Package 'dtrSurv'

May 8, 2026

Type Package

Title Dynamic Treatment Regimes for Survival Analysis

Version 1.5

Date 2025-05-04

Author Shannon T. Holloway [aut, cre],
Hunyong Cho [aut]

Maintainer Shannon T. Holloway <shannon.t.holloway@gmail.com>

Description Provides methods for estimating multi-stage optimal
dynamic treatment regimes for survival outcomes with dependent censoring.
Cho, H., Holloway, S. T., and Kosorok, M. R. (2022) <doi:10.1093/biomet/asac047>.

License GPL-2

NeedsCompilation yes

Repository CRAN

Encoding UTF-8

Depends methods, stats, survival

RoxygenNote 7.1.2

Collate 'VerifyCriticalValue.R' 'VerifyData.R' 'VerifyERT.R'
'VerifyModels.R' 'VerifyRandomSplit.R' 'VerifyReplace.R'
'VerifySampleSize.R' 'VerifySplitRule.R' 'VerifySurvivalTime.R'
'VerifyTieMethod.R' 'VerifyTimePoints.R' 'VerifyTxName.R'
'VerifyUniformSplit.R' 'VerifyUsePrevTime.R'
'class_CriticalValue.R' 'class_CriticalValueMean.R'
'class_CriticalValueSurvival.R' 'class_TreeConditions.R'
'class_TreeType.R' 'criticalValue.R' 'class_TimeInfo.R'
'class_Parameters.R' 'predictSurvTree.R' 'class_SurvRF.R'
'survRF.R' 'shiftMat.R' 'class_Optimal.R' 'class_DTRSurvStep.R'
'class_DTRSurv.R' 'dtrSurv.R'

Date/Publication 2025-05-04 17:30:05 UTC

Contents

---

dtrSurv *Dynamic Treatment Regime for Survival Analysis*

Description

Provides methods for estimating multi-stage optimal dynamic treatment regimes for survival outcomes with dependent censoring.

Usage

```
dtrSurv(
  data,
  txName,
  models,
  ...,
  usePrevTime = TRUE,
  timePoints = "quad",
  nTimes = 100L,
  tau = NULL,
  criticalValue = "mean",
  evalTime = NULL,
  splitRule = NULL,
  ERT = TRUE,
  uniformSplit = NULL,
  sampleSize = NULL,
  replace = NULL,
  randomSplit = 0.2,
  tieMethod = "random",
  minEvent = 3L,
  nodeSize = 6L,
  nTree = 10L,
  mTry = NULL,
  pooled = FALSE,
  stratifiedSplit = NULL,
  stageLabel = "."
)
```

Arguments

data	A data.frame object. The full dataset including treatments received, all stage covariates, observed times, and censoring indicators. Can be provided as a matrix object if column headers are included. Can contain missing data coded as NA, but cannot contain NaN.
txName	A character vector object. The treatment variable name for each decision point. Each element corresponds to the respective decision point (element 1 = 1st decision; element 2 = 2nd decision, etc.).
models	A list object or a single formula. The models for each decision point. For list objects, each element corresponds to the respective decision point. Each element contains a formula defining the response as a Surv() object and the covariate structure of the model. Note that this model should not include any terms of order > 1. If using a single formula and the number of decision points is > 1, it is assumed that 'models' is a common formula to be used across all decision points. See details for further discussion.
...	Ignored. Present only to require named inputs.
usePrevTime	A logical object. If TRUE, previous times are included in the common formula model given in 'models'. This input is ignored if 'models' is not specified as a single common formula.
timePoints	A character object or a numeric vector object. If a character object, must be one of {"quad", "uni", "exp"} indicating the distribution from which the time points are to be calculated. For character input, input 'nTimes' must also be provided. If a numeric vector, the time points to be used. If 0 is not the first value, it will be concatenated by the software.
nTimes	An integer object. The total number of time points to be generated and considered. Used in conjunction with input 'timePoints' when 'timePoints' is a character; ignored otherwise.
tau	A numeric object or NULL. The study length. If NULL, the maximum timePoint is used.
criticalValue	A character object. Must be one of {"mean", "surv.prob", "surv.mean"}. The estimator for the value of a treatment rule. For "mean": the mean survival time; for "surv.prob": the mean survival probability at time 'evalTime'; for "surv.mean": first the mean survival probability is used, if ties exist across treatments, the mean survival time is used to identify the optimal.
evalTime	A numeric object or NULL. If numeric, the time at which the survival probability is to be estimated to determine the optimal treatment rule; 'criticalValue' must be one of {"surv.prob", "surv.mean"}. If NULL, 'criticalValue' must be {"mean"}.
splitRule	A character object OR NULL. Must be one of {"logrank", "mean"} indicating the test used to determine an optimal split. If NULL and 'criticalValue' = 'mean', takes value 'mean'. If NULL and 'criticalValue' = 'surv.prob' or 'surv.mean', takes value 'logrank'.
ERT	A logical object. If TRUE, the Extremely Randomized Trees algorithm is used to select the candidate variable.

`uniformSplit` A logical object. If 'ERT' and 'uniformSplit' are TRUE, the random cutoff is sampled from a uniform distribution over the range of available covariate values. If 'ERT' is TRUE and 'uniformSplit' is FALSE, a case is randomly selected and the cutoff is taken to be the mean cutoff between it and the next largest covariate value. If 'ERT' is FALSE, input is ignored.

`sampleSize` A numeric object, numeric vector object, or NULL. The fraction (0 < sampleSize <= 1) of the data to be used for each tree in the forest. If only one value is given, it is assumed to be the fraction for all decision points. If a vector is given, the length must be equal to the total number of decision points and each element corresponds to its respective decision point. If NULL and 'ERT' is TRUE, sampleSize defaults to 1.0. If NULL and 'ERT' is FALSE, sampleSize defaults to 0.632.

`replace` A logical object or NULL. If TRUE, the sample drawn for each of the nTree trees may have duplicate records. If FALSE, no individual is present in the sample for than once. If NULL, 'replace' = !'ERT'.

`randomSplit` A numeric object. The probability that a random split will occur. Must be 0 < randomSplit < 1.

`tieMethod` A character object. Must be one of {"first", "random"}. If multiple splits lead to the same value, the method by which the tie is broken.

`minEvent` An integer object. The minimum number of events that must be present in a node.

`nodeSize` An integer object. The minimum number of individuals that must be present in a node.

`nTree` An integer object. The number of trees to grow.

`mTry` An integer or integer vector object. The maximum number of covariates to sample for each split. If a vector, each element corresponds to its respective decision point.

`pooled` A logical object. If TRUE, data are pooled for the analysis. If FALSE, data is separated into groups based on treatment received and a tree is grown for each treatment group.

`stratifiedSplit` A numeric object. The stratified random split coefficient. Covariates for which the number of splits (s_i) is less than s*stratifiedSplit/d are explored preferentially total number of covariates under consideration).

`stageLabel` A character object. If using a common formula, the character used to separate the covariate from the decision point label. See details.

Details

If using a common formula for all decision points, i.e., 'models' is a single formula object, your data must follow a specific format. Specifically, if 'stageLabel' = ".", covariates must be named as xxx.1 for the first decision point, xxx.2 for the second, xxx.3 for the third, etc. The exact structure of the 'xxx' can be generally defined; however, it cannot contain the stageLabel. For example, if the column names are (Y.1, Y.2, d.1, d.2, A.1, A.2, X.1, X.2) 'models' = Surv(Y,d) ~ X + A would lead to Surv(Y.1,d.1) ~ X.1 + A.1 as the first stage model; and Surv(Y.2,d.2) ~ X.2 + A.2 as the

second stage. Further, baseline covariates can be used rather than stage dependent. In this case, the covariates should have no stageLabel. For example, if the column names are (Y.1, Y.2, d.1, d.2, A.1, A.2, X1, X2) where X1 and X2 are baseline 'models' = Surv(Y,d) ~ X1 + X2 + A would lead to Surv(Y.1,d.1) ~ X1 + X2 + A.1 as the first stage model; and Surv(Y.2,d.2) ~ X1 + X2 + A.2 as the second stage.

Y.k is the length of Stage k so that (Y.1 + Y.2 + ... + Y.K) is the overall observed failure time, d.k is the censoring status at Stage k, d.k = 0 if a subject was censored at Stage k, and 1 if he/she experienced failure during that stage or moved to Stage k+1. A.k is the treatment at Stage k, k=1,2,..., K. Note that every quantity here is stage-wide. In other words, Y.2 is the length of Stage 2 and is not cumulative from the baseline.

When one experienced censoring or failure at Stage k, it should be that Y.j = 0 for all j > k and instantaneous failure (Y.k < 1e-8) is not allowed; E.g., when d.(k-1) = 1 and Y.k = 0, the person is considered died at Stage k-1, but when d.(k-1) = 1 and Y.k = 2, the person made it to Stage k and either experienced failure or censoring (depending on d.k) during Stage k.

Any subject with missing values at Stage k will be ignored.

Value

An S4 object of class DTRSurv containing the key results and input parameters of the analysis. The information contained therein should be accessed through convenience functions stage(), show(), print(), and predict().

References

Cho, H., Holloway, S.T., and Kosorok, M.R. Multi-stage optimal dynamic treatment regimes for survival outcomes with dependent censoring. Submitted.

See Also

predict for retrieving the optimal treatment and/or the optimal survival curves. stage for retrieving stage results as a list. show for presenting the analysis results.

Examples

```
dt <- data.frame("Y.1" = sample(1:100,100,TRUE), "Y.2" = sample(1:100,100,TRUE),
                 "D.1" = rbinom(100, 1, 0.9), "D.2" = rbinom(100,1,0.9),
                 "A.1" = rbinom(100, 1, 0.5), "A.2" = rbinom(100,1,0.5),
                 "X.1" = rnorm(100), "X.2" = rnorm(100))

dtrSurv(data = dt,
        txName = c("A.1", "A.2"),
        models = list(Surv(Y.1,D.1)~X.1+A.1, Surv(Y.2,D.2)~X.2+A.2+Y.1))

# common formula
dtrSurv(data = dt,
        txName = c("A.1", "A.2"),
        models = Surv(Y,D)~X+A,
        usePrevTime = TRUE,
        stageLabel = ".")
```

```
# common formula and pooled analysis
dtrSurv(data = dt,
        txName = c("A.1", "A.2"),
        models = Surv(Y,D)~X+A,
        usePrevTime = TRUE,
        stageLabel = ".",
        pooled = TRUE)

dt <- data.frame("Y.1" = sample(1:100,100,TRUE), "Y.2" = sample(1:100,100,TRUE),
                 "D.1" = rbinom(100, 1, 0.9), "D.2" = rbinom(100,1,0.9),
                 "A.1" = rbinom(100, 1, 0.5), "A.2" = rbinom(100,1,0.5),
                 "X1" = rnorm(100), "X2" = rnorm(100))

# common formula with only baseline covariates
dtrSurv(data = dt,
        txName = c("A.1", "A.2"),
        models = Surv(Y,D)~X1+X2+A)

# common formula with only baseline covariates
# cutoff selected from indices
dtrSurv(data = dt,
        txName = c("A.1", "A.2"),
        models = Surv(Y,D)~X1+X2+A,
        ERT = TRUE, uniformSplit = FALSE)

# common formula with only baseline covariates
# not extremely random trees
dtrSurv(data = dt,
        txName = c("A.1", "A.2"),
        models = Surv(Y,D)~X1+X2+A,
        ERT = FALSE)

# common formula with only baseline covariates
# survival probability
dtrSurv(data = dt,
        txName = c("A.1", "A.2"),
        models = Surv(Y,D)~X1+X2+A,
        criticalValue = 'surv.prob')
```

`predict` *Prediction Method*

Description

Method to estimate the value for new data or to retrieve estimated value for training data

Usage

```
## S4 method for signature 'DTRSurv'
predict(object, ..., newdata, stage = 1, findOptimal = TRUE)
```

Arguments

object	A DTRSurv object. The object returned by a call to dtrSurv().
...	Ignored. Used to require named inputs.
newdata	NULL or a data.frame object. If NULL, this method retrieves the estimated value for the training data. If a data.frame, the value is estimated based on the data provided.
stage	An integer object. The stage for which predictions are desired.
findOptimal	A logical object. If TRUE, the value is estimated for all treatment options and that leading to the maximum value for each individual is used to estimate the value.

Value

a list object containing a matrix of the predicted survival function (survFunc), the estimated mean survuval (mean), and the estimated survival probability (if critical value is surv.mean or surv.prob)

Examples

```
dt <- data.frame("Y.1" = sample(1:100,100,TRUE), "Y.2" = sample(101:200,100,TRUE),
                 "D.1" = rbinom(100, 1, 0.9), "D.2" = rbinom(100,1,0.9),
                 "A.1" = rbinom(100, 1, 0.5), "A.2" = rbinom(100,1,0.5),
                 "X.1" = rnorm(100), "X.2" = rnorm(100))

result <- dtrSurv(data = dt,
                  txName = c("A.1", "A.2"),
                  models = list(Surv(Y.1,D.1)~X.1+A.1,
                                Surv(Y.2,D.2)~X.2+A.2+Y.1))

tt <- predict(object = result)
tt <- predict(object = result, stage = 1)
tt <- predict(object = result, findOptimal = FALSE)
tt <- predict(object = result, newdata = dt)
tt <- predict(object = result, newdata = dt, stage = 1)
tt <- predict(object = result, newdaata = dt, findOptimal = FALSE)
```

---

print *Print Analysis Results*

Description

Prints the key results of the analysis.

Usage

```
## S4 method for signature 'DTRSurv'
print(x, ...)
```

Arguments

x	A DTRSurv object. The value returned by dtrSurv().
...	Ignored.

Value

No return value, called to display key results.

Examples

```
dt <- data.frame("Y.1" = sample(1:100,100,TRUE), "Y.2" = sample(101:200,100,TRUE),
                 "D.1" = rbinom(100, 1, 0.9), "D.2" = rbinom(100,1,0.9),
                 "A.1" = rbinom(100, 1, 0.5), "A.2" = rbinom(100,1,0.5),
                 "X.1" = rnorm(100), "X.2" = rnorm(100))

result <- dtrSurv(data = dt,
                  txName = c("A.1", "A.2"),
                  models = list(Surv(Y.1,D.1)~X.1+A.1,
                                Surv(Y.2,D.2)~X.2+A.2+Y.1))

print(x = result)
```

---

show *Show Analysis Results*

Description

Shows the key results of the analysis.

Usage

```
## S4 method for signature 'DTRSurv'
show(object)
```

Arguments

object	A DTRSurv object. The value returned by dtrSurv().

Value

No return value, called to display key results.

Examples

```
dt <- data.frame("Y.1" = sample(1:100,100,TRUE), "Y.2" = sample(101:200,100,TRUE),
                 "D.1" = rbinom(100, 1, 0.9), "D.2" = rbinom(100,1,0.9),
                 "A.1" = rbinom(100, 1, 0.5), "A.2" = rbinom(100,1,0.5),
                 "X.1" = rnorm(100), "X.2" = rnorm(100))

result <- dtrSurv(data = dt,
                  txName = c("A.1", "A.2"),
                  models = list(Surv(Y.1,D.1)~X.1+A.1,
                                Surv(Y.2,D.2)~X.2+A.2+Y.1))

show(object = result)
```

---

stage *Retrieve Stage Results as a List*

Description

Returns the key results from all stages or one stage of the Q-learning algorithm.

Usage

```
stage(object, ...)

## S4 method for signature 'DTRSurv'
stage(object, ..., q)
```

Arguments

object	A DTRSurv object. The value returned by dtrSurv().
...	Ignored. Used to require named inputs.
q	An integer object. (optional) The stage for which results are desired. If q is not provided, all stage results will be returned.

Value

A list object containing the key results for each requested stage. If q is not provided, a list of these results will be returned, where the ith element of that list corresponds to the ith decision point.

Examples

```
dt <- data.frame("Y.1" = sample(1:100,100,TRUE), "Y.2" = sample(101:200,100,TRUE),
                 "D.1" = rbinom(100, 1, 0.9), "D.2" = rbinom(100,1,0.9),
                 "A.1" = rbinom(100, 1, 0.5), "A.2" = rbinom(100,1,0.5),
                 "X.1" = rnorm(100), "X.2" = rnorm(100))
```

```
result <- dtrSurv(data = dt,
                  txName = c("A.1", "A.2"),
                  models = list(Surv(Y.1,D.1)~X.1+A.1,
                                Surv(Y.2,D.2)~X.2+A.2+Y.1))

tt <- stage(object = result)
```

Index