COMPREHENDING THE CENTRAL LIMIT THEOREM

Lyndell M. Kerley, Ph.D.
Department of Mathematics
East Tennessee State University
Johnson City,TN 37614

Abstract

Simulation will be used to illustrate the Central Limit Theorem and the concept of testing a hypothesis.

Introduction

STATEMENT OF THE CENTRAL LIMIT THEOREM

No matter what type of distribution a random variable X has, provided its mean μ and variance σ^2 exist, the sampling distribution of sample means, where each random sample has size n, will have
(1) a mean = μ
(2) a standard deviation = $\sigma / \sqrt{n}$ and
(3) will be approximately normally distributed.

In order to show that the Central Limit Theorem is valid regardless of the choice of the distribution, a random variable X will have each of the following density functions.

Normal

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-((x-\mu)/\sigma)/2}, \text{ all real } x$$

Exponential

$$f(x) = \frac{1}{\lambda} e^{-x/\lambda}, \quad x > 0$$

Chi-Square

$$f(x) = \frac{x^{n/2-1} e^{-x/2}}{2^{n/2}\,\Gamma(n/2)}, \quad x > 0$$

Standard Uniform

$$f(x) = 1, \quad 0 < x < 1$$

Triangular

$$f(x) = \begin{cases} 1 + x, & -1 < x < 0 \\ 1 - x, & 0 \le x < 1 \end{cases}$$

Letting X have one of the five listed density functions, an approximation to the distribution of sampling means for a fixed sample size can be obtained by simulating the drawing of repeated samples and calculating the mean for each sample. The computed sample means obtained using simulation will then be used to see if (1) and (2) in the statement of the Central Limit Theorem hold and (3) the computed sample means will be tested for normality using a result by Kolmogorov and Smirnov as in (Hoel) and (Romano).

Observing that the distribution of X is known and using (1) of the Central Limit Theorem, the concept of testing the hypothesis that the mean of X = known mean of X will be illustrated by computing the percentage that the true null hypothesis is rejected.

Generation of Random Variates

If f(x) is the density function of a random variable X, then

$$F(x) = \int_{-\infty}^{x} f(t)dt.$$

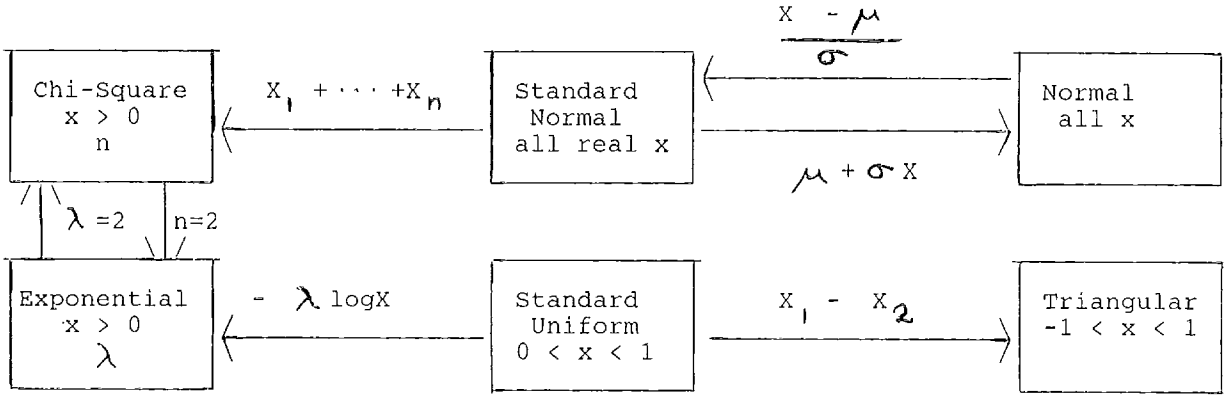
One way to generate random variates is to select a random number R, $0 < R < 1$, and solve $F(x) = R$ for x where x then represents a random variate. In the case of the standard uniform distribution and the exponential distribution, we see

$$R = x \qquad \text{implies} \quad x = R \text{ and}$$

$$R = 1 - e^{-x/\lambda} \qquad \text{implies} \quad x = -\lambda \ln(1 - R)$$

where it is also noted that $E(X)$ = mean of exponential distribution = λ .

The figure below by (Leemis) shows what transformations are needed to handle the remaining cases.



Approximation of the Standard Normal Distribution Function

In order to approximate the standard normal distribution function F, a polygonal function P is constructed by joining the points (-5,0), (-4,0.00003), (-3.5, F(-3.5)), (-3,F(-3)), . . , (3.5,F(3.5)),(4,0.99997), (5,1) where the ordinates of the points are found by using a normal cumulative frequency table. In the computer program that follows, the array X1 will represent the abscissas of the points and the array Y1 will represent the ordinates of the points.

Outline of Computer Program

```
Program Central_Limit_Theory;

Type
   Ar1 = Array[1..500] of real;

Var
   X, Y :   Ar1;
   . . . .
   . .

Procedure Approx_Normal(Var X1, Y1 : Ar1;
                        Var Size_of_Array : Integer);
(* Defines two 1-dimensional arrays X1 and Y1 for the normal case as mentioned earlier *)

Function Straight_Line_Interpolate(x : real; Size_of_Array : Integer;
                                   X, Y : Ar1 ) : Real;
(* For X[1] <= x <= X[Size_of_Array], interpolate at x.  That is, find J using a binary
search as in (Stubbs and Weber) so X[J-1] <=x< X[J]. Then define the function =  y  where
(x,y)  is the point on the line joining the points (X[J-1],  Y[J-1]) and (X[J],Y[J]). *)
```

```
Function Generate_Random_Variate(R : Real; Size_of_Array : Integer;
                                 X, Y : Ar1 ) : Real;
(* Having defined arrays X and Y both of size=Size_of_Array earlier in the program  which
approximate a given distribution function where 0 < Y[J] < 1 where 1 <=J <= Size_of_Array
and X represents values of the random variable, define
            function = Straight_Line_Interpolate(R,Size_of_Array,Y,X).
                 Note  that  R  must satisfy 0 <= R <= 1 *)

Function Distribution_Fcn(x : Real; Size_of_Array : Integer;
                          X, Y : Ar1 ) : Real;
(* Having defined arrays X and Y both of size=Size_of_Array earlier in the program  which
approximate a given distribution function where 0 < Y[J] < 1 where 1 <=J <= Size_of_Array
and X represents values of the random variable, define
            function = Straight_Line_Interpolate(x,Size_of_Array,X,Y). *)

Procedure Sort(Var A : Ar1 ; Size_of_Array : Integer);
(* Sorts an array A in ascending order.  This is needed in the Normality test of sampling
means based on Kolmogorov and Smirnov. The quick sort is very fast *)


Procedure Compute_Mean_SD( A : Ar1;  Size_of_Array:Integer;
                           Var Mean_of_Array, SD_of_Array: Real);
(* Given an array A of numbers, the mean and standard deviation are computed. *)



Procedure Test_CLT_Theory( Sigma, Mu : real;
                           Mean, Standard_Deviation : real;
                           Sample_Size : Integer);
(* Sigma = standard deviation of population                    *)
(* Mu    = mean of population                                  *)
(* Standard_Deviation = standard deviation of sampling means   *)
(* Mean               = mean of sampling means                 *)
(*                      Procedure compares                     *)
(*                          Mu and Mean                        *)
(*           Standard_Deviation and Sigma/Sqrt(Sample_Size)    *)
(* Central Limit Theorem says they  are approximately equal    *)


Procedure Testing_of_Null_Hypoth(Mu, Sigma, Sample_mean,
                                 Sample_SD : Real ;
                                 Sample_Size : Integer ;
                                 Var Cur_No_of_Rejections:Integer);

(* Mu = known mean of population, Sigma = known standard deviation of population, Sample_
Size  = prescribed sample size,  Sample_mean = computed sample mean for a given  sample,
Sample_SD  =  computed standard  deviation for a given  sample,  Cur_No_of_Rejections  =
current number of times that NULL HYPOTHESIS has been rejected as described below.


Define  Z = (Sample_mean - Mu)/(Sigma/Sqrt(Sample_Size)).    If Z > 1.96  or  Z <  -1.96,
increment Cur_No_of_Rejections by one thereby denoting a rejection of the null hypothesis
that  the mean  of the population = Mu which  we know to be  true  by our  choice  of one
of  the  five  known probability distributions.  1.96 implies a five per  cent  level  of
confidence. *)


Procedure Confidence_Results(Var Cur_No_of_Rejections,
                             No_of_Samples : Integer);
(*  Computes Per Cent = 100*Cur_No_of_Rejections/No_of_Samples which should be less  than
or equal to five per cent according to statistical theory. *)


Function Test_Table(No_of_Samples : Integer) : Real;
(*  This function gives the D statistic of the Kolmogorov-Smirnov Test for Normality at a
5%  level  of significance which is based on sample size.  Such a table can be  found  in
(Hoel) and (Romano) *)
```

```
Procedure Normality_Test( A : Ar1; Sigma, Mu : Real;
                          No_of_Samples : Integer);
(* A = Array of Sampling Means,  Sigma = standard deviation of Sampling Means,  Mu = Mean
of Sampling means, No_of_Samples=Number of computed sample means which will be the number
of times that one chooses to sample.


Having sorted the array A in ascending order, this procedure constructs a table
consisting of rows with one row for each sample mean A[J] with headings A[J], Obs prop,
Z-Stat, Exp Prop, Diff, and Abs Diff where Obs prop = J/No_of_Samples, Z_Stat = (A[J] -
Mu)/Sigma, Exp Prop = Distribution_Fcn(Z_Stat,Size_of_Array,X1,Y1) where X1 and Y1 arose
from an invocation of the procedure Approx_Normal, Diff = Obs - Exp Prop, Abs Diff =
Absolute value of Diff. By using a loop, each row of the table is constructed and the
largest value of Abs Diff in the table is denoted D. Setting D1 =
Test_Table(No_of_Samples), we compare D and D1. By the non-parametric one-sample test for
normality by Kolmogorov and Smirnov, we reject the hypothesis of normality if D > D1 and
fail to reject the hypothesis if D <= D1. *)



Procedure Results(Var A : Ar1,
                  Var Sample_SD, Sample_Mean : Real;
                  Var No_of_Samples: Integer);
Var
    . . . . . .

Begin
   Approx_Normal(X1,Y1,Size_of_Array); (* Needed for normality test at least *)
   Writeln('Enter the number ( > 0 ) of samples to be drawn');
   Readln(No_of_Samples);
   Writeln('Enter the desired fixed sample size ( > 0 )');
   Readln(Sample_Size);
   Writeln('Enter either 1, 2, 3, 4 or 5');
   Writeln('where 1 = Exponential Distribution with specified mean');
   Writeln('where 2 = Normal Distribution with specified mean and standard deviation');
   Writeln('where 3 = Chi Square Distribution with 3 degrees of freedom (df)');
   Writeln('where 4 = Standard Uniform Distribution');
   Writeln('where 5 = Triangular Distribution');
   Readln(Code);
   Case Code of
   1 : Begin
          Writeln('Enter the mean of the Exponential Distribution');
          Readln(Mu);
          Sigma := Mu;
       End;
   2 : Begin
          Writeln('Enter the mean of the Normal Distribution');
          Readln(Mu);
          Writeln('Enter the standard deviation of the Normal Distribution');
          Readln(Sigma);
       End;




   3 : Begin  (* Df = degrees of freedom of Chi-Square distribution *)
          Df := 3;  Mu := Df;  Sigma := Sqrt(2*Df); (* Chi-Square with 3 Df *)
       End;
   4 : Begin
          Mu := 0.5; Sigma := Sqrt(1/12); (* Uniform Distribution*)
       End;
   5 : Begin
          Mu := 0; Sigma := 1/Sqrt(6) (* Triangular Distribution *)
       End;
   End;  (* Case *)

(* Having chosen X to have a particular distribution, we now     *)
(* sample from the known population                              *)

   Cur_No_of_Rejections := 0;  (* Initialize no of rejections *)
                               (* of NULL HYPOTH = Mu         *)
```

```
   For I := 1 to No_of_Samples Do

   Begin
      Sum := 0;         (* Initialize. Used for computing sample   *)
      Sum_sqrs := 0;    (* mean and sample standard deviation.     *)
      For J := 1 to Sample_Size Do     (* Draw 1 sample of size =  *)
      Begin                            (*     Sample_Size          *)

      Case Code of   (* Code was  chosen  earlier *)
          (* Random is random number generator with 0 < Random < 1 *)

      1 : Y := -Mu*Ln(1 - Random);      (* Exponential Distribution *)
      2 : Begin
             Y := Random;  (* Normal Distribution *)
             (* Generate a standard normal variable N(0,1) *)
             Y := Generate_Random_Variate(Y,Size_of_Array,X1,Y1);
             Y := Mu + Y*Sigma;
          End;
      3 : Begin
             Y := Random;
             (* Generate a standard normal variable N(0,1) *)
             Y := Generate_Random_Variate(Y,Size_of_Array,X1,Y1);
             Y := Y*Y (* Square of N(0,1) variable is Chi-square variable with 1 df *)
             Y2 := -2*Ln(1 - Random) (* Chi-square with 2 df noting the figure *)
             Y := Y + Y2 (*Chi-square with 3 df by additivity of chi-squares *)
             (* REMARK : Rather than df = 3, the above concepts can be used *)
             (*          to handle df for any positive integer.             *)
          End;
      4 : Y := Random;  (* Standard Uniform Distribution with mean = 0.5 *)
      5 : Begin
             Y := Random;
             Y2 := Random;
             Y := Y - Y2; (* Triangular Distribution noting the figure *)
          End;


      End; (* Case *)
      Sum := Sum + Y;
      Sum_sqrs := Sum_sqrs + Y*Y;
      End; (* J *)
      Sample_Mean := Sum/Sample_Size;
      SS          := Sample_Size;
      Sample_SD   := Sqrt((Sum_sqrs - Sum*Sum/SS)/(SS - 1));
      Testing_of_Null_Hypoth(Mu,Sigma,Sample_Mean,Sample_SD,
                             (Sample_Size,Cur_No_of_Rejections);
      A[I] := Sample_Mean; (* Save sample means in an array *)
   End; (* I *)
   Confidence_Results(Cur_No_of_Rejections,No_of_Samples);
   Compute_Mean_SD(A,No_of_Samples,Mean,Sd);
   Test_CLT_Theory(Sigma,Mu,Mean,Sd,Sample_Size);
End; (* Results *)

Begin   (*  Program Central_Limit_Theory  *)

   (* First check parts 1 and 2 of Central Limit Theorem  *)
   (* and check concept of testing a hypothesis.          *)

   Results(A,Sd,Mean,No_of_Samples);

   (* Check normality of sample means *)

   Normality_Test(A,Sd,Mean,No_of_Samples);

End.  (* Program Central_Limit_Theory  *)
```

Results from computer runs

Distribution	Sample Size	No of Samples	Population Parameters Mu		$\sigma/\sqrt{n}$	Simulation Results Computed Mean		Standard Deviation	Percentage of Rejections of True Null Hypothesis	D Statistic
				*	#	*		#		
Normal	30	100	10	60	1.83	59.79		1.73	4	0.063
Exponential	30	100	10	10	1.83	9.97		1.76	4	0.046
Chi-Sq (3df)	30	100	2.45	3	.447	3.04		.430	5	0.041
Uniform	30	100	.289	.5	.053	.499		.053	1	0.069
Triangular	30	100	.408	0	.074	.004		.071	3	0.053

Analysis of Computer Results

Part (1) of the Central Limit Theorem appears to be verified by comparing the two columns that are headed by * with the first of two columns being the theoretical result and the second column being the simulated one. Part (2) of the Central Limit Theorem is similar except the two columns headed by # are to be considered.

The percentage of rejections of the true null hypothesis should be less than or equal to five per cent. This holds in all the cases.

From a D-Statistic with a sample size of 100, D = 0.136, which is greater than any of the values in the last column of the above table. Thus (3) of the Central Limit Theorem is shown since normality cannot be rejected in any of the cases in the table.

Suggestions for Classroom Use

The following modifications to the program may be useful.
(1) Use only the uniform distribution and illustrate (1) and (2) of the Central Limit Theorem as a programming assignment in a beginning structured programming course.
(2) In a basic statistics course, have the students use the program without their being concerned with the actual programming being involved.
(3) In a numerical analysis course, emphasize integration techniques such as Simpson's Rule and Romberg integration in the evaluation of the distribution functions. Also discuss the solving of equations R = F(x) where R = a random number and F(x) is a distribution function by using such techniques as Newton-Raphson, bisection, and false position.
(4) Without an emphasis on programming but with an emphasis on statistical concepts, a careful analysis of the computer output would be useful in a senior level statistics course. Also additional distribution functions might be added and studied.
(5) In a senior level simulation course, all aspects of the program would need to be analyzed carefully.

The author will be happy to supply the code for the complete computer program to anyone making a request. Moreover, if one will send a diskette, a copy can be made for use on an IBM PC and returned.

References

1. P.A. Bobillier, B. C. Kahan, and A. R. Probst, Simulation with GPSS and GPSSV, Prentice Hall, Inc., Englewood Cliff, NJ, 1976, pp 460-464, p 96.

2. John Freund and Ronald Walpole, Mathematical Statistics, Third Edition, Prentice Hall, Englewood Cliff, NJ, 1980.

3. Paul Hoel, Introduction to Mathematical Statistics, John Wiley & Sons, Inc., New York, NY, 1971, pp 324-329, p 401.

4. Lawrence M. Leemis, "Relationships Among Common Univariate Distributions" , The American Statistician, May 1986, Vol. 40, No. 2, pp. 143-146.

5. William Mendenhall, Introduction to Probability and Statistics, Seventh Edition, Duxbury Press, Boston, Mass, 1987.

6. A. Romano, Applied Statistics for Science and Industry, Allyn and Bacon. Inc, Boston, Mass 1977, pp 187-197, p 461.

7. D. Stubbs and N. Webre, Data Structures with Abstract Data Types and Modula-2, Brooks/Cole Publishing Co., Monterey, CA 1987, pp 163-169.