Testing the Tests: Using Random Number Generators to Improve Empirical Tests

Paul Leopardi[1]

Abstract The implementer of an empirical test for random number generators is faced with some difficult problems, especially if the test is based on a statistic which is known only approximately: How can the test be tested? How can the approximation be improved? When is it good enough? A number of principles can be applied to these problems. These principles are illustrated using implementations of the overlapping serial "Monkey" tests of Marsaglia and Zaman.

1 Introduction

For many empirical tests of random number generators (RNGs), the distribution of the test statistic is known only approximately or asymptotically. The use of two-level testing with such empirical tests and "known good" random number generators can reveal the goodness of fit between the empirical distribution of the test statistic and the approximate theoretical distribution [10, Section 3.1] [11, Section 3]. Two-level testing with a battery of tests can reveal which of the tests use approximations which give a better fit for the size of the test used in the battery.

This paper describes the improvement of the implementation of the overlapping serial "Monkey" tests of Marsaglia and Zaman [18, 17] in the TestU01 suite [13, 14].

The remainder of this paper is organized as follows. Section 2 describes some of the principles of two-level testing with a battery of tests. Section 3 describes how the PseudoDIEHARD battery of TestU01 was tested. Section 4 describes Marsaglia and Zaman's Monkey tests, shows how various differences with the TestU01 implementation were detected, gives some new results for the theoretical moments for these tests, and describes the improvements made to the tests in TestU01. Section 5 summarizes the results of the tests using the improved version of the Pseudo-

Mathematical Sciences Institute, Australian National University.
`http://www.maths.anu.edu.au/~leopardi`

DIEHARD battery of TestU01. Section 6 gives a brief summary of the computation of the variances which are used in the Monkey tests.

2 Two-level testing with a battery of tests

The general idea of two-level testing with a battery of tests is that a battery may yield b p-values; when the battery is repeated r times on disjoint subsequences generated by an RNG, this yields a set of br p-values. This set is then tested using one or more statistical tests for uniformity.

Two-level testing with a battery of tests is not a good idea in general if the intention is to test RNGs, since the individual tests must be short so that the battery can be repeated enough times to give a meaningful result within a reasonable runtime. A short test is less likely to give significant results on an individual RNG than a longer test of the same type. Even worse, since the statistics on which a test is based may be only approximate, performing a two-level test can lead to false rejection of an RNG [10, Section 3.1] [11, Section 3]. For example, a series of papers by Kao and Tang apparently wrongly rejects generators on the basis of failing two-level tests [7, 25, 26].

The key to understanding repeated testing with a battery of empirical tests is to realize that the outcome involves two independent null hypotheses.

$\mathscr{H}_0$: The RNG under test generates a $U(0,1)$ sequence.
$\mathscr{H}_1$: Each test of the battery, when applied to a $U(0,1)$ sequence, yields a p-value from $U(0,1)$.

These can be combined into the hypothesis:

$\mathscr{H}_2$: The battery, when repeatedly applied to the RNG under test, yields a sequence of independent p-values from $U(0,1)$.

For a battery of tests on a single RNG, if $\mathscr{H}_2$ fails it may be hard to distinguish failures of $\mathscr{H}_0$ from failures of $\mathscr{H}_1$. A possible solution is to look for consistent failures of tests across multiple different "known good" RNGs. This approach may not always work, since no RNG actually satisfies $\mathscr{H}_0$ [10, Section 3.4], but there are enough RNGs which work well enough to make this approach feasible.

Once a failure of $\mathscr{H}_1$ for a test battery is detected, the next step is to repeat testing but use each type of test in isolation, or equivalently, to extract from each sequence of p-values produced by a run of the battery those p-values produced by each type of test.

Some tests of a battery may produce multiple correlated p-values. If this causes $\mathscr{H}_1$ to detectably fail for the battery as a whole, this failure will also be detected for the individual test.

For tests using a statistic with a discrete distribution, $\mathscr{H}_1$ is never strictly true, but this type of failure may only be revealed when the number of repetitions of the test is large relative to the number of different p-values which the test can yield.

Models of an empirical test are given by L'Ecuyer and Hellekalek [12, Section 3.1] and by L'Ecuyer and Simard [14, Section 3]. Essentially, a test is a two-step process.

1. The test generates a value y taken by a test statistic Y, where Y is a real-valued function of a number of values generated by the RNG.
2. The test computes a p-value $p := 1 - f(y)$ by using an approximation f to the theoretical distribution function F of the test statistic Y, where F is defined by

$$F(y) := 1 - P[Y \geq y].$$

Possible causes of the failure of $\mathscr{H}_1$ for a particular test may therefore include:

1. The implementation of the test does not match its description in the literature, and actually generates a test statistic $Y' \neq Y$;
2. The function f is not a good approximation to F for the particular parameters used by the test;
3. The implementation of the test actually computes a function $f' \neq f$, giving a different approximation to F from the one described in the literature.

Deeper investigation of the cause of the failure of $\mathscr{H}_1$ for a particular test may therefore require examination of the source code of the test, as well as its description.

3 Initial testing of PseudoDIEHARD in TestU01

TestU01 is a collection of "Utilities for empirical statistical testing of uniform random number generators" [14]. It contains a library of empirical tests, arranged into batteries. Typical use of TestU01 is to test an RNG using the Small Crush, Crush and Big Crush batteries in succession. TestU01 also includes the PseudoDIEHARD battery, which is based on Marsaglia's DIEHARD battery [16].

To investigate the PseudoDIEHARD battery of TestU01, two high quality generators were used: Mersenne Twister mt19937 [20, 19], and Brent Xorgens xor4096 [1, 2]. For the remainder of this paper, the Mersenne Twister mt19937 generator is referred to as MT and the Brent Xorgens xor4096 generator is referred to as BX. Both generators pass all tests of the Small Crush battery. BX passes all tests of the Crush and Big Crush batteries while MT fails Crush and Big Crush in tests of linear complexity [14].

The first testing method used is to perform 64 repetitions of the PseudoDIEHARD battery using an RNG with each of 4 different seeds, yielding a sequence of $N = 32\,256$ p-values, and submit this sequence to a second-level test. The two-sided one sample Kolmogorov test is used to compare the empirical cumulative distribution function (CDF) of the sequence of p-values to the CDF for $U(0,1)$. Hypothesis $\mathscr{H}_2$ is rejected if the second-level test yields a p-value less than 0.001.

The results using TestU01 version 0.6.1 are as follows.

	D	p
BX:	0.0118	0.0002466
MT:	0.0136	1.408×10^{-5}

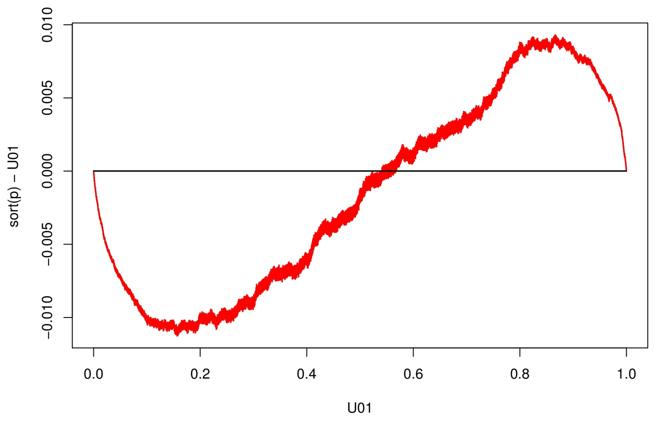
We see that hypothesis $\mathscr{H}_2$ is rejected for both generators, throwing suspicion on hypothesis $\mathscr{H}_1$.

One way to gain more confidence in these results is to increase the number of repetitions. Under hypothesis $\mathscr{H}_0$ this should yield an empirical distribution of p-values which more closely approximates the distribution which is produced by the PseudoDIEHARD test battery. The results using TestU01 0.6.1 PseudoDIEHARD repeated 1024 times, giving 129 024 p-values each, are as follows.

	D	p
BX:	0.0113	1.221×10^{-14}
MT:	0.011	5.951×10^{-14}

Figure 1 plots the difference between the p-values from 1024 repetitions of TestU01 0.6.1 PseudoDIEHARD using BX, sorted in increasing order, and the corresponding values of the $U(0,1)$ distribution, in this case the numbers $(k - 1/2)/N$, for k from 1 to $N = 129\,024$. A systematic pattern resembling a sideways S is easily visible.

Fig. 1 1024 × PseudoDIEHARD 0.6.1 (BX)



When the p-values for each type of test are extracted from a run of 1024 repetitions of TestU01 0.6.1 PseudoDIEHARD, the tests which produce failures of hypothesis $\mathscr{H}_1$ are seen to be the Run test, the OQSO test and the DNA test.

The Run test implemented in TestU01 0.6.1 is essentially the test described in the 1981 version of Knuth [8], with a slight difference. The RNG is called $n+1$ times rather than n times. The improvement to the Run test, which was incorporated into

TestU01 version 1.2.1, essentially consists of implementing the description given in the 1998 version of Knuth [9, pp. 66-69]. Details of this improvement are omitted.

4 Overlapping serial tests

Of the 126 p-values generated by PseudoDIEHARD, 82 come from three overlapping serial (Monkey) tests [18]: 23 OPSO tests, 28 OQSO tests, and 31 DNA tests.

These Monkey tests use an alphabet of size α, form a string of length $n = 2^{21}$ by taking $n \times \log_2 \alpha$ bits from an RNG, and examine the $n-t+1$ overlapping words of length t. According to [18], the number of missing words should be approximately normal with expected value μ and variance σ^2 as given by Table 1.

Table 1 Marsaglia and Zaman's 1993 Monkey test means and standard deviations

	α	t	μ	σ
OPSO:	2^{10}	2	141 909.4653	290.27
OQSO:	2^5	4	141 909.4737	290
DNA:	4	10	141 910.5378	290

In TestU01 the Monkey tests are treated as instances of the overlapping Collision sparse serial tests [13, p. 106, pp. 121–122] [15].

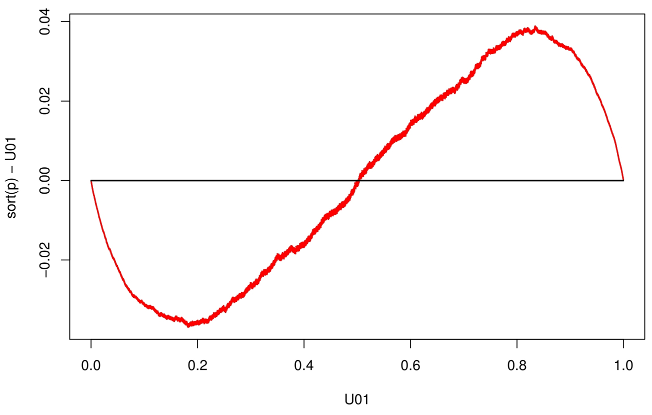
The two-level tests for the Monkey tests use the same method as the Run test. The PseudoDIEHARD battery is repeated 1024 times. From each subsequence of 126 p-values generated by each repetition, the corresponding p-values are extracted: 28 for the OQSO tests and 31 for the DNA tests.

The Kolmogorov test results using the corresponding sequences of p-values extracted from 1024 repetitions of the TestU01 0.6.1 PseudoDIEHARD battery are as follows.

		D	p
OQSO	BX:	0.0085	0.033 11
	MT:	0.0061	0.24
DNA	BX:	0.0389	$< 2.2 \times 10^{-16}$
	MT:	0.0374	$< 2.2 \times 10^{-16}$

Hypothesis $\mathscr{H}_2$ is rejected for the DNA test for both generators. Figure 2 shows the difference between the sorted p-values and the $U(0,1)$ distribution for BX. The distinct sideways S shaped curve of the graph casts suspicion on the variance used to calculate the p-values for the DNA test.

Fig. 2 DNA tests from 1024 $\times$ PseudoDIEHARD 0.6.1 (BX)



For all three Monkey tests, TestU01 0.6.1 calculates the values $k := \alpha^t$ and $\lambda := n/\alpha^t = 2$ and then obtains

$$\mu = k\mathrm{e}^{-\lambda} = 2^{20}\mathrm{e}^{-2} \simeq 141\,909.329955,$$
$$\sigma = k\mathrm{e}^{-\lambda}(1-3\mathrm{e}^{-\lambda}) = 2^{10}\sqrt{\mathrm{e}^{-2}-3\mathrm{e}^{-4}} \simeq 290.3331.$$

These values of μ and σ agree with those of Table 1 to the nearest integer, except for the expected value for the DNA test.

As part of the project which produced the DIEHARD battery of tests [16], Marsaglia in 1995 produced a revised version of his joint paper with Zaman [18]. The newer paper [17] revises the values of σ for the OQSO and DNA tests to 295 and 339 respectively. These revised values were obtained by simulation.

The author submitted a patch to TestU01 to use the revised values of σ for the OQSO and DNA tests. The patch also sets the number of words used to calculate λ to $n-t+1$ so that $\lambda = (n-t+1)/k$, matching the description of the overlapping collision test in the User's Guide [13, Version 0.6.1 p. 121, Version 1.2.1 p. 131]. This patch is used in TestU01 1.2.1.

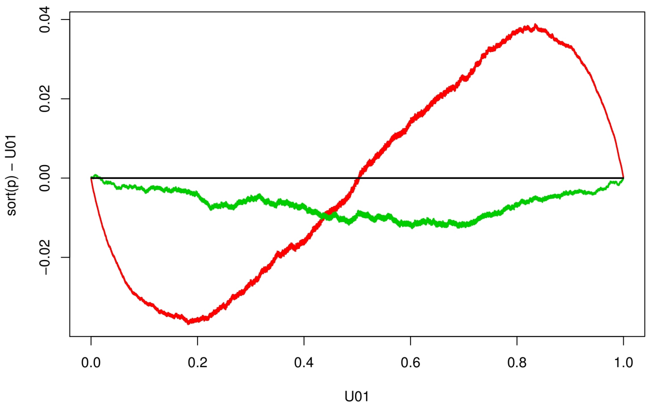
The Kolmogorov test results, using the corresponding sequences of p-values extracted from 1024 repetitions of the TestU01 1.2.1 PseudoDIEHARD battery, are as follows.

		D	p
OQSO	BX:	0.0109	0.002 0921
	MT:	0.0093	0.0141
DNA	BX:	0.0127	6.802×10^{-5}
	MT:	0.0109	0.001 052

Hypothesis $\mathscr{H}_2$ is still rejected for the DNA test for BX, and the p-values for DNA for MT and OQSO for BX are suspiciously low.

Figure 3 shows a U shaped curve which represents the difference between the sorted p-values for the patched DNA test using BX, and the $U(0,1)$ distribution. The sideways S shaped curve from Figure 2 is also shown for comparison. It appears that at least for the DNA test, TestU01 1.2.1 no longer uses an inaccurate variance but instead uses an inaccurate mean.

Fig. 3 DNA tests from 1024 $\times$ PseudoDIEHARD 1.2.1 (BX)



A deeper investigation into the source code of TestU01 1.2.1 reveals two key differences between this implementation and Marsaglia and Zaman's description of the Monkey tests [18, 17].

1. Different test.
   The OPSO, OQSO and DNA tests as described by Marsaglia and Zaman [18, 17] each use a string of length n. Their implementations in TestU01 use words on a *cycle* of length n rather than a string, yielding n not $n-t+1$ words of length t [13, p. 106, pp. 121–122] [15]. This produces a different test with a different expected outcome.
   The expected number of missing words in the cyclic case is different from the non-cyclic case. Using the methods of Guibas and Odlyzko [6], and Rivals and Rahmann [23], and Maple code provided by Edlin and Zeilberger [3], the corresponding μ for the cyclic versions of the OPSO, OQSO and DNA tests was calculated to be:

OPSO:	141 909.194 619 723 81
OQSO:	141 909.194 525 907 72
DNA:	141 909.184 583 083 19

   The corresponding σ is not yet known.

2. Different number of words.
   In the Marsaglia and Zaman [18, 17] (string) versions of the OPSO, OQSO and DNA tests, the number of words in the sequence is $n-t+1$. In the TestU01 1.2.1 (cyclic) implementation of these tests, the number of words in the sequence is n, yet the number of words used to calculate the expected value μ and the variance σ^2 is $n-t+1$. As a result, TestU01 1.2.1 sets μ to the following values.

OPSO:	141 910.329 955
OQSO:	141 912.329 955
DNA:	141 918.329 955

The effect of the combination of differences 1 and 2 results in an inaccurate expected value being used to calculate the p-value.

Precise values of the expected value and variance for the number of missing words in the Monkey tests were calculated using the methods of Guibas and Odlyzko [6], and Rivals and Rahmann [23, 22], with the help of Maple code provided by Noonan and Zeilberger [21], and Edlin and Zeilberger [3]. Calculation of the variance for the OPSO test needs the calculation of 6 generating functions, the OQSO test needs 55, and the DNA test needs 4592. The methods used for the calculation are described in Section 6 below. The resulting values of μ and σ are listed in Table 2 to 20 decimal places.

Table 2 Improved Monkey test means and standard deviations

	μ	σ
OPSO:	141 909.329 955 006 918 91	290.462 263 403 751 797 69
OQSO:	141 909.600 532 131 639 00	294.655 872 365 832 448 93
DNA:	141 910.402 604 762 935 66	337.290 150 690 427 643 65

An improved patch for the TestU01 implementations of the Monkey tests eliminates differences 1 and 2 above, and uses the improved means and standard deviations listed in Table 2. The Kolmogorov test results using the corresponding sequences of p-values extracted from 1024 repetitions of the improved TestU01 PseudoDIEHARD battery are as follows.

		D	p
OQSO	BX:	0.008	0.05186
	MT:	0.006	0.2456
DNA	BX:	0.0038	0.7527
	MT:	0.0034	0.8589

5 Final TestU01 results

The summarized Kolmogorov test results for 1024 repetitions of the PseudoDIEHARD battery, for different versions of TestU01 for the generators MT and BX, are as follows.

Version		D	p
0.6.1	BX:	0.0113	1.221×10^{-14}
	MT:	0.011	5.951×10^{-14}
1.2.1	BX:	0.0063	7.01×10^{-5}
	MT:	0.0056	0.000 6376
Improved	BX:	0.0032	0.1352
	MT:	0.0025	0.3982

The improved version no longer results in rejection of hypothesis $\mathscr{H}_2$.

6 Computation of the variances for the Monkey tests

The following analysis is based on that of Rahmann and Rivals [22]. The problem is to find the mean and variance of the distribution of the number of missing words in a random string. A random string S of length n is formed from an alphabet of size α, with each character equally likely. The string S contains $n-t+1$ overlapping words of length t. There are therefore α^n possible strings S_i, and α^t possible words W_j.

For the remainder of this section, consider α and t to be fixed. Define the indicator variable $v_{i,j}$ to be 1 if word W_j is missing from string S_i, and 0 otherwise. The number of words missing from string S_i is thus

$$X_i := \sum_j v_{i,j}.$$

The probability that both words W_j and W_k are missing from a random string S of length n is

$$a_{j,k}^{(n)} := \alpha^{-n} \sum_i v_{i,j} v_{i,k}.$$

Define the generating functions $A_{j,k}(z) := \sum_n a_{j,k}^{(n)} z^n$ and $A_j := A_{j,j}$. Define the random variable $X^{(n)}$ to be the number of words missing from a random string S of length n. The expected value of $X := X^{(n)}$ is then

$$\mathbb{E}[X] = \alpha^{-n} \sum_i X_i = \alpha^{-n} \sum_i \sum_j v_{i,j} = \sum_j a_{j,j}^{(n)}.$$

The variance is $\mathrm{Var}[X] = \mathbb{E}[X^2 - X] + \mathbb{E}[X] - (\mathbb{E}[X])^2$, with

$$\mathbb{E}[X^2 - X] = \alpha^{-n} \sum_i \sum_{j \neq k} v_{i,j} v_{i,k} = \sum_{j \neq k} a^{(n)}_{j,k}.$$

Given words B and C of length t, with $B = B_0 \ldots B_{t-1}$ etc. define the (word overlap) correlation vector BC by $BC_s = 1$ if $B_{r+s} = C_r$ for $r \in \{0, \ldots, t-s-1\}$, and $BC_s = 0$ otherwise. Figure 4 shows an example of a correlation vector. The correlation vectors BB, CC are called autocorrelations [5, 23].

Fig. 4 The correlation vector for the words B=DANGER, C=ANGERS.

$$\begin{array}{rl} B: & \boxed{\mathrm{D\ A\ N\ G\ E\ R}} \\ C: & \boxed{\mathrm{A\ N\ G\ E\ R\ S}} \\ & \quad\boxed{\mathrm{A\ N\ G\ E\ R\ S}} \\ & \qquad \ldots \\ BC: & \boxed{0\ 1\ 0\ 0\ 0\ 0} \end{array}$$

For the correlation vector v, define the correlation polynomial

$$P_v(z) := v_0 + v_1 z + \ldots + v_{t-1} z^{t-1}.$$

For $P_j := P_{W_j W_j}$, the generating function A_j is given by Guibas and Odlyzko [6, Theorem 1.1], and Rahmann and Rivals [22, Lemma 2.1] as

$$A_j(z) = \frac{P_j(z/\alpha)}{(z/\alpha)^t + (1-z)P_j(z/\alpha)}.$$

For $P_{g,h} := P_{W_g,W_h}$, the correlation matrix is $M_{j,k}(z) := \begin{bmatrix} P_{j,j}(z) & P_{j,k}(z) \\ P_{k,j}(z) & P_{k,k}(z) \end{bmatrix}$.

Given $M := \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$, define $M^V := \begin{bmatrix} m_{22} & m_{21} \\ m_{12} & m_{11} \end{bmatrix}$ and

$R(M) := m_{11} + m_{22} - m_{12} - m_{21}$.

Define the equivalence class $[M] := \{M,\ M^T,\ M^V,\ M^{TV}\}$, so that

$$[M_{j,k}(z)] = \{M_{j,k}(z),\ M^T_{j,k}(z),\ M_{k,j}(z),\ M^T_{k,j}(z)\}.$$

Note that $M' \in [M]$ implies $\det M' = \det M$ and $R(M') = R(M)$.

The generating function $A_{j,k}$ for the pair W_j, W_k is given by Rahmann and Rivals [22, Lemma 3.2] as

$$A_{j,k}(z) = \frac{Q_{j,k}(z/\alpha)}{(1-z)Q_{j,k}(z/\alpha) + (z/\alpha)^t R_{j,k}(z/\alpha)},$$

where $Q_{j,k}(z) := \det M_{j,k}(z)$, and $R_{j,k}(z) := R(M_{j,k}(z))$. See also [6, 21, 24].

Standard methods are used to obtain $a^{(n)}_{j,k} = [z^n]A_{j,k}(z)$ from each $A_{j,k}(z)$. See e.g. Graham, Knuth and Patashnik [4, Section 7.3].

We could simply sum $a^{(n)}_{j,k}$ for all $\alpha^{2t} - \alpha^t$ word pairs $W_j \neq W_k$, but for Marsaglia's tests, $\alpha^{2t} = 2^{40}$. So instead we enumerate correlation classes and count the word pairs for each class.

Each word pair W_j,W_k containing β distinct letters yields a partition of the set $\{0,\ldots,2t-1\}$ into β nonempty subsets, which is equivalent to a restricted growth string of length $2t$ having exactly β distinct letters. The string S of length $2t$ is a restricted growth string if $S_k \leqslant S_j + 1$ for each j from 0 to $k-1$, for k from 1 to $2t-1$.

Each permutation of the alphabet preserves the correlation matrix. The set of word pairs having β distinct letters splits under the symmetry group $\mathbb{S}_\alpha$ into orbits of size $\alpha!/(\alpha-\beta)!$.

Define $N[M](\alpha) = \sharp\{(j,k) \mid M_{j,k} = [M]\}$, the number of word pairs associated to the correlation class $[M]$. For $\alpha \leqslant 2t$, the following algorithm is used to determine all correlation classes $[M]$, and find $N[M](\alpha)$ for each one.

For each β from 1 to α, for each restricted growth string of length $2t$ having exactly β distinct letters:

1. Find the correlation class for the corresponding word pair.
2. Add $\frac{\alpha!}{(\alpha-\beta)!}$ to the count for the correlation class.

For each correlation class $[M]$, $N[M](\alpha)$ is a polynomial in α of maximum degree $2t$. For $\alpha > 2t$, to find $N[M](\alpha)$, first find $N[M](\gamma)$ for γ from 1 to $2t$, and then interpolate the resulting polynomial.

Acknowledgements This paper is the result of correspondence with Richard Simard of the University of Montreal, and with Eric Rivals at LIRMM, Montpellier. Art Owen of Stanford encouraged this paper and the presentation on which it is based. Jörg Arndt assisted in programming and proofreading. The combinatorial, statistical and programming techniques used were refined through various discussions with Sacha van Albada, Peter Drysdale, and others at Complex Systems, School of Physics, University of Sydney; and Jörg Arndt, Sylvain Forêt, John Maindonald, Judy-anne Osborn, and others at the Mathematical Sciences Institute, Australian National University.

References

1. Brent, R.P.: Some uniform and normal random number generators (2006–2008). URL `http://wwwmaths.anu.edu.au/~brent/random.html`
2. Brent, R.P.: Some long-period random number generators using shifts and xors. ANZIAM Journal **48 (CTAC2006)**(1), C188–C202 (2007). URL `http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/40`

3. Edlin, A.E., Zeilberger, D.: The Goulden-Jackson cluster method for cyclic words. Advances in Applied Mathematics **25**, 228–232 (2000)
4. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics, second edn. Addison-Wesley, New Jersey (1994)
5. Guibas, L.J., Odlyzko, A.M.: Periods in strings. Journal of Combinatorial Theory Series A **30**, 19–43 (1981)
6. Guibas, L.J., Odlyzko, A.M.: String overlaps, pattern matching, and nontransitive games. Journal of Combinatorial Theory Series A **30**, 183–208 (1981)
7. Kao, C., Tang, H.C.: Several extensively tested multiple recursive random number generators. Comput. Math. Appl. **36**(6), 129–136 (1998)
8. Knuth, D.E.: Seminumerical Algorithms, *The Art of Computer Programming*, vol. 2, second edn. Addison-Wesley, Reading, Massachusetts (1981)
9. Knuth, D.E.: Seminumerical Algorithms, *The Art of Computer Programming*, vol. 2, third edn. Addison-Wesley, Reading, Massachusetts (1998)
10. L'Ecuyer, P.: Testing random number generators. In: WSC '92: Proceedings of the 24th conference on Winter simulation, pp. 305–313. ACM, New York, NY, USA (1992). DOI 10.1145/167293.167354
11. L'Ecuyer, P.: Random number generators and empirical tests. In: Monte Carlo and quasi-Monte Carlo methods 1996 (Salzburg), *Lecture Notes in Statistics*, vol. 127, pp. 124–138. Springer, New York (1998)
12. L'Ecuyer, P., Hellekalek, P.: Random number generators: selection criteria and testing. In: Random and quasi-random point sets, *Lecture Notes in Statistics*, vol. 138, pp. 223–265. Springer, New York (1998)
13. L'Ecuyer, P., Simard, R.: TestU01: a software library in ANSI C for empirical testing of random number generators: User's guide, detailed version. Département d'Informatique et de Recherche Opérationnelle Université de Montréal (2005). URL `http://www.iro.umontreal.ca/~simardr/testu01/tu01.html`
14. L'Ecuyer, P., Simard, R.: TestU01: a C library for empirical testing of random number generators. ACM Trans. Math. Software **33**(4), Art. 22, 40 (2007)
15. L'Ecuyer, P., Simard, R., Wegenkittl, S.: Sparse serial tests of uniformity for random number generators. SIAM Journal on Scientific Computing **24**(2), 652–668 (2002). DOI 10.1137/S1064827598349033
16. Marsaglia, G.: The Marsaglia random number CDROM including the Diehard battery of tests of randomness (1995). URL `http://www.stat.fsu.edu/pub/diehard/`
17. Marsaglia, G.: Monkey tests for random number generators, (revised extract, 1995). Journal of Statistical Software **14**(13), 1–10 (2005). URL `http://www.jstatsoft.org/v14/i13/supp/1`
18. Marsaglia, G., Zaman, A.: Monkey tests for random number generators. Computers and Mathematics with Applications **26**(9), 1–10 (1993)
19. Matsumoto, M.: Mersenne Twister with improved initialization (2002). URL `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html`
20. Matsumoto, M., Nishimura, T.: Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation **8**(1), 3–30 (1998). DOI 10.1145/272991.272995
21. Noonan, J., Zeilberger, D.: The Goulden-Jackson cluster method: Extensions, applications, and implementations. J. Difference Eq. Appl. **5**, 355–377 (1999)
22. Rahmann, S., Rivals, E.: On the distribution of the number of missing words in random texts. Combinatorics, Probability and Computing **12**, 73–87 (2003)
23. Rivals, E., Rahmann, S.: Combinatorics of periods in strings. Journal of Combinatorial Theory Series A **104**(1), 95–113 (2003)
24. Rukhin, A.L.: Distribution of the number of words with a prescribed frequency and tests of randomness. Advances in Probability **34**(4), 775–797 (2002)
25. Tang, H.C.: A statistical analysis of the screening measure of multiple recursive random number generators of orders one and two. J. Statist. Comput. Simulation **71**(4), 345–356 (2001)
26. Tang, H.C., Kao, C.: Searching for good multiple recursive random number generators via a genetic algorithm. INFORMS J. Comput. **16**(3), 284–290 (2004)