INTRODUCTION & PURPOSE

The main purpose of the "Sampling" application is to enhance the understanding of the results that occur when various types of distributions are randomly sampled. Such understanding is especially needed for those who do experimental research and need to compare the results of having treated two or more groups differently, and need to be able to appreciate whether any differences found might have happened by chance alone.

The concept of "**sampling**" in research always begins with the idea of some "**population of interest**" (or domain of interest) to the researcher. This population of interest to the researcher might be certain people, or certain animals, or certain inanimate things. Often times, the population of interest is extremely large and it would be impossible to measure every case in that population. Sometimes, every case measured can be expensive and it may be prohibitively expensive to measure a very large sample of cases from that population. Because of various kinds of constraints against measuring an entire population, one must settle for measuring only a "**sample**" of cases from that population. If one desires to generalize results obtained (from the sample of cases actually measured) that was a "**representative sample**" of the population of interest. This can be accomplished by what is called "**random sampling**."

Random sampling is accomplished when one selects a case from the population to be part of the sample that will be used. If the sampling is truly random, then "**every case in the population must have an equal chance of being selected**". When this is actually done, statistics can show that the average performance of the randomly sampled group is an **"unbiased" estimate** of what the average performance of the population would be. This isn't meant to suggest that the average performance of a randomly drawn sample will be exactly the same as the populations average performance, but if we continue measuring different random samples, the average performance of each sample may vary, but the average of those sample averages will keep getting closer and closer to the actual average of the population. Further, the extent of variation among those average sample performances can be used to obtain an unbiased estimate of the extent of variance in performance of the entire population.

Finally, the shape of the way performance is distributed within a specific population will also effect the results of random sampling, and while there are hundreds of differently shaped distributions, there are a few which can be used to describe a lot of different populations. Distributions are often described by four characteristics which are referred to as their moments, and the values most frequently used to describe them are:

(a) **mean**: the value (i.e., score) most representative of the sample or population and, thus, the central part of the distribution;
(b) **variance** or **standard deviation**: to what extent do the scores vary from the mean or central part;

(c) **skewness**: to what extent is the distribution of scores symmetric around the center; and
(d) **kurtosis**: to what extent is the distribution of scores peaked in the center or fairly flat.

More specifically, the purposes of the application is to visually show the user:
(a) what the names and shapes of some of the more frequently used distributions are;
(b) what the four moments are for each distribution;
(c) what the shapes of the means of samples taken from a given distribution will be;
(d) how the number of cases in the sample drawn will effect the results

THE APPROACH USED FOR THE PROGRAM

As currently envisioned, the user will be able to learn about various distributions and then to select and watch the program create random samples of various sizes from a selected distribution and show how the means of those random samples spread around the true mean of the population and how the means of the smaller sample sizes are more spread out than the means of samples that have more cases in them.

As the program is accomplishing the sampling, it is also collecting data about each mean so that, at the end of each run, it can also compute the four moments of the distributions of the means of the samples. These results can provide insight into such things as "The Central Limit Theorem" and how the "Unbiased" Variance of the sampled means. can be used to get an unbiased estimate of the variance of the population.

Finally, one can envision storing the results of a given run, such that they could be read in to show that replications of the runs would give highly similar results.

SECTIONS OF THE PROGRAM

SECTION 1. Definition of Arrays Needed

Rather than identifying all the arrays that will be needed, each section of the program will try to identify the needed storage arrays required by that section. A diagram of some of the main arrays that will be needed for the actual sampling is shown on gthe following page.

SECTION 2. Data needed to Instruct the User

When the program is initiated, the User should be given the option of learning (or being refreshed) on what the program does and how to make the program do what it does. Much of what is covered in the earlier "Introduction & Purpose" discussion might be modified and used for this purpose. Whatever is needed for this purpose can be placed in a storage file and read in as needed. *(NOTE: This section of the program can be added later.)*

m = type of sample

NC(1,m) = cases req.

NC(2,m) = cases actual

NK(1,m) = samples req.

NK(2,m) = samples act.

NC's & NK's	m=1	m=2	m=3	m=4	m=5	m=6
NC(1,m)	5	10	20	40	80	3200
NC(2,m)						
NK(1,m)	640	320	160	80	40	1
NK(2,m)						

X¹ is a random score drawn from a given distribution. the powers of **X¹** are then calculated.

X
X(1)=R¹
X(2)=R²
X(3)=R³
X(4)=R⁴

SX=SumX^p	1	2	3	4	5	6
SX(1,m)=∑X¹						
SX(2,m)=∑X²						
SX(3,m)=∑X³						
SX(4,m)=∑X⁴						

Mn = Mean

Vr = Variance

Sk = Skewness

Ku = Kurtosis

Stat
XMn
XVar
XSk
XKu

SS=SumStat	1	2	3	4	5	6
SS(1,m)=∑Mn						
SS(2,m)=∑Var						
SS(3,m)=∑Sk						
SS(4,m)=∑Ku						

StatSq
XMn²
XVar²
XSk²
XKu²

SSS=SumStatSq	1	2	3	4	5	6
SSS(1, =∑Mn²						
SSS(2, =∑Vr²						
SSS(3, =∑Sk²						
SSS(4, =∑Ku²						

"AvgStats" and the "VarStats" are the means and variances of the four statistics and are computed at the end of the run.

Option = Unbiased Var

AS=AvgStat	1	2	3	4	5	6
AS(1, =AvgMn						

AS	VS=VarStat	1	2	3	4	5	6
AS	VS(1, =AvgMn						
AS	VS(2, =AvgVr						
	VS(3, =AvgSk						
	VS(4, =AvgKu						

SECTION 3. Data Needed about Different Distributions

One kind of data that might be needed to be stored for each distribution is simply verbal data that discusses when such a distribution would be appropriate to describe some population of interest. *(NOTE: This section of the program can be added later.)*

A second kind of stored data might be equations for that distribution's mean, variance, skewness, kurtosis, and its probability distribution function. *(NOTE: This section of the program can be added later.)*

A third type of stored data might be in a Kx2xN array where K is the number of the distribution, N represents the length of the continuum and each cell in one of the two rows contains the appropriate score along the continuum and the other row contains the probability of that score. The would allow the drawing of the shape of the distribution.

All of the "Distribution Data" could be stored in a separate file and only read in when needed. A "DISTRIB(2,N) array within the main program could be used for reading in and holding the data for any "file-stored" distribution, or if the user is free to describe any distribution that he can think of.

SECTION 4. Start the Running the "Sampling" Application

This is where the User is given the opportunity to select (or create) a given distribution from which samples will be drawn. The main options, from which the User can select, will be:

(a) type of distribution (e.g., normal, uniform, binomial, etc.);
(b) if the selected distribution can be either, should it be "continuous" or "discrete";
(c) if "discrete" is selected, how many equal-interval cuts should be made in it;
(d) what is the smallest "sample size" desired (selected from a list of options);
(e) what will the total number of sampled cases be (selected from a list of options).

Initially, not all options will be selected by the User. The program should be able to be revised to allow such choices. This section of the program will use the selected options to initiate certain arrays and clear all arrays where data will subsequently be stored.

```
Start.A.Run:
Zero.All.Arrays:
        FOR M = 1 TO 6
            FOR J = 1 TO 4
            IF J < 3 THEN NC(J,M) = 0
            IF J < 3 THEN NK(J,M) = 0
            SX(J,M) = 0
            SS(J,M) = 0
            SSS(J.M) = 0
            AS(J,M) = 0
            VS(J,M) = 0
```

```
                    NEXT J
                NEXT M
USER.Selections:
        USER Selects Shape of Distribution to Use
                NDIST       = (1=Normal; 2= Uniform, …)
        USER Selects If Distribution is to be Continuous or Discrete
                NCUT        = (1=Continuous; 2= Discrete)
                GOSUB Read.In.Distribution
-------------------------------------------------------------------------------------------------------------------
SUBROUTINE
Read.In.Distribution:
        For the time being, if we stick with Continuous Normal (Gaussian) and Uniform
distributions we should be able to just use a "Rand" function to get an "R"
        RETURN
-------------------------------------------------------------------------------------------------------------------
                IF NCUT     = 2 THEN
                    GOSUB Define.Discrete.Cuts
-------------------------------------------------------------------------------------------------------------------
SUBROUTINE
Define.Discrete.Cuts
        For the time being, if we stick with Continuous distributions we won't need this.
But from a program planning point of view, this is where the calls to such a subroutine
should be in our program.
        RETURN
GOSUB Get.Sampling.Parameters
-------------------------------------------------------------------------------------------------------------------
                END IF
-------------------------------------------------------------------------------------------------------------------
SUBROUTINE
Get.Sampling.Parameters
                NLARGE = 3200:    REM = largest sample cases
                NSMALL  =      5:    REM = smallest sample cases
        RETURN
        For the time being, we can actually use this as the subroutine.  Later, we might
give the user some options
-------------------------------------------------------------------------------------------------------------------
Fill.Required:     REM Fills in Other Cells for Required Cases and Samples
        N(1,1) = NSMALL:  N(1,6) = NLARGE
                FOR M= 1 TO 6
                    IF N(1,M) = 0 THEN N(1,M) = 2 * N(1,M-1)
                    K(1,M) = NLARGE/N(1,M)
                NEXT M
```

SECTION 5. The Creation of Samples

This is actually the Main section of the program as it gets the random score ® for each case in a sample and creates the data for its mean, plots it, and computes data so that data on the distribution of the sample means can calculated later.

```
Get.A.Sample:
        M = 1
        FOR I = 1 TO NSMALL
                GOSUB Get.R:       REM  R = value of random case from distribution
```

SUBROUTINE
Get.R:

For the time being, if we stick with Continuous Normal (Gaussian) and Uniform distributions we should be able to just use a "Rand" function to get an "R". When we start using non-symmetric distributions, this subroutine will have to change.

```
IF NDist = 1 THEN R =  (use normally distributed random value function)
        IF NDist = 2 THEN R =  (use uniformly distributed random value function) -.5
        RETURN
```

```
                FOR IP = 1 TO 4:    REM  Gets the four powers of R
                        X(IP) = R**(IP):       SX(IP,M) = SX(IP,M) + X(IP)
                NEXT IP
        NEXT I
        NC(2,M) = NSMALL
        GOSUB Compute.Sample.Stats
        IF NK(2,1) < NK(1,1) THEN GOTO Get.A.Sample
GOTO Wrap.Up
```

Specs for the "Wrap.Up" section haven't been addressed as of now. Its not tough as all it does is to find the means of all the four values for describing the distributions of the sample means. All the data have been collected and the computations are fairly simple.

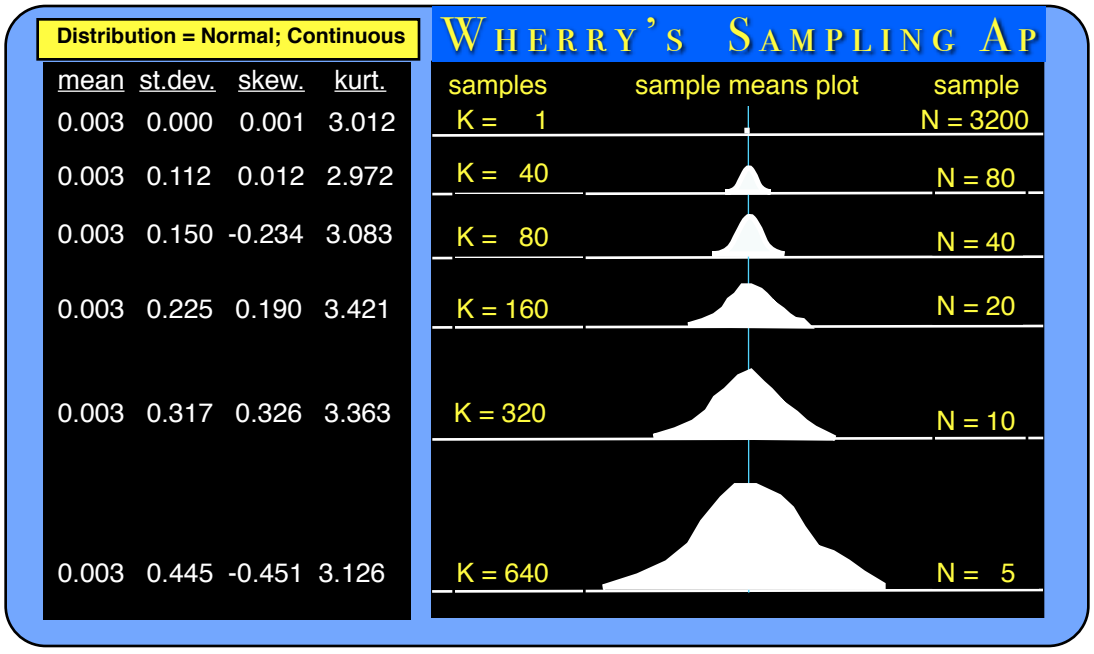
```
---------------------------------------------------------------------------------------------------------------
SUBROUTINE
Compute.Sample.Stats:
        IF NC(2,M) = NC(1,M) THEN
                Cases = NC(1,M)
                UnBias = Cases/(Cases-1)
        Compute.Mean:
                XMn1 = SX(1,M)/Cases:                                   Stat(1) = XMn1
        Compute.Variance.And.SD:
                X2Mn = SX(2,M)/Cases
                XMn2 = XMn*XMn*Unbias
                XVar   = X2Mn - XMn2:                                   Stat(2) = XVar
                XSD    = SQRT(XVr)
        Compute.Skewness:
                X3Mn = SX(3,M)/Cases
                XMn3 = XMn1*XMn2
                XSk    = X3Mn - 3*XMn1*X2Mn + 2*XMn3:                   Stat(3) = XSk
        Compute.Kurtosis:
                X4Mn = SX(4,M)/Cases
                XMn4 = XMn2*XMn2
                XKu = X4Mn-4*XMn1*X3Mn+6*XMn2*X2Mn-5*XMn4:  Stat(4) = XKu
                GOSUB Show.This.Sample.Mean
        Store.Sums.Of.Stats:
        This section takes the sums of the scores from a smaller sample and stores it a
data for the next larger sample.  But it only does it when all the data for the smaller
sample has been used.  This actually saves time.  After the transfer of the summed data
to the larger sample bin, the program determines if this larger sample now has enough
cases in it, such that its mean can now be computed.
                IF M < 6 THEN
                        FOR IStat = 1TO 4
                                SX(IStat,M+1) = SX(IStat,M+1) + SX(IStat,M)
                                SX(IStat,M) = 0
                                SS(IStat,M) = SS(IStat,M) + Stat(IStat)
                                SSS(IStat,M) = SSS(IStat,M) + Stat(IStat)*Stat(IStat)
                        NEXT IStat
                        NC(2,M+1) = NC(2,M+1) + NC(2,M)
                        NC(2,M) = 0
                        NK(2,M) = NK(2,M) + 1
                        M = M + 1
                ENDIF
                GOTO Compute.Sample.Stats
        END IF
        RETURN
---------------------------------------------------------------------------------------------------------------
```

Show.This.Sample.Mean:

This is where the USER can watch, on the screen, the resulting means for each sample size. Consider that the screen is painted black when the sampling actually begins. The portion on the left of the screen is where the wrap-up stats will be shown.

We have five different sample sizes (5, 10, 20, 40, and 80) . The vertical axis of the screen can be divided into five sections, with, say, 5-case samples at the lowest section, 10-case samples next, etc. The 5-case samples will have 640 samples to be plotted. The 10-case samples will only have 320 means to be plotted, so more space is required for small-case samples. A horizontal white line can be drawn across the screen for each sample size. The line represent zero means have been located above that line. I can imagine, say, a 100-cell array for each of the sample sizes. Those arrays are used to store how many means have fallen into each cell. When a mean is computed, its value is rounded to the nearest discrete number which should provide the data for the horizontal location where that mean should be indicated. The program then determines how many means have already fallen into that cell. One is then added to that cell and its value determines the vertical location on the screen where a new pixel will be placed to indicate that a mean has occurred at that location. The vertical location of the horizontal lines provides the location for zero means for that sample size.

A 7x7 array can be used to hold the horizontal and vertical screen position for the rows and columns of the headings and wrap-up data as well as for the headings and data and baselines for the plotted data. The same routine can then be used to plot each mean for each sample size by using the "M" from the program to determine which vertical position is appropriate for plotting the mean for that sample size.



(This is just a suggestion of what the screen might look like)