8-2007

A Statistical Evaluation of Algorithms for Independently Seeding Pseudo-Random Number Generators of Type Multiplicative Congruential (Lehmer-Class).

Robert Grisham Stewart
East Tennessee State University

A Statistical Evaluation of Algorithms for Independently Seeding Pseudo-Random Number Generators of Type Multiplicative Congruential (Lehmer-Class)

A dissertation

presented to

the faculty of the Department of Educational Leadership and Policy Analysis

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Doctor of Education

---

by

Robert Grisham Stewart

August 2007

Keywords: Random number generation, Congruential generator, Seeding algorithms, Statistical independence, Serial correlation

ABSTRACT

A Statistical Evaluation of Algorithms for

Independently Seeding Pseudo-Random Number Generators of

Type Multiplicative Congruential (Lehmer-Class)

by

Robert Grisham Stewart

To be effective, a linear congruential random number generator (LCG) should produce values that are (a) uniformly distributed on the unit interval (0,1) excluding endpoints and (b) substantially free of serial correlation. It has been found that many statistical methods produce inflated Type I error rates for correlated observations. Theoretically, independently seeding an LCG under the following conditions attenuates serial correlation: (a) simple random sampling of seeds, (b) non-replicate streams, (c) non-overlapping streams, and (d) non-adjoining streams. Accordingly, 4 algorithms (each satisfying at least 1 condition) were developed: (a) zero-leap, (b) fixed-leap, (c) scaled random-leap, and (d) unscaled random-leap. Note that the latter satisfied all 4 independent seeding conditions.

To assess serial correlation, univariate and multivariate simulations were conducted at 3 equally spaced intervals for each algorithm (N=24) and measured using 3 randomness tests: (a) the serial correlation test, (b) the runs up test, and (c) the white noise test. A one-way balanced multivariate analysis of variance (MANOVA) was used to test 4 hypotheses: (a) omnibus, (b) contrast of unscaled vs. others, (c) contrast of scaled vs. others, and (d) contrast of fixed vs. others. The MANOVA assumptions of independence, normality, and homogeneity were satisfied.

In sum, the seeding algorithms did not differ significantly from each other (omnibus hypothesis). For the contrast hypotheses, only the fixed-leap algorithm differed significantly from all other algorithms. Surprisingly, the scaled random-leap offered the least difference among the algorithms (theoretically this algorithm should have produced the second largest difference). Although not fully supported by the research design used in this study, it is thought that the unscaled random-leap algorithm is the best choice for independently seeding the multiplicative congruential random number generator. Accordingly, suggestions for further research are proposed.

CONTENTS

Page

ABSTRACT 2

LIST OF TABLES 8

LIST OF FIGURES 9

Chapter

1. INTRODUCTION 10

Background of the Study 11

Purpose of the Study 12

Null Hypotheses 12

Terms 13

Algorithm 13

Pseudo-Random Number 13

Independently Seeded 14

Simulations vs. Monte Carlo Studies 14

Assumptions 14

Delimitations 14

Value for Increment (b) 14

Value for Modulus (m) 15

Value for Multiplier (a) 15

Limitations 15

Dependent Variables 15

Planned Comparisons 16

Non-Directional Hypotheses 16

Chapter Page

2. REVIEW OF THE LITERATURE 17

The Statistical Independence of Observations 17

Forms of Dependence 17

Effects of Dependence 18

Test Development 18

Effect Assessments 19

Adjustments for Dependence 20

Design-Phase 21

Model-Phase 21

The Statistical Randomness of Numbers 22

Production of Random Numbers 22

Human Processes 22

Physical Processes 23

Arithmetical Processes 24

Evaluation of Random Numbers 25

Distributional Properties 25

Testing Protocols 25

The Lehmer Class of Random Number Generators 26

Value for Increment (b) 27

Value for Modulus (m) 27

Value for Multiplier (a) 28

Effectiveness Studies 28

Seeding Algorithms 28

Theoretical Conditions 29

Published Programs 30

Chapter Page

3. METHODS 31

Algorithm Development 31

Algorithm 1: Zero-Leap 31

Algorithm 2: Fixed-Leap 31

Algorithm 3: Scaled Random-Leap 32

Algorithm 4: Unscaled Random-Leap 32

Algorithm Evaluation 33

Simulation Design 33

Number of Experiments 33

Number of Samples per Experiment 33

Number of Observations per Sample 33

Units of Analysis 34

Dependent Variables 35

Independence Measure 36

Monotonicity Measure 36

Periodicity Measure 38

Data Analysis 38

Independence Assumption 39

Normality Assumption 39

Homogeneity Assumption 40

4. RESULTS 42

Model Assumptions 42

Independence 42

Normality 45

Univariate Analyses 45

Chapter Page

Multivariate Analyses 51

Homogeneity 55

Model Outcomes 57

Hypothesis 1 (Omnibus): All Algorithms Equal 58

Hypothesis 2 (Contrast): Unscaled Algorithm vs. Others 59

Hypothesis 3 (Contrast): Scaled Algorithm vs. Others 60

Hypothesis 4 (Contrast): Fixed Algorithm vs. Others 61

5. CONCLUSIONS 62

Algorithm Development 62

Statistical Comparisons 62

Further Research 64

REFERENCES 66

APPENDICES 74

Appendix A: SAS Program for Producing Independent Seeds 74

Appendix B: SAS Program for Evaluating Seeds 82

Appendix C: SAS Program for Spacing Start Seeds 89

Appendix D: SAS Program for the Intraclass Correlation Coefficient 92

Appendix E: SAS Dataset of Study p-Values 93

VITA 94

LIST OF TABLES

Table Page

1. p-values for Univariate Tests of Normality 46
2. p-values for Univariate Tests of Normality (Transformed)... 46
3. MANOVA for the Hypothesis of No Overall Algorithm Effect 58
4. MANOVA for the Hypothesis of No Overall Unscaled vs. Others Effect 59
5. MANOVA for the Hypothesis of No Overall Scaled vs. Others Effect......................... 60
6. MANOVA for the Hypothesis of No Overall Fixed vs. Others Effect 61

LIST OF FIGURES

Figure Page

1. Residual ACF Plot (Correlogram) for Serial Correlation Test 43

2. Residual ACF Plot (Correlogram) for Runs Up Test 44

3. Residual ACF Plot (Correlogram) for White Noise Test 44

4. Box-plot for Serial Correlation Test... 47

5. Box-plot for Runs Up Test 47

6. Box-plot for White Noise Test.. 48

7. Q-Q Plot for Serial Correlation Test 48

8. Q-Q Plot for Serial Correlation Test (De-Trended) 49

9. Q-Q Plot for Runs Up Test 49

10. Q-Q Plot for Runs Up Test (De-Trended) 50

11. Q-Q Plot for White Noise Test 50

12. Q-Q Plot for White Noise Test (De-Trended) 51

13. Bivariate Ellipse Plot for Serial correlation vs. Runs Up 52

14. Bivariate Ellipse Plot for Serial Correlation vs. White Noise 52

15. Bivariate Ellipse Plot for Runs Up vs. White Noise 53

16. Multivariate Q-Q Plot (Chi)... 53

17. Multivariate Q-Q Plot (Gamma) 54

18. Multivariate P-P Plot (Gamma) 54

19. Multivariate Outlier Plot (Chi)... 55

20. Spread-Versus-Level Plot for Serial Correlation Test 56

21. Spread-Versus-Level Plot for Runs Up Test 56

22. Spread-Versus-Level Plot for White Noise Test.. 57

CHAPTER 1

INTRODUCTION

The field of computational statistics has become indispensable to modern civilization. Indeed, essential contributions include (a) facilitating experimentation through enhancements to random selection (sampling) and random assignment (permuting), (b) simulating (modeling) physical processes, (c) aiding attempts to develop analytic solutions for mathematical expressions, (d) providing surrogates for mathematical expressions that are analytically intractable, and (e) improving the robustness of statistical methods through the enhancement of resampling schemes. One should note that advancements in digital computer technology (viz., increases in memory capacity and computational speed) have played a major role in the field's success.

An indispensable tool of the computational statistician is the pseudo-random number (PRN). Indeed, PRNs serve as basic building blocks in that most probability distributions can be sampled with numbers that are uniform and on the interval (0,1 [excluding endpoints]) (Morgan, 1984). The term "pseudo-random number" emphasizes the deterministic process that results from combining arithmetic formula with digital computers (i.e., the quantity of non-repeating random numbers is finite; moreover, any number can be predicted if the input value is given). Although true random numbers are available (e.g., physical devices can be used as inputs to a digital computer), PRNs have seen the widest use. Accordingly, substantial literature has been produced along two lines: (a) the development of pseudo-random number generators and (b) the assessment of generator performance. To date, the linear congruential family of generators has been the most widely applied and the most thoroughly studied method for producing PRNs.

Background of the Study

To be effective, a linear congruential generator (LCG) should produce values that are (a) uniformly distributed on the unit interval (0,1 [excluding endpoints]) and (b) substantially free of serial correlation (Gentle, 1998). Satisfying the former is not problematic for LCGs (see Fan, Felsovalyi, Sivo, & Keenan, 2002 for empirical results and Knuth, 1981, for a theoretical framework). As to the latter condition, the Principium of Seeding (Mihram, 1972) prescribes two criteria: (a) the values used to start the generator --- the "seeds" --- should be independent (i.e., each seed should be compiled from a validated source of random digits; e.g., the RAND table; RAND Corporation, 1955) and (b) the seeds should not be replicated. However, these criteria constitute a minimum degree of independence (i.e., for large applications a chance exists that at least two sequences could overlap to the extent that only one value is unique; Clark & Woodward, 1992, p. 14).

To further attenuate serial correlation, two additional conditions have been proposed: (a) non-overlapping sequences (Clark & Woodward, 1992; Fan et al., 2002; Gentle, 1998) and (b) non-adjoining sequences, that is, a fixed (or random) number of values are leaped (skipped, discarded) to allow space between sequences (Gentle; Kaplan, 1981). Accordingly, four seeding algorithms can be defined that meet the criteria of non-overlapping and non-replicate sequences: (a) zero-leap, (b) fixed-leap, (c) scaled random-leap, and (d) unscaled random-leap. (Note that the latter three also satisfy the condition of non-adjacent sequences.) Although each algorithm satisfies the conditions for "theoretical independence", the degree of "empirical independence" (i.e., the lack of serial correlation achieved in applications) is not known.

Depending on the application, pseudo-random numbers can produce correlational structures that have disastrous effects. More generally, Stevens (1996) surmised that departures from independence (viz., correlated observations) would increase the number of times the null hypotheses would be falsely rejected (i.e., relationships would be assumed where none existed [cf. Type I errors]). In the case of analysis of variance (ANOVA) of correlated data, Scariano and Davenport (1987) found Type I error rates that were inflated nearly 10 times the level of significance (i.e., tests that specified a .05 alpha level were actually tested at .50). It is plausible that seeding algorithms could attenuate the serial correlation found in pseudo-random numbers. And subsequently improve the correlational properties of applications that use pseudo-random numbers.

Purpose of the Study

The objectives of this study were three-fold: (a) to develop an algorithm that satisfies the four theoretical conditions for producing independent seeds, (b) to statistically compare the performance of four algorithms (zero-leap, fixed-leap, scaled random-leap, and unscaled random-leap), and (c) to identify areas for further research.

Null Hypotheses

Theoretically, at least one of the four seeding algorithms should differ in ability to attenuate serial correlation (Gentle, 1998; Kaplan, 1981; Mihram, 1972). Moreover, algorithms that randomly vary the distance between sequences should decrease serial correlation (Kaplan). Therefore, one would expect that the unscaled random-leap would produce the least serial correlation followed by the scaled random-leap, and lastly the fixed leap. Accordingly, the following relationships are of interest:

1. There will be no difference (at the .01 level) in the output of four seeding algorithms (one-way explanatory variable) on three measures of dependence: (a) serial correlation test, (b) runs up test, (c) white noise test.

2. There will be no difference (at the .01 level) in the output of the unscaled random algorithm and all other algorithms on three measures of dependence.

3. There will be no difference (at the .01 level) in the output of the scaled random algorithm and all other algorithms on three measures of dependence.

4. There will be no difference (at the .01 level) in the output of the fixed interval algorithm and all other algorithms on three measures of dependence.

Terms

Algorithm

According to Skiena (1998, p. 3), "an algorithm is a procedure to accomplish a specific task . . . and is the idea behind any computer program." In designing algorithms, three properties are desirable: correctness, efficiency, and ease of use (Skiena). Moreover, an algorithm "will always produce correct results" while a heuristic will "do a good job without providing any guarantee" for the results (Skiena, p. 4).

Pseudo-Random Number

According to Sedgewick (1983, p. 33), "a *random* number is a precisely defined mathematical concept: every number should be equally likely to occur." Often the term arbitrary number is used instead of random number; however, "one is saying that one doesn't really care what number one gets: almost any number will do" (Sedgewick, p.33). Numbers produced using deterministic methods (e.g., digital computer algorithms) are not random, however, because such numbers "seem to be randomly drawn from some known distribution" the term pseudo-random number is appropriate (Gentle, 1998, p. 1). Note that quasi-random numbers "correspond to samples from a U(0,1) distribution" but are largely dependent (Gentle).

Independently Seeded

According to Mihram (1972, p. 251), an encounter is independently seeded if "each successive seed . . . [is] randomly determined by selecting its value from some published source of random numbers . . . *and* the successive seeds . . . [are] inspected to forbid repetitions in the set."

Simulation vs. Monte Carlo Studies

The terms "simulation" and "Monte Carlo" are often used interchangeably in the literature (for an example see Stevens, 1996, p. 237); however, each term does have a specific sense: simulation – the generation of sample data from a known distribution; Monte Carlo – an evaluation framework for solving mathematical problems or comparing statistical methods (Gentle, 1998).

Assumptions

The three statistical requirements for the multivariate analysis of variance (MANOVA) are the core assumptions for this study. These are (a) multivariate independence, (b) multivariate normality, and (c) homogeneity of the variance-covariance matrices. A detailed analysis of each assumption can be found in Chapter 4.

Delimitations

The core delimitations for this study involve selecting values for the constants in the linear congruential generator equation. A discussion of each constant follows:

Value for Increment (b)

To obtain the full cycle length of a linear congruential generator the increment (b) must be greater than 0. However, "large cycle lengths do not necessarily result in sequences of 'good' pseudo-random numbers" (Morgan, 1984, p. 59). Moreover, letting b=0 removes the addition step thereby increasing computational efficiency. Accordingly, the increment was set at 0 for this study.

Value for Modulus (m)

The modulus is the number of unique values produced by the generator. Several "shuffling" techniques have been developed to extend the modulus (period) of the linear congruential family of generators (see Gentle, 1998). However, these techniques were not applied to the generator used in this study.

Value for Multiplier (a)

Over 500 million multipliers are valid for the linear congruential generator (n.b., the multipliers must be a primitive root of the modulus; Gentle, 1998). Subsequently, 17 multipliers have been subjected to extensive exploratory analysis (see Hoaglin, 1976). Of these, the value 397204094 was found to be optimal and subsequently was the only multiplier used in this study.

Limitations

The primary limitations of this study stem from constraints imposed by the multivariate analysis of variance (MANOVA). A discussion of each constraint follows:

Dependent Variables

The number of dependent variables should be reduced to the most relevant for three reasons: (a) small discrepancies among many variables may mask real discrepancies in a few; (b) generally, test power declines as variables count increases; and (c) the lack of reliability among measures may confound model interpretation (Stevens, 1996, p. 227). Accordingly, three dependent variables (measures) were selected for this study: (a) the serial correlation test, (b) the runs up test, and (c) the Bartlett Komolgorov Smirnoff white noise test (Bartlett, 1966). Two additional measures were applicable: (a) the runs down test and (b) Fisher's white noise test. Consequently, these measures (if added or substituted) could have produced different outcomes for the hypotheses defined in this study.

Planned Comparisons

To reduce the probability of spurious results (i.e., Type I errors), planned comparisons should be reduced to the most essential (Stevens, 1986). More specifically, the number of planned comparisons "should not exceed the number of degrees of freedom [df] for the effect" (Littell, Freund, & Spector, 1991, p. 71). Accordingly, three planned comparisons (contrasts) were allowed for this study (i.e., df = 3). Of the four algorithms, the unscaled random-leap, scaled random-leap, and fixed-leap were compared singly to all other algorithms. The zero-leap algorithm was excluded based on its theoretically low performance potential. Consequently, the zero-leap algorithm (if added or substituted) could have produced different outcomes for the hypotheses defined in this study.

Non-Directional Hypotheses

A balanced one-way multivariate analysis of variance (MANOVA) was used in this study. Consequently, a MANOVA does not allow the testing of directional hypotheses; therefore, it was not possible to determine which algorithm provides the best attenuation for serial correlation. At this point in the investigation, it was decided that testing for an effect (regardless of direction) was more useful that testing for a direction (with the possibility of missing an effect).

CHAPTER 2

REVIEW OF THE LITERATURE

The Statistical Independence of Observations

Fisherian (frequency-based) statistical methods require that variables be (a) independent and (b) identically distrubuted (Freiberger & Grenander, 1971). Throughout the literature this concept is commonly abbreviated as IID (or i.i.d) and is generally synonymous with the terms "statistical independence", "stochastic independence" and, "dependence (antonym sense)". Although complete independence requires no relationship between observations, the following linear definition (relationship) is commonly cited: "Let X_i and X_j be observations sampled from an infinite population with mean mu and variance sigma squared. The observations X_i and X_j are non-independent if $E[(X_i – mu)(X_j – mu)]$ does not equal zero" (Kenny & Judd, 1986, p. 423). Indeed, it is widely acknowledged that the presentation of completely independent observations is seldom encountered in practice – thereby requiring independence to be assumed for statistical modeling (for pedagogical examples of this methodological trade-off see Scariano and Davenport, 1987; for a conceptual framework see Kruskal, 1998).

Forms of Dependence

As to the statistical dependence of observations, Kenny and Judd (1986) identified three forms. The first is group based, which is manifested in several ways: (a) selecting (or assigning) subjects non-randomly (e.g., convenience sampling), (b) exposing subjects to the same artifacts (e.g., students having the same teacher), and (c) allowing subjects to interact during an experiment. The second form of dependence is due to sequence (i.e., observations taken from an experimental unit over time). Two basic patterns are formed by sequence: (a) the first-order moving-average model and (b) the first-order autoregressive model. Two effects can contribute to sequence dependence: (a) cycle effects (e.g., subjects may alter their behavior depending on the day of the week) and (b) subject effects (e.g., subjects may alter behavior depending on intrinsic factors). The final form of dependence is due to space, "that is, observations that are nearer together in space may be more or less similar than those farther away in space" (Kenny & Judd, 1986, p. 425). The pattern formed by space dependence is referred to as the nearest neighbor model (and is analogous to the moving average model for sequences). Moreover, spatial dependence manifests according to the same factors as group dependence (i.e., non-random selection or assignment, central artifacts, and subject interaction).

Effects of Dependence

Substantial literature has been devoted to studying the effects of dependence. Accordingly, contributions can be divided along two lines: (a) Developing tests that detect dependence and (b) assessing the effects of dependence on model outcomes. A discussion of each line follows:

Test Development. Christensen and Bedrick (1997) acknowledged that while dependence can take many forms, testing methodology has mainly focused on serial and spatial effects. Moreover, dependence testing in general has lagged behind other diagnostic areas (e.g., testing for non-normality, heteroscedasticity, and lack of fit). In justifying this latency, Christensen and Bedrick (1997) pointed out that "unlike other assumptions, independence is not a property of the population in question. Independence is a property of the manner in which the population is being sampled. . . . as a result, there is no way to check independence without thinking hard about the method of sampling" (p. 1006). Furthermore, Murray (1972) noted that "testing for independence can be a Herculean task since there are literally an infinite number of possible relationships among the observations" (p. 534).

Despite these challenges, three general measures of dependence are available: (a) the intraclass correlation for group data, (b) the autocorrelation for sequence data, and (c) the adjusted autocorrelation for spatial data (Kenny & Judd, 1986). Other noteworthy test methodologies follow: (a) Murray (1972) developed a composite approach that tests for errors in data against classical assumptions (independence, normality, zero mean, and homogeneous variance); (b) Kiefer (1982) derived the Lagrange multiplier (score statistic) to test the independence hypothesis of multivariate probit models; (c) Hearne, Clark, and Hatch (1983) presented a likelihood ratio test for serial correlation patterns found in univariate repeated-measures designs; (d) Kenny and Judd (1986) offered two equations for the calculation of bias for three cases of dependence (groups, sequence, and space); however, these are limited to the one-way discrete analysis of variance case; (e) Beran (1992) offered a review of tests for long-range dependence (i.e., the Hurst effect); (f) Christensen and Bedrick (1997) proposed replicate lack-of-fit tests based on rational subgroup formation which are suitable for detecting serial, spatial, and other forms of correlation; and (g) Fisher and Switzer (2001) proposed the Chi-plot to assess bivariate dependence structures.

Effect Assessment. To date, literature on the effects of dependence on model outcomes can be divided along two lines: (a) analytic frameworks and (b) empirical studies of error rates. As to the former, Kenny and Judd (1986, pp. 426-427) identified two additional consequences of dependence on the $\underline{F}$-ratio: (a) increased mean square variability and (b) correlated means squares that produce distributions other than $\underline{F}$. Scariano and Davenport (1987) constructed a family of non-identity error correlation matrices useful for studying the relationship of Type I and II errors in the one-way analysis of variance case.

As to empirical studies of error rates, the following contributions are noteworthy: (a) In a pioneering contribution, Gastwirth and Rubin (1971) showed that minimal serial correlation strongly inflated the significance levels of the mean, the sign test, and the Wilcoxon test; (b) Similarly, Moore (1982) and Gleser and Moore (1983, 1985) used a proof approach that showed inflated Type I error rates for chi squared and empiric distribution tests of fit; (c) In the case of univariate repeated measures, Hearne et al. (1983) showed that covariance matrices with serial correlation (simplex pattern) obtained probability levels that exceed the nominal level (.05) by more than threefold; (d) Kenny and Judd (1986) demonstrated considerable biasing effects in the case of "mean squares used to test the effect of some discrete independent variable" (p. 422); (e) As to post-hoc analyses for ANOVA, Pavur (1988) studied the effect of two correlation patterns (i.e., same correlations for any two observations [a] within a group [b] in different groups) on the Type I error rates of four multiple comparison procedures (Fisher's least significant difference [LSD], Tukey's honestly significant difference [HSD], Student-Newman-Keuls significant difference [SNK], and Scheffe`'s significant difference [SSD]) and concluded that "small correlations can be amplified by the number of replications in a one-way layout and these correlations can thus easily inflate the Type I error [with the least affect for the LSD method]" (Pavur, 1988, p. 173); and (f) In the case of meta-analysis of correlation coefficients, Tracz, Elmore, and Pohlmann (1992, p. 879) found that "the assumption of independence was violated when more than one predictor with an intercorrelation exceeding zero were used [in a Monte Carlo simulation]" (see also Tracz & Elmore, 1985, for an earlier exploratory study).

Adjustments for Dependence

Given that (a) the assumption of independence is seldom meet and (b) dependence can have extreme effects on tests of significance, many methods for countering dependence have been proposed. Accordingly, these methods fall into one of two categories: (a) design-phase and (b) model-phase. A discussion of each follows:

Design-Phase. To improve independence, random sampling and permuting of units (subjects) have been widely employed in the design of experiments. Kendall and Babington-Smith (1938) cogently described this relationship as "a random method of selection, applied to the characteristic C of a Universe U, as a method which is independent of C in U" (p. 151). The most common applications are (a) randomly sampling elements from a population and (b) randomly assigning elements to different categories. For either case, two basic actions are required to obtain randomness. First, each element is assigned an identification number which is usually sequential (i.e., 1, 2, 3, …, N; where N is the total number of elements). Second, an identification number is produced so that each number has an equal chance of selection (i.e., a random number). In the case of sampling, the element with an identification-random number match is selected from the population. In the case of assignment, the element with an identification-random number match is repositioned as the first element. Note that an identification number can be used again (selection with replacement) or excluded (selection without replacement). The selection of elements without replacement and with reranking (as in random assignment) is often called a random permutation. Many variations on these basic procedures have been proposed (see Brysbaert, 1991; Manly, 1997; Sedgewick, 1977) .

Model-Phase. To date, model-based adjustments for dependence vary according to data type and modeling procedure. For example, Gleser and Olkin (1994) and Tracz, Newman, and McNeil (1986) proposed methods for the special case of meta-analytic data. More generally, Lahiri (2003) and Manly (1997) discussed resampling adjustments applicable to temporal and spatial data (n.b., Manly also includes grouped data, e.g., ANOVA). Within the context of multivariate analysis of variance, Stevens (1996) suggested four approaches: (a) adjusting the significance level to account for inflated error rates, (b) using the means as the units of analysis, (c) using hierarchical modeling (see Raudenbush & Bryk, 2002), and (d) using the quasi-F or pseudogroup procedures developed by Myers, DiCecco, and Lorch (1981) (and featured in Myers & Well, 2003).

The Statistical Randomness of Numbers

Random numbers are the basic buildings blocks for the field of computational statistics. Accordingly, the literature of random numbers can be divided along two lines: (a) methods for producing random numbers and (b) methods for evaluating random numbers. A discussion of each line follows:

Production of Random Numbers

Historically, two simultaneous challenges have confronted computational statisticians: producing a sufficient quantity of random numbers with acceptable properties (quality). To date, three processes have been used: (a) human, (b) physical, and (c) arithmetical. A discussion of each process follows:

Human Processes. The literature on producing random numbers (digits) using human processes is voluminous --- yet inconclusive (see Nickerson, 2002, for an exhaustive theoretical and empirical review). Indeed, the prevailing hypothesis is that humans are not good producers of a random series (for a recent study see Boland & Hutchinson, 2000). However, the alternative hypothesis is also likely (see Kareev, 1992; Wagenaar, 1971). Wagenarr (1972) pointed out that although many studies had been conducted, comparisons were difficult given that researchers used conflicting constructs for randomness and different experimental conditions.

Physical Processes. An effective method for producing random numbers involves manipulating physical devices. In the simplest case, one can use dice to obtain random numbers (e.g., see Hamaker, 1949, for a dice throwing technique to obtain random sampling numbers). However, more elaborate methods exist, such as the randomizing machine described by Kendall and Babington-Smith (1938):

> The machine consists of a disc divided into ten equal sections, on which the digits 0 to 9 are inscribed. The disc rotates rapidly at a speed which can, if necessary, be made constant to a high degree of approximation by means of a tuning-fork. The experiment is conducted in a dark room, and the disc is illuminated from time to time by an electric spark or by a flash of a neon lamp, which is of such short duration that the disc appears to be at rest. At each flash a number is chosen from the apparently stationary disc by means of a pointer fixed in space. In the actual experiment, the disc was rotated by an electric motor at about 250 revolutions per minute. It was illuminated by a neon lamp in parallel with a condenser in an independent electric circuit which was broken by means of a key. Owing to experimental conditions, the time between the making of the circuit and the passing of the flash varied, but to add an extra element of randomness the key was tapped irregularly by the experimenter. Flashes occurred, on the average, about once in three or four seconds. (p. 157)

The RAND Corporation's electronic roulette wheel is another complicated approach:

> Briefly, a random frequency pulse source, providing on the average about 100,000 pulses per second, was gated about once per second by a constant frequency pulse. Pulse standardization circuits passed the pulses through a 5-place binary counter. In principle the machine was a 32-place roulette wheel which made, on the average, about 3000 revolutions per trial and produced one number per second. A binary-to-decimal converter was used which converted 20 of the 32 numbers (the other twelve were discarded) and retained only the final digit of two-digit numbers; this final digit was fed into an IBM punch to produce finally a punched card table of random digits. This table was subjected to fairly exhaustive tests and it was found that it still contained small but statistically significant biases. The table was regarded as reasonably satisfactory because the deviations from expectations in the various tests were all very small--the largest being less than 2 per cent--and no further effort was made to generate better numbers with the machine. However, the table was transformed by adding pairs of digits modulo 10 in order to improve the distribution of the digits. (RAND Corporation, 1955, p. 1)

More recently, methods have been proposed that provide input to a digital computer. For example, Millenson and Sullivan (1969) proposed using the leakage from a reversed biased diode, while Schmidt (1977) proposed using a Geiger counter. Teichroew (1965) noted that values from a physical process must be stored if an application is to be replicated. Moreover, because physical processes are indeterministic, values must be exhaustively tested prior to application (Teichroew). Ultimately, these factors have hampered widespread use of physical processes.

Arithmetical Processes. The most prevalent process for producing random numbers results from coupling arithmetical formulas with modern digital computers. Indeed, with the advent of digital computers, it is possible to produce many samples from a variety of statistical distributions. In using random numbers (e.g., Monte Carlo studies), Mooney (1997) advised that although the number of experiments should be small (to reduce interdependence), the number of samples per experiment should be as high as possible. Consequently, the quantity of pseudo-random numbers required for modern applications easily exceeds those provided in print tables or obtainable from local physical devices (Mooney).

To date, three computer-based arithmetical processes have been developed: (a) linear congruential generators (Gentle, 1998), (b) feedback shift registers (Gentle), and (c) combination (compound) generators (see Collings, 1987; L'Ecuyer, 1988; MacLaren & Marsaglia, 1965; Wichmann & Hill, 1982, 1984). Of these, the linear congruential have been the most successful. Indeed, Morgan (1984, p. 64), pointed out that "the advantage of congruential generators is that they can be shown to possess certain desirable features and to give guaranteed cycle lengths." Specific congruential generators are (a) linear, (b) multiple recursive, (c) lagged Fibonacci, (d) add-with-carry, (e) subtract-with-borrow, (f) multiply-with-carry, (g) inverse, and (h) matrix.

Evaluation of Random Numbers

The computational efficiency afforded by pseudo-random numbers is offset by the weaker distributional properties of said numbers (note that true random numbers have better properties but are more difficult to apply). Consequently, suggestions for evaluating pseudo-random numbers are prominent in the literature that can be divided along two lines: (a) distributional properties and (b) test protocols. A discussion of each line follows:

Distributional Properties. Useful pseudo-random numbers will satisfy the distributional properties of uniformity and randomness. The following definition of uniformity is commonly cited: "Let α and β be real numbers, $0 \leq \alpha < \beta \leq 1$, and Ui be the random sequence $\{U_i: i=1,2,\ldots\}$, where $0 \leq U_i \leq 1$. If the proportion of U_i's satisfying alpha $\leq U_i < \beta - \alpha$ (as the number of deviates generated approaches infinity), the sequence U_i is said to be uniform" (Clark & Woodward, 1992, p. 13). Subsequently, the concept of randomness involves a more complicated expression (Clark & Woodward, 1992). For U_i, "if the $Pr(\alpha_1 \leq U_i < \beta_1$ and $\alpha_2 \leq U_{i+1} < \beta_2) = (\beta_1 - \alpha_1)(\beta_2 - \alpha_2)$ for any four numbers α_1, β_1, α_2, and β_2, where $0 \leq \alpha_j \leq \beta_j \leq 1$ and $1 \leq j \leq k$. A sequence U_i is ∞-distributed if it is k-distributed for all k=1,2,… A sequence that is ∞-distributed can be considered random (Knuth, 1981)" (as cited in Clark & Woodward, 1992, p. 13).

Testing Protocols. Generally, two forms of random number evaluation are proposed: (a) theoretical (global) tests of the generator and (b) empirical (local) tests of generator output. Based on an evaluation of generators using a known analytical solution, Ferrenberg, Landau, and Wong (1992) developed the following protocol: (a) use generators with the best theoretical test results and (b) conduct empirical tests of output for each new application, ignoring how well the generator has performed in prior applications. (Note that Manly, 1997, acknowledged that the latter is likely skipped by practitioners). For applications that involve normal distributions, Bang, Schumacker, and Schlieve (1998) developed the following protocol: (a) test the number distribution for normality, (b) use a large sample size (i.e., $n > 10{,}000$), (c) test the starting seed values for normality, (d) consider the extent that numbers will be allowed to repeat (if at all), (e) examine the serial correlation of number sequences, and (f) test the number distribution for uniformity.

As to the testing of random numbers, Fan et al. (2002) advised using as many tests as possible. However, an infinite number of tests for the randomness of sample values are possible (Morgan, 1984). Consequently, a discussion of all testing methods is beyond the scope of this study (but see Fan et al., chap. 3; Gentle, 1998, chap. 6; Gruenberger & Jaffray, 1965; Kennedy & Gentle, 1980, chap. 6; Knuth, 1981, chap. 3; Mihram, 1972, chap. 2; Morgan, 1984, chap. 6; Rubinstein, 1981, chap. 2; Sedgewick, 1983, pp. 40-42; Strube, 1983; Tuckwell, 1988, pp. 90-97). Note that Chapter 3 – Methods contains a detailed discussion of the three empirical tests used in this study.

The Lehmer Class of Random Number Generators

In 1948, Lehmer (1951), proposed the most prevalent method for producing pseudo-random numbers with digital computers: the linear congruential generator (LCG). The basic form of the linear congruential family of generators is the following recursive equation (which follows the notation used by Morgan, 1984): $x_i + 1 = (ax_i + b) \bmod m$; where a, b, and m are fixed integer constants (called the "multiplier", the "increment", and the "modulus" respectively); where x and i are non-negative integers (called the "sequence" and "index", respectively) with $i \geq 0$. Note that some software programs allow a negative value for x to indicate that the computer's clock value will be the initial seed (see SAS Institute, Inc, 1990a, p. 592). The sequence values (x_i) are scaled on the unit interval (0,1; excluding endpoints) as follows: $u_i = x_i/m$ (Gentle, 1998, p. 6). The first value for x (called the "seed) must be supplied externally and is often represented as x0 since it is not produced by the generator (this value can be included in the output if desired; see Fan et al., 2002, pp. 38-39). The values for a, b, and m are optimized to satisfy four properties: efficiency, periodicity, uniformity, and randomness (Clark & Woodward, 1992). A discussion of each constant follows:

Value for Increment (b)

When the increment value is equal to 1, the term "additive" is used to classify the linear congruential generator. When the addition operation is eliminated (i.e., increment=b=0) an increase in computational efficiency is gained (Gentle, 1998). Subsequently, this generator is classified as "multiplicative congruential". Note that the term "mixed congruential" is also applied to generators where b is not equal to 0 (Gentle).

Value for Modulus (m)

"The period of the random number generator is the number of values produced by the generator before it begins repeating the sequence" (Clark & Woodward, p. 13). In an ideal world, all generator periods would be infinite; however, the period of the linear congruential generator cannot exceed the modulus (Clark & Woodward, 1992). Moreover, x_i cannot = 0 for the multiplicative generator (i.e., 0 will be returned for every successive x_i), hence the modulus is reduced by 1. The maximal period is attained "if (and only if) *m* is a prime and the multiplier, *a*, is a primitive root modulo *m* . . . [which] is a number such that the smallest positive k satisfying a_k=1 mod m is m-1" (Gentle, 1998, p. 7). The most prevalent modulus value is likely the Mersenne prime 2,147,483,647 (programmed as 2^{31}-1) (Gentle). Note that scaling x_i on the unit interval (0,1 excluding endpoints) further reduces the modulus by 1 to 2,147,483,646 (or 2^{31}-2) (Fan et al., 2002, p. 27).

Value for Multiplier (a)

To attain the maximal period, a multiplier must be a primitive root of the modulus (Gentle, 1998, p. 7). Indeed, more than 500 million multipliers satisfy this criterion (Clark & Woodward, 1992, p. 13). Accordingly, Fishman and Moore (1982) evaluated 17 multipliers using statistical tests for randomness and uniformity (from 1 to 3 dimensions). Most of these multipliers were identified by an exploratory study conducted by Hoaglin (1976). The value 397204094 was found to be an optimal multiplier (even after the approximation formulas used by Hoaglin were replaced with exact methods; Clark & Woodward, p. 13). (Note that the SAS System uses this multiplier to produce uniform random variates; SAS Institute, 1990a, p. 592.)

Effectiveness Studies

The linear congruential method is the most widely researched of the random number generators. Although a complete review of is not possible here, the following studies are noteworthy: (a) Coveyou (1959) proposed multiplier and increment constants aimed at reducing the serial correlation of number sequences; (b) Peach (1961) showed that harmonics (sub-periods) significantly reduce the variance of long number sequences vs. the theoretical values; (c) DeMatteis and Pagnutti (1988, p.595) found strong autocorrelations existing between parallel generator runs and concluded that "only small fractions of the sequences can be safely used"; and (d) Eichenauer-Herrmann and Grothe (1989) replicated the findings of DeMatteis and Pagnutti when prime moduli were specified for multiplicative congruential generators.

Seeding Algorithms

The literature regarding independently seeding random number generators of type multiplicative congruential can be divided along two lines: (a) theoretical conditions and (b) published programs. A discussion of each line follows:

Theoretical Conditions. With regard to obtaining independent samples from a pseudo-random number generator, Mihram (1972) offered several points: even if parameters are altered (i.e., independent experiments occur), sequence seeds should be independently seeded that is, (a) each seed should be compiled from a published source of random digits (e.g., the RAND table; RAND Corporation, 1955) and (b) any repeating seed values must be discarded. These points are cogently summarized as the Principium of Seeding: In any sequence of n encounters with a verified stochastic simulation model, the n successive random number seeds shall be selected randomly [from among the admissible set of seeds p. 252] and independently, though repetitions shall be forbidden (Mihram, p. 251).

The condition of "non-repeating seeds" imposed by Mihram (1972) to achieve sequence independence, only allows that two sequences of the same length (number of values) will not have identical values. Indeed, when combined with the condition of "random seed selection" a chance exists that at least two sequences could overlap to the extent that only 1 value between sequences is unrepeated (constituting a minimum degree of independence). Clark and Woodward (1992) concluded that for a moderately sized application such a chance is small given the period (i.e., over 2 billion numbers for a multiplicative congruential generator) . However, for applications that involve a large number of long sequences it is recommended that the condition of "non-overlapping sequences" be met (Clark & Woodward, p. 14; Fan et al., 2002, pp. 38-39; Gentle, 1998, pp. 14, 36, 169). The general condition of "non-overlapping sequences" can be further refined based on the interval length (i.e., the number of values skipped [leaped, discarded]) allowed between sequences. In general, three intervals are recognized: zero, fixed, and random. Gentle advised using either fixed or random intervals. Kaplan (1981) concluded that random intervals should provide the best results, based on his exploratory analysis of the statistical independence of seeding linear congruential generators (additive, multiplicative, and mixed).

Published Programs. Several computer programs based on independent seeding principles were found in the literature. Kelly (2000) provided a SAS program that adds a small perturbation (64) to each consecutive seed obtained from multiplicative congruential generator (cf. mixed congruential generator)– thereby producing random and unrepeated (independent) seeds (n.b., an alternative approach would be to use the consecutive seeds as sampling (serial) numbers for the population of seeds). Clark and Woodward (1992, p. 18) and Fan et al. (2002, pp. 38-39) developed SAS computer programs that produce non-overlapping sequences. The SAS programs by Clark and Woodward (1992, p. 18) and Fan et al. will support the "zero-leap" and "fixed-leap" conditions (albeit in straightforward applications). These programs more closely match systematic random sampling (Scheaffer, Mendenhall, & Ott, 1996, chap. 7) rather than the simple random sampling without replacement imposed by Mihram's Principium of Seeding. Examples of computer programs for the "random-leap" condition were not found (see Chapter 3 for description of a program that uses approximate simple random sampling and ensures "non-overlapping sequences").

CHAPTER 3

METHODS

In this chapter, two issues are addressed: (a) the development of effective seeding algorithms and (b) the evaluation of algorithm performance. A discussion of each follows:

Algorithm Development

Four algorithms that produce seeds for multiplicative congruential generators were developed (or refined) for this study. These are (a) zero-leap, (b) fixed-leap, (c) scaled random-leap, and (d) unscaled random-leap. Features common to each algorithm are: (a) a single computer session is used to produce seeds for an entire simulation study, (b) interval length and sample size are separate parameters, and (c) the generator's period cannot be exceeded (which prevents production of overlapping sequences). A discussion of each algorithm follows:

Algorithm 1: Zero Leap

The zero-leap algorithm is the simplest of the four algorithms and is the most economical with regard to value output (i.e., all values in the period can be used). However, it has the weakest sampling properties of the four algorithms (i.e., the starting seed is the only random component since successive values are in serial order). Moreover, many values could have no chance of inclusion if far enough away from the starting seed.

Algorithm 2: Fixed Leap

The fixed-leap algorithm provides better theoretical sampling properties than the zero-leap. The number of values leaped is computed by subtracting the period from the total number of values required then dividing by the number of seeds. In this way, the entire period is traversed by the algorithm (allowing all values a chance to be included). However, this algorithm is closer to systematic sampling than the simple random sampling imposed by Mihram's (1972)

Principium of Seeding. Depending on the circumstances, the efficiency of systematic sampling can differ from designs that have a direct random component (Madow, 1946). Indeed, Madow advised that "the chief danger in applying a systematic design occurs when the data have a periodic formation, and the sampling interval chosen is equal to the period of the data" (p. 213).

Algorithm 3: Scaled Random-Leap

An application of the random-leap algorithm was not found in the literature. Accordingly, the method for computing the fixed-leap distance was applied. The maximum number of skips between seeds is used to scale the last uniform random number in a sample (say from 1 to 100) that then becomes the number of values leaped after that sample. Although this algorithm offers better random sampling properties than the zero or fixed-leap algorithms, it has the potential to exclude more values (reducing the number of values available for an application).

Algorithm 4: Unscaled Random-Leap

Of the four algorithms considered, the unscaled random-leap provides the closest match to Mihram's (1972) Principium of Seeding (i.e., using simple random sampling to obtain seeds) while satisfying the conditions of non-replicated, non-overlapping, and non-adjoining seeds. Specifically, a block of seeds produced by the generator become sampling numbers for the entire population of seeds. To prevent the chance that sampling numbers will produce overlapping seeds, the distance between each sampling number is computed and checked against the sample size parameter. If the sequences overlap, then that sampling number is rejected. (Note that extra sampling numbers are generated to allow for any rejects.)

Algorithm Evaluation

In this section, four issues regarding evaluation of the seeding algorithms are considered. These are (a) simulation design, (b) units of analysis, (c) dependent variables, and (d) data analysis. A discussion of each follows:

Simulation Design

A simulation study approach was used to evaluate the four seeding algorithms. The primary design constraint for a simulation is the total number of unique uniform values required. Three factors affect this value: (a) the number of experiments in the simulation, (b) the number of samples per experiment, and (c) the number of observations per sample. A discussion of each follows:

Number of Experiments. In practice, the number of experiments for a given simulation can vary greatly (Mooney, 1997). Mooney strongly admonished delimiting the number of experiments in a simulation to those offering the most variation. Moreover, the number of experiments must be balanced against computational limitations (e.g., the amount of time allotted to run a simulation, the length of the random number generator period).

Number of Samples per Experiment. As to the number of samples (trials), Mooney (1997, p. 58), advised that "there are no general theoretical guidelines for the number of trials required for experimental results to be valid." Generally, from 1-100 trials are suggested for exploratory work, while as many trials as possible are suggested for confirmatory work (e.g., 1,000-25,000: Mooney, p. 58).

Number of Observations per Sample. In determining the number of sample observations, two factors must be considered: (a) the number of variables in the sample and (b) the type of probability distribution(s) each variable should approximate. The former assumes that one would conduct either a univariate simulation or multivariate simulation (but not both with regard to the same problem). For the latter, values from probability distributions can be obtained by transforming one (or more) values from a uniform distribution (see Gentle 1998; Johnson, 1987; Mooney, 1997; Morgan, 1984). Note that competing methods for transforming uniform values exist as do alternatives to uniform transformation methods (see Gentle; Morgan). For example, one method for generating four (multivariate) correlated non-normal values (see Headrick & Sawilowsky, 1999a, 1999b) requires 6 standard normal values with each of these requiring 2 uniform values (i.e., 12 uniform values are required to generate 4 correlated non-normal values). However, other methods require fewer uniform values (e.g., Fleishman's, 1978, equation requires only 2 uniform values per multivariate value).

Accordingly, this simulation study was composed of two designs: (a) a univariate design of 18 experiments, with 10,000 samples per experiment, with 6 experiments of sample size 10, 50, and 100 respectively and (b) a multivariate design with 8 experiments, with 10,000 samples per experiment with 4 experiments of sample size 10 and 50 respectively [each with multiples of 4 and 8 to account for multiple variables and or distributional transformations].

Units of Analysis

The stream of 2^{31}-2 of unrepeated random values offers a source of variation. That is, the effectiveness among the algorithms will likely vary depending on where they are initiated because the quality of randomness varies within the stream. This variation was used to increase the units of analysis by initiating each of the two simulations at three different equally spaced points in the random number stream yielding a total of 24 observations for this study (2 simulations x 4 algorithms x 3 starting points). Although arbitrary, three intervals should maximize coverage of variation within the stream while maintaining independence among the units of analysis (Mooney, 1997, pp. 62-63). Equally spaced starting seeds were obtained for each simulation (design 1: 684543030, 1400370912, 2116198794; design 2: 123535106, 839362988, 1555190870) using the following procedure (Mihram, 1972, p. 251):

1. Select a 10-digit number from the RAND table using the manual procedure (RAND Corporation, 1955).
2. If the 10-digit value is equal to 0, greater than 8589934584 (i.e., the largest equiprobable seed value) or already listed, then discard and return to step 1.
3. If the 10-digit value is greater than 2147483646, then subtract 2147483646 until the value is less than or equal to 2147483646 and list as the first seed.
4. To obtain the second seed, add p (where p = 2147483646 divided by the number of intervals [3]) to the first seed and list. To obtain the third seed, add p to the second seed and list.
5. Repeat steps 1-4 for each experiment (for the SAS program see Appendix C).
6. Pass each set seed to the "check for overlapping sequences" routine of the produce_seeds SAS program (Appendix A).
7. If a seed is rejected, repeat steps 1-4 to obtain a replacement set and repeat step 6.

Dependent Variables

An infinite number of tests for the randomness of sample values are possible (Morgan, 1984). Fan et al. (2002) suggested using as many tests as possible to determine randomness. However, two constraints guided the selection of dependent variables for this study: (a) the need for tests that provide a meaningful single measure (e.g., p-value) based on an entire stream of numbers (say 1,800,000) rather than substreams (samples of size 10) and (b) the need for the number of dependent variables in a multivariate k-populations design to be as small as theoretically possible (Stevens 1996, p. 227). Accordingly, three fundamental properties of randomness were the dependent variables for this study: (a) independence, (b) monotonicity, and (c) periodicity. A discussion of the measure for each property follows:

Independence Measure. To measure independence, the serial correlation test (as defined by Anderson, 1942) was applied to each simulation (for the SAS program see Fan et al., 2002, pp. 30-34; for examples see Kennedy & Gentle, 1980, pp. 170-171 and Knuth, 1981, pp. 70-71). Note that the serial test (Good, 1957; Knuth, 1981, p. 60; Morgan, 1984, p. 140-142) is not an analogue of the serial correlation test. Applications of the serial correlation test have been based on Pearson's product-moment correlation coefficient using the standard score formula (Hinkle, Wiersma, & Jurs, 1994, chap. 5). Although any cyclically shifted sequence can be used to lag the test (Knuth, 1981, p. 71), the immediate successor sequence is preferred because of its simplicity. Accordingly, correlations were computed between the ith and (i+j)th random number where j=1,2, . . . n and n = the highest value for lagging (in this case 100). To obtain a probability value (p-value) for each simulation, the arithmetic mean of the lagged p-values was computed. A small p-value indicates lack of independence (i.e., rejection of the two-sided null hypothesis that the correlation is equal to 0).

Monotonicity Measure. To evaluate monotonicity, the runs up test was applied to each simulation (for discussions see Gentle, 1998; Kennedy & Gentle, 1980; Knuth, 1981; Morgan, 1984). Note that different runs test can be constructed thru various combinations of the following: (a) 3 problem forms (i.e., constant probability events with either fixed or unfixed sample sizes [Bradley, 1968, chap. 11] and non-constant probability events with unfixed sample sizes [Bradley, chap. 12]); (b) 3 event forms (i.e., runs up [i.e., monotonic increasing sequences], runs down [i.e., monotonic decreasing sequences], or runs up and down); (c) 3 metric forms (i.e., total number of runs, length of runs, or length of longest run); (d) 3 alternative hypotheses (one-sided [i.e., either too few runs or too many runs] or two-sided [i.e., monotonic pattern unknown]); and (e) distributional form (e.g., binomial, chi-square, normal). More specifically, a modified version of the non-constant probability, runs up and down, length of runs, two-sided chi-square test (Wallis & Moore, 1941) was applied (for the SAS program see Appendix B). Indeed, a constant probability test is not applicable since the process for randomness is of interest (Bradley, p. 271). Moreover, the chi-square length of runs test is more sensitive to departures from randomness than the total number of runs test based on the normal distribution (Kennedy & Gentle; Wallis & Moore). Finally, a two-tailed test is appropriate since a pattern of too many (or to few) runs is not known a priori.

The run (phase) lengths of a sequence are not completely independent of each other (Wallis & Moore, 1941). Accordingly, Wallis and Moore (p. 403) acknowledged that "very large and very small values . . . [for their chi-square test statistic] are a little more likely" while the mean and variance exceed those of the ordinary application. Indeed, Knuth (1981, p. 65), Levene and Wolfowitz (1944, p. 66) and Morgan (1984, p. 144) disapprove of the usual chi-square test (given the negative correlation that exists among the various run lengths; Kennedy & Gentle, 1980, p. 172). Subsequently, Levene and Wolfowitz proposed adjusting the expected values for the runs up and down chi-square test using a variance-covariance matrix (for a cogent example see Kennedy & Gentle; for a version based on the number of runs up see Knuth; Morgan). For reliable asymptotic results, Levene and Wolfowitz recommend a minimum sample size of 100 while Knuth advises 4000.

Because the samples values are scaled on the unit interval (0,1 [excluding endpoints]; Gentle, 1998, p. 6) and are non-repeating for 2^{31}-2 values, the following test assumptions (see

Bratley, 1992, p. 278) are satisfied: (a) each observation position is unique, and (b) each observation value is unique (i.e., no ties between observations for either position or value). The probability value for each sample will be used as the measure of within sample independence. A small p-value indicates lack of independence (i.e., rejection of the two-sided null hypothesis that expected number of runs for each length corresponds to the observed "on the probability sense of order square root n"; Levene & Wolfowitz, 1944, p. 68).

Periodicity Measure. The Bartlett Komolgorov Smirnoff white noise test (as defined by Bartlett, 1966) was used to evaluate the periodicity of each simulation (for the SAS program see SAS Institute Inc., 1986). Note that Fisher's Kappa (Wei, 1990) also provides a test for the white noise null hypothesis (by checking for a single sinusoidal component formed in white noise) (for the SAS program see Woodfiel, 1991). The Bartlett test is preferred because it checks for a more complex departure from white noise (viz., by accumulating "departures from the white noise hypothesis over all frequencies" [SAS Institute Inc., p. 217]).

Data Analysis

Overall, this study employed a "second order" analysis (i.e., analyzing the output from statistical tests [e.g., p-values] with other statistical tests), which is an accepted practice for simulation studies (see Gentle, 1998, p. 158; Morgan, 1984, pp. 145-148). Accordingly, the p-values obtained from the three dependent measures were modeled using a balanced one-way multivariate analysis of variance (MANOVA). Stevens' (1996) framework was used to evaluate the statistical assumptions required for this model (i.e., multivariate independence, normality, and homogeneity). Moreover, Keselman's (2005) protocol for accessing and improving multivariate normality was integrated with Steven's framework. A discussion of each assumption follows:

Independence Assumption. Of the three MANOVA assumptions, independent-identically distributed observations (IID) is the most critical for two reasons: (a) non-independence will likely inflate the Type I error rates for linear models and (b) IID is a requirement for unbiased testing of normality and homogeneity assumptions (Stevens, 1996). Although the data were sampled in a manner designed to reduce non-independence (i.e., the seeding algorithms used random sampling schemes [albeit to varying degrees]), substantial serial correlation is known to exist in the population of values. Accordingly, the intraclass correlation coefficient (ICC) for groups (as defined by Kenny & Judd, 1986, p.204) was computed for each dependent variable (for the SAS program see Appendix D). Subsequently, the sample autocorrelation function (ACF) for residuals (as defined by Kundu & Basu, 2004, p. 229) was plotted for each dependent variable (for adaptable SAS programs see Twagilimana, 2005, p. 6 and Piegorsch & Bailer, 2005, p. 238). Note that the minimum number of observations for an ACF should be greater than 50 (Box & Jenkins, 1970). Consequently, ACFs were not plotted for each independent variable group as the sample size was deemed insufficient (i.e., n=6).

Normality Assumption. In testing the assumption of multivariate normality (MVN), Johnson and Wichern (1998) advocated testing univariate normality (UVN) first because it is a requirement (but not a guarantee) for MVN. In a comparative study of UVN tests, Seier (2002, p. 1) found that the Shapiro-Wilk test was the best overall choice ("but was not uniformly most powerful against all alternatives considered"). Accordingly, the Shapiro-Wilk test was computed for each dependent variable (for the SAS program see D'Agostino, Belanger, & D'Agostino, 1990). Subsequently, univariate box-plots and quantile-quantile (Q-Q) plots were prepared for each dependent variable (for the SAS programs see Friendly, 1991). Note that Stevens (1996) computed the Shaprio-Wilk test for each group of the independent variable where the smallest sample size was 8, eventhough D'Agostino et al. mandated a minimum sample size of 9 for UVN hypothesis testing. Consequently, the Shapiro-Wilk test was not computed for each independent variable group as the sample size was deemed insufficient (i.e., n=6).

For the case of MVN, Looney (1995, p. 69) advised using multiple tests given that there are a variety of departures from multivariate normality, therefore, "one test . . . [cannot] be expected to be uniformly most powerful against all [others]." In choosing MVN tests, Looney pointed out three difficulties: (a) Over 50 MVN tests have been proposed (see Mecklin & Mundfrom, 2004, for a bibliography and critical review); (b) studies of relative power are few; and (c) many tests require special computations for significance values. Consequently, Looney suggested that tests be delimited to those with reliable (published) software for calculating test statistics and p-values. Accordingly, three tests met this criterion: (a) Mardia's (1970) tests for skewness and kurtosis and (b) the cube root test developed by Mudholkar McDermott and Srivastava (1992) (for the SAS programs see Khattree & Naik, 1999). As to multivariate graphical analyses, the following plots were prepared: (a) bivariate ellipse plots, (b) quantile-quantile (Q-Q) plots for the chi and gamma distributions, (c) a probability-probability (P-P) plot, and (d) an outlier plot (for the SAS programs see Friendly, 1991 [a]; Khattree & Naik,[b, c, d]).

Homogeneity Assumption. Of the three MANOVA assumptions, homogeneity of the covariance matrices is the most restrictive (i.e., the most likely not to be satisfied in practice; Stevens, 1996). Accordingly, this assumption was tested using the Bartlett likelihood ratio test (as defined by Morrison, 1976, pp. 268-273 and programmed by the SAS Institute, Inc., 1990b, pp. 690-691). Subsequently, spread-versus-level (SVL) values (as defined by Hoaglin, 1991) were plotted for each dependent variable (for an adaptable SAS program see Friendly, 1991).

Although alternative formulations for the SVL plot exist (for definitions see Emerson & Soto, 1982; for commentary see Myers & Well, 2003; for SAS programs see Friendly), Hoaglin's definition is the most appropriate for the exploratory analysis of variance approach used in this study (Hoaglin).

CHAPTER 4

RESULTS

A balanced one-way multivariate analysis of variance (MANOVA) was performed on three dependent variables: (a) the serial correlation test, (b) the runs up test, and (c) the white noise test (Bartlett Komolgorov Smirnov). Results from this analysis are summarized in two sections: (a) model assumptions and (b) model outcomes. A discussion of each section follows:

Model Assumptions

Independence

The intraclass correlation coefficient (ICC) for strength of association was computed for each dependent variable: serial correlation test ($\underline{r}$=0.14341), runs up test ($\underline{r}$=0.11511), white noise test ($\underline{r}$=0.44903). "A positive value means that scores within groups are on average more similar than scores between groups" (Kenny & Judd, 1986, p. 424). Given that each dependent variable has a positive correlation coefficient, it is reasonable to assume that dependence would likely cause the actual alpha level to be several times greater than the assumed alpha. Consequently, four options for dealing with correlated observations are commonly cited: (a) deleting observations from each independent variable group, (b) tightening the significance level, (c) using the means as the units of analysis, and (d) using alternative models (e.g., hierarchical modeling [Raudenbush & Bryk, 2002]; quasi-$\underline{F}$ and pseudogroup procedures [Myers, DiCecco, & Lorch, 1981; Myers & Well, 2003]). Stevens (1996, p. 241) suggested testing hypotheses "at a more stringent level of significance" (viz., .01) if the assumed error rate is expected to be inflated by a factor of at least 10. Moreover, Scariano and Davenport (1987) demonstrated that for a group size of 3 with 10 observations, alpha levels were .2227 for an ICC of .10 and .5379 for an ICC of .30. Accordingly, MANOVA hypotheses and contrasts were tested at the .01 level of significance (rather than .05) to adjust for the positive intraclass correlation of the three dependent variables.

The sample autocorrelation function (ACF) was plotted for each dependent variable (see Figures 1-3). “Large values [of residuals], especially at the beginning of the plot [i.e., small lags], signal presence of autocorrelation [i.e., the residuals still contain information that must be explained]” (Wiersma, 2004, p. 145). Non-significant autocorrelation coefficients were obtained for each dependent variable. Moreover, the pattern of positive and negative correlations were equally dispersed across lags, indicating a lack of autocorrelation. Note that non-linear dependencies can still exist for the residuals.

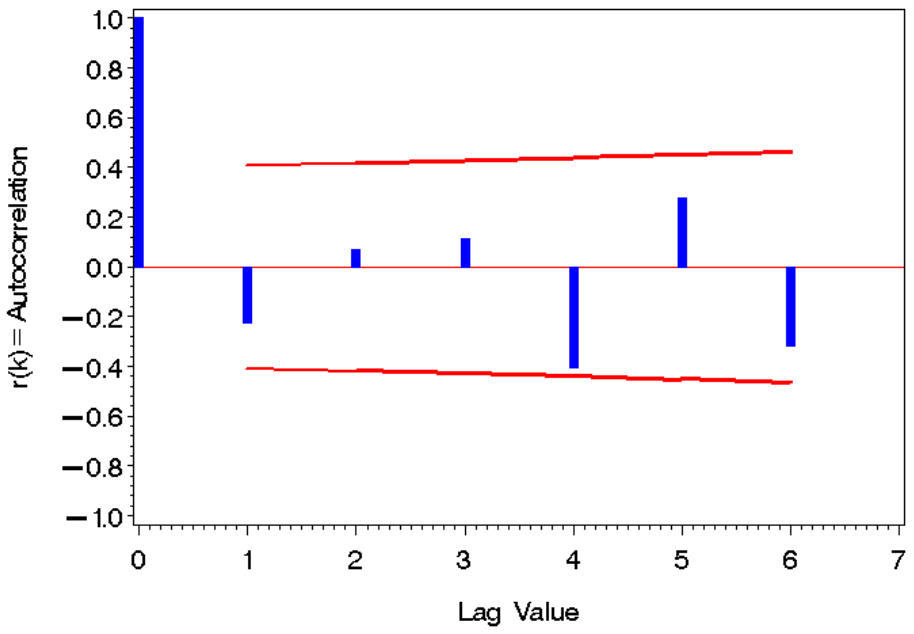

Figure 1. ACF Plot (correlogram) of MANOVA Residuals for the Serial Correlation Test.

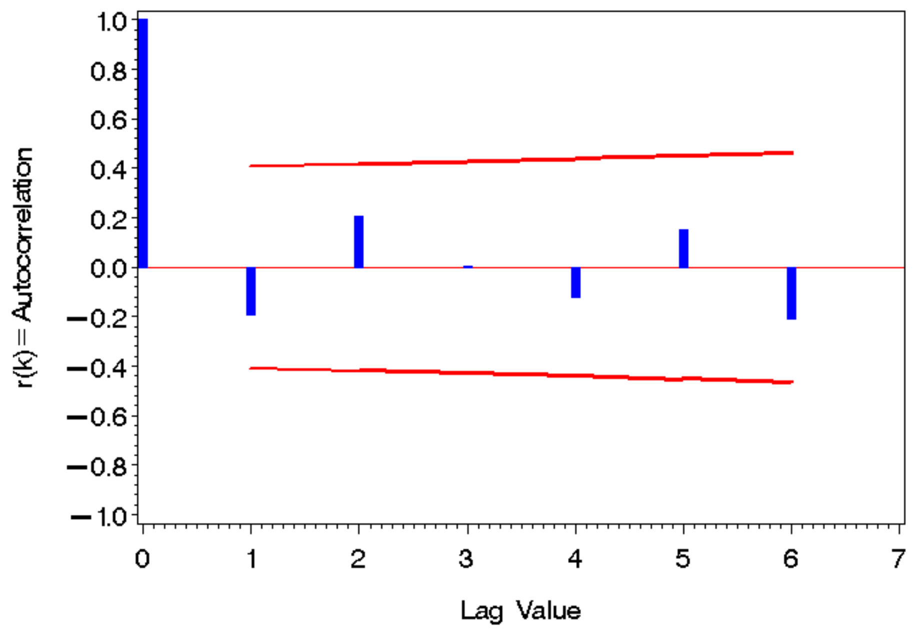

Figure 2. ACF Plot (correlogram) of MANOVA Residuals for the Runs Up Test.

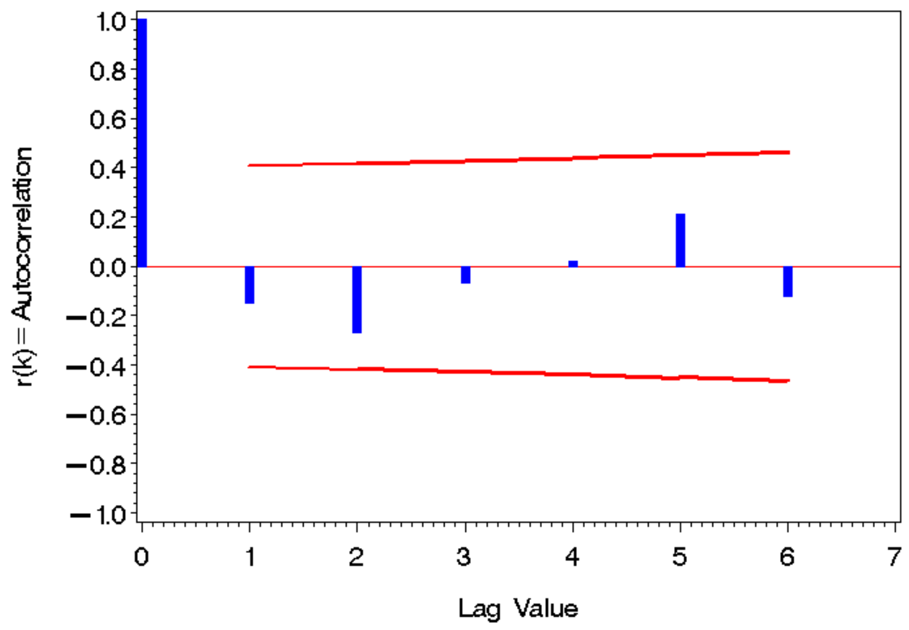

Figure 3. ACF Plot (correlogram) of MANOVA Residuals for the White Noise Test.

Normality

To test the assumption of normality, both univariate and multivariate anlayses were used. A discussion of each follows.

Univariate Analyses. Each dependent variable was tested for univariate normality (as outlined by D'Agostino et al., 1990; Looney, 1995). Table 1 contains the p-values of four univariate tests for normality. All null hypotheses (normality) were supported for the serial and runs up tests. However, the white noise test obtained a significant p-value (0.0047) for the Pearson kurtosis test as well as for both omnibus tests --- ignoring the Bonferroni adjustment (Shapiro-Wilk $\underline{W} = 0.0317$, and D'Agostino-Pearson $\underline{K}^2 = 0.0171$). Analysis of the histogram and the large negative value for Fisher's kurtosis ($\underline{g}_2 = -1.455$) indicate that the empirical distribution is likely bimodal. Although the MANOVA is quite robust to non-normality, Stevens (1996) suggested applying transformations rather than relying on model robustness. Accordingly, the Box-Cox family of power transformations were applied to each dependent variable (for the SAS program see Timm & Mieczkowski, 1997). Subsequently, small improvements for the white noise normality hypotheses were obtained (see Table 2). Moreover, an analysis of box-plots (see Figures 4-6) and quantile-quantile (Q-Q) plots (see Figures 7-12) of the transformed data indicate that a moderate outlying value is likely affecting white noise normality. Given a Bonferroni adjusted alpha level of .0125 (i.e., .05 shared among the four tests) all univariate normality hypotheses were supported. (Note that the kurtosis p-value for the white noise test was .0125.) Although univariate (marginal) normality is a requirement for multivariate (joint) normality, it does not guarantee joint normality (Stevens, 1996).

Table 1

p-values for Univariate Tests of Normality

	Dependent Variable		
Statistic	Serial Correlation	Runs Up	White Noise
Shapiro-Wilk	0.1293	0.4013	0.0316
Skewness	0.3489	0.9336	0.7170
Kurtosis	0.9316	0.0787	0.0047*
D’Agostino Pearson**	0.6425	0.2124	0.0171

Note. *p < .0125. ** Chi Square (2 df)

Table 2

p-values for Univariate Tests of Normality (Transformed)

	Dependent Variable		
Statistic	Serial Correlation	Runs Up	White Noise
Shapiro-Wilk	0.1439	0.4924	0.0222
Skewness	0.3935	0.6928	0.4324
Kurtosis	0.9277	0.1773	0.0125*
D’Agostino Pearson**	0.6920	0.3723	0.0326

Note. *p < .0125. ** Chi Square (2 df)

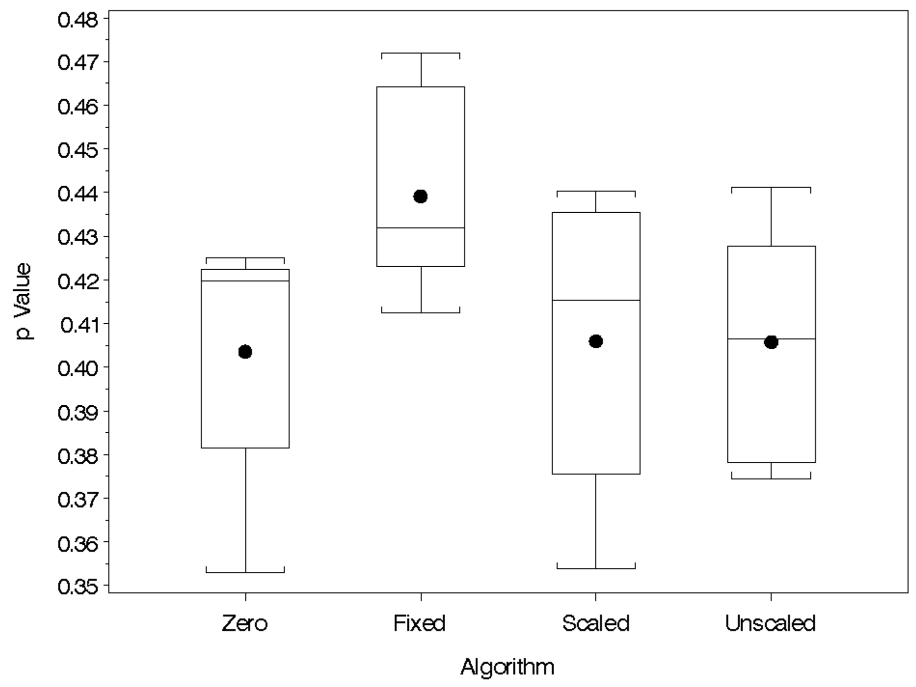

Figure 4. Box-Plot of Serial Correlation Test p-Values.

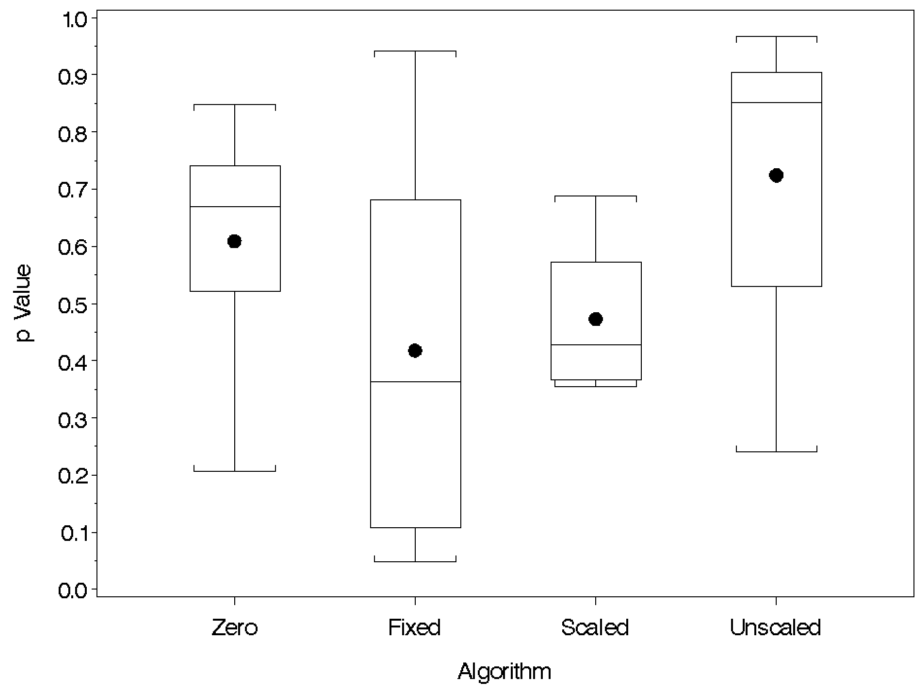

Figure 5. Box-Plot of Runs Up Test p-Values.

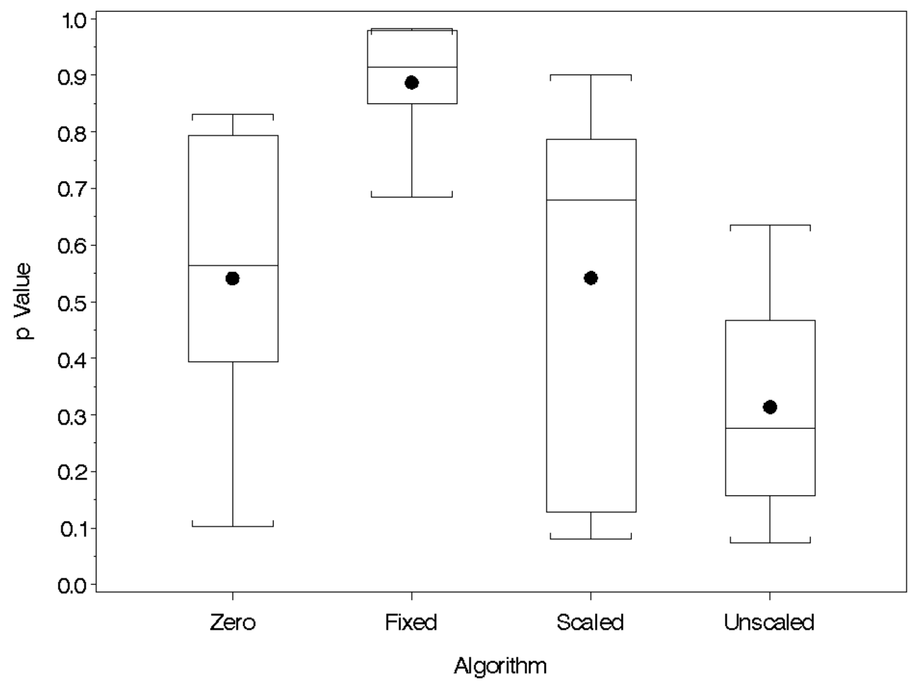

Figure 6. Box-Plot of White Noise Test p-Values.

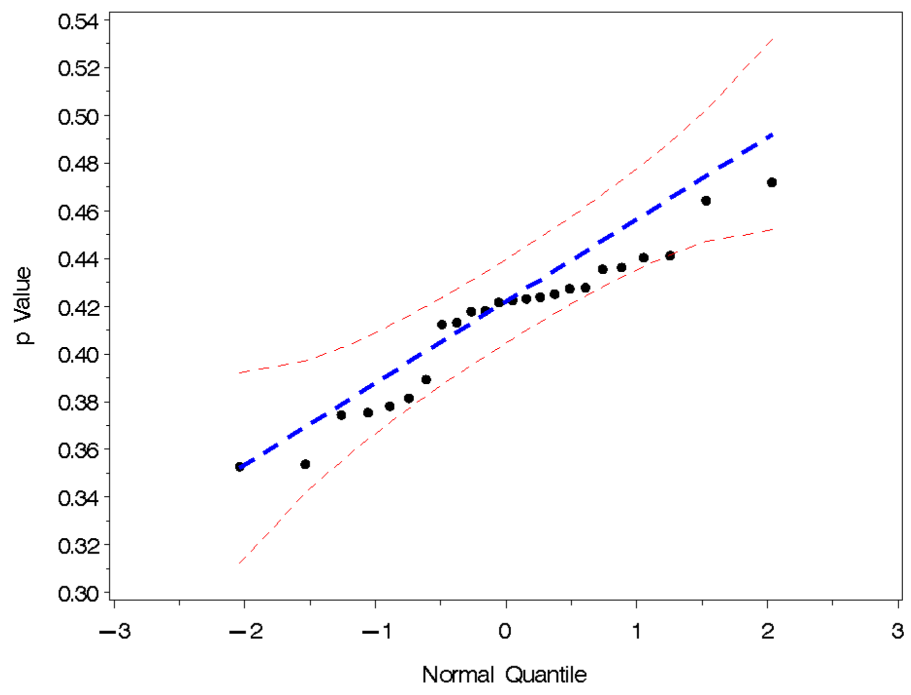

Figure 7. Q-Q Plot of Serial Correlation Test p-Values.

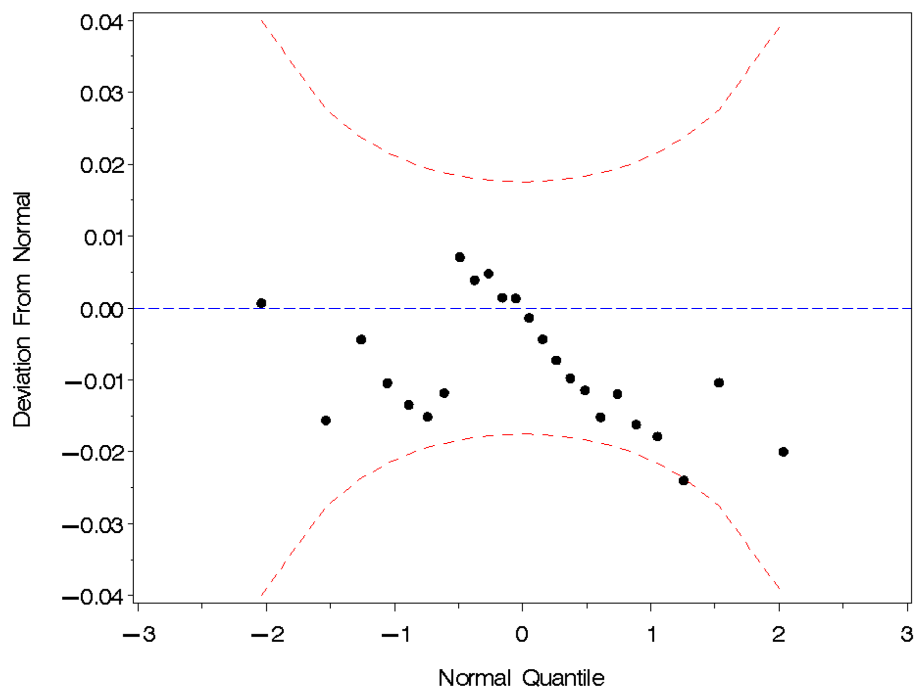

Figure 8. Q-Q Plot of Serial Correlation Test p-Values (de-trended).

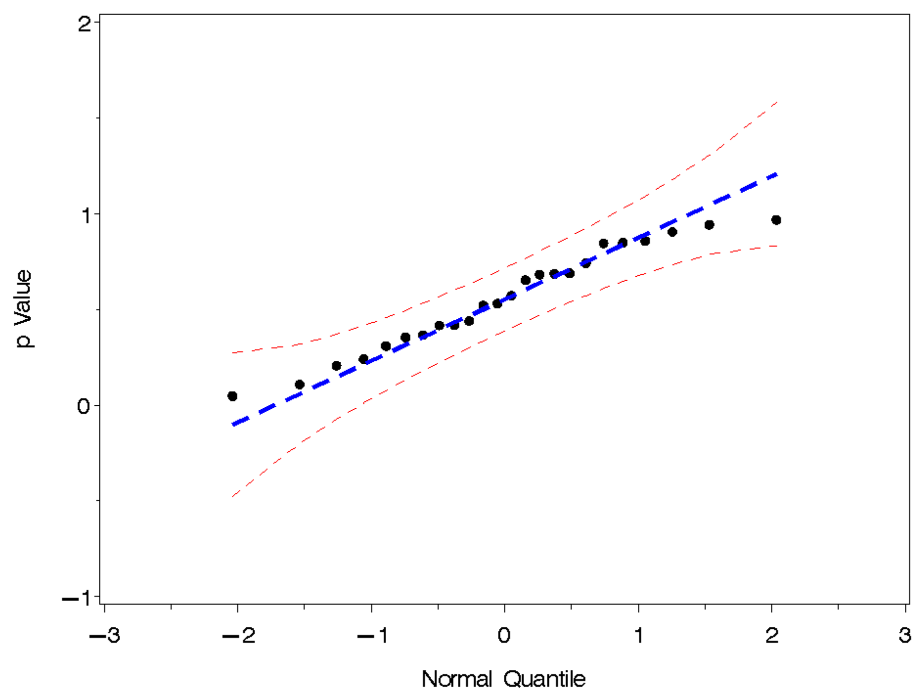

Figure 9. Q-Q Plot of Runs Up Test p-Values.

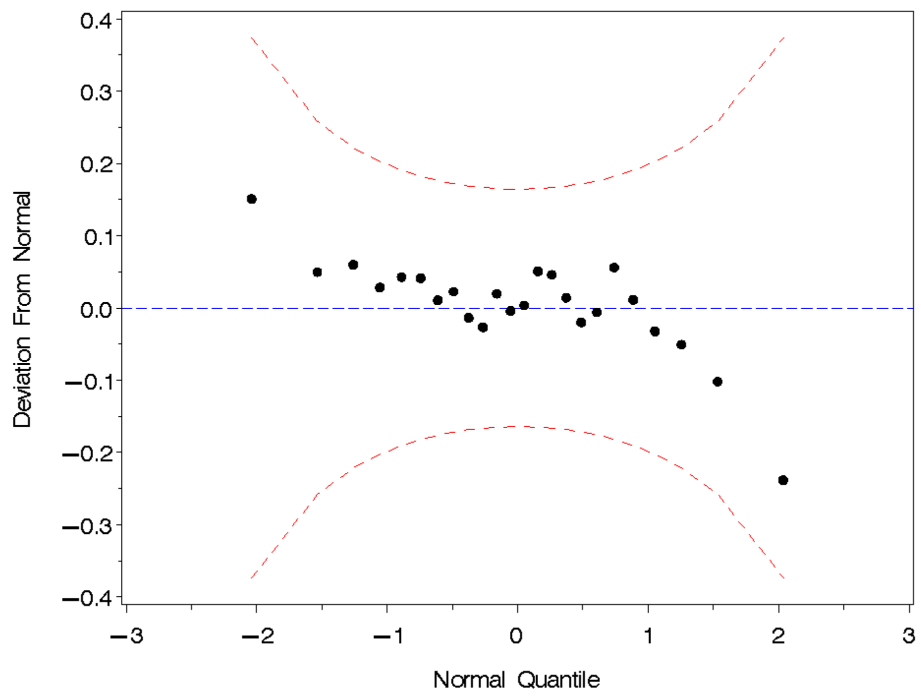

Figure 10. Q-Q Plot of Runs Up Test p-Values (de-trended).

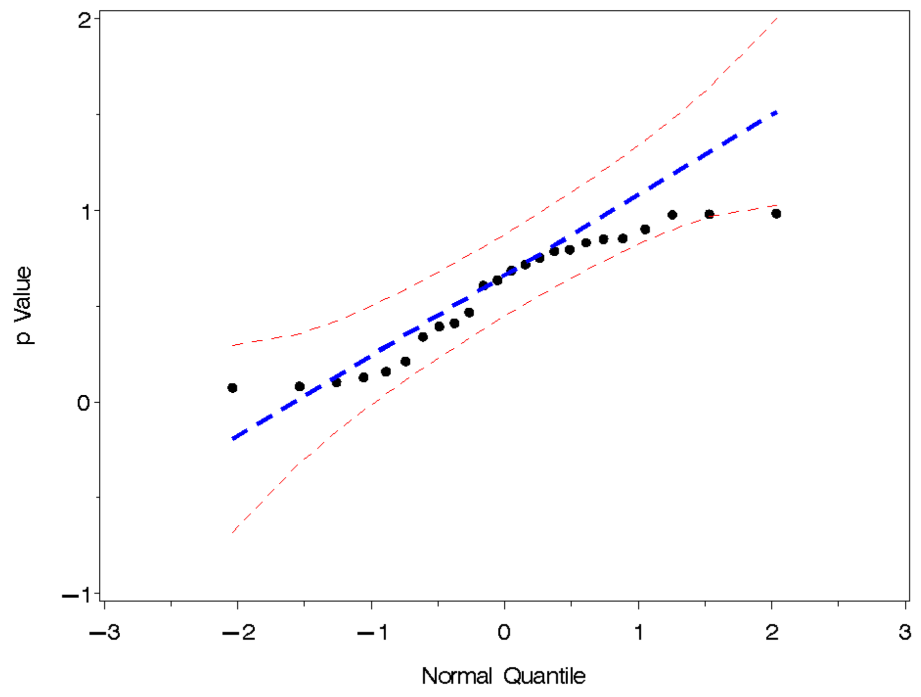

Figure 11. Q-Q Plot of White Noise Test p-Values.

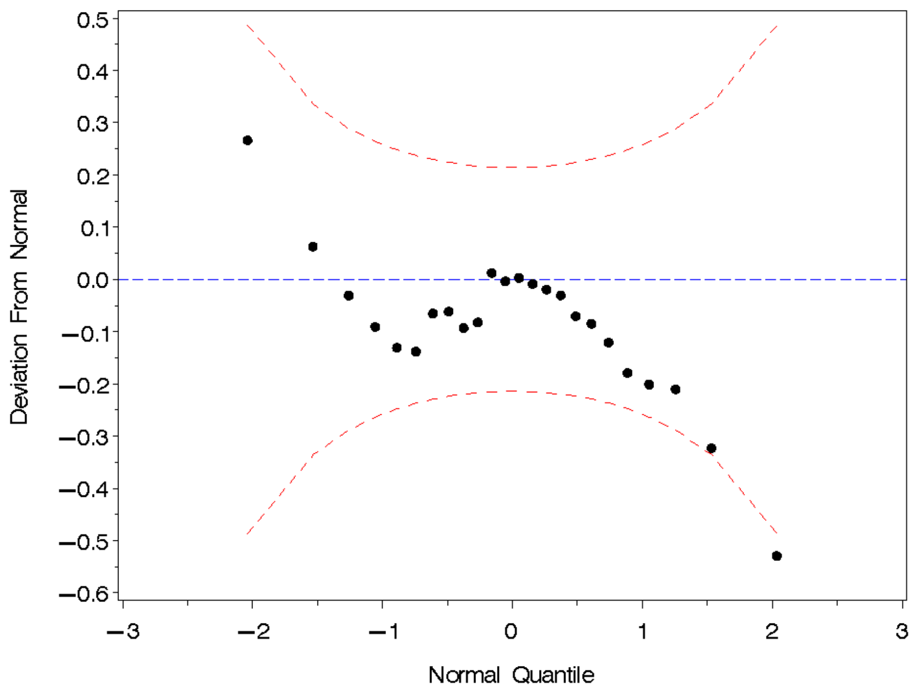

Figure 12. Q-Q Plot of White Noise Test p-Values (de-trended).

Multivariate Analyses. For the transformed data, moderate departures from multivariate normality were indicated by graphical analyses. Specifically, (a) the bivariate plots were semi-elliptical (see Figures 13-15) and (b) the Q-Q plots (to include the P-P plot) departed from the reference line (see Figures 16-18). Although one or more outlying values was suspected (see Osborne & Overbay, 2004), the outlier plot did not show any drastic departures from the 1 scale line (see Figure 19) (Johnson & Wichern, 1998). Given a Bonferonni adjusted alpha level of 0.025, all significance tests supported the null hypothesis. The p-values for Mardia's (1970) tests for multivariate skewness and kurtosis were 0.6298 and 0.1770 respectively. Additionally, the cube root test p-value was 0.1481 (Mudholkar, McDermott, & Srivastava, 1992).

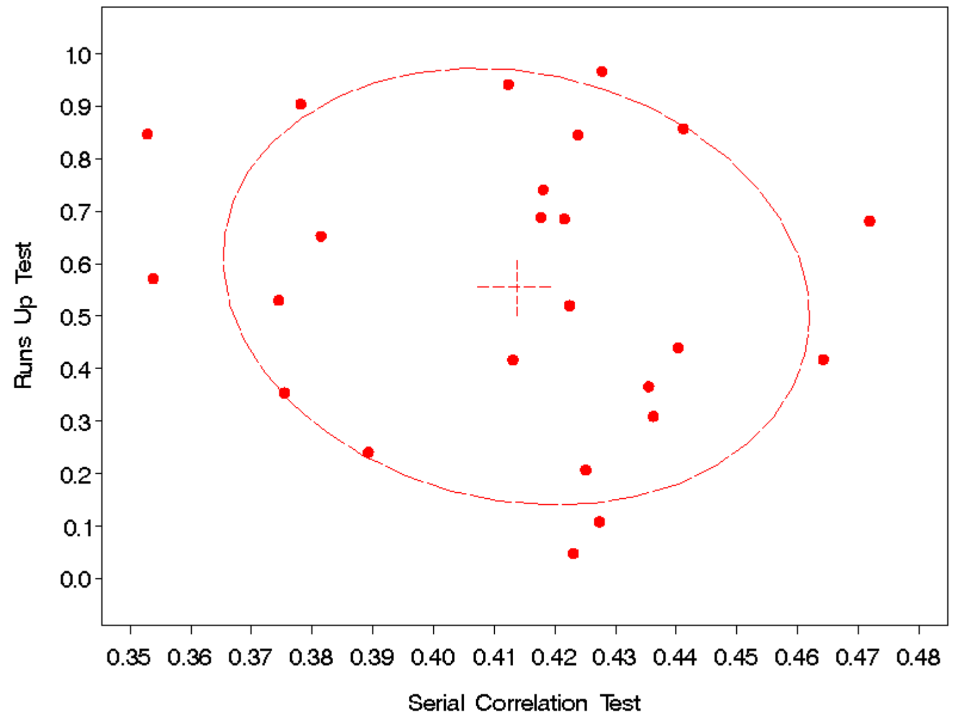

Figure 13. Bivariate Plot of Serial Correlation Test vs. Runs Up Test p-Values.

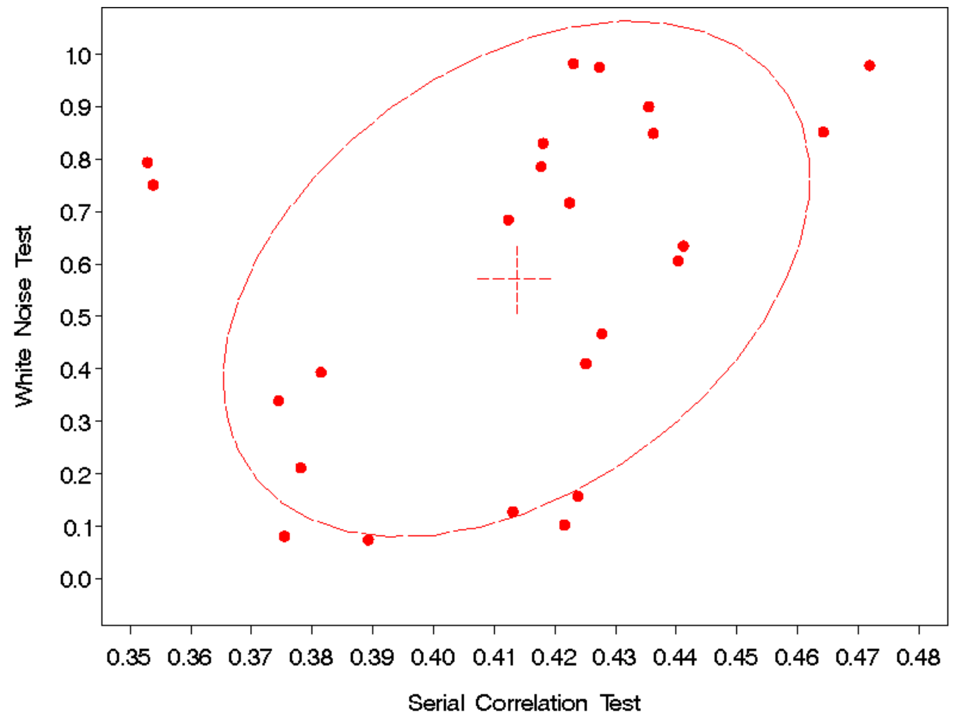

Figure 14. Bivariate Plot of Serial Correlation Test vs. White Noise Test p-Values.

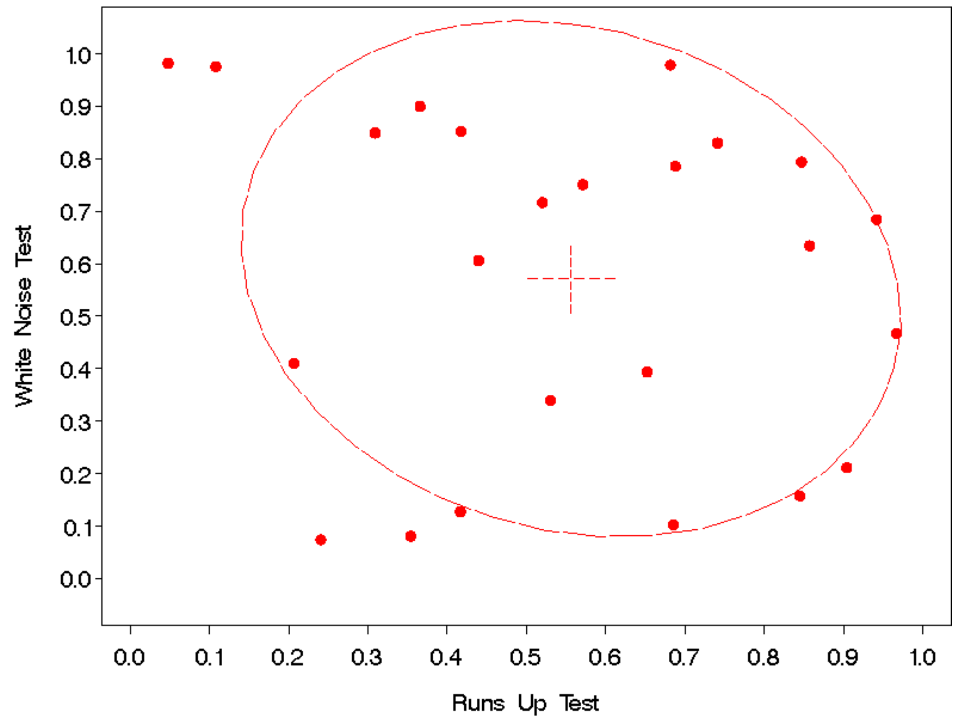

Figure 15. Bivariate Plot of Runs Up Test vs. White Noise Test p-Values.

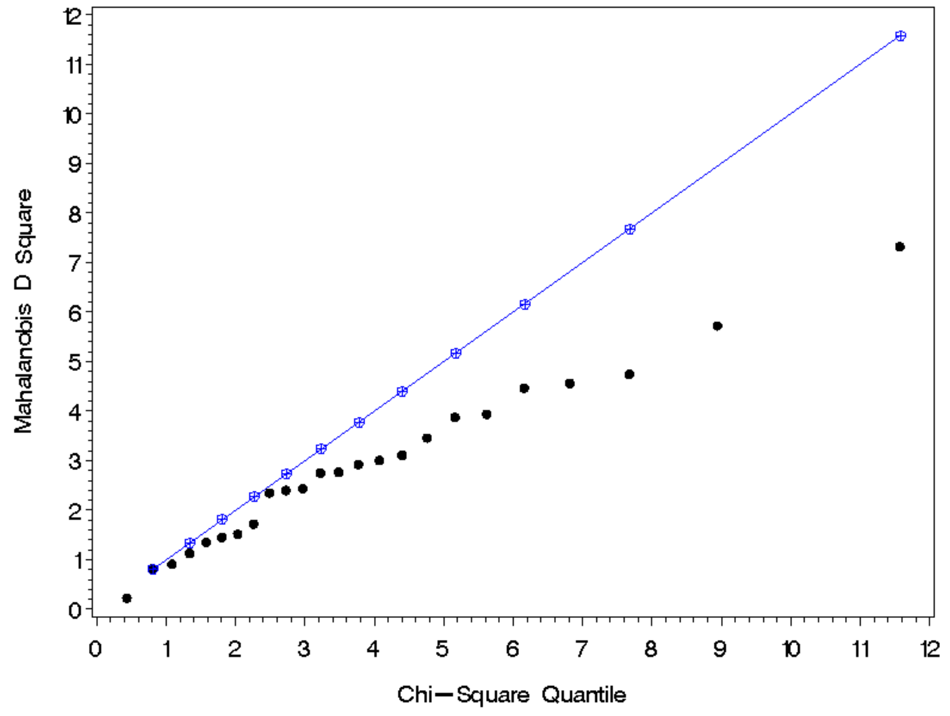

Figure 16. Multivariate Q-Q Plot of p-Values (chi).

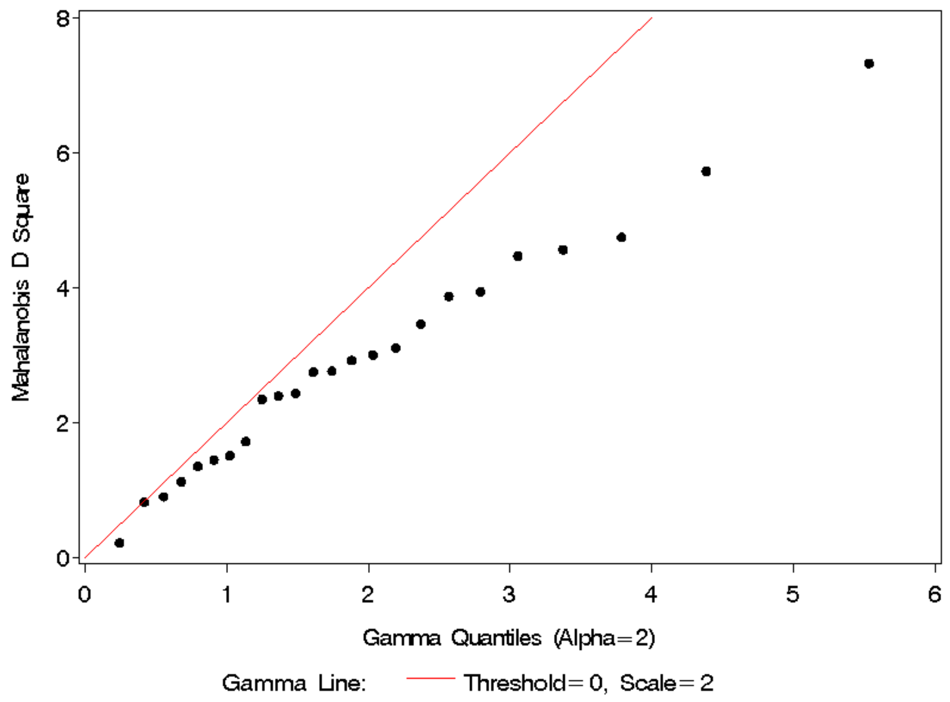

Figure 17. Multivariate Q-Q Plot of p-Values (gamma).

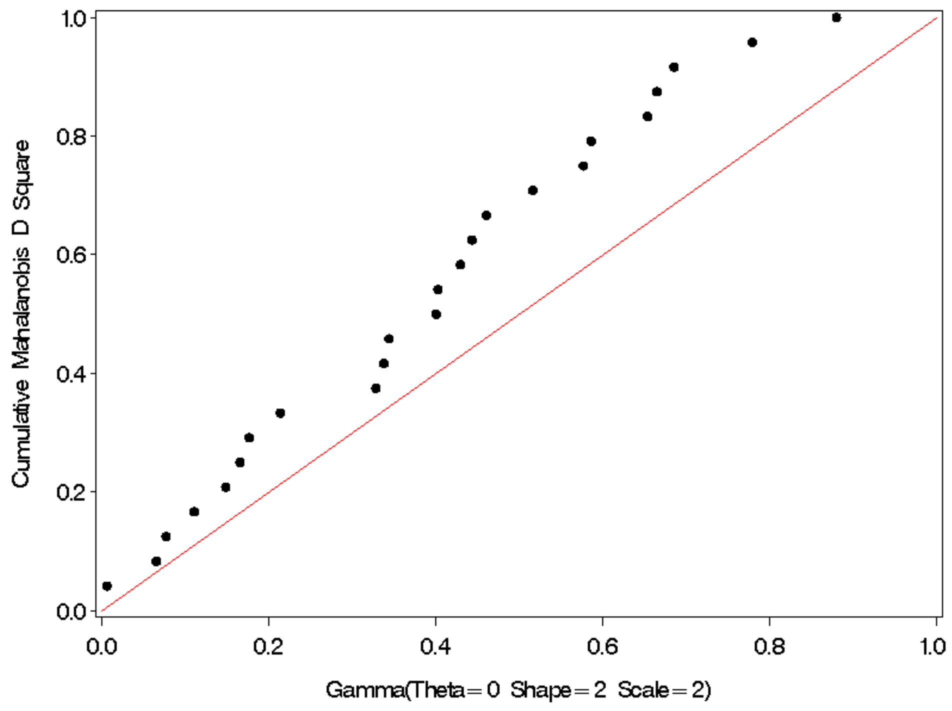

Figure 18. Multivariate P-P Plot of p-Values (gamma).

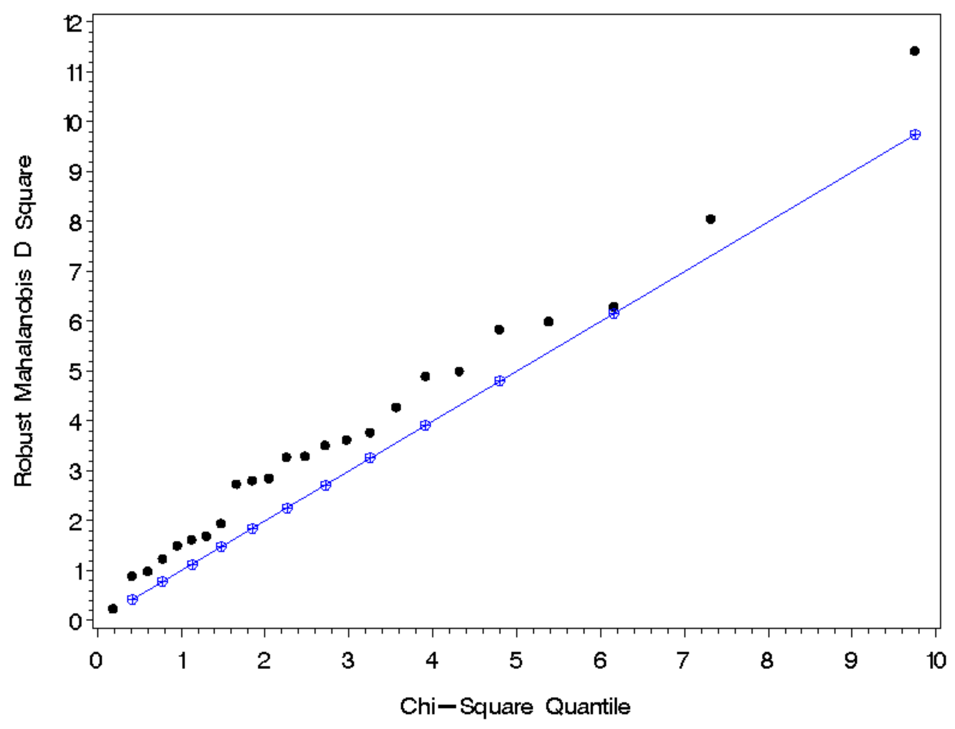

Figure 19. Multivariate Outlier Plot of p-Values (chi).

Homogeneity

The assumption of homogeneity of the covariance matrices was tested using the Bartlett likelihood ratio test. For the transformed data, the hypothesis of homogeneity was supported (df=18, p-value= 0.4561). Subsequently, the spread-versus-level plots for each dependent variable (see Figures 20-22) suggested power transformations for the serial correlation and white noise variables (slope=-3.13 and -0.52 respectively; power [i.e., 1-slope]= 4 and 2 respectively). Although transformations would likely improve homogeneity (and normality), in this case the measurement scale is not arbitrary (i.e., it is expected that the measurement scale [p-values] will vary linearly with the independent variable levels [the degree of randomness attained by the seeding algorithms]Myers & Well, 2003). Consequently, gauging effects (if any) would likely be difficult as “the relative distances among means may change” (Myers & Well, 2003, p. 224).

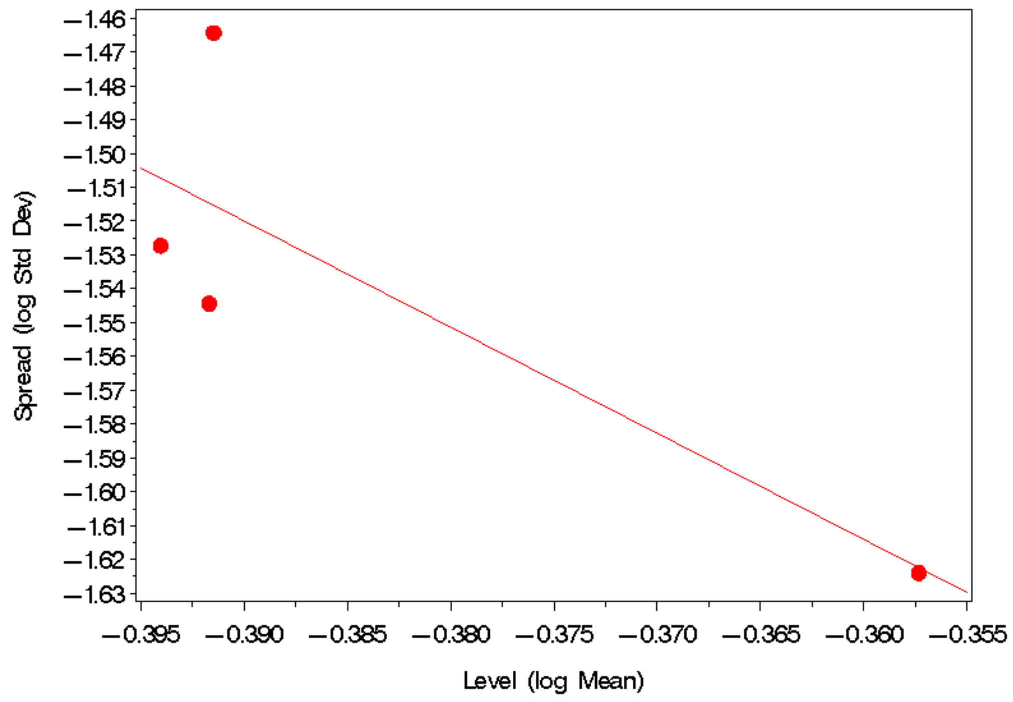

Figure 20. Spread-Versus-Level Plot for the Serial Correlation Test.

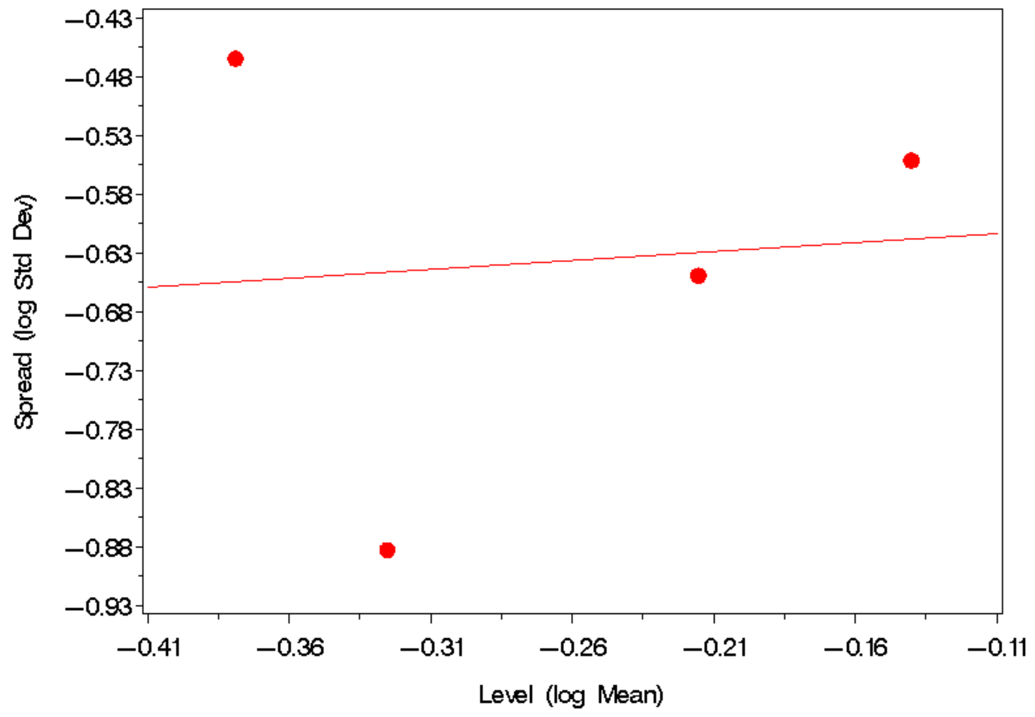

Figure 21. Spread-Versus-Level Plot for the Runs Up Test.

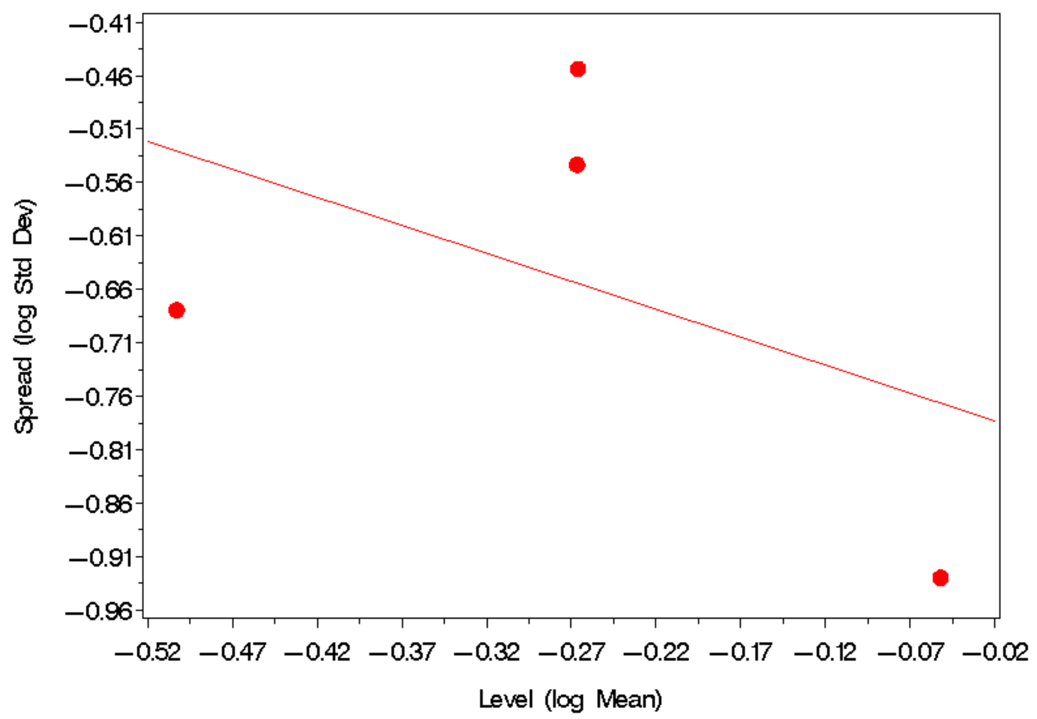

Figure 22. Spread-Versus-Level Plot for the White Noise Test.

<u>Model Outcomes</u>

In this section, the results of the hypothesis and contrasts for the MANOVA design are presented in the following order: (a) omnibus hypothesis -- no difference among seeding algorithms at the .01 level, (b) contrast 1: unscaled random algorithm vs. all others, (c) contrast 2: scaled random algorithm vs. all others, and (d) contrast 3: fixed algorithm vs. all others. Given that all model assumptions have been satisfied (independence, normality, and homogeneity), any of the four tests statistics (i.e., Wilk’s lamda, Roy’s largest root, the Hotelling-Lawley trace, the Pillai-Bartlett trace) are valid. However, preference is given to Wilk's lambda.

Table 3

MANOVA for the Hypothesis of No Overall Algorithm Effect

Statistic	df(num)	df(den)	F	p
Wilks' Lambda	9	43.958	2.26	0.0356
Pillai's Trace	9	60	1.90	0.0692
Hotelling-Lawley Trace	9	25.285	2.64	0.0263
Roy's Greatest Root	3	20	8.32	0.0009

Note. F Statistic for Roy's Greatest Root is an upper bound.

Hypothesis 1 stated that there would be no significant difference (at the .01 level) among four seeding algorithms as measured by (a) the serial test, (b) the runs up test, and (c) the white noise test. Although hypothesis 1 was supported (Wilk's lambda p-value= 0.0356), it should be noted that the significance level was adjusted downward to compensate for the moderate dependence of the white noise test. The overall strength of the p-value does suggest that the algorithms differ (if not significantly).

Table 4

MANOVA for the Hypothesis of No Overall Unscaled vs. Others Effect

Statistic	df(num)	df(den)	F	p
Wilks' Lambda	3	18	4.09	0.0223
Pillai's Trace	3	18	4.09	0.0223
Hotelling-Lawley Trace	3	18	4.09	0.0223
Roy's Greatest Root	3	18	4.09	0.0223

Note. F Statistic for Roy's Greatest Root is an upper bound.

Hypothesis 2 (contrast 1) stated that there would be no significant difference (at the .01 level) between the unscaled random seeding algorithm and all other algorithms. Hypothesis 2 was supported (Wilk's lambda p-value= 0.0223). This outcome is surprising because this algorithm has the best theoretical properties among the other algorithms. However, the overall strength of the p-value suggests that the unscaled algorithm differs from the others (if not significantly).

Table 5

MANOVA for the Hypothesis of No Overall Scaled vs. Others Effect

Statistic	df(num)	df(den)	F	p
Wilks' Lambda	3	18	0.41	0.7477
Pillai's Trace	3	18	0.41	0.7477
Hotelling-Lawley Trace	3	18	0.41	0.7477
Roy's Greatest Root	3	18	0.41	0.7477

Note. F Statistic for Roy's Greatest Root is an upper bound.

Hypothesis 3 (contrast 2) stated that there would be no significant difference (at the .01 level) between the scaled random algorithm and all other algorithms. Hypothesis 3 was supported (Wilk's lambda p-value= 0.7477). This outcome is surprising because theoretically the scaled random algorithm has better theoretical properties than either the zero-leap or fixed-leap algorithms. The strength of the p-value suggests that the scaled random algorithm does not differ substantially from the others.

Table 6

MANOVA Hypothesis of No Overall Fixed vs. Others Effect

Statistic	df(num)	df(den)	F	p
Wilks' Lambda	3	18	6.03	0.0050
Pillai's Trace	3	18	6.03	0.0050
Hotelling-Lawley Trace	3	18	6.03	0.0050
Roy's Greatest Root	3	18	6.03	0.0050

Note. F Statistic for Roy's Greatest Root is an upper bound.

Hypothesis 4 (contrast 3) stated that there would be no significant difference (at the .01 level) between the fixed seeding algorithm and all other algorithms. Hypothesis 4 was not supported (Wilk's lambda p-value= 0.0050). This outcome is surprising because intuitively one would not expect the fixed-leap algorithm to differ significantly in its ability to disrupt the serial correlation patterns in the random number stream. Indeed, one would expect that either the scaled random or unscaled random algorithms would be better disrupters of serial correlation patterns.

In sum, the MANOVA assumptions of independence, normality, and homogeneity were satisfied. Seeding algorithms did not differ significantly from each other. However, the fixed-leap algorithm differed significantly from all other algorithms. Surprisingly, the scaled random-leap had the least difference among the algorithms (theoretically this algorithm should have produced the second largest difference).

CHAPTER 5

CONCLUSIONS

The objectives of this study were three-fold: (a) to develop an algorithm that satisfies the four theoretical conditions for producing independent seeds, (b) to statistically compare the performance of the four algorithms, and (c) to identify areas for further research. A discussion of each objective follows:

Algorithm Development

Four theoretical conditions for producing independent seeds have been proposed. Accordingly, seeds should be (a) obtained using simple random sampling, (b) non-replicated, (c) non-overlapping, and (d) non-adjoining (ideally values should be leaped between sequences). Of these, obtaining all seeds using simple random sampling is the most difficult to satisfy. Indeed, the algorithms proposed (prior to this study) only allow the starting seed to be randomly selected. Moreover, manually compiling all seeds (e.g., 100,000) from a source of random digits would be a formidable task. Consequently, an algorithm satisfying all four criteria had to be developed for this study. To satisfy the random sampling condition, an initial block of seeds become sampling numbers for selecting seeds from the entire random number stream.

Statistical Comparisons

As to statistical comparisons, the primary implication of this study follows: Seeding algorithms have an effect on three properties (independence, monotonicity, and periodicity) of pseudo-random numbers obtained from multiplicative congruential generators. Specifically, the fixed-leap algorithm differed significantly from all other algorithms. Intuitively, one would not expect the systematic sampling approach of the fixed-leap algorithm to simultaneously differ from the algorithms following a random sampling approach (viz., unscaled and scaled random-leap) and the zero-leap algorithm (which offers minimal randomness). However, Madow (1946, p. 213) pointed out that in some circumstances systematic sampling could offer improved efficiencies over random sampling designs but warned that "the chief danger in applying a systematic design occurs when the data have a periodic formation, and the sampling interval chosen is equal to the period of the data." In this case the latter is troubling given that by chance a fixed-leap seeds could match the periodicity of the serial correlation, thereby yielding undesirable pseudo-random numbers.

Although the unscaled random-leap differed substantially from all other algorithms, significant differences were not obtained. This finding is a surprise. Indeed, intuitively one would expect this algorithm to have the best randomness properties because it satisfies the four theoretical conditions of independent seeding. A possible explanation involves the degree that the non-adjoining condition was satisfied. Although randomly selected, a small leap value (say 100) between two (or more) sequences may not have been sufficient to attenuate serial correlation (i.e., some sequences likely approximated the zero-leap algorithm --- theoretically the weakest of the four algorithms). One solution would be to parameterize a minimum leap value (say 1000) to further strengthen the non-adjoining condition.

A second surprise finding involves the lack of insignificance obtained by the scaled random-leap when compared with all other algorithms. Indeed, the strength of the p-value (Wilk's lamba = 0.7477) offered little support for differences. As with the unscaled random-leap, it is plausible that small leap values caused one or more sequences to approximate a joined sequence (as produced by the zero-leap algorithm). It should be noted that the zero-leap algorithm could not be compared singly with all other algorithms (due to modeling constraints governing the number of planned comparisons). Although a glaring limitation, it is likely this algorithm would not have differed significantly from the others.

Caution should be exercised given the exploratory and inconclusive nature of these findings. One is tempted to favor the fixed-leap algorithm; however, it could yield disastrous results if the leap matches the periodic component of the generator's serial correlation. Although not fully supported by the research design used in this study, it is thought that the unscaled random-leap algorithm is the best choice for independently seeding the multiplicative congruential random number generator.

Further Research

As to further research, a first priority should be to resolve the methodological limitations exposed by this study. Specific suggestions are:

1. The fixed-leap and unscaled random-leap algorithms should be compared using the runs up test and the following directional hypothesis: the mean of the runs up test p-values will be greater for the fixed-leap algorithm than the unscaled random-leap (at the .025 level).
2. The zero-leap and unscaled random-leap algorithms should be compared using the runs up test and the following directional hypothesis: the mean of the runs up test p-values will be greater for the zero-leap algorithm than the unscaled random-leap (at the .025 level).
3. The number of equally spaced seeds (for both simulation designs) should be increased from three to nine (see Chapter 3 for selection procedures). This should make it more likely that the algorithms will encounter increased serial correlation effects. However, it may also increase dependence among the observations causing (a) positive intraclass

correlation coefficients and (b) larger Bonferroni type adjustments for the significance level. Alternatively, a collinear MANVOA may be applicable (see Langsrud, 2002).

4. The use of exact tests to measure the dependent variables should be considered (e.g., Dieter & Ahrens, 1971, have developed an exact serial correlation test for pseudo-random number sequences based on Dedekind sums).

5. More broadly, the assessment protocol established by this study should be applied to other classes of random number generators that require seeding.

REFERENCES

Anderson, T. W. (1958). An introduction to multivariate statistical analysis. New York: John Wiley and Sons.

Anderson, R. L. (1942). Distribution of the serial correlation coefficient. The Annals of Mathematical Statistics, 13(1), 1-13.

Bang, J. W., Schumacker, R. E., & Schlieve, P. L. (1998). Random-number generator validity in simulation studies: An investigation of normality. Educational and Psychological Measurement, 58, 430-450.

Bartlett, M. S. (1966). An introduction to stochastic processes (2nd ed.). Cambridge, MA: Cambridge University Press.

Beran, J. (1992). Statistical methods for data with long-range dependence. Statistical Science, 7, 404-416.

Boland, P. J., & Hutchinson, K. (2000). Student selection of random digits. The Statistician, Journal of the Royal Statistical Society – Series D, 49, 519-529.

Box, G. E. P., & Jenkins, G. M. (1970). Time series analysis forecasting and control. San Francisco: Holden Day.

Bradley, J. V. (1968). Distribution-free statistical tests. Englewood Cliffs, NJ: Prentice-Hall.

Brysbaert, M. (1991). Algorithms for randomness in the behavioral sciences: A tutorial. Behavior, Research Methods, Instruments, and Computers, 23(2), 45-60.

Christensen, R., & Bedrick, E. J. (1997). Testing the independence assumption in linear models. Journal of the American Statistical Association, 92, 1006-1016.

Clark, M. R., & Woodward, D. E. (1992). Generating random numbers with Base SAS software. Observations: The Technical Journal for SAS Software Users, 1(4), 12-19.

Collings, B. J. (1987). Compound random number generators. Journal of the American Statistical Association, 82, 525-527.

Coveyou, R. R. (1960). Serial correlation in the generation of pseudo-random numbers. Journal of the Association for Computing Machinery, 7(1), 72-74.

D’Agostino, R. B., Belanger, A., & D’Agostino, R. B., Jr. (1990). A suggestion for using powerful and informative tests of normality. The American Statistician, 44, 316-321.

De Matteis, A., & Pagnutti, S. (1988). Parallelization of random number generators and long-range correlations. Numerische Mathematik, 53, 595-608.

Dieter, U., & Ahrens, J. (1971). An exact determination of serial correlations of pseudo-random numbers. Numerische Mathematik, 17, 101-123.

Eichenauer-Herrmann, J., & Grothe, H. (1989). A remark on long-range correlations in multiplicative congruential pseudo random number generators. Numerische Mathematik, 56, 609-611.

Emerson, J. D., & Stoto, M. A. (1982). Exploratory methods for choosing power transformations. Journal of the American Statistical Association, 77, 103-108.

Fan, X., Felsovalyi, A., Sivo, S. A., & Keenan, S. C. (2002). SAS for Monte Carlo studies: A guide for quantitative researchers. Cary, NC: SAS Institute.

Ferrenberg, A. M., Landau, D. P., & Wong, Y. J. (1992). Monte Carlo simulations: Hidden errors from 'good' random number generators. Physical Review Letters, 69, 3382-3384.

Fishman, G. S., & Moore, L. R. (1982). A statistical evaluation of multiplicative congruential random number generators with modulus 2^{31}-1. Journal of the American Statistical Association, 77, 129-136.

Fishman, G. S., & Moore, L. R. (1986). An exhaustive analysis of multiplicative congruential random number generators with modulus 2^{31}-1. SIAM Journal on Scientific and Statistical Computing, 7(1), 24-45.

Freiberger, W., & Grenander, U. (1971). A course in computational probability and statistics. New York: Springer-Verlag.

Friendly, M. (1991). SAS System for statistical graphics. Cary, NC: SAS Institute, Inc.

Gastwirth, J. L., & Rubin, H. (1971). Effect of dependence on the level of some one-sample tests. Journal of the American Statistical Association, 66, 816-820.

Gentle, J. E. (1998). Random number generation and Monte Carlo methods. New York: Springer-Verlag.

Gleser, L. J., & Moore, D. S. (1983). The effect of dependence on chi-squared and empiric distribution tests of fit. The Annals of Statistics, 11, 1100-1108.

Gleser, L. J., & Moore, D. S. (1985). The effect of positive dependence on chi-squared tests for categorical data. Journal of the Royal Statistical Society. Series B (Methodological), 47, 459-465.

Gleser L. J., & Olkin, I. (1994). Stochastically dependent effect sizes. In H. Cooper & L. V. Hedges (Eds.), The handbook of research synthesis (pp. 339-356). New York: Russell Sage Foundation.

Good, I. J. (1957). On the serial test for random sequences. The Annals of Mathematical Statistics, 28, 262-264.

Gruenberger, F., & Jaffray, G. (1965). Problems for computer solution. New York: John Wiley and Sons.

Hamaker, H. C. (1949). A simple technique for producing random sampling numbers. Proceedings, Koninklijke Nederlandse Akademie van Wetenschappen, 52, 145-150.

Headrick, T. C., & Sawilowsky, S. S. (1999a). Simulating correlated multivariate nonnormal distributions: Extending the Fleishman power method. Psychometrika, 64, 25-35.

Headrick, T. C., & Sawilowsky, S. S. (1999b). Errata for "Simulating correlated multivariate nonnormal distributions: Extending the Fleishman power method". Psychometrika, 64, 251.

Hearne, E. M., III, Clark, G. M., & Hatch, J. P. (1983). A test for serial correlation in univariate repeated-measures analysis. Biometrics, 39, 237-243.

Hinkle, D. E., Wiersma, W., & Jurs, S. G. (1994). Applied statistics for the behavioral sciences (3rd ed.). Boston: Houghton-Mifflin.

Hoaglin, D. C. (1976). Theoretical properties of congruential random number generators: An empirical view. Memorandum NS-349, Harvard University, Department of Statistics.

Hoaglin, D. C. (1991). Fundamentals of exploratory data analysis. New York: Wiley-IEEE.

Johnson, M. E. (1987). Multivariate statistical simulation. New York: John Wiley and Sons.

Johnson, R. A., & Wichern, D. W. (1998). Applied multivariate statistical analysis (4th ed.). Upper Saddle River, NJ: Prentice-Hall.

Kaplan, H. L. (1981). Effective random seeding of random number generators. Behavior Research Methods and Instrumentation, 13(2), 283-289.

Kareev, Y. (1992). Not that bad after all: Generation of random sequences. Journal of Experimental Psychology: Human Perception and Performance, 18(4), 1189-1194.

Kelly, J. K. (2000). Producing a table of seeds for random number generation. Proceedings of the Southeast SAS Users Group (SESUG), USA, 4, P-408.

Kendall, M. G., & Babington-Smith, B. (1938). Randomness and random sampling numbers. Journal of the Royal Statistical Society, 101(1), 147-166.

Kennedy, W. J., Jr., & Gentle, J. E. (1980). Statistical computing. New York: Marcel Dekker.

Kenny, D. A., & Judd, C. M. (1986). Consequences of violating the independence assumption in the analysis of variance. Psychological Bulletin, 99, 422-431.

Keselman, H. J. (2005). Multivariate normality tests. In B. S. Everitt & D. C. Howell (Eds.), Encyclopedia of statistics in behavioral science (Volume 3, 1373-1379). New York: Wiley.

Khattree, R., & Naik, D. N. (1999). Applied multivariate statistics with SAS software (2nd ed.). Cary, NC: SAS Institute.

Kiefer, N. M. (1982). Testing for dependence in multivariate probit models. Biometrika, 69(1), 161-166.

Knuth, D. E. (1981). The art of computer programming, Vol. 2: Seminumerical algorithms (2nd ed.). Reading, MA: Addison-Wesley.

Kruskal, W. (1988). Miracles and statistics: The casual assumption of independence. Journal of the American Statistical Association, 83, 929-940.

Kundu, D., & Basu, A. (2004). Statistical computing: Existing methods and recent developments. Oxford, United Kingdom: Alpha Science International, Ltd.

Lahiri, S. N. (2003). Resampling methods for dependent data. New York: Springer-Verlag.

Langsrud, O. (2002). 50-50 multivariate analysis of variance for collinear responses. The Statistician, 51, 305-317.

L'Ecuyer, P. (1988). Efficient and portable combined random number generators. Communications of the ACM, 31, 742-749, 774.

Lehmer, D. H. (1951). Second symposium on large-scale digital calculating machinery. Cambridge, MA: Harvard University Press.

Levene, H. & Wolfowitz, J. (1944). The covariance matrix of runs up and down. The Annals of Mathematical Statistics, 15(1), 58-69.

Lewis, P. A. W., Goodman, A. S., & Miller, J. M. (1969). A pseudo-random number generator for the System/360. IBM Systems Journal, 136-146.

Littell, R. C., Freund, R. J., & Spector, P. C. (1991). SAS system for linear models (3rd ed.). Cary, NC: SAS Institute.

Looney, S. W. (1995). How to use tests for univariate normality to assess multivariate normality. The American Statistician, 49(1), 64-70.

MacLaren, M. D., & Marsaglia, G. (1965). Uniform random number generators. Journal of the ACM, 12, 83-89.

Madow, L. H. (1946). Systematic sampling and its relation to other sampling designs. Journal of the American Statistical Association, 41, 204-217.

Manly, B. F. J. (1997). Randomization, bootstrap, and Monte Carlo methods in biology (2nd ed.). New York: Chapman and Hall.

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. Biometrika, 57, 519-530.

Marsaglia, G. (1972). The structure of linear congruential sequences in applications of number theory to numerical analysis. New York: Academic Press.

Mecklin, C. J., & Mundfrom, D. J. (2004). An appraisal and bibliography of tests for multivariate normality. International Statistical Review, 72, 123-138.

Mihram, G. A. (1972). Simulation: Statistical foundations and methodology. New York: Academic Press.

Millenson, J. R., & Sullivan, G. D. (1969). A hardware random number generator for use with computer control of probabilistic contingencies. Behavior Research Methods and Instrumentation, 1, 194-196.

Mooney, C. Z. (1997). Monte Carlo simulation. Thousand Oaks, CA: Sage.

Moore, D. S. (1982). The effect of dependence on chi squared tests of fit. The Annals of Statistics, 10, 1163-1171.

Morgan, B. J. T. (1984). Elements of simulation. Boca Raton, FL: CRC Press.

Morrison, D.F. (1976). Multivariate statistical methods. New York: McGraw-Hill.

Mudholkar, G. S., McDermott, M., & Srivastava, D. K. (1992). A test for variate normality. Biometrika, 79, 850-854.

Murray, T. W. (1972). An empirical examination of the classical assumptions concerning errors in data. Journal of the American Statistical Association, 67, 530-537.

Myers, J. L., DiCecco, J. V., & Lorch, R. F., Jr. (1981). Group dynamics and individual performances: Pseudo-group and quasi-F analyses. Journal of Personality and Social Psychology,40, 86-98.

Myers, J. L., & Well, A. D. (2003). Research design and statistical analysis (2nd ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

Nickerson, R. S. (2002). The production and perception of randomness. Psychological Review, 109, 330-357.

Osborne, J. W., & Overbay, A. (2004). The power of outliers (and why researchers should always check for them). Practical Assessment, Research and Evaluation, 9(6). Retrieved July 7, 2007 from http://PAREonline.net/getvn.asp?v=9&n=6

Pavur, R. (1988). Type I error rates for multiple comparison procedures with dependent data. The American Statistician, 42, 171-173.

Peach, P. (1961). Bias in pseudo-random numbers. Journal of the American Statistical Association, 56, 610-618.

Piegorsch, W. W., & Bailer, J. A. (2005). Analyzing environmental data. New York: John Wiley and Sons.

RAND Corporation (1955). A million random digits with 100,000 normal deviates. Glencoe, IL: Free Press.

Raudenbush, S. W., & Bryk, A. S. (2002). Hierarchical linear models: Applications and data analysis methods. (2nd ed.). Thousand Oaks, CA: Sage.

Rubinstein, R. Y. (1981). Simulation and the Monte Carlo method. New York: John Wiley and Sons.

SAS Institute, Inc. (1986). SAS System for forecasting time series. Cary NC: SAS Institute.

SAS Institute, Inc. (1990a). SAS Language: Reference, version 6 (1st ed). Cary, NC: SAS Institute.

SAS Institute, Inc. (1990b). SAS/STAT user's guide, version 6, Vol 1: ACECLUS-FREQ (4th ed.). Cary, NC: SAS Institute.

SAS Institute, Inc. (1990c). SAS/STAT user's guide, version 6, Vol 2: GLM-VARCOMP (4th ed.). Cary, NC: SAS Institute.

Scariano, S., & Davenport, J. (1987). The effects of violations of the independence assumption in the one way ANOVA. The American Statistician, 41, 123-129.

Scheaffer, R. L., Mendenhall, W., III, & Ott, R. L. (1996). Elementary survey sampling (5th ed.). Belmont, CA: Wadsworth.

Schmidt, H. (1977). A simple random number generator for use with minicomputers. Journal of the American Society for Psychical Research, 71, 171-176.

Sedgewick, R. (1977). Permutation generation methods. Computing Surveys, 9, 137-164.

Sedgewick, R. (1983). Algorithms. Reading, MA: Addison-Wesley.

Seier, E. (2002). Comparison of tests of univariate normality. Retrieved April 18, 2007, from http://interstat.statjournals.net/YEAR/2002/abstracts/0201001.php

Skiena, S. S. (1998). The algorithm design manual. New York: Springer-Verlag.

Stevens, J. S. (1996). Applied multivariate statistics for the social sciences (3rd ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

Strube, M. J. (1983). Tests of randomness for pseudorandom number generators. Behavior Research Methods and Instrumentation, 15, 536-537.

Teichroew, D. (1965). A history of distribution sampling prior to the era of the computer and its relevance to simulation. Journal of the American Statistical Association, 60(309), 27-49.

Timm, N. H., & Mieczkowski, T. A. (1997). Univariate and multivariate general linear models: Theory and applications using SAS software. Cary, NC: SAS Institute, Inc.

Tracz, S. M., & Elmore, P. B. (1985). The effect of the violation of the assumption of independence when combining correlation coefficients in a meta-analysis. Paper presented at the annual meeting of the American Educational Research Association, Chicago, IL. (ERIC Document Reproduction Service No. ED 266 151)

Tracz, S. M., Elmore, P. B., & Pohlmann, J. T. (1992). Correlational meta-analysis: Independent and nonindependent cases. Educational and Psychological Measurement, 52, 879-888.

Tracz, S. M., Newman, I., & McNeil, K. (1986). Tests of dependence in meta-analysis using multiple linear regression. Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA. (ERIC Document Reproduction Service No. ED 276 761)

Tuckwell, H. C. (1988). Elementary applications of probability. New York: Chapman and Hall.

Twagilimana, J. (2005). Time dependent data exploration and preprocessing: Doing it all by SAS. Proceedings of the Midwest SAS User Group (MWSUG), USA, 6, SS900.

Wagenaar, W. A. (1971). Serial non-randomness as a function of duration and monotony of a randomization task. Acta Psychologica, 35(1) 78-87.

Wagenaar, W. A. (1972). Generation of random sequences by human subjects: A critical survey of literature. Psychological Bulletin, 77(1), 65-72.

Wallis, W. A., & Moore, G. H. (1941). A significance test for time series analysis. Journal of the American Statistical Association, 36, 401-409.

Wei, W. W. S. (1990). Time series analysis: Univariate and multivariate methods. Redwood City, CA: Addison Wesley.

Wichmann, B. A., & Hill, I. D. (1982). An efficient and portable pseudo-random number generator. Applied Statistics, 31, 188-190 (corrections, 1984, ibid. 33, 123).

Wiersma, B. G. (2004). Environmental monitoring. Boca Raton, FL: CRC Press.

Woodfiel (1991). Appendix 1: Calculating p-values for Fisher's kappa. Retrieved July 13, 2007, from http://ftp.sas.com/techsup/download/observations/4q91/woodfiel/woodfiel.sas

APPENDICES

Appendix A: SAS Program for Producing Independent Seeds

```
/*****************************************************************
Program: PRODUCE_SEEDS

Parameters:

      Seed – The random number for starting the generator (value cannot be
      greater than 2147483646).

      Interval – Specifies one of the following seeding algorithms:

            Zero - The zero-leap algorithm is the simplest of the four
            algorithms and is the most economical with regard to value output
            (i.e., all values in the period can be used). However, it has the
            weakest sampling properties of the four algorithms (i.e., the
            starting seed is the only random component since successive
            values are in serial order). Moreover, many values could have no
            chance of inclusion if far enough away from the starting seed.

            Fixed - The fixed-leap algorithm provides better theoretical
            sampling properties than the zero-leap. The number of values
            leaped is computed by subtracting the period from the total
            number of values required then dividing by the number of seeds.
            In this way, the entire period is traversed by the algorithm
            (allowing all values a chance to be included).  Note that Madow
            (1946, p. 213) advised that “the chief danger in applying a
            systematic design occurs when the data have a periodic
            formation, and the sampling interval chosen is equal to the
            period of the data.”

            Scaled – The scaled random leap algorithm uses the maximum number
            of skips between seeds to scale the last uniform random number in
            a sample (say from 1 to 100) which then becomes the number of
            values leaped after that sample. Although this algorithm offers
            better random sampling properties than the zero or fixed-leap
            algorithms,it has the potential to exclude more values (reducing
            the number of values available for an application).

            Unscaled - Of the four algorithms considered, the unscaled
            random-leap provides the closest match to Mihram’s Principium of
            Seeding. A block of seeds produced by the generator become
            sampling numbers for the entire population of seeds. To prevent
            the chance that sampling numbers will produce overlapping seeds,
            the distance between each sampling number is computed and checked
            against the sample size parameter. If the sequences overlap, then
            that sampling number is rejected. (Note that extra sampling
            numbers are generated to allow for any rejects.)

      Simulation_design – An object-oriented SAS IML macro containing values
      for (a) the number of samples per experiment and (b) the size of each
      sample. Simply replace the existing values with a new design.
      Random_number_generator – An object-oriented SAS IML macro specifying
      values for the constants of the multiplicative congruential generator.

      Output_path – The file path for saving the seeds.

      Extra – Specifies the number of extra values to generate to allow for
      rejected seeds.

      Check – Checks if enough seeds (the extra parameter) have been
      specified (1=Yes, 0=No).

Notes:

      1. The main macro contains the experiments used to evaluate the
         performance of the four seeding algorithms. Simply replace these
         variables with a new design.

      2. The start and finish parameters of the main macro allow experiments
         to be run in parallel (if desired).

Author:

      Robert Grisham Stewart
      Claudius G. Clemmer College of Education
      Dept. of Educational Leadership and Policy Analysis
      East Tennessee State University
      phone: (423) 282-4124

Released:  August 2007

Changes:   None

References:

      Madow, L. H. (1946). Systematic sampling and its relation to other
            sampling designs. Journal of the American Statistical
            Association, 41(234), 204-217.

      Mihram, G. A. (1972). Simulation: Statistical foundations and
            methodology. New York: Academic Press.

/*************************************************************************/
%macro check_seed ;
   if (seed > ((2##31)-2)) then
      do ;
         print 'ERROR: Start seed cannot be greater than 2147483646' ,,
               '-----------------  Start Seed   ------------------' ,,
               seed[format=10.0] ;
         exit = 1 ;
      end ;
   if (seed < 1) then
      do ;
         print 'ERROR: Start seed cannot be less than 1' ,,
               '-----------------  Start Seed   ------------------' ,,
               seed[format=10.0] ;
         exit = 1 ;
      end ;
```

```
%mend ;
/*******************************************************************/
%macro check_totunifs ;
   if (totunifs > (2##31-2)) then
      do ;
         print 'ERROR: Total uniforms exceed period of 2147483646' ,,
               '----------------- Total Uniforms -----------------' ,,
               totunifs[format=10.0] ;
         exit = 1 ;
      end ;
%mend ;
/*******************************************************************/
%macro check_totskips ;
   if (totskips < totseeds) then
      do ;
         maxskips_per_seed = int(totskips/totseeds) ;
         print 'ERROR: Maximum skips per seed must be greater than 0' ,,
               '-------------- Maximum Skips Per Seed --------------' ,,
               maxskips_per_seed[format=10.0] ;
         exit = 1 ;
      end ;
%mend ;
/*******************************************************************/
%macro seed_nunifs ;
   do i=1 to ncol(nunifs) ;
      _nunifs_=_nunifs_//repeat(nunifs[,i],nsamples[,i],1) ;
   end ;
   _matx_=_seed_||_nunifs_ ;
   free _seed_ _nunifs_ ;
%mend ;
/*******************************************************************/
%macro print_report ;
   print '----------------- Interval Type ------------------' ,,
         %upcase("&interval.") ,,,,
         '-----------------  Output Path  ------------------' ,,
         %upcase("&output_path.") ,,,,
         '-----------------  Start Seed   ------------------' ,,
         seed1 ,,,,
         '----------------- Design Matices -----------------' ,,
         nsamples[format=10.0] ,, nunifs[format=10.0] ,,,,
         '----------------- Design Totals ------------------' ,,
         totseeds[format=10.0] totunifs[format=10.0]  ;
%mend ;
/*******************************************************************/
%macro matrix2data
     ( matx = ,
       data = ,
       vars = ) ;
      varname = &vars ;
      create &data
      from &matx [colname=varname] ;
      append from &matx ;
      close &data ;
%mend ;
/*******************************************************************/
%macro matrix2file
     ( matx = ,
       path = ,
       wd = ,
       s = ) ;
   start matrix2file(matx) ;
      file "&path." ;
         do i=1 to nrow(matx) ;
            do j=1 to ncol(matx) ;
               put (matx[i,j]) &wd &s @ ;
            end ;
            put ;
         end ;
      closefile "&path." ;
   finish ;
   run matrix2file(&matx) ;
%mend ;
/**********************************************************************/
%macro produce_seeds
     ( seed = ,
       interval = ,
       simulation_design = ,
       random_number_generator = ,
       output_path = ,
       extra = ,
       check = ) ;
   %local exit cnt err msg1 msg2 msg3 msg4 msg5 msg6 msg7 ;
      proc iml ;
         start matrix2macro(matx,mvar) ;
            if type(matx)='N' then
               matx=trim(char(matx)) ;
            call execute('%let ', mvar, '=', matx, ';') ;
         finish ;
         exit = 0 ;
         seed = &seed ;
         interval = %upcase("&interval.") ;
         %&simulation_design
         totseeds = sum(nsamples) ;
         totunifs = sum(nsamples#nunifs) ;
         totskips = ((2##31-2)-totunifs) ;
         %check_seed
         %check_totunifs
         %if (%upcase("&interval.") = "FIXED") |
             (%upcase("&interval.") = "SCALED") %then
            %check_totskips ;
         run matrix2macro(exit,{exit}) ;
      quit ;
   %if (&exit=0) %then
   %do ;
      %if (%upcase("&interval.") = "UNSCALED") %then
      %do ;
         proc iml ;
            exit = 0 ;
            err = 0 ;
            chk = &check ;
            seed = &seed ;
            %&simulation_design
            totseeds = sum(nsamples) ;
            totunifs = sum(nsamples#nunifs) ;
```

```
%if (%index(&extra,.)>0) %then
   %do ;
      allseeds=totseeds+int(totseeds*&extra) ;
   %end ;
%else
   %do ;
      allseeds=(totseeds+&extra) ;
   %end ;
s_all = repeat({.},allseeds,1) ;
s_tot = repeat({.},totseeds,1) ;
do i=1 to allseeds ;
   %&random_number_generator
   seed=u*((2##31)-1) ;
   s_all[i,]=seed ;
end ;
temp = s_all ;
temp[rank(s_all),] = s_all ;
s_all = temp ;
free temp ;
do i=1 to ncol(nsamples) ;
   x = x//repeat(nunifs[,i],nsamples[,i],1) ;
end ;
i=1 ;
j=2 ;
k=1 ;
do while(exit=0) ;
   dif = abs((s_all[j,]-s_all[i,])) ;
   if (dif >= x[k,]) then
      do ;
         s_tot[k,]=s_all[i,] ;
         i = j ;
         k = (k+1) ;
         if (k=totseeds) then
            exit = 1 ;
      end ;
   j=(j+1) ;
   if (exit=0) then
      if (j>nrow(s_all)) then
         do ;
            exit = 1 ;
            err = 1 ;
         end ;
end ;
if (err=0) then
   do ;
      s_tot[k,]=s_all[i,] ;
      s_last = s_all[i,] ;
      free s_all ;
      s_next = (s_last + x[totseeds,]) ;
      free x ;
      if ( s_next>((2##31)-2) ) then
         if ( (s_next-((2##31)-2))>s_tot[1,] ) then
            err = 1 ;
    end ;
if (chk=1) & (err=1) then
   print 'ERROR: Increase value of extra parameter.' ;
if (chk=1) & (err=0) then
```

```
         do ;
            nextra = (j-1) - totseeds ;
            print 'Number of Extra Seeds' ,, nextra ;
         end ;
      if (chk=0) & (err=0) then
         do ;
            %matrix2data(matx=s_tot,data=_s_,vars='_s_') ;
         end ;
   quit ;
   %if (%sysfunc(exist(_s_))=1) %then
      %do ;
         proc iml ;
            use _s_ ;
            read all into s ;
            call delete (work,_s_) ;
            seed1 = &seed ;
            seed = seed1 ;
            %&simulation_design
            totseeds = sum(nsamples) ;
            totunifs = sum(nsamples#nunifs) ;
            _seed_ = repeat({.},totseeds,1) ;
            cnt = 1 ;
            do i=s[1,] to s[totseeds,] ;
               if (i=s[cnt,]) then
                  do ;
                     _seed_[cnt,] = seed ;
                     cnt = (cnt+1) ;
                  end ;
               %&random_number_generator
               seed = u*((2##31)-1) ;
            end ;
            %seed_nunifs
            %matrix2file(matx=_matx_,path=&output_path,wd=10.0,s=+1)
            %print_report
         quit ;
      %end ;
   %else
      %do ;
         %let msg1 = The unscaled interval will produce ;
         %let msg2 = overlapping samples for the simulation design. ;
         %let msg3 = Try the following: ;
         %let msg4 = (a) Increase the value for the extra parameter ;
         %let msg5 = (b) Use a different start seed ;
         %let msg6 = (c) Use a different interval ;
         %let msg7 = (d) Alter the simulation design. ;
         %put ERROR: &msg1 &msg2 &msg3 &msg4 &msg5 &msg6 &msg7 ;
      %end ;
%end ;
%else
   %do ;
      proc iml ;
         seed1 = &seed ;
         seed = seed1 ;
         interval = %upcase("&interval.") ;
         %&simulation_design
         totseeds = sum(nsamples) ;
         totunifs = sum(nsamples#nunifs) ;
```

```
               totskips = ((2##31-2)-totunifs) ;
               if (interval = 'ZERO') then
                  do ;
                     maxskips_per_seed = 0 ;
                     nskips = 0 ;
                  end ;
               else
                  do ;
                     maxskips_per_seed = int(totskips/totseeds) ;
                     nskips = maxskips_per_seed ;
                  end ;
               _seed_=repeat({.},totseeds,1) ;
               cnt = 0 ;
               do i=1 to ncol(nsamples) ;
                  do j=1 to nsamples[,i] ;
                     cnt = (cnt+1) ;
                     _seed_[cnt,]=seed ;
                     do k=1 to (nunifs[,i] + nskips) ;
                        %&random_number_generator
                        seed = u*((2##31)-1) ;
                     end ;
                     if (interval ='SCALED') then
                        nskips = int(u*(maxskips_per_seed+1)) ;
                  end ;
               end ;
               %seed_nunifs
               %matrix2file(matx=_matx_,path=&output_path,wd=10.0,s=+1)
               %print_report
               print maxskips_per_seed[format=10.0] nskips[format=10.0] ;
            quit ;
         %end ;
   %end ;
%mend ;
/*********************************************************************/
%macro sim1 ;
   nsamples = repeat({10000},1,18) ;
   nunifs   = repeat({10},1,6)||repeat({50},1,6)||repeat({100},1,6) ;
%mend ;
%macro sim2 ;
   nsamples = repeat({10000},1,8) ;
   mult     = { 4  4  8  8  4  4  8  8} ;
   nobs     = {10 50 10 50 10 50 10 50} ;
   nunifs   = mult#nobs ;
%mend ;
/*********************************************************************/
%macro mcg32 ;
   u = (mod(16807*seed,2##31-1))/((2##31)-1) ;
%mend ;
%macro mcg64 ;
   u = uniform(seed) ;
%mend ;
/*********************************************************************/
%macro main(start=,finish=,dir=,ext=) ;
   %let s1 = 684543030 ;
   %let s2 = 1400370912 ;
   %let s3 = 2116198794 ;
   %let s4 = 123535106 ;
   %let s5 = 839362988 ;
   %let s6 = 1555190870 ;
   %let x1  = x1_sim1_mcg64_zero_&s1 ;
   %let x2  = x2_sim1_mcg64_zero_&s2 ;
   %let x3  = x3_sim1_mcg64_zero_&s3 ;
   %let x4  = x4_sim1_mcg64_fixed_&s1 ;
   %let x5  = x5_sim1_mcg64_fixed_&s2 ;
   %let x6  = x6_sim1_mcg64_fixed_&s3 ;
   %let x7  = x7_sim1_mcg64_scaled_&s1 ;
   %let x8  = x8_sim1_mcg64_scaled_&s2 ;
   %let x9  = x9_sim1_mcg64_scaled_&s3 ;
   %let x10 = x10_sim1_mcg64_unscaled_&s1 ;
   %let x11 = x11_sim1_mcg64_unscaled_&s2 ;
   %let x12 = x12_sim1_mcg64_unscaled_&s3 ;
   %let x13 = x13_sim2_mcg64_zero_&s4 ;
   %let x14 = x14_sim2_mcg64_zero_&s5 ;
   %let x15 = x15_sim2_mcg64_zero_&s6 ;
   %let x16 = x16_sim2_mcg64_fixed_&s4 ;
   %let x17 = x17_sim2_mcg64_fixed_&s5 ;
   %let x18 = x18_sim2_mcg64_fixed_&s6 ;
   %let x19 = x19_sim2_mcg64_scaled_&s4 ;
   %let x20 = x20_sim2_mcg64_scaled_s5 ;
   %let x21 = x21_sim2_mcg64_scaled_&s6 ;
   %let x22 = x22_sim2_mcg64_unscaled_&s4 ;
   %let x23 = x23_sim2_mcg64_unscaled_&s5 ;
   %let x24 = x24_sim2_mcg64_unscaled_&s6 ;

   %do i=&start %to &finish ;
      %produce_seeds
         ( simulation_design = %scan(&&&x&i,2,_) ,
           random_number_generator = %scan(&&&x&i,3,_) ,
           interval = %scan(&&&x&i,4,_) ,
           seed = %scan(&&&x&i,5,_) ,
           output_path = &dir.&&&x&i..&ext ,
           extra = 826 ,
           check = 1 )
   %end ;

%mend ;
%main(start=1,finish=24,dir=c:\,ext=.txt)
```

Appendix B: SAS Program for Evaluating Seeds

```
/*******************************************************************
Program:   EVALUATE_SEEDS

Parameters:

      Exp – Name of the file containing the seeds for given experiment.

      Dir – Specifies the directory holding the file (e.g., c:\)

      Ext – Specifies the extension of the file (e.g., .txt)


Notes:

      1. The main macro contains the experiments used to evaluate the
      performance of the four seeding algorithms. Simply replace these
      variables with a new design.

      2. The start and finish parameters of the main macro allow experiments
      to be run in parallel (if desired).

      3. The seral correlation and Fisher’s Kappa macro must be added to the
      program using the %include statement.


Author:

      Robert Grisham Stewart
      Claudius G. Clemmer College of Education
      Dept. of Educational Leadership and Policy Analysis
      East Tennessee State University
      phone: (423) 282-4124


Released:  August 2007

Changes:   None

References:

      Fan, X., Felsovalyi, A., Sivo, S. A., & Keenan, S. C. (2002). SAS for
            Monte Carlo studies: A guide for quantitative researchers. Cary,
            NC: SAS Institute.

      Woodfiel (1991). Appendix 1: Calculating P-Values for Fisher's Kappa.
            Retrieved July 13, 2007, from
            http://ftp.sas.com/techsup/download/observations/4q91/woodfiel/wo
            odfiel.sas

********************************************************************/
%macro put_data
     ( data = ,   /* name of data set */
       vars = ,   /* name of variable(s) */
       path = ,   /* path to file w/ dir. and folder(s)
```

```
                        [e.g., c:/sas/my_folder/ ] */
       name = ,   /* name of file w/o id (if any)
                        [e.g., file_1 would be file_] */
       id = ,     /* file id (if any)
                        [e.g., 1 ] */
       ext = ,    /* extension of file w/dot
                        [e.g., .sas, .txt ] */
       mode = ) ; /* mode for writing to file:
                        old = replaces contents
                            = replaces contents
                        mod = appends to contents */

   %if "&data." ne "%str()" and
       "&vars." ne "%str()" and
       "&path." ne "%str()" %then
      %do ;

         data _null_ ;
            set &data ;
            file "&path.&name.&id.&ext" &mode ;
            put &vars ;
         run ;

      %end ;
   %else
      %do ;
         %if "&data." eq "%str()" %then
            %put NOTE: [PUT_DATA] Value for required parameter "data" is
null. ;
         %if "&vars." eq "%str()" %then
            %put NOTE: [PUT_DATA] Value for required parameter "vars" is
null. ;
         %if "&path." eq "%str()" %then
            %put NOTE: [PUT_DATA] Value for required parameter "path" is
null. ;
      %end ;

%mend put_data ;
/***************************************************************************/
%macro seed2unif
     ( data = ,
       temp = ,
        out = ) ;
   %local i nsamples nobs ;
   %if (%sysfunc(exist(&out))=1) %then
      %delete_data(&out) ;
   %let nsamples=%nobs(&data) ;
   %printlog(no)
   %do i=1 %to &nsamples ;
      data &temp (keep=&out) ;
         set &data (firstobs=&i obs=&i) ;
         do i=1 to nunifs ;
            &out=ranuni(seed) ;
            output ;
         end ;
      run ;
      proc append out = &out
```

```
                  new = &temp ;
      run ;
   %end;
   %printlog(yes)
   %delete_data(&temp &data)
%mend ;
/**************************************************************************/
%macro nobs(dsname) ;
   %local dsid nobs rc ;
   %let dsid = %sysfunc(open(&dsname,i)) ;
   %let nobs = %sysfunc(attrn(&dsid,nobs)) ;
   %let rc = %sysfunc(close(&dsid)) ;
   &nobs
%mend ;
/*********************************************************************/
%macro delete_data
     ( datname ,
       libname = work ,
       memtypes = data ,
       opts = nolist ) ;
   proc datasets
      library = &libname
      memtype = (&memtypes) &opts ;
      delete &datname / memtype = &memtypes ;
   run ;
   quit ;
%mend ;
/*********************************************************************/
%macro printlog(action) ;
   %if (%upcase("&action.") = "YES") %then
      %do ;
         options source source2 notes %str(;)
      %end ;
   %else
      %do ;
         options nosource nosource2 non-otes %str(;)
      %end ;
%mend ;
/********************************************************************/
%macro runstest
     ( data = ,
       test = up ) ;
   %local i op1 op2 ;
   %let op1 = > ;
   %let op2 = < ;
   %if (%upcase("&test.") = "DOWN") %then
      %do ;
         %let op1 = < ;
         %let op2 = > ;
      %end ;
   proc iml ;
      use &data ;
      read all var _num_ into data ;
      %do i=1 %to 6 ;
         c&i = 0 ;
      %end ;
      n = nrow(data) ;
```

```
i = 1 ;
do while(i<n) ;
   if (data[i,] &op1 data[i+1,]) then
      do ;
         if (i=(n-1)) then
            c1=c1+2 ;
         else
            c1=c1+1 ;
         i=(i+1) ;
      end ;
   else
      do ;
         runlen=1 ;
         exit=0 ;
         do while(exit=0) ;
            if (data[i,] &op2 data[i+1,]) then
               do ;
                  runlen = (runlen+1) ;
                  i=(i+1) ;
                  if (i=n) then
                     exit = 1 ;
               end ;
            else
               do ;
                  exit=1 ;
                  i=(i+1) ;
                  if (i=n) then
                     c1=(c1+1) ;
               end ;
         end ;
         if (runlen=2) then
            c2 = (c2+1) ;
         else if (runlen=3) then
            c3 = (c3+1) ;
         else if (runlen=4) then
            c4 = (c4+1) ;
         else if (runlen=5) then
            c5 = (c5+1) ;
         else if (runlen>=6) then
            c6 = (c6+1) ;
      end ;
end ;
c = c1//c2//c3//c4//c5//c6 ;
free / c n ;
b1 = 1/6 ;
b2 = 5/24 ;
b3 = 11/120 ;
b4 = 19/720 ;
b5 = 29/5040 ;
b6 = 1/840 ;
b = b1//b2//b3//b4//b5//b6 ;
a ={ 4529.4 9044.9 13568  18091  22615  27892 ,
     9044.9  18097 27139  36187  45234  55789 ,
      13568  27139 40721  54281  67852  83685 ,
      18091  36187 54281  72414  90470 111580 ,
      22615  45234 67852  90470 113262 139476 ,
      27892  55789 83685 111580 139476 172860 } ;
      x = repeat({.},6,6) ;
      do i=1 to 6 ;
         do j=1 to 6 ;
            x[i,j] = (c[i,]-n*b[i,])*(c[j,]-n*b[j,])*a[i,j] ;
         end ;
      end ;
      s = sum(x) ;
      * print s ;
      v = 1/n*sum(x) ;
      p = probchi(v,6,0) ;
      * print n ,, a,, b,, c,, v,, p ;
      test = %upcase("&test.") ;
      if ( test = 'UP' ) then
         print 'RUNS UP TEST' ,, v p ;
      else
         print 'RUNS DOWN TEST' ,, v p ;
   quit ;
%mend ;
/***********************************************************/
%macro whitetest(data=,var=) ;
   proc spectra whitetest data=&data ;
      var &var ;
   run ;
%mend ;
/**********************************************************/
%include 'd:\corrtest.sas' ;
/**********************************************************/
%include 'd:\fisher_kappa_pvalue.sas' ;
/**********************************************************/
%macro evaluate_seeds
     ( exp = ,
       dir = ,
       ext = .txt ) ;

  %let path = &dir.&&&exp..&ext ;

  data seeds ;
    infile "&path" ;
    input seed nunifs ;
  run ;

  data seeds ;
     set seeds (firstobs=1 obs=1000) ;
  run ;

  %seed2unif( data = seeds ,
              temp = temp ,
              out = u )
  title1 "&&&exp" ;
  /*-------------------------------------------------*/
  %corrtest (data=u, var=u, hlag=100)
  %put_data ( data = work ,
              vars = pvalue ,
              path = c:\Documents and Settings\zrgs1\Desktop,
              name = &&&exp ,
              id   = _corr,
              ext  = .txt )
  /*-------------------------------------------------------*/
  title2 'Runs Up Test';
  %runstest (data=u, test=up)
  /*-------------------------------------------------------*/
  title2 'Runs Down Test';
  %runstest(data=u, test=down)
  /*-------------------------------------------------------*/
  title2 'White Noise Test' ;
  %whitetest(data=u, var=u)
  /*-------------------------------------------------------*/
  title2 'Fisher Kappa (White Noise Test)';
  %fisher_kappa_pvalue(data=u, var=u, out=fish)
  %put_data ( data = fish ,
              vars = period r z mz gprob ,
              path = c:\Documents and Settings\zrgs1\Desktop,
              name = &&&exp ,
              id   = _fish,
              ext  = .txt )
  /*--------------------------------------------------------*/
  proc datasets lib=work nolist memtype=data kill ;
  run; quit;

%mend ;
/**********************************************************/
%macro main(start=,finish=,dir=,ext=);
   %let s1 = 684543030 ;
   %let s2 = 1400370912 ;
   %let s3 = 2116198794 ;
   %let s4 = 123535106 ;
   %let s5 = 839362988 ;
   %let s6 = 1555190870 ;
   %let x1  = x1_sim1_mcg64_null_&s1 ;
   %let x2  = x2_sim1_mcg64_null_&s2 ;
   %let x3  = x3_sim1_mcg64_null_&s3 ;
   %let x4  = x4_sim1_mcg64_fixed_&s1 ;
   %let x5  = x5_sim1_mcg64_fixed_&s2 ;
   %let x6  = x6_sim1_mcg64_fixed_&s3 ;
   %let x7  = x7_sim1_mcg64_scaled_&s1 ;
   %let x8  = x8_sim1_mcg64_scaled_&s2 ;
   %let x9  = x9_sim1_mcg64_scaled_&s3 ;
   %let x10 = x10_sim1_mcg64_unscaled_&s1 ;
   %let x11 = x11_sim1_mcg64_unscaled_&s2 ;
   %let x12 = x12_sim1_mcg64_unscaled_&s3 ;
   %let x13 = x13_sim2_mcg64_null_&s4 ;
   %let x14 = x14_sim2_mcg64_null_&s5 ;
   %let x15 = x15_sim2_mcg64_null_&s6 ;
   %let x16 = x16_sim2_mcg64_fixed_&s4 ;
   %let x17 = x17_sim2_mcg64_fixed_&s5 ;
   %let x18 = x18_sim2_mcg64_fixed_&s6 ;
   %let x19 = x19_sim2_mcg64_scaled_&s4 ;
   %let x20 = x20_sim2_mcg64_scaled_s5 ;
   %let x21 = x21_sim2_mcg64_scaled_&s6 ;
   %let x22 = x22_sim2_mcg64_unscaled_&s4 ;
   %let x23 = x23_sim2_mcg64_unscaled_&s5 ;
   %let x24 = x24_sim2_mcg64_unscaled_&s6 ;

   %do i=&start %to &finish;
```

```
      %evaluate_seeds ( exp = x&i ,
                        dir = &dir ,
                        ext = &ext )
   %end ;

%mend ;

%main(start=1,finish=24,dir=c:\,ext=.txt)
```

Appendix C: SAS Program for Spacing Start Seeds

```
/*******************************************************************
Program:   SPACE_SEEDS

Parameters:

      Nseeds – The number of seeds to be spaced.

      Path – Specifies the directory holding the file (e.g., c:\)
             to include file name and extension.


Notes: None

Author:

      Robert Grisham Stewart
      Claudius G. Clemmer College of Education
      Dept. of Educational Leadership and Policy Analysis
      East Tennessee State University
      phone: (423) 282-4124


Released:  August 2007

Changes:   None

References: None
********************************************************************/

%macro names(prefix=, n=) ;
   %local i ;
   %do i=1 %to &n ;
      &prefix&i
   %end ;
%mend ;

%macro enter_seeds ;
   %window enter_seeds
      #9  @10 "Count of Seed Sets........ &count"
      #10 @10 "Number of Seeds per Set... &nseeds"
      #12 @10 "INSTRUCTIONS:"
      #14 @10 "1. Press enter to activate the cursor."
      #16 @10 "2. Select an option from below:"
      #18 @13 "A. To create a set of equally spaced seeds, Type a 10 digit
seed value."
      #19 @13 "B. To save the seed set(s) and exit the program type 0."
      #21 @10 "3. Press enter to accept value."
      #24 @10 seed 10 ;
%mend ;

%window err_msg
   #10 @10 "ERROR: Seed value cannot be greater tahn 8589934584."
   #12 @10 "Press enter to continue.";
```

```
%macro space_seeds (nseeds=,path=) ;
   %local i j rc seed count ;
   %let count = 0;
   %if (%sysfunc(exist(out_seeds))=1) %then
   %do ;
      proc datasets ;
         delete out_seeds ;
      run ;
      quit ;
   %end ;
   %enter_seeds
   %display enter_seeds ;
   %do %while (&seed^=0) ;
      data new_seeds (drop= rc nskips) ;
         seed1=%trim(%left(&seed)) ;
         if (seed1 <= 8589934584) then
         do ;
            rc = 1 ;
            do while (seed1>2147483646) ;
               seed1 = (seed1 - 2147483646) ;
            end ;
            nskips = 2147483646/&nseeds ;
            %do i=2 %to &nseeds ;
               %let j=%eval(&i-1) ;
               seed&i = seed&j + nskips ;
               if (seed&i > 2147483646) then
                  seed&i=(seed&i-2147483646) ;
            %end ;
         end ;
         else
            rc = 0 ;
         call symput('rc',trim(left(rc))) ;
      run;
      %if (&rc=1) %then
         %do ;
            proc append out = out_seeds
                        new = new_seeds ;
            run ;
            %let count = %eval(&count+1) ;
         %end ;
      %else
         %do ;
            %let seed = _ ;
            %display err_msg ;
         %end ;
      %enter_seeds
      %display enter_seeds ;
   %end ;
   %if (%sysfunc(exist(out_seeds))=1) %then
   %do ;
      data _null_ ;
         set out_seeds ;
         file "&path" ;
         put %names(prefix=seed,n=&nseeds) ;
      run ;
   %end ;
%mend ;
```

```
/*--------------------------------------------------------*/

%space_seeds
   ( nseeds = 3 ,
     path = c:\equal_seeds.txt )
```

Appendix D: SAS Program for the Intraclass Correlation Coefficient

```
%macro icc (data=,classvar=,depvar=) ;
   ods output overallanova=MS ;
   proc anova data=&data ;
      class &classvar ;
      model &depvar = &classvar ;
   run ;
   ods output close ;
   data msb ;
      set ms (firstobs=1 obs=1) ;
      msb = ms ;
   run ;
   data msw ;
      set ms (firstobs=2 obs=2) ;
      msw = ms ;
   run ;
   data icc ;
      merge msb msw ;
      m=6 ;
      icc = (msb-msw) / (msb+(msw*(m-1))) ;
      keep dependent icc  ;
   run ;
   proc print data = icc ;
      var dependent icc ;
   run ;
%mend ;
%icc (data=diss,classvar=alg,depvar=m_st)
%icc (data=diss,classvar=alg,depvar=m_ru)
%icc (data=diss,classvar=alg,depvar=m_bks)
```

Appendix E: SAS Dataset of Study p-Values

```
data diss ;
   input sim mcg seed alg $ m_st m_ru m_rd m_bks ;
   m_st_bc = (m_st**1.3) ;
   m_ru_bc = (m_ru**0.8) ;
   m_bks_bc = (m_bks**0.75) ;
   label m_st_bc  = 'Serial Correlation'
         m_ru_bc  = 'Runs Up'
         m_bks_bc = 'White Noise' ;
cards ;
1 64 68 1 0.5179 0.1397 0.4767 0.3052
1 64 68 2 0.5542 0.3358 0.5568 0.8079
1 64 68 3 0.5276 0.2848 0.8792 0.8692
1 64 68 4 0.4698 0.4528 0.5123 0.2369
1 64 14 1 0.5146 0.6239 0.7657 0.0481
1 64 14 2 0.5200 0.0623 0.0132 0.9683
1 64 14 3 0.4497 0.4971 0.3018 0.6828
1 64 14 4 0.5167 0.8107 0.6228 0.0854
1 64 21 1 0.4488 0.8130 0.8243 0.7354
1 64 21 2 0.5283 0.2307 0.0603 0.8045
1 64 21 3 0.5321 0.3583 0.9274 0.5135
1 64 21 4 0.5204 0.9590 0.5678 0.3625
2 64 12 1 0.5154 0.4423 0.3056 0.6418
2 64 12 2 0.5059 0.9281 0.8522 0.6035
2 64 12 3 0.5066 0.3351 0.1830 0.0645
2 64 12 4 0.5329 0.8253 0.1178 0.5454
2 64 83 1 0.4765 0.5866 0.3498 0.2887
2 64 83 2 0.5612 0.6197 0.0531 0.9720
2 64 83 3 0.4707 0.2734 0.3348 0.0351
2 64 83 4 0.4840 0.1689 0.1985 0.0312
2 64 15 1 0.5113 0.6879 0.4840 0.7808
2 64 15 2 0.5160 0.0226 0.4592 0.9771
2 64 15 3 0.5110 0.6270 0.4946 0.7253
2 64 15 4 0.4733 0.8821 0.9720 0.1262
run ;
```

VITA

ROBERT G. STEWART

Personal Data:	Date of Birth: April 24, 1969 Place of Birth: Bristol Virginia Martial Status: Single
Education:	East Tennessee State University, Johnson City, Tennessee; Educational Leadership and Policy Analysis, Ed.D., 2007 East Tennessee State University, Johnson City, Tennessee; Engineering Technology, M.S., 1996 East Tennessee State University, Johnson City, Tennessee; Engineering Technology, B.S., 1992 Public Schools, Jonesborough, Tennessee
Professional Experience:	Doctoral Fellow, East Tennessee State University, 2003 Instructor, Virginia Intermont College; Bristol, Virginia, 2001 Graduate Assistant, East Tennessee State University, College of Applied Science and Technology, 1997-2000 Research Assistant, Tennessee Institute for Economic Development, 1997 Research Intern, Department of Human Resources, Sprint, 1996 Graduate Assistant, East Tennessee State University, College of Applied Science and Technology, 1994-1996
Publications:	Acknowledged in Bonett, Douglas G. and Price, Robert M. (2005). "Inferential Methods for the Tetrachoric Correlation Coefficient." Journal of Educational and Behavioral Statistics, 39(2), pp. 1-13.
Honors and Awards:	Phi Kappa Phi-Honor Society (April 1996) Epsilon Pi Tau Honor Society (April 1996) Gamma Beta Phi Honor Society (December 1995) Dean's List, East Tennessee State University, 1989