

Package: CVN (via r-universe)

October 12, 2026

Title Covariate-Varying Networks

Version 1.1

Description Inferring high-dimensional Gaussian graphical networks that change with multiple discrete covariates. Louis Dijkstra, Arne Godt, Ronja Foraita (2024)
<[doi:10.48550/arXiv.2407.19978](https://doi.org/10.48550/arXiv.2407.19978)>.

License GPL-3

URL <https://github.com/bips-hb/CVN>, <https://bips-hb.github.io/CVN/>

BugReports <https://github.com/bips-hb/CVN/issues>

Depends R (>= 4.0.0)

Imports crayon, doParallel, dplyr, foreach, ggplot2, glasso, igraph, magrittr, Matrix, methods, parallel, progress, Rcpp, reshape2, stats, utils, visNetwork

Suggests htmltools, knitr, rmarkdown

LinkingTo Rcpp

LazyData true

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Spelling Arne, Foraita, Godt, Ronja

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev

Repository <https://bips-hb.r-universe.dev>

Date/Publication 2025-10-23 07:48:13 UTC

RemoteUrl <https://github.com/bips-hb/CVN>

RemoteRef HEAD

RemoteSha 7d878b78d4698c7cf36ad1e24dc20e954f229d0c

Contents

CVN-package	2
combine_cvn	3
combine_cvn_interpolated	4
create_edges_visnetwork	5
create_nodes_visnetwork	6
create_weight_matrix	7
CVN	8
cvn_edge_summary	11
determine_information_criterion_cvn	12
extract_cvn	12
find_core_graph	13
find_unique_edges	14
genlassoRcpp	14
grid	15
hamming_distance	16
hamming_distance_adj_matrices	16
hd_weight_matrix	17
interpolate	18
matrix_A_inner_ADMM	19
plot.cvn	20
plot_hamming_distances	21
plot_hamming_distances_cvn	22
plot_information_criterion	23
plot_weight_matrix	24
print.cvn	25
relative_difference_precision_matrices	25
set_attributes_to_edges_visnetwork	26
strip_cvn	27
updateZRcpp	28
visnetwork	29
visnetwork_cvn	30
Index	32

 CVN-package

Covariate-varying Networks

Description

Inferring high-dimensional Gaussian graphical networks that change with multiple discrete covariates.

Author(s)

Maintainer: Ronja Foraita <foraita@leibniz-bips.de> ([ORCID](#))

Authors:

- Louis Dijkstra ([ORCID](#))

Other contributors:

- Lukas Burk ([ORCID](#)) [contributor]
- Leibniz Inst Prev Res & Epidemiology - BIPS ([ROR](#)) [copyright holder, funder]
- Deutsche Forschungsgemeinschaft (DFG) [funder]

References

Dijkstra L, Godt A, Foraita R *Inferring High-Dimensional Dynamic Networks Changing with Multiple Covariates* (2024), Arxiv, <https://arxiv.org/abs/2407.19978>.

See Also

Useful links:

- <https://github.com/bips-hb/CVN>
- <https://bips-hb.github.io/CVN/>
- Report bugs at <https://github.com/bips-hb/CVN/issues>

Examples

```
data(grid)
W <- create_weight_matrix(type = "grid", k=3, l=3, plot = FALSE)

cvn <- CVN(grid, W, lambda1 = 1, lambda2 = 1:2,
           n_cores = 1,
           eps = 1e-2, maxiter = 1000, verbose = TRUE)
```

combine_cvn

Combine CVNs of size l

Description

Function to combine single CVNs. Handy, if CVNs are computed in parallel using foreach loop and n_cores = 1

Usage

```
combine_cvn(cvn_list, minimal = TRUE)
```

Arguments

cvn_list a list of CVNs, each of size 1 (one lambda/gamma pair)
 minimal logical; stripped CVN (default = TRUE)

Value

a CVN of size length(cvn_list)

Examples

```
data(grid)

#' Choice of the weight matrix W. Each of 2 covariates has 3 categories
#' (uniform random)
W_grid <- create_weight_matrix("grid", k = 3, l = 3)

fit1 <- CVN(data = grid, W = W_grid, lambda1 = 1, lambda2 = 0.5,
            gamma_ebic = 0.75,
            n_cores = 1, eps = 1e-2, maxiter = 200, minimal = TRUE)

fit2 <- CVN(data = grid, W = W_grid, lambda1 = 2, lambda2 = 0.75,
            gamma_ebic = 0.75,
            n_cores = 1, eps = 1e-2, maxiter = 200, minimal = TRUE)

(fit <- combine_cvn(list(fit1, fit2)))
```

combine_cvn_interpolated

Combine CVN object with interpolated CVN object

Description

Combine an interpolated CVN to an original CVN

Usage

```
combine_cvn_interpolated(cvn, cvn_interpolated)
```

Arguments

cvn A CVN fit with m graphs
 cvn_interpolated A interpolated CVN from a fitted CVN model with m graphs

Value

A 'cvn_interpolated' object (see [interpolate](#)).

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)

interpolate <- interpolate(fit, c(0,0,0,0,0,0,0,0.5,0.5), truncate = 0.05)
fit10 <- combine_cvn_interpolated(fit, interpolate)
```

```
create_edges_visnetwork
```

Create a data.frame for the Edges for visNetwork

Description

In order to visualize a graph, we need to create a data.frame that can be used by the visNetwork package. This function returns the needed data.frame given a adjacency matrix.

Usage

```
create_edges_visnetwork(adj_matrix)
```

Arguments

`adj_matrix` A symmetric adjacency matrix

Value

Data frame that be used as input for visNetwork

Examples

```
nodes <- create_nodes_visnetwork(n_nodes = 5, labels = LETTERS[1:5])

adj_matrix <- matrix(c(0, 1, 0, 1, 0,
                      1, 0, 1, 0, 0,
                      0, 1, 0, 0, 0,
                      1, 0, 0, 0, 1,
                      0, 0, 0, 1, 0), ncol = 5)

edges <- create_edges_visnetwork(adj_matrix)

shared_edges <- data.frame(from = c(1,2), to = c(4, 3))

edges <- set_attributes_to_edges_visnetwork(edges,
                                             subset_edges = shared_edges,
                                             width = c(3, .5),
                                             color = c("red", "blue"))

visnetwork(nodes, edges)
```

`create_nodes_visnetwork`*Nodes for the visNetwork package*

Description

Creates a data frame that can be used for the visNetwork package.

Usage

```
create_nodes_visnetwork(n_nodes, labels = 1:n_nodes)
```

Arguments

<code>n_nodes</code>	Number of nodes in the graph
<code>labels</code>	The labels for the individual nodes (Default: 1:n_nodes)

Value

Data frame with two columns: id and title

Examples

```
nodes <- create_nodes_visnetwork(n_nodes = 5, labels = LETTERS[1:5])

adj_matrix <- matrix(c(0, 1, 0, 1, 0,
                      1, 0, 1, 0, 0,
                      0, 1, 0, 0, 0,
                      1, 0, 0, 0, 1,
                      0, 0, 0, 1, 0), ncol = 5)

edges <- create_edges_visnetwork(adj_matrix)

shared_edges <- data.frame(from = c(1,2), to = c(4, 3))

edges <- set_attributes_to_edges_visnetwork(edges,
                                             subset_edges = shared_edges,
                                             width = c(3, .5),
                                             color = c("red", "blue"))

visnetwork(nodes, edges)
```

Description

This function generates different weight matrices for two external covariates, W . There are several types:

full All graphs are fully connected with weight 1

glasso All graphs are disconnected with weight 0. This mimics the GLASSO, where each graph is estimated independently

grid A weight matrix for a $k \times l$ grid

uniform-random Fully-connected, but the entries are drawn from a uniform distribution

Usage

```
create_weight_matrix(  
  type = c("full", "glasso", "grid", "uniform-random"),  
  k,  
  l,  
  plot = FALSE  
)
```

Arguments

type	The type of weight matrix
k	Number of categories in the first external covariate
l	Number of categories in the second external covariate
plot	If TRUE, the weight matrix is plotted (default: FALSE)

Value

Weight matrix

Examples

```
W <- create_weight_matrix(type="grid", k=3, l=2)
```

Description

Estimates a covariate-varying network model (CVN), i.e., m Gaussian graphical models that change with (multiple) external covariate(s). The smoothing between the graphs is specified by the $(m \times m)$ -dimensional weight matrix W . The function returns the estimated precision matrices for each graph.

Usage

```
CVN(
  data,
  W,
  lambda1 = 1:2,
  lambda2 = 1:2,
  gamma1 = NULL,
  gamma2 = NULL,
  rho = 1,
  eps = 1e-04,
  maxiter = 100,
  truncate = 1e-05,
  rho_genlasso = 1,
  eps_genlasso = 1e-10,
  maxiter_genlasso = 100,
  truncate_genlasso = 1e-04,
  n_cores = min(length(lambda1) * length(lambda2), detectCores() - 1),
  normalized = FALSE,
  warmstart = TRUE,
  minimal = FALSE,
  gamma_ebic = 0.5,
  verbose = TRUE
)
```

Arguments

data	A list with matrices, each entry associated with a single graph. The number of columns should be the same for each matrix. Number of observations can differ
W	The $(m \times m)$ -dimensional symmetric weight matrix W
lambda1	Vector with different λ_1 . LASSO penalty terms (Default: 1:2)
lambda2	Vector with different λ_2 . The global smoothing parameter values (Default: 1:2)
gamma1	A vector of γ_1 's LASSO penalty terms, where $\gamma_1 = \frac{2\lambda_1}{mp(1-p)}$. If gamma1 is set, the value of lambda1 is ignored. (Default: NULL).
gamma2	A vector of γ_2 's global smoothing parameters, where that $\gamma_2 = \frac{4\lambda_2}{m(m-1)p(p-1)}$. If gamma2 is set, the value of lambda2 is ignored.(Default: NULL).

rho	The ρ penalty parameter for the global ADMM algorithm (Default: 1)
eps	If the relative difference between two update steps is smaller than ϵ , the algorithm stops. (Default: 1e-4)
maxiter	Maximum number of iterations (Default: 100)
truncate	All values of the final $\hat{\Theta}_i$'s below truncate will be set to 0. (Default: 1e-5)
rho_genlasso	The ρ penalty parameter for the ADMM algorithm used to solve the generalized LASSO (Default: 1)
eps_genlasso	If the relative difference between two update steps is smaller than ϵ , the algorithm stops. (Default: 1e-10)
maxiter_genlasso	Maximum number of iterations for solving the generalized LASSO problem (Default: 100)
truncate_genlasso	All values of the final $\hat{\beta}$ below truncate_genlasso will be set to 0. (Default: 1e-4)
n_cores	Number of cores used (Default: max. number of cores - 1, or the total number penalty term pairs if that is less)
normalized	Data is normalized if TRUE. Otherwise the data is only centered (Default: FALSE)
warmstart	If TRUE, use the glasso package for estimating the individual graphs first (Default: TRUE)
minimal	If TRUE, the returned cvn is minimal in terms of memory, i.e., Theta, data and Sigma are not returned (Default: FALSE)
gamma_ebic	Gamma value for the eBIC (Default: 0.5)
verbose	Verbose (Default: TRUE)

Value

A CVN object containing the estimates for all the graphs for each different value of (λ_1, λ_2) . General results for the different values of (λ_1, λ_2) can be found in the data frame `results`. It consists of multiple columns, namely:

id	The id. This corresponds to the indices of the lists
lambda1	λ_1 value
lambda2	λ_2 value
gamma1	γ_1 value
gamma2	γ_2 value
converged	whether algorithm converged or not
value	value of the negative log-likelihood function
n_iterations	number of iterations of the ADMM
aic	Akaike information criterion
bic	Bayesian information criterion
ebic	Extended Bayesian information criterion

edges_median	Median number of edges across the m estimated graphs
edges_iqr	Interquartile range of edges across the m estimated graphs

The estimates of the precision matrices and the corresponding adjacency matrices for the different values of (λ_1, λ_2) can be found

Theta	A list with the estimated precision matrices $\{\hat{\Theta}_i(\lambda_1, \lambda_2)\}_{i=1}^m$, (only if <code>minimal = FALSE</code>)
adj_matrices	A list with the estimated adjacency matrices corresponding to the estimated precision matrices in Theta. The entries are 1 if there is an edge, 0 otherwise. The matrices are sparse using package Matrix

In addition, the input given to the CVN function is stored in the object as well:

Sigma	Empirical covariance matrices $\{\hat{\Sigma}_i\}_{i=1}^m$, (only if <code>minimal = FALSE</code>)
m	Number of graphs
p	Number of variables
n_obs	Vector of length m with number of observations for each graph
data	The data, but then normalized or centered (only if <code>minimal = FALSE</code>)
W	The $(m \times m)$ -dimensional weight matrix W
maxiter	Maximum number of iterations for the ADMM
rho	The ρ ADMM's penalty parameter
eps	The stopping criterion ϵ
truncate	Truncation value for $\{\hat{\Theta}_i\}_{i=1}^m$
maxiter_genlasso	Maximum number of iterations for the generalized LASSO
rho_genlasso	The ρ generalized LASSO penalty parameter
eps_genlasso	The stopping criterion ϵ for the generalized LASSO
truncate_genlasso	Truncation value for β of the generalized LASSO
n_lambda_values	Total number of (λ_1, λ_2) value combinations
normalized	If TRUE, data was normalized. Otherwise data was only centered
warmstart	If TRUE, warmstart was used
minimal	If TRUE, data, Theta and Sigma are not added
hits_border_aic	If TRUE, the optimal model based on the AIC hits the border of (λ_1, λ_2)
hits_border_bic	If TRUE, the optimal model based on the BIC hits the border of (λ_1, λ_2)
gamma_ebic	Gamma value used to calculate eBIC

Reusing Estimates

When estimating the graph for different values of λ_1 and λ_2 , we use the graph estimated (if available) for other λ_1 and λ_2 values closest to them.

Examples

```
data(grid)

#' Choice of the weight matrix W. Each of 2 covariates has 3 categories
#' (uniform random)
W <- create_weight_matrix("uniform-random", k = 3, l = 3)

# lambdas:
lambda1 = 1 # can also be lambda1 = 1:2
lambda2 = 1

(fit <- CVN(data = grid,
            W = W,
            lambda1 = lambda1, lambda2 = lambda2,
            n_cores = 1,
            eps = 1e-2, maxiter = 200, # fast but imprecise
            verbose = TRUE))
```

cvn_edge_summary

*Edge Overview***Description**

Function that gives an overview about the number of edges in each subgraph over all fitted CVN models.

Usage

```
cvn_edge_summary(cvn)
```

Arguments

cvn A cvn object

Value

A data.frame showing the number of edges in each subgraph, the number of edges present in all graphs (core edges) and the number of edges that are unique for a subgraph

Examples

```
data(grid)
W <- create_weight_matrix("grid", 3, 3)

# lambdas:
lambda1 = c(1, 1.5)
lambda2 = .2

fit <- CVN(grid, W, lambda1 = lambda1, lambda2 = lambda2, n_cores = 1)
```

```
# Edge summary for a list of CVN models
cvn_edge_summary(fit)

# Edge summary for a single CVN
fit2 <- extract_cvn(fit, id = 2)
cvn_edge_summary(fit2)
```

determine_information_criterion_cvn

Information Criteria for a cvn object

Description

Determines a given information criteria for a cvn object, see [CVN](#).

Usage

```
determine_information_criterion_cvn(cvn, gamma = 0.5)
```

Arguments

cvn	A CVN object, see CVN
gamma	The gamma value for the eBIC (default: 0.5)

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)
determine_information_criterion_cvn(fit, gamma = 0.7)
```

extract_cvn

Extract CVN

Description

Function that extracts one CVN if the CVN object contains more than one. Its helpful when just one tuning parameter constellation wants to be, e.g., plotted.

Usage

```
extract_cvn(cvn, id)
```

Arguments

cvn A CVN object, see [CVN](#)
 id Integer; Which id from the CVN object should be extracted

Value

A CVN object

Examples

```
## Not run:
# Example code of usage (not run because of longer running time)

data(grid)
W <- create_weight_matrix("grid", 3, 3)
fit <- CVN(grid, W, n_cores = 1,
           lambda1 = c(0.5, 1),
           lambda2 = c(0.1, 0.5))
(fit2 <- extract_cvn(fit, id = 2))

## End(Not run)
```

find_core_graph	<i>The Core Graph</i>
-----------------	-----------------------

Description

Finds the 'core graph', i.e., those edges that are present in all of the estimated graphs

Usage

```
find_core_graph(cvn)
```

Arguments

cvn A cvn object

Value

A list of adjacency matrix, one for each value of (lambda1, lambda2)

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)
find_core_graph(fit)
```

find_unique_edges	<i>Unique Edges</i>
-------------------	---------------------

Description

Function that finds edges that are only present in one subgraph of the CVN model.

Usage

```
find_unique_edges(cvn)
```

Arguments

cvn	A cvn object
-----	--------------

Value

A list of adjacency matrix, one for each value of (lambda1, lambda2), showing the edges that are unique for each single subgraph.

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)

ue <- find_unique_edges(fit)

# Graph 3 has unique edges
ue[[1]][[3]]
```

genlassoRcpp	<i>Generalized LASSO</i>
--------------	--------------------------

Description

Solving Generalized LASSO with fixed $\lambda = 1$ Solves efficiently the generalized LASSO problem of the form

$$\hat{\beta} = \operatorname{argmin} \frac{1}{2} \|y - \beta\|_2^2 + \|D\beta\|_1$$

where β and y are m -dimensional vectors and D is a $(c \times m)$ -matrix where $c \geq m$. We solve this optimization problem using an adaption of the ADMM algorithm presented in Zhu (2017).

Usage

```
genlassoRcpp(Y, W, m, eta1, eta2, a, rho, max_iter, eps, truncate)
```

Arguments

Y	The y vector of length m
W	The weight matrix W of dimensions $m \times m$
m	The number of graphs
eta1	Equals λ_1 / rho
eta2	Equals λ_2 / rho
a	Value added to the diagonal of $-D'D$ so that the matrix is positive definite, see matrix_A_inner_ADMM
rho	The ADMM's parameter
max_iter	Maximum number of iterations
eps	Stopping criterion. If differences are smaller than ϵ , algorithm is halted
truncate	Values below truncate are set to 0

Value

The estimated vector $\hat{\beta}$

Author(s)

Louis Dijkstra

References

Zhu, Y. (2017). An Augmented ADMM Algorithm With Application to the Generalized Lasso Problem. *Journal of Computational and Graphical Statistics*, 26(1), 195–204. <https://doi.org/10.1080/10618600.2015.111449>

See Also

[genlasso_wrapper](#)

grid	<i>Data for a grid of graphs (3 x 3)</i>
------	--

Description

Data generated for 9 graphs in total, organized in a grid of (3x3). Each data set has 10 variables drawn from a multinormal distribution given 9 adjacency matrices reflecting a graph structure that follows an Erdős-Rényi graph with probability 0.5. The number of edges that are removed or added in each step of the x- and y direction is 2.

Usage

data(grid)

hamming_distance	<i>Structural Hamming Distance for a cvn Object</i>
------------------	---

Description

Returns the structural Hamming distances

Usage

```
hamming_distance(cvn, verbose = TRUE)
```

Arguments

cvn	A cvn object created by the CVN function
verbose	If TRUE, shows a progress bar

Value

A list of symmetric matrices. Each matrix contains the structural Hamming distances between the different graphs. Each item in the list corresponds to one (λ_1, λ_2) pair

See Also

[CVN](#)

hamming_distance_adj_matrices	<i>Structural Hamming Distance</i>
-------------------------------	------------------------------------

Description

Returns the structural Hamming distance between multiple graphs

Usage

```
hamming_distance_adj_matrices(adj_matrices)
```

Arguments

adj_matrices	A list of adjacency matrices
--------------	------------------------------

Value

Matrix of Hamming distances

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)

hamming_distance_adj_matrices(fit$adj_matrices[[1]])
```

hd_weight_matrix	<i>Heatmap Plot of the Weight Matrix</i>
------------------	--

Description

Returns a heatmap of the weight matrix

Usage

```
hd_weight_matrix(
  W,
  title = "",
  legend_label = "weight",
  add_counts_to_cells = TRUE,
  add_ticks_labels = TRUE,
  t = -6,
  r = -8
)
```

Arguments

W	Weight matrix
title	Title plot (Default is none)
legend_label	Title of the legend (Default: "weight")
add_counts_to_cells	If TRUE, counts from the matrix are added to the plot (Default: TRUE)
add_ticks_labels	If TRUE, the number corresponding to the graph is add to the plot (Default: TRUE)
t	Distance between tick labels and x-axis (Default: -6)
r	Distance between tick labels and y-axis (Default: -8)

Value

A heatmap plot

See Also

[plot_weight_matrix](#)

Examples

```
W_uniform <- round(create_weight_matrix(type="uniform-random", 3, 2), 2)
hd_weight_matrix(W_uniform, title = "2 external covariates with 3 and 2 categories respectively")
```

interpolate

*Interpolation of a Graph***Description**

Estimates a graph for which there are no observation based on a previously fitted CVN model

Usage

```
interpolate(cvn, weights, truncate = NULL)
```

Arguments

cvn	A CVN fit with m graphs
weights	A vector of length m with the regression coefficients
truncate	Truncation value. When a value in the precision matrix is considered 0. If NULL, the same truncation is used as for the fitted CVN model (Default)

Value

A 'cvn_interpolated' object, which is a list with

adj_matrices	A list of adjacency matrix. One for each pair of (λ_1, λ_2) values. The entries are 1 if there is an edge, 0 otherwise. The matrices are sparse using package Matrix
m	Number of graphs
p	Number of variables
weights	The weights used for interpolation
truncate	Truncation value
n_lambda_values	Total number of (λ_1, λ_2) value combinations

results. It consists of two columns:

lambda1	λ_1 value
lambda2	λ_2 value

Examples

```

path <- system.file("cvnfit.rda", package = "CVN")
load(path)

interpolate <- interpolate(fit, c(0,0,0,0,0,0,0,0,0.5,0.5), truncate = 0.05)
fit10 <- combine_cvn_interpolated(fit, interpolate)
fit10 <- visnetwork_cvn(fit10)

if (requireNamespace("htmltools", quietly = TRUE)) {
  htmltools::browsable(htmltools::tagList(
    htmltools::div(style = "display: grid; grid-template-columns: repeat(3, 1fr);
      column-gap: 1px; row-gap: -10px",
      fit10$plots[[1]][8:10]))
} else {
  fit10$plots[[1]][8:10]
}

```

matrix_A_inner_ADMM	<i>Determine matrix A for inner-ADMM for the Z-update step</i>
---------------------	--

Description

The Z -update step requires us to solve a special Generalized LASSO problem of the form

$$\hat{\beta} = \operatorname{argmin} \frac{1}{2} \|y - \beta\|_2^2 + \|D\beta\|_1$$

where β and y are m -dimensional vectors and D is a $(c \times m)$ -matrix where $c = (m^2 + m)/2$. We solve this optimization problem using an adaption of the ADMM algorithm presented in Zhu (2017). This algorithm requires the choice of a matrix A such that $A - D'D$ is positive semidefinite. In order to optimize the ADMM, we choose the matrix A to be diagonal with a fixed value a . This function determines the smallest value of a such that $A - D'D$ is indeed positive semidefinite. We do this by determining the largest eigenvalue

Usage

```
matrix_A_inner_ADMM(W, eta1, eta2)
```

Arguments

W	Weight matrix W
eta1, eta2	The values $\eta_1 = \lambda_1/\rho$ and $\eta_2 = \lambda_2/\rho$

Value

Value of a

Author(s)

Louis Dijkstra

References

Zhu, Y. (2017). An Augmented ADMM Algorithm With Application to the Generalized Lasso Problem. *Journal of Computational and Graphical Statistics*, 26(1), 195–204.
[doi:10.1080/10618600.2015.1114491](https://doi.org/10.1080/10618600.2015.1114491)

plot.cvn

Plot Function for CVN Object Class

Description

Custom plot method for CVN objects.

Usage

```
## S3 method for class 'cvn'
plot(x, ...)
```

Arguments

x	cvn object
...	Additional arguments to pass to visnetwork_cvn

See Also

CVN

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)
fit <- plot(fit)
fit$plots[[1]][[4]]
```

plot_hamming_distances

Heat Map of the Distances between Graphs

Description

Returns a heat map of the distance matrix for a particular CVN

Usage

```
plot_hamming_distances(
  distance_matrix,
  absolute = TRUE,
  limits = c(NA, NA),
  title = "",
  legend_label = "Hamming Distance",
  add_counts_to_cells = TRUE,
  add_ticks_labels = TRUE,
  t = -6,
  r = -8
)
```

Arguments

distance_matrix	Symmetric matrix with distances
absolute	If FALSE, rescaled to $[0, 1]$
limits	The limits for the values of the Hamming distance
title	Title plot (Default is none)
legend_label	Title of the legend (Default: "Hamming Distance")
add_counts_to_cells	If TRUE, counts from the matrix are added to the plot (Default: TRUE)
add_ticks_labels	If TRUE, the number corresponding to the graph is add to the plot (Default: TRUE)
t	Distance between tick labels and x-axis (Default: -6)
r	Distance between tick labels and y-axis (Default: -8)

Value

A heatmap plot

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)
hd <- hamming_distance(fit, verbose = FALSE)

plot_hamming_distances(hd$distance[[1]])
```

```
plot_hamming_distances_cvn
```

Heatmaps for a CVN

Description

Creates all the heatmaps for a CVN, a heatmap for each pair of (λ_1, λ_2)

Usage

```
plot_hamming_distances_cvn(
  cvn,
  absolute = TRUE,
  same_range = TRUE,
  titles = rep("", cvn$n_lambda_values),
  legend_label = "Hamming Distance",
  add_counts_to_cells = TRUE,
  add_ticks_labels = TRUE,
  t = -6,
  r = -8,
  verbose = TRUE
)
```

Arguments

cvn	A cvn object
absolute	If FALSE, rescaled to $[0, 1]$
same_range	If TRUE, all heatmaps have the same range of values of the Hamming distance shown (Default: TRUE)
titles	Title of the plots (Default is none)
legend_label	Title of the legend (Default: "Hamming Distance")
add_counts_to_cells	If TRUE, counts from the matrix are added to the plot (Default: TRUE)
add_ticks_labels	If TRUE, the number corresponding to the graph is add to the plot (Default: TRUE)
t	Distance between tick labels and x-axis (Default: -6)
r	Distance between tick labels and y-axis (Default: -8)
verbose	If TRUE, shows progress bar (Default: TRUE)

Value

List of plots

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)

plot_hamming_distances_cvn(fit)
```

plot_information_criterion

Heat Map of an Information Criterion (AIC or BIC)

Description

Returns a heat map of the AIC or BIC for a fitted CVN

Usage

```
plot_information_criterion(
  cvn,
  criterion = c("bic", "aic", "ebic"),
  use_gammas = TRUE,
  show_minimum = TRUE,
  title = "",
  xlabel = NULL,
  ylabel = NULL,
  legend_label = NULL,
  limits = c(NA, NA)
)
```

Arguments

cvn	Fitted CVN, see CVN
criterion	The information criterion, must be either 'aic' or 'bic'. Default: 'bic'
use_gammas	If TRUE, plots the γ -values. Otherwise, the λ -values are used
show_minimum	If TRUE, an orange dot is put on the point with the minimum value of the information criterion is. If FALSE, no dot is added. Default: TRUE.
title	Title plot (Default is none)
xlabel	Label for the x -axis. Default depends on use_gammas. If use_gammas = TRUE, then the label is 'gamma1'. Otherwise, 'lambda1'
ylabel	Label for the y -axis. Default depends on use_gammas. If use_gammas = TRUE, then the label is 'gamma1'. Otherwise, 'lambda1'

legend_label	Title for the legend. Default depends on criterion. If 'aic', then the label is 'AIC'. Otherwise, 'BIC'.
limits	The limits for the values of the Hamming distance

Value

A heatmap plot

Examples

```
data(grid)
m <- 9

W <- create_weight_matrix(type="grid", 3, 3)
cvn <- CVN(grid, W,
            lambda1 = 1:2, lambda2 = .5, n_cores = 1)
cvn$results

# The smaller the IC value, the better the fit.
plot_information_criterion(cvn, criterion = "aic")
plot_information_criterion(cvn, criterion = "bic")
plot_information_criterion(cvn, criterion = "ebic")
```

plot_weight_matrix	<i>Plot weight matrix as a grid</i>
--------------------	-------------------------------------

Description

This function requires the igraph package to be installed.

Usage

```
plot_weight_matrix(W, k, l)
```

Arguments

W	The weight matrix which equals an adjacency matrix
k	Number of categories in the first external covariate
l	Number of categories in the second external covariate

Value

An igraph object

See Also

[hd_weight_matrix](#)

Examples

```
# Requires to have the igraph package to be installed
W <- create_weight_matrix(type = "full", k = 2, l = 3)
plot_weight_matrix(W, k=2, l=3)
```

print.cvn

*Print CVN***Description**

Print contents of CVN object

Usage

```
## S3 method for class 'cvn'
print(x, ...)
```

Arguments

x	Object of class 'CVN'
...	Additional arguments to pass

See Also

[CVN](#)

Examples

```
path <- system.file("cvnfit.rda", package = "CVN")
load(path)

print(fit)
```

relative_difference_precision_matrices

*The Relative Difference between two Precision Matrices***Description**

Returns the relative $L1$ difference between precision matrix $\Theta(k+1)$ (parameter Theta_new) and $\Theta(k)$ (parameter Theta_old).

Usage

```
relative_difference_precision_matrices(Theta_new, Theta_old)
```

Arguments

Theta_new	A list with matrices with the updated values of Θ
Theta_old	A list with matrices with the old values of Θ

Details

This is used for checking whether the stopping condition has been met.

Value

The relative difference between $\Theta(k + 1)$ and $\Theta(k)$

```
set_attributes_to_edges_visnetwork
```

Add Attributes to Subset of Edges for visNetwork

Description

A subset of edges can be assign a different thickness or color.

Usage

```
set_attributes_to_edges_visnetwork(
  edges,
  subset_edges,
  width = c(NA, NA),
  color = c(NULL, NULL)
)
```

Arguments

edges	A data.frame create by create_edges_visnetwork
subset_edges	A list with the elements from and to. Both from and to are vectors of the same length denoting the different edges
width	Vector with two values. The first is assigned to the edges in the subset given by subset_edges. The second value is assigned to the rest. If width = c(NA,NA), no width is assigned
color	Vector with two values. The first is assigned to the edges in the subset given by subset_edges. The second value is assigned to the rest. If color = c(NULL, NULL), no color is assigned

Value

A data frame that can be used by the visNetwork package

Examples

```

nodes <- create_nodes_visnetwork(n_nodes = 5, labels = LETTERS[1:5])

adj_matrix <- matrix(c(0, 1, 0, 1, 0,
                      1, 0, 1, 0, 0,
                      0, 1, 0, 0, 0,
                      1, 0, 0, 0, 1,
                      0, 0, 0, 1, 0), ncol = 5)

edges <- create_edges_visnetwork(adj_matrix)

shared_edges <- data.frame(from = c(1,2), to = c(4, 3))

edges <- set_attributes_to_edges_visnetwork(edges,
                                             subset_edges = shared_edges,
                                             width = c(3, .5),
                                             color = c("red", "blue"))

visnetwork(nodes, edges)

```

strip_cvn

*Strip CVN***Description**

Function that removes most of the items to make the CVN object more memory sufficient. This is especially important when the graphs are rather larger

Usage

```
strip_cvn(cvn)
```

Arguments

cvn Object of class 'CVN'

Value

Reduced CVN where Theta, data and Sigma are removed

updateZRcpp

The Z-update Step

Description

A C implementation of the Z -update step. We solve a generalized LASSO problem repeatedly for each of the individual edges

Usage

```
updateZRcpp(m, p, Theta, Y, W, eta1, eta2, a, rho, max_iter, eps, truncate)
```

Arguments

m	The number of graphs
p	The number of variables
Theta	A list of matrices with the Θ -matrices
Y	A list of matrices with the Y -matrices
W	The weight matrix W of dimensions $m \times m$
eta1	Equals λ_1 / ρ
eta2	Equals λ_2 / ρ
a	Value added to the diagonal of $-D'D$ so that the matrix is positive definite, see matrix_A_inner_ADMM
rho	The ADMM's parameter
max_iter	Maximum number of iterations
eps	Stopping criterion. If differences are smaller than ϵ , algorithm is halted
truncate	Values below truncate are set to 0

Value

The estimated vector $\hat{\beta}$

Author(s)

Louis Dijkstra

visnetwork	<i>A visNetwork plot</i>
------------	--------------------------

Description

Creates a visNetwork plot given a list of nodes and edges. The nodes data frame can be created with [create_nodes_visnetwork](#); the edges with [create_edges_visnetwork](#). In order to highlight edges, you can use [set_attributes_to_edges_visnetwork](#).

Usage

```
visnetwork(
  nodes,
  edges,
  node_titles = nodes$id,
  title = "",
  igraph_layout = "layout_in_circle"
)
```

Arguments

nodes	A data frame with rows equal to the number of nodes with columns "id" and "title". Can be created with create_nodes_visnetwork .
edges	A data frame with columns "from" and "to". Each row represents an edge between two nodes (integer)
node_titles	Vector with title of the nodes (Default: 1:p)
title	A list with n_lambda_values vectors. Each vector is of the length m. Regulates the titles of the graphs (Default: no title)
igraph_layout	igraph layout (default: layout_in_circle)

Value

A visNetwork plot

See Also

CVN, [create_nodes_visnetwork](#)

Examples

```
nodes <- create_nodes_visnetwork(n_nodes = 5, labels = LETTERS[1:5])

adj_matrix <- matrix(c(0, 1, 0, 1, 0,
                      1, 0, 1, 0, 0,
                      0, 1, 0, 0, 0,
                      1, 0, 0, 0, 1,
                      0, 0, 0, 1, 0), ncol = 5)
```

```

edges <- create_edges_visnetwork(adj_matrix)

shared_edges <- data.frame(from = c(1,2), to = c(4, 3))

edges <- set_attributes_to_edges_visnetwork(edges,
                                             subset_edges = shared_edges,
                                             width = c(3, .5),
                                             color = c("red", "blue"))

visnetwork(nodes, edges)

```

visnetwork_cvn	<i>All visNetwork plots for a CVN object</i>
----------------	--

Description

Creates all visNetwork plots, see [visnetwork](#), for all graphs in a cvn object

Usage

```

visnetwork_cvn(
  cvn,
  node_titles = 1:cvn$p,
  titles = lapply(1:cvn$n_lambda_values, function(i) sapply(1:cvn$m, function(j) "")),
  show_core_graph = TRUE,
  width = c(3, 1),
  color = c("red", "blue"),
  igraph_layout = "layout_in_circle",
  verbose = TRUE
)

```

Arguments

cvn	A cvn object, see CVN
node_titles	Vector with title of the nodes (Default: 1:p)
titles	A list with n_lambda_values vectors. Each vector is of the length m. Regulates the titles of the graphs (Default: no title)
show_core_graph	Shall the core graph be visualized (Default = TRUE)
width	Edge width of the core graph
color	String vector with two colors. The first color marks the edges in the core graph (Default: c("red", "blue"))
igraph_layout	igraph layout (default: layout_in_circle)
verbose	Verbose (Default: TRUE)

Value

The cvn input which is extended by the list element 'plots'

See Also

[CVN](#), [visnetwork](#)

Examples

```
## Not run:  
path <- system.file("cvnfit.rda", package = "CVN")  
load(path)  
fit_plot <- visnetwork_cvn(fit)  
fit_plot$plots[[1]][[1]]  
  
## End(Not run)
```

Index

- * **datasets**
 - grid, [15](#)
- * **graphical**
 - CVN-package, [2](#)
- * **models**
 - CVN-package, [2](#)
- combine_cvn, [3](#)
- combine_cvn_interpolated, [4](#)
- create_edges_visnetwork, [5](#), [26](#)
- create_nodes_visnetwork, [6](#), [29](#)
- create_weight_matrix, [7](#)
- CVN, [8](#), [12](#), [13](#), [16](#), [23](#), [25](#), [30](#), [31](#)
- CVN-package, [2](#)
- cvn_edge_summary, [11](#)
- determine_information_criterion_cvn,
[12](#)
- extract_cvn, [12](#)
- find_core_graph, [13](#)
- find_unique_edges, [14](#)
- genlasso_wrapper, [15](#)
- genlassoRcpp, [14](#)
- glasso, [9](#)
- grid, [15](#)
- hamming_distance, [16](#)
- hamming_distance_adj_matrices, [16](#)
- hd_weight_matrix, [17](#), [24](#)
- interpolate, [4](#), [18](#)
- Matrix, [10](#), [18](#)
- matrix_A_inner_ADMM, [15](#), [19](#), [28](#)
- plot.cvn, [20](#)
- plot_hamming_distances, [21](#)
- plot_hamming_distances_cvn, [22](#)
- plot_information_criterion, [23](#)
- plot_weight_matrix, [17](#), [24](#)
- print.cvn, [25](#)
- relative_difference_precision_matrices,
[25](#)
- set_attributes_to_edges_visnetwork, [26](#),
[29](#)
- strip_cvn, [27](#)
- updateZRcpp, [28](#)
- visnetwork, [29](#), [30](#), [31](#)
- visnetwork_cvn, [20](#), [30](#)