

Package ‘mvgb’

June 22, 2022

Title Multivariate Probabilities of Scale Mixtures of Multivariate Normal Distributions via the Genz and Bretz (2002) QRSVN Method

Version 0.0.3

Description Generates multivariate subgaussian stable probabilities using the QRSVN algorithm as detailed in Genz and Bretz (2002) <[DOI:10.1198/106186002394](https://doi.org/10.1198/106186002394)> but by sampling positive stable variates not chi/sqrt(nu).

Depends R (>= 3.4.0)

License LGPL (>= 2.1)

Encoding UTF-8

RoxygenNote 7.1.2

URL <https://github.com/swihart/mvgb>

BugReports <https://github.com/swihart/mvgb/issues>

NeedsCompilation yes

Author Alan Genz [aut] (wrote mvtdstpack.f),
Bruce Swihart [aut, cre] (<<https://orcid.org/0000-0002-4216-9942>>)

Maintainer Bruce Swihart <bruce.swihart@gmail.com>

Repository CRAN

Date/Publication 2022-06-22 08:20:06 UTC

R topics documented:

| | |
|--------------|----------|
| mvgb | 2 |
| pmvss | 2 |
| Index | 6 |

| | |
|------|--|
| mvgb | <i>Multivariate Probabilities of Scale Mixtures of Multivariate Normal Distributions</i> |
|------|--|

Description

The QRVSN algorithm is used in a broader context: using the Genz and Bretz (2002) algorithm to calculate multivariate distribution probabilities.

Multivariate Subgaussian Stable Distribution

[pmvss](#) – multivariate subgaussian stable distribution probabilities

| | |
|-------|--|
| pmvss | <i>Multivariate Subgaussian Stable Distribution Probabilities (via positive stable variates)</i> |
|-------|--|

Description

Computes the the distribution function of the multivariate subgaussian stable distribution for arbitrary limits, alpha, shape matrices, and location vectors. This function is unlike `mvtnorm::pmvt` in two ways:

1. The QRVSN method is used for every dimension n (including bivariate and trivariate),
2. The QRVSN method on positive stable variates, not $\chi/\sqrt{\nu}$.

Usage

```
pmvss(
  lower,
  upper,
  alpha,
  Q,
  delta = rep(0, NROW(Q)),
  maxpts = 25000,
  abseps = 0.001,
  releps = 0
)
```

Arguments

| | |
|-------|--|
| lower | lower bounds of integration must be length n , finite |
| upper | upper bounds of integration must be length n , finite |
| alpha | real number between 0 and 2 that cannot have more than 6 digits precision. 0.123456 is okay; 0.1234567 is not. Thus alpha in [0.000001, 1.999999]. |

| | |
|--------|--|
| Q | shape matrix |
| delta | location vector must have length equal to the number of rows of Q. Defaults to the 0 vector. |
| maxpts | (description from FORTRAN code) INTEGER, maximum number of function values allowed. This parameter can be used to limit the time. A sensible strategy is to start with MAXPTS = 1000*N, and then increase MAXPTS if ERROR is too large. (description from mvtnorm::GenzBretz) maximum number of function values as integer. The internal FORTRAN code always uses a minimum number depending on the dimension. (for example 752 for three-dimensional problems). |
| abseps | absolute error tolerance |
| releps | relative error tolerance as double. |

Value

Returns the variables from the MVTDST function (QRVSN algorithm):

| | |
|--------|---|
| N | INTEGER, the number of variables. |
| NU | INTEGER, the number of degrees of freedom. If NU < 1, then an MVN probability is computed. |
| LOWER | DOUBLE PRECISION, array of lower integration limits. |
| UPPER | DOUBLE PRECISION, array of upper integration limits. |
| INFIN | INTEGER, array of integration limits flags: if INFIN(I) < 0, Ith limits are (-infinity, infinity); if INFIN(I) = 0, Ith limits are (-infinity, UPPER(I)]; if INFIN(I) = 1, Ith limits are [LOWER(I), infinity); if INFIN(I) = 2, Ith limits are [LOWER(I), UPPER(I)]. |
| CORREL | DOUBLE PRECISION, array of correlation coefficients; the correlation coefficient in row I column J of the correlation matrix should be stored in CORREL(J + ((I-2)*(I-1))/2), for J < I. The correlation matrix must be positive semi-definite. |
| DELTA | DOUBLE PRECISION, array of non-centrality parameters. |
| MAXPTS | INTEGER, maximum number of function values allowed. This parameter can be used to limit the time. A sensible strategy is to start with MAXPTS = 1000*N, and then increase MAXPTS if ERROR is too large. |
| ABSEPS | DOUBLE PRECISION absolute error tolerance. |
| RELEPS | DOUBLE PRECISION relative error tolerance. |

```
error DOUBLE PRECISION estimated absolute error,  
with 99% confidence level.  
  
value DOUBLE PRECISION estimated value for the integral  
  
inform INTEGER, termination status parameter:  
if INFORM = 0, normal completion with ERROR < EPS;  
if INFORM = 1, completion with ERROR > EPS and MAXPTS  
function values used; increase MAXPTS to  
decrease ERROR;  
if INFORM = 2, N > 1000 or N < 1.  
if INFORM = 3, correlation matrix not positive semi-definite.  
  
RND ignore; this initializes RNG
```

References

Genz, A. and Bretz, F. (2002), Methods for the computation of multivariate t-probabilities. *Journal of Computational and Graphical Statistics*, **11**, 950–971.

<http://www.math.wsu.edu/faculty/genz/homepage>

<http://www.math.wsu.edu/faculty/genz/software/fort77/mvtdstpack.f>

Examples

```
Q = structure(c(1, 0.85, 0.85, 0.85, 0.85,  
              0.85, 1, 0.85, 0.85, 0.85,  
              0.85, 0.85, 1, 0.85, 0.85,  
              0.85, 0.85, 0.85, 1, 0.85,  
              0.85, 0.85, 0.85, 0.85, 1),  
             .Dim = c(5L,5L))  
  
## default maxpts=25000 doesn't finish with error < abseps  
mvgb::pmvss(lower=rep(-1,5),  
            upper=rep(1,5),  
            alpha=1,  
            Q=Q,  
            maxpts=25000)[c("value","inform","error")]  
  
## increase maxpts to get inform value 0 (that is, error < abseps)  
mvgb::pmvss(lower=rep(-1,5),  
            upper=rep(1,5),  
            alpha=1,  
            Q=Q,  
            maxpts=25000*350)[c("value","inform","error")]  
  
set.seed(10)  
shape_matrix <- structure(c(1, 0.9, 0.9, 0.9, 0.9, 0.9, 1, 0.9, 0.9, 0.9, 0.9,  
                          0.9, 1, 0.9, 0.9, 0.9, 0.9, 0.9, 1, 0.9, 0.9, 0.9, 0.9,  
                          0.9, 0.9, 1, 0.9, 0.9, 0.9, 0.9, 0.9, 1, 0.9, 0.9, 0.9,  
                          0.9, 0.9, 0.9, 1, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 1),
```

```
1), .Dim = c(5L, 5L))  
  
mvgb::pmvss(lower=rep(-2,5),  
            upper=rep(2,5),  
            alpha=1.7,  
            Q=shape_matrix,  
            delta=rep(0,5),  
            maxpts=25000*30)[c("value", "inform", "error")]
```

Index

* **distribution**

pmvss, [2](#)

mvgb, [2](#)

pmvss, [2, 2](#)