Package 'Runuran'

October 22, 2021

Type Package

Title R Interface to the 'UNU.RAN' Random Variate Generators

Version 0.35

Date 2021-10-22

Depends R (>= 3.0.0)

Imports methods, stats

Suggests testthat (>= 2.0.0)

Description
Interface to the 'UNU.RAN' library for Universal Non-Uniform RANdom variate generators.
Thus it allows to build non-uniform random number generators from quite arbitrary
distributions. In particular, it provides an algorithm for fast numerical inversion
for distribution with given density function.
In addition, the package contains densities, distribution functions and quantiles
from a couple of distributions.

Collate unuran_distr.R unuran_cont.R unuran_discr.R unuran_cmv.R
Runuran.R universal.R distributions.R deprecated.R options.R
utils.R zzz.R

License GPL (>= 2)

URL https://statmath.wu.ac.at/unuran/

RoxygenNote 7.1.1

NeedsCompilation yes

Author Josef Leydold [aut, cre],
Wolfgang H"ormann [aut]

Maintainer Josef Leydold <josef.leydold@wu.ac.at>

Repository CRAN

Date/Publication 2021-10-22 18:50:02 UTC

R topics documented:

`Runuran-package` *Runuran – R interface to Universal Non-Uniform RANdom variate generators library*

Description

R interface to the UNU.RAN library for Universal Non-Uniform RANdom variate generators

Details

Package:	Runuran
Type:	Package
Version:	0.35
Date:	2021-10-22
License:	GPL 2 or later

Runuran provides an interface to the UNU.RAN library for universal non-uniform random number generators. It provides a collection of so called automatic methods for non-uniform random variate generation. Thus it is possible to draw samples from uncommon distributions. Nevertheless, (some of) these algorithms are also well suited for standard distribution like the normal distribution. Moreover, sampling from distributions like the generalized hyperbolic distribution is very fast. Such distributions became recently popular in financial engineering.

Runuran compiles four sets of functions of increasing power (and thus complexity):

[Special Generator] – Generators for particular distributions. Their syntax is similar to the corresponding R built-in functions.

[Universal] – Functions that offer an interface to a carefully selected collection of UNU.RAN methods with their most important parameters.

[Distribution] – Functions that create objects for important distributions. These objects can then be used in combination with one of the universal methods which is best suited for a particular application.

[Advanced] – Wrapper to the UNU.RAN string API. This gives access to all UNU.RAN methods and their variants.

We have marked all functions in their corresponding help page by one these four tags.

An introduction to **Runuran** with examples together with a very short survey on non-uniform random variate generation can be found in the package vignette (which can be displayed using `vignette("Runuran")`).

[Special Generator]

These functions have similar syntax to the analogous R built-in generating functions (if these exist) but have optional domain arguments `lb` and `ub`, i.e., these calls also allow to draw samples from truncated distributions:

`ur...(n,distribution parameters,lb ,ub)`

Compared to the corresponding R functions these `ur...` functions have a different behavior:

- `ur...` functions are often much faster for large samples (e.g., a factor of about 5 for the t distribution). For small samples they are slow.
- All `ur...` functions allow to sample from truncated versions of the original distributions. Therefore the arguments `lb` (lower border) and `ub` (upper border) are available for all these functions.
- Almost all `ur...` functions are based on fast numerical inversion algorithms. This is important for example for generating order statistics, quasi-Monte Carlo methods or random vectors from copulas.
- All `ur...` functions do **not** allow vectors as arguments (to be more precise: they only use the first element of the vector).

However, we recommend to use the more flexible approach described in the next sections below.

A list of all available special generators can be found in `Runuran.special.generators`.

[Universal]

These functions allow access to a selected collection of UNU.RAN methods. They require some data about the target distribution as arguments and return an instance of a UNU.RAN generator object that is implemented as an S4 class `unuran`. These can then be used to draw samples from the desired distribution by means of function `ur`. Methods that implement an *inversion* method can also be used for quantile function `uq`.

Currently the following methods are available by such functions.

Continuous Univariate Distributions:

Function		*Method*
`arou.new`	...	Automatic Ratio-of-Uniforms method
`ars.new`	...	Adaptive Rejection Sampling
`itdr.new`	...	Inverse Transformed Density Rejection
`pinv.new`	...	Polynomial interpolation of INVerse CDF
`srou.new`	...	Simple Ratio-Of-Uniforms method
`tabl.new`	...	TABLe based rejection
`tdr.new`	...	Transformed Density Rejection

Discrete Distributions:

Function		*Method*
`dari.new`	...	Discrete Automatic Rejection Inversion
`dau.new`	...	Alias-Urn Method

dgt.new	...	Guide-Table Method for discrete inversion

Multivariate Distributions:

Function		*Method*
hitro.new	...	Hit-and-Run with Ratio-of-Uniforms method
vnrou.new	...	Multivariate Naive Ratio-Of-Uniforms method

[Distribution]

Coding the required functions for particular distributions can be tedious. Thus we have compiled a set of functions that create UNU.RAN distribution objects that can directly be used with the functions from section [Universal].

A list of all available distributions can be found in Runuran.distributions.

[Advanced]

This interface provides the most flexible access to UNU.RAN. It requires three steps:

1. Create a unuran.distr object that contains all required information about the *target distribution*. We have three types of distributions:

Function		*Type of distribution*
unuran.cont.new	...	continuous distributions
unuran.discr.new	...	discrete distributions
unuran.cmv.new	...	multivariate continuous distributions

   The functions from section [Distribution] creates such objects for particular distributions.

2. Choose a *generation method* and create a unuran object using function unuran.new. This function takes two argument: the distribution object created in Step 1, and a string that contains the chosen UNU.RAN method and (optional) some parameters to adjust this method to the given target distribution. We refer to the UNU.RAN for more details on this "method string".
3. Use this object to *draw samples* from the target distribution using ur or uq.

Function		
ur	...	draw sample
uq	...	compute quantile (inverse CDF)
unuran.details	...	show unuran object

Density and distribution function

UNU.RAN distribution objects and generator objects may also be used to compute density and distribution function for a given distribution by means of ud and up.

Uniform random numbers

All UNU.RAN methods use the R built-in random number generator as source of (pseudo-) random numbers. Thus the generated samples depend on the state `.Random.seed` and can be controlled by the R functions `RNGkind` and `set.seed`.

Warning

`unuran` objects cannot be saved and restored in later R sessions, nor is it possible to copy such objects to different nodes in a computer cluster.

However, `unuran` objects for *some* generation methods can be "packed", see `unuran.packed`. Then these objects can be handled like any other R object (and thus saved and restored).

All other objects **must** be **newly created** in a new R session! (Using a restored object does not work as the `"unuran"` object is then broken.)

Note

The interface has been changed compared to the DSC 2003 paper.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

J. Leydold and W. H\"ormann (2000-2008): UNU.RAN User Manual, see https://statmath.wu.ac.at/unuran/.

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

G. Tirler and J. Leydold (2003): Automatic Nonuniform Random Variate Generation in R. In: K. Hornik and F. Leisch, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20–22, Vienna, Austria.

See Also

All objects are implemented as respective S4 classes `unuran`, `unuran.distr`, `unuran.cont`, `unuran.discr`, `unuran`.

See `Runuran.special.generators` for an overview of special generators and `Runuran.distributions` for a list of ready-to-use distributions suitable for the automatic methods.

`arou.new` *UNU.RAN generator based on Automatic Ratio-Of-Uniforms method (AROU)*

Description

UNU.RAN random variate generator for continuous distributions with given probability density function (PDF). It is based on the Automatic Ratio-Of-Uniforms method ('AROU').

[Universal] – Rejection Method.

Usage

```
arou.new(pdf, dpdf=NULL, lb, ub, islog=FALSE, ...)
aroud.new(distr)
```

Arguments

`pdf`	probability density function. (R function)
`dpdf`	derivative of `pdf`. (R function)
`lb`	lower bound of domain; use `-Inf` if unbounded from left. (numeric)
`ub`	upper bound of domain; use `Inf` if unbounded from right. (numeric)
`islog`	whether `pdf` is given as log-density (the `dpdf` must then be the derivative of the log-density). (boolean)
`...`	(optional) arguments for `pdf`.
`distr`	distribution object. (S4 object of class `"unuran.cont"`)

Details

This function creates an `unuran` object based on 'AROU' (Automatic Ratio-Of-Uniforms method). It can be used to draw samples of a continuous random variate with given probability density function using `ur`.

The density `pdf` must be positive but need not be normalized (i.e., it can be any multiple of a density function). The derivative `dpdf` of the (log-) density is optional. If omitted, numerical differentiation is used. Notice, however, that this might cause some round-off errors such that the algorithm fails. This is in particular the case when the density function is provided instead of the log-density.

The given `pdf` must be $T_{-0.5}$-concave (with implies unimodal densities with tails not higher than $(1/x^2)$; this includes all log-concave distributions).

It is recommended to use the log-density (instead of the density function) as this is numerically more stable.

Alternatively, one can use function `aroud.new` where the object `distr` of class `"unuran.cont"` must contain all required information about the distribution.

The setup time of this method depends on the given PDF, whereas its marginal generation times are almost independent of the target distribution.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Section 4.8 (Automatic Ratio-Of-Uniforms).

See Also

ur, tdr.new, unuran.cont, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a Gaussian distribution
pdf <- function (x) { exp(-0.5*x^2) }
gen <- arou.new(pdf=pdf, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Create a sample of size 100 for a
## Gaussian distribution (use logPDF)
logpdf <- function (x) { -0.5*x^2 }
gen <- arou.new(pdf=logpdf, islog=TRUE, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Same example but additionally provide derivative of log-density
## to prevent possible round-off errors
logpdf <- function (x) { -0.5*x^2 }
dlogpdf <- function (x) { -x }
gen <- arou.new(pdf=logpdf, dpdf=dlogpdf, islog=TRUE, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Draw sample from Gaussian distribution with mean 1 and
## standard deviation 2. Use 'dnorm'.
gen <- arou.new(pdf=dnorm, lb=-Inf, ub=Inf, mean=1, sd=2)
x <- ur(gen,100)

## Draw a sample from a truncated Gaussian distribution
## on domain [5,Inf)
logpdf <- function (x) { -0.5*x^2 }
gen <- arou.new(pdf=logpdf, lb=5, ub=Inf, islog=TRUE)
x <- ur(gen,100)

## Alternative approach
distr <- udnorm()
gen <- aroud.new(distr)
x <- ur(gen,100)
```

`ars.new` *UNU.RAN generator based on Adaptive Rejection Sampling (ARS)*

Description

UNU.RAN random variate generator for continuous distributions with given probability density function (PDF). It is based on Adaptive Rejection Sampling ('ARS').

[Universal] – Rejection Method.

Usage

```
ars.new(logpdf, dlogpdf=NULL, lb, ub, ...)
arsd.new(distr)
```

Arguments

`logpdf`	log-density function. (R function)
`dlogpdf`	derivative of `logpdf`. (R function)
`lb`	lower bound of domain; use `-Inf` if unbounded from left. (numeric)
`ub`	upper bound of domain; use `Inf` if unbounded from right. (numeric)
`...`	(optional) arguments for `logpdf`.
`distr`	distribution object. (S4 object of class `"unuran.cont"`)

Details

This function creates a `unuran` object based on 'ARS' (Adaptive Rejection Sampling). It can be used to draw samples from continuous distributions with given probability density function using `ur`.

Function `logpdf` is the logarithm the density function of the target distribution. It must be a concave function (i.e., the distribution must be log-concave). However, it need not be normalized (i.e., it can be a log-density plus some arbitrary constant).

The derivative `dlogpdf` of the log-density is optional. If omitted, numerical differentiation is used. Notice, however, that this might cause some round-off errors such that the algorithm fails.

Alternatively, one can use function `arsd.new` where the object `distr` of class `"unuran.cont"` must contain all required information about the distribution.

The setup time of this method depends on the given PDF, whereas its marginal generation times are almost independent of the target distribution.

'ARS' is a special case of method 'TDR' (see `tdr.new`). It is a bit slower and less flexible but numerically more stable. In particular, it is useful if one wants to sample from truncated distributions with extreme truncation points; or when the integral of the given "density" function is only known to be extremely large or small. However, this assumes that the log-density is computed analytically and not by just using `log(pdf(x))`.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Chapter 4 (Tranformed Density Rejection).

W. R. Gilks and P. Wild (1992): Adaptive rejection sampling for Gibbs sampling. Applied Statistics 41(2), pp. 337–348.

See Also

ur, tdr.new, unuran.cont, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a
## Gaussian distribution (use logPDF)
lpdf <- function (x) { -0.5*x^2 }
gen <- ars.new(logpdf=lpdf, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Same example but additionally provide derivative of log-density
## to prevent possible round-off errors
lpdf <- function (x) { -0.5*x^2 }
dlpdf <- function (x) { -x }
gen <- ars.new(logpdf=lpdf, dlogpdf=dlpdf, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Draw a sample from a truncated Gaussian distribution
## on domain [100,Inf)
lpdf <- function (x) { -0.5*x^2 }
gen <- ars.new(logpdf=lpdf, lb=50, ub=Inf)
x <- ur(gen,100)

## Alternative approach
distr <- udnorm()
gen <- arsd.new(distr)
x <- ur(gen,100)
```

dari.new *UNU.RAN generator based on Discrete Automatic Rejection Inversion (DARI)*

Description

UNU.RAN random variate generator for discrete distributions with given probability mass function (PMF). It is based on Discrete Automatic Rejection Inversion ('DARI').

[Universal] – Rejection Method.

Usage

```
dari.new(pmf, lb, ub, mode=NA, sum=1, ...)
darid.new(distr)
```

Arguments

pmf	probability mass function. (R function)
lb	lower bound of domain; use -Inf if unbounded from left. (numeric, integer)
ub	upper bound of domain; use Inf if unbounded from right. (numeric, integer)
mode	mode of distribution. (integer)
sum	sum over all "probabilities". (numeric)
...	(optional) arguments for pmf.
distr	distribution object. (S4 object of class "unuran.discr")

Details

This function creates an unuran object based on 'DARI' (Discrete Automatic Rejection Inversion). It can be used to draw samples of a discrete random variate with given probability mass function using ur.

Function pmf must be postive but need not be normalized (i.e., it can be any multiple of a probability mass function).

The given function must be $T_{-0.5}$-concave; this includes all log-concave distributions.

In addition the algorithm requires the location of the mode. If omitted then it is computed by a slow numerical search.

If the sum over all probabilities is different from 1 then a rough estimate of this sum is required.

Alternatively, one can use function darid.new where the object distr of class "unuran.discr" must contain all required information about the distribution.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Section 10.2 (Tranformed Probability Rejection).

See Also

ur, unuran.discr, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a Binomial distribution
## with 1000 number if observations and probability 0.2
gen <- dari.new(pmf=dbinom, lb=0, ub=1000, size=1000, prob=0.2)
x <- ur(gen,100)

## Create a sample from a distribution with PMF
##  p(x) = 1/x^3, x >= 1  (Zipf distribution)
zipf <- function (x) { 1/x^3 }
gen <- dari.new(pmf=zipf, lb=1, ub=Inf)
x <- ur(gen,100)

## Alternative approach
distr <- udbinom(size=100,prob=0.3)
gen <- darid.new(distr)
x <- ur(gen,100)
```

---

dau.new *UNU.RAN generator based on the Alias method (DAU)*

Description

UNU.RAN random variate generator for discrete distributions with given probability vector. It applies the Alias-Urn method ('DAU').

[Universal] – Patchwork Method.

Usage

```
dau.new(pv, from=1)
daud.new(distr)
```

Arguments

pv	vector of non-negative numbers (need not sum to 1). (numeric vector)
from	index of first entry in vector. (integer)
distr	distribution object. (S4 object of class "unuran.discr")

Details

This function creates a unuran object based on 'DAU' (Discrete Alias-Urn method). It can be used to draw samples of a discrete random variate with given probability vector using ur.

Vector pv must be postive but need not be normalized (i.e., it can be any multiple of a probability vector).

The method runs fast in constant time, i.e., marginal sampling times do not depend on the length of the given probability vector. Whereas their setup times grow linearly with this length.

Notice that the range of random variates is from:(from+length(pv)-1).

Alternatively, one can use function daud.new where the object distr of class "unuran.discr" must contain all required information about the distribution.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Section 3.2 (The Alias Method).

A.J. Walker (1977): An efficient method for generating discrete random variables with general distributions. ACM Trans. Model. Comput. Simul. 3, pp.253–256.

See Also

ur, unuran.discr, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a
## binomial distribution with size=115, prob=0.5
gen <- dau.new(pv=dbinom(0:115,115,0.5), from=0)
x <- ur(gen,100)

## Alternative approach
distr <- udbinom(size=100,prob=0.3)
gen <- daud.new(distr)
x <- ur(gen,100)
```

dgt.new *UNU.RAN generator based on table guided discrete inversion (DGT)*

Description

UNU.RAN random variate generator for discrete distributions with given probability vector. It applies the Guide-Table Method for discrete inversion ('DGT').

[Universal] – Inversion Method.

Usage

```
dgt.new(pv, from=1)
dgtd.new(distr)
```

Arguments

pv	vector of non-negative numbers (need not sum to 1). (numeric vector)
from	index of first entry in vector. (integer)
distr	distribution object. (S4 object of class "unuran.discr")

Details

This function creates an unuran object based on 'DGT' (Discrete Guide-Table method). It can be used to draw samples of a discrete random variate with given probability vector using ur. It also allows to compute quantiles by means of uq.

Vector pv must be postive but need not be normalized (i.e., it can be any multiple of a probability vector).

The method runs fast in constant time, i.e., marginal sampling times do not depend on the length of the given probability vector. Whereas their setup times grow linearly with this length.

Notice that the range of random variates is from: (from+length(pv)-1).

Alternatively, one can use function dgtd.new where the object distr of class "unuran.discr" must contain all required information about the distribution.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Section 3.1.2 (Indexed Search).

H.C. Chen and Y. Asau (1974): On generating random variates from an empirical distribution. AIIE Trans. 6, pp.163–166.

See Also

ur, uq, unuran.discr, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a
## binomial distribution with size=115, prob=0.5
gen <- dgt.new(pv=dbinom(0:115,115,0.5),from=0)
x <- ur(gen,100)

## Alternative approach
distr <- udbinom(size=100,prob=0.3)
gen <- dgtd.new(distr)
x <- ur(gen,100)
```

---

hitro.new *UNU.RAN generator based on Hit-and-Run sampler (HITRO)*

Description

UNU.RAN random variate generator for continuous multivariate distributions with given probability density function (PDF). It is based on the Hit-and-Run algorithm in combinaton with the Ratio-of-Uniforms method ('HITRO').

[Universal] – MCMC (Markov chain sampler).

Usage

```
hitro.new(dim=1, pdf, ll=NULL, ur=NULL, mode=NULL, center=NULL,
          thinning=1, burnin=0, ...)
```

Arguments

dim	number of dimensions of the distribution. (integer)
pdf	probability density function. (R function)
ll,ur	lower left and upper right vertex of a rectangular domain of the pdf. The domain is only set if both vertices are not NULL. Otherwise, the domain is unbounded by default. (numeric vectors)
mode	location of the mode. (numeric vector)
center	point in "typical" region of distribution, e.g. the approximate location of the mode. If omitted the mode is used. If the mode is not given either, the origin is used. (numeric vector)
thinning	thinning factor. (positive integer)
burnin	length of burnin-in phase. (positive integer)
...	(optional) arguments for pdf

Details

Beware: MCMC sampling can be dangerous!

This function creates a `unuran` object based on the Hit-and-Run algorithm in combinaton with the Ratio-of-Uniforms method ('HITRO'). It can be used to draw samples of a continuous random vector with given probability density function using `ur`.

The algorithm works best with log-concave distributions. Other distributions work as well but convergence can be slower.

The density must be provided by a function `pdf` which must return non-negative numbers and but need not be normalized (i.e., it can be any multiple of a density function).

The `center` is used as starting point of the Hit-and-Run algorithm. It is thus important, that the `center` is contained in the (interior of the) domain. Alternatively, one could provide the location of the `mode`. However, this requires its exact position whereas `center` allows any point in the "typical" region of the distribution.

If the `mode` is given, then it is used to obtain an upper bound on the `pdf` and thus its location should be given sufficiently accurate.

The 'HITRO' algorithm is a MCMC samplers and thus it does not produce a sequence of independent variates. The drawn sample follows the target distribution only approximately. The dependence between consecutive vectors can be decreased when only a subsequence is returned (and the other elements are erased). This is called "thinning" of the Markov chain and can be controlled by the `thinning` factor. A thinning factor k means that only every k-th element is returned.

Markov chains also depend on the chosen starting point (i.e., the `center` in this implementation of the algorithm). This dependence can be decreased by erasing the first part of the chain. This is called the "burn-in" of the Markov chain and its length is controlled by the argument `burnin`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

R. Karawatzki, J. Leydold, and K. P\"otzelberger (2005): Automatic Markov Chain Monte Carlo Procedures for Sampling from Multivariate Distributions. Research Report Series / Department of Statistics and Mathematics, Nr. 27, December 2005 Department of Statistics and Mathematics, Wien, Wirtschaftsuniv., 2005. https://epub.wu.ac.at/id/eprint/1400

See Also

`ur`, `unuran.new`, `unuran`.

Examples

```
## Create a sample of size 100 for a
## Gaussian distribution
mvpdf <- function (x) { exp(-sum(x^2)) }
gen <- hitro.new(dim=2, pdf=mvpdf)
x <- ur(gen,100)
```

```
## Use mode of Gaussian distribution.
## Reduce auto-correlation by thinning and burn-in.
##  mode at (0,0)
##  thinning factor 3
##    (only every 3rd vector in the sequence is returned)
##  burn-in of length 1000
##    (the first 100 vectors in the sequence are discarded)
mvpdf <- function (x) { exp(-sum(x^2)) }
gen <- hitro.new(dim=2, pdf=mvpdf, mode=c(0,0), thinning=3, burnin=1000)
x <- ur(gen,100)

## Gaussian distribution restricted to the rectangle [1,2]x[1,2]
##  (don't forget to provide a starting point using 'center')
mvpdf <- function (x) { exp(-sum(x^2)) }
gen <- hitro.new(dim=2, pdf=mvpdf, center=c(1.1,1.1), ll=c(1,1), ur=c(2,2))
x <- ur(gen,100)
```

`itdr.new` *UNU.RAN generator based on Inverse Transformed Density Rejection (ITDR)*

Description

UNU.RAN random variate generator for continuous distributions with given probability density function (PDF). It is based on the Inverse Transformed Density Rejection method ('ITDR').

[Universal] – Rejection Method.

Usage

```
itdr.new(pdf, dpdf, lb, ub, pole, islog=FALSE, ...)
itdrd.new(distr)
```

Arguments

`pdf`	probability density function. (R function)
`dpdf`	derivative of `pdf`. (R function)
`pole`	pole of distribution. (numeric)
`lb`	lower bound of domain; use `-Inf` if unbounded from left. (numeric)
`ub`	upper bound of domain; use `Inf` if unbounded from right. (numeric)
`islog`	whether `pdf` is given as log-density (the `dpdf` must then be the derivative of the log-density). (boolean)
`...`	(optional) arguments for `pdf`.
`distr`	distribution object. (S4 object of class `"unuran.cont"`)

Details

This function creates a unuran object based on "ITDR" (Inverse Transformed Density Rejection). It can be used to draw samples of a continuous random variate with given probability density function using ur.

The density pdf must be positive but need not be normalized (i.e., it can be any multiple of a density function). The algorithm is especially designed for distributions with unbounded densities. Thus the algorithm needs the position of the pole. Moreover, the given function must be monotone on its domain.

The derivative dpdf is essential. (Numerical derivation does not work as it results in serious round-off errors.)

Alternatively, one can use function itdrd.new where the object distr of class "unuran.cont" must contain all required information about the distribution.

The setup time of this method depends on the given PDF, whereas its marginal generation times are almost independent of the target distribution.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2007): Inverse transformed density rejection for unbounded monotone densities. ACM Trans. Model. Comput. Simul. 17(4), Article 18, 16 pages. DOI: 10.1145/1276927.1276931

See Also

ur, unuran.cont, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a Gamma(0.5) distribution
pdf <- function (x) { x^(-0.5)*exp(-x) }
dpdf <- function (x) { (-x^(-0.5) - 0.5*x^(-1.5))*exp(-x) }
gen <- itdr.new(pdf=pdf, dpdf=dpdf, lb=0, ub=Inf, pole=0)
x <- ur(gen,100)

## Alternative approach
distr <- udgamma(shape=0.5)
gen <- itdrd.new(distr)
x <- ur(gen,100)
```

`mixt.new` *UNU.RAN generator for finite mixture of distributions*

Description

UNU.RAN random variate generator for a finite mixture of continuous or discrete distributions. The components are given as `unuran` objects.

[Universal] – Composition Method.

Usage

```
mixt.new(prob, comp, inversion=FALSE)
```

Arguments

`prob`	weights of mixture ("probabilities"); these must be non-negative numbers but need not sum to 1. (numeric vector)
`comp`	components of mixture. (list of S4 object of class `"unuran"`)
`inversion`	whether inversion method should be used. (boolean)

Details

Given a set of probability density functions $p_1(x), \ldots, p_n(x)$ (called the mixture components) and weights $w_1, \ldots, w_n$ such that $w_i \geq 0$ and $\sum w_i = 1$, the sum

$$q(x) = \sum_{i=1}^{n} w_i \, p_i(x)$$

is called the mixture density.

Function `mixt.new` creates an `unuran` object for a finite mixture of continuous or discrete univariate distributions. It can be used to draw samples of a continuous random variate using `ur`.

The weights `prob` must be a vector of non-negative numbers (not all equal to 0) but need not sum to 1.

`comp` is a list of `"unuran"` generator objects. Each of which must sample from a continuous or discrete univariate distribution.

If `inversion` is `TRUE`, then the inversion method is used for sampling from the mixture distribution. However, the following conditions must be satisfied:

- Each component (`unuran` object) must use implement an inversion method (i.e., the quantile funtion `uq` must work).
- The domains of the components must not overlapping.
- The components must be order with respect to their domains.

If one of these conditions is violated, then initialization of the mixture object fails.

The setup time is fast, whereas its marginal generation times strongly depend on the average generation times of its components.

Value

An object of class "unuran".

Note

Each component in comp must correspond to a continuous or discrete univariate distribution. In particular this also includes mixtures of distributions. Thus mixtures can also be defined recursively.

Moreover, none of these components must be packed (see unuran.packed).

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Section 2.3 (Composition).

See Also

ur, uq, unuran.new, unuran.

Examples

```
## Create a mixture of an Exponential and a Half-normal distribution
unr1 <- unuran.new(udnorm(lb=-Inf, ub=0))
unr2 <- unuran.new(udexp())
mix <- mixt.new( c(1,1), c(unr1, unr2) )
x <- ur(mix,100)

## Now use inversion method:
## It is important that
##  1. we use a inversion for each component
##  2. the domains to not overlap
##  3. the components are ordered with respect to their domains
unr1 <- pinvd.new(udnorm(lb=-Inf, ub=0))
unr2 <- pinvd.new(udexp())
mix <- mixt.new( c(1,1), c(unr1, unr2), inversion=TRUE )
x <- ur(mix,100)

## We also can compute the inverse distribution function
##x <- uq(mix,0.90)

## Create a mixture of Exponential and Geometric distrbutions
unr1 <- unuran.new(udexp())
unr2 <- unuran.new(udgeom(0.7))
mix <- mixt.new( c(0.6,0.4), c(unr1, unr2) )
x <- ur(mix,100)
```

`pinv.new` *UNU.RAN generator based on Polynomial interpolation of INVerse CDF (PINV)*

Description

UNU.RAN random variate generator for continuous distributions with given probability density function (PDF) or cumulative distribution function (CDF). It is based on the Polynomial interpolation of INVerse CDF ('PINV').

[Universal] – Inversion Method.

Usage

```
pinv.new(pdf, cdf, lb, ub, islog=FALSE, center=0,
         uresolution=1.e-10, smooth=FALSE, ...)
pinvd.new(distr, uresolution=1.e-10, smooth=FALSE)
```

Arguments

`pdf`	probability density function. (R function)
`cdf`	cumulative distribution function. (R function)
`lb`	lower bound of domain; use `-Inf` if unbounded from left. (numeric)
`ub`	upper bound of domain; use `Inf` if unbounded from right. (numeric)
`islog`	whether `pdf` and `cdf` are given by their corresponding logarithms. (boolean)
`center`	"typical" point of distribution. (numeric)
`...`	(optional) arguments for `pdf` and `cdf`.
`distr`	distribution object. (S4 object of class `"unuran.cont"`)
`uresolution`	maximal acceptable u-error. (numeric)
`smooth`	whether the inverse CDF is differentiable. (boolean)

Details

This function creates an `unuran` object based on 'PINV' (Polynomial interpolation of INVerse CDF). It can be used to draw samples of a continuous random variate with given probability density function `pdf` or cumulative distribution function `cdf` by means of `ur`. It also allows to compute quantiles by means of `uq`.

Function `pdf` must be positive but need not be normalized (i.e., it can be any multiple of a density function). The set of points where the `pdf` is strictly positive must be connected. The `center` is a point where the `pdf` is not too small, e.g., (a point near) the mode of the distribution.

If the density `pdf` is given, then the algorithm automatically computes the CDF using Gauss-Lobatto integration. If the `cdf` is given but not the `pdf` then the CDF is used instead of the PDF. However, we found in our experiments that using the PDF is numerically more stable.

Alternatively, one can use function `pinvd.new` where the object `distr` of class `"unuran.cont"` must contain all required information about the distribution.

The algorithm approximates the inverse of the CDF of the distribution by means of Newton interpolation between carefully selected nodes. The approxiating functing is thus continuous. Argument `smooth` controls whether this function is also differentiable("smooth") at the nodes. Using `smooth=TRUE` requires the `pdf` of the distribution. It results in a higher setup time and memory consumption. Thus using `smooth=TRUE` is not *not recommended*, unless differentiability is important.

The approximation error is estimated by means of the the u-error, i.e., $|CDF(G(U)) - U|$, where G denotes the approximation of the inverse CDF. The error can be controlled by means of argument `uresolution`.

When sampling from truncated distributions with extreme truncation points, it is recommended to provide the log-density by setting `islog=TRUE`. Then the algorithm is numerically more stable.

The setup time of this method depends on the given PDF, whereas its marginal generation times are independent of the target distribution.

Value

An object of class `"unuran"`.

Remark

Using function `up` generator objects that implement method 'PINV' may also be used to approximate the cumulative distribution function of the given distribution when only the density is given. The approximation error is about one tenth of the requested `uresolution`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

G. Derflinger, W. H\"ormann, and J. Leydold (2010): Random variate generation by numerical inversion when only the density is known. ACM Trans. Model. Comput. Simul., 20:4, #18

See Also

`ur`, `uq`, `up`, `unuran.cont`, `unuran.new`, `unuran`.

Examples

```
## Create a sample of size 100 for a Gaussian distribution
pdf <- function (x) { exp(-0.5*x^2) }
gen <- pinv.new(pdf=pdf, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Create a sample of size 100 for a
## Gaussian distribution (use logPDF)
logpdf <- function (x) { -0.5*x^2 }
gen <- pinv.new(pdf=logpdf, islog=TRUE, lb=-Inf, ub=Inf)
```

```
x <- ur(gen,100)

## Draw sample from Gaussian distribution with mean 1 and
## standard deviation 2. Use 'dnorm'.
gen <- pinv.new(pdf=dnorm, lb=-Inf, ub=Inf, mean=1, sd=2)
x <- ur(gen,100)

## Draw a sample from a truncated Gaussian distribution
## on domain [2,Inf)
gen <- pinv.new(pdf=dnorm, lb=2, ub=Inf)
x <- ur(gen,100)

## Improve the accuracy of the approximation
gen <- pinv.new(pdf=dnorm, lb=-Inf, ub=Inf, uresolution=1e-15)
x <- ur(gen,100)

## We have to provide a 'center' when PDF (almost) vanishes at 0.
gen <- pinv.new(pdf=dgamma, lb=0, ub=Inf, center=4, shape=5)
x <- ur(gen,100)

## We also can force a smoother approximation
gen <- pinv.new(pdf=dnorm, lb=-Inf, ub=Inf, smooth=TRUE)
x <- ur(gen,100)

## Alternative approach
distr <- udnorm()
gen <- pinvd.new(distr)
x <- ur(gen,100)
```

`Runuran.distributions` *UNU.RAN distribution objects*

Description

Create objects for particular distributions suitable for using with generation methods from the UNU.RAN library.

Details

Runuran provides an interface to the UNU.RAN library for universal non-uniform random number generators. This is a very flexible and powerful collection of sampling routines, where the user first has to specify the target distribution and then has to choose an appropriate sampling method.

Creating an object for a particular distribution can be a bit tedious especially if the target distribution has a more complex density function. Thus we have compiled a set of functions that provides ready-to-use distribution objects. Moreover, using these object often results in faster setup time than objects created with pure R code.

These functions share a similar syntax and naming scheme (only ud is prefixed) with analogous R built-in functions that provide density, distribution function and quantile:

`ud...(distribution parameters,lb ,ub)`

Currently generators for the following distributions are implemented.

Continuous Univariate Distributions (26):

Function		*Distribution*
udbeta	...	Beta
udcauchy	...	Cauchy
udchi	...	Chi
udchisq	...	Chi-squared
udexp	...	Exponential
udf	...	F
udfrechet	...	Frechet (Extreme value type II)
udgamma	...	Gamma
udghyp	...	Generalized Hyperbolic
udgig	...	Generalized Inverse Gaussian
udgumbel	...	Gumbel (Extreme value type I)
udhyperbolic	...	Hyperbolic
udig	...	Inverse Gaussian (Wald)
udlaplace	...	Laplace (double exponential)
udlnorm	...	Log Normal
udlogis	...	Logistic
udlomax	...	Lomax (Pareto of second kind)
udmeixner	...	Meixner
udnorm	...	Normal (Gaussian)
udpareto	...	Pareto (of first kind)
udpowerexp	...	Powerexponential (Subbotin)
udrayleigh	...	Rayleigh
udslash	...	Slash
udt	...	t (Student)
udvg	...	Variance Gamma
udweibull	...	Weibull (Extreme value type III)

Discrete Distributions (6):

Function		*Distribution*
udbinom	...	Binomial
udgeom	...	Geometric
udhyper	...	Hypergeometric
udlogarithmic	...	Logarithmic
udnbinom	...	Negative Binomial
udpois	...	Poisson

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

See Also

Runuran-package.

Examples

```
## Create an object for a gamma distribution with shape parameter 5.
distr <- udgamma(shape=5)
## Create the UNU.RAN generator object. use method PINV (inversion).
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen, 100)
## Compute some quantiles for Monte Carlo methods
x <- uq(gen, (1:9)/10)

## Analogous for half normal distribution
distr <- udnorm(lb=0, ub=Inf)
gen <- pinvd.new(distr)
x <- ur(gen, 100)
x <- uq(gen, (1:9)/10)

## Analogous for a generalized hyperbolic distribution
distr <- udghyp(lambda=-1.0024, alpha=39.6, beta=4.14, delta=0.0118, mu=-0.000158)
gen <- pinvd.new(distr)
x <- ur(gen, 100)
x <- uq(gen, (1:9)/10)

## It is also possible to compute density or distribution functions.
## However, this might not work for all generator objects.
##    Density
x <- ud(gen, 1.2)
##    Cumulative distribution function
x <- up(gen, 1.2)
```

Runuran.options *Set or return options for Runuran library*

Description

Library **Runuran** has some parameters which (usually) affect the behavior of its functions. These can be set for the whole session via `Runuran.options`.

Usage

```
Runuran.options(...)
```

Arguments

`...`	A list may be given as the only argument, or any number of arguments may be in the `name=value` form, a character string for the name of a parameter, or no argument at all may be given.

Details

The function provides a tool to control the behavior of library **Runuran**. A list may be given as the only argument, or any number of arguments may be in the `name=value` form. If no arguments are specified then the function returns the current settings of all parameters. If a single option name is given as character string, then its value is returned (or `NULL` if it does not exist). Option *values* may be abbreviated.

Currently used parameters in alphabetical order:

error.level verbosity level of error messages and warnings from the underlying UNU.RAN library. It has no effect on messages from the routines in this package. Warnings are useful for analysing possible problems with the selected combinations of distribution and method. However, they can produce quite a lot of output if the conditions of the method is appropriate for a distribution or the distribution has properties like very heavy tails or very high peaks.

The following levels can be set:

`"default"`: same as `"warning"`.

`"none"`: all error messages and warnings are suppressed.

`"error"`: only show error messages.

`"warning"`: show error messages and some of the warnings.

`"all"`: show all error messages and warnings.

Value

`Runuran.options` returns a list with the updated values of the parameters. If the argument list is not empty, the returned list is invisible. If a single character string is given, then the value of the corresponding parameter is returned (or `NULL` if the parameter is not used).

Author(s)

Josef Leydold <josef.leydold@wu.ac.at>

Examples

```
## save current options
oldval <- Runuran.options()

## show current options
Runuran.options("error.level")

## suppress all UNU.RAN error messages and warnings
Runuran.options(error.level="none")

## restore Runuran options
Runuran.options(oldval)
```

`Runuran.special.generators` *Generators for distributions based on methods from the UNU.RAN library*

Description

Generators for particular distributions. Their syntax is similar to the corresponding R built-in functions.

Details

Runuran provides an interface to the UNU.RAN library for universal non-uniform random number generators. This is a very flexible and powerful collection of sampling routines, where the user first has to specify the target distribution and then has to choose an appropriate sampling method. However, we found that this approach is a little bit confusing for the beginner.

Thus we have prepared easy-to-use sampling functions for standard distributions to facilitate the use of the package. All these functions share a similar syntax and naming scheme (only `u` is prefixed) with their analogous R built-in generating functions (if these exist) but have optional domain arguments `lb` and `ub`, i.e., these calls also allow to draw samples from truncated distributions:

`ur...(n,distribution parameters,lb ,ub)`

These functions also show the interested user how we used the more powerful functions. We recommend to directly use these more flexible functions. Then one has faster marginal generation times and one may choose the best generation method for one's application.

Currently generators for the following distributions are implemented.

Continuous Univariate Distributions (24):

Function		*Distribution*
`urbeta`	...	Beta
`urburr`	...	Burr
`urcauchy`	...	Cauchy
`urchi`	...	Chi
`urchisq`	...	Chi-squared
`urexp`	...	Exponential
`urextremeI`	...	Gumbel (extreme value type I)
`urextremeII`	...	Frechet (extreme value type II)
`urf`	...	F
`urgamma`	...	Gamma
`urgig`	...	GIG (generalized inverse Gaussian)
`urhyperbolic`	...	Hyperbolic
`urlaplace`	...	Laplace
`urlnorm`	...	Log-Normal
`urlogis`	...	Logistic
`urlomax`	...	Lomax
`urnorm`	...	Normal (Gaussian)

urpareto	...	Pareto (of first kind)
urplanck	...	Planck
urpowerexp	...	Powerexponential (Subbotin)
urrayleigh	...	Rayleigh
urt	...	t (Student)
urtriang	...	Triangular
urweibull	...	Weibull

Discrete Distributions (6):

Function		*Distribution*
urbinom	...	Binomial
urgeom	...	Geometric
urhyper	...	Hypergeometric
urlogarithmic	...	Logarithmic
urnbinom	...	Negative Binomial
urpois	...	Poisson

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

See Also

Runuran-package, Runuran.distributions.

Examples

```
## draw a sample of size 100 from a
## gamma distribution with shape parameter 5
x <- urgamma(n=100, shape=5)

## draw a sample of size 100 from a
## half normal distribution
x <- urnorm(n=100, lb=0, ub=Inf)
```

---

srou.new *UNU.RAN generator based on Simple Ratio-Of-Uniforms Method (SROU)*

Description

UNU.RAN random variate generator for continuous distributions with given probability density function (PDF). It is based on the Simple Ratio-Of-Uniforms Method ('SROU').

[Universal] – Rejection Method.

Usage

```
srou.new(pdf, lb, ub, mode, area, islog=FALSE, r=1, ...)
sroud.new(distr, r=1)
```

Arguments

pdf	probability density function. (R function)
lb	lower bound of domain; use -Inf if unbounded from left. (numeric)
ub	upper bound of domain; use Inf if unbounded from right. (numeric)
mode	location of the mode. (numeric)
area	area below pdf. (numeric)
islog	whether pdf is given as log-density (the dpdf must then be the derivative of the log-density). (boolean)
...	(optional) arguments for pdf.
distr	distribution object. (S4 object of class "unuran.cont")
r	adjust algorithm to heavy-tailed distribution. (numeric)

Details

This function creates a unuran object based on 'SROU' (Simple Ratio-Of-Uniforms Method). It can be used to draw samples of a continuous random variate with given probability density function using ur.

The density pdf must be positive but need not be normalized (i.e., it can be any multiple of a density function). It must be T_c-concave for $c = -r/(r+1)$; this includes all log-concave distributions.

The (exact) location of the mode and the area below the pdf are essential.

Alternatively, one can use function sroud.new where the object distr of class "unuran.cont" must contain all required information about the distribution.

The acceptance probability decreases with increasing parameter r. Thus it should be as small as possible. On the other hand it must be sufficiently large for heavy tailed distributions. If possible, use the default r=1.

Compared to tdr.new it has much slower marginal generation times but has a faster setup and is numerically more robust. Moreover, It also works for unimodal distributions with tails that are heavier than those of the Cauchy distribution.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. Sections 6.3 and 6.4.

See Also

ur, unuran.cont, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a Gaussian distribution.
pdf <- function (x) { exp(-0.5*x^2) }
gen <- srou.new(pdf=pdf, lb=-Inf, ub=Inf, mode=0, area=2.506628275)
x <- ur(gen,100)

## Create a sample of size 100 for a Gaussian distribution.
## Use 'dnorm'.
gen <- srou.new(pdf=dnorm, lb=-Inf, ub=Inf, mode=0, area=1)
x <- ur(gen,100)

## Alternative approach
distr <- udnorm()
gen <- sroud.new(distr)
x <- ur(gen,100)
```

---

tabl.new *UNU.RAN generator based on TABLe based Rejection (TABL)*

Description

UNU.RAN random variate generator for continuous distributions with given probability density function (PDF). It is based on the TABLe based rejection method ('TABL').

[Universal] – Rejection Method.

Usage

```
tabl.new(pdf, lb, ub, mode, islog=FALSE, ...)
tabld.new(distr)
```

Arguments

pdf	probability density function. (R function)
lb	lower bound of domain; use -Inf if unbounded from left. (numeric)
ub	upper bound of domain; use Inf if unbounded from right. (numeric)
mode	location of the mode. (numeric)
islog	whether pdf is given as log-density (the dpdf must then be the derivative of the log-density). (boolean)
...	(optional) arguments for pdf.
distr	distribution object. (S4 object of class "unuran.cont")

Details

This function creates an unuran object based on 'TABL' (TABLe based rejection). It can be used to draw samples of a continuous random variate with given probability density function using ur.

The density pdf must be positive but need not be normalized (i.e., it can be any multiple of a density function).

The given pdf must be unimodal.

Alternatively, one can use function tabld.new where the object distr of class "unuran.cont" must contain all required information about the distribution.

The setup time of this method depends on the given PDF, whereas its marginal generation times are almost independent of the target distribution.

Value

An object of class "unuran".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Section 5.1 ("Ahrens Method").

See Also

ur, tdr.new, unuran.cont, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a Gaussian distribution
pdf <- function (x) { exp(-0.5*x^2) }
gen <- tabl.new(pdf=pdf, lb=-Inf, ub=Inf, mode=0)
x <- ur(gen,100)

## Create a sample of size 100 for a
## Gaussian distribution (use logPDF)
logpdf <- function (x) { -0.5*x^2 }
gen <- tabl.new(pdf=logpdf, islog=TRUE, lb=-Inf, ub=Inf, mode=0)
x <- ur(gen,100)

## Draw sample from Gaussian distribution with mean 1 and
## standard deviation 2. Use 'dnorm'.
gen <- tabl.new(pdf=dnorm, lb=-Inf, ub=Inf, mode=1, mean=1, sd=2)
x <- ur(gen,100)

## Draw a sample from a truncated Gaussian distribution
## on domain [5,Inf)
logpdf <- function (x) { -0.5*x^2 }
```

```
gen <- tabl.new(pdf=logpdf, lb=5, ub=Inf, mode=5, islog=TRUE)
x <- ur(gen,100)

## Alternative approach
distr <- udnorm()
gen <- tabld.new(distr)
x <- ur(gen,100)
```

`tdr.new` *UNU.RAN generator based on Transformed Density Rejection (TDR)*

Description

UNU.RAN random variate generator for continuous distributions with given probability density function (PDF). It is based on the Transformed Density Rejection method ('TDR').

[Universal] – Rejection Method.

Usage

```
tdr.new(pdf, dpdf=NULL, lb, ub, islog=FALSE, ...)
tdrd.new(distr)
```

Arguments

`pdf`	probability density function. (R function)
`dpdf`	derivative of `pdf`. (R function)
`lb`	lower bound of domain; use `-Inf` if unbounded from left. (numeric)
`ub`	upper bound of domain; use `Inf` if unbounded from right. (numeric)
`islog`	whether `pdf` is given as log-density (the `dpdf` must then be the derivative of the log-density). (boolean)
`...`	(optional) arguments for `pdf`.
`distr`	distribution object. (S4 object of class `"unuran.cont"`)

Details

This function creates an `unuran` object based on 'TDR' (Transformed Density Rejection). It can be used to draw samples of a continuous random variate with given probability density function using `ur`.

The density `pdf` must be positive but need not be normalized (i.e., it can be any multiple of a density function). The derivative `dpdf` of the (log-) density is optional. If omitted, numerical differentiation is used. Notice, however, that this might cause some round-off errors such that the algorithm fails. This is in particular the case when the density function is provided instead of the log-density.

The given `pdf` must be $T_{-0.5}$-concave (with implies unimodal densities with tails not higher than $(1/x^2)$; this includes all log-concave distributions).

It is recommended to use the log-density (instead of the density function) as this is numerically more stable.

Alternatively, one can use function `tdrd.new` where the object `distr` of class `"unuran.cont"` must contain all required information about the distribution.

The setup time of this method depends on the given PDF, whereas its marginal generation times are almost independent of the target distribution.

There exists a variant of 'TDR' which is numerically more stable (albeit a bit slower and less flexible) which is avaible via the `ars.new` function.

Value

An object of class `"unuran"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. See Chapter 4 (Tranformed Density Rejection).

See Also

`ur`, `ars.new`, `unuran.cont`, `unuran.new`, `unuran`.

Examples

```
## Create a sample of size 100 for a Gaussian distribution
pdf <- function (x) { exp(-0.5*x^2) }
gen <- tdr.new(pdf=pdf, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Create a sample of size 100 for a
## Gaussian distribution (use logPDF)
logpdf <- function (x) { -0.5*x^2 }
gen <- tdr.new(pdf=logpdf, islog=TRUE, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Same example but additionally provide derivative of log-density
## to prevent possible round-off errors
logpdf <- function (x) { -0.5*x^2 }
dlogpdf <- function (x) { -x }
gen <- tdr.new(pdf=logpdf, dpdf=dlogpdf, islog=TRUE, lb=-Inf, ub=Inf)
x <- ur(gen,100)

## Draw sample from Gaussian distribution with mean 1 and
## standard deviation 2. Use 'dnorm'.
gen <- tdr.new(pdf=dnorm, lb=-Inf, ub=Inf, mean=1, sd=2)
x <- ur(gen,100)
```

```
## Draw a sample from a truncated Gaussian distribution
## on domain [5,Inf)
logpdf <- function (x) { -0.5*x^2 }
gen <- tdr.new(pdf=logpdf, lb=5, ub=Inf, islog=TRUE)
x <- ur(gen,100)

## Alternative approach
distr <- udnorm()
gen <- tdrd.new(distr)
x <- ur(gen,100)
```

---

ud *Density function for "unuran" object*

Description

Evaluates the probability density function (PDF) or probability mass function (PMF) for a "unuran" object for a continuous and discrete distribution, respectively.

Usage

```
ud(obj, x, islog = FALSE)
```

Arguments

`obj`	one of • a distribution object of class `"unuran.cont"` that contains the PDF, or • a distribution object of class `"unuran.discr"` that contains the PMF, or • a generator object (class `"unuran"`) that contains the PDF and PMF, resp.
`x`	vector of x values. (numeric)
`islog`	if `TRUE`, the log-density is returnd. (boolean)

Details

The routine evaluates the probability density function of a distribution stored in a UNU.RAN distribution object or UNU.RAN generator object. If `islog` is `TRUE`, then the logarithm of the density is returned.

If the PDF (or its respective logarithm) is not available in the object, then `NA` is returned and a warning is thrown.

Note: when the log-density is not given explicitly (by setting `islog=TRUE` in the corresponding routing like `unuran.cont.new` or in an **Runuran** built-in distribution), then `NA` is returned even if the density is given.

Important: Routine `ud` just evaluates the density function that is stored in `obj`. It ignores the boundaries of the domain of the distribution, i.e., it does not return `0` outside the domain unless the implementation of the PDF handles this case correctly. This behavior is in particular important when **Runuran** built-in distributions are truncated by explicitly setting the domain boundaries.

Note

The generator object must not be packed (see unuran.packed).

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

See Also

unuran.cont, unuran.discr, unuran.

Examples

```
## Create an UNU.RAN distribution object (for standard Gaussian)
## and evaluate density for some points
distr <- udnorm()
ud(distr, 1.5)
ud(distr, -3:3)

## Create an UNU.RAN generator object (for standard Gaussian)
## and evaluate density of underyling distribution
gen <- tdrd.new(udnorm())
ud(gen, 1.5)
ud(gen, -3:3)
```

---

udbeta *UNU.RAN object for Beta distribution*

Description

Create UNU.RAN object for a Beta distribution with with parameters shape1 and shape2.

[Distribution] – Beta.

Usage

```
udbeta(shape1, shape2, lb=0, ub=1)
```

Arguments

shape1,shape2	positive shape parameters of the Beta distribution.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Beta distribution with parameters `shape1` $= a$ and `shape2` $= b$ has density

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^a (1-x)^b$$

for $a > 0$, $b > 0$ and $0 \leq x \leq 1$.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 25, p. 210.

See Also

`unuran.cont`.

Examples

```
## Create distribution object for beta distribution
distr <- udbeta(shape1=3,shape2=7)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

`udbinom` *UNU.RAN object for Binomial distribution*

Description

Create UNU.RAN object for a Binomial distribution with parameters `size` and `prob`.

[Distribution] – Binomial.

Usage

```
udbinom(size, prob, lb=0, ub=size)
```

Arguments

size	number of trials (one or more).
prob	probability of success on each trial.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Binomial distribution with size $= n$ and prob $= p$ has probability mass function

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

for $x = 0, \ldots, n$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.discr".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and A.W. Kemp (1992): Univariate Discrete Distributions. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 3, p. 105.

See Also

unuran.discr.

Examples

```
## Create distribution object for Binomial distribution
dist <- udbinom(size=100, prob=0.33)
## Generate generator object; use method DGT (inversion)
gen <- dgtd.new(dist)
## Draw a sample of size 100
x <- ur(gen,100)
```

udcauchy *UNU.RAN object for Cauchy distribution*

Description

Create UNU.RAN object for a Cauchy distribution with location parameter `location` and scale parameter `scale`.

[Distribution] – Cauchy.

Usage

```
udcauchy(location=0, scale=1, lb=-Inf, ub=Inf)
```

Arguments

`location`	location parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Cauchy distribution with location l and scale s has density

$$f(x) = \frac{1}{\pi s}\left(1 + \left(\frac{x-l}{s}\right)^2\right)^{-1}$$

for all x.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 16, p. 299.

See Also

`unuran.cont`.

Examples

```
## Create distribution object for Cauchy distribution
distr <- udcauchy()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udchi *UNU.RAN object for Chi distribution*

Description

Create UNU.RAN object for a Chi distribution with df degrees of freedom.

[Distribution] – Chi.

Usage

```
udchi(df, lb=0, ub=Inf)
```

Arguments

df	degrees of freedom (strictly positive). Non-integer values allowed.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Chi distribution with df$= n > 0$ degrees of freedom has density

$$f(x) = x^{n-1} e^{-x^2/2}$$

for $x > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 18, p. 417.

See Also

unuran.cont.

Examples

```
## Create distribution object for chi-squared distribution
distr <- udchi(df=5)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udchisq *UNU.RAN object for Chi-Squared distribution*

Description

Create UNU.RAN object for a Chi-squared (χ^2) distribution with df degrees of freedom.

[Distribution] – Chi-squared.

Usage

```
udchisq(df, lb=0, ub=Inf)
```

Arguments

df	degrees of freedom (strictly positive). Non-integer values allowed.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Chi-squared distribution with df$= n > 0$ degrees of freedom has density

$$f_n(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2}$$

for $x > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 18, p. 416

See Also

unuran.cont.

Examples

```
## Create distribution object for chi-squared distribution
distr <- udchisq(df=5)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udexp *UNU.RAN object for Exponential distribution*

Description

Create UNU.RAN object for an Exponential distribution with rate `rate` (i.e., mean `1/rate`).

[Distribution] – Exponential.

Usage

```
udexp(rate=1, lb=0, ub=Inf)
```

Arguments

`rate`	(strictly positive) rate parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Exponential distribution with rate λ has density

$$f(x) = \lambda e^{-\lambda x}$$

for $x \geq 0$.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 19, p. 494.

See Also

unuran.cont.

Examples

```
## Create distribution object for standard exponential distribution
distr <- udexp()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udf *UNU.RAN object for F distribution*

Description

Create UNU.RAN object for an F distribution with mean with df1 and df2 degrees of freedom. [Distribution] – F.

Usage

```
udf(df1, df2, lb=0, ub=Inf)
```

Arguments

df1, df2	(strictly positive) degrees of freedom. Non-integer values allowed.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The F distribution with df1 = n_1 and df2 = n_2 degrees of freedom has density

$$f(x) = \frac{\Gamma(n_1/2 + n_2/2)}{\Gamma(n_1/2)\Gamma(n_2/2)} \left(\frac{n_1}{n_2}\right)^{n_1/2} x^{n_1/2-1} \left(1 + \frac{n_1 x}{n_2}\right)^{-(n_1+n_2)/2}$$

for $x > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 27, p. 332

See Also

unuran.cont.

Examples

```
## Create distribution object for F distribution
distr <- udf(df1=3,df2=6)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udfrechet *UNU.RAN object for Frechet distribution*

Description

Create UNU.RAN object for a Frechet (Extreme value type II) distribution with shape parameter shape, location parameter location and scale parameter scale.

[Distribution] – Frechet (Extreme value type II).

Usage

```
udfrechet(shape, location=0, scale=1, lb=location, ub=Inf)
```

Arguments

shape	(strictly positive) shape parameter.
location	location parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Frechet distribution function with shape k, location l and scale s is

$$F(x) = \exp(-(\frac{x-l}{s})^{-k})$$

for $x \geq l$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 22, p. 2.

See Also

unuran.cont.

Examples

```
## Create distribution object for Frechet distribution
distr <- udfrechet(shape=2)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

udgamma *UNU.RAN object for Gamma distribution*

Description

Create UNU.RAN object for a Gamma distribution with parameters `shape` and `scale`.

[Distribution] – Gamma.

Usage

```
udgamma(shape, scale=1, lb=0, ub=Inf)
```

Arguments

`shape`	(strictly positive) shape parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution
`ub`	upper bound of (truncated) distribution

Details

The Gamma distribution with parameters `shape` $= \alpha$ and `scale` $= \sigma$ has density

$$f(x) = \frac{1}{\sigma^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\sigma}$$

for $x \geq 0$, $\alpha > 0$ and $\sigma > 0$. (Here $\Gamma(\alpha)$ is the function implemented by R's `gamma()` and defined in its help.)

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 17, p. 337.

See Also

`unuran.cont`.

Examples

```
## Create distribution object for gamma distribution
distr <- udgamma(shape=4)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udgeom *UNU.RAN object for Geometric distribution*

Description

Create UNU.RAN object for a Geometric distribution with parameter prob.

[Distribution] – Geometric.

Usage

```
udgeom(prob, lb = 0, ub = Inf)
```

Arguments

prob	probability of success in each trial. 0 < prob <= 1.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Geometric distribution with prob $= p$ has density

$$p(x) = p(1-p)^x$$

for $x = 0, 1, 2, \ldots$, $0 < p \leq 1$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.discr".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and A.W. Kemp (1992): Univariate Discrete Distributions. 2nd edition, John Wiley & Sons, Inc., New York. Sect. 5.2, p. 201

Examples

```
## Create distribution object for Geometric distribution
dist <- udgeom(prob=0.33)
## Generate generator object; use method DARI
gen <- darid.new(dist)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udghyp *UNU.RAN object for Generalized Hyperbolic distribution*

Description

Create UNU.RAN object for a Generalized Hyperbolic distribution with shape parameter `lambda`, shape parameter `alpha`, asymmetry (shape) parameter `beta`, scale parameter `delta`, and location parameter `mu`.

[Distribution] – Generalized Hyperbolic.

Usage

```
udghyp(lambda, alpha, beta, delta, mu, lb=-Inf, ub=Inf)
```

Arguments

`lambda`	shape parameter.
`alpha`	shape parameter (must be strictly larger than absolute value of `beta`).
`beta`	asymmetry (shape) parameter.
`delta`	scale parameter (must be strictly positive).
`mu`	location parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The generalized hyperbolic distribution with parameters λ, α, β, δ, and μ has density

$$f(x) = \kappa \, (\delta^2 + (x-\mu)^2)^{1/2(\lambda-1/2)} \cdot \exp(\beta(x-\mu)) \cdot K_{\lambda-1/2}\left(\alpha\sqrt{\delta^2+(x-\mu)^2}\right)$$

where the normalization constant is given by

$$\kappa = \frac{\left(\sqrt{\alpha^2-\beta^2}/\delta\right)^{\lambda}}{\sqrt{2\pi}\,\alpha^{\lambda-1/2}\,K_{\lambda}\left(\delta\sqrt{\alpha^2-\beta^2}\right)}$$

$K_\lambda(t)$ is the modified Bessel function of the third kind with index λ.

Notice that $\alpha > |\beta|$ and $\delta > 0$.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

Barndorff-Nielsen, O., Blaesild, P., 1983. Hyperbolic distributions. In: Johnson, N. L., Kotz, S., Read, C. B. (Eds.), Encyclopedia of Statistical Sciences. Vol. 3. Wiley, New York, p. 700–707.

Prause, K., 1997. Modelling financial data using generalized hyperbolic distributions. FDM preprint 48, University of Freiburg.

Prause, K., 1999. The generalized hyperbolic model: Estimation, financial derivatives, and risk measures. Ph.D. thesis, University of Freiburg.

See Also

unuran.cont.

Examples

```
## Create distribution object for generalized hyperbolic distribution
distr <- udghyp(lambda=-1.0024, alpha=39.6, beta=4.14, delta=0.0118, mu=-0.000158)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udgig *UNU.RAN object for Generalized Inverse Gaussian distribution*

Description

Create UNU.RAN object for a Generalized Inverse Gaussian distribution. Two parametrizations are available.

[Distribution] – Generalized Inverse Gaussian.

Usage

```
udgig(theta, psi, chi, lb=0, ub=Inf)
udgiga(theta, omega, eta=1, lb=0, ub=Inf)
```

Arguments

`theta`	shape parameter.
`psi, chi`	shape parameters (must be strictly positive).
`omega, eta`	shape parameters (must be strictly positive).
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The generalized inverse Gaussian distribution with parameters θ, ψ, and χ has density proportional to

$$f(x) = x^{\theta-1} \exp\left(-\frac{1}{2}\left(\psi x + \frac{\chi}{x}\right)\right)$$

where $\psi > 0$ and $\chi > 0$.

An alternative parametrization used parameters θ, ω, and η and has density proportional to

$$f(x) = x^{\theta-1} \exp\left(-\frac{\omega}{2}\left(\frac{x}{\eta} + \frac{\eta}{x}\right)\right)$$

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Note

These two parametrizations can be converted into each other by means of the following transformations:

$$\psi = \frac{\omega}{\eta}, \quad \chi = \omega\eta$$

$$\omega = \sqrt{\chi\psi}, \quad \eta = \sqrt{\frac{\chi}{\psi}}$$

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 15, p. 284.

See Also

`unuran.cont`.

Examples

```
## Create distribution object for GIG distribution
distr <- udgig(theta=3, psi=1, chi=1)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udgumbel *UNU.RAN object for Gumbel distribution*

Description

Create UNU.RAN object for a Gumbel (Extreme value type I) distribution location parameter `location` and scale parameter `scale`.

[Distribution] – Gumbel (Extreme value type I).

Usage

```
udgumbel(location=0, scale=1, lb=-Inf, ub=Inf)
```

Arguments

`location`	location parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Gumbel distribution function with location l and scale s is

$$F(x) = \exp(-\exp(-\frac{x-l}{s}))$$

for all x.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 22, p. 2.

See Also

unuran.cont.

Examples

```
## Create distribution object for Gumbel distribution
distr <- udgumbel()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udhyper *UNU.RAN object for Hypergeometric distribution*

Description

Create UNU.RAN object for a Hypergeometric distribution with parameters m, n, and k.

[Distribution] – Hypergeometric.

Usage

```
udhyper(m, n, k, lb=max(0,k-n), ub=min(k,m))
```

Arguments

m	the number of white balls in the urn.
n	the number of black balls in the urn.
k	the number of balls drawn from the urn.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Hypergeometric distribution is used for sampling *without* replacement. The density of this distribution with parameters m, n and k (named Np, $N - Np$, and n, respectively in the reference below) is given by

$$p(x) = \binom{m}{x}\binom{n}{k-x} \Big/ \binom{m+n}{k}$$

for $x = 0, \ldots, k$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.discr".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and A.W. Kemp (1992): Univariate Discrete Distributions. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 6, p. 237.

Examples

```
## Create distribution object for Hypergeometric distribution
dist <- udhyper(m=15,n=5,k=7)
## Generate generator object; use method DGT (inversion)
gen <- dgtd.new(dist)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udhyperbolic *UNU.RAN object for Hyperbolic distribution*

Description

Create UNU.RAN object for a Hyperbolic distribution with location parameter mu, tail (shape) parameter alpha, asymmetry (shape) parameter beta, and scale parameter delta.

[Distribution] – Hyperbolic.

Usage

```
udhyperbolic(alpha, beta, delta, mu, lb=-Inf, ub=Inf)
```

Arguments

alpha	tail (shape) parameter (must be strictly larger than absolute value of beta).
beta	asymmetry (shape) parameter.
delta	scale parameter (must be strictly positive).
mu	location parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The hyperbolic distribution with parameters μ,α,β, and δ has density proportional to

$$f(x) = \exp(-\alpha\sqrt{(\delta^2 + (x-\mu)^2)} + \beta * (x-\mu))$$

where $\alpha > |\beta|$ and $\delta > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

See Also

unuran.cont.

Examples

```
## Create distribution object for hyperbolic distribution
distr <- udhyperbolic(alpha=3,beta=2,delta=1,mu=0)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udig *UNU.RAN object for Inverse Gaussian distribution*

Description

Create UNU.RAN object for a Inverse Gaussian (Wald) distribution with mean mu and shape parameter lambda.

[Distribution] – Inverse Gaussian (Wald).

Usage

```
udig(mu, lambda, lb=0, ub=Inf)
```

Arguments

mu	mean (strictly positive).
lambda	shape parameter (strictly positive).
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The inverse Gaussian distribution with mean μ and shape parameter λ has density

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}} \exp(-\frac{\lambda(x-\mu)^2}{2\mu^2 x})$$

where $\mu > 0$ and $\lambda > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 15, p. 259.

See Also

unuran.cont.

Examples

```
## Create distribution object for inverse Gaussian distribution
distr <- udig(mu=3, lambda=2)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udlaplace *UNU.RAN object for Laplace distribution*

Description

Create UNU.RAN object for a Laplace (double exponential) distribution with location parameter location and scale parameter scale.

[Distribution] – Laplace (double exponential).

Usage

```
udlaplace(location=0, scale=1, lb = -Inf, ub = Inf)
```

Arguments

location	location parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Laplace distribution with location l and scale s has density

$$f(x) = \exp(-\frac{|x-l|}{s})$$

for all x.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 24, p. 164.

See Also

unuran.cont.

Examples

```
## Create distribution object for standard Laplace distribution
distr <- udlaplace()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

udlnorm *UNU.RAN object for Log Normal distribution*

Description

Create UNU.RAN object for a Log Normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`.

[Distribution] – Log Normal.

Usage

```
udlnorm(meanlog=0, sdlog=1, lb=0, ub=Inf)
```

Arguments

`meanlog`	mean of the distribution on the log scale.
`sdlog`	standard deviation of the distribution on the log scale.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The log normal distribution has density

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp -(\log(x) - \mu)^2/(2sigma^2)$$

where μ is the mean and σ the standard deviation of the logarithm.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 14, p. 207.

See Also

`unuran.cont`.

Examples

```
## Create distribution object for log normal distribution
distr <- udlnorm()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udlogarithmic *UNU.RAN object for Logarithmic distribution*

Description

Create UNU.RAN object for a Logarithmic distribution with shape parameter shape.

[Distribution] – Logarithmic.

Usage

```
udlogarithmic(shape, lb = 1, ub = Inf)
```

Arguments

shape	shape parameter. Must be between 0 and 1.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Logarithmic distribution with parameters shape $= \theta$ has density

$$f(x) = -\log(1-\theta)\theta^x/x$$

for $x = 1, 2, \ldots$ and $0 < \theta < 1$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.discr".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and A.W. Kemp (1992): Univariate Discrete Distributions. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 7, p. 285.

See Also

unuran.discr.

Examples

```
## Create distribution object for Logarithmic distribution
dist <- udlogarithmic(shape=0.3)
## Generate generator object; use method DARI
gen <- darid.new(dist)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udlogis *UNU.RAN object for Logistic distribution*

Description

Create UNU.RAN object for a Logistic distribution with parameters `location` and `scale`.

[Distribution] – Logistic.

Usage

```
udlogis(location=0, scale=1, lb=-Inf, ub=Inf)
```

Arguments

`location`	location parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Logistic distribution with `location` $= \mu$ and `scale` $= \sigma$ has distribution function

$$F(x) = \frac{1}{1 + e^{-(x-\mu)/\sigma}}$$

and density

$$f(x) = \frac{1}{\sigma} \frac{e^{(x-\mu)/\sigma}}{(1 + e^{(x-\mu)/\sigma})^2}$$

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 23, p. 115.

See Also

unuran.cont.

Examples

```
## Create distribution object for standard logistic distribution
distr <- udlogis()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udlomax *UNU.RAN object for Lomax distribution*

Description

Create UNU.RAN object for a Lomax distribution (Pareto distribution of second kind) with shape parameter shape and scale parameter scale.

[Distribution] – Lomax (Pareto of second kind).

Usage

```
udlomax(shape, scale=1, lb=0, ub=Inf)
```

Arguments

shape	(strictly positive) shape parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Lomax distribution with parameters shape $= \alpha$ and scale $= \sigma$ has density

$$f(x) = \alpha\sigma^{\alpha}(x+\sigma)^{-(\alpha+1)}$$

for $x \geq 0$, $\alpha > 0$ and $\sigma > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 20, p. 575.

See Also

unuran.cont.

Examples

```
## Create distribution object for Lomax distribution
distr <- udlomax(shape=2)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udmeixner *UNU.RAN object for Meixner distribution*

Description

Create UNU.RAN object for a Meixner distribution with scale parameter alpha, asymmetry (shape) parameter beta, shape parameter delta and location parameter mu.

[Distribution] – Meixner.

Usage

```
udmeixner(alpha, beta, delta, mu, lb=-Inf, ub=Inf)
```

Arguments

alpha	scale parameter (must be strictly positive).
beta	asymmetry (shape) parameter (must be larger than $-\pi$ and smaller than π).
delta	shape parameter (must be strictly positive).
mu	location parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Mexiner distribution with parameters α, β, δ, and μ has density

$$f(x) = \kappa \, \exp(\beta(x-\mu)/\alpha) \, |\Gamma\left(\delta + i(x-\mu)/\alpha\right)|^2$$

where the normalization constant is given by

$$\kappa = \frac{\left(2\cos\left(\beta/2\right)\right)^{2\delta}}{2\alpha\pi\,\Gamma\left(2\delta\right)}$$

The symbol i denotes the imaginary unit, that is, we have to evaluate the gamma function $\Gamma(z)$ for complex arguments $z = x + i\,y$.

Notice that $\alpha > 0$, $|\beta| < \pi$ and $\delta > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Kemal Dingec <unuran@statmath.wu.ac.at>.

References

Grigelionis, B., 1999. Processes of Meixner type. Lithuanian Mathematical Journal, Vol. 39, p. 33–41.

Schoutens, W., 2001. The Meixner Processes in Finance. Eurandom Report 2001-002, Eurandom, Eindhoven.

See Also

unuran.cont.

Examples

```
## Create distribution object for meixner distribution
distr <- udmeixner(alpha=0.0298, beta=0.1271, delta=0.5729, mu=-0.0011)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

udnbinom *UNU.RAN object for Negative Binomial distribution*

Description

Create UNU.RAN object for a Negative Binomial distribution with parameters `size` and `prob`.

[Distribution] – Negative Binomial.

Usage

```
udnbinom(size, prob, lb = 0, ub = Inf)
```

Arguments

`size`	target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive.
`prob`	probability of success in each trial. `0 < prob <= 1`.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Negative Binomial distribution with `size` $= n$ and `prob` $= p$ has density

$$p(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!}p^n(1-p)^x$$

for $x = 0, 1, 2, \ldots$, $n > 0$ and $0 < p \leq 1$. This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.discr"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and A.W. Kemp (1992): Univariate Discrete Distributions. 2nd edition, John Wiley & Sons, Inc., New York. Sect. 5.1, p. 200.

See Also

`unuran.discr`.

Examples

```
## Create distribution object for Negative Binomial distribution
dist <- udnbinom(size=100, prob=0.33)
## Generate generator object; use method DARI
gen <- darid.new(dist)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udnorm *UNU.RAN object for Normal distribution*

Description

Create UNU.RAN object for a Normal (Gaussian) distribution with mean equal to `mean` and standard deviation to `sd`.

[Distribution] – Normal (Gaussian).

Usage

```
udnorm(mean=0, sd=1, lb=-Inf, ub=Inf)
```

Arguments

`mean`	mean of distribution.
`sd`	standard deviation.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The normal distribution with mean μ and standard deviation σ has density

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

where μ is the mean of the distribution and σ the standard deviation.

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 13, p. 80.

See Also

unuran.cont.

Examples

```
## Create distribution object for standard normal distribution
distr <- udnorm()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)

## Create distribution object for positive normal distribution
distr <- udnorm(lb=0, ub=Inf)
## ... and draw a sample
gen <- pinvd.new(distr)
x <- ur(gen,100)
```

---

udpareto *UNU.RAN object for Pareto distribution*

Description

Create UNU.RAN object for a Pareto distribution (of first kind) with shape parameters k and a.

[Distribution] – Pareto (of first kind).

Usage

```
udpareto(k, a, lb=k, ub=Inf)
```

Arguments

k	(strictly positive) shape and location parameter.
a	(strictly positive) shape parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Pareto distribution with parameters k and a has density

$$f(x) = ak^a x^{-(a+1)}$$

for $x \geq k$, $k > 0$ and $a > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 20, p. 574.

See Also

unuran.cont.

Examples

```
## Create distribution object for Pareto distribution
distr <- udpareto(k=3,a=2)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udpois *UNU.RAN object for Poisson distribution*

Description

Create UNU.RAN object for a Poisson distribution with parameter lambda.

[Distribution] – Poisson.

Usage

```
udpois(lambda, lb = 0, ub = Inf)
```

Arguments

lambda	(non-negative) mean.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Poisson distribution has density

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

for $x = 0, 1, 2, \ldots$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.discr".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and A.W. Kemp (1992): Univariate Discrete Distributions. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 4, p. 151.

See Also

unuran.discr.

Examples

```
## Create distribution object for Poisson distribution
dist <- udpois(lambda=2.3)
## Generate generator object; use method DARI
gen <- darid.new(dist)
## Draw a sample of size 100
x <- ur(gen,100)
```

udpowerexp *UNU.RAN object for Powerexponential distribution*

Description

Create UNU.RAN object for a Powerexponential (Subbotin) distribution with shape parameter shape.

[Distribution] – Powerexponential (Subbotin).

Usage

```
udpowerexp(shape, lb=-Inf, ub=Inf)
```

Arguments

shape	(strictly positive) shape parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Powerexponential distribution with parameter shape $= \tau$ has density

$$f(x) = \frac{1}{2\Gamma(1+1/\tau)} \exp(-|x|^\tau)$$

for all x and $\tau > 0$. (Here $\Gamma(\alpha)$ is the function implemented by R's gamma() and defined in its help.)

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Note

This function is wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 24, p. 195.

See Also

unuran.cont.

Examples

```
## Create distribution object for powerexponential distribution
distr <- udpowerexp(shape=4)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udrayleigh *UNU.RAN object for Rayleigh distribution*

Description

Create UNU.RAN object for a Rayleigh distribution with scale parameter scale.

[Distribution] – Rayleigh.

Usage

```
udrayleigh(scale=1, lb=0, ub=Inf)
```

Arguments

scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Rayleigh distribution with scale parameter scale $= \sigma$ has density

$$f(x) = \frac{1}{sigma^2} x \exp(-\frac{1}{2}(\frac{x}{sigma})^2)$$

for $x \geq 0$ and $\sigma > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 18, p. 456.

See Also

unuran.cont.

Examples

```
## Create distribution object for standard Rayleigh distribution
distr <- udrayleigh()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udslash *UNU.RAN object for Slash distribution*

Description

Create UNU.RAN object for a Slash distribution.

[Distribution] – Slash.

Usage

```
udslash(lb=-Inf, ub=Inf)
```

Arguments

lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The slash distribution has density

$$f(x) = \frac{1}{\sqrt{2\pi}}(1 - exp(-x^2/2))/x^2$$

for $x \neq 0$ and $\frac{1}{\sqrt{2\pi}}$ otherwise. It is the distribution of the ratio of a unit normal variable to an independent standard uniform (0,1) variable.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 12, p. 63.

See Also

unuran.cont.

Examples

```
## Create distribution object for a slash distribution
distr <- udslash()
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udt *UNU.RAN object for Student t distribution*

Description

Create UNU.RAN object for a Student t distribution with with df degrees of freedom.

[Distribution] – t (Student).

Usage

```
udt(df, lb=-Inf, ub=Inf)
```

Arguments

df	degrees of freedom (strictly positive). Non-integer values allowed.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The t distribution with `df` $= \nu$ degrees of freedom has density

$$f(x) = \frac{\Gamma((\nu+1)/2)}{\sqrt{\pi\nu}\Gamma(\nu/2)}(1+x^2/\nu)^{-(\nu+1)/2}$$

for all real x. It has mean 0 (for $\nu > 1$) and variance $\frac{\nu}{\nu-2}$ (for $\nu > 2$).

The domain of the distribution can be truncated to the interval (`lb`,`ub`).

Value

An object of class `"unuran.cont"`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 28, p. 362.

See Also

`unuran.cont`.

Examples

```
## Create distribution object for t distribution
distr <- udt(df=4)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udvg *UNU.RAN object for Variance Gamma distribution*

Description

Create UNU.RAN object for a Variance Gamma distribution with shape parameter `lambda`, shape parameter `alpha`, asymmetry (shape) parameter `beta`, and location parameter `mu`.

[Distribution] – Variance Gamma.

Usage

```
udvg(lambda, alpha, beta, mu, lb=-Inf, ub=Inf)
```

Arguments

lambda	shape parameter (must be strictly positive).
alpha	shape parameter (must be strictly larger than absolute value of beta).
beta	asymmetry (shape) parameter.
mu	location parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The variance gamma distribution with parameters λ, α, β, and μ has density

$$f(x) = \kappa \, |x-\mu|^{\lambda-1/2} \cdot \exp(\beta(x-\mu)) \cdot K_{\lambda-1/2}\left(\alpha|x-\mu|\right)$$

where the normalization constant is given by

$$\kappa = \frac{\left(\alpha^2-\beta^2\right)^{\lambda}}{\sqrt{\pi}\,(2\alpha)^{\lambda-1/2}\,\Gamma\left(\lambda\right)}$$

$K_\lambda(t)$ is the modified Bessel function of the third kind with index λ. $\Gamma(t)$ is the Gamma function.

Notice that $\alpha > |\beta|$ and $\lambda > 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Note

For $\lambda \leq 0.5$, the density has a pole at μ.

Author(s)

Josef Leydold and Kemal Dingec <unuran@statmath.wu.ac.at>.

References

Eberlein, E., von Hammerstein, E. A., 2004. Generalized hyperbolic and inverse Gaussian distributions: limiting cases and approximation of processes. In Seminar on Stochastic Analysis, Random Fields and Applications IV, Progress in Probability 58, R. C. Dalang, M. Dozzi, F. Russo (Eds.), Birkhauser Verlag, p. 221–264.

Madan, D. B., Seneta, E., 1990. The variance gamma (V.G.) model for share market returns. Journal of Business, Vol. 63, p. 511–524.

Raible, S., 2000. L\'evy Processes in Finance: Theory, Numerics, and Empirical Facts. Ph.D. thesis, University of Freiburg.

See Also

unuran.cont.

Examples

```
## Create distribution object for variance gamma distribution
distr <- udvg(lambda=2.25, alpha=210.5, beta=-5.14, mu=0.00094)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

udweibull *UNU.RAN object for Weibull distribution*

Description

Create UNU.RAN object for a Weibull (Extreme value type III) distribution with with parameters shape and scale.

[Distribution] – Weibull (Extreme value type III).

Usage

```
udweibull(shape, scale=1, lb=0, ub=Inf)
```

Arguments

shape	(strictly positive) shape parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Weibull distribution with shape parameter a and scale parameter σ has density given by

$$f(x) = (a/\sigma)(x/\sigma)^{a-1} \exp(-(x/\sigma)^a)$$

for $x \geq 0$.

The domain of the distribution can be truncated to the interval (lb,ub).

Value

An object of class "unuran.cont".

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap. 21, p. 628.

See Also

unuran.cont.

Examples

```
## Create distribution object for Weibull distribution
distr <- udweibull(shape=3)
## Generate generator object; use method PINV (inversion)
gen <- pinvd.new(distr)
## Draw a sample of size 100
x <- ur(gen,100)
```

---

unuran-class *Class "unuran" – Universal Non-Uniform RANdom variate generators*

Description

The class unuran provides an interface to the UNU.RAN library for universal non-uniform random number generators. It uses the R built-in uniform random number generator.

[Advanced] – UNU.RAN generator object.

Objects from the Class

Objects can be created by calls of the form new("unuran",distribution,method).

distribution: A character string that describes the target distribution (see UNU.RAN User Manual) or one of the S4 classes unuran.cont, unuran.discr, or unuran.cmv that holds information about the distribution.

method: A character string that describes the chosen generation method, see UNU.RAN User Manual. If omitted method "auto" (automatic) is used.

See unuran.new for short introduction and examples for this interface.

Methods

The class unuran provides the following methods for handling objects:

ur signature(object = "unuran"): Get a random sample from the stream object.

r signature(object = "unuran"): Same as ur.

initialize signature(.Object = "unuran"): Initialize unuran object. (For Internal usage only).

print signature(x = "unuran"): Print info about unuran object.

show signature(x = "unuran"): Same as print.

Warning

unuran objects cannot be saved and restored in later R sessions, nor is it possible to copy such objects to different nodes in a computer cluster.

However, unuran objects for *some* generation methods can be "packed", see unuran.packed. Then these objects can be handled like any other R object (and thus saved and restored).

All other objects **must** be **newly created** in a new R session! (Using a restored object does not work as the unuran is then broken.)

Note

The interface has been changed compared to the DSC 2003 paper.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

J. Leydold and W. H\"ormann (2000-2007): UNU.RAN User Manual, see https://statmath.wu.ac.at/unuran/.

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

G. Tirler and J. Leydold (2003): Automatic Nonuniform Random Variate Generation in R. In: K. Hornik and F. Leisch, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20–22, Vienna, Austria.

See Also

unuran.new and ur for faster creation and sampling routines, unuran.details for a more verbose version of show.

unuran.packed allows to pack *some* unuran objects.

For distribution objects see unuran.cont, unuran.discr, and unuran.cmv.

---

unuran.cmv-class *Class "unuran.cmv" for Continuous Multivariate Distribution*

Description

Class unuran.cmv provides an interface to UNU.RAN objects for continuous multivariate distributions. The interface might be changed in future releases. **Do not use unnamed arguments!**

[Advanced] – Continuous Multivariate Distribution Object.

Details

Create a new instance of a `unuran.cmv` object using

```
new ("unuran.cmv",dim=1,pdf=NULL,ll=NULL,ur=NULL,mode=NULL,center=NULL,name=NA).
```

dim number of dimensions of the distribution. (integer)

pdf probability density function. (R function)

ll,ur lower left and upper right vertex of a rectangular domain of the `pdf`. The domain is only set if both vertices are not `NULL`. Otherwise, the domain is unbounded by default. (numeric vectors)

mode location of the mode. (numeric vector – optional)

center point in "typical" region of distribution, e.g. the approximate location of the mode. It is used by several methods to locate the main region of the distribution. If omitted the `mode` is implicitly used. If the `mode` is not given either, the origin is used. (numeric vector – optional)

name name of distribution. (string)

The user is responsible that the given informations are consistent. It depends on the chosen method which information must be given / are used. It is important, that the mode is contained in the (closure of the) domain.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

J. Leydold and W. H\"ormann (2000-2007): UNU.RAN User Manual, see https://statmath.wu.ac.at/unuran/.

See Also

`unuran.cmv.new`, `unuran.new`, `unuran`.

Examples

```
## Create distribution with given PDF
mvpdf <- function (x) { exp(-sum(x^2)) }
mvdist <- new("unuran.cmv", dim=2, pdf=mvpdf)

## Restrict domain to rectangle [0,1]x[0,1] and set
## mode to (0,0)
mvpdf <- function (x) { exp(-sum(x^2)) }
mvdist <- new("unuran.cmv", dim=2, pdf=mvpdf, ll=c(0,0), ur=c(1,1), mode=c(0,0))
```

`unuran.cmv.new` *Create a UNU.RAN continuous multivariate distribution object*

Description

Create a new UNU.RAN object for a continuous multivariate distribution. The interface might be changed in future releases. **Do not use unnamed arguments!**

[Advanced] – Continuous Multivariate Distribution.

Usage

```
unuran.cmv.new(dim=1, pdf=NULL, ll=NULL, ur=NULL,
               mode=NULL, center=NULL, name=NA)
```

Arguments

`dim`	number of dimensions of the distribution. (integer)
`pdf`	probability density function. (R function)
`ll,ur`	lower left and upper right vertex of a rectangular domain of the `pdf`. The domain is only set if both vertices are not `NULL`. Otherwise, the domain is unbounded by default. (numeric vectors)
`mode`	location of the mode. (numeric vector – optional)
`center`	point in "typical" region of distribution, e.g. the approximate location of the mode. It is used by several methods to locate the main region of the distribution. If omitted the `mode` is implicitly used. If the `mode` is not given either, the origin is used. (numeric vector – optional)
`name`	name of distribution. (string)

Details

Creates an instance of class `unuran.cmv`.

The user is responsible that the given informations are consistent. It depends on the chosen method which information must be given / are used.

Note

`unuran.cmv.new(...)` is an alias for `new("unuran.cmv",...)`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

See Also

unuran.cmv, unuran.new, unuran.

Examples

```
## Get a distribution object with given pdf and mode
mvpdf <- function (x) { exp(-sum(x^2)) }
mvd <- unuran.cmv.new(dim=2, pdf=mvpdf, mode=c(0,0))

## Restrict domain to rectangle [0,1]x[0,1] and set
## mode to (0,0)
mvpdf <- function (x) { exp(-sum(x^2)) }
mvd <- unuran.cmv.new(dim=2, pdf=mvpdf, ll=c(0,0), ur=c(1,1), mode=c(0,0))
```

---

unuran.cont-class *Class "unuran.cont" for Continuous Distribution*

Description

Class unuran.cont provides an interface to UNU.RAN objects for continuous distributions. The interface might be changed in future releases. **Do not use unnamed arguments!**

[Advanced] – Continuous Distribution Object.

Details

Create a new instance of a unuran.cont object using

```
new ("unuran.cont",cdf=NULL,pdf=NULL,dpdf=NULL,islog=FALSE,lb=NA,ub=NA,mode=NA,center=NA,area=NA,nam
```

cdf cumulative distribution function. (R function)

pdf probability density function. (R function)

dpdf derivative of the pdf. (R function)

islog whether the given cdf and pdf are given by their logarithms (the dpdf is then the derivative of the logarithm). (boolean)

lb lower bound of domain; use -Inf if unbounded from left. (numeric)

ub upper bound of domain; use Inf if unbounded from right. (numeric)

mode mode of distribution. (numeric)

center typical point ("center") of distribution. If not given the mode is used. (numeric)

area area below pdf; used for computing normalization constants if required. (numeric)

name name of distribution. (string)

The user is responsible that the given informations are consistent. It depends on the chosen method which information must be given / are used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

J. Leydold and W. H\"ormann (2000-2007): UNU.RAN User Manual, see https://statmath.wu.ac.at/unuran/.

See Also

unuran.cont.new, unuran.new, unuran.

Examples

```
## Create continuous distribution with given logPDF and its derivative
pdf <- function (x) { -0.5*x^2 }
dpdf <- function (x) { -x }
distr <- new("unuran.cont", pdf=pdf, dpdf=dpdf, islog=TRUE, lb=-Inf, ub=Inf)
```

---

unuran.cont.new *Create a UNU.RAN continuous univariate distribution object*

Description

Create a new UNU.RAN object for a continuous univariate distribution. The interface might be changed in future releases. **Do not use unnamed arguments!**

[Advanced] – Continuous Distribution.

Usage

```
unuran.cont.new( cdf=NULL, pdf=NULL, dpdf=NULL, islog=FALSE,
                 lb=NA, ub=NA, mode=NA, center=NA, area=NA, name=NA)
```

Arguments

cdf	cumulative distribution function. (R function)
pdf	probability density function. (R function)
dpdf	derivative of the pdf. (R function)
islog	whether the given cdf and pdf are given by their logarithms (the dpdf is then the derivative of the logarithm). (boolean)
lb	lower bound of domain; use -Inf if unbounded from left. (numeric)
ub	upper bound of domain; use Inf if unbounded from right. (numeric)
mode	mode of distribution. (numeric)
center	typical point (“center”) of distribution. If not given the mode is used. (numeric)

area	area below pdf; used for computing normalization constants if required. (numeric)
name	name of distribution. (string)

Details

Creates an instance of class unuran.cont.

The user is responsible that the given informations are consistent. It depends on the chosen method which information must be given / are used.

Note

unuran.cont.new(...) is an alias for new("unuran.cont",...).

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

See Also

unuran.cont, unuran.new, unuran.

Examples

```
## Get a distribution object with given pdf, domain and mode
mypdf <- function (x) { exp(-x) }
distr <- unuran.cont.new(pdf=mypdf, islog=FALSE, lb=0, ub=Inf, mode=0)

## This object can now be used to create an generator object.
## 1. select a method using a Runuran function:
gen <- pinvd.new(distr, uresolution=1e-12)

## 2. directly use the UNU.RAN string API
gen <- unuran.new(distr, method="pinv; u_resolution=1e-12")
```

`unuran.details` *Information on a given "unuran" generator object*

Description

Prints type of `unuran` generator, data used from distribution, parameter for algorithm, performance characteristic, and hints to adjust the performance of the generator. It also returns a list that contains some of these data.

[Advanced] – Print object.

Usage

```
unuran.details(unr, show=TRUE, return.list=FALSE, debug=FALSE)
```

Arguments

`unr`	an `unuran` object.
`show`	whether the data are printed on the console. (boolean)
`return.list`	whether some of the data are returned in a list. (boolean)
`debug`	if TRUE, store additional data in returned list. This might be useful to examine a method. (boolean)

Details

If `show` is `TRUE` then this routine prints data about the generator object to the console.

If `return.list` is `TRUE` then a list that contains some of these data is returned. This is an experimental feature and components of the list may be extended in future releases.

The components of the returned list depend on the particular method. However, the following are common to all objects:

`method` string that contains the name of the generation method.

`type` one of the following strings that describes the type of the generation method:

- `"inv"` inversion method
- `"ar"` acceptance-rejection method
- `"iar"` acceptance-rejection whether inversion is used for the proposal distribution
- `"mcmc"` Markov chain Monte Carlo sampler
- `"other"` none of the above methods

`distr.class` one of the following strings that describes the class of the distribution:

- `"cont"` univariate continuous distribution
- `"discr"` univariate discrete distribution
- `"cont"` multivariate continuous distribution

In addition the following components may be available:

`area.pdf` area below density function of the distribution.

`area.hat` area below hat function for an acceptance-rejection method.

`rejection.constant` rejection constant for an acceptance-rejection method. It given as the ratio `area.hat / area.pdf`.

`area.squeeze` area below squeeze function for an acceptance-rejection method. `area.hat / area.squeeze` can be used as upper bound for the rejection constant.

`intervals` integer that contains the number of subintervals into which the domain of the target distribution is split for constructing a hat function / approximating function.

`truncated.domain` vector of length 2 that contains upper and lower boundary of the 'computational domain' that is used for constructing an approximating function.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

See Also

unuran.

Examples

```
## Create a generator object
distr <- udnorm()
gen <- tdrd.new(distr)

## print data about object on console
unuran.details(gen)

## get list with some of these data
data <- unuran.details(gen,return.list=TRUE)
```

---

unuran.discr-class *Class "unuran.discr" for Discrete Distribution*

Description

Class `unuran.discr` provides an interface to UNU.RAN objects for discrete distributions. The interface might be changed in future releases. **Do not use unnamed arguments!**

[Advanced] – Discrete Distribution Object.

Details

Create a new instance of a `unuran.discr` object using

```
new ("unuran.discr",cdf=NULL,pv=NULL,pmf=NULL,lb=NA,ub=NA,mode=NA,sum=NA,name=NA).
```

cdf cumulative distribution function. (R function)

pv probability vector. (numeric vector)

pmf probability mass function. (R function)

lb lower bound of domain; use `-Inf` if unbounded from left. (numeric, integer)

ub upper bound of domain; use `Inf` if unbounded from right; when `pmf` is not given, the default `ub=Inf` is used. (numeric, integer)

mode mode of distribution. (numeric, integer)

sum sum over `pv` / `pmf`; used for computing normalization constants if required. (numeric)

name name of distribution. (string)

The user is responsible that the given informations are consistent. It depends on the chosen method which information must be given / are used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

J. Leydold and W. H\"ormann (2000-2007): UNU.RAN User Manual, see https://statmath.wu.ac.at/unuran/.

See Also

unuran.discr.new, unuran.new, unuran.

Examples

```
## Create discrete distribution with given probability vector
## (the PV need not be normalized)
pv <- c(1.,2.,1.5,0.,3.,1.2)
dpv <- new("unuran.discr", pv=pv, lb=1)

## Create discrete distribution with given PMF
pmf <- function(x) dbinom(x,100,0.3)
dpmf <- new("unuran.discr",pmf=pmf,lb=0,ub=100)
```

`unuran.discr.new` *Create a UNU.RAN discrete univariate distribution object*

Description

Create a new UNU.RAN object for a discrete univariate distribution. The interface might be changed in future releases. **Do not use unnamed arguments!**

[Advanced] – Discrete Distribution.

Usage

```
unuran.discr.new(cdf=NULL, pv=NULL, pmf=NULL, lb=NA, ub=NA,
                 mode=NA, sum=NA, name=NA)
```

Arguments

`cdf`	cumulative distribution function. (R function)
`pv`	probability vector. (numeric vector)
`pmf`	probability mass function. (R function)
`mode`	mode of distribution. (numeric, integer)
`lb`	lower bound of domain; use `-Inf` if unbounded from left. (numeric, integer)
`ub`	upper bound of domain; use `Inf` if unbounded from right; when `pmf` is not given, the default `ub=Inf` is used. (numeric, integer)
`sum`	sum over `pv` / `pmf`; used for computing normalization constants if required. (numeric)
`name`	name of distribution. (string)

Details

Creates an instance of class `unuran.discr`. For more details see also `unuran.new`.

The user is responsible that the given informations are consistent. It depends on the chosen method which information must be given / are used.

Note

`unuran.discr.new(...)` is an alias for `new("unuran.discr",...)`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

See Also

unuran.discr, unuran.new, unuran.

Examples

```
## Create a distribution object with given PV and mode
mypv <- dbinom(0:100,100,0.3)
distr <- new("unuran.discr", pv=mypv, lb=0, mode=30)

## Create discrete distribution with given probability vector
## (the PV need not be normalized)
pv <- c(1.,2.,1.5,0.,3.,1.2)
dpv <- new("unuran.discr", pv=pv, lb=1)

## Create discrete distribution with given PMF
pmf <- function(x) dbinom(x,100,0.3)
dpmf <- new("unuran.discr", pmf=pmf, lb=0, ub=100)
```

---

unuran.distr-class *Virtual class "unuran.distr"*

Description

The virtual class unuran.distr provides an interface to UNU.RAN objects for distributions.

The following classes extend this class:

class unuran.cont for univariate continuous distributions, see unuran.cont.

class unuran.discr for discrete distributions, see unuran.discr.

class unuran.cmv for multivariate continuous distributions, see unuran.cmv.

[Advanced] – Distribution Object.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

J. Leydold and W. H\"ormann (2000-2007): UNU.RAN User Manual, see https://statmath.wu.ac.at/unuran/.

See Also

unuran.cont, unuran.discr, unuran.cmv; unuran.

`unuran.is.inversion` *Test whether a "unuran" generator object implements an inversion method*

Description

Test whether a given `unuran` generator object implements an inversion method.

[Advanced] – Test type of method.

Usage

```
unuran.is.inversion(unr)
```

Arguments

`unr`	a `unuran` object.

Details

A `unuran` object may implement quite a couple of generation methods. This function tests whether the method used in `unr` is an (approximate) inversion method.

It returns `TRUE` when `unr` implements an inversion method and `FALSE` otherwise.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

See Also

`unuran`.

Examples

```
## PINV is an inversion method
unr <- pinvd.new(udnorm())
unuran.is.inversion(unr)

## TDR is a rejection method
unr <- tdrd.new(udnorm())
unuran.is.inversion(unr)
```

`unuran.new` *Create a UNU.RAN object*

Description

Create a new `unuran` object in package **Runuran** that can be used for sampling from the specified distribution. The function `ur` can then be used to draw a random sample.

[Advanced] – Create generator object.

Usage

```
unuran.new(distr,method="auto")
```

Arguments

`distr`	a string or an S4 class describing the distribution.
`method`	a string describing the random variate generation method.

Details

This function creates an instance of S4 class `unuran` which contains a generator for the target distribution. This distribution has to be provided as an instance of S4 class `unuran.distr`. Depending on the type of distribution such an instance can be created by

`unuran.cont.new` for univariate continuous distributions,

`unuran.discr.new` for discrete distributions, and

`unuran.cmv.new` for multivariate continuous distributions.

The generation can be chosen by passing `method` to the UNU.RAN String API. The default method, `"auto"` tries to find an appropriate method for the given distribution. However, this method is experimental and is yet not very powerfull.

Once a `unuran` object has been created it can be used to draw random samples from the target distribution using `ur`.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

See Also

See unuran for the UNU.RAN class of generators. See unuran.details for printing details about the generator object, and ur and uq for sampling and quantile function, respectively.

For distribution objects see unuran.cont, unuran.discr, and unuran.cmv.

runif, .Random.seed about random number generation in R.

Examples

```
## Use method 'TDR' (Transformed Density Rejection) to
## draw a sample of size 10 from a hyperbolic distribution with PDF
##   f(x) = const * exp(-sqrt(1+x^2))
## restricted to domain [-1,2].

## We first have to define functions that return the log-density and
## its derivative, respectively. (We also could use the density itself.)
lf  <- function (x) { -sqrt(1+x^2) }
dlf <- function (x) { -x/sqrt(1+x^2) }

## Next create the continuous distribution object.
d <- unuran.cont.new(pdf=lf,dpdf=dlf,islog=TRUE,lb=-1,ub=2)

## Create 'unuran' object. We choose method 'TDR' with
## immediate acceptance (IA) and parameter c=0.
gen <- unuran.new(distr=d, method="tdr; variant_ia; c=0")

## Now we can use this object to draw the sample.
## (Of course we can repeat this step as often as required.)
ur(gen,10)

## Here is some information about our generator object.
unuran.details(gen)
```

---

unuran.packed-method *Pack "unuran" object*

Description

Pack unuran object in package **Runuran** and report its status (packed/unpacked).

Packed unuran objects can be saved and loaded or sent to other nodes in a computer cluster (which is not possible for unpacked object).

FIXME

Usage

```
## S4 method for signature 'unuran'
unuran.packed(unr)
unuran.packed(unr) <- value
```

Arguments

unr	a unuran object.
value	TRUE to pack the object.

Details

A unuran object contains a pointer to an external object in library UNU.RAN. Thus it cannot be saved and restored in later R sessions, nor is it possible to copy such an object to different nodes in a computer cluster.

By "packing" an unuran object all required data are copied from the external object into an R list and stored in the unuran object while the external UNU.RAN object is destroyed. Thus the object can be handled like any other R object. Moreover, it can be still used as argument for ur and uq (which may have even faster execution times then). Packed unuran objects cannot be unpacked any more.

Notice that currently only objects that implement method 'PINV' can be packed.

Methods

Currently only objects that implement method 'PINV' can be packed.

Note

Note that due to limitations of floating point arithmetic the output of a uq call with the same input value for u may slightly differ for the packed and unpacked version.

See Also

unuran, pinv.new.

Examples

```
## create a unuran object for half-normal distribution using methed 'PINV'
gen <- pinv.new(dnorm,lb=0,ub=Inf)

## status of object
unuran.packed(gen)

## draw a random sample of size 10
x <- ur(gen,10)

## pack unuran object
unuran.packed(gen) <- TRUE
unuran.packed(gen)

## draw a random sample of size 10
x <- ur(gen,10)

## Not run:
## unpacking is not supported
unuran.packed(gen) <- FALSE    ## results in error
```

```
## End(Not run)
```

`unuran.verify.hat` *Verify hat and squeezes in a "unuran" generator object*

Description

Verify hat function and squeezes in a given unuran generator that implements a rejection method.

[Advanced] – Verify rejection method.

Usage

```
unuran.verify.hat(unr, n=1e5, show=TRUE)
```

Arguments

unr	an unuran object.
n	sample size. (integer)
show	whether the result is printed on the console. (boolean)

Details

UNU.RAN is a library of so called black-box algorithms. For algorithms based on the rejection method this means that hat and squeezes are created automatically during the setup. Obviously not all algorithms work for all distribution. Then usually the setup fails (which is good, since then one does not silently obtain a random sample from a distribution other then the requested.)

Although we have tested these algorithms with a lot of distributions (including those with extreme properties) there is still some (minor) chance that hat and squeezes are computed without any warnings, but are incorrect, i.e., the inequalities

$$squeeze(x) \leq density(x) \leq hat(x)$$

are not satisfied for all x. This might happen due to serious round-off errors for densities with extreme properties (e.g., sharp and narrow peaks). But it also might be caused by some incorrect additional information about the distribution given by the user which has not been detected by various checks during the setup. If one is unsure about his or her chosen generation method one can check these inequalities.

Routine `unuran.verify.hat` allows to run generator `unr` and check whether the two inequalities are violated. This is done for every point x that is sampled from the hat distribution. This includes also those points that are rejected. The function counts the occurrences of such evaluations and returns the ratio of this number and the sample size `n`. (It is thus a little bit too high since the total number of generated but rejected points is not known.) Yet, it does not provide any information about the magnitude of violation of the inequality.

If `show` is `TRUE` then this routine prints this ratio and some diagnostcs to the console.

Routine `unuran.verify.hat` does not work for algorithms that do not implement a rejection method.

Value

Ratio of number occurrences where the hat and squeezes violate the inequality and the sample size.

Note

Due to round-off errors there might exist a few points where the ratio $density(x)/hat(x)$ is slightly larger than 1. In our experiments we observed a few cases where this ratio was as large as $1 + 10^{-8}$ for some points although we could proof (using real numbers instead of floating point numbers) that hat and squeeze are computed correctly.

On the other hand, there are cases where, due to the limitation of floating point arithmetic, it is not possible to sample from the target distribution at all. The Gamma distribution with extremely small shape parameter, say 0.0001, is such an example. Then the continuous Gamma distribution degenerates to a point distribution with only a few points with significant mass.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

See Also

unuran.

Examples

```
## Create a generator object that implements a rejection method
unr <- tdrd.new(udnorm())

## Verify hat and squeeze
unuran.verify.hat(unr)
```

---

up *Distribution function for "unuran" object*

Description

Evaluates the (approximate) cumulative distribution function (CDF) of a "unuran" object for a continuous or discrete distribution.

Usage

```
up(obj, x)
```

Arguments

obj	one of • a distribution object of class "unuran.cont" that contains the CDF, or • a distribution object of class "unuran.discr" that contains the CDF, or • a generator object (class "unuran") that contains a CDF or implements method 'PINV'.
x	vector of x values. (numeric)

Details

The routine evaluates the cumulative distribution function of a distribution stored in a UNU.RAN distribution object or UNU.RAN generator object.

For the computation of the CDF the following alternatives are tried (in the given order):

1. The CDF is available in object obj: the function is evaluated and the result is returned.

   Important: In this case routine up just evaluates the CDF but ignores the boundaries of the domain of the distribution, i.e., it does not return 0 and 1, resp., outside the domain unless the implementation of the CDF handles this case correctly. This behavior is in particular important when **Runuran** built-in distributions are truncated by explicitly setting the domain boundaries.

2. Object obj is a generator object that implements method 'PINV': In this case an approximate value for the CDF is returned. The approximation error is about one tenth of the requested uresolution for method 'PINV'.

3. Neither the CDF nor its approximation is available in object obj: NA is returned and a warning is thrown.

Note

The generator object must not be packed (see unuran.packed).

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

See Also

unuran.cont, unuran.discr, unuran, pinv.new.

Examples

```
## Create an UNU.RAN distribution object (for standard Gaussian)
## and evaluate distribution function for some points
distr <- udnorm()
up(distr, 1.5)
up(distr, -3:3)

## Create an UNU.RAN generator object (for standard Gaussian)
## and evaluate distribution function of underyling distribution
unr <- tdrd.new(udnorm())
up(unr, 1.5)
up(unr, -3:3)

## Create an UNU.RAN generator object that does not contain
## the CDF but implements method PINV.
unr <- pinv.new(pdf=function(x){exp(-x)}, lb=0,ub=Inf)
up(unr, 0:5)
```

---

uq *Quantile function for "unuran" object*

Description

Evaluates quantile of distribution approximately using a unuran object that implements an inversion method.

[Universal] – Quantile Function.

Usage

```
uq(unr, U)
```

Arguments

unr	a unuran object that implements an inversion menthod.
U	vector of probabilities.

Details

The routine evaluates the quantiles (inverse CDF) for a given (vector of) probabilities approximately. It requires a unuran object that implements an inversion method. Currently these are

- 'HINV'
- 'NINV'
- 'PINV'

for continuous distributions and

- 'DGT'

for discrete distributions.

uq returns the left boundary of the domain of the distribution if argument U is less than or equal to 0 and the right boundary if U is greater than or equal to 1.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

See Also

unuran,unuran.new.

Examples

```
## Compute quantiles of normal distribution using method 'PINV'
gen <- pinv.new(pdf=dnorm, lb=-Inf, ub=Inf)
uq(gen,seq(0,1,0.05))

## Compute quantiles of user-defined distribution using method 'PINV'
pdf <- function (x) { exp(-x) }
gen <- pinv.new(pdf=pdf, lb=0, ub=Inf, uresolution=1.e-12)
uq(gen,seq(0,1,0.05))

## Compute quantiles of binomial distribution using method 'DGT'
gen <- dgt.new(pv=dbinom(0:1000,1000,0.4), from=0)
uq(gen,seq(0,1,0.05))


## Compute quantiles of normal distribution using method 'HINV'
## (using 'advanced' interface)
gen <- unuran.new("normal()","hinv")
uq(gen,0.975)
uq(gen,c(0.025,0.975))

## Compute quantiles of user-defined distributio using method 'HINV'
## (using 'advanced' interface)
cdf <- function (x) { 1.-exp(-x) }
pdf <- function (x) { exp(-x) }
dist <- new("unuran.cont", cdf=cdf, pdf=pdf, lb=0, ub=Inf)
gen <- unuran.new(dist, "hinv; u_resolution=1.e-12")
uq(gen,seq(0,1,0.05))
```

ur *Sample from a distribution specified by a "unuran" object*

Description

Get random sample from a unuran object in package **Runuran**.

[Universal] – Sampling Function.

Usage

```
ur(unr, n=1)
unuran.sample(unr, n=1)
```

Arguments

unr	a unuran object.
n	sample size.

Note

unuran.sample is just an (older) longer name for ur.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class.

Examples

```
## Draw random sample of size 10 from normal distribution using
## method 'TDR'
unr <- unuran.new("normal","tdr")
x <- ur(unr,n=10)
```

urbeta *UNU.RAN Beta random variate generator*

Description

UNU.RAN random variate generator for the Beta distribution with parameters shape1 and shape2. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Beta.

Usage

```
urbeta(n, shape1, shape2, lb = 0, ub = 1)
```

Arguments

n	size of required sample.
shape1, shape2	positive shape parameters of the Beta distribution.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Beta distribution with parameters shape1 $= a$ and shape2 $= b$ has density

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^a (1-x)^b$$

for $a > 0$, $b > 0$ and $0 \leq x \leq 1$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Beta distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R. Compared to rbeta, urbeta is faster, especially for larger sample sizes. However, in opposition to rbeta vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rbeta for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urbeta(n=1000,shape1=2,shape2=5)
```

---

urbinom *UNU.RAN Binomial random variate generator*

Description

UNU.RAN random variate generator for the Binomial distribution with parameters size and prob. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Binomial.

Usage

```
urbinom(n, size, prob, lb = 0, ub = size)
```

Arguments

n	size of required sample.
size	number of trials (one or more).
prob	probability of success on each trial.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Binomial distribution with size $= n$ and prob $= p$ has density

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

for $x = 0, \ldots, n$.

The generation algorithm uses guide table based inversion. The parameters lb and ub can be used to generate variates from the Binomial distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to rbinom, urbinom is faster, especially for larger sample sizes. However, in opposition to rbinom vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rbinom for the R built-in generator.

Examples

```
## Create a sample of size 1000 from the binomial distribution
x <- urbinom(n=1000,size=10,prob=0.3)
```

---

urburr *UNU.RAN Burr random variate generator*

Description

UNU.RAN random variate generator for the Burr distribution with shape1 and shape2. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Burr.

Usage

```
urburr(n, a, b, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
a,b	positive shape parameters of the Burr distribution.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Burr distribution with parameters a and b has density

$$f(x) = a(b-1)x^{a-1}/(1+x^a)^b$$

for $x \geq 0$, $a \geq 1$ and $b \geq 2$.

The generation algorithm uses transformed density rejection 'TDR'. The parameters lb and ub can be used to generate variates from the Burr distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urburr(n=1000,a=2,b=3)
```

---

urcauchy *UNU.RAN Cauchy random variate generator*

Description

UNU.RAN random variate generator for the Cauchy distribution with location parameter location and scale parameter scale. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Cauchy.

Usage

```
urcauchy(n, location=0, scale=1, lb = -Inf, ub = Inf)
```

Arguments

n	size of required sample.
location	location parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

If `location` or `scale` are not specified, they assume the default values of `0` and `1` respectively.

The Cauchy distribution with location l and scale s has density

$$f(x) = \frac{1}{\pi s}\left(1 + \left(\frac{x-l}{s}\right)^2\right)^{-1}$$

for all x.

The generation algorithm uses fast numerical inversion. The parameters `lb` and `ub` can be used to generate variates from the Cauchy distribution truncated to the interval (`lb`,`ub`).

Note

This function is wrapper for the UNU.RAN class in R. Compared to `rcauchy`, `urcauchy` is faster, especially for larger sample sizes. However, in opposition to `rcauchy` vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation, `unuran` for the UNU.RAN class, and `rcauchy` for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urcauchy(n=1000)
```

`urchi` *UNU.RAN Chi random variate generator*

Description

UNU.RAN random variate generator for the Chi (χ) distribution with `df` degrees of freedom. It also allows sampling from the truncated distribution. (Do not confuse with the Chi-Squared (χ^2) distribution!)

[Special Generator] – Sampling Function: Chi.

Usage

```
urchi(n, df, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
df	degrees of freedom (strictly positive, but can be non-integer).
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Chi distribution with df$= n > 0$ degrees of freedom has density

$$f(x) = x^{n-1} e^{-x^2/2}$$

for $x > 0$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Chi distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap.18, p.417.

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urchi(n=1000,df=3)
```

urchisq *UNU.RAN Chi-Squared random variate generator*

Description

UNU.RAN random variate generator for the Chi-Squared (χ^2) distribution with df degrees of freedom. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Chi-squared.

Usage

```
urchisq(n, df, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
df	degrees of freedom (strictly positive, but can be non-integer).
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Chi-squared distribution with df$= n > 0$ degrees of freedom has density

$$f_n(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2}$$

for $x > 0$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Chi-squared distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R. Compared to rchisq, urchisq is faster, especially for larger sample sizes. However, in opposition to rchisq vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rchisq for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urchisq(n=1000,df=3)
```

---

urdgt *UNU.RAN discrete random variate generator*

Description

UNU.RAN random variate generator for discrete distributions with given probability vector. It applies the Guide-Table Method (urdgt) for discrete inversion or the Alias-Urn Method (urdau).

[**Deprecated**. Use dgt.new or dau.new instead.]

Usage

```
urdgt(n, probvector, from = 0, by = 1)
urdau(n, probvector, from = 0, by = 1)
```

Arguments

n	size of required sample.
probvector	vector of non-negative numbers (need not sum to 1).
from	number corresponding to the first probability in probvector.
by	"from + (k-1)*by" is the number corresponding to the k-th probability in probvector.

Details

These routines generate a sample of discrete random variates with given probability vector. This vector must be provided by probvector, a vector of non-negative numbers which need not necessarily sum up to 1.

Method 'DGT' uses a guide-table based inversion method.

Method 'DAU' implements the *Alias-Urn* method.

Both methods run in constant time, i.e., the marginal sampling times do not depend on the length of the given probability vector. Whereas their setup times grow linearly with this length.

Note

urdgt() and urdau() are very fast for probvector not longer than about 1000.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

H.C. Chen and Y. Asau (1974): On generating random variates from an empirical distribution. AIIE Trans. 6, pp.163–166.

A.J. Walker (1977): An efficient method for generating discrete random variables with general distributions. ACM Trans. Model. Comput. Simul. 3, pp.253–256.

See Also

dau.new, dgt.new, and ur for replacement.

---

urexp *UNU.RAN Exponential random variate generator*

Description

UNU.RAN random variate generator for the Exponential distribution with rate rate (i.e., mean 1/rate). It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Exponential.

Usage

```
urexp(n, rate=1, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
rate	(strictly positive) rate parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

If rate is not specified, it assumes the default value of 1.

The Exponential distribution with rate λ has density

$$f(x) = \lambda e^{-\lambda x}$$

for $x \geq 0$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Exponential distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to `rexp`, `urexp` is faster, especially for larger sample sizes. However, in opposition to `rexp` vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation, `unuran` for the UNU.RAN class, and `rexp` for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urexp(n=1000)
```

`urextremeI` *UNU.RAN Extreme value type I (Gumbel-type) random variate generator*

Description

UNU.RAN random variate generator for the Extreme value type I (Gumbel-type) distribution with location parameter `location` and scale parameter `scale`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Gumbel (extreme value type I).

Usage

```
urextremeI(n, location=0, scale=1, lb=-Inf, ub=Inf)
```

Arguments

`n`	size of required sample.
`location`	location parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

If `location` or `scale` are not specified, they assume the default values of `0` and `1` respectively.

The Gumbel distribution with location l and scale s has distribution function

$$F(x) = \exp(-\exp(-\frac{x-l}{s}))$$

for all x.

The generation algorithm uses fast numerical inversion. The parameters `lb` and `ub` can be used to generate variates from the Gumbel distribution truncated to the interval (`lb`,`ub`).

Note

This function is wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap.22, p.2.

See Also

`runif` and `.Random.seed` about random number generation and `unuran` for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urextremeI(n=1000)
```

`urextremeII` *UNU.RAN Extreme value type II (Frechet-type) random variate generator*

Description

UNU.RAN random variate generator for the Extreme value type II (Frechet-type) distribution with shape parameter `shape`, location parameter `location` and scale parameter `scale`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Frechet (extreme value type II).

Usage

```
urextremeII(n, shape, location=0, scale=1, lb=location, ub=Inf)
```

Arguments

n	size of required sample.
shape	(strictly positive) shape parameter.
location	location parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

If location or scale are not specified, they assume the default values of 0 and 1 respectively.

The Frechet distribution function with shape k, location l and scale s is

$$F(x) = \exp(-(\frac{x-l}{s})^{-k})$$

for $x \geq l$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Frechet distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap.22, p.2.

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urextremeII(n=1000,shape=2)
```

urf *UNU.RAN F random variate generator*

Description

UNU.RAN random variate generator for the F distribution with with df1 and df2 degrees of freedom. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: F.

Usage

```
urf(n, df1, df2, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
df1, df2	(strictly positive) degrees of freedom. Non-integer values allowed.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The F distribution with df1 = n_1 and df2 = n_2 degrees of freedom has density

$$f(x) = \frac{\Gamma(n_1/2+n_2/2)}{\Gamma(n_1/2)\Gamma(n_2/2)} \left(\frac{n_1}{n_2}\right)^{n_1/2} x^{n_1/2-1} \left(1+\frac{n_1 x}{n_2}\right)^{-(n_1+n_2)/2}$$

for $x > 0$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the F distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R. Compared to rf, urf is faster, especially for larger sample sizes. However, in opposition to rf vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rf for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urf(n=1000,df1=3,df2=5)
```

---

urgamma *UNU.RAN Gamma random variate generator*

Description

UNU.RAN random variate generator for the Gamma distribution with parameters shape and scale. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Gamma.

Usage

```
urgamma(n, shape, scale=1, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
shape	(strictly positive) shape parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution
ub	upper bound of (truncated) distribution

Details

If scale is omitted, it assumes the default value of 1.

The Gamma distribution with parameters shape $= \alpha$ and scale $= \sigma$ has density

$$f(x) = \frac{1}{\sigma^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\sigma}$$

for $x \geq 0$, $\alpha > 0$ and $\sigma > 0$. (Here $\Gamma(\alpha)$ is the function implemented by R's gamma() and defined in its help.)

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Gamma distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to `rgamma`, `urgamma` is faster, especially for larger sample sizes. However, in opposition to `rgamma` vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation, `unuran` for the UNU.RAN class, and `rgamma` for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urgamma(n=1000,shape=2)
```

---

urgeom *UNU.RAN Geometric random variate generator*

Description

UNU.RAN random variate generator for the Geometric distribution with parameter `prob`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Geometric.

Usage

```
urgeom(n, prob, lb = 0, ub = Inf)
```

Arguments

`n`	size of required sample.
`prob`	probability of success in each trial. `0 < prob <= 1`.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Geometric distribution with `prob` $= p$ has density

$$p(x) = p(1-p)^x$$

for $x = 0, 1, 2, \ldots$, $0 < p \leq 1$.

The generation algorithm uses guide table based inversion for $p > 0.02$ and method 'DARI' otherwise. The parameters `lb` and `ub` can be used to generate variates from the Geometric distribution truncated to the interval (`lb`,`ub`).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to `rgeom`, `urgeom` is faster, especially for larger sample sizes. However, in opposition to `rgeom` vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation, `unuran` for the UNU.RAN class, and `rgeom` for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urgeom(n=1000,prob=0.2)
```

---

urgig — *UNU.RAN Generalized Inverse Gaussian Distribution variate generator*

Description

UNU.RAN random variate generator for the Generalized Inverse Gaussian Distribution with parameters `lambda` and `omega`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: GIG (generalized inverse Gaussian).

Usage

```
urgig(n, lambda, omega, lb=1.e-12, ub=Inf)
```

Arguments

n	size of required sample.
lambda	(strictly positive) shape parameter.
omega	(strictly positive) shape parameter.
lb	lower bound of (truncated) distribution
ub	upper bound of (truncated) distribution

Details

The Generalized Inverse Gaussian distribution with parameters lambda $= \lambda$ and omega $= \omega$ has a density proportional to

$$f(x) \sim x^{\lambda-1} \exp(-(\omega/2)(x + 1/x))$$

for $x \geq 0$, $\lambda > 0$ and $\omega > 0$.

The generation algorithm uses transformed density rejection 'TDR'. The parameters lb and ub can be used to generate variates from the distribution truncated to the interval (lb,ub).

The generation algorithm works for $\lambda \geq 1$ and $\omega > 0$ and for $\lambda > 0$ and $\omega \geq 0.5$.

Note

This function is wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap.15, p.284.

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urgig(n=1000,lambda=2,omega=3)
```

urhyper *UNU.RAN Hypergeometric random variate generator*

Description

UNU.RAN random variate generator for the Hypergeometric distribution. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Hypergeometric.

Usage

```
urhyper(nn, m, n, k, lb=max(0,k-n), ub=min(k,m))
```

Arguments

nn	number of observations.
m	the number of white balls in the urn.
n	the number of black balls in the urn.
k	the number of balls drawn from the urn.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Hypergeometric distribution is used for sampling *without* replacement. The density of this distribution with parameters m, n and k (named Np, $N - Np$, and n, respectively in the reference below) is given by

$$p(x) = \binom{m}{x}\binom{n}{k-x} \Big/ \binom{m+n}{k}$$

for $x = 0, \ldots, k$.

The generation algorithm uses guide table based inversion. The parameters lb and ub can be used to generate variates from the Hypergeometric distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to rhyper, urhyper is faster, especially for larger sample sizes. However, in opposition to rhyper vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation, `unuran` for the UNU.RAN class, and `rhyper` for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urhyper(nn=20,m=15,n=5,k=7)
```

`urhyperbolic` *UNU.RAN Hyperbolic random variate generator*

Description

UNU.RAN random variate generator for the Hyperbolic distribution with parameters `shape` and `scale`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Hyperbolic.

Usage

```
urhyperbolic(n, shape, scale=1, lb = -Inf, ub = Inf)
```

Arguments

`n`	size of required sample.
`shape`	(strictly positive) shape parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

If `scale` is omitted, it assumes the default value of `1`.

The Hyperbolic distribution with parameters `shape` $= \alpha$ and `scale` $= \sigma$ has density proportional to

$$f(x) \sim \exp(-\alpha\sqrt{1+(\frac{x}{s})^2})$$

for all x, $\alpha > 0$ and $\sigma > 0$.

The generation algorithm uses transformed density rejection 'TDR'. The parameters `lb` and `ub` can be used to generate variates from the Hyperbolic distribution truncated to the interval (`lb`,`ub`).

Note

This function is wrapper for the UNU.RAN class in R.

Do not confuse with `rhyper`

that samples from the (discrete) *hypergeometric* distribution.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation and `unuran` for the UNU.RAN class.

Examples

```
## Create a sample of size 1000 from Hyperbolic distribution with shape=3
x <- urhyperbolic(n=1000,shape=3)
```

---

urlaplace *UNU.RAN Laplace random variate generator*

Description

UNU.RAN random variate generator for the Laplace (double exponential) distribution with location parameter `location` and scale parameter `scale`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Laplace.

Usage

```
urlaplace(n, location=0, scale=1, lb = -Inf, ub = Inf)
```

Arguments

`n`	size of required sample.
`location`	location parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

If `location` or `scale` are not specified, they assume the default values of `0` and `1` respectively.

The Laplace distribution with location l and scale s has density

$$f(x) = \exp(-\frac{|x-l|}{s})$$

for all x.

The generation algorithm uses fast numerical inversion. The parameters `lb` and `ub` can be used to generate variates from the Laplace distribution truncated to the interval (`lb`,`ub`).

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap.24, p.164.

See Also

`runif` and `.Random.seed` about random number generation and `unuran` for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urlaplace(n=1000)
```

`urlnorm` *UNU.RAN Log-Normal random variate generator*

Description

UNU.RAN random variate generator for the Log-Normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Log-Normal.

Usage

```
urlnorm(n, meanlog=0, sdlog=1, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
meanlog, sdlog	mean and standard deviation of the distribution on the log scale. If not not specified they assume the default values of 0 and 1, respectively.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Log-Normal distribution has density

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

where μ and σ are the mean and standard deviation of the logarithm.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Log-Normal distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R. Compared to rlnorm, urlnorm is faster, especially for larger sample sizes. However, in opposition to rlnorm vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rlnorm for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urlnorm(n=1000)
```

`urlogarithmic` *UNU.RAN Logarithmic random variate generator*

Description

UNU.RAN random variate generator for the Logarithmic distribution with shape parameter `shape`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Logarithmic.

Usage

```
urlogarithmic(n, shape, lb = 1, ub = Inf)
```

Arguments

`n`	size of required sample.
`shape`	shape parameter. Must be between 0 and 1.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Logarithmic distribution with parameters `shape` $= \theta$ has density

$$f(x) = -\log(1-\theta)\theta^x/x$$

for $x = 1, 2, \ldots$ and $0 < \theta < 1$.

The generation algorithm uses guide table based inversion when the tails are not too heavy and method 'DARI' otherwise. The parameters `lb` and `ub` can be used to generate variates from the Logarithmic distribution truncated to the interval (`lb`,`ub`).

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg.

N.L. Johnson, S. Kotz, and A.W. Kemp (1992): Univariate Discrete Distributions, 2nd edition. John Wiley & Sons, Inc., New York. Chap.7, p.285.

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urlogarithmic(n=1000,shape=0.3)
```

---

urlogis *UNU.RAN Logistic random variate generator*

Description

UNU.RAN random variate generator for the Logistic distribution with parameters location and scale. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Logistic.

Usage

```
urlogis(n, location=0, scale=1, lb=-Inf, ub=Inf)
```

Arguments

n	size of required sample.
location	location parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

If location or scale are omitted, they assume the default values of 0 and 1 respectively.

The Logistic distribution with location $= \mu$ and scale $= \sigma$ has distribution function

$$F(x) = \frac{1}{1 + e^{-(x-\mu)/\sigma}}$$

and density

$$f(x) = \frac{1}{\sigma} \frac{e^{(x-\mu)/\sigma}}{(1 + e^{(x-\mu)/\sigma})^2}$$

The generation algorithm uses inversion. The parameters lb and ub can be used to generate variates from the Logistic distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to `rlogis`, `urlogis` is faster, especially for larger sample sizes. However, in opposition to `rlogis` vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation, `unuran` for the UNU.RAN class, and `rlogis` for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urlogis(n=1000)
```

---

urlomax *UNU.RAN Lomax random variate generator*

Description

UNU.RAN random variate generator for the Lomax distribution (Pareto distribution of second kind) with shape parameter `shape` and scale parameter `scale`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Lomax.

Usage

```
urlomax(n, shape, scale=1, lb=0, ub=Inf)
```

Arguments

`n`	size of required sample.
`shape`	(strictly positive) shape parameter.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

If scale is omitted, it assumes the default value of 1.

The Lomax distribution with parameters shape $= \alpha$ and scale $= \sigma$ has density

$$f(x) = \alpha\sigma^{\alpha}(x+\sigma)^{-(\alpha+1)}$$

for $x \geq 0$, $\alpha > 0$ and $\sigma > 0$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Lomax distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap.20, p.575.

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urlomax(n=1000,shape=2)
```

---

urnbinom *UNU.RAN Negative Binomial random variate generator*

Description

UNU.RAN random variate generator for the Negative Binomial distribution with with parameters size and prob. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Negative Binomial.

Usage

```
urnbinom(n, size, prob, lb = 0, ub = Inf)
```

Arguments

n	size of required sample.
size	target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive.
prob	probability of success in each trial. 0 < prob <= 1.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Negative Binomial distribution with size $= n$ and prob $= p$ has density

$$p(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!}p^n(1-p)^x$$

for $x = 0, 1, 2, \ldots$, $n > 0$ and $0 < p \leq 1$. This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached.

The generation algorithm uses guide table based inversion when the tails are not too heavy and method 'DARI' otherwise. The parameters lb and ub can be used to generate variates from the Negative Binomial distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R. Compared to rnbinom, urnbinom is faster, especially for larger sample sizes. However, in opposition to rnbinom vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rnbinom for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urnbinom(n=1000,size=10,prob=0.3)
```

urnorm *UNU.RAN Normal random variate generator*

Description

UNU.RAN random variate generator for the Normal distribution with mean equal to mean and standard deviation to sd. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Normal (Gaussian).

Usage

```
urnorm(n, mean = 0, sd = 1, lb = -Inf, ub = Inf)
```

Arguments

n	size of required sample.
mean	mean of distribution.
sd	standard deviation.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

If mean or sd are not specified they assume the default values of 0 and 1, respectively.

The normal distribution has density

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

where μ is the mean of the distribution and σ the standard deviation.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Normal distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to rnorm, urnorm is faster, especially for larger sample sizes. However, in opposition to rnorm vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rnorm for the R built-in normal random variate generator.

Examples

```
## Create a sample of size 1000
x <- urnorm(n=1000)
```

---

urpareto *UNU.RAN Pareto random variate generator*

Description

UNU.RAN random variate generator for the Pareto distribution (of first kind) with shape parameters k and a. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Pareto (of first kind).

Usage

```
urpareto(n, k, a, lb=k, ub=Inf)
```

Arguments

n	size of required sample.
k	(strictly positive) shape and location parameter.
a	(strictly positive) shape parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Pareto distribution with parameters k and a has density

$$f(x) = ak^a x^{-(a+1)}$$

for $x \geq k$, $k > 0$ and $a > 0$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Pareto distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap.20, p.574.

See Also

`runif` and `.Random.seed` about random number generation and `unuran` for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urpareto(n=1000,k=2,a=3)
```

`urplanck` *UNU.RAN Planck random variate generator*

Description

UNU.RAN random variate generator for the Planck distribution with shape parameter `a`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Planck.

Usage

```
urplanck(n, a, lb = 1.e-12, ub = Inf)
```

Arguments

`n`	size of required sample.
`a`	(strictly positive) shape parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

The Planck distribution with parameter `a` has density proportional to

$$f(x) \sim \frac{x^a}{\exp(x) - 1}$$

for $x \geq 0$ and $a \geq 1$.

The generation algorithm uses transformed density rejection 'TDR'. The parameters `lb` and `ub` can be used to generate variates from the Planck distribution truncated to the interval (`lb`,`ub`).

Note

This function is wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urplanck(n=1000,a=2)
```

---

urpois *UNU.RAN Poisson random variate generator*

Description

UNU.RAN random variate generator for the Poisson distribution with parameter lambda. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Poisson.

Usage

```
urpois(n, lambda, lb = 0, ub = Inf)
```

Arguments

n	size of required sample.
lambda	(non-negative) mean.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Poisson distribution has density

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

for $x = 0, 1, 2, \ldots$.

The generation algorithm uses guide table based inversion when the tails are not too heavy and method 'DARI' otherwise. The parameters `lb` and `ub` can be used to generate variates from the Poisson distribution truncated to the interval (`lb`,`ub`).

Note

This function is wrapper for the UNU.RAN class in R. Compared to `rpois`, `urpois` is faster, especially for larger sample sizes. However, in opposition to `rpois` vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation, `unuran` for the UNU.RAN class, and `rpois` for the R built-in generator.

Examples

```
## Create a sample of size 1000 from Poisson distribution with lamda=2.3
x <- urpois(n=1000,lambda=2.3)
```

---

urpowerexp *UNU.RAN Powerexponential random variate generator*

Description

UNU.RAN random variate generator for the Powerexponential (Subbotin) distribution with shape parameter `shape`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Powerexponential (Subbotin).

Usage

```
urpowerexp(n, shape, lb = -Inf, ub = Inf)
```

Arguments

n	size of required sample.
shape	(strictly positive) shape parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Powerexponential distribution with parameter shape $= \tau$ has density

$$f(x) = \frac{1}{2\Gamma(1+1/\tau)} \exp(-|x|^\tau)$$

for all x and $\tau > 0$. (Here $\Gamma(\alpha)$ is the function implemented by R's gamma() and defined in its help.)

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Powerexponential distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

N.L. Johnson, S. Kotz, and N. Balakrishnan (1995): Continuous Univariate Distributions, Volume 2. 2nd edition, John Wiley & Sons, Inc., New York. Chap.24, p.195.

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urpowerexp(n=1000, shape=4)
```

urrayleigh *UNU.RAN Rayleigh random variate generator*

Description

UNU.RAN random variate generator for the Rayleigh distribution with scale parameter `scale`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Rayleigh.

Usage

```
urrayleigh(n, scale=1, lb = 0, ub = Inf)
```

Arguments

`n`	size of required sample.
`scale`	(strictly positive) scale parameter.
`lb`	lower bound of (truncated) distribution.
`ub`	upper bound of (truncated) distribution.

Details

If `scale` is omitted, it assumes the default value of `1`.

The Rayleigh distribution with scale parameter `scale` $= \sigma$ has density

$$f(x) = \frac{1}{sigma^2} x \exp(-\frac{1}{2}(\frac{x}{sigma})^2)$$

for $x \geq 0$ and $\sigma > 0$.

The generation algorithm uses fast numerical inversion. The parameters `lb` and `ub` can be used to generate variates from the Rayleigh distribution truncated to the interval (`lb`,`ub`).

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

N.L. Johnson, S. Kotz, and N. Balakrishnan (1994): Continuous Univariate Distributions, Volume 1. 2nd edition, John Wiley & Sons, Inc., New York. Chap.18, p.456.

See Also

runif and .Random.seed about random number generation and unuran for the UNU.RAN class.

Examples

```
## Create a sample of size 1000 from Rayleigh distribution with scale=1
x <- urrayleigh(n=1000,scale=1)
```

---

urt *UNU.RAN Student t random variate generator*

Description

UNU.RAN random variate generator for the Student t distribution with with df degrees of freedom. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: t (Student).

Usage

```
urt(n, df, lb = -Inf, ub = Inf)
```

Arguments

n	size of required sample.
df	degrees of freedom (> 0, maybe non-integer).
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The t distribution with df $= \nu$ degrees of freedom has density

$$f(x) = \frac{\Gamma((\nu+1)/2)}{\sqrt{\pi\nu}\Gamma(\nu/2)}(1+x^2/\nu)^{-(\nu+1)/2}$$

for all real x. It has mean 0 (for $\nu > 1$) and variance $\frac{\nu}{\nu-2}$ (for $\nu > 2$).

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the t distribution truncated to the interval (lb,ub).

Note

This function is a wrapper for the UNU.RAN class in R. Compared to rt, urt is faster, especially for larger sample sizes. However, in opposition to rt vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rt for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urt(n=1000,df=4)
```

---

urtriang *UNU.RAN Triangular random variate generator*

Description

UNU.RAN random variate generator for the Triangular distribution with shape parameters a, m and b. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Triangular.

Usage

```
urtriang(n, a, m, b, lb=a, ub=b)
```

Arguments

n	size of required sample.
a,b	left and right boundary of domain
m	mode of distribution
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Triangular distribution with domain (a, b) and mode m has a density proportional to

$$f(x) \sim (x - a)/(m - a)$$

for $a \leq x \leq m$, and

$$f(x) \sim (b - x)/(b - m)$$

for $m \leq x \leq b$.

The generation algorithm uses fast numerical inversion. The parameters `lb` and `ub` can be used to generate variates from the Triangular distribution truncated to the interval (`lb`,`ub`).

Note

This function is a wrapper for the UNU.RAN class in R.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

`runif` and `.Random.seed` about random number generation and `unuran` for the UNU.RAN class.

Examples

```
## Create a sample of size 1000
x <- urtriang(n=1000,a=-1,m=0,b=2)
```

`urweibull` *UNU.RAN Weibull random variate generator*

Description

UNU.RAN random variate generator for the Weibull distribution with with parameters `shape` and `scale`. It also allows sampling from the truncated distribution.

[Special Generator] – Sampling Function: Weibull.

Usage

```
urweibull(n, shape, scale=1, lb=0, ub=Inf)
```

Arguments

n	size of required sample.
shape	(strictly positive) shape parameter.
scale	(strictly positive) scale parameter.
lb	lower bound of (truncated) distribution.
ub	upper bound of (truncated) distribution.

Details

The Weibull distribution with shape parameter a and scale parameter σ has density given by

$$f(x) = (a/\sigma)(x/\sigma)^{a-1} \exp(-(x/\sigma)^a)$$

for $x \geq 0$.

The generation algorithm uses fast numerical inversion. The parameters lb and ub can be used to generate variates from the Weibull distribution truncated to the interval (lb,ub).

Note

This function is wrapper for the UNU.RAN class in R. Compared to rweibull, urweibull is faster, especially for larger sample sizes. However, in opposition to rweibull vector arguments are ignored, i.e. only the first entry is used.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg

See Also

runif and .Random.seed about random number generation, unuran for the UNU.RAN class, and rweibull for the R built-in generator.

Examples

```
## Create a sample of size 1000
x <- urweibull(n=1000,shape=3)
```

use.aux.urng-method *Use auxiliary random number generator for Runuran objects*

Description

Some UNU.RAN methods that are based on the rejection method can make use of a second auxiliary uniform random number generator. It is only used when a rejection has occurred, see below for details. This allows to keep two streams of random variates (almost) synchronized. This is in particular necessary for variance reduction methods like common or antithetic random variates.

[Advanced] – Use auxiliary URNG for rejection method.

Usage

```
## S4 method for signature 'unuran'
use.aux.urng(unr)
use.aux.urng(unr) <- value
set.aux.seed(seed)
```

Arguments

unr	a unuran generator object.
value	TRUE when an auxiliary URNG is used, FALSE when no auxiliary URNG is used (the default).
seed	seed for the auxiliary URNG.

Details

Variance reduction techniques like common or antithetic random variates require correlated streams of non-uniform random variates. Such streams can be easily created by means of the inversion method using the same source of uniform random numbers (URNs). However, the quantile function (inverse CDF) or even the CDF often is not available in closed form and thus numerical method are required that are expensive or only return approximate random numbers or both.

On the other hand two streams of non-uniformly distributed random variates are completely uncorrelated when the acceptance-rejection method is used. Then the two streams run "out-of-sync" when the first rejection occurs.

Schmeiser and Kachitvichyanukul, however, have shown that this problem can be overcome when the proposal point is generated by inversion and a fixed number k of URNs is used for generating one non-uniform random variate. This can be accomplished by means of a second auxiliary stream of URNs which is used when the required number of URNs exceeds k.

By this approach two streams of non-uniform random variates run synchronized except when a rejection occurs in one of the two streams. Therefore the two generated streams are respected mixtures of highly correlated streams and independent streams. The induced correlation thus decreases when the rejection constants of the acceptance-rejection algorithms used for generating the two streams increases.

In UNU.RAN some of the acceptance-rejection algorithms make use of a second auxiliary stream of URNs. It is implemented in one of the following ways:

- The primary uniform random number generator is used for the first loop of the acceptance-rejection step. When a rejection occurs the algorithms switches to auxiliary generator until the proposal point is accepted. Thus exactly *two* URNs from the primary generator are used to generate one non-uniform random variate.
- The primary uniform random number generator is used just for the first proposal point and then switches to the auxiliary generator until the proposal point is accepted. Thus exactly *one* URN from the primary generator is used to generate one non-uniform random variate.

The call `use.aux.urng(unr)` returns `FALSE` if this feature is disabled for Runuran generator object `unr` (the default) and `TRUE` if this feature is enabled. It auxiliary URNs are not supported at all for object `unr` then `use.aux.urng(unr)` returns `NA`.

The replacement method `use.aux.urng(unr) <-TRUE` enables this feature for generator `unr`. It can be disabled by means of `use.aux.urng(unr) <-FALSE`. (One gets an error if this feature is not supported at all.)

The seed of the auxiliary uniform random number generator can be set by means of `set.aux.seed(seed)`. The auxiliary generator is a combined multiple recursive generator (MRG31k3p) by L'Ecuyer and Touzin and is built into package **Runuran**. Currently it cannot be replaced by some other generator.

Value

`use.aux.urng` returns
`TRUE`, if using the auxiliary generator is enabled,
`FALSE`, it is disabled, and
`NA`, if this feature does not exist at all.

`set.aux.seed` returns `NULL`, invisibly.

Methods

Currently the following UNU.RAN methods support this feature. (Currently the last four methods are only available via `unuran.new`, see the UNU.RAN manual for more details.)

method name	Runuran call	URN per variate
"tdr" (ps)	`tdr.new`	2
"arou" (ia)	–	1
"tabl" (ia=false)	–	2
"tdr" (gw)	–	2
"tdr" (ia)	–	1
inversion		1

Note

Using an auxiliary uniform random number generator generator is only useful if the rejection constant is close to 1.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation, Sect. 8.4.2. Springer-Verlag, Berlin Heidelberg

B. W. Schmeiser and V. Kachitvichyanukul (1986): Correlation induction without the inverse transformation. In: Proc. 1986 Winter Simulation Conf., J. Wilson, J. Henriksen, S. Roberts (eds.), 266-274.

B. W. Schmeiser and V. Kachitvichyanukul (1990): Non-inverse correlation induction: guidelines for algorithm development. J. Comput. Appl. Math. 31, 173-180.

W. H\"ormann and G. Derflinger (1994): Universal generators for correlation induction. In: Compstat, Proceedings in Computational Statistics, R. Dutter and W. Grossmann (eds.), 52-57.

J. Leydold, E. Janka, and W. H\"ormann (2002): Variants of Transformed Density Rejection and Correlation Induction. In: Monte Carlo and Quasi-Monte Carlo Methods 2000, K.-T. Fang, F. Hickernell, and H. Niederreiter (eds.), 345–356.

P. L'Ecuyer and R. Touzin (2000): Fast combined multiple recursive generators with multipliers of the form a = +/- 2^q +/- 2^r. In: J.A. Jones, R.R. Barton, K. Kang, and P.A. Fishwick (eds.), Proc. 2000 Winter Simulation Conference, 683-689.

See Also

tdr.new.

Examples

```
## Create respective generators for normal and exponential distribution.
## Use method TDR
gen1 <- tdrd.new(udnorm())
gen2 <- tdrd.new(udexp())

## The two streams are independent even we use the same seed
set.seed(123); x1 <- ur(gen1,1e5)
set.seed(123); x2 <- ur(gen2,1e5)
cor(x1,x2)

## We can enable the auxiliary URNG and get correlated streams
use.aux.urng(gen1) <- TRUE
use.aux.urng(gen2) <- TRUE
set.seed(123); x1 <- ur(gen1,1e5)
set.seed(123); x2 <- ur(gen2,1e5)
cor(x1,x2)

## This feature can be disabled again
use.aux.urng(gen1)
use.aux.urng(gen1) <- FALSE
use.aux.urng(gen2) <- FALSE

## Notice that TDR cannot simply mixed with an inversion method
## as the number of URNG per random point differs
gen3 <- pinvd.new(udexp())
set.seed(123); x3 <- ur(gen3,1e5)
```

```
cor(x1,x3)

## But a trick would do this
set.seed(123); x3 <- ur(gen3,2*1e5)
x3 <- x3[seq(1,2*1e5,2)]
cor(x1,x3)
## or ...
set.seed(123); u3 <- runif(2*1e5); u3 <- u3[seq(1,2*1e5,2)]
x3 <- uq(gen3,u3)
cor(x1,x3)

## Maybe method AROU is more appropriate
gen4 <- unuran.new(udnorm(), "arou")
use.aux.urng(gen4) <- TRUE
set.seed(123); x3 <- ur(gen3,1e5)
set.seed(123); x4 <- ur(gen4,1e5)
cor(x3,x4)
```

`vnrou.new` *UNU.RAN generator based on Multivariate Naive Ratio-Of-Uniforms method (VNROU)*

Description

UNU.RAN random variate generator for continuous multivariate distributions with given probability density function (PDF). It is based on the Multivariate Naive Ratio-Of-Uniforms method ('VNROU').

[Universal] – Rejection Method.

Usage

```
vnrou.new(dim=1, pdf, ll=NULL, ur=NULL, mode=NULL, center=NULL, ...)
```

Arguments

`dim`	number of dimensions of the distribution. (integer)
`pdf`	probability density function. (R function)
`ll,ur`	lower left and upper right vertex of a rectangular domain of the `pdf`. The domain is only set if both vertices are not `NULL`. Otherwise, the domain is unbounded by default. (numeric vectors)
`mode`	location of the mode. (numeric vector)
`center`	point in "typical" region of distribution, e.g. the approximate location of the mode. If omitted the `mode` is used. If the `mode` is not given either, the origin is used. (numeric vector)
`...`	(optional) arguments for `pdf`

Details

This function creates a unuran object based on the naive ratio-of-uniforms method ('VNROU'), i.e., a bounding rectangle for the acceptance region is estimated and use for sampling proposal points. It can be used to draw samples of a continuous random vector with given probability density function using ur.

The algorithm works with unimodal distributions provided that the tails are not too "high" in every direction.

The density must be provided by a function pdf which must return non-negative numbers and which need not be normalized (i.e., it can be any multiple of a density function).

The center is used as starting point for computing the bounding rectangle. Alternatively, one also could provide the location the mode. However, this requires its exact position whereas center allows any point in the "typical" region of the distribution.

The setup can be accelerated when the mode is given.

Author(s)

Josef Leydold and Wolfgang H\"ormann <unuran@statmath.wu.ac.at>.

References

W. H\"ormann, J. Leydold, and G. Derflinger (2004): Automatic Nonuniform Random Variate Generation. Springer-Verlag, Berlin Heidelberg. Section 11.1.4.

See Also

ur, unuran.new, unuran.

Examples

```
## Create a sample of size 100 for a Gaussian distribution
mvpdf <- function (x) { exp(-sum(x^2)) }
gen <- vnrou.new(dim=2, pdf=mvpdf)
x <- ur(gen,100)

## Use mode of Gaussian distribution to accelerate set-up.
mvpdf <- function (x) { exp(-sum(x^2)) }
gen <- vnrou.new(dim=2, pdf=mvpdf, mode=c(0,0))
x <- ur(gen,100)

## Gaussian distribution restricted to the rectangle [1,2]x[1,2]
##  (don't forget to provide a point inside domain using 'center')
mvpdf <- function (x) { exp(-sum(x^2)) }
gen <- vnrou.new(dim=2, pdf=mvpdf, ll=c(1,1), ur=c(2,2), center=c(1.5,1.5))
x <- ur(gen,100)
```

Index