Package 'AR'

January 20, 2025

Type Package

Title Another Look at the Acceptance-Rejection Method

Version 1.1

Date 2018-05-02

Author Abbas Parchami (Department of Statistics, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran)

Maintainer Abbas Parchami <parchami@uk.ac.ir>

Description In mathematics, 'rejection sampling' is a basic technique used to generate observations from a distribution. It is also commonly called 'the Acceptance-Rejection method' or 'Accept-Reject algorithm' and is a type of Monte Carlo method. 'Acceptance-Rejection method' is based on the observation that to sample a random variable one can perform a uniformly random sampling of the 2D cartesian graph, and keep the samples in the region under the graph of its density function. Package 'AR' is able to generate/simulate random data from a probability density function by Acceptance-Rejection method. Moreover, this package is a useful teaching resource for graphical presentation of Acceptance-Rejection method. From the practical point of view, the user needs to calculate a constant in Acceptance-Rejection method, which package 'AR' is able to compute this constant by optimization tools. Several numerical examples are provided to illustrate the graphical presentation for the Acceptance-Rejection Method.

License LGPL (>= 3)

Imports DISTRIB

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-02 03:54:28 UTC

Contents

AR-package *Another Look at the Acceptance-Rejection Method*

Description

There are many distributions for which the inverse transform method and even general transformations will fail to be able to generate the required random variables. For these cases, we must turn to indirect methods; that is, methods in which we generate a candidate random variable and only accept it subject to passing a test. This class of methods is extremely powerful and will allow us to simulate from virtually any distribution; see (Robert and Casella, 2010) for more detailes. These so-called `Accept-Reject methods` only require us to know the functional form of the density $f_X(.)$ of interest (called the target density) up to a multiplicative constant. We use a simpler (to simulate) density f_Y, called the instrumental or candidate density, to generate the random variable for which the simulation is actually done. The constraints we impose on this candidate density f_Y are that:

(i) Y be simulate-able (the data simulation from Y be actually possible).

(ii) There is a constant c with $\frac{f_X(x)}{f_Y(x)} \leq c$ for all $x \in S_X = \{x : f_X(x) > 0\}$.

(iii) f_X and f_Y have compatible supports (i.e., $S_X \subseteq S_Y$).

In this case, X can be simulated as follows by Accept-Reject method. First, we generate y from $Y \sim f_Y$ and, independently, we generate u from $U \sim U(0, 1)$. If

$$u \leq \frac{f_X(y)}{c\, f_Y(y)},$$

then we set $x = y$. If the inequality is not satisfied, we then discard/reject y and u and start again (Robert and Casella, 2010).

Details

Package `AR` provides a useful tool for teaching students to understand the theoritical idea behind the `Accept-Reject method`. This package works with only one function, i.e. function `AR.Sim` which can generate random sample/vector on the basis of the `Accept-Reject method`.

Author(s)

Abbas Parchami

Maintainer: Abbas Parchami <parchami@uk.ac.ir>

References

Iacus, S.M., Simulation and Inference for Stochastic Differential Equations: With R Examples, Springer, New York (2008).

Jones, O., Maillardet, R, Robinson, A., Introduction to Scientific Programming and Simulation Using R, Chapman & Hall/CRC, Boca Raton (2009).

Robert, C.P., Casella, G., Introducing Monte Carlo Methods with R, New York: Springer (2010).

Vasishth, S., Broe, M., The Foundations of Statistics: A Simulation-based Approach, Springer (2010).

Wikipedia, the free encyclopedia, Rejection sampling, https://en.wikipedia.org/wiki/Rejection_sampling

---

AR.Sim *Graphical Visualization for Accept-Reject Method*

Description

Package AR provides a graphical presentation for Accept-Reject method by drawing three figures which their explanations are as follow:

Explanation of Figure 1:

Moreover, even when the Rejection Accept-Reject method is applied, it is always hard to optimize the constant c for the likelihood ratio. Although, the algorithm works with a bigger constant c (with respect to optimal/minimum possible c), but increasing c cause high rejection rate and the algorithm can be very in-efficient. The first figure show three curves $f_X(x)$, $f_Y(x)$ and $\frac{f_X(x)}{f_Y(x)}$. Moreover, the optimum c (minimum possible c, such that $\frac{f_X(x)}{f_Y(x)} \leq c$) calculated as the maximum height of the curve $\frac{f_X(x)}{f_Y(x)}$, which is also shown on the first figure.

Explanation of Figure 2:

To visualize the motivation behind the Acceptance-Rejection method, imagine graphing curve $\frac{f_X(y)}{c\, f_Y(y)}$ onto a large rectangular board and throwing darts at it. Assume that the x-positions of these darts/points are uniformly distributed around the board and the distribution of y-positions of them are based on Y distribution. Now, remove all of the darts/points that are outside the area under the curve $\frac{f_X(y)}{c\, f_Y(y)}$. The x-positions of the remaining darts will be distributed according to the random variable's density of X within the area under the curve. Since, it can be prove that

$$P\left[Y \leq y \mid U \leq \frac{f_X(Y)}{c\, f_Y(Y)}\right] = P\left(X \leq x\right).$$

Explanation of Figure 3:

For another graphical presentation of the motivation behind the Acceptance-Rejection method, assumes that the considered board (which is presented in explanation of Figure 2) is not necessarily rectangular but is shaped according to some distribution that we know how to generate sample from it ($c.f_Y(y)$). Therefore, if y-positions of random points/darts be equal to $u.c.f_Y(y)$, then all darts/points will be land under the curve $c.f_Y(y)$. The acceptance condition in the Acceptance-Rejection method is

$$u \leq \frac{f_X(y)}{c\, f_Y(y)},$$

or equivalently

$$u.c.f_Y(y) \leq f_X(y),$$

and it means that after omitting the extra/red random darts/points from the board (which are not satisfy in the acceptance condition), the x-positions of the remaining darts/points will be distributed according to the distribution of X.

Usage

```
AR.Sim(n, f_X, Y.dist, Y.dist.par, xlim = c(0, 1), S_X = xlim, Rej.Num = TRUE,
       Rej.Rate = TRUE, Acc.Rate = TRUE)
```

Arguments

n	The number/length of data which must be generated/simulated from f_X density.
f_X	The density f_X of interest for simulation (called the target density)
Y.dist	The distribution name of the random variable Y, which used to generate the random data from f_Y. Precisely, Y.dist is the name of f_Y density which is match with DISTRIB Package. For example, use Y.dist = "norm", when $Y \sim N(\mu, \sigma^2)$.
Y.dist.par	A vector of Y distribution parameters with considered ordering in stats package and also is match with DISTRIB Package. For example, use Y.dist.par = c(2,3), when $Y \sim N(\mu = 2, \sigma^2 = 9)$.
xlim	NULL or a numeric vector of length 2; if non-NULL it provides the defaults for c(from, to) and, unless add=TRUE, selects the x-limits of the available plot. Its default is xlim=c(0,1).
S_X	The support of X with default S_X = xlim , which is needed for calculating the optimum value of constant c.
Rej.Num	A logical argument with default TRUE for calculate the number of rejections in Accept-Reject method. If Rej.Num = FALSE, then the number of rejections is not reported.
Rej.Rate	A logical argument with default TRUE for calculate the ratio of rejections in Accept-Reject method (i.e. Rej.Num / n). If Rej.Rate = FALSE, then the ratio of rejections is not reported.
Acc.Rate	A logical argument with default TRUE for calculate the ratio of acceptances in Accept-Reject method (i.e. 1 - Rej.Rate). If Acc.Rate = FALSE, then the ratio of acceptances is not reported.

Value

A vector of generated/simulated data from random variable X with length n.

Optimum value for c, i.e. the minimum possible value for c.

References

Robert, C.P., Casella, G., Introducing Monte Carlo Methods with R, New York: Springer (2010).

Wikipedia, the free encyclopedia, Rejection sampling, https://en.wikipedia.org/wiki/Rejection_sampling

Examples

```
# Example 1:
data = AR.Sim( n = 150,
         f_X = function(y){dbeta(y,2.7,6.3)},
         Y.dist = "unif", Y.dist.par = c(0,1),
```

```
          Rej.Num = TRUE,
          Rej.Rate = TRUE,
          Acc.Rate = FALSE
          )

# QQ-plot
q <- qbeta(ppoints(100), 2.7, 6.3)
qqplot(q, data, cex=0.6, xlab="Quantiles of Beta(2.7,6.3)",
       ylab="Empirical Quantiles of simulated data")
abline(0, 1, col=2)


# ------------------------------------------------------
# Example 2: From Page 54 of (Robert and Casella, 2009)
f_X = function(x) dbeta(x,2.7,6.3)
Simulation1 <- AR.Sim(n=300, f_X, Y.dist = "unif", Y.dist.par = c(0,1))
Simulation2 <- AR.Sim(n=2000, f_X, Y.dist="beta", Y.dist.par=c(2,6) )
Simulation3 <- AR.Sim(n=1000, f_X, Y.dist="beta", Y.dist.par=c(1.5,3.7) )
Simulation4 <- AR.Sim(n=250, f_X, Y.dist="norm", Y.dist.par=c(.5,.2) )
Simulation5 <- AR.Sim(n=200,  f_X, Y.dist="exp", Y.dist.par=3 )
Simulation6 <- AR.Sim( 400 ,  f_X, Y.dist="gamma", Y.dist.par=c(2,5) )

hist(Simulation1, prob=TRUE)#, col="gray20")
hist(Simulation2, prob=TRUE, add=TRUE, col="gray35")
hist(Simulation3, prob=TRUE, add=TRUE, col="gray60")
hist(Simulation4, prob=TRUE, add=TRUE, col="gray75")
hist(Simulation5, prob=TRUE, add=TRUE, col="gray85")
hist(Simulation6, prob=TRUE, add=TRUE, col="gray100")
curve(f_X(x), add=TRUE, col=2, lty=2, lwd=3)

#compare empirical and theoretical percentiles:
p <- seq(.1, .9, .1)
Qhat1 <- quantile(Simulation1, p) #Empirical quantiles of simulated sample
Qhat2 <- quantile(Simulation2, p) #Empirical quantiles of simulated sample
Qhat3 <- quantile(Simulation3, p) #Empirical quantiles of simulated sample
Qhat4 <- quantile(Simulation4, p) #Empirical quantiles of simulated sample
Qhat5 <- quantile(Simulation5, p) #Empirical quantiles of simulated sample
Qhat6 <- quantile(Simulation6, p) #Empirical quantiles of simulated sample
Q <- qbeta(p, 2.7, 6.3)    #Theoretical quantiles of Be(2.7,6.3)
round( rbind(Q, Qhat1, Qhat2, Qhat3, Qhat4, Qhat5, Qhat6), 3)

# Compute p-value of Kolmogorov-Smirnov test:
ks.test(Simulation1, "pbeta", 2.7, 6.3)$p.value
ks.test(Simulation2, "pbeta", 2.7, 6.3)$p.value
ks.test(Simulation3, "pbeta", 2.7, 6.3)$p.value
ks.test(Simulation4, "pbeta", 2.7, 6.3)$p.value
ks.test(Simulation5, "pbeta", 2.7, 6.3)$p.value
ks.test(Simulation6, "pbeta", 2.7, 6.3)$p.value


# ------------------------------------------------------
# Example 3: Simulate Truncated N(5,2^2) at l=3 and r=14 in left and rigth sides, respectively.
```

```
mu = 5
sigma = 2
l = 3
r = 14
n = 400
f_X = function(x)  dnorm(x,mu,sigma) *
                   as.integer(l<x & x<r) / (pnorm(r,mu,sigma)-pnorm(l,mu,sigma))

Sim1 <- AR.Sim(n, f_X, S_X=c(l,r), Y.dist="norm", Y.dist.par=c(5,4), xlim=c(l-1,r+1) )
head(Sim1, 15)
hist(Sim1, prob=TRUE, col="lightgreen")
curve(f_X(x), add=TRUE, col=2, lty=2, lwd=3) # Truncated pdf of N(5,2^2) at l and r
c2 = 1 / (pnorm(r,mu,sigma)-pnorm(l,mu,sigma)) ; c2 #See page 15 jozve
```

Index