Package 'clusterGGM'

October 17, 2025

Type Package

Title Sparse Gaussian Graphical Modeling with Variable Clustering

Version 0.1.0

Date 2025-10-13

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.9), dplyr, parallel, rlang, stats

Suggests mvtnorm

LinkingTo Rcpp, RcppEigen

Description Perform sparse estimation of a Gaussian graphical model (GGM) with node aggregation through variable clustering. Currently, the package implements the clusterpath estimator of the Gaussian graphical model (CGGM) (Touw, Alfons, Groenen & Wilms, 2025; <doi:10.48550/arXiv.2407.00644>).

License GPL (>= 3)

URL https://github.com/aalfons/clusterGGM

BugReports https://github.com/aalfons/clusterGGM/issues

Author Daniel J.W. Touw [aut] (ORCID: <https://orcid.org/0000-0003-3074-5401>), Andreas Alfons [aut, cre] (ORCID: <https://orcid.org/0000-0002-2513-3788>), Ines Wilms [aut] (ORCID: <https://orcid.org/0000-0003-3269-4601>), Patrick J.F. Groenen [ths] (ORCID: <https://orcid.org/0000-0001-6683-8971>, PhD advisor of Daniel J.W. Touw)

Maintainer Andreas Alfons <alfons@ese.eur.nl>

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-10-17 20:20:13 UTC

Contents

`clusterGGM-package` *Sparse Gaussian Graphical Modeling with Variable Clustering*

Description

Perform sparse estimation of a Gaussian graphical model (GGM) with node aggregation through variable clustering. Currently, the package implements the clusterpath estimator of the Gaussian graphical model (CGGM) (Touw, Alfons, Groenen & Wilms, 2025; <doi:10.48550/arXiv.2407.00644>).

Details

The DESCRIPTION file:

Package:	clusterGGM
Type:	Package
Title:	Sparse Gaussian Graphical Modeling with Variable Clustering
Version:	0.1.0
Date:	2025-10-13
Depends:	R (>= 3.5.0)
Imports:	Rcpp (>= 1.0.9), dplyr, parallel, rlang, stats
Suggests:	mvtnorm
LinkingTo:	Rcpp, RcppEigen
Description:	Perform sparse estimation of a Gaussian graphical model (GGM) with node aggregation through variable clu
License:	GPL (>= 3)
URL:	https://github.com/aalfons/clusterGGM
BugReports:	https://github.com/aalfons/clusterGGM/issues
Authors@R:	c(person("Daniel J.W.", "Touw", role = "aut", comment = c(ORCID = "0000-0003-3074-5401")), person("An
Author:	Daniel J.W. Touw [aut] (ORCID: <https://orcid.org/0000-0003-3074-5401>), Andreas Alfons [aut, cre] (ORC
Maintainer:	Andreas Alfons <alfons@ese.eur.nl>
Encoding:	UTF-8
RoxygenNote:	7.3.3

Index of help topics:

cggm	Clusterpath Estimator of the Gaussian Graphical Model
cggm_cv	Cross Validation for the Clusterpath Estimator of the Gaussian Graphical Model
cggm_refit	Refit the Gaussian Graphical Model for a Given Aggregation and Sparsity Structure
clusterGGM-package	Sparse Gaussian Graphical Modeling with Variable Clustering
clusterpath_weights	Compute the Weight Matrix for the Clusterpath Penalty
cv_folds	Create Cross-Validation Folds
get_Theta	Extract the Estimated Precision Matrix
get_clusters	Extract the Cluster Assignment
lasso_weights	Compute the Weight Matrix for the Lasso Penalty
min_clusters	Calculate the Minimum Number of Clusters

Author(s)

Daniel J.W. Touw [aut] (ORCID: <https://orcid.org/0000-0003-3074-5401>), Andreas Alfons [aut, cre] (ORCID: <https://orcid.org/0000-0002-2513-3788>), Ines Wilms [aut] (ORCID: <https://orcid.org/0000-0003-3269-4601>), Patrick J.F. Groenen [ths] (ORCID: <https://orcid.org/0000-0001-6683-8971>, PhD advisor of Daniel J.W. Touw)

Maintainer: Andreas Alfons <alfons@ese.eur.nl>

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

Useful links:

- https://github.com/aalfons/clusterGGM
- Report bugs at https://github.com/aalfons/clusterGGM/issues

---

cggm *Clusterpath Estimator of the Gaussian Graphical Model*

Description

Compute the clusterpath estimator of the Gaussian Graphical Model (CGGM) for fixed values of the tuning parameters to obtain a sparse estimate with variable clustering of the precision matrix or the covariance matrix.

Usage

```
cggm(
  S,
  W_cpath,
  lambda_cpath,
  W_lasso = NULL,
  lambda_lasso = 0,
  eps_lasso = 0.005,
  gss_tol = 0.005,
  conv_tol = 1e-07,
  fusion_threshold = NULL,
  tau = 0.001,
  max_iter = 5000,
  expand = FALSE,
  max_difference = 0.01,
  verbose = 0
)
```

Arguments

S	The sample covariance matrix of the data.
W_cpath	The weight matrix used in the clusterpath penalty.
lambda_cpath	A numeric vector of tuning parameters for regularization. Should be a sequence of monotonically increasing values.
W_lasso	The weight matrix used in the lasso penalty. Defaults to NULL, which is interpreted as all weights being zero (no penalization).
lambda_lasso	The penalty parameter used for the lasso penalty. Defaults to 0 (no penalization).
eps_lasso	Parameter that governs the quadratic approximation of the lasso penalty. Within the interval c(-eps_lasso, eps_lasso) the absolute value function is approximated by a quadratic function. Defaults to 0.005.
gss_tol	The tolerance value used in the golden section search (GSS) algorithm. Defaults to 0.005.
conv_tol	The tolerance used to determine convergence. Defaults to 1e-7.
fusion_threshold	The threshold for fusing two clusters. If NULL, defaults to tau times the median distance between the rows of solve(S).
tau	The parameter used to determine the fusion threshold. Defaults to 0.001.
max_iter	The maximum number of iterations allowed for the optimization algorithm. Defaults to 5000.
expand	Determines whether the vector lambda should be expanded with additional values in order to find a sequence of solutions that (a) terminates in the minimum number of clusters and (b) has consecutive solutions for Theta that are not too different from each other. The degree of difference between consecutive solutions that is allowed is determined by max_difference. Defaults to FALSE.

`max_difference` The maximum allowed difference between consecutive solutions of Theta if `expand = TRUE`. The difference is computed as `norm(Theta[i-1]-Theta[i], "F") / norm(Theta[i-1], "F")`. Defaults to 0.01.

`verbose` Determines the amount of information printed during the optimization. Slows down the algorithm significantly. Defaults to 0.

Value

An object of class `"CGGM"` with the following components:

`A, R` Lists of matrices. Each pair of matrices with the same index parametrize the estimated precision matrix for the corresponding value of the aggregation parameter `lambda_cpath`. It is not recommended to use these directly, instead use the accessor function `get_Theta()` to extract the estimated precision matrix for a given index of the aggregation parameter.

`clusters` An integer matrix in which each row contains the cluster assignment of each variable for the corresponding value of the aggregation parameter `lambda_cpath`. Use the accessor function `get_clusters()` to extract the cluster assignment for a given index of the aggregation parameter.

`lambdas` A vector with the values for the aggregation parameter `lambda_cpath` for which the CGGM loss function has been minimized.

`Theta` List of matrices. Contains the solution to the minimization procedure for each value of the aggregation parameter `lambda_cpath`. It is not recommended to use these directly, instead use the accessor function `get_Theta()` to extract the estimated precision matrix for a given index of the aggregation parameter.

`losses` A vector with the values of the minimized CGGM loss function for each value of the aggregation parameter `lambda_cpath`.

`cluster_counts` An integer vector containing the number of clusters obtained for each value of the aggregation parameter `lambda_cpath`.

`loss_progression`
A list of vectors. Contains, for each value of the aggregation parameter `lambda_cpath`, the value of the loss function for each iteration of the minimization procedure. This is only part of the output if `expand = FALSE`.

`loss_timings` A list of vectors. Contains, for each value of the aggregation parameter `lambda_cpath`, the cumulative elapsed computation time in each iteration of the minimization procedure.

`fusion_threshold`
The threshold value used to determine whether two clusters should be clustered.

`cluster_solution_index`
An integer vector containing the index of the value of the aggregation parameter `lambda_cpath` for which a certain number of clusters was attained. For example, `cluster_solution_index[2]` yields the index of the smallest value for `lambda_cpath` for which a solution with two clusters was found. Contains -1 if there is no value for `lambda_cpath` with that number of clusters.

`n` The number of values of the aggregation parameter `lambda_cpath` for which the CGGM loss function was minimized.

`inputs` A list of the inputs of the function, used internally and in `cggm_refit()`. It consists of eight components:

- `S` (the sample covariance matrix)
- `W_cpath` (the weight matrix for the clusterpath penalty)
- `gss_tol` (the tolerance for the GSS algorithm)
- `conv_tol` (the convergence tolerance)
- `max_iter` (the maximum number of iterations)
- `lambda_lasso` (the penalty parameter for the lasso penalty)
- `eps_lasso` (parameter used for the quadratic approximation of the lasso penalty)
- `W_lasso` (the weight matrix for the lasso penalty)

Note

The function interface and output structure are still experimental and may change in the next version.

Author(s)

Daniel J.W. Touw

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

`clusterpath_weights()`, `lasso_weights()`, `cggm_refit()`, `cggm_cv()`

Examples

```
## CGGM can be used to estimate a clustered precision matrix

# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Estimate the covariance matrix
S <- cov(X)

# Compute the weight matrix for the clusterpath (clustering) weights
W_cpath <- clusterpath_weights(S, phi = 1, k = 2)
```

```
# Compute the weight matrix for the lasso (sparsity) weights
W_lasso <- lasso_weights(S)

# Set values to be used for the aggregation parameter
lambdas <- seq(0, 0.2, by = 0.01)

# Estimate the precision matrix for each value of the aggregation
# parameter and a fixed value of the sparsity parameter
fit <- cggm(S, W_cpath = W_cpath, lambda_cpath = lambdas,
            W_lasso = W_lasso, lambda_lasso = 0.2)

# The index of the first value for lambda for which there are 2 clusters
keep <- fit$cluster_solution_index[2]

# Accessor function that retrieve the solution with 2 clusters
get_Theta(fit, index = keep)
get_clusters(fit, index = keep)


# Often, it is not clear which values of the aggregation parameter
# make up the right sequence. But it can be expanded automatically.
fit <- cggm(S, W_cpath = W_cpath, lambda_cpath = lambdas,
            W_lasso = W_lasso, lambda_lasso = 0.2,
            expand = TRUE)

# A solution with 2 clusters
keep <- fit$cluster_solution_index[2]
get_Theta(fit, index = keep)
get_clusters(fit, index = keep)


## CGGM can also be used to estimate a clustered covariance matrix

# Generate data
set.seed(3)
Sigma <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = Sigma)

# Estimate the covariance matrix and compute its inverse
S <- cov(X)
S_inv <- solve(S)

# Compute the weight matrix for the clusterpath (clustering) weights.
# The input is now the sample precision matrix.
W_cpath <- clusterpath_weights(S_inv, phi = 1, k = 2)
```

```
# Compute the weight matrix for the lasso (sparsity) weights.
# The input is again the sample precision matrix.
W_lasso <- lasso_weights(S_inv)

# Set values to be used for the aggregation parameter
lambdas <- seq(0, 0.2, by = 0.01)

# Use the sample precision matrix to estimate the covariance matrix
# for each value of the aggregation parameter and a fixed value of
# the sparsity parameter
fit <- cggm(S_inv, W_cpath = W_cpath, lambda_cpath = lambdas,
            W_lasso = W_lasso, lambda_lasso = 0.2, expand = TRUE)

# A solution with 2 clusters
keep <- fit$cluster_solution_index[2]
get_Theta(fit, index = keep)
get_clusters(fit, index = keep)
```

`cggm_cv` *Cross Validation for the Clusterpath Estimator of the Gaussian Graphical Model*

Description

Perform cross validation to tune the weight matrix parameters `phi` and `k` (for k-nearest-neighbors) as well as the aggregation parameter `lambda_cpath` and the sparsity parameter `lambda_lasso` of the clusterpath estimator of the Gaussian Graphical Model (CGGM) in order to obtain a sparse estimate with variable clustering of the precision matrix or the covariance matrix. The scoring metric is the negative log-likelihood (lower is better).

Usage

```
cggm_cv(
  X,
  tune_grid,
  kfold = 5,
  folds = NULL,
  connected = TRUE,
  fit = TRUE,
  refit = TRUE,
  lasso_unit_weights = FALSE,
  estimate_Sigma = FALSE,
  verbose = 0,
  n_jobs = 1,
  ...
)
```

Arguments

X	The n times p matrix holding the data, with n observations and p variables.
tune_grid	A data frame with values of the tuning parameters. Each row is a combination of parameters that is evaluated. The columns have the names of the tuning parameters and should include k and phi. The sparsity parameter lambda_lasso and the aggregation parameter lambda are optional. If there is no column named lambda_lasso, the sparsity parameter is set to 0. If there is no column named lambda, an appropriate range for the aggregation parameter is selected for each combination of k, phi, and lambda_lasso.
kfold	The number of folds. Defaults to 5.
folds	Optional argument to manually set the folds for the cross validation procedure. If this is not NULL, it overrides the kfold argument. Defaults to NULL.
connected	Logical, indicating whether connectedness of the weight matrix should be ensured. Defaults to TRUE. See clusterpath_weights().
fit	Logical, indicating whether the cross-validation procedure should consider the result from cggm(), before refitting is applied. Defaults to TRUE. At least one of fit and refit should be TRUE.
refit	Logical, indicating whether the cross-validation procedure should also consider the refitted result from cggm(). See also cggm_refit(). Defaults to TRUE. At least one of fit and refit should be TRUE.
lasso_unit_weights	Logical, indicating whether the weights in the sparsity penalty should be all one or decreasing in the magnitude of the corresponding element of the inverse of the sample covariance matrix. Defaults to FALSE.
estimate_Sigma	Logical, indicating whether CGGM should be used to estimate the covariance matrix based on the sample precision matrix. Defaults to FALSE.
verbose	Determines the amount of information printed during the cross validation. Defaults to 0.
n_jobs	Number of parallel jobs used for cross validation. If 0 or smaller, uses the maximum available number of physical cores. Defaults to 1 (sequential).
...	Additional arguments to be passed down to cggm() and cggm_refit().

Value

An object of class "CGGM_CV" with the following components:

fit	A list with cross-validation results for CGGM without the refitting step. It consists of four components: • final (an object of class "CGGM" corresponding to the final model fit using the optimal values of the tuning parameters; see cggm()) • scores (a data frame containing the values of the tuning parameters and the corresponding cross-validation scores) • opt_index (the index of the optimal aggregation parameter lambda_cpath in the final model fit) • opt_tune (a data frame containing the values of the tuning parameters)

`refit`	A list with cross-validation results for CGGM including the refitting step. It contains the same four components as above, except that `final` is an object of class `"CGGM_refit"` (see `cggm_refit()`).
`raw_cv_results`	A list of raw cross-validation results before restructuring.
`best`	A character string indicating whether the optimal model fit without the refitting step (`"fit"`) or including the refitting step ("refit") has a better cross-validation score.

Note

The function interface and output structure are still experimental and may change in the next version.

Author(s)

Daniel J.W. Touw, modifications by Andreas Alfons

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

`clusterpath_weights()`, `lasso_weights()`, `cggm()`, `cggm_refit()`

Examples

```
# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Use cross-validation to select the tuning parameters
fit_cv <- cggm_cv(
  X = X,
  tune_grid = expand.grid(
    phi = 1,
    k = 2,
    lambda_lasso = c(0, 0.02),
    lambda = seq(0, 0.2, by = 0.01)
  ),
  folds = cv_folds(nrow(X), 5)
)
```

```
# The best solution has 2 clusters
get_Theta(fit_cv)
get_clusters(fit_cv)
```

---

cggm_refit *Refit the Gaussian Graphical Model for a Given Aggregation and Sparsity Structure*

Description

Estimate the parameters of a clustered and sparse precision matrix or covariance matrix based on a restricted negative log-likelihood loss function. The restrictions are given by the provided aggregation and sparsity structure. This function is different from `cggm()`, as there are no aggregation and sparsity penalties on the precision or covariance matrix.

Usage

```
cggm_refit(cggm_output, verbose = 0)
```

Arguments

`cggm_output`	An object of class `"CGGM"` as returned by `cggm()`.
`verbose`	Determines the amount of information printed during the optimization. Defaults to `0`.

Value

An object of class `"CGGM_refit"` with the following components:

`A`, `R`	Lists of matrices. Each pair of matrices with the same index parametrize the estimated precision matrix after the refitting step given the aggregation structure found with the corresponding value of the aggregation parameter `lambda_cpath` (and sparsity structure found with the value of the sparsity parameter `lambda_lasso`). It is not recommended to use these directly, instead use the accessor function `get_Theta()` to extract the estimated precision matrix for a given index of the aggregation parameter.
`clusters`	An integer matrix in which each row contains the cluster assignment of each variable for the corresponding value of the aggregation parameter `lambda_cpath`. Use the accessor function `get_clusters()` to extract the cluster assignment for a given index of the aggregation parameter.
`lambdas`	A vector with the values for the aggregation parameter `lambda_cpath` for which the CGGM loss function has been minimized.
`Theta`	List of matrices. Contains the solution to the minimization procedure for each value of the aggregation parameter `lambda_cpath`. It is not recommended to use these directly, instead use the accessor function `get_Theta()` to extract the estimated precision matrix for a given index of the aggregation parameter.

`cluster_counts` An integer vector containing the number of clusters obtained for each value of the aggregation parameter `lambda_cpath`.

`cluster_solution_index`
An integer vector containing the index of the value of the aggregation parameter `lambda_cpath` for which a certain number of clusters was attained. For example, `cluster_solution_index[2]` yields the index of the smallest value for `lambda_cpath` for which a solution with two clusters was found. Contains -1 if there is no value for `lambda_cpath` with that number of clusters.

`n` The number of values of the aggregation parameter `lambda_cpath` for which the CGGM loss function was minimized.

Note

The function interface and output structure are still experimental and may change in the next version.

Author(s)

Daniel J.W. Touw

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

`cggm()`, `cggm_cv()`

Examples

```
# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Estimate the covariance matrix
S <- cov(X)

# Compute the weight matrix for the clusterpath (clustering) weights
W_cpath <- clusterpath_weights(S, phi = 1, k = 2)

# Compute the weight matrix for the lasso (sparsity) weights
W_lasso <- lasso_weights(S)
```

```
# Set values to be used for the aggregation parameter
lambdas <- seq(0, 0.2, by = 0.01)

# Estimate the precision matrix while automatically expanding
# the sequence of values for the aggregation parameter
fit <- cggm(S, W_cpath = W_cpath, lambda_cpath = lambdas,
            W_lasso = W_lasso, lambda_lasso = 0.2,
            expand = TRUE)

# Apply the refitting step to the results, estimating the
# precision matrix based on the clustering and sparsity
# patterns but without additional shrinkage
refit <- cggm_refit(fit)

# A solution with 2 clusters
keep <- refit$cluster_solution_index[2]
get_Theta(refit, index = keep)
get_clusters(refit, index = keep)
```

`clusterpath_weights` *Compute the Weight Matrix for the Clusterpath Penalty*

Description

Compute the (possibly sparse) weight matrix for the clusterpath penalty in the clusterpath estimator of the Gaussian graphical model (CGGM). Weights are computed based on a distance measure so that variables that are close are clustered more quickly (higher weight) than variables that are far apart (lower weight). Only neighboring variables thereby receive a nonzero weight. Additionally, groups of variables that would not be connected via nonzero weights due to the sparsity of the weight matrix can still be connected by applying a minimum spanning tree algorithm.

Usage

```
clusterpath_weights(S, phi, k, connected = TRUE)
```

Arguments

`S`	The sample covariance matrix of the data.
`phi`	Tuning parameter of the weights.
`k`	The number of nearest neighbors that should be used to set weights to a nonzero value. If `0 < k < ncol(S)`, the dense weight matrix will be made sparse, otherwise the dense matrix is returned.
`connected`	A logical indicating whether a connected weight matrix should be enforced. Defaults to `TRUE`.

Value

A weight matrix for the clusterpath penalty.

Author(s)

Daniel J.W. Touw

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

`lasso_weights()`, `cggm()`, `cggm_refit()`, `cggm_cv()`

Examples

```
# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Estimate the covariance matrix
S <- cov(X)

# Compute the weight matrix for the clusterpath (clustering) weights
W_cpath <- clusterpath_weights(S, phi = 1, k = 2)
W_cpath
```

---

cv_folds *Create Cross-Validation Folds*

Description

Obtain indices for splitting observations into K blocks to be to be folded into training and test data during K-fold cross-validation.

Usage

```
cv_folds(n, K = 5L)
```

Arguments

n	an integer giving the number of observations to be split.
K	an integer giving the number of blocks into which the observations should be split (the default is five).

Value

A list of indices giving the blocks of observations to be folded into training and test data during cross-validation.

Author(s)

Andreas Alfons

See Also

cggm_cv()

Examples

```
cv_folds(20, K = 5)
```

---

get_clusters *Extract the Cluster Assignment*

Description

Extract a cluster assignment obtained via the clusterpath estimator of the Gaussian graphical model (CGGM).

Usage

```
get_clusters(object, ...)

## S3 method for class 'CGGM'
get_clusters(object, index, ...)

## S3 method for class 'CGGM_refit'
get_clusters(object, index, ...)

## S3 method for class 'CGGM_CV'
get_clusters(object, which = NULL, ...)
```

Arguments

object	an object from which to extract the cluster assignment.
...	additional arguments are currently ignored.
index	an integer specifying the step along the clusterpath for which to extract the cluster assignment.

`which` a character string specifying for which solution to extract the cluster assignment. Possible values are `"refit"` for the solution including the refitting step (see `cggm_refit()`), or `"fit"` for the solution without without the refitting step (see `cggm()`). If `NULL` (the default), the solution with the better cross-validation score is used.

Value

An integer vector giving the obtained cluster assignment for each variable.

For the `"CGGM_CV"` method (see `cggm_cv()`), the returned cluster assignment corresponds to the optimal values of the tuning parameters.

Author(s)

Daniel J.W. Touw

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

`cggm()`, `cggm_refit()`, `cggm_cv()`

`get_Theta()`

Examples

```
# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Estimate the covariance matrix
S <- cov(X)

# Compute the weight matrix for the clusterpath (clustering) weights
W_cpath <- clusterpath_weights(S, phi = 1, k = 2)

# Compute the weight matrix for the lasso (sparsity) weights
W_lasso <- lasso_weights(S)

# Set values to be used for the aggregation parameter
lambdas <- seq(0, 0.2, by = 0.01)
```

```
# Estimate the precision matrix while automatically expanding
# the sequence of values for the aggregation parameter
fit <- cggm(S, W_cpath = W_cpath, lambda_cpath = lambdas,
            W_lasso = W_lasso, lambda_lasso = 0.2,
            expand = TRUE)

# Cluster membership for 2 clusters
get_clusters(fit, index = fit$cluster_solution_index[2])


# Apply the refitting step to the results, estimating the
# precision matrix based on the clustering and sparsity
# patterns but without additional shrinkage
refit <- cggm_refit(fit)

# Cluster membership for 2 clusters
get_clusters(refit, index = refit$cluster_solution_index[2])
```

`get_Theta` *Extract the Estimated Precision Matrix*

Description

Extract a (block-structured and sparse) precision matrix obtained via the clusterpath estimator of the Gaussian graphical model (CGGM).

Usage

```
get_Theta(object, ...)

## S3 method for class 'CGGM'
get_Theta(object, index, ...)

## S3 method for class 'CGGM_refit'
get_Theta(object, index, ...)

## S3 method for class 'CGGM_CV'
get_Theta(object, which = NULL, ...)
```

Arguments

`object`	an object from which to extract the precision matrix.
`...`	additional arguments are currently ignored.
`index`	an integer specifying the step along the clusterpath for which to extract the precision matrix.

`which` a character string specifying for which solution to extract the precision matrix. Possible values are `"refit"` for the solution including the refitting step (see `cggm_refit()`), or `"fit"` for the solution without without the refitting step (see `cggm()`). If `NULL` (the default), the solution with the better cross-validation score is used.

Value

The estimated (block-structured and sparse) precision matrix.

For the `"CGGM_CV"` method (see `cggm_cv()`), the returned precision matrix corresponds to the optimal values of the tuning parameters.

Author(s)

Daniel J.W. Touw

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

`cggm()`, `cggm_refit()`, `cggm_cv()`

`get_clusters()`

Examples

```
# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Estimate the covariance matrix
S <- cov(X)

# Compute the weight matrix for the clusterpath (clustering) weights
W_cpath <- clusterpath_weights(S, phi = 1, k = 2)

# Compute the weight matrix for the lasso (sparsity) weights
W_lasso <- lasso_weights(S)

# Set values to be used for the aggregation parameter
lambdas <- seq(0, 0.2, by = 0.01)
```

```
# Estimate the precision matrix while automatically expanding
# the sequence of values for the aggregation parameter
fit <- cggm(S, W_cpath = W_cpath, lambda_cpath = lambdas,
            W_lasso = W_lasso, lambda_lasso = 0.2,
            expand = TRUE)

# Precision matrix with 2 clusters
get_Theta(fit, index = fit$cluster_solution_index[2])


# Apply the refitting step to the results, estimating the
# precision matrix based on the clustering and sparsity
# patterns but without additional shrinkage
refit <- cggm_refit(fit)

# Precision matrix with 2 clusters
get_Theta(refit, index = refit$cluster_solution_index[2])
```

`lasso_weights` *Compute the Weight Matrix for the Lasso Penalty*

Description

Compute the weight matrix for the lasso penalty in the clusterpath estimator of the Gaussian graphical model (CGGM).

Usage

```
lasso_weights(S, unit = FALSE)
```

Arguments

`S`	The sample covariance matrix of the data.
`unit`	A logical indicating whether the weights should be all one or based on the inverse of `S`.

Value

A weight matrix for the lasso penalty.

Author(s)

Daniel J.W. Touw

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

clusterpath_weights(), cggm(), cggm_refit(), cggm_cv()

Examples

```
# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Estimate the covariance matrix
S <- cov(X)

# Compute the weight matrix for the lasso (sparsity) weights
W_lasso <- lasso_weights(S)
W_lasso
```

---

min_clusters *Calculate the Minimum Number of Clusters*

Description

Compute the minimum number of clusters achievable by the clusterpath penalty using the provided weight matrix.

Usage

```
min_clusters(W)
```

Arguments

W	The weight matrix for the clusterpath penalty.

Value

An integer giving the minimum number of clusters.

Author(s)

Daniel J.W. Touw

References

D.J.W. Touw, A. Alfons, P.J.F. Groenen and I. Wilms (2025) *Clusterpath Gaussian Graphical Modeling*. arXiv:2407.00644. doi:10.48550/arXiv.2407.00644.

See Also

clusterpath_weights(), cggm(), cggm_cv()

Examples

```
# Generate data
set.seed(3)
Theta <- matrix(
  c(2, 1, 0, 0,
    1, 2, 0, 0,
    0, 0, 4, 1,
    0, 0, 1, 4),
  nrow = 4
)
X <- mvtnorm::rmvnorm(n = 100, sigma = solve(Theta))

# Estimate the covariance matrix
S <- cov(X)


# Compute the weight matrix for the clusterpath (clustering) weights
# without enforcing connectedness
W_cpath <- clusterpath_weights(S, phi = 1, k = 1, connected = FALSE)

# The smallest number of clusters is 2
min_clusters(W_cpath)


# Compute the weight matrix for the clusterpath (clustering) weights
# with enforcing connectedness (default behavior)
W_cpath <- clusterpath_weights(S, phi = 1, k = 1, connected = TRUE)

# The smallest number of clusters is 1
min_clusters(W_cpath)
```

Index