Measurement Canada
An Agency of Industry Canada
Mesures Canada
Un organisme d'Industrie Canada

Specifications

Category: **STATISTICAL METHOD**	Specification: **S-S-01 (rev.1)**	Page: **1 of 13**
Document(s):	Issue Date: **2008-11-10**	Effective Date: **2008-11-10**
	Supersedes: **S-S-01**	

Specifications for Random Sampling and Randomization

1.0 Scope

1.1 These specifications define algorithms for random sampling and randomization and are applicable whenever regulations or other specifications reference them for the purposes of random sampling or randomization.

1.2 These specifications are applicable to such situations as:

(a) acceptance sampling of discrete units presented for inspection in lots;

(b) sampling for survey purposes;

(c) auditing of quality management system results; and,

(d) selecting experimental units, allocating treatments to them, and determining evaluation order in the conduct of designed experiments.

1.3 These specifications also include information to facilitate auditing or other external review of random sampling or randomization results where this is required by Measurement Canada or quality management personnel in accredited organizations.

1.4 No normative references are applicable to these specifications. For informative references, refer to the Bibliography in the appendix.

2.0 Authority

These specifications are issued under the authority of section 19 of the *Electricity and Gas Inspection Regulations*.

3.0 Terms, Definitions, Symbols, and Abbreviations

3.1 Terms and Definitions

3.1.1 Lot

Definite part of a population (3.1.2) constituted under essentially the same conditions as the population with respect to the sampling (3.1.8) purpose.

NOTE: The sampling purpose may, for example, be to determine lot acceptability, or to estimate the mean value of a particular characteristic.

3.1.2 Population

Totality of items under consideration.

3.1.3 Pseudo-independent Random Sampling

Sampling (3.1.8) where a sample (3.1.7) of *n* sampling units (3.1.9) is taken from a population (3.1.2) in accordance with a table of random numbers or a computer algorithm designed such that each of the possible combinations of *n* sampling units has a particular probability of being taken.

3.1.4 Random Sample

Sample (3.1.7) selected by random sampling (3.1.5).

3.1.5 Random Sampling

Sampling (3.1.8) where a sample (3.1.7) of *n* sampling units (3.1.9) is taken from a population (3.1.2) in such a way that each of the possible combinations of *n* sampling units has a particular probability of being taken.

3.1.6 Randomization

Process by which a set of items are set into a random order.

NOTE: If, from a population (3.1.2) consisting of the natural numbers 1 to n, numbers are drawn at random (i.e. in such a way that all numbers have the same chance of being drawn), one by one, successively, without replacement, until the population is exhausted, the numbers are said to be drawn "in random order".

If these *n* numbers have been associated in advance with *n* distinct units or *n* distinct treatments that are then re-arranged in the order in which the numbers are drawn, the order of the units or treatments is said to be randomized.

3.1.7 Sample

Subset of a population (3.1.2) made up of one or more sampling units (3.1.9).

3.1.8 Sampling

Act of drawing or constituting a sample (3.1.7).

3.1.9 Sampling Unit

One of the individual parts into which a population (3.1.2) is divided.

3.1.10 Sampling Without Replacement

Sampling (3.1.8) in which each sampling unit (3.1.9) is taken from the population (3.1.2) once only without being returned to the population.

3.1.11 Seed

Numerical value or set of values used to initialize a pseudo-independent random sampling (3.1.3) algorithm or to establish a starting point in a table of random numbers.

3.1.12 Simple Random Sample

Sample (3.1.7) selected by simple random sampling (3.1.13).

3.1.13 Simple Random Sampling

Sampling (3.1.8) where a sample (3.1.7) of n sampling units (3.1.9) is taken from a population (3.1.2) in such a way that all possible combinations of n sampling units have the same probability of being taken.

3.2 Symbols and Abbreviations

The key symbols and abbreviations used in these specifications are as follows:

mod modulo operator ($a \bmod b = a - b \lfloor a / b \rfloor$)

N lot size

n sample size

n_i size of the i^{th} sample

U uniformly-distributed random real variable on the open range (0, 1)

x_i the i^{th} value of the variable x

$\lfloor z \rfloor$ floor function of z (returns the integer portion of real value z)

4.0 Pseudo-independent Random Sampling Computer Algorithms

4.1 Overview

4.1.1 These specifications adopt a specific system of algorithms developed in bibliographic references [1, 5, and 8]. The algorithms have been designed to possess the mathematical and statistical properties required for random sampling, as well as to be portable with respect to implementation in different programming languages on different computer platforms and to facilitate verification and auditing of the selected sample values, which might be required for regulatory purposes.

4.1.2 The system of algorithms involves two major sub-systems:

(a) an optional initialization algorithm that automatically generates a quasi-random seed integer based on elapsed time from a reference date; and,

(b) a random number generator.

4.1.3 For verification or auditing purposes, the optional initialization algorithm mentioned in 4.1.2 a) and described in 4.2 would be by-passed with a manually-entered seed value. This value needs to be within the integer range from 1 and 2 147 483 398 inclusive. A copy of this input value is saved for records purposes when required. However, in general usage for quality control and designed experiment applications, there should be infrequent need to by-pass the option of automatic random seed generation, which should be the default option in practice.

NOTE: The presentations of the steps of the algorithms in this clause have been kept in a more mathematical format to aid in programming.

4.2 Initialization Algorithm

4.2.1 The initialization algorithm consists of:

(a) an elapsed time computation algorithm, referenced to a fixed past date and time; and,

(b) a random number generation algorithm based on the uniform distribution, called a random number of times based on the output of item a) above, to obtain a random seed based on the time-based input.

4.2.2 The following algorithm determines the number of seconds that has elapsed since 2000-01-01 00:00:00 to the current date and time:

(a) Capture the computer system's date and time to a string variable, save a copy of the variable for records purposes, and then parse the string into its time components (i.e. year, month, day, hour, minute, and second).

(b) Compute the number of fully elapsed days d_e since the reference time point, using the current date's full four-digit year y, month m_1, and day d numerical values processed as follows:

If $m_1 < 3$ then let $m_1 = m_1 + 12$ and let $y = y - 1$

$$d_e = d + \lfloor (153\ m_1 - 457) / 5 \rfloor + 365\ y + \lfloor y / 4 \rfloor - \lfloor y / 100 \rfloor + \lfloor y / 400 \rfloor - 730\ 426$$

NOTE: The equation for d_e may be slightly simplified for calendar years up to and including 2099 by replacing the terms following $\lfloor y / 4 \rfloor$ by "− 730 441".

(c) Compute the total number of seconds s_e elapsed since the reference date using the quantity obtained in step b) and the time of day (in 24-hour "hh:mm:ss" format) captured in the string variable in step a) in accordance with the following equation:

$$s_e = 86400\ d_e + 3600\ h + 60\ m_2 + s$$

where h, m_2, and s are the hours, minutes, and seconds respectively.

NOTE: Some programming languages have built-in functions to perform the calculation of s_e directly. Such intrinsic functions need to be validated before use, to ensure the effects of leap years and daylight saving time are properly handled.

(d) The value resulting from step (c) is the initializing seed for the random seed generator and is used to obtain the final seed. A copy of this value is saved to a separate variable for records purposes when required.

(e) The number of times j that the subsequent random number generator is to be called is a random integer between 1 and 100 inclusive, based on the two least significant digits of the value obtained in step (c) increased by 1, which may be expressed as follows:

$$j = s_e - 100 \lfloor s_e / 100 \rfloor + 1$$

4.2.3 The random number generator for the automatic seed generation (initialization function) algorithm takes the form of the linear congruential recurrence relation:

$$x_{i+1} = 40\ 692\ x_i \mod 2\ 147\ 483\ 399$$

which can be implemented on computers capable of handling 32-bit integers via the following steps:

(a) $k = \lfloor x_i / 52\,774 \rfloor$

(b) $x_{i+1} = 40\,692\,(x_i - 52\,774\,k) - 3\,791\,k$

(c) If $x_{i+1} < 0$ then let $x_{i+1} = x_{i+1} + 2\,147\,483\,399$

4.2.4 Generate the seed to the random sampling algorithm by assigning the result from 4.2.2 (c) to x_i and then calling the formula in 4.2.3 j times per step 4.2.2 (e), replacing x_i with x_{i+1} each time until the required number of calls are made.

4.2.5 The final value of x_{i+1} resulting from step 4.2.4 is a random integer between 1 and 2 147 483 398 inclusive and serves as the initial seed to the random sampling algorithm described in 4.3 [in particular, the value y_i in step 4.3.6 (b)]. A copy of this value is saved to a separate variable for records purposes when required.

4.3 Random Number Generation Algorithm

4.3.1 The random number generation algorithm consists of:

(a) a shuffling array that is populated by a uniform-distribution random number generation algorithm; and,

(b) a combination, uniform-distribution random number generation algorithm.

4.3.2 Create a 32-element array A to serve as a means of shuffling the output of the random sampling algorithm.

4.3.3 The following random number generator is used to populate the shuffling array:

$x_{i+1} = 40\,014\,x_i \mod 2\,147\,483\,563$

which can be implemented on 32-bit computers via the following steps:

(a) $k = \lfloor x_i / 53\,668 \rfloor$

(b) $x_{i+1} = 40\,014\,(x_i - 53\,668\,k) - 12\,211\,k$

(c) If $x_{i+1} < 0$ then let $x_{i+1} = x_{i+1} + 2\,147\,483\,563$

4.3.4 Initialize the array *A* by assigning the result from 4.1.3 or 4.2.5 to x_i and then calling the generator given in 4.3.3 (a) 40 times, replacing x_i with x_{i+1} on each call, discarding the first 8 values, and then assigning each of the remaining 32 output values of x_{i+1} to the array in reverse order (i.e. from element 32 down to element 1).

4.3.5 Set element 1 of array *A* (i.e. *A*[1]) as the initializing value *k* to the combination random number generation algorithm.

4.3.6 The combination random number generator for random sample generation takes the form of the following combination of linear congruential recurrence relations and array index determination steps:

(a) $x_{i+1} = 40\,014\ x_i \mod 2\,147\,483\,563$

(b) $y_{i+1} = 40\,692\ y_i \mod 2\,147\,483\,399$

(c) $J = \lfloor 32\ k / 2\,147\,483\,563 \rfloor + 1$

(d) $k = A[J] - y_{i+1}$

(e) $A[J] = x_{i+1}$

(f) If $k < 1$ then let $k = k + 2\,147\,483\,562$

NOTE: The two random number generators above are those described in 4.2.3 and 4.3.3 (refer to those clauses if 32-bit equivalent implementations are required).

4.3.7 The algorithm in 4.3.6 is initialized by setting x_i to the final value of x_{i+1} from 4.3.4 and setting y_i to the value referenced in 4.2.5. The values x_{i+1} and y_{i+1} serve as the subsequent values of x_i and y_i for all subsequent calls to the algorithm. A random index *J* to the shuffling array *A* is calculated using the value of *k* (from 4.3.5 initially), and the difference between *A*[*J*] and y_{i+1} is assigned to *k*, while *A*[*J*] is updated with x_{i+1}. Finally, the value of *k* is altered if necessary to produce a positive value.

4.3.8 The output of the random sampling algorithm is the value *k*, which is a random number between 1 and 2 147 483 562 inclusive, scaled as a standard uniformly-distributed real variable *U* over the range from 0 to 1, exclusive of the endpoint values of this range, as follows: $U = k / 2\,147\,483\,563$.

4.3.9 The output from 4.3.8 may be scaled as a uniformly-distributed integer variable *L* over the range from 1 to *N*, inclusive, as follows: $L = \lfloor N\ U \rfloor + 1$.

4.3.10 To generate a random sample, steps 4.3.6 to 4.3.9 are repeated until the desired number of random values is obtained.

4.4 Audit Records

When records are required to be maintained for audit purposes by Measurement Canada or a responsible authority, record the operator identifier, lot identifier, lot size, sample size(s), type of sampling employed, and lists of the units in the lot and in the sample(s).

In addition, with respect to the algorithms, record the manually entered seed per 4.1.3, or if the random seed generator is used then record the:

(a) computer system's date and time used to compute this initial seed;

(b) initial seed's value per 4.2.2 (d); and,

(c) final seed's value per 4.2.5.

5.0 Random Sampling Methods

5.1 General

5.1.1 This clause provides algorithms for random sampling strategies commonly used in legal metrology work.

5.1.2 Throughout this clause, U is defined as a random real variable, uniformly-distributed in the range from 0 to 1, exclusive of the endpoint values of the range, such as provided by the algorithm in 4.

5.2 Single Sampling

A single random sample of n distinct units from a lot of N units is generated without replacement by the following method:

(a) Generate a random real value U.

(b) Set L equal to $\lfloor N\,U \rfloor + 1$.

(c) Verify that the value of L has not been previously generated; if it is distinct, store the value, otherwise discard it.

(d) Repeat steps (a) to (c) until n different values of L are obtained.

(e) Optionally, sort the values in ascending order.

NOTE: If the resulting values of a single sample are not sorted, that sample may be used for sequential sampling inspection by inspecting each unit in the order selected.

5.3 Multiple Sampling

Multiple random samples of n_i distinct units from a lot of N units are generated without replacement by the following method:

(a) Generate a single sample of n_t distinct units from a lot of N units without replacement, where n_t is the total of the individual sample sizes n_i, leaving the values in original output order (i.e. unsorted).

(b) Take the first n_1 resulting values as the first sample, the next n_2 resulting values as the second sample, and so forth.

(c) Optionally, sort the values of each component sample in ascending order.

6.0 Revision

The purpose of Revision 1 is to update the presentation of these specifications in a manner consistent with the ISO international standard that Canada is developing on the subject. This specification does not introduce any substantive changes to the Agency's web application that has been in use for several years.

Alan E. Johnston
President
Measurement Canada

Appendix A - (Informative)

A. Tests of Algorithm Implementation

A.1 General

This appendix provides information to assist software developers with testing the correctness of their implementations of the random sampling algorithms in the specification.

A.2 Seed Calculation Tests

For the manually input date and time in the first column of the table below, the seed values in the second and third columns should result at the points in the algorithm indicated by the clause references.

Date and time	Seed 1 4.2.2 (c)	Seed 2 4.2.5
2009-01-15 16:16:16	285 351 376	1 774 249 844
2009-07-15 08:08:08	300 960 488	150 009 464
2010-01-15 16:16:16	316 887 376	1 593 377 912
2010-07-15 08:08:08	332 496 488	1 451 476 477

A.3 Tests of Component Random Number Generation Algorithms

Using initializing seeds of x_0 = 1and y_0 = 1, as applicable, for each of the random number generators and calling each generator 10 000 times, produces the following output:

(a) for x_{i+1} = 40 014 x_i mod 2 147 483 563 (4.3.3), $x_{10\,000}$ = 1 919 456 777;

(b) for y_{i+1} = 40 692 y_i mod 2 147 483 399 (4.2.3), $y_{10\,000}$ = 2 006 618 587; and,

(c) for the combined generator with shuffle array (4.3.6), $A[J]_{J = 10\,000}$ = 1 701 364 455.

A.4 Step-by-step Overall Test of Implementation

Using an initializing date and time of 2009-01-15 16:16:16, the intermediate and final outputs of the algorithms are as follows:

(a) per step 4.2.2 (b), d_e = 3 302

(b) per step 4.2.2 (c), s_e = 285 351 376

(c) per step 4.2.2 (e), j = 77

(d) per step 4.2.4 and 4.2.5, result = 1 774 249 844;

(e) per step 4.3.4, the 32 values in array A are:

J	$A[J]$	J	$A[J]$	J	$A[J]$	J	$A[J]$
1	1 773 883 525	9	12 989 333	17	1 843 118 480	25	925 629 865
2	1 376 260 681	10	1 236 571 744	18	1 301 824 472	26	879 056 303
3	324 244 626	11	150 838 841	19	2 024 723 015	27	257 361 492
4	616 012 910	12	1 379 547 554	20	1 640 100 338	28	1 402 037 236
5	1 753 573 598	13	1 594 841 833	21	1 715 924 041	29	1 031 539 864
6	238 867 782	14	363 535 288	22	1 979 383 646	30	981 619 081
7	591 860 039	15	643 814 074	23	1 293 133 612	31	81 117 341
8	64 148 416	16	1 662 338 174	24	504 407 049	32	2 036 123 857

(f) per step 4.3.5, k = 1 773 883 525;

(g) per step 4.3.6 (a), x_{i+1} = 1 548 645 074;

(h) per step 4.3.6 (b), y_{i+1} = 1 530 261 067;

(i) per step 4.3.6 (c), J = 27;

(j) per step 4.3.6 (d), k = -1 272 899 575;

(k) per step 4.3.6 (e), $A[J]$ = 1 548 645 074;

(l) per step 4.3.6 (f), k = 874 583 987.

Appendix B - (Informative)

B. Internet Random Sampling Application

B.1 General

B.1.1 Measurement Canada has developed an on-line internet application that implements the algorithms defined in this specification. The application is designed to generate one or more pseudo-independent random samples without replacement from a finite lot.

B.1.2 Subject to the provisions of the disclaimer associated with the application, its output may be used to satisfy legal requirements for sample selection and auditability under legislation enforced by Measurement Canada.

B.1.3 The application will also be of assistance to software developers as it can be used to supplement the tests in Appendix A to verify the correctness of user implementations.

B.2 Accessing the Application

B.2.1 The application can be accessed from Measurement Canada's Web site at http://www.mc.ic.gc.ca/.

Appendix C - (Informative)

C. Bibliography

[1] Bays, C. and Durham, S.D. (1976). Improving a Poor Random Number Generator. ACM Transactions on Mathematical Software, Vol. 2, No. 1 (March), pp. 59-64.

[2] ISO 3534-1:2006, Statistics — Vocabulary and symbols — Part 1: General statistical terms and terms used in probability.

[3] ISO 3534-2:2006, Statistics — Vocabulary and symbols — Part 2: Applied statistics.

[4] ISO 3534-3:1999, Statistics — Vocabulary and symbols — Part 3: Design of experiments.

[5] L'Ecuyer, P. (1988). An Efficient and Portable Combined Random Number Generator. Communications of the ACM, Vol. 31, No. 6 (June), pp. 742-749, 774.

[6] Marsaglia, G. (2003). Random Number Generators. Journal of Modern Applied Statistical Methods, Vol. 2, No. 1 (May), pp. 2-13.

[7] Park, S.K. and Miller, K.W. (1988). Random Number Generators: Good Ones are Hard to Find. Communications of the ACM, Vol. 31, No. 10 (October), pp. 1192-1201.

[8] Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992, 2001). Numerical Recipes in Fortran 77: The Art of Scientific Computing, Second Edition (Volume 1 of Fortran Numerical Recipes), Cambridge University Press, Cambridge, UK.