Package ‘PermAlgo’

April 2, 2015

Title Permutational Algorithm to Simulate Survival Data

Version 1.1

Date 2015-03-31

Author Marie-Pierre Sylvestre, Thad Edens, Todd MacKenzie, Michal Abrahamowicz

Maintainer Marie-Pierre Sylvestre
<marie-pierre.sylvestre@umontreal.ca>

Description This version of the permutational algorithm generates a dataset in which event and censoring times are conditional on an user-specified list of covariates, some or all of which are time-dependent.

License GPL-2

LazyLoad yes

Repository CRAN

Date/Publication 2015-04-02 00:11:52

NeedsCompilation no

R topics documented:

PermAlgo-package 1
permalgorithm 3

Index **8**

`PermAlgo-package` *Generate Event Times Conditional On Time-Dependent Covariates*

Description

This version of the permutational algorithm generates a dataset in which event and censoring times are conditional on an user-specified list of covariates, some or all of which are time-dependent. Event times and censoring times also follow user-specified distributions.

Details

Package:	PermAlgo
Type:	Package
Version:	1.0
Date:	2010-08-24
License:	GPL-2
LazyLoad:	yes

The package contains one function avialable to the user, `permalgorithm`. The gist of the algorithm is to perform a one-to-one matching of `n` observed times with `n` independently generated vectors of covariates values. The matching is performed based on a permutation probability law derived from the partial likelihood of Cox's Proportional Hazards (PH) model.

Author(s)

Marie-Pierre Sylvestre, Thad Evans, Todd MacKenzie, Michal Abrahamowicz

Maintainer: Marie-Pierre Sylvestre <marie-pierre.sylvestre.chum@ssss.gouv.qc.ca>

References

This algorithm is an extension of the permutational algorithm first introduced by Abrahamowicz, MacKenzie and Esdaile, and described in details by MacKenzie and Abrahamowicz. The current version of the permutational algorithm is a flexible tool to generate event and censoring times that follow user-specified distributions and that are conditional on user-specified covariates. It has been validated through simulations in Sylvestre and Abrahamowicz. Please reference the manuscript by Sylvestre and Abrahamowicz cited below if the results of this program are used in any published material.

Sylvestre M.-P., Abrahamowicz M. (2008) Comparison of algorithms to generate event times conditional on time-dependent covariates. *Statistics in Medicine* **27(14)**:2618–34.

Abrahamowicz M., MacKenzie T., Esdaile J.M. (1996) Time-dependent hazard ratio: modelling and hypothesis testing with application in lupus nephritis. *JASA* **91**:1432–9.

MacKenzie T., Abrahamowicz M. (2002) Marginal and hazard ratio specific random data generation: Applications to semi-parametric bootstrapping. *Statistics and Computing* **12(3)**:245–252.

Examples

```
# Example - Generating adverse event conditional on use
# of prescription drugs

# Prepare the matrice of covariate (Xmat)
# Here we simulate daily exposures to 2 prescription drugs over a
# year. Drug prescriptions can start any day of follow-up, and their
# duration is a multiple of 7 days. There can be multiple prescriptions
# for each individuals over the year and interuptions of drug use in
# between.
```

```
# Additionaly, there is a time-independant binary covarite (sex).

n=500 # subjects
m=365 # days

# Generate the matrix of three covariate, in a 'long' format.
Xmat=matrix(ncol=3, nrow=n*m)

# time-independant binary covariate
Xmat[,1] <- rep(rbinom(n, 1, 0.3), each=m)

# Function to generate an individual time-dependent exposure history
# e.g. generate prescriptions of different durations and doses.
TDhist <- function(m){
  start <- round(runif(1,1,m),0) # individual start date
  duration <-  7 + 7*rpois(1,3) # in weeks
  dose <-  round(runif(1,0,10),1)
  vec <- c(rep(0, start-1), rep(dose, duration))
  while (length(vec)<=m){
    intermission <- 21 + 7*rpois(1,3) # in weeks
    duration <-  7 + 7*rpois(1,3) # in weeks
    dose <-  round(runif(1,0,10),1)
    vec <- append(vec, c(rep(0, intermission), rep(dose, duration)))}
  return(vec[1:m])}

# create TD var
Xmat[,2] <- do.call("c", lapply(1:n, function(i) TDhist(m)))
Xmat[,3] <- do.call("c", lapply(1:n, function(i) TDhist(m)))

# genereate vectors of event and censoring times prior to calling the
# function for the algorithm

eventRandom <- round(rexp(n, 0.012)+1,0)
censorRandom <- round(runif(n, 1,870),0)

# Generate the survival data conditional on the three covariates

data <- permalgorithm(n, m, Xmat, XmatNames=c("sex", "Drug1", "Drug2"),
eventRandom = eventRandom, censorRandom=censorRandom, betas=c(log(2),
log(1.04), log(0.99)), groupByD=FALSE )

# could use survival library and check whether the data was generated
# properly using coxph(Surv(Start, Stop, Event) ~ sex + Drug1 + Drug2,
# data)
```

---

permalgorithm *Generate Event Times Conditional On Time-Dependent Covariates*

Description

This version of the permutational algorithm generates a dataset in which event and censoring times are conditional on an user-specified list of covariates, some or all of which are time-dependent. Event times and censoring times also follow user-specified distributions.

Usage

```
permalgorithm(numSubjects, maxTime, Xmat, XmatNames = NULL,
eventRandom = NULL, censorRandom = NULL, betas, groupByD = FALSE)
```

Arguments

`numSubjects` is the number of subjects generated.

`maxTime` is a non-zero integer represening the maximum length of follow-up.

`Xmat` is the matrix of covariates values in a counting process format where every line represent one and only one time interval, during which all covariate values for a given subject remains constant. Consequently, `Xmat` should have `numSubjects*maxTime` rows. Each column of `Xmat` corresponds to a different covariates on which the event is conditionned. For fixed-in-time covariates, the same value should be replicated in each of `maxTime` row for a given subject.

`XmatNames` a an optional vector of character strings representing the names of each of the covariates in `Xmat`.

`eventRandom` represents individual event times. `eventRandom` can be a vector of nonegative integer values or a random generating function with argument n. In both cases, its values must be smaller or equal to `maxTime`. If left unspecified, then the algorithm generates event times based on an uniform distribtuion [1, maxTime].

`censorRandom` represents individual censoring times. `censorRandom` can be a vector of nonegative integer values or a random generating function with argument n. In both cases, its values must be smaller or equal to `maxTime`. The default is Uniform[1,maxTime].

`betas` is a vector of regression coefficients (log hazard) that represent the magnitude of the relationship between each of the covariates and the risk of an event. The length of `betas` should correspond to the number of columns in `Xmat`.

`groupByD` groupByD is an option that, when enabled, increases the computational efficiency of the algorithm by replacing the individual assignment of event times and censoring times by grouped assignements. The side effect of this option is that it generates datasets that are, on average, slightly less consistent with the model described by `betas` that those generated with the groupByD option set to FALSE. Still, groupByD=TRUE may be usefull to generate large datasets where `maxTime` is much smaller than `numSubjects` so that many ties are expected. Default is FALSE.

Details

The gist of the algorithm is to perform a one-to-one matching of `n` observed times with independently generated vectors of covariates values. The matching is performed based on a permutation probability law derived from the partial likelihood of Cox's Proportional Hazards (PH) model.

The number of events obtained in the data.frame returned by the function depends on both the distribution of event enventRandom and censoring times censorRandom. In the simplest case where the distirbution of eventRandom is Uniform over follow-up U[1,m], and the censoring is random, the number of observed events in the data.frame returnd by the algorithm is determined by the upper bound of the Uniform distribution of censorRandom. For example, setting the distribution of censorRandom to U[1,m] will lead to approximately half of the subjects to experience an event during follow-up, while setting the distribution of censorRandom to U[1,3/2] will lead to approximately two thirds of the observed times to be events.

Subjects without an event before or on maxTime and who are not censored before maxTime are censored on maxTime (administrative censoring).

*** Warning *** Currently the algorithm only takes Xmat in matrix format. Consequently, factor variables are not allowed. Instead, users need to code them with binary indicators.

Value

A data.frame object with columns corresponding to

Id	Identifies the rows of the data.frame that corresponds to each of the n individuals.
Event	Indicator of event. Event = 1 when event occurs and 0 otherwise.
Fup	Individual follow-up time.
Start	For counting process formulation. Represents the start of each time interval.
Stop	For counting process formulation. Represents the end of each time interval.
Xmat	The values of the covariates specified in Xmat.

Author(s)

Marie-Pierre Sylvestre, Thad Evans, Todd MacKenzie, Michal Abrahamowicz

References

This algorithm is an extension of the permutational algorithm first introduced by Abrahamowicz, MacKenzie and Esdaile, and described in details by MacKenzie and Abrahamowicz. The current version of the permutational algorithm is a flexible tool to generate event and censoring times that follow user-specified distributions and that are conditional on user-specified covariates. This is especially useful whenever at least one of the covariate is time-dependent so that conventional inversion methods are difficult to implement.

The algorithm has been validated through simulations in Sylvestre and Abrahamowicz. Please reference the manuscript by Sylvestre and Abrahamowicz, cited below, if this program is used in any published material.

Sylvestre M.-P., Abrahamowicz M. (2008) Comparison of algorithms to generate event times conditional on time-dependent covariates. *Statistics in Medicine* **27(14)**:2618–34.

Abrahamowicz M., MacKenzie T., Esdaile J.M. (1996) Time-dependent hazard ratio: modelling and hypothesis testing with application in lupus nephritis. *JASA* **91**:1432–9.

MacKenzie T., Abrahamowicz M. (2002) Marginal and hazard ratio specific random data generation: Applications to semi-parametric bootstrapping. *Statistics and Computing* **12(3)**:245–252.

Examples

```
# Example 1 - Generating adverse event conditional on use
# of prescription drugs

# Prepare the matrice of covariate (Xmat)
# Here we simulate daily exposures to 2 prescription drugs over a
# year. Drug prescriptions can start any day of follow-up, and their
# duration is a multiple of 7 days. There can be multiple prescriptions
# for each individuals over the year and interuptions of drug use in
# between.

# Additionaly, there is a time-independant binary covarite (sex).

n=500 # subjects
m=365 # days

# Generate the matrix of three covariate, in a 'long' format.
Xmat=matrix(ncol=3, nrow=n*m)

# time-independant binary covariate
Xmat[,1] <- rep(rbinom(n, 1, 0.3), each=m)

# Function to generate an individual time-dependent exposure history
# e.g. generate prescriptions of different durations and doses.
TDhist <- function(m){
  start <- round(runif(1,1,m),0) # individual start date
  duration <-  7 + 7*rpois(1,3) # in weeks
  dose <-  round(runif(1,0,10),1)
  vec <- c(rep(0, start-1), rep(dose, duration))
  while (length(vec)<=m){
    intermission <- 21 + 7*rpois(1,3) # in weeks
    duration <-  7 + 7*rpois(1,3) # in weeks
    dose <-  round(runif(1,0,10),1)
    vec <- append(vec, c(rep(0, intermission), rep(dose, duration)))}
  return(vec[1:m])}

# create TD var
Xmat[,2] <- do.call("c", lapply(1:n, function(i) TDhist(m)))
Xmat[,3] <- do.call("c", lapply(1:n, function(i) TDhist(m)))

# genereate vectors of event and censoring times prior to calling the
# function for the algorithm

eventRandom <- round(rexp(n, 0.012)+1,0)
censorRandom <- round(runif(n, 1,870),0)

# Generate the survival data conditional on the three covariates

data <- permalgorithm(n, m, Xmat, XmatNames=c("sex", "Drug1", "Drug2"),
eventRandom = eventRandom, censorRandom=censorRandom, betas=c(log(2),
log(1.04), log(0.99)), groupByD=FALSE )
```

```
# could use survival library and check whether the data was generated
# properly using coxph(Surv(Start, Stop, Event) ~ sex + Drug1 + Drug2,
# data)


# Example 2 - Generating Myocardial Infarction (MI) conditional on
# biennial measures of systolic blood pressure (like in the
# Framingham data).

m = 16 # exams
n <- 10000 # individuals

# Very crude way to generate the data,  meant as an example only!
sysBP <- rnorm(n*m,  120, 15)

# by not submitting event and censor time, one let the algorithm
# generate them from uniform distributions over the follow-up time.

data2 <- permalgorithm(n, m, sysBP, XmatNames="sysBP", betas=log(1.01),
groupByD=FALSE )
```

Index

∗Topic **survival**
PermAlgo-package, 1
permalgorithm, 3

PermAlgo (PermAlgo-package), 1
PermAlgo-package, 1
permalgorithm, 3