Package 'prCGGM'

November 25, 2026

Type Package

Title Prior-Informed Conditional Gaussian Graphical Models

Version 0.2.0

Description Estimates Gaussian graphical models whose mean and precision matrix depend linearly on sample-level covariates, while integrating prior knowledge on the network structure. A typical application is the reconstruction of personalised protein-protein interaction networks from proteomics data, with clinical or demographic variables as covariates and interaction confidence scores from the 'STRING' database as prior knowledge. Prior knowledge enters through a weighted sparse-group lasso penalty that is applied only to the population-level network, so that covariate-specific changes remain entirely data-driven. Node-wise regressions are fitted with 'glmnet' and 'sparsegl'; the penalty and the structural hyperparameters are selected per node by the Bayesian information criterion of the least-squares refit (or, optionally, by its extended version, cross-validation or stability selection), the selected edges are refitted by least squares, and personalised networks can be predicted for new individuals from their covariates. Estimation runs sequentially, in parallel on a single machine, or as 'SLURM' array jobs on computing clusters. The method is described in Mapelli et al. (2026) <doi:10.48550/arXiv.2606.31805>.

License GPL-3

URL https://github.com/AlessiaMapelli/prCGGM

BugReports https://github.com/AlessiaMapelli/prCGGM/issues

Encoding UTF-8

Language en-GB

LazyData true

Depends R (>= 4.1.0)

Imports glmnet, graphics, grDevices, parallel, sparsegl, stats, utils

Suggests corrplot, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Alessia Mapelli [aut, cre],
Francesca Ieva [ths],
Politecnico di Milano [cph]

Maintainer Alessia Mapelli <alessia.mapelli@polimi.it>

Repository CRAN

Date/Publication 2026-11-25 17:50:02 UTC

Contents

compare_networks 2
plot.prcggm 3
plot_network 4
plot_network_difference 5
prcggm 5
prcggm_example 10
prcggm_slurm_prepare 11
predict.prcggm 15
simulate_prcggm_data 16
summary.prcggm 18

Index **20**

`compare_networks` *Compare two networks*

Description

Computes a summary measure of the difference between two adjacency matrices.

Usage

```
compare_networks(
  network1,
  network2,
  method = c("difference", "correlation", "frobenius")
)
```

Arguments

network1, network2	Square numeric matrices of the same size.
method	"difference" (default) for the mean absolute difference, "correlation" for the Pearson correlation of the upper-triangular entries, or "frobenius" for the Frobenius norm of the difference.

Value

A single number.

Examples

```
a <- matrix(c(0, 0.4, 0, 0.4, 0, -0.3, 0, -0.3, 0), 3, 3)
b <- matrix(c(0, 0.1, 0, 0.1, 0, -0.1, 0, -0.1, 0), 3, 3)
compare_networks(a, b, method = "frobenius")
```

---

plot.prcggm *Plot a network component of a fitted model*

Description

Draws one network component of a fitted model, or the personalised network of one individual, as a heatmap with plot_network().

Usage

```
## S3 method for class 'prcggm'
plot(x, component = "Baseline", newdata = NULL, ...)
```

Arguments

x	A "prcggm" object returned by prcggm().
component	Name of the component to draw: "Baseline" (default) or one of the covariate effects in names(x$networks). Ignored if newdata is supplied.
newdata	Optional one-row data frame of covariates; if supplied, the personalised network of that individual is drawn.
...	Further arguments passed to plot_network(), such as threshold or main.

Value

Called for its side effect; returns x invisibly.

Examples

```
data(prcggm_example)
set.seed(1)
fit <- prcggm(prcggm_example$x,
              covariates = prcggm_example$covariates,
              prior = prcggm_example$prior,
              tune = FALSE)
plot(fit)
plot(fit, component = "x1_1")
```

`plot_network` *Plot a network as a heatmap*

Description

Draws an adjacency matrix (for example a component of a fitted model or a personalised network from `predict.prcggm()`) as a heatmap. The **corrplot** package is used if installed, otherwise `stats::heatmap()`.

Usage

```
plot_network(network, labels = NULL, threshold = 0, main = "Network")
```

Arguments

`network`	Square numeric matrix.
`labels`	Optional node labels. By default the row or column names of `network`, or `Node_1`, `Node_2`, and so on.
`threshold`	Entries with absolute value below `threshold` are set to zero before plotting. Default 0.
`main`	Plot title.

Value

Called for its side effect; returns the plotted matrix invisibly.

Examples

```
net <- matrix(c(0, 0.4, 0, 0.4, 0, -0.3, 0, -0.3, 0), 3, 3)
plot_network(net, labels = c("A", "B", "C"))
```

plot_network_difference

Plot the difference between two networks

Description

Draws `network1 - network2` with `plot_network()`, for example to compare the personalised networks of two individuals.

Usage

```
plot_network_difference(
  network1,
  network2,
  labels = NULL,
  main = "Network difference"
)
```

Arguments

`network1`, `network2`	Square numeric matrices of the same size.
`labels`	Optional node labels, as in `plot_network()`.
`main`	Plot title.

Value

Called for its side effect; returns the difference matrix invisibly.

Examples

```
a <- matrix(c(0, 0.4, 0.4, 0), 2, 2)
b <- matrix(c(0, 0.1, 0.1, 0), 2, 2)
plot_network_difference(a, b)
```

prcggm

Fit a prior-informed conditional Gaussian graphical model

Description

Estimates a Gaussian graphical model in which both the mean and the precision matrix depend linearly on subject-level covariates, while prior knowledge on the network (for example 'STRING' interaction confidence scores) reduces the penalty on supported edges of the population-level network.

Usage

```
prcggm(
  x,
  covariates = NULL,
  prior = NULL,
  screening = FALSE,
  gamma = NULL,
  screening_matrix = NULL,
  mean_estimation = TRUE,
  lambda_mean = NULL,
  lambda_mean_type = c("min", "1se"),
  lambda_prec = NULL,
 selection = c("bic_relaxed", "ebic_relaxed", "bic", "ebic", "cv_min", "cv_1se",
    "stability_bic", "stability_cv"),
  refit = TRUE,
  selection_control = list(),
  tune = TRUE,
  alpha_grid = c(0.75, 0.5, 0.9, 0.99),
  theta_grid = c(1, 0.8, 1.1, 1.3),
  random_search = FALSE,
  p_random = 0.5,
  nfolds = 5,
  symm_method = c("AND", "OR"),
  ncores = 1L,
  verbose = FALSE,
  lambda_prec_type = NULL
)
```

Arguments

x	Numeric $n \times p$ matrix or data frame of expression data, with the n subjects in rows and the p features (network nodes) in columns. Missing values are not allowed.
covariates	Optional data frame of subject-level covariates with n rows, in the same order as the rows of x. Columns can be numeric, factor, character or logical; they are encoded into q columns as described in *Covariate encoding*. If NULL (default), a single population network is estimated ($q = 0$).
prior	Optional symmetric $p \times p$ matrix of prior edge confidence in [0, 1], with rows and columns in the same order as the columns of x. Edges with entries close to 1 are penalised less in the baseline network. If NULL (default), no prior knowledge is used.
screening	Logical; if TRUE, predictors weakly correlated with the response node (absolute correlation of the residuals not above gamma) are excluded from each node-wise regression, unless they have prior support. Default FALSE.
gamma	Correlation threshold in [0, 1] for screening. If NULL (default), the 20th percentile of the $p(p-1)/2$ absolute pairwise correlations is used.

`screening_matrix`
: Optional $p \times p$ matrix with entries in $\{0, 1\}$; entry $(j, k) = 0$ excludes feature k from the regression of node j. When supplied, it is used for screening instead of the correlation rule. Prior-supported pairs are never excluded.

`mean_estimation`
: Logical; if `TRUE` (default) the covariate effects on the mean are estimated and removed first. Set to `FALSE` only if `x` has already been centred conditionally on the covariates.

`lambda_mean`
: Optional fixed lasso penalty for the mean regressions. If `NULL` (default), it is selected by cross-validation with `glmnet::cv.glmnet()`.

`lambda_mean_type`
: Cross-validation rule used to select `lambda_mean`, with the same meaning as in `glmnet::cv.glmnet()`: `"min"` (default) selects `lambda.min`, the value that minimises the cross-validated error; `"1se"` selects `lambda.1se`, the largest value whose error is within one standard error of the minimum, which gives sparser models. Ignored if `lambda_mean` is supplied or if $q = 1$, in which case the mean is estimated by least squares.

`lambda_prec`
: Optional fixed penalty for the node-wise network regressions. If `NULL` (default), it is selected with the rule given by `selection`.

`selection`
: Rule used to select the penalty of the node-wise network regressions and the hyperparameters: `"bic_relaxed"` (default), `"ebic_relaxed"`, `"bic"`, `"ebic"`, `"cv_min"`, `"cv_1se"`, `"stability_bic"` or `"stability_cv"`. See *Selection rules*.

`refit`
: Logical; if `TRUE` (default), the predictors selected for each node are refitted by least squares, so that the edge weights are not shrunk. The selected edges do not change. Always `TRUE` for stability selection.

`selection_control`
: List of settings of the selection rules, any of `ebic_gamma` (EBIC parameter, default 0.5), `stab_B` (number of subsamples, default 50), `stab_fraction` (subsample size as a fraction of n, default 0.5), `stab_pi` (selection-frequency threshold, default 0.75) and `stab_pfer` (bound on the expected number of false selections per node, default 1).

`tune`
: Logical; if `TRUE` (default) α and ϑ are selected over `alpha_grid` and `theta_grid` with the rule given by `selection`. If `FALSE`, the first value of each grid is used.

`alpha_grid`
: Candidate values in [0, 1] for α, the balance between element-wise sparsity ($\alpha \to 1$) and group sparsity ($\alpha \to 0$).

`theta_grid`
: Positive candidate values for ϑ, the penalty on covariate-specific effects relative to the baseline penalty.

`random_search`
: Logical; if `TRUE`, each node evaluates a random subset of the α and ϑ hyperparameter grid instead of the full grid. Default `FALSE`.

`p_random`
: Proportion of the grid evaluated when `random_search` is `TRUE`. Default 0.5.

`nfolds`
: Number of cross-validation folds, used by `lambda_mean` selection and by the `"cv_min"`, `"cv_1se"` and `"stability_cv"` rules. Default 5.

`symm_method`
: Symmetrisation rule: `"AND"` (default) keeps an edge only if it is selected in both node-wise regressions, `"OR"` if it is selected in either. `"AND"` is recommended with `"bic_relaxed"`, `"OR"` with `"ebic_relaxed"` (see *Selection rules*).

`ncores` Number of cores used to fit the node-wise regressions in parallel with the **parallel** package. Default 1 (sequential). For very large networks see `prcggm_slurm_prepare()`.

`verbose` Logical; print progress messages. Default `FALSE`.

`lambda_prec_type` Deprecated; use `selection`. `"min"` and `"1se"` correspond to `selection = "cv_min"` and `"cv_1se"`.

Details

Throughout, n is the number of subjects, p the number of features (network nodes) and q the number of covariate columns after encoding (see *Covariate encoding*).

Estimation follows two steps. First, each feature is regressed on the covariates with a lasso penalty and the residuals are kept. Second, for each node j the residuals are regressed on the residuals of the other nodes and on their interactions with the covariates, using the sparse-group lasso penalty

$$\alpha\lambda\|(1 - W_{j,-j}) \odot d_j^0\|_1 + \vartheta\alpha\lambda\|d_j^{-0}\|_1 + (1-\alpha)\lambda \sum_{h=1}^{q} \|d_j^h\|_2,$$

where d_j^0 are the population-level (baseline) coefficients, d_j^h the coefficients modulated by covariate h and W the prior weight matrix. The prior therefore only affects the baseline network. For each node, the penalty λ and the structural parameters α and ϑ are selected with the rule given by `selection` (see *Selection rules*), and by default the selected predictors are refitted by least squares, which removes the shrinkage of the edge weights without changing the selected edges. Node-wise estimates are finally combined into symmetric matrices with the "OR" or "AND" rule.

Selection rules. With $BIC = n\log(RSS_j/n) + \log(n)df_j$, where df_j is the number of non-zero coefficients of node j, and the extended BIC (Chen and Chen, 2008) $EBIC = BIC + 2\gamma\log\binom{P}{df_j}$, where P is the number of predictors and γ is `selection_control$ebic_gamma`:

`"bic_relaxed"` **(default)** λ, α and ϑ jointly minimise the BIC over the λ path and the hyperparameter grid, with RSS_j the residual sum of squares of the least-squares refit of the predictors selected at each λ ("relaxed" criterion). No cross-validation is needed. Recommended with `symm_method = "AND"` (the default), which removes the edges it over-selects in single node-wise regressions; this combination is the most accurate for $n \geq 5000$ in the paper's simulations, in particular for binary covariate effects.

`"ebic_relaxed"` As `"bic_relaxed"` with the EBIC. More conservative; recommended with `symm_method = "OR"`, the most accurate combination for small samples ($n \approx 1000$) and for continuous covariate effects at moderate n.

`"bic"`, `"ebic"` As the relaxed rules but with RSS_j of the penalised fit. The lasso shrinkage inflates this RSS by an amount that grows with n, so these rules choose ever smaller λ and select increasingly dense networks as n grows. Kept for comparison.

`"cv_min"`, `"cv_1se"` λ by `nfolds`-fold cross-validation within each hyperparameter combination (`lambda.min` or `lambda.1se`), then the combination with the lowest BIC. `"cv_min"` is the rule of version 0.1.0; it selects increasingly dense networks as n grows.

`"stability_bic"`, `"stability_cv"` Stability selection (Meinshausen and Buhlmann, 2010) at the hyperparameters chosen by `"bic_relaxed"` or `"cv_min"`: the λ path is refitted on `stab_B` random subsamples of size `stab_fraction` $\times n$ (the same subsamples for all nodes), the

path is restricted to the region where the average number of selected predictors is at most $\sqrt{PFER(2\pi - 1)P}$, and the predictors selected in a proportion of at least π of the subsamples somewhere in that region are kept. The selected predictors are always refitted. More accurate, especially for covariate effects, at a higher computational cost.

If `lambda_prec` is supplied, λ is fixed and only the hyperparameters are selected, by BIC.

Depending on the inputs, the same function fits a standard neighbourhood-selection GGM (no covariates, no prior), a prior-weighted GGM (prior only), a conditional GGM (covariates only) or the full prior-informed conditional GGM.

Covariate encoding. Numeric covariates are standardised and give one column each. Factors (and character or logical columns) with C levels are dummy coded into $C-1$ columns with the first level as the reference, so the baseline network refers to the reference group; use `factor()` or `relevel()` to choose the reference level. The total number of resulting columns is q.

Reproducibility. Cross-validation folds (shared by all nodes and hyperparameter values), the random-search subsets and the subsamples of stability selection (shared by all nodes) are drawn from the current random number generator state before any model is fitted, so calling `set.seed()` beforehand makes the fit reproducible, also when `ncores > 1`.

Value

An object of class `"prcggm"`, a list with elements:

`networks` Named list of symmetric $p \times p$ adjacency matrices of partial-correlation-type edge weights: `"Baseline"` (the population network) followed by one matrix per covariate effect, named after the encoded covariate (for example `"sex_Male"`).

`precision` Named list with the corresponding precision matrix components $\hat{\Delta}_h$. The diagonal of `"Baseline"` holds the inverse residual variances.

`mean_effects` $p \times (q + 1)$ matrix of covariate effects on the mean (intercept first), or `NULL` if `mean_estimation = FALSE`.

`residuals` Matrix of residuals from the mean step, used as input to the network step.

`coefficients` Matrix of node-wise regression coefficients (one row per node) before symmetrisation; refitted by least squares if `refit = TRUE`.

`coefficients_penalised` The penalised coefficients of the selected models, or `NULL` for stability selection.

`sigma2` Residual variance of each node-wise regression.

`tuning` Data frame with, for each node, the selected `lambda` (`NA` for stability selection), `alpha`, `theta`, the `bic` and the number of selected predictors `df` of the reported model, and the computing time in minutes.

`covariates` Encoding information used by `predict.prcggm()`.

`settings` List of the main estimation settings.

`n`, `p`, `q` Number of subjects, features and encoded covariates.

`call` The matched call.

References

Mapelli, A., Massi, M. C., Cuccuru, G., Di Angelantonio, E. and Ieva, F. (2026). Prior-informed conditional Gaussian graphical models: an application to protein interaction network reconstruction. arXiv:2606.31805. doi:10.48550/arXiv.2606.31805

Chen, J. and Chen, Z. (2008). Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3), 759–771.

Meinshausen, N. and Buhlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B*, 72(4), 417–473.

Zhang, J. and Li, Y. (2023). High-dimensional Gaussian graphical regression models with covariates. *Journal of the American Statistical Association*, 118(543), 2088–2100.

See Also

`predict.prcggm()`, `summary.prcggm()`, `plot.prcggm()`, `prcggm_slurm_prepare()`.

Examples

```
data(prcggm_example)
set.seed(1)
fit <- prcggm(prcggm_example$x,
              covariates = prcggm_example$covariates,
              prior = prcggm_example$prior,
              tune = FALSE)
fit
summary(fit)


# Tune alpha and theta on a small grid
fit_tuned <- prcggm(prcggm_example$x,
                    covariates = prcggm_example$covariates,
                    prior = prcggm_example$prior,
                    alpha_grid = c(0.5, 0.9), theta_grid = c(1, 1.3))
fit_tuned$tuning
```

---

prcggm_example *Example data for prCGGM*

Description

A small simulated data set with 1000 subjects, 10 proteins and one binary covariate, generated from a conditional Gaussian graphical model with a scale-free baseline network and an Erdos-Renyi covariate-specific network, as in the simulation study of Mapelli et al. (2026). The prior weight matrix is "perfect": it equals 1 on the true baseline edges and 0 elsewhere.

Usage

```
prcggm_example
```

Format

A list with three elements:

`x` Numeric matrix with 1000 rows (subjects) and 10 columns (`Prot1` to `Prot10`) of expression values.

`covariates` Data frame with one factor column, `x1`, with levels `"0"` (reference) and `"1"`.

`prior` 10 x 10 prior weight matrix with entries in {0, 1}.

Source

Simulated; see `data-raw/prcggm_example.R` in the package sources.

Examples

```
data(prcggm_example)
str(prcggm_example)
```

`prcggm_slurm_prepare` *Run the estimation as SLURM array jobs*

Description

For large networks the node-wise regressions can be distributed over a computing cluster, one 'SLURM' array task per node, in three steps:

1. `prcggm_slurm_prepare()` validates the inputs, runs the mean step and saves everything the node-wise regressions need in `dir`. It also copies template job scripts to `dir` and returns the `sbatch` command to submit.
2. Each array task calls `prcggm_node()` (through the copied `prcggm_node_job.R` script) and saves the result of one node in `dir`.
3. Once all tasks have finished, `prcggm_slurm_collect()` reads the node results and returns the same `"prcggm"` object as `prcggm()`.

Usage

```
prcggm_slurm_prepare(
  x,
  covariates = NULL,
  prior = NULL,
  dir,
  screening = FALSE,
  gamma = NULL,
  screening_matrix = NULL,
  mean_estimation = TRUE,
  lambda_mean = NULL,
  lambda_mean_type = c("min", "1se"),
```

```
  lambda_prec = NULL,
  selection = c("bic_relaxed", "ebic_relaxed", "bic", "ebic", "cv_min", "cv_1se",
    "stability_bic", "stability_cv"),
  refit = TRUE,
  selection_control = list(),
  tune = TRUE,
  alpha_grid = c(0.75, 0.5, 0.9, 0.99),
  theta_grid = c(1, 0.8, 1.1, 1.3),
  random_search = FALSE,
  p_random = 0.5,
  nfolds = 5,
  symm_method = c("AND", "OR"),
  copy_templates = TRUE,
  verbose = FALSE,
  lambda_prec_type = NULL
)

prcggm_node(input, node, output_dir = dirname(input))

prcggm_slurm_collect(dir, output_dir = dir)
```

Arguments

x	Numeric $n \times p$ matrix or data frame of expression data, with the n subjects in rows and the p features (network nodes) in columns. Missing values are not allowed.
covariates	Optional data frame of subject-level covariates with n rows, in the same order as the rows of x. Columns can be numeric, factor, character or logical; they are encoded into q columns as described in *Covariate encoding*. If NULL (default), a single population network is estimated ($q = 0$).
prior	Optional symmetric $p \times p$ matrix of prior edge confidence in [0, 1], with rows and columns in the same order as the columns of x. Edges with entries close to 1 are penalised less in the baseline network. If NULL (default), no prior knowledge is used.
dir	Directory where inputs, job scripts and node results are written. It is created if needed. There is no default: choose a location on a file system shared by the cluster nodes.
screening	Logical; if TRUE, predictors weakly correlated with the response node (absolute correlation of the residuals not above gamma) are excluded from each node-wise regression, unless they have prior support. Default FALSE.
gamma	Correlation threshold in [0, 1] for screening. If NULL (default), the 20th percentile of the $p(p-1)/2$ absolute pairwise correlations is used.
screening_matrix	Optional $p \times p$ matrix with entries in {0, 1}; entry $(j, k) = 0$ excludes feature k from the regression of node j. When supplied, it is used for screening instead of the correlation rule. Prior-supported pairs are never excluded.

`mean_estimation`
: Logical; if `TRUE` (default) the covariate effects on the mean are estimated and removed first. Set to `FALSE` only if `x` has already been centred conditionally on the covariates.

`lambda_mean`
: Optional fixed lasso penalty for the mean regressions. If `NULL` (default), it is selected by cross-validation with `glmnet::cv.glmnet()`.

`lambda_mean_type`
: Cross-validation rule used to select `lambda_mean`, with the same meaning as in `glmnet::cv.glmnet()`: `"min"` (default) selects `lambda.min`, the value that minimises the cross-validated error; `"1se"` selects `lambda.1se`, the largest value whose error is within one standard error of the minimum, which gives sparser models. Ignored if `lambda_mean` is supplied or if $q = 1$, in which case the mean is estimated by least squares.

`lambda_prec`
: Optional fixed penalty for the node-wise network regressions. If `NULL` (default), it is selected with the rule given by `selection`.

`selection`
: Rule used to select the penalty of the node-wise network regressions and the hyperparameters: `"bic_relaxed"` (default), `"ebic_relaxed"`, `"bic"`, `"ebic"`, `"cv_min"`, `"cv_1se"`, `"stability_bic"` or `"stability_cv"`. See *Selection rules*.

`refit`
: Logical; if `TRUE` (default), the predictors selected for each node are refitted by least squares, so that the edge weights are not shrunk. The selected edges do not change. Always `TRUE` for stability selection.

`selection_control`
: List of settings of the selection rules, any of `ebic_gamma` (EBIC parameter, default 0.5), `stab_B` (number of subsamples, default 50), `stab_fraction` (subsample size as a fraction of n, default 0.5), `stab_pi` (selection-frequency threshold, default 0.75) and `stab_pfer` (bound on the expected number of false selections per node, default 1).

`tune`
: Logical; if `TRUE` (default) α and ϑ are selected over `alpha_grid` and `theta_grid` with the rule given by `selection`. If `FALSE`, the first value of each grid is used.

`alpha_grid`
: Candidate values in $[0, 1]$ for α, the balance between element-wise sparsity ($\alpha \to 1$) and group sparsity ($\alpha \to 0$).

`theta_grid`
: Positive candidate values for ϑ, the penalty on covariate-specific effects relative to the baseline penalty.

`random_search`
: Logical; if `TRUE`, each node evaluates a random subset of the α and ϑ hyperparameter grid instead of the full grid. Default `FALSE`.

`p_random`
: Proportion of the grid evaluated when `random_search` is `TRUE`. Default 0.5.

`nfolds`
: Number of cross-validation folds, used by `lambda_mean` selection and by the `"cv_min"`, `"cv_1se"` and `"stability_cv"` rules. Default 5.

`symm_method`
: Symmetrisation rule: `"AND"` (default) keeps an edge only if it is selected in both node-wise regressions, `"OR"` if it is selected in either. `"AND"` is recommended with `"bic_relaxed"`, `"OR"` with `"ebic_relaxed"` (see *Selection rules*).

`copy_templates`
: Logical; copy the template job scripts to `dir`. Default `TRUE`.

`verbose`
: Logical; print progress messages. Default `FALSE`.

`lambda_prec_type`
: Deprecated; use `selection`. `"min"` and `"1se"` correspond to `selection = "cv_min"` and `"cv_1se"`.

`input`
: Path of the input file written by `prcggm_slurm_prepare()`.

`node`
: Index of the node (feature) to estimate, from 1 to `p`.

`output_dir`
: Directory where the node result is saved. Defaults to the directory of `input`.

Details

Edit the copied `prcggm_node.sbatch` file to match your cluster (partition, memory, time, and how R is loaded) before submitting. The template files are also available in `system.file("slurm", package = "prCGGM")`.

Value

`prcggm_slurm_prepare()` returns, invisibly, a list with the path of the saved input file (`input`), the number of nodes (`p`) and the `sbatch` command to run (`command`), which is also printed as a message.

`prcggm_node()` returns, invisibly, the path of the saved node result.

`prcggm_slurm_collect()` returns a `"prcggm"` object, as `prcggm()`.

Examples

```
## Not run:
# On the cluster login node
job <- prcggm_slurm_prepare(expr, covariates = covs, prior = W,
                            dir = "/scratch/my_project/prcggm")
# Submit job$command from a shell, e.g.
#   sbatch --array=1-366 /scratch/my_project/prcggm/prcggm_node.sbatch ...
# When all array tasks have completed:
fit <- prcggm_slurm_collect("/scratch/my_project/prcggm")

## End(Not run)

# The same three steps run locally, without SLURM:
data(prcggm_example)
dir <- file.path(tempdir(), "prcggm_slurm")
set.seed(1)
job <- prcggm_slurm_prepare(prcggm_example$x,
                            covariates = prcggm_example$covariates,
                            prior = prcggm_example$prior,
                            tune = FALSE, dir = dir,
                            copy_templates = FALSE)
for (i in seq_len(job$p)) prcggm_node(job$input, i)
fit <- prcggm_slurm_collect(dir)
unlink(dir, recursive = TRUE)
```

`predict.prcggm` *Predict personalised networks*

Description

Computes the personalised network of new individuals as the linear combination of the estimated baseline and covariate-specific components, $\hat{A}(x) = \hat{A}_0 + \sum_h x_h \hat{A}_h$.

Usage

```
## S3 method for class 'prcggm'
predict(object, newdata = NULL, type = c("network", "precision"), ...)
```

Arguments

`object`	A `"prcggm"` object returned by `prcggm()`.
`newdata`	Data frame with the covariates of the new individuals, containing every covariate used in the fit. If `NULL` (default), or if the model has no covariates, the baseline network is returned.
`type`	`"network"` (default) for the adjacency matrices of edge weights, or `"precision"` for the precision matrices.
`...`	Not used.

Details

Covariates in `newdata` are encoded exactly as in the estimation data: numeric covariates are standardised with the training mean and standard deviation, and factors are dummy coded against the training reference level.

Value

A list with one matrix per row of `newdata`, named `Subject_1`, `Subject_2`, and so on. Without `newdata`, a list holding the baseline matrix only.

See Also

`prcggm()`, `compare_networks()`.

Examples

```
data(prcggm_example)
set.seed(1)
fit <- prcggm(prcggm_example$x,
              covariates = prcggm_example$covariates,
              prior = prcggm_example$prior,
              tune = FALSE)
new_subjects <- data.frame(x1 = factor(c("0", "1")))
nets <- predict(fit, new_subjects)
compare_networks(nets$Subject_1, nets$Subject_2, method = "frobenius")
```

`simulate_prcggm_data` *Simulate data from a conditional Gaussian graphical model*

Description

Generates expression data whose precision matrix depends linearly on the covariates, $\Omega(x) = \Delta_0 + \sum_h x_h \Delta_h$, together with a prior weight matrix of the chosen quality, following the simulation design of Mapelli et al. (2026).

Usage

```
simulate_prcggm_data(
  n = 500,
  p = 10,
  n_binary = 1,
  n_continuous = 0,
  edge_prob = NULL,
  coef_range = c(0.35, 0.5),
  effect_range = c(0.175, 0.25),
  exponent = 2.5,
  prior = c("perfect", "noisy", "none"),
  sensitivity = 0.8,
  specificity = 0.9,
  confidence_range = c(0.7, 1),
  eig_margin = 0.1,
  power = NULL
)
```

Arguments

`n`	Number of subjects.
`p`	Number of features (nodes).
`n_binary`, `n_continuous`	Number of binary (returned as factors with levels `"0"` and `"1"`) and continuous covariates.
`edge_prob`	Edge probability of the covariate networks. If `NULL` (default), $1/(p-1)$.
`coef_range`	Range of the absolute values of the non-zero off-diagonal entries of the baseline network; signs are random.
`effect_range`	Range of the absolute values of the non-zero entries of the covariate networks; signs are random.
`exponent`	Exponent of the power-law degree distribution of the baseline network.
`prior`	Quality of the simulated prior: `"perfect"` (weight 1 on the edges of the true baseline, 0 elsewhere), `"noisy"` (true edges kept with probability `sensitivity`, false edges added with probability `1 - specificity`, with random confidence scores) or `"none"`.

`sensitivity`, `specificity`
: Parameters of the noisy prior.

`confidence_range`
: Range of the confidence scores of true edges in the noisy prior; false edges get scores between 0.1 and `confidence_range[1]` - `0.1`.

`eig_margin`
: Minimum eigenvalue of the precision matrix over the covariate range.

`power`
: Deprecated and ignored (preferential-attachment power of version 0.1.0); see `exponent`.

Details

- **Baseline network** Δ_0: `p` edges drawn from a fitness model (Goh et al., 2001), whose degree distribution follows a power law with exponent `exponent`; non-zero entries $\pm$U(`coef_range`).
- **Covariate networks** Δ_h, one per covariate: Erdos-Renyi graphs with edge probability `edge_prob` (by default $1/(p-1)$, about $p/2$ edges); non-zero entries $\pm$U(`effect_range`) and zero diagonal.
- **Diagonal**: one constant d for all nodes and all covariate values, the smallest that keeps the minimum eigenvalue of $\Omega(x)$ at least `eig_margin` over the whole covariate range. Because $\Omega(x)$ is linear in x, checking the 2^q corners of the covariate range is enough, so $\Omega(x)$ is positive definite for every subject. The conditional variances $1/d$ do not depend on the covariates, so the node-wise regressions are exactly linear in the covariates, as assumed by `prcggm()`.
- **Covariates**: binary ones are Bernoulli(0.5), continuous ones Uniform(0, 1). Every covariate has an effect.

Target of the baseline. `prcggm()` standardises numeric covariates and codes factors against their first level, so its baseline network is the network of a subject with the continuous covariates at their mean and the binary ones at the reference level 0: $T_0 = \Delta_0 + \sum_{h \in continuous} \bar{x}_h \Delta_h$. The true baseline returned in `truth_precision` and `truth_networks` is T_0, and the prior is generated from the edges of T_0. Likewise, the true effects of continuous covariates are returned per standard deviation of the covariate, $sd(x_h)\Delta_h$, the scale of the estimates of `prcggm()`.

Value

A list with elements:

`x` The $n \times p$ matrix of simulated expression data.

`covariates` Data frame of the simulated covariates, named `x1`, `x2`, ...: first the binary ones (factors), then the continuous ones.

`prior` The $p \times p$ prior weight matrix (all zero for `prior = "none"`).

`truth_precision` Named list of the true precision components on the scale of the `precision` element of a fitted `prcggm()` model: `"Baseline"` (T_0, with diagonal d) and one component per covariate.

`truth_networks` Named list of the true networks on the scale of the `networks` element of a fitted `prcggm()` model, the node-wise regression coefficients $-(\Delta_h)_{jk}/d$, with zero diagonal. Their sign is opposite to that of the precision components.

`diagonal` The common diagonal d.

Both lists use the same names as the components of a fitted `prcggm()` model (`"Baseline"`, `"x1_1"`, `"x2"`, ...), so that, for example, `fit$networks$x1_1` estimates `sim$truth_networks$x1_1`.

References

Goh, K.-I., Kahng, B. and Kim, D. (2001). Universal behavior of load distribution in scale-free networks. *Physical Review Letters*, 87(27), 278701.

Examples

```
set.seed(1)
sim <- simulate_prcggm_data(n = 200, p = 8)
str(sim, max.level = 1)

# Compare the estimated baseline network with the truth
fit <- prcggm(sim$x, covariates = sim$covariates, prior = sim$prior,
              mean_estimation = FALSE, tune = FALSE)
compare_networks(fit$networks$Baseline, sim$truth_networks$Baseline,
                 method = "correlation")
```

`summary.prcggm` *Summarise a fitted prCGGM model*

Description

Gives an overview of a fitted `"prcggm"` object: the size of the problem, the model that was fitted, the structure of each estimated network component and the hyperparameters selected across nodes.

Usage

```
## S3 method for class 'prcggm'
summary(object, ...)
```

Arguments

`object`	A `"prcggm"` object returned by `prcggm()` or `prcggm_slurm_collect()`.
`...`	Not used.

Details

The summary reports:

- the number of subjects n, features p and encoded covariate columns q, whether prior knowledge was used and the symmetrisation rule;
- for each network component (the baseline network and one component per covariate effect), the number of edges, the density (edges divided by the $p(p-1)/2$ possible pairs) and the mean absolute weight of the non-zero edges;
- how often each combination of α and ϑ was selected by BIC across the p node-wise regressions.

Printing the returned object shows these results; the full estimates remain available in the fitted object (see `prcggm()`).

Value

An object of class `"summary.prcggm"`, a list with elements:

`components` Data frame with one row per network component and columns `component`, `edges`, `density` and `mean_abs_weight`.

`tuning` Contingency table of the selected `alpha` (rows) and `theta` (columns) across nodes.

`fit` List with `n`, `p`, `q`, `prior` (logical), `symm_method` and `selection` (the selection rule, with `refit`).

See Also

`prcggm()`, `plot.prcggm()`, `predict.prcggm()`.

Examples

```
data(prcggm_example)
set.seed(1)
fit_tuned <- prcggm(prcggm_example$x,
              covariates = prcggm_example$covariates,
              prior = prcggm_example$prior,
              alpha_grid = c(0.5, 0.9), theta_grid = c(1, 1.3))
summary(fit_tuned)
```

Index

∗ **datasets**
prcggm_example, 10

compare_networks, 2
compare_networks(), *15*

factor(), *9*

glmnet::cv.glmnet(), *7*, *13*

plot.prcggm, 3
plot.prcggm(), *10*, *19*
plot_network, 4
plot_network(), *3*, *5*
plot_network_difference, 5
prcggm, 5
prcggm(), *3*, *11*, *14, 15*, *17–19*
prcggm_example, 10
prcggm_node *(*prcggm_slurm_prepare*)*, 11
prcggm_slurm_collect
*(*prcggm_slurm_prepare*)*, 11
prcggm_slurm_collect(), *18*
prcggm_slurm_prepare, 11
prcggm_slurm_prepare(), *8*, *10*
predict.prcggm, 15
predict.prcggm(), *4*, *9, 10*, *19*

relevel(), *9*

set.seed(), *9*
simulate_prcggm_data, 16
stats::heatmap(), *4*
summary.prcggm, 18
summary.prcggm(), *10*