Package 'sox'

March 22, 2023

Type Package

Title Structured Learning in Time-Dependent Cox Models

Version 1.0

Date 2023-03-21

Description Efficient procedures for fitting and cross-validating the structurally-regularized time-dependent Cox models. The penalty term is a weighted sum of infinity norms of (overlapping) groups of coefficients, which can select variables following a user-specified grouping structure.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.5.0), survival, glmnet

Imports Rcpp (>= 1.0.10)

LinkingTo Rcpp

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.2.3

LazyData true

Copyright file inst/COPYRIGHTS

NeedsCompilation yes

Author Yi Lian [aut, cre],
Guanbo Wang [aut],
Archer Y. Yang [aut],
Mireille E. Schnitzer [aut],
Robert W. Platt [aut],
Rui Wang [aut],
Marc Dorais [aut],
Sylvie Perreault [aut],
Julien Mairal [ctb],
Yuansi Chen [ctb]

Maintainer Yi Lian <yi.lian@mail.mcgill.ca>

Repository CRAN

Date/Publication 2023-03-22 09:00:02 UTC

R topics documented:

`sox-package` *sox*

Description

Efficient procedures for fitting and cross-validating the structurally-regularized time-dependent Cox models. The penalty term is a weighted sum of infinity norms of (overlapping) groups of coefficients, which can select variables following a user-specified grouping structure.

`plot_sox_cv` *plots for* `sox_cv`

Description

Plot the cross-validation curves produced by `sox_cv`.

Usage

```
plot_sox_cv(sox_cv_obj)
```

Arguments

`sox_cv_obj`	The `sox_cv` object.

Value

The plot is the `cvm` (red dot) for each lambda with its standard error (vertical bar). The two vertical dashed lines corresponds to the `lambda.min` and `lambda.1se`

See Also

`sox`, `sox_cv`.

Examples

```
grp <- matrix(c(0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,
                1, 1, 0, 0, 0,
                0, 0, 0, 0, 0,
                0, 1, 0, 1, 0),
              ncol = 5, byrow = TRUE)
grp.var <- matrix(c(1, 0, 0, 0, 0, #A1
                    1, 0, 0, 0, 0, #A2
                    0, 0, 0, 1, 0, #C1
                    0, 0, 0, 1, 0, #C2
                    0, 1, 0, 0, 0, #B
                    0, 0, 1, 0, 0, #A1B
                    0, 0, 1, 0, 0, #A2B
                    0, 0, 0, 0, 1, #C1B
                    0, 0, 0, 0, 1  #C2B
                   ), ncol = 5, byrow = TRUE)
eta_g <- rep(1, 5)
x <- as.matrix(sim[, c("A1","A2","C1","C2","B",
                       "A1B","A2B","C1B","C2B")])
lam.seq <- 10^seq(0, -2, by = -0.2)

cv <- sox_cv(x = x,
                ID = sim$Id,
                time = sim$Start,
                time2 = sim$Stop,
                event = sim$Event,
                lambda = lam.seq,
                group = grp,
                group_variable = grp.var,
                penalty_weights = eta_g,
                nfolds = 5,
                tol = 1e-4,
                maxit = 1e3,
                verbose = FALSE)
plot_sox_cv(cv)
```

---

plot_sox_sp *plots for* sox *and* sox_cv

Description

Plot the solution path produced by sox or sox_cv.

Usage

```
plot_sox_sp(
  sox_obj,
  plot_min = FALSE,
```

```
  plot_1se = FALSE,
  type = "l",
  log = "x",
  ...
)
```

Arguments

sox_obj	The sox or sox_cv object.
plot_min	Logical, whether a vertical line at lambda.min acquired by sox_cv is plotted. Not available if sox_obj is a sox fit.
plot_1se	Logical, whether a vertical line at lambda.1se acquired by sox_cv is plotted. Not available if sox_obj is a sox fit.
type	Graphical argument to be passed to matplot, a character string (length 1 vector) or vector of 1-character strings indicating the type of plot for each column of y, see plot.default for all possible types. Default is "l" for lines.
log	Graphical argument to be passed to matplot, a character string which contains "x" if the x axis is to be logarithmic, "y" if the y axis is to be logarithmic, "" if neither, "xy" or "yx" if both axes are to be logarithmic. Default is "x".
...	Further arguments of matplot and ultimately of plot.default for some.

Value

Produces a coefficient profile plot of the coefficient paths for a fitted sox or sox_cv object.

See Also

sox, sox_cv.

Examples

```
grp <- matrix(c(0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,
                1, 1, 0, 0, 0,
                0, 0, 0, 0, 0,
                0, 1, 0, 1, 0),
              ncol = 5, byrow = TRUE)
grp.var <- matrix(c(1, 0, 0, 0, 0, #A1
                    1, 0, 0, 0, 0, #A2
                    0, 0, 0, 1, 0, #C1
                    0, 0, 0, 1, 0, #C2
                    0, 1, 0, 0, 0, #B
                    0, 0, 1, 0, 0, #A1B
                    0, 0, 1, 0, 0, #A2B
                    0, 0, 0, 0, 1, #C1B
                    0, 0, 0, 0, 1  #C2B
                   ), ncol = 5, byrow = TRUE)
eta_g <- rep(1, 5)
x <- as.matrix(sim[, c("A1","A2","C1","C2","B",
```

```
                        "A1B","A2B","C1B","C2B")])
lam.seq <- 10^seq(0, -2, by = -0.2)
# plot solution path from a sox fit
fit <- sox(x = x,
              ID = sim$Id,
              time = sim$Start,
              time2 = sim$Stop,
              event = sim$Event,
              lambda = lam.seq,
              group = grp,
              group_variable = grp.var,
              penalty_weights = eta_g,
              tol = 1e-4,
              maxit = 1e3,
              verbose = FALSE)
plot_sox_sp(sox_obj = fit)

# plot solution path from a sox_cv fit
cv <- sox_cv(x = x,
               ID = sim$Id,
               time = sim$Start,
               time2 = sim$Stop,
               event = sim$Event,
               lambda = lam.seq,
               group = grp,
               group_variable = grp.var,
               penalty_weights = eta_g,
               nfolds = 5,
               tol = 1e-4,
               maxit = 1e3,
               verbose = FALSE)
plot_sox_sp(sox_obj = cv, plot_min = TRUE, plot_1se = TRUE)
```

`sim` *A simulated demo dataset* `sim`

Description

A simulated demo dataset `sim`

Usage

```
data(sim)
```

Format

A simulated data frame that is used to illustrate the use of the sox package. The max follow-up time for each subject is set to be 5. The total number of subject is 50.

Id The ID of each subject.

Event During the time from `Start` to `Stop`, if the subject experience the event. We use the function `permalgorithm` in the `R` package `PermAlgo` to generate the Event.

Start Start time.

Stop Stop time.

Fup The total follow-up time for the subject.

Covariates A1, A2, C1, C2, B, A1B, A2B, C1B, C2B. The dataset contains 5 variables (9 columns after one-hot encoding). Variable A is a e 3-level categorical variable, which results in 2 binary variables (A1 and A2), the same with the variable C. B is a continuous variable. The interaction term AB and CB are also two 3-level categorical variables. The code for generating the covariates is given below.

See Also

`PermAlgo`

Examples

```
# generate B
gen_con=function(m){
 X=rnorm(m/5)
 XX=NULL
 for (i in 1:length(X)) {
   if (length(XX)<m){
   X.rep=rep(X[i],round(runif(1,5,10),0))
   XX=c(XX,X.rep)
   }
 }
 return(XX[1:m])
}
# generate A and C
gen_cat=function(m){
 X=sample.int(3, m/5,replace = TRUE)
 XX=NULL
 for (i in 1:length(X)) {
   if (length(XX)<m){
     X.rep=rep(X[i],round(runif(1,5,10),0))
     XX=c(XX,X.rep)
   }
 }
 return(XX[1:m])
}

# generate covariate for one subject
gen_X=function(m){
 A=gen_cat(m);B=gen_con(m);C=gen_cat(m)
 A1=ifelse(A==1,1,0);A2=ifelse(A==2,1,0)
 C1=ifelse(C==1,1,0);C2=ifelse(C==2,1,0)
 A1B=A1*B;A2B=A2*B
 C1B=C1*B;C2B=C2*B
 return(as.matrix(cbind(A1,A2,C1,C2,B,A1B,A2B,C1B,C2B)))
}
```

```
# generate covariate for all subject
gen_X_n=function(m,n){
 Xn=NULL
 for (i in 1:n) {
   X=gen_X(m)
   Xn=rbind(Xn,X)
 }
 return(Xn)
}

 n=50;m=5
 covariates=gen_X_n(m,n)
 # generate outcomes
 # library(PermAlgo)
 # data <- permalgorithm(n, m, covariates,
 #                       XmatNames = c("A1","A2","C1","C2","B","A1B","A2B","C1B","C2B"),
 #                       #change according to scenario 1/2
 #                     betas = c(rep(log(3),2),rep(0,2), log(4), rep(log(3),2),rep(0,2)),
 #                       groupByD=FALSE )
 # fit.original = coxph(Surv(Start, Stop, Event) ~ . ,data[,-c(1,3)])
```

---

sox *fit a (time-dependent) Cox model with structured variable selection*

Description

Fit a (time-dependent) Cox model via penalized maximum likelihood, where the penalization is a weighted sum of infinity norm of (overlapping) groups of coefficients. The regularization path is computed at a grid of values for the regularization parameter lambda.

Usage

```
sox(
  x,
  ID,
  time,
  time2,
  event,
  lambda,
  group,
  group_variable,
  penalty_weights,
  par_init,
  stepsize_init = 1,
  stepsize_shrink = 0.8,
  tol = 1e-05,
  maxit = 1000L,
  verbose = FALSE
)
```

Arguments

x	Predictor matrix with dimension $nm*p$, where n is the number of subjects, m is the maximum observation time, and p is the number of predictors. See Details.
ID	The ID of each subjects, each subject has one ID (many rows in x share one ID).
time	Represents the start of each time interval.
time2	Represents the stop of each time interval.
event	Indicator of event. event = 1 when event occurs and event = 0 otherwise.
lambda	Sequence of regularization coefficients λ's.
group	$G*G$ matrix describing the relationship between the groups of variables, where G represents the number of groups. Denote the i-th group of variables by g_i. The (i,j) entry is 1 if and only if $i \neq j$ and g_i is a child group (subset) of g_j, and is 0 otherwise. See Examples and Details.
group_variable	$p*G$ matrix describing the relationship between the groups and the variables. The (i,j) entry is 1 if and only if variable i is in group g_j, but not in any child group of g_j, and is 0 otherwise. See Examples and Details.
penalty_weights	Optional, vector of length G specifying the group-specific penalty weights. If not specified, the default value is $\mathbf{1}_G$. Modify with caution.
par_init	Optional, vector of initial values of the optimization algorithm. Default initial value is zero for all p variables.
stepsize_init	Initial value of the stepsize of the optimization algorithm. Default is 1.
stepsize_shrink	Factor in $(0,1)$ by which the stepsize shrinks in the backtracking linesearch. Default is 0.8.
tol	Convergence criterion. Algorithm stops when the l_2 norm of the difference between two consecutive updates is smaller than tol.
maxit	Maximum number of iterations allowed.
verbose	Logical, whether progress is printed.

Details

The predictor matrix should be of dimension $nm*p$. Each row records the values of covariates for one subject at one time, for example, the values at the day from time (Start) to time2 (Stop). An example dataset sim is provided. The dataset has the same format produced by the R package **PermAlgo**. The specification of arguments group and group_variable for the grouping structure can be found in http://thoth.inrialpes.fr/people/mairal/spams/doc-R/html/doc_spams006.html#sec27, the same as the grouping structure specification in the R package **spams**.

In the Examples below, $p = 9, G = 5$, the group structure is:

$$g_1 = \{A_1, A_2, A_1B, A_2B\},$$
$$g_2 = \{B, A_1B, A_2B, C_1B, C_2B\},$$
$$g_3 = \{A_1B, A_2B\},$$
$$g_4 = \{C_1, C_2, C_1B, C_2B\},$$
$$g_5 = \{C_1B, C_2B\}.$$

where g_3 is a subset of g_1 and g_2, and g_5 is a subset of g_2 and g_4.

Value

A list with the following three elements.

`lambdas`	The user-specified regularization coefficients `lambda` sorted in decreasing order.
`estimates`	A matrix, with each column corresponding to the coefficient estimates at each λ in `lambdas`.
`iterations`	A vector of number of iterations it takes to converge at each λ in `lambdas`.

Examples

```
# g3 in g1 -> grp_31 = 1
# g3 in g2 -> grp_32 = 1
# g5 in g2 -> grp_52 = 1
# g5 in g4 -> grp_54 = 1
grp <- matrix(c(0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,
                1, 1, 0, 0, 0,
                0, 0, 0, 0, 0,
                0, 1, 0, 1, 0),
              ncol = 5, byrow = TRUE)

# Variable A1 is in g1 only: grp.var_11 = 1
# Variable A1B is in g1 and g3, but g3 is a child group of g1,
# so grp.var_63 = 1 while grp.var_61 = 0.
grp.var <- matrix(c(1, 0, 0, 0, 0, #A1
                    1, 0, 0, 0, 0, #A2
                    0, 0, 0, 1, 0, #C1
                    0, 0, 0, 1, 0, #C2
                    0, 1, 0, 0, 0, #B
                    0, 0, 1, 0, 0, #A1B
                    0, 0, 1, 0, 0, #A2B
                    0, 0, 0, 0, 1, #C1B
                    0, 0, 0, 0, 1  #C2B
                   ), ncol = 5, byrow = TRUE)
eta_g <- rep(1, 5)
x <- as.matrix(sim[, c("A1","A2","C1","C2","B",
                       "A1B","A2B","C1B","C2B")])
lam.seq <- 10^seq(0, -2, by = -0.2)

fit <- sox(x = x,
              ID = sim$Id,
              time = sim$Start,
              time2 = sim$Stop,
              event = sim$Event,
              lambda = lam.seq,
              group = grp,
              group_variable = grp.var,
              penalty_weights = eta_g,
              tol = 1e-4,
              maxit = 1e3,
              verbose = FALSE)
```

sox_cv *cross-validation for* sox

Description

Conduct cross-validation (cv) for sox.

Usage

```
sox_cv(
  x,
  ID,
  time,
  time2,
  event,
  lambda,
  group,
  group_variable,
  penalty_weights,
  par_init,
  nfolds = 10,
  stepsize_init = 1,
  stepsize_shrink = 0.8,
  tol = 1e-05,
  maxit = 1000L,
  verbose = FALSE
)
```

Arguments

x	Predictor matrix with dimension $nm*p$, where n is the number of subjects, m is the maximum observation time, and p is the number of predictors. See Details.
ID	The ID of each subjects, each subject has one ID (many rows in x share one ID).
time	Represents the start of each time interval.
time2	Represents the stop of each time interval.
event	Indicator of event. event = 1 when event occurs and event = 0 otherwise.
lambda	Sequence of regularization coefficients λ's.
group	$G*G$ matrix describing the relationship between the groups of variables, where G represents the number of groups. Denote the i-th group of variables by g_i. The (i,j) entry is 1 if and only if $i \neq j$ and g_i is a child group (subset) of g_j, and is 0 otherwise. See Examples and Details.
group_variable	$p*G$ matrix describing the relationship between the groups and the variables. The (i,j) entry is 1 if and only if variable i is in group g_j, but not in any child group of g_j, and is 0 otherwise. See Examples and Details.

`penalty_weights`
: Optional, vector of length G specifying the group-specific penalty weights. If not specified, the default value is $\mathbf{1}_G$. Modify with caution.

`par_init`
: Optional, vector of initial values of the optimization algorithm. Default initial value is zero for all p variables.

`nfolds`
: Optional, the folds of cross-validation. Default is 10.

`stepsize_init`
: Initial value of the stepsize of the optimization algorithm. Default is 1.

`stepsize_shrink`
: Factor in $(0, 1)$ by which the stepsize shrinks in the backtracking linesearch. Default is 0.8.

`tol`
: Convergence criterion. Algorithm stops when the l_2 norm of the difference between two consecutive updates is smaller than `tol`.

`maxit`
: Maximum number of iterations allowed.

`verbose`
: Logical, whether progress is printed.

Details

For each lambda, 10 folds cross-validation (by default) is performed. The cv error is defined as follows. Suppose we perform K-fold cross-validation, denote $\hat{\beta}^{-k}$ by the estimate obtained from the rest of $K-1$ folds (training set). The error of the k-th fold (test set) is defined as $2(P-Q)$ divided by R, where P is the log partial likelihood evaluated at $\hat{\beta}^{-k}$ using the entire dataset, Q is the log partial likelihood evaluated at $\hat{\beta}^{-k}$ using the training set, and R is the number of events in the test set. We do not use the negative log partial likelihood evaluated at $\hat{\beta}^{-k}$ using the test set because the former definition can efficiently use the risk set, and thus it is more stable when the number of events in each test set is small (think of leave-one-out). The cv error is used in parameter tuning. To account for balance in outcomes among the randomly formed test set, we divide the deviance $2(P-Q)$ by R. To get the estimated coefficients that has the minimum cv error, use `sox()$Estimates[sox()$Lambdas==sox_cv()$lambda.min]`. To apply the 1-se rule, use `sox()$Estimates[sox()$Lambdas==sox_cv()$lambda.1se]`.

Value

A list.

`lambdas`
: A vector of lambda used for each cross-validation.

`cvm`
: The cv error averaged across all folds for each lambda.

`cvsd`
: The standard error of the cv error for each lambda.

`cvup`
: The cv error plus its standard error for each lambda.

`cvlo`
: The cv error minus its standard error for each lambda.

`nzero`
: The number of non-zero coefficients at each lambda.

`sox.fit`
: A sox fit for the full data at all lambdas.

`lambda.min`
: The lambda such that the `cvm` reach its minimum.

`lambda.1se`
: The maximum of lambda such that the `cvm` is less than the minimum the `cvup` (the minmum of `cvm` plus its standard error).

See Also

sox, plot_sox_cv.

Examples

```
grp <- matrix(c(0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,
                1, 1, 0, 0, 0,
                0, 0, 0, 0, 0,
                0, 1, 0, 1, 0),
              ncol = 5, byrow = TRUE)
grp.var <- matrix(c(1, 0, 0, 0, 0, #A1
                    1, 0, 0, 0, 0, #A2
                    0, 0, 0, 1, 0, #C1
                    0, 0, 0, 1, 0, #C2
                    0, 1, 0, 0, 0, #B
                    0, 0, 1, 0, 0, #A1B
                    0, 0, 1, 0, 0, #A2B
                    0, 0, 0, 0, 1, #C1B
                    0, 0, 0, 0, 1  #C2B
                   ), ncol = 5, byrow = TRUE)
eta_g <- rep(1, 5)
x <- as.matrix(sim[, c("A1","A2","C1","C2","B",
                       "A1B","A2B","C1B","C2B")])
lam.seq <- 10^seq(0, -2, by = -0.2)

cv <- sox_cv(x = x,
                ID = sim$Id,
                time = sim$Start,
                time2 = sim$Stop,
                event = sim$Event,
                lambda = lam.seq,
                group = grp,
                group_variable = grp.var,
                penalty_weights = eta_g,
                nfolds = 5,
                tol = 1e-4,
                maxit = 1e3,
                verbose = FALSE)
```

Index