Package: Tinflex (via r-universe)

September 8, 2024

Type Package

Title A Universal Non-Uniform Random Number Generator

Version 2.4

Date 2023-03-21

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

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

Imports graphics, methods, stats

Suggests Runuran

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

License GPL (>= 2)

NeedsCompilation yes

Date/Publication 2023-03-21 20:50:05 UTC

Repository https://jleydold.r-universe.dev

RemoteUrl https://github.com/cran/Tinflex

RemoteRef HEAD

RemoteSha e02d070edc59890d8f75701359bc2f5ad69470fb

Contents

`Tinflex-package` *Tinflex – Universal non-uniform random number generator*

Description

`Tinflex` is a universal non-uniform random number generator based on the acceptence-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:	2.4
Date:	2023-03-21
License:	GPL 2 or later

Package **Tinflex** serves three purposes:

1. The installed package provides a fast routine for sampling from any distribution that has a piecewise twice differentiable density function.
2. It provides C routines functions that could be used in other packages (see the installed C header files).
3. 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 targent distribution;
- its first derivative;
- its second derivative (optionally);
- 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`.

There are variants of the method. The first one uses the second derivative to determine regions whre the transformed density is convex, concave, or has a single inflection points. The second variant estimates the signs on the second derivative by means of the first derivative. Thus it is easier to use at the expense of a more complex algorithm.

There are two different implementation: Routine `Tinflex.setup` is implemented mainly in R and serves (together with `Tinflex:::Tinflex.sample.R`) as a reference implementation of the published algorithm. Nevertheless, the sampling routine `Tinflex.sample` runs quite fast.

Routine `Tinflex.setup.C` on the other hand is implemented entirely in C. So it also allows to link to the underlying C code from other packages.

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).

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/s1122201193064. See also Research Report Series / Department of Statistics and Mathematics Nr. 110, Department of Statistics and Mathematics, WU Vienna University of Economics and Business, https://epub.wu.ac.at/id/eprint/3158.

W. Hörmann, and J. Leydold (2022), A Generalized Transformed Density Rejection Algorithm, in: Advances in Modeling and Simulation, Ch. 14, doi:10.1007/9783031101939_14, accepted for publication.. See also Research Report Series / Department of Statistics and Mathematics Nr. 135, Department of Statistics and Mathematics, WU Vienna University of Economics and Business, https://research.wu.ac.at/de/publications/a-generalized-transformed-density-rejection-algorithm.

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.C(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)

## With Version 2.0 the setup also works without providing the
## second derivative of the log-density
gen <- Tinflex.setup.C(lpdf, dlpdf, d2lpdf=NULL, ib=c(-Inf,0,Inf), rho=1.1)
Tinflex.sample(gen, n=10)
```

`plot.Tinflex` *Plot Tinflex Generator Objects*

Description

Plotting methods for generator objects of classes `"Tinflex"` and `"TinflexC"`. The plot shows the (transformed) density, hat and squeeze.

Usage

```
## S3 method for class 'Tinflex'
plot(x, from, to, is.trans=FALSE, n=501, ...)
## S3 method for class 'TinflexC'
plot(x, from, to, is.trans=FALSE, n=501, ...)
```

Arguments

x	an object of class "Tinflex" or "TinflexC".
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)
n	the number of x values at which (transformed) PDF to evaluate. (integer)
...	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" or "TinflexC". 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.

---

print.Tinflex *Print Tinflex Generator Objects*

Description

Print methods for generator objects of class "Tinflex" or "TinflexC".

Usage

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

Arguments

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

Details

These are the print methods for objects of classes "Tinflex" and "TinflexC".

Author(s)

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

See Also

print. Tinflex.setup. Tinflex.setup.C. 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" or "TinflexC".

Usage

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

Arguments

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

Details

Routine Tinflex.sample.C allows objects of class "TinflexC" only and thus is a bit faster than the same call with routine Tinflex.sample.

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 Objects*

Description

Create a generator object of class `"Tinflex"` or `"TinflexC"`.

Usage

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

Arguments

`lpdf`	log-density of targent distribution. (function)
`dlpdf`	first derivative of log-density. (function)
`d2lpdf`	second derivative of log-density. (function, optional)
`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 `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 targent 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 targent distribution, `lpdf`;
- its first derivative `dlpdf`;
- its second derivative `d2lpdf` (optionally);
- 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. Moreover, in the corresponding intervals the transformed density must be convex.

Argument `d2lpdf` is optional. If `d2lpdf=NULL`, then a variant of the method is used, that determines intervals where the transformed density is concave or convex without means of the second derivative of the log-density.

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 $(-\inf, a]$ or $[a, \inf)$, one has to select `cT` 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 `cT` 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 `cT`, then it is concave for all smaller values of `cT`.

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) evalutions 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

Routine `Tinflex.setup` returns an 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"`.

Routine `Tinflex.setup.C` is equivalent to `Tinflex.setup` but does all computations entirely in C. It returns an object of class `"TinflexC"` which is equivalent to class `"Tinflex"` but stores all data in an C structure instead of an R list.

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).

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/s1122201193064. See also Research Report Series / Department of Statistics and Mathematics Nr. 110, Department of Statistics and Mathematics, WU Vienna University of Economics and Business, https://epub.wu.ac.at/id/eprint/3158.

W. Hörmann, and J. Leydold (2022), A Generalized Transformed Density Rejection Algorithm, in: Advances in Modeling and Simulation, Ch. 14, doi:10.1007/9783031101939_14, accepted for publication.. See also Research Report Series / Department of Statistics and Mathematics Nr. 135, Department of Statistics and Mathematics, WU Vienna University of Economics and Business, https://research.wu.ac.at/de/publications/a-generalized-transformed-density-rejection-algorithm.

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 + 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.C(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)

## With Version 2.0 the setup also works without providing the
## second derivative of the log-density
gen <- Tinflex.setup.C(lpdf, dlpdf, d2lpdf=NULL, ib=c(-Inf,0,Inf), rho=1.1)
Tinflex.sample(gen, n=10)

## ----------------------------------------------------------------------
## Example 2: Exponential Power Distribution
##   Density  f(x) = exp( -|x|^alpha )
##   with alpha > 0  [ >= 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.C(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)

## With Version 2.0 the setup also works without providing the
## second derivative of the log-density
gen <- Tinflex.setup.C(lpdf, dlpdf, d2lpdf=NULL, ib=ib, cT=-0.5, rho=1.1)
Tinflex.sample(gen, n=10)

## ----------------------------------------------------------------------
## Example 3: Generalized Inverse Gaussian Distribution
##   Density  f(x) = x^(lambda-1) * exp(-omega/2 * (x+1/x))   x>= 0
##   with 0 < lambda < 1 and 0 < omega <= 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.C(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)

## With Version 2.0 the setup also works without providing the
## second derivative of the log-density
gen <- Tinflex.setup.C(lpdf, dlpdf, d2lpdf=NULL, ib=ib, cT=cT, rho=1.1)
Tinflex.sample(gen, n=10)
```

Index