Package 'graCRR'

December 24, 2025

Type Package

Title Graph-Assisted Convoluted Rank Regression

Version 0.1.0

Author Ziyu Xiang [aut, cre], Xiangyong Tan [aut], Shishuo Guo [aut], Xu Liu [aut]

Maintainer Ziyu Xiang <xiangziyu@stu.sufe.edu.cn>

Description Provides methods for estimation and inference in high-dimensional linear models using graph-assisted convoluted rank regression (CRR). The package implements a Forward and Backward Stagewise (Fabs) algorithm with lasso and Laplacian penalties, supports EBIC-based selection of the Laplacian penalty, and provides graph-assisted debiased estimators with asymptotic normal inference, including standard errors, confidence intervals, and p-values. An optional multiplier bootstrap procedure is also provided for interval construction.

License GPL (>= 2)

Imports mnormt, Matrix, glmnet, flare

Repository github

URL https://github.com/Ziyu-Xiang/graCRR

BugReports https://github.com/Ziyu-Xiang/graCRR/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

NeedsCompilation yes

Archs x64

Contents

`Data_simu` *Simulate high-dimensional linear model data*

Description

Generate data from a linear regression model with a structured covariance matrix for the design, and various choices of error distributions.

Usage

```
Data_simu(n, p, s0, xcov, error, signal=1)
```

Arguments

`n`	Sample size.
`p`	Number of predictors p. It must be a multiple of 5; otherwise the function stops with an error.
`s0`	Number of nonzero coefficients in the true regression vector $\boldsymbol{\beta}$. The first `s0` coordinates of $\boldsymbol{\beta}$ are set to `signal`, and the remaining are zeros.
`xcov`	A character string specifying the covariance structure of the design matrix $\mathbf{X}$. Possible values are: `"AR1"` Block-diagonal AR(1) covariance with cluster size 5 and correlation parameter $\rho = 0.5$. `"AR2"` Block-diagonal AR(1) covariance with cluster size 5 and correlation parameter $\rho = 1/1.01$, corresponding to a highly correlated design. `"Band"` Block-diagonal banded (moving-average-type) covariance. A random vector $\boldsymbol{\rho}$ of length 5 is generated from `runif(5)` and normalized to have unit ℓ_2-norm; the within-block covariance is then constructed from $\boldsymbol{\rho}$ and replicated along the diagonal.
`error`	A character string specifying the distribution of the error term $\boldsymbol{\varepsilon}$. Possible values are: `"norm"` Standard normal errors, $\varepsilon_i \sim N(0,1)$. `"mnorm"` Mixture normal errors: with probability 0.95, $N(0,1)$; with probability 0.05, $N(0,100)$. `"t3"` t-distributed errors with 3 degrees of freedom, $\varepsilon_i \sim t_3$. `"cauchy"` t-distributed errors with 1 degree of freedom, i.e., standard Cauchy errors.
`signal`	Signal strength for the nonzero coefficients. The first `s0` entries of the true regression vector $\boldsymbol{\beta}$ are set to `signal`. Default is `1`.

Details

The design covariance matrix Σ is first constructed according to the argument `xcov`, and the corresponding precision matrix is computed as $\Theta = \Sigma^{-1}$. The matrix `G` is then obtained by thresholding the entries of Θ: $G[i,j]$ is set to $|\,\Theta_{ij}\,|$ if $|\,\Theta_{ij}\,| > 10^{-8}$ and to 0 otherwise. Thus, `G` can be interpreted as a weighted adjacency matrix; if only the graph structure is of interest, it can be further binarized by treating all nonzero entries as weight 1.

Value

A list with the following components:

`X`	An $n \times p$ design matrix simulated from a multivariate normal distribution with mean zero and the specified covariance matrix.
`y`	The response vector of length n, generated as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$.
`beta_T`	The true regression coefficient vector $\boldsymbol{\beta}$ of length p, where the first `s0` entries equal `signal` and the rest are zeros.
`G`	A square matrix representing the weighted graph structure of the predictors. Each entry $G[i, j]$ indicates the weight of the edge between node i and node j. A value of zero means no edge is present. Diagonal entries are ignored.

References

Tan, X., Zhang, X., Cui, Y., and Liu, X. (2024). Uncertainty quantification in high-dimensional linear models incorporating graphical structures with applications to gene set analysis. *Bioinformatics*, **40**(9), btae541.

Cai, L., Guo, X., Lian, H., and Zhu, L. (2025). Statistical inference for high-dimensional convoluted rank regression. *Journal of the American Statistical Association*, 1–25.

Zhong, P.-S. and Chen, S. X. (2011). Tests for high-dimensional regression coefficients with factorial designs. *Journal of the American Statistical Association*, **106**(493), 260–274.

Examples

```
set.seed(123)
dat <- Data_simu(n = 50, p = 100, s0 = 5,
                 xcov = "AR1", error = "norm", signal = 1)
str(dat)
```

`debias_lap` *Graph-assisted debiased estimator for high-dimensional linear models*

Description

Compute a debiased estimator for high-dimensional linear models fitted using the convoluted rank loss with ℓ_1 and Laplacian penalties.

Usage

```
debias_lap(X, y, beta, G, h)
```

Arguments

`X`	An $n \times p$ design matrix of predictors.
`y`	A numeric response vector of length n.
`beta`	A numeric vector of length p giving an initial estimator of the regression coefficients, i.e., the solution to a graph-assisted convoluted rank regression problem with ℓ_1 and Laplacian penalties.

G — A square matrix representing the weighted graph structure of the predictors. Each entry $G[i, j]$ indicates the weight of the edge between node i and node j. A value of zero means no edge is present. Diagonal entries are ignored.

h — A positive bandwidth parameter used in the smoothed (convoluted) rank loss and its derivatives. It should match the bandwidth used to obtain the initial estimator `beta`.

Value

A numeric vector of length p giving the debiased estimator of the regression coefficients.

References

Tan, X., Zhang, X., Cui, Y., and Liu, X. (2024). Uncertainty quantification in high-dimensional linear models incorporating graphical structures with applications to gene set analysis. *Bioinformatics*, **40**(9), btae541.

Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **76**(1), 217–242.

Examples

```
set.seed(123)

## Simulate data (see Data_simu for details)
dat <- Data_simu(n = 50, p = 100, s0 = 5,
                 xcov = "AR1", error = "norm", signal = 1)
X <- dat$X
y <- dat$y
G <- dat$G

## Suppose we start from a simple initial estimator
beta_init <- rep(0, ncol(X))

beta_db <- debias_lap(X, y, beta_init, G, h = 1)
```

---

ebic.Lfabs — *A forward and backward stagewise algorithm for high-dimensional convoluted rank regression with sparse Laplacian shrinkage*

Description

Fit a high-dimensional linear model using the convoluted rank loss with ℓ_1 and graph Laplacian penalties, where the Laplacian penalty parameter `lambda2` is selected in a data-driven way using EBIC. The function evaluates a grid of candidate `lambda2` values, applies `Lfabs` at each value, and chooses the Laplacian penalty that yields the smallest EBIC. The graph structure is encoded by `G`.

Usage

```
ebic.Lfabs(X, y, G, h = 1, nlambda2 = 5, weight = NULL)
```

Arguments

X	An $n \times p$ design matrix of predictors.
y	A numeric response vector of length n.
G	A square matrix representing the weighted graph structure of the predictors. Each entry $G[i, j]$ indicates the weight of the edge between node i and node j. A value of zero means no edge is present. Diagonal entries are ignored.
h	A positive bandwidth parameter used in the smoothed (convoluted) rank loss and its derivatives. Default is 1.
nlambda2	The number of candidate lambda2 values in the grid for EBIC-based selection. Default is 5.
weight	A weight vector of length p for the adaptive lasso penalty. Default is NULL, which corresponds to weight 1 for all β_j, i.e., the standard lasso.

Details

The function first fits the model with lambda2 = 0 via Lfabs to obtain a reference ℓ_1 tuning parameter. It then constructs a grid of candidate Laplacian penalties lambda2 on a logarithmic scale and, for each candidate, calls Lfabs and records the minimum EBIC along the corresponding solution path. The lambda2 value with the smallest EBIC is selected, and the final model is refitted at this optimal Laplacian penalty.

Value

A list with the following components:

Beta	A matrix containing the coefficient estimates along the solution path (for the selected lambda2), one column for each step.
beta	A numeric vector of length p giving the coefficient estimates at the EBIC-selected optimal ℓ_1 tuning parameter for the chosen lambda2.
lambda	A numeric vector containing the sequence of ℓ_1 tuning parameters along the path for the selected lambda2.
lambda2	The selected Laplacian penalty parameter lambda2 that minimizes EBIC over the candidate grid.
direction	A numeric vector indicating the sign of the change in the updated coefficient at each iteration. 1 indicates a forward step; 0 indicates a backward step.
iter	Integer scalar giving the number of iterations.
ebic	A numeric vector of EBIC values along the solution path (for the selected lambda2).
opt	Integer index of the EBIC-selected optimal model along the path.

References

Shi, X., Huang, Y., Huang, J., and Ma, S. (2018). A forward and backward stagewise algorithm for nonconvex loss functions with adaptive lasso. *Computational Statistics & Data Analysis*, **124**, 235–251.

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

Tan, X., Zhang, X., Cui, Y., and Liu, X. (2024). Uncertainty quantification in high-dimensional linear models incorporating graphical structures with applications to gene set analysis. *Bioinformatics*, **40**(9), btae541.

See Also

Lfabs

Examples

```
set.seed(123)

## Simulate data (see Data_simu for details)
dat <- Data_simu(n = 50, p = 100, s0 = 5,
                 xcov = "AR1", error = "norm", signal = 1)
X <- dat$X
y <- dat$y
G <- dat$G

fit <- ebic.Lfabs(X, y, G, h = 1, nlambda2 = 5)
beta_hat <- fit$beta
lambda2_hat <- fit$lambda2
```

---

infer_boot *Bootstrap-based inference for graph-assisted convoluted rank regression*

Description

Perform coordinatewise inference for high-dimensional linear models fitted via graph-assisted convoluted rank regression. Given an initial estimator, its debiased version, and a graph structure G, the function applies a multiplier bootstrap based on the smoothed rank loss to construct marginal confidence intervals for each coefficient. When the true coefficients are provided, the function additionally evaluates empirical coverage in simulation studies.

Usage

```
infer_boot(X, y, beta_hat, beta_true = NULL, debias, G, h, alpha = 0.05, B = 1000)
```

Arguments

X	An $n \times p$ design matrix of predictors.
y	A numeric response vector of length n.
beta_hat	A numeric vector of length p giving the initial estimator of the regression coefficients (e.g., the output from Lfabs or ebic.Lfabs).
beta_true	Optional. A numeric vector of length p containing the true regression coefficients. If provided, it is used to assess empirical coverage of the confidence intervals in simulation studies; if NULL (default), the coverage component is not returned.
debias	A numeric vector of length p giving the debiased estimator of the regression coefficients, for example from debias_lap.
G	A square matrix representing the weighted graph structure of the predictors. Each entry $G[i, j]$ indicates the weight of the edge between node i and node j. A value of zero means no edge is present. Diagonal entries are ignored.

h	A positive bandwidth parameter used in the smoothed (convoluted) rank loss and its derivatives. Default is 1. It should match the bandwidth used in estimation.
alpha	Nominal significance level for the marginal confidence intervals. Default is 0.05.
B	Number of bootstrap replications used in the algorithm. Default is 1000.

Value

A list with the following components:

cover	If beta_true is provided, a logical vector of length p. The j-th entry is TRUE if the true coefficient β_j^* is contained in the constructed confidence interval, and FALSE otherwise. If beta_true = NULL, this component is NULL.
length	A numeric vector of length p giving the lengths of the marginal confidence intervals for each coefficient.
lower	A numeric vector of length p giving the lower endpoints of the marginal confidence intervals.
upper	A numeric vector of length p giving the upper endpoints of the marginal confidence intervals.

References

Tan, X., Zhang, X., Cui, Y., and Liu, X. (2024). Uncertainty quantification in high-dimensional linear models incorporating graphical structures with applications to gene set analysis. *Bioinformatics*, **40**(9), btae541.

Cai, L., Guo, X., Lian, H., and Zhu, L. (2025). Statistical inference for high-dimensional convoluted rank regression. *Journal of the American Statistical Association*, 1–25.

Examples

```
set.seed(123)

## Simulate data (see Data_simu for details)
dat <- Data_simu(n = 50, p = 100, s0 = 5,
                 xcov = "AR1", error = "norm", signal = 1)
X <- dat$X
y <- dat$y
G <- dat$G
beta_true <- dat$beta_T

## Initial estimator
fit_init <- ebic.Lfabs(X, y, G, h = 1)
beta_hat <- fit_init$beta

## Debiased estimator
beta_db <- debias_lap(X, y, beta_hat, G, h = 1)

## Bootstrap-based inference
res <- infer_boot(X, y, beta_hat, beta_true, beta_db, G, h = 1)
```

`infer_norm` *Normal-approximation inference for graph-assisted convoluted rank regression*

Description

Perform coordinatewise inference for high-dimensional linear models fitted via graph-assisted convoluted rank regression. Given an initial estimator, its debiased version, and a graph structure `G`, the function constructs marginal confidence intervals for each coefficient using a normal approximation. It also reports standard error estimates and two-sided p-values. When the true coefficients are provided, the function additionally evaluates empirical coverage in simulation studies.

Usage

```
infer_norm(X, y, beta_hat, beta_true = NULL, debias, G, h, alpha = 0.05,
  inv_method = "Tan2024")
```

Arguments

`X`	An $n \times p$ design matrix of predictors.
`y`	A numeric response vector of length n.
`beta_hat`	A numeric vector of length p giving the initial estimator of the regression coefficients (e.g., the output from `Lfabs` or `ebic.Lfabs`).
`beta_true`	Optional. A numeric vector of length p containing the true regression coefficients. If provided, it is used to assess empirical coverage of the confidence intervals in simulation studies; if `NULL` (default), the coverage component is not returned.
`debias`	A numeric vector of length p giving the debiased estimator of the regression coefficients, for example from `debias_lap`.
`G`	A square matrix representing the weighted graph structure of the predictors. Each entry $G[i, j]$ indicates the weight of the edge between node i and node j. A value of zero means no edge is present. Diagonal entries are ignored.
`h`	A positive bandwidth parameter used in the smoothed (convoluted) rank loss and its derivatives. Default is `1`. It should match the bandwidth used in estimation.
`alpha`	Nominal significance level for the marginal confidence intervals. Default is `0.05`.
`inv_method`	A character string specifying the method used to estimate the inverse Gram (or precision) matrix in the variance formula. Possible values are: `"Tan2024"` The graph-assisted inverse Gram matrix estimator proposed by Tan, Zhang, Cui, and Liu (2024). In the implementation, the inverse Gram matrix is computed via the function `calculate_inv_gram(X, G)` from the **GCDL** package, which incorporates the graph structure `G`. `"Cai2025"` The precision matrix estimator used in Cai, Guo, Lian, and Zhu (2025). In the implementation, a sparse precision matrix is estimated using the function `sugm` from the **flare** package. Default is `"Tan2024"`.

Value

A list with the following components:

`cover`	If `beta_true` is provided, a logical vector of length p. The j-th entry is `TRUE` if the true coefficient β_j^* is contained in the constructed confidence interval, and `FALSE` otherwise. If `beta_true = NULL`, this component is `NULL`.
`length`	A numeric vector of length p giving the lengths of the marginal confidence intervals for each coefficient.
`lower`	A numeric vector of length p giving the lower endpoints of the marginal confidence intervals.
`upper`	A numeric vector of length p giving the upper endpoints of the marginal confidence intervals.
`Inv_Gram`	The estimated inverse Gram/precision matrix used in the inference procedure.
`sehat`	A numeric vector of length p giving the estimated standard errors for each coordinate of the debiased estimator.
`p_value`	A numeric vector of length p giving two-sided p-values based on the normal approximation for testing the null hypotheses $H_0 : \beta_j = 0, j = 1, \ldots, p$.

References

Tan, X., Zhang, X., Cui, Y., and Liu, X. (2024). Uncertainty quantification in high-dimensional linear models incorporating graphical structures with applications to gene set analysis. *Bioinformatics*, **40**(9), btae541.

Cai, L., Guo, X., Lian, H., and Zhu, L. (2025). Statistical inference for high-dimensional convoluted rank regression. *Journal of the American Statistical Association*, 1–25.

Examples

```
set.seed(123)

## Simulate data (see Data_simu for details)
dat <- Data_simu(n = 50, p = 100, s0 = 5,
                 xcov = "AR1", error = "norm", signal = 1)
X <- dat$X
y <- dat$y
G <- dat$G
beta_true <- dat$beta_T

## Initial estimator
fit_init <- ebic.Lfabs(X, y, G, h = 1)
beta_hat <- fit_init$beta

## Debiased estimator
beta_db <- debias_lap(X, y, beta_hat, G, h = 1)

## Normal-approximation inference
res <- infer_norm(X, y, beta_hat,
                  beta_true = beta_true,
                  debias = beta_db,
                  G = G, h = 1,
                  inv_method = "Tan2024")
```

`Lfabs` *A forward and backward stagewise algorithm for high-dimensional convoluted rank regression with sparse Laplacian shrinkage*

Description

Fit a high-dimensional linear model using the convoluted rank loss with ℓ_1 and graph Laplacian penalties. The function computes a solution path over the ℓ_1 tuning parameter `lambda` while fixing the Laplacian penalty parameter `lambda2`, and selects the optimal model along the path using EBIC. The graph structure is encoded by `G` and determines which predictors are penalized jointly.

Usage

```
Lfabs(X, y, lambda2, G, h = 1, stopping = TRUE, eps = 0.02, xi = 1e-6,
    iter = 1e4, lambda.min = 1e-4, weight = NULL)
```

Arguments

`X`	An $n \times p$ design matrix of predictors.
`y`	A numeric response vector of length n.
`lambda2`	Nonnegative tuning parameter for the graph Laplacian penalty term $\frac{1}{2}\lambda_2 \boldsymbol{\beta}^\top L \boldsymbol{\beta}$, where L is the graph Laplacian matrix derived from `G`. This value is held fixed while the ℓ_1 tuning parameter varies along a path.
`G`	A square matrix representing the weighted graph structure of the predictors. Each entry $G[i, j]$ indicates the weight of the edge between node i and node j. A value of zero means no edge is present. Diagonal entries are ignored.
`h`	A positive bandwidth parameter used in the smoothed (convoluted) rank loss and its derivatives. Default is `1`.
`stopping`	Logical indicator of whether to stop the iterations when `lambda` is less than `lambda.min`. Default is `TRUE`.
`eps`	The step size for updating coefficients. Default is `0.02`.
`xi`	A small positive threshold used in the algorithm. Default is `1e-6`.
`iter`	The maximum number of iterations allowed. Default is `1e4`.
`lambda.min`	The smallest value for `lambda`, used as a stopping criterion for the solution path. Default is `1e-4`.
`weight`	A weight vector of length p for the adaptive lasso penalty. Default is `NULL`, which corresponds to weight 1 for all β_j, i.e., the standard lasso.

Details

The Laplacian shrinkage level is fixed, which means `lambda2` is predetermined. EBIC is evaluated along the path and is used to select the optimal model.

Value

A list with the following components:

`Beta`	A matrix containing the coefficient estimates along the solution path, one column for each step.
`beta`	A numeric vector of length p giving the coefficient estimates at the EBIC-selected optimal tuning parameter.
`lambda`	A numeric vector containing the sequence of ℓ_1 tuning parameters along the path.
`direction`	A numeric vector indicating the sign of the change in the updated coefficient at each iteration. `1` indicates a forward step; `0` indicates a backward step.
`iter`	Integer scalar giving the number of iterations.
`ebic`	A numeric vector of EBIC values along the solution path.
`opt`	Integer index of the EBIC-selected optimal model along the path.

References

Shi, X., Huang, Y., Huang, J., and Ma, S. (2018). A forward and backward stagewise algorithm for nonconvex loss functions with adaptive lasso. *Computational Statistics & Data Analysis*, **124**, 235–251.

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

Tan, X., Zhang, X., Cui, Y., and Liu, X. (2024). Uncertainty quantification in high-dimensional linear models incorporating graphical structures with applications to gene set analysis. *Bioinformatics*, **40**(9), btae541.

Examples

```
set.seed(123)

## Simulate data (see Data_simu for details)
dat <- Data_simu(n = 50, p = 100, s0 = 5,
                 xcov = "AR1", error = "norm", signal = 1)
X <- dat$X
y <- dat$y
G <- dat$G

fit <- Lfabs(X, y, lambda2 = 1, G = G, h = 1)
beta_hat <- fit$beta
```

Index