Package 'TimeVTree'

October 12, 2022

Type Package

Title Survival Analysis of Time Varying Coefficients Using a
Tree-Based Approach

Version 0.3.1

Date 2018-03-12

Imports survival, grDevices, graphics, stats

Description Estimates time varying regression effects under Cox type models in
survival data using classification and regression tree. The codes in this package were
originally written in S-Plus for the paper ``Survival Analysis with Time-Varying Regression
Effects Using a Tree-Based Approach,'' by Xu, R. and Adak, S. (2002) <doi:10.1111/j.0006-
341X.2002.00305.x>, Biometrics, 58: 305-315.
Development of this package was supported by NIH grants AG053983 and AG057707,
and by the UCSD Altman Translational Research Institute, NIH grant UL1TR001442.
The content is solely the responsibility of the authors and does not necessarily
represent the official views of the NIH.
The example data are from the Honolulu Heart Program/Honolulu Asia Ag-
ing Study (HHP/HAAS).

License GPL-2

RoxygenNote 6.0.1

NeedsCompilation no

Author Sudeshna Adak [aut],
Ronghui Xu [aut],
Euyhyun Lee [trl, cre],
Steven Edland [ctb],
Lon White [ctb]

Maintainer Euyhyun Lee <e4lee@ucsd.edu>

Repository CRAN

Date/Publication 2018-03-12 18:01:06 UTC

R topics documented:

---

alcohol *Alcohol Consumption Data*

Description

This data set contains subjects' age at the time of death, and alcohol drinking habits. The data set includes 7990 subjects and 7610 events.

Usage

```
data('alcohol')
```

Format

- time: Subject's age at death (possibly right censored)

- event: Outcome indicator.
  1 = death
  0 = censored

- alc
  0 = no alcohol consumption
  1 = moderate alcohol consumption
  4 = excessive alcohol consumption

Source

Data is from the Honolulu Heart Program/Honolulu Asia Aging Study (HHP/HAAS). The HHP/HAAS was reviewed and approved by the Kuakini Hospital IRB, Kuakini Hospital, Honolulu, HI.

---

bootstrap	*Bootstrap to Correct for Over-optimism due to Adaptive Splitting*

Description

This function is used to obtain the bias-corrected cost. One may select the final subtree with the lowest bootstrap estimated cost, with or without the additional AIC/BIC as in Xu and Adak (2002).

Usage

```
bootstrap(B = 20, nodetree, subtrees, survtime, survstatus,
          x, D = 4, minfail = 30, alphac = 2)
```

Arguments

B	Number of bootstrap samples. Default is 20.
nodetree	Full grown tree with original data. Output from `output.coxphout`
subtrees	Pruned subtrees with original data. Output from `prune`
survtime	survival time/follow up time of subjects
survstatus	survival status of subjects.
x	a data frame of covariates. In case of single covariate, use [,,drop =FALSE] to keep the data frame structure
D	maximum depth the tree will grow. Default depth is 4.
minfail	minimum number of unique event required in each block. Default is 10
alphac	Predetermined penalty parameter

Details

The implemented cost here is the negative log partial likelihood. Each bootstrap sample is used to grow a full tree and then pruned to obtain the set of subtrees. The bias is estimated by the average of the differences between the cost of a bootstrapped subtree itself and the cost of sending the original data down the bootstrapped subtree. The bias-corrected cost is then obtained by subtracting this bias from the original cost. Predetermined penalty parameter can be used to account for the dimension of covariates, via Akaike information criteria (AIC), Schwarz Bayesian information criteria (BIC), or the 0.95 quantile of the chi-square distribution.

Value

bcoef	coefficient values from each bootstrap sample
btree	Tree related information from each bootstrapped sample. Types of information are the same as the ones from `output.coxphout`
bomega	Bias at each subtree for each bootstrapped data, the average of which gives the overall bootstrap estimated bias
bootcost	cost based on the bootstrapped data
ori.boot	negative log partial likelihood of the original data fitted to the model given by bootstrapped data

References

Xu, R. and Adak, S. (2002), Survival Analysis with Time-Varying Regression Effects Using a Tree-Based Approach. Biometrics, 58: 305-315.

Examples

```
## Not run:
data('alcohol')
require(survival)

coxtree <- coxph.tree(alcohol[,'time'], alcohol[,'event'],
                      x = alcohol[,'alc', drop = FALSE], D = 4)
nodetree <- output.coxphout(coxtree)

subtrees <- prune(nodetree)

#This function requires output from output.coxphout, prune, and the original data set.

store.mult.cont <- bootstrap(B=20, nodetree, subtrees, alcohol[,'time'],
                             alcohol[,'event'], x = alcohol[,'alc', drop = FALSE],
                             D=4,minfail=20, alphac=2)

## End(Not run)
```

---

coxph.tree   *Function to Grow the Tree Using the Score Statistic*

Description

This funtion finds the optimal cutpoints for the time-varying regression effects and grows the 'full tree' using the score statistic.

Usage

```
coxph.tree(survtime, survstatus, x, D = 3, method = "breslow", minfail = 10,
           iter.max = 20, eps = 1e-06, type = 'mod')
```

Arguments

survtime	survival time/ follow up time of subjects
survstatus	survival status of subjects. 0 for censored and 1 for event
x	a data frame of covariates. In case of single covariate, use [,,drop =F] to keep the data frame structure
D	maximum depth the tree will grow. Default depth is 3.
method	argument for coxph function. Default is 'breslow'. See coxph for more details.
minfail	minimum number of unique events required in each block. Default is 10

iter.max	the maximum number of iteration in coxph; default is 20. See [coxph](#) for more details.
eps	argument for coxph function; default is 0.000001. See [coxph](#) for more details.
type	method to calculate the score statistic. Two options are available: 'mod' for the modified score statistic and 'ori' for the original score statistic. Default value is 'mod.' Modified score statistic is used in the bootstrap part

Details

coxph.tree takes in survival time, survival status, and covariates to grow the full tree. It follows one of the stopping rules: 1) when the pre-specified depth is reached, or 2) the number of events in a node is less than a prespecified number, or 3) the maximized score statistic is less than a default value (0.0001).

Currently, data need to be arranged in descending order of time and with no missing.

Value

coxph.tree returns an object of class 'coxphtree.'

The function [output.coxphout](#) is used to obtain and print a summary of the result.

An object of class 'coxphtree' is a list containing the following components:

D	Depth value specified in the argument
coef	coefficient values of predictors. First number represents depth and second number represents block number
lkl	Likelihood ratio value of each node
breakpt	Starting point of each node. Starting point of node at Depth= 0 to maximum Depth = D+1 is shown.
ntree	Number of cases in each node
nevent	Number of events in each node
nblocks	Number of blocks in each depth
nodes	Indicator that indicates whether the block was eligible for further split
nodetree	A table with depth, block, node, left right, maximum score, start time, end time, # of cases, and # of events
scoretest	Maximum score at each block
xnames	Name of predictors
failtime	The time when events occurred without duplicates
summary	coxph output of each block
pvalue	p-value to test validity of a change point against none

References

Xu, R. and Adak, S. (2002), Survival Analysis with Time-Varying Regression Effects Using a Tree-Based Approach. Biometrics, 58: 305-315.

Examples

```
##Call in alcohol data set
data('alcohol')
require(survival)

coxtree <- coxph.tree(alcohol[,'time'], alcohol[,'event'],
                      x = alcohol[,'alc', drop = FALSE], D = 4)

nodetree <- output.coxphout(coxtree)

subtrees <- prune(nodetree)
```

---

elbow.tree   *Finding the Final Tree using the Elbow Method*

Description

elbow.tree is like final.tree, but instead of using the minimum cost it uses the 'elbow' of the costs. It is similar to the elbow AIC or BIC approaches in the literature.

Usage

```
elbow.tree(nodetree=nodetree, subtrees=subtrees, omega, alphac=2)
```

Arguments

nodetree	Fully grown tree from the original data. Output from [output.coxphout]
subtrees	Pruned subtrees from the original data. Output from [prune]
omega	Bias (i.e. third index of the output) from [bootstrap]. Look at the value section of [bootstrap] for more information.
alphac	Predetermined penalty parameter

Details

One can take the output (table) generated by this function and plot the (penalized) bias-corrected cost of each subtrees, then (visually) identify the 'elbow' as the selected subtree.

Value

subtree	output from prune with an additional column 'cost' that contains bootstrap estimate of each subtree
cost.p	This column contains the (penalized) bias-corrected cost of each subtree

Examples

```
## Not run:
data('alcohol')
require(survival)

coxtree <- coxph.tree(alcohol[,'time'], alcohol[,'event'],
                      x = alcohol[,'alc', drop = FALSE], D = 4)
nodetree <- output.coxphout(coxtree)

subtrees <- prune(nodetree)

store.mult.cont <- bootstrap(B=20, nodetree, subtrees, alcohol[,'time'],
                              alcohol[,'event'], x = alcohol[,'alc', drop = FALSE],
                              D=4,minfail=20, alphac=2)

Balph <- 0.5 * 2 * log(nrow(alcohol))
elbow.tree <- elbow.tree(nodetree, subtrees, store.mult.cont[[3]], alphac= Balph)

## End(Not run)
```

---

final.tree   *Finding the Final Tree After Bootstrap*

Description

final.tree uses bias-corrected costs obtained from bootstrap function and the predetermined penalty parameter to find the optimal tree from the set of subtrees.

Usage

```
final.tree(nodetree=nodetree, subtrees=subtrees, omega, alphac=2)
```

Arguments

nodetree	Fully grown tree from the original data. Output from output.coxphout
subtrees	Pruned subtrees from the original data. Output from prune
omega	Bias (i.e. third index of the output) from bootstrap. Look at the value section of bootstrap for more information.
alphac	Predetermined penalty parameter

Details

final.tree is part of the bootstrap function but can be used to try different penalty parameters without re-running bootstrap.

Value

subtree	output from prune with an additional column 'cost' that contains bootstrap estimate of each subtree
final	A tree with lowest cost value after applying predetermined penalty

References

Xu, R. and Adak, S. (2002), Survival Analysis with Time-Varying Regression Effects Using a Tree-Based Approach. Biometrics, 58: 305-315.

Examples

```
## Not run:
data('alcohol')
require(survival)

coxtree <- coxph.tree(alcohol[,'time'], alcohol[,'event'],
                      x = alcohol[,'alc', drop = FALSE], D = 4)
nodetree <- output.coxphout(coxtree)

subtrees <- prune(nodetree)

store.mult.cont <- bootstrap(B=20, nodetree, subtrees, alcohol[,'time'],
                             alcohol[,'event'], x = alcohol[,'alc', drop = FALSE],
                             D=4,minfail=20, alphac=2)

Balph <- 0.5 * 2 * log(nrow(alcohol))
final.tree <- final.tree(nodetree, subtrees, store.mult.cont[[3]], alphac= Balph)

## End(Not run)
```

---

mat.tvbeta	*Beta coefficient estimate at each time point*

Description

Function that ouputs beta coefficient estimate of each covariate at each observation time point for a given tree, which can be used to plot the time-varying coefficients.

Usage

```
mat.tvbeta(indx, fulltree, subtrees = NULL, survtime, survstatus, x)
```

Arguments

indx	Index number of a subtree that needs to be analyzed
fulltree	output of output.coxphout.
subtrees	(Optional) output of prune.

survtime	survival time/ follow up time of subjects
survstatus	survival status of subjects. 0 for alive and 1 for dead
x	a data frame of covariates. In case of single covariate, use [„drop =F] to keep the data frame structure

Value

For each predictor, `mat.tvbeta` gives the coefficient values at each observation time for a given subtree. The function outputs a matrix that can be used to plot the time-varying coefficient estimates over time. The number of rows in the matrix is the # of observations and the number of columns is the product of the # of covariates and the # of specified subtrees.

References

Xu, R. and Adak, S. (2002), Survival Analysis with Time-Varying Regression Effects Using a Tree-Based Approach. Biometrics, 58: 305-315.

Examples

```
#This function requires output from output.coxphout, prune, and the original data set.
data('alcohol')
require(survival)

coxtree <- coxph.tree(alcohol[,'time'], alcohol[,'event'],
                      x = alcohol[,'alc', drop = FALSE], D = 4)
nodetree <- output.coxphout(coxtree)

subtrees <- prune(nodetree)

#creating matrix of beta coefficients at each event time point for all subtrees
k <- nrow(subtrees)
for (l in 1:k) {
    print(paste("Tree #",l))
    coeftmp <- mat.tvbeta(l,nodetree,subtrees,alcohol[,'time'], alcohol[,'event'],
                  x = data.frame(model.matrix(~alc, data=alcohol)[,-c(1), drop = FALSE]))
    if (l == 1) coef <- coeftmp
    if (l > 1) coef <- cbind(coef,coeftmp)
  }


##Creating plot of all subtrees for each predictor:

p <- ncol(coef)/k #Number of variables
x = data.frame(model.matrix(~alc, data=alcohol)[,-c(1), drop = FALSE])
xnames <- xname(x)
xnames <- c('Alcohol 1', 'Alcohol 4')
#Subsetting data
coefnew <- data.frame(coef)
survtime <- alcohol[,'time']
#Setting desired depth (All the subtrees)
kk <- nrow(subtrees)
```

```
for (j in 1:p) {
  matplot(survtime,coefnew[,seq(from=j,to=kk*p,by=p)],type="l",lty=1:kk,col= (1:kk)+1
          ,xlab="Survival Time",ylab=" ")
  title(main=paste('all:', xnames[j]))
  legend('bottomleft', legend = paste('tree number', 1:kk), lty=1:kk,col= (1:kk)+1)
  }

##Creating a plot showing changes in coefficient of two predictors in full tree
#creating matrix of beta coefficients at each event time point for full tree
coeftmp <- mat.tvbeta(1,nodetree,subtrees,alcohol[,'time'], alcohol[,'event'],
                  x = data.frame(model.matrix(~alc, data=alcohol)[,-c(1), drop = FALSE]))
coefnew <- coeftmp
matplot(survtime,coefnew,type="l",lty=1:2,col= (1:2)+1,xlab="Survival Time",ylab=" ")
legend('bottomleft', legend = c("Alcohol 1", "Alcohol 4"), lty=1:2,col= (1:2)+1)
```

---

optimal.cutpoint	*Function to Find the First Cutpoint and its P Value*

Description

This function finds the first optimal cutpoint for the time-varying regression effects based on the maximized score statistics and calculates p-value based on a formula from Davies (1987) and O'Quigley and Pessione (1991). This is for depth 1 only.

Usage

```
optimal.cutpoint(survtime, survstatus, x, method = "breslow", acpf = 10,
            iter.max = 20, eps = 1e-06)
```

Arguments

survtime	survival time/ follow up time of subjects
survstatus	survival status of subjects. 0 for censored and 1 for an event
x	a data frame of covariates. In case of a single covariate, use [,,drop =F] to keep the data frame structure
method	argument for coxph function. Default is 'breslow'. See [coxph](#) for more details.
acpf	The search for the optimal cutpoint starts from the ((acpf/2)+1)th event until the (k - (acpf/2))th event, where k is the total number of events. Default is 10.
iter.max	the maximum number of iteration in coxph; default is 20. See [coxph](#) for more details.
eps	argument for coxph function; default is 0.000001. See [coxph](#) for more details.

Details

`optimal.cutpoint` takes in survival time, survival status, and covariates to find the first optimal cutpoint.

Currently, data need to be arranged in descending order of time and with no missing.

Value

optimal.cutpoint returns the following information:

breakpt	optimal cutpoint
scoretest	Maximum score associated with the optimal cut point
summary	3 output from coxph fitted with 1) entire data, 2) data before the optimal cutpoint, and 3) data after the optimal cutpoint.
pvalue	p-value to test the existance of a change point against none

References

Davies, R. (1987). Hypothesis Testing when a Nuisance Parameter is Present Only Under the Alternatives. Biometrika, 74(1), 33-43.

O'Quigley, J., and Pessione, F. (1991). The Problem of a Covariate-Time Qualitative Interaction in a Survival Study. Biometrics, 47(1), 101-115.

Examples

```
##Call in alcohol data set
data('alcohol')
require(survival)

coxtree <- optimal.cutpoint(alcohol[,'time'], alcohol[,'event'],
                    x = alcohol[,'alc', drop = FALSE])
```

---

output.coxphout	*Summary of coxph.tree output*

Description

This funtion organizes coxph.tree output into a format that can be used as an input for prune, plot_coxphtree, and mat.tvbeta.

Usage

```
output.coxphout(coxout)
```

Arguments

coxout	output from coxph.tree

Value

output.coxphout returns a table with following columns.

Depth	Depth value specified in the argument
Block	Time intervals present at each depth
Node	Unique number assigned to each block
Left	Node of a block that was divided into the left side in the next depth
Right	Node of a block that was divided into the right side in the next depth
Score	Modified score statistic of each node
lkl	Likelihood ratio value of each node
Start	Starting time of the node
End	Ending time of the node
# of Cases	Number of observations in each node
# of Events	Number of events in each node

References

Xu, R. and Adak, S. (2002), Survival Analysis with Time-Varying Regression Effects Using a Tree-Based Approach. Biometrics, 58: 305-315.

---

plot_coxphtree	*Plotting of Full Tree and Subtrees*

Description

This functin uses the full tree and subtrees (optional) to create visual outputs of the tree(s) and segments.

Usage

```
plot_coxphtree(fulltree, subtrees = NULL, mm = 3, start = 0, pdf = FALSE, file.name)
```

Arguments

fulltree	output of output.coxphout.
subtrees	(Optional) output of prune.
mm	Number of subtrees plot to be placed in one page. Default is 3
start	Sets starting point for segments. Useful if the minimum event time is far away from 0.
pdf	Do you want to export the plots in pdf format? Default is FALSE. When set as FALSE, all plots need to be cleared before running this function to avoid 'Plot rendering error.'
file.name	Name for the pdf file output.

Details

plot_coxphtree takes an output from output.coxphout and creates treeplot and barplot showing blocks at each depth. If an output from prune is also included in the argument, the function creates treeplot and barplot for each subtree. In the barplot, end nodes are in dark blue color.

References

Xu, R. and Adak, S. (2002), Survival Analysis with Time-Varying Regression Effects Using a Tree-Based Approach. Biometrics, 58: 305-315.

Examples

```
#This function requires output from output.coxphout and prune(optional)
data('alcohol')
require(survival)

coxtree <- coxph.tree(alcohol[,'time'], alcohol[,'event'],
                      x = alcohol[,'alc', drop = FALSE], D = 4)
nodetree <- output.coxphout(coxtree)

subtrees <- prune(nodetree)

plot_coxphtree(nodetree, subtrees, start = 70, pdf = FALSE)
```

---

prune | *Function to Prune Using the Score Statistic*

Description

This function merges over-segmented intervals to create optimally pruned subtrees.

Usage

```
prune(fulltree)
```

Arguments

fulltree | output from output.coxphout

Details

prune uses the CART algorithm and -log (partial likelihood) as cost to find the optimally pruned subtrees.

Value

prune returns a matrix with the following columns, where each row is an optimally pruned subtree:

K	subtrees number 1, 2, etc. Tree #1 is the full tree
N[1]	Number of terminal nodes
alpha	penalty parameter corresponding to the subtree
S[1]	-log(partial likelihood) of the subtree
pruneoff	Node that was removed from the previous larger subtree to obtain the current subtree

References

Xu, R. and Adak, S. (2002), Survival Analysis with Time-Varying Regression Effects Using a Tree-Based Approach. Biometrics, 58: 305-315.

Examples

```
##Call in alcohol data set
data('alcohol')
require(survival)

coxtree <- coxph.tree(alcohol[,'time'], alcohol[,'event'],
                      x = alcohol[,'alc', drop = FALSE], D = 4)
nodetree <- output.coxphout(coxtree)

subtrees <- prune(nodetree)
```

Index