Package 'VeccTMVN'

November 16, 2026

Type Package

Title Multivariate Normal Probabilities using Vecchia Approximation

Version 1.3.3

Date 2026-11-16

Author Jian Cao [aut, cre],
Matthias Katzfuss [aut]

Maintainer Jian Cao <jcao2416@gmail.com>

Description Under a different representation of the multivariate normal (MVN) probability, we can use the Vecchia approximation to sample the integrand at a linear complexity with respect to n. Additionally, both the SOV algorithm from Genz (92) and the exponential-tilting method from Botev (2017) can be adapted to linear complexity. The reference for the method implemented in this package is Jian Cao and Matthias Katzfuss (2024) ``Linear-Cost Vecchia Approximation of Multivariate Normal Probabilities'' <doi:10.48550/arXiv.2311.09426>. Two major references for the development of our method are Alan Genz (1992) ``Numerical Computation of Multivariate Normal Probabilities'' <doi:10.1080/10618600.1992.10477010> and Z. I. Botev (2017) ``The Normal Law Under Linear Restrictions: Simulation and Estimation via Minimax Tilting'' <doi:10.48550/arXiv.1603.04166>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.10), Matrix (>= 1.5-3), GpGp (>= 0.4.0), truncnorm (>= 1.0-8), GPvecchia, TruncatedNormal, nleqslv

Suggests testthat (>= 3.0.0), lhs, mvtnorm

Config/testthat/edition 3

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/JCatwood/VeccTMVN

BugReports https://github.com/JCatwood/VeccTMVN/issues

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-11-16 18:50:02 UTC

Contents

FIC_reorder_univar . . . 2
find_nn_corr . . . 3
get_sp_inv_chol . . . 4
loglk_censor_MVN . . . 5
mvrandn . . . 6
mvrandt . . . 7
pmvn . . . 8
pmvn_MLMC . . . 9
pmvt . . . 10
pmvt_MLMC . . . 11
ptmvrandn . . . 13
univar_order . . . 14
VeccTMVN . . . 14
Vecc_reorder . . . 15

Index **17**

`FIC_reorder_univar` *Univariate ordering under FIC approximation, first m chosen by m iter of dense univariate reordering*

Description

Univariate ordering under FIC approximation, first m chosen by m iter of dense univariate reordering

Usage

```
FIC_reorder_univar(
  a,
  b,
  m,
  locs = NULL,
  covName = NULL,
  covParms = NULL,
  covMat = NULL
)
```

Arguments

`a`	lower bound vector for TMVN
`b`	upper bound vector for TMVN
`m`	Vecchia conditioning set size
`locs`	location (feature) matrix n X d
`covName`	covariance function name from the ‘GpGp‘ package

covParms	parameters for ‘covName‘
covMat	dense covariance matrix, not needed when ‘locs‘ is not null

Value

a vector of new order based on FIC assumption and maxmin ordering

Examples

```
library(VeccTMVN)
n1 <- 5
n2 <- 5
n <- n1 * n2
m <- 5
locs <- as.matrix(expand.grid((1:n1) / n1, (1:n2) / n2))
covparms <- c(2, 0.1, 0)
cov_name <- "matern15_isotropic"
a <- rep(-Inf, n)
b <- seq(from = -3, to = 3, length.out = n)
cat("The output order should be roughly 1 to ", n)
cat(FIC_reorder_univar(a, b, m, locs, cov_name, covparms))
```

---

find_nn_corr

Find ordered nearest neighbors based on a correlation Matrix. Assuming the absolute value of the correlation is monotonically decreasing with distance. Returns an n X (m + 1) matrix similar to ‘GpGp::find_ordered_nn‘.

Description

Find ordered nearest neighbors based on a correlation Matrix. Assuming the absolute value of the correlation is monotonically decreasing with distance. Returns an n X (m + 1) matrix similar to ‘GpGp::find_ordered_nn‘.

Usage

```
find_nn_corr(corrMat, m)
```

Arguments

corrMat	the correlation matrix
m	the number of nearest neighbors

Value

an n X (m + 1) matrix

Examples

```
library(GpGp)
library(VeccTMVN)
set.seed(123)
d <- 3
n <- 100
locs <- matrix(runif(d * n), n, d)
covparms <- c(2, 0.01, 0)
cov_mat <- GpGp::matern15_isotropic(covparms, locs)
m <- 10
NNarray_test <- GpGp::find_ordered_nn(locs, m = m)
NNarray <- find_nn_corr(cov_mat, m)
cat("Number of mismatch is", sum(NNarray != NNarray_test, na.rm = TRUE))
```

---

get_sp_inv_chol *Get the inverse upper Cholesky factor under the Vecchia approximation*

Description

Get the inverse upper Cholesky factor under the Vecchia approximation

Usage

```
get_sp_inv_chol(covMat, NNarray)
```

Arguments

covMat	the covariance matrix
NNarray	n X (m + 1) matrix representing the nearest neighbor indices among previous observations. This is typically the return of GpGp::find_ordered_nn

Value

upper Cholesky of the inverse of ‘covMat‘

Examples

```
library(GpGp)
n1 <- 10
n2 <- 10
n <- n1 * n2
locs <- as.matrix(expand.grid((1:n1) / n1, (1:n2) / n2))
covparms <- c(2, 0.3, 0)
cov_mat <- GpGp::matern15_isotropic(covparms, locs)
m <- 30
NNarray <- GpGp::find_ordered_nn(locs, m = m)
# Vecchia approx ------------------------------
```

```
U_Vecc <- get_sp_inv_chol(cov_mat, NNarray)
U <- solve(chol(cov_mat))
cat("Frobenius norm of the difference is", sqrt(sum((U - U_Vecc)^2)))
```

`loglk_censor_MVN` *Compute censored multivariate normal (MVN) log-probabilities that have spatial covariance matrices using Vecchia approximation*

Description

Compute censored multivariate normal (MVN) log-probabilities that have spatial covariance matrices using Vecchia approximation

Usage

```
loglk_censor_MVN(
  locs,
  indCensor,
  y,
  bCensor,
  covName = NULL,
  covParms = NULL,
  m = 30,
  NLevel1 = 10,
  NLevel2 = 1000,
  verbose = TRUE
)
```

Arguments

`locs`	location (feature) matrix n X d
`indCensor`	indices of locations that have only censored observations
`y`	observed (not censored) values, of length n
`bCensor`	upper bound, above which observations are not censored, can be different for different locations, of length 1 or n
`covName`	covariance function name from the 'GpGp' package
`covParms`	parameters for 'covName'
`m`	Vecchia conditioning set size
`NLevel1`	first level Monte Carlo sample size
`NLevel2`	second level Monte Carlo sample size
`verbose`	verbose level

Value

estimated MVN probability and estimation error

`mvrandn` *Simulate truncated multivariate normal (TMVN) using the Vecchia approximation*

Description

Simulate truncated multivariate normal (TMVN) using the Vecchia approximation

Usage

```
mvrandn(
  lower,
  upper,
  mean,
  locs = NULL,
  covName = "matern15_isotropic",
  covParms = c(1, 0.1, 0),
  m = 30,
  sigma = NULL,
  N = 1000,
  verbose = FALSE
)
```

Arguments

`lower`	lower bound vector for TMVN
`upper`	upper bound vector for TMVN
`mean`	MVN mean
`locs`	location (feature) matrix n X d
`covName`	covariance function name from the ‘GpGp‘ package
`covParms`	parameters for ‘covName‘
`m`	Vecchia conditioning set size
`sigma`	dense covariance matrix, not needed when ‘locs‘ is not null
`N`	number of samples required
`verbose`	verbose level

Value

n X N matrix of generated samples

`mvrandt` *Simulate truncated multivariate normal (TMVT) using the Vecchia approximation*

Description

Simulate truncated multivariate normal (TMVT) using the Vecchia approximation

Usage

```
mvrandt(
  lower,
  upper,
  delta,
  df,
  locs = NULL,
  covName = "matern15_isotropic",
  covParms = c(1, 0.1, 0),
  m = 30,
  sigma = NULL,
  N = 1000,
  verbose = FALSE
)
```

Arguments

`lower`	lower bound vector for TMVT
`upper`	upper bound vector for TMVT
`delta`	MVT shifting parameter
`df`	degrees of freedom
`locs`	location (feature) matrix n X d
`covName`	covariance function name from the ‘GpGp‘ package
`covParms`	parameters for ‘covName‘
`m`	Vecchia conditioning set size
`sigma`	dense covariance matrix, not needed when ‘locs‘ is not null
`N`	number of samples required
`verbose`	verbose level

Value

n X N matrix of generated samples

pmvn *Compute multivariate normal (MVN) probabilities that have spatial covariance matrices using Vecchia approximation*

Description

Compute multivariate normal (MVN) probabilities that have spatial covariance matrices using Vecchia approximation

Usage

```
pmvn(
  lower,
  upper,
  mean,
  locs = NULL,
  covName = "matern15_isotropic",
  covParms = c(1, 0.1, 0),
  m = 30,
  sigma = NULL,
  reorder = 0,
  NLevel1 = 12,
  NLevel2 = 10000,
  verbose = FALSE,
  retlog = FALSE,
  ...
)
```

Arguments

lower	lower bound vector for TMVN
upper	upper bound vector for TMVN
mean	MVN mean
locs	location (feature) matrix n X d
covName	covariance function name from the 'GpGp' package
covParms	parameters for 'covName'
m	Vecchia conditioning set size
sigma	dense covariance matrix, not needed when 'locs' is not null
reorder	whether to reorder integration variables. '0' for no, '1' for FIC-based univariate ordering, '2' for Vecchia-based univariate ordering, and '3' for univariate reordering, which appeared faster than '2'
NLevel1	first level Monte Carlo sample size
NLevel2	second level Monte Carlo sample size
verbose	verbose or not

retlog	TRUE or FALSE for whether to return loglk or not
...	could be m_ord for conditioning set size for reordering

Value

estimated MVN probability and estimation error

---

pmvn_MLMC *Applying the multi-level Monte Carlo (MLMC) technique to the pmvn function The function uses* NLevel1 = 1 *for* m = m2 *and the same exponential tilting parameter as* m = m1 *to compute one MC estimate. This MC estimate is used to correct the bias from the Vecchia approximation*

Description

Applying the multi-level Monte Carlo (MLMC) technique to the pmvn function The function uses NLevel1 = 1 for m = m2 and the same exponential tilting parameter as m = m1 to compute one MC estimate. This MC estimate is used to correct the bias from the Vecchia approximation

Usage

```
pmvn_MLMC(
  lower,
  upper,
  mean,
  locs = NULL,
  covName = "matern15_isotropic",
  covParms = c(1, 0.1, 0),
  m1 = 30,
  m2 = 100,
  sigma = NULL,
  reorder = 0,
  NLevel1 = 12,
  NLevel2 = 10000,
  verbose = FALSE,
  retlog = FALSE,
  ...
)
```

Arguments

lower	lower bound vector for TMVN
upper	upper bound vector for TMVN
mean	MVN mean
locs	location (feature) matrix n X d

covName	covariance function name from the 'GpGp' package
covParms	parameters for 'covName'
m1	the smaller Vecchia conditioning set size for Level 1 MC
m2	the bigger Vecchia conditioning set size for Level 2 MC
sigma	dense covariance matrix, not needed when 'locs' is not null
reorder	whether to reorder integration variables. '0' for no, '1' for FIC-based univariate ordering, '2' for Vecchia-based univariate ordering, and '3' for univariate reordering, which appeared faster than '2'
NLevel1	first level Monte Carlo sample size
NLevel2	second level Monte Carlo sample size
verbose	verbose or not
retlog	TRUE or FALSE for whether to return loglk or not
...	could be m_ord for conditioning set size for reordering

Value

estimated MVN probability and estimation error

`pmvt` *Compute multivariate Student-t (MVT) probabilities that have spatial covariance matrices using Vecchia approximation*

Description

Compute multivariate Student-t (MVT) probabilities that have spatial covariance matrices using Vecchia approximation

Usage

```
pmvt(
  lower,
  upper,
  delta,
  df,
  locs = NULL,
  covName = "matern15_isotropic",
  covParms = c(1, 0.1, 0),
  m = 30,
  sigma = NULL,
  reorder = 0,
  NLevel1 = 12,
  NLevel2 = 10000,
  verbose = FALSE,
  retlog = FALSE,
  ...
)
```

Arguments

lower	lower bound vector for TMVT
upper	upper bound vector for TMVT
delta	MVT shifting parameter
df	degrees of freedom
locs	location (feature) matrix n X d
covName	covariance function name from the 'GpGp' package
covParms	parameters for 'covName'
m	Vecchia conditioning set size
sigma	dense covariance matrix, not needed when 'locs' is not null
reorder	whether to reorder integration variables. '0' for no, '1' for FIC-based univariate ordering, '2' for Vecchia-based univariate ordering, and '3' for univariate reordering, which appeared faster than '2'
NLevel1	first level Monte Carlo sample size
NLevel2	second level Monte Carlo sample size
verbose	verbose or not
retlog	TRUE or FALSE for whether to return loglk or not
...	could be m_ord for conditioning set size for reordering

Value

estimated MVT probability and estimation error

---

pmvt_MLMC — *Applying the multi-level Monte Carlo (MLMC) technique to the pmvt function The function uses* NLevel1 = 1 *for* m = m2 *and the same exponential tilting parameter as* m = m1 *to compute one MC estimate. This MC estimate is used to correct the bias from the Vecchia approximation*

Description

Applying the multi-level Monte Carlo (MLMC) technique to the pmvt function The function uses NLevel1 = 1 for m = m2 and the same exponential tilting parameter as m = m1 to compute one MC estimate. This MC estimate is used to correct the bias from the Vecchia approximation

Usage

```
pmvt_MLMC(
  lower,
  upper,
  delta,
  df,
  locs = NULL,
  covName = "matern15_isotropic",
  covParms = c(1, 0.1, 0),
  m1 = 30,
  m2 = 100,
  sigma = NULL,
  reorder = 0,
  NLevel1 = 12,
  NLevel2 = 10000,
  verbose = FALSE,
  retlog = FALSE,
  ...
)
```

Arguments

lower	lower bound vector for TMVT
upper	upper bound vector for TMVT
delta	MVT shifting parameter
df	degrees of freedom
locs	location (feature) matrix n X d
covName	covariance function name from the 'GpGp' package
covParms	parameters for 'covName'
m1	the smaller Vecchia conditioning set size for Level 1 MC
m2	the bigger Vecchia conditioning set size for Level 2 MC
sigma	dense covariance matrix, not needed when 'locs' is not null
reorder	whether to reorder integration variables. '0' for no, '1' for FIC-based univariate ordering, '2' for Vecchia-based univariate ordering, and '3' for univariate reordering, which appeared faster than '2'
NLevel1	first level Monte Carlo sample size
NLevel2	second level Monte Carlo sample size
verbose	verbose or not
retlog	TRUE or FALSE for whether to return loglk or not
...	could be m_ord for conditioning set size for reordering

Value

estimated MVT probability and estimation error

ptmvrandn *Simulate partially censored multivariate normal (MVN) at censored locations using the Vecchia approximation*

Description

Simulate partially censored multivariate normal (MVN) at censored locations using the Vecchia approximation

Usage

```
ptmvrandn(
  locs,
  indCensor,
  y,
  bCensor,
  covName = NULL,
  covParms = NULL,
  m = 30,
  N = 1000,
  verbose = TRUE,
  reorder = TRUE
)
```

Arguments

locs	location (feature) matrix n X d
indCensor	indices of locations that have only censored observations
y	observed (not censored) values, of length n
bCensor	upper bound, above which observations are not censored, can be different for different locations, of length 1 or n
covName	covariance function name from the 'GpGp' package
covParms	parameters for 'covName'
m	Vecchia conditioning set size
N	number of samples required
verbose	verbose level
reorder	whether to Vecchia univariate variable reordering

Value

n X N matrix of generated samples

`univar_order` *Univariate variable reordering, described in Genz and Bretz (2009) If failed due to PD singularity, the unfinished order will be returned and a warning will be issued*

Description

Univariate variable reordering, described in Genz and Bretz (2009) If failed due to PD singularity, the unfinished order will be returned and a warning will be issued

Usage

```
univar_order(a, b, sigma)
```

Arguments

`a`	lower integration limits
`b`	upper integration limits
`sigma`	covariance matrix

Value

the new order

`VeccTMVN` *VeccTMVN*

Description

Compute multivariate normal probabilities and sample from multivariate truncated normal distribution, taking advantage of the Vecchia approximation

Author(s)

jcao2416@gmail.com

Vecc_reorder *Univariate ordering under Vecchia approximation*

Description

Univariate ordering under Vecchia approximation

Usage

```
Vecc_reorder(
  a,
  b,
  m,
  locs = NULL,
  covName = NULL,
  covParms = NULL,
  covMat = NULL
)
```

Arguments

a	lower bound vector for TMVN
b	upper bound vector for TMVN
m	Vecchia conditioning set size
locs	location (feature) matrix n X d
covName	covariance function name from the 'GpGp' package
covParms	parameters for 'covName'
covMat	dense covariance matrix, not needed when 'locs' is not null

Value

new order, nearest neighbor matrix, and coefficient matrix

Examples

```
library(lhs)
library(GpGp)
library(VeccTMVN)
set.seed(123)
n <- 100
m <- 5
locs <- lhs::geneticLHS(n, 2)
covparms <- c(1, 0.1, 0)
cov_name <- "matern15_isotropic"
cov_mat <- get(cov_name)(covparms, locs)
a <- rep(-Inf, n)
```

```
b <- runif(n)
odr_TN <- TruncatedNormal::cholperm(cov_mat, a, b)$perm
rslt <- Vecc_reorder(a, b, m,
  locs = locs, covName = cov_name,
  covParms = covparms
)
# compare order
cat(rslt$order)
cat(odr_TN)
```

Index

FIC_reorder_univar, 2
find_nn_corr, 3

get_sp_inv_chol, 4

loglk_censor_MVN, 5

mvrandn, 6
mvrandt, 7

pmvn, 8
pmvn_MLMC, 9
pmvt, 10
pmvt_MLMC, 11
ptmvrandn, 13

univar_order, 14

Vecc_reorder, 15
VeccTMVN, 14