ORNL/TM-10868

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Generation of an Ordered Simple Random Sample Without Replacement Using One Random Number

Y. H. Etheridge
T. Wright

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4500N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this report, send in name with report and the library will arrange a loan.

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22161
NTIS price codes—Printed Copy: A02 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or 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 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. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-10868

Engineering Physics and Mathematics Division

Mathematical Sciences Section

GENERATION OF AN ORDERED SIMPLE RANDOM SAMPLE WITHOUT REPLACEMENT USING ONE RANDOM NUMBER

Y. H. Etheridge

and

T. Wright

Date Published - August 1988

Research sponsored by the
Applied Mathematical Sciences Research Program
U.S. Department of Energy
Office of Energy Research

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES

3 4456 0281459 6

Table of Contents

Description and Purpose 1

Structure: User Interface 2

Auxiliary Software 2

Precision 2

Time 3

Compiler Considerations 3

Acknowledgements 3

References 3

Generation of an Ordered Simple Random Sample Without Replacement Using One Random Number

by Y. H. Etheridge and T. Wright

Oak Ridge National Laboratory, U.S.A.

Keywords: Computation of Binomial Coefficients; One Random Number; Simple Random Sampling

Algorithm Implementation Language

All of the algorithms are implemented in the C programming language.

Description and Purpose

Nigam and Gupta(1984) presented by examples a method for the identification of the **n** units to be included in a simple random sample (**SRS**) for selection from a finite universe of **N**(> **n**) distinct and identifiable units. The method calls for the generation of only one random number to select a **SRS** of **n** units without having to list all $\binom{N}{n}$ possible samples - practically all other methods require the generation of between **n** and **N** random numbers. Wright(1989) has given a slightly faster and generalized algorithm and demonstrated that the generalized algorithm to generate a **SRS** has a natural relation with Pascal's Triangle. The slight improvement in speed is a direct result of the establishment of a link with Pascal's Triangle. Below is a program that executes the generalized algorithm as described by Wright(1989).

Assume a listing of the units in a finite universe numbered consecutively as $U_1, U_2, \ldots, U_N$ and that the desire is to select without replacement a **SRS** of size **n**. After computing $\binom{N}{n}$ using **mpfactor** as described below, one randomly selects one integer **r** from the set $\{1, 2, \ldots, \binom{N}{n}\}$. The **r** uniquely identifies the units to be included in the sample. The values for **N**, **n**, and **r** are submitted to the program **sample** and one receives the **SRS** to be observed. This sample has the following desirable properties (i) exactly **n** distinct units will be selected and (ii) units appear in the correct sequence (order), avoiding the need for subsequent sorting.

<u>Example 1.</u> If **N**=6 and **n**=4, then $\binom{6}{4}=15$. For **r**=12, the **SRS** to observe is U_2, U_3, U_4, and U_6.

<u>Example 2.</u> If **N**=500 and **n**=50, then

$$\binom{500}{50}=2314422827984300469017756871661048812545657819062792522329327913362690.$$

For **r**=2314422827×10^{60} the **SRS** to observe is

U_{167}, U_{172}, U_{174}, U_{183}, U_{187}, U_{199}, U_{200}, U_{203}, U_{204}, U_{212}, U_{217}, U_{219}, U_{224},
U_{231}, U_{241}, U_{244}, U_{258}, U_{260}, U_{265}, U_{266}, U_{280}, U_{283}, U_{290}, U_{296}, U_{301}, U_{302},
U_{308}, U_{309}, U_{311}, U_{320}, U_{328}, U_{333}, U_{347}, U_{355}, U_{359}, U_{390}, U_{394}, U_{402}, U_{412},
U_{417}, U_{418}, U_{424}, U_{427}, U_{443}, U_{445}, U_{463}, U_{474}, U_{475}, U_{484}, U_{487}.

Generation of a **SRS** with just one random number is important because it is exactly consistent with the manner in which simple random sampling is defined (Cochran, 1977; Cassel, Särndal, and Wretman, 1977), and it minimizes the concern with independence violations that accompany situations that call for a sequence (two or more) of random numbers generated by computers.

Structure: User Interface

sample N n r

where:

sample is the name of the program which generates the sample associated with $\begin{bmatrix} N \\ n \end{bmatrix}$ and the multiprecision random number, r.

mpfactor N n

where:

mpfactor generates the multiple precision calculation of $\begin{bmatrix} N \\ n \end{bmatrix}$. Using the algorithm in Goetgheluck(1987), $\begin{bmatrix} N \\ n \end{bmatrix}$ is computed by writing it as a product of primes.

Auxiliary Software

Both programs **sample** and **mpfactor** utilize the procedures **mpbinomial**, **prime**, and **smin**. **Mpbinomial** determines multiple precision binomial coefficients of its input parameters while **prime** supplies the primality of its input parameter. **Smin** is an adjunct to the multiple precision library software so that numeric character strings can be converted to multiple precision binary format.

Precision

Testing of these procedures has been carried out on a VAX[1] 11/780 running BSD 4.3 UNIX[2], a Sequent Balance 8000 running DYNIX[3] (a variant of BSD 4.2 UNIX), and an IBM PC/AT running UNIX System V Release 2. The binomial coefficient parameters have been limited to 16-bit precision, but the multiple precision of the generated output is limited only by available memory. For example the 281-digit output of the binomial coefficient $\begin{bmatrix} 1000 \\ 353 \end{bmatrix}$ is easily generated by the presented implementation of the algorithm. While the generation of the sample for N=500 things taken n=250 at a time using the 150-digit number, $r=1167443157\times10^{140}$, is calculated in an agreeable time as demonstrated below.

---

[1] VAX is a trademark of Digital Equipment Corporation.

[2] UNIX is a trademark of AT&T Bell Laboratories.

[3] DYNIX is a trademark of Sequent Computer Systems, Inc.

Time

Since the algorithm is somewhat dependent on the relative efficiency of the determination of $\begin{bmatrix} N \\ n \end{bmatrix}$ it is instructive to measure this calculation via **mpfactor**. The generation times of $\begin{bmatrix} 1000 \\ 353 \end{bmatrix}$ by **mpfactor** by each of these machines is 1.4 seconds, 1.9 seconds, and 2.4 seconds, respectively. The determination of the sample of n = 250 from a universe of N = 500 using the 150-digit number, $r = 1167443157 \times 10^{140}$, requires 3.6 seconds on a VAX, 5.3 seconds on a Sequent, and 10.3 seconds on a PC/AT. Finally, since the previously described example 2 can be considered a more "normal" instance for applying this algorithm the resultant times for the VAX, Sequent, and PC/AT are 2.0 seconds, 2.9 seconds, and 9.6 seconds, respectively.

Compiler Considerations

The crux of this algorithm implementation is that of using multiple precision arithmetic to carry out the dominant calculations. Therefore, the BSD UNIX library, **libmp**, is used on the VAX and Sequent machines. This library has been rewritten to run on UNIX System V and PC DOS. Examples, for compiling and generating executible procedures for BSD UNIX follow:

```
cc -o mpfactor mpfactor.c mpbinomial.c prime.c -lmp
```

and

```
cc -o sample sample.c mpbinomial.c prime.c smin.c -lmp
```

Acknowledgements

The authors are grateful to Elmon Leach who assisted with some of the programming, and to T. Dunigan and G. Ostrouchov for their helpful comments.

References

Cassel, C.-M., Särndal, C.-E., and Wretman, J. H. (1977). *Foundations of Inference in Survey Sampling*, John Wiley and Sons, New York.

Cochran, W. G. (1977). *Sampling Techniques*, 3rd Edition, John Wiley and Sons, New York.

Goetgheluck, P. (1987). "Computing Binomial Coefficients", *The American Mathematical Monthly* **94**, #4, 360-365.

Nigam, A. K. and Gupta, V. K. (1984). "A Method of Sampling with Equal or Unequal Probabilities without Replacement". *Applied Statistics* 33 #2, 227-229.

Wright, T. (1989). "A Note on Pascal's Triangle and Simple Random Sampling", *The College Mathematics Journal* **20** #1.

```
/*
 * mpfactor:
 *
 * Generate a binomial coefficient.
 */

#include <stdio.h>
#include <mp.h>

main(narg, parg)                                                    main
  int narg;
  char **parg;
{
  int N, n;
  MINT *mpbinomial();

  if (narg != 3)
      printf("Useage factor N n\n");
  N = atoi(*(parg + 1));              /* Convert the external ASCII strings to */
  n = atoi(*(parg + 2));              /*    internal Integer. */
  printf("Binomial coefficient of %d over %d:\n", N, n);
  mout(mpbinomial(N, n));             /* Print the multiple precision value */
                                      /*  for the binomial coefficient. */
}
```

```
/*
 * sample:
 *
 * Determine the ordered listing of the n units to be included in
 * a simple random sample from a universe of N to be derived without
 * explicitly listing all combinations of the N objects taken n at a time.
 */

#include <stdio.h>
#include <mp.h>

main(narg, parg)                                                    main
  int narg;
  char **parg;
{
  int N, n;
  MINT *mpbinomial(), *smin();
  MINT *prev, *next, sum, *r;
  int i, j, k, v;
  int l = 1;

  while (narg > 4)
    narg--;
  if (narg != 4)
  {
      printf("Useage: sample N n s\n");
  }
  N = atoi(*(parg + 1));
  n = atoi(*(parg + 2));

  r = smin(*(parg + 3));              /* Get the multiple precision random number. */

  printf("Universe is %d, sample size is %d, random sample is ", N, n);
  mout(r);

  k = 1;
```

```
    i = N - 1;
    j = n - 1;
    next = mpbinomial(i, j);
    move(next, &sum);
    prev = itom(0);
    do
    {
      MINT *mpj;
      int rem;

      while((v = mcmp(r, &sum)) > 0)          /* Applying the Algorithm */
      {                                       /* as given in Wright(1987). */
        MINT *Nj;

        move(&sum, prev);
        Nj = itom(i - j);
        mult(next, Nj, next);
        xfree(Nj);
        sdiv(next, i, next, &rem);
        if (rem != 0)
        {
          printf("sample: impossible mp divide nonzero remainder<%d>.\n", rem);
          exit(1);
        }
        madd(prev, next, &sum);
        i--;
        k++;
      }

      printf("U(%d)", k);                 /* Found a unit. */
      if (l++ >= 10)                      /* Print at most 10 units to a line. */
      {
        printf("\n");
        l = 1;
      }
      if (v == 0 || j <= 0 || i <= 0)     /* Found all units? */
        break;

      mpj = itom(j);
      mult(next, mpj, next);
      j--;
      xfree(mpj);
      sdiv(next, i, next, &rem);
      if (rem != 0)
      {
        printf("sample: impossible nonzero remainder for next sum<%d>.\n", rem);
        exit(2);
      }
      madd(prev, next, &sum);
      i--;
      k++;
    } while (1);

    while (j > 0)                         /* Show remaining units, if any. */
    {
      printf("U(%d)", N - j + 1);
      j--;
    }
    printf("\n");
}
```

mpbinomial

```
/*
 * mpbinomial:
 *
 * Generate a multiple precision binomial coefficient.
 */

#include <mp.h>

MINT *
mpbinomial(N, n)
  int N;          /* The first parameter in the binomial expansion of N over n. */
  int n;          /* The second parameter. */
{
  int prime();
  int i;
  MINT *prod;
  MINT *term;

  if ((N > 0  &&  N < n)  ||  n < 0)
    return(itom(0));

  /*  We will assume that N and n are non-negative. */

  prod = itom(1);          /* Set initial value for product calculation. */

  n = n < N - n ? n : N - n;      /* Use symmetry to calculate end terms. */

  for (i = 2; i <= N; i++)        /* Scan to N, cancel n-factors, and */
                                  /* determine prime elements in remaining */
                                  /* terms, according to Goetgheluck's */
                                  /* algorithm. */
  {
    int l = N;
    int m = n;
    int e = 0;
    int r = 0;

    if (!prime(i))
      continue;
    if (i <= (l - m))
    {
      if (i <= l / 2)
      {
        if ((i*i > l)  &&  (l % i  <  m % i))
          e = 1;
        else
        {
          do
          {
            int a = l % i;
            int b = m % i + r;

            l = l / i;
            m = m / i;

            if (a >= b)
              r = 0;
            else
            {
              e++;
              r = 1;
            }
          } while (l > 0);
        }
      }
```

```
        else
            e = 0;
      }
      else
        e = 1;
      for (; e > 0; e--)
      {
        term = itom(i);
        mult(prod, term, prod);
      }
    }
    return(prod);
}
```

prime

```
/*
 * prime(n):  Simplistic test n for primality.
 */

#define PRIME 1
#define NOTPRIME 0
int
prime(n)
  int n;            /* Test n for primality. */
{
  int i;

  if (n == 2)
    return(PRIME);
  if (n % 2 == 0)
    return(NOTPRIME);      /* n is even */
  for (i = 3; i < n; i += 2)          /* March through the remaining odds. */
    if (n % i == 0)
      return(NOTPRIME);    /* n not prime */
  return(PRIME);
}
```

smin

```
/*
 * Convert string to multi-precision number.  An adaptation of the libmp
 * entry, min.
 */

#include <stdio.h>
#include <mp.h>

MINT *
smin(s)
  char *s;   /* String to be converted. */
{
  MINT x,y,ten;
  MINT *a;
  int sign;
  short qten,qy;
  char *c;

  a = itom(0);
  xfree(a);
  sign = 1;
  ten.len = 1;
  ten.val = &qten;
```

```
    qten = 10;
    x.len = 0;
    y.len = 1;
    y.val = &qy;
    c = s;
    while (*c != '\0')
    {
      switch(*c)
      {
        case '\\':
          c++;
          break;
        case '\t':
        case '\n':
          a->len *= sign;
          xfree(&x);
          return(a);
        case ' ':
          continue;
        case '-':
          sign = -sign;
          break;
        default:
          if(*c>='0' && *c<= '9')
          {
            qy = *c - '0';
            mult(&x,&ten,a);
            madd(a,&y,a);
            move(a,&x);
            break;
          }
          else
          {
            c--;
            a->len *= sign;
            return(a);
          }
      }
      c++;
    }
    return(a);
}
```

ORNL/TM-10868

INTERNAL DISTRIBUTION

1. B. R. Appleton
2. D. J. Downing
3. T. H. Dunigan
4-8. Y. H. Etheridge
9-10. R. F. Harbison
11-15. J. K. Ingersoll
16. E. Leach
17-21. F. C. Maienschein
22. G. Ostrouchov
23-27. R. C. Ward
28-32. T. Wright
33. J. J. Dorning (Consultant)
34. G. H. Golub (Consultant)
35. R. M. Haralick (Consultant)
36. D. Steiner (Consultant)
37. Central Research Library
38. K-25 Plant Library
39. ORNL Patent Office
40. Y-12 Technical Library /Document Reference Section
41. Laboratory Records - RC
42-43. Laboratory Records Department

EXTERNAL DISTRIBUTION

44. Dr. Donald M. Austin, ER-7, GTN, Office of Scientific Computing, Office of Energy Research, Washington, DC 20545
45. David Hall, Statistics, Systems Department, Pacific Northwest Laboratory, P.O. Box 999, Richland, Washington 99352
46. George P. McCabe, Jr., Department of Statistics, Purdue University, West Lafayette, Indiana 47907
47. Dr. Ronald Peierls, Applied Mathematics Department, Brookhaven National Laboratory, Upton, New York 11973
48. Douglas S. Robson, Biometrics Unit, Cornell University, Ithaca, New York 14850
49. Dr. L. R. Shenton, Advanced Computational Methods Center, University of Georgia, Athens, Georgia 30602
50. Daniel L. Solomon, Department of Statistics, North Carolina State University, P.O. Box 5457, Raleigh, North Carolina 27650
51. Ray A. Waller, S-1, Statistics, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, New Mexico 87545
52. Office of Assistant Manager for Energy Research and Development, U. S. Department of Energy, Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, Tennessee 37831-8600

53-63. Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37830