Package 'TMNImoment'

June 18, 2023

Type Package

Title Multivariate Normal/Independent Distributions with Double Truncation

Version 0.1.0

Author Wan-Lun Wang and Tsung-I Lin

Maintainer Wan-Lun Wang <wangwl@gs.ncku.edu.tw>

Description This package provides functions related to the multivariate normal/independent (MNI) distributions with double truncations considered in Lin and Wang (2023). The considered MNI distributions contain the multivariate normal (MVN), multivariate t (MVT), multivariate slash (MSL), multivariate contaminated normal (MCN) and multivariate variance-gamma (MVG) distributions. The `dmni' and `pmni' functions calculate the probability density and distribution functions for the MNI distributions. The `dtmni' and `ptmni' calculate the the probability density and distribution functions for the truncated MNI distributions, and the `rtmni' function generates random number (vector) from the truncated MNI distributions. The `TMNI.moment' function computes the first two moments of doubly truncated MNI distributions theoretically.

License GPL-2

Depends cubature, mvtnorm, tmvtnorm, invgamma, Bessel

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

R topics documented:

TMNImoment-package *Multivariate Normal/Independent Distributions with Double Truncation*

Description

This package provides functions related to the multivariate normal/independent (MNI) distributions with double truncations considered in Lin and Wang (2023). The considered MNI distributions contain the multivariate normal (MVN), multivariate t (MVT), multivariate slash (MSL), multivariate contaminated normal (MCN) and multivariate variance-gamma (MVG) distributions. The 'dmni' and 'pmni' functions calculate the probability density and distribution functions for the MNI distributions. The 'dtmni' and 'ptmni' calculate the the probability density and distribution functions for the truncated MNI distributions, and the 'rtmni' function generates random number (vector) from the truncated MNI distributions. The 'TMNI.moment' function computes the first two moments of doubly truncated MNI distributions theoretically.

Details

Package:	TMNImoment
Type:	Package
Version:	1.0
Date:	2023-01-01
License:	GPL-2

Author(s)

Authors: Wan-Lun Wang and Tsung-I Lin
Maintainer: Wan-Lun Wang <wangwl@gs.ncku.edu.tw>

References

Tsung-I Lin and Wan-Lun Wang (2023). On moments of truncated multivariate normal/independent distributions. Submitted.

See Also

Related functions:
`dmni`: This function is to calculate the probability density function of the MNI distribution.
`pmni`: This function is to calcualte the distribution function of the MNI distribution.
`rtmni`: This function is to generate random number (vector) of the truncated MNI distribution.
`dtmni`: This function is to calculate the probability density function of the truncated MNI distribution.
`ptmni`: This function is to calcualte the distribution function of the truncated MNI distribution.
`TMNI.moment`: This function is to compute the first two moments of truncated MNI distributions theoretically.

mni *Density and Distribution Functions for Multivariate Normal/Independent (MNI) Distributions*

Description

These functions calculate the probability density and distribution functions of the MNI distribution with location vector equal to mu, scale-covariance matrix equal to Sigma and harmonizing parameters equal to nu (for MVT and MSL), nu and rho (for MCN) and alpha and beta (for MVG).

Usage

```
dmni(x, mu, Sigma=diag(length(mu)),
     nu=NULL, rho=NULL, alpha=NULL, beta=NULL,
     distr=c('MVN', 'MVT', 'MSL', 'MCN', 'MVG'))
pmni(lower=rep(-Inf, length(mu)), upper=rep(Inf, length(mu)),
     mu, Sigma=diag(length(mu)),
     nu=NULL, rho=NULL, alpha=NULL, beta=NULL,
     distr=c('MVN', 'MVT', 'MSL', 'MCN', 'MVG'))
```

Arguments

x	vector of quantiles of length p
mu	Location parameter vector of length p
Sigma	Nonsingular scale-covariance matrix, default is diag(length(mu))
nu	Harmonizing parameter, if distr='MVT', nu is the degrees of freedom; if distr='MSL', nu is the shape parameter; and if distr='MCN', 0<nu<1 is the mixing proportion
rho	Degree of contamination, must be located between 0 and 1
alpha	Shape parameter, if distr='MVG', alpha is required and must be a positive value
beta	Scale parameter, if distr='MVG', beta is required and must be a positive value
distr	Distributional options are 'MVN' (multivariate normal), 'MVT' (multivariate t), 'MSL' (multivariate slash), 'MCN' (multivariate contaminated normal) and 'MVG' (multivariate variance-gamma)
lower	Lower integration limits, a numeric vector of length p, default is rep(-Inf, length(mu))
upper	Upper integration limits, a numeric vector of length p, default is rep(Inf, length(mu))

Value

dmni	gives the density
pmni	gives the distribution function

See Also

tmni

Examples

```
# Test example 1 (p=1)
mu=0; Sigma=2
y1 = rtmni(n=1000, mu, Sigma=Sigma, distr='MVN')
y2 = rtmni(n=1000, mu, Sigma=Sigma, nu=5, distr='MVT')
y3 = rtmni(n=1000, mu, Sigma=Sigma, nu=2, distr='MSL')
y4 = rtmni(n=1000, mu, Sigma=Sigma, nu=0.25, rho=0.2, distr='MCN')
y5 = rtmni(n=1000, mu, Sigma=Sigma, alpha=2, beta=1, distr='MVG')
y6 = rtmni(n=1000, mu, Sigma=Sigma, alpha=1, beta=2, distr='MVG')

#win.graph(width=15, height=10)
par(mfrow=c(2,3), mar=c(3,2,3,0.5))
hist(y1, nclass=20, prob=TRUE, ylim=c(0,0.3), ylab='', col=5,
     main='Histogram of N samples')
curve(dmni(x, mu=mu, Sigma=Sigma, distr='MVN'),
      min(y1), max(y1), n=100, col=2, lwd=1.5, lty=1, add=TRUE)
legend("topright", c('MVN'), col=2, lty=1, bty='n')

hist(y2, nclass=20, prob=TRUE, ylim=c(0,0.3), ylab='', col=5,
     main='Histogram of T samples')
curve(dmni(x, mu=mu, Sigma=Sigma, nu=5, distr='MVT'),
      min(y2), max(y2), n=100, col=2, lwd=1.5, lty=1, add=TRUE)
curve(dmni(x, mu=mu, Sigma=Sigma, distr='MVN'),
      min(y2), max(y2), n=100, col=4, lwd=1, lty=2, add=TRUE)
legend("topright", c('MVT','MVN'), col=c(2,4), lty=c(1,2), bty='n')

hist(y3, nclass=20, prob=TRUE, ylim=c(0,0.3), ylab='', col=5,
     main='Histogram of SL samples')
curve(dmni(x, mu=mu, Sigma=Sigma, nu=2, distr='MSL'),
      min(y3), max(y3), n=100, col=2, lwd=1.5, lty=1, add=TRUE)
curve(dmni(x, mu=mu, Sigma=Sigma, distr='MVN'),
      min(y3), max(y3), n=100, col=4, lwd=1, lty=2, add=TRUE)
legend("topright", c('MSL','MVN'), col=c(2,4), lty=c(1,2), bty='n')

hist(y4, nclass=20, prob=TRUE, ylim=c(0,0.3), ylab='', col=5,
     main='Histogram of CN samples')
curve(dmni(x, mu=mu, Sigma=Sigma, nu=0.25, rho=0.2, distr='MCN'),
      min(y4), max(y4), n=100, col=2, lwd=1.5, lty=1, add=TRUE)
curve(dmni(x, mu=mu, Sigma=Sigma, distr='MVN'),
      min(y4), max(y4), n=100, col=4, lwd=1, lty=2, add=TRUE)
legend("topright", c('MCN','MVN'), col=c(2,4), lty=c(1,2), bty='n')

hist(y5, nclass=20, prob=TRUE, ylim=c(0,0.3), ylab='', col=5,
     main='Histogram of VG samples')
curve(dmni(x, mu=mu, Sigma=Sigma, alpha=2, beta=1, distr='MVG'),
      min(y5), max(y5), n=100, col=2, lwd=1.5, lty=1, add=TRUE)
curve(dmni(x, mu=mu, Sigma=Sigma, distr='MVN'),
      min(y5), max(y5), n=100, col=4, lwd=1, lty=2, add=TRUE)
legend("topright", c('MVG','MVN'), col=c(2,4), lty=c(1,2), bty='n')

hist(y6, nclass=20, prob=TRUE, ylim=c(0,0.5), ylab='', col=5,
     main='Histogram of DE samples')
curve(dmni(x, mu=mu, Sigma=Sigma, alpha=1, beta=2, distr='MVG'),
      min(y6), max(y6), n=100, col=2, lwd=1.5, lty=1, add=TRUE)
curve(dmni(x, mu=mu, Sigma=Sigma, distr='MVN'),
      min(y6), max(y6), n=100, col=4, lwd=1, lty=2, add=TRUE)
```

```
legend("topright", c('MDE','MVN'), col=c(2,4), lty=c(1,2), bty='n')

# Test example 2 (p=2)
mu=c(1,2)
Sigma=matrix(c(3,1,1,2), 2, 2)

Y1 = rtmni(n=500, mu, Sigma=Sigma, distr='MVN')
x1 = y1 = seq(min(Y1[,1]), max(Y1[,1]), length=30)
Y2 = rtmni(n=500, mu, Sigma=Sigma, nu=5, distr='MVT')
x2 = y2 = seq(min(Y2[,1]), max(Y2[,1]), length=30)
Y3 = rtmni(n=500, mu, Sigma=Sigma, nu=2, distr='MSL')
x3 = y3 = seq(min(Y3[,1]), max(Y3[,1]), length=30)
Y4 = rtmni(n=500, mu, Sigma=Sigma, nu=0.25, rho=0.2, distr='MCN')
x4 = y4 = seq(min(Y4[,1]), max(Y4[,1]), length=30)
Y5 = rtmni(n=500, mu, Sigma=Sigma, alpha=2, beta=1, distr='MVG')
x5 = y5 = seq(min(Y5[,1]), max(Y5[,1]), length=30)
Y6 = rtmni(n=500, mu, Sigma=Sigma, alpha=1, beta=2, distr='MVG')
x6 = y6 = seq(min(Y6[,1]), max(Y6[,1]), length=30)

den1 = den2 = den3 = den4 = den5 = den6 = matrix(NA, 30, 30)
for(i in 1: 30)for(j in 1: 30){
  den1[i,j] = dmni(c(x1[i], y1[j]), mu=mu, Sigma=Sigma, distr='MVN')
  den2[i,j] = dmni(c(x2[i], y2[j]), mu=mu, Sigma=Sigma, nu=5, distr='MVT')
  den3[i,j] = dmni(c(x3[i], y3[j]), mu=mu, Sigma=Sigma, nu=2, distr='MSL')
  den4[i,j] = dmni(c(x4[i], y4[j]), mu=mu, Sigma=Sigma, nu=0.25, rho=0.2, distr='MCN')
  den5[i,j] = dmni(c(x5[i], y5[j]), mu=mu, Sigma=Sigma, alpha=2, beta=1, distr='MVG')
  den6[i,j] = dmni(c(x6[i], y6[j]), mu=mu, Sigma=Sigma, alpha=1, beta=2, distr='MVG')
}

#win.graph(width=15, height=10)
par(mfrow=c(2,3), mar=c(3,2,3,0.5))
plot(Y1, pch=18, cex=0.5, xlab='Variable 1', ylab='Variable 2',
    las = 1, col = "gray30", main='MVN')
contour(x1, y1, den1, lty=2, col=2, drawlabels=FALSE, add=TRUE,
    levels = quantile(den1, prob=c(seq(0, 0.9, 0.2), seq(0.9, 1, 0.02))))
plot(Y2, pch=18, cex=0.5, xlab='Variable 1', ylab='Variable 2',
    las = 1, col = "gray30", main='MVT')
contour(x2, y2, den2, lty=2, col=2, drawlabels=FALSE, add=TRUE,
    levels = quantile(den2, prob=c(seq(0, 0.9, 0.2), seq(0.9, 1, 0.02))))
plot(Y3, pch=18, cex=0.5, xlab='Variable 1', ylab='Variable 2',
    las = 1, col = "gray30", main='MSL')
contour(x3, y3, den3, lty=2, col=2, drawlabels=FALSE, add=TRUE,
    levels = quantile(den3, prob=c(seq(0, 0.9, 0.2), seq(0.9, 1, 0.02))))
plot(Y4, pch=18, cex=0.5, xlab='Variable 1', ylab='Variable 2',
    las = 1, col = "gray30", main='MCN')
contour(x4, y4, den4, lty=2, col=2, drawlabels=FALSE, add=TRUE,
    levels = quantile(den4, prob=c(seq(0, 0.9, 0.2), seq(0.9, 1, 0.02))))
plot(Y5, pch=18, cex=0.5, xlab='Variable 1', ylab='Variable 2',
    las = 1, col = "gray30", main='MVG')
contour(x5, y5, den5, lty=2, col=2, drawlabels=FALSE, add=TRUE,
    levels = quantile(den5, prob=c(seq(0, 0.9, 0.2), seq(0.9, 1, 0.02))))
plot(Y6, pch=18, cex=0.5, xlab='Variable 1', ylab='Variable 2',
    las = 1, col = "gray30", main='MDE')
contour(x6, y6, den6, lty=2, col=2, drawlabels=FALSE, add=TRUE,
    levels = quantile(den6, prob=c(seq(0, 0.9, 0.2), seq(0.9, 1, 0.02))))
```

tmni *Generator, Density and Distribution Functions for Truncated Multivariate Normal/Independent (TMNI) Distributions*

Description

These functions provide the density and distribution functions and a random number (vector) generator for the truncated multivariate normal/independent distribution with location equal to mu and scale-covariance matrix Sigma, harmonizing parameters equal to nu (for MVT and MSL), nu and rho (for MCN) and alpha and beta (for MVG), and truncation region located between lower and upper.

Usage

```
rtmni(n, mu, Sigma=diag(length(mu)),
      nu=NULL, rho=NULL, alpha=NULL, beta=NULL,
      distr=c('MVN', 'MVT', 'MSL', 'MCN', 'MVG'),
      lower=rep(-Inf,length(mu)), upper=rep(Inf, length(mu)))
dtmni(x, mu, Sigma=diag(length(mu)),
      nu=NULL, rho=NULL, alpha=NULL, beta=NULL,
      distr=c('MVN', 'MVT', 'MSL', 'MCN', 'MVG'),
      a.low=rep(-Inf, length(mu)), a.upp=rep(Inf, length(mu)))
ptmni(lower=rep(-Inf, length(mu)), upper=rep(Inf, length(mu)),
      mu, Sigma=diag(length(mu)),
      nu=NULL, rho=NULL, alpha=NULL, beta=NULL,
      distr=c('MVN', 'MVT', 'MSL', 'MCN', 'MVG'),
      a.low=rep(-Inf, length(mu)), a.upp=rep(Inf, length(mu)))
```

Arguments

n	Number of random points to be sampled. Must be an integer >= 1
x	vector of quantiles of length p
mu	Location parameter vector of length p
Sigma	Nonsingular scale-covariance matrix, default is diag(length(mu))
nu	Harmonizing parameter, if distr='MVT', nu is the degrees of freedom and nu>4 is required to calculate the second moment of TMVT distribution. If distr='MSL', nu is the shape parameter and nu>1 is required to calculate the second moment of TMSL distribution. If distr='MCN', nu is the mixing proportion and 0<nu<1
rho	Degree of contamination, must be located between 0 and 1
alpha	Shape parameter, if distr='MVG', alpha is required and must be a positive value
beta	Scale parameter, if distr='MVG', beta is required and must be a positive value
distr	Distributional options are 'MVN' (multivariate normal), 'MVT' (multivariate t), 'MSL' (multivariate slash), 'MCN' (multivariate contaminated normal) and 'MVG' (multivariate variance-gamma)
lower	Lower integration limits, a numeric vector of length p, default is rep(-Inf, length(mu))
upper	Upper integration limits, a numeric vector of length p, default is rep(Inf, length(mu))
a.low	Vector of lower truncation points of length p, default is rep(-Inf,length(mu))
a.upp	Vector of upper truncation points of length p, default is rep(Inf, length(mu))

Value

rtmni	generates an n by p matrix of random samples
dtmcn	gives the density
ptmcn	gives the distribution function

See Also

dmni, pmni, TMNI.moment

Examples

```
# Test Example
mu=c(1,2)
Sigma=matrix(c(3,1,1,2), 2, 2)
ubd = 1
a2.low=c(-Inf,-Inf); a2.upp=c(ubd,ubd)
a3.low=c(-ubd,-ubd); a3.upp=c(Inf,Inf)
a4.low=c(-ubd,-ubd); a4.upp=c(ubd,ubd)

x1 = x2 = seq(-5, 5, length=25)
m = length(x1)
yMVN1 = yMVN2 = yMVN3 = yMVN4 =
yMVT1 = yMVT2 = yMVT3 = yMVT4 =
yMSL1 = yMSL2 = yMSL3 = yMSL4 =
yMCN1 = yMCN2 = yMCN3 = yMCN4 =
yMVG1 = yMVG2 = yMVG3 = yMVG4 =
yMDE1 = yMDE2 = yMDE3 = yMDE4 = matrix(0, m, m)
for(i in 1: m)for(j in 1: m){
  x12 = c(x1[i],x2[j])
# MVN
  yMVN1[i, j] = dmni(x12,  mu, Sigma, distr='MVN')
  yMVN2[i, j] = dtmni(x12, mu, Sigma, distr='MVN', a.low=a2.low, a.upp=a2.upp)
  yMVN3[i, j] = dtmni(x12, mu, Sigma, distr='MVN', a.low=a3.low, a.upp=a3.upp)
  yMVN4[i, j] = dtmni(x12, mu, Sigma, distr='MVN', a.low=a4.low, a.upp=a4.upp)
# MVT
  yMVT1[i, j] = dmni(x12,  mu, Sigma, nu=5, distr='MVT')
  yMVT2[i, j] = dtmni(x12, mu, Sigma, nu=5, distr='MVT', a.low=a2.low, a.upp=a2.upp)
  yMVT3[i, j] = dtmni(x12, mu, Sigma, nu=5, distr='MVT', a.low=a3.low, a.upp=a3.upp)
  yMVT4[i, j] = dtmni(x12, mu, Sigma, nu=5, distr='MVT', a.low=a4.low, a.upp=a4.upp)
# MSL
  yMSL1[i, j] = dmni(x12,  mu, Sigma, nu=2, distr='MSL')
  yMSL2[i, j] = dtmni(x12, mu, Sigma, nu=2, distr='MSL', a.low=a2.low, a.upp=a2.upp)
  yMSL3[i, j] = dtmni(x12, mu, Sigma, nu=2, distr='MSL', a.low=a3.low, a.upp=a3.upp)
  yMSL4[i, j] = dtmni(x12, mu, Sigma, nu=2, distr='MSL', a.low=a4.low, a.upp=a4.upp)
# MCN
  yMCN1[i, j] = dmni(x12,  mu, Sigma, nu=0.25, rho=0.2, distr='MCN')
 yMCN2[i, j] = dtmni(x12, mu, Sigma, nu=0.25, rho=0.2, distr='MCN', a.low=a2.low, a.upp=a2.upp)
 yMCN3[i, j] = dtmni(x12, mu, Sigma, nu=0.25, rho=0.2, distr='MCN', a.low=a3.low, a.upp=a3.upp)
 yMCN4[i, j] = dtmni(x12, mu, Sigma, nu=0.25, rho=0.2, distr='MCN', a.low=a4.low, a.upp=a4.upp)
# MVG
  yMVG1[i, j] = dmni(x12,  mu, Sigma, alpha=2, beta=1, distr='MVG')
 yMVG2[i, j] = dtmni(x12, mu, Sigma, alpha=2, beta=1, distr='MVG', a.low=a2.low, a.upp=a2.upp)
 yMVG3[i, j] = dtmni(x12, mu, Sigma, alpha=2, beta=1, distr='MVG', a.low=a3.low, a.upp=a3.upp)
 yMVG4[i, j] = dtmni(x12, mu, Sigma, alpha=2, beta=1, distr='MVG', a.low=a4.low, a.upp=a4.upp)
# MDE
  yMDE1[i, j] = dmni(x12,  mu, Sigma, alpha=1, beta=2, distr='MVG')
```

```
  yMDE2[i, j] = dtmni(x12, mu, Sigma, alpha=1, beta=2, distr='MVG', a.low=a2.low, a.upp=a2.upp)
  yMDE3[i, j] = dtmni(x12, mu, Sigma, alpha=1, beta=2, distr='MVG', a.low=a3.low, a.upp=a3.upp)
  yMDE4[i, j] = dtmni(x12, mu, Sigma, alpha=1, beta=2, distr='MVG', a.low=a4.low, a.upp=a4.upp)
}

#win.graph(width=30, height=40)
par(mfrow=c(7,5), mar=c(2.5,2.25,0.25,0.5))
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext(expression(paste('(',-infinity,',',infinity,')')), 1, line=1, cex=1, font=2)
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext(expression(paste(A[1],'=(',-infinity,',',1,')')), 1, line=1, cex=1, font=2)
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext(expression(paste(A[2],'=(',-1,',',infinity,')')), 1, line=1, cex=1, font=2)
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext(expression(paste(A[3],'=(',-1,',',1,')')), 1, line=1, cex=1, font=2)

# MVN
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext('TMVN', 1, line=-4, cex=0.8, font=2)
contour(x1, x2, yMVN1, nlevels=5, lty=1, col=2, xlim=c(-5,5), ylim=c(-2.5,6), las=1)
contour(x1, x2, yMVN2, nlevels=5, lty=1, col=2, xlim=c(-5,1), ylim=c(-1,1.2), las=1)
contour(x1, x2, yMVN3, nlevels=5, lty=1, col=2, xlim=c(-1.5,6), ylim=c(0,5), las=1)
contour(x1, x2, yMVN4, nlevels=5, lty=1, col=2, xlim=c(-1.2,1), ylim=c(-0.2,1.2), las=1)

# MVT
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext('TMVT', 1, line=-4, cex=0.8, font=2)
contour(x1, x2, yMVT1, nlevels=5, lty=1, col=2, xlim=c(-5,5), ylim=c(-2.5,6), las=1)
contour(x1, x2, yMVT2, nlevels=5, lty=1, col=2, xlim=c(-5,1), ylim=c(-1,1.2), las=1)
contour(x1, x2, yMVT3, nlevels=5, lty=1, col=2, xlim=c(-1.5,6), ylim=c(0,5), las=1)
contour(x1, x2, yMVT4, nlevels=5, lty=1, col=2, xlim=c(-1.2,1), ylim=c(-0.2,1.2), las=1)

# MSL
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext('TMSL', 1, line=-4, cex=0.8, font=2)
contour(x1, x2, yMSL1, nlevels=5, lty=1, col=2, xlim=c(-5,5), ylim=c(-2.5,6), las=1)
contour(x1, x2, yMSL2, nlevels=5, lty=1, col=2, xlim=c(-5,1), ylim=c(-1,1.2), las=1)
contour(x1, x2, yMSL3, nlevels=5, lty=1, col=2, xlim=c(-1.5,6), ylim=c(0,5), las=1)
contour(x1, x2, yMSL4, nlevels=5, lty=1, col=2, xlim=c(-1.2,1), ylim=c(-0.2,1.2), las=1)

# MCN
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext('TMCN', 1, line=-4, cex=0.8, font=2)
contour(x1, x2, yMCN1, nlevels=5, lty=1, col=2, xlim=c(-5,5), ylim=c(-2.5,6), las=1)
contour(x1, x2, yMCN2, nlevels=5, lty=1, col=2, xlim=c(-5,1), ylim=c(-1,1.2), las=1)
contour(x1, x2, yMCN3, nlevels=5, lty=1, col=2, xlim=c(-1.5,6), ylim=c(0,5), las=1)
contour(x1, x2, yMCN4, nlevels=5, lty=1, col=2, xlim=c(-1.2,1), ylim=c(-0.2,1.2), las=1)

# MVG
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext('TMVG', 1, line=-4, cex=0.8, font=2)
contour(x1, x2, yMVG1, nlevels=5, lty=1, col=2, xlim=c(-5,5), ylim=c(-2.5,6), las=1)
contour(x1, x2, yMVG2, nlevels=5, lty=1, col=2, xlim=c(-5,1), ylim=c(-1,1.2), las=1)
contour(x1, x2, yMVG3, nlevels=5, lty=1, col=2, xlim=c(-1.5,6), ylim=c(0,5), las=1)
contour(x1, x2, yMVG4, nlevels=5, lty=1, col=2, xlim=c(-1.2,1), ylim=c(-0.2,1.2), las=1)
```

```
# MDE
plot(c(0, 1), c(0, 3), type='n', xaxt='n', yaxt='n', xlab='', ylab='', bty='n')
mtext('TMDE', 1, line=-4, cex=0.8, font=2)
contour(x1, x2, yMDE1, nlevels=5, lty=1, col=2, xlim=c(-5,5), ylim=c(-2.5,6), las=1)
contour(x1, x2, yMDE2, nlevels=5, lty=1, col=2, xlim=c(-5,1), ylim=c(-1,1.2), las=1)
contour(x1, x2, yMDE3, nlevels=5, lty=1, col=2, xlim=c(-1.5,6), ylim=c(0,5), las=1)
contour(x1, x2, yMDE4, nlevels=5, lty=1, col=2, xlim=c(-1.2,1), ylim=c(-0.2,1.2), las=1)
```

`TMNI.moment` *First Two Moments of Truncated Multivariate Normal/Independent (TMNI) Distributions*

Description

This function calculates the first two moments of the TMNI distribution with location vector, scale-covariance matrix, harmonizing parameter and truncation region.

Usage

```
TMNI.moment(mu, Sigma=diag(length(mu)),
            nu=NULL, rho=NULL, alpha=NULL, beta=NULL,
            distr=c('MVN', 'MVT', 'MSL', 'MCN', 'MVG'),
            a.low=rep(-Inf, length(mu)), a.upp=rep(Inf, length(mu)))
```

Arguments

`mu`	location number or vector
`Sigma`	Nonsingular scale-covariance matrix, default is diag(length(mu))
`nu`	Harmonizing parameter, if distr='MVT', nu is the degrees of freedom and nu>4 is required to calculate the second moment of TMVT distribution. If distr='MSL', nu is the shape parameter and nu>1 is required to calculate the second moment of TMSL distribution. If distr='MCN', nu is the mixing proportion and 0<nu<1
`rho`	Degree of contamination, must be located between 0 and 1
`alpha`	Shape parameter, if distr='MVG', alpha is required and must be a positive value
`beta`	Scale parameter, if distr='MVG', beta is required and must be a positive value
`distr`	Distributional options are 'MVN' (multivariate normal), 'MVT' (multivariate t), 'MSL' (multivariate slash), 'MCN' (multivariate contaminated normal) and 'MVG' (multivariate variance-gamma).
`a.low`	Lower (left) truncation bound on the random vector, default is rep(-Inf, nrow(mu))
`a.upp`	Upper (right) truncation bound on the random vector, default is rep(Inf, nrow(mu))

Value

`EY`	The first moment
`EYY`	The second moment
`CovY`	The covariance matrix

Author(s)

Wan-Lun Wang <wangwl@gs.ncku.edu.tw> and Tsung-I Lin <tilin@nchu.edu.tw>

References

Tsung-I Lin and Wan-Lun Wang (2023). On moments of truncated multivariate normal/independent distributions. Submitted.

See Also

mni, tmni,

Examples

```
# Test Examples
mu=c(1,2,3)
Sigma=matrix(c(7,3,2,3,2,1,2,1,2), 3,3)
ubd = 1
a.low=c(-ubd,-ubd,-ubd); a.upp=c(ubd,ubd,ubd)

### MVN ###
y0 = rtmni(n=500, mu, Sigma, distr='MVN', lower=a.low, upper=a.upp)
colMeans(y0)
cov(y0)

MY0 = TMNI.moment(mu, Sigma, distr='MVN', a.low=a.low, a.upp=a.upp)
MY0$EY
MY0$CovY

### MVT ###
y1 = rtmni(n=500, mu, Sigma, nu=5, distr='MVT', lower=a.low, upper=a.upp)
colMeans(y1)
cov(y1)

MY1 = TMNI.moment(mu, Sigma, nu=5, distr='MVT', a.low=a.low, a.upp=a.upp)
MY1$EY
MY1$CovY

### MSL ###
y2 = rtmni(n=500, mu, Sigma, nu=2, distr='MSL', lower=a.low, upper=a.upp)
colMeans(y2)
cov(y2)

MY2 = TMNI.moment(mu, Sigma, nu=2, distr='MSL', a.low=a.low, a.upp=a.upp)
MY2$EY
MY2$CovY

### MCN ###
y3 = rtmni(n=500, mu, Sigma, nu=0.25, rho=0.2, distr='MCN', lower=a.low, upper=a.upp)
colMeans(y3)
cov(y3)

MY3 = TMNI.moment(mu, Sigma, nu=0.25, rho=0.2, distr='MCN', a.low=a.low, a.upp=a.upp)
MY3$EY
MY3$CovY

### MVG ###
y4 = rtmni(n=500, mu, Sigma, alpha=2, beta=1, distr='MVG', lower=a.low, upper=a.upp)
colMeans(y4)
cov(y4)
```

```
MY4 = TMNI.moment(mu, Sigma, alpha=2, beta=1, distr='MVG', a.low=a.low, a.upp=a.upp)
MY4$EY
MY4$CovY

### MDE ###
y5 = rtmni(n=500, mu, Sigma, alpha=1, beta=2, distr='MVG', lower=a.low, upper=a.upp)
colMeans(y5)
cov(y5)

MY5 = TMNI.moment(mu, Sigma, alpha=1, beta=2, distr='MVG', a.low=a.low, a.upp=a.upp)
MY5$EY
MY5$CovY
```

Index