Package ‘VecDep’

January 20, 2025

Type Package

Title Measuring Copula-Based Dependence Between Random Vectors

Version 0.1.3

Description Provides functions for estimation (parametric, semi-parametric and non-parametric) of copula-based dependence coefficients between a finite collection of random vectors, including phi-dependence measures and Bures-Wasserstein dependence measures. An algorithm for agglomerative hierarchical variable clustering is also implemented. Following the articles De Keyser & Gijbels (2024) <doi:10.1016/j.jmva.2024.105336>, De Keyser & Gijbels (2024) <doi:10.1016/j.ijar.2023.109090>, and De Keyser & Gijbels (2024) <doi:10.48550/arXiv.2404.07141>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/StevenDeKeyser98/VecDep

BugReports https://github.com/StevenDeKeyser98/VecDep/issues

Depends R (>= 4.4.0)

Imports ElliptCopulas (>= 0.1.4.1), HAC (>= 1.1-1), hash (>= 2.2.6.3), sets (>= 1.0-25), covglasso (>= 1.0.3), expm (>= 1.0-0), magic (>= 1.6-1), pbapply (>= 1.7-2), Rmpfr (>= 0.9-5), reticulate (>= 1.39.0), gtools (>= 3.9.5)

Suggests mvtnorm, ggplot2, extraDistr, fossil, dendextend, copula, knitr, rmarkdown, testthat (>= 3.0.0)

Language en-US

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Steven De Keyser [aut, cre] (<https://orcid.org/0000-0002-3469-8692>), Irène Gijbels [ctb] (<https://orcid.org/0000-0002-4443-9803>)

Maintainer Steven De Keyser <steven.dekeyser@kuleuven.be>

Repository CRAN

Date/Publication 2024-11-14 13:50:19 UTC

Contents

`betakernelestimator` *betakernelestimator*

Description

This function computes the non-parametric beta kernel copula density estimator.

Usage

```
betakernelestimator(input, h, pseudos)
```

Arguments

`input`	The copula argument at which the density estimate is to be computed.
`h`	The bandwidth to be used in the beta kernel.
`pseudos`	The (estimated) copula observations from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).

Details

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_k)$ with $\mathbf{X}_i = (X_{i1}, \ldots, X_{id_i})$, and samples $X_{ij}^{(1)}, \ldots, X_{ij}^{(n)}$ from X_{ij} for $i = 1, \ldots, k$ and $j = 1, \ldots, d_i$, the beta kernel estimator for the copula density of $\mathbf{X}$ equals, at $\mathbf{u} = (u_{11}, \ldots, u_{kd_k}) \in \mathbb{R}^q$,

$$\widehat{c}_{\mathrm{B}}(\mathbf{u}) = \frac{1}{n} \sum_{\ell=1}^{n} \prod_{i=1}^{k} \prod_{j=1}^{d_i} k_{\mathrm{B}} \left(\widehat{U}_{ij}^{(\ell)}, \frac{u_{ij}}{h_n} + 1, \frac{1 - u_{ij}}{h_n} + 1 \right),$$

where $h_n > 0$ is a bandwidth parameter, $\widehat{U}_{ij}^{(\ell)} = \widehat{F}_{ij}(X_{ij}^{(\ell)})$ with

$$\widehat{F}_{ij}(x_{ij}) = \frac{1}{n+1} \sum_{\ell=1}^{n} 1 \left(X_{ij}^{(\ell)} \leq x_{ij} \right)$$

the (rescaled) empirical cdf of X_{ij}, and

$$k_{\mathrm{B}}(u, \alpha, \beta) = \frac{u^{\alpha-1}(1-u)^{\beta-1}}{B(\alpha, \beta)},$$

with B the beta function.

Value

The beta kernel copula density estimator evaluated at the input.

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

See Also

`transformationestimator` for the computation of the Gaussian transformation kernel copula density estimator, `hamse` for local bandwidth selection for the beta kernel or Gaussian transformation kernel copula density estimator, `phinp` for fully non-parametric estimation of the Φ-dependence between k random vectors.

Examples

```
q = 3
n = 100

# Sample from multivariate normal distribution with identity covariance matrix
sample = mvtnorm::rmvnorm(n,rep(0,q),diag(3),method = "chol")

# Copula pseudo-observations
pseudos = matrix(0,n,q)
for(j in 1:q){pseudos[,j] = (n/(n+1)) * ecdf(sample[,j])(sample[,j])}

# Argument at which to estimate the density
input = rep(0.5,q)

# Local bandwidth selection
h = hamse(input,pseudos = pseudos,n = n,estimator = "beta",bw_method = 1)

# Beta kernel estimator
est_dens = betakernelestimator(input,h,pseudos)

# True density
true = copula::dCopula(input, copula::normalCopula(0, dim = q))
```

---

bwd1 *bwd1*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the correlation-based Bures-Wasserstein coefficient $\mathcal{D}_1$ between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$.

Usage

```
bwd1(R, dim)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$.

Details

Given a correlation matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1k} \\ \mathbf{R}_{12}^{\mathrm{T}} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1k}^{\mathrm{T}} & \mathbf{R}_{2k}^{\mathrm{T}} & \cdots & \mathbf{R}_{kk} \end{pmatrix},$$

the coefficient $\mathcal{D}_1$ equals

$$\mathcal{D}_1(\mathbf{R}) = \frac{d_W^2(\mathbf{R}, \mathbf{I}_q) - \sum_{i=1}^{k} d_W^2(\mathbf{R}_{ii}, \mathbf{I}_{d_i})}{\sup_{\mathbf{A} \in \Gamma(\mathbf{R}_{11}, \dots, \mathbf{R}_{kk})} d_W^2(\mathbf{A}, \mathbf{I}_q) - \sum_{i=1}^{k} d_W^2(\mathbf{R}_{ii}, \mathbf{I}_{d_i})},$$

where d_W stands for the Bures-Wasserstein distance, $\Gamma(\mathbf{R}_{11}, \dots, \mathbf{R}_{kk})$ denotes the set of all correlation matrices with diagonal blocks $\mathbf{R}_{ii}$ for $i = 1, \dots, k$, and $\mathbf{I}_q$ is the identity matrix. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$ between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

See Also

bwd2 for the computation of the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$, bwd1avar for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_1$.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

bwd1(R,dim)
```

---

bwd1asR0 *bwd1asR0*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function simulates a sample from the asymptotic distribution of the plug-in estimator for the correlation-based Bures-Wasserstein coefficient $\mathcal{D}_1$ between $\mathbf{X}_1, ..., \mathbf{X}_k$ given that the entire correlation matrix $\mathbf{R}$ is equal to $\mathbf{R}_0$ (correlation matrix under independence of $\mathbf{X}_1, ..., \mathbf{X}_k$). The argument dim should be in ascending order. This function requires importation of the python modules "numpy" and "scipy".

Usage

```
bwd1asR0(R, dim, M)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$, in ascending order.
M	The sample size.

Details

A sample of size M is drawn from the asymptotic distribution of the plug-in estimator $\mathcal{D}_1(\widehat{\mathbf{R}}_n)$ at $\mathbf{R}_0 = \text{diag}(\mathbf{R}_{11}, \ldots, \mathbf{R}_{kk})$, where $\widehat{\mathbf{R}}_n$ is the sample matrix of normal scores rank correlations. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

To create a Python virtual environment with "numpy" and "scipy", run:

install_tensorflow()

reticulate::use_virtualenv("r-tensorflow", required = FALSE)

reticulate::py_install("numpy")

reticulate::py_install("scipy")

Value

A sample of size M from the asymptotic distribution of the plug-in estimator for the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$ under independence of $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

See Also

bwd1 for the computation of the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$, bwd2 for the computation of the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$, bwd1avar for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_1$, bwd2avar for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_2$, bwd2asR0 for sampling from the asymptotic distribution of the plug-in estimator for $\mathcal{D}_2$ under the hypothesis of independence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$, estR for the computation of the sample matrix of normal scores rank correlations, otsort for rearranging the columns of sample such that dim is in ascending order.

Examples

```
q = 5
dim = c(2,3)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

R0 = createR0(R,dim)

# Check whether scipy module is available (see details)
```

```
have_scipy = reticulate::py_module_available("scipy")

if(have_scipy){

sample = bwd1asR0(R0,dim,1000)

}
```

---

bwd1avar *bwd1avar*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the asymptotic variance of the plug-in estimator for the correlation-based Bures-Wasserstein coefficient $\mathcal{D}_1$ between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$. The argument dim should be in ascending order.

Usage

```
bwd1avar(R, dim)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$, in ascending order.

Details

The asymptotic variance of the plug-in estimator $\mathcal{D}_1(\widehat{\mathbf{R}}_n)$ is computed at $\mathbf{R}$, where $\widehat{\mathbf{R}}_n$ is the sample matrix of normal scores rank correlations. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The asymptotic variance of the plug-in estimator for the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$ between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

See Also

bwd1 for the computation of the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$, bwd2 for the computation of the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$, bwd2avar for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_2$, bwd1asR0 for sampling from the asymptotic distribution of the plug-in estimator for $\mathcal{D}_1$ under the hypothesis of independence between $\mathbf{X}_1, \dots, \mathbf{X}_k$, bwd2asR0 for sampling from the asymptotic distribution of the plug-in estimator for $\mathcal{D}_2$ under the hypothesis of independence between $\mathbf{X}_1, \dots, \mathbf{X}_k$, estR for the computation of the sample matrix of normal scores rank correlations, otsort for rearranging the columns of sample such that dim is in ascending order.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

bwd1avar(R,dim)
```

---

bwd2 *bwd2*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the correlation-based Bures-Wasserstein coefficient $\mathcal{D}_2$ between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$.

Usage

```
bwd2(R, dim)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$.

Details

Given a correlation matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1k} \\ \mathbf{R}_{12}^{\text{T}} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1k}^{\text{T}} & \mathbf{R}_{2k}^{\text{T}} & \cdots & \mathbf{R}_{kk} \end{pmatrix},$$

the coefficient $\mathcal{D}_2$ equals

$$\mathcal{D}_2(\mathbf{R}) = \frac{d_W^2(\mathbf{R}, \mathbf{R}_0)}{\sup_{\mathbf{A} \in \Gamma(\mathbf{R}_{11}, \ldots, \mathbf{R}_{kk})} d_W^2(\mathbf{A}, \mathbf{R}_0)},$$

where d_W stands for the Bures-Wasserstein distance, $\Gamma(\mathbf{R}_{11}, \ldots, \mathbf{R}_{kk})$ denotes the set of all correlation matrices with diagonal blocks $\mathbf{R}_{ii}$ for $i = 1, \ldots, k$, and the matrix $\mathbf{R}_0 = \mathrm{diag}(\mathbf{R}_{11}, \ldots, \mathbf{R}_{kk})$ is the correlation matrix under independence. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$ between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

See Also

`bwd1` for the computation of the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$, `bwd2avar` for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_2$.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

bwd2(R,dim)
```

`bwd2asR0` *bwd2asR0*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function simulates a sample from the asymptotic distribution of the plug-in estimator for the correlation-based Bures-Wasserstein coefficient $\mathcal{D}_2$ between $\mathbf{X}_1, ..., \mathbf{X}_k$ given that the entire correlation matrix $\mathbf{R}$ is equal to $\mathbf{R}_0$ (correlation matrix under independence of $\mathbf{X}_1, ..., \mathbf{X}_k$). The argument dim should be in ascending order. This function requires importation of the python modules "numpy" and "scipy".

Usage

```
bwd2asR0(R, dim, M)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$, in ascending order.
M	The sample size.

Details

A sample of size M is drawn from the asymptotic distribution of the plug-in estimator $\mathcal{D}_2(\widehat{\mathbf{R}}_n)$ at $\mathbf{R}_0 = \text{diag}(\mathbf{R}_{11}, \ldots, \mathbf{R}_{kk})$, where $\widehat{\mathbf{R}}_n$ is the sample matrix of normal scores rank correlations. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

To create a Python virtual environment with "numpy" and "scipy", run:

install_tensorflow()

reticulate::use_virtualenv("r-tensorflow", required = FALSE)

reticulate::py_install("numpy")

reticulate::py_install("scipy")

Value

A sample of size M from the asymptotic distribution of the plug-in estimator for the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$ under independence of $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

See Also

bwd1 for the computation of the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$, bwd2 for the computation of the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$, bwd1avar for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_1$, bwd2avar for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_2$, bwd1asR0 for sampling from the asymptotic distribution of the plug-in estimator for $\mathcal{D}_1$ under the hypothesis of independence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$, estR for the computation of the sample matrix of normal scores rank correlations, otsort for rearranging the columns of sample such that dim is in ascending order.

Examples

```
q = 5
dim = c(2,3)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

R0 = createR0(R,dim)

# Check whether scipy module is available (see details)
```

```
have_scipy = reticulate::py_module_available("scipy")

if(have_scipy){

sample = bwd2asR0(R0,dim,1000)

}
```

---

bwd2avar *bwd2avar*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the asymptotic variance of the plug-in estimator for the correlation-based Bures-Wasserstein coefficient $\mathcal{D}_2$ between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$. The argument dim should be in ascending order.

Usage

```
bwd2avar(R, dim)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$, in ascending order.

Details

The asymptotic variance of the plug-in estimator $\mathcal{D}_2(\widehat{\mathbf{R}}_n)$ is computed at $\mathbf{R}$, where $\widehat{\mathbf{R}}_n$ is the sample matrix of normal scores rank correlations. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The asymptotic variance of the plug-in estimator for the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$ between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

See Also

bwd1 for the computation of the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$, bwd2 for the computation of the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$, bwd1avar for the computation of the asymptotic variance of the plug-in estimator for $\mathcal{D}_1$, bwd1asR0 for sampling from the asymptotic distribution of the plug-in estimator for $\mathcal{D}_1$ under the hypothesis of independence between $\mathbf{X}_1, \dots, \mathbf{X}_k$, bwd2asR0 for sampling from the asymptotic distribution of the plug-in estimator for $\mathcal{D}_2$ under the hypothesis of independence between $\mathbf{X}_1, \dots, \mathbf{X}_k$, estR for the computation of the sample matrix of normal scores rank correlations, otsort for rearranging the columns of sample such that dim is in ascending order.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

bwd2avar(R,dim)
```

---

covgpenal *covgpenal*

Description

This function computes the empirical penalized Gaussian copula covariance matrix with the Gaussian log-likelihood plus a lasso-type penalty as objective function. Model selection is done by choosing omega such that BIC is maximal.

Usage

```
covgpenal(
  S,
  n,
  omegas,
  derpenal = function(t, omega) {
     derSCAD(t, omega, 3.7)
 },
  nsteps = 1
)
```

Arguments

S	The sample matrix of normal scores covariances.
n	The sample size.
omegas	The candidate values for the tuning parameter in $[0, \infty)$.

derpenal The derivative of the penalty function to be used (default = scad with parameter $a = 3.7$).

nsteps The number of weighted covariance graphical lasso iterations (default = 1).

Details

The aim is to solve/compute

$$\widehat{\mathbf{\Sigma}}_{\text{LT},n} \in \text{arg min}_{\mathbf{\Sigma}>0} \left\{ \ln|\mathbf{\Sigma}| + \text{tr}\left(\mathbf{\Sigma}^{-1}\widehat{\mathbf{\Sigma}}_n\right) + P_{\text{LT}}\left(\mathbf{\Sigma}, \omega_n\right) \right\},$$

where the penalty function P_{LT} is of lasso-type:

$$P_{\text{LT}}\left(\mathbf{\Sigma}, \omega_n\right) = \sum_{ij} p_{\omega_n}\left(\Delta_{ij}\left|\sigma_{ij}\right|\right),$$

for a certain penalty function p_{ω_n} with penalty parameter ω_n, and σ_{ij} the (i,j)'th entry of $\mathbf{\Sigma}$ with $\Delta_{ij} = 1$ if $i \neq j$ and $\Delta_{ij} = 0$ if $i = j$ (in order to not shrink the variances). The matrix $\widehat{\mathbf{\Sigma}}_n$ is the matrix of sample normal scores covariances.

In case $p_{\omega_n}(t) = \omega_n t$ is the lasso penalty, the implementation for the (weighted) covariance graphical lasso is available in the R package 'covglasso' (see the manual for further explanations). For general penalty functions, we perform a local linear approximation to the penalty function and iteratively do (nsteps, default = 1) weighted covariance graphical lasso optimizations.

The default for the penalty function is the scad (derpenal = derivative of scad penalty), i.e.,

$$p'_{\omega_n,\text{scad}}(t) = \omega_n\left[1\left(t \leq \omega_n\right) + \frac{\max\left(a\omega_n - t, 0\right)}{\omega_n(a-1)} 1\left(t > \omega_n\right)\right],$$

with $a = 3.7$ by default.

For tuning ω_n, we maximize (over a grid of candidate values) the BIC criterion

$$\text{BIC}\left(\widehat{\mathbf{\Sigma}}_{\omega_n}\right) = -n\left[\ln\left|\widehat{\mathbf{\Sigma}}_{\omega_n}\right| + \text{tr}\left(\widehat{\mathbf{\Sigma}}_{\omega_n}^{-1}\widehat{\mathbf{\Sigma}}_n\right)\right] - \ln(n)\text{df}\left(\widehat{\mathbf{\Sigma}}_{\omega_n}\right),$$

where $\widehat{\mathbf{\Sigma}}_{\omega_n}$ is the estimated candidate covariance matrix using ω_n and df (degrees of freedom) equals the number of non-zero entries in $\widehat{\mathbf{\Sigma}}_{\omega_n}$, not taking the elements under the diagonal into account.

Value

A list with elements "est" containing the (lasso-type) penalized matrix of sample normal scores rank correlations (output as provided by the function "covglasso.R"), and "omega" containing the optimal tuning parameter.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

Fop, M. (2021). covglasso: sparse covariance matrix estimation, R package version 1.0.3. url: https://CRAN.R-project.org/package=covglasso.

Wang, H. (2014). Coordinate descent algorithm for covariance graphical lasso. Statistics and Computing 24:521-529. doi: https://doi.org/10.1007/s11222-013-9385-5.

See Also

grouplasso for group lasso estimation of the normal scores rank correlation matrix.

Examples

```
q = 10
dim = c(5,5)
n = 100

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

# Sparsity on off-diagonal blocks
R0 = createR0(R,dim)

# Sample from multivariate normal distribution
sample = mvtnorm::rmvnorm(n,rep(0,q),R0,method = "chol")

# Normal scores
scores = matrix(0,n,q)
for(j in 1:q){scores[,j] = qnorm((n/(n+1)) * ecdf(sample[,j])(sample[,j]))}

# Sample matrix of normal scores covariances
Sigma_est = cov(scores) * ((n-1)/n)

# Candidate tuning parameters
omega = seq(0.01, 0.6, length = 50)

Sigma_est_penal = covgpenal(Sigma_est,n,omega)
```

---

createR0 *createR0*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function constructs the correlation matrix under independence of $\mathbf{X}_1, ..., \mathbf{X}_k$, given the entire correlation matrix $\mathbf{R}$.

Usage

```
createR0(R, dim)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$.

Details

Given a correlation matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1k} \\ \mathbf{R}_{12}^{\mathrm{T}} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1k}^{\mathrm{T}} & \mathbf{R}_{2k}^{\mathrm{T}} & \cdots & \mathbf{R}_{kk} \end{pmatrix},$$

the matrix $\mathbf{R}_0 = \mathrm{diag}(\mathbf{R}_{11}, \ldots, \mathbf{R}_{kk})$, being the correlation matrix under independence of $\mathbf{X}_1, \ldots, \mathbf{X}_k$, is returned.

Value

The correlation matrix under independence of $\mathbf{X}_1, \ldots, \mathbf{X}_n$.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

createR0(R,dim)
```

---

cvomega *cvomega*

Description

This functions selects the omega tuning parameter for ridge penalization of the empirical Gaussian copula correlation matrix via cross-validation. The objective function is the Gaussian log-likelihood, and a grid search is performed using K folds.

Usage

```
cvomega(sample, omegas, K)
```

Arguments

`sample`	A sample from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
`omegas`	A grid of candidate penalty parameters in $[0, 1]$.
`K`	The number of folds to be used.

Details

The loss function is the Gaussian log-likelihood, i.e., given an estimated (penalized) Gaussian copula correlation matrix (normal scores rank correlation matrix) $\widehat{\mathbf{R}}_n^{(-j)}$ computed on a training set leaving out fold j, and $\widehat{\mathbf{R}}_n^{(j)}$ the empirical (non-penalized) Gaussian copula correlation matrix computed on test fold j, we search for the tuning parameter that minimizes

$$\sum_{j=1}^{K} \left[\ln \left(\left| \widehat{\mathbf{R}}_n^{(-j)} \right| \right) + \text{tr} \left\{ \widehat{\mathbf{R}}_n^{(j)} \left(\widehat{\mathbf{R}}_n^{(-j)} \right)^{-1} \right\} \right].$$

The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The optimal ridge penalty parameter minimizing the cross-validation error.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

Warton, D.I. (2008). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. Journal of the American Statistical Association 103(481):340-349.
doi: https://doi.org/10.1198/016214508000000021.

See Also

estR for computing the (Ridge penalized) empirical Gaussian copula correlation matrix.

Examples

```
q = 10
n = 50

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

# Sample from multivariate normal distribution
sample = mvtnorm::rmvnorm(n,rep(0,q),R,method = "chol")

# 5-fold cross-validation with Gaussian likelihood as loss for selecting omega
omega = cvomega(sample = sample,omegas = seq(0.01,0.999,len = 50),K = 5)

R_est = estR(sample,omega = omega)
```

ellcopest *ellcopest*

Description

This functions performs improved kernel density estimation of the generator of a meta-elliptical copula by using Liebscher's algorithm, combined with a shrinkage function.

Usage

```
ellcopest(
  dataU,
  Sigma_m1,
  h,
  grid,
  niter = 10,
  a,
  Kernel = "epanechnikov",
  shrink,
  verbose = 1,
  startPoint = "identity",
  prenormalization = FALSE,
  normalize = 1
)
```

Arguments

dataU	The (estimated) copula observations from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
Sigma_m1	The (estimated) inverse of the scale matrix of the meta-elliptical copula.
h	The bandwidth of the kernel.
grid	The grid of values on which to estimate the density generator.
niter	The number of iterations used in the MECIP (default = 10).
a	The tuning parameter to improve the performance at 0.
Kernel	The kernel used for the smoothing (default = "epanechnikov").
shrink	The shrinkage function to further improve the performance at 0 and guarantee the existence of the AMISE bandwidth.
verbose	See the "EllDistrEst.R" function of the R package 'ElliptCopulas'.
startPoint	See the "EllDistrEst.R" function of the R package 'ElliptCopulas'.
prenormalization	See the "EllDistrEst.R" function of the R package 'ElliptCopulas'.
normalize	A value in $\{1, 2\}$ indicating the normalization procedure that is applied to the estimated generator (default = 1).

Details

The context is the one of a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_k)$,

with $\mathbf{X}_i = (X_{i1}, \ldots, X_{id_i})$ for $i = 1, \ldots, k$, having a meta-elliptical copula. This means that there exists a generator $g_{\mathcal{R}} : (0, \infty) \rightarrow \mathbb{R}$ and a quantile function Q, such that the random vector $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_k)$ with

$$\mathbf{Z}_i = (Z_{i1}, \ldots, Z_{id_i}) = ((Q \circ F_{i1})(X_{i1}), \ldots, (Q \circ F_{id_i})(X_{id_i}))$$

for $i = 1, \ldots, k$, where F_{ij} is the cdf of X_{ij}, has a multivariate elliptical distribution. Denoting $\widehat{F}_{ij}(x_{ij}) = \frac{1}{n+1}\sum_{\ell=1}^{n} 1\left(X_{ij}^{(\ell)} \leq x_{ij}\right)$ for the (rescaled) empirical cdf of X_{ij} based on a sample $X_{ij}^{(1)}, \ldots, X_{ij}^{(n)}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, d_i$, and $\widehat{\mathbf{R}}$ for an estimator of the scale matrix $\mathbf{R}$, this function estimates $g_{\mathcal{R}}$ by using the MECIP (Meta-Elliptical Copula Iterative Procedure) of Derumigny & Fermanian (2022).

This means that we start from an initial guess $\widehat{g}_{\mathcal{R},0}$ for the generator $g_{\mathcal{R}}$, based on which we obtain an estimated sample from $\mathbf{Z}$ through the quantile function corresponding to $\widehat{g}_{\mathcal{R},0}$. Based on this estimated sample, we then obtain an estimator $\widehat{g}_{\mathcal{R},1}$ using the function `elldistrest`, performing improved kernel estimation with shrinkage function. This procedure is repeated for a certain amount (niter) of iterations to obtain a final estimate for $g_{\mathcal{R}}$.

The estimator without the shrinkage function α is implemented in the R package 'ElliptCopulas'. We use this implementation and bring in the shrinkage function.

In order to make $g_{\mathcal{R}}$ identifiable, an extra normalization procedure is implemented in line with an extra constraint on $g_{\mathcal{R}}$. When normalize = 1, this corresponds to $\mathbf{R}$ being the correlation matrix of $\mathbf{Z}$. When normalize = 2, this corresponds to the identifiability condition of Derumigny & Fermanian (2022).

Value

The estimates for $g_{\mathcal{R}}$ at the grid points.

References

Derumigny, A., Fermanian, J.-D., Ryan, V., van der Spek, R. (2024). ElliptCopulas, R package version 0.1.4.1. url: https://CRAN.R-project.org/package=ElliptCopulas.

Derumigny, A. & Fermanian, J.-D. (2022). Identifiability and estimation of meta-elliptical copula generators. Journal of Multivariate Analysis 190:104962.
doi: https://doi.org/10.1016/j.jmva.2022.104962.

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

Liebscher, E. (2005). A semiparametric density estimator based on elliptical distributions. Journal of Multivariate Analysis 92(1):205-225. doi: https://doi.org/10.1016/j.jmva.2003.09.007.

See Also

`elldistrest` for improved kernel estimation of the elliptical generator of an elliptical distribution, `elliptselect` for selecting optimal tuning parameters for the improved kernel estimator of the elliptical generator, phiellip for estimating the Φ-dependence between k random vectors having a meta-elliptical copula.

Examples

```
q = 4

# Sample size
n = 1000

# Grid on which to evaluate the elliptical generator
grid = seq(0.005,100,by = 0.005)

# Degrees of freedom
nu = 7

# Student-t generator with 7 degrees of freedom
g_q = ((nu/(nu-2))^(q/2))*(gamma((q+nu)/2)/(((pi*nu)^(q/2))*gamma(nu/2))) *
      ((1+(grid/(nu-2)))^(-(q+nu)/2))

# Density of squared radius
R2 = function(t,q){(gamma((q+nu)/2)/(((nu-2)^(q/2))*gamma(nu/2)*gamma(q/2))) *
                   (t^((q/2)-1)) * ((1+(t/(nu-2)))^(-(q+nu)/2))}

# Sample from 4-dimensional Student-t distribution with 7 degrees of freedom
# and identity covariance matrix
sample = ElliptCopulas::EllDistrSim(n,q,diag(q),density_R2 = function(t){R2(t,q)})

# Copula pseudo-observations
pseudos = matrix(0,n,q)
for(j in 1:q){pseudos[,j] = (n/(n+1)) * ecdf(sample[,j])(sample[,j])}

# Shrinkage function
shrinkage = function(t,p){1-(1/((t^p) + 1))}

# Tuning parameter selection
opt_parameters = elliptselect(n,q,seq((3/4)-(1/q)+0.01,1-0.01,len = 200),
                                  seq(0.01,2,len = 200))

# Optimal tuning parameters
a = opt_parameters$Opta ; p = opt_parameters$Optp ; h = opt_parameters$Opth

# Estimated elliptical generator
g_est = ellcopest(dataU = pseudos,Sigma_m1 = diag(q),h = h,grid = grid,a = a,
                  shrink = function(t){shrinkage(t,p)})

plot(grid,g_est,type = "l", xlim = c(0,8))
lines(grid,g_q,col = "green")
```

elldistrest *elldistrest*

Description

This functions performs improved kernel density estimation of the generator of an elliptical distribution by using Liebscher's algorithm, combined with a shrinkage function.

Usage

```
elldistrest(
  Z,
  mu = 0,
  Sigma_m1,
  grid,
  h,
  Kernel = "epanechnikov",
  a,
  shrink,
  mpfr = FALSE,
  precBits = 100,
  dopb = FALSE,
  normalize = 1
)
```

Arguments

Z	A sample from a q-dimensional random vector $\mathbf{Z}$ ($n \times q$ matrix with observations in rows, variables in columns).
mu	The (estimated) mean of $\mathbf{Z}$ (default $= 0$).
Sigma_m1	The (estimated) inverse of the scale matrix of $\mathbf{Z}$.
grid	The grid of values on which to estimate the density generator.
h	The bandwidth of the kernel.
Kernel	The kernel used for the smoothing (default = "epanechnikov").
a	The tuning parameter to improve the performance at 0.
shrink	The shrinkage function to further improve the performance at 0 and guarantee the existence of the AMISE bandwidth.
mpfr	See the "EllDistrEst.R" function of the R package 'ElliptCopulas'.
precBits	See the "EllDistrEst.R" function of the R package 'ElliptCopulas'.
dopb	See the "EllDistrEst.R" function of the R package 'ElliptCopulas'.
normalize	A value in $\{1, 2\}$ indicating the normalization procedure that is applied to the estimated generator (default = 1).

Details

The context is the one of a q-dimensional random vector $\mathbf{Z}$ following an elliptical distribution with generator $g_{\mathcal{R}} : (0, \infty) \to \mathbb{R}$ and scale matrix $\mathbf{R}$ such that the density of $\mathbf{Z}$ is given by

$$h(\mathbf{z}) = |\mathbf{R}|^{-1/2} g_{\mathcal{R}} \left(\mathbf{z}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{z}\right),$$

for $\mathbf{z} \in \mathbb{R}^q$. Suppose that a sample $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(n)}$ from $\mathbf{Z}$ is given, and let $\widehat{\mathbf{R}}$ be an estimator for the scale matrix $\mathbf{R}$. Then, when defining

$$\widehat{\mathbf{Y}}^{(\ell)} = \widehat{\mathbf{R}}^{-1/2}\mathbf{Z}^{(\ell)}$$

for $\ell = 1, \ldots, n$, this function computes the estimator $\widehat{g}^{\mathrm{I}}_{\mathcal{R}}$ for $g_{\mathcal{R}}$ given by

$$\widehat{g}^{\mathrm{I}}_{\mathcal{R}}(t) = c^{\mathrm{I}}(t) \sum_{\ell=1}^{n} \left\{ k\left(\frac{\psi(t) - \psi\left(\left|\left|\widehat{\mathbf{Y}}^{(\ell)}\right|\right|^2\right)}{h_n \alpha\left(\psi(t)\right)} \right) + k\left(\frac{\psi(t) + \psi\left(\left|\left|\widehat{\mathbf{Y}}^{(\ell)}\right|\right|^2\right)}{h_n \alpha\left(\psi(t)\right)} \right) \right\},$$

where $c^{\mathrm{I}}(t) = [\Gamma(q/2)/(\pi^{q/2} n h_n \alpha(\psi(t)))] t^{-q/2+1} \psi'(t)$, with k the kernel and h_n the bandwidth. The function

$$\psi(t) = -a + \left(a^{q/2} + t^{q/2}\right)^{2/q},$$

with $a > 0$ a tuning parameter was introduced by Liebscher (2005), and the shrinkage function $\alpha(t)$ yields further estimation improvement. We suggest to take (for $q > 2$)

$$\alpha(t) = 1 - \frac{1}{t^{\delta} + 1},$$

where $\delta \in (3/4 - 1/q, 1)$ is another tuning parameter. When $q = 2$, one can just take $\alpha(t) = 1$, and the value of a does not matter.

The estimator without the shrinkage function α is implemented in the R package 'ElliptCopulas'. We use this implementation and bring in the shrinkage function.

In order to make $g_{\mathcal{R}}$ identifiable, an extra normalization procedure is implemented in line with an extra constraint on $g_{\mathcal{R}}$. When normalize = 1, this corresponds to $\mathbf{R}$ being the correlation matrix of $\mathbf{Z}$. When normalize = 2, this corresponds to the identifiability condition of Derumigny & Fermanian (2022).

Value

The estimates for $g_{\mathcal{R}}$ at the grid points.

References

Derumigny, A., Fermanian, J.-D., Ryan, V., van der Spek, R. (2024). ElliptCopulas, R package version 0.1.4.1. url: https://CRAN.R-project.org/package=ElliptCopulas.

Derumigny, A. & Fermanian, J.-D. (2022). Identifiability and estimation of meta-elliptical copula generators. Journal of Multivariate Analysis 190:104962.
doi: https://doi.org/10.1016/j.jmva.2022.104962.

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

Liebscher, E. (2005). A semiparametric density estimator based on elliptical distributions. Journal of Multivariate Analysis 92(1):205-225. doi: https://doi.org/10.1016/j.jmva.2003.09.007.

See Also

`ellcopest` for improved kernel estimation of the elliptical generator of a meta-elliptical copula, `elliptselect` for selecting optimal tuning parameters for the improved kernel estimator of the elliptical generator, `phiellip` for estimating the Φ-dependence between k random vectors having a meta-elliptical copula.

Examples

```
q = 4

# Sample size
n = 1000

# Grid on which to evaluate the elliptical generator
grid = seq(0.005,100,by = 0.005)

# Degrees of freedom
nu = 7

# Student-t generator with 7 degrees of freedom
g_q = ((nu/(nu-2))^(q/2))*(gamma((q+nu)/2)/(((pi*nu)^(q/2))*gamma(nu/2))) *
      ((1+(grid/(nu-2)))^(-(q+nu)/2))

# Density of squared radius
R2 = function(t,q){(gamma((q+nu)/2)/(((nu-2)^(q/2))*gamma(nu/2)*gamma(q/2))) *
                   (t^((q/2)-1)) * ((1+(t/(nu-2)))^(-(q+nu)/2))}

# Sample from 4-dimensional Student-t distribution with 7 degrees of freedom
# and identity covariance matrix
sample = ElliptCopulas::EllDistrSim(n,q,diag(q),density_R2 = function(t){R2(t,q)})

# Shrinkage function
shrinkage = function(t,p){1-(1/((t^p) + 1))}

# Tuning parameter selection
opt_parameters = elliptselect(n,q,seq((3/4)-(1/q)+0.01,1-0.01,len = 200),
                                  seq(0.01,2,len = 200))

# Optimal tuning parameters
a = opt_parameters$Opta ; p = opt_parameters$Optp ; h = opt_parameters$Opth

# Estimated elliptical generator
g_est = elldistrest(Z = sample, Sigma_m1 = diag(q), grid = grid, h = h, a = a,
                    shrink = function(t){shrinkage(t,p)})
```

```
plot(grid,g_est,type = "l", xlim = c(0,8))
lines(grid,g_q,col = "green")
```

`elliptselect` *elliptselect*

Description

This functions selects optimal tuning parameters for improved kernel estimation of the generator of an elliptical distribution.

Usage

```
elliptselect(n, q, pseq, aseq)
```

Arguments

`n`	The sample size.
`q`	The total dimension.
`pseq`	Candidate values for the δ parameter of the shrinkage function.
`aseq`	Candidate values for the a parameter of the Liebscher function.

Details

When using the function `elldistrest` for estimating an elliptical generator $g_{\mathcal{R}}$ based on a kernel k with bandwidth h_n, the function

$$\psi(t) = -a + \left(a^{q/2} + t^{q/2}\right)^{2/q},$$

and the shrinkage function (for $q > 3$)

$$\alpha(t) = 1 - \frac{1}{t^\delta + 1},$$

this function selects h_n, δ and a in the following way.

Use the normal generator $g_{\mathcal{R}}(t) = e^{-t/2}/(2\pi)^{q/2}$ as reference generator, and define

$$\Psi(t) = \frac{\pi^{q/2}}{\Gamma(q/2)} \left(\psi^{-1}(t)\right)' \left(\psi^{-1}(t)\right)^{q/2-1} g_{\mathcal{R}}\left(\psi^{-1}(t)\right),$$

as well as

$$h_n^{\text{opt}} = \left\{ \frac{\left(\int_{-1}^1 k^2(t)dt\right) \left(\int_0^\infty \alpha(t)^{-1}\Psi(t)dt\right)}{\left(\int_{-1}^1 t^2 k(t)dt\right)^2 \left(\int_0^\infty \left(\alpha(t)^2\Psi''(t)\right)^2 dt\right)} \right\}^{1/5} n^{-1/5}.$$

When $q = 2$, take $\alpha(t) = 1$ (there is no need for shrinkage), and take h_n^{opt}. The value of a does not matter.

When $q > 2$, specify a grid of candidate δ-values in $(3/4 - 1/q, 1)$ and a grid of a-values in $(0, \infty)$. For each of these candidate values, compute the corresponding optimal (AMISE) bandwidth h_n^{opt}. Take the combination of parameters that minimizes (a numerical approximation of) the (normal reference) AMISE given in equation (20) of De Keyser & Gijbels (2024).

Value

A list with elements "Opta" containing the optimal a, "Optp" containing the optimal δ, and "Opth" containing the optimal h_n.

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

See Also

`elldistrest` for improved kernel estimation of the elliptical generator of an elliptical distribution, `ellcopest` for improved kernel estimation of the elliptical generator of a meta-elliptical copula, `phiellip` for estimating the Φ-dependence between k random vectors having a meta-elliptical copula.

Examples

```
q = 4
n = 1000
opt_parameters = elliptselect(n,q,seq((3/4)-(1/q)+0.01,1-0.01,len = 200),
                              seq(0.01,2,len = 200))
```

---

estphi *estphi*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function estimates the Φ-dependence between $\mathbf{X}_1, ..., \mathbf{X}_k$ by estimating the joint and marginal copula densities.

Usage

```
estphi(sample, dim, est_method, phi)
```

Arguments

`sample`	A sample from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
`dim`	The vector of dimensions $(d_1, ..., d_k)$.
`est_method`	The method used for estimating the Φ-dependence.
`phi`	The function Φ.

Details

When $\mathbf{X}$ has copula density c with marginal copula densities c_i of $\mathbf{X}_i$ for $i = 1, \ldots, k$, the Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$ equals

$$\mathcal{D}_\Phi\left(\mathbf{X}_1, \ldots, \mathbf{X}_k\right) = \mathbb{E}\left\{\frac{\prod_{i=1}^k c_i(\mathbf{U}_i)}{c\left(\mathbf{U}\right)} \Phi\left(\frac{c(\mathbf{U})}{\prod_{i=1}^k c_i(\mathbf{U}_i)}\right)\right\},$$

for a certain continuous, convex function $\Phi : (0, \infty) \to \mathbb{R}$, and with $\mathbf{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_k) \sim c$.

This functions allows to estimate $\mathcal{D}_\Phi$ in several ways (options for est_method)

- list("hac", type = type, M = M) for parametric estimation by fitting a hierarchical Archimedean copula (hac) via pseudo-maximum likelihood estimation, using a generator of type = type and a simulated Monte Carlo sample of size M in order to approximate the expectation, see also the functions `mlehac` and `phihac`,
- list("nphac", estimator = estimator, type = type) for fully non-parametric estimation using the beta kernel estimator or Gaussian transformation kernel estimator using a fitted hac (via pseudo-maximum likelihood estimation) of type = type to find locally optimal bandwidths, see also the function `phinp`,
- list("np", estimator = estimator, bw_method = bw_method) for fully non-parametric estimation using the beta kernel estimator or Gaussian transformation kernel estimator, see `phinp` for different bw_method arguments (either 1 or 2, for performing local bandwidth selection),
- list("ellip", grid = grid) for semi-parametric estimation through meta-elliptical copulas, with bandwidths determined by the `elliptselect` function, see also the function `phiellip`.

Value

The estimated Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

See Also

phihac for computing the Φ-dependence between all the child copulas of a hac object with two nesting levels, phinp for fully non-parametric estimation of the Φ-dependence between k random vectors, phiellip for estimating the Φ-dependence between k random vectors having a meta-elliptical copula.

Examples

```
# Hierarchical Archimedean copula setting
q = 4
dim = c(2,2)

# Sample size
n = 1000

# Four dimensional hierarchical Gumbel copula
# with parameters (theta_0,theta_1,theta_2) = (2,3,4)
hac = gethac(dim,c(2,3,4),type = 1)

# Sample
sample =  suppressWarnings(HAC::rHAC(n,hac))

# Several estimators for the mutual information between two random vectors of size 2

est_phi_1 = estphi(sample,dim,list("hac",type = 1,M = 10000),function(t){t * log(t)})
est_phi_2 = estphi(sample,dim,list("nphac",estimator = "beta",type = 1),
                                   function(t){t * log(t)})
est_phi_3 = estphi(sample,dim,list("nphac",estimator = "trans",type = 1),
                                   function(t){t * log(t)})
est_phi_4 = estphi(sample,dim,list("np",estimator = "beta",bw_method = 1),
                                   function(t){t * log(t)})
est_phi_5 = estphi(sample,dim,list("np",estimator = "trans",bw_method = 1),
                                   function(t){t * log(t)})
est_phi_6 = estphi(sample,dim,list("np",estimator = "beta",bw_method = 2),
                                   function(t){t * log(t)})
est_phi_7 = estphi(sample,dim,list("np",estimator = "trans",bw_method = 2),
                                   function(t){t * log(t)})

true_phi = phihac(hac,dim,10000,function(t){t * log(t)})

# Gaussian copula setting

q = 4
dim = c(2,2)

# Sample size
n = 1000

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))
```

```
# Sample from 4-dimensional normal distribution
sample = mvtnorm::rmvnorm(n,rep(0,q),R,method = "chol")

# Estimate mutual information via MECIP procedure
est_phi = estphi(sample,dim,list("ellip",grid = seq(0.005,100,by = 0.005)),
                              function(t){t * log(t)})

true_phi = minormal(R,dim)
```

---

estR *estR*

Description

This function computes the sample Q-scores rank correlation matrix. A ridge penalization is possible.

Usage

```
estR(
  sample,
  omega = 1,
  Q = function(t) {
     stats::qnorm(t)
 }
)
```

Arguments

`sample`	A sample from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
`omega`	The penalty parameter for ridge penalization (default = 1, meaning no penalization).
`Q`	The quantile function to be applied to the copula pseudo-observations (default = qnorm()).

Details

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i = (X_{i1}, \ldots, X_{id_i})$ a d_i dimensional random vector, i.e., $q = d_1 + ... + d_k$, the sample Q-scores rank correlation matrix is given as

$$\widehat{\mathbf{R}}_n = \begin{pmatrix} \widehat{\mathbf{R}}_{11} & \widehat{\mathbf{R}}_{12} & \cdots & \widehat{\mathbf{R}}_{1k} \\ \widehat{\mathbf{R}}_{12}^{\mathrm{T}} & \widehat{\mathbf{R}}_{22} & \cdots & \widehat{\mathbf{R}}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{\mathbf{R}}_{1k}^{\mathrm{T}} & \widehat{\mathbf{R}}_{2k}^{\mathrm{T}} & \cdots & \widehat{\mathbf{R}}_{kk} \end{pmatrix} \quad \text{with} \quad \left(\widehat{\mathbf{R}}_{im}\right)_{jt} = \widehat{\rho}_{ij,mt} = \frac{\frac{1}{n}\sum_{\ell=1}^{n} \widehat{Z}_{ij}^{(\ell)} \widehat{Z}_{mt}^{(\ell)}}{\frac{1}{n}\sum_{\ell=1}^{n} \left[Q\left(\frac{\ell}{n+1}\right)\right]^2},$$

for $i, m = 1, \ldots, k$, $j = 1, \ldots, d_i$, and $t = 1, \ldots, d_m$, based on the observed Q-scores

$$\widehat{Z}_{ij}^{(\ell)} = Q\left(\frac{n}{n+1}\widehat{F}_{ij}\left(X_{ij}^{(\ell)}\right)\right) = Q\left(\frac{1}{n+1}\sum_{t=1}^{n} 1\left\{X_{ij}^{(t)} \leq X_{ij}^{(\ell)}\right\}\right),$$

for $\ell = 1, \ldots, n$, where $\widehat{F}_{ij}$ is the empirical cdf of the sample $X_{ij}^{(1)}, \ldots, X_{ij}^{(n)}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, d_i$. The underlying assumption is that the copula of $\mathbf{X}$ is meta-elliptical. The default for Q is the standard normal quantile function (corresponding to the assumption of a Gaussian copula). Ridge penalization (especially in the Gaussian copula setting) with penalty parameter omega = ω boils down to computing

$$\omega\widehat{\mathbf{R}}_n + (1-\omega)\mathbf{I}_q,$$

where $\mathbf{I}_q$ stands for the identity matrix.

Value

The (ridge penalized) sample Q-scores rank correlation matrix.

References

De Keyser, S. & Gijbels, I. (2024). Some new tests for independence among continuous random vectors.

Warton, D.I. (2008). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. Journal of the American Statistical Association 103(481):340-349.
doi: https://doi.org/10.1198/016214508000000021.

See Also

cvomega for selecting omega using K-fold cross-validation in case of a Gaussian copula.

Examples

```
# Multivariate normal copula setting

q = 10
n = 50

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

# Sample from multivariate normal distribution
sample = mvtnorm::rmvnorm(n,rep(0,q),R,method = "chol")

# 5-fold cross-validation with Gaussian likelihood as loss for selecting omega
omega = cvomega(sample = sample,omegas = seq(0.01,0.999,len = 50),K = 5)

R_est = estR(sample,omega = omega)

# Multivariate Student-t copula setting

q = 10
```

```
n = 500

# Degrees of freedom
nu = 7

# Density of R^2, with R the radius of the elliptical distribution
# Identifiability contraint is that R is the correlation matrix
R2 = function(t,q){(gamma((q+nu)/2)/(((nu-2)^(q/2)) * gamma(nu/2) * gamma(q/2))) *
                   (t^((q/2)-1)) * ((1+(t/(nu-2)))^(-(q+nu)/2))}

# Univariate quantile function, with unit variance
Q = function(t){extraDistr::qlst(t,nu,0,sqrt((nu-2)/nu))}

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

# Sample from multivariate Student-t distribution
# with correlation matrix R and nu degrees of freedom
sample = EllipCopulas::EllDistrSim(n,q,t(chol(R)),density_R2 = function(t){R2(t,q)})

R_est = estR(sample,Q = Q)
```

---

gethac *gethac*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function construct a hac object (hierarchical Archimedean copula) with two nesting levels given the specified dimensions and parameters of the root and k child copulas.

Usage

```
gethac(dim, thetas, type)
```

Arguments

`dim`	The vector of dimensions $(d_1, ..., d_k)$.
`thetas`	The parameters $(\theta_0, \theta_1, \dots, \theta_k)$.
`type`	The type of Archimedean copula.

Details

A hierarchical (or nested) Archimedean copula C with two nesting levels and k child copulas is given by

$$C(\mathbf{u}) = C_0\left(C_1(\mathbf{u}_1), \dots, C_k(\mathbf{u}_k)\right),$$

where $\mathbf{u} = (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \mathbb{R}^q$ with $\mathbf{u}_i \in \mathbb{R}^{d_i}$ for $i = 1, \dots, k$. The (k-dimensional) copula C_0 is called the root copula, and the (d_i-dimensional) copulas C_i are the child copulas.

They all belong to the class of Archimedean copulas, and we denote θ_i for the parameter of C_i for $i = 0, 1, \ldots, k$. A sufficient condition to guarantee that C indeed is a copula, is that $C_0, C_1, \ldots, C_k$ are all a particular member of this class of Archimedean copulas (e.g., Clayton), and such that $\theta_0 \leq \theta_i$ for $i = 1, \ldots, k$ (sufficient nesting condition).

When a certain child copula C_i is one dimensional ($\mathbf{X}_i$ is one dimensional), θ_i can be any number. It must hold that length(thetas) $= k + 1$.

Many functions for working with nested Archimedean copulas are developed in the R package 'HAC', and the function gethac utilizes these functions to quickly construct a hac object that is useful for modelling the dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$. See also the R package 'HAC' for the different possibilities of type (specified by a number in $\{1, \ldots, 10\}$).

Value

A hac object with two nesting levels and k child copulas.

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

Okhrin, O., Ristig, A. & Chen, G. (2024). HAC: estimation, simulation and visualization of hierarchical Archimedean copulae (HAC), R package version 1.1-1.
url: https://CRAN.R-project.org/package=HAC.

See Also

phihac for computing the Φ-dependence between all the child copulas of a hac object with two nesting levels, Helhac for computing the Hellinger distance between all the child copulas of a hac object with two nesting levels, mlehac for maximum pseudo-likelihood estimation of the parameters of a hac object with two nesting levels.

Examples

```
dim = c(3,5,1,2)
thetas = c(2,2,3,1,4)

# 11 dimensional nested Gumbel copula with
# (theta_0,theta_1,theta_2,theta_3,theta_4) = (2,2,3,1,4),
# where the value of theta_3 could be anything,
# because the third random vector is one dimensional

HAC = gethac(dim,thetas,type = 1)
```

grouplasso *grouplasso*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the empirical penalized Gaussian copula covariance matrix with the Gaussian log-likelihood plus the grouped lasso penalty as objective function, where the groups are the diagonal and off-diagonal blocks corresponding to the different random vectors. Model selection is done by choosing omega such that BIC is maximal.

Usage

```
grouplasso(Sigma, S, n, omegas, dim, step.size = 100, trace = 0)
```

Arguments

`Sigma`	An initial guess for the covariance matrix (typically equal to S).
`S`	The sample matrix of normal scores covariances.
`n`	The sample size.
`omegas`	The candidate values for the tuning parameter in $[0, \infty)$.
`dim`	The vector of dimensions $(d_1, ..., d_k)$.
`step.size`	The step size used in the generalized gradient descent, affects the speed of the algorithm (default = 100).
`trace`	Controls how verbose output should be (default = 0, meaning no verbose output).

Details

Given a covariance matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} & \cdots & \mathbf{\Sigma}_{1k} \\ \mathbf{\Sigma}_{12}^{\text{T}} & \mathbf{\Sigma}_{22} & \cdots & \mathbf{\Sigma}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Sigma}_{1k}^{\text{T}} & \mathbf{\Sigma}_{2k}^{\text{T}} & \cdots & \mathbf{\Sigma}_{kk} \end{pmatrix},$$

the aim is to solve/compute

$$\widehat{\mathbf{\Sigma}}_{\text{GLT},n} \in \text{arg min}_{\mathbf{\Sigma}>0} \left\{ \ln|\mathbf{\Sigma}| + \text{tr}\left(\mathbf{\Sigma}^{-1}\widehat{\mathbf{\Sigma}}_n\right) + P_{\text{GLT}}\left(\mathbf{\Sigma}, \omega_n\right) \right\},$$

where the penalty function P_{GLT} is of group lasso-type:

$$P_{\text{GLT}}\left(\mathbf{\Sigma}, \omega_n\right) = 2\sum_{i,j=1,j>i}^{k} p_{\omega_n}\left(\sqrt{d_i d_j}\,||\mathbf{\Sigma}_{ij}||_{\text{F}}\right) + \sum_{i=1}^{k} p_{\omega_n}\left(\sqrt{d_i(d_i-1)}\,||\mathbf{\Delta}_i * \mathbf{\Sigma}_{ii}||_{\text{F}}\right),$$

for a certain penalty function p_{ω_n} with penalty parameter ω_n, and $\mathbf{\Delta}_i \in \mathbb{R}^{d_i \times d_i}$ a matrix with ones as off-diagonal elements and zeroes on the diagonal (in order to avoid shrinking the variances, the operator $*$ stands for elementwise multiplication).

For now, the only possibility in this function for p_{ω_n} is the lasso penalty $p_{\omega_n}(t) = \omega_n t$. For other penalties (e.g., scad), one can do a local linear approximation to the penalty function and iteratively perform weighted group lasso optimizations (similar to what is done in the function covgpenal).

Regarding the implementation, we used the code available in the R package 'spcov' (see the manual for further explanations), but altered it to the context of a group-lasso penalty.

For tuning ω_n, we maximize (over a grid of candidate values) the BIC criterion

$$\text{BIC}\left(\widehat{\mathbf{\Sigma}}_{\omega_n}\right) = -n\left[\ln\left|\widehat{\mathbf{\Sigma}}_{\omega_n}\right| + \text{tr}\left(\widehat{\mathbf{\Sigma}}_{\omega_n}^{-1}\widehat{\mathbf{\Sigma}}_n\right)\right] - \ln(n)\text{df}\left(\widehat{\mathbf{\Sigma}}_{\omega_n}\right),$$

where $\widehat{\mathbf{\Sigma}}_{\omega_n}$ is the estimated candidate covariance matrix using ω_n and df (degrees of freedom) equals

$$\text{df}\left(\widehat{\mathbf{\Sigma}}_{\omega_n}\right) = \sum_{i,j=1,j>i}^{k} 1\left(\left|\left|\widehat{\mathbf{\Sigma}}_{\omega_n,ij}\right|\right|_{\text{F}} > 0\right)\left(1 + \frac{\left|\left|\widehat{\mathbf{\Sigma}}_{\omega_n,ij}\right|\right|_{\text{F}}}{\left|\left|\widehat{\mathbf{\Sigma}}_{n,ij}\right|\right|_{\text{F}}}(d_i d_j - 1)\right)$$
$$+ \sum_{i=1}^{k} 1\left(\left|\left|\mathbf{\Delta}_i * \widehat{\mathbf{\Sigma}}_{\omega_n,ii}\right|\right|_{\text{F}} > 0\right)\left(1 + \frac{\left|\left|\mathbf{\Delta}_i * \widehat{\mathbf{\Sigma}}_{\omega_n,ii}\right|\right|_{\text{F}}}{\left|\left|\mathbf{\Delta}_i * \widehat{\mathbf{\Sigma}}_{n,ii}\right|\right|_{\text{F}}}\left(\frac{d_i(d_i-1)}{2} - 1\right)\right) + q,$$

with $\widehat{\mathbf{\Sigma}}_{\omega_n,ij}$ the (i,j)'th block of $\widehat{\mathbf{\Sigma}}_{\omega_n}$, similarly for $\widehat{\mathbf{\Sigma}}_{n,ij}$.

Value

A list with elements "est" containing the (group lasso) penalized matrix of sample normal scores rank correlations (output as provided by the function "spcov.R"), and "omega" containing the optimal tuning parameter.

References

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

Bien, J. & Tibshirani, R. (2022). spcov: sparse estimation of a covariance matrix, R package version 1.3. url: https://CRAN.R-project.org/package=spcov.

Bien, J. & Tibshirani, R. (2011). Sparse Estimation of a Covariance Matrix. Biometrika 98(4):807-820. doi: https://doi.org/10.1093/biomet/asr054.

See Also

covgpenal for (elementwise) lasso-type estimation of the normal scores rank correlation matrix.

Examples

```
q = 10
dim = c(5,5)
n = 100

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))
```

```
# Sparsity on off-diagonal blocks
R0 = createR0(R,dim)

# Sample from multivariate normal distribution
sample = mvtnorm::rmvnorm(n,rep(0,q),R0,method = "chol")

# Normal scores
scores = matrix(0,n,q)
for(j in 1:q){scores[,j] = qnorm((n/(n+1)) * ecdf(sample[,j])(sample[,j]))}

# Sample matrix of normal scores covariances
Sigma_est = cov(scores) * ((n-1)/n)

# Candidate tuning parameters
omega = seq(0.01, 0.6, length = 50)

Sigma_est_penal = grouplasso(Sigma_est, Sigma_est, n, omega, dim)
```

`hamse` *hamse*

Description

This function performs local bandwidth selection based on the amse (asymptotic mean squared error) for the beta kernel or Gaussian transformation kernel copula density estimator.

Usage

```
hamse(input, cop = NULL, pseudos = NULL, n, estimator, bw_method)
```

Arguments

`input`	The copula argument at which the optimal local bandwidth is to be computed.
`cop`	A fitted reference hac object, in case bw_method = 0 (default = NULL).
`pseudos`	The (estimated) copula observations from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns), in case bw_method = 1 (default = NULL).
`n`	The sample size.
`estimator`	Either "beta" or "trans" for the beta kernel or the Gaussian transformation kernel copula density estimator.
`bw_method`	A number in $\{0, 1\}$ specifying the method used for computing the bandwidth.

Details

When estimator = "beta", this function computes, at a certain input, a numerical approximation of the optimal local bandwidth (for the beta kernel copula density estimator) in terms of the amse (asymptotic mean squared error) given in equation (27) of De Keyser & Gijbels (2024). When estimator = "trans" (for the Gaussian transformation kernel copula density estimator), this optimal bandwidth is given at the end of Section 5.2 in De Keyser & Gijbels (2024).

Of course, these optimal bandwidths depend upon the true unknown copula. If bw_method = 0, then the given fitted (e.g., via MLE using mlehac) hac object (hierarchical Archimedean copula) cop is used as reference copula. If bw_method = 1, then a non-parametric (beta or Gaussian transformation) kernel copula density estimator based on the pseudos as pivot is used. This pivot is computed using the big O bandwidth (i.e., $n^{-2/(q+4)}$ in case of the beta estimator, and $n^{-1/(q+4)}$ for the transformation estimator, with q the total dimension).

Value

The optimal local bandwidth (in terms of amse).

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

See Also

betakernelestimator for the computation of the beta kernel copula density estimator, transformationestimator for the computation of the Gaussian transformation kernel copula density estimator, phinp for fully non-parametric estimation of the Φ-dependence between k random vectors.

Examples

```
q = 4
dim = c(2,2)

# Sample size
n = 1000

# Four dimensional hierarchical Gumbel copula
# with parameters (theta_0,theta_1,theta_2) = (2,3,4)
HAC = gethac(dim,c(2,3,4),type = 1)

# Sample
sample = suppressWarnings(HAC::rHAC(n,HAC))

# Copula pseudo-observations
pseudos = matrix(0,n,q)
for(j in 1:q){pseudos[,j] = (n/(n+1)) * ecdf(sample[,j])(sample[,j])}

# Maximum pseudo-likelihood estimator to be used
```

```
# as reference copula for bw_method = 0
est_cop = mlehac(sample,dim,1,c(2,3,4))

h_1 = hamse(rep(0.5,q),cop = est_cop,n = n,estimator = "beta",bw_method = 0)
h_2 = hamse(rep(0.5,q),cop = est_cop,n = n,estimator = "trans",bw_method = 0)
h_3 = hamse(rep(0.5,q),pseudos = pseudos,n = n,estimator = "beta",bw_method = 1)
h_4 = hamse(rep(0.5,q),pseudos = pseudos,n = n,estimator = "trans",bw_method = 1)

est_dens_1 = betakernelestimator(rep(0.5,q),h_1,pseudos)
est_dens_2 = transformationestimator(rep(0.5,q),h_2,pseudos)
est_dens_3 = betakernelestimator(rep(0.5,q),h_3,pseudos)
est_dens_4 = transformationestimator(rep(0.5,q),h_4,pseudos)

true = HAC::dHAC(c("X1" = 0.5, "X2" = 0.5, "X3" = 0.5, "X4" = 0.5), HAC)
```

Helhac *Helhac*

Description

This function computes the Hellinger distance between all the child copulas of a hac object obtained by the function gethac, i.e., given a q-dimensional random vector $\mathbf{X} = (\mathbf{X_1}, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, where $\mathbf{X}_1, ..., \mathbf{X}_k$ are connected via a hierarchical Archimedean copula with two nesting levels, Helhac computes the Hellinger distance between $\mathbf{X}_1, ..., \mathbf{X}_k$.

Usage

```
Helhac(cop, dim, M)
```

Arguments

cop	A hac object as provided by the function gethac.
dim	The vector of dimensions $(d_1, ..., d_k)$.
M	The size of the Monte Carlo sample used for approximating the integral of the Hellinger distance.

Details

When $\mathbf{X}$ has copula density c with marginal copula densities c_i of $\mathbf{X}_i$ for $i = 1, \ldots, k$, the Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$ equals

$$\mathcal{D}_\Phi \left(\mathbf{X}_1, \ldots, \mathbf{X}_k\right) = \int_{[0,1]^q} \prod_{i=1}^{k} c_i(\mathbf{u}_i) \Phi \left(\frac{c(\mathbf{u})}{\prod_{i=1}^{k} c_i(\mathbf{u}_i)} \right),$$

for a certain continuous, convex function $\Phi : (0, \infty) \to \mathbb{R}$. The Hellinger distance corresponds to $\Phi(t) = (\sqrt{t} - 1)^2$, and $\mathcal{D}_{(\sqrt{t}-1)^2}$ could be approximated by $\widehat{\mathcal{D}}_{(\sqrt{t}-1)^2}$ as implemented in the function phihac. Yet, for this specific choice of Φ, it is better to first simplify $\mathcal{D}_{(\sqrt{t}-1)^2}$ to

$$\mathcal{D}_{(\sqrt{t}-1)^2}(\mathbf{X}_1, \dots, \mathbf{X}_k) = 2 - 2 \int_{[0,1]^q} \sqrt{c(\mathbf{u}) \prod_{i=1}^{k} c_i(\mathbf{u}_i)} d\mathbf{u},$$

and then, by drawing a sample of size M from c, say $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(M)}$, with $\mathbf{U}^{(\ell)} = (\mathbf{U}_1^{(\ell)}, \dots, \mathbf{U}_k^{(\ell)})$, approximate it by

$$\widetilde{D}_{(\sqrt{t}-1)^2} = 2 - \frac{2}{M} \sum_{\ell=1}^{M} \sqrt{\frac{\prod_{i=1}^{k} c_i\left(\mathbf{U}_i^{(\ell)}\right)}{c\left(\mathbf{U}^{(\ell)}\right)}}.$$

The function Helhac computes $\widetilde{\mathcal{D}}_{(\sqrt{t}-1)^2}$ when c is a hierarchical Archimedean copula with two nesting levels, as produced by the function gethac.

Value

The Hellinger distance between $\mathbf{X}_1, \dots, \mathbf{X}_k$ (i.e., between all the child copulas of the hac object).

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

See Also

gethac for creating a hac object with two nesting levels, phihac for computing the Φ-dependence between all the child copulas of a hac object with two nesting levels, mlehac for maximum pseudo-likelihood estimation of the parameters of a hac object with two nesting levels.

Examples

```
dim = c(2,2)
thetas = c(2,3,4)

# 4 dimensional nested Gumbel copula with (theta_0,theta_1,theta_2) = (2,3,4)
HAC = gethac(dim,thetas,type = 1)

# Hellinger distance based on Monte Carlo sample of size 10000
Hel = Helhac(HAC,dim,10000)
```

Helnormal *Helnormal*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the correlation-based Hellinger distance between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$.

Usage

```
Helnormal(R, dim)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$.

Details

Given a correlation matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1k} \\ \mathbf{R}_{12}^{\mathrm{T}} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1k}^{\mathrm{T}} & \mathbf{R}_{2k}^{\mathrm{T}} & \cdots & \mathbf{R}_{kk} \end{pmatrix},$$

the Hellinger distance equals

$$\mathcal{D}^{\mathcal{N}}_{(\sqrt{t}-1)^2}(\mathbf{R}) = 2 - 2\frac{2^{q/2}|\mathbf{R}|^{1/4}}{\left|\mathbf{I}_q + \mathbf{R}_0^{-1}\mathbf{R}\right|^{1/2} \prod_{i=1}^{k} |\mathbf{R}_{ii}|^{1/4}},$$

where $\mathbf{I}_q$ denotes the identity matrix, and $\mathbf{R}_0 = \mathrm{diag}(\mathbf{R}_{11}, \ldots, \mathbf{R}_{kk})$ is the correlation matrix under independence of $\mathbf{X}_1, \ldots, \mathbf{X}_k$. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The correlation-based Hellinger distance between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

See Also

`minormal` for the computation of the Gaussian copula mutual information, `Helnormalavar` for the computation of the asymptotic variance of the plug-in estimator for the Gaussian copula Hellinger distance.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

Helnormal(R,dim)
```

Helnormalavar *Helnormalavar*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the asymptotic variance of the plug-in estimator for the correlation-based Hellinger distance between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$.

Usage

```
Helnormalavar(R, dim)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$.

Details

The asymptotic variance of the plug-in estimator $\mathcal{D}_{(\sqrt{t}-1)^2}(\widehat{\mathbf{R}}_n)$ is computed at $\mathbf{R}$, where $\widehat{\mathbf{R}}_n$ is the sample matrix of normal scores rank correlations. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The asymptotic variance of the correlation-based Hellinger distance between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

See Also

minormal for the computation of the mutual information, Helnormal for the computation of the Hellinger distance, minormalavar for the computation of the asymptotic variance of the plug-in estimator for the mutual information, estR for the computation of the sample matrix of normal scores rank correlations.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

Helnormalavar(R,dim)
```

Icluster *Icluster*

Description

This function clusters the columns (variables) of a dataset via agglomerative hierarchical variable clustering using estimated multivariate similarities (dependence coefficients) between random vectors.

Usage

```
Icluster(
  data,
  est_method,
  max_dim = Inf,
  norm = NULL,
  link = "average",
  trace = 1
)
```

Arguments

data	The dataset ($n \times q$ matrix with observations in rows, variables in columns) whose columns need to be clustered.
est_method	The method for estimating the similarity between two clusters of variables.
max_dim	The maximum dimension of the random vectors for which no link function is used when computing the similarity (default = Inf).
norm	A possible normalization function applied to the dependence measure (default = NULL, meaning no normalization).
link	The link function to be used when max_dim is exceeded (default = "average").
trace	Controls how verbose output should be (default = 1, showing the progress).

Details

Suppose that the q variables (of which we have n observations in data) are $\mathcal{S} = \{X_1, \ldots, X_q\}$. Then, most important in hierarchical variable clustering, is computing the similarity

$$\mathcal{D}(\mathbb{X}, \mathbb{Y})$$

between two disjoint subsets of variables $\mathbb{X}, \mathbb{Y} \subset \mathcal{S}$. In particular, the main algorithm is as follows:

- Each object $\{X_i\}$ forms a separate cluster, i.e., $\aleph_1 = \{\{X_1\}, \ldots, \{X_q\}\}$ is the initial feature partition.

For $i = 1, 2, \ldots, q-1$:

- For each pair of disjoint clusters $\mathbb{X}, \mathbb{Y} \in \aleph_i$, compute the similarity $\mathcal{D}(\mathbb{X}, \mathbb{Y})$.
- Define $\aleph_{i+1} = (\aleph_i \setminus \{\widetilde{\mathbb{X}}, \widetilde{\mathbb{Y}}\}) \cup \{\widetilde{\mathbb{X}} \cup \widetilde{\mathbb{Y}}\}$, where $\widetilde{\mathbb{X}}, \widetilde{\mathbb{Y}}$ are the clusters having maximal similarity according to the previous step.
- The algorithm stops with $\aleph_q = \{\{X_1, \ldots, X_q\}\}$.

We call $\{\aleph_1, \ldots, \aleph_q\}$ the hierarchy constructed throughout the algorithm, and define, for $i \in \{1, \ldots, q\}$, Adiam$(\aleph_i) = |\aleph_i|^{-1} \sum_{\mathbb{X} \in \aleph_i}$ diam$(\mathbb{X})$, with

$$\text{diam}(\mathbb{X}) = \begin{cases} \min\limits_{\{X,Y\} \subset \mathbb{X}} \mathcal{D}(X, Y) & \text{if } |\mathbb{X}| > 1 \\ 1 & \text{if } |\mathbb{X}| = 1, \end{cases}$$

and Msplit$(\aleph_i) = \max_{\mathbb{X} \in \aleph_i}$ split$(\mathbb{X})$, with

$$\text{split}(\mathbb{X}) = \max_{\substack{X \in \mathbb{X} \\ Y \in \aleph_i \setminus \mathbb{X}}} \mathcal{D}(X, Y) \quad \text{for} \quad \{\mathbb{X}\} \neq \aleph_i.$$

Adiam stands for the average diameter of a partition (measuring the homogeneity, which we want to be large), while Msplit stands for the maximum split of a partition (measuring the non-separation, which we want to be small).

For measuring the similarity $\mathcal{D}(\mathbb{X}, \mathbb{Y})$, we approach $\mathbb{X}$ and $\mathbb{Y}$ as being two random vectors (let's say of dimensions d_1 and d_2 respectively). For $\mathcal{D}$, we take an estimated dependence measure between (two) random vectors. The following options are possible:

- list("phi", "mi", "Gauss", omegas = omegas) for the estimated Gaussian copula mutual information. Use omegas = 1 for no penalty, or a sequence of omegas for a ridge penalty tuned via 5-fold cross-validation, see also the functions `minormal`, `estR`, and `cvomega`,
- list("phi", "Hel", "Gauss", omegas = omegas) for the estimated Gaussian copula Hellinger distance. Use omegas = 1 for no penalty, or a sequence of omegas for a ridge penalty tuned via 5-fold cross-validation, see also the functions `Helnormal`, `estR`, and `cvomega`,
- list("phi", phi(t), "hac", type = type, M = M) for general Φ-dependence with specified function phi(t) = $\Phi(t)$, estimated by fitting (via pseudo maximum likelihood estimation) a hierarchical Archimedean copula of given type = type, and computed based on a Monte Carlo sample of size M in order to approximate the expectation, see also the functions `mlehac`, `phihac` and `estphi`,

- list("phi", phi(t), "nphac", estimator = estimator, type = type) for general Φ-dependence with specified function phi(t) = $\Phi(t)$, estimated via non-parametric beta kernel estimation or Gaussian transformation kernel estimation, and local bandwidth selection, by using a fitted (via pseudo maximum likelihood) hierarchical Archimedean copula as reference copula, see also the functions phinp and estphi,
- list("phi", phi(t), "np", estimator = estimator, bw_method = bw_method) for general Φ-dependence with specified function phi(t) = $\Phi(t)$, estimated via non-parametric beta kernel estimation or Gaussian transformation kernel estimation, and local bandwidth selection, either by using a non-parametric kernel estimator as reference copula if bw_method = 1, or by using a big O bandwidth rule if bw_method = 2, see also the functions phinp and estphi,
- list("phi", phi(t), "ellip", grid = grid) for general Φ-dependence with specified function phi(t) = $\Phi(t)$, estimated via the improved MECIP procedure on the specified grid, and parameter selection done via the function elliptselect using the Gaussian generator as reference generator, see also the functions phiellip and estphi,
- list("ot", coef = coef, omegas = omegas) for Gaussian copula Bures-Wasserstein dependence measures, either coefficient $\mathcal{D}_1$ (coef = 1) or coefficient $\mathcal{D}_2$ (coef = 2). Use omegas = 1 for no penalty, or a sequence of omegas for a ridge penalty tuned via 5-fold cross-validation, see also the functions bwd1, bwd2, estR, and cvomega.

When $d_1 + d_2 >$ max_dim, the specified link function (say L) is used for computing the similarity between $\mathbb{X}$ and $\mathbb{Y}$, i.e.,

$$\mathcal{D}\left(\mathbb{X}, \mathbb{Y}\right) = L\left(\{\mathcal{D}(X, Y) : X \in \mathbb{X}, Y \in \mathbb{Y}\}\right),$$

which by default is the average of all inter-pairwise similarities. Other options are "single" for the minimum, and "complete" for the maximum.

The function norm (say N) is a possible normalization applied to the similarity measure, i.e., instead of computing $\mathcal{D}$ (using the method specified by est_method), the similarity becomes $N \circ \mathcal{D}$. The default is $N(t) = t$, meaning that no normalization is applied.

Value

A list with elements "hierarchy" containing the hierarchy constructed throughout the algorithm (a hash object), "all" containing all similarities that were computed throughout the algorithm (a hash object), "diam" containing the average diameters of all partitions created throughout the algorithm (a vector), and "split" containing the maximum splits of all partitions created throughout the algorithm (a vector).

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

De Keyser, S. & Gijbels, I. (2024). High-dimensional copula-based Wasserstein dependence. doi: https://doi.org/10.48550/arXiv.2404.07141.

See Also

minormal for the computation of the Gaussian copula mutual information, Helnormal for the computation of the Gaussian copula Hellinger distance, estphi for several approach to estimating the Φ-dependence between k random vectors, bwd1 for the computation of the first Bures-Wasserstein dependence coefficient $\mathcal{D}_1$, bwd2 for the computation of the second Bures-Wasserstein dependence coefficient $\mathcal{D}_2$.

Examples

```
q = 20

# We will impose a clustering
# {{X1,X2},{X3,X4,X5},{X6,X7,X8},{X9,X10,X11,X12,X13},{X14,X15,X16,X17,X18,X19,X20}}
dim = c(2,3,3,5,7)

# Sample size
n = 200

# Twenty dimensional hierarchical Gumbel copula with parameters
# (theta_0,theta_1,theta_2,theta_3,theta_4,theta_5) = (2,3,4,5,6,7)
hac = gethac(dim,c(2,3,4,5,6,7),type = 1)

# So, strongest cluster is {X14,X15,X16,X17,X18,X19,X20}, then {X9,X10,X11,X12,X13},
# then {X6,X7,X8}, then {X3,X4,X5}, and finally {X1,X2}

# Sample
sample =  suppressWarnings(HAC::rHAC(n,hac))

# Cluster using different methods

# Gaussian copula based methods

Clustering1 =  Icluster(data = sample,
                        est_method = list("phi", "mi", "Gauss", omegas = 1))

# 5-cluster partition
Clustering1$hierarchy$Aleph_16

Clustering2 =  Icluster(data = sample,
                        est_method = list("phi", "mi", "Gauss",
                                          omegas = seq(0.01,0.999,len = 50)))

# 5-cluster partition
Clustering2$hierarchy$Aleph_16

Clustering3 =  Icluster(data = sample,
                        est_method = list("phi", "mi", "Gauss", omegas = 1),
                        max_dim = 2)

# 5-cluster partition
Clustering3$hierarchy$Aleph_16
```

```
Clustering4 =  Icluster(data = sample,
                        est_method = list("phi", "Hel", "Gauss", omegas = 1))

# 5-cluster partition
Clustering4$hierarchy$Aleph_16

Clustering5 =  Icluster(data = sample,
                        est_method = list("ot", coef = 1, omegas = 1))

# 5-cluster partition
Clustering5$hierarchy$Aleph_16

Clustering6 =  Icluster(data = sample,
                        est_method = list("ot", coef = 2, omegas = 1))

# 5-cluster partition
Clustering6$hierarchy$Aleph_16

Clustering7 =  Icluster(data = sample,
                        est_method = list("ot", coef = 2, omegas = 1), max_dim = 4)

# 5-cluster partition
Clustering7$hierarchy$Aleph_16

# Parametric hierarchical Archimedean copula approach

Clustering8 = Icluster(data = sample,
                       est_method = list("phi", function(t){t * log(t)}, "hac",
                                         type = 1, M = 1000), max_dim = 4)

# 5-cluster partition
Clustering8$hierarchy$Aleph_16

Clustering9 = Icluster(data = sample,
                       est_method = list("phi", function(t){(sqrt(t)-1)^2}, "hac",
                                         type = 1, M = 1000), max_dim = 2)

# 5-cluster partition
Clustering9$hierarchy$Aleph_16

# Non-parametric approaches

Clustering10 = Icluster(data = sample,
                        est_method = list("phi", function(t){t * log(t)}, "nphac",
                                       estimator = "beta", type = 1), max_dim = 3)

# 5-cluster partition
Clustering10$hierarchy$Aleph_16

Clustering11 = Icluster(data = sample,
                        est_method = list("phi", function(t){t * log(t)}, "nphac",
                                     estimator = "trans", type = 1), max_dim = 3)
```

```
# 5-cluster partition
Clustering11$hierarchy$Aleph_16

Clustering12 = Icluster(data = sample,
                        est_method = list("phi", function(t){t * log(t)}, "np",
                                     estimator = "beta", bw_method = 1), max_dim = 3)

# 5-cluster partition
Clustering12$hierarchy$Aleph_16

Clustering13 = Icluster(data = sample,
                        est_method = list("phi", function(t){t * log(t)}, "np",
                                     estimator = "trans", bw_method = 2), max_dim = 3)

# 5-cluster partition
Clustering13$hierarchy$Aleph_16

Clustering14 = Icluster(data = sample,
                        est_method = list("phi", function(t){(sqrt(t)-1)^2}, "np",
                                      estimator = "trans", bw_method = 1), max_dim = 2)

# 5-cluster partition
Clustering14$hierarchy$Aleph_16

# Semi-parametric meta-elliptical copula approach
# Uncomment to run (takes a long time)

# Clustering15 = Icluster(data = sample,
#                         est_method = list("phi", function(t){t * log(t)}, "ellip",
#                                      grid = seq(0.005,100,by = 0.005)), max_dim = 2)

# 5-cluster partition
# Clustering15$hierarchy$Aleph_16
```

install_tensorflow *install_tensorflow*

Description

This function installs a python virtual environment.

Usage

```
install_tensorflow(envname = "r-tensorflow")
```

Arguments

envname	Name of the environment.

Value

No return value, used for creating a python virtual environment.

`minormal` *minormal*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the correlation-based mutual information between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$.

Usage

```
minormal(R, dim)
```

Arguments

`R`	The correlation matrix of $\mathbf{X}$.
`dim`	The vector of dimensions $(d_1, ..., d_k)$.

Details

Given a correlation matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1k} \\ \mathbf{R}_{12}^{\mathsf{T}} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1k}^{\mathsf{T}} & \mathbf{R}_{2k}^{\mathsf{T}} & \cdots & \mathbf{R}_{kk} \end{pmatrix},$$

the mutual information equals

$$\mathcal{D}_{t\ln(t)}^{\mathcal{N}}(\mathbf{R}) = -\frac{1}{2}\ln\left(\frac{|\mathbf{R}|}{\prod_{i=1}^{k}|\mathbf{R}_{ii}|}\right).$$

The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The correlation-based mutual information between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

See Also

`Helnormal` for the computation of the Gaussian copula Hellinger distance, `minormalavar` for the computation of the asymptotic variance of the plug-in estimator for the Gaussian copula mutual information, `miStudent` for the computation of the Student-t mutual information.

Examples

```
q = 10
dim = c(1,2,3,4)
# AR(1) correlation matrix with correlation 0.5

R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

minormal(R,dim)
```

---

minormalavar *minormalavar*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the asymptotic variance of the plug-in estimator for the correlation-based mutual information between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$.

Usage

```
minormalavar(R, dim)
```

Arguments

`R`	The correlation matrix of $\mathbf{X}$.
`dim`	The vector of dimensions $(d_1, ..., d_k)$.

Details

The asymptotic variance of the plug-in estimator $\mathcal{D}_{t\ln(t)}(\widehat{\mathbf{R}}_n)$ is computed at $\mathbf{R}$, where $\widehat{\mathbf{R}}_n$ is the sample matrix of normal scores rank correlations. The underlying assumption is that the copula of $\mathbf{X}$ is Gaussian.

Value

The asymptotic variance of the correlation-based mutual information between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

See Also

`minormal` for the computation of the mutual information, `Helnormal` for the computation of the Hellinger distance, `Helnormalavar` for the computation of the asymptotic variance of the plug-in estimator for the Hellinger distance, `estR` for the computation of the sample matrix of normal scores rank correlations.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

minormalavar(R,dim)
```

---

miStudent *miStudent*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function computes the Student-t mutual information between $\mathbf{X}_1, ..., \mathbf{X}_k$ given the entire correlation matrix $\mathbf{R}$ and the degrees of freedom nu.

Usage

```
miStudent(R, dim, nu)
```

Arguments

R	The correlation matrix of $\mathbf{X}$.
dim	The vector of dimensions $(d_1, ..., d_k)$.
nu	The degrees of freedom.

Details

Given a correlation matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1k} \\ \mathbf{R}_{12}^{\mathrm{T}} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1k}^{\mathrm{T}} & \mathbf{R}_{2k}^{\mathrm{T}} & \cdots & \mathbf{R}_{kk} \end{pmatrix},$$

and a certain amount of degrees of freedom $\nu > 0$, the Student-t mutual information equals

$$\mathcal{D}_{t\ln(t)}^{\mathrm{S}}(\mathbf{R}, \nu) = -\frac{1}{2} \ln \left(\frac{|\mathbf{R}|}{\prod_{i=1}^{k} |\mathbf{R}_{ii}|} \right) + K(\nu),$$

where

$$K(\nu) = \ln\left(\frac{\Gamma((q+\nu)/2)\Gamma(\nu/2)^{k-1}}{\prod_{i=1}^{k}\Gamma((d_i+\nu)/2)}\right) + \sum_{i=1}^{k}\left[\frac{d_i+\nu}{2}\psi((d_i+\nu)/2)\right] - \frac{q+\nu}{2}\psi((q+\nu)/2) - \frac{\nu}{2}(k-1)\psi(\nu/2),$$

with Γ the gamma function and ψ the digamma function. The underlying assumption is that the copula of $\mathbf{X}$ is Student-t.

Value

The Student-t mutual information between $\mathbf{X}_1, ..., \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

See Also

minormal for the computation of the Gaussian copula mutual information.

Examples

```
q = 10
dim = c(1,2,3,4)

# AR(1) correlation matrix with correlation 0.5
R = 0.5^(abs(matrix(1:q-1,nrow = q, ncol = q, byrow = TRUE) - (1:q-1)))

# Degrees of freedom
nu = 7

miStudent(R,dim,nu)
```

---

mlehac *mlehac*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function performs maximum pseudo-likelihood estimation for the parameters of a hierarchical Archimedean copula with two nesting levels of a specific type, used for modelling the dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$.

Usage

```
mlehac(sample, dim, type, start_val = NULL)
```

Arguments

`sample`	A sample from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
`dim`	The vector of dimensions $(d_1, ..., d_k)$.
`type`	The type of Archimedean copula.
`start_val`	The starting values for the parameters $(\theta_0, \theta_1, ..., \theta_k)$ of the hierarchical Archimedean copula.

Details

Under the assumption that $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_k)$ has a hierarchical Archimedean copula with two nesting levels, i.e.,

$$C(\mathbf{u}) = C_0\left(C_1(\mathbf{u}_1), \ldots, C_k(\mathbf{u}_k)\right),$$

where $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_k) \in \mathbb{R}^q$ with $\mathbf{u}_i \in \mathbb{R}^{d_i}$ for $i = 1, \ldots, k$, and with θ_i the parameter of C_i for $i = 0, 1, \ldots, k$ (see the function `gethac`), this functions performs maximum pseudo-likelihood estimation for $\boldsymbol{\theta}_C = (\theta_0, \theta_1, \ldots, \theta_k)$. This means that for $\widehat{F}_{ij}(x_{ij}) = \frac{1}{n+1}\sum_{\ell=1}^{n} 1\left(X_{ij}^{(\ell)} \leq x_{ij}\right)$ the (rescaled) empirical cdf of X_{ij} based on a sample $X_{ij}^{(1)}, \ldots, X_{ij}^{(n)}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, d_i$ (recall that $\mathbf{X}_i = (X_{i1}, \ldots, X_{id_i})$), we look for

$$\widehat{\boldsymbol{\theta}}_{C,n}^{\text{NP}} = \text{arg max}_{\boldsymbol{\theta}_C} \sum_{\ell=1}^{n} \ln\left\{c\left(\widehat{F}_{11}\left(X_{11}^{(\ell)}\right), \ldots, \widehat{F}_{kd_k}\left(X_{kd_k}^{(\ell)}\right); \boldsymbol{\theta}_C\right)\right\},$$

where $c(\cdot; \boldsymbol{\theta}_C)$ is the copula density of the hierarchical Archimedean copula.

We assume that C_i belongs to the same family of Archimedean copulas (e.g., Clayton) for $i = 0, \ldots, k$, and make use of the R package 'HAC'.

In case the starting values (start_val) are not specified, the starting value for θ_0 is put equal to 1.9 and the starting values for θ_i with $i \in \{1, \ldots, k\}$ are determined by performing maximum pseudo-likelihood estimation to the d_i-dimensional marginals with starting value 2.

Value

The maximum pseudo-likelihood estimates for $(\theta_0, \theta_1, \ldots, \theta_k)$.

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

Okhrin, O., Ristig, A. & Chen, G. (2024). HAC: estimation, simulation and visualization of hierarchical Archimedean copulae (HAC), R package version 1.1-1.
url: https://CRAN.R-project.org/package=HAC.

See Also

`gethac` for creating a hac object with two nesting levels, `phihac` for computing the Φ-dependence between all the child copulas of a hac object with two nesting levels, `Helhac` for computing the Hellinger distance between all the child copulas of a hac object with two nesting levels.

Examples

```
dim = c(2,2)
thetas = c(2,3,4)

# Sample size
n = 1000

# 4 dimensional nested Gumbel copula with (theta_0,theta_1,theta_2) = (2,3,4)
HAC = gethac(dim,thetas,type = 1)

# Sample
sample = suppressWarnings(HAC::rHAC(n,HAC))

# Maximum pseudo-likelihood estimator with starting values equal to thetas
HAC_est_1 = mlehac(sample,dim,1,thetas)

# Maximum pseudo-likelihood estimator with starting values
# theta_0 = 1.9, and theta_1, theta_2 determined by maximum
# pseudo-likelihood estimation for marginal child copulas

HAC_est_2 = mlehac(sample,dim,1)
```

---

otsort   *otsort*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function sorts the columns (variables) of a sample of $\mathbf{X}$ such that the dimensions are in ascending order.

Usage

```
otsort(sample, dim)
```

Arguments

`sample`	A sample from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
`dim`	The vector of dimensions $(d_1, ..., d_k)$, in order as given in sample.

Details

The columns of sample are rearranged such that the data corresponding to the random vector $\mathbf{X}_i$ having the smallest dimension d_i comes first, then the random vector with second smallest dimension, and so on.

Value

A list with elements "sample" containing the ordered sample, and "dim" containing the ordered dimensions.

Examples

```
q = 10
n = 50
dim = c(2,3,1,4)

# Sample from multivariate normal distribution
sample = mvtnorm::rmvnorm(n,rep(0,q),diag(q), method = "chol")

ordered = otsort(sample,dim)
```

---

phiellip *phiellip*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function estimates the Φ-dependence between $\mathbf{X}_1, ..., \mathbf{X}_k$ by estimating the joint and marginal meta-elliptical copula generators via the MECIP.

Usage

```
phiellip(sample, dim, phi, grid, params, normalize = 1)
```

Arguments

`sample`	A sample from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
`dim`	The vector of dimensions $(d_1, ..., d_k)$.
`phi`	The function Φ.
`grid`	The grid of values on which to estimate the density generators.
`params`	The tuning parameters to be used when estimating the density generators.
`normalize`	A value in $\{1, 2\}$ indicating the normalization procedure that is applied to the estimated generator (default = 1).

Details

When $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_k)$ has a meta-elliptical copula with generator $g_{\mathcal{R}}$, marginal generators $g_{\mathcal{R}_i}$ of $\mathbf{X}_i$ for $i = 1, \ldots, k$, and scale matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1k} \\ \mathbf{R}_{12}^{\mathrm{T}} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{1k}^{\mathrm{T}} & \mathbf{R}_{2k}^{\mathrm{T}} & \cdots & \mathbf{R}_{kk} \end{pmatrix},$$

the Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$ equals

$$\mathcal{D}_\Phi\left(\mathbf{X}_1, \ldots, \mathbf{X}_k\right) = \mathbb{E}\left\{\frac{\prod_{i=1}^k g_{\mathcal{R}_i}\left(\mathbf{Z}_i^{\mathrm{T}} \mathbf{R}_{ii}^{-1} \mathbf{Z}_i\right) |\mathbf{R}|^{1/2}}{g_{\mathcal{R}}\left(\mathbf{Z}^{\mathrm{T}} \mathbf{R}^{-1} \mathbf{Z}\right) \prod_{i=1}^k |\mathbf{R}_{ii}|^{1/2}} \Phi\left(\frac{g_{\mathcal{R}}\left(\mathbf{Z}^{\mathrm{T}} \mathbf{R}^{-1} \mathbf{Z}\right) \prod_{i=1}^k |\mathbf{R}_{ii}|^{1/2}}{\prod_{i=1}^k g_{\mathcal{R}_i}\left(\mathbf{Z}_i^{\mathrm{T}} \mathbf{R}_{ii}^{-1} \mathbf{Z}_i\right) |\mathbf{R}|^{1/2}}\right)\right\},$$

where (recall that $\mathbf{X}_i = (X_{i1}, \ldots, X_{id_i})$ for $i = 1, \ldots, k$)

$$\mathbf{Z}_i = (Z_{i1}, \ldots, Z_{id_i}) = \left(\left(Q \circ F_{i1}\right)\left(X_{i1}\right), \ldots, \left(Q \circ F_{id_i}\right)\left(X_{id_i}\right)\right),$$

and $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_k)$, with Q the quantile function corresponding to $g_{\mathcal{R}}$.

The expectation $\mathbb{E}$ is replaced by the empirical mean using the estimated sample $\widehat{\mathbf{Z}}^{(1)}, \ldots, \widehat{\mathbf{Z}}^{(n)}$ with $\widehat{\mathbf{Z}}^{(\ell)} = (\widehat{\mathbf{Z}}_1^{(\ell)}, \ldots, \widehat{\mathbf{Z}}_k^{(\ell)})$ for $\ell = 1, \ldots, n$, where

$$\widehat{\mathbf{Z}}_i^{(\ell)} = \left(\widehat{Z}_{i1}^{(\ell)}, \ldots, \widehat{Z}_{id_i}^{(\ell)}\right) = \left(\left(\widehat{Q} \circ \widehat{F}_{i1}\right)\left(X_{i1}^{(\ell)}\right), \ldots, \left(\widehat{Q} \circ \widehat{F}_{id_i}\right)\left(X_{id_i}^{(\ell)}\right)\right),$$

for $i = 1, \ldots, k$. Here, $\widehat{Q}$ will be the quantile function corresponding to the final estimator for $g_{\mathcal{R}}$, and

$$\widehat{F}_{ij}(x_{ij}) = \frac{1}{n+1} \sum_{\ell=1}^{n} 1\left(X_{ij}^{(\ell)} \leq x_{ij}\right)$$

is the (rescaled) empirical cdf of X_{ij} based on a sample $X_{ij}^{(1)}, \ldots, X_{ij}^{(n)}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, d_i$.

The estimation of $\mathbf{R}$ is done via its relation with the Kendall's tau matrix, see the function "KTMatrixEst.R" in the R package 'ElliptCopulas' of Derumigny et al. (2024).

For estimating $g_{\mathcal{R}}$ and $g_{\mathcal{R}_i}$ for $i = 1, \ldots, k$, the function `ellcopest` is used. This function requires certain tuning parameters (a bandwidth h, a parameter a, and a parameter δ for the shrinkage function). Suppose that there are m marginal random vectors (among $\mathbf{X}_1, \ldots, \mathbf{X}_k$) that are of dimension strictly larger than one. Then, all tuning parameters should be given as

$$\text{params} = \text{list}(\text{"h"} = (h, h_1, \ldots, h_m), \text{"a"} = (a, a_1, \ldots, a_m), \text{"p"} = (\delta, \delta_1, \ldots, \delta_m)),$$

i.e., (h, a, δ) will be used for estimating $g_{\mathcal{R}}$, and (h_i, a_i, δ_i) will be used for estimating $g_{\mathcal{R}_i}$ for $i = 1, \ldots, k$.

When $d_i = 1$ for a certain $i \in \{1, \ldots, k\}$, the function "Convert_gd_To_g1.R" from the R package 'ElliptCopulas' is used to estimate $g_{\mathcal{R}_i}$.

In order to make $g_{\mathcal{R}}$ identifiable, an extra normalization procedure is implemented in line with an extra constraint on $g_{\mathcal{R}}$. When normalize = 1, this corresponds to $\mathbf{R}$ being the correlation matrix of $\mathbf{Z}$. When normalize = 2, this corresponds to the identifiability condition of Derumigny & Fermanian (2022).

Value

The estimated Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$.

References

Derumigny, A., Fermanian, J.-D., Ryan, V., van der Spek, R. (2024). ElliptCopulas, R package version 0.1.4.1. url: https://CRAN.R-project.org/package=ElliptCopulas.

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

See Also

elldistrest for improved kernel estimation of the elliptical generator of an elliptical distribution, ellcopest for improved kernel estimation of the elliptical generator of a meta-elliptical copula, elliptselect for selecting optimal tuning parameters for the improved kernel estimator of the elliptical generator.

Examples

```
q = 4
dim = c(2,2)

# Sample size
n = 1000

# Grid on which to evaluate the elliptical generator
grid = seq(0.005,100,by = 0.005)

# Degrees of freedom
nu = 7

# Student-t generator with 7 degrees of freedom
g_q = ((nu/(nu-2))^(q/2))*(gamma((q+nu)/2)/(((pi*nu)^(q/2))*gamma(nu/2))) *
                          ((1+(grid/(nu-2)))^(-(q+nu)/2))

# Density of squared radius
R2 = function(t,q){(gamma((q+nu)/2)/(((nu-2)^(q/2))*gamma(nu/2)*gamma(q/2))) *
                   (t^((q/2)-1)) * ((1+(t/(nu-2)))^(-(q+nu)/2))}

# Sample from 4-dimensional Student-t distribution with 7 degrees of freedom
# and identity covariance matrix
sample = EllipCopulas::EllDistrSim(n,q,diag(q),density_R2 = function(t){R2(t,q)})

# Tuning parameter selection for g_R
opt_parameters_joint = elliptselect(n,q,seq((3/4)-(1/q)+0.01,1-0.01,len = 200),
                                        seq(0.01,2,len = 200))

# Optimal tuning parameters for g_R
a = opt_parameters_joint$Opta ; p = opt_parameters_joint$Optp ;
                                h = opt_parameters_joint$Opth

# Tuning parameter selection for g_R_1 (same for g_R_2)
opt_parameters_marg = elliptselect(n,2,seq((3/4)-(1/2)+0.01,1-0.01,len = 200),
                                       seq(0.01,2,len = 200))

# Optimal tuning parameters for g_R_1 (same for g_R_2)
a1 = opt_parameters_marg$Opta ; p1 = opt_parameters_marg$Optp ;
                                h1 = opt_parameters_marg$Opth

a2 = a1 ; p2 = p1 ; h2 = h1
params = list("h" = c(h,h1,h2), "a" = c(a,a1,a2), "p" = c(p,p1,p2))

# Mutual information between two random vectors of size 2
```

```
est_phi = phiellip(sample, dim, function(t){t * log(t)}, grid, params)
```

---

phihac *phihac*

Description

This function computes the Φ-dependence between all the child copulas of a hac object obtained by the function gethac, i.e., given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, where $\mathbf{X}_1, ..., \mathbf{X}_k$ are connected via a hierarchical Archimedean copula with two nesting levels, phihac computes the Φ-dependence between $\mathbf{X}_1, ..., \mathbf{X}_k$.

Usage

```
phihac(cop, dim, M, phi)
```

Arguments

cop	A hac object as provided by the function gethac.
dim	The vector of dimensions $(d_1, ..., d_k)$.
M	The size of the Monte Carlo sample used for approximating the integral of the Φ-dependence.
phi	The function Φ.

Details

When $\mathbf{X}$ has copula density c with marginal copula densities c_i of $\mathbf{X}_i$ for $i = 1, \ldots, k$, the Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$ equals

$$\mathcal{D}_\Phi\left(\mathbf{X}_1, \ldots, \mathbf{X}_k\right) = \int_{[0,1]^q} \prod_{i=1}^{k} c_i(\mathbf{u}_i) \Phi\left(\frac{c(\mathbf{u})}{\prod_{i=1}^{k} c_i(\mathbf{u}_i)}\right),$$

for a certain continuous, convex function $\Phi : (0, \infty) \to \mathbb{R}$. By drawing a sample of size M from c, say $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(M)}$, with $\mathbf{U}^{(\ell)} = (\mathbf{U}_1^{(\ell)}, \ldots, \mathbf{U}_k^{(\ell)})$, we can approximate $\mathcal{D}_\Phi$ by

$$\widehat{\mathcal{D}}_\Phi = \frac{1}{M} \sum_{\ell=1}^{M} \frac{\prod_{i=1}^{k} c_i\left(\mathbf{U}_i^{(\ell)}\right)}{c\left(\mathbf{U}^{(\ell)}\right)} \Phi\left(\frac{c\left(\mathbf{U}^{(\ell)}\right)}{\prod_{i=1}^{k} c_i\left(\mathbf{U}_i^{(\ell)}\right)}\right).$$

The function phihac computes $\widehat{\mathcal{D}}_\Phi$ when c is a hierarchical Archimedean copula with two nesting levels, as produced by the function gethac.

Value

The Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$ (i.e., between all the child copulas of the hac object).

References

De Keyser, S. & Gijbels, I. (2024). Parametric dependence between random vectors via copula-based divergence measures. Journal of Multivariate Analysis 203:105336.
doi: https://doi.org/10.1016/j.jmva.2024.105336.

See Also

gethac for creating a hac object with two nesting levels, Helhac for computing the Hellinger distance between all the child copulas of a hac object with two nesting levels, mlehac for maximum pseudo-likelihood estimation of the parameters of a hac object with two nesting levels.

Examples

```
dim = c(2,2)
thetas = c(2,3,4)

# 4 dimensional nested Gumbel copula with (theta_0,theta_1,theta_2) = (2,3,4)
HAC = gethac(dim,thetas,type = 1)

# Mutual information based on Monte Carlo sample of size 10000
Phi_dep = phihac(HAC,dim,10000,function(t){t * log(t)})
```

---

phinp *phinp*

Description

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_k)$ with $\mathbf{X}_i$ a d_i-dimensional random vector, i.e., $q = d_1 + ... + d_k$, this function estimates the Φ-dependence between $\mathbf{X}_1, ..., \mathbf{X}_k$ by estimating the joint and marginal copula densities via fully non-parametric copula kernel density estimation.

Usage

```
phinp(sample, cop = NULL, dim, phi, estimator, bw_method)
```

Arguments

sample	A sample from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).
cop	A fitted reference hac object, in case bw_method = 0 (default = NULL).
dim	The vector of dimensions $(d_1, ..., d_k)$.
phi	The function Φ.
estimator	Either "beta" or "trans" for the beta kernel or the Gaussian transformation kernel copula density estimator.
bw_method	A number in $\{0, 1, 2\}$ specifying the method used for computing optimal local bandwidths.

Details

When $\mathbf{X}$ has copula density c with marginal copula densities c_i of $\mathbf{X}_i$ for $i = 1, \ldots, k$, the Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$ equals

$$\mathcal{D}_\Phi\left(\mathbf{X}_1, \ldots, \mathbf{X}_k\right) = \mathbb{E}\left\{\frac{\prod_{i=1}^k c_i(\mathbf{U}_i)}{c\left(\mathbf{U}\right)} \Phi\left(\frac{c(\mathbf{U})}{\prod_{i=1}^k c_i(\mathbf{U}_i)}\right)\right\},$$

for a certain continuous, convex function $\Phi : (0, \infty) \to \mathbb{R}$, and with $\mathbf{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_k) \sim c$.

The expectation $\mathbb{E}$ is replaced by the empirical mean using the estimated copula sample $\widehat{\mathbf{U}}^{(1)}, \ldots, \widehat{\mathbf{U}}^{(n)}$ with $\widehat{\mathbf{U}}^{(\ell)} = (\widehat{\mathbf{U}}_1^{(\ell)}, \ldots, \widehat{\mathbf{U}}_k^{(\ell)})$ for $\ell = 1, \ldots, n$, where (recall that $\mathbf{X}_i = (X_{i1}, \ldots, X_{id_i})$ for $i = 1, \ldots, k$)

$$\widehat{\mathbf{U}}_i^{(\ell)} = \left(\widehat{U}_{i1}^{(\ell)}, \ldots, \widehat{U}_{id_i}^{(\ell)}\right) = \left(\widehat{F}_{i1}\left(X_{i1}^{(\ell)}\right), \ldots, \widehat{F}_{id_i}\left(X_{id_i}^{(\ell)}\right)\right).$$

Hereby, $\widehat{F}_{ij}(x_{ij}) = \frac{1}{n+1}\sum_{\ell=1}^n 1\left(X_{ij}^{(\ell)} \leq x_{ij}\right)$ is the (rescaled) empirical cdf of X_{ij} based on a sample $X_{ij}^{(1)}, \ldots, X_{ij}^{(n)}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, d_i$.

The joint copula density c and marginal copula densities c_i for $i = 1, \ldots, k$ are estimated via fully non-parametric copula kernel density estimation. When estimator = "beta", the beta kernel copula density estimator is used. When estimator = "trans", the Gaussian transformation kernel copula density estimator is used.

Bandwidth selection is done locally by using the function `hamse`. When bw_method = 0, then the given fitted (e.g., via MLE using `mlehac`) hac object (hierarchical Archimedean copula) cop is used as reference copula. When bw_method = 1, then a non-parametric (beta or Gaussian transformation) kernel copula density estimator based on the pseudos as pivot is used. This pivot is computed using the big O bandwidth (i.e., $n^{-2/(q+4)}$ in case of the beta estimator, and $n^{-1/(q+4)}$ for the transformation estimator, with q the total dimension). When bw_method = 2, the big O bandwidths are taken.

Value

The estimated Φ-dependence between $\mathbf{X}_1, \ldots, \mathbf{X}_k$.

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

See Also

`betakernelestimator` for the computation of the beta kernel copula density estimator, `transformationestimator` for the computation of the Gaussian transformation kernel copula density estimator, `hamse` for local bandwidth selection for the beta kernel or Gaussian transformation kernel copula density estimator.

Examples

```
q = 4
dim = c(2,2)

# Sample size
n = 500

# Four dimensional hierarchical Gumbel copula
# with parameters (theta_0,theta_1,theta_2) = (2,3,4)
HAC = gethac(dim,c(2,3,4),type = 1)

# Sample
sample =  suppressWarnings(HAC::rHAC(n,HAC))

# Maximum pseudo-likelihood estimator to be used as reference copula for bw_method = 0
est_cop = mlehac(sample,dim,1,c(2,3,4))

# Estimate mutual information between two random vectors of size 2 in different ways

est_phi_1 = phinp(sample,cop = est_cop,dim = dim,phi = function(t){t * log(t)},
                  estimator = "beta",bw_method = 0)
est_phi_2 = phinp(sample,cop = est_cop,dim = dim,phi = function(t){t * log(t)},
                  estimator = "trans",bw_method = 0)
est_phi_3 = phinp(sample,dim = dim,phi = function(t){t * log(t)},
                  estimator = "beta",bw_method = 1)
est_phi_4 = phinp(sample,dim = dim,phi = function(t){t * log(t)},
                  estimator = "trans",bw_method = 1)
est_phi_5 = phinp(sample,dim = dim,phi = function(t){t * log(t)},
                  estimator = "beta",bw_method = 2)
est_phi_6 = phinp(sample,dim = dim,phi = function(t){t * log(t)},
                  estimator = "trans",bw_method = 2)
```

---

transformationestimator

transformationestimator

Description

This function computes the non-parametric Gaussian transformation kernel copula density estimator.

Usage

```
transformationestimator(input, h, pseudos)
```

Arguments

- `input` The copula argument at which the density estimate is to be computed.
- `h` The bandwidth to be used in the Gaussian kernel.
- `pseudos` The (estimated) copula observations from a q-dimensional random vector $\mathbf{X}$ ($n \times q$ matrix with observations in rows, variables in columns).

Details

Given a q-dimensional random vector $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_k)$ with $\mathbf{X}_i = (X_{i1}, \dots, X_{id_i})$, and samples $X_{ij}^{(1)}, \dots, X_{ij}^{(n)}$ from X_{ij} for $i = 1, \dots, k$ and $j = 1, \dots, d_i$, the Gaussian transformation kernel estimator for the copula density of $\mathbf{X}$ equals, at $\mathbf{u} = (u_{11}, \dots, u_{kd_k}) \in \mathbb{R}^q$,

$$\widehat{c}_{\mathrm{T}}(\mathbf{u}) = \frac{1}{nh_n^q \prod_{i=1}^k \prod_{j=1}^{d_i} \phi\left(\Phi^{-1}\left(u_{ij}\right)\right)} \sum_{\ell=1}^n \prod_{i=1}^k \prod_{j=1}^{d_i} \phi\left(\frac{\Phi^{-1}(u_{ij}) - \Phi^{-1}\left(\widehat{U}_{ij}^{(\ell)}\right)}{h_n}\right),$$

where $h_n > 0$ is a bandwidth parameter, $\widehat{U}_{ij}^{(\ell)} = \widehat{F}_{ij}(X_{ij}^{(\ell)})$ with

$$\widehat{F}_{ij}(x_{ij}) = \frac{1}{n+1} \sum_{\ell=1}^n 1\left(X_{ij}^{(\ell)} \leq x_{ij}\right)$$

the (rescaled) empirical cdf of X_{ij}, and Φ the standard normal distribution function with corresponding quantile function Φ^{-1} and density function ϕ.

Value

The Gaussian transformation kernel copula density estimator evaluated at the input.

References

De Keyser, S. & Gijbels, I. (2024). Hierarchical variable clustering via copula-based divergence measures between random vectors. International Journal of Approximate Reasoning 165:109090. doi: https://doi.org/10.1016/j.ijar.2023.109090.

See Also

`betakernelestimator` for the computation of the beta kernel copula density estimator, `hamse` for local bandwidth selection for the beta kernel or Gaussian transformation kernel copula density estimator, `phinp` for fully non-parametric estimation of the Φ-dependence between k random vectors.

Examples

```
q = 3
n = 100

# Sample from multivariate normal distribution with identity covariance matrix
sample = mvtnorm::rmvnorm(n,rep(0,q),diag(3),method = "chol")

# Copula pseudo-observations
```

```
pseudos = matrix(0,n,q)
for(j in 1:q){pseudos[,j] = (n/(n+1)) * ecdf(sample[,j])(sample[,j])}

# Argument at which to estimate the density
input = rep(0.5,q)

# Local bandwidth selection
h = hamse(input,pseudos = pseudos,n = n,estimator = "trans",bw_method = 1)

# Gaussian transformation kernel estimator
est_dens = transformationestimator(input,h,pseudos)

# True density
true = copula::dCopula(rep(0.5,q), copula::normalCopula(0, dim = q))
```

Index