Package ‘anMC’

Type Package
Title Compute High Dimensional Orthant Probabilities
Version 0.2.5
Date 2023-08-22
Maintainer Dario Azzimonti <dario.azzimonti@gmail.com>
Description Computationally efficient method to estimate orthant probabilities of high-
dimensional Gaussian vectors. Further implements a function to compute conservative esti-
mates of excursion sets under Gaussian random field priors.
URL https://doi.org/10.1080/10618600.2017.1360781
License GPL-3
Encoding UTF-8
Depends mvtnorm
Imports Rcpp (>= 0.11.1)
Suggests DiceKriging, TruncatedNormal
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.2.3
NeedsCompilation yes
Author Dario Azzimonti [aut, cre, cph]
  (<https://orcid.org/0000-0001-5080-3061>)
Repository CRAN
Date/Publication 2023-08-22 10:30:02 UTC

R topics documented:

  ANMC_Gauss ................................................................. 2
  conservativeEstimate .................................................. 3
  get_chronotime .......................................................... 5
  MC_Gauss ................................................................. 6
  mvmnormArma ............................................................ 7
  ProbaMax ................................................................. 8
ANMC_Gauss

ProbaMin ................................................................. 12
selectActiveDims ....................................................... 16
selectQdims ............................................................. 17
trmvnorm_rej_cpp ...................................................... 19

Index 21

ANMC_Gauss  ANMC estimate for the remainder

Description

Asymmetric nested Monte Carlo estimation of \( P(\max X^{-q} > \text{threshold}|\max X^q \leq \text{threshold}) \)
where \( X \) is a normal vector. It is used for the bias correction in \text{ProbaMax} and \text{ProbaMin}.  

Usage

\[
\text{ANMC_Gauss}( \text{compBdg, problem, delta = 0.4, type = "M", trmvrnorm = trmvrnorm_rej_cpp, typeReturn = 0, verb = 0})
\]

Arguments

\begin{itemize}
\item \textbf{compBdg} total computational budget in seconds.
\item \textbf{problem} list defining the problem with mandatory fields
  \begin{itemize}
  \item \textbf{muEq} mean vector of \( X^q \);
  \item \textbf{sigmaEq} covariance matrix of \( X^q \);
  \item \textbf{threshold} fixed threshold \( t \);
  \item \textbf{muEmq} mean vector of \( X^{-q} \);
  \item \textbf{wwCondQ} = “weights” for \( X^{-q}|X^q \) [the vector \( \Sigma^{-q,q}(\Sigma^q)^{-1} \)];
  \item \textbf{sigmaCondQChol} = Cholesky factorization of the conditional covariance matrix \( \Sigma^{-q,q} \);
  \end{itemize}
\item \textbf{delta} total proportion of budget assigned to initial estimate (default 0.4), the actual proportion used might be smaller.
\item \textbf{type} type of excursion: "m", for minimum below threshold or "M", for maximum above threshold.
\item \textbf{trmvrnorm} function to generate truncated multivariate normal samples, it must have the following signature \( \text{trmvrnorm}(n,\mu,\sigma,\text{upper},\text{lower},\verb) \), where
  \begin{itemize}
  \item \( n \): number of simulations;
  \end{itemize}
\end{itemize}
conservativeEstimate

Computationally efficient conservative estimate

Description

Computes conservative estimates with two step GANMC procedure for a Gaussian vector. The probability is approximated with a biased low dimensional estimator and the bias is corrected with a MC estimator.

- mu: mean vector of the Normal variable of dimension \( d \);
- sigma: covariance matrix of dimension \( d \times d \);
- upper: vector of upper limits of length \( d \);
- lower: vector of lower limits of length \( d \);
- verb: the level of verbosity 3 basic, 4 extended.

It must return a matrix \( d \times n \) of realizations. If not specified, the rejection sampler trmvnorm_rej_cpp is used.

typeReturn integer chosen between

- 0 a number with only the probability estimation;
- 1 light return: a list with the probability estimator, the variance of the estimator, the vectors of conditional quantities used to obtain \( m^* \) and the system dependent parameters;
- 2 heavy return: the same list as light return with also the computational times and additional intermediate parameters.

verb level of verbosity (0,1 for this function), also sets the verbosity of trmvnorm (to verb-1).

Value

A list containing the estimated probability of excursion, see typeReturn for details.

References


Usage

```r
conservativeEstimate(
  alpha = 0.95,
  pred,
  design,
  threshold,
  pn = NULL,
  type = ">",
  verb = 1,
  lightReturn = T,
  algo = "GANMC"
)
```

Arguments

- **alpha**: probability of conservative estimate.
- **pred**: list containing mean vector (`pred$mean`) and covariance matrix (`pred$cov`).
- **design**: a matrix of size `length(pred$mean) x` (input space dimension) that contains the design where `pred$mean` was computed.
- **threshold**: threshold, real number.
- **pn**: coverage probability function, vector of the same length as `pred$mean` (if not specified it is computed).
- **type**: type of excursion: ">" for excursion above threshold or "<" for below.
- **verb**: level of verbosity, integer from 1–7.
- **lightReturn**: boolean for light return.
- **algo**: choice of algorithm for computing probabilities ("GANMC", "GMC").

Value

A list containing the conservative estimate (set), the Vorob'ev level (lvs). If `lightReturn=FALSE`, it also returns the actual probability of the set (proba) and the variance of this estimate (vars).

References


Examples

```r
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
  stop("DiceKriging needed for this example to work. Please install it.",
       call. = FALSE)
}
# Compute conservative estimate of excursion set of testfun below threshold
# Initialize
testfun<-function(x){return(((3*x^2+7*x-3)*exp(-1*(x)^2)*cos(5*pi*x^2)-1.2*x^2))}
mDet<- 1500

# Uniform design points
set.seed(42)
doe<-runif(n = 8)
res<-testfun(doe)
threshold<-0
# create km
smallKm <- DiceKriging::km(design = matrix(doe,ncol=1),
  response = res,covtype = "matern5_2",coef.trend = -1,coef.cov = c(0.05),coef.var = 1.1)
# prediction at newdata
newdata<-data.frame(matrix(seq(0,1,,mDet),ncol=1)); colnames(newdata)<-colnames(smallKm@X)
pred<-DiceKriging::predict.km(object = smallKm,newdata = newdata,type = "UK",cov.compute = TRUE)

## Not run:
# Plot (optional)
plot(seq(0,1,,mDet),pred$mean,type='l'
  points(doe,res,pch=10)
  abline(h = threshold)
  lines(seq(0,1,,mDet),pred$mean+pred$sd,lty=2,col=1)
  lines(seq(0,1,,mDet),pred$mean-pred$sd,lty=2,col=1)

## End(Not run)
# Compute the coverage function
pn<-pnorm((threshold-pred$mean)/pred$sd)

## Not run:
pred$cov <- pred$cov + 1e-7*diag(nrow = nrow(pred$cov),ncol = ncol(pred$cov))
CE<-conservativeEstimate(alpha = 0.95,pred = pred,design = as.matrix(newdata),
  threshold = threshold,type = "<",verb=1, pn=pn,algo = "ANMC")
points(newdata[CE$set,],rep(-0.1,mDet)[CE$set],col=4,pch="-",cex=2)
## End(Not run)
```

get_chronotime  Measure elapsed time with C++11 chrono library

Description

Returns a time indicator that can be used to accurately measure elapsed time. The C++11 clock used is `chrono::high_resolution_clock`. 
Usage

get_chronotime()

Value

A double with the number of nanoseconds elapsed since a fixed epoch.

Examples

# Measure 1 second sleep
initT<-get_chronotime()
Sys.sleep(1)
measT<-(get_chronotime()-initT)*1e-9
cat("1 second passed in ",measT," seconds.\n")

---

MC_Gauss

\textit{MC estimate for the remainder}

Description

Standard Monte Carlo estimate for \(P(\max X^{-q} > \text{threshold}|\max X^q \leq \text{threshold})\) or \(P(\min X^{-q} < \text{threshold}|\min X^q \geq \text{threshold})\) where \(X\) is a normal vector. It is used for the bias correction in \texttt{ProbaMax} and \texttt{ProbaMin}.

Usage

\begin{verbatim}
MC_Gauss(
    compBdg, 
    problem,
    delta = 0.1,
    type = "M",
    trmvrnorm = trmvrnorm_rej_cpp,
    typeReturn = 0,
    verb = 0,
    params = NULL
)
\end{verbatim}

Arguments

compBdg total computational budget in seconds.
problem list defining the problem with mandatory fields:
\begin{itemize}
  \item muEq = mean vector of \(X^q\);
  \item sigmaEq = covariance matrix of \(X^q\);
  \item threshold = threshold;
  \item muEmq = mean vector of \(X^{-q}\);
  \item wwCondQ = “weights” for \(X^{-q}|X^q\) [ the vector \(\Sigma^{-q,q}(\Sigma^q)^{-1}\)];
\end{itemize}
mvrnormArma

- sigmaCondQChol = Cholesky factorization of the conditional covariance matrix $\Sigma^{-q|q}$.

delta total proportion of budget assigned to initial estimate (default 0.1), the actual proportion used might be smaller.

type type of excursion: "m", for minimum below threshold or "M", for maximum above threshold.

trmvrnorm function to generate truncated multivariate normal samples, it must have the following signature trmvrnorm(n,mu,sigma,upper,lower,verb), where
  - n: number of simulations;
  - mu: mean vector of the Normal variable of dimension d;
  - sigma: covariance matrix of dimension dxd;
  - upper: vector of upper limits of length d;
  - lower: vector of lower limits of length d;
  - verb: the level of verbosity 3 basic, 4 extended.

It must return a matrix dxdn of realizations. If not specified, the rejection sampler trmvrnorm_rej_cpp is used.

typeReturn integer: 0 (only the estimate) or 1 (heavy return with variance of the estimate, parameters of the estimator and computational times).

verb the level of verbosity, also sets the verbosity of trmvrnorm (to verb-1).

params system dependent parameters (if NULL they are estimated).

Value

A list containing the estimated probability of excursion, see typeReturn for details.

References


Usage

\texttt{mvrnormArma(n, mu, sigma, chol)}

Arguments

- \texttt{n} \hspace{1cm} \text{number of simulations.}
- \texttt{mu} \hspace{1cm} \text{mean vector.}
- \texttt{sigma} \hspace{1cm} \text{covariance matrix or Cholesky decomposition of the matrix (see chol).}
- \texttt{chol} \hspace{1cm} \text{integer, if 0 sigma is a covariance matrix, otherwise it is the Cholesky decomposition of the matrix.}

Value

A matrix of size \(dxn\) containing the samples.

Examples

```r
# Simulate 1000 realizations from a multivariate normal vector
mu <- rep(0,200)
Sigma <- diag(rep(1,200))
realizations<-mvrnormArma(n=1000,mu = mu,sigma=Sigma, chol=0)
empMean<-rowMeans(realizations)
empCov<-cov(t(realizations))
# check if the sample mean is close to the actual mean
maxErrorOnMean<-max(abs(mu-empMean))
# check if we can estimate correctly the covariance matrix
maxErrorOnVar<-max(abs(rep(1,200)-diag(empCov)))
maxErrorOnCov<-max(abs(empCov[lower.tri(empCov)]))
## Not run:
plot(density(realizations[2,]))
## End(Not run)
```

---

ProbaMax

\textit{Probability of exceedance of maximum of Gaussian vector}

Description

Computes \(P(\max X > \text{threshold})\) with choice of algorithm between ANMC\textunderscore Gauss and MC\textunderscore Gauss. By default, the computationally expensive sampling parts are computed with the Rcpp functions.

Usage

\texttt{ProbaMax(}
\hspace{1cm} \texttt{cBdg},
\hspace{1cm} \texttt{threshold},
\hspace{1cm} \texttt{mu},
\hspace{1cm} \texttt{...})}
Sigma,
E = NULL,
q = NULL,
 pn = NULL,
 lightReturn = T,
 method = 4,
 verb = 0,
 Algo = "ANMC",
 trmvrnorm = trmvrnorm_rej_cpp,
 pmvnorm_usr = pmvnorm
)

Arguments

cBdg computational budget.
threshold threshold.
mu mean vector.
Sigma covariance matrix.
E discretization design for the field. If NULL, a simplex-lattice design n,n is used, with n=length(mu). In this case the choice of method=4,5 are not advised.
q number of active dimensions, it can be either
  • an integer: in this case the optimal q active dimension are chosen;
  • a numeric vector of length 2: this is the range where to search for the best
    number of active dimensions;
  • NULL: q is selected as the best number of active dimensions in the feasible
    range.

pn coverage probability function evaluated with mu, Sigma. If NULL it is computed automatically.
lightReturn boolean, if TRUE light return.
method method chosen to select the active dimensions. See selectActiveDims for de-
tails.
verb level of verbosity (0-5), selects verbosity also for ANMC_Gauss (verb-1) and
MC_Gauss (verb-1).
Algo choice of algorithm to compute the remainder Rq ("ANMC" or "MC").
trmvrnorm function to generate truncated multivariate normal samples, it must have the follow-
ing signature trmvrnorm(n,mu,sigma,upper,lower,verb), where
  • n: number of simulations;
  • mu: mean vector of the Normal variable of dimension d;
  • sigma: covariance matrix of dimension dxd;
  • upper: vector of upper limits of length d;
  • lower: vector of lower limits of length d;
  • verb: the level of verbosity 3 basic, 4 extended.

It must return a matrix dxn of realizations. If not specified, the rejection sampler
trmvrnorm_rej_cpp is used.
pmvnorm_usr function to compute core probability on active dimensions. Inputs:

• lower: the vector of lower limits of length d.
• upper: the vector of upper limits of length d.
• mean: the mean vector of length d.
• sigma: the covariance matrix of dimension d.

returns a the probability value with attribute "error", the absolute error. Default is the function pmvnorm from the package mvtnorm.

Value

A list containing

• probability: The probability estimate
• variance: the variance of the probability estimate
• q: the number of selected active dimensions

If lightReturn=F then the list also contains:

• aux_probabilities: a list with the probability estimates: probability the actual probability, pq the biased estimator $p_q$, $R_q$ the conditional probability $R_q$
• Eq: the points of the design $E$ selected for $p_q$
• indQ: the indices of the active dimensions chosen for $p_q$
• resRq: The list returned by the MC method used for $R_q$

References


Examples

```r
## Not run:
# Compute probability P(X \in (-\infty,0]) with X~N(\mu,Sigma)
d<-200 # example dimension
mu<-rep(0,d) # mean of the normal vector
# correlation structure (Miwa et al. 2003, Craig 2008, Botev 2016)
Sigma<-0.5*diag(d)+ 0.5*rep(1,d)%*%t(rep(1,d))
```
pANMC <- ProbaMax(cBdg = 20, q = min(50, d/2), E = seq(0, 1, d), threshold = 0, mu = mu, Sigma = Sigma, pn = NULL, lightReturn = TRUE, method = 3, verb = 2, Algo = "ANMC")
proba <- 1 - pANMC$probability

# Percentage error
abs(1 - pANMC$probability - 1/(d+1))/(1/(d+1))

# Implement ProbaMax with user defined function for active dimension probability estimate
if(!requireNamespace("TruncatedNormal", quietly = TRUE)) {
  stop("Package TruncatedNormal needed for this example to work. Please install it.\n\n    call. = FALSE")
}

# define pmvnorm_usr with the function mvNcdf from the package TruncatedNormal
pmvnorm_usr <- function(lower, upper, mean, sigma){
  pMET <- TruncatedNormal::mvNcdf(l = lower-mean, u = upper-mean, Sig = sigma, n = 5e4)
  res <- pMET$prob
  attr(res, "error") <- pMET$relErr
  return(res)
}

pANMC <- ProbaMax(cBdg = 20, q = min(50, d/2), E = seq(0, 1, d), threshold = 0, mu = mu, Sigma = Sigma, pn = NULL, lightReturn = TRUE, method = 3, verb = 2, Algo = "ANMC", pmvnorm_usr = pmvnorm_usr)
proba <- 1 - pANMC$probability

# Percentage error
abs(1 - pANMC$probability - 1/(d+1))/(1/(d+1))

# Implement ProbaMax with user defined function for truncated normal sampling
if(!requireNamespace("tmg", quietly = TRUE)) {
  stop("Package tmg needed for this example to work. Please install it.\n\n    call. = FALSE")
}

trmvrnorm_usr <- function(n, mu, sigma, upper, lower, verb){
  M <- chol2inv(chol(sigma))
  r <- as.vector(M%*%mu)
  if(all(lower == -Inf) && all(upper == Inf)){
    f <- NULL
    g <- NULL
  }else{
    if(all(lower == -Inf)){
      f <- -diag(length(mu))
      g <- upper
      initial <- (upper - 1)/2
    }else if(all(upper == Inf)){
      f <- diag(length(mu))
      g <- -lower
      initial <- -2*(lower + 1)
    }else{
      # Implement ProbaMax with user defined function for truncated normal sampling
    
    }# Implement ProbaMax with user defined function for truncated normal sampling
  }
\begin{verbatim}
    f<-rbind(-diag(length(mu)), diag(length(mu)))
g<-c(upper,-lower)
    initial<-c(upper-lower)/2
}
reals_tmg<-tmg::rtmg(n=n,M=M,r=r,initial = initial,f=f,g=g)
return(t(reals_tmg))
}
pANMC<-ProbaMax(cBdg=20, q=min(50,d/2), E=seq(0,1,,d), threshold=0, mu=mu, Sigma=Sigma, pn = NULL, lightReturn = TRUE, method = 3, verb = 2, Algo = "ANMC", trmvrnorm=trmvrnorm_usr)
proba<-1-pANMC$probability
# Percentage error
abs(1-pANMC$probability-1/(d+1))/(1/(d+1))
## End(Not run)
\end{verbatim}

**ProbaMin**  

**Probability of exceedance of minimum of Gaussian vector**

**Description**

Computes \( P(\min X \leq \text{threshold}) \) with choice of algorithm between ANMC_Gauss and MC_Gauss. By default, the computationally expensive sampling parts are computed with the Rcpp functions.

**Usage**

\begin{verbatim}
ProbaMin(
    cBdg,
    threshold,  
    mu,  
    Sigma,   
    E = NULL,  
    q = NULL, 
    pn = NULL, 
    lightReturn = T, 
    method = 4, 
    verb = 0,  
    Algo = "ANMC", 
    trmvrnorm = trmvrnorm_rej_cpp,  
    pmvnorm_usr = pmvnorm 
)
\end{verbatim}

**Arguments**

- `cBdg`: computational budget.
threshold threshold.
mu mean vector.
Sigma covariance matrix.
E discretization design for the field. If NULL, a simplex-lattice design n,n is used, with n=length(mu). In this case the choice of method=4,5 are not advised.
q number of active dimensions, it can be either
• an integer: in this case the optimal q active dimension are chosen;
• a numeric vector of length 2: this is the range where to search for the best number of active dimensions;
• NULL: q is selected as the best number of active dimensions in the feasible range.

pn coverage probability function evaluated with mu, Sigma. If NULL it is computed automatically.

lightReturn boolean, if TRUE light return.
method method chosen to select the active dimensions. See selectActiveDims for details.
verb level of verbosity (0-5), selects verbosity also for ANMC_Gauss (verb-1) and MC_Gauss (verb-1).
Algo choice of algorithm to compute the remainder Rq ("ANMC" or "MC").
trmvrnorm function to generate truncated multivariate normal samples, it must have the following signature trmvrnorm(n,mu,sigma,upper,lower,verb), where
• n: number of simulations;
• mu: mean vector of the Normal variable of dimension d;
• sigma: covariance matrix of dimension dxd;
• upper: vector of upper limits of length d;
• lower: vector of lower limits of length d;
• verb: the level of verbosity 3 basic, 4 extended.

It must return a matrix dxn of realizations. If not specified, the rejection sampler trmvrnorm_rej_cpp is used.

pmvnorm_usr function to compute core probability on active dimensions. Inputs:
• lower: the vector of lower limits of length d.
• upper : the vector of upper limits of length d.
• mean: the mean vector of length d.
• sigma: the covariance matrix of dimension d.

returns a the probability value with attribute "error", the absolute error. Default is the function pmvnorm from the package mvtnorm.

Value
A list containing
• probability: The probability estimate
• variance: the variance of the probability estimate
• q: the number of selected active dimensions

If lightReturn=F then the list also contains:
• aux_probabilities: a list with the probability estimates: probability the actual probability, pq the biased estimator \( p_q \), Rq the conditional probability \( R_q \)
• Eq: the points of the design \( E \) selected for \( p_q \)
• indQ: the indices of the active dimensions chosen for \( p_q \)
• resRq: The list returned by the MC method used for \( R_q \)

References


Examples

```r
## Not run:
# Compute probability P(X \in [0,\infty]) with X\sim N(0,\Sigma)
d<-200 # example dimension
mu<-rep(0,d) # mean of the normal vector
# correlation structure (Miwa et al. 2003, Craig 2008, Botev 2016)
Sigma<-0.5*diag(d)+ 0.5*rep(1,d)%*%t(rep(1,d))
pANMC<-ProbaMin(cBdg=20, q=min(50,d/2), E=seq(0,1,,d), threshold=0, mu=mu, Sigma=Sigma, pn = NULL, lightReturn = TRUE, method = 3, verb = 2, Algo = "ANMC")
proba<-1-pANMC$probability
# Percentage error
abs(1-pANMC$probability-1/(d+1))/(1/(d+1))

# Implement ProbaMin with user defined function for active dimension probability estimate
if(!requireNamespace("TruncatedNormal", quietly = TRUE)) {
  stop("TruncatedNormal needed for this example to work. Please install it.", call. = FALSE)
}
# define pmvnorm usr with the function mvNcdf from the package TruncatedNormal
pmvnorm_usr<-function(lower,upper,mean,sigma){
  pMET<-TruncatedNormal::mvNcdf(l = lower-mean,u = upper-mean,Sig = sigma,n = 5e4)
}```
res<-pMET$prob
attr(res,"error")<-pMET$relErr
return(res)
}
pANMC<-ProbaMin(cBdg=20, q=min(50,d/2), E=seq(0,1,,d), threshold=0, mu=mu, Sigma=Sigma,
pn = NULL, lightReturn = TRUE, method = 3, verb = 2, Algo = "ANMC",pmvnorm_usr=pmvnorm_usr)
proba<-1-pANMC$probability

# Percentage error
abs(1-pANMC$probability-1/(d+1))/(1/(d+1))

# Implement ProbaMin with user defined function for truncated normal sampling
if(!requireNamespace("tmg", quietly = TRUE)) {
  stop("Package tmg needed for this example to work. Please install it.",
call. = FALSE)
}
trmvrnorm_usr<-function(n,mu,sigma,upper,lower,verb){
  M<-chol2inv(chol(sigma))
  r=as.vector(M%*%mu)
  if(all(lower==-Inf) && all(upper==Inf)){
    f<- NULL
    g<- NULL
  }else{
    if(all(lower==-Inf)){
      f<-diag(length(mu))
      g<-upper
      initial<-(upper-1)/2
    }else if(all(upper==Inf)){
      f<-diag(length(mu))
      g<- -lower
      initial<-2*(lower+1)
    }else{
      f<-rbind(-diag(length(mu)),diag(length(mu)))
      g<-c(upper,-lower)
      initial<-(upper-lower)/2
    }
  }
  reals_tmg<-tmg::rtmg(n=n,M=M,r=r,initial = initial,f=f,g=g)
  return(t(reals_tmg))
}
pANMC<-ProbaMin(cBdg=20, q=min(50,d/2), E=seq(0,1,,d), threshold=0, mu=mu, Sigma=Sigma,
pn = NULL, lightReturn = TRUE, method = 3, verb = 2, Algo = "ANMC",trmvrnorm=trmvrnorm_usr)
proba<-1-pANMC$probability

# Percentage error
abs(1-pANMC$probability-1/(d+1))/(1/(d+1))

## End(Not run)
selectActiveDims  Select active dimensions for small dimensional estimate

Description
The function selectActiveDims selects the active dimensions for the computation of $p_q$ with an heuristic method.

Usage
selectActiveDims(
  q = NULL,  
  E, 
  threshold, 
  mu, 
  Sigma, 
  pn = NULL, 
  method = 1, 
  verb = 0, 
  pmvnorm_usr = pmvnorm
)

Arguments
- **q**: either the fixed number of active dimensions or the range where the number of active dimensions is chosen with selectQdims. If NULL the function selectQdims is called.
- **E**: discretization design for the field.
- **threshold**: threshold.
- **mu**: mean vector.
- **Sigma**: covariance matrix.
- **pn**: coverage probability function based on threshold, mu and Sigma. If NULL it is computed.
- **method**: integer chosen between
  - 0 selects by taking equally spaced indexes in mu;
  - 1 samples from pn;
  - 2 samples from pn*(1-pn);
  - 3 samples from pn adjusting for the distance (tries to explore all modes);
  - 4 samples from pn*(1-pn) adjusting for the distance (tries to explore all modes);
  - 5 samples with uniform probabilities.
- **verb**: level of verbosity: 0 returns nothing, 1 returns minimal info
- **pmvnorm_usr**: function to compute core probability on active dimensions. Inputs:
The function `selectQdims` iteratively selects the number of active dimensions and the dimensions themselves for the computation of $p_q$. The number of dimensions is increased until $p_q - p_{q-1}$ is smaller than the error of the procedure.

**Value**

A vector of integers denoting the chosen active dimensions of the vector $mu$.

**References**


Arguments

\( E \)

- discretization design for the field.

\( \text{threshold} \)

- threshold.

\( \mu \)

- mean vector.

\( \Sigma \)

- covariance matrix.

\( \text{pn} \)

- coverage probability function based on \( \text{threshold}, \mu \) and \( \Sigma \). If \text{NULL} it is computed.

\( \text{method} \)

- integer chosen between
  - 0 selects by taking equally spaced indexes in \( \mu \);
  - 1 samples from \( \text{pn} \);
  - 2 samples from \( \text{pn}^*(1-\text{pn}) \);
  - 3 samples from \( \text{pn} \) adjusting for the distance (tries to explore all modes);
  - 4 samples from \( \text{pn}^*(1-\text{pn}) \) adjusting for the distance (tries to explore all modes);
  - 5 samples with uniform probabilities.

\( \text{reducedReturn} \)

- boolean to select the type of return. See Value for further details.

\( \text{verb} \)

- level of verbosity: 0 returns nothing, 1 returns minimal info.

\( \text{limits} \)

- numeric vector of length 2 with \( q_{\text{min}} \) and \( q_{\text{max}} \). If \text{NULL} initialized at \( c(10,300) \)

\( \text{pmvnorm}_\text{usr} \)

- function to compute core probability on active dimensions. Inputs:
  - \text{lower}: the vector of lower limits of length \( d \).
  - \text{upper}: the vector of upper limits of length \( d \).
  - \text{mean}: the mean vector of length \( d \).
  - \text{sigma}: the covariance matrix of dimension \( d \).

returns a the probability value with attribute "error", the absolute error. Default is the function \text{pmvnorm} from the package \text{mvtnorm}.

Value

If \( \text{reducedReturn} = \text{F} \) returns a list containing

- \( \text{indQ} \): the indices of the active dimensions chosen for \( p_q \);
- \( p_q \): the biased estimator \( p_q \) with attribute \text{error}, the estimated absolute error;
- \( E_q \): the points of the design \( E \) selected for \( p_q \);
- \( \mu_Eq \): the subvector of \( \mu \) selected for \( p_q \);
- \( \Sigma_Eq \): the submatrix of \( \Sigma \) composed by the indexes selected for \( p_q \).

Otherwise it returns only \( \text{indQ} \).
### trmvrnorm_rej_cpp

Sample from truncated multivariate normal distribution with C++

### Description

Simulates realizations from a truncated multivariate normal with mean mu, covariance matrix sigma in the bounds lower upper.

### Usage

```
trmvrnorm_rej_cpp(n, mu, sigma, lower, upper, verb)
```

### Arguments

- `n`: number of simulations.
- `mu`: mean vector.
- `sigma`: covariance matrix.
- `lower`: vector of lower bounds.
- `upper`: vector of upper bounds.
- `verb`: level of verbosity: if lower than 3 nothing, 3 minimal, 4 extended.

### Value

A matrix of size $dxn$ containing the samples.

### References


---

### References


Examples

# Simulate 1000 realizations from a truncated multivariate normal vector
mu <- rep(0,10)
Sigma <- diag(rep(1,10))
upper <- rep(3,10)
lower <- rep(-0.5,10)
realizations<-trmvrnorm_rej_cpp(n=1000,mu = mu, sigma=Sigma, lower =lower, upper= upper,verb=3)
empMean<-rowMeans(realizations)
empCov<-cov(t(realizations))
# check if the sample mean is close to the actual mean
maxErrorOnMean<-max(abs(mu-empMean))
# check if we can estimate correctly the covariance matrix
maxErrorOnVar<-max(abs(rep(1,200)-diag(empCov)))
maxErrorOnCov<-max(abs(empCov[lower.tri(empCov)]))
## Not run:
plot(density(realizations[1,]))
hist(realizations[1,],breaks="FD")

## End(Not run)
Index

ANMC_Gauss, 2, 9, 13
conservativeEstimate, 3
get_chronotime, 5
MC_Gauss, 6, 9, 13
mvrnormArma, 7
pmvnorm, 10, 13, 17, 18
ProbaMax, 2, 6, 8
ProbaMin, 2, 6, 12
selectActiveDims, 9, 13, 16
selectQdims, 16, 17
trmvrnorm_rej_cpp, 3, 7, 9, 13, 19