drawnorm — Draw sample from multivariate normal distribution

Description

drawnorm draws a sample from a multivariate normal distribution with desired means and covariance matrix. The default is orthogonal data with mean 0 and variance 1. The covariance matrix may be singular. The values generated are a function of the current random-number seed or the number specified with set seed(); see [R] **set seed**.

Quick start

Generate independent variables x and y, where x has mean 2 and standard deviation 0.5 and y has mean 3 and standard deviation 1

```
drawnorm x y, means(2,3) sds(.5,1)
```

Same as above, but create dataset of 1,000 observations on x and y with means stored in vector m and standard deviations stored in vector sd

```
drawnorm x y, means(m) sds(sd) n(1000)
```

Same as above, and set the seed for the random-number generator to reproduce results

```
drawnorm x y, means(m) sds(sd) n(1000) seed(81625)
```

Sample from bivariate standard normal distribution with covariance between x and y of 0.5 stored in variance–covariance matrix C

```
matrix C = (1, .5 \ .5, 1)
drawnorm x y, cov(C)
```

Sample from a trivariate standard normal distribution with correlation between x and y of 0.4, x and z of 0.3, and y and z of 0.6 stored in correlation matrix C

```
matrix C = (1, .4, .3 \ .4, 1, .6 \ .3, .6, 1)
drawnorm x y z, corr(C)
```

Same as above, but avoid typing full matrix by specifying correlations in vector v treated as a lower triangular matrix

```
matrix v = (1, .4, 1, .3, .6, 1)
drawnorm x y z, corr(v) cstorage(lower)
```

Menu

Data > Create or change data > Other variable-creation commands > Draw sample from normal distribution

Syntax

drawnorm *newvarlist* [, *options*]

options	Description
Main	
clear	replace the current dataset
double	generate variable type as double; default is float
n(#)	generate # observations; default is current number
sds(*vector*)	standard deviations of generated variables
corr(*matrix* | *vector*)	correlation matrix
cov(*matrix* | *vector*)	covariance matrix
cstorage(full)	store correlation/covariance structure as a symmetric $k \times k$ matrix
cstorage(lower)	store correlation/covariance structure as a lower triangular matrix
cstorage(upper)	store correlation/covariance structure as an upper triangular matrix
forcepsd	force the covariance/correlation matrix to be positive semidefinite
means(*vector*)	means of generated variables; default is means(0)
Options	
seed(#)	seed for random-number generator

Options

Main

clear specifies that the dataset in memory be replaced, even though the current dataset has not been saved on disk.

double specifies that the new variables be stored as Stata doubles, meaning 8-byte reals. If double is not specified, variables are stored as floats, meaning 4-byte reals. See [D] **Data types**.

n(#) specifies the number of observations to be generated. The default is the current number of observations. If n(#) is not specified or is the same as the current number of observations, drawnorm adds the new variables to the existing dataset; otherwise, drawnorm replaces the data in memory.

sds(*vector*) specifies the standard deviations of the generated variables. sds() may not be specified with cov().

corr(*matrix* | *vector*) specifies the correlation matrix. If neither corr() nor cov() is specified, the default is orthogonal data.

cov(*matrix* | *vector*) specifies the covariance matrix. If neither cov() nor corr() is specified, the default is orthogonal data.

cstorage(full | lower | upper) specifies the storage mode for the correlation or covariance structure in corr() or cov(). The following storage modes are supported:

full specifies that the correlation or covariance structure is stored (recorded) as a symmetric $k \times k$ matrix.

`lower` specifies that the correlation or covariance structure is recorded as a lower triangular matrix. With k variables, the matrix should have $k(k+1)/2$ elements in the following order:

$$\mathrm{C}_{11}\ \mathrm{C}_{21}\ \mathrm{C}_{22}\ \mathrm{C}_{31}\ \mathrm{C}_{32}\ \mathrm{C}_{33}\ \ldots\ \mathrm{C}_{k1}\ \mathrm{C}_{k2}\ \ldots\ \mathrm{C}_{kk}$$

`upper` specifies that the correlation or covariance structure is recorded as an upper triangular matrix. With k variables, the matrix should have $k(k+1)/2$ elements in the following order:

$$\mathrm{C}_{11}\ \mathrm{C}_{12}\ \mathrm{C}_{13}\ \ldots\ \mathrm{C}_{1k}\ \mathrm{C}_{22}\ \mathrm{C}_{23}\ \ldots \mathrm{C}_{2k}\ \ldots\ \mathrm{C}_{(k-1k-1)}\ \mathrm{C}_{(k-1k)}\ \mathrm{C}_{kk}$$

Specifying `cstorage(full)` is optional if the matrix is square. `cstorage(lower)` or `cstorage(upper)` is required for the vectorized storage methods. See *Example 2: Storage modes for correlation and covariance matrices*.

`forcepsd` modifies the matrix *C* to be positive semidefinite (psd), and so be a proper covariance matrix. If *C* is not positive semidefinite, it will have negative eigenvalues. By setting negative eigenvalues to 0 and reconstructing, we obtain the least-squares positive-semidefinite approximation to *C*. This approximation is a singular covariance matrix.

`means(`*vector*`)` specifies the means of the generated variables. The default is `means(0)`.

Options

`seed(#)` specifies the initial value of the random-number seed used by the `runiform()` function. The default is the current random-number seed. Specifying `seed(#)` is the same as typing `set seed #` before issuing the `drawnorm` command.

Remarks and examples

▷ Example 1

Suppose that we want to draw a sample of 1,000 observations from a normal distribution $N(\mathbf{M},\mathbf{V})$, where **M** is the mean matrix and **V** is the covariance matrix:

```
. matrix M = 5, -6, 0.5
. matrix V = (9, 5, 2 \ 5 , 4 , 1 \ 2, 1, 1)
. matrix list M
M[1,3]
    c1  c2  c3
r1   5  -6  .5
. matrix list V
symmetric V[3,3]
    c1  c2  c3
r1   9
r2   5   4
r3   2   1   1
. drawnorm x y z, n(1000) cov(V) means(M)
(obs 1,000)
```

```
. summarize

    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
           x |      1,000      5.0424    3.061953  -5.065592   15.96129
           y |      1,000   -5.914462    2.012488  -12.25234   .3326397
           z |      1,000    .5181909    1.017397   -2.59316   3.884182

. correlate, cov
(obs=1,000)

             |        x        y        z
-------------+---------------------------
           x |  9.37556
           y |  5.14201  4.05011
           z |  2.17972  1.07222   1.0351
```

◁

❑ Technical note

The values generated by `drawnorm` are a function of the current random-number seed. To reproduce the same dataset each time `drawnorm` is run with the same setup, specify the same seed number in the `seed()` option.

❑

▷ Example 2: Storage modes for correlation and covariance matrices

The three storage modes for specifying the correlation or covariance matrix in `corr2data` and `drawnorm` can be illustrated with a correlation structure, `C`, of 4 variables. In full storage mode, this structure can be entered as a 4×4 Stata matrix:

```
. matrix C = ( 1.0000,  0.3232,  0.1112,  0.0066 \ ///
               0.3232,  1.0000,  0.6608, -0.1572 \ ///
               0.1112,  0.6608,  1.0000, -0.1480 \ ///
               0.0066, -0.1572, -0.1480,  1.0000 )
```

Elements within a row are separated by commas, and rows are separated by a backslash, \. We use the input continuation operator `///` for convenient multiline input; see [P] **comments**. In this storage mode, we probably want to set the row and column names to the variable names:

```
.  matrix rownames C = price trunk headroom rep78
.  matrix colnames C = price trunk headroom rep78
```

This correlation structure can be entered more conveniently in one of the two vectorized storage modes. In these modes, we enter the lower triangle or the upper triangle of `C` in rowwise order; these two storage modes differ only in the order in which the $k(k+1)/2$ matrix elements are recorded. The lower storage mode for C comprises a vector with $4(4+1)/2 = 10$ elements, that is, a 1×10 or 10×1 Stata matrix, with one row or column,

```
.  matrix C = ( 1.0000,  ///
                0.3232,  1.0000,  ///
                0.1112,  0.6608,  1.0000,  ///
                0.0066, -0.1572, -0.1480,  1.0000)
```

or more compactly as

```
.  matrix C = ( 1, 0.3232, 1, 0.1112, 0.6608, 1, 0.0066, -0.1572, -0.1480, 1 )
```

C may also be entered in upper storage mode as a vector with $4(4+1)/2 = 10$ elements, that is, a 1×10 or 10×1 Stata matrix,

```
. matrix C = ( 1.0000,  0.3232,  0.1112,  0.0066, ///
                        1.0000,  0.6608, -0.1572, ///
                                 1.0000, -0.1480, ///
                                          1.0000 )
```

or more compactly as

```
. matrix C = ( 1, 0.3232, 0.1112, 0.0066, 1, 0.6608, -0.1572, 1, -0.1480, 1 )
```

◁

Methods and formulas

Results are asymptotic. The more observations generated, the closer the correlation matrix of the dataset is to the desired correlation structure.

Let $\mathbf{V} = \mathbf{A}'\mathbf{A}$ be the desired covariance matrix and $\mathbf{M}$ be the desired mean matrix. We first generate $\mathbf{X}$, such that $\mathbf{X} \sim N(\mathbf{0}, \mathbf{I})$. Let $\mathbf{Y} = \mathbf{A}'\mathbf{X} + \mathbf{M}$, then $\mathbf{Y} \sim N(\mathbf{M}, \mathbf{V})$.

References

Canette, I. 2013. Fitting ordered probit models with endogenous covariates with Stata's gsem command. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2013/11/07/fitting-ordered-probit-models-with-endogenous-covariates-with-statas-gsem-command/.

Chen, M. 2015. Generating nonnegatively correlated binary random variates. *Stata Journal* 15: 301–308.

Gould, W. W. 2012a. Using Stata's random-number generators, part 2: Drawing without replacement. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2012/08/03/using-statas-random-number-generators-part-2-drawing-without-replacement/.

———. 2012b. Using Stata's random-number generators, part 3: Drawing with replacement. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2012/08/29/using-statas-random-number-generators-part-3-drawing-with-replacement/.

Huber, C. 2014. How to simulate multilevel/longitudinal data. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2014/07/18/how-to-simulate-multilevellongitudinal-data/.

Lee, S. 2015. Generating univariate and multivariate nonnormal data. *Stata Journal* 15: 95–109.

Lindsey, C. 2015a. Probit model with sample selection by mlexp. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2015/10/22/probit-model-with-sample-selection-by-mlexp/.

———. 2015b. Using mlexp to estimate endogenous treatment effects in a probit model. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2015/11/05/using-mlexp-to-estimate-endogenous-treatment-effects-in-a-probit-model/.

Also see

[D] **corr2data** — Create dataset with specified correlation structure

[R] **set seed** — Specify random-number seed and state