Package ‘missDeaths’

September 23, 2024

Date 2024-09-23

Title Simulating and Analyzing Time to Event Data in the Presence of Population Mortality

Version 2.8

Author Tomaz Stupnik [aut, cre], Maja Pohar Perme [ctb]

Maintainer Tomaz Stupnik <tomaz.stupnik@guest.arnes.si>

Description Implements two methods: a nonparametric risk adjustment and a data imputation method that use general population mortality tables to allow a correct analysis of time to disease recurrence. Also includes a powerful set of object oriented survival data simulation functions.

License GPL (>= 2)

Imports Rcpp (>= 0.11.1), mitools

LinkingTo Rcpp

Depends survival, rms, relsurv, cmprsk, MASS, methods

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-09-23 13:30:20 UTC

Contents

`md.binom` *md.binom*

Description

Creates information of a Bernoulli distributed covariate with the specified probability. This function call must be added to the `md.simparams` object.

Usage

```
md.binom(name, prob = 0.5)
```

Arguments

`name`	name of the covariate
`prob`	probability of success (having a value of 1)

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.binom("X", 0.25)

## End(Not run)
```

md.censor *Censoring simulated survival data*

Description

The md.censor function takes the data set simulated by the md.simulate and transforms it into a right censored sample by adding two columns to the original data set:
- time specifies the time to event or censoring, whichever comes first, specified in days and
- status specifies the censoring indicator and equals 0 if the individual is censored or <>0 in case of event.

Usage

```
md.censor(data, start, end, eventcolumns)
```

Arguments

data	data.frame containig event times and covariates
start	column name specifying start dates (left censoring)
end	column name specifying end dates (right censoring)
eventcolumns	vector of column names specifying a single event time or multiple event times (in case of competing risks)

Value

data.frame containing original data and columns time, maxtime and status

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
  md.sex("sex", 0.5) +
    md.uniform("birth", as.Date("1915-1-1"), as.Date("1930-1-1")) +
      md.uniform("start", as.Date("2000-1-1"), as.Date("2005-1-1")) +
        md.death("death", survexp.us, "sex", "birth", "start") +
          md.eval("age", "as.numeric(start - birth)/365.2425", 80, FALSE) +
            md.exp("event", "start", c("age", "sex"), c(0.1, 2), 0.05/365.2425)

data = md.simulate(sim, 1000)
complete = md.censor(data, "start", as.Date("2010-1-1"), c("event", "death"))

## End(Not run)
```

md.constant *md.constant*

Description

Creates information of a covariate that contains a fixed value (either numeric or date). This function call must be added to the `md.simparams` object.

Usage

```
md.constant(name, value)
```

Arguments

`name`	name of the covariate
`value`	value of the covariate

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.constant("start", as.Date("1990-1-1"))

## End(Not run)
```

md.D *Prepare compatible demographic information*

Description

Utility function that returns a data.frame containing basic demographic information compatible with the `md.survnp`, `md.survcox` and `md.impute` functions.

Usage

```
md.D(age, sex, year)
```

Arguments

`age`	vector of patient ages specified as number of days or number of years.
`sex`	vector containing 1 for males and 2 for females
`year`	vector of years of entry into the study can either be supplied as vector of start dates or as vector of years specified in number of days from origin (1-1-1970).

See Also

`md.survcox`, `md.survnp`

`md.data` *md.data*

Description

Creates information of covariates that are copies of covariates from an existing data set. This function call must be added to the `md.simparams` object.

Usage

```
md.data(data, randomsample = FALSE)
```

Arguments

`data`	data.frame
`randomsample`	controls whether the rows of the dataset are randomly sampled (with replaceent) or simply copied

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.data(data)

## End(Not run)
```

`md.death` *md.death*

Description

Creates information of a time of death variable distributed according to the specified population mortality table and demographic information. This function call must be added to the `md.simparams` object.

Usage

```
md.death(name, poptable, sexcol, birthcol, startcol)
```

Arguments

`name`	name of the column
`poptable`	population mortality table used to simulate times od death
`sexcol`	name of the column (covariate) specifying birth date
`birthcol`	name of the column (covariate) specifying birth date
`startcol`	name of the column (covariate) specifying the start date from which to calculate time of death

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.sex("sex", 0.5) +
     md.uniform("birth", as.Date("1930-1-1"), as.Date("1970-1-1")) +
       md.uniform("start", as.Date("2005-1-1"), as.Date("2010-1-1")) +
         md.death("death", survexp.us, "sex", "birth", "start")

## End(Not run)
```

`md.eval` *md.eval*

Description

Creates information of a covariate that is calculated (from other covariates) by evaluating a specified formula. This function call must be added to the `md.simparams` object.

Usage

```
md.eval(name, eval, center = Inf, invisible = FALSE)
```

Arguments

`name`	name of the covariate
`eval`	string specifying the formula to calculate the covariate
`center`	expected value of the calculated covariate
`invisible`	specifies whether the calculated covariate is included in the simulated dataset

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
    md.uniform("birth", as.Date("1915-1-1"), as.Date("1930-1-1")) +
      md.uniform("start", as.Date("2000-1-1"), as.Date("2005-1-1")) +
          md.eval("age", "as.numeric(start - birth)/365.2425", 80, FALSE)

## End(Not run)
```

`md.exp` *md.exp*

Description

Creates information of an event time variable distributed according to the specified exponential distribution. This function call must be added to the `md.simparams` object.

Usage

```
md.exp(name, startcol, covariates, betas, lambda)
```

Arguments

`name`	name of the column
`startcol`	column name specifying individual study start dates or a start date
`covariates`	names of covariate columns used
`betas`	betas for the corresponding covariate columns
`lambda`	baseline lambda

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
  md.uniform("X1", 0.5) +
    md.norm("X2") +
      md.exp("event", as.Date("1915-1-1"), c("X1", "X2"), c(0.1, 0.2), 0.05/365.2425)

## End(Not run)
```

`md.impute`	*Correctly impute missing information of possible deaths using population mortality*

Description

An iterative approach is used in this method to estimate the conditional distribution required to correctly impute the times of deaths using population mortality tables.

Note, that simply imputing expected survival times may seem intuitive, but does not give unbiased estimates, since the right censored individuals are not a random subsample of the patients.

Usage

```
md.impute(data, f, maxtime, D, ratetable, iterations = 4)
```

Arguments

`data`	a data.frame in which to interpret the variables named in the formula.
`f`	a formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`maxtime`	maximum potential observation time (number of days). where `status`=0 equals `time`. where `status`=1 equals potential time of right censoring if no event would be observed.
`D`	demographic information compatible with `md.survcox`, `md.impute` and `md.survnp`, see `md.D`.
`ratetable`	a population mortality table, default is `slopop`
`iterations`	the number of iteration steps to be performed, default is 4

Value

an array of times with imputed times of death that can be used instead of the unavailable complete data set to get unbiased estimates, ie. in `coxph`.

References

Stupnik T., Pohar Perme M. (2015) "Analysing disease recurrence with missing at risk information." Statistics in Medicine 35. p1130-43. `https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.6766`

See Also

`md.survcox`

Examples

```
library(missDeaths)
data(slopop)

data(observed)
observed$time = observed$time*365.2425
D = md.D(age=observed$age*365.2425, sex=observed$sex, year=(observed$year - 1970)*365.2425)
newtimes = md.impute(observed, Surv(time, status) ~ age + sex + iq + elevation,
  observed$maxtime*365.2425, D, slopop, iterations=4)

#Cumulative incidence function
cif = survfit(Surv(observed$time, observed$status)~1)
cif$surv = 1 - cif$surv
cif$upper = 1 - cif$upper
cif$lower = 1 - cif$lower
plot(cif)

#Net survival (NOTE: std error is slightly underestimated!)
surv.net = survfit(Surv(newtimes, observed$status)~1)
summary(surv.net, times=c(3,9)*365.2425)
plot(surv.net)

#Event free survival (NOTE: std error is slightly underestimated!)
surv.efs = survfit(Surv(newtimes, 1 * (observed$status | (newtimes != observed$time)))~1)
summary(surv.efs, times=c(3,9)*365.2425)
plot(surv.efs)
```

`md.mvnorm` *md.mvnorm*

Description

Creates information of a vector of multi-normal covariates with the specified array of means and covariance matrix. This function call must be added to the `md.simparams` object.

Usage

```
md.mvnorm(names, means = rep(0, length(names)), cov = diag(ncol(names)))
```

Arguments

`names`	vector of covariate names
`means`	vector of means, default is `rep(0, length(names))`
`cov`	covariance matrix, default is `diag(ncol(names))`

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.mvnorm(c("X1", "X2"), c(100, 0), matrix(c(225, 3, 2, 1), 2, 2))

## End(Not run)
```

`md.norm` *md.norm*

Description

Creates information of a normally distributed numeric covariate with the specified mean and standard deviation. This function call must be added to the `md.simparams` object.

Usage

```
md.norm(name, mean = 0, sd = 1)
```

Arguments

`name`	name of the covariate
`mean, sd`	mean and standard deviation

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.norm("X", 0, 1)

## End(Not run)
```

`md.sample` *md.sample*

Description

Creates information of a covariate that represents a random sample (with replacement) of the provided values. This function call must be added to the `md.simparams` object.

Usage

```
md.sample(name, array, weights=NULL)
```

Arguments

`name`	name of the covariate
`array`	vector of elements from which to choose from
`weights`	vector of probability weights for obtaining the elements of the vector being sampled or NULL if all values have equal probabilities

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.sample("X", c(0, 1, 2), c(0.2, 0.3, 0.5))

## End(Not run)
```

`md.sex` *md.sex*

Description

Creates information of a sex covariate that is a special case of Bernoulli covariate specifying 1 for male and 2 for female sex. This function call must be added to the `md.simparams` object.

Usage

```
md.sex(name, maleperc = 0.5, asfactor = FALSE)
```

Arguments

`name`	name of the covariate
`maleperc`	percentage of males (value = 1) versus females (value = 2)
`asfactor`	specifies whether the resulting sex covariate is factorized using as.factor or not

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.sex("sex", 0.5)

## End(Not run)
```

md.simparams *md.simparams*

Description

Constructs an md.simparams object that holds the parameters required to generate the simulated data set. The parameters specifying covariates and event time variables are appended to the md.simparams by adding the appropriate function call

Usage

```
md.simparams()
```

See Also

`md.constant`, `md.uniform`, `md.binom`, `md.norm`, `md.mvnorm`, `md.sex`, `md.exp`, `md.death`

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.sex("sex", 0.5) +
   md.uniform("birth", as.Date("1930-1-1"), as.Date("1970-1-1")) +
   md.uniform("start", as.Date("2005-1-1"), as.Date("2010-1-1")) +
   md.death("death", survexp.us, "sex", "birth", "start")

## End(Not run)
```

md.simulate *md.simulate*

Description

Creates a simulated dataset using the provided simulation parameters.

Usage

```
md.simulate(sim, N)
```

Arguments

`sim`	a `md.simparams` object containg simulation parameters
`N`	number of observations

Examples

```
## Not run:
library(missDeaths)
ratetable = survexp.us

sim = md.simparams() +
          md.sex("sex", 1) +
          md.uniform("Z1") +
          md.mvnorm(c("Z2", "Z3"), c(100, 0), matrix(c(225, 3, 2, 1), 2, 2)) +
          md.sample("Z4", c(1, 2, 3, 4), c(0.25, 0.25, 0.25, 0.25)) +
          md.uniform("birth", as.Date("1930-1-1"), as.Date("1970-1-1")) +
          md.constant("start", as.Date("1990-1-1")) +
          md.death("death", ratetable, "sex", "birth", "start") +
          md.eval("age", "as.numeric(start - birth)/365.2425", 80, FALSE) +
          md.exp("event", "start", c("age", "sex", "Z1", "Z2"),
             c(0.1, 2, 1, 0.01), 0.0001)

data = md.simulate(sim, 1000)

## End(Not run)
```

---

md.survcox *Fit a proportional hazards regression model over disease recurrence data with missing information of possible deaths*

Description

An iterative approach is used in this method to estimate the conditional distribution required to correctly impute the times of deaths using population mortality tables.

Note, that simply imputing expected survival times may seem intuitive, but does not give unbiased estimates, since the right censored individuals are not a random subsample of the patients.

Usage

```
md.survcox(data, f, maxtime, D, ratetable, iterations = 4, R = 50)
```

Arguments

`data`	a data.frame in which to interpret the variables named in the formula.
`f`	a formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the `Surv` function.
`maxtime`	maximum potential observation time (number of days). where `status`=0 equals `time`. where `status`=1 equals potential time of right censoring if no event would be observed.

`D`	demographic information compatible with `ratetable`, see `md.D`.
`ratetable`	a population mortality table, default is `slopop`
`iterations`	the number of iteration steps to be performed, default is 4
`R`	the number of multiple imputations performed to adjust the estimated variance of estimates, default is 50.

Value

if `R` equals 1 then an object of class `coxph.object` representing the fit.

if `R` > 1 then the result of the `MIcombine` of the coxph objects.

References

Stupnik T., Pohar Perme M. (2015) "Analysing disease recurrence with missing at risk information." Statistics in Medicine 35. p1130-43. https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.6766

See Also

`md.impute`, `MIcombine`

Examples

```
## Not run: 
library(missDeaths)
data(slopop)

data(observed)
observed$time = observed$time*365.2425
D = md.D(age=observed$age*365.2425, sex=observed$sex, year=(observed$year - 1970)*365.2425)

#fit a cox model (NOTE: estimated std error is slightly underestimated!)
md.survcox(observed, Surv(time, status) ~ age + sex + iq + elevation,
  observed$maxtime*365.2425, D, slopop, iterations=4, R=1)

#multiple imputations to correct the stimated std error
md.survcox(observed, Surv(time, status) ~ age + sex + iq + elevation,
  observed$maxtime*365.2425, D, slopop, iterations=4, R=50)

## End(Not run)
```

`md.survnp`	*Nonparametric analysis of disease recurrence with missing information of possible deaths*

Description

Estimates the Net and Event free survial using a is non-parametric approach that aims to correct all individuals using the unconditional survival time distribution obtained from the population mortality table.

The idea comes from realizing that the number of observed events in the data equals the number which would be observed in case of a complete data set, but the number of patients at risk does not. Hence, this method adjusts the observed number at risk to mimic the one we would get if the data was complete.

Usage

```
md.survnp(time, status, maxtime, D, ratetable, conf.int = 0.95)
```

Arguments

`time`	the time to event (number of days)
`status`	the status indicator, 0=right censored, 1=event at `time`
`maxtime`	maximum potential observation time (number of days). where `status`=0 equals `time`. where `status`=1 equals potential time of right censoring if no event would be observed.
`D`	demographic information compatible with `ratetable`, see `md.D`.
`ratetable`	a population mortality table, default is `slopop`
`conf.int`	desired coverage of the estimated confidence interval

Value

A list with components giving the estimates of net and event free survival.

`time`	times where the estimates are calculated (number of days)
`Y.net`	adjusted number of patients at risk at each time in a hypothetical world where patients don't die
`Y.efs`	adjusted number of patients at risk at each time
`surv.net`	the estimated Net survival
`std.err.net`	the estimated standard error of Net survival estimates
`surv.efs`	the estimated Event free survival
`std.err.efs`	the estimated standard error of Event free survival estimates

References

Stupnik T., Pohar Perme M. (2015) "Analysing disease recurrence with missing at risk information." Statistics in Medicine 35. p1130-43. https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.6766

Examples

```
## Not run:
library(missDeaths)
library(cmprsk)
data(slopop)

data(observed)
D = md.D(age=observed$age*365.2425, sex=observed$sex, year=(observed$year - 1970)*365.2425)
np = md.survnp(observed$time*365.2425, observed$status, observed$maxtime*365.2425, D, slopop)

#calculate net survival at 3 and 9 years
w = list(list(time=np$time, est=np$surv.net, var=(np$std.err.net)^2))
timepoints(w, times=c(3,9)*365.2425)

#plot the net and event free survival curves
plot(np$time, np$surv.net)
plot(np$time, np$surv.efs)

## End(Not run)
```

`md.uniform` *md.uniform*

Description

Creates information of a uniformly distributed numeric or date covariate with the specified lower and upper limits. This function call must be added to the `md.simparams` object.

Usage

```
md.uniform(name, min = 0, max = 1)
```

Arguments

`name`	name of the covariate
`min, max`	lower and upper limits of the distribution. Must be finite (either numeric or date)

Examples

```
## Not run:
library(missDeaths)

sim = md.simparams() +
   md.uniform("X", 0, 1)

## End(Not run)
```

`missDeaths` *Simulating and analyzing time to event data in the presence of population mortality*

Description

In analysis of time to event data we may have a situation where we know that a certain non-negligible competing risk exists, but is not recorded in the data. Due to competing nature of the risks, ignoring such a risk may significantly impact the at-risk group and thus lead to biased estimates.

This problem can be found in several national registries of benign diseases, medical device implantations (e.g. hip, knee or heart pacemaker) etc. where law obliges physicians to report events whereas the information on patient deaths is unavailable; it is hence unclear how many devices are still in use at a given time.

Under the assumption that the survival of an individual is not influenced by the event under study, general population mortality tables can be used to obtain unbiased estimates of the measures of interest or to verify the assumption that the bias introduced by the non-recorded deaths is truly negligible.

Two approaches are implemented in the missdeaths package:
- an iterative imputation method `md.survcox` and
- a mortality adjusted at risk function `md.survnp`.

The package also includes a comprehensive set of functions to simulate data that can be used for better understanding of these methods (See `md.simulate`).

Author(s)

Tomaz Stupnik <tomaz.stupnik@gmail.com> and Maja Pohar Perme

References

Stupnik T., Pohar Perme M. (2015) "Analysing disease recurrence with missing at risk information." Statistics in Medicine 35. p1130-43. https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.6766

Examples

```
## Not run:
library(missDeaths)
ratetable = survexp.us

sim = md.simparams() +
          md.sex("sex", 0.5) +
          md.binom("Z1", 0.5) +
          md.mvnorm(c("Z2", "Z3"), c(100, 0), matrix(c(225, 3, 2, 1), 2, 2)) +
          md.sample("Z4", c(1, 2, 3, 4), c(0.25, 0.25, 0.25, 0.25)) +
          md.uniform("birth", as.Date("1925-1-1"), as.Date("1950-1-1")) +
          md.uniform("start", as.Date("2000-1-1"), as.Date("2005-1-1")) +
          md.death("death", ratetable, "sex", "birth", "start") +
          md.eval("age", "as.numeric(start - birth)/365.2425", 80, FALSE) +
          md.exp("event", "start", c("age", "sex", "Z1", "Z2"),
             c(0.1, 2, 1, 0.01), 0.05/365.2425)
data = md.simulate(sim, 1000)

#construct a complete right censored data set
complete = md.censor(data, "start", as.Date("2010-1-1"), c("event", "death"))

#construct an observed right censored data set with non-reported deaths
observed = complete
observed$time[which(complete$status == 2)] = observed$maxtime[which(complete$status == 2)]
observed$status[which(complete$status == 2)] = 0

#estimate coefficients of the proportional hazards model
D = md.D(age=observed$age, sex=observed$sex, year=observed$start)
md.survcox(observed, Surv(time, status) ~ age + sex + Z1 + Z2,
          observed$maxtime, D, ratetable, iterations=4, R=50)

#estimate net- and event-free survival
np = md.survnp(observed$time, observed$status, observed$maxtime, D, ratetable)
w = list(list(time=np$time, est=np$surv.net, var=(np$std.err.net)^2))
timepoints(w, times=c(3,9)*365.2425)
plot(np$time/365.2425, np$surv.net)
plot(np$time/365.2425, np$surv.efs)

## End(Not run)
```

`observed` *A simulated dataset with non-recorded deaths*

Description

This data set is used to illustrate the missDeaths functions.

Format

A `data.frame` containing 10000 observations.

Index