

# Package ‘covdepGE’

July 22, 2025

**Title** Covariate Dependent Graph Estimation

**Version** 1.0.1

**Date** 2022-09-16

**Language** en-US

**BugReports** <https://github.com/JacobHelwig/covdepGE/issues>

**URL** <https://github.com/JacobHelwig/covdepGE>

**Description** A covariate-dependent approach to Gaussian graphical modeling as described in Dasgupta et al. (2022). Employs a novel weighted pseudo-likelihood approach to model the conditional dependence structure of data as a continuous function of an extraneous covariate. The main function, `covdepGE::covdepGE()`, estimates a graphical representation of the conditional dependence structure via a block mean-field variational approximation, while several auxiliary functions (`inclusionCurve()`, `matViz()`, and `plot.covdepGE()`) are included for visualizing the resulting estimates.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**LinkingTo** Rcpp, RcppArmadillo

**Imports** doParallel, foreach, ggplot2, glmnet, latex2exp, MASS,  
parallel, Rcpp, reshape2, stats

**Suggests** testthat (>= 3.0.0), covr, vdiff

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Jacob Helwig [cre, aut],  
Sutanoy Dasgupta [aut],  
Peng Zhao [aut],  
Bani Mallick [aut],  
Debdeep Pati [aut]

**Maintainer** Jacob Helwig <jacob.a.helwig@tamu.edu>

**Repository** CRAN

**Date/Publication** 2022-09-16 15:56:08 UTC

## Contents

covdepGE-package . . . . .	2
covdepGE . . . . .	3
generateData . . . . .	8
inclusionCurve . . . . .	11
matViz . . . . .	13
plot.covdepGE . . . . .	15
<b>Index</b>	<b>17</b>

---

covdepGE-package	<i>covdepGE: Covariate Dependent Graph Estimation</i>
------------------	---

---

## Description

A covariate-dependent approach to Gaussian graphical modeling as described in Dasgupta et al. (2022). Employs a novel weighted pseudo-likelihood approach to model the conditional dependence structure of data as a continuous function of an extraneous covariate. The main function, `covdepGE::covdepGE()`, estimates a graphical representation of the conditional dependence structure via a block mean-field variational approximation, while several auxiliary functions (`inclusionCurve()`, `matViz()`, and `plot.covdepGE()`) are included for visualizing the resulting estimates.

## Author(s)

**Maintainer:** Jacob Helwig <jacob.a.helwig@tamu.edu>

Authors:

- Sutanoy Dasgupta <sutanoy@stat.tamu.edu>
- Peng Zhao <pzhao@stat.tamu.edu>
- Bani Mallick <bmallick@stat.tamu.edu>
- Debdeep Pati <debdeep@stat.tamu.edu>

## References

(1) Sutanoy Dasgupta, Peng Zhao, Prasenjit Ghosh, Debdeep Pati, and Bani Mallick. An approximate Bayesian approach to covariate-dependent graphical modeling, pages 1–59, 2022.

## See Also

Useful links:

- <https://github.com/JacobHelwig/covdepGE>
- Report bugs at <https://github.com/JacobHelwig/covdepGE/issues>

---

 covdepGE

*Covariate Dependent Graph Estimation*


---

### Description

Model the conditional dependence structure of  $X$  as a function of  $Z$  as described in (1)

### Usage

```

covdepGE(
  X,
  Z = NULL,
  hp_method = "hybrid",
  ssq = NULL,
  sbsq = NULL,
  pip = NULL,
  nssq = 5,
  nsbsq = 5,
  npip = 5,
  ssq_mult = 1.5,
  ssq_lower = 1e-05,
  snr_upper = 25,
  sbsq_lower = 1e-05,
  pip_lower = 1e-05,
  pip_upper = NULL,
  tau = NULL,
  norm = 2,
  center_X = TRUE,
  scale_Z = TRUE,
  alpha_tol = 1e-05,
  max_iter_grid = 10,
  max_iter = 100,
  edge_threshold = 0.5,
  sym_method = "mean",
  parallel = FALSE,
  num_workers = NULL,
  prog_bar = TRUE
)

```

### Arguments

$X$	$n \times p$ numeric matrix; data matrix. For best results, $n$ should be greater than $p$
$Z$	NULL OR $n \times q$ numeric matrix; extraneous covariates. If NULL, $Z$ will be treated as constant for all observations, i.e.:  $Z \leftarrow \text{rep}(0, \text{nrow}(X))$

If  $Z$  is constant, the estimated graph will be homogeneous throughout the data.  
NULL by default

hp_method	<p>character in <code>c("grid_search", "model_average", "hybrid")</code>; method for selecting hyperparameters from the the hyperparameter grid. The grid will be generated as the Cartesian product of <code>ssq</code>, <code>sbsq</code>, and <code>pip</code>. Fix <math>X_j</math>, the <math>j</math>-th column of <math>X</math>, as the response; then, the hyperparameters will be selected as follows:</p> <ul style="list-style-type: none"> <li>• If "grid_search", the point in the hyperparameter grid that maximizes the total ELBO summed across all <math>n</math> regressions will be selected</li> <li>• If "model_average", then all posterior quantities will be an average of the variational estimates resulting from the model fit for each point in the hyperparameter grid. The unnormalized averaging weights for each of the <math>n</math> regressions are the exponentiated ELBO</li> <li>• If "hybrid", then models will be averaged over <code>pip</code> as in "model_average", with <math>\sigma^2</math> and <math>\sigma_\beta^2</math> chosen for each <math>\pi</math> in <code>pip</code> by maximizing the total ELBO over the grid defined by the Cartesian product of <code>ssq</code> and <code>sbsq</code> as in "grid_search"</li> </ul> <p>"hybrid" by default</p>
ssq	<p>NULL OR numeric vector with positive entries; candidate values of the hyperparameter <math>\sigma^2</math> (prior residual variance). If NULL, <code>ssq</code> will be generated for each variable <math>X_j</math> fixed as the response as:</p> <pre>ssq &lt;- seq(ssq_lower, ssq_upper, length.out = nssq)</pre> <p>NULL by default</p>
sbsq	<p>NULL OR numeric vector with positive entries; candidate values of the hyperparameter <math>\sigma_\beta^2</math> (prior slab variance). If NULL, <code>sbsq</code> will be generated for each variable <math>X_j</math> fixed as the response as:</p> <pre>sbsq &lt;- seq(sbsq_lower, sbsq_upper, length.out = nsbsq)</pre> <p>NULL by default</p>
pip	<p>NULL OR numeric vector with entries in <math>(0, 1)</math>; candidate values of the hyperparameter <math>\pi</math> (prior inclusion probability). If NULL, <code>pip</code> will be generated for each variable <math>X_j</math> fixed as the response as:</p> <pre>pip &lt;- seq(pip_lower, pi_upper, length.out = npip)</pre> <p>NULL by default</p>
nssq	positive integer; number of points to generate for <code>ssq</code> if <code>ssq</code> is NULL. 5 by default
nsbsq	positive integer; number of points to generate for <code>sbsq</code> if <code>sbsq</code> is NULL. 5 by default
npip	positive integer; number of points to generate for <code>pip</code> if <code>pip</code> is NULL. 5 by default
ssq_mult	<p>positive numeric; if <code>ssq</code> is NULL, then for each variable <math>X_j</math> fixed as the response:</p> <pre>ssq_upper &lt;- ssq_mult * stats::var(X_j)</pre> <p>Then, <code>ssq_upper</code> will be the greatest value in <code>ssq</code> for variable <math>X_j</math>. 1.5 by default</p>

ssq_lower	positive numeric; if ssq is NULL, then ssq_lower will be the least value in ssq. 1e-5 by default
snr_upper	positive numeric; upper bound on the signal-to-noise ratio. If sbsq is NULL, then for each variable $X_j$ fixed as the response:  <pre>s2_sum &lt;- sum(apply(X, 2, stats::var)) sbsq_upper &lt;- snr_upper / (pip_upper * s2_sum)</pre> Then, sbsq_upper will be the greatest value in sbsq. 25 by default
sbsq_lower	positive numeric; if sbsq is NULL, then sbsq_lower will be the least value in sbsq. 1e-5 by default
pip_lower	numeric in (0, 1); if pip is NULL, then pip_lower will be the least value in pip. 1e-5 by default
pip_upper	NULL OR numeric in (0, 1); if pip is NULL, then pip_upper will be the greatest value in pip. If sbsq is NULL, pip_upper will be used to calculate sbsq_upper. If NULL, pip_upper will be calculated for each variable $X_j$ fixed as the response as:  <pre>lasso &lt;- glmnet::cv.glmnet(X, X_j) non0 &lt;- sum(glmnet::coef.glmnet(lasso, s = "lambda.1se")[-1] != 0) non0 &lt;- min(max(non0, 1), p - 1) pip_upper &lt;- non0 / p</pre> NULL by default
tau	NULL OR positive numeric OR numeric vector of length $n$ with positive entries; bandwidth parameter. Greater values allow for more information to be shared between observations. Allows for global or observation-specific specification. If NULL, use 2-step KDE methodology as described in (2) to calculate observation-specific bandwidths. NULL by default
norm	numeric in $[1, \infty]$ ; norm to use when calculating weights. Inf results in infinity norm. 2 by default
center_X	logical; if TRUE, center X column-wise to mean 0. TRUE by default
scale_Z	logical; if TRUE, center and scale Z column-wise to mean 0, standard deviation 1 prior to calculating the weights. TRUE by default
alpha_tol	positive numeric; end CAVI when the Frobenius norm of the change in the alpha matrix is within alpha_tol. 1e-5 by default
max_iter_grid	positive integer; if tolerance criteria has not been met by max_iter_grid iterations during grid search, end CAVI. After grid search has completed, CAVI is performed with the final hyperparameters selected by grid search for at most max_iter iterations. Does not apply to hp_method = "model_average". 10 by default
max_iter	positive integer; if tolerance criteria has not been met by max_iter iterations, end CAVI. 100 by default
edge_threshold	numeric in (0, 1); a graph for each observation will be constructed by including an edge between variable $i$ and variable $j$ if, and only if, the $(i, j)$ entry of the symmetrized posterior inclusion probability matrix corresponding to the observation is greater than edge_threshold. 0.5 by default

sym_method	character in c("mean", "max", "min"); to symmetrize the posterior inclusion probability matrix for each observation, the $(i, j)$ and $(j, i)$ entries will be post-processed as sym_method applied to the $(i, j)$ and $(j, i)$ entries. "mean" by default
parallel	logical; if TRUE, hyperparameter selection and CAVI for each of the $p$ variables will be performed in parallel using foreach. Parallel backend may be registered prior to making a call to covdepGE. If no active parallel backend can be detected, then parallel backend will be automatically registered using:  doParallel::registerDoParallel(num_workers)  FALSE by default
num_workers	NULL OR positive integer less than or equal to parallel::detectCores(); argument to doParallel::registerDoParallel if parallel = TRUE and no parallel backend is detected. If NULL, then:  num_workers <- floor(parallel::detectCores() / 2)  NULL by default
prog_bar	logical; if TRUE, then a progress bar will be displayed denoting the number of remaining variables to fix as the response and perform CAVI. If parallel, no progress bar will be displayed. TRUE by default

### Value

Returns object of class covdepGE with the following values:

graphs	list with the following values: <ul style="list-style-type: none"> <li>• graphs: list of <math>n</math> numeric matrices of dimension <math>p \times p</math>; the <math>l</math>-th matrix is the adjacency matrix for the <math>l</math>-th observation</li> <li>• unique_graphs: list; the <math>l</math>-th element is a list containing the <math>l</math>-th unique graph and the indices of the observation(s) corresponding to this graph</li> <li>• inclusion_probs_sym: list of <math>n</math> numeric matrices of dimension <math>p \times p</math>; the <math>l</math>-th matrix is the symmetrized posterior inclusion probability matrix for the <math>l</math>-th observation</li> <li>• inclusion_probs_asym: list of <math>n</math> numeric matrices of dimension <math>p \times p</math>; the <math>l</math>-th matrix is the posterior inclusion probability matrix for the <math>l</math>-th observation prior to symmetrization</li> </ul>
variational_params	list with the following values: <ul style="list-style-type: none"> <li>• alpha: list of <math>p</math> numeric matrices of dimension <math>n \times (p - 1)</math>; the <math>(i, j)</math> entry of the <math>k</math>-th matrix is the variational approximation to the posterior inclusion probability of the <math>j</math>-th variable in a weighted regression with variable <math>k</math> fixed as the response, where the weights are taken with respect to observation <math>i</math></li> <li>• mu: list of <math>p</math> numeric matrices of dimension <math>n \times (p - 1)</math>; the <math>(i, j)</math> entry of the <math>k</math>-th matrix is the variational approximation to the posterior slab mean for the <math>j</math>-th variable in a weighted regression with variable <math>k</math> fixed as the response, where the weights are taken with respect to observation <math>i</math></li> </ul>

- `ssq_var`: list of  $p$  numeric matrices of dimension  $n \times (p - 1)$ ; the  $(i, j)$  entry of the  $k$ -th matrix is the variational approximation to the posterior slab variance for the  $j$ -th variable in a weighted regression with variable  $k$  fixed as the response, where the weights are taken with respect to observation  $i$

#### hyperparameters

list of  $p$  lists; the  $j$ -th list has the following values for variable  $j$  fixed as the response:

- `grid`: matrix of candidate hyperparameter values, corresponding ELBO, and iterations to converge
- `final`: the final hyperparameters chosen by grid search and the ELBO and iterations to converge for these hyperparameters

#### model\_details

list with the following values:

- `elapsed`: amount of time to fit the model
- `n`: number of observations
- `p`: number of variables
- `ELBO`: ELBO summed across all observations and variables. If `hp_method` is "model\_average" or "hybrid", this ELBO is averaged across the hyperparameter grid using the model averaging weights for each variable
- `num_unique`: number of unique graphs
- `grid_size`: number of points in the hyperparameter grid
- `args`: list containing all passed arguments of length 1

#### weights

list with the following values:

- `weights`:  $n \times n$  numeric matrix. The  $(i, j)$  entry is the similarity weight of the  $i$ -th observation with respect to the  $j$ -th observation using the  $j$ -th observation's bandwidth
- `bandwidths`: numeric vector of length  $n$ . The  $i$ -th entry is the bandwidth for the  $i$ -th observation

## References

- (1) Sutanoy Dasgupta, Peng Zhao, Prasenjit Ghosh, Debdeep Pati, and Bani Mallick. An approximate Bayesian approach to covariate-dependent graphical modeling. pages 1–59, 2022.
- (2) Sutanoy Dasgupta, Debdeep Pati, and Anuj Srivastava. A Two-Step Geometric Framework For Density Modeling. *Statistica Sinica*, 30(4):2155–2177, 2020.

## Examples

```
## Not run:
library(ggplot2)

# get the data
set.seed(12)
data <- generateData()
X <- data$X
Z <- data$Z
interval <- data$interval
```

```

prec <- data$true_precision

# get overall and within interval sample sizes
n <- nrow(X)
n1 <- sum(interval == 1)
n2 <- sum(interval == 2)
n3 <- sum(interval == 3)

# visualize the distribution of the extraneous covariate
ggplot(data.frame(Z = Z, interval = as.factor(interval))) +
  geom_histogram(aes(Z, fill = interval), color = "black", bins = n %% 5)

# visualize the true precision matrices in each of the intervals

# interval 1
matViz(prec[[1]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 1, observations 1,...,", n1))

# interval 2 (varies continuously with Z)
cat("\nInterval 2, observations ", n1 + 1, ",...,", n1 + n2, sep = "")
int2_mats <- prec[interval == 2]
int2_inds <- c(5, n2 %% 2, n2 - 5)
lapply(int2_inds, function(j) matViz(int2_mats[[j]], incl_val = TRUE) +
  ggtitle(paste("True precision matrix, interval 2, observation", j + n1)))

# interval 3
matViz(prec[[length(prec)]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 3, observations ",
    n1 + n2 + 1, ",...,", n1 + n2 + n3))

# fit the model and visualize the estimated graphs
(out <- covdepGE(X, Z))
plot(out)

# visualize the posterior inclusion probabilities for variables (1, 3) and (1, 2)
inclusionCurve(out, 1, 2)
inclusionCurve(out, 1, 3)

## End(Not run)

```

---

generateData

*Generate Covariate-Dependent Data*


---

### Description

Generate a 1-dimensional extraneous covariate and  $p$ -dimensional Gaussian data with a precision matrix that varies as a continuous function of the extraneous covariate. This data is distributed similar to that used in the simulation study from (1)



**Usage**

```
generateData(p = 5, n1 = 60, n2 = 60, n3 = 60, Z = NULL, true_precision = NULL)
```

**Arguments**

<code>p</code>	positive integer; number of variables in the data matrix. 5 by default
<code>n1</code>	positive integer; number of observations in the first interval. 60 by default
<code>n2</code>	positive integer; number of observations in the second interval. 60 by default
<code>n3</code>	positive integer; number of observations in the third interval. 60 by default
<code>Z</code>	NULL or numeric vector; extraneous covariate values for each observation. If NULL, Z will be generated from a uniform distribution on each of the intervals
<code>true_precision</code>	NULL OR list of matrices of dimension $p \times p$ ; true precision matrix for each observation. If NULL, the true precision matrices will be generated dependent on Z. NULL by default

**Value**

Returns list with the following values:

<code>X</code>	a $(n1 + n2 + n3) \times p$ numeric matrix, where the $i$ -th row is drawn from a $p$ -dimensional Gaussian with mean 0 and precision matrix <code>true_precision[[i]]</code>
<code>Z</code>	a $(n1 + n2 + n3) \times 1$ numeric matrix, where the $i$ -th entry is the extraneous covariate $z_i$ for observation $i$
<code>true_precision</code>	list of $n1 + n2 + n3$ matrices of dimension $p \times p$ ; the $i$ -th matrix is the precision matrix for the $i$ -th observation
<code>interval</code>	vector of length $n1 + n2 + n3$ ; interval assignments for each of the observations, where the $i$ -th entry is the interval assignment for the $i$ -th observation

**Extraneous Covariate**

If `Z = NULL`, then the generation of Z is as follows:

The first  $n1$  observations have  $z_i$  from from a uniform distribution on the interval  $(-3, -1)$  (the first interval).

Observations  $n1 + 1$  to  $n1 + n2$  have  $z_i$  from from a uniform distribution on the interval  $(-1, 1)$  (the second interval).

Observations  $n1 + n2 + 1$  to  $n1 + n2 + n3$  have  $z_i$  from a uniform distribution on the interval  $(1, 3)$  (the third interval).

**Precision Matrices**

If `true_precision = NULL`, then the generation of the true precision matrices is as follows:

All precision matrices have 2 on the diagonal and 1 in the  $(2, 3)/(3, 2)$  positions.

Observations in the first interval have a 1 in the  $(1, 2)/(1, 2)$  positions, while observations in the third interval have a 1 in the  $(1, 3)/(3, 1)$  positions.

Observations in the second interval have 2 entries that vary as a linear function of their extraneous covariate. Let  $\beta = 1/2$ . Then, the (1, 2)/(2, 1) positions for the  $i$ -th observation in the second interval are  $\beta \cdot (1 - z_i)$ , while the (1, 3)/(3, 1) entries are  $\beta \cdot (1 + z_i)$ .

Thus, as  $z_i$  approaches  $-1$  from the right, the associated precision matrix becomes more similar to the matrix for observations in the first interval. Similarly, as  $z_i$  approaches  $1$  from the left, the matrix becomes more similar to the matrix for observations in the third interval.

## Examples

```
## Not run:
library(ggplot2)

# get the data
set.seed(12)
data <- generateData()
X <- data$X
Z <- data$Z
interval <- data$interval
prec <- data$true_precision

# get overall and within interval sample sizes
n <- nrow(X)
n1 <- sum(interval == 1)
n2 <- sum(interval == 2)
n3 <- sum(interval == 3)

# visualize the distribution of the extraneous covariate
ggplot(data.frame(Z = Z, interval = as.factor(interval))) +
  geom_histogram(aes(Z, fill = interval), color = "black", bins = n %% 5)

# visualize the true precision matrices in each of the intervals

# interval 1
matViz(prec[[1]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 1, observations 1,...,", n1))

# interval 2 (varies continuously with Z)
cat("\nInterval 2, observations ", n1 + 1, ", ..., ", n1 + n2, sep = "")
int2_mats <- prec[interval == 2]
int2_inds <- c(5, n2 %% 2, n2 - 5)
lapply(int2_inds, function(j) matViz(int2_mats[[j]], incl_val = TRUE) +
  ggtitle(paste("True precision matrix, interval 2, observation", j + n1)))

# interval 3
matViz(prec[[length(prec)]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 3, observations ",
    n1 + n2 + 1, ", ..., ", n1 + n2 + n3))

# fit the model and visualize the estimated graphs
(out <- covdepGE(X, Z))
plot(out)
```

```
# visualize the posterior inclusion probabilities for variables (1, 3) and (1, 2)
inclusionCurve(out, 1, 2)
inclusionCurve(out, 1, 3)

## End(Not run)
```

---

inclusionCurve	<i>Plot PIP as a Function of Index</i>
----------------	--

---

### Description

Plot the posterior inclusion probability of an edge between two variables as a function of observation index

### Usage

```
inclusionCurve(
  out,
  col_idx1,
  col_idx2,
  line_type = "solid",
  line_size = 0.5,
  line_color = "black",
  point_shape = 21,
  point_size = 1.5,
  point_color = "#500000",
  point_fill = "white"
)
```

### Arguments

out	object of class covdepGE; return of covdepGE function
col_idx1	integer in $[1, p]$ ; column index of the first variable
col_idx2	integer in $[1, p]$ ; column index of the second variable
line_type	linetype; ggplot2 line type to interpolate the probabilities. "solid" by default
line_size	positive numeric; thickness of the interpolating line. 0.5 by default
line_color	color; color of interpolating line. "black" by default
point_shape	shape; shape of the points denoting observation-specific inclusion probabilities; 21 by default
point_size	positive numeric; size of probability points. 1.5 by default
point_color	color; color of probability points. "#500000" by default
point_fill	color; fill of probability points. Only applies to select shapes. "white" by default

**Value**

Returns ggplot2 visualization of inclusion probability curve

**Examples**

```
## Not run:
library(ggplot2)

# get the data
set.seed(12)
data <- generateData()
X <- data$X
Z <- data$Z
interval <- data$interval
prec <- data$true_precision

# get overall and within interval sample sizes
n <- nrow(X)
n1 <- sum(interval == 1)
n2 <- sum(interval == 2)
n3 <- sum(interval == 3)

# visualize the distribution of the extraneous covariate
ggplot(data.frame(Z = Z, interval = as.factor(interval))) +
  geom_histogram(aes(Z, fill = interval), color = "black", bins = n %% 5)

# visualize the true precision matrices in each of the intervals

# interval 1
matViz(prec[[1]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 1, observations 1,...,", n1))

# interval 2 (varies continuously with Z)
cat("\nInterval 2, observations ", n1 + 1, ", ..., ", n1 + n2, sep = "")
int2_mats <- prec[interval == 2]
int2_inds <- c(5, n2 %% 2, n2 - 5)
lapply(int2_inds, function(j) matViz(int2_mats[[j]], incl_val = TRUE) +
  ggtitle(paste("True precision matrix, interval 2, observation", j + n1)))

# interval 3
matViz(prec[[length(prec)]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 3, observations ",
    n1 + n2 + 1, ", ..., ", n1 + n2 + n3))

# fit the model and visualize the estimated graphs
(out <- covdepGE(X, Z))
plot(out)

# visualize the posterior inclusion probabilities for variables (1, 3) and (1, 2)
inclusionCurve(out, 1, 2)
inclusionCurve(out, 1, 3)
```

```
## End(Not run)
```

---

```
matViz
```

```
Visualize a matrix
```

---

## Description

Create a visualization of a matrix

## Usage

```
matViz(
  x,
  color1 = "white",
  color2 = "#500000",
  grid_color = "black",
  incl_val = FALSE,
  prec = 2,
  font_size = 3,
  font_color1 = "black",
  font_color2 = "white",
  font_thres = mean(x)
)
```

## Arguments

x	matrix; matrix to be visualized
color1	color; color for low entries. "white" by default
color2	color; color for high entries. "#500000" by default
grid_color	color; color of grid lines. "black" by default
incl_val	logical; if TRUE, the value for each entry will be displayed. FALSE by default
prec	positive integer; number of decimal places to round entries to if incl_val is TRUE. 2 by default
font_size	positive numeric; size of font if incl_val is TRUE. 3 by default
font_color1	color; color of font for low entries if incl_val is TRUE. "black" by default
font_color2	color; color of font for high entries if incl_val is TRUE. "white" by default
font_thres	numeric; values less than font_thres will be displayed in font_color1 if incl_val is TRUE. mean(x) by default

## Value

Returns ggplot2 visualization of matrix

**Examples**

```

## Not run:
library(ggplot2)

# get the data
set.seed(12)
data <- generateData()
X <- data$X
Z <- data$Z
interval <- data$interval
prec <- data$true_precision

# get overall and within interval sample sizes
n <- nrow(X)
n1 <- sum(interval == 1)
n2 <- sum(interval == 2)
n3 <- sum(interval == 3)

# visualize the distribution of the extraneous covariate
ggplot(data.frame(Z = Z, interval = as.factor(interval))) +
  geom_histogram(aes(Z, fill = interval), color = "black", bins = n %% 5)

# visualize the true precision matrices in each of the intervals

# interval 1
matViz(prec[[1]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 1, observations 1,...,", n1))

# interval 2 (varies continuously with Z)
cat("\nInterval 2, observations ", n1 + 1, ", ..., ", n1 + n2, sep = "")
int2_mats <- prec[interval == 2]
int2_inds <- c(5, n2 %% 2, n2 - 5)
lapply(int2_inds, function(j) matViz(int2_mats[[j]], incl_val = TRUE) +
  ggtitle(paste("True precision matrix, interval 2, observation", j + n1)))

# interval 3
matViz(prec[[length(prec)]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 3, observations ",
    n1 + n2 + 1, ", ..., ", n1 + n2 + n3))

# fit the model and visualize the estimated graphs
(out <- covdepGE(X, Z))
plot(out)

# visualize the posterior inclusion probabilities for variables (1, 3) and (1, 2)
inclusionCurve(out, 1, 2)
inclusionCurve(out, 1, 3)

## End(Not run)

```

---

plot.covdepGE	<i>Plot the Graphs Estimated by covdepGE</i>
---------------	--

---

## Description

Create a list of the unique graphs estimated by covdepGE

## Usage

```
## S3 method for class 'covdepGE'  
plot(x, graph_colors = NULL, title_sum = TRUE, ...)
```

## Arguments

x	object of class covdepGE; return of covdepGE function
graph_colors	NULL OR vector; the $j$ -th element is the color for the $j$ -th graph. If NULL, all graphs will be colored with "#500000". NULL by default
title_sum	logical; if TRUE the indices of the observations corresponding to the graph will be included in the title. TRUE by default
...	additional arguments will be ignored

## Value

Returns list of ggplot2 visualizations of unique graphs estimated by covdepGE

## Examples

```
## Not run:  
library(ggplot2)  
  
# get the data  
set.seed(12)  
data <- generateData()  
X <- data$X  
Z <- data$Z  
interval <- data$interval  
prec <- data$true_precision  
  
# get overall and within interval sample sizes  
n <- nrow(X)  
n1 <- sum(interval == 1)  
n2 <- sum(interval == 2)  
n3 <- sum(interval == 3)  
  
# visualize the distribution of the extraneous covariate  
ggplot(data.frame(Z = Z, interval = as.factor(interval))) +  
  geom_histogram(aes(Z, fill = interval), color = "black", bins = n %% 5)
```

```
# visualize the true precision matrices in each of the intervals

# interval 1
matViz(prec[[1]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 1, observations 1,...,", n1))

# interval 2 (varies continuously with Z)
cat("\nInterval 2, observations ", n1 + 1, ",...,", n1 + n2, sep = "")
int2_mats <- prec[interval == 2]
int2_inds <- c(5, n2 %% 2, n2 - 5)
lapply(int2_inds, function(j) matViz(int2_mats[[j]], incl_val = TRUE) +
  ggtitle(paste("True precision matrix, interval 2, observation", j + n1)))

# interval 3
matViz(prec[[length(prec)]], incl_val = TRUE) +
  ggtitle(paste0("True precision matrix, interval 3, observations ",
    n1 + n2 + 1, ",...,", n1 + n2 + n3))

# fit the model and visualize the estimated graphs
(out <- covdepGE(X, Z))
plot(out)

# visualize the posterior inclusion probabilities for variables (1, 3) and (1, 2)
inclusionCurve(out, 1, 2)
inclusionCurve(out, 1, 3)

## End(Not run)
```



# Index

`_PACKAGE` (covdepGE-package), [2](#)

`covdepGE`, [3](#)

`covdepGE-method` (covdepGE), [3](#)

`covdepGE-package`, [2](#)

`generateData`, [8](#)

`inclusionCurve`, [11](#)

`matViz`, [13](#)

`plot.covdepGE`, [15](#)