Package 'TruncatedNormal'

November 10, 2015

Type Package

Title Truncated Multivariate Normal

Version 1.0

Date 2015-11-08

Author Zdravko I. Botev

Maintainer Zdravko Botev <botev@unsw.edu.au>

Description A collection of functions to deal with the truncated univariate and multivariate normal distributions.

License GPL-3

Depends randtoolbox

NeedsCompilation no

Repository CRAN

Date/Publication 2015-11-10 11:19:19

R topics documented:

`TruncatedNormal-package` *Truncated Normal Distribution Toolbox*

Description

The routines include:

- generator of **independent and identically distributed** random vectors from the truncated univariate and multivariate distributions;
- (Quasi-) Monte Carlo estimator and a **deterministic upper bound** of the cumulative distribution function of the multivariate normal;
- algorithm for the accurate computation of the quantile function of the normal distribution in the extremes of its tails.

Details

Package:	Truncated Normal
Type:	Package
Version:	1.0
Date:	2015-11-08
License:	GPL-3

`mvNcdf(l,u,Sig,n)` uses a Monte Carlo sample of size n to estimate the cumulative distribution function, $Pr(l < X < u)$, of the d-dimensional multivariate normal with zero-mean and covariance Σ, that is, X has $N(0, \Sigma)$ distribution;

`mvNqmc(l,u,Sig,n)` provides a Quasi Monte Carlo algorithm for medium dimensions (say, $d < 50$), in addition to the faster Monte Carlo algorithm in `mvNcdf`;

`mvrandn(l,u,Sig,n)` simulates n **identically and independently distributed** random vectors X from $N(0, \Sigma)$, conditional on $l < X < u$;

`norminvp(p,l,u)` computes the quantile function at $0 \leq p \leq 1$ of the univariate $N(0, 1)$ distribution truncated to $[l, u]$, and with high precision in the tails;

`trandn(l,u)` is a fast random number generator from the univariate $N(0, 1)$ distribution truncated to $[l, u]$.

Author(s)

Z. I. Botev, email: <botev@unsw.edu.au> and web page: http://web.maths.unsw.edu.au/~zdravkobotev/

References

- Z. I. Botev (2015), *The Normal Law Under Linear Restrictions: Simulation and Estimation via Minimax Tilting*, submitted to JRSS(B)

- Z. I. Botev and P. L'Ecuyer (2015), *Efficient Estimation and Simulation of the Truncated Multivariate Student-t Distribution*, Proceedings of the 2015 Winter Simulation Conference, Huntington Beach, CA, USA
- Gibson G. J., Glasbey C. A., Elston D. A. (1994), *Monte Carlo evaluation of multivariate normal integrals and sensitivity to variate ordering*, In: Advances in Numerical Methods and Applications, pages 120–126

See Also

Matlab toolbox: http://web.maths.unsw.edu.au/~zdravkobotev/

Examples

```
trandn(4,5) # generate a variable from the truncated normal on [4,5]
```

---

lupus *Latent Membranous Lupus Nephritis Dataset*

Description

The data represents two clinical measurements (covariates), which are used to predict the occurrence of latent membranous lupus nephritis. The dataset consists of measurements on 55 patients of which 18 have been diagnosed with latent membranous lupus.

Usage

```
data("lupus")
```

Format

The format is: "response" "const" "x1" "x2"

Details

The data was transcribed manually into this format from Table 1, page 22, of Dyk and Meng (2001).

References

D. A. van Dyk and X.-L. Meng (2001) *The art of data augmentation (with discussion)*. Journal of Computational and Graphical Statistics, volume 10, pages 1-50.

See Also

See Also as mvrandn, which uses this dataset.

Examples

```
data("lupus")
```

mvNcdf *truncated multivariate normal cumulative distribution*

Description

computes an estimator and a deterministic upper bound of the probability $Pr(l < X < u)$, where X is a zero-mean multivariate normal vector with covariance matrix Σ, that is, X is drawn from $N(0, \Sigma)$ infinite values for vectors u and l are accepted; Monte Carlo method uses sample size n;

Usage

```
mvNcdf(l, u, Sig, n)
```

Arguments

`l`	lower truncation limit
`u`	upper truncation limit
`Sig`	covariance matrix of $N(0, \Sigma)$
`n`	Monte Carlo simulation effort — the larger the n, the smaller the relative error of the estimator.

Details

Suppose you wish to estimate $p = Pr(l < AX < u)$, where A is a full rank matrix and X is drawn from $N(\mu, \Sigma)$, then you simply compute $p = Pr(l - A\mu < AY < u - A\mu)$, where Y is drawn from $N(0, A\Sigma A^\top)$.

Value

`est` with structure

`$prob`	estimated value of probability $Pr(l < X < u)$
`$relErr`	estimated relative error of estimator
`$upbnd`	theoretical upper bound on true $Pr(l < X < u)$

Note

For small dimensions, say $d < 50$, better accuracy may be obtained by using the (usually slower) quasi-Monte Carlo version `mvNqmc` of this algorithm.

Author(s)

Z. I. Botev, email: <botev@unsw.edu.au> and web page: http://web.maths.unsw.edu.au/~zdravkobotev/

References

Z. I. Botev (2015), *The Normal Law Under Linear Restrictions: Simulation and Estimation via Minimax Tilting*, submitted to JRSS(B)

See Also

See also: `mvNqmc` and `mvrandn`

Examples

```
d=15;l=1:d;u=l+Inf;
Sig=matrix(rnorm(d^2),d,d)*2;Sig=Sig%*%t(Sig)
n=10^3 # size of simulation effort
x=mvNcdf(l,u,Sig,10^4) # compute the probability
print(x)
```

---

mvNqmc *truncated multivariate normal cumulative distribution (quasi-Monte Carlo)*

Description

computes an estimator and a deterministic upper bound of the probability $Pr(l < X < u)$, where X is a zero-mean multivariate normal vector with covariance matrix Σ, that is, X is drawn from $N(0, \Sigma)$ infinite values for vectors u and l are accepted;

Usage

```
mvNqmc(l, u, Sig, n)
```

Arguments

`l`	lower truncation limit
`u`	upper truncation limit
`Sig`	covariance matrix of $N(0, \Sigma)$
`n`	total randomized quasi-Monte Carlo simulation effort; higher values yield more accurate results;

Details

Quasi-Monte Carlo version: This version uses a Quasi Monte Carlo (QMC) pointset of size `ceiling(n/12)` and estimates the relative error using 12 independent randomized QMC estimators; QMC is slower than ordinary Monte Carlo, but is also likely to be more accurate when $d < 50$. For high dimensions, say $d > 50$, you may obtain the same accuracy using the (typically faster) `mvNcdf`.

The non-zero mean case, that is, $N(\mu, \Sigma)$: Suppose you wish to estimate $p = Pr(l < AX < u)$, where A is a full rank matrix and X drawn from $N(\mu, \Sigma)$. Then, you simply compute $p = Pr(l - A\mu < AY < u - A\mu)$, where Y is drawn from $N(0, A\Sigma A^\top)$.

Value

`est` with structure

`$prob`	estimated value of probability $Pr(l < X < u)$
`$relErr`	estimated relative error of estimator
`$upbnd`	theoretical upper bound on true $Pr(l < X < u)$

Author(s)

Z. I. Botev, email: <botev@unsw.edu.au> and web page: http://web.maths.unsw.edu.au/~zdravkobotev/

References

Z. I. Botev (2015), *The Normal Law Under Linear Restrictions: Simulation and Estimation via Minimax Tilting*, submitted to JRSS(B)

See Also

See also: `mvNcdf` and `mvrandn`

Examples

```
d=15;l=1:d;u=l+Inf;
Sig=matrix(rnorm(d^2),d,d)*2;Sig=Sig%*%t(Sig)
n=10^3 # size of simulation effort
x=mvNqmc(l,u,Sig,10^4)
print(x)
```

---

mvrandn *truncated multivariate normal generator*

Description

simulates n **identically and independently distributed** random vectors from the d-dimensional $N(0, \Sigma)$ distribution (zero-mean normal with covariance Σ) conditional on $l < X < u$; infinite values for l and u are accepted;

Usage

```
mvrandn(l, u, Sig, n)
```

Arguments

`l`	lower truncation limit
`u`	upper truncation limit
`Sig`	Covariance matrix of $N(0, \Sigma)$
`n`	number of simulated vectors

Details

Bivariate normal: Suppose we wish to simulate a bivariate X from $N(\mu, \Sigma)$, conditional on $X_1 - X_2 < -6$. We can recast this as the problem of simulation of Y from $N(0, A\Sigma A^\top)$ (for an appropriate matrix A) conditional on $l - A\mu < Y < u - A\mu$ and then setting $X = \mu + A^{-1}Y$. See the example code below.

Exact posterior simulation for Probit regression: Consider the Bayesian Probit Regression model applied to the lupus dataset. Let the prior for the regression coefficients β be $N(0, \nu^2 I)$. Then, to simulate from the Bayesian posterior exactly, we first simulate Z from $N(0, \Sigma)$, where $\Sigma = I + \nu^2 X X^\top$, conditional on $Z \geq 0$. Then, we simulate the posterior regression coefficients, β, of the Probit regression by drawing $(\beta|Z)$ from $N(CX^\top Z, C)$, where $C^{-1} = I/\nu^2 + X^\top X$. See the example code below.

Value

output is a d by n array storing the random vectors, X, drawn from $N(0, \Sigma)$, conditional on $l < X < u$;

Note

Algorithm may not work or be very inefficient if Σ is close to being rank deficient.

Author(s)

Z. I. Botev, email: <botev@unsw.edu.au> and web page: http://web.maths.unsw.edu.au/~zdravkobotev/

References

Z. I. Botev (2015), *The Normal Law Under Linear Restrictions: Simulation and Estimation via Minimax Tilting*, submitted to JRSS(B)

See Also

See also: mvNqmc and mvNcdf

Examples

```
# Bivariate example.

Sig=matrix(c(1,0.9,0.9,1),2,2);mu=c(-3,0);l=c(-Inf,-Inf);u=c(-6,Inf);A=matrix(c(1,0,-1,1),2,2);
 n=10^3; # number of sampled vectors
 Y=mvrandn(l-A%*%mu,u-A%*%mu,A%*%Sig%*%t(A),n);
 X=rep(mu,n)+solve(A,diag(2))%*%Y; # now apply the inverse map as explained above
 plot(X[1,],X[2,]) # provide a scatterplot of exactly simulated points

# Exact Bayesian Posterior Simulation Example.

data("lupus"); # load lupus data
Y = lupus[,1]; # response data
X = lupus[,-1]  # construct design matrix
```

```
m=dim(X)[1]; d=dim(X)[2]; # dimensions of problem
X=diag(2*Y-1)%*%X; # incorporate response into design matrix
nu=sqrt(10000); # prior scale parameter
C=solve(diag(d)/nu^2+t(X)%*%X);
L=t(chol(t(C))); # lower Cholesky decomposition
Sig=diag(m)+nu^2*X%*%t(X); # this is covariance of Z given beta
l=rep(0,m);u=rep(Inf,m);
est=mvNcdf(l,u,Sig,10^3); # estimate acceptance probability of Crude Monte Carlo
print(est$upbnd/est$prob) # estimate the reciprocal of acceptance probability
n=10^4 # number of iid variables
z=mvrandn(l,u,Sig,n); # sample exactly from auxiliary distribution
beta=L%*%matrix(rnorm(d*n),d,n)+C%*%t(X)%*%z; # simulate beta given Z and plot boxplots of marginals
boxplot(t(beta)) # plot the boxplots of the marginal distribution of the coefficients in beta
print(rowMeans(beta)) # output the posterior means
```

`norminvp` *normal quantile function with precision*

Description

computes with tail-precision the quantile function of the standard normal distribution at $0 \leq p \leq 1$, and truncated to the interval $[l, u]$; Infinite values for vectors l and u are accepted;

Usage

```
norminvp(p, l, u)
```

Arguments

`p`	quantile at $0 \leq p \leq 1$
`l`	lower truncation limit
`u`	upper truncation limit

Details

Suppose we wish to simulate a random variable Z drawn from $N(\mu, \sigma^2)$ and conditional on $l < Z < u$ using the inverse transform method. To achieve this, first compute `X=norminvp(runif(1),(l-mu)/sig,(u-mu)/sig` and then set `Z=mu+sig*X`

Value

A real number – the quantile value of the truncated normal distribution.

Note

If you wish to simulate truncated normal variables fast, use `trandn`. Using `norminvp` is advisable only when needed, for example, in quasi-Monte Carlo or antithetic sampling, where the inverse transform method is unavoidable.

Author(s)

Z. I. Botev, email: <botev@unsw.edu.au> and web page: http://web.maths.unsw.edu.au/~zdravkobotev/

References

Z. I. Botev (2015), *The Normal Law Under Linear Restrictions: Simulation and Estimation via Minimax Tilting*, submitted to JRSS(B)

See Also

See Also as trandn

Examples

```
d=150;l=1:d;u=l+Inf;
x=norminvp(runif(d),l,u) # simulate via inverse transform method
```

---

trandn *fast truncated normal generator*

Description

efficient state-of-the-art generator of a vector of length(l)=length(u) from the standard multivariate normal distribution, truncated over the region $[l, u]$; infinite values for u and l are accepted;

Usage

```
trandn(l,u)
```

Arguments

l	lower truncation limit
u	upper truncation limit

Details

If you wish to simulate a random variable Z drawn from $N(\mu, \sigma^2)$ conditional on $l < Z < u$, then first simulate X=trandn((l-mu)/sig,(u-mu)/sig) and set Z=mu+sig*X;

Value

random variable(s) drawn from the truncated normal distribution

Note

Use norminvp for the (slower) inverse transform method of simulating truncated normal variables.

Author(s)

Z. I. Botev, email: <botev@unsw.edu.au> and web page: http://web.maths.unsw.edu.au/~zdravkobotev/

References

Z. I. Botev (2015), *The Normal Law Under Linear Restrictions: Simulation and Estimation via Minimax Tilting*, submitted to JRSS(B)

See Also

See also: norminvp

Examples

```
d=150;l=1:d;u=l+Inf;
x=trandn(l,u)
print(x)
```

Index