

Package ‘Tinflex’

June 8, 2017

Type Package

Title A Universal Non-Uniform Random Number Generator

Version 1.2

Date 2017-06-08

Author Josef Leydold, Carsten Botts and Wolfgang H{"ormann

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

Imports graphics, stats

Suggests Runuran, rvgtest

Description A universal non-uniform random number generator
for quite arbitrary distributions with piecewise twice
differentiable densities.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-06-08 15:21:10 UTC

R topics documented:

Tinflex-package	2
plot.Tinflex	4
print.Tinflex	5
Tinflex.sample	6
Tinflex.setup	6

Index	11
--------------	-----------

Tinflex-package

Tinflex – Universal non-uniform random number generator

Description

Tinflex is a universal non-uniform random number generator based on the acceptance-rejection method for all distributions that have a piecewise twice differentiable density function. Required input includes the log-density function of the target distribution and its first and second derivatives.

Details

Package: Tinflex
Type: Package
Version: 1.2
Date: 2017-06-08
License: GPL 2 or later

Package **Tinflex** serves two purposes:

1. The installed package provides a fast routine for sampling from any distribution that has a piecewise twice differentiable density function.
2. The R source (including comments) presents all details of the general sampling method which are not entirely worked out in our paper cited in the see references below.

Algorithm **Tinflex** is a universal random variate generator based on transformed density rejection which is a variant of the acceptance-rejection method. The generator first computes and stores hat and squeeze functions and then uses these functions to generate variates from the distribution of interest. Since the setup procedure is separated from the generation procedure, many samples can be drawn from the same distribution without rerunning the (expensive) setup.

The algorithm requires the following data about the distribution (for further details see [Tinflex.setup](#)):

- the log-density of the target distribution;
- its first and second derivative;
- a starting partition of its domain such that each subinterval contains at most one inflection point of the transformed density;
- a transformation for the density (default is the logarithm transformation).

The following routines are provided.

[Tinflex.setup](#) computes hat and squeeze. The table is then stored in a generator object of class "Tinflex".

[Tinflex.sample](#) draws a random sample from a particular generator object.

[print.Tinflex](#) prints the properties a generator object of class "Tinflex".

[plot.Tinflex](#) plots density, hat and squeeze functions for a given generator object of class "Tinflex".

For further details see [Tinflex.setup](#).

Warning

It is very important to note that the user is responsible for the correctness of the supplied arguments. Since the algorithm works (in theory) for all distributions with piecewise twice differentiable density functions, it is not possible to detect improper arguments. It is thus recommended that the user inspect the generator object visually by means of the `plot` method (see `plot.Tinflex` for details).

Package `rvgtest` provides a test suite for non-uniform random number generators. (Approximate distribution functions are available through method `pinv.new` in package `Runuran`.)

Note

Routine `Tinflex.sample` is implemented both as pure R code (routine `Tinflex.sample.R`) for documenting the algorithm as well as C code for fast performance.

Author(s)

Josef Leydold <josef.leydold@wu.ac.at>, Carsten Botts and Wolfgang Hörmann.

References

C. Botts, W. Hörmann, and J. Leydold (2013), Transformed Density Rejection with Inflection Points, *Statistics and Computing* 23(2), 251–260, DOI: 10.1007/s11222-011-9306-4 (see also Research Report Series / Department of Statistics and Mathematics Nr. 110, Department of Statistics and Mathematics, WU Wien, <http://epub.wu.ac.at/>).

See Also

See `Tinflex.setup` for further details.

Package `Runuran` provides a set of many other automatic non-uniform sampling algorithms.

Examples

```
## Bimodal density
## f(x) = exp( -|x|^alpha + s*|x|^beta + eps*|x|^2 )
## with alpha > beta >= 2 and s, eps > 0

alpha <- 4.2
beta <- 2.1
s <- 1
eps <- 0.1

## Log-density and its derivatives.
lpdf <- function(x) { -abs(x)^alpha + s*abs(x)^beta + eps*abs(x)^2 }
dlpdf <- function(x) { (sign(x) * (-alpha*abs(x)^(alpha-1)
+ s*beta*abs(x)^(beta-1) + 2*eps*abs(x))) }
d2lpdf <- function(x) { (-alpha*(alpha-1)*abs(x)^(alpha-2)
+ s*beta*(beta-1)*abs(x)^(beta-2) + 2*eps) }

## Parameter cT=0 (default):
## There are two inflection points on either side of 0.
ib <- c(-Inf, 0, Inf)
```

```
## Create generator object.
gen <- Tinflex.setup(lpdf, dlpdf, d2lpdf, ib=c(-Inf,0,Inf), rho=1.1)

## Print data about generator object.
print(gen)

## Draw a random sample
Tinflex.sample(gen, n=10)

## Inspect hat and squeeze visually in original scale
plot(gen, from=-2.5, to=2.5)
## ... and in transformed (log) scale.
plot(gen, from=-2.5, to=2.5, is.trans=TRUE)
```

plot.Tinflex

Plot Tinflex Generator Object

Description

Plotting method for generator objects of class "Tinflex". The plot shows the (transformed) density, hat and squeeze.

Usage

```
## S3 method for class 'Tinflex'
plot(x, from, to, is.trans=FALSE, ...)
```

Arguments

x	an object of class "Tinflex".
from,to	the range over which the function will be plotted. (numeric)
is.trans	if TRUE then the transformed density and its hat and squeezes are plotted. (logical)
...	arguments to be passed to methods, such as graphical parameters (see par). In particular the following argument may be useful:
	ylim limit for the plot range: see plot.window . It has sensible defaults if omitted.

Details

This is the [print](#) method for objects of class "Tinflex". It plots the given density function (blue) in the domain (from,to) as well as hat function (red) and squeeze (green) of the acceptance-rejection algorithm. If `is.trans` is set to TRUE, then density function, hat and squeeze are plotted on the transformed scale. Notice that the latter only gives a sensible picture if parameter `cT` is the same for all intervals.

Author(s)

Josef Leydold <josef.leydold@wu.ac.at>, Carsten Botts and Wolfgang Hörmann.

See Also

[plot](#), [plot.function](#). See [Tinflex.setup](#) for examples.

<code>print.Tinflex</code>	<i>Print Tinflex Generator Object</i>
----------------------------	---------------------------------------

Description

Print method for generator objects of class "Tinflex".

Usage

```
## S3 method for class 'Tinflex'  
print(x, debug=FALSE, ...)
```

Arguments

<code>x</code>	an object of class "Tinflex".
<code>debug</code>	enable/disable the display of detailed information about the object. (logical)
<code>...</code>	additional arguments to print .

Details

This is the [print](#) method for objects of class "Tinflex".

Author(s)

Josef Leydold <josef.leydold@wu.ac.at>, Carsten Botts and Wolfgang Hörmann.

See Also

[print.Tinflex.setup](#). See [Tinflex.setup](#) for examples.

Tinflex.sample *Draw Random Sample from Tinflex Generator Object*

Description

Draw a random sample from a generator object of class "Tinflex".

Usage

```
Tinflex.sample(gen, n=1)
```

Arguments

gen	an object of class "Tinflex".
n	sample size. (integer)

Author(s)

Josef Leydold <josef.leydold@wu.ac.at>, Carsten Botts and Wolfgang Hörmann.

See Also

See [Tinflex.setup](#) for examples.

Tinflex.setup *Create Tinflex Generator Object*

Description

Create a generator object of class "Tinflex".

Usage

```
Tinflex.setup(lpdf, dlpdf, d2lpdf, ib, cT=0, rho=1.1, max.intervals=1001)
```

Arguments

lpdf	log-density of target distribution. (function)
dlpdf	first derivative of log-density. (function)
d2lpdf	second derivative of log-density. (function)
ib	break points for partition of domain of log-density. (numeric vector of length greater than 1)
cT	parameter for transformation T_c . (numeric vector of length 1 or of length $\text{length}(ib)-1$)
rho	performance parameter: requested upper bound for ratio of area below hat to area below squeeze. (numeric)
max.intervals	maximal numbers of intervals. (numeric)

Details

Algorithm `Tinflex` is a flexible algorithm that works (in theory) for all distributions that have a piecewise twice differentiable density function. The algorithm is based on the transformed density rejection algorithm which is a variant of the acceptance-rejection algorithm where the density of the target distribution is transformed by means of some transformation T_c . Hat and squeeze functions of the density are then constructed by means of tangents and secants.

The algorithm uses family T_c of transformations, where

$$T_c(x) = \begin{cases} \log(x) & \text{for } c = 0, \\ \text{sign}(c) x^c & \text{for } c \neq 0. \end{cases}$$

Parameter c is given by argument `cT`.

The algorithm requires the following input from the user:

- the log-density of the target distribution, `lpdf`;
- its first derivative `d1pdf`;
- its second derivative `d2lpdf`;
- a starting partition `ib` of the domain of the target distribution such that each subinterval contains at most one inflection point of the transformed density;
- the parameter(s) `cT` of the transformation either for the entire domain or alternatively for each of the subintervals of the partition.

The starting partition of the domain of the target distribution into non-overlapping intervals has to satisfy the following conditions:

- The partition points must be given in ascending order (otherwise they are sorted anyway).
- The first and last entry of this vector are the boundary points of the domain of the distribution. In the case when the domain of the distribution is unbounded, the respective points are `-Inf` and `Inf`.
- Within each interval of the partition, the transformed density possesses at most one inflection point (including all finite boundary points).
- If a boundary point is infinite, or the density vanishes at the boundary point, then the transformed density must be concave near the corresponding boundary point and in the corresponding tail, respectively.
- If the log-density `lpdf` has a pole or cusp at some point x , then this must be added to the starting partition point. Moreover, it has to be counted as inflection point.

Parameter `cT` is either a single numeric, that is, the same transformation T_c is used for all subintervals of the domain, or it can be set independently for each of these intervals. In the latter case `length(cT)` must be equal to the number of intervals, that is, equal to `length(ib)-1`. For the choice of `cT` the following should be taken into consideration:

- `cT=0` (the default) is most robust against numeric underflow or overflow.
- `cT=-0.5` has the fastest marginal generation time.
- One should always use `cT=0` or `cT=-0.5` for intervals that contain a point where the derivative of the (log-) density vanishes (e.g., an extremum). For other values of `cT`, the algorithm is less accurate.

- For unbounded intervals $(-\infty, a]$ or $[a, \infty)$, one has to select c_T such that $0 \geq c_T > -1$.
- For an interval that contains a pole at one of its boundary points (i.e., there the density is unbounded), one has to select c_T such that $c_T < -1$ and the transformed density is convex.
- If the transformed density is concave in some interval for a particular value of c_T , then it is concave for all smaller values of c_T .

Parameter `rho` is a performance parameter. It defines an upper bound for ratio of the area below the hat function to the area below the squeeze function. This parameter is an upper bound of the rejection constant. More importantly, it provides an approximation to the number of (time consuming) evaluations of the log-density function `lpdf`. For `rho=1.01`, the log-density function is evaluated once for a sample of 300 random points. However, values of `rho` close to 1 also increase the table size and thus make the setup more expensive.

Parameter `max.intervals` defines the maximal number of subintervals and thus the maximal table size. Putting an upper bound on the table size prevents the algorithm from accidentally exhausting all of the computer memory due to invalid input. It is very unlikely that one has to increase the default value.

Value

Object of class "Tinflex" that stores the random variate generator (density, hat and squeeze functions, cumulated areas below hat). For details see sources of the algorithm or execute `print(gen, debug=TRUE)` with an object `gen` of class "Tinflex".

Warning

It is very important to note that the user is responsible for the correctness of the supplied arguments. Since the algorithm works (in theory) for all distributions with piecewise twice differentiable density functions, it is not possible to detect improper arguments. It is thus recommended that the user inspect the generator object visually by means of the `plot` method (see `plot.Tinflex` for details).

Package `rvgtest` provides a test suite for non-uniform random number generators. (Approximate distribution functions are available through method `pinv.new` in package `Runuran`.)

Author(s)

Josef Leydold <josef.leydold@wu.ac.at>, Carsten Botts and Wolfgang Hörmann.

References

C. Botts, W. Hörmann, and J. Leydold (2013), Transformed Density Rejection with Inflection Points, *Statistics and Computing* 23(2), 251–260, DOI: 10.1007/s11222-011-9306-4 (see also Research Report Series / Department of Statistics and Mathematics Nr. 110, Department of Statistics and Mathematics, WU Wien, <http://pub.wu.ac.at/>).

See Also

See `Tinflex.sample` for drawing random samples, `plot.Tinflex` and `print.Tinflex` for printing and plotting objects of class "Tinflex".

Examples

```

## Example 1: Bimodal density
## Density  $f(x) = \exp(-|x|^\alpha + s|x|^\beta + \text{eps}|x|^2)$ 
## with  $\alpha > \beta \geq 2$  and  $s, \text{eps} > 0$ 

alpha <- 4.2
beta <- 2.1
s <- 1
eps <- 0.1

## Log-density and its derivatives.
lpdf <- function(x) { -abs(x)^alpha + s*abs(x)^beta + eps*abs(x)^2 }
dlpdf <- function(x) { (sign(x) * (-alpha*abs(x)^(alpha-1)
+ s*beta*abs(x)^(beta-1) + 2*eps*abs(x))) }
d2lpdf <- function(x) { (-alpha*(alpha-1)*abs(x)^(alpha-2)
+ s*beta*(beta-1)*abs(x)^(beta-2) + 2*eps) }

## Parameter cT=0 (default):
## There are two inflection points on either side of 0.
ib <- c(-Inf, 0, Inf)

## Create generator object.
gen <- Tinflex.setup(lpdf, dlpdf, d2lpdf, ib=c(-Inf,0,Inf), rho=1.1)

## Print data about generator object.
print(gen)

## Draw a random sample
Tinflex.sample(gen, n=10)

## Inspect hat and squeeze visually in original scale
plot(gen, from=-2.5, to=2.5)
## ... and in transformed (log) scale.
plot(gen, from=-2.5, to=2.5, is.trans=TRUE)

## -----
## Example 2: Exponential Power Distribution
## Density  $f(x) = \exp(-|x|^\alpha)$ 
## with  $\alpha > 0$  [  $\geq 0.015$  due to limitations of FPA ]

alpha <- 0.5

## Log-density and its derivatives.
lpdf <- function(x) { -abs(x)^alpha }
dlpdf <- function(x) { if (x==0) {0} else {-sign(x) * alpha*abs(x)^(alpha-1)}}
d2lpdf <- function(x) { -alpha*(alpha-1)*abs(x)^(alpha-2) }

## Parameter cT=-0.5:
## There are two inflection points on either side of 0 and
## a cusp at 0. Thus we need a partition point that separates
## the inflection points from the cusp.
ib <- c(-Inf, -(1-alpha)/2, 0, (1-alpha)/2, Inf)

```

```

## Create generator object with c = -0.5.
gen <- Tinflex.setup(lpdf, dlpdf, d2lpdf, ib=ib, cT=-0.5, rho=1.1)

## Print data about generator object.
print(gen)

## Draw a random sample.
Tinflex.sample(gen, n=10)

## Inspect hat and squeeze visually in original scale
plot(gen, from=-4, to=4)
## ... and in transformed (log) scale.
plot(gen, from=-4, to=4, is.trans=TRUE)

## -----
## Example 3: Generalized Inverse Gaussian Distribution
## Density  $f(x) = x^{(\lambda-1)} * \exp(-\omega/2 * (x+1/x))$   $x \geq 0$ 
## with  $0 < \lambda < 1$  and  $0 < \omega \leq 0.5$ 

la <- 0.4 ## lambda
om <- 1.e-7 ## omega

## Log-density and its derivatives.
lpdf <- function(x) { ifelse (x==0., -Inf, ((la - 1) * log(x) - om/2*(x+1/x))) }
dlpdf <- function(x) { if (x==0) { Inf} else {(om + 2*(la-1)*x-om*x^2)/(2*x^2)} }
d2lpdf <- function(x) { if (x==0) {-Inf} else {-(om - x + la*x)/x^3} }

## Parameter cT=0 near 0 and cT=-0.5 at tail:
ib <- c(0, (3/2*om/(1-la) + 2/9*(1-la)/om), Inf)
cT <- c(0,-0.5)

## Create generator object.
gen <- Tinflex.setup(lpdf, dlpdf, d2lpdf, ib=ib, cT=cT, rho=1.1)

## Print data about generator object.
print(gen)

## Draw a random sample.
Tinflex.sample(gen, n=10)

## Inspect hat and squeeze visually in original scale
plot(gen, from=0, to=5)

```

Index

*Topic **datagen**

- plot.Tinflex, 4
- print.Tinflex, 5
- Tinflex-package, 2
- Tinflex.sample, 6
- Tinflex.setup, 6

*Topic **distribution**

- plot.Tinflex, 4
- print.Tinflex, 5
- Tinflex-package, 2
- Tinflex.sample, 6
- Tinflex.setup, 6

*Topic **package**

- Tinflex-package, 2

par, 4

pinv.new, 3, 8

plot, 5

plot.function, 5

plot.Tinflex, 2, 3, 4, 8

plot.window, 4

print, 4, 5

print.Tinflex, 2, 5, 8

Runuran, 3, 8

rvgtest, 3, 8

Tinflex (Tinflex-package), 2

Tinflex-package, 2

Tinflex.sample, 2, 3, 6, 8

Tinflex.setup, 2, 3, 5, 6, 6