LBNL-1003516

ERNEST ORLANDO LAWRENCE

BERKELEY NATIONAL LABORATORY

Random Numbers Generation for Lifecycle-Cost Analysis of Energy Consuming Equipment

Helcio Blum, Edson Okwelum

Energy Analysis and Environmental Impacts Division

October, 2015

This work was supported by the Office of Energy Efficiency and Renewable Energy (Solar Technologies Office) of the U.S. Department of Energy under Lawrence Berkeley National Laboratory Contract No. DE-AC02-05CH1131

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, or The Regents of the University of California.

Ernest Orlando Lawrence Berkeley National Laboratory is an equal opportunity employer.

Random Numbers Generation for Lifecycle-Cost Analysis of Energy Consuming Equipment

Helcio Blum, Edson Okwelum

Energy Efficiency Standards Group, Energy Analysis and Environmental Impacts Division,

Energy Technologies Area, Lawrence Berkeley National Laboratory

1 Cyclotron Rd Bld 90, Berkeley CA, 94720

Abstract

Lifecycle Cost Analysis has a key role in the development of energy efficiency standards for appliances, lighting, and other equipment. The analysis includes Monte Carlo simulations to estimate – from a sample of thousands of consumers with diverse profiles – the net-present value of all, lifetime costs associated with a piece of equipment to provide a certain amount of energy service. This report introduces a set of pseudo-random number generation functions developed in MS Excel VBA to support those simulations. It further compares the effectiveness of these functions to the effectiveness of similar ones available in a broadly used commercial software program. The effectiveness of the functions is evaluated from 17 sets of 1000 samples, each sample comprised of 10,000 pseudo-random numbers drawn from custom uniform, normal, triangular, Weibull and categorical distributions. All samples pass three relevant tests of pseudo-randomness, namely long period, uniformity and independence. The samples also prove to be statistically equivalent to their corresponding peers generated by the commercial software program.

Random Number Generation for

Lifecycle-Cost Analysis of Energy Consuming Equipment

Table of Contents

1. Introduction 5
2. Lifecycle Cost Analysis of Energy Consuming Equipment 5
3. Random Number Generation Functions 6
4. Effectiveness Evaluation 8
5. Conclusions 14

References 15
Appendix A: VBA Source Codes 17
Appendix B: Density and QQ Plots of EES Samples 20
Appendix C: Kolmogorov-Smirnov Test 28
Appendix D: Kullback-Leibler Divergence 34
Appendix E: Chi-Square Goodness-of-Fit Test 40
Appendix F: Equivalence and Non-Inferiority Tests 43
Appendix G: R Source Code 47

List of Tables

Table 1: Analytical forms of quantile functions 8
Table 2: Samples description (continuous distributions) 12
Table 3: Samples description (categorical distributions) 12
Table A.4: Parameters to the Functions 17
Table C.5: Descriptive statistics of the distributions of KS statistics 29
Table C.6: Equivalence evaluated from the KS test 33
Table D.7: Descriptive statistics of the distributions of KL statistics 34
Table D.8: Equivalence evaluated from the KL divergence test 39
Table E.9: Descriptive statistics of the distributions of $\chi 2$ statistics 40
Table E.10: Equivalence evaluated from the $\chi 2$ test 41
Table F.11: Results from equivalence test for KS statistics 44
Table F.12: Results from equivalence test for KL statistics 45
Table F.13: Results from equivalence test for $\chi 2$ statistics 45
Table F.14: Results from non-inferiority tests 46
Table G.15: R Programs 47

List of Figures

Figure 1: Graphical distributions of sequences of paired* pseudo-random numbers 11
Figure 2: Two-stage benchmark process 13

Figure B.3: Histogram and QQ plot of the samples generated from the uniform distribution 20
Figure B.4: Histograms and QQ plots of the samples generated from the normal distributions 21
Figure B.5: Histograms and QQ plots of the samples generated from the symmetric and bounded triangular distributions 22
Figure B.6: Histograms and QQ plots of the samples generated from the asymmetric triangular distributions 23
Figure B.7: Histograms and QQ plots of the samples generated from the one-side triangular distributions 24
Figure B.8: Histograms and QQ plots of the samples generated from the Weibull distributions W_a and W_b 25
Figure B.9: Histograms and QQ plots of the samples generated from the Weibull distributions W_c and W_d 26
Figure B.10: Mass distribution of the samples generated from the categorical distributions 27
Figure C.11: Density distribution and QQ plots of KS statistics for the Uniform Distribution 29
Figure C.12: Density distribution and QQ plots of KS statistics for Normal Distributions 30
Figure C.13: Density distribution and QQ plots of KS statistics for Triangular Distributions 30
Figure C.14: Density distribution and QQ plots of KS statistics for Triangular Distributions 31
Figure C.15: Density distribution and QQ plots of KS statistics for Weibull Distributions 32
Figure D.16: Density distribution and QQ plots of KL statistics for the Uniform Distribution 35
Figure D.17: Density distribution and QQ plots of KL statistics for Normal Distributions 36
Figure D.18: Density distribution and QQ plots of KL statistics for Triangular Distributions 36
Figure D.19: Density distribution and QQ plots of KL statistics for Triangular Distributions 37
Figure D.20: Density distribution and QQ plots of KL statistics for Weibull Distributions 38
Figure E.21: Density distribution and QQ plots of the $\chi 2$ statistics for Categorical Distributions 42

List of Boxes

Box A.1: VBA Source Code for Function eesUniform 18
Box A.2: VBA Source Code for Function eesNormal 18
Box A.3: VBA Source Code for Function eesTriangular 18
Box A.4: VBA Source Code for Function eesWeibull 19
Box A.5: VBA Source Code for Function eesCateg 19
Box G.6: R Source Code for the KS Analysis 48
Box G.7: R Source Code for the KL Analysis 49
Box G.8: R Source Code for the Chi-Square Analysis 51
Box G.9: R Source Code for the Non-Inferiority Test 52

Random Number Generation for Lifecycle Cost Analysis of Energy Consuming Equipment

Helcio Blum, Edson Okwelum

1. Introduction

The Energy Efficiency Standards Group (EESG)[1] at LBNL produces technical, economic, and environmental analyses to support the U.S. Department of Energy (DOE) in the development of energy efficiency standards for appliances, lighting, and other equipment. The analyses evaluate the potential benefits from new energy efficiency standards that would shift the market for an energy service towards more efficient equipment. Lifecycle Cost (LCC) Analysis is used as the underlying methodological framework to evaluate those benefits at the level of individual consumers. The benefits are estimated from changes in the *net-present value* (NPV) of all, lifetime costs associated with a particular appliance, lighting device or type of equipment, to provide a certain amount of energy service under different scenarios of minimum energy performance standards (*meps*). Lifetime costs will vary across consumers, as they are likely to use the equipment with different intensity and be located in different regions of the country. The latter may eventually affect the equipment retail price, its usage, and the energy price paid to operate the equipment. Monte Carlo simulation is used to generate a sample of consumers with a diversity of socio-economic and geographical profiles. The sample is then used to estimate the impacts from new *meps* on each consumer according to its specific profile.

This report introduces a set of pseudo-random number generation functions especially developed to support Monte Carlo simulation, as part of the LCC analysis, using Microsoft Excel (MS Excel). In the following, we describe the methodology used to perform the LCC analysis, and introduce the pseudo-random number generation functions developed. We then describe and report results from the process used to evaluate the functions, which includes statistical tests and the benchmark of the functions against a broadly used commercial software program.[2] We finally conclude with a summary of the experiment and results obtained. A set of seven appendices provide additional, detailed information that includes the source code of the functions, as well as the methods, results and tools used in the statistical tests.

2. Lifecycle Cost Analysis of Energy Consuming Equipment

Lifecycle Cost Analysis is a method to assess cost-effectiveness of investments. It is also known as *whole cost accounting* or *total cost of ownership* (when the investment refers to a good). The method balances initial costs with lifetime costs associated with the investment. In this case, (a) initial costs refer to equipment purchase, as well as shipment and installation costs, and (b) lifetime costs include equipment maintenance and repair costs, as well as energy costs, all estimated over the life of a unit. The lifecycle

[1] The EESG is a research group with the Energy Analysis and Environmental Impacts Division, Energy Technologies Area, Lawrence Berkeley National Laboratory (https://ees.lbl.gov/).
[2] The commercial software used to benchmark the MS Excel pseudo-random number generators introduced in this report is Oracle's Crystal Ball (http://www.oracle.com/us/products/applications/crystalball/overview/index.html).

cost LCC_c incurred by a consumer c to enjoy the energy service provided by a unit of a certain type of energy consuming equipment is calculated as:

$$LCC_c = TIC_c + \sum_{i=1,LT_c} OPER_c \quad [1]$$

where:

$$TIC_c = RET_c + INST_c \quad [1a]$$

$$OPER_c = MR_c + EC_c * P_c \quad [1b]$$

and:

LCC_c	Lifecycle cost incurred by consumer c,
TIC_c	Total installed cost incurred by consumer c,
LT_c	Lifetime (in years) of the unit operated by consumer c,
$OPER_c$	Total lifetime operating costs incurred by consumer c,
RET_c	Retail price paid by consumer c,
$INST_c$	Installation costs incurred by consumer c,
MR_c	Annual maintenance and repair costs incurred by consumer c,
EC_c	Annual energy consumption of the unit operated by consumer c,
P_c	Energy price paid by consumer c.

In the LCC analysis performed during the development of new energy efficiency standards, Equation [1] is evaluated for a sample comprised of thousands of consumers. For each consumer, LT_c and all variables in Equations [1a] and [1b] are sampled from custom probability distributions, which typically rely on the normal, uniform, triangular, Weibull or categorical distributions, and for which long sequences of random numbers need to be generated.

3. Random Number Generation Functions

Random number generators have been used for a long time. They seek at generating sequences of numbers or symbols that lack any pattern. In the past, sequences of random numbers were generated, for example, from dice, coins and spinning wheels. With the advent of computers, several algorithms for generating random numbers have been proposed. They have been classified as *pseudo*-random number generators (pRNG), as they rely on deterministic algorithms to produce *apparently random* sequences of

numbers that meet certain distribution properties.[3] According to Hellekalek (1998) "there are no 'safe' [pseudo-random number] generators." Nevertheless, the sequences of numbers they generate have been considered random enough to support simulations.

We rely on the *inverse transformation method* (Devroye, 1986) to generate sequences of pseudo-random numbers w_i sampled from five types of probability distributions: uniform, normal, triangular, Weibull and categorical. The method has been broadly used for sampling numbers at random from any probability distribution given its *quantile function* (QF), which is the inverse of its *cumulative distribution function* (CDF). The idea is to sample a sequence of numbers u_i uniformly distributed between 0 and 1, and use them as a probability to calculate w_i such that:[4]

$$\Pr(x \in \mathbb{R} | x \leq w_i) = F(w_i) = u_i \qquad [2]$$

where F is the CDF of the distribution for which the pseudo-random numbers w_i are to be generated. The method works as follows: (a) draw a random number u_i from the standard uniform distribution ($u_i \sim U(0,1)$); (b) compute w_i such that $F(w_i) = u_i$; (c) take w_i as the pseudo-random number drawn from the distribution described by F.

A stream of LCC_c values can be calculated from Equation [1] using this approach to generate sequences of pseudo-random numbers according to the probability distributions of the variables involved in that calculation. Let F_v be the cumulative distribution function that describes a variable v in Equations [1], [1a] or [1b], and $Q_v \equiv F_v^{-1}$. The equations can be rewritten to estimate the LCC_c of a sampled consumer as:

$$LCC_c = TIC_c + \sum\nolimits_{i=1, Q_{LT}(u_{LT})} OPER_c \qquad [3]$$

$$TIC_c = Q_{RET}(u_{RET}) + Q_{INST}(u_{INST}) \qquad [3a]$$

$$OPER_c = Q_{MR}(u_{MR}) + Q_{EC}(u_{EC}) * Q_P(u_P) \qquad [3b]$$

where $0 \leq u_v < 1$ is drawn from the standard uniform distribution.

In Equations [3], [3a] and [3b], the variables that comprise the LCC calculation are represented as random numbers, which can be sampled from different, custom probability distributions using the inverse transformation method. A computational function to generate random numbers uniformly distributed between 0 and 1, as well as one that implements the quantile functions of the desired probability distributions are then necessary.

[3] In fact, repeatability – the ability to repeat a sequence of pseudo-random numbers – is an appreciated characteristic of a pRNG for the application to which the functions introduced in this report were developed. As the LCC spreadsheets developed by the EESG are publicly deployed, they enable other parties to perform the same LCC analysis, which are expected to lead to same results. Using sequences of true (rather than pseudo-) random numbers would lead to similar, statistically equivalent results, yet not exactly the same.

[4] The inverse method is recommended by Hormann (1993) when generating non-uniform pseudo-random numbers from uniform distributions, particularly when the uniform pseudo-random numbers are generated using the linear-congruential method (LCG). See Section 4 for more on the algorithm and other characteristics of LCG.

MS Excel provides two standard uniform distribution pRNG: (a) function RAND(), available at spreadsheet level, and (b) function RND(), available in Microsoft Visual Basic for Applications (VBA). We rely on RND() to obtain random numbers $u_i \in \{\mathbb{R}|0 \leq u_i < 1\}$. The random numbers u_i are then used as the argument to a QF, according to the type and other additional characteristics of the probability distribution from which a pseudo-random number w_i is to be sampled.[5] Table 1 presents the analytical forms of the QFs corresponding to the probability distributions covered in this report. The table includes the formulas of the distributions and the names of the corresponding pRNG functions developed in MS Excel VBA. Appendix A presents the VBA source code of these functions.

Table 1: Analytical forms of quantile functions

Distribution	**Analytical Form**[a]	**Proposed Function**
Uniform	$u \cdot (b-a) + a$	eesUniform
Normal[b,c]	MS Excel function NORM.INV	eesNormal
Triangular[c]	$\begin{cases} \sqrt{u \cdot (b-a) \cdot (c-a)} + a, \ u < (c-a)/(b-a) \\ b - \sqrt{(1-u) \cdot (b-a) \cdot (b-c)}, \ u \geq (c-a)/(b-a) \end{cases}$	eesTriangular
Weibull[c]	$\theta + \lambda \cdot \sqrt[k]{-ln(1-u)}$	eesWeibull
Categorical[d]	$\begin{cases} C_1, \ u < \Pr(C_1) \\ C_k, \ \sum_{i=1,k-1} \Pr(C_i) \leq u < \sum_{i=1,k} \Pr(C_i) \ \text{for } k = 2..K \end{cases}$	eesCateg

[a] In the formulas, a and b represent the lower and upper limits of a uniform and triangular distributions; c represents the mode of a triangular distribution; and θ, λ and k represent respectively the location, scale and shape of a Weibull distribution.

[b] There is no analytical form for the quantile function of the normal distribution. One can approximate it using the sum: $\sum_{n \geq 0} 2\pi^{\frac{2n+1}{2}} \frac{C_{2n+1}}{(2n+1)!} \left(x - \frac{1}{2}\right)^{2n+1}$ where $C_1 = 1, C_3 = 1, C_5 = 7, C_7 = 127$ ... (Dominici, 2003). We rely on an MS Excel native function to calculate u_i given z.

[c] A bounded version of this distribution is also available, and refers to the special case of the former distribution where the pseudo-random numbers w_i generated are further subject to $L \leq w_i \leq U$, where L and U are respectively the lower and upper bounds of the range of pseudo-random numbers to be generated. The lower and upper bounds are implemented within the corresponding pRNG function.

[d] C_i is the i-th of the K categories in the distribution, and $\Pr(C_i)$ is the probability associated to the i-th category.

4. Effectiveness Evaluation

The effectiveness of the pseudo-random number sequences generated by a pRNG depends on the quality of both the standard uniform pRNG that underlies the inverse transformation method and the implementation of the inverse transformation method itself. The evaluation of the pRNG functions introduced in this report (hereafter referred to as 'the proposed functions') is therefore developed in two steps: First, the MS Excel VBA RND() function, which is the building block of the proposed functions, is tested for three basic requirements for any standard uniform pRNG; then, the pseudo-random number

[5] In case of normal distribution, we replace any $u_i = 0$ by a new sampled random number $u_i' > 0$.

sequences generated by the proposed functions are benchmarked based on goodness-of-fit and other statistical tests.

Evaluating the RND() Function

The basic requirements for a standard uniform pRNG are that the sequences of pseudo-random numbers they generate present long period, uniformity and independence. The long period requirement refers to the (undesirable) existence of cycles within sequences generated by the pRNG. The RND() function develops upon the *linear-congruential method* (LCG) to produce sequences of pseudo-random numbers between 0 and 1 (Microsoft, 2004). The LCG generates sequences of uniformly distributed pseudo-random numbers based on the formula:

$$X_{n+1} = (a \cdot X_n + c) \bmod m \qquad [4]$$

where X_0 is the sequence's *seed*, a is the *multiplier*, c is the *increment*, and m is the *modulus*.

The resulting sequence can be normalized to the interval [0, 1) by dividing the pseudo-random numbers by m. Depending on the parameters used in an LCG algorithm, the period of a sequence is at most equals to the parameter *modulus* of the LCG formula. According to Hull and Dobell (1962), the conditions for an LCG sequence to have full period[6] are:

(i) c is relatively prime to m,

(ii) $a \equiv 1 \pmod{p}$ if p is a prime factor or m, and

(iii) $a \equiv 1 \pmod{4}$ if 4 is a factor of m.

When m is a power of 2, it is only necessary to have c odd and $a \equiv 1 \pmod{4}$ to meet the requirements above (Hull and Dobell, 1962). The MS Excel VBA RND() function uses the following parameters for the LCG algorithm: a=1140671485, c=12820163, and m=2^{24} (Microsoft, 2004). We can therefore conclude that the RND() function is most likely to generate full period sequences,[7] which are much longer than the typical 10,000-trial samples used in the LCC analysis. We nevertheless create 1000 samples V_j of 10,000 pseudo-random numbers $v_{i,j}$ to test it. No repeated values were found, either within or across all samples V_j. Since all samples were generated in a single spreadsheet and at the same time, one can also assume that they comprise a single sequence of 1000 times 10,000 numbers, for which no repeated values were found.

The uniformity requirement for a standard uniform pRNG refers to how evenly the pseudo-random numbers generated are distributed. To test RND() for uniformity we (a) use the 1000 samples V_j of pseudo-random numbers in the interval [0,1) created above, (b) regress the empirical cumulative distribution $F(V_j)$ of each sample V_j against a sequence U of 10,000 consecutive numbers uniformly distributed between 0 and 1, and (c) test the regression coefficients for their ideal, expected values that would indicate that the samples V_j meet the uniformity requirement. We use the following linear regression model:

[6] A period equals to the *modulus*.
[7] W. Santy has empirically found that for the RND() function in VB the period is 2^{24}. See Kyd (2011).

$$F(V_j) = \alpha_j + \beta_j \cdot U + \varepsilon_j \quad [5]$$

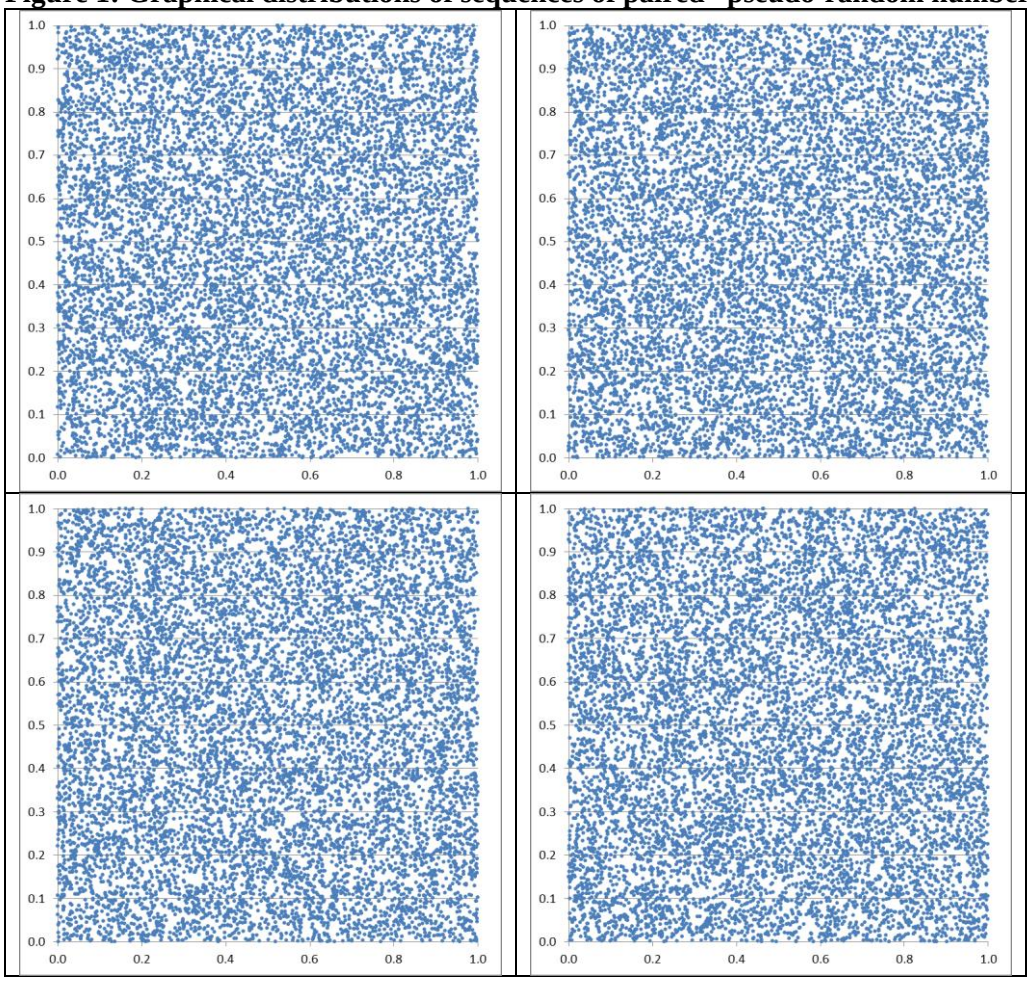
where $F(V_j)$ is the cumulative distribution of V_j, $U = [0.0001, 0.0002, \ldots, 1]'$ is a vector of 10,000 monotonically increasing, uniformly distributed numbers, and ε_j is the vector of statistical errors corresponding to sample V_j. The linear regression in model [5] attempts to measure how accurately the cumulative distribution of the empirical sequence V_j approximates the cumulative distribution of the theoretical, standard uniform distribution. Ideally $\alpha_j = 0$, $\beta_j = 1$ and $\varepsilon_j = 0$, $\forall j$.

We estimate the 1000 values of α_j and β_j, and their corresponding standard-errors, t-Stat values and p-values. The intercepts α_j range from -0.01180 to 0.01278, with mean equals to -0.00015 and 95% confidence interval [-0.00724, 0.00694]. The average standard-error of α_j is 0.00005. The t-Stat values range from -190 to 186, with mean equals to -3. The p-values range from 0 to 0.975, with mean 0.007. For 98.5% of the samples the p-value is equal or lower than 0.05, which indicates that the intercept in model [5] – despite being very small – matters. However, a one-sample *t*-test of the 1000 values of α_j, to test the null-hypothesis that their mean is equal to zero, leads to a p-value equals to 0.17. For a significance level of 0.05 the null-hypothesis cannot be rejected. We therefore assume that $\alpha_j = 0$ at that significance level.

The coefficients β_j range from 0.98552 to 1.01233, with mean equals to 1.00003 and 95% confidence interval [0.99116, 1.00891]. The average standard-error of β_j is 0.00009. The t-Stat values range from 4614 to 26406, with mean equals to 12695. All p-values are equal to zero. A one-sample *t*-test of the 1000 values of β_j, to test the null-hypothesis that their mean is equal to one, leads to a p-value equals to 0.81. For a significance level of 0.05 the null-hypothesis cannot be rejected. We therefore assume that $\beta_j = 1$ at that significance level. Based on these two statistical results we assume that the cumulative distributions of the 1000 samples V_j approximate the cumulative distribution of a standard uniform distribution and, consequently, that RND() meets the uniformity requirement.

We finally evaluate whether the sequences generated by RND() meet the independence requirement. The independence requirement refers to the inexistence of any autocorrelation between the values of a sample. All samples generated by the proposed functions have close to zero autocorrelation with lags varying from 1 to 40. Hence, we assume the samples generated by the proposed functions meet the independence requirement. Figure 1 exemplifies how a sequence of pairs $z = (v_{i,1}, v_{i+lag,1})$, derived from sample V_1, is graphically distributed for lag values equal to 1, 2, 5 and 10 (Petrila et al, 2014). In all cases, the sequenced pairs uniformly cover the entire range and no pattern can be visually identified, which reinforces the inexistence of correlation between sequenced numbers with different lags. Based on the results above, we conclude that the MS Excel VBA RND() function, after being tested for long period, uniformity and independence, meets relevant requirements of pseudo-randomness for a standard uniform pRNG. It is therefore a robust resource to supply the proposed functions with sequences of pseudo-random numbers drawn from the standard uniform distribution.

Figure 1: Graphical distributions of sequences of paired* pseudo-random numbers



* Lags are 1, 2, 5 and 10, from left to right and top-down.

Benchmarking the Proposed Functions

We proceed with the evaluation process by benchmarking a set of custom pseudo-random number sequences generated by the proposed functions against similar sequences generated by a commercial software program (hereafter referred to as 'the commercial software').[2] We use both pRNG tools to create 17 sets of 1000 samples, each sample comprised by 10,000 pseudo-random numbers drawn from a custom distribution. Table 2 and Table 3 describe the 17 sets of custom samples generated. Appendix B presents probability density and QQ plots for all samples for the continuous distributions in Table 2, and probability mass plots for the categorical ones in Table 3. The plots also include the corresponding theoretical distributions.

The benchmark follows a two-stage process. First we use goodness-of-fit statistical tests to benchmark the samples against their corresponding theoretical distributions, precisely calculated from the distributions' analytical forms. Then, we evaluate – for each statistical test – if the distributions of the goodness-of-fit statistic calculated for the proposed functions and the commercial software in the first stage can be assumed to come from the same population, in which case we conclude that the proposed

functions are as effective as the commercial software to produce pseudo-random number sequences for that type of distribution. This is possible under the assumption that the results from benchmarking the 1000 samples generated by each tool, for a given type of distribution, against their corresponding theoretical distribution, represent a stochastic measure of the effectiveness of the tools to produce samples of pseudo-random numbers distributed according to that type of distribution. Figure 2 illustrates the two-stage benchmark process.

Table 2: Samples description (continuous distributions)

Distribution		**Description**
Uniform	U	$\text{Lower} = -1, \text{Upper} = +1$
Normal	N	$\text{Mean} = 0, \text{SD} = 1$
	N_B	$\text{Mean} = 0, \text{SD} = 1, \text{Bounded to } [-1.5, +2.0]$
Triangular	T	$\text{Lower} = -1, \text{Mode} = 0, \text{Upper} = +1$
	T_B	$\text{Lower} = -1, \text{Mode} = 0, \text{Upper} = +1, \text{Bounded to } [-0.25, +0.75]$
	T_R	$\text{Lower} = -1, \text{Mode} = -0.5, \text{Upper} = +1$
	T_L	$\text{Lower} = -1, \text{Mode} = +0.5, \text{Upper} = +1$
	T_D	$\text{Lower} = -1, \text{Mode} = -1, \text{Upper} = +1$
	T_U	$\text{Lower} = -1, \text{Mode} = +1, \text{Upper} = +1$
Weibull	W_a	$\text{Location} = +1, \text{Shape} = 1.5, \text{Scale} = 5$
	W_b	$\text{Location} = +1, \text{Shape} = 2, \text{Scale} = 10$
	W_c	$\text{Location} = +1, \text{Shape} = 2, \text{Scale} = 15$
	W_d	$\text{Location} = +1, \text{Shape} = 3, \text{Scale} = 20$

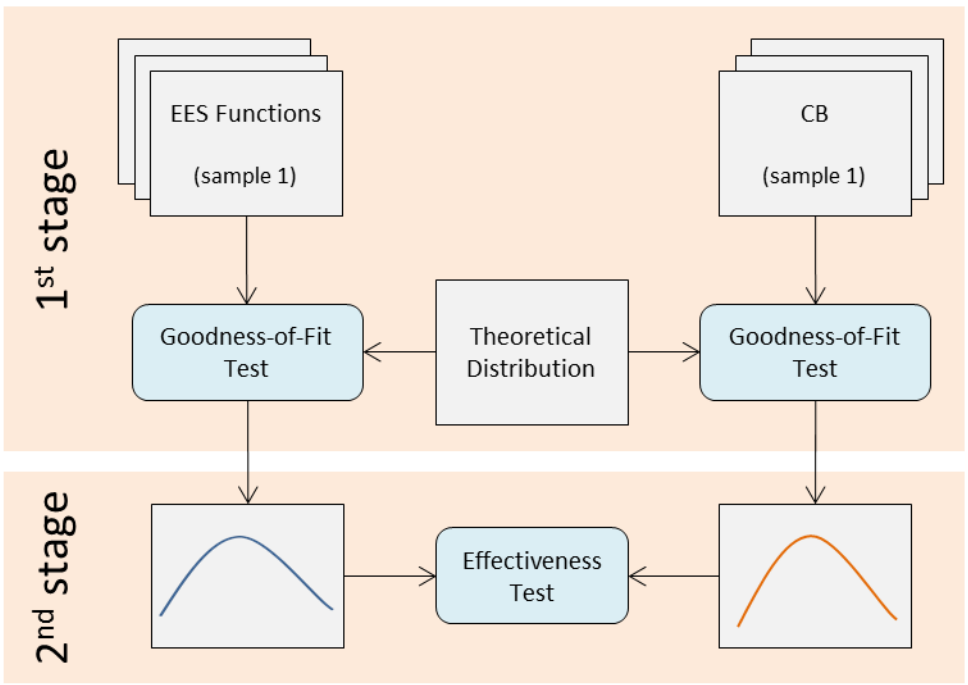
Table 3: Samples description (categorical distributions)

Distribution	**Frequencies by Category**									
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
C_2	0.4	0.6								
C_3	0.2	0.5	0.3							
C_5	0.1	0.2	0.1	0.5	0.1					
C_{10}	0.130	0.011	0.107	0.138	0.094	0.121	0.109	0.115	0.046	0.129

We rely on three goodness-of-fit statistical tests for the first stage. For the samples generated from the continuous distributions we use the *Kolmogorov-Smirnov* (KS) test and the *Kullback-Leibler* (KL) divergence. The two methods are broadly used to compare samples of continuous numbers, and present relevant characteristics for this benchmark: Whereas the KS test expresses similarity based on a worst

case approach, looking at the highest deviation between the two cumulative distributions being compared, the KL divergence expresses similarity based on an integral approach, accounting for deviations in the probability functions all over the range of the distributions. As for the samples generated from the categorical distributions, we use the *Chi-Square Goodness-of-fit* (χ^2) test to assess how well each sample fits to its corresponding theoretical distribution.

Figure 2: Two-stage benchmark process



Note: In the first stage we use a goodness-of-fit statistical test to compare each of the 1000 samples generated by each tool with its corresponding theoretical distribution. This leads to 1000 values of the statistical test for each tool. We use two different statistical tests in that stage for the continuous distributions, and one test for the categorical ones. The results from each goodness-of-fit test expresses the effectiveness of the tool in generating sequences that meet the characteristics of the distribution to which the sequences are expected to fit. In the second stage we compare the effectiveness of each tool by testing if the distributions of the 1000 results from each statistical test calculated for each tool can be assumed to be statistically equivalent.

In the second stage of the benchmark process, we use a variant of the KS test to compare the distributions of the 1000 statistics calculated for the samples generated by the proposed functions and the commercial software for each custom distribution. We then complete the benchmark process with the *equivalence*- and *non-inferiority* tests, where we assess – based on the effectiveness of each tool calculated from each of the statistical tests above – whether the proposed functions can be considered, respectively, equivalent and no worse than the commercial software. Appendix C, D and E describe the theoretical approach and present detailed results for the KS, KL and χ^2 statistical tests. Appendix F

describes the equivalence and non-inferiority tests, and presents results for those tests. All statistical analyses are developed in R (R Core Team, 2015). Appendix G presents the R source codes.

Results show that the proposed functions can be considered equivalent to the commercial software within an equivalence interval ranging from less than 0.1% to 2.4%, depending on the distribution type and the statistical test. In a few cases, the statistical tests calculated for the proposed functions result in a mean value that is higher than the corresponding mean calculated for the commercial software. This could indicate that, for those cases, the proposed functions are not as effective as the commercial software. However, for all those cases, the proposed functions are estimated to be non-inferior than the latter for a margin of non-inferiority equals to 0.1%, and with statistical significance lower than 0.001.

5. Conclusions

LBNL performs LCC analysis for DOE using MS Excel spreadsheets. As part of the LCC analysis, a commercial software program is currently used to generate pseudo-random numbers in support to Monte Carlo simulations. MS Excel VBA provides a pseudo-random number generator function that produces pseudo-random sequences of real numbers between 0 and 1. Using the inverse transformation method these sequences can be translated into sequences of pseudo-random numbers sampled from other custom distributions. This report introduces a set of pseudo-random generation functions coded in MS Excel VBA. The functions rely on the inverse transformation method and the MS Excel VBA pseudo-random number generator function RND() to produce samples of real numbers or categorical data distributed according to a select set of types of distributions.

The functions are evaluated from two perspectives. First, we test the MS Excel VBA RND() function, which is the building block of the proposed functions, for three relevant requirements for pseudo-randomness: long period, uniformity and independence. The RND() function passes the three tests. Consequently, we conclude that the sequences generated by the proposed functions can also be considered pseudo-randomly distributed.

We then evaluate the effectiveness of the proposed functions, for which we generate two groups of 17 sets of 1000 samples, with each sample comprised by 10,000 random numbers drawn from a custom distribution. The two groups refer to (a) samples generated from the proposed functions introduced in this report, and (b) samples generated from the commercial software program currently used by the EESG. The samples generated from each tool for 13 continuous distributions are evaluated with the Kolmogorov-Smirnov test and the Kullback-Leibler divergence measure. The samples generated from each tool for the four categorical distributions are evaluated with the Chi-Square Goodness-of-Fit test. The distributions of the 1000 values of these three statistics, calculated for each sample generated by each tool, are then compared using the two-sample Kolmogorov-Smirnov test. Results from the latter test show that the distributions of the statistics calculated for the samples generated from each tool are statistically equivalent.

We further compare the differences of the means of the distributions of the three statistics calculated for the samples generated from each tool using the equivalence and the non-inferiority tests. Results for each of the statistics show that the proposed functions can be considered as effective as and non-inferior

than the ones embedded in the commercial software program used to benchmark them, in generating samples of pseudo-random numbers distributed according to the custom distributions they implement. The proposed functions are, in addition, transparent and publicly available, and can be used in MS Excel spreadsheets as an alternative to the latter.

Acknowledgments. This work was supported by the Office of Energy Efficiency and Renewable Energy (Solar Technologies Office) of the U.S. Department of Energy under Lawrence Berkeley National Laboratory Contract No. DE-AC02-05CH1131. We acknowledge Scott Young, LBNL, for his contribution to the design of the evaluation process, as well as Alex B. Lekov and Samir Touzani, LBNL, for their valuable comments on a draft version of this report.

References

Christensen, E (2007): "Methodology of superiority vs. equivalence trials and non-inferiority trials." *Journal of Hepatology* 46 (5) :947-954.

Devroye, Luc (1986): *Non-uniform random variate generation*. Springer-Verlag, New York, nY, USA.

Dominici, D E (2003): "The Inverse of the Cumulative Standard Normal Probability Function, Integral Transforms and Special Functions." *Integral Transforms and Special Functions* 14 (4): 281-292.

Hellekalek, P (1998): "Good random number generators are (not so) easy to find." Mathematics and Computers in Simulation 46 (5-6): 485-505.

Hormann, W (1993): *The Quality of Non-uniform Random Numbers*. Preprint Series #7, Department of Applied Statistics and Data Processing, Wirtschaftsuniversitat Wien, Vienna, Austria. http://statistik.wu-wien.ac.at/[8]

Hull, T E & Dobell, A R (1962): "Random number generators." SIAM Review 4 (3): 230-254.

Kyd, C (2011): "Excel Random Numbers … A Reader's Comment." ExcelUser.com. http://exceluser.com/blog/508/excel-random-numbers-a-readers-comment.html[8]

Kullback, S & Leibler, R A (1951): "On information and sufficiency." *Annals of Mathematical Statistics* 22 (1): 79-86.

Lesaffre, E (2008): "Superiority, Equivalence, and Non-Inferiority Trials." Bulletin of the NYU Hospital for Joint Diseases 66 (2): 150-154.

Massey, F J Jr (1951): "The Kolmogorov-Smirnov Test for Goodness of Fit." *Journal of the American Statistical Association* 46 (253): 68-78.

Microsoft (2004): "INFO: How Visual Basic Generates Pseudo-Random Numbers for the RND Function." Article ID: 231847. Last Review: June 24, 2004. Revision: 3.0. Microsoft Corporation, WA, USA. http://support.microsoft.com/en-us/kb/231847[8]

[8] Webpage accessed in April, 2015.

NIST/SEMATECH (2013): *e-Handbook of Statistical Methods* (Last updated: 10/30/2013). National Institute of Standards and Technology, U.S. Department of Commerce. http://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm[8]

Petrila, I, Manta, V and Ungureano, F (2014): "Uniformity and Correlation Test Parameters for Random Number Generators." *Proceedings of the 18th International Conference on Systems Theory, Control and Computing.* Sinaia, Romania.

Pettitt, A N & Stephens, M A (1977): "The Kolmogorov-Smirnov Goodness-of-Fit Statistic with Discrete and Grouped Data." *Technometrics* 19 (2): 205-210.

R Core Team (2015): *R*: *A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/[8]

Appendix A: VBA Source Codes

Table A.4 describes the parameters to the random number generation function. Box A.1 through Box A.5 present the VBA source code of the functions.

Table A.4: Parameters to the Functions

Distribution	MS Excel Function	Parameter	
		Name[a]	Description
Uniform	eesUniform	**Lower**	Lower bound
		Upper	Upper bound
Normal	eesNormal	**Mean**	Mean
		SD	Standard deviation
		lowerCut[b]	Lower limit
		Uppercut[b]	Upper limit
Triangular	eesTriangular	**Lower**	Lower bound
		Upper	Upper bound
		Mode	Likeliest value
		lowerCut[b]	Lower limit
		uppercut[b]	Upper limit
Weibull	eesWeibull	**Shp**	Shape
		Scl	Scale
		Loc	Location
		lowerCut[b]	Lower limit
		uppercut[b]	Upper limit
Categorical	eesCateg	**Labels**	List of categories[c]
		Frequency	List of frequencies[c]

[a] Parameters in bold are mandatory.
[b] Lower and upper limits apply to bounded distributions only.
[c] Both lists should have same number of elements.

Box A.1: VBA Source Code for Function eesUniform

```
Function eesUniform(Lower, Upper) As Double
 eesUniform = Lower + Rnd() * (Upper - Lower)
End Function
```

Box A.2: VBA Source Code for Function eesNormal

```
Function eesNormal(Mean, SD, Optional lowerCut = -1E+32, Optional upperCut = 1E+32) As Double
 cummLow = (lowerCut - Mean) / SD
 cummUp = (upperCut - Mean) / SD
 myRND = Rnd()
 zRnd = Application.WorksheetFunction.NormDist(cummLow, Mean, SD, True) + _
        IIf(myRND = 0, Rnd(), myRND) * (Application.WorksheetFunction.NormDist(cummUp, Mean, SD, True) -
Application.WorksheetFunction.NormDist(cummLow, Mean, SD, True))
 eesNormal = Application.WorksheetFunction.NormInv(zRnd, Mean, SD) * SD + Mean
End Function
```

Box A.3: VBA Source Code for Function eesTriangular

```
Function eesTriangular(Lower, Upper, Optional Mode, Optional lowerCut, Optional upperCut) As Double
 If IsMissing(Mode) Or IsEmpty(Mode) Then Mode = (Lower + Upper) / 2
 Mode = Application.WorksheetFunction.Max(Lower, Application.WorksheetFunction.Min(Upper, Mode))
 If IsMissing(lowerCut) Then lowerCut = Lower
 lowerCut = Application.WorksheetFunction.Max(Lower, Application.WorksheetFunction.Min(Upper, lowerCut))
 If IsMissing(upperCut) Then upperCut = Upper
 upperCut = Application.WorksheetFunction.Max(Lower, Application.WorksheetFunction.Min(Upper, upperCut))
 If Mode = Lower Then
  cummLow = 0
 Else
  cummLow = (lowerCut - Lower) ^ 2 / ((Upper - Lower) * (Mode - Lower))
 End If
 If Mode = Upper Then
  cummUp = 1
 Else
  cummUp = 1 - (Upper - upperCut) ^ 2 / ((Upper - Lower) * (Upper - Mode))
 End If
 zRnd = cummLow + Rnd() * (cummUp - cummLow)
 If zRnd < (Mode - Lower) / (Upper - Lower) Then
  x = Sqr(zRnd * (Upper - Lower) * (Mode - Lower)) + Lower
 Else
  x = Upper - Sqr((1 - zRnd) * (Upper - Lower) * (Upper - Mode))
 End If
 eesTriangular = x
End Function
```

Box A.4: VBA Source Code for Function eesWeibull

```
Function eesWeibull(Shp, Scl, Optional Loc = 0, Optional lowerCut = -1E+32, Optional upperCut = 1E+32) As Double
 x = Loc + Scl * (-Log(1 - Rnd())) ^ (1 / Shp)
 While x < lowerCut Or x > upperCut
  x = Loc + Scl * (-Log(1 - Rnd())) ^ (1 / Shp)
 Wend
 eesWeibull = x
End Function
```

Box A.5: VBA Source Code for Function eesCateg

```
Function eesCateg(Labels, Frequency) As Variant
 myRandom = Rnd()
 Dim p As Double
 p = Frequency(1)
 i = 1
 While myRandom >= p
  i = i + 1
  p = p + Frequency(i)
 Wend
 If IsMissing(Labels) Then
  eesCateg = i
 Else
  eesCateg = Labels(i)
 End If
End Function
```

Appendix B: Density and QQ Plots of EES Samples

This appendix presents histograms and QQ plots from the 1000 samples generated by the proposed functions for each of the continuous distributions listed in Table 2 (Figure B.3 to Figure B.9), and probability mass plots for each of the categorical distributions in Table 3 (Figure B.10). The shaded areas show the variability of results across samples. The histograms also include a red curve (red dot markers in case of categorical distributions) that represents the theoretical distribution against which each sample is benchmarked in Section 4. In the QQ-plots, the horizontal axis refers to the quantiles of the samples generated by the proposed functions, and the vertical one to the theoretical distributions.

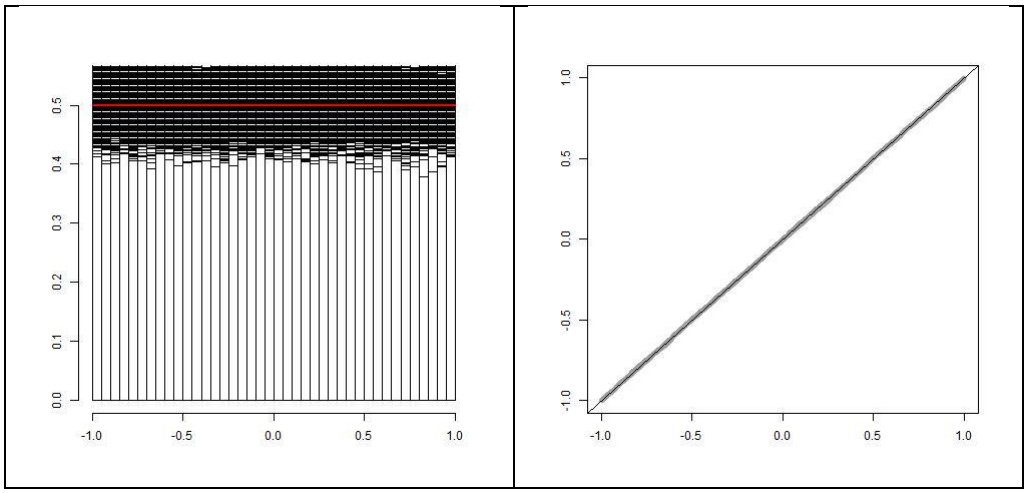


Figure B.3: Histogram and QQ plot of the samples generated from the uniform distribution

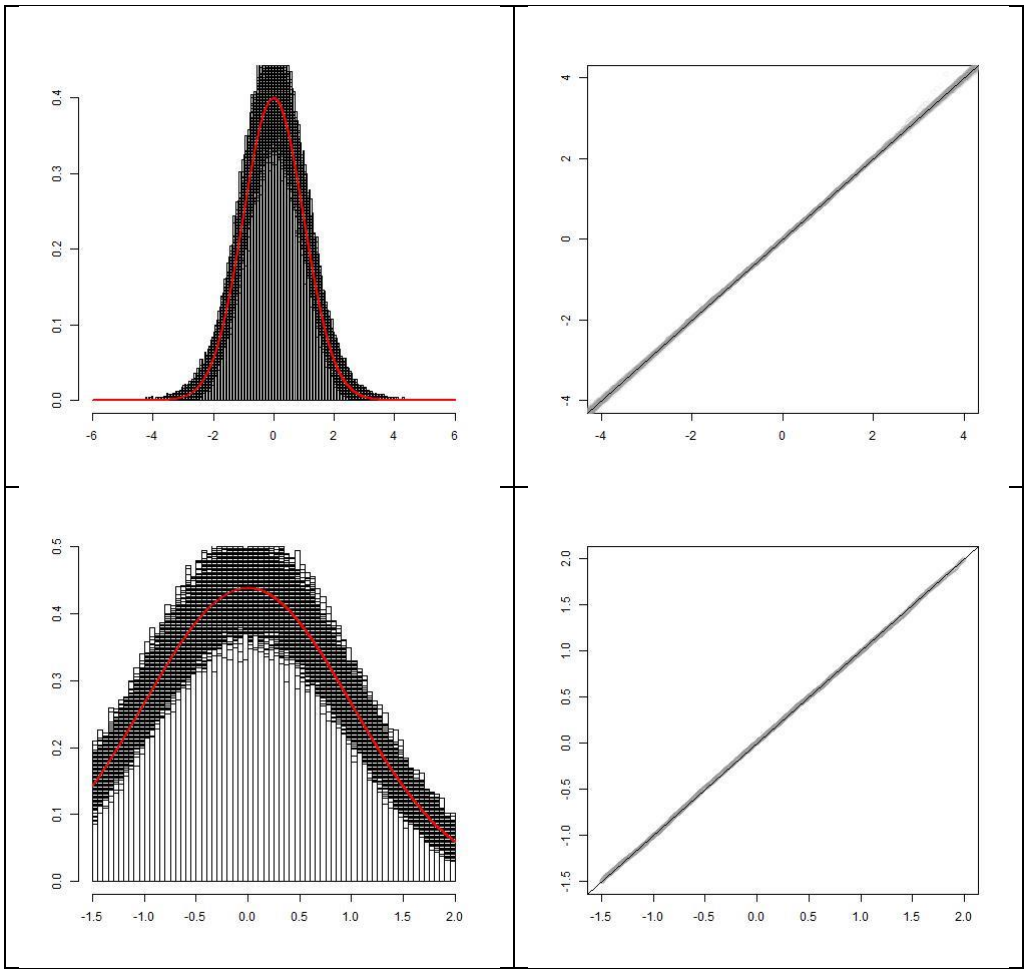


Figure B.4: Histograms and QQ plots of the samples generated from the normal distributions
Top to bottom: Distributions N and N_B.

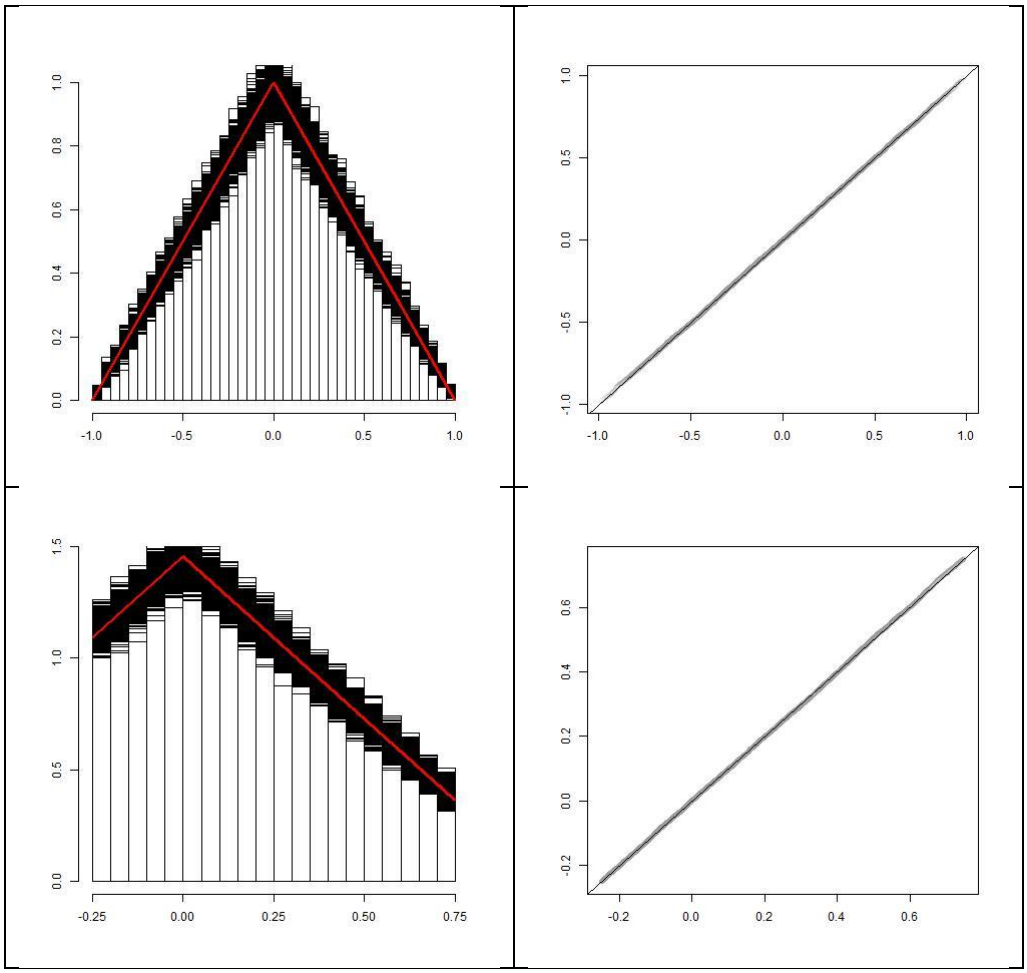


Figure B.5: Histograms and QQ plots of the samples generated from the symmetric and bounded triangular distributions

Top to bottom: Distributions T and T_B.

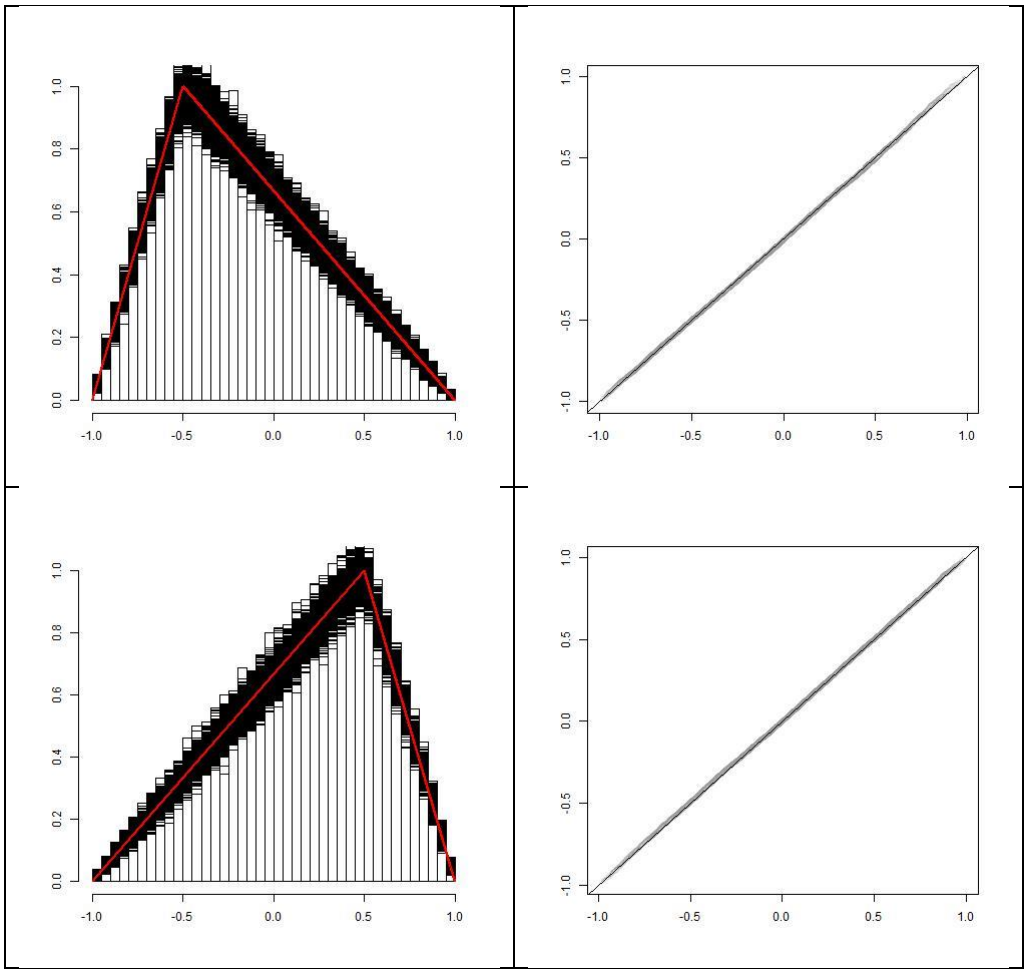


Figure B.6: Histograms and QQ plots of the samples generated from the asymmetric triangular distributions

Top to bottom: Distributions T_R and T_L.

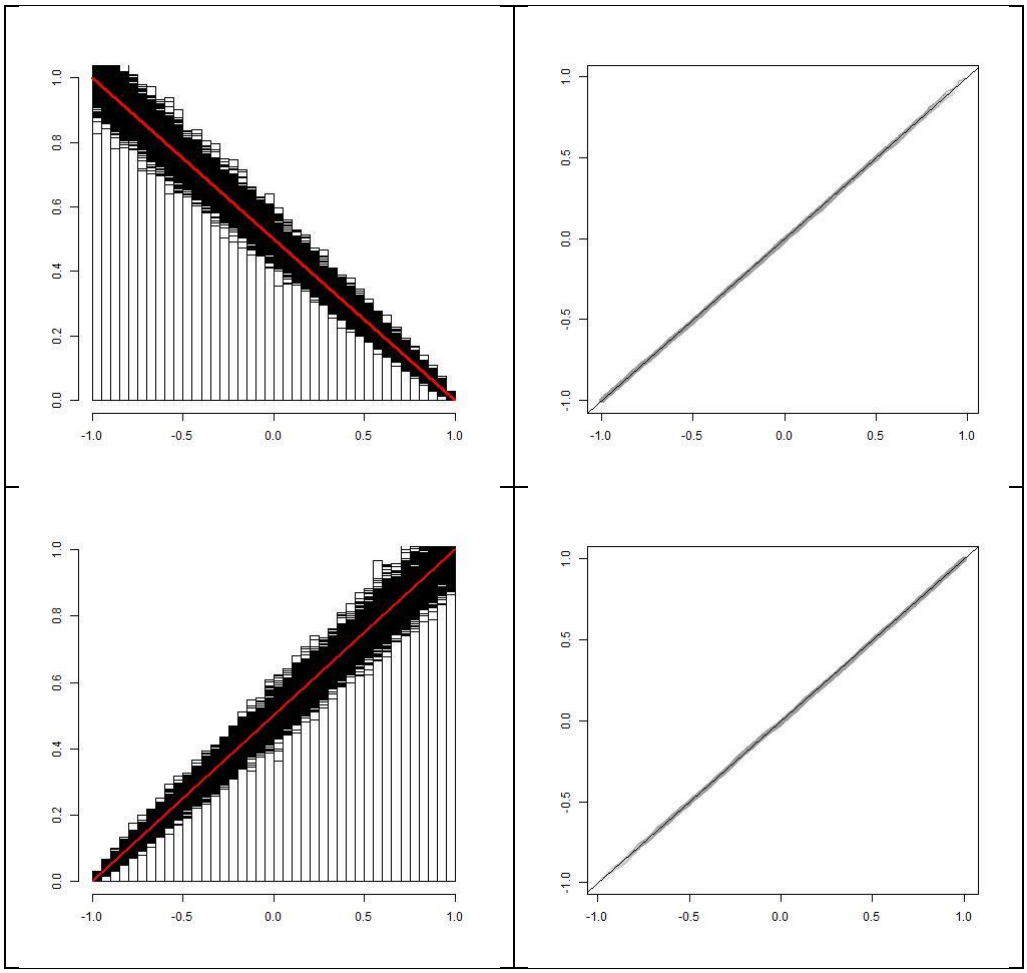


Figure B.7: Histograms and QQ plots of the samples generated from the one-side triangular distributions

Top to bottom: Distributions T_D and T_U.

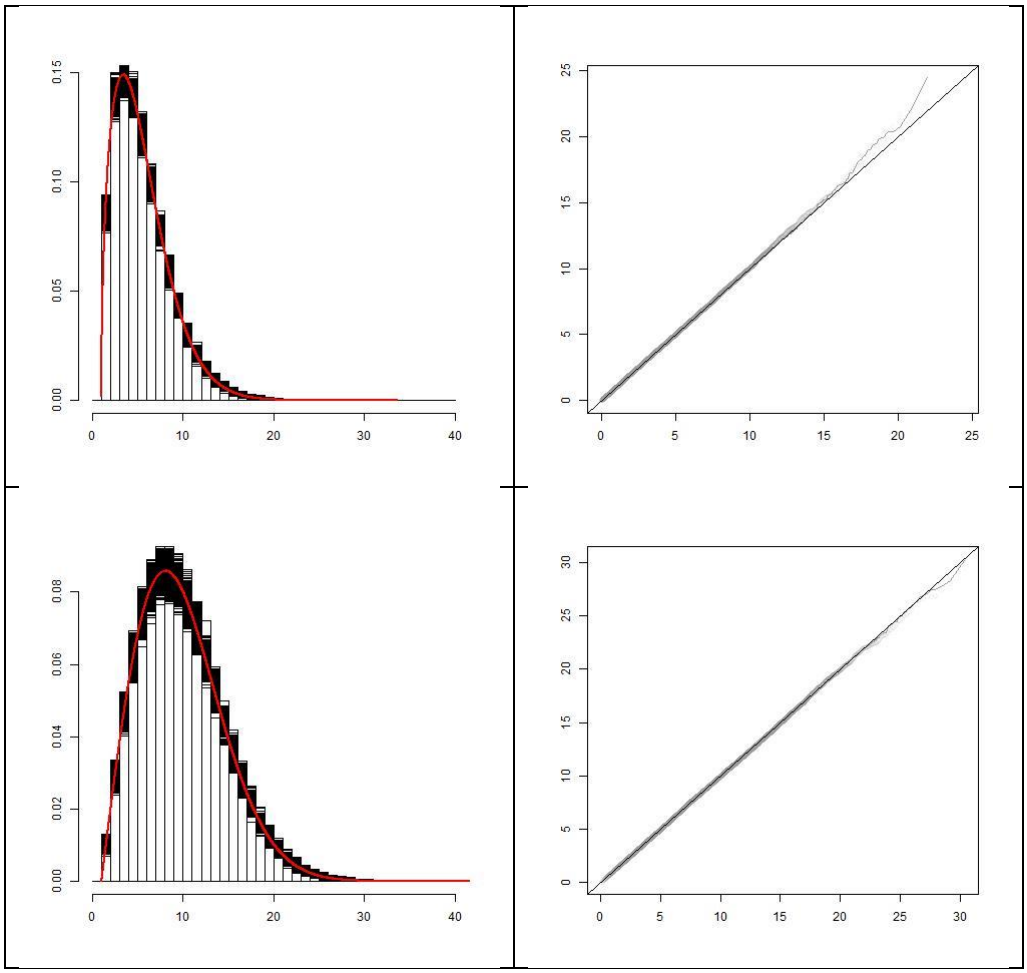


Figure B.8: Histograms and QQ plots of the samples generated from the Weibull distributions W_a and W_b

Top to bottom: Distributions W_a and W_b.

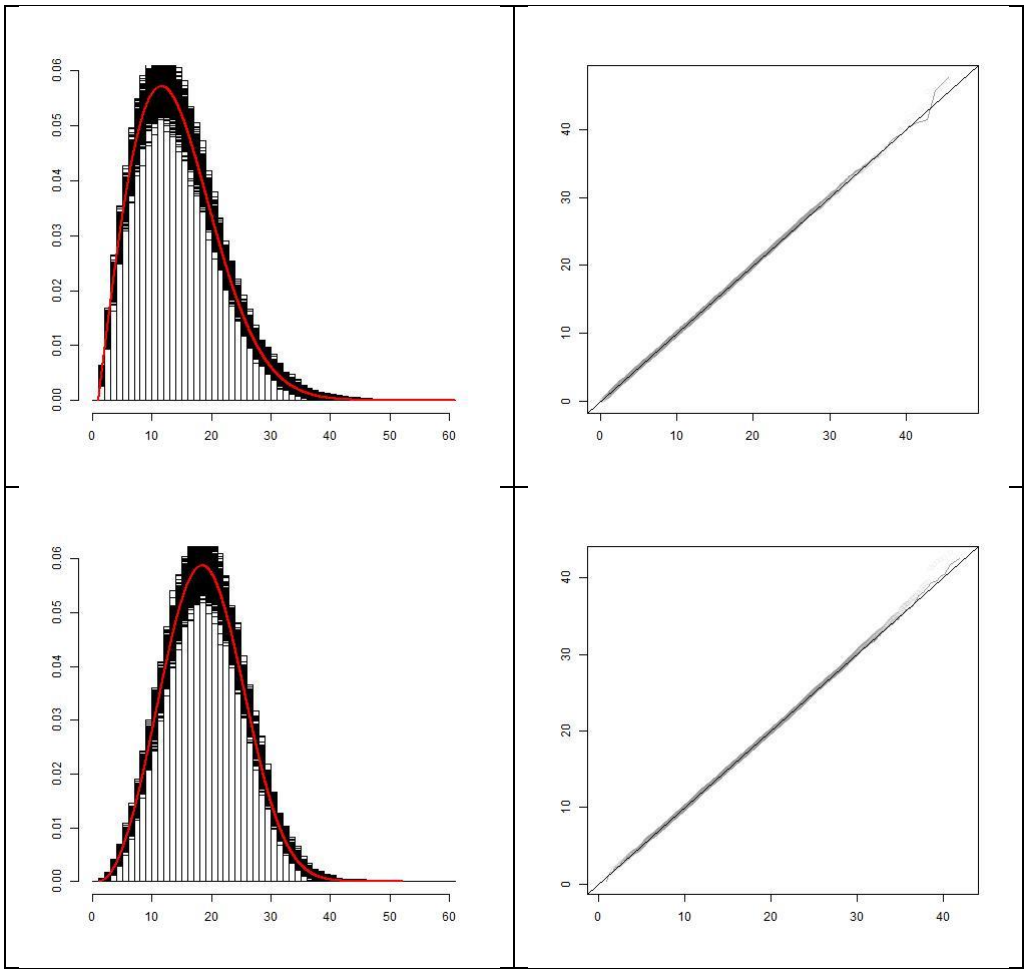


Figure B.9: Histograms and QQ plots of the samples generated from the Weibull distributions W_c and W_d

Top to bottom: Distributions W_c and W_d.

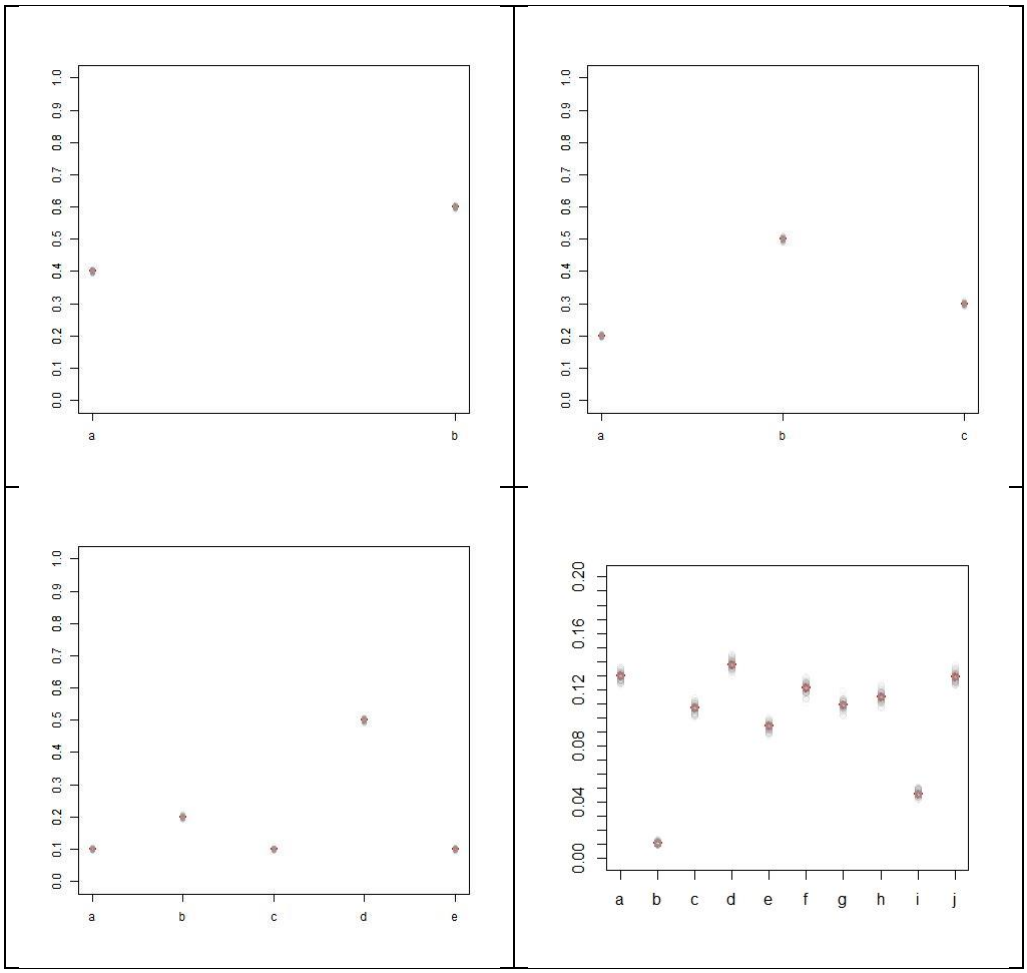


Figure B.10: Mass distribution of the samples generated from the categorical distributions

Top row, left to right: Distributions C_2 and C_3.

Bottom row, left to right: Distributions C_5 and C_{10}.

Appendix C: Kolmogorov-Smirnov Test

The KS test[9] is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to compare: (a) a sample with a reference probability distribution (one-sample KS test), or (b) two samples (two-sample KS test). The KS statistic D quantifies a distance between the distributions being compared. In case of the one-sample KS test, the statistic refers to the distance between the empirical distribution function underlying the sample and the CDF of the reference distribution. In case of the two-sample KS test, the statistic measures the distance between the empirical distributions of the two samples. In both cases, the distance is calculated as the maximum absolute difference between the CDFs of the two distributions being compared. We calculate the one-sample KS statistics $D_{d,s}^{E}$ and $D_{d,s}^{C}$ for each of the samples $W_{d,s}^{E} = \left[w_{d,s,i}^{E}\right]$ and $W_{d,s}^{C} = \left[w_{d,s,i}^{C}\right]$, $s = 1, \dots, 1000$ generated respectively by the proposed functions and the commercial software, for each continuous distribution d listed in Table 2 as:

$$D_{d,s}^{E} = \sup_{w_{s,i}^{E}} \left| F_{d}^{E}\left(W_{d,s}^{E}\right) - F_{d}\left(W_{d,s}^{E}\right) \right| \qquad [6a]$$

$$D_{d,s}^{C} = \sup_{w_{s,i}^{C}} \left| F_{d}^{C}\left(W_{d,s}^{C}\right) - F_{d}\left(W_{d,s}^{C}\right) \right| \qquad [6b]$$

where $F_{d}^{E}\left(W_{d,s}^{E}\right)$ and $F_{d}^{C}\left(W_{d,s}^{C}\right)$ are respectively the CDFs of the samples $W_{d,s}^{E}$ and $W_{d,s}^{C}$, and $F_{d}\left(W_{d,s}^{E}\right)$ and $F_{d}\left(W_{d,s}^{C}\right)$ the CDFs of the reference distribution d calculated for the same values that comprise the samples $W_{d,s}^{E}$ and $W_{d,s}^{C}$.

We proceed with the benchmark based on the KS test by comparing – for each distribution d in Table 2 – the distributions of the 1000 values of $D_{d,s}^{E}$ with the distributions of the 1000 values of $D_{d,s}^{C}$. Table C.5 presents descriptive statistics for distributions $D_{d,s}^{E}$ and $D_{d,s}^{C}$. Figure C.11 to Figure C.15 compare the distributions of $D_{d,s}^{E}$ and $D_{d,s}^{C}$. Results show that the distributions of the KS statistics calculated from samples generated by the proposed functions and the commercial software have mostly the same shape, with very close mean, median, and lower- and upper bound values.

[9] For more details see, for example, Massey (1951).

Table C.5: Descriptive statistics of the distributions of KS statistics

Dist (d)	**Proposed Functions** $(D^E_{d,s})$				**Commercial Software** $(D^C_{d,s})$			
	Min	Median	Max	Mean	Min	Median	Max	Mean
U	0.00369	0.00829	0.02000	0.00865	0.00386	0.00806	0.01992	0.00848
N	0.00368	0.00816	0.02249	0.00867	0.00362	0.00806	0.01909	0.00849
N_B	0.00510	0.01160	0.03050	0.01222	0.00490	0.01155	0.02820	0.01222
T	0.00349	0.00822	0.01791	0.00857	0.00395	0.00833	0.02180	0.00876
T_B	0.00369	0.00813	0.02203	0.00862	0.00349	0.00829	0.02083	0.00873
T_R	0.00394	0.00814	0.01864	0.00861	0.00407	0.00832	0.01789	0.00874
T_L	0.00388	0.00828	0.02122	0.00874	0.00381	0.00842	0.01994	0.00872
T_D	0.00338	0.00819	0.01985	0.00857	0.00344	0.00822	0.01972	0.00869
T_U	0.00338	0.00819	0.01985	0.00857	0.00335	0.00810	0.02177	0.00859
W_a	0.00375	0.00824	0.01995	0.00864	0.00329	0.00812	0.02064	0.00859
W_b	0.00343	0.00820	0.01933	0.00866	0.00418	0.00852	0.02113	0.00888
W_c	0.00335	0.00818	0.01758	0.00865	0.00342	0.00839	0.01981	0.00877
W_d	0.00369	0.00813	0.02203	0.00862	0.00393	0.00824	0.01990	0.00862

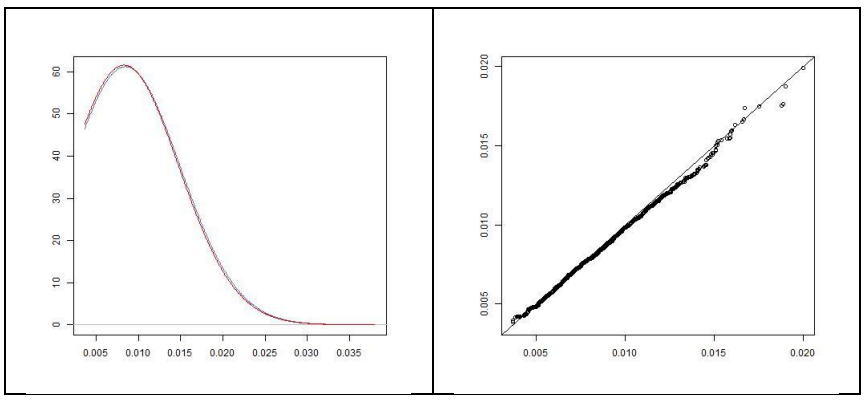


Figure C.11: Density distribution and QQ plots of KS statistics for the Uniform Distribution
Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

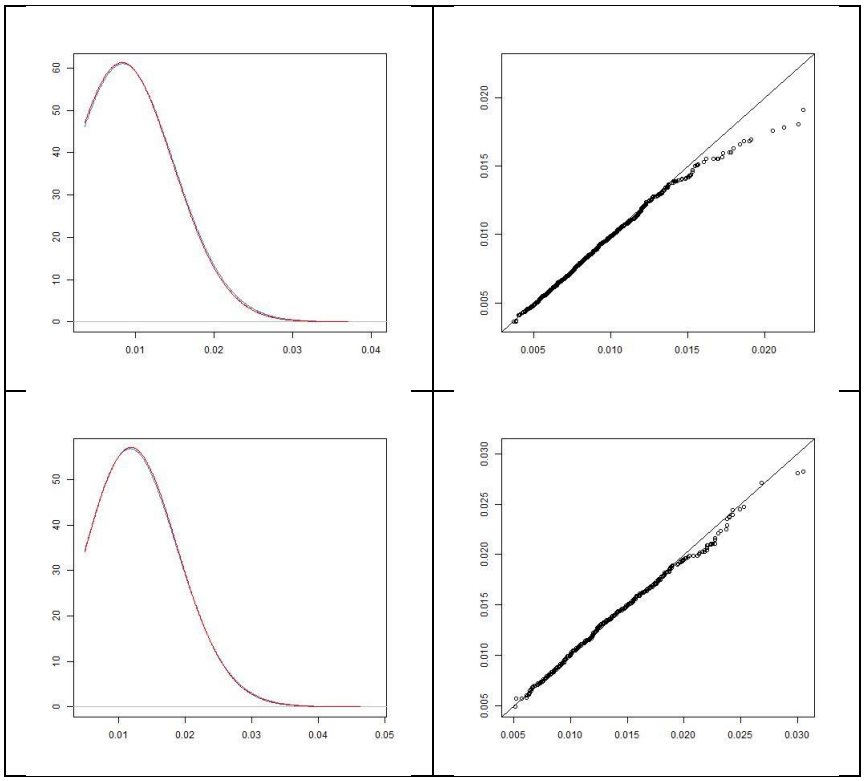


Figure C.12: Density distribution and QQ plots of KS statistics for Normal Distributions
From top to bottom: Distributions N and N_B. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

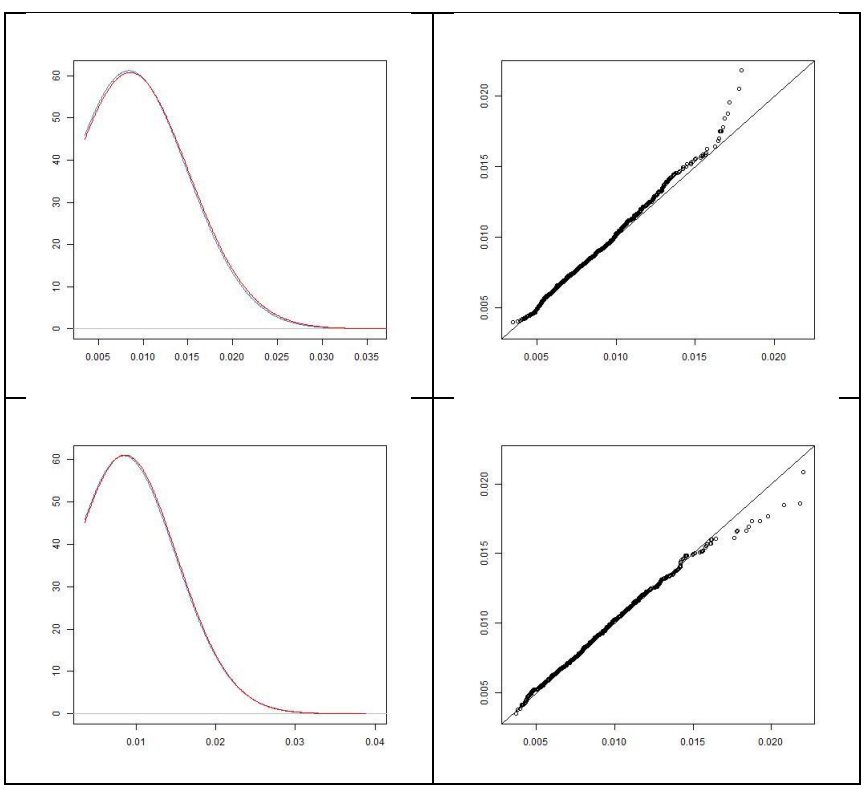


Figure C.13: Density distribution and QQ plots of KS statistics for Triangular Distributions
From top to bottom: Distributions T and T_B. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

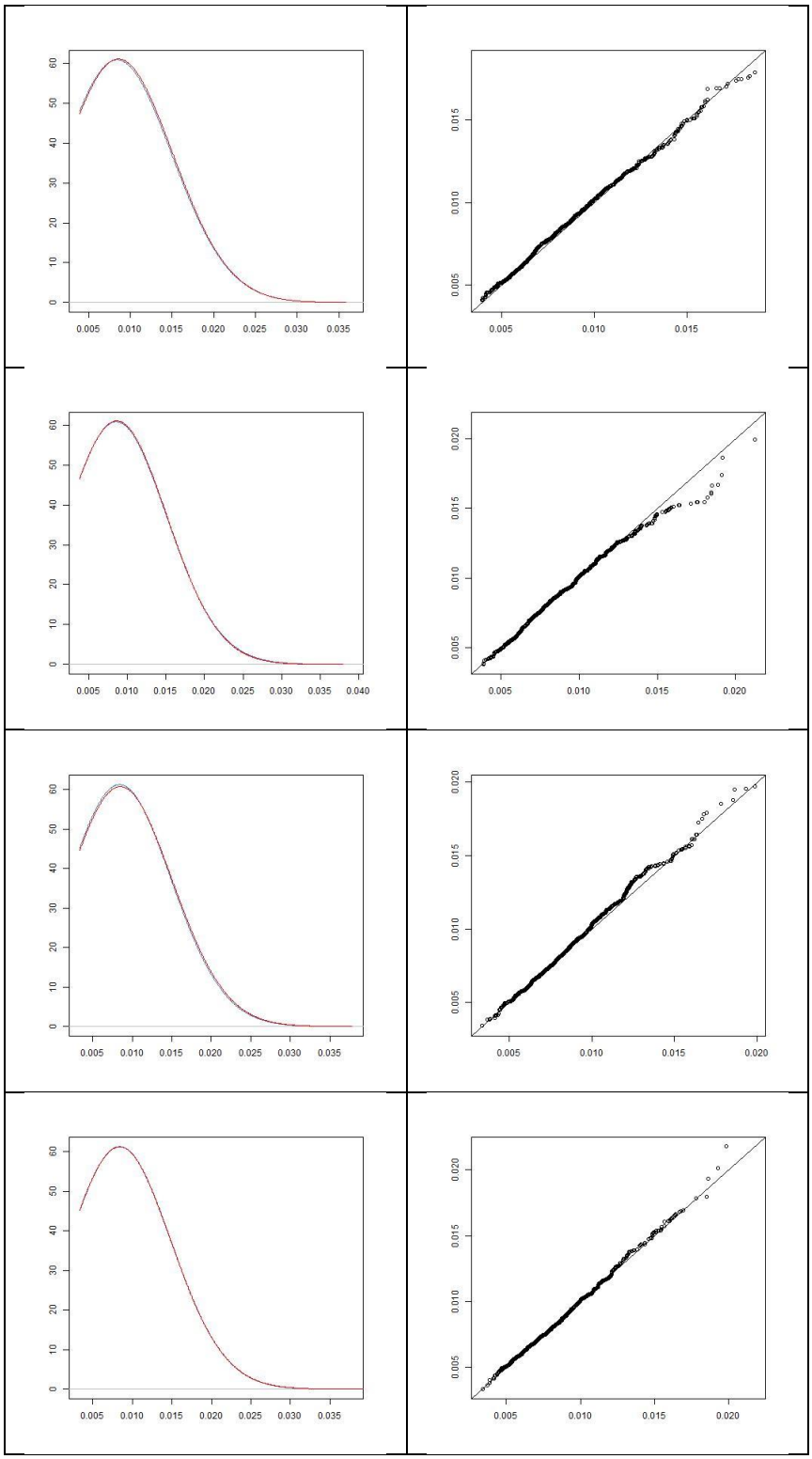


Figure C.14: Density distribution and QQ plots of KS statistics for Triangular Distributions
From top to bottom: Distributions T_R, T_L, T_D and T_U. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

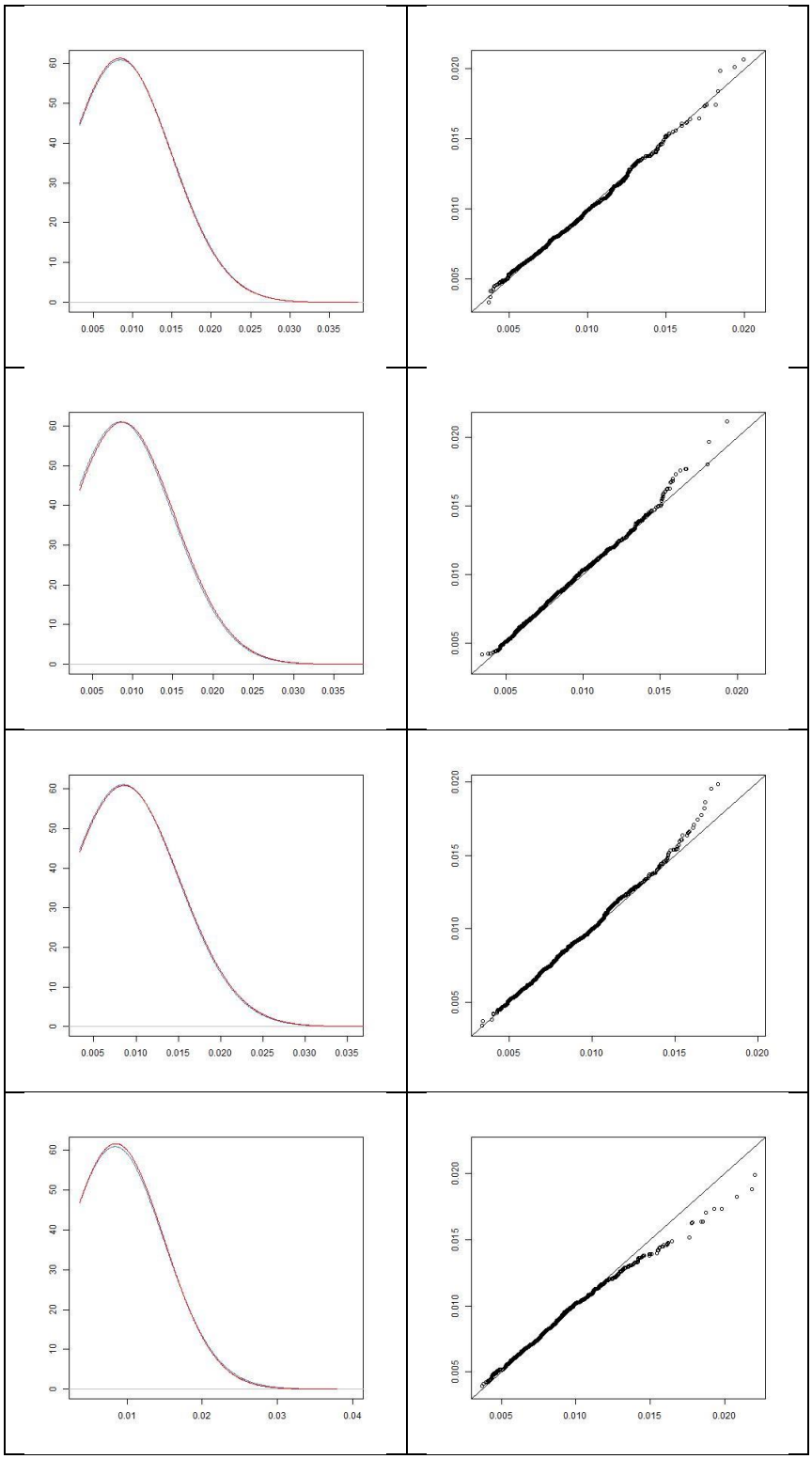


Figure C.15: Density distribution and QQ plots of KS statistics for Weibull Distributions
From top to bottom: Distributions W_a, W_b, W_c and W_d. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

In order to compare the effectiveness of the proposed functions and the commercial software using the KS test approach, we perform the two-sample KS test over the distributions of the 1000 values of $D_{d,s}^{E}$ and $D_{d,s}^{C}$ to verify if they can be assumed to be random samples from the same population. We first calculate the KS statistic D_d^{KS} for each distribution d as:

$$D_d^{KS} = \sup_{D_{d,s}} \left| F_E\left(D_{d,s}\right) - F_C\left(D_{d,s}\right) \right| \qquad [7]$$

where $D_{d,s} = D_{d,s}^{E} \cup D_{d,s}^{C}$ and $F_E\left(D_{d,s}\right)$ and $F_C\left(D_{d,s}\right)$ are respectively the cumulative distributions of $D_{d,s}^{E}$ and $D_{d,s}^{C}$ calculated over $D_{d,s}$. Table C.6 shows the values of D_d^{KS} calculated for each continuous distribution d covered in this report. We then compare each D_d^{KS} with the KS critical value D'_α calculated as:

$$D'_\alpha = F_{KS}^{-1}(\alpha) \cdot \sqrt{\frac{n_E + n_C}{n_E \cdot n_C}} = F_{KS}^{-1}(\alpha) \cdot \sqrt{\frac{2}{1000}} \qquad [8]$$

where α is the significance level, F_{KS}^{-1} is the inverse of the KS CDF, and $n_E = n_C = 1000$ are the sizes of the two samples being compared. For a given distribution d, the results from the proposed functions and the commercial software can be considered equivalent – at the significance level of α – if $D_d^{KS} \leq D'_\alpha$. For $\alpha = 0.05$, $F_{KS}^{-1} = 1.36$ and the KS critical value D'_α is equal to 0.06082. Results in Table C.6 show that all D_d^{KS} are lower than the critical value D'_α. Therefore the proposed functions and the commercial software present statistically equivalent effectiveness when the samples generated are compared with their corresponding theoretical distributions using the KS test.

Table C.6: Equivalence evaluated from the KS test

Distribution		**Two-sample KS Statistic (D_d^{KS})**
Uniform	U	0.041
Normal	N	0.044
	N_B	0.036
Triangular	T	0.040
	T_B	0.033
	T_R	0.051
	T_L	0.029
	T_D	0.035
	T_U	0.026
Weibull	W_a	0.035
	W_b	0.055
	W_c	0.041
	W_d	0.031

Appendix D: Kullback-Leibler Divergence

The KL divergence (Kullback and Leibler, 1951) is a non-symmetric measure of the difference between two probability distributions that expresses the information lost when one distribution is used to approximate the other. In this study, the KL divergence measure is used to evaluate the extent to what the empirical distributions $P_d^E(W_{d,s}^E)$ and $P_d^C(W_{d,s}^C)$, calculated respectively from the samples $W_{d,s}^E = [w_{d,s,i}^E]$ and $W_{d,s}^C = [w_{d,s,i}^C]$, $s = 1, \dots, 1000$ generated by the proposed functions and the commercial software, can be used to approximate their corresponding theoretical distributions. We calculate the KL statistics $K_{d,s}^E$ and $K_{d,s}^C$ for each of the samples generated by the proposed functions and the commercial software, for each continuous distribution d listed in Table 2 as:[10]

$$K_{d,s}^E\left(P_d(W_{d,s}^E)||P_d^E(W_{d,s}^E)\right) = \sum_i P_d(W_{d,s}^E)\ln\frac{P_d(W_{d,s}^E)}{P_d^E(W_{d,s}^E)} \quad [9a]$$

$$K_{d,s}^C\left(P_d(W_{d,s}^C)||P_d^C(W_{d,s}^C)\right) = \sum_i P_d(W_{d,s}^C)\ln\frac{P_d(W_{d,s}^C)}{P_d^E(W_{d,s}^C)} \quad [9b]$$

where $P_d^E(W_{d,s}^E)$ and $P_d^C(W_{d,s}^C)$ are respectively the PDFs of the samples $W_{d,s}^E$ and $W_{d,s}^C$, and $P_d(W_{d,s}^E)$ and $P_d(W_{d,s}^C)$ the PDFs of the reference distribution d calculated for the same values that comprise the samples $W_{d,s}^E$ and $W_{d,s}^C$.

We proceed with the benchmark based on the KL divergence by comparing – for each distribution d in Table 2 – the distribution of the 1000 values of $K_{d,s}^E$ with the distribution of the 1000 values of $K_{d,s}^C$. Table D.7 presents descriptive statistics for distributions $K_{d,s}^E$ and $K_{d,s}^C$. Figure D.16 to Figure D.20 compare the distributions of $K_{d,s}^E$ and $K_{d,s}^C$.

In order to compare the effectiveness of the proposed functions and the commercial software measured with the KL divergence approach, we perform – for each continuous distribution d – the two-sample KS test (as described in Appendix C) over the distributions of the 1000 values of $K_{d,s}^E$ and $K_{d,s}^C$ to verify if they can be assumed to be random samples of the same population. The approach is similar to the one used in Appendix C to compare the results from the one-sample KS tests as represented in Equation [7]. We first calculate the KS statistic D_d^{KL} for each continuous distribution d as:

$$D_d^{KL} = \sup_{K_{d,s}}\left|F_E(K_{d,s}) - F_C(K_{d,s})\right| \quad [10]$$

where $K_{d,s} = K_{d,s}^E \cup K_{d,s}^C$ and $F_E(K_{d,s})$ and $F_C(K_{d,s})$ are respectively the cumulative distributions of $K_{d,s}^E$ and $K_{d,s}^C$ calculated over $K_{d,s}$. Table D.8 shows the values of D_d^{KL} calculated for each continuous distribution d covered in this report.

Table D.7: Descriptive statistics of the distributions of KL statistics

[10] The empirical distributions that we compare with their corresponding theoretical distributions are derived from the sequences $w_{d,s}^E$ and $w_{d,s}^C$ of pseudo-random numbers. We therefore use the KL formulation to compare categorical probability distributions (even though the theoretical distributions are continuous).

Dist (d)	**Proposed Functions** $(K_{d,s}^{E})$				**Commercial Software** $(K_{d,s}^{C})$			
	Min	Median	Max	Mean	Min	Median	Max	Mean
U	0.00012	0.00084	0.00289	0.00090	0.00009	0.00081	0.00283	0.00088
N	-0.00032	0.00131	4.65096	0.38614	-0.00047	0.00115	4.44697	0.37657
N_B	0.00002	0.00052	0.00214	0.00060	0.00004	0.00052	0.00197	0.00059
T	0.00014	0.00085	0.00275	0.00091	0.00008	0.00085	0.00262	0.00092
T_B	0.00013	0.00092	0.00304	0.00099	0.00012	0.00094	0.00295	0.00100
T_R	0.00010	0.00083	0.00291	0.00090	0.00014	0.00084	0.00289	0.00092
T_L	0.00013	0.00085	0.00267	0.00090	0.00013	0.00083	0.00310	0.00090
T_D	0.00012	0.00084	0.00285	0.00090	0.00015	0.00084	0.00279	0.00090
T_U	0.00011	0.00082	0.00277	0.00089	0.00008	0.00082	0.00406	0.00090
W_a	0.48174	0.50617	0.53151	0.50608	0.48660	0.50610	0.53122	0.50621
W_b	0.29218	0.30678	0.32153	0.30675	0.29113	0.30684	0.32280	0.30679
W_c	0.29143	0.30644	0.32151	0.30666	0.28979	0.30689	0.32157	0.30684
W_d	0.14213	0.14846	0.15632	0.14844	0.14076	0.14842	0.15642	0.14847

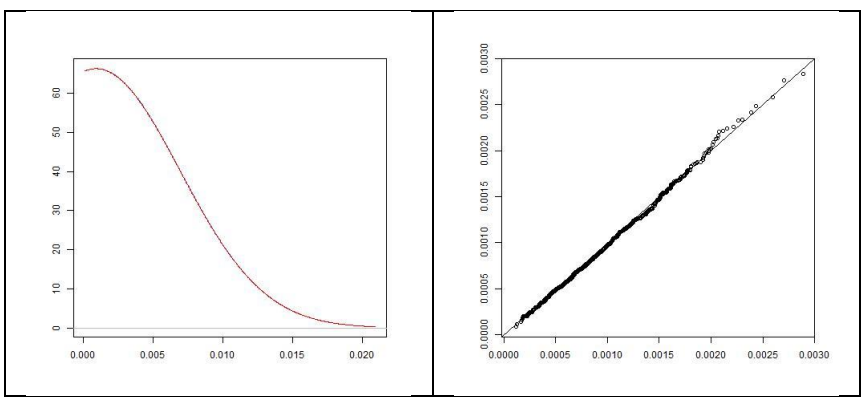


Figure D.16: Density distribution and QQ plots of KL statistics for the Uniform Distribution
Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

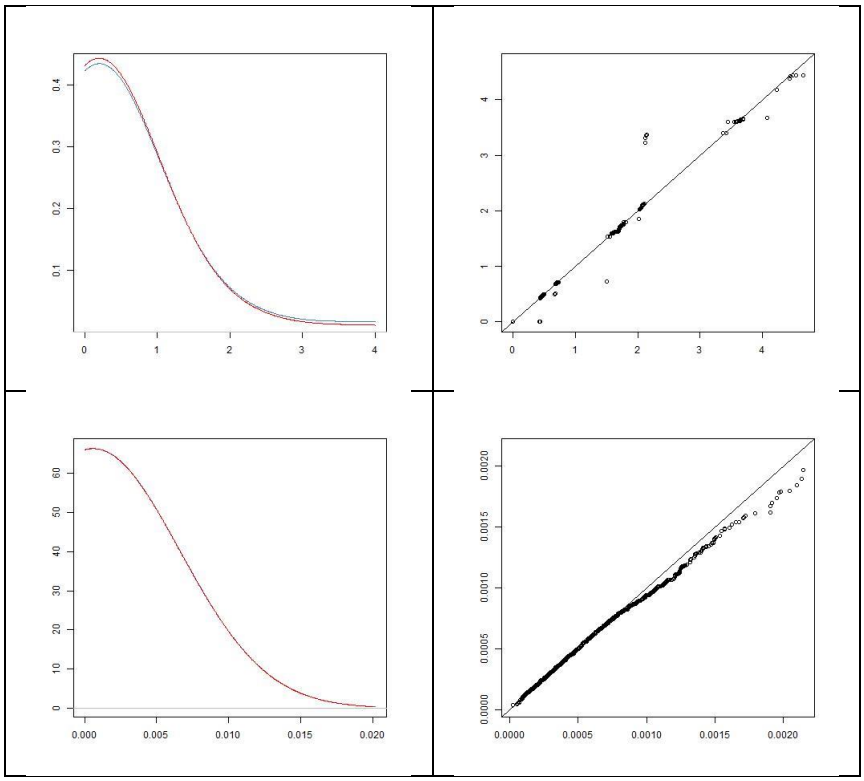


Figure D.17: Density distribution and QQ plots of KL statistics for Normal Distributions
Top to bottom: Distributions N and N_B. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

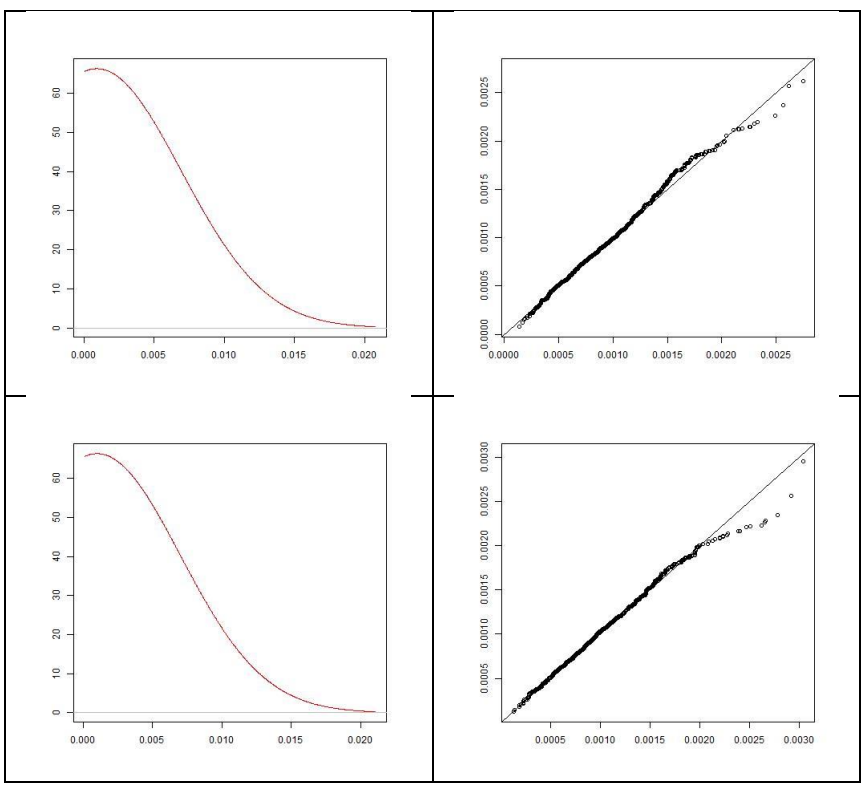


Figure D.18: Density distribution and QQ plots of KL statistics for Triangular Distributions
Top to bottom: Distributions T and T_B. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

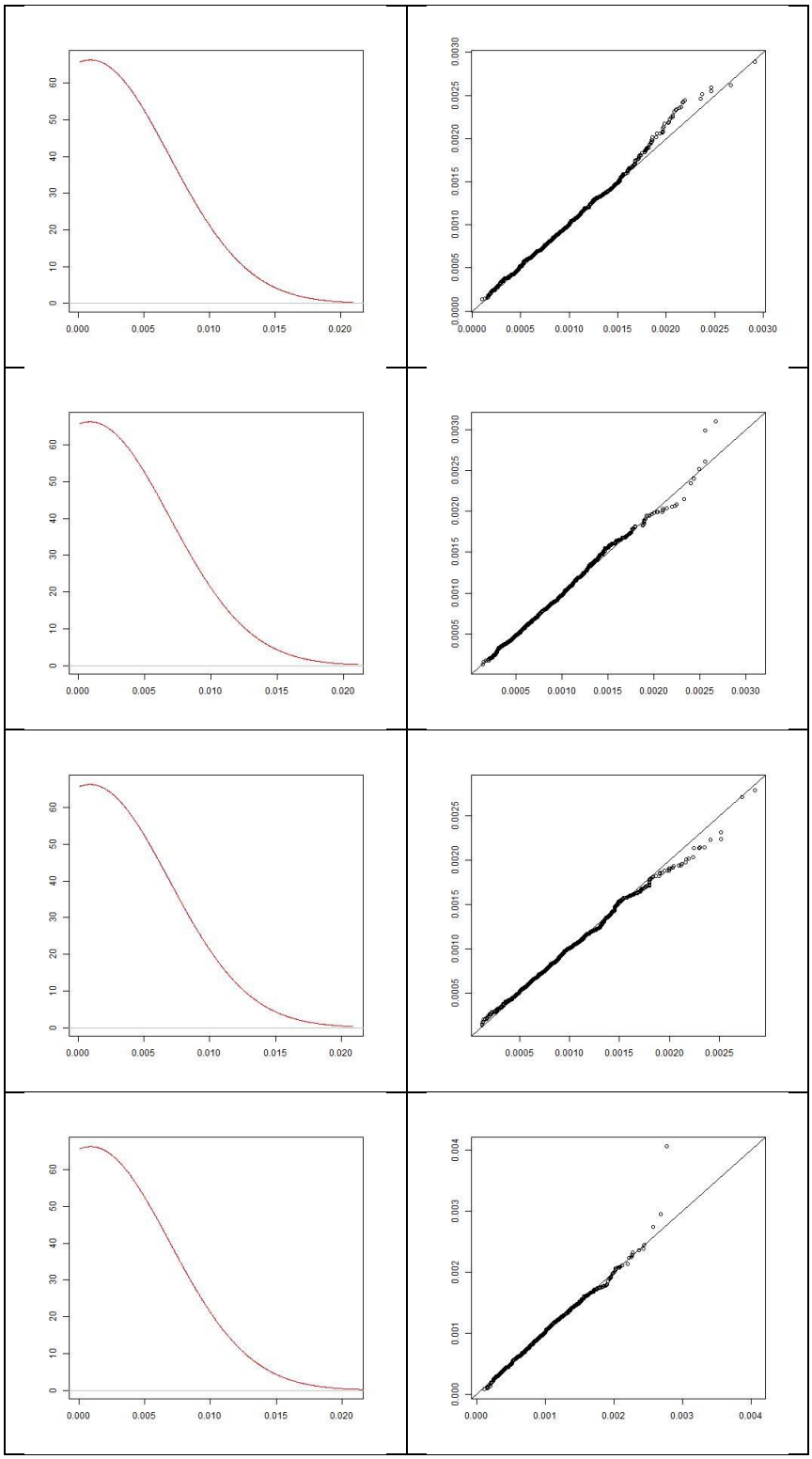


Figure D.19: Density distribution and QQ plots of KL statistics for Triangular Distributions
Top to bottom: Distributions T_R, T_L, T_D and T_U. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

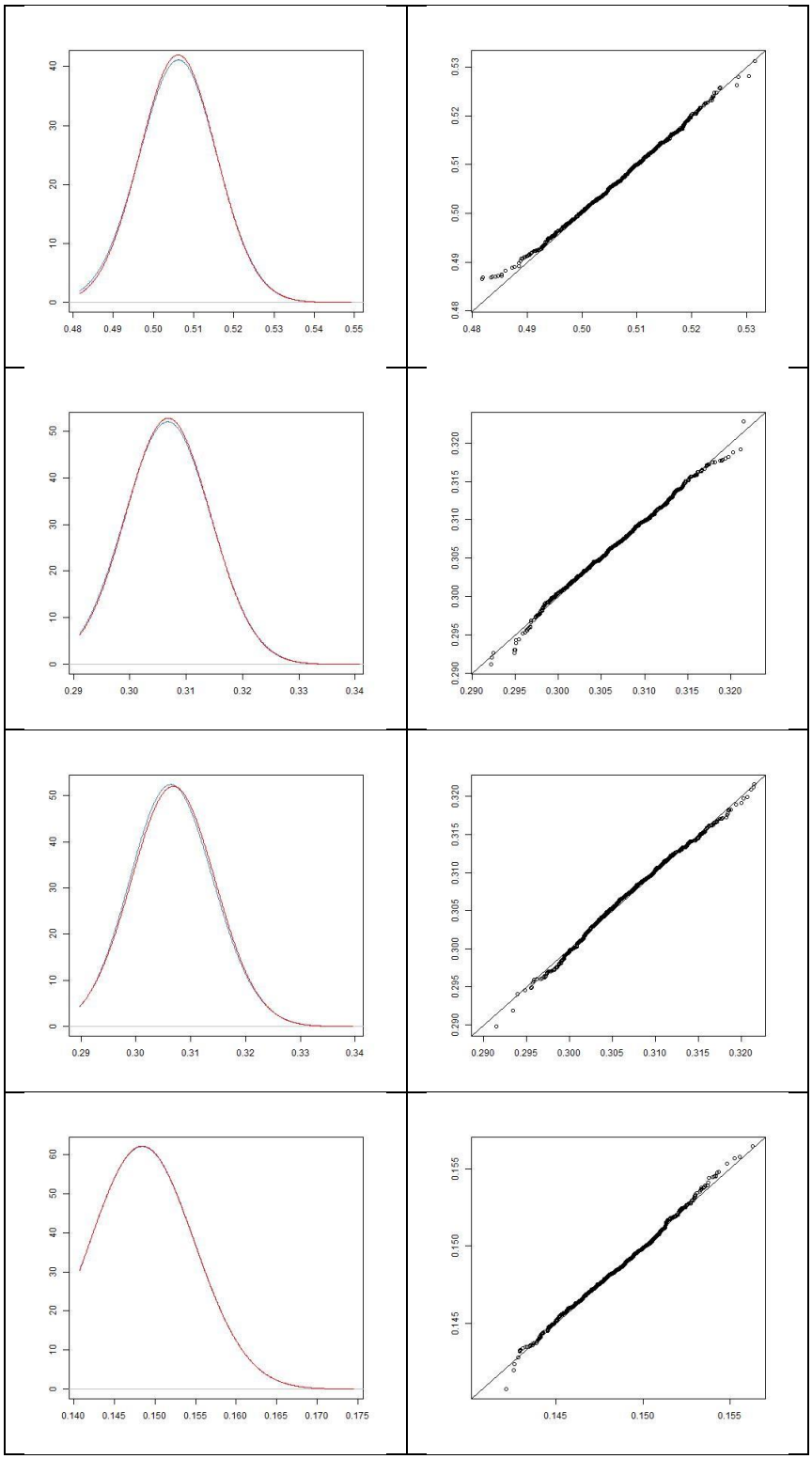


Figure D.20: Density distribution and QQ plots of KL statistics for Weibull Distributions
Top to bottom: Distributions W_a, W_b, W_c and W_d. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

We then compare each D_d^{KL} with the KS critical value D'_α calculated in Appendix C. Results in Table D.8 show that all D_d^{KL} are lower than the critical value 0.06082. Therefore the proposed functions and the commercial software present statistically equivalent effectiveness when the samples they generate are compared with their corresponding theoretical distributions using the KL divergence test.

Table D.8: Equivalence evaluated from the KL divergence test

Distribution		**Two-sample KS Statistic (D_d^{KL})**
Uniform	U	0.048
Normal	N	0.040
	N_B	0.035
Triangular	T	0.030
	T_B	0.029
	T_R	0.035
	T_L	0.024
	T_D	0.030
	T_U	0.030
Weibull	W_a	0.021
	W_b	0.026
	W_c	0.053
	W_d	0.034

Appendix E: Chi-Square Goodness-of-Fit Test

The χ^2 test[11] is used to assess whether a sample of data comes from a population with a specific distribution. The test applies to binned data (i.e., data organized into categories), which makes it a relevant tool to evaluate how well the samples generated for the categorical distributions listed in Table 3 approximate their corresponding theoretical distributions. We calculate the χ^2 statistics $\chi^E_{d,s}$ and $\chi^C_{d,s}$ over the g_d categories that comprise the distribution d, for each of the $s = 1, \dots, 1000$ samples generated by the proposed functions and the commercial software for distribution d, as:

$$\chi^E_{d,s} = \sum_{g_d} \frac{\left(P^E_d(g_d) - P_d(g_d)\right)^2}{P_d(g_d)} \qquad [11a]$$

$$\chi^C_{d,s} = \sum_{g_d} \frac{\left(P^C_d(g_d) - P_d(g_d)\right)^2}{P_d(g_d)} \qquad [11b]$$

where $P^E_d(g_d)$ and $P^C_d(g_d)$ are respectively the probability mass distributions (PMF) of the samples $W^E_{d,s} = [w^E_{d,s,i}]$ and $W^C_{d,s} = [w^C_{d,s,i}]$, $s = 1, \dots, 1000$ generated by the proposed functions and the commercial software, and $P_d(g_d)$ the PMF of the reference distribution d, with all PMFs calculated for the set of categories g_d that comprise the reference distribution d and all samples $W^E_{d,s}$ and $W^C_{d,s}$ corresponding to distribution d.

We proceed with the benchmark based on the χ^2 test by comparing – for each distribution d in Table 3 – the distribution of the 1000 values of $\chi^E_{d,s}$ with the distribution of the 1000 values of $\chi^C_{d,s}$. Table E.9 presents descriptive statistics for distributions $\chi^E_{d,s}$ and $\chi^C_{d,s}$. Figure E.21 compares the distributions of $\chi^E_{d,s}$ and $\chi^C_{d,s}$. Results in Table E.9 show that for all categorical distributions C_x , the distributions of their corresponding $\chi^E_{x,s}$ and $\chi^C_{x,s}$ are almost the same. They are consequently not distinguishable in the charts in Figure E.21.

Table E.9: Descriptive statistics of the distributions of χ^2 statistics

Dist (d)	**Proposed Functions** ($\chi^E_{d,s}$)				**Commercial Software** ($\chi^C_{d,s}$)			
	Min	Median	Max	Mean	Min	Median	Max	Mean
C_2	0.000	0.400	10.800	0.956	0.000	0.454	11.900	1.007
C_3	0.000	1.424	16.503	2.117	0.001	1.331	16.122	1.976
C_5	0.098	3.360	18.238	3.959	0.080	3.366	20.106	4.037
C_{10}	1.184	8.278	27.824	9.032	0.544	8.204	28.250	8.926

[11] For more on the χ^2 test please refer, for example, to NIST/SEMATECH (2013).

In order to compare the effectiveness of the proposed functions and the commercial software measured with the χ^2 test, we perform – for each categorical distribution d – the two-sample KS test (as described in Appendix C) over the distributions of the 1000 values of $\chi_{d,s}^{E}$ and $\chi_{d,s}^{C}$ to verify if they can be assumed to be random samples of the same population. The approach is similar to the one used in Appendix C to compare the results from the one-sample KS tests as represented in [7]. We first calculate the KS statistic D_d^{χ} for each categorical distribution d as:

$$D_d^{\chi} = \sup_{K_{d,s}} \left| F_E(\chi_{d,s}) - F_C(\chi_{d,s}) \right| \qquad [12]$$

where $\chi_{d,s} = \chi_{d,s}^{E} \cup \chi_{d,s}^{C}$ and $F_E(\chi_{d,s})$ and $F_C(\chi_{d,s})$ are respectively the cumulative distributions of $\chi_{d,s}^{E}$ and $\chi_{d,s}^{C}$ calculated over $\chi_{d,s}$. Table E.10 shows the values of D_d^{χ} calculated for each categorical distribution d covered in this report.

We then compare each D_d^{χ} with the KS critical value D'_{α} calculated in Appendix C. Results presented in Table E.10 show that all D_d^{χ} are lower than the critical value 0.06082. Therefore the effectiveness of the proposed functions and the commercial software are statistically equivalent when the samples they generate are compared with their corresponding theoretical distributions using the χ^2 test.

Table E.10: Equivalence evaluated from the χ^2 test

Distribution		**Two-sample KS Statistic (D_d^{χ})**
Categorical	C_2	0.037
	C_3	0.052
	C_5	0.028
	C_{10}	0.021

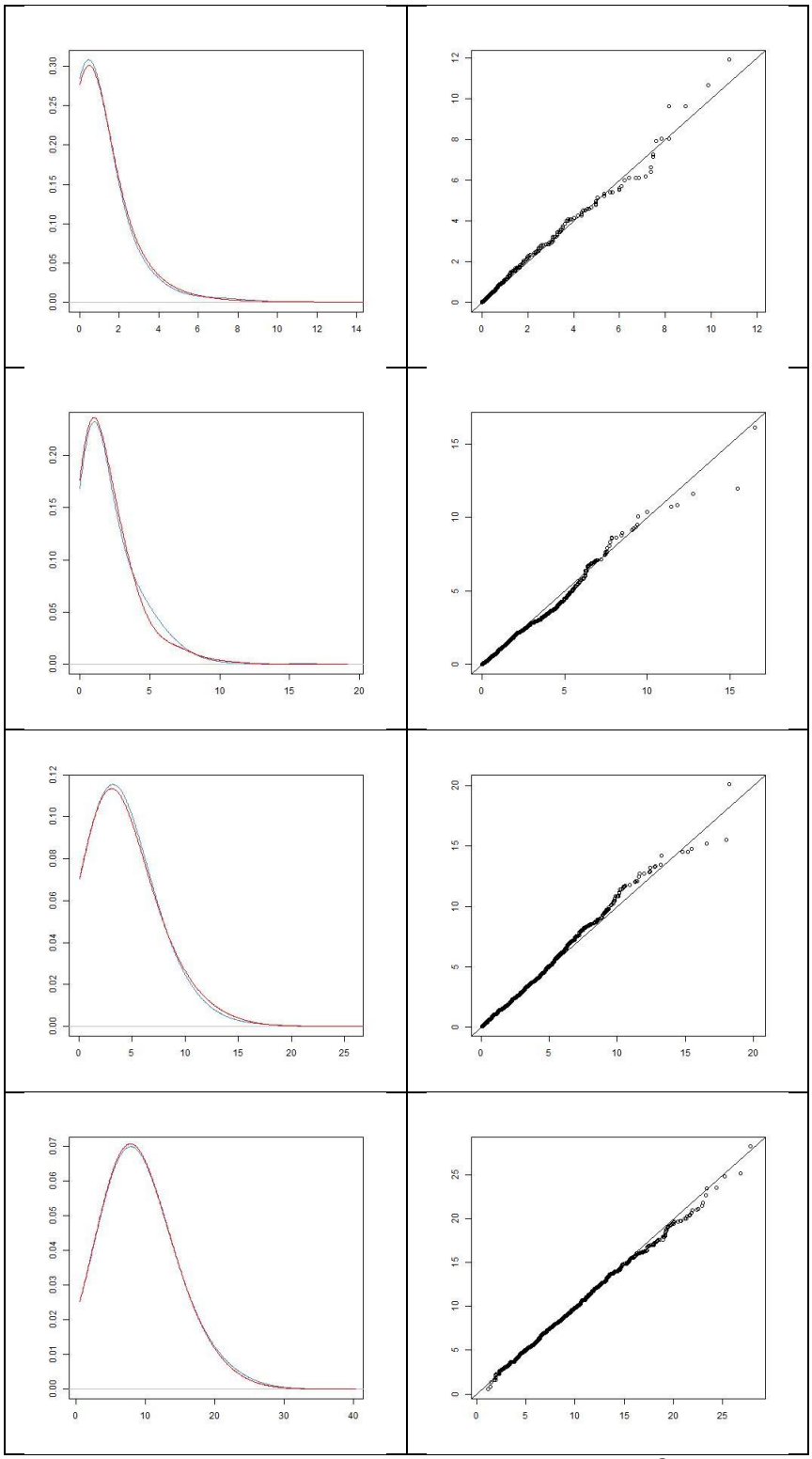


Figure E.21: Density distribution and QQ plots of the χ^2 statistics for Categorical Distributions
Top to bottom: Distributions C_2, C_3, C_5 and C_{10}. Blue: Proposed functions, Red: Commercial software.
QQ-Plot: Horizontal quantiles are from the proposed functions, vertical quantiles are from the commercial software.

Appendix F: Equivalence and Non-Inferiority Tests

The *equivalence test* is used to assess whether two measurements are close enough to be considered equivalents. Equivalence can be claimed if the confidence interval of the difference between the means of the two measurements lies completely within acceptable, pre-defined equivalence limits. The test therefore requires two parameters: a confidence interval for the difference of the means, and a range of values within which the difference of the means is accepted as negligible.

We apply the equivalence test to the means of the distributions of the KS ($D_{d,s}^{E}$ and $D_{d,s}^{C}$), KL ($K_{d,s}^{E}$ and $K_{d,s}^{C}$) and χ^2 ($\chi_{d,s}^{E}$ and $\chi_{d,s}^{C}$) statistics calculated in Appendix C, D and E – for each distribution d – from the samples produced by the proposed functions and the commercial software. Table F.11 to Table F.13 present the means and their differences (including the 95% confidence interval of the differences), calculated for each statistical test performed above. The differences between the means are calculated as:

$$\mu_d^T = \mu_{d,CB}^T - \mu_{d,EES}^T \qquad [13]$$

where $\mu_{d,CB}^T$ and $\mu_{d,EES}^T$ are respectively the means of the statistic T calculated for distribution d from the samples generated by the commercial software and proposed functions, and μ_d^T is the difference of those means.

We further examine the differences of the means for each distribution and – rather than defining an acceptable equivalence interval – we report the smaller relative equivalence interval within which the samples generated by the proposed functions would be accepted as equivalent to the ones generated by the commercial software. The equivalence intervals range from less than 0.1% to 2.4% for the KS statistics (Table F.11), from less than 0.1% to 2.7% for the KL statistics (Table F.12), and from 1.2% to 7.1% for the χ^2 statistics (Table F.13). As for the differences between the means of the statistics, results in Table F.11 to Table F.13 show that for most of the distributions they are positive. This indicates that – for the samples generated for this benchmark, and according to the corresponding statistics – the proposed functions are on average more effective than the commercial software in generating samples for those distributions.[12] For the cases where the differences between the means are negative, we proceed with the evaluation performing a non-inferiority test.

The non-inferiority test has been broadly used in randomized clinical trials to show that an experimental treatment or a new therapy is not (much) worse than a standard treatment or an established therapy (or placebo) (See, for example, Christensen, 2007 or Lesaffre, 2008).[13] Since the proposed functions presented in this report can be viewed as alternatives to those from the commercial software, we parallel the comparative assessment of the effectiveness of the former against the latter with that of medical experimental treatments or new therapies. Similar to the equivalence test, the non-inferiority one relies on the confidence interval of the difference of the means of two samples, which in this case are the six sets of the 1000 values calculated for the KS, KL and χ^2 statistics from the samples generated by the proposed functions and the commercial software. Unlike the equivalence test, however, the non-inferiority test is designed as a one-sided test that attempts to demonstrate, in this case, that the upper

[12] Ideally, results from the three statistics (KS, KL and χ^2) would be zero.
[13] The null-hypothesis we want to be rejected, in this case, is that the differences of the means of the test statistics are greater or equal to zero plus the margin.

bound of the confidence interval of the difference between the means lies below the upper limit of the range within which the difference of the means is accepted as negligible.

We apply the non-inferiority test for the cases in Table F.11 to Table F.13 where the difference between the means is negative. Table F.14 presents the means, differences of the means and confidence intervals for those cases. These are the same results presented in Table F.11 to Table F.13. The table also includes the *t*-statistics and p-values from the non-inferiority test, for a margin of non-inferiority equals to 0.1%. The latter refers to the relative difference between the two means for which the proposed functions can be accepted as non-inferior to the commercial software. Results show that for all cases, the statistics calculated for the proposed functions are (statistically significantly) not greater than 100.1% of their corresponding values calculated for the commercial software. This indicates that the proposed functions are – within a margin of 0.1% – non-inferior than the commercial software in generating pseudo-random numbers, even when the former seems less effective than the latter.

Table F.11: Results from equivalence test for KS statistics

Distribution		**Commercial Software**	**Proposed Functions**	**Difference [95% CI]**		**Equivalence Interval**
Uniform	U	0.00848	0.00865	-0.00017	[-0.00040, 0.00005]	±2.1%
Normal	N	0.00849	0.00867	-0.00018	[-0.00041, 0.00005]	±2.1%
	N_B	0.01222	0.01222	-0.00001	[-0.00033, 0.00032]	±0.0%
Triangular	T	0.00876	0.00857	0.00019	[-0.00004, 0.00042]	±2.2%
	T_B	0.00873	0.00862	0.00011	[-0.00012, 0.00035]	±1.3%
	T_R	0.00874	0.00861	0.00013	[-0.00010, 0.00036]	±1.5%
	T_L	0.00872	0.00874	-0.00002	[-0.00025, 0.00022]	±0.2%
	T_D	0.00869	0.00857	0.00012	[-0.00011, 0.00035]	±1.4%
	T_U	0.00859	0.00857	0.00001	[-0.00021, 0.00024]	±0.2%
Weibull	W_a	0.00859	0.00864	-0.00005	[-0.00028, 0.00018]	±0.5%
	W_b	0.00888	0.00866	0.00021	[-0.00002, 0.00044]	±2.4%
	W_c	0.00877	0.00865	0.00012	[-0.00011, 0.00035]	±1.4%
	W_d	0.00862	0.00862	0.00000	[-0.00023, 0.00022]	±0.0%

Table F.12: Results from equivalence test for KL statistics

Distribution		Commercial Software	Proposed Functions	Difference [95% CI]		Equivalence Interval
Uniform	U	0.00088	0.00090	-0.00002	[-0.00006, 0.00001]	±2.6%
Normal	N	0.37657	0.38614	-0.00957	[-0.07200, 0.05286]	±2.5%
	N_B	0.00059	0.00060	-0.00002	[-0.00005, 0.00001]	±2.7%
Triangular	T	0.00092	0.00091	0.00001	[-0.00003, 0.00005]	±1.0%
	T_B	0.00100	0.00099	0.00001	[-0.00003, 0.00005]	±1.0%
	T_R	0.00092	0.00090	0.00002	[-0.00002, 0.00006]	±2.4%
	T_L	0.00090	0.00090	0.00000	[-0.00004, 0.00003]	±0.3%
	T_D	0.00090	0.00090	0.00000	[-0.00004, 0.00004]	±0.0%
	T_U	0.00090	0.00089	0.00001	[-0.00003, 0.00005]	±0.8%
Weibull	W_a	0.50621	0.50608	0.00013	[-0.00053, 0.00079]	±0.0%
	W_b	0.30679	0.30675	0.00004	[-0.00038, 0.00045]	±0.0%
	W_c	0.30684	0.30666	0.00018	[-0.00024, 0.00060]	±0.1%
	W_d	0.14847	0.14844	0.00002	[-0.00018, 0.00023]	±0.0%

Table F.13: Results from equivalence test for χ^2 statistics

Distribution		Commercial Software	Proposed Functions	Difference [95% CI]		Equivalence Interval
Categorical	C_2	1.00654	0.95554	0.05101	[-0.07218, 0.17419]	±5.1%
	C_3	1.97613	2.11694	-0.14081	[-0.31832, 0.03670]	±7.1%
	C_5	4.03724	3.95929	0.07795	[-0.16685, 0.32274]	±1.9%
	C_{10}	8.92555	9.03223	-0.10668	[-0.48625, 0.27289]	±1.2%

Table F.14: Results from non-inferiority tests

Distribution		Commercial Software	Proposed Functions	Difference [95% CI]		t-Stat	p-Value
KS Statistics							
Uniform	U	0.00848	0.00865	-0.00017	[-0.00040, 0.00005]	1.453	0.000570
Normal	N	0.00849	0.00867	-0.00018	[-0.00041, 0.00005]	1.422	0.000568
	N_B	0.01222	0.01222	-0.00001	[-0.00033, 0.00032]	-0.041	0.000569
Triangular	T_L	0.00872	0.00874	-0.00002	[-0.00025, 0.00022]	0.061	0.000569
Weibull	W_a	0.00859	0.00864	-0.00005	[-0.00028, 0.00018]	0.326	0.000568
KL Statistics							
Uniform	U	0.00088	0.00090	-0.00002	[-0.00006, 0.00001]	1.144	0.000588
Normal	N	0.37657	0.38614	-0.00957	[-0.07200, 0.05286]	0.289	0.000521
	N_B	0.00059	0.00060	-0.00002	[-0.00005, 0.00001]	0.991	0.000571
χ^2 *Statistics*							
Categorical	C_3	1.97613	2.11694	-0.14081	[-0.31832, 0.03670]	1.533	0.000540
	C_{10}	8.92555	9.03223	-0.10668	[-0.48625, 0.27289]	0.505	0.000588

Appendix G: R Source Code

Table G.15 describes the R programs developed to support the analysis described in this report. Box G.6 through Box G.9 present the R source code of the programs.

Table G.15: R Programs

Analysis	*Scope in Report*	*R Program*
KS Analysis	Figures C.11-15 Tables C.5-6, F.11	KS.R
KL Analysis	Figures D.16-20 Tables D.7-8, F.12	KL.R
Chi-Square Analysis	Figures E.21 Tables E.9-10, F.13	Chi2.R
Non-Inferiority Test	Table F.14	Non-Inferior.R

Box G.6: R Source Code for the KS Analysis

```
options(java.parameters="-Xmx4g")
library(XLConnect)
library(VGAM)
library(truncdist)
library(truncnorm)
library(equivalence)
library(openxlsx)

myDist = readWorksheetFromFile("SampleNames.xlsm","Continuous")
myDist
fDist = array(myDist[4:9],dimnames=list(rows=myDist[,1],cols=c("d","p1","p2","p3","L","U")))
nameCols = c("Stat","pV")
ksSumm = matrix(nrow=nrow(myDist),ncol=2)
colnames(ksSumm) = nameCols
rownames(ksSumm) = myDist$D

for (iDist in 1:nrow(myDist)) {
 print(paste((myDist$D[iDist]),"EES"))
 mySamples = read.xlsx(paste("EES ",myDist$Name[iDist],".xlsx",sep=""), sheet=1)

 ksEES = matrix(nrow=1000,ncol=2)
 colnames(ksEES) = nameCols
 for (iSample in 1:1000) {
  ks = if(rownames(fDist)[iDist]=="U")  ks.test(mySamples[iSample], punif, min=fDist$p1[iDist], max=fDist$p2[iDist], alternative="t") else
       if(rownames(fDist)[iDist]=="N")  ks.test(mySamples[iSample], pnorm, mean=fDist$p1[iDist], sd=fDist$p2[iDist], alternative="t") else
       if(rownames(fDist)[iDist]=="Nb") ks.test(mySamples[iSample],
          rtruncnorm(10000, mean=fDist$p1[iDist], sd=fDist$p2[iDist], a=fDist$L[iDist], b=fDist$U[iDist]), alternative="t") else
       if(rownames(fDist)[iDist]=="Tb") ks.test(mySamples[iSample], ptrunc, spec="triangle", lower=fDist$p1[iDist], upper=fDist$p2[iDist],
          theta=fDist$p3[iDist], a=fDist$L[iDist], b=fDist$U[iDist], alternative="t") else
       if(substr(rownames(fDist)[iDist],1,1)=="T") ks.test(mySamples[iSample], ptriangle, lower=fDist$p1[iDist], upper=fDist$p2[iDist],
          theta=fDist$p3[iDist], alternative="t") else
       if(substr(rownames(fDist)[iDist],1,1)=="W") ks.test(mySamples[iSample]-1, pweibull, shape=fDist$p1[iDist], scale=fDist$p2[iDist], alternative="t")
  ksEES[iSample,1] = ks$statistic
  ksEES[iSample,2] = ks$p.value
 }
 rm(mySamples)
 gc()
 writeWorksheetToFile("KS.xlsx", data=myDist$D[iDist], sheet="EES", startRow=2, startCol=2+3*(iDist-1),header=FALSE)
 writeWorksheetToFile("KS.xlsx", data=ksEES, sheet="EES", startRow=3, startCol=2+3*(iDist-1))

 print(paste((myDist$D[iDist]),"CB"))
 mySamples = read.xlsx(paste("CB ",myDist$Name[iDist],".xlsx",sep=""), sheet=1)

 ksCB = matrix(nrow=1000,ncol=2)
 colnames(ksCB) = nameCols
 for (iSample in 1:1000) {
  ks = if(rownames(fDist)[iDist]=="U")  ks.test(mySamples[iSample], punif, min=fDist$p1[iDist], max=fDist$p2[iDist], alternative="t") else
       if(rownames(fDist)[iDist]=="N")  ks.test(mySamples[iSample], pnorm, mean=fDist$p1[iDist], sd=fDist$p2[iDist], alternative="t") else
       if(rownames(fDist)[iDist]=="Nb") ks.test(mySamples[iSample],
          rtruncnorm(10000, mean=fDist$p1[iDist], sd=fDist$p2[iDist], a=fDist$L[iDist], b=fDist$U[iDist]), alternative="t") else
       if(rownames(fDist)[iDist]=="Tb") ks.test(mySamples[iSample], ptrunc, spec="triangle", lower=fDist$p1[iDist], upper=fDist$p2[iDist],
          theta=fDist$p3[iDist], a=fDist$L[iDist], b=fDist$U[iDist], alternative="t") else
       if(substr(rownames(fDist)[iDist],1,1)=="T") ks.test(mySamples[iSample], ptriangle, lower=fDist$p1[iDist], upper=fDist$p2[iDist],
          theta=fDist$p3[iDist], alternative="t") else
       if(substr(rownames(fDist)[iDist],1,1)=="W") ks.test(mySamples[iSample]-1, pweibull, shape=fDist$p1[iDist], scale=fDist$p2[iDist], alternative="t")
```

```
  ksCB[iSample,1] = ks$statistic
  ksCB[iSample,2] = ks$p.value
 }
 rm(mySamples)
 gc()
 writeWorksheetToFile("KS.xlsx", data=myDist$D[iDist], sheet="CB", startRow=2, startCol=2+3*(iDist-1), header=FALSE)
 writeWorksheetToFile("KS.xlsx", data=ksCB, sheet="CB", startRow=3, startCol=2+3*(iDist-1))

 ksDist = ks.test(ksEES[,1], ksCB[,1], alternative="t")
 ksSumm[iDist,1] = ksDist$statistic
 ksSumm[iDist,2] = ksDist$p.value

 ksEquiv = rtost(ksCB[,1], ksEES[,1], alpha=0.05, tr=0)
 writeWorksheetToFile("KS.xlsx", data=ksEquiv$mean.diff, sheet="Equivalence", startRow=3+iDist, startCol=5, header=FALSE)
 writeWorksheetToFile("KS.xlsx", data=ksEquiv$se.diff,   sheet="Equivalence", startRow=3+iDist, startCol=6, header=FALSE)

 jpeg(file=paste("ks",myDist$D[iDist],".jpeg"))
 plot(density(ksEES[,1], bw=0.006, from=min(min(ksEES[,1]),min(ksCB[,1]))), xlab="", main="", ylab="", col="steelblue")
 lines(density(ksCB[,1], bw=0.006, from=min(min(ksEES[,1]),min(ksCB[,1]))), col="red")
 dev.off()

 jpeg(file=paste("kSQQ",myDist$D[iDist],".jpeg"))
 qqplot(ksEES[,1],ksCB[,1], plot.it=TRUE, xlab="", ylab="", xlim=c(min(ksEES[,1],ksCB[,1]),max(ksEES[,1],ksCB[,1])),
                                                            ylim=c(min(ksEES[,1],ksCB[,1]),max(ksEES[,1],ksCB[,1])))
 abline(0,1)
 dev.off()
}

writeWorksheetToFile("KS.xlsx", data=ksSumm, sheet="KS's KS", startRow=2, startCol=2)
```

Box G.7: R Source Code for the KL Analysis

```
options(java.parameters="-Xmx4g")
library(XLConnect)
library(VGAM)
library(truncdist)
library(entropy)
library(mc2d)
library(FNN)
library(equivalence)
library(openxlsx)
myDist = readWorksheetFromFile("SampleNames.xlsm","Continuous")
myDist
fDist = array(myDist[4:9],dimnames=list(rows=myDist[,1],cols=c("d","p1","p2","p3","L","U")))
nameCols = c("Stat","pV")
ksSumm = matrix(nrow=nrow(myDist),ncol=2)
colnames(ksSumm) = nameCols
rownames(ksSumm) = myDist$D

for (iDist in 1:nrow(myDist)) {
 print(paste((myDist$D[iDist]),"EES"))
 mySamples = read.xlsx(paste("EES ",myDist$Name[iDist],".xlsx",sep=""), sheet=1)

 klEES = matrix(nrow=1000,ncol=2)
 colnames(klEES) = nameCols
```

```
for (iSample in 1:1000) {
 kl = if(rownames(fDist)[iDist]=="U") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts,
         (hist(runif(10000, min=fDist$p1[iDist], max=fDist$p2[iDist]), plot=F, breaks=8))$counts) else
      if(rownames(fDist)[iDist]=="N") KL.empirical((hist(mySamples[,iSample], plot=F, breaks=6, prob=F, ylim=c(-4,4)))$counts,
         (hist(rnorm(10000, mean=fDist$p1[iDist], sd=fDist$p2[iDist]), plot=F, breaks=6, prob=F, ylim=c(-4,4)))$counts) else
      if(rownames(fDist)[iDist]=="Nb") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts,
         (hist(truncdist::rtrunc(10000, spec="norm", mean=fDist$p1[iDist], sd=fDist$p2[iDist], a=fDist$L[iDist], b=fDist$U[iDist]),
          plot=F,breaks=8))$counts) else
      if(rownames(fDist)[iDist]=="Tb") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts,
         (hist(truncdist::rtrunc(10000, spec="triangle", lower=fDist$p1[iDist], upper=fDist$p2[iDist], theta=fDist$p3[iDist], a=fDist$L[iDist],
          b=fDist$U[iDist]),plot=F, breaks=8))$counts) else
      if(substr(rownames(fDist)[iDist],1,1)=="T") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts, (hist(rtriang(10000,
         min=fDist$p1[iDist], max=fDist$p2[iDist], mode=fDist$p3[iDist]),plot=F, breaks=8))$counts) else
      if(substr(rownames(fDist)[iDist],1,1)=="W") KL.empirical(mySamples[,iSample]-1,rweibull(10000, shape=fDist$p1[iDist], scale=fDist$p2[iDist]))
 klEES[iSample,1] = kl
 klEES[iSample,2] = 0
}
rm(mySamples)
gc()
writeWorksheetToFile("KL.xlsx", data=myDist$D[iDist], sheet="EES", startRow=2, startCol=2+3*(iDist-1),header=FALSE)
writeWorksheetToFile("KL.xlsx", data=klEES, sheet="EES", startRow=3, startCol=2+3*(iDist-1))

print(paste((myDist$D[iDist]),"CB"))
mySamples = read.xlsx(paste("CB ",myDist$Name[iDist],".xlsx",sep=""), sheet=1)

klCB = matrix(nrow=1000,ncol=2)
colnames(klCB) = nameCols
for (iSample in 1:1000) {
 kl = if(rownames(fDist)[iDist]=="U") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts,
         (hist(runif(10000, min=fDist$p1[iDist], max=fDist$p2[iDist]), plot=F, breaks=8))$counts) else
      if(rownames(fDist)[iDist]=="N") KL.empirical((hist(mySamples[,iSample], plot=F, breaks=6, prob=F, ylim=c(-4,4)))$counts,
         (hist(rnorm(10000, mean=fDist$p1[iDist], sd=fDist$p2[iDist]), plot=F, breaks=6, prob=F, ylim=c(-4,4)))$counts) else
      if(rownames(fDist)[iDist]=="Nb") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts,
         (hist(truncdist::rtrunc(10000, spec="norm", mean=fDist$p1[iDist], sd=fDist$p2[iDist], a=fDist$L[iDist], b=fDist$U[iDist]),
          plot=F,breaks=8))$counts) else
      if(rownames(fDist)[iDist]=="Tb") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts,
         (hist(truncdist::rtrunc(10000, spec="triangle", lower=fDist$p1[iDist], upper=fDist$p2[iDist], theta=fDist$p3[iDist], a=fDist$L[iDist],
          b=fDist$U[iDist]),plot=F, breaks=8))$counts) else
      if(substr(rownames(fDist)[iDist],1,1)=="T") KL.plugin((hist(mySamples[,iSample], plot=F, breaks=8))$counts, (hist(rtriang(10000,
         min=fDist$p1[iDist], max=fDist$p2[iDist], mode=fDist$p3[iDist]),plot=F, breaks=8))$counts) else
      if(substr(rownames(fDist)[iDist],1,1)=="W") KL.empirical(mySamples[,iSample]-1,rweibull(10000, shape=fDist$p1[iDist], scale=fDist$p2[iDist]))
 klCB[iSample,1] = kl
 klCB[iSample,2] = 0
}
rm(mySamples)
gc()
writeWorksheetToFile("KL.xlsx", data=myDist$D[iDist], sheet="CB", startRow=2, startCol=2+3*(iDist-1),header=FALSE)
writeWorksheetToFile("KL.xlsx", data=klCB, sheet="CB", startRow=3, startCol=2+3*(iDist-1))

ksDist = ks.test(klEES[,1], klCB[,1], alternative="t")
ksSumm[iDist,1] = ksDist$statistic
ksSumm[iDist,2] = ksDist$p.value

klEquiv = rtost(klCB[,1], klEES[,1], alpha=0.05, tr=0, epsilon=1)
writeWorksheetToFile("KL.xlsx", data=klEquiv$mean.diff, sheet="Equivalence", startRow=3+iDist, startCol=5, header=FALSE)
writeWorksheetToFile("KL.xlsx", data=klEquiv$se.diff,   sheet="Equivalence", startRow=3+iDist, startCol=6, header=FALSE)
```

```
 jpeg(file=paste("kl",myDist$D[iDist],".jpeg"))
 plot(density(klEES[,1], bw=0.006, from=min(min(klEES[,1]),min(klCB[,1]))), xlab="", main="", ylab="", col="steelblue")
 lines(density(klCB[,1], bw=0.006, from=min(min(klEES[,1]),min(klCB[,1]))), col="red")
 dev.off()

 jpeg(file=paste("klQQ",myDist$D[iDist],".jpeg"))
 qqplot(klEES[,1],klCB[,1], plot.it=TRUE, xlab="", ylab="", xlim=c(min(klEES[,1],klCB[,1]),max(klEES[,1],klCB[,1])),
                                                            ylim=c(min(klEES[,1],klCB[,1]),max(klEES[,1],klCB[,1])))
 abline(0,1)
 dev.off()
}

writeWorksheetToFile("KS.xlsx", data=ksSumm, sheet="KL's KS", startRow=2, startCol=2)
```

Box G.8: R Source Code for the Chi-Square Analysis

```
options(java.parameters="-Xmx4g")
library(XLConnect)
library(vcd)
library(equivalence)
library(openxlsx)

myDist = readWorksheetFromFile("SampleNames.xlsm","Discrete")
myDist
fDist = array(myDist[3],dimnames=list(rows=myDist[,1],cols="nCat"))
nameCols = c("Stat","pV")
ksSumm = matrix(nrow=nrow(myDist),ncol=2)
colnames(ksSumm) = nameCols
rownames(ksSumm) = myDist$D

for (iDist in 1:nrow(myDist)) {
 print(paste((myDist$D[iDist]),"EES"))
 mySamples = read.xlsx(paste("EES ",myDist$Name[iDist],".xlsx",sep=""), sheet=1)

 chiEES = matrix(nrow=1000,ncol=2)
 colnames(chiEES) = nameCols
 for (iSample in 1:1000) {
  chi = chisq.test(table(mySamples[,iSample]), p=myDist[iDist,4:{4+(myDist$nCat[iDist]-1)}])
  chiEES[iSample,1] = chi$statistic
  chiEES[iSample,2] = chi$p.value
 }
 rm(mySamples)
 gc()
 writeWorksheetToFile("Chi2.xlsx", data=myDist$D[iDist], sheet="EES", startRow=2, startCol=2+3*(iDist-1),header=FALSE)
 writeWorksheetToFile("Chi2.xlsx", data=chiEES, sheet="EES", startRow=3, startCol=2+3*(iDist-1))

 print(paste((myDist$D[iDist]),"CB"))
 mySamples = read.xlsx(paste("CB ",myDist$Name[iDist],".xlsx",sep=""), sheet=1)

 chiCB = matrix(nrow=1000,ncol=2)
 colnames(chiCB) = nameCols
 for (iSample in 1:1000) {
  chi = chisq.test(table(mySamples[,iSample]), p=myDist[iDist,4:{4+(myDist$nCat[iDist]-1)}])
  chiCB[iSample,1] = chi$statistic
  chiCB[iSample,2] = chi$p.value
```

```
 }
 rm(mySamples)
 gc()
 writeWorksheetToFile("Chi2.xlsx", data=myDist$D[iDist], sheet="CB", startRow=2, startCol=2+3*(iDist-1),header=FALSE)
 writeWorksheetToFile("Chi2.xlsx", data=chiCB, sheet="CB", startRow=3, startCol=2+3*(iDist-1))

 ksDist = ks.test(chiEES[,1], chiCB[,1], alternative="t")
 ksSumm[iDist,1] = ksDist$statistic
 ksSumm[iDist,2] = ksDist$p.value

 chiEquiv = rtost(chiCB[,1], chiEES[,1], alpha=0.05, tr=0)
 writeWorksheetToFile("Chi2.xlsx", data=chiEquiv$mean.diff, sheet="Equivalence", startRow=3+iDist, startCol=5, header=FALSE)
 writeWorksheetToFile("Chi2.xlsx", data=chiEquiv$se.diff,   sheet="Equivalence", startRow=3+iDist, startCol=6, header=FALSE)

 jpeg(file=paste("chi",myDist$D[iDist],".jpeg"))
 plot(density(chiEES[,1], bw=myDist$BW[iDist], from=min(min(chiEES[,1]),min(chiCB[,1]))), xlab="", main="", ylab="", col="steelblue")
 lines(density(chiCB[,1], bw=myDist$BW[iDist], from=min(min(chiEES[,1]),min(chiCB[,1]))), col="red")
 dev.off()

 jpeg(file=paste("chiQQ",myDist$D[iDist],".jpeg"))
 qqplot(chiEES[,1],chiCB[,1], plot.it=TRUE, xlab="", ylab="", xlim=c(min(chiEES[,1],chiCB[,1]),max(chiEES[,1],chiCB[,1])),
                                                              ylim=c(min(chiEES[,1],chiCB[,1]),max(chiEES[,1],chiCB[,1])))
 abline(0,1)
 dev.off()
}

writeWorksheetToFile("Chi2.xlsx", data=ksSumm, sheet="Chi's KS", startRow=2, startCol=2)
```

Box G.9: R Source Code for the Non-Inferiority Test

```
# Non-Inferiority for KS Test Stats
ksData = openxlsx::read.xlsx("KS.xlsx", sheet="Equivalence", colNames=T)
ksData$sd = (ksData$SE*sqrt(2000))/2
kappa = 1
ksData$z = (ksData $EES-ksData$CB-(0.001*ksData$CB))/(ksData$sd*sqrt((1+1/kappa)/1000))
Power = pnorm(ksData$z-qnorm(1-alpha))+pnorm(-ksData$z-qnorm(1-alpha))
ksData$power.inf = power.t.test(n=1000,delta=0.001*ksData$CB,sd=ksData$sd,sig.level=0.001,alternative="two.sided",type="two.sample")$power
writeWorksheetToFile("KS.xlsx", data=ksData, sheet="NIF's KS", startRow=2, startCol=2)

# Non-I nferiority for KL Test Stats
klData = openxlsx::read.xlsx("KL.xlsx", sheet="Equivalence", colNames=T)
klData$sd = klData$SE*sqrt(2000)/2
klData$z = (klData$EES-klData$CB-(0.001*klData$CB))/(klData$sd*sqrt((1+1/kappa)/1000))
klData$power.inf = power.t.test(n=1000,delta=0.001*klData$CB,sd=klData$sd,sig.level=0.001,alternative="two.sided",type="two.sample")$power
writeWorksheetToFile("KL.xlsx", data=klData, sheet="NIF's KS", startRow=2, startCol=2)

# Non-I nferiority for Chi Test Stats
chi2Data = openxlsx::read.xlsx("Chi2.xlsx", sheet="Equivalence", colNames=T)
kappa=1
chi2Data$sd = chi2Data$SE*sqrt(2000)/2
chi2Data$z=(chi2Data$EES-chi2Data$CB-(0.001*chi2Data$CB))/(chi2Data$sd*sqrt((1+1/kappa)/1000))
chi2Data$power.inf = power.t.test(n=1000,delta=0.001*chi2Data$CB,sd=chi2Data$sd,sig.level=0.001,alternative="two.sided",type="two.sample" )$power
writeWorksheetToFile("Chi2.xlsx", data=chi2Data, sheet="NIF's Chi2", startRow=2, startCol=2)
```