Package 'MSN'

June 4, 2026

Type Package

Title Multivariate Survival Data with Network Structures

Version 0.1.0

Description Implements a semi-parametric estimation framework combined with a boosting algorithm to marginally estimate the conditional cumulative distribution function of survival times given informative covariates. It then utilizes the graphical lasso method to reconstruct network structures among multivariate time-to-event variables, accommodating both multivariate outcomes measured within a single dataset and survival times integrated from heterogeneous (multi-source) datasets..

License GPL-3

Encoding UTF-8

Imports MASS, glasso, survival, stats

RoxygenNote 7.3.2

NeedsCompilation no

Author Li-Pang Chen [aut, cre]

Maintainer Li-Pang Chen <lchen723@nccu.edu.tw>

Repository CRAN

Date/Publication 2026-06-04 12:10:09 UTC

Contents

MSN-package *Multivariate Survival Data with Network Structures*

Description

Implements a semi-parameric estimation framework combined with a boosting algorithm to marginally estimate conditional cumulative distribution functions of survival times given informative covariates. Based on these marginal estimates, the graphical lasso method is applied to reconstruct network structures among multivariate survival outcomes.

Details

The MSN package provides robust functions to analyze multivariate time-to-event outcomes while characterizing their underlying network dependencies.

The package features a cross-validated semi-parametric boosting approach to estimate marginal conditional cumulative distribution functions for survival data under covariate adjustment. The functions offer multiple algorithmic options for users, including the specification of boosting steps, learning rates, and constants for thresholding small values in the estimators. Based on the marginalized estimates, the implied covariance structure is determined via pairwise Kendall's tau. The graphical lasso is subsequently implemented to reconstruct network structures, fully supporting data integrated from heterogeneous sources as well as multiple outcomes within a single dataset.

Author(s)

Chen, L.-P.

Maintainer: Li-Pang Chen <lchen723@nccu.edu.tw>

network_estimate *The network estimation for multiple survival times*

Description

Identifies the network structure among multiple survival times or time-to-event variables, accommodating both multivariate outcomes within a single dataset and data integrated from heterogeneous sources.

Usage

```
network_estimate(Y, delta, X, num = 10, rho = 0.1, max_iter = 2000,
                 learning.rate = 0.5, kappa=0.5, stop_value = 0.001)
```

Arguments

`Y`	A list of multivariate survival times or events across samples, utilized for training the estimated conditional cumulative distribution functions.
`delta`	A list of censoring indicators for multivariate time-to-event outcomes, used for training the estimated conditional cumulative distribution functions.
`X`	A list of covariate matrices for the multiple time-to-event processes, used for training the estimated conditional cumulative distribution functions.
`num`	A positive integer specifying the number of grid points used to evaluate the integral of the estimated cumulative distribution function.
`rho`	A positive tuning parameter controlling the sparsity of the graphical lasso estimator.
`max_iter`	A positive integer indicating the total number of boosting iterations to be executed.
`learning.rate`	A positive step-size parameter (learning rate) that scales the contribution of each boosting iteration.
`kappa`	A positive thresholding constant used to zero out small estimated coefficients in the β estimator.
`stop_value`	A positive tolerance value used as the convergence criterion for early stopping the boosting procedure.

Details

This function operates in three sequential steps. It first adopts `semi_estimation` to compute the marginal cumulative distribution functions for each time-to-event variable. These estimated distributions are then utilized to evaluate Kendall's tau for pairwise outcomes. Finally, the graphical lasso method is implemented to reconstruct the underlying network structure based on the pairwise association matrix.

Value

`cov_matrix`	An $J \times J$ covariance matrix where the elements are determined by the pairwise Kendall's tau coefficients.
`pre_matrix`	An $J \times J$ precision matrix determined by the graphical lasso.
`num_edge`	A positive number of edges in the estimated graph.

Author(s)

Chen, L.-P.

Examples

```
library(MASS)
p = 5
J = 5
n = 10
```

```
X1 = mvrnorm(n, rep(0, p), diag(1, p))

beta = matrix(0, nrow = p, ncol = J)
for (j in 1:J) {
  beta[sample(1:p, 3), j] <- 1
}

P = matrix(0,J,J)
P[1,2]=1; P[3,4:5]=1
P[2,1]=1; P[4:5,3]=1

diag(P) = max(eigen(P)$value)+0.1
Sigma = cov2cor(solve(P))
Z = mvrnorm(n, mu = rep(0, J), Sigma=Sigma)
U = pnorm(Z)

event = NULL
event[[1]] = -log(1 - U[,1]) / exp(X1 %*% beta[,1])
event[[2]] = (-log(1 - U[,2]) /  exp(X1 %*% beta[,2]))
event[[3]] = sqrt(-2 * log(1 - U[,3]) / exp(X1%*%beta[,3]))
event[[4]] = exp(X1%*%beta[,4] + qnorm(U[,4]))
event[[5]] = exp(X1%*%beta[,5] + qnorm(U[,5]))


Y = NULL; delta = NULL; X = NULL
for(j in 1:J) {
Y[[j]] = pmin(event[[j]], rexp(n,1))
delta[[j]] = (event[[j]]< rexp(n,1))*1
X[[j]] = X1
}


net_est = network_estimate(Y = Y, delta = delta, X = X, rho = 0.05)
```

`network_estimate_BIC` *The Bayesian information criterion for determining the tuning parameter*

Description

Implements the Bayesian Information Criterion (BIC) to select the optimal tuning parameter for the graphical lasso method.

Usage

```
network_estimate_BIC(Y, delta, X, num = 10, rho_seq = seq(0,0.5,length=10),
                     max_iter = 2000, learning.rate = 0.5, kappa=0.5,
                     stop_value = 0.001)
```

Arguments

Y	A list of multivariate survival times or events across samples, utilized for training the estimated conditional cumulative distribution functions.
delta	A list of censoring indicators for multivariate time-to-event outcomes, used for training the estimated conditional cumulative distribution functions.
X	A list of covariate matrices for the multiple time-to-event processes, used for training the estimated conditional cumulative distribution functions.
num	A positive integer specifying the number of grid points used to evaluate the integral of the estimated cumulative distribution function.
rho_seq	A user-specified numeric vector representing the candidate sequence of tuning parameters for the graphical lasso.
max_iter	A positive integer indicating the total number of boosting iterations to be executed.
learning.rate	A positive step-size parameter (learning rate) that scales the contribution of each boosting iteration.
kappa	A positive thresholding constant used to zero out small estimated coefficients in the β estimator.
stop_value	A positive tolerance value used as the convergence criterion for early stopping the boosting procedure.

Details

The function conducts a grid search across candidate tuning parameters to compute their corresponding BIC scores. By balancing model fit against the number of non-zero edges, it determines the optimal tuning parameter for the graphical lasso algorithm. Users can subsequently pass this optimal parameter into `network_estimate` to perform the final network structure detection and visualization.

Value

bic_seq	A numeric vector of BIC values corresponding to each candidate tuning parameter specified in `rho_seq`. The length of this vector matches the length of `rho_seq`.
opt_rho	The optimal tuning parameter selected via the minimization of the BIC criterion.
pre_matrix	An $J \times J$ precision matrix determined by the graphical lasso.
num_edge	A positive number of edges in the estimated graph.

Author(s)

Chen, L.-P.

Examples

```
library(MASS)
p = 5
J = 5
n = 10

X1 = mvrnorm(n, rep(0, p), diag(1, p))

beta = matrix(0, nrow = p, ncol = J)
for (j in 1:J) {
  beta[sample(1:p, 3), j] <- 1
}

P = matrix(0,J,J)
P[1,2]=1; P[3,4:5]=1
P[2,1]=1; P[4:5,3]=1

diag(P) = max(eigen(P)$value)+0.1
Sigma = cov2cor(solve(P))
Z = mvrnorm(n, mu = rep(0, J), Sigma=Sigma)
U = pnorm(Z)

event = NULL
event[[1]] = -log(1 - U[,1]) / exp(X1 %*% beta[,1])
event[[2]] = (-log(1 - U[,2]) /  exp(X1 %*% beta[,2]))
event[[3]] = sqrt(-2 * log(1 - U[,3]) / exp(X1%*%beta[,3]))
event[[4]] = exp(X1%*%beta[,4] + qnorm(U[,4]))
event[[5]] = exp(X1%*%beta[,5] + qnorm(U[,5]))


Y = NULL; delta = NULL; X = NULL
for(j in 1:J) {
Y[[j]] = pmin(event[[j]], rexp(n,1))
delta[[j]] = (event[[j]]< rexp(n,1))*1
X[[j]] = X1
}


net_est_bic = network_estimate_BIC(Y = Y, delta = delta, X = X,
                                   rho_seq = seq(0,0.8,length=5))
```

`semi_estimation` *Estimation of the cumulative distribution function of the survival time given the covariates.*

Description

Implements a gradient boosting algorithm to select informative covariates and marginally estimate the conditional cumulative distribution functions of survival times. Based on the fitted model, this function further evaluates predicted cumulative distribution values for user-specified survival times and covariate profiles.

Usage

```
semi_estimation(y_vec, X_target, Y_train, Delta_train, X_train, max_iter = 2000,
                learning.rate = 0.5, kappa=0.5, stop_value = 0.001)
```

Arguments

y_vec	A user-specified numeric vector or matrix at which the estimated conditional cumulative distribution function is to be evaluated.
X_target	A user-specified matrix or data frame of new covariate profiles, used for evaluating the estimated conditional cumulative distribution function.
Y_train	A list of multivariate survival times or events across samples, utilized for training the estimated conditional cumulative distribution functions.
Delta_train	A list of censoring indicators for multivariate time-to-event outcomes, used for training the estimated conditional cumulative distribution functions.
X_train	A list of covariate matrices for the multiple time-to-event processes, used for training the estimated conditional cumulative distribution functions.
max_iter	A positive integer indicating the total number of boosting iterations to be executed.
learning.rate	A positive step-size parameter (learning rate) that scales the contribution of each boosting iteration.
kappa	A positive thresholding constant used to zero out small estimated coefficients in the β estimator.
stop_value	A positive tolerance value used as the convergence criterion for early stopping the boosting procedure.

Details

The function executes a semi-parametric boosting approach to identify critical risk factors and marginally estimate the conditional cumulative distribution function. After finalizing the optimization process, the resulting estimator can be further queried by the user. By passing designated survival times and new covariate matrices into the corresponding arguments, the function computes the predicted marginal cumulative probabilities, facilitating flexible post-estimation analysis.

Value

est_beta	A p-dimensional numeric vector containing the estimated coefficients (estimates) for the covariates.
est_F	A numeric matrix of the estimated conditional cumulative distribution function values, evaluated at the specified survival time-points y_vec and covariate profiles X_target.

Author(s)

Chen, L.-P.

Examples

```
library(MASS)
p = 5
J = 5
n = 10

X1 = mvrnorm(n, rep(0, p), diag(1, p))

beta = matrix(0, nrow = p, ncol = J)
for (j in 1:J) {
  beta[sample(1:p, 3), j] <- 1
}

P = matrix(0,J,J)
P[1,2]=1; P[3,4:5]=1
P[2,1]=1; P[4:5,3]=1

diag(P) = max(eigen(P)$value)+0.1
Sigma = cov2cor(solve(P))
Z = mvrnorm(n, mu = rep(0, J), Sigma=Sigma)
U = pnorm(Z)

event = NULL
event[[1]] = -log(1 - U[,1]) / exp(X1 %*% beta[,1])
event[[2]] = (-log(1 - U[,2]) /  exp(X1 %*% beta[,2]))
event[[3]] = sqrt(-2 * log(1 - U[,3]) / exp(X1%*%beta[,3]))
event[[4]] = exp(X1%*%beta[,4] + qnorm(U[,4]))
event[[5]] = exp(X1%*%beta[,5] + qnorm(U[,5]))


Y = NULL; delta = NULL; X = NULL
for(j in 1:J) {
Y[[j]] = pmin(event[[j]], rexp(n,1))
delta[[j]] = (event[[j]]< rexp(n,1))*1
X[[j]] = X1
}

semi_est = semi_estimation(y_vec = seq(0.1,max(Y[[J]]),length=10), X_target = X[[J]],
     Y_train = Y[[J]], Delta_train = delta[[J]], X_train = X[[J]])
```

Index