

Package ‘LTRCforests’

July 3, 2020

Version 0.5.0

Date 2020-07-01

Title Ensemble Methods for Survival Data with Time-Varying Covariates

Author Weichi Yao [aut, cre],
Halina Frydman [aut],
Denis Larocque [aut],
Jeffrey S. Simonoff [aut]

Maintainer Weichi Yao <wy635@stern.nyu.edu>

Depends R (>= 3.5.0)

Imports stats, utils, survival, ipred, parallel, prodlim, partykit

Suggests randomForestSRC, LTRCtrees

Description Implements the conditional inference forest and random survival forest algorithm to modeling left-truncated right-censored data with time-invariant covariates, and (left-truncated) right-censored survival data with time-varying covariates. It also provides functions to tune the parameters and evaluate the model fit. See Yao et al. (2020) <arXiv:2006.00567>.

NeedsCompilation yes

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Repository CRAN

Date/Publication 2020-07-03 10:50:10 UTC

R topics documented:

LTRCforests-package	2
ltrccf	3
ltrcrsf	5
pbcsample	8
predictProb	9

print	10
sbrier_ltrc	11
tune.ltrccf	12
tune.ltrcrsf	15

Index	18
--------------	-----------

LTRCforests-package	<i>Constructs forest methods for left-truncated and right-censored (LTRC) survival data</i>
---------------------	---

Description

Constructs a LTRC conditional inference forest (LTRCCF) or a LTRC random survival forest (LTRCRSF) for left-truncated and right-censored data, it also allows for (left-truncated) right-censored survival data with time-varying covariates. The main functions of this package are [ltrccf](#) and [ltrcrsf](#).

Details

Problem setup and existing methods: Continuous-time survival data with time-varying covariates are common in practice. Methods like the Cox proportional hazards model rely on restrictive assumptions such as proportional hazards and a log-linear relationship between the hazard function and covariates. Furthermore, because these methods are often parametric, nonlinear effects of variables must be modeled by transformations or expanding the design matrix to include specialized basis functions for more complex data structures in real world applications. The functions [LTRCIT](#) and [LTRCART](#) provide a conditional inference tree method and a survival tree method for left-truncated right-censored survival data, which also allows for right-censored survival data with time-varying covariates. Tree estimators are nonparametric and as such often exhibit low bias and high variance. Ensemble methods like bagging and random forest can reduce variance while preserving low bias.

LTRC forests: This package implements [ltrccf](#) and [ltrcrsf](#). [ltrccf](#) extends the conditional inference forest (see [cforest](#)) to LTRC survival data. It uses LTRC conditional inference survival trees (see [LTRCIT](#)) as base learners. [ltrcrsf](#) extends the random survival forest (see [rfsrf](#)) to left-truncated right-censored survival data. It uses LTRC survival trees with Poisson splitting rule (see [LTRCART](#)) as base learners. The main functions [ltrccf](#) and [ltrcrsf](#) fit a corresponding LTRC forest for LTRC data, with parameter `mtry` tuned by [tune.ltrccf](#) or [tune.ltrcrsf](#). This tuning procedure relies on the evaluation of the out-of-bag errors, which is performed by the function [sbrier_ltrc](#). [print](#) prints summary output for [ltrccf](#) objects and [ltrcrsf](#) objects. [predictProb](#) constructs survival function estimates for [ltrccf](#) objects and [ltrcrsf](#) objects.

For (left-truncated) right-censored survival data with time-varying covariates, one can first reformat the data structure to one with LTRC observations, where the multiple records of a subject become a list of pseudo-subjects and are treated independently. This procedure is usually referred to as the Andersen-Gill method (Andersen and Gill, 1982). Then LTRC forest methods can be applied on this reformatted dataset.

Overall, the methods in this package can handle all combinations of left truncation, right censoring, time-invariant covariates, and time-varying covariates. If one is in the traditional case with right censored data and time-invariant covariates, however, then it is recommended to use the functions [cforest](#) and [rfsrc](#) directly to construct conditional inference forests and random survival forests, respectively.

References

Andersen, P. and Gill, R. (1982). Cox's regression model for counting processes, a large sample study. *Annals of Statistics*, **10**, 1100-1120.

See Also

[ltrccf](#), [ltrcrsf](#), [predictProb](#), [print](#), [tune.ltrccf](#), [tune.ltrcrsf](#), [sbrier_ltrc](#)

ltrccf

Fit a LTRC conditional inference forest

Description

An implementation of the random forest and bagging ensemble algorithms utilizing LTRC conditional inference trees [LTRCIT](#) as base learners for left-truncated right-censored survival data with time-invariant covariates. It also allows for (left-truncated) right-censored survival data with time-varying covariates.

Usage

```
ltrccf(
  formula,
  data,
  id,
  mtry = NULL,
  ntree = 100L,
  bootstrap = c("by.sub", "by.root", "by.user", "none"),
  samptype = c("swor", "swr"),
  sampfrac = 0.632,
  samp = NULL,
  na.action = "na.omit",
  stepFactor = 2,
  trace = TRUE,
  applyfun = NULL,
  cores = NULL,
  control = partykit::ctree_control(teststat = "quad", testtype = "Univ", minsplit =
    max(ceiling(sqrt(nrow(data))), 20), minbucket = max(ceiling(sqrt(nrow(data))), 7),
    minprob = 0.01, mincriterion = 0, saveinfo = FALSE)
)
```

Arguments

formula	a formula object, with the response being a <code>Surv</code> object, with form <code>Surv(tleft,tright,event)</code> .
data	a data frame containing n rows of left-truncated right-censored observations. For time-varying data, this should be a data frame containing pseudo-subject observations based on the Andersen-Gill reformulation.
id	variable name of subject identifiers. If this is present, it will be searched for in the data data frame. Each group of rows in data with the same subject id represents the covariate path through time of a single subject. If not specified, the algorithm then assumes data contains left-truncated and right-censored survival data with time-invariant covariates.
mtry	number of input variables randomly sampled as candidates at each node for random forest algorithms. The default mtry is tuned by <code>tune.ltrccf</code> .
ntree	an integer, the number of the trees to grow for the forest. <code>ntree = 100L</code> is set by default.
bootstrap	bootstrap protocol. (1) If id is present, the choices are: "by.sub" (by default) which bootstraps subjects, "by.root" which bootstraps pseudo-subjects. Both can be with or without replacement (by default sampling is without replacement; see the option <code>perturb</code> below); (2) If id is not specified, it bootstraps the data by sampling with or without replacement. Regardless of the presence of id, if "none" is chosen, data is not bootstrapped at all, and is used in every individual tree. If "by.user" is chosen, the bootstrap specified by <code>samp</code> is used.
samptype	choices are <code>swor</code> (sampling without replacement) and <code>swr</code> (sampling with replacement). The default action here is sampling without replacement.
sampfrac	a fraction, determining the proportion of subjects to draw without replacement when <code>samptype = "swor"</code> . The default value is <code>0.632</code> . To be more specific, if id is present, $0.632 * N$ of subjects with their pseudo-subject observations are drawn without replacement (N denotes the number of subjects); otherwise, $0.632 * n$ is the requested size of the sample.
samp	Bootstrap specification when <code>bootstype = "by.user"</code> . Array of dim $n \times ntree$ specifying how many times each record appears in each bootstrap sample.
na.action	action taken if the data contains NA's. The default "na.omit" removes the entire record if any of its entries is NA (for x-variables this applies only to those specifically listed in formula). See function <code>cforest</code> for other available options.
stepFactor	at each iteration, mtry is inflated (or deflated) by this value, used when mtry is not specified (see <code>tune.ltrccf</code>). The default value is 2.
trace	whether to print the progress of the search of the optimal value of mtry, when mtry is not specified (see <code>tune.ltrccf</code>). <code>trace = TRUE</code> is set by default.
applyfun	an optional lapply-style function with arguments <code>function(X,FUN,...)</code> . It is used for computing the variable selection criterion. The default is to use the basic lapply function unless the <code>cores</code> argument is specified (see below). See <code>ctree_control</code> .
cores	numeric. See <code>ctree_control</code> .

`control` a list of control parameters, see [ctree_control](#). `control` parameters `minsplit`, `minbucket` have been adjusted from the [cforest](#) defaults. Other default values correspond to those of the default values used by [ctree_control](#).

Details

This function extends the conditional inference survival forest algorithm in [cforest](#) to fit left-truncated and right-censored data, which allow for time-varying covariates.

Value

An object belongs to the class `ltrccf`, as a subclass of [cforest](#).

References

Andersen, P. and Gill, R. (1982). Cox's regression model for counting processes, a large sample study. *Annals of Statistics*, **10**, 1100-1120.

See Also

[predictProb](#) for prediction and [tune.ltrccf](#) for mtry tuning.

Examples

```
#### Example with time-varying data pbcsample
library(survival)
Formula = Surv(Start, Stop, Event) ~ age + alk.phos + ast + chol + edema
## Fit an LTRCCF on the time-invariant data, with mtry tuned with stepFactor = 3.
LTRCCFobj = ltrccf(formula = Formula, data = pbcsample, ntree = 50L, stepFactor = 3)
```

ltrcrsf

Fit a LTRC random survival forest with Poisson splitting rule

Description

An implementation of the random forest algorithms utilizing LTRC `rpart` trees [LTRCART](#) as base learners for left-truncated right-censored survival data with time-invariant covariates. It also allows for (left-truncated) right-censored survival data with time-varying covariates.

Usage

```
ltrcrsf(
  formula,
  data,
  id,
  ntree = 100L,
  mtry = NULL,
  nodesize = max(ceiling(sqrt(nrow(data))), 15),
```

```

bootstrap = c("by.sub", "by.root", "by.node", "by.user", "none"),
samptype = c("swor", "swr"),
sampfrac = 0.632,
samp = NULL,
na.action = "na.omit",
stepFactor = 2,
trace = TRUE,
nodedepth = NULL,
nsplit = 10L,
ntime
)

```

Arguments

formula	a formula object, with the response being a Surv object, with form <code>Surv(tleft, tright, event)</code> .
data	a data frame containing n rows of left-truncated right-censored observations. For time-varying data, this should be a data frame containing pseudo-subject observations based on the Andersen-Gill reformulation.
id	variable name of subject identifiers. If this is present, it will be searched for in the data data frame. Each group of rows in data with the same subject id represents the covariate path through time of a single subject. If not specified, the algorithm then assumes data contains left-truncated and right-censored survival data with time-invariant covariates.
ntree	an integer, the number of the trees to grow for the forest. <code>ntree = 100L</code> is set by default.
mtry	number of input variables randomly sampled as candidates at each node for random forest like algorithms. The default <code>mtry</code> is tuned by tune.ltrcrsf .
nodesize	an integer, forest average terminal node size. It has been adjusted from the rfsrc default 15 for survival families.
bootstrap	bootstrap protocol. (1) If <code>id</code> is present, the choices are: "by.sub" (by default) which bootstraps subjects, "by.root" which bootstraps pseudo-subjects. Both can be with or without replacement (by default sampling is without replacement; see the option <code>samptype</code> below). (2) If <code>id</code> is not specified, the default is "by.root" which bootstraps the data by sampling with or without replacement; if "by.node" is chosen, data is bootstrapped with replacement at each node while growing the tree. Regardless of the presence of <code>id</code> , if "none" is chosen, data is not bootstrapped at all, and is used in every individual tree. If "by.user" is chosen, the bootstrap specified by <code>samp</code> is used.
samptype	choices are <code>swor</code> (sampling without replacement) and <code>swr</code> (sampling with replacement). The default action here is sampling without replacement.
sampfrac	a fraction, determining the proportion of subjects to draw without replacement when <code>samptype = "swor"</code> . The default value is 0.632. To be more specific, if <code>id</code> is present, $0.632 * N$ of subjects with their pseudo-subject observations are drawn without replacement (N denotes the number of subjects); otherwise, $0.632 * n$ is the requested size of the sample.

samp	Bootstrap specification when <code>bootstype = "by.user"</code> . Array of dim $n \times ntree$ specifying how many times each record appears in each bootstrap sample.
na.action	action taken if the data contains NA's. The default <code>"na.omit"</code> removes the entire record if any of its entries is NA (for x-variables this applies only to those specifically listed in <code>formula</code>). See function <code>rfsrc</code> for other available options.
stepFactor	at each iteration, <code>mtry</code> is inflated (or deflated) by this value, used when <code>mtry</code> is not specified (see <code>tune.ltrcrsf</code>). The default value is 2.
trace	whether to print the progress of the search of the optimal value of <code>mtry</code> if <code>mtry</code> is not specified (see <code>tune.ltrcrsf</code>). <code>trace = TRUE</code> is set by default.
nodedepth	maximum depth to which a tree should be grown. The default behaviour is that this parameter is ignored.
nsplit	an non-negative integer value for number of random splits to consider for each candidate splitting variable. This significantly increases speed. When zero or NULL, the algorithm uses much slower deterministic splitting where all possible splits are considered. <code>nsplit = 10L</code> by default.
ntime	an integer value used for survival to constrain ensemble calculations to a grid of <code>ntime</code> time points. Alternatively if a vector of values of length greater than one is supplied, it is assumed these are the time points to be used to constrain the calculations (note that the constrained time points used will be the observed event times closest to the user supplied time points). If no value is specified, the default action is to use all observed event times. Further details can be found in <code>rfsrc</code> .

Details

This function extends the random survival forest algorithm in `rfsrc` to fit left-truncated and right-censored data, which allows for time-varying covariates.

Value

An object belongs to the class `ltrcrsf`, as a subclass of `rfsrc`.

References

Andersen, P. and Gill, R. (1982). Cox's regression model for counting processes, a large sample study. *Annals of Statistics*, **10**, 1100-1120.

See Also

`predictProb` for prediction and `tune.ltrcrsf` for `mtry` tuning.

Examples

```
#### Example with time-varying data pbcSAMPLE
library(survival)
Formula = Surv(Start, Stop, Event) ~ age + alk.phos + ast + chol + edema
# Built a LTRCRSF forest (based on bootstrapping subjects without replacement)
# on the time-varying data by specifying id:
```

```
LTRCRSFobj = ltrcrsf(formula = Formula, data = pbcsample, id = ID, stepFactor = 3,
                      ntree = 30L)
LTRCRSFobj
```

pbcsample

Sample Mayo Clinic Primary Biliary Cirrhosis Data

Description

A sample real dataset with time varying covariates. It contains multiple records of measurements of risk factors at multiple time points from 10 patients with primary biliary cirrhosis (PBC), constructed from [pbcseq](#) in the R package [survival](#). The data structure has been reformatted with left-truncated right-censored pseudo-subject observations based on the Andersen-Gill reformulation method.

Value

a data frame with 57 rows and 9 variables:

ID	Patient id
Start	the left truncation time point for the corresponding pseudo-subject observation
Stop	the right censoring time point for the corresponding pseudo-subject observation
Event	a binary value, with 1 indicating the event occurring at the corresponding Stop, 0 indicating right-censored
age	patient's age at entry, in years
alk.phos	alkaline phosphatase (U/liter)
ast	aspartate aminotransferase (U/ml)
chol	serum cholesterol (mg/dl)
edema	0=no edema, 0.5=untreated or successfully treated, 1=edema despite diuretic therapy

Source

P. A. Murtaugh, E. R. Dickson, G. M. V. Dam, M. Malinchoc, P. M. Grambsch, A. L. Langworthy, and C. H. Gips. (1989). Primary biliary cirrhosis: Prediction of shortterm survival based on repeated patient visits. *Hepatology*, **20**, 126-134.

References

Andersen, P. and Gill, R. (1982). Cox's regression model for counting processes, a large sample study. *Annals of Statistics*, **10**, 1100-1120.

predictProb	<i>Compute a Survival Curve from a LTRCCF model or a LTRCRSF model</i>
-------------	--

Description

Constructs a monotone nonincreasing estimated survival curve from a LTRCCF model or a LTRCRSF model for any given (left-truncated) right-censored survival data with time-varying covariates. It can also compute survival function estimates for left-truncated right-censored data with time-invariant covariates.

Usage

```
predictProb(
  object,
  newdata = NULL,
  newdata.id,
  OOB = FALSE,
  time.eval,
  time.tau = NULL
)
```

Arguments

object	an object as returned by <code>ltrccf</code> or by <code>ltrcrsf</code> .
newdata	an optional data frame containing the test data (with the names of the variables the same as those in data from object).
newdata.id	optional variable name of subject identifiers for newdata. If this is present, it will be searched for in the newdata data frame. Each group of rows in newdata with the same subject id represents the covariate path through time of a single subject, and the result will contain one curve per subject. If it is not specified, then an estimated survival curve is returned for each row of newdata.
OOB	a logical specifying whether out-of-bag predictions are desired (only if newdata = NULL).
time.eval	a vector of time points, at which the estimated survival probabilities will be computed.
time.tau	an optional vector, with the <i>i</i> -th entry giving the upper time limit for the computed survival probabilities for the <i>i</i> -th data of interest (i.e., only computes survival probabilities at <code>time.eval[time.eval <= time.tau[i]]</code> for the <i>i</i> -th data of interest). If <code>OOB = TRUE</code> , the length of <code>time.tau</code> is equal to the length of data used to train the object; If <code>OOB = FALSE</code> , the length of <code>time.tau</code> is equal to the length of newdata, or equal to the length of data if newdata is not given. The default NULL is simply to set all entries of <code>time.tau</code> equal to the maximum value of <code>time.eval</code> , so that all estimated survival probabilities are computed at the same <code>time.eval</code> .

Value

A list containing:

`survival.id` subject identifiers.
`survival.obj` an object of class `Surv`.
`survival.probs` the estimated survival probabilities for each data of interest. It is a list if the length of the estimated values differs from one to another; otherwise, it is a matrix with the number of columns equal to the number of the data of interest, number of rows equal to the number of the time points at which the estimated survival probabilities are computed.
`survival.tau` the input value `time.tau`.
`survival.times` the input value `time.eval`.

See Also

[sbrier_ltrc](#) for evaluation of model fit

Examples

```
#### Example with data pbcsample
library(survival)
Formula <- Surv(Start, Stop, Event) ~ age + alk.phos + ast + chol + edema
## Fit an LTRC conditional inference forest on time-varying data
LTRCCFobj <- ltrccf(formula = Formula, data = pbcsample, id = ID,
                   mtry = 3, ntree = 50L)

## Construct an estimated survival estimate for the second subject
tpnt <- seq(0, max(pbcsample$Stop), length.out = 50)
newData <- pbcsample[pbcsample$ID == 2, ]
Pred <- predictProb(object = LTRCCFobj, newdata = newData, newdata.id = ID,
                   time.eval = tpnt)
## Since time.tau = NULL, Pred$survival.probs is in the matrix format, with dimensions:
dim(Pred$survival.probs) # length(time.eval) x nrow(newdata)
## Plot the estimated survival curve
plot(Pred$survival.times, Pred$survival.probs, type = "l", col = "red",
     xlab = "Time", ylab = "Survival probabilities")
```

print

Print Summary Output of a ltrccf object or a ltrcrsf object

Description

Print summary output after a LTRCCF or a LTRCRSF model is built. This is the default print method for objects in the class of `ltrccf` or `ltrcrsf`.

Usage

```
print(x)
```

Arguments

x an object of class `ltrccf` or `ltrcrsf`.

See Also

[ltrccf](#), [ltrcrsf](#)

Examples

```
library(survival)
Formula = Surv(Start, Stop, Event) ~ age + alk.phos + ast + chol + edema
# Built a LTRCCF forest on the time-varying data by specifying id, with mtry specified:
LTRCCFobj = ltrccf(formula = Formula, data = pbcsample, id = ID, mtry = 3, ntree = 50L)
print(LTRCCFobj)

# Built a LTRCCF forest on the time-invariant data, with resampling, with mtry specified:
LTRCCFobj = ltrccf(formula = Formula, data = pbcsample, samptype = "swr",
                   mtry = 3, ntree = 50L)
print(LTRCCFobj)
```

sbrier_ltrc

Model fit evaluation for LTRC forests.

Description

Compute the (integrated) Brier score to evaluate the model fit for (left-truncated) right-censored survival data with time-varying covariates, as well as left-truncated right-censored data with time-invariant covariates.

Usage

```
sbrier_ltrc(obj, id = NULL, pred, type = c("IBS", "BS"))
```

Arguments

obj an object of class [Surv](#), formed on left-truncated right-censored observations (which are pseudo-subject observations in the time-varying case).

id an optional vector as subject identifiers for obj.

pred a list. This should contain 1) either a matrix or a list of survival probabilities named `survival.probs`; 2) a sequence of time points `survival.times`; 3) a vector of upper time limits `survival.tau`. See the values returned by [predictProb](#).

type a character string denoting the type of scores returned. If type = "IBS", the integrated Brier score up to the last time point in pred\$surv.times that is not larger than the minimum value of pred\$surv.tau is returned. If type = "BS", the Brier score at every time point in pred\$surv.times up to the minimum value of pred\$surv.tau is returned. type = "IBS" is set by default.

Value

If type = "IBS", this returns the integrated Brier score.

If type = "BS", this returns BScore, the Brier scores and Time, the time points at which the scores are computed.

Examples

```
### Example with dataset pbcsample
library(survival)
Formula = Surv(Start, Stop, Event) ~ age + alk.phos + ast + chol + edema
## Fit an LTRC conditional inference forest on time-varying data
LTRCCFobj = ltrccf(formula = Formula, data = pbcsample, id = ID, mtry = 3, ntree = 50L)

# Time points
tpnt = seq(0, 6000, by = 100)
# Set different upper time limits for each of the subjects
tau = seq(4001, 6200, length.out = length(unique(pbcsample$ID)))
## Obtain estimation at time points tpnt
Predobj = predictProb(object = LTRCCFobj, time.eval = tpnt, time.tau = tau)

## Compute the integrated Brier score:
pbcobj = Surv(pbcsample$Start, pbcsample$Stop, pbcsample$Event)
IBS = sbrier_ltrc(obj = pbcobj, id = pbcsample$ID, pred = Predobj, type = "IBS")

## Compute the Brier score at each value of tpnt
BS = sbrier_ltrc(obj = pbcobj, id = pbcsample$ID, pred = Predobj, type = "BS")
## Plot the Brier scores
plot(BS$Time, BS$BScore, pch = 20, xlab = "Time", ylab = "Brier score", col = 2)
## As one can see, the Brier scores are returned at all tpnt up to 4000,
## this is because the algorithm set the last evaluation time point
## to be 4000 based on the value of time.eval and time.tau
## (max(tpnt[tpnt <= min(tau)]) == 4000).
```

tune.ltrccf

Tune mtry to the optimal value with respect to out-of-bag error for a LTRCCF model

Description

Starting with the default value of mtry, search for the optimal value (with respect to out-of-bag error estimate) of mtry for `ltrccf`.

Usage

```
tune.ltrccf(
  formula,
  data,
  id,
  mtryStart = NULL,
  stepFactor = 2,
  time.eval = NULL,
  time.tau = NULL,
  ntreeTry = 100L,
  bootstrap = c("by.sub", "by.root", "none", "by.user"),
  samptype = c("swor", "swr"),
  sampfrac = 0.632,
  samp = NULL,
  na.action = "na.omit",
  trace = TRUE,
  doBest = FALSE,
  plot = FALSE,
  applyfun = NULL,
  cores = NULL,
  control = partykit::ctree_control(teststat = "quad", testtype = "Univ", mincriterion
    = 0, saveinfo = FALSE, minsplit = max(ceiling(sqrt(nrow(data))), 20), minbucket =
    max(ceiling(sqrt(nrow(data))), 7), minprob = 0.01)
)
```

Arguments

formula	a formula object, with the response being a Surv object, with form <code>Surv(tleft,tright,event)</code> .
data	a data frame containing n rows of left-truncated right-censored observations.
id	variable name of subject identifiers. If this is present, it will be searched for in the data data frame. Each group of rows in data with the same subject id represents the covariate path through time of a single subject. If not specified, the algorithm then assumes data contains left-truncated and right-censored survival data with time-invariant covariates.
mtryStart	starting value of mtry; default is <code>sqrt(nvar)</code> .
stepFactor	at each iteration, mtry is inflated (or deflated) by this value. The default value is 2.
time.eval	a vector of time points, at which the estimated survival probabilities are evaluated.
time.tau	an optional vector, with the i -th entry giving the upper time limit for the computed survival probabilities for the i -th data (i.e., only computes survival probabilities at <code>time.eval[time.eval <= time.tau[i]]</code> for the i -th data of interest).
ntreeTry	number of trees used at the tuning step.
bootstrap	bootstrap protocol. (1) If id is present, the choices are: "by.sub" (by default) which bootstraps subjects, "by.root" which bootstraps pseudo-subjects. Both

can be with or without replacement (by default sampling is without replacement; see the option `perturb` below); (2) If `id` is not specified, it bootstraps the data by sampling with or without replacement. Regardless of the presence of `id`, if "none" is chosen, data is not bootstrapped at all, and is used in every individual tree. If "by.user" is chosen, the bootstrap specified by `samp` is used.

<code>samptype</code>	choices are <code>swor</code> (sampling without replacement) and <code>swr</code> (sampling with replacement). The default action here is sampling without replacement.
<code>sampfrac</code>	a fraction, determining the proportion of subjects to draw without replacement when <code>samptype = "swor"</code> . The default value is <code>0.632</code> . To be more specific, if <code>id</code> is present, $0.632 * N$ of subjects with their pseudo-subject observations are drawn without replacement (N denotes the number of subjects); otherwise, $0.632 * n$ is the requested size of the sample.
<code>samp</code>	Bootstrap specification when <code>bootstype = "by.user"</code> . Array of dim $n \times ntree$ specifying how many times each record appears in each bootstrap sample.
<code>na.action</code>	action taken if the data contains NA's. The default "na.omit" removes the entire record if any of its entries is NA (for x-variables this applies only to those specifically listed in <code>formula</code>). See function <code>cforest</code> for other available options.
<code>trace</code>	whether to print the progress of the search. <code>trace = TRUE</code> is set by default.
<code>doBest</code>	whether to run a <code>ltrccf</code> object using the optimal <code>mtry</code> found. <code>doBest = FALSE</code> is set by default.
<code>plot</code>	whether to plot the out-of-bag error as a function of <code>mtry</code> . <code>plot = FALSE</code> is set by default.
<code>applyfun</code>	an optional <code>lapply</code> -style function with arguments <code>function(X, FUN, ...)</code> . It is used for computing the variable selection criterion. The default is to use the basic <code>lapply</code> function unless the <code>cores</code> argument is specified (see below). See <code>ctree_control</code> .
<code>cores</code>	numeric. See <code>ctree_control</code> .
<code>control</code>	a list with control parameters, see <code>cforest</code> . The default values correspond to those of the default values used by <code>ltrccf</code> .

Value

If `doBest = FALSE` (default), this returns the optimal `mtry` value of those searched.

If `doBest = TRUE`, this returns the `ltrccf` object produced with the optimal `mtry`.

See Also

`sbrier_ltrc` for evaluation of model fit when searching for the optimal value of `mtry`.

Examples

```
### Example with data pbcsample
library(survival)
Formula = Surv(Start, Stop, Event) ~ age + alk.phos + ast + chol + edema
## mtry tuned by the OOB procedure with stepFactor 3, number of trees built 50.
mtryT = tune.ltrccf(formula = Formula, data = pbcsample, id = ID, stepFactor = 3,
```

```
ntreeTry = 50L, plot = TRUE)
```

tune.ltrcrsf	<i>Tune mtry to the optimal value with respect to out-of-bag error for a LTRCRSF model</i>
--------------	--

Description

Starting with the default value of `mtry`, search for the optimal value (with respect to out-of-bag error estimate) of `mtry` for `ltrcrsf`.

Usage

```
tune.ltrcrsf(
  formula,
  data,
  id,
  mtryStart = NULL,
  stepFactor = 2,
  time.eval = NULL,
  time.tau = NULL,
  ntreeTry = 100L,
  bootstrap = c("by.sub", "by.root", "by.node", "by.user", "none"),
  samptype = c("swor", "swr"),
  sampfrac = 0.632,
  samp = NULL,
  na.action = "na.omit",
  trace = TRUE,
  doBest = FALSE,
  plot = FALSE,
  ntime,
  nsplit = 10L,
  nodesizeTry = max(ceiling(sqrt(nrow(data))), 15),
  nodedepth = NULL
)
```

Arguments

<code>formula</code>	a formula object, with the response being a <code>Surv</code> object, with form <code>Surv(tleft,tright,event)</code> .
<code>data</code>	a data frame containing <code>n</code> rows of left-truncated right-censored observations.
<code>id</code>	variable name of subject identifiers. If this is present, it will be searched for in the data data frame. Each group of rows in <code>data</code> with the same subject <code>id</code> represents the covariate path through time of a single subject. If not specified, the algorithm then assumes <code>data</code> contains left-truncated and right-censored survival data with time-invariant covariates.

mtryStart	starting value of mtry; default is $\sqrt{\text{nvar}}$.
stepFactor	at each iteration, mtry is inflated (or deflated) by this value, used when mtry is not specified (see tune.ltrcrsf). The default value is 2.
time.eval	a vector of time points, at which the estimated survival probabilities are evaluated.
time.tau	an optional vector, with the i -th entry giving the upper time limit for the computed survival probabilities for the i -th data (i.e., only computes survival probabilities at $\text{time.eval}[\text{time.eval} \leq \text{time.tau}[i]]$ for the i -th data of interest).
ntreeTry	number of trees used at the tuning step.
bootstrap	bootstrap protocol. (1) If <code>id</code> is present, the choices are: "by.sub" (by default) which bootstraps subjects, "by.root" which bootstraps pseudo-subjects. Both can be with or without replacement (by default sampling is without replacement; see the option <code>samptype</code> below). (2) If <code>id</code> is not specified, the default is "by.root" which bootstraps the data by sampling with or without replacement; if "by.node" is chosen, data is bootstrapped with replacement at each node while growing the tree. Regardless of the presence of <code>id</code> , if "none" is chosen, the data is not bootstrapped at all. If "by.user" is chosen, the bootstrap specified by <code>samp</code> is used.
samptype	choices are <code>swor</code> (sampling without replacement) and <code>swr</code> (sampling with replacement). The default action here is sampling without replacement.
sampfrac	a fraction, determining the proportion of subjects to draw without replacement when <code>samptype = "swor"</code> . The default value is 0.632. To be more specific, if <code>id</code> is present, $0.632 * N$ of subjects with their pseudo-subject observations are drawn without replacement (N denotes the number of subjects); otherwise, $0.632 * n$ is the requested size of the sample.
samp	Bootstrap specification when <code>bootstype = "by.user"</code> . Array of dim $n \times \text{ntree}$ specifying how many times each record appears in each bootstrap sample.
na.action	action taken if the data contains NA's. The default "na.omit" removes the entire record if any of its entries is NA (for x-variables this applies only to those specifically listed in <code>formula</code>). See function rfsrc for other available options.
trace	whether to print the progress of the search. <code>trace = TRUE</code> is set by default.
doBest	whether to run a ltrcrsf object using the optimal mtry found. <code>doBest = FALSE</code> is set by default.
plot	whether to plot the out-of-bag error as a function of mtry. <code>plot = FALSE</code> is set by default.
ntime	an integer value used for survival to constrain ensemble calculations to a grid of <code>ntime</code> time points. Alternatively if a vector of values of length greater than one is supplied, it is assumed these are the time points to be used to constrain the calculations (note that the constrained time points used will be the observed event times closest to the user supplied time points). If no value is specified, the default action is to use all observed event times.
nsplit	an non-negative integer value for number of random splits to consider for each candidate splitting variable. This significantly increases speed. When zero or NULL, the algorithm uses much slower deterministic splitting where all possible splits are considered. <code>nsplit = 10L</code> by default.

nodesizeTry	forest average terminal node size used at the tuning step. It has been adjusted from the rfsrc default 15 for survival families.
nodedepth	maximum depth to which a tree should be grown. The default behaviour is that this parameter is ignored.

Value

If doBest = FALSE (default), this returns the optimal mtry value of those searched.

If doBest = TRUE, this returns the [ltrcrsf](#) object produced with the optimal mtry.

See Also

[sbrier_ltrc](#) for evaluation of model fit for the optimal value of mtry.

Examples

```
### Example with data pbcsample
library(survival)
Formula = Surv(Start, Stop, Event) ~ age + alk.phos + ast + chol + edema
## mtry tuned by the OOB procedure with stepFactor 3, number of trees built 50.
mtryT = tune.ltrcrsf(formula = Formula, data = pbcsample, id = ID, stepFactor = 3,
                    ntreeTry = 20L, plot = TRUE)
```

Index

cforest, [2–5](#), [14](#)
ctree_control, [4](#), [5](#), [14](#)

LTRCART, [2](#), [5](#)
ltrccf, [2](#), [3](#), [3](#), [9–12](#), [14](#)
LTRCforests-package, [2](#)
LTRCIT, [2](#), [3](#)
ltrcrsf, [2](#), [3](#), [5](#), [9–11](#), [15–17](#)

pbcsample, [8](#)
pbcseq, [8](#)
predictProb, [2](#), [3](#), [5](#), [7](#), [9](#), [11](#)
print, [2](#), [3](#), [10](#)

rfsrc, [2](#), [3](#), [6](#), [7](#), [16](#), [17](#)

sbrier_ltrc, [2](#), [3](#), [10](#), [11](#), [14](#), [17](#)
Surv, [4](#), [6](#), [10](#), [11](#), [13](#), [15](#)
survival, [8](#)

tune.ltrccf, [2–5](#), [12](#)
tune.ltrcrsf, [2](#), [3](#), [6](#), [7](#), [15](#), [16](#)