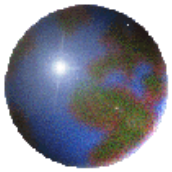


Monte Carlo Sampling
In
LINGO

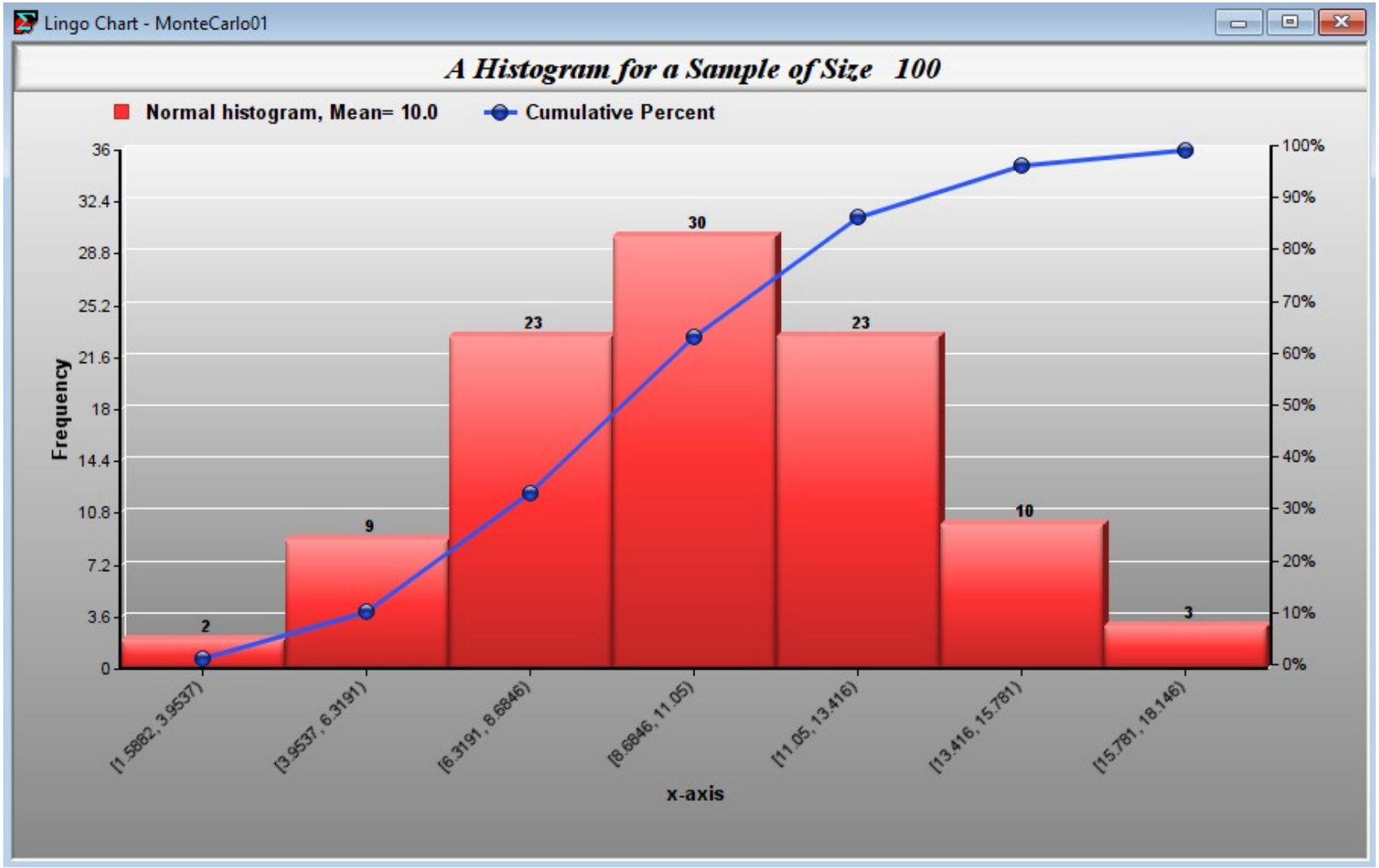
www.lindo.com

LINGO has a wide range of capabilities.
Here we illustrate how to do Monte Carlo sampling.

Keywords: Charts, Cholesky, Monte carlo, Sampling

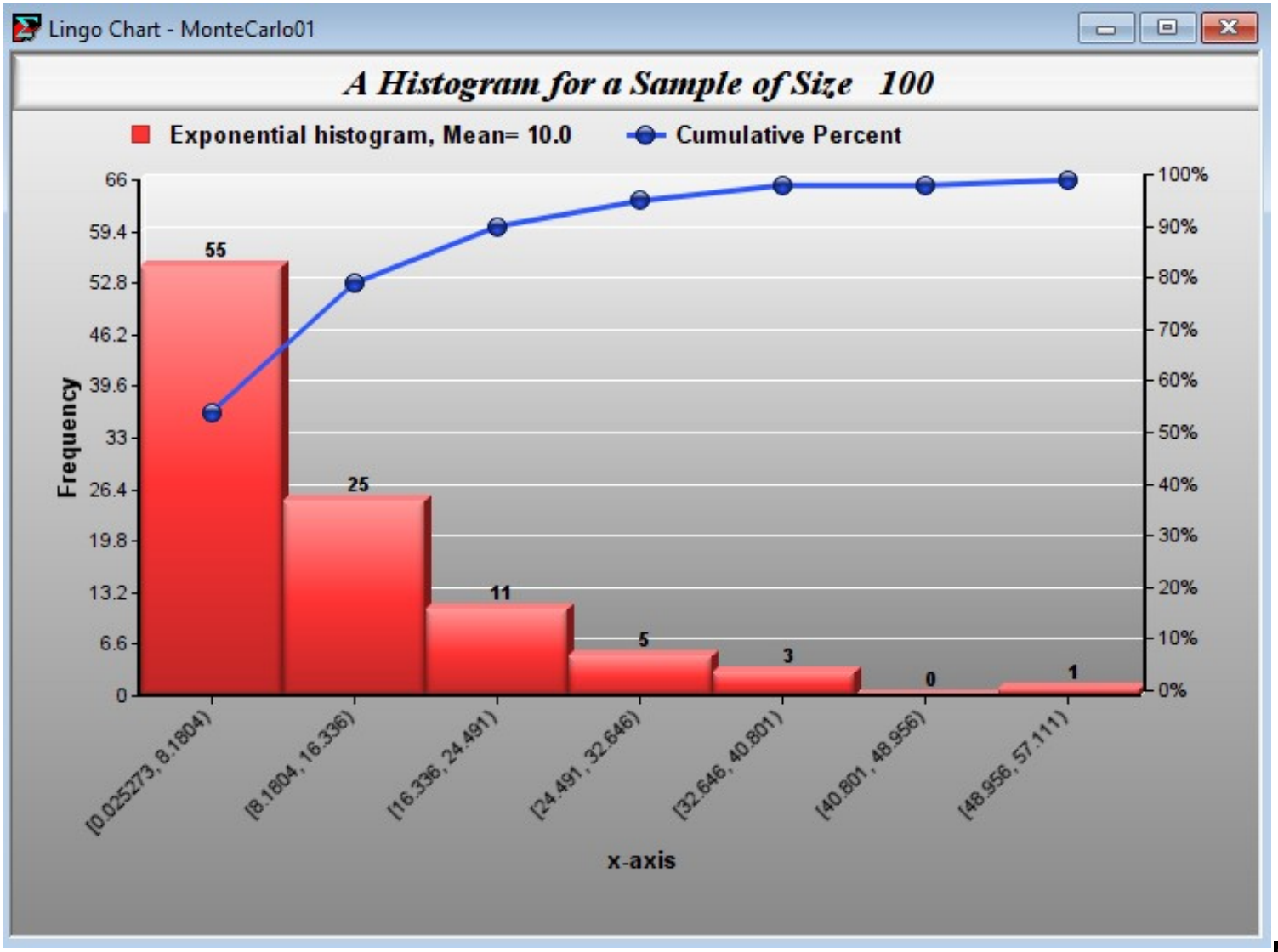
LINDO SYSTEMS INC.

LINGO Can Display the Histogram of a Sample…



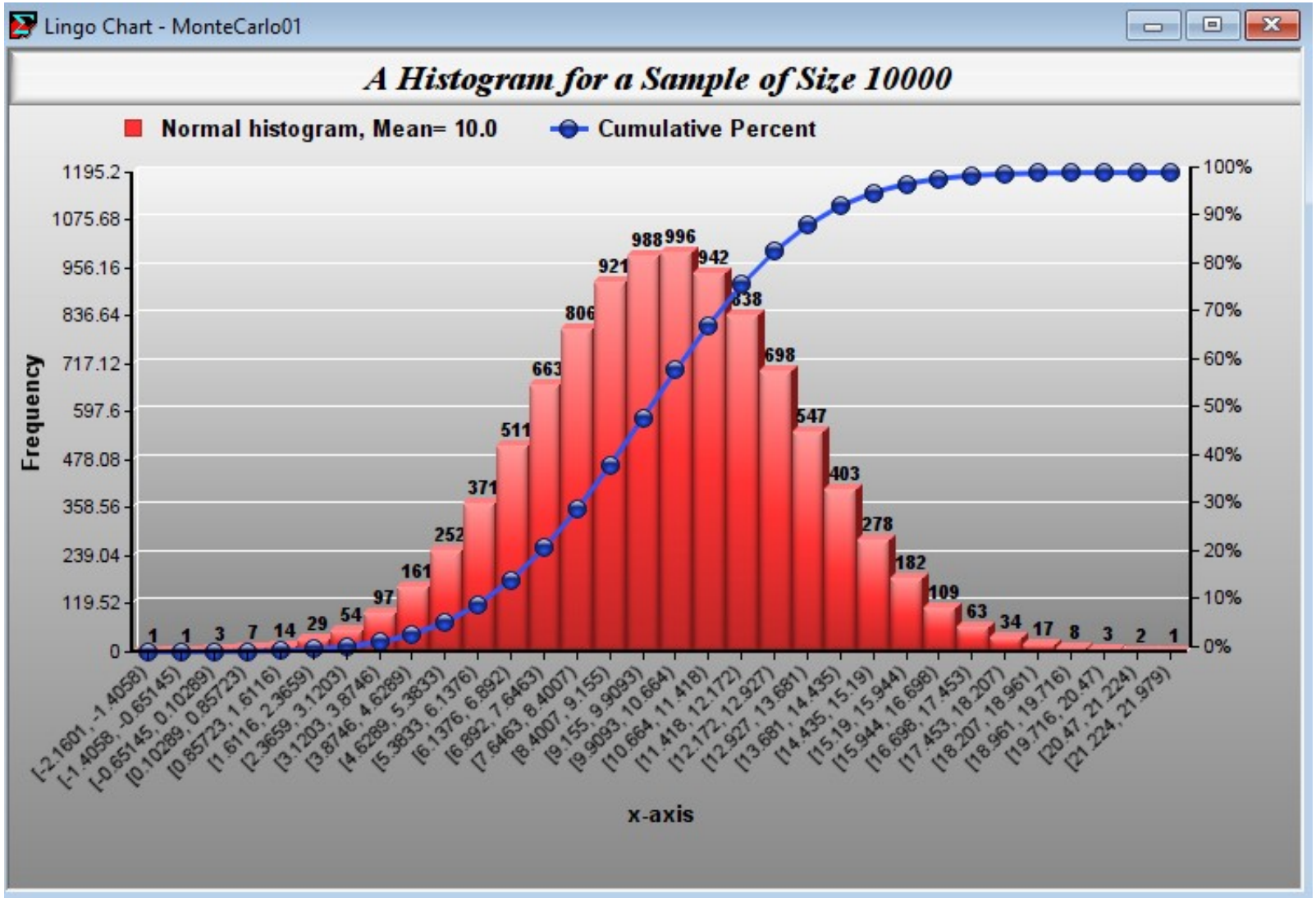
LINDO SYSTEMS INC.

Histogram of a Sample from an Exponential Distribution



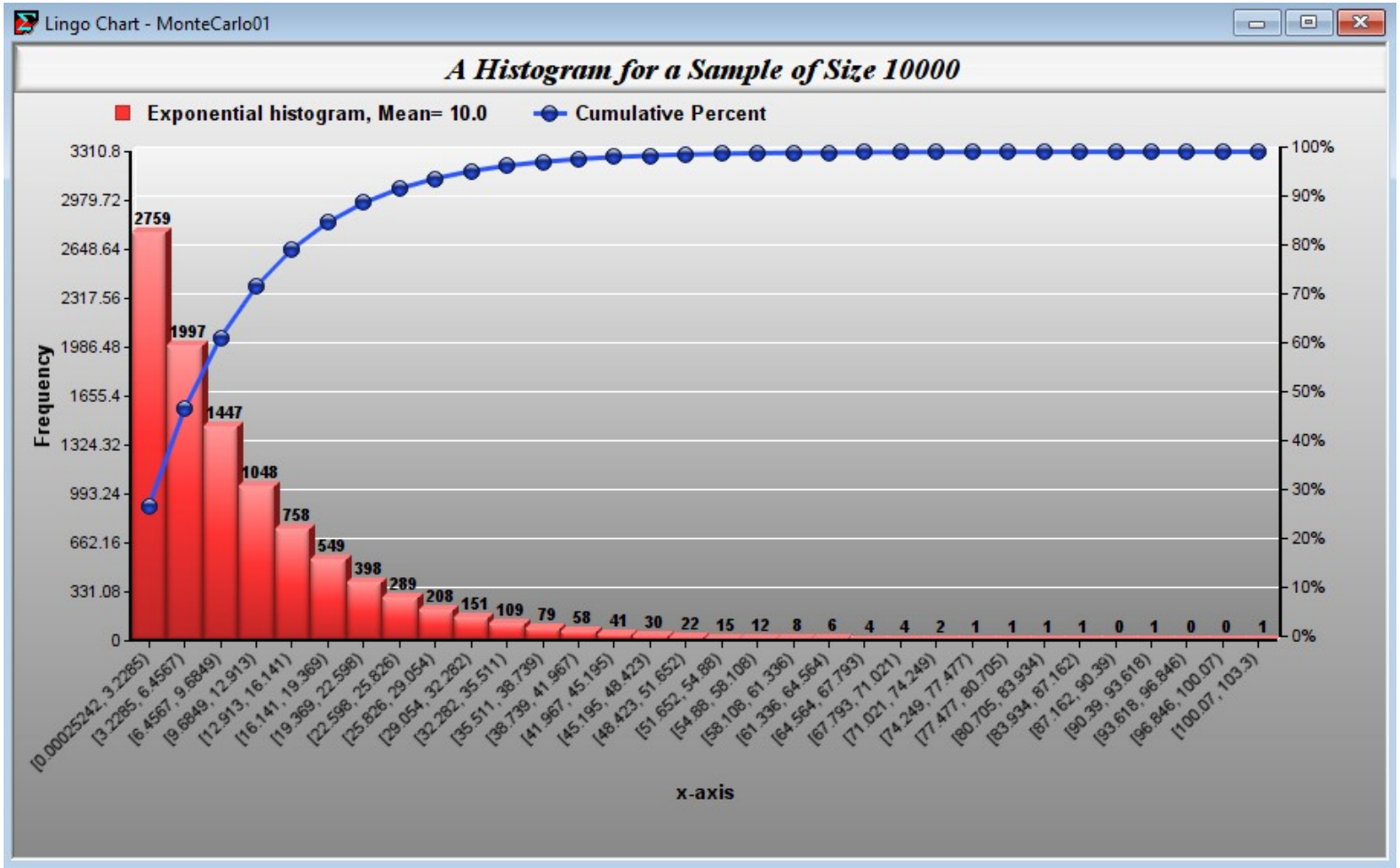
LINDO SYSTEMS INC.

Larger Sample Size gives a Smoother Histogram

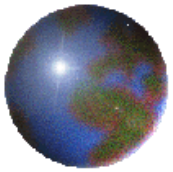


LINDO SYSTEMS INC.

Larger Sample Size gives a Smoother Histogram



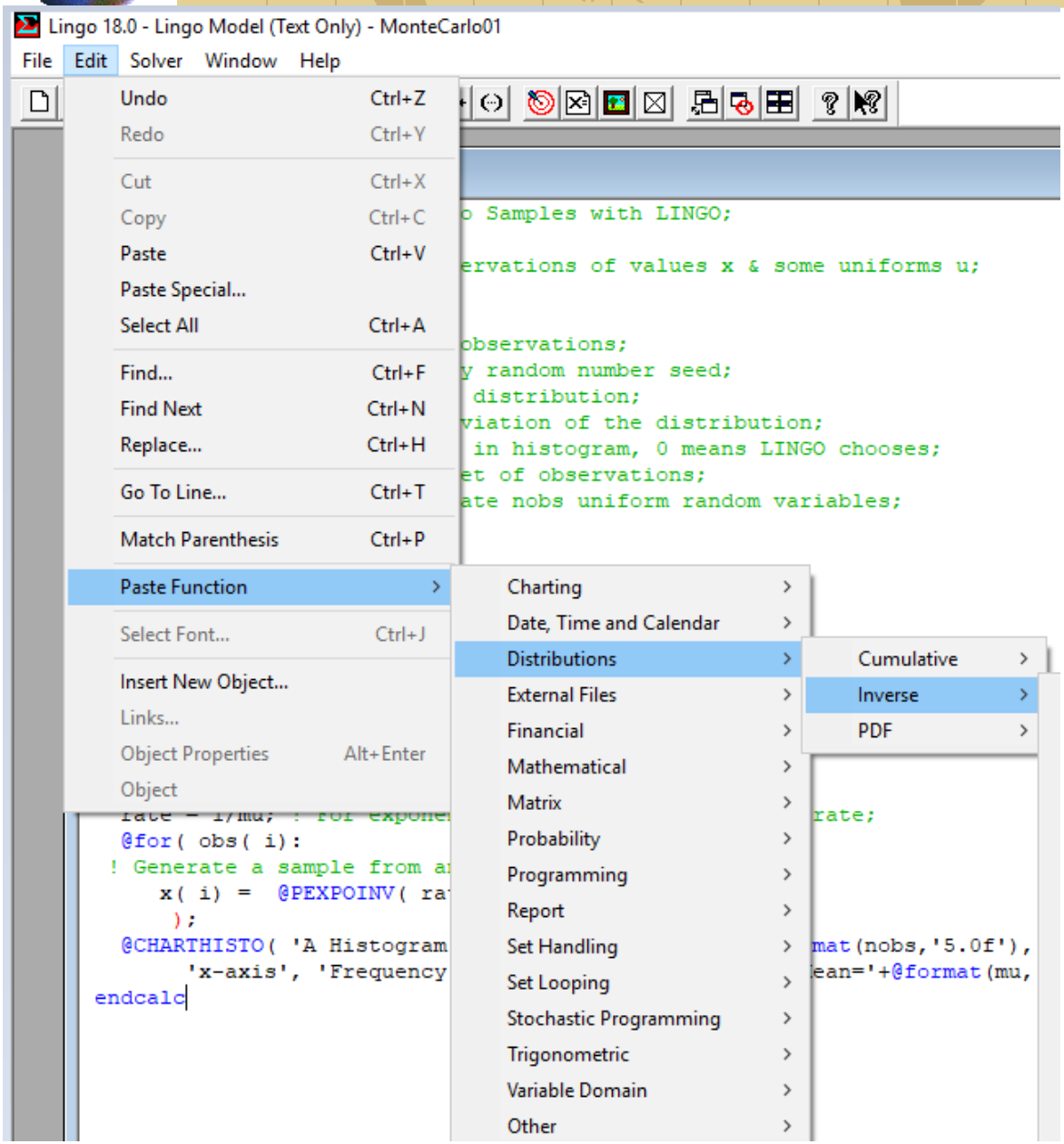
LINDO SYSTEMS INC.



The Code to do Sampling Is Not Long…

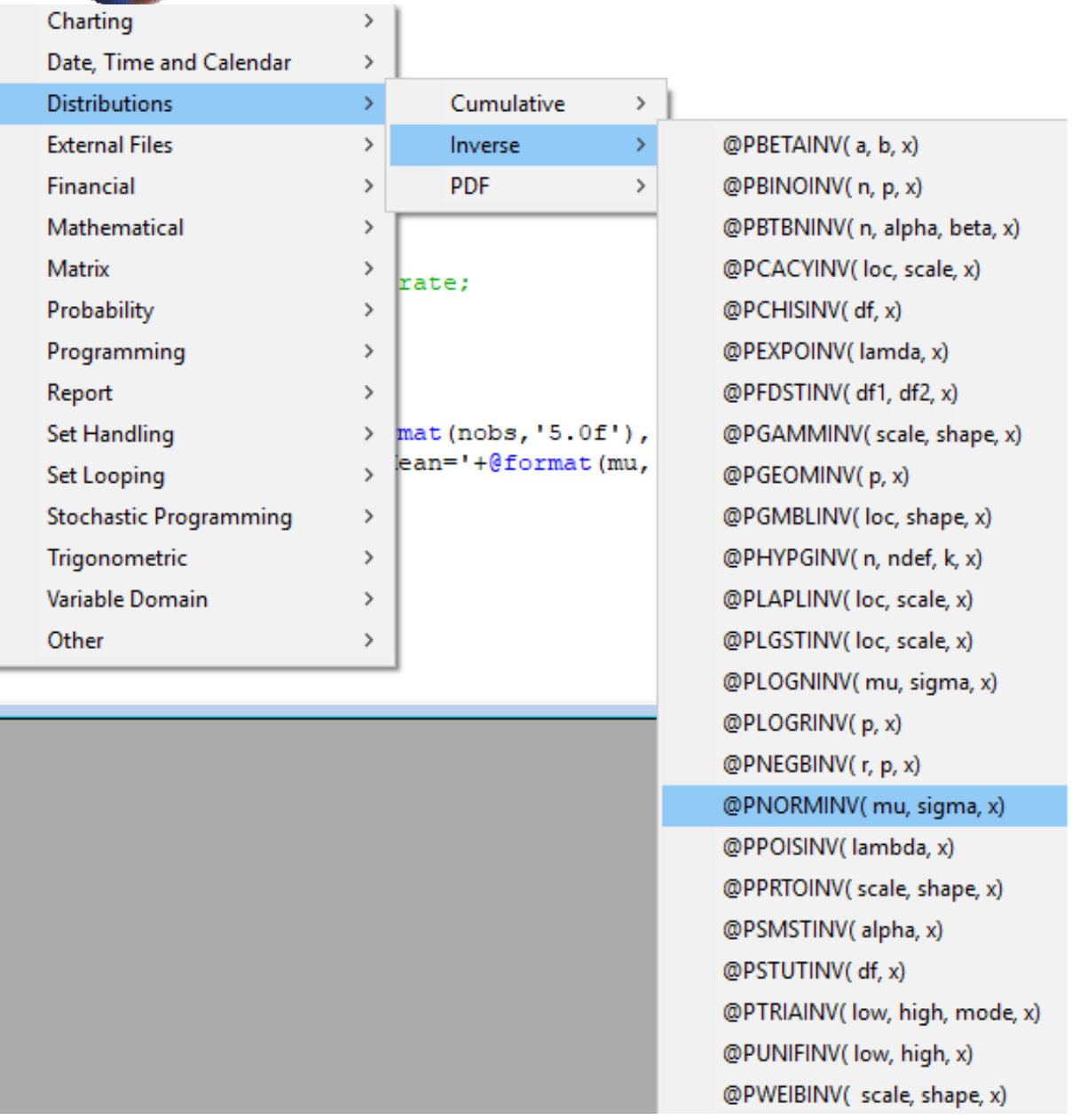
```
! Generating some Monte Carlo Samples with LINGO (MonteCarlo.lng);
sets:
 obs: x, u;   ! A set of observations of values x & some uniforms u;
endsets
data:
  nobs = 10000; ! Number of observations;
  seed = 34187; ! An arbitrary random number seed;
  mu = 10;      ! Mean of the distribution;
  sigma =  3;   ! Standard deviation of the distribution;
  nbins = 13;   ! Number bins in histogram, 0 means LINGO chooses;
  obs = 1..nobs;     ! The set of observations;
  u = @qrand( seed); ! Generate nobs uniform random variables;
enddata
calc:
  @for( obs( i):
 ! Generate sample from Normal distribution using Normal cdf inverse;
     x( i) =  @PNORMINV( mu, sigma, u( i));   );
 ! Display a histogram;
  @CHARTHISTO( 'A Histogram for a Sample of Size '+@format(nobs,'5.0f'),
       'x-axis', 'Frequency', 'Normal histogram, Mean='+@format(mu,'5.1f'), 0, x);

  rate = 1/mu; ! For exponential, the parameter is the rate;
  @for( obs( i):
 ! Generate a sample from an Exponential distribution;
     x( i) =  @PEXPOINV( rate, u( i));     );
  @CHARTHISTO( 'A Histogram for a Sample of Size '+@format(nobs,'5.0f'),
       'x-axis', 'Frequency', 'Exponential histogram,Mean='+@format(mu,'5.1f'), 0, x);
endcalc
```

It is easy to find the available distributions..



LINDO SYSTEMS INC.

There are many distributions available…

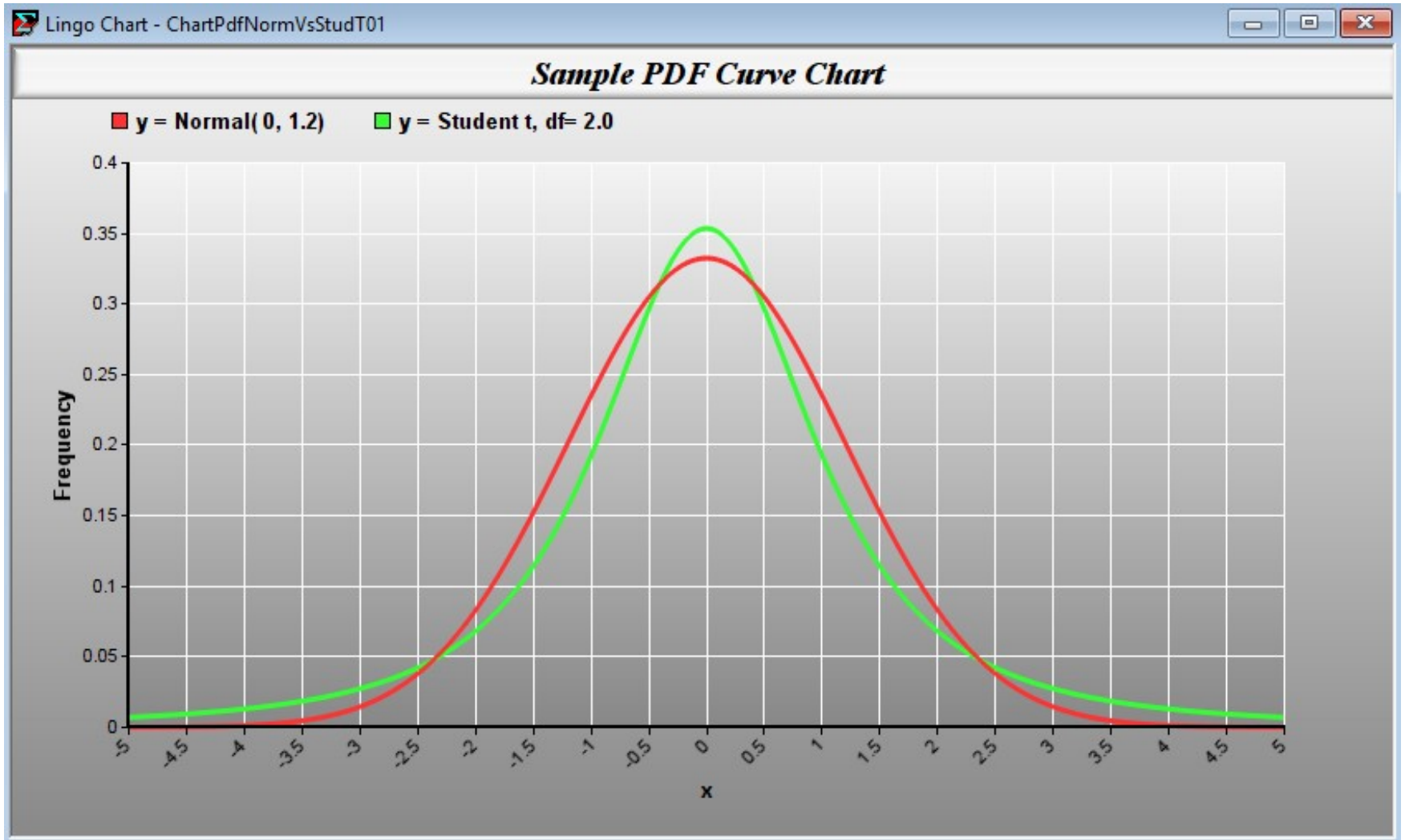


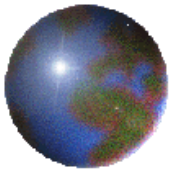
Beta,
Binomial, BetaBinomial,
Cauchy,
ChiSquare,
Exponential,
F,
Gamma,
Geometric,
Gumbel,
Hypergeometric,
Laplace,
Logistic,
LogNormal,
Logarithmic,
Negative binomial,
Normal,
Poisson,
Pareto,
Symmetric stable,
Student *t*,
Triangular,
Uniform, Weibull…

LINDO SYSTEMS INC.

You Can Also Graph Probability Distributions Directly

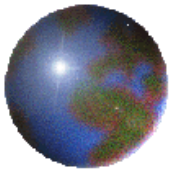
Notice longer tails of Student *t* distribution relative to Normal.





The LINGO Code is Simple…

```
! Graph Normal distribution vs. Student t
(ChartPdfNormVsStudT);
PROCEDURE PDFCOMPUTE:
  y1 = @PNORMPDF( 0, sigma, x);
  y2 = @PSTUTPDF( df, x);
ENDPROCEDURE
calc:
   sigma = 1.2;! Standard deviation for Normal distribution;
   df = 2;     ! Degrees of freedom for Student t distribution;
   LB = -5;    ! Lower bound for x;
   UB =  5;    ! Upper bound for x;
   @CHARTPCURVE( 'Sample PDF Curve Chart', 'x', 'Frequency',
       PDFCOMPUTE, x, LB, UB,
      'y = Normal( 0, '+@format( sigma,'3.1f')+')', y1,
      'y = Student t, df= '+@format( df,'3.1f'),    y2);
 endcalc
```



Multi-Variate Distributions

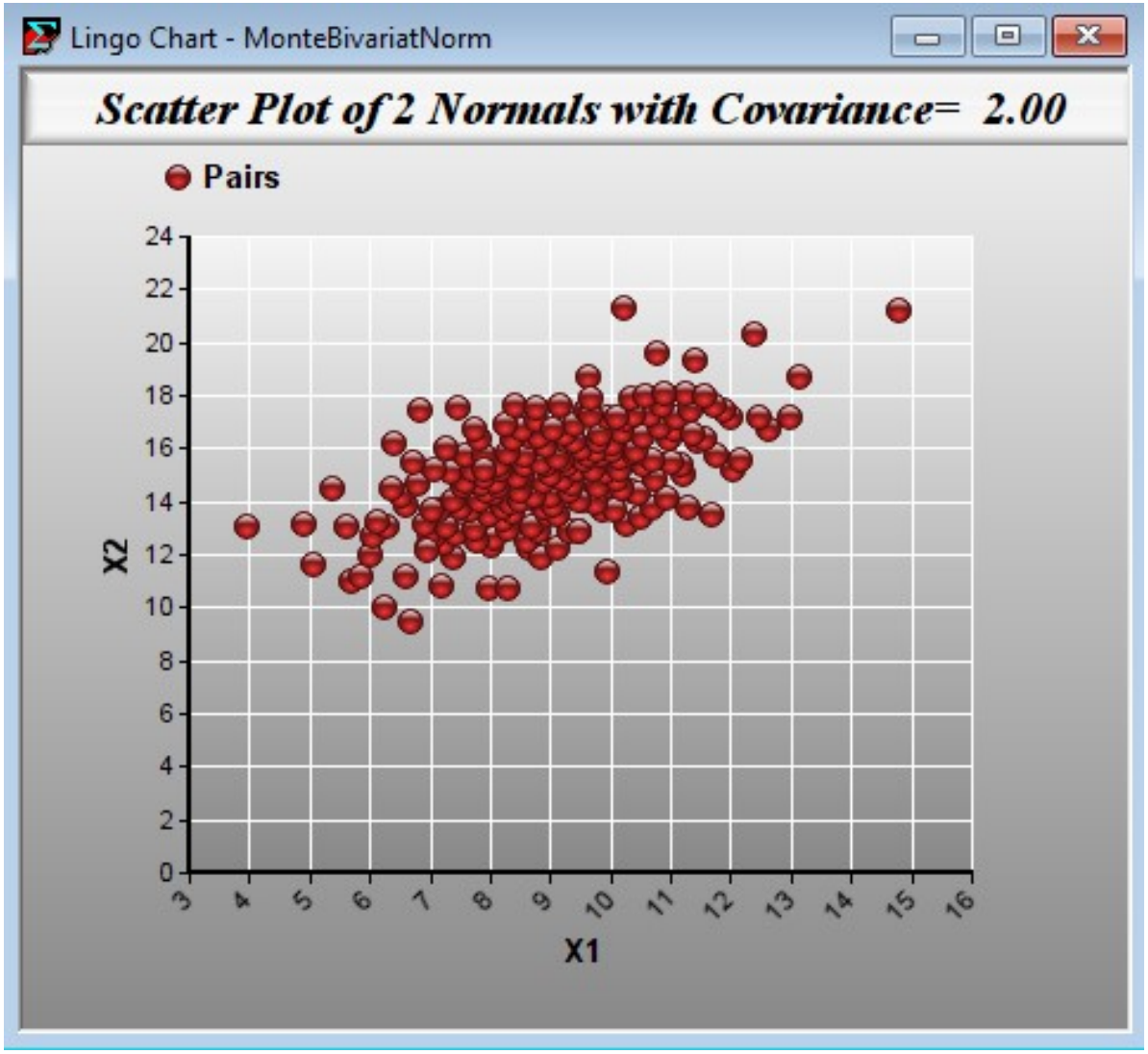
Suppose you want to generate
a sample from a multi-variate Normal distribution
with the following parameters.

```
mu = 9 15; ! Means of multi-variate Normal;
 covar =   ! Covariance matrix;
      3  2
      2  4;
```

Notice that the two variables are positively correlated.

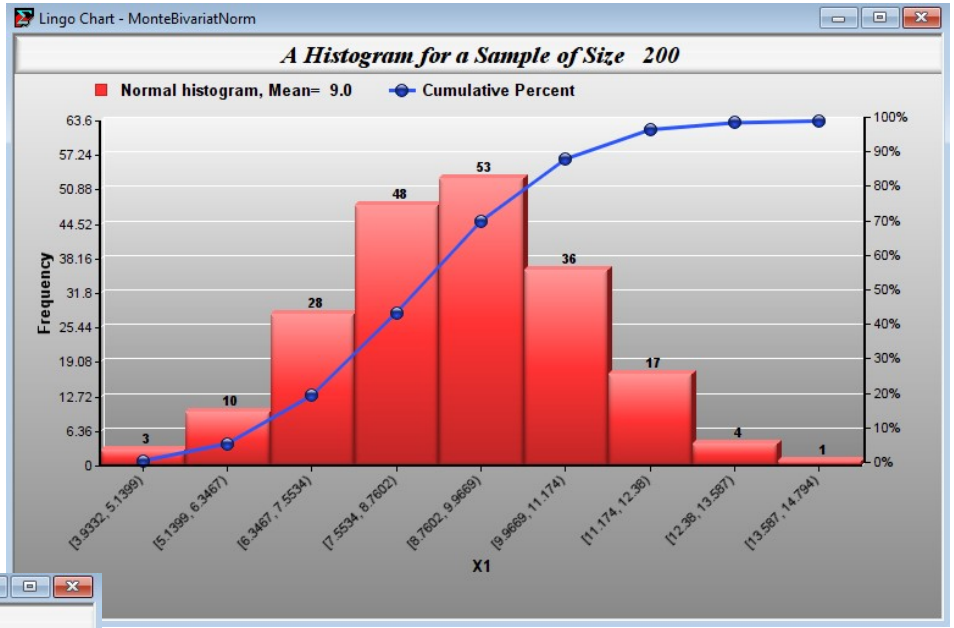
Scatter Plot

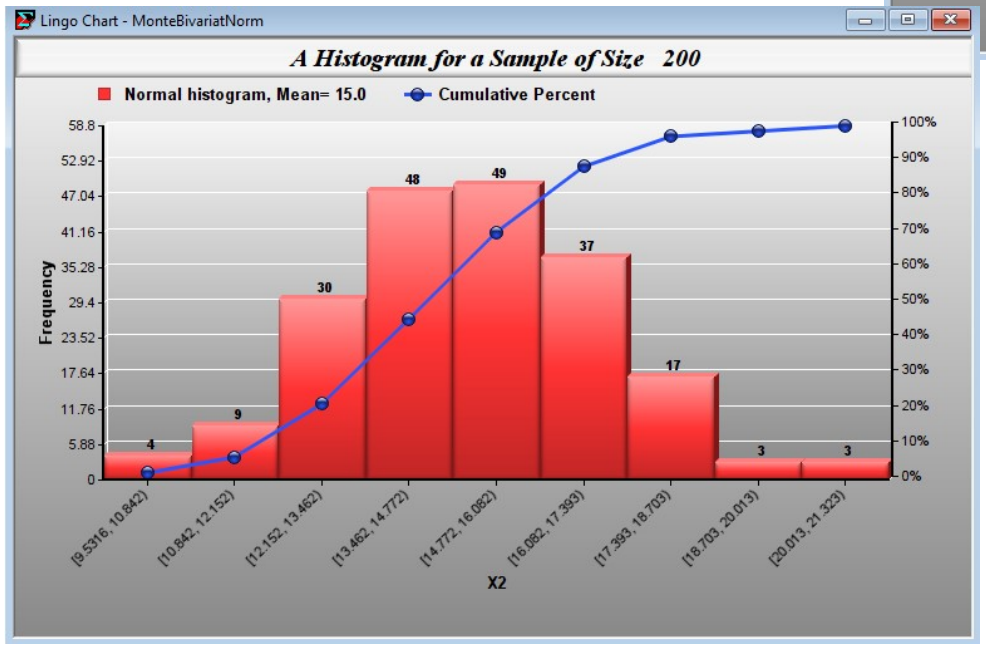
Notice the positive correlation in the scatter plot.



LINDO SYSTEMS INC.

The Marginal Distributions for X1 and X2 Look Normal





LINDO SYSTEMS INC.

Generating Multi-variate Normal Random Variables

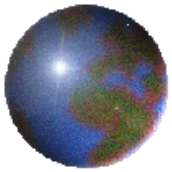
The most common way of generating multi-variate Normal random variables *X*, is to

1) compute the Cholesky factorization, loosely speaking, the matrix square root of the covariance matrix, call it *sigma*,

2) generate some Normal random variables, *Z*, with mean 0 and standard deviation 1 and then

3) use the matrix equivalent of

X = mu + sigma Z;*



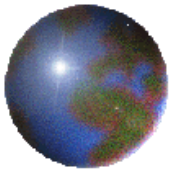
Cholesky Factorizaion/Matrix square root

```
Original Covariance Matrix
                1           2
  1   3.000000   2.000000
  2   2.000000   4.000000

Cholesky factorization( matrix square root)
                1           2
  1   1.732051   0.000000
  2   1.154701   1.632993
```

Notice that :

1.732051*1.732051 = 3, and

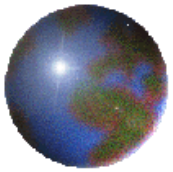
1.154701*1.154701 + 1.632993*1.632993 = 4



The Complete LINGO Code is:

! Declarations:

```
! Generating some Multi-variate Normal Samples with
LINGO (MonteBivariateNorm.lng);
sets:
 var: mu;
 vxv( var, var): covar, sigma;
 obs: x1, x2;
 oxv( obs, var): u, x; ! A set of observations of
values x & some uniforms u;
endsets
data:
  nobs = 200;  ! Number of observations;
  obs = 1..nobs;     ! The set of observations;
  seed = 32187;! An arbitrary random number seed;
  mu = 9 15;   ! Means of the multi-variate
distribution;
  covar =      ! Covariance matrix;
       3  2
       2  4;
  u = @qrand( seed); ! Generate nobs uniform random
variables;
enddata
```



The Computational part is:

```
calc:
 ! Compute the 'Square root'/Cholesky factor of covariancematrix;
    sigma = @CHOLESKY( covar);

 ! Generate some correlated Normal random variables.
   Loosely speaking:  x = mu + covar*z, where z are standard Normal;
 @for( obs( i):
   @for( var( j):
     x( i, j) = mu( j) + @sum( var( k): sigma( j, k)* @PNORMINV( 0, 1, u( i, k)));
      ); );

! Do a scatter plot of the variables 1 and 2;
 @for( obs( i):
   x1( i) = x( i, 1);
   x2( i) = x( i, 2);
     );
 @CHARTSCATTER( 'Scatter Plot of 2 Normals with Covariance= '+@format(covar(1,2),'5.2f'),
                'X1', 'X2', 'Pairs', x1, x2);
! Look at the histogram of each random variable;
 @CHARTHISTO( 'A Histogram for a Sample of Size '+@format(nobs,'5.0f'),
       'X1', 'Frequency', 'Normal histogram, Mean='+@format(mu(1),'5.1f'), 0, x1);
 @CHARTHISTO( 'A Histogram for a Sample of Size '+@format(nobs,'5.0f'),
       'X2', 'Frequency', 'Normal histogram, Mean='+@format(mu(2),'5.1f'), 0, x2);
endcalc
data:
    @text() = 'Original Covariance Matrix';
    @text() = @TABLE( covar);
    @text() = ' ';
    @text() = 'Cholesky factorization( matrix square root)';
    @text() = @TABLE( sigma);
enddata
```