Package ‘rvgtest’

February 20, 2015

Type Package
Title Tools for Analyzing Non-Uniform Pseudo-Random Variate Generators
Version 0.7.4
Date 2014-02-26
Author Josef Leydold and Sougata Chaudhuri <sgtchaudhuri@gmail.com>
Maintainer Josef Leydold <josef.leydold@wu.ac.at>
Depends R (>= 3.0.0)
Suggests Runuran (>= 0.18), testthat
Description Test suite for non-uniform pseudo-random number generators.
License GPL-2
URL http://statmath.wu.ac.at/unuran/
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-02-26 15:11:17

R topics documented:

rvgtest-package .................................................. 2
Perturbation .......................................................... 5
plot.rvgt.ftable .................................................. 6
plot.rvgt.htest ................................................... 7
plot.rvgt.ierror ................................................... 8
rvgt.chisq .......................................................... 10
rvgt.ftable .......................................................... 11
rvgt.Mtest .......................................................... 14
uerror ............................................................... 15
xerror ............................................................... 18

Index 21
Description

Suite for testing non-uniform random number generators.

Details

Package: rvgtest
Type: Package
Version: 0.7.4
Date: 2014-02-26
License: GPL 2 or later
LazyLoad: yes

rvgtest is a set of tools to investigate the quality of non-uniform pseudo-random random generators (RVG). Thus it provides functions to visualize and test for possible defects. There are three main reasons for such defects and errors:

1. Errors in the design of algorithms – The proof for theorem that claims the correctness of the algorithm is wrong.
2. Implementation errors – Mistakes in computer programs.
3. Limitations of floating point arithmetic and round-off errors in implementations of these algorithms in real world computers.

Of course testing software is a self-evident part of software engineering. Implementation errors usually result in large deviations from the requested distribution and thus errors of type 2 are easily detected. However, this need not always be the case, for example for rather complicated algorithms like those based on patchwork methods.

The same holds for errors of type 1. In the best of all worlds, there exists a correct proof for the validity of the algorithm. In our world however human can err. Then the deviations are rather small, since otherwise it would have been detected when testing the implementation for errors of type 2.

Errors of type 3 can be a problem when the requested distribution has extreme properties. E.g., it is no problem to generate a sample of beta distributed random variates with shape parameters 0.001 using \texttt{rbeta(n=100, shape1=0.001, shape2=0.001)}. However, due the limited resolution of floating point numbers it behaves like a discrete distribution (especially near 1). It is not always obvious whether such round-off errors will influence ones simulation results.

It is the purpose of this package to provide some tools to find possible errors in RVGs. However, observing a defect in (the implementation of) a pseudo-random variate generator by purely statistical tools may require a large sample size which that exceeds the memory when held in a single array in \texttt{R}. (Nevertheless, there is some chance that this defect causes an error in a particular simulation...
with a moderate sample size.) Hence we have implemented routines that can run tests on very large sample sizes (which are only limited by the available runtimes).

Currently there are two toolsets for testing random variate generators for univariate distributions:

1. Testing based on histograms for all kinds of RVGs.
2. Estimating errors of RVGs that are based on numerical inversion methods.

1. Histograms

A frequently used method for testing univariate distributions is based on the following strategy: Draw a sample, compute a histogram and run a goodness-of-fit test on the resulting frequency table.

We have implemented a three step procedure:

1. Create tables that can hold the information of huge random samples.
2. Perform some test for the null hypothesis on these tables.
3. Visualize these tables as well the results of the tests.

The advantages of this procedure are:

- Huge total sample sizes are possible (only limited by available runtime but not by memory).
- Can run multiple tests on the same random sample.
- Inspect data visually.

In addition there are also some random functions for introducing defects in other random variate generators artificially. Thus one may investigate the power of tests.

**List of Routines:**

Data generation: `rvgt.ftable`.


Visualization: `plot` (see `plot.rvgt.ftable`, `plot.rvgt.htest` for the respective syntax of the call).

Perturbation of RVGs: `pertadd`, `pertsb`.

2. Numerical Inversion

Random variate generators that are based on inverting the distribution function preserve the structure of the underlying uniform random number generator. Given the fact that state-of-the-art uniform random number generators are well tested, it is sufficient to estimate (maximal) approximation errors.

Let $G^{-1}$ denote the approximate inverse distribution function (quantile function) and $F$ the (exact) cumulative distribution. Then the following measures for the approximation errors are implemented:

**u-error:** $\epsilon_u(u) = |u - F(G^{-1}(u))|$

**absolute x-error:** $\epsilon_x(u) = |F^{-1}(u) - G^{-1}(u)|$

**relative x-error:** $\epsilon_x(u)/|F^{-1}(u)|$
We are convinced that the u-error is the most convenient measure for the approximation error in
the framework of Monte Carlo simulation. E.g., goodness-of-fit tests like the chi-square test or the
Kolmogorov-Smirnov test look at this deviation.
As for the histogram based tests we have implemented in such a way that sample sizes are not
limited by memory. Again data generation and visualization is separated into to routines.

**List of Routines:** Data generation: `uerror`, `xerror`.
Visualization: `plot` (see `plot.rvgt ierror` for the syntax of the call).

**Author(s)**
Josef Leydold <josef.leydold@wu.ac.at>, Sougata Chaudhuri <sgtchaudhuri@gmail.com>

**Examples**
```r
define a function uerror for u-error calculation
```

```r
# 1. Histogram
# -----------------------------
#
# Use a poor Gaussian random variate generator
# (otherwise we should not see a defect).
# RNGkind(normal.kind="Buggy Kinderman-Ramage")
#
# Create table of bin counts.
# Use a sample of 20 times 100 random variates.
table <- rvgt.ftable(n=1e5, rep=20, rdist=rnorm, qdist=qnorm)
#
# Plot histogram for (cumulated) data
# plot(table)
#
# Perform a chi-square goodness-of-fit test and plot result
# r1 <- rvgt.chisq(table)
# plot(r1)
#
# Perform M-test and plot both results
# r2 <- rvgt.Mtest(table)
# plot.rvgt.htest(list(r1,r2))
#
# -----------------------------
# 2. Numerical Inversion
# -----------------------------
#
# Create a table of u-errors for spline interpolation of
# the inverse CDF of the standard normal distribution.
aqn <- splinefun(x=pnorm((-100:100)*0.05), y=(-100:100)*0.05,
    method="monoH.FC")
#
# Use a sample of size of 10^5 random variates.
uerrn <- uerror(n=1e5, aqdist=aqn, pdist=pnorm)
#
# Plot u-errors
# plot(uerrn)
```
Perturbation

Generate Random Variates from a Perturbed RVG

Description
These functions perturb the distribution of a given random variate generator. Thus it allows to investigate the power of tests for random variate generators.

Usage

pertadd(n, rdist = rnorm, ..., min = 0, max = 1, p = 0.001)
pertsub(n, rdist = rnorm, ..., min = 0, max = 1, p = 0.001)

Arguments

n sample size.
rdist given RVG to be perturbed.
... parameters for given random variate generator.
min, max left and right boundary of perturbed domain.
p strength of perturbation.

Details

pertadd generates random variates from a mixture of rdist and a uniform distribution on the interval (min, max). The uniform distribution is chosen with probability p.
pertsub generates random variates from the rdist but rejects all points in the interval (min, max) with probability p.

By varying the width of uniform distribution (min, max) and probability of error p, different levels of “artificial” error can be introduced. Thus it allows to investigate the power of tests for random variate generators.

Value

A vector of random variates from the perturbed distribution is returned.

Author(s)

Sougata Chaudhuri <sgtchaudhuri@gmail.com>, Josef Leydold <josef.leydold@wu.ac.at>

Examples

## Generating random sample with default settings
x <- pertadd(n=1000)
y <- pertsub(n=1000)

## Generating random sample, with parent distribution as exponential.
plot.rvgt.ftable

x <- pertadd(n=1000, rdist=rexp, rate=2, min=2, max=3, p=0.005)
y <- pertsub(n=1000, rdist=rexp, rate=2, min=2, max=3, p=0.005)

## Conducting chi-square test on random variates generated from function
## pertsub with parent distribution as exponential.
ft <- rvgt.ftable(n=1e4, rep=10,
                   rdist=function(n){pertsub(n, rdist=rexp, p=0.1)},
                   qdist=qexp, breaks=1001)
plot(ft)
rvgt.chisq(ft)

## Conducting chi-square test on random variates generated from function
## pertadd with parent distribution as Weibull, shape=1, scale=2.
ft <- rvgt.ftable(n=1e4, rep=10,
                   rdist=function(n,...){pertadd(n, rdist=rweibull,...,p=0.05)},
                   qdist=qweibull, shape=1, scale=2)
plot(ft)
rvgt.chisq(ft)

plot.rvgt.ftable

Plot RVG Frequency Table

Description

Method for plotting the density of an RVG frequency table.

Usage

## S3 method for class 'rvgt.ftable'
plot(x, rows, alpha = 0.01, ...)

Arguments

x
  object of class "rvgt.ftable".
rows
  integer or array of integers that indicate the rows of the table for which the
density is plotted. If missing all rows are used.
alpha
  significance level for plotting critical values.
...  
  further graphical parameters.

Details

plot creates a density plot for the given RVG frequency table. The display zooms into the union
of the range of frequency values and 2 times the confidence intervals for the frequencies. Thus it
visualizes significant deviations from the uniform distributions. The critical values for significance
level alpha are marked by a red (dashed) line.

rows is either a number or a vector of numbers that indicate the rows of RVG tabel x which are
merged (cumulated) for the plot. If this argument is missing, all rows are merged.
Author(s)
Sougata Chaudhuri <sgtchaudhuri@gmail.com>, Josef Leydold <josef.leydold@wu.ac.at>

See Also
rvgt.ftable for creating the frequency table; hist for the height of the drawn rectangle.

Examples

## Create a frequency table for normal distribution and show histogram.
## Use a sample of size of 5 times 10^5 random variates.
ft <- rvgt.ftable(n=1e5,rep=5, rdist=rnorm,qdist=qnorm, mean=1,sd=2)

## Plot histogram
plot(ft)

## Plot histogram for row 1 only
plot(ft,rows=1)

## Plot histogram for rows 2 and 4 only
plot(ft,rows=c(2,4))

## Same with the buggy random variate generator
## (try to increase sample size 'n' or 'rep')
RNGkind(normal.kind="Buggy Kinderman-Ramage")
ft <- rvgt.ftable(n=1e5,rep=1, rdist=rnorm,qdist=qnorm)
plot(ft)

plot.rvgt.htest

**Plot p-Values against Sample Size**

Description
Method for plotting the p-values of test results on RVG frequency tables.

Usage

```r
## S3 method for class 'rvgt.htest'
plot(x, alpha = 0.001, ...)
```

Arguments

- `x`: an object of class "rvgt.htest" or a list of such objects.
- `alpha`: significance level.
- `...`: further graphical parameters.
Details

Plot common logarithm of p-values stored in an object of class "rvgt.htest" (i.e., p-values of tests on RVG frequency tables). The p-values are plotted against increasing sample size. The significance level alpha is marked by a red dashed line. This allows for visible inspection of test results.

Argument x must be an object of class "rvgt.htest" that contains information about the test and p-values. Such an object is created by rvgt.chisq or rvgt.Mtest.

Alternatively, when the method is called by its full name, plot.rvgt.htest, then the first argument x may also be a list of such objects. Then p-values for multiple experiments will be plotted in the same graph with different colors. Thus one can compare the power of different tests or the results for different generation methods.

Author(s)

Sougata Chaudhuri <sgtchaudhuri@gmail.com>, Josef Leydold <josef.LEYDOLD@wu.ac.at>

See Also

rvgt.chisq, rvgt.Mtest.

Examples

```r
## Create a frequency table for normal distribution and show histogram.
## Use a sample of size of 5 times 10^5 random variates.
table <- rvgt.ftable(n=1e5,rep=5, rdist=rnorm,qdist=qnorm, mean=1,sd=2)

## Perform a chi-square goodness-of-fit test and plot result
r1 <- rvgt.chisq(table)
plot(r1)

## Perform M-test
r2 <- rvgt.Mtest(table)
plot(r2)

## Create a plot that contains the results of both tests
plot.rvgt.htest(list(r1,r2))
```

---

plot.rvgt.ierror  
*Plot Errors in Inversion Methods*

Description

Method for plotting errors in numerical inversion methods.

Usage

```r
## S3 method for class 'rvgt.ierror'
plot(x, maxonly=FALSE, tol=NA, ...)
```
Arguments

x               an object of class "rvgt.ierror" or a list of such objects.
maxonly         logical. If TRUE, only show maximal errors.
tol             maximal tolerated error (optional).
...             further graphical parameters.

Details

Plot errors stored in an object of class "rvgt.ierror". The function plots range, interquartile range, median, and maximum. If maxonly is TRUE, then only the maximal errors are plotted.

If present, the maximal tolerated u-error tol is marked by a blue dashed line. Furthermore, tol also can be used to control the plotting range for the error. The plotting range depends on the maximal observed error. If tol is given, then the range is between 1.1*tol and 10*tol.

Argument x must be an object of class "rvgt.ierror" that contains information about u-errors. Such an object is created by uerror or xerror.

Alternatively, when the method is called by its full name, plot.rvgt.ierror, then the first argument x may also be a list of such objects. Then maximal errors for multiple experiments will be plotted in the same graph with different colors (that is, with maxonly=TRUE). Thus one can compare errors of different inverse distribution functions.

Author(s)

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

See Also

uerror, xerror.

Examples

## Create a table of u-errors for spline interpolation of
## the inverse CDF of the standard normal distribution and
## the beta distribution
aqn <- splinefun(x=pnorm(-100:100)*0.05, y=(-100:100)*0.05,
                    method="monoH.FC")
ueern <- uerror(n=1e5, aqdist=aqn, pdist=pnorm)

aqb <- splinefun(x=pbeta(0:100)*0.01, shape1=2, shape2=5),
             y=(0:100)*0.01, method="monoH.FC")
ueerrb <- uerror(n=1e5, aqdist=aqb, pdist=pbeta, shape1=2, shape2=5)

## Plot u-errors of the normal distribution
plot(ueern)

## Plot maximal u-errors of the normal distribution
plot(ueern, maxonly=TRUE)

## Compare the u-errors of these two distributions and
## draw maximal tolerated error
Perform Chi-Square goodness-of-fit Test on RVG Frequency Table

Description
Perform a chi-square goodness-of-fit test on the RVG frequency table.

Usage
rvgt.chisq(ftable)

Arguments
ftable object of class "rvgt.ftable".

Details
rvgt.chisq performs a chi-square goodness-of-fit test on the bin counts of the RVG frequency tables created by means of rvgt.ftable. The null hypothesis is that the bin counts are distributed according to the expected probabilities (i.e., proportional to the lengths of the histogram cells).
If the ftable contains the bin counts for more than one repetitions then the test is repeated with column sums of increasing number of rows which corresponds to increasing sampling sizes. This allows for getting an idea of the power of the test.

Value
An object of class "rvgt.htest" which is a list with components:

- type character string containing type of test; equals "chisq".
- n sample size for every row in ftable.
- rep number of rows in ftable.
- breaks number of break points in ftable (i.e., number of bins + 1).
- pval vector of p-values.

Author(s)
Sougata Chaudhuri <sgtchaudhuri@gmail.com>, Josef Leydold <josef.leydold@wu.ac.at>

See Also
See plot.rvgt.htest for the syntax of the plotting method.
Examples

```r
## Create a frequency table for normal distribution and
c## conduct a chi-square goodness-of-fit test on this data.
c## Use a sample of size of 5 times 10^5 random variates.
table <- rvgt.ftable(n=1e5, rep=5, rdist=rnorm, qdist=qnorm, mean=1, sd=2)

## Perform test
result <- rvgt.chisq(table)

## Plot result
plot(result)
```

rvgt.ftable

Create RVG Frequency Table for Random Variate Generator

Description

Function for creating frequency tables for random variate generators. Thus a histogram is computed and the bin counts are stored in an array which can be used to visualize possible defects of the pseudo-random variate generator and run goodness-of-fit tests.

The function only works for generators for univariate distributions.

Usage

```r
rvgt.ftable(n, rep=1, rdist, qdist, pdist, ...,
breaks = 101, trunc=NULL, exactu=FALSE, plot=FALSE)
```

Arguments

- `n`: sample size for one repetition (≥ 100).
- `rep`: number of repetitions.
- `rdist`: random variate generator for a univariate distribution.
- `qdist`: quantile function for the distribution.
- `pdist`: cumulative distribution function for distribution.
- `...`: parameters to be passed to `rdist`, `qdist` and `pdist`.
- `breaks`: one of:
  - a single number giving the number of cells of histogram; or
  - a vector giving the breakpoints between histogram cells (in uniform scale).
    Notice that in the latter case the break points are automatically sorted and the first and last entry is set to 0 and 1, resp. Moreover, they must be different from each other.
- `trunc`: boundaries of truncated domain. (optional)
- `exactu`: logical. If TRUE then the exact locations of the given break points are used. Otherwise, these points are slightly shifted in order to accelerate execution time, see details below.
- `plot`: logical. If TRUE, a histogram is plotted.
Details

rvgt.ftable returns tables of bin counts similar to the `hist` function. Bins can be specified either by the number of break points between the cells of the histogram, or by a list of break points in the \( u \)-scale. In the former case the break points are constructed such that all bins of the histogram have equal probability for the distribution under the null hypothesis, i.e., the break points are equidistributed in the \( u \)-scale using the formula \( u_i = i/(breaks - 1) \) where \( i = 0, \ldots, breaks - 1 \).

When the quantile function \( qdist \) is given, then these points are transformed into breaking points in the \( x \)-scale using \( qdist(u_i) \). Thus the histogram can be computed directly for random points \( X \) that are generated by means of \( rdist \).

Otherwise the cumulative distribution function \( pdist \) must be given. If \( exactu \) is \text{TRUE}, then all non-uniform random points \( X \) are first transformed into uniformly distributed random numbers \( U = pdist(X) \) for which the histogram is created. This is slower than directly using \( X \) but it is numerically more robust as round-off error in \( qdist \) have much more influence than those in \( pdist \).

If \( trunc \) is given, then functions \( qdist \) and \( pdist \) are rescaled to this given domain. It is recommended to provide \( pdist \) even when \( qdist \) is given.

If \( exactu \) is \text{FALSE} and the quantile function \( qdist \) is missing, then the first sample of size \( n \) is used to estimate the quantiles for the given break points using function `quantile`. The break points in \( u \)-scale are then recomputed using these quantiles by means of the given probability function \( pdist \). This is usually (much) faster than calling \( pdist \) on each generated point. However, the break points are slightly perturbated (but this does not effect the correctness of the frequency table).

The argument \( rep \) allows to create multiple such arrays of bin counts and store these in one table. Thus has two advantages:

- It allows for huge total sample sizes that would otherwise exceed the available memory, and
- it can be used to visualize test results for increasing sample sizes, or
- allows for a two-level test.

For \textit{discrete} distributions function \( pdist \) must be given and both arguments \( qdist \) and \( exactu \) are ignored. Moreover, the given break points have to be adjusted according to the probability function of the discrete distribution. In particular this means that bins have to be collapsed when the probability of some number is larger than difference of break points in \( u \)-scale. Thus there resulting tables may contain less break points than requested.

The type of distribution (continuous or discrete) is autodetected by the function.

Value

An object of class "rvgt.ftable" which is a list with components:

- \( n \) sample size.
- \( rep \) number of repetitions.
- \( ubreaks \) an array of break points in \( u \)-scale.
- \( xbreaks \) an array of break points in \( x \)-scale.
- \( count \) a matrix of \( rep \) rows and \( (breaks - 1) \) columns that contains the bin counts. The results for each repetition are stored row wise.
- \( dtype \) a string that contains the type of the distribution: "cont" or "discr".
Note

It is important that all given functions – rdist, qdist, and pdist – accept the same arguments passed to rvgt.ftable via ....

The random variate generator rdist can alternatively be a generator object form the Runuran package.

Author(s)
Sougata Chaudhuri <sgtchaudhuri@gmail.com>, Josef Leydold <josef.leydold@wu.ac.at>

References

See Also
See plot.rvgt.ftable for the syntax of the plotting method.

Examples

```
## Create a frequency table for normal distribution with mean 1 and
## standard deviation 2. Number of bins should be 50.
## Use a sample of size of 5 times 10^5 random variates.
ft <- rvgt.ftable(n=1e5,rep=5, rdist=rnorm,qdist=qnorm, breaks=51, mean=1, sd=2)

## Show histogram
plot(ft)

## Run a chi-square test
rvgt.chisq(ft)

## The following allows to plot a histogram in a single call.
rvgt.ftable(n=1e5,rep=5, rdist=rnorm,qdist=qnorm, plot=TRUE)

## Use the cumulative distribution function when the quantile function
## is not available or if its round-off errors have serious impact.
ft <- rvgt.ftable(n=1e5,rep=5, rdist=rnorm,pdist=pnorm )
plot(ft)

## Create a frequency table for the normal distribution with
## non-equidistributed break points
ft <- rvgt.ftable(n=1e5,rep=5, rdist=rnorm,qdist=qnorm, breaks=1/(1:100))
plot(ft)

## A (naive) generator for a truncated normal distribution
rdist <- function(n) {
x <- numeric(n)
for (i in 1:n){ while(TRUE){ x[i] <- rnorm(1); if (x[i]<1) break } }
return(x)
}
```
rvgt.Mtest

Perform M-Test on RVG Frequency Table

Description

Perform an adjusted residual test ("M-Test") on the RVG frequency table.

Usage

rvgt.Mtest(ftable)

Arguments

ftable object of class "rvgt.ftable".

Details

rvgt.Mtest performs an adjusted residual test ("M-Test") on the bin counts of the RVG frequency tables created by means of rvgt.ftable. The null hypothesis is that the bin counts are distributed according to the expected probabilities (i.e., proportional to the lengths of the histogram cells).

If the ftable contains the bin counts for more than one repetitions then the test is repeated with column sums of increasing number of rows which corresponds to increasing sampling sizes. This allows for getting an idea of the power of the test.

Value

An object of class "rvgt.htest" which is a list with components:

- type character string containing type of test; equals "M-test".
- n sample size for every row in ftable.
- rep number of rows in ftable.
- breaks number of break points in ftable (i.e., number of bins + 1).
- pval vector of p-values.

Author(s)

Sougata Chaudhuri <sgtchaudhuri@gmail.com>, Josef Leydold <josef.leydold@wu.ac.at>
References


See Also

See `plot.rvgt.htest` for the syntax of the plotting method.

Examples

```r
## Create a frequency table for normal distribution and
## conduct an M-test on this data.
## Use a sample of size of 5 times 10^5 random variates.
table <- rvgt.ftable(n=1e5, rep=5, rdist=rnorm, qdist=qnorm, mean=1, sd=2)

## Perform test
result <- rvgt.Mtest(table)

## Plot result
plot(result)
```

---

**uerror** *Create Table of U-Errors for Numerical Inversion Method*

**Description**

Function for creating a table of u-errors of a numerical inversion method (i.e., it uses an approximate quantile function of the target distribution). Thus the domain of the inverse distribution function is partitioned into intervals for which maxima, minima and some other quantiles of the u-errors are computed.

Currently the function only works for generators for continuous univariate distribution.

**Usage**

```r
uerror(n, aqdist, pdist, ..., trunc=NULL, udomain=c(0,1),
       res=1000, tails=FALSE, plot=FALSE)
```

**Arguments**

- `n` sample size for one repetition.
- `aqdist` approximate inverse distribution function (quantile function) for a continuous univariate distribution.
- `pdist` cumulative distribution function for distribution.
- `...` parameters to be passed to `pdist`.
- `trunc` boundaries of truncated domain. (optional)
The u-error of an approximate inverse distribution function (quantile function) $G^{-1}$ for some $u \in (0, 1)$ is given by

$$\epsilon_u(u) = |u - F(G^{-1}(u))|$$

where $F$ denotes the (exact) cumulative distribution. It is a convenient measure for approximation errors in non-uniform random variate generators based on numerical inversion, see the reference below for our arguments.

Computing, plotting and analyzing of such u-errors can be quite time consuming.

$$\epsilon_u(u)$$

is a very volatile function and requires the computation at a lot of points. For plotting we can condense the information by partitioning $(0, 1)$ into intervals of equal length. In each of these the u-error is computed at equidistributed points and some quantiles (see below) are estimated and stored. Thus we save memory and it is much faster to plot and compare u-errors for different methods or distributions.

If `trunc` is given, then function `pdist` is rescaled to this given domain. Notice, however, that this has some influence on the accuracy of the results of the distribution function `pdist`.

Using argument `udomain` it is possible to restrict the domain of the given (approximate) quantile function `aqdist`, i.e., of its argument $u$.

When `tails=TRUE` we use additional $n$ points for the first and last interval (which correspond to the tail regions of the distribution).

**Value**

An object of class "rvgt.ierror" which is a list with components:

- `n` sample size.
- `res` resolution (number of intervals).
- `kind` kind of error (string).
- `udomain` domain for $u$.
- `min` an array of minimum u-errors (of length `res`).
- `lqr` an array of lower quartile of u-errors (of length `res`).
- `med` an array of median u-errors (of length `res`).
- `uqr` an array of upper quartile of u-errors (of length `res`).
- `max` an array of maximum u-errors (of length `res`).
- `mad` an array of mean absolute deviations (of length `res`).
- `mse` an array of mean squared errors (of length `res`).
**Note**

It should be noted that `uerror` computes the numerical error of the composed function `pdist(aqdist(u))`. Thus one needs a distribution function `pdist` that is numerically (much) more accurate than `aqdist`.

The random variate generator `rdist` can alternatively be a generator object from the `Runuran` package.

**Author(s)**

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

**References**


**See Also**

See `plot.rvgt.ierror` for the syntax of the plotting method. See `xerror` for computing x-errors.

**Examples**

```r
## Create a table of u-errors for spline interpolation of
## the inverse CDF of the standard normal distribution
aqn <- splinefun(x=pnorm((-100:100)*0.05), y=(-100:100)*0.05,
                method="mono.H.FC")
## Use a sample of size of 1e5 random variates.
uerrn <- uerror(n=1e5, aqdist=aqn, pdist=pnorm)

## Plot u-errors
plot(uerrn)

## Investigate tails more accurately, and use
## a resolution of 1000 intervals.
uerrn <- uerror(n=1e5, aqdist=aqn, pdist=pnorm, res=1000, tails=TRUE)

## Same for a gamma distribution.
## But this time we immediately plot the error.
aaq <- splinefun(x=pgamma((0:500)*0.1, shape=5),
                y=(0:500)*0.1, method="mono.H.FC")
uerrg <- uerror(n=1e5, aqdist=aqg, pdist=pgamma, shape=5,
                plot=TRUE)

## Compute u-error for a subdomain of a beta distribution
aqb <- splinefun(x=pbeta((0:100)*0.01, shape1=2, shape2=5),
                y=(0:100)*0.01, method="mono.H.FC")
uerrb <- uerror(n=1e5, aqdist=aqb, pdist=pbeta, shape1=2, shape2=5,
                udomain=c(0.6,0.65), plot=TRUE)

## Show all u-errors in one plot
```
Create Table of X-Errors for Numerical Inversion Method

**Description**

Function for creating a table of x-errors of a numerical inversion method (i.e., it uses an approximate quantile function of the target distribution). Thus the domain of the inverse distribution function is partitioned into intervals for which maxima, minima and some other quantiles of the x-errors are computed.

Currently the function only works for generators for continuous univariate distribution.

**Usage**

```r
xerror(n, aqdist, qdist, ..., trunc=NULL, udomain=c(0,1),
       res=1000, kind=c("abs","rel"), tails=FALSE, plot=FALSE)
```

**Arguments**

- `n`: sample size for one repetition.
- `aqdist`: approximate inverse distribution function (quantile function) for a continuous univariate distribution.
- `qdist`: (Exact) quatile function of distribution.
- `...`: parameters to be passed to `qdist`.
- `trunc`: boundaries of truncated domain. (optional)
- `udomain`: domain of investigation for (approximate) quantile function `aqdist`.
- `res`: resolution (number of intervals).
- `kind`: kind of x-error.
- `tails`: logical. If TRUE, then the tail regions are treated more accurately. However, this doubles the given sample size.
- `plot`: logical. If TRUE, the (range of the) x-errors is plotted.
Details

The absolute x-error of an approximate inverse distribution function (quantile function) $G^{-1}$ for some $u \in (0, 1)$ is given by

$$\epsilon_x(u) = |F^{-1}(u) - G^{-1}(u)|$$

where $F^{-1}$ denotes the (exact) quantile function of the distribution. The relative x-error is then defined as

$$\epsilon_x(u) / |F^{-1}(u)|$$

Computing, plotting and analyzing of such x-errors can be quite time consuming. $\epsilon_x(u)$ is a very volatile function and requires the computation at a lot of points. For plotting we can condense the information by partitioning $(0,1)$ into intervals of equal length. In each of these the x-error is computed at equidistributed points and some quantiles (see below) are estimated and stored. Thus we save memory and it is much faster to plot and compare x-errors for different methods or distributions.

If `trunc` is given, then function `qdist` is rescaled to this given domain. Notice, however, that this has some influence on the accuracy of the results of the “exact” quantile function `qdist`.

Using argument `udomain` it is possible to restrict the domain of the given (approximate) quantile function `aqdist`, i.e., of its argument $u$.

When `tails=TRUE` we use additional $n$ points for the first and last interval (which correspond to the tail regions of the distribution).

Value

An object of class "rvgt.ierror", see `uerror` for details.

Note

It should be noted that `xerror` computes the difference between the approximate inversion function `aqdist(u)` and the given ‘exact’ quantile function `qdist`. Thus one needs a quantile function `qdist` that is numerically (much) more accurate than `aqdist`.

The random variate generator `rdist` can alternatively be a generator object form the `Runuran` package.

Author(s)

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

See Also

See `plot.rvgt.ierror` for the syntax of the plotting method. See `uerror` for computing u-errors.
## Examples

```r
## Create a table of absolute x-errors for spline interpolation of
## the inverse CDF of the standard normal distribution.
aq <- splinefun(x=pnorm((-100:100)/0.05), y=(-100:100)/0.05,
               method="monoh.FC")
## Use a sample of size of 10*5 random variates.
xerr <- xerror(n=1e5, aqdist=aq, qdist=qnorm, kind="abs")

## Plot x-errors
plot(xerr)

## Same for the relative error.
## But this time we use a resolution of 500, and
## we immediately plot the error.
xerr <- xerror(n=1e5, aqdist=aq, qdist=qnorm,
               res=500, kind="rel", plot=TRUE)

## An inverse CDF for a truncated normal distribution
aqtn <- splinefun(x=(pnorm((0:100)/0.015) - pnorm(0))/(pnorm(1.5)-pnorm(0)),
             y=(0:100)/0.015, method="monoh.FC")
xerrtn <- xerror(n=1e5, aqdist=aqtn, qdist=qnorm, trunc=c(0,1.5),
                 plot=TRUE)
```
Index

*Topic **datagen**
  Perturbation, 5
  rvgt.ftable, 11
  rvgtest-package, 2
  uerror, 15
  xerror, 18

*Topic **distribution**
  Perturbation, 5
  plot.rvgt.ftable, 6
  plot.rvgt.htest, 7
  plot.rvgt.ierror, 8
  rvgt.chisq, 10
  rvgt.ftable, 11
  rvgt.Mtest, 14
  rvgtest-package, 2
  uerror, 15
  xerror, 18

*Topic **hplot**
  plot.rvgt.ftable, 6
  plot.rvgt.htest, 7
  plot.rvgt.ierror, 8
  rvgtest-package, 2

*Topic **htest**
  plot.rvgt.htest, 7
  plot.rvgt.ierror, 8
  rvgt.chisq, 10
  rvgt.Mtest, 14
  rvgtest-package, 2

*Topic **package**
  rvgtest-package, 2

hist, 7, 12

pertadd, 3
pertadd (Perturbation), 5
pertsb, 3
pertsb (Perturbation), 5
Perturbation, 5
plot.rvgt.ftable, 3, 6, 13
plot.rvgt.htest, 3, 7, 10, 15

plot.rvgt.ierror, 4, 8, 17, 19
quantile, 12

Runuran, 13, 17, 19
rvgt.chisq, 3, 8, 10
rvgt.ftable, 3, 7, 10, 11, 14
rvgt.Mtest, 3, 8, 14
rvgtest (rvgtest-package), 2
rvgtest-package, 2
uerror, 4, 9, 15, 19
xerror, 4, 9, 17, 18

hist, 7, 12

pertadd, 3
pertadd (Perturbation), 5
pertsb, 3
pertsb (Perturbation), 5
Perturbation, 5
plot.rvgt.ftable, 3, 6, 13
plot.rvgt.htest, 3, 7, 10, 15

plot.rvgt.ierror, 4, 8, 17, 19
quantile, 12

Runuran, 13, 17, 19
rvgt.chisq, 3, 8, 10
rvgt.ftable, 3, 7, 10, 11, 14
rvgt.Mtest, 3, 8, 14
rvgtest (rvgtest-package), 2
rvgtest-package, 2
uerror, 4, 9, 15, 19
xerror, 4, 9, 17, 18