator is a member of a general class called "Multiplicative Congruential Generators with Modulus 2^{31}-1". A description of the algorithm for these generators s given, along with a description of the transform method used to obtain the RANNOR generator from the RANUNI. Finally, some common missuses of the random number generators in SAS programming is discussed.

The need to generate a stream of random numbers that follows a given distribution occurs often in data processing, statistical and operations research. A common use of generators in the SAS System is in programs that take a random sample from a large data set. Other common uses include the investigation of the distribution of statistics whose theoretical distribution is difficult to calculate, and randomization of residuals in bootstrap studies.

While a program to obtain a random sample might be quite involved (see the SAS Applications Guide, 1987 Edition), the random aspect of the selection is usually quite simple, and involves the RANUNI generator. In the investigation of distributions, the statistic of interest can sometimes be expressed as a complicated function of basic random variables with common distributions. These latter random variables can be generated using the appropriate SAS random number generator. Randomization of residuals in bootstrap applications is again a sampling problem that typically uses the RANUNI generator.

We now focus attention on sequences and generators that are uniform between 0 and 1. A basic theorem in probability theory shows how to construct a sequence $(Y_j:j=0,1,..)$ following a distribution $F(\bullet)$ from a uniform sequence $(U_j:j=0,1,..)$ between 0 and 1. By letting

$$Y_i = F^{-1}(U_i),\ i=0,1,..,$$

the sequence $Y_0,Y_1,..$ has the desired distribution.

The computation of $F^{-1}(\bullet)$ presents problems for some distributions. Well known methods, such as Box-Muller for the Normal (Box and Muller, 1958), and acceptance-rejection for the Gamma (Fishman, 1978), are available to get around this difficulties for these common distributions. Marsaglia (1984), gives an "Exact-Approximate" method for generating random variates for a given distribution $F(\bullet)$ without ever computing $F^{-1}(\bullet)$.

In the applications described above it is clear that the validity of the results depend crucially on the "randomness" and the distributional properties of the generators. Before discussing validation, we must define terms such as "randomness" and "uniform".

A stream of numbers $U_0,U_1,...$ between 0 and 1 is "uniform" if for any $0 <= \alpha < \beta <= 1$, the proportion of U's satisfying $\alpha < U < \beta$ tends to $\beta-\alpha$. This simple concept can be tested for the RANUNI in a data step. (see figure 1) The notion of randomness is more subtle. A simple example due to Knuth (1981) shows a uniform sequence that is not random. Let $(U_j:j=0,1,..)$ and $U'_j:j=0,1,...)$ be two uniform sequences and construct the sequence $W_0,W_1,...$ by

$W_0 = (1/2)*U_0$

$W_1 = (1/2)*U'_0 + 1/2$

$W_2 = (1/2)*U_1$

$W_3 = (1/2)*U'_1 + 1/2$, and so on.

This sequence would be "uniform" in the sense described above, but is certainly not random. To describe "randomness" we must consider consecutive numbers in the stream.

For example, let $0 < v_1 < w_1 < 1$, $0 < v_2 < w_2 < 1$. For the stream of numbers $(U_0,U_1,....)$, let S(j) denote the statement

$v_1 < U_j < w_1$, and

$v_2 < U_{j+1} < w_2$.

Let $\nu(n)$ be the number of times S(j), j=0,..,n-1 is true. Then a reasonably "pair-wise random", or 2-distributed sequence would have the property

$$\nu(n)/n \to (w_1-v_1)(w_2-v_2) \text{ as } n \to \infty.$$

Note that the sequence described above does not have this property for the choice $v_1=v_2=1/2$, and $w_1=w_2=1$.

This idea can be generalized for k-tuples in the following way. Let $0 <= v_j < w_j <= 1$, j=1,..,k, and define S(j) as the statement

$$v_1<=U_j<w_1, \ \ldots\ , \ v_k<=U_{j+k-1}<w_k.$$

Let $\nu(n)$ be the number of times S(j), j=0,...,n-1 is true. Then the sequence $U_0,U_1,\ldots$ is k-distributed if

$$\nu(n)/n \to (w_1-v_1)(w_2-v_2)\ldots (w_k-v_k),$$

as $n \to \infty$.

A sequence $U_0,U_1,\ldots$ is ∞-distributed if it is k-distributed for all k=1,2,... A sequence that is that is ∞-distributed can be considered random (Knuth, (1981), pp. 152-7). However, there are obvious difficulties in empirically showing a given sequence is ∞-distributed.

In practice, to determine if a given sequence $U_0,U_1,\ldots$ is random, a small set of tests are applied. These tests are logically divided into theoretical and empirical tests. Empirical tests use actual numbers; calculations are performed on a finite subset of consecutive numbers from the sequence. Theoretical tests use number-theoretic results applied to the entire sequence. For the generators considered here, the sequences are defined by recursion relations.

Empirical tests include the Equidistribution test (or frequency test), the Serial test, the Gap test, the Poker test, the Coupon Collectors, Permutation test, Run test, Maximum of t test, Collision test, and Serial Correlation test (Neiderreiter, 1976, and Knuth, 1981), and tests considered by Fishman and More (1982), and tests based on empirical distribution functions (Stephens, 1974).

Theoretical tests include the spectral test (Coveyou and MacPherson, 1967), and the lattice test (Marsaglia, 1972). These tests exploit the fact that linear congruential generators induce a pattern similar to lattice structures studied in solid state physics. A very readable explanation of these theoretical tests is given in Fishman and Moore, (1986)

While most of the empirical tests consider one-dimensional properties of generators, uniformity in higher dimensions is important in some applications. Empirical tests such as the Serial Test and the theorectical tests mentioned above have been used in considering uniformity of generators in dimensions 2-6. A well known example of a generator that performs reasonably well in one dimension and poorly for higher dimensions is the RANDU generator (Fishman and Moore, 1982,1986). This generator was formerly in the IBM Scientific Subroutine Library.

With this short review of the definition of a uniform random sequence, and the means of testing such a sequence, we turn to the algorithms used to generate such sequences on a digital computer.

Since the advent of the computer, a number of methods of generating uniform random numbers has been suggested. Variations of John Von Neumman's "middle square" method have been proposed, while Knuth (1981), as an example of a poor generator, shows an algorithm that itself if random. However, the most popular generator in use today is a type called "linear congruential generators", introduced by D. H. Lehmer in 1949 (Lehmer, 1951). The SAS RANUNI generator is in this class.

A linear congruential generators is completely defined by the four quantities (U_0,M,A,C). U_0 is the seed, "A" is the multiplier, "C" is the increment, and "M" is the modulus. The random sequence is defined recursively by

$$U_{n+1} = (A*U_n + C) \bmod M, \quad n = 0,1,2,\ldots.$$

The period T of a generator is the number of elements produced by the generator before it begins repeating. It is clear that any generator of the form given above can have period no larger than the modulus M. It is also clear that a linear congruential generator is completely deterministic. For example, suppose we take C = 0, M = 999, and A = 173. The MG1 algorithm (Payne, Rabung, and Bogyo, 1969) can be implemented using a SAS DATA step to show the effect of a finite period.

First we consider the MG1 algorithm in generic code. Letting A, X, and Y be variables capable of holding p binary digits, and W a temporary variable holding up to 2*p binary digits. For $M = 2^{p-1}-1$, A set equal to the multiplier, and X initially containing the seed, the algorithm is

1. W <-- X * A

2. Y <-- INT($W/2^{P-1}$)

3. X <-- W - Y * 2^{P-1}

4. X <-- X + Y

5. Y <-- INT($X/2^{P-1}$)

6. IF Y ¬= 0, X <-- X - (2^{P-1} - 1)

7. RETURN X

(see figure 2)

The seeming paradox of using a completely determistic sequence that repeats after a finite number of terms is handled by:

1) making the period so large as to be practically

infinite, and

2) testing the sequence for uniformity

and randomness using the tests described above.

An additional consideration in random number generation is efficiency. By letting c = 0 in the definition of linear congruential generators, the addition operation is avoided. Letting the modulus M be a power of 2 allows the modulus operation to be performed as a shift operation, increasing computational efficiency.

In fact, if $M = 2^{\beta}$, where β is the word length of the computer, even the shift operation is eliminated. In the IBM 360/370 architecture, the word length is 32 bits. Hence a candidate modulus M is 2^{32}. Marsaglia (1972) introduced a generator with M is 2^{32}, A = 69069, U_0 odd. The RANDU generator mentioned above has M = 2^{31}, A = 65539.

A problem with generator using $M = 2^{\beta}$ is that the period T is at most M/4. If, instead, M is a prime number, and the multiplier A is a "primitive root" of M, then a maximal period of M - 1 is attained. By choosing M = 2^{31}-1, and a primitive root multiplier, generators almost as efficient as those with M = 2^{32} are possible, while the period T is substantially increased (Fishman and Moore, 1982).

The SAS RANUNI generator uses M = 2^{31}-1, A = 397204094, C = 0, and $U_0 > 0$ (Actually, a seed $U_0 <= 0$ induces special action for RANUNI, this will be discussed below). The MG1 algorithm is implemented in 370 assembler.

This generator has been tested extensively in the literature. Hoaglin (1976) considered 50 multipliers for M=2^{31}-1, all of which were primitive roots of M to insure maximum period. In this study, the spectral and lattice tests were applied to these multipliers for dimensions 2-6. Included in this study was the RANUNI multiplier, A = 397204094.

Fishman and Moore (1982) subject 17 multipliers to a battery of empirical tests for independence and uniformity in 1,2, and 3 dimensions. The multipliers were chosen as follows. Multipliers I-II are from the APL, IMSL, and SIMSCRIPT packages, III-VIII were the "best" multipliers in the Hoaglin study, IX-XVI were the "poorest" from the Hoaglin study, and for reference, the RANDU generator was included as multiplier XVII. The RANUNI generator, corresponding to multiplier V, performed well in these tests.

In a revision to Hoaglin (1976), the algorithm used to approximate the lattice test calculations was replaced by an exact method substantially changing the rankings of the generators he considered. Fishman and Moore (1986) recompute the theoretical tests to find an optimal multiplier among all primitive root multipliers for M = 2^{31}-1 (some 500 million of them). The RANUNI generator is no longer among the best multipliers using these criteria.

It is interesting to note however, that when Fishman and Moore (1986) subject the top five multipliers to the same battery of tests used in their earlier (1982) paper, the RANUNI generator performs about the same as these five generators.

Clearly the ratios used in Fishman and Moore (1986) are useful for ranking and picking out grossly deficient multipliers. However, it is not clear how to compare, for example, a S_1 ratio of .8673 for k=2 dimensions for the best multiplier with the corresponding RANUNI value of S_1 = .5564. The results of the empirical tests indicate very little difference in the two generators in independence, and uniformity in 2 and 3 dimensions.

In addition to the empirical tests on the RANUNI in the literature, SAS Institute Inc. has done extensive testing on all the generators available in the DATA STEP using an empirical distribution function (EDF) approach described by Stephens (1974). These tests involve the Kolmogorov-Smirnov, Cramer-von Mises, Watson, and Anderson-Darling statistics. As discussed in

Stephens (1974), these tests have good power, comparable to other well known tests.

The hypothesis being tested here is that the stream $(Y_j:j=0,1,..)$ comes from a completely specified distribution $F(\cdot)$. All the generators were tested for various seeds and sample sized. Most of the tests did not reject at the 15% level. The most familiar of these tests, the Kolmogorov-Smirnov, is now presented for 500 samples of 10,000 observations for the RANNOR generator (see figure 3).

The RANNOR Generator

The most common distribution in Statistics is the Normal distribution. The RANNOR generator produces Normal random deviates with mean 0 and variance 1. The RANNOR generator uses the RANUNI along with the Polar, or Box-Muller method to generate its stream (Box and Muller, 1958, and Knuth, 1981). An outline of the Polar method is now given.

Consider the probability element using the joint density of two independent normal (0,1) random variables:

$$f(x_1,x_2)dx_1dx_2 = (1/2\pi)\exp(-(x_1^2+x_2^2)/2)dx_1dx_2$$

Converting to polar coordinates (x_1,x_2) --> (r,θ) and factoring yields:

$$f(r,\theta)rdrd\theta = \{\exp(-r^2/2)d(r^2/2)\}\{d\theta/2\pi\}$$

The first density is exponential in the variable $r^2/2$, while the second is uniform on the interval $(0,2\pi)$.

Random deviates corresponding to these distributions are easily constructed by letting

$$\theta = 2\pi U, \text{ and}$$

$$r = (-2\log(U'))^{1/2},$$

where U, U' are uniform (0,1).

Finally, letting

$$Y = R\cos(\theta) = (-2\log(U'))^{1/2}\cos(2\pi U),$$

Y has a Normal (0,1) distribution.

The Polar methods is an exact result, assuring that if $U_0,U_1,\ldots$ are U(0,1), then Y, generated as above, is N(0,1). We now consider the remaining continuous generators, outlining their method of generation and giving references for more details.

RANEXP

The RANEXP generator produces exponentially distributed random variates with scale parameter 1. The simple form of the distribution function, $F(x) = 1 - \exp(-x)$, allows the inverse-transform method to be used. The general form for the exponential distribution is

$$F(x) = 1 - \exp(-\lambda x),\ x > 0,\ \lambda > 0.$$

By forming $Z = \lambda X$, where $X \sim 1 - \exp(-x)$, Z has an exponential distribution for a general $\lambda > 0$.

RANGAM

The RANGAM generator produces random variates that have the Gamma distribution with shape parameter α. The usual form of the Gamma distribution is

$$F(x) = (\lambda^\alpha/\Gamma(\alpha))x^{\alpha-1}\exp(-\lambda x),\ x>0,\ \lambda>0,\ \alpha>0$$

(Mood, Graybill, and Boes, 1974). The RANGAM generator with second argument ALPHA produces random variates with α = ALPHA, and $\lambda = 1$. To obtain a stream with a general $\lambda > 0$, use the RANGAM generator with the appropriate shape parameter to obtain $(X_j:j=0,1,..)$. Form $(Y_j:j=0,1,..)$ by

$$Y_j = \lambda X_j.$$

For details on producing Chi-square and Beta distributed random deviates from RAMGAM, see the SAS USER'S GUIDE: Basics, Version 5 Edition, pp. 265-7. When α = ALPHA is an integer, $\Gamma(\alpha)$ can be computed as a factorial, and the distribution is easily inverted, allowing use of the inverse-transform method.

When α is not an integer, two independent streams are generated, one for α=INT(ALPHA) (using the inverse-transform method) and the other for α=ALPHA-INT(ALPHA) (using acceptance-rejection method). From the reproductive property of the gamma distribution, the sum of these two gammas has the desired distribution (see Fishman, 1978).

The RANCAU generator uses the acceptance-rejection method to produce Cauchy distributed random variates with scale parameter 1 and location parameter 0. For more details, see the SAS USER'S GUIDE: Basics, Version 5 Edition, p. 265.

The RANTRI generator produces random variates that have a triangular distribution on (0,1) with height parameter h, $0 < h < 1$. The inverse-transform method is used for generation. For more details on transforming outside the unit interval, see the SAS USER'S GUIDE: Basics, Version 5 Edition, p. 269.

Assignments and Calls

For computation efficiency, all assignments to a given generator initialize and use only one stream from RANUNI. For example, consider the program log and listing in figure 4.

Note that in the second DATA STEP in this example, the second seed, SEED2, was ignored.

By using CALL routines, separate streams are produced for each call. In fact identical observations can be produced in the same DATA STEP by the statements:

```
SEED1=12345;  SEED2=12345;

CALL RANUNI(SEED1,X1);

CALL RANUNI(SEED2,X2);
```

Two separate, but identical streams are produced by these calls. Note, however, that a second or subsequent assignment to a different generator will result in initializing a new (RANUNI) stream. This can cause collinearity problems for certain combinations of generators. For example, the RANEXP generator uses the inverse transform method:

$$Y = -\log(U), \text{ for } U \sim U(0,1).$$

Clearly, the pair (Y,U) generated by the above relation will be highly correlated. This problem can be avoided by using different seeds in RANUNI and RANEXP. The RANBIN generator also exhibits strong correlations with RANUNI when both are used in the same DATA STEP with the same seed (see figure 5).

Seed values

Positive integer seed values U_0 are used to intialize a stream. Recall the definition of a prime modulus congruential generator:

$$U_{n+1} = (A*U_n + C) \bmod M, \quad n = 0,1,2,\ldots.$$

U_1 and all subsequent U's are determined by U_0. Clearly $U_0=0$ cannot be used in this definition, but an assignment using a 0 seed in the DATA STEP (or PROC IML) to any generator results in the system clock time being used as the initial seed.

On OS and CMS, the system clock time is units of 10 milliseconds, starting from midnight on any given day, while on the VMS system, the system clock is arbitrary units the order of 10 milliseconds. While these clocks generate reasonable seed values, the resulting stream cannot be reproduced.

Negative seed values also have a special action, in which the system clock time is used as a seed value **each** time an assignment is made. The resulting generator is no longer a prime modulus mulplicative generator. Depending on system conditions, the generator can produce groups of autocorrelated, or even identical numbers. Nothing is known on the properties of this method, and it is not reccomended.

References

Box, G. E. P. and Muller, M. E.,(1958), "A Note on the Generation of Normal Deviates", Annals of Mathematical Statistics XXIX, 610-611.

Coveyou, R. R. and MacPherson, R. D.,(1967), "Fourier Analysis of Uniform Random Generators", J. Assoc. Comput. Mach., 14, pp.100-119.

Fishman, G. S.,(1978),Principles of Discrete Even Simulation,, New York: John Wiley & Sons.

Fishman, G. S., and Moore, L. R.,(1982), "A Statistical Evaluation of Multiplicative Congruential Random Number Generators with Modulus 2^{31}-1",JASA, vol. 77, No. 377.

Fishman, G. S., and Moore, L. R.,(1986), "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus 2^{31}-1",SIAM J. SCI. STAT COMPUT. vol. 7, No. 1.

Hoaglin, D. (1976), "Theoretical Properties of Congruential Random-Number Generators: An empirical View", Memorandum NS-340, Harvard University, Department of Statistics. (Revision in personal correspondence).

Knuth, D. E., (1981), The Art of Computer Programming Vol. 2: Semi-numerical Algorithms, 2nd ed., Addison-Wesley, Reading, MA.

Learmonth, G. P., (1976), "Empirical Tests of Multipliers for the Prime Modulus Random Number", 9-th interface symposium on computer science and statistics., Harvard University and Massachuestts Institute of Technology.

Lehmer, D. H., (1951), 2nd Symposium on Large-Scale Digital Calculating Machinery, Cambridge: Harvard University Press, pp. 141-146.

Marsaglia, G., (1972), The Structure of Linear Congruential Sequences, in Applications of Number Theory to Numerical Analysis, Zaremba, S. K., ed., Academic Press: New York.

Marsaglia, G., (1984), "The Exact-Approximate Method for Generating Random Variables in a Computer", JASA, vol. 79, No. 385.

Mood, A. M., Graybill, F.A., and Boes, D. C., (1974) Introduction to the Theory of Statistics, Third Edition, McGraw-Hill.

Neiderreiter, H., (1976), "The Serial Test for Linear Congruential Psuedo-Random Numbers", Bull. Amer. Math Soc., 84, pp. 273-4.

Stephens, M. A., (1974), "EDF Statistics for Goodness of Fit and Some Comparisions", JASA, vol. 69, No. 347.

```
1          /*************************************************************/
2          /*          S A S   T E S T   L I B R A R Y                  */
3          /*                                                           */
4          /*  NAME: FIGURE 1                                           */
5          /*                                                           */
6          /*  MISC: DEFINITION OF UNIFORMITY: FOR ANY 0 < A < B < 1,   */
7          /*        THE PROPORTION OF UNIFORM RANDOM DEVIATES FALLING  */
8          /*        IN THE INTERVAL (A,B) SHOULD APPROACH B-A AS THE   */
9          /*        SAMPLE SIZE GOES TO INFINITY.                      */
10         /*************************************************************/
11        OPTIONS NODATE DQUOTE NOTEXT82 LS=80;
12        %LET SEED1 = 7831137;
13        %LET SEED2 = 9977311;
14        %LET ALPHA = .2;
15        %LET BETA  = .4;
16        %LET N = 1E6;
17        DATA A;
18         DO KK = 1 TO &N;
20          X1 = RANUNI(&SEED1);
22          CALL RANUNI(&SEED2,X2);
24          TRUE1 = (.2 < X1 < .4);
25          TRUE2 = (.2 < X2 < .4);
26          TTRUE1 + TRUE1;
27          TTRUE2 + TRUE2;
28         END;
29         P1 = TTRUE1/&N;
31         P2 = TTRUE2/&N;
33        RUN;

NOTE: THE VARIABLE X2 IS UNINITIALIZED.
NOTE: DATA SET WORK.A HAS 1 OBSERVATIONS AND 9 VARIABLES. 250 OBS/TRK.
NOTE: THE DATA STATEMENT USED 36.51 SECONDS AND 20DK.

34        PROC PRINT DATA=A SPLIT='#';
35         VAR P1 P2;
36         ID;
37        TITLE 'FIGURE 1: PROPORTION OF THE RANUNI STREAM';
38        TITLE2 "FALLING IN THE INTERVAL (&ALPHA,&BETA) FOR";
39        TITLE3 "A SAMPLE SIZE OF &N FOR ASSIGNMENT AND CALL";
40        LABEL P1= 'PROPORTION#FROM ASSIGN'
41              P2= 'PROPORTION# FROM CALL  ';
42        RUN;
NOTE: THE PROCEDURE PRINT USED 0.10 SECONDS AND 224K
      AND PRINTED PAGE 1.
```

FIGURE 1: PROPORTION OF THE RANUNI STREAM

FALLING IN THE INTERVAL (.2,.4) FOR
A SAMPLE SIZE OF 1E6 FOR ASSIGNMENT AND CALL

OBS	PROPORTION FROM ASSIGN	PROPORTION FROM CALL
1	0.200167	0.200631

```
1          /*************************************************************/
2          /*          S A S   T E S T   L I B R A R Y                  */
3          /*                                                           */
4          /*  NAME: FIGURE 2                                           */
5          /*                                                           */
6          /*  MISC: ALGORITHM FOR A MULTIPLICATIVE GENERATOR WITH      */
7          /*        M = (10**3 - 1) * 999, A = 173, C = 0, X0 = 15     */
8          /*        RETURNS SEQUENCE (Z(N)=X(N)/M, 0<=Z(N)<=1.         */
9          /*                                                           */
10         /*************************************************************/
11        OPTIONS NODATE LS=80;
12        %LET N = 1E3;  /*-- SAMPLE SIZE  ---*/
13        %LET SEED = 15;  /*-- SEED  --*/
14        DATA UNIFORM;
15         RETAIN X &SEED;   /*-- INITIAL X TO SEED VALUE  --*/
17          M  = 999;
18          M1 = 999+1;
19          A  = 173;
20          DO K = 1 TO &N;
22           W = A * X;
23           Y = INT(W/M1);
24           X = W - Y*M1;
25           X = X + Y;
26           Y = INT(X/M1);
27           IF Y ~= 0 THEN X = X - M;
28           Z = X/M;
29           OUTPUT;
30          END;
31        RUN;

NOTE: DATA SET WORK.UNIFORM HAS 1000 OBSERVATIONS AND 8 VARIABLES. 280
OBS/TRK.
NOTE: THE DATA STATEMENT USED 0.15 SECONDS AND 200K.

32        PROC FREQ DATA=UNIFORM;
33         TABLES X;
34         TITLE 'FIGURE 2: FREQUENCY FOR UNIFORM';
35         TITLE2 'DEVIATES FROM MG1 ALGORITHM WITH';
36         TITLE3 'MODULUS=(10**31)-1, A=173, X0=15, C=0';
37         TITLE4 'NOTE CLUMPING OF UNNORMALIZED POINTS';
38        RUN;
NOTE: THE PROCEDURE FREQ USED 0.15 SECONDS AND 420K
      AND PRINTED PAGE 1.
```

FIGURE 2: FREQUENCY FOR UNIFORM

DEVIATES FROM MG1 ALGORITHM WITH
MODULUS=(10**31)-1, A=173, X0=15, C=0
NOTE CLUMPING OF UNNORMALIZED POINTS

X	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
15	55	5.5	55	5.5
24	55	5.5	110	11.0
150	56	5.6	166	16.6
156	55	5.5	221	22.1
240	56	5.6	277	27.7
384	56	5.6	333	33.3
402	56	5.6	389	38.9
438	55	5.5	444	44.4
498	56	5.6	500	50.0
501	55	5.5	555	55.5
561	56	5.6	611	61.1
597	56	5.6	667	66.7
615	55	5.5	722	72.2
759	55	5.5	777	77.7
843	56	5.6	833	83.3
849	55	5.5	888	88.8
975	56	5.6	944	94.4
984	56	5.6	1000	100.0

```
1
 /*************************************************************/
 /*          S A S   T E S T   L I B R A R Y                  */
 /*                                                           */
 /*  NAME: FIGURE 3                                           */
 /*                                                           */
 /*  MISC: KOLMOGOROV-SMIRNOV TEST ON N2 SAMPLES, EACH        */
 /*        HAVING N1 OBSERVATIONS. CRITICAL LEVELS ARE FROM   */
 /*        TABLE 1.0 OF STEPHENS, 1974.                       */
 /*                                                           */
 /*************************************************************/
OPTIONS DQUOTE NONOTES NODATE LS=80;
%LET SEED1=553311;
%LET SEED2=975131;
%LET N1=10000;              /*-- N1 WILL BE THE SIZE OF EACH SAMPLE --*/
%LET N2=500;               /*-- N2 WILL BE THE NUMBER OF SAMPLES --*/
%MACRO KOLMOG;
 %DO JJ = 1 %TO &N2;
   DATA A;
    DO KK = 1 TO &N1;
     X1 = RANNOR(&SEED1);
     CALL RANNOR(&SEED2,X2);
     OUTPUT;
    END;
     CALL SYMPUT('SEED1',INT( 1E5 * ABS(X1) ));
     CALL SYMPUT('SEED2',INT( 1E5 * ABS(X2) ))
     RUN;
    PROC SORT DATA=A(KEEP=X1) OUT=B1; BY X1;
    RUN;
    DATA K1;
     RETAIN DP DN 0;
     LABEL D1 = 'K-S FROM &ASSIGNMENT';
     SET B1 END = EOF;
     DP = MAX( DP, _N_/&N1 - PROBNORM(X1) );
     DN = MAX( DN, PROBNORM(X1) -  (_N_-1)/&N1 );
     D1 = MAX( DP, DN );
     IF EOF THEN OUTPUT;
    RUN;
    PROC SORT DATA=A(KEEP=X2) OUT=B2; BY X2;
    RUN;
    DATA K2;
     RETAIN DP DN 0;
     LABEL D2 = 'K-S FROM # CALL ';
     SET B2 END = EOF;
     DP = MAX( DP, _N_/&N1 - PROBNORM(X2) );
     DN = MAX( DN, PROBNORM(X2) -  (_N_-1)/&N1 );
     D2  = MAX( DP, DN );
     IF EOF THEN OUTPUT;
    RUN;
    DATA K3; MERGE K1 K2;
    RUN;
    %IF &JJ=1 %THEN %DO;
       DATA K; SET K3;
    %END;
    %ELSE %DO;
      DATA K; SET K K3;
      RUN;
    %END;
 %END;

%MEND;


2
%KOLMOG;
DATA IN1.KOLMOG; LENGTH SIG1 SIG2 $ 8;
  SET K;
 ROOTN1 = SQRT(&N1);
 DMOD1 = D1 * ( ROOTN1 + 0.12 + 0.11/ROOTN1 ); /*-- MODIFIED K-S --*/
 DMOD2 = D2 * ( ROOTN1 + 0.12 + 0.11/ROOTN1 );
                /*--   SEE TABLE 1.0 IN STEPHENS, (1974)  --*/
 IF DMOD1 < 1.138 THEN SIG1 = 'NS 0.15 ';
 ELSE IF DMOD1 < 1.224 THEN SIG1 = 'NS 0.10 ';
 ELSE IF DMOD1 < 1.358 THEN SIG1 = 'NS 0.05 ';
 ELSE IF DMOD1 < 1.480 THEN SIG1 ='NS 0.025';
 ELSE IF DMOD1 < 1.628 THEN SIG1 = 'NS 0.01 ';
 ELSE IF DMOD1 >= 1.628 THEN SIG1 = 'S 0.01 ';

 IF DMOD2 < 1.138 THEN SIG2 = 'NS 0.15 ';
 ELSE IF DMOD2 < 1.224 THEN SIG2 = 'NS 0.10 ';
 ELSE IF DMOD2 < 1.358 THEN SIG2 = 'NS 0.05 ';
 ELSE IF DMOD2 < 1.480 THEN SIG2 ='NS 0.025';
 ELSE IF DMOD2 < 1.628 THEN SIG2 = 'NS 0.01 ';
 ELSE IF DMOD2 >= 1.628 THEN SIG2 = 'S 0.01 ';
RUN;
PROC FREQ DATA=IN1.KOLMOG;
 TABLES SIG1 SIG2;
  TITLE 'FIGURE 3';
  TITLE2 'KOLMOGOROV-SMIRNOV D-STATISTIC RESULTS';
  TITLE3 'FOR ASSIGNMENT (SIG1) AND CALL (SIG2)';
  TITLE4 'EACH OF 500 SAMPLES HAD 10,000 OBSERVATIONS';
RUN;
```

FIGURE 3

KOLMOGOROV-SMIRNOV D-STATISTIC RESULTS
FOR ASSIGNMENT (SIG1) AND CALL (SIG2)
EACH OF 500 SAMPLES HAD 10,000 OBSERVATIONS

SIG1	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
NS 0.01	5	1.0	5	1.0
NS 0.025	17	3.4	22	4.4
NS 0.05	21	4.2	43	8.6
NS 0.10	45	9.0	88	17.6
NS 0.15	406	81.2	494	98.8
S 0.01	6	1.2	500	100.0

SIG2	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
NS 0.01	4	0.8	4	0.8
NS 0.025	18	3.6	22	4.4
NS 0.05	20	4.0	42	8.4
NS 0.10	43	8.6	85	17.0
NS 0.15	409	81.8	494	98.8
S 0.01	6	1.2	500	100.0

```
1          /*************************************************************/
2          /*          S A S   T E S T   L I B R A R Y                  */
3          /*                                                           */
4          /*  NAME: FIGURE 4                                           */
5          /*                                                           */
6          /*  MISC: ILLUSTRATION OF MULTIPLE ASSIGNMENTS TO THE SAME   */
7          /*        GENERATOR IN A DATA STEP. THE SECOND SEED IN THE   */
8          /*        SECOND DATA STEP IS IGNORED. NOTE FROM THE         */
9          /*        PRINT OF THE MERGED DATA SET THAT X2 AND X3 ARE    */
10         /*        GIVEN ALTERNATIVE VALUES OF X1                     */
11         /*************************************************************/
12        OPTIONS NODATE LS=80;
13        DATA A1;
14         SEED1 = 12345;
15         DO KK = 1 TO 5;
16          X1 = RANUNI( SEED1 );
17          OUTPUT;
18          LABEL X1 = 'FROM SINGLE#ASSIGNMENT';
19         END;
20         DROP KK;
21        RUN;

NOTE: DATA SET WORK.A1 HAS 5 OBSERVATIONS AND 2 VARIABLES. 953 OBS/TRK.
NOTE: THE DATA STATEMENT USED 0.08 SECONDS AND 196K.

22        DATA A2;
23         SEED1 = 12345;
24         SEED2 = 99999;
25         DO KK = 1 TO 5;
26          X2 = RANUNI( SEED1 );
27          X3 = RANUNI( SEED2 );
28          OUTPUT;
29          LABEL X2 = ' FROM 1ST #ASSIGNMENT';
30          LABEL X3 = ' FROM 2ND #ASSIGNMENT';
31         END;
32         DROP KK;
33        RUN;

NOTE: DATA SET WORK.A2 HAS 5 OBSERVATIONS AND 4 VARIABLES. 529 OBS/TRK.
NOTE: THE DATA STATEMENT USED 0.06 SECONDS AND 128K.

34        DATA B; MERGE A1 A2;
35        RUN;

NOTE: DATA SET WORK.B HAS 5 OBSERVATIONS AND 5 VARIABLES. 433 OBS/TRK.
NOTE: THE DATA STATEMENT USED 0.06 SECONDS AND 176K.

36        PROC PRINT DATA=B SPLIT='#';
37         VAR X1 X2 X3;
38         TITLE 'FIGURE 4';

39         TITLE2 'COMPARISION OF A SINGLE ASSIGNMENT AND';
40         TITLE3 'MULTIPLE ASSIGNEMTS TO RANUNI';
```

FIGURE 4

COMPARISION OF A SINGLE ASSIGNMENT AND
MULTIPLE ASSIGNEMTS TO RANUNI

OBS	FROM SINGLE ASSIGNMENT	FROM 1ST ASSIGNMENT	FROM 2ND ASSIGNMENT
1	0.362924	0.362924	0.745195
2	0.745195	0.831059	0.276277
3	0.831059	0.183824	0.728883
4	0.276277	0.077893	0.734318
5	0.183824	0.707254	0.764080

```
1          /*************************************************************/
2          /*          S A S   T E S T   L I B R A R Y                  */
3          /*                                                           */
4          /*  NAME: FIGURE 5                                           */
5          /*                                                           */
6          /*  MISC: INDUCED CORRELATIONS BETWEEN RANUNI, RANEXP AND    */
7          /*        RANBIN GENERATORS BY USING THE SAME SEED IN CALL   */
8          /*        OR ASSIGNMENT.                                     */
9          /*                                                           */
10         /*************************************************************/
11        OPTIONS NODATE LS=80;
12        DATA A;
13         SEED1 = 12345;
14         N = 50;
15         P = 0.4;
16         DO KK = 1 TO 1E4;
17          X1 = RANUNI( SEED1 );
18          X2 = RANEXP(SEED1 );
19          X3 = RANBIN( SEED1, N, P );
20          OUTPUT;
21         END;
22         DROP KK;
23        RUN;

NOTE: DATA SET WORK.A HAS 10000 OBSERVATIONS AND 6 VARIABLES. 366 OBS/TRK.
NOTE: THE DATA STATEMENT USED 1.76 SECONDS AND 196K.

24        PROC CORR DATA = A  NOSIMPLE;
25         VAR X1 X2 X3;
26         TITLE 'FIGURE 5';
27         TITLE2 'INDUCED CORRELATION BETWEEN RANUNI, RANEXP AND';
28         TITLE3 'RANBIN BY USING SAME SEED';
29        RUN;
NOTE: THE PROCEDURE CORR USED 0.88 SECONDS AND 216K
      AND PRINTED PAGE 1.
```

FIGURE 5

INDUCED CORRELATION BETWEEN RANUNI, RANEXP AND
RANBIN BY USING SAME SEED

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10000

	X1	X2	X3
X1	1.00000 0.0000	-0.86373 0.0001	0.97373 0.0001
X2	-0.86373 0.0001	1.00000 0.0000	-0.88960 0.0001
X3	0.97373 0.0001	-0.88960 0.0001	1.00000 0.0000