Title

runiform() — Uniform and nonuniform pseudorandom variates

Syntax

real matrix `runiform(`*real scalar r*, *real scalar c*`)`

string scalar `rseed()`

void `rseed(`*string scalar newseed*`)`

void `rseed(`*real scalar newseed*`)`

real matrix `rbeta(`*real scalar r*, *real scalar c*, *real matrix a*, *real matrix b*`)`

real matrix `rbinomial(`*real scalar r*, *real scalar c*, *real matrix n*, *real matrix p*`)`

real matrix `rchi2(`*real scalar r*, *real scalar c*, *real matrix df*`)`

real matrix `rdiscrete(`*real scalar r*, *real scalar c*, *real colvector p*`)`

real matrix `rgamma(`*real scalar r*, *real scalar c*, *real matrix a*, *real matrix b*`)`

real matrix `rhypergeometric(`*real scalar r*, *real scalar c*, *real matrix N*, *real matrix K*, *real matrix n*`)`

real matrix `rnbinomial(`*real scalar r*, *real scalar c*, *real matrix n*, *real matrix p*`)`

real matrix `rnormal(`*real scalar r*, *real scalar c*, *real matrix m*, *real matrix s*`)`

real matrix `rpoisson(`*real scalar r*, *real scalar c*, *real matrix m*`)`

real matrix `rt(`*real scalar r*, *real scalar c*, *real matrix df*`)`

Description

`runiform(`*r*, *c*`)` returns an $r \times c$ real matrix containing uniformly distributed random variates on [0,1). `runiform()` is the same function as Stata's `runiform()` function.

`rseed()` returns the current random-variate seed in an encrypted string form. `rseed()` returns the same thing as Stata's `c(seed)`; see [R] **set seed** and [P] **creturn**.

rseed(*newseed*) sets the seed: a string previously obtained from rseed() can be specified for the argument or an integer number can be specified. rseed() has the same effect as Stata's set seed command; see [R] **set seed**.

rbeta(*r*, *c*, *a*, *b*) returns an $ir \times jc$ real matrix containing beta random variates. The real-valued matrices *a* and *b* contain the beta shape parameters. The matrices *a* and *b* must be r-conformable, where $i =$ max(rows(*a*),rows(*b*)) and $j =$ max(cols(*a*),cols(*b*)).

rbinomial(*r*, *c*, *n*, *p*) returns an $ir \times jc$ real matrix containing binomial random variates. The real-valued matrices *n* and *p* contain the number of trials and the probability parameters. The matrices *n* and *p* must be r-conformable, where $i =$ max(rows(*n*),rows(*p*)) and $j =$ max(cols(*n*),cols(*p*)).

rchi2(*r*, *c*, *df*) returns an $ir \times jc$ real matrix containing chi-squared random variates. The real-valued matrix *df* contains the degrees of freedom parameters, where $i =$ rows(*df*) and $j =$ cols(*df*).

rdiscrete(*r*, *c*, *p*) returns an $r \times c$ real matrix containing random variates from the discrete distribution specified by the probabilities in the vector *p* of length *k*. The range of the discrete variates is 1, 2, . . . , *k*. The alias method of Walker (1977) is used to sample from the discrete distribution.

rgamma(*r*, *c*, *a*, *b*) returns an $ir \times jc$ real matrix containing gamma random variates. The real-valued matrices *a* and *b* contain the gamma shape and scale parameters, respectively. The matrices *a* and *b* must be r-conformable, where $i =$ max(rows(*a*),rows(*b*)) and $j =$ max(cols(*a*),cols(*b*)).

rhypergeometric(*r*, *c*, *N*, *K*, *n*) returns an $ir \times jc$ real matrix containing hypergeometric random variates. The integer-valued matrix *N* contains the population sizes, the integer-valued matrix *K* contains the number of elements in each population that have the attribute of interest, and the integer-valued matrix *n* contains the sample size. The matrices *N*, *K*, and *n* must be r-conformable, where $i =$ max(rows(*N*),rows(*K*),rows(*n*)) and $j =$ max(cols(*N*),cols(*K*),cols(*n*)).

rnbinomial(*r*, *c*, *n*, *p*) returns an $ir \times jc$ real matrix containing negative binomial random variates. When the elements of the matrix *n* are integer-valued, rnbinomial() returns the number of failures before the *n*th success, where the probability of success on a single draw is contained in the real-valued matrix *p*. The elements of *n* can also be nonintegral but must be positive. The matrices *n* and *p* must be r-conformable, where $i =$ max(rows(*n*),rows(*p*)) and $j =$ max(cols(*n*),cols(*p*)).

rnormal(*r*, *c*, *m*, *s*) returns an $ir \times jc$ real matrix containing normal (Gaussian) random variates. The real-valued matrices *m* and *s* contain the mean and standard deviation parameters, respectively. The matrices *m* and *s* must be r-conformable, where $i =$ max(rows(*m*),rows(*s*)) and $j =$ max(cols(*m*),cols(*s*)).

rpoisson(*r*, *c*, *m*) returns an $ir \times jc$ real matrix containing Poisson random variates. The real-valued matrix *m* contains the Poisson mean parameters, where $i =$ rows(*m*) and $j =$ cols(*m*).

rt(*r*, *c*, *df*) returns an $ir \times jc$ real matrix containing Student's t random variates. The real-valued matrix *df* contains the degrees-of-freedom parameters, where $i =$ rows(*df*) and $j =$ cols(*df*).

Remarks and examples stata.com

The functions described here generate random variates. The parameter limits for each generator are the same as those documented for Stata's random-number functions, except for rdiscrete(), which has no Stata equivalent.

In the example below, we generate and summarize 1,000 random normal deviates with a mean of 3 and standard deviation of 1.

```
: rseed(13579)
: x = rnormal(1000, 1, 3, 1)
: meanvariance(x)
                 1
    +---------------+
  1 |  2.942981879  |
  2 |  1.061992176  |
    +---------------+
```

The next example uses a 1×3 vector of gamma shape parameters to generate a 1000×3 matrix of gamma random variates, X.

```
: a = (0.5,1.5,2.5)
: rseed(13579)
: X = rgamma(1000,1,a,1)
: mean(X)
                 1              2              3
    +----------------------------------------------+
  1 |  .5339343609    1.510028772    2.451447187   |
    +----------------------------------------------+

: diagonal(variance(X))'
                 1              2              3
    +----------------------------------------------+
  1 |  .6129729256    1.669457192    2.284915684   |
    +----------------------------------------------+
```

The first column of X contains gamma variates with shape parameter 0.5, the second column contains gamma variates with shape parameter 1.5, and the third column contains gamma variates with shape parameter 2.5.

Below we generate a 4×3 matrix of beta variates where we demonstrate the use of two r-conformable parameter matrices, a and b.

```
: a = (0.5,1.5,2.5)
: b = (0.5,0.75,1.0\1.25,1.5,1.75)
: rseed(13579)
: rbeta(2,1,a,b)
                 1              2              3
    +----------------------------------------------+
  1 |   .668359305   .3238859912    .7785175363    |
  2 |   .266459731   .7665943496     .634730294    |
  3 |  .0373430126   .9246702534     .851879254    |
  4 |  .1903514438   .5012842811    .8759050005    |
    +----------------------------------------------+
```

The 4×3 shape-parameter matrices used to generate these beta variates are given below:

```
: J(2,1,J(rows(b),1,a))
          1      2      3
    +--------------------+
  1 |    .5    1.5    2.5  |
  2 |    .5    1.5    2.5  |
  3 |    .5    1.5    2.5  |
  4 |    .5    1.5    2.5  |
    +--------------------+
```

```
: J(2,1,b)
          1      2      3
    ┌──────────────────────┐
  1 │    .5    .75      1  │
  2 │  1.25    1.5   1.75  │
  3 │    .5    .75      1  │
  4 │  1.25    1.5   1.75  │
    └──────────────────────┘
```

Conformability

`runiform(`r, c`)`:

r:	1×1
c:	1×1
result:	$r \times c$

`rseed()`:

result:	1×1

`rseed(`*newseed*`)`:

newseed:	1×1
result:	*void*

`rbeta(`r, c, a, b`)`:

r:	1×1
c:	1×1
a:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
b:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
result:	$r \times c$ or $ir \times c$ or $r \times jc$ or $ir \times jc$

`rbinomial(`r, c, n, p`)`:

r:	1×1
c:	1×1
n:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
p:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
result:	$r \times c$ or $ir \times c$ or $r \times jc$ or $ir \times jc$

`rchi2(`r, c, *df*`)`:

r:	1×1
c:	1×1
df:	$i \times j$
result:	$ir \times jc$

`rdiscrete(`r,c,p`)`:

r:	1×1
c:	1×1
p:	$k \times 1$
result:	$r \times c$

`rgamma(`r`, `c`, `a`, `b`):`

r:	1×1
c:	1×1
a:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
b:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
result:	$r \times c$ or $ir \times c$ or $r \times jc$ or $ir \times jc$

`rhypergeometric(`r`, `c`, `N`, `K`, `n`):`

r:	1×1
c:	1×1
N:	1×1
K:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
n:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
result:	$r \times c$ or $ir \times c$ or $r \times jc$ or $ir \times jc$

`rnbinomial(`r`, `c`, `n`, `p`):`

r:	1×1
c:	1×1
n:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
p:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
result:	$r \times c$ or $ir \times c$ or $r \times jc$ or $ir \times jc$

`rnormal(`r`, `c`, `m`, `s`):`

r:	1×1
c:	1×1
m:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
s:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
result:	$r \times c$ or $ir \times c$ or $r \times jc$ or $ir \times jc$

`rpoisson(`r`, `c`, `m`):`

r:	1×1
c:	1×1
m:	$i \times j$
result:	$ir \times jc$

`rt(`r`, `c`, `*df*`):`

r:	1×1
c:	1×1
df:	1×1 or $i \times 1$ or $1 \times j$ or $i \times j$
result:	$r \times c$ or $ir \times c$ or $r \times jc$ or $ir \times jc$

Diagnostics

All random-variate generators abort with an error if $r < 0$ or $c < 0$.

`rseed(`*seed*`)` aborts with error if a string seed is specified and it is malformed (was not obtained from `rseed()`).

`rnormal(`r`, `c`, `m`, `s`)`, `rbeta(`r`, `c`, `a`, `b`)`, `rbinomial(`r`, `c`, `n`, `p`)`, `rhypergeometric(`r`, `c`, `N`, `K`, `n`)`, and `rnbinomial(`r`, `c`, `k`, `p`)` abort with an error if the parameter matrices do not conform. See *r-conformability* in [M-6] **Glossary** for rules on matrix conformability.

`rdiscrete()` aborts with error if the probabilities in p are not in [0,1] or do not sum to 1.

References

Gould, W. W. 2012a. Using Stata's random-number generators, part 1. The Stata Blog: Not Elsewhere Classified. http://blog.stata.com/2012/07/18/using-statas-random-number-generators-part-1/.

———. 2012b. Using Stata's random-number generators, part 2: Drawing without replacement. The Stata Blog: Not Elsewhere Classified. http://blog.stata.com/2012/08/03/using-statas-random-number-generators-part-2-drawing-without-replacement/.

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

———. 2012d. Using Stata's random-number generators, part 4: Details. The Stata Blog: Not Elsewhere Classified. http://blog.stata.com/2012/10/24/using-statas-random-number-generators-part-4-details/.

Walker, A. J. 1977. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software* 3: 253–256.

Also see

[M-4] **standard** — Functions to create standard matrices

[M-4] **statistical** — Statistical functions