Package 'gicf'

May 16, 2025

Type Package

Title Penalised Likelihood Estimation of a Covariance Matrix

Description Penalised likelihood estimation of a covariance matrix via the ridge-regularised cov-glasso estimator described in Cibinel et al. (2024) <doi:10.48550/arXiv.2410.02403>.

Version 1.0

Date 2025-04-16

Author Luca Cibinel [cre, aut],
Alberto Roverato [aut] (ORCID: <https://orcid.org/0000-0001-7984-3593>),
Veronica Vinciotti [aut] (ORCID:
<https://orcid.org/0000-0002-2625-7977>),
Michael Fop [ctb] (ORCID: <https://orcid.org/0000-0003-3936-2757>),
Mathias Drton [ctb] (ORCID: <https://orcid.org/0000-0001-5614-3025>)

Maintainer Luca Cibinel <lcibinel@gmail.com>

License GPL-3

Imports Rcpp (>= 1.0.12), stats

Suggests mvtnorm

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-05-16 09:00:02 UTC

Contents

gicf-package *Penalised Likelihood Estimation of a Covariance Matrix*

Description

Penalised likelihood estimation of a covariance matrix via the ridge-regularised covglasso estimator described in Cibinel et al. (2024) <doi:10.48550/arXiv.2410.02403>.

Details

This package optimises the penalised loglikelihood function of a Gaussian covariance graphical model via an iterative coordinate descent algorithm.

Author(s)

Luca Cibinel [cre, aut], Alberto Roverato [aut] (<https://orcid.org/0000-0001-7984-3593>), Veronica Vinciotti [aut] (<https://orcid.org/0000-0002-2625-7977>), Michael Fop [ctb] (<https://orcid.org/0000-0003-3936-2757>), Mathias Drton [ctb] (<https://orcid.org/0000-0001-5614-3025>)

Maintainer: Luca Cibinel <lcibinel@gmail.com>

References

Cibinel, L., A. Roverato, and V. Vinciotti (2024). A unified approach to penalized likelihood estimation of covariance matrices in high dimensions. arXiv, arXiv:2410.02403.

gcgmloglik *Gaussian Covariance Graphical Model Loglikelihood function*

Description

Computes the penalised loglikelihood function of a Gaussian covariance graph model.

Usage

```
gcgmloglik(Sigma, S, n, lambda = 0, kappa = 0)
```

Arguments

`Sigma`	The covariance matrix.
`S`	The sample covariance matrix.
`n`	The size of the observed dataset.
`lambda`	A non-negative lasso parameter.
`kappa`	A non-negative ridge regularisation parameter.

Details

When imposing sparsity on the covariance matrix of a multivariate Gaussian distribution, the resulting model can be interpreted as a covariance graphical model, i.e., the independence structure of the components of the random vector can be encoded by a graph in which the nodes are identified with the variables and a missing edge between two nodes implies that the corresponding variables are marginally independent.

In particular, this model admits both a ridge and a lasso penalty, resulting in the loglikelihood function

$$-\log|\Sigma| - \text{trace}(\Sigma^{-1}S) - \lambda\|\Sigma - \text{diag}(\Sigma)\|_1 - \kappa\|\Sigma^{-1}\|_1,$$

where $\lambda, \kappa \geq 0$.

Value

The value of the penalised loglikelihood function.

Examples

```
# An example with a banded covariance matrix
library(mvtnorm)

set.seed(1234)

p <- 10
n <- 500

# Create banded covariance matrix with three bands
band1 <- cbind(1:(p - 1), 2:p)
band2 <- cbind(1:(p - 2), 3:p)
band3 <- cbind(1:(p - 3), 4:p)
idxs <- rbind(band1, band2, band3)

Sigma <- matrix(0, p, p)
Sigma[idxs] <- 0.5
Sigma <- Sigma + t(Sigma)
diag(Sigma) <- 2

# Generate data
data <- rmvnorm(n, sigma = Sigma)
S <- cov(data) * (n - 1)/n

# Fix a value of lambda and kappa
lambda <- 0.07
kappa <- 0.5

# Gaussian loglikelihood
print("Gaussian loglikelihood:")
print(gcgmloglik(Sigma, S, n))

# Penalised Gaussian loglikelihood
print("Penalised Gaussian loglikelihood:")
print(gcgmloglik(Sigma, S, n, lambda, kappa))
```

`gicf` *Penalised maximum likelihood covariance matrix estimation*

Description

Estimation of a sparse covariance matrix via the ridge-regularised covglasso estimator described in Cibinel et al. (2024).

Usage

```
gicf(
  data = NULL,
  S = NULL,
  n = NULL,
  lambda = 0,
  kappa = 0,
  max.iter = 2500,
  tol = 1e-04,
  Sigma.init = NULL,
  adj = NULL
)
```

Arguments

`data`	A numerical matrix whose rows contain the observations of multivariate normal random vector. If `NULL`, the sample covariance matrix S and the dataset size n must be provided.
`S`	The sample covariance matrix. Must be provided if data is `NULL`.
`n`	The dataset size. Must be provided if data is `NULL`.
`lambda`	A vector of non-negative lasso parameters. For efficency purposes, should be sorted from largest to smallest.
`kappa`	A non-negative ridge regularisation parameter.
`max.iter`	The maximum number of iterations allowed for the coordinate descent algorithm.
`tol`	A numerical tolerance below which quantities are treated as zero.
`Sigma.init`	The initial guess for the coordinate descent algorithm. Defaults to the diagonal of the sample covariance matrix.
`adj`	An optional matrix whose pattern of zeroes is enforced onto the final output of the algorithm.

Details

This function computes the ridge-regularised covglasso estimator of the covariance matrix of a multivariate normal distribution, that is it computes the maximum of the penalised log-likelihood

$$-\log|\Sigma| - \text{trace}(\Sigma^{-1}S) - \lambda\|\Sigma - \text{diag}(\Sigma)\|_1 - \kappa\|\Sigma^{-1}\|_1,$$

where $\lambda, \kappa \geq 0$. The optimum is computed via a coordinate descent algorithm, resulting in an approach which unifies and extends the methods of Chaudhuri et. al (2007), Warton (2008), Bien and Tibshirani (2011) and Wang (2014).

Value

If a scalar value for `lambda` is provided, a list containing the following elements.

`sigma`	The estimate of the covariance matrix.
`omega`	The inverse of the estimated covariance matrix.
`loglik`	The (unpenalised) log-likelihood at the optimum.
`loglikpen`	The (penalised) log-likelihood at the optimum.
`it`	The number of iterations needed to reach convergence.

If a vector of values of `lambda` is provided, the output is a list in which each entry is itself a list, structured as above, associated with the corresponding value of `lambda`.

References

Chaudhuri, S., M. Drton, and T. S. Richardson (2007). Estimation of a covariance matrix with zeros. Biometrika 94 (1), 199–216.

Cibinel, L., A. Roverato, and V. Vinciotti (2024). A unified approach to penalized likelihood estimation of covariance matrices in high dimensions. arXiv, arXiv:2410.02403.

Bien, J. and R. J. Tibshirani (2011). Sparse estimation of a covariance matrix. Biometrika 98 (4), 807–820.

Wang, H. (2014). Coordinate descent algorithm for covariance graphical lasso. Statistics and Computing 24, 521–529.

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.

Examples

```
# An example with a banded covariance matrix
library(mvtnorm)

set.seed(1234)

p <- 10
n <- 500

# Create banded covariance matrix with three bands
band1 <- cbind(1:(p - 1), 2:p)
band2 <- cbind(1:(p - 2), 3:p)
```

```
band3 <- cbind(1:(p - 3), 4:p)
idxs <- rbind(band1, band2, band3)

Sigma <- matrix(0, p, p)
Sigma[idxs] <- 0.5
Sigma <- Sigma + t(Sigma)
diag(Sigma) <- 2

# Generate data
data <- rmvnorm(n, sigma = Sigma)

# Fit a path of estimates
lambdas <- seq(0, 0.15, 0.01)
fit <- gicf(data, lambda = lambdas, kappa = 0.1)

# Explore one particular estimate
onefit <- fit[[5]]
image(onefit$sigma != 0)

# Redo the fit, but this time fix the correct sparsity pattern
fit2 <- gicf(data, lambda = lambdas, kappa = 0.1, adj = Sigma)

onefit2 <- fit2[[5]]
image(onefit2$sigma != 0)
```

`Hyperparameters` *Maximum effective range of regularisation parameters*

Description

Compute the effective range of the regularisation paramerters κ and λ.

Usage

```
lambdamax(S, kappa = 0, adj = 1 - diag(1, nrow(S)))

kappamax(S, lambda, adj = 1 - diag(1, nrow(S)))
```

Arguments

`S`	The sample covariance matrix.
`kappa, lambda`	The non-negative ridge regularisation/lasso shrinkage parameters.
`adj`	An optional matrix whose pattern of zeroes is to be enforced onto the final output of the Generalised Iterative Conditional Fitting algorithm.

Details

These utility functions describe the boundary of the region

$$\mathcal{H} = \{(\kappa, \lambda) \in \mathbb{R}^2_{\geq 0} : \lambda \leq \lambda_{MAX}(\kappa)\},$$

with

$$\lambda \leq \lambda_{MAX}(\kappa) \iff \kappa \leq \kappa_{MAX}(\lambda),$$

$$\lambda_{MAX}(\kappa) = \max_{i,j \in \mathcal{G}} \frac{|s_{ij}|}{(s_{ii} + \kappa)(s_{jj} + \kappa)},$$

$$\kappa_{MAX}(\lambda) = \max_{\substack{i,j \in \mathcal{G} \\ g_{ij}(\lambda) \geq 0}} \left\{ \sqrt{\frac{1}{4}(s_{ii} + s_{jj}) + g_{ij}(\lambda)} - \frac{1}{2}(s_{ii} + s_{jj}) \right\},$$

$$g_{ij}(\lambda) = \frac{|s_{ij}|}{\lambda} - s_{ii}s_{jj}.$$

Here S is the sample covariance matrix and $\mathcal{G}$ is a graph whose adjacency matrix has the same sparsity pattern as `adj`. If the parameters (κ, λ) lay outside of $\mathcal{H}$, and the starting point of the Generalised Iterative Conditional Fitting algorithm is $\text{diag}(S + \kappa I)$, then the output will also be $\text{diag}(S + \kappa I)$.

Value

`lambdamax` returns a scalar value representing $\lambda_{MAX}(\kappa)$. `kappamax` returns a scalar value representing $\kappa_{MAX}(\lambda)$

Examples

```
# An example with a banded covariance matrix
library(mvtnorm)

set.seed(1234)

p <- 10
n <- 500

# Create banded covariance matrix with three bands
band1 <- cbind(1:(p - 1), 2:p)
band2 <- cbind(1:(p - 2), 3:p)
band3 <- cbind(1:(p - 3), 4:p)
idxs <- rbind(band1, band2, band3)

Sigma <- matrix(0, p, p)
Sigma[idxs] <- 0.5
Sigma <- Sigma + t(Sigma)
diag(Sigma) <- 2

# Generate data
data <- rmvnorm(n, sigma = Sigma)
S <- cov(data) * (n - 1)/n
```

```
# Fix a value of lambda and compute k_max
lambda <- 0.07
k.max <- kappamax(S, lambda = lambda)

# Check that fit is diagonal
fit <- gicf(S = S, n = n, lambda = lambda, kappa = k.max)
image(fit$sigma != 0)


# Fix a value of kappa and compute l_max
kappa <- 1.15
l.max <- lambdamax(S, kappa = kappa)

# Check that fit is diagonal
fit <- gicf(S = S, n = n, lambda = l.max, kappa = kappa)
image(fit$sigma != 0)

# Repeat steps above, but with correct sparsity pattern specified
lambda <- 0.07
k.max <- kappamax(S, lambda = lambda, adj = Sigma)
fit <- gicf(S = S, n = n, lambda = lambda, kappa = k.max, adj = Sigma)
image(fit$sigma != 0)

kappa <- 1.15
l.max <- lambdamax(S, kappa = kappa, adj = Sigma)
fit <- gicf(S = S, n = n, lambda = l.max, kappa = kappa, adj = Sigma)
image(fit$sigma != 0)
```

Index