Package ‘CFC’

May 25, 2019

Type Package
Title Cause-Specific Framework for Competing-Risk Analysis
Version 1.1.2
Date 2019-05-25
Author Mansour T.A. Sharabiani, Alireza S. Mahani
Maintainer Alireza S. Mahani <alireza.s.mahani@gmail.com>
Description Numerical integration of cause-specific survival curves to arrive at cause-
specific cumulative incidence functions, with three usage modes: 1) Convenient API for parametric survival regression followed by competing-risk analysis, 2) API for CFC, accepting user-specified survival functions in R, and 3) Same as 2, but accepting survival functions in C++.
License GPL (>= 2)
Imports Rcpp (>= 0.12.1), abind, survival, doParallel, foreach
LinkingTo Rcpp, RcppArmadillo, RcppProgress
Suggests BSGW, BayesMixSurv, Hmisc, randomForestSRC, RcppArmadillo
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-05-25 21:20:03 UTC

R topics documented:

bmt ......................................................... 2
cfc ......................................................... 2
cfc.pbasis .................................................. 5
cfc.prepdata ............................................. 7
cfc.survreg .............................................. 8
cfc.survreg.survprob .................................. 10
cfc.tbasis ............................................... 11
summary.cfc ........................................... 14
summary.cfc.pbasis ................................... 16
summary.cfc.survreg .................................. 17
summary.cfc.tbasis ................................... 18
The Bone Marrow Transplant Data

Description

Bone marrow transplant data with 408 rows and 5 columns.

Format

The data has 408 rows and 5 columns.

- **cause**: a numeric vector code. Survival status. 1: dead from treatment related causes, 2: relapse, 0: censored.
- **time**: a numeric vector. Survival time.
- **platelet**: a numeric vector code. Platelet 1: more than $10^9$ per L, 0: less.
- **tcell**: a numeric vector. T-cell depleted BMT 1:yes, 0:no.
- **age**: a numeric vector code. Age of patient, scaled and centered ((age-35)/15).

Source

Simulated data (taken from R package 'timereg')

---

cfc

Cause-specific competing-risk survival analysis

Description

Using adaptive generalized Newton-Cotes for calculating cumulative incidence functions.

Usage

cfc(f.list, args.list, n, tout, Nmax = 100L, rel.tol = 1e-05, ncores = 1)

Arguments

- **f.list**: In R mode, this is a list of survival functions, one per cause. Each survival function must have the prototype $f(t, \text{args}, n)$, where $t$ is a vector of time-from-index values, $\text{args}$ is a list of all arguments needed by the function, and $n$ is the iterator that allows the function to produce a different output for each observation and/or Bayesian sample. The output is a vector of survival probabilities at times $t$ for that particular observation/sample combination with iterator value $n$. In C++ mode, this is a list of lists, one per cause. Each list must contain pointers to three C++ functions, in order: 1) survival function of type func with prototype defined as `typedef arma::vec (*func)(arma::vec x, void* arg, int n)`
cfc (using RcppArmadillo's vec class) and a similar interpretation as its R counterpart, 2) initializer function of type initfunc, with prototype defined as typedef void* (*initfunc)(List) where List is the Rcpp wrapper class for R lists, and 3) resource de-allocator function of type freefunc with this prototype: typedef void (*freefunc)(void *arg).

See vignette for C++ example.

args.list List of arguments (each one a list), one per cause, to be supplied to the survival functions in f.list.

n Range of iterator (starting at 1) for survival functions. This can be the product of number of observations, and number of MCMC samplers in a Bayesian survival function.

tout Vector of time points for which cumulative incidence functions are requested.

Nmax Maximum number of subdivisions in the interval [0, max(tout)) to be created in the adaptive quadrature algorithm.

rel.tol Threshold for relative integration error, used as stoppage criterion. It is calculated as the maximum relative error at time point max(tout) across all causes. Each relative error number is the difference between the Simpson-based and trapezoidal-based numbers, divided by the Simpson-based number.

ncores Number of parallel threads to use. This is currently only implemented in C++ mode.

Value

An object of class cfc, which is a list with the following elements:

ci Array of dimensions (length(tout), length(f.list), n), cumulative incidence functions for all causes and all values of the iterator, evaluated at all time points indicated by tout.

s Array of same dimensions as ci, containing the (unadjusted) survival functions for all causes and all values of the iterator, evaluated at all time points indicated by tout.

is.maxiter Binary Array of length n, where 1 indicates that subdivision process for quadrature problem applied to survival functions at iteration n reached maximum set by Nmax before converging, and 0 otherwise.

n.maxiter Number of iterations that did not converge, i.e., sum(is.maxiter).

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

References


See Also

cfc.survreg

Examples

## Not run:

library("survival") # used for constructing survival formulas
library("BSGW") # used for Bayesian survival regression

data("bmt")
# splitting data into training and prediction sets
idx.train <- sample(1:nrow(bmt), size = 0.7 * nrow(bmt))
idx.pred <- setdiff(1:nrow(bmt), idx.train)
nobs.train <- length(idx.train)
nobs.pred <- length(idx.pred)

# prepare data and formula for Bayesian cause-specific survival regression
# using R package BSGW
out.prep <- cfc.prepdata(Surv(time, cause) ~ platelet + age + tcell, bmt)
f1 <- out.prep$formula.list[[1]]
f2 <- out.prep$formula.list[[2]]
dat <- out.prep$dat
tmax <- out.prep$tmax

# estimating cause-specific models
# set nsmp to larger number in real-world applications
nsmp <- 10
reg1 <- bsgw(f1, dat[idx.train, ], control = bsgw.control(iter = nsmp)
  , ordweib = T, print.level = 0)
reg2 <- bsgw(f2, dat[idx.train, ], control = bsgw.control(iter = nsmp)
  , ordweib = T, print.level = 0)

# defining survival function for this model
survfunc <- function(t, args) {
  nobs <- args$nobs; natt <- args$natt; nsmp <- args$nsmp
  alpha <- args$alpha; beta <- args$beta; X <- args$X
  idx.smp <- floor((n - 1) / nobs) + 1
  idx.obs <- n - (idx.smp - 1) * nobs
  return (exp(- t ^ alpha[idx.smp] *
            exp(sum(X[idx.obs, ] * beta[idx.smp, ]))));
}

# preparing function and argument lists
X.pred <- as.matrix(cbind(1, bmt[idx.pred, c("platelet", "age", "tcell")]))
arg.1 <- list(nobs = nobs.pred, natt = 4, nsmp = nsmp
  , alpha = exp(reg1$smp$betas), beta = reg1$smp$beta, X = X.pred)
arg.2 <- list(nobs = nobs.pred, natt = 4, nsmp = nsmp
  , alpha = exp(reg2$smp$betas), beta = reg2$smp$beta, X = X.pred)
arg.list <- list(arg.1, arg.2)
f.list <- list(survfunc, survfunc)
cfc.pbasis

Cause-specific competing-risk survival analysis in probability deno-
mination

Description

Constructing cumulative incidence and event-free probability functions from cause-specific survival times give for a fixed set of probabilities.

Usage

cfc.pbasis(t1, t2, probs, unity.tol = 1e-06, diff.tol = 0.01,
  diff.tol.policy = c("all", "mean"))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Multi-dimensional array containing survival times for cause 1 (i.e. exponential of the negative integral of hazard function). First dimension must correspond to probabilities at which times are calculated. Elements with same time, but distributed in the space of remaining dimensions, are treated independently. These dimensions can correspond, e.g., to observations or samples (in Bayesian frameworks). Survival times must be increasing along the first dimension.</td>
</tr>
<tr>
<td>t2</td>
<td>Multi-dimensional array containing survival times for cause 2. See note for t1.</td>
</tr>
<tr>
<td>probs</td>
<td>Probabilities for which survival times are provided in t1 and t2. Must begin at 1.0 and be decreasing.</td>
</tr>
<tr>
<td>unity.tol</td>
<td>Tolerance for difference of survival probability from 1.0 at time=0.0. In other words, we require that abs(probs[1] - 1.0) &lt; unity.tol.</td>
</tr>
<tr>
<td>diff.tol</td>
<td>Tolerance for change in survival probabilities from one time point to the next. Large changes lead to higher errors during numerical integration.</td>
</tr>
<tr>
<td>diff.tol.policy</td>
<td>If &quot;mean&quot;, then average change in survival probabilities are compared to diff.tol. If &quot;all&quot;, each values is compared. The latter is more strict.</td>
</tr>
</tbody>
</table>
Details

For each ‘row’ of t1, and t2, all elements are processed independently. To combine the survival curves from corresponding elements of t1 and t2, we first form a ‘comon denominator’ time vector by combining the two time vectors and sorting the results (after removing duplicates). We limit the maximum value in the combined time vector to minimum of the two maxima from each cause. Next, we use interpolation to find survival probabilities of each cause at all the time points in the combined time vector. Finally, we call the function cfc.tbasis.

Value

If t1 and t2 are one-dimensional, a matrix with columns named "time", "ci1", "ci2" and "efp" is returned. For multi-dimensional arrays, a list is returned with one such matrix for each element of the consolidated dimension representing all but the first dimension of t1 and t2.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

References


See Also

cfc.tbasis

Examples

```r
## Not run:

# prepare data for cause-specific competing-risk analysis
data(bmt)
bmt$status1 <- 1*(bmt$cause==1)
bmt$status2 <- 1*(bmt$cause==2)
f1 <- Surv(time, status1) ~ platelet + age + tcell
f2 <- Surv(time, status2) ~ platelet + age + tcell

# perform weibull regression on each cause independently
library(survival)
reg1 <- survreg(f1, bmt)
reg2 <- survreg(f2, bmt)

# predict times for given probabilities
# transpose predictions so that first dimension
# is time/probability (use first 50 observations for speed)
pvec <- seq(from=1.0, to = 0.1, length.out = 100)
```
cfc.prepdata

```r
cfc.prepdata
```

# Utility function for CFC data preparation

## Description

Preparing a data frame and formulas for cause-specific competing-risk survival analysis. It expands the multi-state status column into a series of binary columns by treating an event for a cause as censoring for all other causes.

## Usage

```r
cfc.prepdata(formul, dat)
```

## Arguments

- `formul`  
  Original survival formula.

- `dat`  
  Original data frame, with status column being an integer with values from 0 to K. The value 0 represents right-censoring, while 1 to K represent the K mutually-exclusive events.

## Details

The output data frame will have K new binary status columns. The K new status columns will be named "status_1", "status_2" through "status_<K>". Each of the output formulas in `formula_list` field will have the corresponding status. Column "status_1" will be 1 wherever status equals 1 in original data frame, and 0 elsewhere, and similarly for the remaining K-1 newly-added status columns.

## Value

A list with the following elements:

- `K`  
  Number of causes.
cfc.survreg

Expanded data frame.

A list of \( k \) formulas, each corresponding to one of the cause-specific survival models to be estimated. See details.

A formula with no left-hand side (time and status variables). This can be used for preparing the model matrix for prediction data sets, which can possibly have no response.

Maximum time to event/censoring extracted from original data frame. This can be used, e.g., during competing-risk analysis.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

References


Examples

data(bmt)
prep.out <- cfc.prepdata(Surv(time, cause) ~ platelet + age + tcell, bmt)

cfc.survreg(formula, data, newdata = NULL, dist = "weibull",
control = survreg.control(), tout, Nmax = 100L,
rel.tol = 1e-05)

Description

Convenient function to build cause-specific, parametric survival models using the \texttt{survival} package. This is followed by application of \texttt{cfc} function to produce cumulative incidence functions.

Usage

cfc.survreg(formula, data, newdata = NULL, dist = "weibull",
control = survreg.control(), tout, Nmax = 100L,
rel.tol = 1e-05)

Arguments

\begin{itemize}
\item \texttt{formula}\hspace{1cm} Survival formula with a multi-state status variable. See \texttt{cfc.prepdata}.
\item \texttt{data}\hspace{1cm} Data frame containing variables listed in \texttt{formula}.
\item \texttt{newdata}\hspace{1cm} Data frame of structure similar to \texttt{data}, perhaps without the time and status columns, to be used for generating cumulative incidence curves.
\end{itemize}
dist

One of survreg\_distributions. It can also be a vector, in which case elements 1 through K (number of causes) will be extracted and assigned to each cause-specific survival model. This allows for using different distributions for different causes.

control

List of survreg control parameters, according to survreg\_control.

tout

Time points, along which to produce the cumulative incidence curves.

Nmax

Maximum number of subdivisions to be used in the cfc quadrature algorithm.

rel\_tol

Threshold for relative error in cfc quadrature, used as a stoppage criterion. See cfc for details.

Value

A list with the following elements:

- \( K \) Number of causes.
- \( \text{formulas} \) List of formulas used in each of the \( K \) cause-specific survival regression models.
- \( \text{regs} \) List of all cause-specific regression objects returned by survreg, one per cause. The \( x \) field of each regression object has been substituted by the model matrix from newdata.
- \( \text{tout} \) Same as input.
- \( \text{cfc} \) An object of class cfc, the output of applying cfc to the parametric survival regression models constructed using survreg from \textbf{survival} package.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

References


See Also

cfc\_prepdata, cfc

Examples

data(bmt)
formul <- Surv(time, cause) ~ platelet + age + tcell
ret <- cfc\_survreg(formul, bmt[1:300, ], bmt[-(1:300), ]
, Nmax = 300, rel\_tol = 1e\-3)
cfc.survreg.survprob  

Survival probability function for survreg models

Description

Function for predicting survival probability as a function of time for survreg regression objects in survival package. It can be used to mix survreg models with other survival models in competing-risk analysis, using CFC package. This function is used inside cfc.survreg.

Usage

cfc.survreg.survprob(t, args, n)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>Time from index. Must be non-negative, but can be a vector.</td>
</tr>
<tr>
<td>args</td>
<td>Regression object that is returned by survreg. If using newdata for prediction, the x field of this object must be updated accordingly.</td>
</tr>
<tr>
<td>n</td>
<td>Observation index, must be between 1 and nrow(args$x).</td>
</tr>
</tbody>
</table>

Value

Vector of survival probabilities at time(s) t.

Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

References


See Also

cfc.survreg

Examples

```r
## Not run:
library("CFC")  # for cfc
data(bmt)
library("randomForestSRC")  # for rfsrc
library("survival")  # for survreg
prep <- cfc.prepdata(Surv(time, cause) ~ platelet + age + tcell, bmt)
f1 <- prep$formula.list[[1]]
```
Description

Constructing cumulative incidence and event-free probability functions from cause-specific survival probabilities evaluated at fixed time points.

Usage

cfc.tbasis(p1, p2, unity.tol = 1e-06, diff.tol = 0.01, diff.tol.policy = c("mean", "all"), check = TRUE)

Arguments

p1 Multi-dimensional array containing survival probabilities for cause 1 (i.e. exponential of the negative integral of hazard function). First dimension must correspond to time points at which probabilities are calculated. Elements with same
time, but distributed in the space of remaining dimensions, are treated independently. These dimensions can correspond, e.g., to observations or samples (in Bayesian frameworks).

- **p2**: Multi-dimensional array containing survival probabilities for cause 2. See note for \( p_1 \).
- **unity.tol**: Tolerance for difference of survival probabilities from 1.0 at time=0.0, which is the first 'row' of arrays \( p_1 \) and \( p_2 \). For example, for two-dimensional arrays, we need \( \text{all}(|p_1[1, \ldots, 1.0] - \text{unity.tol}|) \), and a similar condition for \( p_2 \).
- **diff.tol**: Tolerance for change in survival probabilities from one time point to the next. Large changes lead to higher errors during numerical integration.
- **diff.tol.policy**: If "mean", then average change in survival probabilities are compared to \( \text{diff.tol} \). If "all", each values is compared. The latter is more strict.
- **check**: Boolean flag indicating whether or not to check probability arrays for validity. Current validity checks are: 1) ensuring all probabilities are between 0.0 and 1.0, 2) all probabilities at time=0.0 are equal to 1.0 (see \( \text{unity.tol} \)), 3) No changes in probabilities from one time point to next are too large (see \( \text{diff.tol} \)). Dimensional consistency between \( p_1 \) and \( p_2 \) is always checked regardless of the value of this flag.

### Details

Assuming one-dimensional \( p_1 \) and \( p_2 \) for clarity, the algorithm calculates cumulative incidence function for cause 1 using a recursive formula: 

\[
ci_1[n+1] = ci_1[n] + dc_i_1[n], \text{where } dc_i_1[n] = 0.5*(p_2[n] + p_2[n+1])
\]

The increment in cumulative incidence function for cause 2 is similarly calculated, 

\[
dc_i_2[n] = 0.5*(p_1[n] + p_1[n+1])*(p_2[n] - p_2[n+1])
\]

These equations guarantee that 

\[
dc_i_1[n] + dc_i_2[n] = p_1[n]*p_2[n] - p_1[n+1]*p_2[n+1]
\]

Event-free probability is simply calculated as \( \text{efp}[n] = p_1[n]*p_2[n] \). Taken together, this numerical integration ensures that 

\[
\text{efp}[n+1] = \text{efp}[n] + dc_i_1[n] + dc_i_2[n] = 0.
\]

### Value

If \( p_1 \) and \( p_2 \) are one-dimensional arrays (i.e. vectors), a matrix with columns named "\( ci_1 \)", "\( ci_2 \)" and "\( \text{efp} \)" is returned, representing the cumulative incidence functions for cause 1 and cause 2 and the event-free probability, evaluated at same time points as \( p_1 \) and \( p_2 \) are provided. If \( p_1 \) and \( p_2 \) are multi-dimensional arrays, a list is returned with elements "\( ci_1 \)", "\( ci_2 \)" and "\( \text{efp} \)"; each one with the same interpretation, and all of the same dimensions as \( p_1 \) and \( p_2 \).

### Note

The integration algorithm described above does not require knowledge of time step. (Alternatively, using hazard functions for integration would have required specification of time step.) Since \( p_1 \) and \( p_2 \) are integrals (followed by exponentiation) of cause-specific hazard functions, using them directly adds to robustness of numerical integration and avoids error accumulation. The returned cumulative incidence and event-free probabilities correspond to the same time points assumed for input cause-specific probabilities.

### Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani
References


See Also
cfc.pbasis

Examples

```r
## Not run:
# prepare data for cause-specific competing-risk analysis
data(bmt)
bmt$status1 <- 1*(bmt$cause==1)
bmt$status2 <- 1*(bmt$cause==2)
f1 <- Surv(time, status1) ~ platelet + age + tcell
f2 <- Surv(time, status2) ~ platelet + age + tcell

# sample-based bayesian weibull regression
library(BSGW)
reg1 <- bsgw(f1, bmt, ordweib = TRUE, control = bsgw.control(iter = 500, burnin = 100, nskip = 50))
reg2 <- bsgw(f2, bmt, ordweib = TRUE, control = bsgw.control(iter = 500, burnin = 100, nskip = 50))

# prediction on a uniform grid of 100 time points
# (use first 50 observations for speed)
pred1 <- predict(reg1, newdata = bmt[1:50,], tvec = 100)
pred2 <- predict(reg2, newdata = bmt[1:50,], tvec = 100)

# permuting dimensions of survival objects to conform with cfc
S1 <- aperm(pred1$smp$s, c(2,1,3))
S2 <- aperm(pred2$smp$s, c(2,1,3))

# cause-specific competing risk analysis - time mode
my.cfc <- cfc.tbasis(S1, S2)

# calculating averages across observations (e.g. patients in the study)
my.summ <- summary(my.cfc, MARGIN = c(1,2))

# plotting mean CI and event-free functions
# as well as their sampled-based confidence intervals
plot(my.summ, t = pred1$tvec)
## End(Not run)
```
Summary and plotting output of `cfc`

**Description**

Summary method for class `cfc`.

**Usage**

```r
### S3 method for class 'cfc'
summary(object
  , f.reduce = function(x) x
  , pval = 0.05, ...)
### S3 method for class 'summary.cfc'
plot(x, which = c(1, 2), ...)
```

**Arguments**

- `object`  
  An object of class "cfc", usually the result of a call to `cfc`.
- `f.reduce`  
  Function to be applied to each sub-array of `object$ci` (cumulative incidence) and `object$s` (survival probability). Desired significance level for confidence intervals produced by `summary.cfc`. We essentially set the argument `probs` to `c(pval/2, 0.5, 1-pval/2)` when calling `quantile`.
- `pval`  
  An object of class "summary.cfc", usually the result of a call to `summary.cfc`.
- `which`  
  Vector of integers, indicating which plot(s) must be produced: 1) cumulative incidence functions, one per cause. For each cause, median and credible bands are plotted vs. time-from-index. 2) (unadjusted) survival functions, one per cause. Similar to (1), median and credible bands are plotted.
- `...`  
  Further arguments to be passed to `f.reduce` (for `summary.cfc`).

**Value**

Recall that the survival probability and cumulative incidence arrays returned by `cfc` are three-dimensional, and their first two dimensions indicate 1) time points and 2) causes. `f.reduce` is expected to produce an array of a fixed length, when applied to each sub-array, `ci[i, j, ]` and `s[i, j, ]`. The end-result is two three-dimensional array, where the first two dimensions are identical to its input arrays. This 3D array is then passed to the `quantile` function to compute median and credible bands. There is a special case where `f.reduce` returns a scalar, rather than an array, when applied to each sub-array. In this case, quantile calculation is meaningless and we return simply these point estimates. In summary, the return object from `summary` is a list with elements: 1) `ci` (cumulative incidence), 2) `s` (survival), and 3) `quantiles`, a boolean flag indicating whether the cumulative incidence and survival arrays returned are quantiles or point estimates.

**Author(s)**

Alireza S. Mahani, Mansour T.A. Sharabiani
References


See Also
cfc, summary.

Examples

```R
## Not run:
library("BSSGWI") # used for Bayesian survival regression

data(bmt)
# splitting data into training and prediction sets
idx.train <- sample(1:nrow(bmt), size = 0.7 * nrow(bmt))
idx.pred <- setdiff(1:nrow(bmt), idx.train)
nobs.train <- length(idx.train)
nobs.pred <- length(idx.pred)

# prepare data and formula for Bayesian cause-specific survival regression
# using R package BSSGW
out.prep <- cfc.prepdata(Surv(time, cause) ~ platelet + age + tcell, bmt)
f1 <- out.prep$formula.list[[1]]
f2 <- out.prep$formula.list[[2]]
dat <- out.prep$dat
tmax <- out.prep$tmax

# estimating cause-specific models
# set nsnmp to larger number in real-world applications
nsmp <- 10
reg1 <- bsgw(f1, dat[idx.train, ], control = bsgw.control(iter = nsmp)
          , ordweib = T, print.level = 0)
reg2 <- bsgw(f2, dat[idx.train, ], control = bsgw.control(iter = nsmp)
          , ordweib = T, print.level = 0)

# defining survival function for this model
survfunc <- function(t, args) {
  nobs <- args$nobs; natt <- args$natt; nsmp <- args$nsmp
  alpha <- args$alpha; beta <- args$beta; X <- args$X
  idx.snp <- floor((n - 1) / nobs) + 1
  idx.obs <- n - (idx.snp - 1) * nobs
  return (exp(- t ^ alpha[idx.snp] *
              exp(sum(X[idx.obs, ] * beta[idx.snp, ]))));
}

# preparing function and argument lists
X.pred <- as.matrix(cbind(1, bmt[idx.pred, c("platelet", "age", "tcell")]))
arg.1 <- list(nobs = nobs.pred, natt = 4, nsmp = nsmp
              , alpha = exp(reg1$snmp$betas), beta = reg1$snmp$beta, X = X.pred)
```
arg.2 <- list(nobs = nobs.pred, natt = 4, nsmp = nsmp
  alpha = exp(reg2$smp$betas), beta = reg2$smp$beta, X = X.pred)
arg.list <- list(arg.1, arg.2)
F.list <- list(survfunc, survfunc)

# cause-specific competing-risk
# set rel.tol to smaller number in real-world applications
out.cfc <- cfc(f.list, arg.list, nobs.pred * nsmp, tout, rel.tol = 1e-2)

# summarizing (and plotting) the results
# this function calculates the population-average CI and survival, one
# per each MCMC sample; therefore, the quantiles produced by the summary
# method, correspondingly, reflect our confidence in population-average values
my.f.reduce <- function(x, nobs, nsmp) {
  return (colMeans(array(x, dim = c(nobs, nsmp))))
}
my.summ <- summary(out.cfc, f.reduce = my.f.reduce, nobs = nobs.pred, nsmp = nsmp)

## End(Not run)

summary.cfc.pbasis  Summary of CFC objects

Description

summary method for class cfc.pbasis.

Usage

## S3 method for class 'cfc.pbasis'
summary(object, ...)
## S3 method for class 'summary.cfc.pbasis'
plot(x, ...)

Arguments

object An object of class "cfc.pbasis", usually the result of a call to cfc.pbasis.
x An object of class "summary.cfc.pbasis", usually the result of a call to summary.cfc.pbasis.
... Further arguments to be passed to/from other methods.

Value

The function summary.cfc.pbasis calculates the average of cumulative incidence and event-free
probability functions at each time point across all elements of the object list. If the object is a
matrix, it is returned without change.
Author(s)
Alireza S. Mahani, Mansour T.A. Sharabiani

References

See Also
The model fitting function is \texttt{cfc.pbasis}. See \texttt{summary} and \texttt{plot} for descriptions of the generic methods. See \texttt{cfc.tbasis} for time-denominated CFC, as well as usage examples.

\begin{verbatim}
summary.cfc.survreg  Summarizing and plotting output of cfc.survreg
\end{verbatim}

Description
summary and method for class \texttt{cfc.survreg}.

Usage
\begin{verbatim}
## S3 method for class 'cfc.survreg'
summary(object, obs.idx = "all", ...)
## S3 method for class 'summary.cfc.survreg'
plot(x, which = c(1, 2), ...)
\end{verbatim}

Arguments
\begin{itemize}
\item \texttt{object} An object of class "cfc.survreg", usually the result of a call to \texttt{cfc.survreg}.
\item \texttt{obs.idx} Index of observations to calculate mean cumulative incidence for; defaults to all observation.
\item \texttt{...} Further arguments to be passed to/from other methods.
\item \texttt{x} An object of class \texttt{summary.cfc.survreg}, usually the output of \texttt{summary.cfc.survreg}.
\item \texttt{which} Vector of integers, indicating which plot(s) must be produced: 1) cumulative incidence functions, one per cause, as a function of time-to-index, all in the same plot, 2) comparison of cumulative incidence function with/without competing-risk adjustment. The unadjusted figure is equivalent to 1 minus the Kaplan-Meyer (i.e., survival) function.
\end{itemize}

Value
\texttt{summary.cfc.surveeg} produces a matrix of dimensions \texttt{length(object$tout)} (number of time points) by \texttt{object$k} (number of causes). See description of which argument for \texttt{plot.summary.cfc.survreg}.
Author(s)

Mansour T.A. Sharabiani, Alireza S. Mahani

References


See Also

The model fitting function is \texttt{cfc.survreg}. See \texttt{summary} and \texttt{plot} for descriptions of the generic methods. For more flexible ways of cause-specific competing-risk analysis, see \texttt{cfc}.

---

**summary.cfc.tbasis**: Summarizing time-denominated CFC objects

### Description

summary method for class \texttt{cfc.tbasis}.

### Usage

```r
## S3 method for class 'cfc.tbasis'
summary(object,
    margin = if (class(object)[2] == "matrix") NULL else 1,
    ...)
## S3 method for class 'summary.cfc.tbasis'
plot(x, t = 1, ci = 0.95, ...)
```

### Arguments

- **object**: An object of class "cfc.tbasis", usually the result of a call to \texttt{cfc.tbasis}.
- **margin**: Dimensions of cumulative incidence and event-free probability arrays to keep while averaging the remaining dimensions. If the \texttt{cfc.tbasis} object is a matrix, no averaging is performed, and the function returns the object unchanged. Note that for list objects, \texttt{margin} must include 1 since it is meaningless to average out the time/probability dimension (which is always the first one).
- **x**: An object of class "summary.cfc.tbasis", usually the result of a call to \texttt{summary.cfc.tbasis}.
- **t**: Regular time step, or vector of time values, used for producing cumulative incidence and event-free probability plots.
- **ci**: Confidence interval used in cumulative incidence and event-free probability plots.
- **...**: Further arguments to be passed to/from other methods.
Value

The function `summary.cfc.tbasis` calculates the average of cumulative incidence and event-free probability functions as directed by `margin`. For example, if the element `ci1` of the object list is three-dimensional, then using `margin=c(1, 2)` causes the last dimension to be averaged out.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

References


See Also

The model fitting function is `cfc.tbasis`. See `summary` and `plot` for descriptions of the generic methods. See `cfc.pbasis` for probability-denominated CFC, as well as usage examples.
Index

*Topic datasets

bmt, 2

cfc, 2, 9, 14, 15, 18

cfc.pbasis, 5, 13, 16, 17, 19

cfc.prepdata, 7, 9

cfc.survreg, 4, 8, 10, 17, 18

cfc.survreg.survprob, 10

cfc.tbasis, 6, 11, 17–19

plot, 17–19

plot.summary.cfc(summary.cfc), 14
plot.summary.cfc.pbasis
  (summary.cfc.pbasis), 16
plot.summary.cfc.survreg
  (summary.cfc.survreg), 17
plot.summary.cfc.tbasis
  (summary.cfc.tbasis), 18

summary, 15, 17–19

summary.cfc, 14, 14
summary.cfc.pbasis, 16
summary.cfc.survreg, 17
summary.cfc.tbasis, 18