Monte Carlo Simulation Design for Evaluating Normal-Based Control Chart Properties

Article *in* Journal of modern applied statistical methods: JMASM · January 2015

DOI: 10.22237/jmasm/1478003580

1 author:

John N Dyer
Georgia Southern University
27 PUBLICATIONS **91** CITATIONS

Journal of Modern Applied Statistical Methods

Volume 15 | Issue 2 Article 35

11-1-2016

Monte Carlo Simulation Design for Evaluating Normal-Based Control Chart Properties

John N. Dyer
Georgia Southern University, jdyer@georgiasouthern.edu

Journal of Modern Applied Statistical Methods
November 2016, Vol. 15, No. 2, 580-626.
doi: 10.22237/jmasm/1478003580

ISSN 1538 − 9472

Monte Carlo Simulation Design for Evaluating Normal-Based Control Chart Properties

John N. Dyer
Georgia Southern University
Statesboro, GA

The advent of more complicated control charting schemes has necessitated the use of Monte Carlo simulation (MCS) methods. Unfortunately, few sources exist to study effective design and validation of MCS methods related to control charting. This paper describes the design, issues, considerations and limitations for conducting normal-based control chart MCS studies, including choice of random number generator, simulation size requirements, and accuracy/error in simulation estimation. This paper also describes two design strategies for MCS for control chart evaluations and provides the programming code. As a result, this paper hopes to establish de facto MCS schemes aimed at guiding researchers and practitioners in validation and control-chart evaluation MCS design.

Keywords: Monte Carlo simulation, statistical process control, random number generation

Introduction

Various control charts exist using a control chart statistic based on the Normal distribution, including the Shewhart, R-Chart, Individuals, S-Chart, Cumulative Sum (CUSUM), Exponentially-Weighted Moving Average (EWMA), Combined EWMA-Shewhart (CES), and Reverse Moving Average (RMA), among others. The performance of many control charts have been investigated using various analytical and numerical methods such as integral equations, saddle-point approximations, and Monte Carlo Markov Chain (MCMC) methods. Monte Carlo simulation (MCS) has also grown in popularity due to the relative ease of programmatic design, and the ability to investigate additional important performance measures of a control chart such as the median run-length (MRL), run-length quantiles, and the cumulative distribution function (CDF). Unfortunately

Dr. Dyer is a Professor of Information Systems. Email him at: jdyer@georgiasouthern.edu.

there are few sources available for researchers and practitioners to study effective design and validation of MCS methods related to control charting.

In general, MCS includes a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results, then running multiple simulations to obtain the distribution of an unknown probabilistic entity. MC methods are used in physical and mathematical problems and are useful when it is difficult or impossible to obtain a closed-form expression, or infeasible to apply a deterministic algorithm. MC methods are mainly used in three distinct problems classes: optimization, numerical integration, and studying probability distributions of random variables. The use of random numbers as input is a defining feature of MCS. This is what turns a deterministic model into a stochastic model. Regardless of the application, simulations of this kind should be impartial, systematic and reproducible. Sources of error need to be controllable or at least isolatable. The basic steps of conducting a Monte Carlo simulation can be summarized as follows (Salleh, 2013).

1. Create a model with appropriate parameters and assumptions.
2. Generate random numbers as inputs to the model.
3. Run the simulation and record the results and desired outputs.
4. Analyze the results with statistical and/or advanced modeling tools.

Although there is no strict definition MCS, in the field of statistical process control (SPC) and control charting, and for the purposes of this paper, it is broadly defined as the use of a programmatic pseudo-random number generation replicating repeated sampling from an assumed underlying statistical distribution, for the purposes of numerical integration of a function of a control-charting statistic and/or studying run-length (RL) properties and performance of the control chart. As such, there are several considerations related to control-charting MCS, including the choice, series length, and precision of the random number generator (RNG), as well as the required simulation size and expected accuracy/error in simulation estimation.

The advantage of using MCS in this manner allows one to more fully investigate the RL properties and performance of a control chart, over a wider array of performance measures including the average run-length (ARL), MRL, standard error of the run-length (SRL) and the CDF of the run-length, as well as percentiles and quartiles. The CDF measures the cumulative proportion or percent of signals given by the i^{th} period following the shift. It should be noted that the CDF completely characterizes the run length distribution, while the ARL is only the mean. Additionally, the MRL can be used in conjunction with the ARL and CDF

since it is a better measure of central tendency for skewed distributions such as the run length distribution. The MRL is defined as the median (50^{th} percentile) number of sampling periods until the control chart signals. Although traditional analytical and numerical methods such as integral equation and saddle-point approximations, as well as MCMC methods, provide good estimates of ARLs at specified control limits (CLs) of a control chart, the methods can be very cumbersome, mathematically complicated, and do not readily allow wider studies of RL properties and performance measures beyond the ARL, or simultaneous evaluation over a wide range of CLs. Additionally, in most cases the MCS can provide equal or better estimates than the traditional methods. One can also use MCS to validate findings based on other methods mentioned above.

Many researchers and practitioners involved in SPC and control charting design and implementation are very familiar with Microsoft Excel and use it extensively for analysis and modeling, regardless of the inherent problems known to exist in Excel and the RNGs employed in Excel (Ahrens & Dieter, 1988; Knusel, 2002; Benneyan, Lloyd, & Plsek, 2003). Additionally, Excel has many built-in functions as well as offering the user a Visual Basic for Applications (VBA) interface for programming in Excel. Excel can also be used as a prototype or beta MCS for initial studies prior to full implementation. As such, this paper does not have the purpose to intensely discuss the advantages, disadvantages, similarities, differences, etc., regarding analytical/numerical/MCMC methods versus MCS, nor is it the purpose to compare and contrast MCS using a myriad of possible programming languages, such as R, Visual Basic, C+, Java, FORTRAN, etc. Instead, this paper's purpose is to describe the basic validation design, issues, considerations and limitations for normal-based control chart MCS design, including choice of RNGs, RNG series length, MCS simulation size, and accuracy/error in MCS estimation, while exemplifying using Excel 2010. The design principles can be easily extended to other programming languages and in a variety of field requiring simulation. It is assumed that the reader is familiar with basics of elementary statistics, control charting and the normal distribution. For detailed introductions to these ideas, the reader is referred to (NIST/SEMATECH, 2012; Montgomery, 1996; Ryan, 2000; Wheeler & Chambers, 1992).

Normal Based Control Charts and Performance Criteria

SPC techniques have been used for decades to monitor and control a process, most often a manufacturing process, but are seeing increased use in fields broadly related to health care (Benneyan et al., 2003; Srinivasan, 2011), information technology

(Abdel-Aziz, Abdel & Darwis, 2008), finance (Golosnoy & Schmid, 2007; Severin & Schmid, 1998), and business process monitoring and improvement (Jaing, Au, & Tsui, 2007). The idea is to plot data (a control chart statistic) over time to aid in determining trends or changes in the process variability. In any process there exists a certain amount of inherent, common cause variability. This common cause variability is usually small, yet unavoidable. In contrast, variability from assignable causes is generally large, and can usually be removed from the process if detected. The primary use of control charts is to detect any assignable causes or process changes as quickly as possible, thus enabling quick action in elimination of the assignable cause.

Control charts can be used to monitor many aspects of the process, but the most common use is for monitoring the process mean and/or variance. To monitor the mean, individual observations or averages (or functions of) are plotted over time, where these plotted values are estimates of the process mean. Likewise, sample ranges or standard deviations are plotted against time as estimates of process variability.

When evaluating control chart performance, the ARL has typically been used to quantify performance of the chart. The ARL is defined as the average number of time periods until the control chart signals, and can be defined for both the in-control (IC) and out-of-control (OC) cases. A more recent alternative performance criterion is the CDF (Dyer & Barrett, 2000; Dyer, Conerly, Adams, & Barrett, 2002; Dyer, Conerly, & Adams, 2003; Dyer, Adams, & Conerly, 2003; Lin & Adams, 1996) and MRL. The CDF measures the cumulative proportion or percent of signals given by the ith period following the shift. It should be noted that the CDF completely characterizes the RL distribution, while the ARL is only the mean. Additionally, the MRL can be used in conjunction with the ARL and CDF since it is a better measure of central tendency for skewed distributions such as the RL distribution (Gan, 1993). The MRL is defined as the median (50^{th} percentile) number of time periods until the control chart signals.

Although there are a myriad of control charting schemes, many are based on an assumption of plotting a control chart statistic related to individual measures or the mean of a subgroup of measures against control-limits that are a function of the normal distribution. These control charts include the Shewhart (Shewhart, 1931/1980; Shewhart, 1939/1986; Roberts, 1959) and Individuals charts (with and without runs-rules) (Nelson, 1984; Western Electric Company, 1956), Multiple-Sampling schemes (Daudin, 1992; He, Grigoryan, & Sigh, 2002; Irianto & Shinozaki, 1998; Teoh & Khoo, 2012; Torng & Lee, 2009), the CUSUM (Page, 1954), EWMA (various schemes) (Hunter, 1986; Ryan, 2000) CES (Lucas &

Saccucci, 1990), and RMA (Dyer, Adams, & Conerly 2003) charts, among others. In many cases a measure of variability is also charted. The Shewhart and Individuals charts are straightforward in determining control-limits and control chart performance since the distribution of the IC and OC sequentially plotted statistics are assumed to follow an independent and identical Normal distribution (iidN), hence the RL properties can be studied using the Geometric distribution. It should be noted that the IC and OC processes follow different iidN distributions.

For example, assuming an IC process, using a Shewhart chart and $\pm 3\sigma$ CLs, the probability of a signal (p) is 0.0027, that is, the probability the plotted statistic exceeds the CLs (although it is a false-alarm), corresponding to ARL = 370. Since the count of the number of sampling periods until a false-alarm occurs (the run length) follows a Geometric distribution (Chen, 1997), the ARL, SRL and MRL are given by

$$\mu = \text{ARL} = \frac{1}{p} \tag{1}$$

$$\sigma = \text{SRL} = \frac{\sqrt{1-p}}{p} \tag{2}$$

$$\text{Med} = \text{MRL} = \frac{\ln(0.50)}{\ln(1-p)} \tag{3}$$

Solving (1), (2), and (3) for $p = 0.0027$ yields ARL = 370, SRL = 370, and MRL = 256. For cases in which consecutive observations are independent, for the Geometric distribution, $\mu \approx \sigma$, that is, ARL ≈ SRL. This result is not true of several common control charting methods such as the EWMA, CUSUM and RMA. Additionally, for the Geometric distribution the first quartile (Q_1) and third quartile (Q_3) RLs are given by

$$Q_1 = \text{QRL}_1 = \frac{\ln(0.75)}{\ln(1-p)} \tag{4}$$

$$Q_3 = \text{QRL}_3 = \frac{\ln(0.25)}{\ln(1-p)} \tag{5}$$

Solving (4) and (5) following the above example yields $QRL_1 = 106$ and $QRL_2 = 512$. In general, the p^{th} percentile run-length value is given by

$$RL_{p^{th}p} = \frac{\ln\left(1 - p^{th}\text{percentile}\right)}{\ln\left(1-p\right)} \tag{6}$$

Now, assume a shift of 1σ in the process mean. The probability of a signal is now $p = 0.0228$. In this case, the ARL ≈ 44, SRL ≈ 43, MRL ≈ 30, $QRL_1 \approx 12$, and $QRL_3 \approx 60$. The IC and OC processes now follow different iidN distributions following the shift in the process mean of the underlying distribution; hence the RL distributions are different iid Geometric (iidG) distributions.

Although the underlying processes used for the CUSUM, EWMA and RMA control chart statistics are assumed to be iidN, the sequentially plotted control-chart statistics are not independent since they are a form of cumulative sums or averages; hence the RL distribution cannot be exactly described as above. The Shewhart chart is said to have "no memory" while the CUSUM, EWMA, RMA (and others) are said to have "memory" (Kalgonda, Koshti, & Ashokan, 2011). Memory refers to if the control-chart statistic uses past data from a previous sampling period. It is known that CLs for control-charts with memory are much different than for the underlying iidN process. In this case of the lack of independence (memory), it can be difficult to use the traditional analytical or numerical methods to study RL properties and control chart performance, hence many MCS studies have been conducted to determine appropriate CLs as well as control chart performance measures (Dyer, Conerly, & Adams, 2003; Fu & Hu, 1999; Lin & Adams, 1996).

Additionally, MCS can also be employed to more readily determine overall RLs when multiple charts are used to monitor the same process, such as simultaneous use of charts for the mean and variability. Most of the aforementioned control-charts also have very limited tabulations of IC and OC ARLs, and sparse literature regarding other important measures like the MRL, percentiles/quartiles, SRL, CDF, or RL distribution studies.

Designing the Validation MCS for Control Chart Evaluation

One advantage of using the design exemplified in this paper is the ability to produce the RL distribution for many different sets of CLs simultaneously. For example, most MCS programs allow specification of a single set of CLs, thus calculating a single set of performance measures (ARL, MRL, SRL, CDF, etc.) from the

resulting simulated run-lengths. Alternatively, this design allows specification of up to 16,380 different sets of CLs, thus calculating as many different sets of summary measures. This would be an absurd case, but it reflects the capability of the design and the ability to use a single simulation run across all desired CLs.

Regardless of the programming language, software, or RNG being used, this author proposes two different designs of a MCS for control charting the mean or individual observations based on an underlying iidN process, as described in the 4 steps below. The two designs, D1 and D2 respectively, differ only in step 4 discussed below. In D1, a series of length *m* random numbers is independently generated *nsim* times, where *nsim* = the number of simulation runs. Then, one RL is recorded for each separate simulation, resulting in an array of *nsim* RLs. For example, setting *m* = 14,000 and *nsim* = 10,000, each independent series of length 14,000 is produced 10,000 times, resulting in 10,000 recorded RLs. In D2, a single very long series of size *m* is generated only one time. The number of RLs within the m random numbers is recorded. The optimal values of *m* and *nsim* for D1 and the optimal value of *m* and expected number of RLs are discussed in a subsequent section.

The basic design steps can be easily modified to accommodate a myriad of various control charting schemes, such as monitoring mean/variability simultaneously, or employing two charts for the mean simultaneously, like the CES control chart, and schemes based on runs-rules and those such as double-sampling. The MCS design steps are as follows.

1. Use a RNG to generate a series of length m, of subsets of size n of pseudo-random iidN variables ($n = 1, 2, 3\ldots$) representing the simulated values of the underlying iidN process (x_i), $i = 1$ to m. The variable n represents a subgroup size from which the appropriate statistic is calculated, such as the subgroup mean. So, the result will be a series of m means of subset size n. It is recommended that each of the m means be standardized to represent the Standard Normal distribution, that is, $z \sim N(\mu = 0, \sigma = 1)$ where z_i is the standardized subgroup mean. Note, that although $n = 1$ for the Individuals chart, n can also be 1 for the CUSUM and EWMA, but is usually ≤ 10.

2. Transform the series of calculated *z* statistics from step 1 to a series of control-chart statistics (CCS), appropriate to the control chart to be studied, e.g., CUSUM, EWMA, RMA, etc.

3. Establish upper/lower CLs (or a range of CLs) by which to compare the series of CCS from step 2.

For Design 1:

4. For the IC process, compare each sequential CCS from step 2 to the CLs established in step 3. When the first occurrence of a CCS exceeds one of the CLs, record the RL as $RL = i$, that is, the location within the series of m statistics wherein CCS exceeded CL. Stop step 4.

Following step 4, repeat steps 1 through 4 *nsim* times (where *nsim* is the number of simulations) and calculate the summary measures like ARL, MRL, SRL, and desired percentiles/quartiles. This will produce a total of *nsim* RLs for each set of estimates. For the simulated OC process, induce a step-shift in the mean of the process in step 1, e.g., a 1σ shift where σ is the process standard deviation. Again, complete the 4 steps *nsim* times and calculate the summary measures, including the CDF.

For Design 2:

4. For the IC process, compare each sequential CCS from step 2 to the CLs established in step 3. When an occurrence of a CCS exceeds one of the CLs, record the RL as $RL = i$, that is, the location within the series of m statistics wherein CCS exceeded one of the CLs. After recording the first *RL*, re-index the series so that next value in the series is 1, and then continue step 4, re-indexing and recording each subsequent *RL* until the entire series has been evaluated. In this case, the series-length m is much longer than D1.

After evaluating the entire series calculate the summary measures like ARL, MRL, SRL, CDF and desired percentiles/quartiles. This will produce an unknown but predictable number of sets of RLs for each set of estimates. For the simulated OC process, induce a step-shift in the mean of the process in step 1, e.g., a 1σ shift where σ is the process standard deviation. Again, complete the 4 steps and calculate the summary measures, including the CDF.

MCS Design Considerations

Because the RL distribution for the N(μ, σ) Shewhart/Individuals control charts is well known, one should first design a validation MCS by which to compare estimated results with known results. There are several important considerations to be made when designing the MCS according to the section above, including the choice of RNGs, the RNG series length and burn-in period, the choice of CLs, the number of simulations to conduct, and the resulting accuracy or error of the MCS. Since we already know the RL distribution of the Shewhart and Individuals chart, any MCS should begin by first validating the study against what is known about the IC $z \sim N(\mu = 0, \sigma = 1)$ process. That is, before transforming data into the CCS for the CUSUM, EWMA and RMA (or others), one should first test the MCS design against known properties and RL distribution of a Shewhart or Individuals chart. If the validation design is adequate, then one can make better assumptions regarding the adequacy of the design when modified for other control charting schemes. Although the literature reflects results using MCS for many control chart studies, few if any provide information regarding the estimated accuracy of results (estimated error) or degree of confidence in estimates, and few adequately describe the MCS design, the RNGs used or a justification of simulation size.

Additionally, when extending the validation design to other control charts, one should first validate at least a few summary measures from the MCS against what is already known in the literature. As such, the topics discussed in the sub-sections should first be applied against the known Shewhart or Individuals results.

Choosing the Random Number Generator

As stated in design step 1 in a previous section, one must use a RNG to generate a series of pseudo-random numbers from an assumed distribution. This can be done by calling an existing RNG function (as in Excel), or employing existing and validated RNG subroutines (e.g., the IMSL subroutine library) in languages such as FORTRAN, C, C++, Java, etc., or by writing one's own RNGs using known and validated algorithms. Note also, that a RNG is more appropriately called a pseudo-random number generator (PRNG), as it is an algorithm for generating a sequence of numbers that approximates the properties of random numbers that are "sufficiently random" to suit the intended use. Regardless of the RNG used, it should meet at least some of the statistical tests of randomness, and have a period-length long enough to not repeat a value in a very long string of pseudo-randomly generated values. Additionally, a good RNG should allow one to set a starting seed, allowing replication of results, and the RNG should have a known period-length.

In any event, the RNG must be implemented programmatically and requires some knowledge of programming.

Methods commonly used for the Normal distribution include the Ziggurat method (Marsaglia & Tsang, 2000), the Box-Muller transform (Box & Muller, 1958), the Marsaglia polar method (Marsaglia & Bray, 1964), the Probit function method, the Abramowitz & Stegun algorithm (Abramowitz & Stegun, 1972), a recent algorithm by Acklam (2014), and methods known as Kinderman-Ramage (Kinderman & Ramage, 1976), and Ahrens-Dieter (Ahrens & Dieter, 1988). Of these, the Ziggurat algorithm is considered superior, but the Box-Muller transform is very good and is a very common implementation in many software and programming languages. Many other algorithms are available for both the normal and other distributions (Korn, Korn, & Kroisandt, 2010; C. Roberts & Casella, 1999).

The task is then to use a Uniform [0, 1] RNG to randomly generate values of p, then solve the desired distribution's cdf for values of x. Common uniform RNGs include implementations using the Wichmann-Hill (WH) method (Wichmann & Hill, 1982), the Mersenne Twister algorithm (MT19937) (Matsumoto & Nishimura, 1998), the Marsaglia Multiply-with-carry (MWC) method (Marsaglia, Zaman, & Marsaglia, 1994), and other various methods in the classes of linear feedback shift register generators and linear congruential generators.

Using Excel 2010 VBA to replicate a series of iidN pseudo-random variates requires use of two built in functions; RND and NORM_INV. The RND function returns a floating-point Uniform [0, 1] random real number, representing a probability (p), where $0 \leq p < 1$. The function has no arguments and the output depends on the initial seed, which is set as a function of the system clock, hence not replicable. Although Microsoft claims to have implemented the Wichmann-Hill generator for the RND function, there are many finding that it was implemented incorrectly, and that it does not pass the DIEHARD test (McCullough & Heiser, 2008). Others suggest that for long periods, RND will create negative numbers, and the period-length is not exactly known. The RND function also returns the value 0. The NORM_INV function returns the inverse of the normal cumulative distribution with specified mean and standard deviation. That is, given a value for p, NORM_INV(p, μ, σ) seeks that value x such that the function NORM_DIST(x) = p. The value of p is the output from the RND function. The NORM_INV function is implemented using the Probit function method. It should be noted that Microsoft claims that that accuracy of the NORM_INV function depends on the accuracy of their NORM_DIST function (which uses the Abramowitz & Stegun algorithm), and the quality of the search procedure in its ability to "home in on" the value of x that

corresponds to the supplied value of p (Microsoft, 2011). They further claim that the accuracy is up to 15 or 16 decimal places. Excel with VBA is a good software and programming platform, and this author has been satisfied with the results of implementing the above Excel functions as well as the MT19937 for the Uniform [0, 1] RNG, which passes the DIEHARD test, and using the Box-Muller transform for the Normal distribution.

Calculating the Series Length

The series length m is the total number of N(μ, σ) random numbers (x) or subgroup averages to be generated and examined sequentially against the CLs in each simulation run. Again, the series of random numbers or subgroup averages should be standardized to represent a $z \sim N(\mu = 0, \sigma = 1)$ distribution. For any given desired ARL estimate, m will be constant, and is chosen as a function of the upper percentiles of the related Geometric distribution.

In D1, the total series length m should be long enough for at least one of the randomly generated z-values to exceed the CLs based on the desired in-control ARL. For example, a Shewhart chart with $\pm 3\sigma$ CLs ($p \approx 0.0027$), the ARL $\approx$ 370. Based on a known Geometric RL distribution with μ = ARL = 370, the 99th percentile value of the RL distribution is 1,702. So, setting m = 1,702, it would be expected that with great probability, at least one of the 1,702 z-values will exceed the CLs. Choosing the 99.9th percentile results in m = 2,554, and the 99.99th percentile results in m = 3,406. Of course, larger ARLs (corresponding to smaller values of p), like ARL = 1,000 (p = 0.001), the series lengths for the 99th, 99.9th, 99.99th and 99.9999th percentiles are m = 4,602, m = 6,903, m = 9,205, and m = 13,808, respectively. For iidN cases it is recommended that the 99.9999th percentile value be chosen for m to avoid the case of any simulation run failing to result in a recordable run-length. This almost certainly ensures that each series m will produce a recordable RL. When designing a MCS to evaluate a range of desired CLs and corresponding ARLs, it is recommended that (7) below be used to select the series length to accommodate the largest expected ARL estimation (eARL) in the study. Note that the multiple of 14 corresponds to the 99.9999th percentile. Alternatively, a multiple of 11 would result in the 99.999th percentile, a multiple of 9 would result in the 99.99th percentile, and so on.

$$m = \text{Max}\left(e\text{ARL}\right) * 14 \tag{7}$$

For example, if a study were performed using multiple CLs corresponding to ARLs of 1000, 900, 800, 700,…, 100, then Max(eARL) = 1000 and m = 14,000. Keep in mind that the series of length m will be generated and evaluated *nsim* times. For example, if m = 14,000 and *nsim* = 10,000, there would be 10,000 RLs recorded for each set of CLs in the completed simulation.

The greater issue of determining m arises when the sequential values of the transformed plotted statistics, CCS, are not independent, like for the CUSUM, EWMA, and RMA. Previous research has shown the CLs need to be adjusted to accommodate the correlation between sequential CCS. This case requires more of a trial and error approach in determining m since the expected RL can be much different than under the iidN assumption, and the distribution of the RL is no longer exactly Geometric. For example, if the underlying distribution is iidN, and $\pm 3\sigma$ CLs are chosen, the resulting ARL ≈ 370 and recommended m = 5,180, use m = 6,000, run several thousand simulations and count the number of simulations for which no CCS exceeded the CLs in each run. If the count is zero then the choice of m should be sufficient. If not, then increase m. Additionally, the control-charts based on cumulative sums or averages are known to have a much larger standard deviation than the $z \sim N(0, 1)$ process, so the value m will likely be longer than the N(0, 1) process.

Alternatively, instead of presetting m, one could increment a variable by 1 until the first CCS exceeds the CLs, record the value of m as the RL, then stop the run and start the next simulation. Although this is suitable for studying only one RL distribution at a time for a specified set of CLs and ARL, this researcher often performs the study over a very wide range of ARLs (up to 50 simultaneously), which is a feature that often makes MCS more desirable and faster than other methods. Recall, the capability of the design and the ability to use a single simulation run across many desired CLs is an advantage of the proposed design, and is not possible when incrementing instead of presetting m.

In D2, only one series m is calculated, but is a much longer length than used in D1. The choice of m and the expected number of resulting RLs depend directly again on the largest ARL to be estimated. Let *dRL* = the desired number of RLs to evaluate properties related to the largest expected ARL (eARL) in the design. So, for D2, the series-length m is given by (8):

$$m = \text{Max}\left(e\text{ARL}\right) * dRL \tag{8}$$

For example, for ARL = 1,000 and *dRL* = 10,000, m = 1,000*10,000 = 10,000,000. So a series length of m = 10,000,000 will result in about 10,000 recordable RLs.

Inversely, choosing m to accommodate the largest ARL estimate, the resulting number of RLs for lessor values of ARLs is given by (9):

$$dRL = \frac{m}{e\text{ARL}} \tag{9}$$

Using the example above, simultaneously evaluate RLs for known ARLs of 1,000, 500, 370, and 50. The expected number of useable RLs will be about 10,000, 20,000, 27,000, and 200,000, respectively. As a result, the error in estimating lower valued ARLs will be dramatically reduced. Keep in mind that the IC ARL is always larger than the OC ARL, and the larger the mean shift (δ) the smaller the OC ARL. For example, for IC ARL = 1,000, the OC ARLs for mean shifts of $\delta = 1$, $\delta = 2$, and $\delta = 3$ are 90.87, 10.16, and 2.59, respectively. Hence, while *nsim* will remain the same, *m* can be decreased significantly and the program will run much faster.

Calculating the Burn in Period

When extending the validation model to evaluating control charts like the CUSUM, EWMA and RMA, or other control-charts with memory, the series of normal random numbers (*z*) and the transformed control-chart statistics (CCS) should have a burn-in period of runs prior to the actual RLs to be evaluated in the OC process. This accommodates the cumulative nature of the transformed statistics required to mimic a steady-state of the IC series prior to evaluating the subsequent desired series-length. Zero-state simulations refer to RLs those that have been initialized at the target starting value of the control statistic. Steady-state simulations refer to RLs that are evaluated after the control chart statistic has reached a steady-state, meaning the process has been "in-control" long enough for the effect of the starting value to become negligible (Lucas & Saccucci, 1990). So zero-state simulations require no burn-in period, but steady-state simulations do require a short burn-in period.

When evaluating the IC process, it is assumed that one starts the process from an IC zero-state, meaning there is no burn-in period. When evaluating the OC process it is assumed the series has reached a steady-state, implying a burn-in period of a stable IC process. There is no body of literature regarding burn-in for MCS, while there are quite a few articles regarding burn-in for MCMC methods. Although the burn-in period is equivocally stated, it is suggested that a burn-in period that is close to and less than the smallest expected MRL being evaluated should be adequate. Beyond those periods one might expect the burn-in process to

drift toward a false-alarm state. Additionally, CLs for many control-charts like the EWMA are a function of asymptotic variance, so the burn-in period should be long enough for the assumption of the asymptotic variance to hold. For example, if evaluating the OC RL properties of an EWMA with parameter λ with corresponding control-limits set to produce an IC ARL = 300 and MRL = 208, then the burn-in period less than 208 should be adequate.

The EWMA parameter λ can be used to determine a burn-in period necessary for asymptotic CLs to be appropriate. The asymptotic time-varying component of the variance of the EWMA is given as $1 - (1 - \lambda)^{2t}$, hence the asymptotic CLs are constant starting at burn-in period determined by solving for the period t such that $1 - (1 - \lambda)^{2t} \approx 1$. This is especially true when evaluating the OC case, assuming the period was stable and in-control during the previous IC periods. As such, the total series-length will be *Burn-in* + *m*, where evaluation starts at the first period of *m* following the last burn-in period. One can easily run several independent simulations using various burn-in periods to determine the most appropriate period.

Additionally, the rational starting value of the burn-in period is a function of the assumed/estimated mean of the control chart IC process. For example, if evaluating the EWMA control-chart with parameter $\lambda = 0.20$ and process $\mu = 5$, the rational starting or target value would be $CCS_1 = 5$. Hence, $CCS_2 = \lambda * z + (1 - \lambda) * CCS_1 = 0.20 * z + 0.80 * 5$, where z is the randomly generated value such that $z \sim N(\mu = 5, \sigma = 1)$.

Establishing Control Limits

Control charts for the mean of a $z \sim N(0, 1)$ process are typically designed based on the desired IC ARL for a process, which in turn determines the control-limits, which is turn determines the OC ARL based on a specified shift in the process mean. In the $z \sim N(\mu, \sigma)$ process, using a Shewhart or Individuals chart, the upper/lower CLs are easily found for any desired IC ARL, since ARL = 1/*p*, where *p* is the probability of an IC signal (false-alarm). For example, for a desired IC ARL = 370, $p = 1/370 = 0.0027$.

Because half of 0.0027 is below the lower CL and half is above the upper CL, one can use a Normal Inverse function (e.g., Excel's function NORM_INV($p/2, \mu, \sigma$)) to determine the lower CLL ≈ -2.9967; hence the upper CLU ≈ +2.9967, which closely correspond to the $\pm 3\sigma$ CLs. Likewise, when a mean shift (δ) occurs, one can determine the expected OC ARL for the shifted process $z \sim N(\mu + \delta\sigma, \sigma)$ as OC ARL = $1/(p_1 + p_2)$, where $p_1 = P(z < CL_L)$ and $p_2 = P(z > CL_U)$, which are not equal probabilities in the OC case. One can use a

Normal Distribution function (e.g., Excel's function NORM_DIST(z, μ, σ)) to calculate p_1 and p_2 thus allowing calculation of the OC ARL.

When using MCS to evaluate RL properties for other control charts, such as the EWMA, one might want to evaluate RL properties over a range of CLs. As previously noted, one advantage of MCS is the ability to evaluate RL properties over a range of CLs simultaneously, like evaluating properties for ARLs ranging from 20 to 1,000 in steps of 5 units, like 20, 25, 30,…, 1,000. In this case a consideration that must be made regarding CLs is how many decimal places are required, which also helps in deciding the unit step-distance from one ARL to the next. For example, if the IC process is $z \sim N(0, 1)$, then CLs = ±2.99 would include ARL values from 359 to 370, while CLs = ±3.00 correspond to ARL values from 371 to 382. The point being that the more CLs that are evaluated simultaneously, the longer the time it takes to run the simulations, some of which may be redundant and overlapping. Conversely, the additional time will always render some additional information. In general, smaller unit step-distances are more appropriate for ranges of smaller expected ARLs, while larger unit step-distances are more appropriate for ranges of larger expected ARLs.

Determining Simulation Size and Error

For D1, the choice of simulation runs, nsim, depends on the accuracy of the RNG being used, the acceptable maximum error in estimation (E) of ARLs, and the $(100 - \alpha)$% degree of confidence in the estimation of the ARL. A common simulation size used in many published articles for MCS is $nsim = 10{,}000$, hence producing 10,000 RLs (Dyer et al., 2002; Dyer, Conerly, & Adams, 2003; Dyer, Adams, & Conerly, 2003; Lin & Adams, 1996). The value is not arbitrary, but is based on the assumption of the large-sample Normal distribution of a proportion (p), (often an unknown, hence assumed value of $p = 0.50$), and a desired 95% confidence level (corresponding to $\alpha = 0.05$ and $z \approx 2$) in estimation of p with a margin of error $E = 0.01$. The value z is the value of the Standard Normal distribution such that $\alpha/2$ is above $+z$ and below $-z$. In this case the formula is given as

$$nsim = p(1-p) * \left(\frac{z}{E}\right)^2 = 0.25\left(\frac{2}{0.01}\right)^2 = 10{,}000$$

Because a primary goal of MCS is evaluating IC ARLs, and ARLs are a function of p (probability of a false alarm), and p is almost always much less than

0.05, this goal then is to have a very small error in estimation of p (much less than 0.01), hence a small error in estimation of the ARL. But, we are estimating ARLs, which in turn provide estimates of p. So, if we assume a large-sample Normal distribution of the ARL, and we know for the Geometric distribution that the mean and standard deviation are approximately equal, that is, $\mu = \text{ARL} \approx \sigma = \text{SRL}$, then the following estimated equation ((10) below) is given based on D1 for the number of simulations necessary to accommodate the largest ARL estimation (when estimating a wider array of ARLs simultaneously), based on a $(100 - \alpha)\%$ degree of confidence and the maximum allowable error in estimation (E) of the largest expected ARL (eARL). Recall, this estimate does not account for the error in the RNG being used, which can increase the number of required simulations. The value z is the same as described in the previous paragraph.

$$nsim \approx \left(\frac{z * \text{Max}\left(e\text{ARL}\right)}{E} \right)^2 \tag{10}$$

Conversely, the estimated error (E) for any specified expected ARL is given by (11):

$$E \approx \frac{z * e\text{ARL}}{\sqrt{nsim}} \tag{11}$$

Since a Shewhart chart with $p = 0.001$ corresponds to ARL = 1,000, using $nsim$ = 10,000 would suggest a maximum error in estimation of $E = 20$ with 95% confidence, not accounting for any additional error in the RNG. A review of the research also reveals that many published MCS based studies evaluate only a very few selected ARLs, like 370, 500, and 1,000. Using the above scenario ($nsim$ = 10,000, $z = 2.00$), for ARL = 500 we expect $E = 10$, and for ARL = 370 we expect $E = 7.4$. Unfortunately, to reduce the error further (or to increase the degree of confidence), requires a substantial increase in $nsim$. For example, in the case above with ARL = 1,000, reducing the desired maximum error to $E = 5$ requires increasing the number of simulations to $nsim$ = 160,000.

In D2, only one long series m is generated, and the resulting expected number of RLs is given by (9). Hence, (10) is modified to estimate the series-length given by (12) below:

$$m \approx \left(\frac{z}{E}\right)^2 * \text{Max}\left(e\text{ARL}\right) * \sqrt{dRL^3} \tag{12}$$

Conversely, the estimated error (E) for any specified ARL is given by (13):

$$E \approx z * \sqrt{\frac{e\text{ARL}}{m}} * \sqrt[4]{dRL^3} \tag{13}$$

In general, all RL distributions appear Geometric to some degree, but are not exactly iidG. As a result, one must be very cautious about extending Shewhart based error estimates to ARLs obtained for other control charts for which the sequential CCS are not iidN (but are correlated), and making assumptions about the RL distribution that is not exactly iidG. Again, as discussed in the beginning of this section, at least a few of the results should be validated against existing literature before making broader conclusions about RL properties and summary measures.

An advantage of MCS is that the method allows additional summaries such as the MRL, SRL, CDF, and quantities such as percentiles and quartiles. As such, one might want to place confidence limit bounds on the resulting estimated MRL and SRL, in essence allowing one to estimate the error on these values following the simulations. It is assumed that the D1 simulation size (*nsim*) or the D2 desired run-length (*dRL*) is determined relative to a desired maximum error in estimation (E) of the maximum ARL under study.

Regarding the MRL, and assuming a large-sample Normal approximation, the upper and lower confidence interval provides two ordered rank locations of the RLs, which in turn allow one to determine the upper and lower confidence intervals around the MRL, which is asymmetric. Let MR_L and MR_U be the lower and upper rank locations, respectively, in an ordered array of RLs of size *nsim* for any given ARL. (14) and (15) are provided to find the rank locations, where z corresponds to the desired $(1 - \alpha)$% degree of confidence.

$$\text{MR}_\text{L} \approx \left(\frac{nsim}{2}\right) - z * \sqrt{\frac{nsim}{4}} \tag{14}$$

$$\text{MR}_\text{U} \approx \left(\frac{nsim}{2}\right) + z * \sqrt{\frac{nsim}{4}} \tag{15}$$

For example, using 95% confidence and *nsim* = 10,000, MR_L = 4,900, and MR_U = 5100, and the ordered RL values at rank locations 4,900 and 5,100 are the asymmetric confidence interval limits on the MRL. For any given *nsim*, the rank locations will always be the same, but larger *nsim* will result in corresponding RLs closer to the true MRL, that is, less error. Confidence intervals for rank locations for other percentiles and/or quartiles can be derived by adjusting each divisor in each of the two formulas above.

Additionally, one can determine the expected error in the MRL by examining the cumulative probabilities of the Geometric distribution for a specified expected ARL at the 50th percentile RL (median), and upper and lower RLs corresponding to 0.50 ± desired maximum error. Additionally, for any given *nsim*, the expected maximum error (*E*) for the MRL estimate with $(1 - \alpha)$% degree of confidence is given by (16), where *e*MRL is the expected MRL, and (11) is inflated by a factor of $\sqrt{0.50 * \pi} \approx 1.25$ (Stigler, 1973).

$$E \approx \frac{z * e\text{ARL}}{\sqrt{nsim}} * 1.25 \tag{16}$$

Regarding the SRL, the large-sample distribution for any given ARL is Normally distributed, hence the confidence limits on the true SRL are a function of the desired $(100 - \alpha)$% degree of confidence, *nsim*, the estimated SRL, and the Chi-square distribution (χ^2) evaluated at functions of $\alpha/2$ and $df = nsim - 1$ degrees of freedom, so that

$$\chi^2_{\text{Lower}} = \chi^2_{\left(1-\frac{\alpha}{2}, df = nsim-1\right)} \quad \text{and} \quad \chi^2_{\text{Upper}} = \chi^2_{\left(\frac{\alpha}{2}, df = nsim-1\right)}$$

Let SRL_L and SRL_U be the lower and upper confidence limits (respectively). (17) and (18) are provided to find the desired $(1 - \alpha)$% degree of confidence interval limits, where SRL is the estimated SRL.

$$\text{SRL}_\text{L} \approx \frac{df * \text{SRL}}{\chi^2_{\text{Lower}}} \tag{17}$$

$$\text{SRL}_\text{U} \approx \frac{df * \text{SRL}}{\chi^2_{\text{Upper}}} \tag{18}$$

Additionally, the expected error can be determined based on $(1 - \alpha)\%$ degree of confidence in the SRL estimation using (13), where *e*SRL is the expected SRL.

$$E \approx \frac{e\text{SRL} * \left(\chi^2_{\text{Upper}} - df\right)}{df} \tag{19}$$

MCS Programs and Validation Design Example

Although MCS can be implemented in a variety of programming languages, several advantages of using Excel 2010 for MCS studies include the available built-in functions, the ability to write special functions, the VBA programming interface for macros, subroutines and functions, built-in analysis and modeling tools, and the ability to use the spread-sheet as a data repository. Even when MCS is performed in programs such as R, C+, and FORTRAN, the data arrays are often imported into Excel for analysis and modeling. It should be noted that the validation design examples have been exemplified in Excel 2010 using VBA, but also compared with results in the VBA implementation using MT19937 and the Box-Muller transform method adapted to VBA (Annen, 2013). As such, the VBA code shown in Appendix A reflect calling two Excel VBA functions; RND which generates the Uniform [0, 1] variables, and NORM_DIST to generate the series of $z \sim N(\mu, \sigma)$ random variables using input from the RND functions. The code sections reflecting the generation of random numbers can be modified to implement any other choice of RNGs. Additionally, there is no error handling code, so one must be careful about such issues as inputting $\sigma \leq 0$ or placing data of any kind into worksheet cells that are not directly related to specific input or output. The screen-shots provided also reflect formatting options set at the worksheet level and not in the VBA code, such as rounding and run-time output formats. All code and a working Excel 2010 workbook is available from the author by request.

The sections below discuss the setup and description of the worksheet and the underlying VBA code for running the validation MCS based on the Individuals control chart, with an assumed IC $N(\mu, \sigma)$ process for both D1 and D2. Recall, the Individuals chart is a Shewhart chart with subgroup size $n = 1$. The assumption for the design is that, regardless of the underlying IC iidN process, the underlying data (x) would be standardized (z) such that $z \sim N(\mu = 0, \sigma = 1)$ process. As such, for the OC process the distribution changes to a $z \sim N(\delta, 1)$ process.

Program Worksheet Inputs

For Design 1: For D1 and a single IC simulation run, we wish to generate quantity *m* of $z \sim N(\mu = 0, \sigma = 1)$ random variables, sequentially compare each *z*-value to the CLs determined by the desired ARLs, then record the location of the RL in the sequence when the first *z*-value in the series exceeds the CLs (for each specified set of CLs). This process is then replicated *nsim* times. The result will be *nsim* RLs (for each set of specified IC ARLs) for which we can calculate the summaries for estimated ARLs, MRLs, and SRLs. Additionally, one may repeat the entire simulation as many times as desired using the Num Runs input. So, the worksheet will include input cells to specify the number of runs (Num Runs), the series length (*m*), the number of simulations (*nsim*), the in-control process mean (μ) and standard deviation (σ), the mean shift (0 for the IC case, $\delta\sigma$ for the OC case), and the desired IC ARLs, which correspond to CLs that will be calculated using the Excel function NORM_INV implemented in the VBA code. Note that the simulation design allows any values of μ and σ, but there is no need to set these values to anything other than 0 and 1, respectively.

Figure 1 is a screen-shot of the formatted Excel 2010 workbook reflecting the initial setup using Num Runs = 1, *m* = 14,000, *nsim* = 10,000, and IC ARLs of 1,000, 500, 370, and 50. The input parameters reflect a desire for a maximum Error of *E* = 20 for estimating ARL = 1,000 with 95% confidence. The desired IC ARLs are entered from the largest to the smallest. With this limited example we wish to use MCS validation to estimate the four ARLs, MRLs and SRLS of an Individuals control chart based on the control-limits corresponding to specified IC probabilities of $p = 0.001$ (ARL = 1000), $p = 0.002$ (ARL = 500), $p = 0.0027$ (ARL =370), and $p = 0.02$ (ARL = 50). The corresponding calculated control-limits will be ±3.2905, ±3.0902, ±2.9997, and ±2.3263, respectively. Recall, we already know the properties of the Individuals based $z \sim N(0, 1)$ process, so the MCS in the example is used to exemplify setup and validation of the estimated MCS results with expected results.

The input cells are as follows:

- B2 (Num Runs) = 1. This value allows one to produce one or more independent simulations runs, hence allowing one to investigate results of the same simulation design over multiple runs.
- B3 (*m*) = 14,000. This value is based on the series length calculation
- $m \approx \text{Max(ARL)}*14 = 1{,}000*14 \approx 14{,}000$.

- B4 (*nsim*) = 10,000. This value is based on a desired maximum error in estimation of $E = 20$ for Max ARL = 1,000, with 95% confidence. Error will necessarily be lower for smaller estimates.
- B5 (μ) = 0 (mean for the underlying IC process assuming a $z \sim N(0, 1)$ process).
- B6 (σ) = 1 (standard deviation for underlying IC process assuming a $z \sim N(0, 1)$ process).
- B7 (μ-Shift) = 0 (0 for the IC process, or > 0 for the OC process $z \sim N(\delta\sigma, 1)$ process).
- F1:I1 (IC ARLs) = 1000 (F1), 500 (G1), 370 (H1), 50 (I1), assuming one wants to evaluate these four ARLs.
- Run Button – A button to execute the VBA subroutine

For Design 2: Figure 2 is a screen-shot of the formatted Excel 2010 worksheet reflecting the initial setup using $m = 10{,}000{,}000$ (B3) while the remainder of input cells are the same as used in the D1 example. Recall that for D2 we wish to generate one very long series of length m of $z \sim N(0, 1)$ random variables, sequentially compare each z to the CLs determined by the desired ARLs, then record the location of each RL in the sequence when each z in the series exceeds the CLs (for each specified set of CLs). The long series is based on the *dRL*, and is almost equivalent to *m*nsim* in D1. The other inputs are the same as D1, with the same error and confidence level. The output will be almost the same is D1, except that we don't know in advance how many RLs will be produced for each set of specified IC ARLs, but can be estimated using (9).

	A	B	C	D	E	F	G	H	I
1	Inputs			Inputs	IC ARLs	1000	500	370	50
2	NumRuns =	1			Upper CLs				
3	m =	14000			Lower CLs				
4	nsim =	10000			Expected ARLs				
5	μ =	0		Outputs	Expected MRLs				
6	σ=	1			Expected SRLs				
7	μ-Shift =	0			Estimated ARLs				
8					Estimated MRLs				
9	Outputs				Estimated SRLs				
10	Start-Time			Blank Runs	Run-Lenghts				
11	End-Time			0					
12	Run-Time								
13	Sim Run =			Run					

Figure 1. Design 1 program input cells

	A	B	C	D	E	F	G	H	I
1	Inputs			Inputs	IC ARLs	1000	500	370	50
2	Num Runs =	1			Upper CLs				
3	m =	10000000			Lower CLs				
4	μ =	0			Expected ARLs				
5	σ =	1			Expected MRLs				
6	μ-Shift =	0		Outputs	Expected SRLs				
7					Estimated ARLs				
8	Outputs				Estimated MRLs				
9	Start-time =				Estimated SRLs				
10	End-time =				Count				
11	Run-time =			Run	Run-Lengths				
12	Sim Run =								

Figure 2. Design 2 program input cells

Program Worksheet Outputs

For Design 1: Figure 3 is a screen-shot reflecting the D1 outputs of one individual run of m = 14,000 and $nsim$ = 10,000 simulations. All output cells are set and/or calculated using VBA code as shown in Appendix A. The worksheet output cells include a timer with Start-time, End-time and Run-time, an indicator of the exact simulation number being run at any given time (if multiple simulations are to be done), cells containing the upper and lower control-limits (based on the input ARLs), expected ARLs, MRL, and SRLs, columns to record the RLs of all simulations for each set of control-limits, and cells for the estimated ARLs, MRLs and SRLs for each IC ARL (or for each OC ARL in the OC process).

The output cells for this example are described as follows. See the screen-shot for actual exemplary values.

- B10 = Start-time, B11 = End-time, B12 = Run-time
- B13 (Sim Run) = changing variable depending on exact simulation being run, starting with 1 and ending with Num Runs in B2. Only used if multiple simulations are being run on the same set of inputs.
- F2:I3 = upper and lower CLs based on input ARLs.
- F4:I4 = expected ARLs based on the input μ-Shift. The values are the same as the input ARLs for the IC process but will change for the OC process, and depend on the mean shift (set in input cell B7).
- F5:I5 = expected MRLs based on the input μ-Shift. The values are the same for the IC process but will change for the OC process, and depend on the mean shift (set in input cell B7).

- F6:I6 = expected SRLs based on the input μ-Shift. The values are the same for the IC process but will change for the OC process, and depend on the mean shift (set in input cell B7).
- F7:I7 = estimated ARLs (average of RLs).
- F8:I8 = estimated MRLs (median of RLs).
- F9:I9 = estimated SRLs (standard error of RLs).
- F10:I10010 = 10,000 recorded RLs for each input ARL.
- D11 = a count of the runs of size *m* out of *nsim* that did not produce a RL for the largest input ARL. If *m* is selected appropriately and large enough then the value will be 0.

For Design 2: Figure 4 is a screen-shot reflecting the D2 outputs of one individual run. All output cells are set and/or calculated using VBA code as shown in Appendix B. The worksheet output cells are the same as D1, with the exceptions

	A	B	C	D	E	F	G	H	I
1	**Inputs**			**Inputs**	**IC ARLs**	**1000**	**500**	**370**	**50**
2	**NumRuns =**	**1**			**Upper CLs**	3.2905	3.0902	2.9997	2.3263
3	**m =**	**14000**			**Lower CLs**	-3.2905	-3.0902	-2.9997	-2.3263
4	**nsim =**	**10000**			**Expected ARLs**	1000	500	370	50
5	**μ =**	**0**		**Outputs**	**Expected MRLs**	693	346	256	34
6	**σ=**	**1**			**Expected SRLs**	1000	500	370	50
7	**μ-Shift =**	**0**			**Estimated ARLs**	1012	505	373	50
8					**Estimated MRLs**	701	346	259	35
9	**Outputs**				**Estimated SRLs**	1020	503	376	49
10	**Start-Time**	2/11/2014 7:27		Blank Runs	**Run-Lenghts**	558	406	406	35
11	**End-Time**	2/11/2014 8:16		0		331	331	190	190
12	**Run-Time**	49:09.2				1470	1470	773	26
13	**Sim Run =**	1		Run		2202	2202	1625	115

Figure 3. Design 1 program output cells

	A	B	C	D	E	F	G	H	I
1	**Inputs**			**Inputs**	**IC ARLs**	**1000**	**500**	**370**	**50**
2	**Num Runs =**	**1**			**Upper CLs**	3.2905	3.0902	2.9997	2.3263
3	**m =**	**10000000**			**Lower CLs**	-3.2905	-3.0902	-2.9997	-2.3263
4	**μ =**	**0**			**Expected ARLs**	1000	500	370	50
5	**σ =**	**1**			**Expected MRLs**	693	346	256	34
6	**μ-Shift =**	**0**		**Outputs**	**Expected SRLs**	1000	500	370	50
7					**Estimated ARLs**	1007	505	370	50
8	**Outputs**				**Estimated MRLs**	696	346	256	35
9	**Start-time =**	**2/11/2014 7:18**			**Estimated SRLs**	1006	509	371	49
10	**End-time =**	**2/11/2014 7:23**			**Count**	9821	19783	27015	200552
11	**Run-time =**	**0:04:42**		Run	**Run-Lengths**				
12	**Sim Run =**	**1**				2353	2154	1007	23
13						581	199	396	156

Figure 4. Design 2 program output cells

that in D2 there is no need for a count of runs that didn't produce a RL value, and the D2 output includes a count of the number of RLs for each set of control-limits. For example, the single simulation resulted in 9,821 RLs for ARL = 1,000 as displayed in cell F10. Cells G10, H10, and I10 display the counts for each of the additional ARLs. Note the much larger RL counts for the smaller ARL calculations.

Program Overview

For Design 1: For D1, when the VBA code is executed for the IC $z \sim N(\mu, \sigma)$ process, the following steps occur programmatically. For each simulation run (Num Runs); (1) The Start-time is stored in cell B10, (2) each of the upper and lower control-limits are calculated and stored in cells starting in cell F2. These values correspond to the desired IC ARLs, based on input choices of μ and σ, and (3) each of the expected ARLs, MRLs and SRLs are calculated and stored in in cells starting in cell F4.

For each simulation (*nsim*); (4) a series of *m* *z*-values are generated and stored in an array. For each set of the control-limits; (5) each *z*-value is sequentially compared to each upper CL and lower CL, one at a time. When the first *z*-value exceeds its CLs, the RL is recorded in an array. After a RL has been recorded for each specified CL, the process terminates and continues to the next simulation. Steps 5 through 7 are continued until all simulations are completed. After all *nsim* simulations are complete, the RL array is copied into the worksheet.

Following the last simulation in step 5; (6) the summary estimated measures (estimated ARL, MRL and SRL) are calculated based on the RLs and stored in cells

starting in cell F7, and (7) the End-time is stored in cell B11 and Run-time is stored in cell B12.

When an OC process is simulated, the user sets the value of the shifted mean in worksheet cell B7, in terms of a shift in σ. For example, regardless of the choice of σ, entering the value "3" in cell B7 implies a 3σ shift. In step 3 the expected OC ARLs, MRLS and SRLs are calculated and stored in cells starting in cell F4. Then in step 5 the z-values are generated as $z \sim N(\mu\text{-Shift}, 1)$. Note that m is now much smaller and chosen to accommodate the largest OC ARL. For example, for IC ARL = 1,000 and corresponding control-limits ±3.2905, a mean-shift of 1σ results in an OC ARL ≈ 91. Hence, $m = \text{Max}(e\text{ARL})*14 = 91*14 = 1{,}275$. Keeping $m = 10{,}000$ will result in an expected error of $E = 1.82$ with 95% confidence.

For Design 2: For D2, when the VBA code is executed for the IC $z \sim N(\mu, \sigma)$ process, the following steps occur programmatically. For each simulation run (Num Runs), the first 4 steps and steps 6 and 7 are the same as D1. For each of the control-limits; (5) each z-value is sequentially compared to each upper CL and lower CL, one at a time. Anytime a z-value exceeds its CLs, the RL is recorded in an array. After all RLs have been recorded for each specified CL, the procedure terminates and the RL array is copied into the worksheet. When an OC process is simulated, D2 is setup the same as D1. Again m is now much smaller and chosen to accommodate the largest OC ARL. Using the previous example, for IC ARL = 1,000 and corresponding control-limits ±3.2905, a mean-shift of 1σ results in an OC ARL ≈ 91. For $dRL = 10{,}000$, $m = \text{Max(ARL)}*dRL = 91*10{,}000 = 900{,}000$.

Other Considerations

It might be questioned why two MCS validation designs are exemplified, when both produce equivalent results. When the IC process is simulated for any control-charting scheme, both designs are adequate and have equal results. But when simulating the OC case for cumulative schemes, like the CES, CUSUM, EWMA and RMA, D1 is required since the cumulative effect of the control-chart statistic depends on previous states, which must be based on the last simulated value of the previous control-chart statistic. That is, each new series of the OC process must be started using the last value of the stable IC burn-in period, which reflects the IC process prior to a process shift. Additionally, the design choice depends on computing time versus the personal computer's (PC) configuration and performance. D1 takes significantly more time to run on a PC since the series-

length *m* is replicated and evaluated *nsim* times, but for most simulations the full series length m is not necessary. Unfortunately, one doesn't know in advance when the first *z* to exceed the CLs will occur, so *m* must be long enough to ensure a recordable RL will occur in each separate *nsim*. The full series of *z*-values for each *nsim* are stored in an array, and since the array size is relatively small, the program is less dependent on the PCs processor and memory to handle large arrays. The trade-off then relates to being able to use a PC with less processing capacity and memory, but requiring greater processing time.

The D2 design is best used to evaluate the OC case when implementing control-charts with run-rules or multiple-sampling plans, wherein the OC statistic at any given time is not dependent on a previous value. D2 creates one very long array of size *m* and indexes through it to count the RLs, and is thus significantly faster. The long array size though may create a limitation on computers with less memory. As a result, D2 might not be a feasible option on some PCs, but for those that can accommodate the large array in memory the processing time in reduced significantly. Additionally, the D2 design with an overly long series-length may still press the limits of any modern PC's ability to dimension an overly large array in memory. In either event, if using Excel 2010 to generate or store RLs, one must be careful of the longest expected RL output since the maximum number of rows is 1,048,576. If the RL is expected to exceed this value then the RLs can be truncated or can be written to a text file.

Design Validation and Error Analysis

For the sake of completion, a limited study was conducted as follows to validate designs and estimate error using two tests (T1 and T2). The test compares and contrasts results using T1 and T2 (shown below) to validate the MCS design for the IC $z \sim N(0, 1)$ process using $m = 14{,}000$, $nsim = 10{,}000$, ARLs of 1000, 500, 370, and 50, respectively, repeated NumRuns = 20 times, and maximum error in estimation corresponding to 95% confidence. The aggregated results are shown in Tables 1 (ARLs), 2 (MRLs), and 3 (SRLs), and are further discussed.

- T1: Use Excel 2010 and VBA implementing built-in RND and NORM_DIST functions.
- T2: Use Excel 2010 and VBA implementing the Mersenne Twister algorithm and Box-Muller transform methods (Annen, 2013).

Tables 1, 2, and 3 reflect simulated estimated ARLs, MRLs, and SRLs as specified above for tests T1 and T2. Note that the results in each table are sorted by ARL, MRL, and SRL in ascending order, respectively. Table 1 displays the observed (estimated) ARLs (versus expected ARLs) for the 20 independent simulation runs (sorted from lowest to highest). The summary statistics provided reflect the aggregated ARLs over the 20 runs. For T1, while the aggregated average ARLs do not all match the expected results, most of individual ARL estimates are very close to the expected ARLs, that is, most of the observed error is within the maximum expected error. The last row of the table reflects the percent of the 20 individual runs for each estimated ARL that are within the expected error, and all but expected ARL = 370 resulted in 95% to 100% of runs within the expected error. ARL = 370 also corresponds to an unexpectedly high standard deviation of RLs (4.35). The average aggregate error for expected ARL = 1,000 (E = 3.85) is well within expected error (4.5), but the error for expected ARL = 500 (E = 4.60) is 205% larger than expected error (2.24), the error for expected ARL = 370 (E = 3.35) is 202% larger than the expected error (1.65), and the error for expected ARL = 50 (E = 0.30) is 134% larger than the expected error (0.22). Although the percent difference is large the actual error is very small.

Regarding T2, the aggregated estimated ARLs are generally quite close to expected ARLs, and the standard deviation between the individual runs are typically close to those of T1, with the exception of estimated ARL = 1,000 with a very large standard deviation (7.13). While each of the 20 individual runs for each estimated ARL is within the expected errors, the average aggregated errors are similar to that of T1, all having larger than expected errors.

Table 2 displays the same summary data as Table 1 but for the MRLs instead. The estimated MRLs are very close to expected MRLS. The results are largely consistent with those found in Table 1, but the maximum errors for T1 and T2 are typically somewhat larger than those for the ARLs. The standard deviations between median RLs as well as the standard deviations of errors are much larger than expected. The last row of the table reflects the percent of 20 individual runs for each MRL that are within the expected error. For T1, 100% of the 20 runs for expected MRL = 693 (ARL = 1,000) and MRL = 256 (ARL = 370) are within expected error, while 90% of runs for expected MRL = 346 (ARL = 500) are within expected error, and 95% of runs for expected MRL = 34 (ARL=50) are within expected error. For T2, 90% of the 20 runs for expected MRL = 693 are within expected error, 95% for MRL = 346, 100% for MRL = 258, and only 85% for MRL = 34.

Table 1. ARL summaries for S1

Expected ARLs	1000		500		370		50	
Simulation	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**Test 1**	**Test 2**
1	994	989	489	494	361	365	49	49
2	995	994	490	494	362	367	49	49
3	996	995	491	495	362	367	49	49
4	997	996	492	495	365	368	50	49
5	998	996	492	496	367	368	50	50
6	998	997	493	498	368	370	50	50
7	998	998	495	498	368	370	50	50
8	999	999	495	498	369	370	50	50
9	1000	1001	496	500	370	371	50	50
10	1002	1001	496	501	370	371	50	50
11	1003	1001	497	502	371	372	50	50
12	1003	1002	497	502	371	373	50	50
13	1003	1003	498	503	371	373	50	50
14	1004	1003	499	504	372	374	50	50
15	1004	1005	499	504	372	374	50	50
16	1005	1008	499	505	372	374	50	50
17	1006	1008	500	505	374	375	50	51
18	1007	1012	502	506	374	376	51	51
19	1007	1014	504	508	375	376	51	51
20	1008	1017	504	508	377	376	51	51
ARL Summary Statistics for 20 Simulation Runs								
Average	1001	1002	496	501	370	372	50	50
Median	1002	1001	496	501	370	372	50	50
Std Dev	4.32	7.13	4.40	4.58	4.35	3.30	1.00	0.64
Minimum	994	989	489	494	361	365	49	49
Maximum	1008	1017	504	508	377	376	51	51
ARL Error Summary Statistics for 20 Simulation Runs								
Expected Error1	20.00		10.00		7.40		1.00	
Expected Error2	4.47		2.24		1.65		0.22	
Average	3.85	5.56	4.60	4.03	3.35	2.99	0.30	0.49
Median	3.50	4.07	4.00	4.26	2.00	2.91	0.00	0.41
Std Dev	2.10	4.60	3.15	2.12	2.73	1.95	0.46	0.39
Minimum	0.00	0.57	0.00	0.08	0.00	0.07	0.00	0.03
Maximum	8.00	17.37	11.00	7.70	9.00	6.46	1.00	1.36
% Within Error	100%	100%	95%	100%	85%	100%	100%	100%

Note: 1 individual run of *nsim* = 10,000, 2 aggregated runs of *nsim* = 200,000

Table 2. MRL summaries for S1

Expected MRLs	693		346		256		34	
Simulation	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**Test 1**	**Test 2**
1	676	675	337	336	248	250	33	34
2	679	675	337	338	251	251	34	34
3	683	677	337	338	251	252	34	34
4	684	681	341	339	251	253	34	34
5	685	683	341	340	252	254	34	34
6	685	684	344	341	252	254	35	35
7	687	687	345	342	253	254	35	35
8	689	687	345	344	255	254	35	35
9	690	688	346	344	255	255	35	35
10	691	690	347	345	256	255	35	35
11	692	694	347	346	256	256	35	35
12	694	695	348	347	257	257	35	35
13	694	696	348	347	259	258	35	35
14	695	696	349	348	259	258	35	35
15	697	699	350	348	259	259	35	35
16	701	703	351	350	260	260	35	35
17	701	705	352	350	260	260	35	35
18	702	705	354	352	260	260	35	36
19	706	712	356	352	262	261	35	36
20	706	718	358	353	262	262	36	36
MRL Summary Statistics for 20 Simulation Runs								
Average	692	692	347	345	256	256	35	35
Median	691	692	347	346	256	256	35	35
Std Dev	8.52	12.15	6.04	5.16	4.17	3.48	0.64	0.56
Minimum	676	674.5	337	336	248	250	33	34
Maximum	706	718	358	353	262	262	36	36
ARL Error Summary Statistics for 20 Simulation Runs								
Expected Error1	17.33		8.65		6.40		0.85	
Expected Error2	3.87		1.93		1.43		0.19	
Average	7.05	9.88	4.75	4.28	3.50	2.95	0.85	0.85
Median	8.00	9.75	4.50	4.00	4.00	2.50	1.00	1.00
Std Dev	4.66	6.56	3.55	2.86	2.06	1.69	0.48	0.55
Minimum	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Maximum	17.00	25.00	12.00	10.00	8.00	6.00	2.00	2.00
% Within Error	100%	90%	90%	95%	95%	100%	95%	85%

Note: 1 individual run of *nsim* = 10,000, 2 aggregated runs of *nsim* = 200,000

Table 3. SRL summaries for S1

Expected SRLs	1000		500		370		50	
Simulation	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**Test 1**	**Test 2**	**Test 1**	**Test 2**
1	1000	984	484	489	361	366	48	48
2	1001	984	485	493	363	367	49	49
3	1001	985	485	496	364	368	49	49
4	1001	985	486	497	365	372	49	49
5	1003	988	486	499	366	372	49	49
6	1005	989	487	499	367	372	49	49
7	1005	992	487	499	368	372	49	49
8	1006	992	487	500	368	372	49	49
9	1006	995	488	501	369	372	49	49
10	1006	1004	488	503	370	373	50	49
11	1007	1007	489	505	370	375	50	49
12	1007	1008	490	506	371	375	50	49
13	1010	1008	490	506	372	376	50	49
14	1010	1008	491	506	373	376	50	50
15	1010	1009	495	507	374	376	50	50
16	1013	1012	495	509	374	378	50	50
17	1013	1012	495	510	376	378	50	50
18	1013	1012	496	510	376	379	50	50
19	1014	1022	498	511	377	381	50	50
20	1014	1029	498	514	378	389	51	51
SRL Summary Statistics for 20 Simulation Runs								
Average	1007	1001	490	503	370	374	50	49
Median	1007	1005	489	504	370	374	50	49
Std Dev	4.57	13.48	4.47	6.46	4.88	5.18	0.66	0.70
Minimum	1000	984	484	489	361	366	48	48
Maximum	1014	1029	498	514	378	389	51	51
SRL Error Summary Statistics for 20 Simulation Runs								
Expected Error1	27.73		13.86		10.26		1.39	
Expected Error2	6.20		0.15		0.11		0.02	
Average	7.25	11.84	10.00	5.83	4.00	5.30	0.55	0.77
Median	6.50	11.49	11.50	5.84	4.00	4.33	0.50	0.58
Std Dev	4.56	5.82	4.44	3.80	2.61	4.15	0.59	0.49
Minimum	0.00	3.52	2.00	0.37	0.00	1.60	0.00	0.02
Maximum	14.00	29.01	16.00	13.69	9.00	18.74	2.00	1.94
% Within Error	100%	95%	70%	95%	100%	90%	95%	95%

Note: 1 individual run of *nsim* = 10,000, 2 aggregated runs of *nsim* = 200,000

Table 3 displays the same summary data as Table 1 but for SRLs instead. For T1, while the aggregated average SRLs do not all match the expected results, most of individual SRL estimates are relatively close to the expected SRLs. The average errors are significantly larger than those of the ARLs and MRLs. The last row of the table reflects the percent of 20 individual runs for each SRL that are within the expected error. For T1, while expected SRLs 1000, 370 and 50 have 95% to 100% of runs within the expected error, ARL = 500 has only 70% of runs within the expected error. The average aggregate errors for all T1 SRLs are larger than expected. Results for T2 are more consistent with what is expected. The estimated SRLs are all very close to expected SRLs, and expected SRL 1,000, 500 and 50 have 95% of runs within the expected error, while SRL = 370 has 90% of runs within expected error.

As previously mentioned, for any given ARL/MRL/SRL estimate and simulation size, any difference between expected errors versus observed errors are due to the choice of RNG, or perhaps implementation and/or numerical precision. While most results are consistent with expected results, and the estimates are relatively adequate, it appears that Excel's implementation of the two RNGs is adequate, and in some cases is superior to the VBA implementation of the Mersenne Twister algorithm and Box-Muller transform methods. Additionally, the T1 design runs about 8-times faster than the T2 design, running on a PC configured with an AMD Phenom II 945 Processor (3.00 GHz), 8 GB of Ram, using Windows 7 (64-bit) and Excel 2010 (32-bit).

Design Modification, Validation, and Evaluation for the EWMA Control Chart

Assuming the D1 design proposed in this paper has validated the simulation results for the $z \sim N(0, 1)$ process, we can now modify the program to transform the series of $z \sim N(0, 1)$ values to a series of EWMA control chart statistics. It is known that the CLs for the EWMA are much narrower than those of the Individuals control-chart. Control-limit equations in the literature relate that the EWMA with parameter $\lambda = 0.25$ and desired IC ARL≈500 has estimated CLs = ±1.134. The corresponding Shewhart IC ARL = 370 with CL = ±3.00.

The D1 program designed is then modified to evaluate a set of 7 different CL values ranging from ±1.14271 to ±1.1240 for which to compare our EWMA statistics. These CLs correspond to Shewhart $z \sim N(0, 1)$ ARLs from 400 to 340 in steps of 10, centered on ARL = 370. Hence we will modify the MCS validation design to provide estimated ARLs, MRLs, and SRLs for the complete set of EWMA

CLs, for both the IC and OC cases. This modified program will thus allow one to validate and then estimate IC and OC RL properties of the EWMA over a wide array of parameter values and any choice of CLs.

The simulation is then designed to accommodate the following three studies.

- Study 1: One individual validation run (Num Runs = 1) of m = 7,000 and $nsim$ = 200,000, $\lambda = 1.00$, for the IC $z \sim N(0, 1)$ process. The simulation requires no burn-in period. The EWMA with $\lambda = 1.00$ corresponds to a Shewhart control-chart, hence the EWMA CLs for $\lambda = 1.00$ are shown in Table 4 (2nd row) correspond to desired Shewhart IC ARLs between 400 and 340. This simulation is to validate the modified design when $\lambda = 1.00$. Under this process we expect the error in estimation of Shewhart desired ARL = 370 to be $E \approx 1$ with 95% degree of confidence.
- Study 2: One individual validation run (Num Runs = 1) of m = 7,000 and $nsim$ = 200,000, $\lambda = 0.25$, for the IC $z \sim N(0, 1)$ process. To maintain consistency with methods and results in literature, there is no burn-in period, nor are time-varying control-limits used at start-up. The modified EWMA CLs with $\lambda = 0.25$ shown in Table 4 correspond to desired EWMA IC ARLs between about 536 and 462. This simulation is to validate the modified design with expected results when $\lambda = 0.25$ with no mean-shift ($\sigma = 0.00$).
- Study 3: One individual estimation run (Num Runs = 1) of m = 1,000 and $nsim$ = 200,000, $\lambda = 0.25$, for the OC $z \sim N(\mu\text{-Shift}, 1)$ process based on CLs in study 2. The simulation uses a burn-in period of 50. This simulation is to estimate RL properties and generate summary estimates (ARL, MRL, SRL) and compare with expected OC results when $\lambda = 0.25$ and mean-shifts of $\sigma = 0.50$, $\sigma = 1.00$, and $\sigma = 1.50$.

For study 1, the observed ARLs, MRLs and SRLs are consistent with what is expected. While many of the observed ARL and MRL estimates equal the expected results, the maximum error for those that do not equal expected results is never more than $E = 1$. While many of the observed SRLs are also equal to expected results, the maximum error never exceeds $E = 2$. Recall in the limited validation study in a subsequent section that the SRL is most often marginally inflated/deflated from expected results. Since the simulation result is consistent

with what is known, the modified design is validated and expected to be highly accurate.

For study 2, focus on the observed EWMA ARL corresponding to Shewhart IC ARL = 370. Much of the comparative literature on the EWMA provide results for $\lambda = 0.25$ and ARL = 370, relating that the expected EWMA IC ARL = 500.

Many studies based on integral equation, MCMC and MCS studies estimate the actual IC ARL between 501 and 503. This simulation result reveals estimated ARL = 501, which is largely consistent with the previous finding. Since the simulation result is consistent with the existing literature, the modified design is again validated and expected to be highly accurate.

Table 4. EWMA validation and estimation summaries

Shewhart IC ARLs	**400**	**390**	**380**	**370**	**360**	**350**	**340**
EWMA CLs λ = 1.00	**3.0233**	**3.0157**	**3.0078**	**2.9997**	**2.9913**	**2.9827**	**2.9738**
Expected IC ARL	400	390	380	370	360	350	340
Observed IC ARL	399	389	380	371	360	350	341
Expected IC MRL	277	270	263	256	249	242	235
Observed IC MRL	278	270	263	257	250	243	237
Expected IC SRL	400	390	380	370	360	350	340
Observed IC SRL	398	389	380	370	359	349	340
EWMA CLs λ = 0.25	**1.1427**	**1.1398**	**1.1368**	**1.1338**	**1.1306**	**1.1274**	**1.124**
			μ-Shift = 0.00				
Observed IC ARL	536	524	512	501	488	475	462
Observed IC MRL	372	364	355	347	338	330	320
Observed IC SRL	533	523	510	503	489	475	463
			μ-Shift = 0.50				
Observed OC ARL	50	49	48	48	47	46	46
Observed OC MRL	36	35	35	34	34	34	33
Observed OC SRL	46	45	44	44	43	43	42
			μ-Shift = 1.00				
Observed OC ARL	11	11	11	11	11	11	11
Observed OC MRL	9	9	9	9	9	9	9
Observed OC SRL	8	8	8	8	7	7	7
			μ-Shift = 1.50				
Observed OC ARL	5	5	5	5	5	5	5
Observed OC MRL	5	5	5	5	5	5	5
Observed OC SRL	3	3	3	3	3	3	3

For study 3, again focus on EWMA IC ARL = 370 and OC ARLs related to the three specified mean-shifts. The findings are consistent with existing literature regarding OC ARLs (Lucas & Saccucci, 1990). Although not a part of this study, it is interested to note the same OC ARLs for mean-shifts δ = 1.00, and δ = 1.50, suggesting that across the specified range of IC ARLs one can expect the same OC ARLs over the specified range of EWMA CLs, hence one would benefit most by choosing the wider CLs to increase the IC ARL. Since the simulation result is consistent with the existing literature, the modified design is again validated and expected to be highly accurate, hence one could feel confident using the design over a wider range of EWMA studies.

Conclusion

Two MCS validation design schemes related to control-charting simulation studies were proposed. The basic design was modified to evaluate the EWMA control-chart. Three EWMA MCS studies were conducted and evaluated, resulting in summaries consistent with existing literature, hence validating the adequacy of the MCS design schemes. Although the MCS design is specific to control-chart evaluation, the basic design and related issues extend to simulation studies in other fields. It is suggested that researchers and practitioners using any MCS design should state results relative to the issues discussed in this paper, including justification of RNGs, simulation size, expected error, burn-in period, and design validation, among others.

References

Abdel-Aziz, A. S., Abdel Dayem, M., & Darwis, G. (2008). *Network intrusion detection system applying multivariate control charts*. Paper presented at INFOS2008, Cairo, Egypt. Retrieved from http://infos2008.fci.cu.edu.eg/infos/NET_02_P011-018.pdf

Abramowitz, M., & Stegun, I. A. (1972). *Handbook of mathematical functions*. New York: Dover.

Acklam, P. J. (2014). *An algorithm for computing the inverse normal cumulative distribution function*. Unpublished manuscript. Retrieved from http://home.online.no/~pjacklam/notes/invnorm/

Ahrens, J. H., & Dieter, U. (1988). Efficient table-free sampling methods for the exponential, Cauchy, and normal distributions. *Communications of the ACM, 31*(11), 1330-1337. doi: 10.1145/50087.50094

Annen, K. (2013). WEB_REG_MT19937 [computer software]. Retrieved from http://www.web-reg.de

Benneyan, J., Lloyd, R., & Plsek, P. (2003). Statistical process control as a tool for research and healthcare improvement. *Quality and Safety in Health Care, 12*(6), 458-464. doi: 10.1136/qhc.12.6.458

Box, G. E., & Muller, M. E. (1958). A note on the generation of random normal deviates. *The Annals of Mathematical Statistics, 29*(2), 610-611. doi: 10.1214/aoms/1177706645

Chen, G. (1997). The mean and standard deviation of the run length distribution of $\bar{X}$ charts when control limits are estimated. *Statistica Sinica, 7*(3), 789-798. Retrieved from http://www3.stat.sinica.edu.tw/statistica/j7n3/j7n314/j7n314.htm

Daudin, J. J. (1992). Double sampling charts. *Journal of Quality Technology, 24*(2), 78-87.

Dyer, J. N., Adams, B. M., & Conerly, M. D. (2003). The reverse moving average control chart and comparisons of forecast based monitoring schemes. *Journal of Quality Technology, 35*(2), 135-155.

Dyer, J. N., & Barrett, J. D. (2000). A simple mathematical description of the impact of forecast recovery for AR(2) processes. *Palmetto Review, 3*, 14-22.

Dyer, J. N., Conerly, M. D., & Adams, B. M. (2003). A simulation study of the impact of forecast recovery for control charts applied to various ARMA processes. *Journal of Modern Applied Statistical Methods, 1*(2), 343-353. Retrieved from http://digitalcommons.wayne.edu/jmasm/vol1/iss2/43/

Dyer, J. N., Conerly, M. D., Adams, B. M., & Barrett J. D. (2002). Multivariate forecast based control charting schemes. *Journal of Business, Industry, and Economics, 1*(2), 145-156.

Fu, M. C., & Hu, J. (1999). Efficient design and sensitivity analysis of control charts using Monte Carlo simulation. *Management Science, 45*(3), 395-413. doi: 10.1287/mnsc.45.3.395

Gan, F. F. (1993). An optimal design of EWMA control charts based on median run length. *Journal of Statistical Computation and Simulation, 45*(3-4), 169-184. doi: 10.1080/00949659308811479

Golosnoy, V., & Schmid, W. (2007). EWMA control charts for monitoring optimal portfolio weights. *Sequential Analysis: Design Methods and Applications, 26*(2), 195-224. doi: 10.1080/07474940701247099

He, D., Grigoryan, A., & Sigh, M. (2002). Design of double- and triple-sampling X-bar control charts using genetic algorithms. *International Journal of Production Research, 40*(6), 1387-1404. doi: 10.1080/00207540110118415

Hunter, J. S. (1986). The exponentially weighted moving average. *Journal of Quality Technology, 18*(4), 203-210.

Irianto, D., & Shinozaki, N. (1998). An optimal double sampling $\bar{X}$ control chart. *International Journal of Industrial Engineering: Theory, Applications and Practice, 5*(3), 226-234.

Jaing, W., Au, T., & Tsui, K.-L. (2007). A statistical process control approach to business activity monitoring. *IIE Transactions, 39*(3), 235-249. doi: 10.1080/07408170600743912

Kalgonda, A., Koshti, V., & Ashokan, K. (2011). Exponentially weighted moving average control chart. *Asian Journal of Management Research, 2*(1), 253-263.

Kinderman, A. J., & Ramage, J. G. (1976). Computer generation of normal random variables. *Journal of the American Statistician, 71*(356), 893-898. doi: 10.1080/01621459.1976.10480965

Knusel, L. (2002). On the reliability of Microsoft Excel XP for statistical purposes. *Computational Statistics & Data Analysis, 39*(1), 109-110. doi: 10.1016/S0167-9473(02)00035-X

Korn, R., Korn, E., & Kroisandt, G. (2010). *Monte Carlo methods and models in finance and insurance*. Boca Raton, FL: CRC Press/Taylor & Francis.

Lin, W. S., & Adams, B. N. (1996). Combined control charts for forecast-based monitoring schemes. *Journal of Quality Technology, 28*(3), 289-302.

Lucas, J. M., & Saccucci, M. S. (1990). Exponentially weighted moving average control schemes: Properties and enhancements. *Technometrics, 32*(1), 1-12. doi: 10.1080/00401706.1990.10484583

Marsaglia, G., & Bray, T. A. (1964). A convenient methods for generating normal variables. *SIAM Review, 6*(3), 260-264. doi: 10.1137/1006063

Marsaglia, G., & Tsang, W. W. (2000). The ziggurat method for generating random variables. *Journal of Statistical Software, 5*(8), 1-7. doi: 10.18637/jss.v005.i08

Marsaglia, G., Zaman, A., & Marsaglia, J. C. (1994). Rapid evaluation of the inverse of the normal distribution function. *Statistics & Probability Letters, 19*(4), 259-266. doi: 10.1016/0167-7152(94)90174-0

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation, 8*(1), 3-30. doi: 10.1145/272991.272995

McCullough, B. D., & Heiser, D. A. (2008). On the accuracy of statistical procedures in Microsoft Excel 2007. Computational Statistics and Data Analysis, 52(10), 4570-4578. doi: 10.1016/j.csda.2008.03.004

Microsoft. (2011). *Description of the NORMSDIST function in Excel*. Retrieved from http://support.microsoft.com/kb/827369

Montgomery, D. C. (1996), *Introduction to statistical quality control* (3rd ed.). New York: Wiley.

Nelson, L. S. (1984). Column: Technical aids: The Shewhart control chart-tests for special cases. *Journal of Quality Technology, 16*(4), 238-239.

NIST/SEMATECH. (2012). *NIST/SEMATECH e-handbook of statistical methods*. Retrieved from http://www.itl.nist.gov/div898/handbook/

Page, E. S. (1954). Continuous inspection schemes. *Biometrika, 41*(1/2), 100-115. doi: 10.2307/2333009

Roberts, C., & Casella, G. (1999). *Monte Carlo statistical methods*. New York: Springer.

Roberts, S. W. (1959). Control chart tests based on geometric moving averages. *Technometrics, 1*(3), 239-250. doi: 10.1080/00401706.1959.10489860

Ryan, T. P. (2000). *Statistical methods for quality improvement* (2nd ed.). New York: Wiley.

Salleh, S. (2013, June 27). Monte Carlo simulation tips and tricks [Web log post]. Retrieved from http://decision-analytics-blog.lumina.com/blog/monte-carlo-simulation-tips-and-tricks

Severin, T., & Schmid, W. (1998). Statistical process control and its application in finance. In G. Bol, G. Nakhaeizadeh, & K.-H. Vollmer (Eds.), *Risk measurement, econometrics and neural networks* (pp. 83-104). New York: Physica-Verlag. doi: 10.1007/978-3-642-58272-1_7

Shewhart, W. A. (1980). *Economic control of quality of manufactured product*. Wilwaukee, WI: American Society for Quality. (Original work published 1931)

Shewhart, W. A. (1986). *Statistical method from the viewpoint of quality control*. Mineola, NY: Dover. (Original work published 1939)

Srinivasan, A. (2011). *Application of information technology and statistical process control in pharmaceutical quality assurance & compliance* (Master's thesis). Retrieved from DSpace@MIT. (http://hdl.handle.net/1721.1/66047)

Stigler, S. (1973). Studies in the history of probability and statistics. XXXII: Laplace, Fisher, and the discovery of the concept of sufficiency. *Biometrika, 60*(3), 439-445. doi: 10.1093/biomet/60.3.439

Teoh, W. L., & Khoo, M. B. (2012). Optimal design of the double sampling $\bar{X}$ chart based on median run length. *International Journal of Chemical Engineering and Applications, 3*(5), 303-306. doi: 10.7763/IJCEA.2012.V3.205

Torng, C., & Lee, P. (2009, March). A modified statistical design model of double sampling $\bar{X}$ control chart. In S. I. Ao, O. Castillo, C. Douglas, D. D. Feng, & J.-A. Lee (Eds.), *Proceedings of the International Multiconference of Engineers and Computer Scientists 2009* (pp. 1761-1764). Hong Kong: International Association of Engineers. Retrieved from http://iaeng.org/publication/IMECS2009/IMECS2009_pp1761-1764.pdf

Western Electric Company. (1956). *Statistical quality control handbook*. Indianapolis: Western Electric Co.

Wheeler, D. J., & Chambers, D. S. (1992). *Understanding statistical process control* (2nd ed.). Knoxville, TN: SPC Press.

Wichmann, B. A., & Hill, I. D. (1982). Algorithm AS 183: An efficient and portable pseudo-random number generator. Applied Statistics, 31(2), 188-190. doi: 10.2307/2347988

Appendix A: Design 1 VBA Code

```
Sub Sim()
' 1. Declare Variables ------------------------------------------------
Range("A1").Select: Dim numRuns As Long, LastRow As Long, LastCol As Long,
RLStartRow As Long, RLEndRow As Long
Dim CLStartRow As Long, CLStartCol As Long, CLEndCol As Long
Dim M As Long, nsim As Long, Mu As Double, Sigma As Double, MuShift As
Long, NumCL As Long, CL As Double
Dim p1 As Double, p2 As Double, expARL As Double, expMRL As Double, expSRL
As Double
Dim Z As Long, a As Long, b As Long, c As Long, d As Long, e As Long, f
As Long
Dim zOut() As Variant, CLArray() As Double, RLOut() As Double
Dim wsOut As Range, calcOut As Range, blankOut As Range
Dim wf As WorksheetFunction: Set wf = Application.WorksheetFunction
Dim Path As String: Path = "User Sets Path Here to Save Results": Dim
xlsmExt As String: xlsmExt = ".xlsm"

' 2. Set Number of Simulation Runs -------------------------------------
numRuns = Range("B2").Value                                    'Number
of Simulation Runs (each saved separately to path above)

' 3. Start Simulation Runs---------------------------------------------
For Z = 1 To numRuns: Range("B13") = Z                        'Display Run

' 4. Delete Previous Time and Blank Count Outputs ----------------------
Range("B10:B12").Select: Selection.ClearContents 'Delete Previous Start-
Time, End-Time, & Run-Time
Range("D11").Select: Selection.ClearContents 'Delete Previous Blank Count

' 5. Delete Previous Estimates and Run-Length Outputs ------------------
LastRow = ActiveSheet.UsedRange.Rows.Count: 'Count Last Row of Worksheet
LastCol = ActiveSheet.UsedRange.Columns.Count 'Count Last Column of
Worksheet
Range(Cells(2,        6),        Cells(LastRow,        LastCol)).Select:
Selection.ClearContents 'Delete previous Estimates and RL outputs
```

```
' 6. Set Start Time ----------------------------------------------------
StartTime = "=Now()": Range("B10") = StartTime: Range("B10") =
Range("B10") ‘

' 7. Set Variables Typed  into Worksheet Inputs ------------------------
M = Range("B3").Value                                    'Series Length
nsim = Range("B4").Value                              'Number of Simulations
Mu = Range("B5").Value                    'IC Mean of Normal Distribution
Sigma = Range("B6").Value              'IC Std Dev of Normal Distribution
MuShift = Mu + Range("B7").Value * Sigma 'OC Mean of Shifted x~N(Mu+Shift,
1)

' 8. Set Range of Control Limits For ARLs Starting in Range("F3") ------
CLStartRow = 2: CLStartCol = 6 'Starting Row/Column of Control Limits
(F3)
NumCL = wf.Count(Range(Cells(1, CLStartCol), Cells(1, LastCol))) 'Count
Number of Control Limits
CLEndCol = CLStartCol + NumCL – 1       'Ending Column of Control Limits
RLStartRow = 10                        'Starting Row of Run-Length Output
RLEndRow = RLStartRow + nsim – 1        'Ending Row of Run-Length Output
Set wsOut = Range(Cells(RLStartRow, CLStartCol), Cells(RLEndRow,
CLEndCol)) 'Set Range of RL Output in Worksheet

' 9. Calculate Control Limits and OC ARLs -----------------------------
ReDim CLArray(1 To 2, 1 To NumCL)      'Re-dimension Control-Limit Array
For a = CLStartCol To CLEndCol
      CLL = wf.Norm_Inv((1 / Cells(1, a)) / 2, Mu, Sigma) 'Calculate
Lower Control Limit Value
      CLU = wf.Norm_Inv(1 - (1 / Cells(1, a)) / 2, Mu, Sigma) 'Calculate
      Upper Control Limit Value
            Cells(2, a) = CLU: Cells(3, a) = CLL 'Copy CLs into Worksheet
            CLArray(1, a - 5) = CLU: CLArray(2, a - 5) = CLL 'Copy CLs
            into CL Array
      'Calculate p1 and p2 for Expected ARL Calculation
            p1 = 1 - wf.Norm_Dist(Cells(2, a), MuShift, Sigma, True)
            'P(z>Upper CL)
```

```
            p2 = wf.Norm_Dist(Cells(3, a), MuShift , Sigma, True)
            'P(z<Lower CL)
      'Calculate Expected ARLs (will be the same for the IC Process)
            expARL = (1 / (p1 + p2))
            Cells(4, a) = Round(expARL, 2) 'Copy Expected ARLs into
            Worksheet
      'Calculate Expected MRLs (will be the same for the IC Process)
            expMRL = (wf.Ln(0.5)) / (wf.Ln(1 - (1 / Cells(4, a)))
            Cells(5, a) = Round(expMRL, 2) 'Copy Expected MRLs into
            Worksheet
      'Calculate Expected SRLs (will be the same for the IC Process)
            expSRL = ((1 - (1 / Cells(4, a))) ^ (1 / 2)) / (1 / Cells(4,
      a))
            Cells(6, a) = Round(expSRL, 2) 'Copy Expected SRLs into
            Worksheet
Next a

' 10. Start Simulations ---------------------------------------------
ReDim RLOut(1 To nsim, 1 To NumCL)        'Re-dimension Run-Length Array
For b = 1 To nsim:  Application.ScreenUpdating = False 'For Each Simulation

' 11. Fill zOut Array with z~N(Mu,Sigma) Random Numbers of Series-Length
M using NormInv Function -------------------------------------------
ReDim zOut(1 To M, 1 To 2)         'Re-dimension Array of Random Numbers
For c = 1 To M      'For Each Random Number to Be Generated in the Array
      p = Rnd            'Use Rnd Function to generate value of p, 0<p<1
            If p <= 0 Then                                  'If p<0 Then
                  p = Rnd                       'Generate new value of p
            End If
      zOut(c, 1) = wf.NormInv(p, MuShift, Sigma) 'Use NormInv Fn to fill
      Array with Random Value
      zOut(c, 2) = c     'Record Location in Array for each Random Value
Next c

' 12. Compare each z with Control Limits and Record Run-Lengths in Run-
Length Array -------------------------------------------------------
For d = 1 To NumCL                   'For Each CL in Control-Limit Array
      For e = 1 To M              'For each Random z-Value in Array zOut
```

```
            If zOut(e, 1) > CLArray(1, d) Or zOut(e, 1) < CLArray(2, d)
            Then 'If z exceeds CLs Then
                  RLOut(b, d) = zOut(e, 2) 'Record Run-Length Location
                  in Run-Length Array
      Exit For                                    'Exit and Move to Next CL
      Else: End If
      Next e                                           'Else Move to Next z
Next d
'-------------------------------------------------------------------------
Next b

' 13. Copy Run-Length Array into Worksheet ----------------------------
wsOut.Value = RLOut: Application.ScreenUpdating = True 'Copy Run-Length
Array into Worksheet Range

' 14. Calculate Estimated ARLs, MRLs and SRLs -------------------------
For f = CLStartCol To CLEndCol: Set calcOut = Range(Cells(RLStartRow, f),
Cells(RLEndRow, f))
      Cells(7, f) = wf.Average(calcOut)                    'Calculate ARL
      Cells(8, f) = wf.Median(calcOut)                     'Calculate MRL
      Cells(9, f) = wf.StDev(calcOut)                      'Calculate SRL
Next f

' 15. Count Blank Run-lengths -----------------------------------------
Set blankOut = Range(Cells(RLStartRow, CLStartCol), Cells(RLEndRow,
CLStartCol))
Range("D11") = wf.CountBlank(blankOut) 'Count Simulations with Blank Run-
Lengths

' 16. Set End Time and Calculate Run Time -----------------------------
EndTime = "=Now()": Range("B11") = StartTime: Range("B11") = Range("B11")
Range("B12") = Range("B11") - Range("B10")

' 17. Save Workbook and Do Next Run Z ---------------------------------
Range("A1").Select: ActiveWorkbook.SaveAs Path & "-M=" & M & "-Z=" & Z &
xlsmExt 'Save Workbook
Next Z                                                       'Next Run
End Sub
```

Appendix B: Design 2 VBA Code

```
Sub Sim()
' 1. Declare Variables ------------------------------------------------
Dim LastRow As Long, LastCol As Long, RLStartRow As Integer, RLCountRow
As Integer, MaxRow As Long
Dim CLStartCol As Integer, CLEndCol As Integer, CLStartRow As Integer,
numRuns As Long, M As Long
Dim Mu As Double, Sigma As Double, MuShift As Double, NumCL As Integer,
CL As Double, p1 As Double, p2 As Double
Dim expARL As Double, expMRL As Double, expSRL As Double, Z As Long, a As
Long, c As Long, d As Long, e As Long
Dim f As Long, g As Long, h As Long, i As Long, RL2 As Long, RL1 As Long,
RLDiff As Long, ZOut() As Variant
Dim RLArray() As Variant, CLArray() As Variant, ROut As Long, MaxCount As
Long, MaxRL() As Long, RLCount As Long
Dim calcOut As Range, wf As WorksheetFunction: Set wf =
Application.WorksheetFunction
Dim Path As String: Path = "User Sets Path to Save Results": Dim xlsmExt
As String: xlsmExt = ".xlsm"

' 2. Set Number of Simulation Runs -------------------------------------
numRuns = Range("B2").Value 'Number of Simulation Runs (each saved
separately to path above)

' 3. Start Simulation Runs ---------------------------------------------
For Z = 1 To numRuns: Range("B13") = Z                    'Display Run

' 4. Delete Previous Time and Blank Count Outputs ----------------------
Range("B9:B11").Select: Selection.ClearContents 'Delete Previous Start-
Time, End-Time, & Run-Time

' 5. Delete Previous Estimates and Run-Length Outputs ------------------
LastRow = ActiveSheet.UsedRange.Rows.Count: 'Count Last Row of Worksheet
LastCol = ActiveSheet.UsedRange.Columns.Count 'Count Last Column of
Worksheet
Range(Cells(2,        6),        Cells(LastRow,        LastCol)).Select:
Selection.ClearContents 'Delete previous Estimates and RL outputs
```

```
' 6. Set Start Time ----------------------------------------------------
StartTime = "=Now()": Range("B9") = StartTime: Range("B9") = Range("B9")
'

'7. Set Variables Typed  into Worksheet Inputs -------------------------
M = Range("B3").Value                                   'Series Length
Mu = Range("B4").Value                    'IC Mean of Normal Distribution
Sigma = Range("B5").Value              'IC Std Dev of Normal Distribution
MuShift = Mu + Range("B6").Value * Sigma 'OC Mean of Shifted x~N(Mu+Shift,
1)

' 8. Set Range of Control Limits For ARLs Starting in Range("F3") ------
CLStartRow = 2: CLStartCol = 6 'Starting Row/Column of Control Limits
(F3)
NumCL = wf.Count(Range(Cells(1, CLStartCol), Cells(1, LastCol))) 'Count
Number of Control Limits
CLEndCol = CLStartCol + NumCL - 1       'Ending Column of Control Limits
RLCountRow=10                                  'Row of Run-Length Counts
RLStartRow = 12                         'Starting Row of Run-Length Output
MaxRow=1048576                               'Last Row in Excel Worksheet

' 9. Calculate Control Limits and OC ARLs -----------------------------
ReDim CLArray(1 To 2, 1 To NumCL)      'Re-dimension Control-Limit Array
For a = CLStartCol To CLEndCol
       CLL = wf.Norm_Inv((1 / Cells(1, a)) / 2, Mu, Sigma) 'Calculate
       Lower Control Limit Value
       CLU = wf.Norm_Inv(1 - (1 / Cells(1, a)) / 2, Mu, Sigma) 'Calculate
       Upper Control Limit Value
              Cells(2, a) = CLU: Cells(3, a) = CLL 'Copy CLs into Worksheet
              CLArray(1, a - 5) = CLU: CLArray(2, a - 5) = CLL 'Copy CLs
              into CL Array
       'Calculate p1 and p2 for Expected ARL Calculation
              p1 = 1 - wf.Norm_Dist(Cells(2, a), MuShift, Sigma, True)
              'P(z>Upper CL)
              p2 = wf.Norm_Dist(Cells(3, a), MuShift , Sigma, True)
              'P(z<Lower CL)
       'Calculate Expected ARLs (will be the same for the IC Process)
              expARL = (1 / (p1 + p2))
```

```
            Cells(4, a) = Round(expARL, 2) 'Copy Expected ARLs into
            Worksheet
      'Calculate Expected MRLs (will be the same for the IC Process)
            expMRL = (wf.Ln(0.5)) / (wf.Ln(1 - (1 / Cells(4, a)))
            Cells(5, a) = Round(expMRL, 2) 'Copy Expected MRLs into
            Worksheet
      'Calculate Expected SRLs (will be the same for the IC Process)
            expSRL = ((1 - (1 / Cells(4, a))) ^ (1 / 2)) / (1 / Cells(4,
      a))
            Cells(6, a) = Round(expSRL, 2) 'Copy Expected SRLs into
            Worksheet
Next a

' 10. Start Simulations - Fill ZOut Array with z~N(Mu,Sigma) Random
Numbers of Series-Length M using NormInv Function----------------------
ReDim ZOut(1 To M, 1 To 2): Application.ScreenUpdating = False 'Re-
dimension Run-Length Array
For c = 1 To M      'For Each Random Number to Be Generated in the Array
      p = Rnd           'Use Rnd Function to generate value of p, 0<p<1
            If p <= 0 Then                                  'If p<0 Then
                  p = Rnd                       'Generate new value of p
            End If
      zOut(c, 1) = wf.NormInv(p, MuShift, Sigma) 'Use NormInv Fn to fill
      Array with Random Value
      zOut(c, 2) = c     'Record Location in Array for each Random Value
Next c

' 11. Compare each z with Control Limits and Record Run-Lengths in Run-
Length Array ------------------------------------------------------------
For d = 1 To NumCL                  'For Each CL in Control-Limit Array
      Rout = 11                                 'Set Row Output to Row 11
      For e = 1 To M              'For each Random z-Value in Array zOut
            If ROut = MaxRow Then 'If Row Out Exceeds Excel's Max Row
            Length
      Exit For   'Exit For Loop If Row Out Exceeds Excel's Max Row Length
            ElseIf zOut(e, 1) > CLArray(1, d) Or zOut(e, 1) < CLArray(2,
            d) Then 'If z exceeds CLs Then
            Rout = Rout + 1                   'Increment Row Output by 1
```

```
            Cells(ROut, d).Value = e          'Record Run-Length Location
            End If
      Next e                                  'Else Move to Next z-value
Next d                                                  'Move to Next CL

' 12. Calculate the Count of Each Run Length for Each CL & Store in Row
10 ------------------------------------------------------------------
For f = CLStartCol To CLEndCol                          'For Each CL
      RLCount = wf.CountA(Range(Cells(RLStartRow, f), Cells(MaxRow, f)))
      'Count Number of Non-Zero Run-Lengths
      Cells(RLCountRow, f) = RLCount     'Copy Count into Worksheet Cells
Next f                                                  'Move to Next CL

' 13. Iteratively Subtract Subsequent RL from Previous RL to Calculate
Indexed Run-Lengths ----------------------------------------------------
LastRow = ActiveSheet.UsedRange.Rows.Count  'Count Last Row of Worksheet
MaxCount = wf.Max(Range(Cells(RLCountRow, CLStartCol), Cells(RLCountRow,
CLEndCol)))
ReDim RLArray(1 To MaxCount, 1 To NumCL)    'Re-dimension Run-Length Array
For g = CLStartCol To CLEndCol                          'For Each CL
      RLCount = Cells(RLCountRow, g)      'Set Count from Worksheet Cells
      For h = 1 To RLCount                      'For Each Row/Column Cell
            RL2 = Cells(h + RLCountRow + 1, g).Value 'Set RL2 = to
            Subsequent Run-Length Value
            RL1 = Cells(h + RLCountRow, g).Value 'Set RL1 = to Previous
            Run0-Length Value
            RLDiff = RL2 - RL1            'Calculate Difference Rl2-Rl1
            RLArray(h, g - (CLStartCol - 1)) = RLDiff 'Copy RLDiff into
            RLArray
      Next h                                'Move to Next Row/Column Cell
Next g                                                  'Move to Next CL

' 14. Copy Run-Length Array into Worksheet ----------------------------
Range(Cells(RLStartRow, CLStartCol), Cells(MaxCount + RLCountRow + 1,
CLEndCol)) = RLArray

' 15. Find & Replace 0 Run-Lengths with Null String --------------------
LastRow = ActiveSheet.UsedRange.Rows.Count    'Count Last Row of Worksheet
```

```
Range(Cells(RLCountRow + 1, CLStartCol), Cells(LastRow, LastCol)).Select
'Find & Replace 0 Run-Lengths
      Selection.Replace  What:="0",  Replacement:="",  LookAt:=xlWhole,
      SearchOrder:=xlByColumns, MatchCase:=True _
      SearchFormat:=False, ReplaceFormat:=False: Range("A1").Select

' 16. Calculate Estimated ARLs, MRLs and SRLs -------------------------
For i= CLStartCol To CLEndCol
RLCount = Cells(RLCountRow, i).Value         'Record RL Count for Each CL
Set calcOut = Range(Cells(RLStartRow, i), Cells(RLCount + RLCountRow + 1,
i)) 'Set Range of Run-Length Output
      Cells(7, i) = wf.Average(calcOut)                  'Calculate ARL
      Cells(8, i) = wf.Median(calcOut)                   'Calculate MRL
      Cells(9, i) = wf.StDev(calcOut)                   'Calculate SRL
Next i

' 17. Set End Time and Calculate Run Time ------------------------------
EndTime = "=Now()": Range("B10") = StartTime: Range("B10") = Range("B10")
Range("B11") = Range("B10") - Range("B9")

' 18. Save Workbook and Do Next Run Z ----------------------------------
Range("A1").Select
ActiveWorkbook.SaveAs Path & "-" & M & "-" & Z & xlsmExt    'Save Workbook
Next Z                                                          'Next Run
End Sub
```