Package ‘prodlim’

March 6, 2017

Title  Product-Limit Estimation for Censored Event History Analysis
Version 1.6.1
Author Thomas A. Gerds
Description Fast and user friendly implementation of nonparametric estimators for censored event history (survival) analysis. Kaplan-Meier and Aalen-Johansen method.
Depends R (>= 2.9.0)
Imports Rcpp (>= 0.11.5), stats, graphics, survival, KernSmooth, lava
LinkingTo Rcpp
Maintainer Thomas A. Gerds <tag@biostat.ku.dk>
License GPL (>= 2)
RoxygenNote 5.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2017-03-06 13:53:09

R topics documented:
atRisk .................................................. 2
backGround ........................................... 3
confInt ................................................ 4
crModel .............................................. 5
dimColor ............................................. 6
EventHistory.frame ................................. 7
getEvent ............................................ 10
getStates .......................................... 11
Hist .................................................. 12
jackknife ......................................... 15
leaveOneOut .................................... 16
List2Matrix ..................................... 17
markTime ......................................... 18
meanNeighbors .................................. 19
atRisk

Drawing numbers of subjects at-risk of experiencing an event below Kaplan-Meier and Aalen-Johansen plots.

Description

This function is invoked and controlled by plot.prodlim.

Usage

\[
\text{atRisk}(x, \text{newdata, times, line, col, labelcol = NULL, interspace, cex, labels, title = "", titlecol = NULL, pos, adj, dist, adjust.labels = TRUE, ...})
\]

Arguments

- \( x \) an object of class `prodlim` as returned by the `prodlim` function.
- \( \text{newdata} \) see `plot.prodlim`
- \( \text{times} \) Where to compute the atrisk numbers.
- \( \text{line} \) Distance of the atrisk numbers from the inner plot.
- \( \text{col} \) The color of the text.
backGround

labelcol  The color for the labels. Defaults to col.
interspace Distance between rows of atrisk numbers.
cex      Passed on to mtext for both atrisk numbers and labels.
labels   Labels for the at-risk rows.
title    Title for the at-risk labels
titlecol The color for the title. Defaults to 1 (black).
pos     The value is passed on to the mtext argument at for the labels (not the atriks numbers).
adj     Passed on to mtext for the labels (not the atriks numbers).
dist    If line is missing, the distance of the upper most atrisk row from the inner plotting region: par$mgp[2].
adjust.labels  If TRUE the labels are left adjusted.
...       Further arguments that are passed to the function mtext.

Details

This function should not be called directly. The arguments can be specified as atRisk.arg in the call to plot.prodlim.

Value

Nil

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

See Also

plot.prodlim, confInt, markTime

Description

Some users like background colors, and it may be helpful to have grid lines to read off e.g. probabilities from a Kaplan-Meier graph. Both things can be controlled with this function. However, it mainly serves plot.prodlim.

Usage

backGround(xlim, ylim, bg = "white", fg = "gray77", horizontal = NULL, vertical = NULL, border = "black")
Arguments

- **xlim** Limits for the xaxis, defaults to `par("usr")[1:2].
- **ylim** Limits for the yaxis, defaults to `par("usr")[3:4].
- **bg** Background color. Can be multiple colors which are then switched at each horizontal line.
- **fg** Grid line color.
- **horizontal** Numerical values at which horizontal grid lines are plotted.
- **vertical** Numerical values at which vertical grid lines are plotted.
- **border** The color of the border around the background.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples

```r
plot(0,0)
backGround(bg="beige",fg="red",vertical=0, horizontal=0)

plot(0,0)
backGround(bg=c("yellow","green"),fg="red",xlim=c(-1,1),ylim=c(-1,1),horizontal=seq(0,1,.1))
backGround(bg=c("yellow","green"),fg="red",horizontal=seq(0,1,.1))
```

---

confInt Add point-wise confidence limits to the graphs of Kaplan-Meier and Aalen-Johansen estimates of survival and cumulative incidence.

Description

This function is invoked and controlled by `plot.prodlim`.

Usage

```r
confInt(x, times, newdata, type, citype, cause, col, lty, lwd, density = 55, ...)
```

Arguments

- **x** an object of class ‘prodlim’ as returned by the `prodlim` function.
- **times** where to compute point-wise confidence limits
- **newdata** see `plot.prodlim`
- **type** Either "cuminc" or "survival" passed to `summary.prodlim` as surv=ifelse(type="cuminc",FALSE,TRUE)

```r
confint
```
### Usage

```r
crModel()
```

### Details

Create a competing risks model with two causes to simulate a right censored event time data without covariates.

This function requires the `lava` package.

### crModel

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ctype</code></td>
<td>If &quot;shadow&quot; then confidence limits are drawn as colored shadows. Otherwise, dotted lines are used to show the upper and lower confidence limits.</td>
</tr>
<tr>
<td><code>cause</code></td>
<td>see <code>plot.prodlim</code></td>
</tr>
<tr>
<td><code>col</code></td>
<td>the colour of the lines.</td>
</tr>
<tr>
<td><code>lty</code></td>
<td>the line type of the lines.</td>
</tr>
<tr>
<td><code>lwd</code></td>
<td>the line thickness of the lines.</td>
</tr>
<tr>
<td><code>density</code></td>
<td>For <code>ctype=&quot;shadow&quot;</code>, the density of the shade. Default is 55 percent.</td>
</tr>
<tr>
<td><code>...</code></td>
<td>Further arguments that are passed to the function <code>segments</code> if <code>type=&quot;bars&quot;</code> and to <code>lines</code> else.</td>
</tr>
</tbody>
</table>

### Value

Nil

### Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

### See Also

`plot.prodlim`, `atRisk`, `markTime`
Value

A structural equation model initialized with four variables: the latent event times of two causes, the latent right censored time, and the observed right censored event time.

Author(s)

Thomas A. Gerds

Examples

library(lava)
m <- crModel()
d <- sim(m, 6)
print(d)

dimColor

Dim a given color to a specified density

Description

This function calls first col2rgb on a color name and then uses rgb to adjust the intensity of the result.

Usage

dimColor(col, density = 55)

Arguments

col Color name or number passed to col2rgb.
density Integer value passed as alpha coefficient to rgb between 0 and 255

Value

A character vector with the color code. See rgb for details.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

rgb col2rgb

Examples

dimColor(2, 33)
dimColor("green", 133)
EventHistory.frame

Description

Extract event history data and design matrix including specials from call

Usage

EventHistory.frame(formula, data, unspecialsDesign = TRUE, specials,
  specialsFactor = TRUE, specialsDesign = FALSE, stripSpecials = NULL,
  stripArguments = NULL, stripAlias = NULL, stripUnspecials = NULL,
  dropIntercept = TRUE, check.formula = TRUE, response = TRUE)

Arguments

  formula           Formula whose left hand side specifies the event history, i.e., either via Surv() or Hist().
  data              Data frame in which the formula is interpreted
  unspecialsDesign  Passed as is to model.design.
  specials          Character vector of special function names. Usually the body of the special functions is function(x)x but e.g., strata from the survival package does treat the values
  specialsFactor    Passed as is to model.design.
  specialsDesign    Passed as is to model.design
  stripSpecials     Passed as specials to strip.terms
  stripArguments    Passed as arguments to strip.terms
  stripAlias        Passed as alias.names to strip.terms
  stripUnspecials   Passed as unspecials to strip.terms
  dropIntercept     Passed as is to model.design
  check.formula     If TRUE check if formula is a Surv or Hist thing.
  response          If FALSE do not get response data (event.history).

Details

Obtain a list with the data used for event history regression analysis. This function cannot be used directly on the user level but inside a function to prepare data for survival analysis.

Value

A list which contains - the event.history (see Hist) - the design matrix (see model.design) - one entry for each special (see model.design)
Author(s)
Thomas A. Gerds <tag@biostat.ku.dk>

See Also
model.frame model.design Hist

Examples

## Here are some data with an event time and no competing risks
## and two covariates X1 and X2.
## Suppose we want to declare that variable X1 is treated differently
## than variable X2. For example, X1 could be a cluster variable, or
## X1 should have a proportional effect on the outcome.
dsurv <- data.frame(time=1:7,  
  status=c(0,1,1,0,0,0,1),  
  X2=c(2.24,3.22,9.59,4.4,3.54,6.81,5.05),  
  X3=c(1,1,1,1,0,0,1),  
  X4=c(44.69,37.41,68.54,38.85,35.9,27.02,41.84),  
  x1=factor(c("a","b","a","c","a","b"),  
    levels=c("c","a","b")))
## We pass a formula and the data
e <- EventHistory.frame(Hist(time,status)=prop(X1)+X2+cluster(X3)+X4,  
  data=dsurv,  
  specials=c("prop","cluster"),  
  stripSpecials=c("prop","cluster"))
names(e)
## The first element is the event.history which is result of the left hand  
## side of the formula:
e$event.history  
## same as
with(dsurv, Hist(time,status))
## to see the structure do
colnames(e$event.history)
unclass(e$event.history)
## in case of competing risks there will be an additional column called event,  
## see help(Hist) for more details
## The other elements are the design, i.e., model.matrix for the non-special covariates
e$design  
## and a data.frame for the special covariates
e$prop
## The special covariates can be returned as a model.matrix
e2 <- EventHistory.frame(Hist(time,status)=prop(X1)+X2+cluster(X3)+X4,  
  data=dsurv,  
  specials=c("prop","cluster"),  
  stripSpecials=c("prop","cluster"),  
  specialsDesign=TRUE)
e2$prop
## and the non-special covariates can be returned as a data.frame
e3 <- EventHistory.frame(Hist(time,status)=prop(X1)+X2+cluster(X3)+X4,
the general idea is that the function is used to parse the combination of formula and data inside another function. Here is an example with competing risks:

```
SampleRegression <- function(formula, data = parent.frame()){
  thecall <- match.call()
  ehf <- EventHistory.frame(formula = formula,
                            data = data,
                            stripSpecials = c("prop","cluster","timevar"),
                            specials = c("prop","timevar","cluster"))
  time <- ehf$event.history[,"time"]
  status <- ehf$event.history[,"status"]
  ## event as a factor
  if (attr(ehf$event.history,"model") == "competing.risks"){
    event <- ehf$event.history[,"event"]
    Event <- getEvent(ehf$event.history)
    list(response = data.frame(time, status, event, Event), X = ehf[-1])
  }
  else{ # no competing risks
    list(response = data.frame(time, status), X = ehf[-1])
  }
}
```

dsurv$outcome <- c("cause1","0","cause2","cause1","cause2","cause2","0")
SampleRegression(Hist(time, outcome ~ prop(X1)+X2+cluster(X3)+X4, dsurv))

```
## let's test if the parsing works
form1 <- Hist(time, outcome!="0") ~ prop(X1)+X2+cluster(X3)+X4
form2 <- Hist(time, outcome) ~ prop(X1)+cluster(X3)+X4
ff <- list(form1, form2)
lapply(ff, function(f)(SampleRegression(f, dsurv)))
```

```
## here is what the riskRegression package uses to distinguish between covariates with time-proportional effects and covariates with time-varying effects:
## Not run:
library(riskRegression)
data(Melanoma)
f <- Hist(time, status) ~ prop(thick)+strata(sex)+age+prop(ulcer, power=1)+timevar(invasion, test=1)
## here the unspecial terms, i.e., the term age is treated as prop
## also, strata is an alias for timvar
EHF <- prodlm:EventHistory.frame(formula,
                               Melanoma[1:10],
                               specials = c("timevar","strata","prop","const","tp"),
                               stripSpecials = c("timevar","prop"),
                               e3$design
```

getEvent

Extract a column from an event history object.

description

Extract a column from an event history object, as obtained with the function *Hist*.

Usage

getEvent(object, mode = "factor", column = "event")

Arguments

- **object**: Object of class "Hist".
- **mode**: Return mode. One of "numeric", "character", or "factor".
- **column**: Name of the column to extract from the object.

Details

Since objects of class "Hist" are also matrices, all columns are numeric or integer valued. To extract a correctly labeled version, the attribute states of the object is used to generate factor levels.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

See Also

*Hist*
getStates

Examples

dat <- data.frame(time=1:5, event=letters[1:5])
x <- with(dat, Hist(time, event))
## inside integer
unclass(x)
## extract event (the extra level "unknown" is for censored data)
getEvent(x)

getStates States of a multi-state model

Description

Extract the states of a multi-state model

Usage

getStates(object, ...)

Arguments

object Object of class prodlim or Hist.

Details

Applying this function to the fit of prodlim means to apply it to fit$model$response.

Value

A character vector with the states of the model.

Author(s)

Thomas A. Gerds
Hist

Create an event history response variable

Description

Functionality for managing censored event history response data. The function can be used as the left hand side of a formula: Hist serves `prodlim` in a similar way as `Surv` from the survival package serves ‘`survfit’`. Hist provides the suitable extensions for dealing with right censored and interval censored data from competing risks and other multi state models. Objects generated with Hist have a print and a plot method.

Usage

```
Hist(time, event, entry = NULL, id = NULL, cens.code = "0",
     addInitialState = FALSE)
```

Arguments

time for right censored data a numeric vector of event times – for interval censored data a list or a data.frame providing two numeric vectors the left and right endpoints of the intervals. See Details.

event A vector or a factor that specifies the events that occurred at the corresponding value of time. Numeric, character and logical values are recognized. It can also be a list or a data.frame for the longitudinal form of storing the data of a multi state model – see Details.

entry Vector of delayed entry times (left-truncation) or list of two times when the entry time is interval censored.

id Identifies the subjects to which multiple events belong for the longitudinal form of storing the data of a multi state model – see Details.

cens.code A character or numeric vector to identify the right censored observations in the values of event. Defaults to "0" which is equivalent to 0.

addInitialState If TRUE, an initial state is added to all ids for the longitudinal input form of a multi-state model.

Details

*Specification of the event times*

If `time` is a numeric vector then the values are interpreted as right censored event times, ie as the minimum of the event times and the censoring times.

If `time` is a list with two elements or data frame with two numeric columns The first element (column) is used as the left endpoints of interval censored observations and the second as the corresponding right endpoints. When the two endpoints are equal, then this observation is treated as an exact uncensored observation of the event time. If the value of the right interval endpoint is either `NA` or `Inf`, then this observation is treated as a right censored observation. Right censored observations
can also be specified by setting the value of event to cens.code. This latter specification of right censored event times overwrites the former: if event equals cens.code the observation is treated as right censored no matter what the value of the right interval endpoint is.

*Specification of the events*

If event is a numeric, character or logical vector then the order of the attribute "state" given to the value of Hist is determined by the order in which the values appear. If it is a factor then the order from the levels of the factor is used instead.

**Normal form of a multi state model**

If event is a list or a data.frame with exactly two elements, then these describe the transitions in a multi state model that occurred at the corresponding time as follows: The values of the first element are interpreted as the from states of the transition and values of the second as the corresponding to states.

**Longitudinal form of a multi state model**

If id is given then event must be a vector. In this case two subsequent values of event belonging to the same value of id are treated as the from and to states of the transitions.

Value

An object of class Hist for which there are print and plot methods. The object's internal is a matrix with some of the following columns:

| time | the right censored times |
| L    | the left endpoints of internal censored event times |
| R    | the right endpoints of internal censored event times |
| status | 0 for right censored, 1 for exact, and 2 for interval censored event times. |
| event | an integer valued numeric vector that codes the events. |
| from | an integer valued numeric vector that codes the from states of a transition in a multi state model. |
| to | an integer valued numeric vector that codes the to states of a transition in a multi state model. |

Further information is stored in attributes. The key to the official names given to the events and the from and to states is stored in an attribute "states".

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>, Arthur Allignol <arthur.allignol@fdm.uni-freiburg.de>

See Also

plot.Hist, summary.Hist, prodlim
Examples

```r
## Right censored responses of a two state model
## ---------------------------------------------

Hist(time=1:10,event=c(0,1,0,0,0,1,0,1,0,0))

## change the code for events and censored observations


TwoStateFrame <- SimSurv(10)
SurvHist <- with(TwoStateFrame,Hist(time,status))
summary(SurvHist)
plot(SurvHist)

## Right censored data from a competing risk model
## ---------------------------------------------

CompRiskFrame <- data.frame(time=1:10,event=c(1,2,0,3,0,1,2,1,2,1))
CRHist <- with(CompRiskFrame,Hist(time,event))
summary(CRHist)
plot(CRHist)

## Interval censored data from a survival model
## with icensFrame

icensFrame <- data.frame(L=c(1,1,3,4,6),R=c(2,NA,3,6,9),event=c(1,1,2,2))
with(icensFrame,Hist(time=list(L,R)))

## Interval censored data from a competing risk model
with(icensFrame,Hist(time=list(L,R),event))

## Multi state model

MultiStateFrame <- data.frame(time=1:10,
  from=c(1,1,3,1,2,4,1,1,2,1),
  to=c(2,3,1,2,4,2,3,2,4,4))
with(MultiStateFrame,Hist(time,event=list(from,to)))

## MultiState with right censored observations

MultiStateFrame1 <- data.frame(time=1:10,
  from=c(1,1,3,2,1,4,1,1,3,1),
  to=c(2,3,1,0,2,2,3,2,0,4))
with(MultiStateFrame1,Hist(time,event=list(from,to)))

## Using the longitudinal input method

MultiStateFrame2 <- data.frame(time=c(0,1,2,3,4,0,1,2,0,1),
  event=c(1,2,3,0,1,2,4,2,1,2),
  id=c(1,1,1,1,2,2,2,2,3,3))
with(MultiStateFrame2,Hist(time,event=event,id=id))
```
Jackknife

Compute jackknife pseudo values.

Description

Compute jackknife pseudo values.

Usage

jackknife(object, times, cause, keepResponse = FALSE, ...)

Arguments

- object: Object of class "prodlim".
- times: Time points at which to compute pseudo values.
- cause: For competing risks the cause of failure.
- keepResponse: If TRUE add the model response, i.e. event time, event status, etc. to the result.
- ...: not used

Details

Compute jackknife pseudo values based on marginal Kaplan-Meier estimate of survival, or based on marginal Aalen-Johansen estimate of cumulative incidence.

Note

The R-package pseudo does a similar job, and appears to be a little faster in small samples, but much slower in large samples. See examples.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

References


See Also

prodlim
Examples

```r
## pseudo-values for survival models
d = SimSurv(20)
f = prodlim(Hist(time, status) ~ 1, data = d)
jackknife(f, times = c(3, 5))

## in some situations it may be useful to attach the
time event history to the result
jackknife(f, times = c(3, 5), keepResponse = TRUE)

# pseudo-values for competing risk models
d = SimCompRisk(10)
f = prodlim(Hist(time, event) ~ 1, data = d)
jackknife(f, times = c(3, 10), cause = 1)
jackknife(f, times = c(3, 10, 17), cause = 2)
```

---

**leaveOneOut**

*Compute jackknife pseudo values.*

**Description**

Compute leave-one-out estimates

**Usage**

```r
leaveOneOut(object, times, cause, lag = FALSE, ...)
```

**Arguments**

- `object`: Object of class "prodlim".
- `times`: Time points at which to compute leave-one-out event/survival probabilities.
- `cause`: For competing risks the cause of interest.
- `lag`: For survival models only. If TRUE lag the result, i.e. compute $S(t-)$ instead of $S(t)$.
- `...`: Not used

**Details**

This function is the work-horse for jackknife

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>
List2Matrix

Reduce list to a matrix or data.frame with names as new columns

Description

This function is used by `summary.prodlim` to deal with results.

Usage

```r
List2Matrix(list, depth, names)
```

Arguments

- `list` A named list which contains nested lists
- `depth` The depth in the list hierarchy until an rbindable object
- `names` Names for the list variables

Details

Reduction is done with rbind.

Value

Matrix or data.frame.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

```r
x = list(a=data.frame(u=1,b=2,c=3), b=data.frame(u=3,b=4,c=6))
List2Matrix(x,depth=1, "X")
```
markTime

Marking product-limit plots at the censored times.

Description

This function is invoked and controlled by plot.prodlim.

Usage

markTime(x, times, nlost, pch, col, ...)

Arguments

x The values of the curves at times.
times The times where there curves are plotted.
nlost The number of subjects lost to follow-up (censored) at times.
pch The symbol used to mark the curves.
col The color of the symbols.
... Arguments passed to points.

Details

This function should not be called directly. The arguments can be specified as atRisk.arg in the call to plot.prodlim.

Value

Nil

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

See Also

plot.prodlim, confInt, atRisk
meanNeighbors

Helper function to obtain running means for prodlim objects.

Description

Compute average values of a variable according to neighborhoods.

Usage

meanNeighbors(x, y, ...)

Arguments

x Object of class "neighborhood".
y Vector of numeric values.
... Not used.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

See Also

neighborhood

Examples

meanNeighbors(x=1:10,y=c(1,10,100,1000,10001,10001,1002,1002,1002))

model.design

Extract a design matrix and specials from a model.frame

Description

Extract design matrix and data specials from a model.frame

Usage

model.design(terms, data, xlev = NULL, dropIntercept = FALSE,
             maxOrder = 1, unspecialsDesign = TRUE, specialsFactor = FALSE,
             specialsDesign = FALSE)
model.design

Arguments

terms          terms object as obtained either with function terms or strip.terms.
data           A data set in which terms are defined.
xlev           a named list of character vectors giving the full set of levels to be assumed for
               the factors. Can have less elements, in which case the other levels are learned
               from the data.
dropIntercept  If TRUE drop intercept term from the design matrix
maxOrder       An error is produced if special variables are involved in interaction terms of
               order higher than max.order.
unspecialsDesign A logical value: if TRUE apply model.matrix to unspecial covariates. If FALSE
               extract unspecial covariates from data.
specialsFactor A character vector containing special variables which should be coerced into a
               single factor. If TRUE all specials are treated in this way, if FALSE none of
               the specials is treated in this way.
specialsDesign A character vector containing special variables which should be transformed
               into a design matrix via model.matrix. If TRUE all specials are treated in this
               way.

Details

The function separates special terms from the unspecial terms and returns a list of design matrices,
one for unspecial terms and one for each special. Some special specials cannot or should not be
evaluated in data. E.g., y=a+dummy(x)+strata(v) the function strata can and should be evaluated,
but in order to have model.frame also evaluate dummy(x) one would be to define and export the
function dummy. Still the term dummy(x) can be used to identify a special treatment of the variable
x. To deal with this case, one can specify stripSpecials="dummy". In addition, the data should in-
clude variables strata(z) and x, not dummy(x). See examples. The function untangle.specials of
the survival function does a similar job.

Value

A list which contains - the design matrix with the levels of the variables stored in attribute 'levels'
- separate data.frames which contain the values of the special variables.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

EventHistory.frame model.frame terms model.matrix getXlevels
Examples

# specials that are evaluated. here ID needs to be defined
set.seed(8)
d <- data.frame(y=rnorm(5),x=factor(c("a","b","b","a","c")),z=c(2,2,7,7,7),v=sample(letters[1:5]))
d$z <- factor(d$z,levels=c(1:8))
ID <- function(x)
f <- formula(y~x+ID(z))
t <- terms(f,special="ID",data=d)
mda <- model.design(terms(t),data=d,specialsFactor=TRUE)
mda$ID
mda$design
##
mdb <- model.design(terms(t),data=d,specialsFactor=TRUE,unspecialsDesign=FALSE)
mdb$ID
mdb$design

# set x-levels
attr(mdb$ID,"levels")
attr(model.design(terms(t),data=d,xlev=list("ID(z)"=1:10),
specialsFactor=TRUE)$ID,"levels")

# special specials (avoid define function SP)
f <- formula(y~x+SP(z)+factor(v))
t <- terms(f,special="SP",data=d)
st <- strip.terms(t,specials="SP",arguments=NULL)
md2a <- model.design(st,data=d,specialsFactor=TRUE,specialsDesign="SP")
md2a$SP
md2b <- model.design(st,data=d,specialsFactor=TRUE,specialsDesign=FALSE)
md2b$SP

# special function with argument
f2 <- formula(y~x+treat(z,power=2)+treat(v,power=-1))
t2 <- terms(f2,special="treat")
list("treat")
model.design(st2,data=d,specialsFactor=FALSE)
model.design(st2,data=d,specialsFactor=TRUE)
model.design(st2,data=d,specialsDesign=TRUE)

library(survival)
data(pbc)
t3 <- terms(Surv(time,status!=0)-factor(edema)*age+strata(I(log(bili)>1))+strata(sex), special=c("strata","cluster"))
st3 <- strip.terms(t3,specials=c("strata"),arguments=NULL)
md3 <- model.design(terms=st3,data=pbc[1:4,])
md3$strata
md3$cluster

f4 <- Surv(time,status)-age+const(factor(edema))+strata(sex,test=0)+prop(bili,power=1)+tp(albumin)
t4 <- terms(f4,specials=c("prop","timevar","strata","tp","const"))
st4 <- strip.terms(t4,
specials=c("prop","timevar"), unspecials="prop")
Nearest neighborhoods for kernel smoothing

Description
Nearest neighborhoods for the values of a continuous predictor. The result is used for the conditional Kaplan-Meier estimator and other conditional product limit estimators.

Usage
neighborhood(x, bandwidth = NULL, kernel = "box")

Arguments
  x  Numeric vector – typically the observations of a continuous random variate.
  bandwidth  Controls the distance between neighbors in a neighborhood. It can be a decimal, i.e. the bandwidth, or the string "smooth", in which case \(N^{(-1/4)}\) is used, \(N\) being the sample size, or NULL in which case the \texttt{dpik} function of the package \texttt{KernSmooth} is used to find the optimal bandwidth.
  kernel  Only the rectangular kernel ("box") is implemented.

Value
An object of class 'neighborhood'. The value is a list that includes the unique values of 'x' (values) for which a neighborhood, consisting of the nearest neighbors, is defined by the first neighbor (first.nbh) of the usually very long vector neighbors and the size of the neighborhood (size.nbh).

Further values are the arguments bandwidth, kernel, the total sample size n and the number of unique values nu.

Author(s)
Thomas Gerds

References
parseSpecialNames

See Also

dpik, prodlim

Examples

d <- SimSurv(20)
neighborhood(d$X2)

Description

Extract from a vector of character strings the names of special functions and auxiliary arguments

Usage

parseSpecialNames(x, special, arguments)

Arguments

x Vector of character strings.
special A character string: the name of the special argument.
arguments A vector which contains the arguments of the special function

Details

Signals an error if an element has more arguments than specified by argument arguments.

Value

A named list of parsed arguments. The names of the list are the special variable names, the elements
are lists of arguments.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

model.design
Examples

```r
## ignore arguments
parseSpecialNames("treat(Z)", special="treat")
## set default to 0
parseSpecialNames(c("log(Z)","a","log(B)"), special="log", arguments=list("base"=0))
## set default to 0
parseSpecialNames(c("log(Z,3)","a","log(B,base=1)"), special="log", arguments=list("base"=0))
## different combinations of order and names
parseSpecialNames(c("log(Z,3)","a","log(B,1)"),
  special="log",
  arguments=list("base"=0))
parseSpecialNames(c("log(Z,1,3)","a","log(B,u=3)"),
  special="log",
  arguments=list("base"=0, "u"=1))
parseSpecialNames(c("log(Z,u=1,base=3)","a","log(B,u=3)"),
  special="log",
  arguments=list("base"=0, "u"=1))
parseSpecialNames(c("log(Z,u=1,base=3)","a","log(B,base=8,u=3)"),
  special="log",
  arguments=list("base"=0, "u"=1))
parseSpecialNames("treat(Z,u=2)",
  special="treat",
  arguments=list("u"=1, "k"=1))
parseSpecialNames(c("treat(Z,1,u=2)", "treat(B,u=2,k=3)"),
  special="treat",
  arguments=list("u"="NA", "k"=NULL))
## does not work to set default to NULL:
parseSpecialNames(c("treat(Z,1,u=2)", "treat(B,u=2)"),
  special="treat",
  arguments=list("u"="NA", "k"=NULL))
```

---

**PercentAxis**

*Percentage-labeled axis.*

**Description**

Use percentages instead of decimals to label the an axis with a probability scale.

**Usage**

`PercentAxis(x, at, ...)`

**Arguments**

- `x` Side of the axis
- `at` Positions (decimals) at which to label the axis.
- `...` Given to axis.
plot.Hist

Author(s)
Thomas Alexander Gerds

See Also
plot.prodlim

Examples

```r
plot(0,0,xlim=c(0,1),ylim=c(0,1),axes=FALSE)
PercentAxis(1,at=seq(0,1,.25))
PercentAxis(2,at=seq(0,1,.25))
```

plot.Hist  Box-arrow diagrams for multi-state models.

Description
Automated plotting of the states and transitions that characterize a multi states model.

Usage
```r
## S3 method for class 'Hist'
plot(x, nrow, ncol, stateLabels, arrowLabels,
    arrowLabelStyle = "symbolic", arrowLabelSymbol = "lambda",
    changeArrowLabelSide, tagBoxes = FALSE, startCountZero = TRUE, oneFitsAll,
    margin, cex, verbose = FALSE, ...)
```

Arguments
- `x` An object of class `Hist`.
- `nrow` the number of graphic rows
- `ncol` the number of graphic columns
- `stateLabels` Vector of names to appear in the boxes (states). Defaults to `attr(x,"state.names")`. The boxes can also be individually labeled by smart arguments of the form `box3.label="diseased",` see examples.
- `arrowLabels` Vector of labels to appear in the boxes (states). One for each arrow. The arrows can also be individually labeled by smart arguments of the form `arrow1.label=paste(expression(eta),...)`, see examples.
- `arrowLabelStyle` Either "symbolic" for automated symbolic arrow labels, or "count" for arrow labels that reflect the number of transitions in the data.
- `arrowLabelSymbol` Symbol for automated symbolic arrow labels. Defaults to "lambda".
changeArrowLabelSide
    A vector of mode logical (TRUE, FALSE) one for each arrow to change the side
    of the arrow on which the label is placed.

tagBoxes
    Logical. If TRUE the boxes are numbered in the upper left corner. The size can
    be controlled with smart argument boxtags.cex. The default is boxtags.cex=1.28.

startCountZero
    Control states numbers for symbolic arrow labels and box tags.

oneFitsAll
    If FALSE then boxes have individual size, depending on the size of the label,
    otherwise all boxes have the same size dependent on the largest label.

margin
    Set the figure margin via par(mar=margin). Less than 4 values are repeated.

cex
    Initial cex value for the state and the arrow
    labels

verbose
    If TRUE echo various things.

... Smart control of arguments for the subroutines text (box label), rect (box), ar-
    rows, text (arrow label). Thus the three dots can be used to draw individual
    boxes with individual labels, arrows and arrow labels. E.g. arrow2.label="any
    label" changes the label of the second arrow. See examples.

Note
    Use the functionality of the unix program ‘dot’ http://www.graphviz.org/About.php via R package
    Rgraphviz to obtain more complex graphs.

Author(s)
    Thomas A Gerds <tag@biostat.ku.dk>

See Also
    HistSmartControl

Examples

## A simple survival model

```r
SurvFrame <- data.frame(time=1:10,status=c(0,1,1,0,1,0,1,0,1,0))
SurvHist <- with(SurvFrame, Hist(time,status))
plot(SurvHist)
plot(SurvHist, box2.col=2, box2.label="experienced\nR user")
plot(SurvHist,
    box2.col=2,
    box1.label="newby",
    box2.label="experienced\nR user",
    oneFitsAll=FALSE,
    arrow1.length=.5,
    arrow1.label="",
    arrow1.lwd=4)
```

## change the cex of all box labels:
plot(SurvHist, 
   box2.col=2, 
   box1.label="newby", 
   box2.label="experienced\nR user", 
   oneFitsAll=FALSE, 
   arrow1.length=.5, 
   arrow1.label="", 
   arrow1.lwd=4, 
   label.cex=1)

## change the cex of single box labels:
plot(SurvHist, 
   box2.col=2, 
   box1.label="newby", 
   box2.label="experienced\nR user", 
   oneFitsAll=FALSE, 
   arrow1.length=.5, 
   arrow1.label="", 
   arrow1.lwd=4, 
   label1.cex=1, 
   label2.cex=2)

## The pbc data set from the survival package
library(survival)
data(pbc)
plot(with(pbc, Hist(time,status)), 
   stateLabels=c("randomized","transplant","dead"), 
   arrowLabelStyle="count")

## two competing risks
comprisk.model <- data.frame(time=1:3,status=1:3)
CRHist <- with(comprisk.model, Hist(time,status,cens.code=2))
plot(CRHist)
plot(CRHist,arrow1.label=paste(expression(eta(s,u))))

plot(CRHist,box2.label="This\nis\nstate 2",arrow1.label=paste(expression(gamma[1](t))))
plot(CRHist,box3.label="Any\nLabel",arrow2.label="any\nLabel")

## change the layout
plot(CRHist, 
   box1.label="Alive", 
   box2.label="Dead\ncause 1", 
   box3.label="Dead\ncause 2", 
   arrow1.label=paste(expression(gamma[1](t))), 
   arrow2.label=paste(expression(eta[2](t))), 
   box1.col=2, 
   box2.col=3, 
   box3.col=4, 
   nrow=2, 
   ncol=3, 
   box1.row=1, 
   box1.column=2,
box2.row=2,
box2.column=1,
box3.row=2,
box3.column=3)

## more competing risks
comprisk.model2 <- data.frame(time=1:4, status=1:4)
CRHist2 <- with(comprisk.model2, Hist(time, status, cens.code=2))
plot(CRHist2, box1.row=2)

## illness-death models
illness.death.frame <- data.frame(time=1:4,
   from=c("Disease\nfree",
          "Diseased",
          "Disease\nfree"),
   to=c("0", "Diseased", "Dead", "Dead"))
IDHist <- with(illness.death.frame, Hist(time, event=list(from,to)))
plot(IDHist)

## illness-death with recovery
illness.death.frame2 <- data.frame(time=1:5,
   from=c("Disease\nfree", "Diseased", "Diseased", "Disease\nfree"),
   to=c("0", "Diseased", "Disease\nfree", "Dead", "Dead"))
IDHist2 <- with(illness.death.frame2, Hist(time, event=list(from,to)))
plot(IDHist2)

## 4 state models
x=data.frame(from=c(1,2,1,3,4), to=c(2,1,3,4,1), time=1:5)
y=with(x, Hist(time=time, event=list(from=from, to=to)))
plot(y)

## moving the label of some arrows
d <- with(d, Hist(time, event=list(from, to)))
plot(d, tagBoxes=TRUE,
     stateLabels=c("Remission\nwithout\nGvHD",
                   "Remission\nwith\nGvHD",
                   "Relapse",
                   "Death\nwithout\nelapse"),
     arrowLabelSymbol='alpha',
     arrowlabel3.x=35,
     arrowlabel3.y=53,
     arrowlabel4.y=54,
     arrowlabel4.x=68)
```

```r

plot.prodlm

Plotting event probabilities over time
```
Description

Function to plot survival and cumulative incidence curves against time.

Usage

```r
# S3 method for class 'prodlim'
plot(x, type = "surv", cause = 1, select, newdata, add = FALSE,
     col, lty, lwd, ylim, xlim, ylab, xlab = "Time", timeconverter,
     legend = TRUE, logrank = FALSE, marktime = FALSE, confint = TRUE,
     automar, atrisk = ifelse(add, FALSE, TRUE), timeorigin = 0, axes = TRUE,
     background = TRUE, percent = TRUE, minatrisk = 0, limit = 10, ...)
```

Arguments

- `x` an object of class ‘prodlim’ as returned by the `prodlim` function.
- `type` Either "surv" or "cuminc" controls what determines the cause of the cumulative incidence function. Currently one cause is allowed at a time, but you may call the function again with add=TRUE to add the lines of the other causes.
- `cause` Select which lines to plot. This can be used when there are many strata or many competing risks to select a subset of the lines. However, a more clean way to select covariate stratat is to use argument newdata. Another application is when there are many competing risks and it is desired (for the stacked plot) to stack and show only a subset of the cumulative incidence functions.
- `select` a data frame containing covariate strata for which to show curves. When omitted element x of object x is used.
- `newdata` if TRUE curves are added to an existing plot.
- `add` color for curves. Default is 1: number(curves)
- `col` line type for curves. Default is 1.
- `lty` line width for all curves. Default is 3.
- `lwd` limits of the y-axis
- `ylim` limits of the x-axis
- `xlim` label for the y-axis
- `ylab` label for the x-axis
- `xlab` timeconverter The strings are allowed: "days2years" (conversion factor: 1/365.25) "months2years" (conversion factor: 1/12) "days2months" (conversion factor 1/30.4368499) "years2days" (conversion factor 365.25) "years2months" (conversion factor 12) "months2days" (conversion factor 30.4368499)
- `timeconverter` if TRUE a legend is plotted by calling the function `legend`. Optional arguments of the function `legend` can be given in the form `legend.x=val` where x is the name of the argument and val the desired value. See also Details.
- `legend` If TRUE, the logrank p-value will be extracted from a call to `survdiff` and added to the legend. This works only for survival models, i.e. Kaplan-Meier with discrete predictors.
marktime if TRUE the curves are tick-marked at right censoring times by invoking the function markTime. Optional arguments of the function markTime can be given in the form confint.x=val as with legend. See also Details.

confint if TRUE pointwise confidence intervals are plotted by invoking the function confInt. Optional arguments of the function confInt can be given in the form confint.x=val as with legend. See also Details.

automar If TRUE the function tries to find suitable values for the figure margins around the main plotting region.

atRisk if TRUE display numbers of subjects at risk by invoking the function atRisk. Optional arguments of the function atRisk can be given in the form atRisk.x=val as with legend. See also Details.

timeOrigin Start of the time axis

axes If true axes are drawn. See details.

background If TRUE the background color and grid color can be controlled using smart arguments SmartControl, such as background.bg="yellow" or background.bg=c("gray66","gray88"). The following defaults are passed to background by plot.prodlim: horizontal=seq(0,1,.25), vertical=NULL, bg="gray77", fg="white". See background for all arguments, and the examples below.

percent If true the y-axis is labeled in percent.

minAtRisk Integer. Show the curve only until the number at-risk is at least minAtRisk

limit When newdata is not specified and the number of lines in element X of object x exceeds limits, only the results for covariate constellations of the first, the middle and the last row in X are shown. Otherwise all lines of X are shown.

... Parameters that are filtered by SmartControl and then passed to the functions plot.legend, axis, atRisk, confInt, markTime, background

Details

From version 1.1.3 on the arguments legend.args, atRisk.args, confint.args are obsolete and only available for backward compatibility. Instead arguments for the invoked functions atRisk, legend, confInt, markTime, axis are simply specified as atRisk.cex=2. The specification is not case sensitive, thus atRisk.cex=2 or atRisk.cex=2 will have the same effect. The function axis is called twice, and arguments of the form axis1.labels, axis1.at are used for the time axis whereas axis2.pos, axis1.labels, etc. are used for the y-axis.

These arguments are processed via ...{} of plotprodlim and inside by using the function SmartControl. Documentation of these arguments can be found in the help pages of the corresponding functions.

Value

The (invisible) object.

Note

Similar functionality is provided by the function plot.survfit of the survival library
**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**See Also**

`plot`, `legend`, `axis`, `prodlim`, `plot.Hist`, `summary.prodlim`, `neighborhood`, `atRisk`, `confInt`, `markTime`, `backGround`

**Examples**

```r
## simulate right censored data from a two state model
set.seed(100)
dat <- SimSurv(100)
# with(dat, plot(Hist(time, status)))

### marginal Kaplan-Meier estimator
kmf <- prodlim(Hist(time, status) ~ 1, data = dat)
plot(kmf)
plot(kmf, timeconverter="years2months")

# change time range
plot(kmf, xlim=c(0,4))

# change scale of y-axis
plot(kmf, percent=FALSE)

# mortality instead of survival
plot(kmf, type="cuminc")

# change axis label and position of ticks
plot(kmf,
     xlim=c(0,10),
     axis1.at=seq(0,10,1),
     axis1.labels=0:10,
     xlab="Years",
     axis2.las=2,
     atrisk.at=seq(0,10,2.5),
     atrisk.title="")

# change background color
plot(kmf,
     xlim=c(0,10),
     confint.type="shadow",
     col=1,
     axis1.at=0:10,
     axis1.labels=0:10,
     xlab="Years",
     axis2.las=2,
     atrisk.at=seq(0,10,2.5),
     atrisk.title="",
     background=TRUE,
```
background.fg="white", background.horizontal=seq(0,1,25/2), background.vertical=seq(0,10,2.5), background.bg=c("gray88")

# change type of confidence limits
plot(kmfitx, xlim=c(0,10), confint.citype="dots", col=4, background=TRUE, background.bg=c("white","gray88"), background.fg="gray77", background.horizontal=seq(0,1,25/2), background.vertical=seq(0,10,2))

### Kaplan-Meier in discrete strata
kmfitx <- prodlim(Hist(time, status) ~ X1, data = dat)
plot(kmfitx)
# move legend
plot(kmfitx, legend.x="bottomleft", atRisk.cex=1.3, atRisk.title="No. subjects")

## Control the order of strata
## since version 1.5.1 prodlim does obey the order of
## factor levels
dat$group <- factor(cut(dat$X2, c(-Inf,0,0.5,Inf)), labels=c("High","Intermediate","Low"))
kmfitG <- prodlim(Hist(time, status) ~ group, data = dat)
plot(kmfitG)

## relevel
dat$group2 <- factor(cut(dat$X2, c(-Inf,0,0.5,Inf)), levels=c("[0.5, Inf]","[0,0.5]","(-Inf,0]"), labels=c("Low","Intermediate","High"))
kmfitG2 <- prodlim(Hist(time, status) ~ group2, data = dat)
plot(kmfitG2)

## add log-rank test to legend
plot(kmfitx, atRisk.cex=1.3, logrank=TRUE, legend.x="topright", atrisk.title="at-risk")

## change atrisk labels
plot(kmfitx, legend.x="bottomleft", atrisk.title="Patients", atrisk.cex=0.9, atrisk.labels=c("X1=0","X1=1"))
# multiple categorical factors

```
kmfitXG <- prodlim(Hist(time,status)-X1+group2, data=dat)
p plot(kmfitXG, select=1:2)
```

### Kaplan-Meier in continuous strata

```
kmfitX2 <- prodlim(Hist(time, status) ~ X2, data = dat)
p plot(kmfitX2, xlim=c(0,10))
```

# specify values of X2 for which to show the curves
```
p plot(kmfitX2, xlim=c(0,10), newdata=data.frame(X2=c(-1.8,0,1.2)))
```

### Cluster-correlated data

```
library(survival)
cdat <- cbind(SimSurv(20), patnr=sample(1:5, size=20, replace=TRUE))
k mfitC <- prodlim(Hist(time, status) ~ cluster(patnr), data = cdat)
p plot(kmfitC)
p plot(kmfitC, atrisk.labels=c("Units","Patients"))
```

```
k mfitC2 <- prodlim(Hist(time, status) ~ X1 + cluster(patnr), data = cdat)
p plot(kmfitC2)
p plot(kmfitC2, atrisk.labels=c("Teeth","Patients","Teeth","Patients"), atrisk.col=c(1,1,2,2))
```

### Cluster-correlated data with strata

```
n = 50
foo = runif(n)
bar = rexp(n)
baz = rexp(n,1/2)
d = stack(data.frame(foo, bar, baz))
d$scl = sample(10, 3*n, replace=TRUE)
fit = prodlim(Surv(values) ~ ind + cluster(cl), data=d)
p plot(fit)
```

### simulate right censored data from a competing risk model

```
datCR <- SimCompRisk(100)
with(datCR, plot(Hist(time, event)))
```

### marginal Aalen-Johansen estimator

```
a jjfit <- prodlim(Hist(time, event) ~ 1, data = datCR)
p plot(a jjfit) # same as plot(a jjfit, cause=1)
```

# cause 2
```
p plot(a jjfit, cause=2)
```

# both in one
```
p plot(a jjfit, cause=1)
p plot(a jjfit, cause=2, add=TRUE, col=2)
```

### stacked plot
plotCompetingRiskModel

Plotting a competing-risk-model.

Description

Plotting a competing-risk-model.

Usage

plotCompetingRiskModel(stateLabels, horizontal = TRUE, ...)

Arguments

stateLabels  Labels for the boxes.
horizontal     The orientation of the plot.
...           Arguments passed to plot.Hist.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>
plotIllnessDeathModel

See Also

plotIllnessDeathModel, plot.Hist

Examples

plotCompetingRiskModel()
plotCompetingRiskModel(labels=c("a","b"))
plotCompetingRiskModel(labels=c("a","b","c"))

plotIllnessDeathModel

Plotting an illness-death-model.

Description

Plotting an illness-death-model using plot.Hist.

Usage

plotIllnessDeathModel(stateLabels, style = 1, recovery = FALSE, ...)

Arguments

stateLabels Labels for the three boxes.
style Either 1 or anything else, switches the orientation of the graph. Hard to explain
recovery Logical. If TRUE there will be an arrow from the illness state to the initial state.
... Arguments passed to plot.Hist.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

See Also

plotCompetingRiskModel, plot.Hist

Examples

plotIllnessDeathModel()
plotIllnessDeathModel(style = 2)
plotIllnessDeathModel(style = 2,
        stateLabels = c("a","b\nc","d"),
        box1.col = "yellow",
        box2.col = "green",
        box3.col = "red")
predict.prodlim  Predicting event probabilities from product limit estimates

Description

Evaluation of estimated survival or event probabilities at given times and covariate constellations.

Usage

```r
## S3 method for class 'prodlim'
predict(object, times, newdata, level=1,
         type = c("surv", "cuminc", "list"), mode = "list", bytime = FALSE,
         cause = 1, ...)
```

Arguments

- `object`: A fitted object of class "prodlim".
- `times`: Vector of times at which to return the estimated probabilities.
- `newdata`: A data frame with the same variable names as those that appear on the right hand side of the 'prodlim' formula. If there are covariates this argument is required.
- `level`: Integer specifying the sorting of the output: ‘0’ sort by time and newdata; ‘1’ only by time; ‘2’ no sorting at all
- `type`: Choice between "surv", "cuminc", "list": "surv": predict survival probabilities only survival models "cuminc": predict cumulative incidences only competing risk models "list": find the indices corresponding to times and newdata. See value. Defaults to "surv" for two-state models and to "cuminc" for competing risk models.
- `mode`: Only for `type`="surv" and `type"="cuminc". Can either be "list" or "matrix". For "matrix" the predicted probabilities will be returned in matrix form.
- `bytime`: Logical. If TRUE and `mode"="matrix" the matrix with predicted probabilities will have a column for each time and a row for each newdata. Only when `object$ covariate.type>1 and more than one time is given.
- `cause`: The cause for predicting the cause-specific cumulative incidence function in competing risk models.
- `...`: Only for compatibility reasons.

Details

Predicted (survival) probabilities are returned that can be plotted, summarized and used for inverse of probability of censoring weighting.
Value

type="surv" A list or a matrix with survival probabilities for all times and all newdata.
type="cuminc" A list or a matrix with cumulative incidences for all times and all newdata.
type="list" A list with the following components:

- times: The argument times carried forward
- predictors: The relevant part of the argument newdata.
- indices: A list with the following components:
  - time: Where to find values corresponding to the requested times
  - strata: Where to find values corresponding to the values of the variables in newdata. Together with time and strata show where to find the predicted probabilities.
- dimensions: A list with the following components:
  - time: The length of times
  - strata: The number of rows in newdata
  - namesNstrata: Labels for the covariate values.

Author(s)

Thomas Alexander Gerds <tag@biostat.ku.dk>

See Also

predictSurvIndividual

Examples

dat <- SimSurv(400)
fit <- prodlim(Hist(time,status)-1,data=dat)

## predict the survival probs at selected times
predict(fit,times=c(10,100,1000))

## works also outside the usual range of the Kaplan-Meier
predict(fit,times=c(-1,0,10,100,1000,10000))

## newdata is required if there are strata
## or neighborhoods (i.e. overlapping strata)
mfit <- prodlim(Hist(time,status)-X1+X2,data=dat)
predict(mfit,times=c(-1,0,10,100,1000,10000),newdata=dat[18:21,])

## this can be requested in matrix form
predict(mfit,times=c(-1,0,10,100,1000,10000),newdata=dat[18:21,],mode="matrix")

## and even transposed
predict(mfit,times=c(-1,0,10,100,1000,10000),newdata=dat[18:21,],mode="matrix",bytime=TRUE)
predictSurvIndividual  

Predict individual survival probabilities

Description

Function to extract the predicted probabilities at the individual event times that have been used for fitting a prodlim object.

Usage

predictSurvIndividual(object, lag = 1)

Arguments

object  
A fitted object of class "prodlim".

lag  
Integer. '0' means predictions at the individual times, 1 means just before the individual times, etc.

Value

A vector of survival probabilities.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

predict.prodlim,predictSurv,

Examples

SurvFrame <- data.frame(time=1:10,status=rbinom(10,1,.5))
x <- prodlim(formula=Hist(time=time,status!=0)=1,data=SurvFrame)  
predictSurvIndividual(x,lag=1)
print.prodlim

Print objects in the prodlim library

Description

Pretty printing of objects created with the functionality of the ‘prodlim’ library.

Usage

## S3 method for class 'prodlim'
print(x, ...)

Arguments

x: Object of class prodlim, Hist and neighborhood.

...: Not used.

Author(s)

Thomas Gerds <tag@biostat.ku.dk>

See Also

summary.prodlim, predict.prodlim

prodlim

Functions for estimating probabilities from right censored data

Description

Functions for estimating probabilities from right censored data

Nonparametric estimation in event history analysis. Featuring fast algorithms and user friendly syntax adapted from the survival package. The product limit algorithm is used for right censored data; the self-consistency algorithm for interval censored data.

Usage

prodlim(formula, data = parent.frame(), subset, na.action = NULL, reverse = FALSE, conf.int = 0.95, bandwidth = NULL, caseweights = NULL, discrete.level = 3, x = TRUE, maxiter = 1000, grid, tol = 7, method = c("npmle", "one.step", "impute.midpoint", "impute.right"), exact = TRUE, type)
Arguments

**formula**
A formula whose left hand side is a `Hist` object. In some special cases it can also be a `Surv` response object, see the details section. The right hand side is as usual a linear combination of covariates which may contain at most one continuous factor. Whether or not a covariate is recognized as continuous or discrete depends on its class and on the argument `discrete.level`. The right hand side may also be used to specify clusters, see the details section.

**data**
A data.frame in which all the variables of `formula` can be interpreted.

**subset**
Passed as argument `subset` to function `subset` which applied to data before the formula is processed.

**na.action**
All lines in data with any missing values in the variables of `formula` are removed.

**reverse**
For right censored data, if `reverse=TRUE` then the censoring distribution is estimated.

**conf.int**
The level (between 0 and 1) for two-sided pointwise confidence intervals. Defaults to 0.95. Remark: only plain Wald-type confidence limits are available.

**bandwidth**
Smoothing parameter for nearest neighborhoods based on the values of a continuous covariate. See function `neighborhood` for details.

**caseweights**
Weights applied to the contribution of each subject to change the number of events and the number at risk. This can be used for bootstrap and survey analysis. Should be a vector of the same length and the same order as `data`.

**discrete.level**
Numeric covariates are treated as factors when their number of unique values exceeds not `discrete.level`. Otherwise the product limit method is applied, in overlapping neighborhoods according to the bandwidth.

**x**
logical value: if TRUE, the full covariate matrix with is returned in component `model.matrix`. The reduced matrix contains unique rows of the full covariate matrix and is always returned in component `x`.

**maxiter**
For interval censored data only. Maximal number of iterations to obtain the nonparametric maximum likelihood estimate. Defaults to 1000.

**grid**
For interval censored data only. When `method=one.step` grid for one-step product limit estimate. Defaults to sorted list of unique left and right endpoints of the observed intervals.

**tol**
For interval censored data only. Numeric value whose negative exponential is used as convergence criterion for finding the nonparametric maximum likelihood estimate. Defaults to 7 meaning exp(-7).

**method**
For interval censored data only. If equal to "rpmle" (the default) use the usual Turnbull algorithm, else the product limit version of the self-consistent estimate.

**exact**
If TRUE the grid of time points used for estimation includes all the L and R endpoints of the observed intervals.

**type**
In two state models either "surv" for the Kaplan-Meier estimate of the survival function or "cuminc" for 1-Kaplan-Meier. Default is "surv" when `reverse==FALSE` and "cuminc" when `reverse==TRUE`. In competing risks models it has to be "cuminc" Aalen-Johansen estimate of the cumulative incidence function.
Details

The response of `formula` (i.e., the left hand side of the `~` operator) specifies the model.

In two-state models – the classical survival case – the standard Kaplan-Meier method is applied. For this the response can be specified as a `Surv` or as a `Hist` object. The `Hist` function allows you to change the code for censored observations, e.g., `Hist(time, status, cens.code="4")`.

Besides a slight gain of computing efficiency, there are some extensions that are not included in the current version of the survival package:

(0) The Kaplan-Meier estimator for the censoring times `reverse=TRUE` is correctly estimated when there are ties between event and censoring times.

(1) A conditional version of the kernel smoothed Kaplan-Meier estimator for at most one continuous predictors using nearest neighborhoods (Beran 1981, Stute 1984, Akritas 1994).

(2) For cluster-correlated data the right hand side of `formula` may identify a `cluster` variable. In that case Greenwood’s variance formula is replaced by the formula of Ying & Wei (1994).

(3) Competing risk models can be specified via `Hist` response objects in `formula`.

The Aalen-Johansen estimator is applied for estimating the cumulative incidence functions for all causes. The advantage over the function `cuminc` of the cmprsk package are user-friendly model specification via `Hist` and sophisticated print, summary, predict and plot methods.

Under construction:

(U0) Interval censored event times specified via `Hist` are used to find the nonparametric maximum likelihood estimate. Currently this works only for two-state models and the results should match with those from the package ‘Icens’.

(U1) Extensions to more complex multi-states models

(U2) The nonparametric maximum likelihood estimate for interval censored observations of competing risks models.

Value

Object of class "prodlim". See `print.prodlim`, `predict.prodlim`, `predict`, `summary.prodlim`, `plot.prodlim`.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Thomas A. Gerds <tag@biostat.ku.dk>

References

Andersen, Borgan, Gill, Keiding (1993) Springer ‘Statistical Models Based on Counting Processes’
Ying, Wei (1994) Journal of Multivariate Analysis 50, 17-29 The Kaplan-Meier estimate for dependent failure time observations

See Also

deprob, predictSurv, predictSurvIndividual, predictCuminc, Hist, neighborhood, Surv, survfit, strata.

Examples

```r
###-----------------------------two-state survival model-----------------------------
dat <- SimSurv(30)
with(dat, plot(Hist(time, status)))
fit <- prodlim(Hist(time, status)~1, data=dat)
print(fit)
plot(fit)
summary(fit)
quantile(fit)

## Subset
fit1a <- prodlim(Hist(time, status)~1, data=dat, subset=dat$X1==1)
fit1b <- prodlim(Hist(time, status)~1, data=dat, subset=dat$X1==1 & dat$X2>0)

### ------------------------clustered data-----------------------------
library(survival)
cdat <- cbind(SimSurv(30), patnr=sample(1:5, size=30, replace=TRUE))
fit <- prodlim(Hist(time, status)~cluster(patnr), data=cdat)
print(fit)
plot(fit)
summary(fit)

###-----------------compare Kaplan-Meier to survival package------------------

dat <- SimSurv(30)
pfit <- prodlim(Surv(time, status)~1, data=dat)
pfit <- prodlim(Hist(time, status)~1, data=dat) ## same thing
sfit <- survfit(Surv(time, status)~1, data=dat, conf.type="plain")
## same result for the survival distribution function
all(round(pfit$surv,12)==round(sfit$surv,12))
summary(pfit, digits=3)
summary(sfit, times=quantile(unique(dat$time)))

###-------------------estimating the censoring survival function----------------

rdat <- data.frame(time=c(1,2,3,3,3,4,5,6,7), status=c(1,0,0,1,0,1,0,1,1,0))
rpfit <- prodlim(Hist(time, status)~1, data=rdat, reverse=TRUE)
rsvfit <- survfit(Surv(time, 1-status)~1, data=rdat, conf.type="plain")
## When there are ties between times at which events are observed
## times at which subjects are right censored, then the convention
## is that events come first. This is not obeyed by the above call to survfit,
```
```
## and hence only prodlim delivers the correct reverse Kaplan-Meier: 
cbind("Wrong":"=rsfit$surv","Correct":"=rpfit$surv")

#-----------------stratified Kaplan-Meier-------------------

pfit.X2 <- prodlim(Surv(time,status)-X2,data=dat)
summary(pfit.X2)
summary(pfit.X2,intervals=TRUE)
plot(pfit.X2)

#--------continuous covariate: Stone-Beran estimate---------

prodlim(Surv(time,status)-X1,data=dat)

#----------both discrete and continuous covariates-----------

prodlim(Surv(time,status)-X2+X1,data=dat)

#----------------interval censored data---------------------

dat <- data.frame(L=1:10,R=c(2,3,12,8,9,10,7,12,12,12)),status=c(1,1,0,1,1,1,0,0,0))
with(dat,Hist(time=list(L,R),event=status))

dat$event=1
npmle.fitml <- prodlim(Hist(time=list(L,R),event=1),data=dat)

#----------------competing risks---------------------------

CompRiskFrame <- data.frame(time=1:100,event=rbinom(100,2,.5),X=rbinom(100,1,.5))

compRiskFrame <- prodlim(Hist(time,event)-X,data=CompRiskFrame)
summary(compRiskFrame)
plot(compRiskFrame)

# Changing the cens.code:

dat <- data.frame(time=1:10,status=c(1,2,1,2,5,5,1,1,2,2))
fit <- prodlim(Hist(time,status)-1,data=dat)
print(fit$model.response)
fit <- prodlim(Hist(time,status,cens.code="2")-1,data=dat)
print(fit$model.response)
plot(fit)
plot(fit,cause="5")

#----------------delayed entry-----------------------------

## left-truncated event times with competing risk endpoint

dat <- data.frame(entry=c(7,3,11,12,11,2,1,7,15,17,3),time=10:20,status=c(1,0,2,2,0,0,1,2,0,2,0))
fitd <- prodlim(Hist(time,time,event=status,entry=entry)-1,data=dat)
summary(fitd)
```
quantile.prodlim

Description
Quantiles for Kaplan-Meier and Aalen-Johansen estimates.

Usage
## S3 method for class 'prodlim'
quantile(x, q, cause = 1, ...)

Arguments
x Object of class "prodlim".
q Quotiles. Vector of values between 0 and 1.
cause For competing risks the cause of interest.
... not used

Author(s)
Thomas Alexander Gerds <tag@biostat.ku.dk>

Examples
library(lava)
set.seed(1)
d=SimSurv(30)
f=prodlim(Hist(time,status)-1,data=d)
f1=prodlim(Hist(time,status)-X1,data=d)
# default: median and IQR
quantile(f)
quantile(f1)
# median alone
quantile(f,.5)
quantile(f1,.5)

# competing risks
set.seed(3)
 dd = SimCompRisk(30)
 ff=prodlim(Hist(time,event)-1,data=dd)
 ff1=prodlim(Hist(time,event)-X1,data=dd)
 # default: median and IQR
quantile(ff)
quantile(ff1)
Calculation of Efron’s re-distribution to the right algorithm to obtain the Kaplan-Meier estimate.

Description
Calculation of Efron’s re-distribution to the right algorithm to obtain the Kaplan-Meier estimate.

Usage
redist(time, status)

Arguments
- time: A numeric vector of event times.
- status: The event status vector takes the value 1 for observed events and the value 0 for right censored times.

Value
Calculations needed to

Author(s)
Thomas A. Gerds <tag@biostat.ku.dk>

See Also
prodlim

Examples
redist(time=c(.35,.4,.51,.51,.73),status=c(0,1,1,0,1))
row.match

Identifying rows in a matrix or data.frame

Description

Function for finding matching rows between two matrices or data.frames. First the matrices or
data.frames are vectorized by row wise pasting together the elements. Then it uses the function
match. Thus the function returns a vector with the row numbers of (first) matches of its first argu-
ment in its second.

Usage

row.match(x, table, nomatch = NA)

Arguments

- **x**: Vector or matrix whose rows are to be matched
- **table**: Matrix or data.frame that contain the rows to be matched against.
- **nomatch**: the value to be returned in the case when no match is found. Note that it is
  coerced to 'integer'.

Value

A vector of the same length as 'x'.

Author(s)

Thomas A. Gerds

See Also

match

Examples

```
tax <- data.frame(num=1:26, abc=letters)
x <- c(3,"c")
row.match(x, tab)
x <- data.frame(n=c(3,8), z=c("c","h"))
row.match(x, tab)
```
Simulate competing risks data

Description

Simulate right censored competing risks data with two covariates X1 and X2. Both covariates have effect exp(1) on the hazards of event 1 and zero effect on the hazard of event 2.

Usage

SimCompRisk(N, ...)

Arguments

N  sample size
...  do nothing.

Details

This function calls crModel, then adds covariates and finally calls sim.lvm.

Value

data.frame with simulated data

Author(s)

Thomas Alexander Gerds

Examples

SimCompRisk(10)

Simulate survival data

Description

Simulate right censored survival data with two covariates X1 and X2, both have effect exp(1) on the hazard of the unobserved event time.

Usage

SimSurv(N, ...)
Arguments

- `N` sample size
- `...` do nothing

Details

This function calls `survModel`, then adds covariates and finally calls `sim.lvm`.

Value
data.frame with simulated data

Author(s)
Thomas Alexander Gerds

References

Examples

SimSurv(10)

sindex

Index for evaluation of step functions.

Description

Returns an index of positions. Intended for evaluating a step function at selected times. The function counts how many elements of a vector, e.g. the jump times of the step function, are smaller or equal to the elements in a second vector, e.g. the times where the step function should be evaluated.

Usage

`sindex(jump.times, eval.times, comp = "smaller", strict = FALSE)`

Arguments

- `jump.times` Numeric vector: e.g. the unique jump times of a step function.
- `eval.times` Numeric vector: e.g. the times where the step function should be evaluated.
- `comp` If "greater" count the number of jump times that are greater (greater or equal when strict==FALSE) than the eval times.
- `strict` If TRUE make the comparison of jump times and eval times strict.
Details

If all jump.times are greater than a particular eval.time the sindex returns 0. This must be considered when sindex is used for subsetting, see the Examples below.

Value

Index of the same length as eval.times containing the numbers of the jump.times that are smaller than or equal to eval.times.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

Examples

test <- list(time = c(1, 1, 5, 5, 2, 7, 9),
            status = c(1, 0, 1, 0, 1, 1, 0))
fit <- prodlim(Hist(time, status)-1, data=test)
jtimes <- fit$time
etimes <- c(0, 5, 2, 8, 10)
fit$surv
  c(1, fit$surv)[1+sindex(jtimes,etimes)]

SmartControl

Function to facilitate the control of arguments passed to subroutines.

Description

Many R functions need to pass several arguments to several different subroutines. Such arguments can are given as part of the three magic dots "...". The function SmartControl reads the dots together with a list of default values and returns for each subroutine a list of arguments.

Usage

SmartControl(call, keys, ignore, defaults, forced, split, ignore.case = TRUE,
             replaceDefaults, verbose = TRUE)

Arguments

call A list of named arguments, as for example can be obtained via list(...).
keys A vector of names of subroutines.
ignore A list of names which are removed from the argument call before processing.
defaults A named list of default argument lists for the subroutines.
forced A named list of forced arguments for the subroutines.
**stopTime**

Stop the time of an event history object

### Description

All event times are stopped at a given time point and corresponding events are censored

### Usage

```r
stopTime(object, stop.time)
```
Arguments

object Event history object as obtained with Hist
stop.time Time point at which to stop the event history object

Value

Stopped event history object where all times are censored at stop.time. All observations with times greater than stop.time are set to stop.time and the event status is set to attr(object,"cens.code"). A new column "stop.time" is equal to 1 for stopped observations and equal to 0 for the other observations.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

Hist

Examples

```r
set.seed(29)
d <- SimSurv(10)
h <- with(d,Hist(time,status))
h
stopTime(h,8)
stopTime(h,5)

## works also with Surv objects
library(survival)
s <- with(d,Surv(time,status))
stopTime(s,5)

## competing risks
set.seed(29)
dr <- SimCompRisk(10)
hr <- with(dr,Hist(time,event))
hr
stopTime(hr,8)
stopTime(hr,5)
```
strip.terms

Strip special functions from terms

Description

Reformulate a terms object such that some specials are stripped off.

Usage

strip.terms(terms, specials, alias.names = NULL, unspecials = NULL,
arguments, keep.response = TRUE)

Arguments

terms Terms object
specials Character vector of specials which should be stripped off
alias.names Optional. A named list with alias names for the specials.
unspecials Optional. A special name for treating all the unspecial terms.
arguments A named list of arguments, one for each element of specials. Elements are
passed to parseSpecialNames.
keep.response Keep the response in the resulting object?

Details

This function is used to remove special specials, i.e., those which cannot or should not be evaluated. IMPORTANT: the unstripped terms need to know about all specials including the aliases. See examples.

Value

Reformulated terms object with an additional attribute which contains the stripped.specials.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

parseSpecialNames reformulate drop.terms
Examples

```r
## parse a survival formula and identify terms which
## should be treated as proportional or timevarying:
f <- Surv(time,status)-age+prop(factor(edema))+timevar(sex,test=0)+prop(bili,power=1)
tt <- terms(f,specials=c("prop","timevar"))
attr(tt,"specials")
st <- strip.terms(tt,specials=c("prop","timevar"),arguments=NULL)
formula(st)
attr(st,"specials")
attr(st,"stripped.specials")

## provide a default value for argument power of proportional treatment
## and argument test of timevarying treatment:
st2 <- strip.terms(tt,
  specials=c("prop","timevar"),
  arguments=list("prop"=list("power"=0),"timevar"=list("test"=0)))
formula(st2)
attr(st2,"stripped.specials")
attr(st2,"stripped.arguments")

## treat all unspecial terms as proportional
st3 <- strip.terms(tt,
  unspecials="prop",
  specials=c("prop","timevar"),
  arguments=list("prop"=list("power"=0),"timevar"=list("test"=0)))
formula(st3)
attr(st3,"stripped.specials")
attr(st3,"stripped.arguments")

## allow alias names: strata for timevar and tp, const for prop.
## IMPORTANT: the unstripped terms need to know about
## all specials including the aliases
f <- Surv(time,status)-age+const(factor(edema))+strata(sex,test=0)+prop(bili,power=1)+tp(albumin)
tt2 <- terms(f,specials=c("prop","timevar","strata","tp","const"))
st4 <- strip.terms(tt2,
  specials=c("prop","timevar"),
  unspecials="prop",
  alias.names=list("timevar"="strata","prop"=c("const","tp")),
  arguments=list("prop"=list("power"=0),"timevar"=list("test"=0)))
formula(st4)
attr(st4,"stripped.specials")
attr(st4,"stripped.arguments")

## test if alias works also without unspecial argument
st5 <- strip.terms(tt2,
  specials=c("prop","timevar"),
  alias.names=list("timevar"="strata","prop"=c("const","tp")),
  arguments=list("prop"=list("power"=0),"timevar"=list("test"=0)))
formula(st5)
attr(st5,"stripped.specials")
attr(st5,"stripped.arguments")
```
library(survival)
data(pbc)
model.design(st4, data=pbc[1:3], specialsDesign=TRUE)
model.design(st5, data=pbc[1:3], specialsDesign=TRUE)

summary.Hist  Summary of event histories

Description
Describe events and censoring patterns of an event history.

Usage
## S3 method for class 'Hist'
summary(object, verbose = TRUE, ...)

Arguments
object An object with class ‘Hist’ derived with Hist
verbose Logical. If FALSE any printing is suppressed.
... Not used

Value
NULL for survival and competing risk models. For other multi-state models, it is a list with the following entries:
states the states of the model
transitions the transitions between the states
trans.frame a data.frame with the from and to states of the transitions

Author(s)
Thomas A. Gerds <tag@biostat.ku.dk>

See Also
Hist, plot.Hist

Examples
icensFrame <- data.frame(L=c(1,1,3,4,6),R=c(2,NA,3,6,9),event=c(1,1,1,2,2))
with(icensFrame, summary(Hist(time=list(L,R))))
Summary method for prodlim objects.

Description

Summarizing the result of the product limit method in life-table format. Calculates the number of subjects at risk and counts events and censored observations at specified times or in specified time intervals.

Usage

```R
## S3 method for class 'prodlim'
summary(object, times, newdata, max.tables = 20,
         surv = TRUE, cause, intervals = FALSE, percent = FALSE,
         showTime = TRUE, asMatrix = FALSE, ...)
```

Arguments

- **object**: An object with class ‘prodlim’ derived with `prodlim`
- **times**: Vector of times at which to return the estimated probabilities.
- **newdata**: A data frame with the same variable names as those that appear on the right hand side of the ‘prodlim’ formula. Defaults to `object$X`.
- **max.tables**: Integer. If `newdata` is not given the value of `max.tables` decides about the maximal number of tables to be shown. Defaults to 20.
- **surv**: Logical. If `newdata` report event probabilities instead of survival probabilities. Only available for `object$model == "survival"`.
- **cause**: The cause for predicting the cause-specific cumulative incidence function in competing risk models.
- **intervals**: Logical. If TRUE count events and censored in intervals between the values of `times`.
- **percent**: Logical. If TRUE all estimated values are multiplied by 100 and thus interpretable on a percent scale.
- **showTime**: If TRUE evaluation times are put into a column of the output table, otherwise evaluation times are shown as rownames.
- **asMatrix**: Control the output format when there are multiple life tables, either because of covariate strata or competing causes or both. If not missing and not FALSE, reduce multiple life tables into a matrix with new columns `X` for covariate strata and `Event` for competing risks.
- **...**: Further arguments that are passed to the print function.

Details

For cluster-correlated data the number of clusters at-risk are are also given. Confidence intervals are displayed when they are part of the fitted object.
Summary

A data.frame with the relevant information.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

See Also

prodlim, summary.Hist

Examples

library(lava)
set.seed(17)
m <- survModel()
distribution(m,"age") <- uniform.lvm(30,80)
distribution(m,"sex") <- binomial.lvm()
m <- categorical(m,"z",K=3)
regression(m,eventtime~age) <- 0.01
regression(m,eventtime~sex) <- -0.4
d <- sim(m,50)
d$sex <- factor(d$sex,levels=c(0,1),labels=c("female","male"))
d$Z <- factor(d$z,levels=c(1,0,2),labels=c("B","A","C"))

# Univariate Kaplan-Meier
# ---------------------------------------------------------------
fit0 <- prodlim(Hist(time,event)-1,data=d)
summary(fit0)

## show survival probabilities as percentage and
## count number of events within intervals of a
## given time-grid:
summary(fit0,times=c(1,5,10,12),percent=TRUE,intervals=TRUE)

## the result of summary has a print function
## which passes ... to print and print.listof
sx <- summary(fit0,times=c(1,5,10,12),percent=TRUE,intervals=TRUE)
print(sx,digits=3)

## show cumulative incidences (1-survival)
summary(fit0,times=c(1,5,10,12),surv=FALSE,percent=TRUE,intervals=TRUE)

# Stratified Kaplan-Meier
# ---------------------------------------------------------------

fit1 <- prodlim(Hist(time,event)-sex,data=d)
print(summary(fit1,times=c(1,5,10),intervals=TRUE,percent=TRUE),digits=3)
summary(fit1,times=c(1,5,10),asMatrix=TRUE,intervals=TRUE,percent=TRUE)
fit2 <- prodlim(Hist(time,event)-Z,data=d)
print(summary(fit2,times=c(1,5,10),intervals=TRUE,percent=TRUE),digits=3)

## Continuous strata (Beran estimator)
# ---------------------------------------------------------------------------
fit3 <- prodlim(Hist(time,event)-age,data=d)
print(summary(fit3,
   times=c(1,5,10),
   newdata=data.frame(age=c(20,50,70)),
   intervals=TRUE,
   percent=TRUE),digits=3)

## stratified Beran estimator
# ---------------------------------------------------------------------------
fit4 <- prodlim(Hist(time,event)-age+sex,data=d)
print(summary(fit4,
   times=c(1,5,10),
   newdata=data.frame(age=c(20,50,70),sex=c("female","male","male")),
   intervals=TRUE,
   percent=TRUE),digits=3)

print(summary(fit4,
   times=c(1,5,10),
   newdata=data.frame(age=c(20,50,70),sex=c("female","male","male")),
   intervals=TRUE,collapse=TRUE,
   percent=TRUE),digits=3)

## assess results from summary
x <- summary(fit4,times=10,newdata=expand.grid(age=c(60,40,50),sex=c("male","female")))
cbind(names(x$table),do.call("rbind",lapply(x$table,round)))

x <- summary(fit4,times=10,newdata=expand.grid(age=c(60,40,50),sex=c("male","female")))

## Competing risks: Aalen-Johansen
# ---------------------------------------------------------------------------
d <- SimCompRisk(30)
crfit <- prodlim(Hist(time,event)-X1,data=d)
summary(crfit,times=c(1,2,5))
summary(crfit,times=c(1,2,5),cause=1,intervals=TRUE)
summary(crfit,times=c(1,2,5),cause=1,asMatrix=TRUE)
summary(crfit,times=c(1,2,5),cause=1:2,asMatrix=TRUE)

# extract the actual tables from the summary
sumfit <- summary(crfit,times=c(1,2,5),print=FALSE)
sumfit$table[[1]] # cause 1
sumfit$table[[2]] # cause 2

# '
Description

Create a survival model to simulate a right censored event time data without covariates

Usage

survModel()

Details

This function requires the lava package.

Value

A structural equation model initialized with three variables: the latent event time, the latent right censored time, and the observed right censored event time.

Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>
Index

+ Topic **Graphics**
  SmartControl, 49
+ Topic **cluster**
  prodlim, 39
+ Topic **misc**
  row.match, 46
  sindex, 48
+ Topic **nonparametric**
  prodlim, 39
+ Topic **smooth**
  neighborhood, 22
+ Topic **survival**
  atRisk, 2
  background, 3
  confInt, 4
  getEvent, 10
  hist, 3
  jackknife, 15
  markTime, 18
  meanNeighbors, 19
  PercentAxis, 24
  plotHist, 25
  plot.prodlim, 29
  plotCompetingRiskModel, 34
  plotIllnessDeathModel, 35
  predict.prodlim, 36
  predictSurvIndividual, 38
  print.prodlim, 39
  predict.cuminc, 42
  predict.cuminc (predict.prodlim), 36
  predictSurv, 38
  predictSurv (predict.prodlim), 36
  col2rgb, 6
  confInt, 3, 4, 18, 30, 31
  crModel, 5
  dimColor, 6
  dpik, 22, 23
  EventHistory.frame, 7, 20
  getEvent, 10
  getStates, 11
  hist, 7, 10, 12, 26, 41, 42, 54
  jackknife, 15, 17
  leaveOneOut, 16
  legend, 30, 31
  lines.prodlim (plot.prodlim), 29
  List2Matrix, 17
  markTime, 3, 5, 18, 30, 31
  meanNeighbors, 19
  model.design, 7, 19
  model.matrix, 20
  neighborhood, 19, 22, 31, 42
  parseSpecialNames, 23
  PercentAxis, 24
  plot, 30, 31
  plotHist, 13, 25, 31, 34, 35, 54
  plot.prodlim, 3, 5, 18, 25, 28, 41, 50
  plot.survfit, 30
  plotCompetingRiskModel, 34, 35
  plotIllnessDeathModel, 35, 35
  predict.prodlim, 36, 38, 39, 41
  predictCuminc, 42
  predictCuminc (predict.prodlim), 36
  predictSurv, 38, 42
  predictSurv (predict.prodlim), 36
predictSurvIndividual, 37, 38, 42
print.Hist (print.prodlm), 39
print.neighborhood (print.prodlm), 39
print.prodlm, 39, 41
prodlm, 12, 13, 15, 23, 31, 39, 55, 56
prodlm-package (prodlm), 39

quantile.prodlm, 44

redist, 45
rgb, 6
row.match, 46

SimCompRisk, 47
SimSurv, 47
sindex, 48
SmartControl, 26, 30, 49
stopTime, 50
strata, 7, 42
strip.terms, 7, 52
summary.Hist, 13, 54, 56
summary.prodlm, 31, 39, 41, 55
Surv, 12, 41, 42
survfit, 42
survModel, 58