

Package ‘prodlim’

January 2, 2012

Title Product Limit Estimation for event history and survival analysis

Version 1.2.9

Author Thomas A. Gerds

Description

Fast and user friendly nonparametric estimation in censored survival (event history) analysis.

Depends R (>= 1.9.1), stats, KernSmooth, survival

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

License GPL (>= 2)

Repository CRAN

Date/Publication 2011-12-15 10:39:53

R topics documented:

atRisk	2
backGround	3
clusterTestData	4
confInt	5
getEvent	6
Hist	7
jackknife	10
markTime	12
meanNeighbors	13
neighborhood	14
NN	15
PercentAxis	16
plot.Hist	16
plot.prodlim	19
plot.SimSurv	22
plotIllnessDeathModel	23
predict.prodlim	24

predictSurvIndividual	26
print.prodlim	27
prodlim	27
quantile.prodlim	31
row.match	32
sampleCompriskFrame	33
SimSurv	33
sindex	37
SmartControl	38
summary.Hist	39
summary.prodlim	40

Index	42
--------------	-----------

atRisk	<i>Drawing numbers of subjects at-risk of experiencing an event below Kaplan-Meier and Aalen-Johannsen plots.</i>
--------	---

Description

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

Usage

```
atRisk(x, newdata, times, line, col, interspace, cex, labels, pos, adj, dist, adjust.labels = TRUE, ...)
```

Arguments

<code>x</code>	an object of class ‘prodlim’ as returned by the <code>prodlim</code> function.
<code>newdata</code>	see <code>plot.prodlim</code>
<code>times</code>	Where to compute the atrisk numbers.
<code>line</code>	Distance of the atrisk numbers from the inner plot.
<code>col</code>	The color of the text.
<code>interspace</code>	Distance between rows of atrisk numbers.
<code>cex</code>	Passed on to <code>mtext</code> for both atrisk numbers and labels.
<code>labels</code>	Labels for the atrisk rows.
<code>pos</code>	The value is passed on to the <code>mtext</code> argument <code>at</code> for the labels (not the atriks numbers).
<code>adj</code>	Passed on to <code>mtext</code> for the labels (not the atriks numbers).
<code>dist</code>	If <code>line</code> is missing, the distance of the upper most atrisk row from the inner plotting region: <code>par()\$mgp[2]</code> .
<code>adjust.labels</code>	If TRUE the labels are left adjusted.
<code>...</code>	Further arguments that are passed to the function <code>mtext</code> .

Details

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

Value

Nil

Author(s)

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

See Also

[plot.prodim](#), [confInt](#), [markTime](#)

backGround	<i>Background and grid color control.</i>
------------	---

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.prodim](#).

Usage

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

Arguments

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

Author(s)

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

Examples

```
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))
```

clusterTestData	<i>Data set with a clustered data structure, for testing the product limit method.</i>
-----------------	--

Description

Data set with a clustered data structure, for testing the product limit method.

Usage

```
data(clusterTestData)
```

Format

A data frame with 42 observations on the following 4 variables.

midtimeX event time

eventX event status

patientid patient number

AnyCrownFracture a binary predictor

Examples

```
data(clusterTestData)
## maybe str(clusterTestData) ; plot(clusterTestData) ...
```

confInt	<i>Add point-wise confidence limits to the graphs of Kaplan-Meier and Aalen-Johannsen estimates of survival and cumulative incidence.</i>
---------	---

Description

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

Usage

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

Arguments

<code>x</code>	an object of class ‘ <code>prodim</code> ’ as returned by the <code>prodim</code> function.
<code>times</code>	where to compute point-wise confidence limits
<code>newdata</code>	see <code>plot.prodim</code>
<code>type</code>	Either “ <code>cuminc</code> ” or “ <code>survival</code> ” passed to <code>summary.prodim</code> as <code>surv=ifelse(type=="cuminc",FALSE,</code>
<code>cotype</code>	If “ <code>shadow</code> ” then confidence limits are drawn as colored shadows. Otherwise, dotted lines are used to show the upper and lower confidence limits.
<code>cause</code>	see <code>plot.prodim</code>
<code>col</code>	the colour of the lines.
<code>lty</code>	the line type of the lines.
<code>lwd</code>	the line thickness of the lines.
<code>density</code>	For <code>cotype="shadow"</code> , the density of the shade. Default is 55 percent.
<code>...</code>	Further arguments that are passed to the function <code>segments</code> if <code>type=="bars"</code> and to <code>lines</code> else.

Details

This function should not be called directly. The arguments can be specified as `Confint.arg` in the call to `plot.prodim`.

Value

Nil

Author(s)

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

See Also

[plot.prodim](#), [atRisk](#), [markTime](#)

`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")
getStates(object)
```

Arguments

<code>object</code>	Object of class "Hist".
<code>mode</code>	Return mode. One of "numeric", "character", or "factor".
<code>column</code>	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](#)

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)
```

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 `prodlm` 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

<code>time</code>	for right censored data a numeric vector of event times – for interval censored data a list or a <code>data.frame</code> providing two numeric vectors the left and right endpoints of the intervals. See <code>Details</code> .
<code>event</code>	A vector or a factor that specifies the events that occurred at the corresponding value of <code>time</code> . Numeric, character and logical values are recognized. It can also be a list or a <code>data.frame</code> for the longitudinal form of storing the data of a multi state model – see <code>Details</code> .
<code>entry</code>	Vector of delayed entry times (left-truncation) or list of two times when the entry time is interval censored.
<code>id</code>	Identifies the subjects to which multiple events belong for the longitudinal form of storing the data of a multi state model – see <code>Details</code> .
<code>cens.code</code>	A character or numeric vector to identify the right censored observations in the values of <code>event</code> . Defaults to "0" which is equivalent to 0.
<code>addInitialState</code>	If <code>TRUE</code> , an initial state is added to all <code>ids</code> 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

```

## 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

Hist(time=1:10,event=c(99,"event",99,99,99,"event",99,"event",99,99),cens.code=99)

TwoStateFrame <- data.frame(time=rlnorm(100),status=rbinom(100,1,.5))
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
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,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

```

`jackknife`*Compute jackknife pseudo values.*

Description

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

Usage

```
jackknife(object, times, keepResponse=FALSE, ...)  
jackknife.survival(object, times, keepResponse=FALSE, ...)  
jackknife.competing.risks(object, times, cause, keepResponse=FALSE, ...)  
leaveOneOut(object, times, ...)  
leaveOneOut.survival(object, times, lag=FALSE, ...)  
leaveOneOut.competing.risks(object, times, cause, ...)
```

Arguments

<code>object</code>	Object of class "prodlm".
<code>times</code>	Time points at which to compute pseudo values.
<code>cause</code>	For competing risks the cause of failure.
<code>keepResponse</code>	If TRUE add the model response, i.e. event time, event status, etc. to the result.
<code>lag</code>	Logical. If TRUE lag the result, i.e. compute $S(t-)$ instead of $S(t)$.
<code>...</code>	

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

Andersen PK & Perme MP (2010). Pseudo-observations in survival analysis *Statistical Methods in Medical Research*, 19(1), 71-99.

See Also

[prodlm](#)

Examples

```

## Not run:

## pseudo-values for survival models

d=SimSurv(100)
f=prodlim(Hist(time,status)~1,data=d)
jackknife(f,times=c(30,50))

## in some situations it may be useful to attach the
## the event time history to the result
jackknife(f,times=c(30,50),keepResponse=TRUE)

# pseudo-values for competing risk models
d=prodlim:::SimCompRisk(100)
f=prodlim(Hist(time,cause)~1,data=d)
jackknife(f,times=c(3,10),cause=1)
jackknife(f,times=c(3,10,17),cause=2)

# comparison to pseudoci
# make sure we get the same
# results with both packages

library(prodlim)
library(pseudo)

set.seed(17)
N <- 200
ddd <- data.frame(time=1:N,cause=rbinom(N,2,.5),X=rbinom(N,1,.5))
ttt <- c(3,5,10)
# ttt <- ddd$time
fff <- prodlim(Hist(time,cause)~1,data=ddd)
system.time(jack <- with(ddd,pseudoci(time,cause,ttt))[, -c(1:2,seq(4,N+2,2))])
system.time({jack2 <- jackknife.competing.risks(fff,times=ttt)})

## check individual 1
all(round(jack2[,1],9)==round(jack[,1],9))

## check all individuals
all(sapply(1:N,function(x){
a <- round(jack[x,],8)
b <- round(jack2[x,],8)
# all(a[!is.na(a)]==b[!is.na(b)])
all(a[!is.na(a)]==b[!is.na(a)])
}))

## the pseudoci function seems only slightly slower
## for small sample sizes (up to ca. 200) but
## much slower for large sample sizes:

set.seed(17)

```

```

N <- 200
ddd <- data.frame(time=1:N,cause=rbinom(N,2,.5),X=rbinom(N,1,.5))
ttt <- c(3,5,10)
# ttt <- ddd$time
fff <- prodlim(Hist(time,cause)~1,data=ddd)
system.time(jack <- with(ddd,pseudoci(time,cause,ttt)[-c(1:2,seq(4,N+2,2))])
system.time({jack2 <- jackknife.competing.risks(fff,times=ttt)})
all(round(jack2[,1],9)==round(jack[,1],9))

set.seed(17)
N <- 2000
ddd <- data.frame(time=1:N,cause=rbinom(N,2,.5),X=rbinom(N,1,.5))
ttt <- c(3,5,10)
# ttt <- ddd$time
fff <- prodlim(Hist(time,cause)~1,data=ddd)
system.time(jack <- with(ddd,pseudoci(time,cause,ttt)[-c(1:2,seq(4,N+2,2))])
system.time({jack2 <- jackknife.competing.risks(fff,times=ttt)})
all(round(jack2[,1],9)==round(jack[,1],9))

## End(Not run)

```

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

<code>x</code>	The values of the curves at times.
<code>times</code>	The times where there curves are plotted.
<code>nlost</code>	The number of subjects lost to follow-up (censored) at times.
<code>pch</code>	The symbol used to mark the curves.
<code>col</code>	The color of the symbols.
<code>...</code>	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(1,10,100,1000,1001,1001,1001,1002,1002,1002))
```

neighborhood	<i>Symmetric neighborhoods for kernel smoothing</i>
--------------	---

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 dpik function of the package 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

Stute, W. "Asymptotic Normality of Nearest Neighbor Regression Function Estimates", *The Annals of Statistics*, 1984,12,917–926.

See Also

[dpik](#), [prodlm](#)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
## Not run:
library(survival)
data(pbc)
neighborhood(pbc$age)

## End(Not run)
```

NN

Dummy function

Description

This is a special function used in the context of the conditional product limit estimator. It is similar to the function `strata` and can be used in a formula to force a stratified analysis in nearest neighborhoods defined by the values of a continuous predictor.

Usage

```
NN(x)
```

Arguments

x A continuous predictor

Value

The argument is passed on as it is: `NN(x)=x`

Author(s)

Thomas Gerds

See Also

[prodlm](#)

Examples

```
## Not run:
require(survival)
data(pbc)
prodlm(Hist(time,status)~NN(age),data=pbc)

## End(Not run)
```

PercentAxis	<i>Percentage-labeled axis.</i>
-------------	---------------------------------

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.

Author(s)

Thomas Alexander Gerds

See Also

[plot.prodlim](#)

Examples

```
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	<i>Box-arrow diagrams for multi-state models.</i>
-----------	---

Description

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

Usage

```
## 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), arrows, 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
## Not run:

## or simply create some data
SurvFrame <- data.frame(time=1:10,status=sample(0:1,10,TRUE))
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=F,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=F,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\n cause 1",box3.label="Dead\n cause 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","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","Disease\nfree","Diseased","Diseased","Disea
IDHist2 <- with(illness.death.frame2,Hist(time,event=list(from,to)))
plot(IDHist2)

## 4 state model
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)

## End(Not run)

```

plot.prodlim

Plotting event probabilities over time

Description

Function to plot survival and cumulative incidence curves against time.

Usage

```

## S3 method for class 'prodlim'
plot(x,
      type,

```

```

cause = 1,
newdata,
add = FALSE,
col,
lty,
lwd,
ylim,
xlim,
xlab = "Time",
ylab,
legend = TRUE,
marktime = FALSE,
confint = TRUE,
automar,
atrisk=ifelse(add,FALSE,TRUE),
timeOrigin,
axes=TRUE,
background=TRUE,
percent=TRUE,
minAtrisk=0,
...)
```

Arguments

x	an object of class 'prodlim' as returned by the prodlim function.
type	controls what part of the object is plotted. Defaults to "survival" for the Kaplan-Meier estimate of the survival function in two state models and to "incidence" for the Aalen-Johansen estimate of the cumulative incidence function in competing risk models
cause	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.
newdata	a data frame containing strata for which plotted curves are desired.
add	if 'TRUE' curves are added to an existing plot.
col	color for curves defaults to 1:number(curves)
lty	line type for curves defaults to 1
lwd	line width for all curves
ylim	limits of the y-axis
xlim	limits of the x-axis
ylab	label for the y-axis
xlab	label for the x-axis
legend	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.

marktime	if TRUE the curves are tick-marked at right censoring times by invoking the function <code>markTime</code> . Optional arguments of the function <code>markTime</code> can be given in the form <code>confint.x=val</code> as with <code>legend</code> . See also <code>Details</code> .
confint	if TRUE pointwise confidence intervals are plotted by invoking the function <code>confInt</code> . Optional arguments of the function <code>confInt</code> can be given in the form <code>confint.x=val</code> as with <code>legend</code> . See also <code>Details</code> .
automar	If TRUE the function tries to get good values for figure margins around the main plotting region.
atrisk	if TRUE display numbers of subjects at risk by invoking the function <code>atRisk</code> . Optional arguments of the function <code>atRisk</code> can be given in the form <code>atrisk.x=val</code> as with <code>legend</code> . See also <code>Details</code> .
timeOrigin	Start of the time axis
axes	If true axes are drawn. See <code>details</code> .
background	If TRUE the background color and grid color can be controlled using smart arguments <code>SmartControl</code> , such as <code>background.bg="yellow"</code> or <code>background.bg=c("gray66","gray88")</code> . The following defaults are passed to <code>background</code> by <code>plot.prodlim</code> : <code>horizontal=seq(0,1,.25)</code> , <code>vertical=NULL</code> , <code>bg="gray77"</code> , <code>fg="white"</code> . See <code>background</code> 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 <code>minAtrisk</code>
...	Parameters that are filtered by <code>SmartControl</code> and then passed to the functions <code>plot</code> , <code>legend</code> , <code>axis</code> , <code>atRisk</code> , <code>confInt</code> , <code>markTime</code> , <code>backGround</code>

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 `plot.prodlim` 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

```
## simulate right censored data from a two state model
dat <- data.frame(time=rexp(100),status=rbinom(100,1,.3),X=rbinom(100,1,.5),Z=rnorm(100,10,3),patnr=sample(1:10,100))
with(dat,plot(Hist(time,status)))

### marginal Kaplan-Meier estimator
kmfit <- prodlim(Hist(time, status) ~ 1, data = dat)
plot(kmfit)
plot(kmfit,percent=TRUE)
plot(kmfit,percent=TRUE,axis1.at=seq(0,kmfit$maxtime,365.25/2),axis1.labels=seq(0,kmfit$maxtime/365.25,1/2),xlab="Time (days)",yprob=TRUE)
plot(kmfit,confint.citype="shadow",col=4,background=TRUE,background.fg="yellow",background.horizontal=seq(0,1,0.1))
plot(kmfit,confint.citype="dots",col=4,background=TRUE,background.bg=c("white","gray88"),background.fg="gray77")

### Kaplan-Meier in discrete strata
kmfitX <- prodlim(Hist(time, status) ~ X, data = dat)
plot(kmfitX)
plot(kmfitX,legend.x="bottomleft",atRisk.cex=1.3)

### Kaplan-Meier in continuous strata
kmfitZ <- prodlim(Hist(time, status) ~ Z, data = dat)
plot(kmfitZ,newdata=data.frame(Z=c(5,7,12)))

### Cluster-correlated data
kmfitC <- prodlim(Hist(time, status) ~ cluster(patnr), data = dat)
plot(kmfitC,atrisk.labels=c("Units","Patients"))

## simulate right censored data from a competing risk model
datCR <- data.frame(time=rexp(100),status=rbinom(100,2,.3),X=rbinom(100,1,.5),Z=rnorm(100,10,3))
with(datCR,plot(Hist(time,status)))

### marginal Aalen-Johansen estimator
ajfit <- prodlim(Hist(time, status) ~ 1, data = datCR)
plot(ajfit)

### conditional Aalen-Johansen estimator
ajfitXZ <- prodlim(Hist(time, status) ~ X+Z, data = datCR)
plot(ajfitXZ,newdata=data.frame(X=c(1,1,0),Z=c(4,10,10)))
plot(ajfitXZ,newdata=data.frame(X=c(1,1,0),Z=c(4,10,10)),cause=2)
```

Description

Survival data are simulated with `SimSurv`. Then this function applies the LinksKaplan-Meier estimator – possibly in strata defined by covariates – and plots the result.

Usage

```
## S3 method for class 'SimSurv'
plot(x, ...)
```

Arguments

<code>x</code>	Data.frame as obtained with <code>SimSurv</code>
<code>...</code>	passed to <code>plot.prodlim</code>

Author(s)

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

See Also

[SimSurv](#), [plot.prodlim](#)

Examples

```
sdat=SimSurv(100)
plot(sdat)
```

`plotIllnessDeathModel` *Plotting an illness-death-model.*

Description

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

Usage

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

Arguments

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

Author(s)

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

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

```
## S3 method for class 'prodlim'
predict(object,
  times,
  newdata,
  level.chaos=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.chaos	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 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 names.strata : 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

<code>object</code>	A fitted object of class "prodlim".
<code>lag</code>	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	<i>Print objects in the prodlim library</i>
---------------	---

Description

Pretty printing of objects created with the functionality of the 'prodlim' library.

Usage

```
## S3 method for class 'prodlim'  
print(x, ...)  
## S3 method for class 'Hist'  
print(x, ...)  
## S3 method for class 'neighborhood'  
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	<i>product limit method</i>
---------	-----------------------------

Description

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,
        reverse=FALSE,
        conf.int=0.95,
        bandwidth=NULL,
        discrete.level=3,
        maxiter=1000,
        grid,
        tol=7,
        method=c("npml", "one.step", "impute.midpoint", "impute.right"),
        exact=TRUE)
```

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 <code>discrete.level</code> . 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 <code>formula</code> can be interpreted.
subset	Expression identifying a subset of the rows of the data for the fitting process.
na.action	A missing-data filter function, applied to the model.frame, after any subset argument has been used. Default is <code>options()\$na.action</code> .
reverse	For right censored data, if <code>reverse=TRUE</code> then the censoring distribution is estimated.
conf.int	The level (between 0 and 1) for two-sided pointwise confidence intervals. Defaults to 0.95.
bandwidth	Smoothing parameter for symmetric nearest neighborhoods based on the values of a continuous covariate. See function <code>neighborhood</code> for details.
discrete.level	Numeric covariates are treated as factors when their number of unique values exceeds <code>not discrete.level</code> . Otherwise the product limit method is applied, in overlapping neighborhoods according to the bandwidth.
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 <code>method=one.step</code> 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 "npml" (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.

Details

The response of formula (ie the left hand side of the '~' operator) specifies the model.

In two-state models – the classic survival case – the standard Kaplan-Meier method is applied. For this the response can be specified as a `Surv` or as a `Hist` object. 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 Kaplan-Meier estimator for at most one continuous predictors using symmetrical 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-Johannsen 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" for which there are `print`, `predict`, `life.table`, `summary` and `plot` methods.

Author(s)

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

References

- Andersen, Borgan, Gill, Keiding (1993) Springer 'Statistical Models Based on Counting Processes'
- Akritas (1994) The Annals of Statistics 22, 1299-1327 Nearest neighbor estimation of a bivariate distribution under random censoring.
- R Beran (1981) <http://anson.ucdavis.edu/~beran/paper.html> 'Nonparametric regression with randomly censored survival data'

Stute (1984) The Annals of Statistics 12, 917–926 ‘Asymptotic Normality of Nearest Neighbor Regression Function Estimates’

Ying, Wei (1994) Journal of Multivariate Analysis 50, 17-29 The Kaplan-Meier estimate for dependent failure time observations

See Also

[predictSurv](#), [predictSurvIndividual](#), [predictCuminc](#), [Hist](#), [neighborhood](#), [Surv](#), [survfit](#), [strata](#), [cluster](#), [NN](#)

Examples

```
##-----two-state survival model-----

dat <- SimSurv(100)
with(dat,plot(Hist(time,status)))
fit <- prodlim(Hist(time,status)~1,data=dat)
print(fit)
plot(fit)
summary(fit)

## -----clustered data-----

dat <- data.frame(time=rexp(100),status=rbinom(100,1,.3),X=rbinom(100,1,.5),Z=rnorm(100,10,3),patnr=sample(1:10,100))
fit <- prodlim(Hist(time,status)~cluster(patnr),data=dat)
print(fit)
plot(fit)
summary(fit)

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

dat <- SimSurv(100)
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)))

##-----censoring survival function-----

rpfit <- prodlim(Hist(time,status)~1,data=dat,reverse=TRUE)
rsfit <- survfit(Surv(time,1-status)~1,data=dat,conf.type="plain")
## not the same if there are ties!
all(round(rpfit$surv,3)==round(rsfit$surv,3))

##-----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(1,1,0,1,1,1,1,0,0,0))
with(dat,Hist(time=list(L,R),event=status))

dat$event=1
npml.e.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))
crFit <- prodlim(Hist(time,event)~X,data=CompRiskFrame)
summary(crFit)
plot(crFit)
summary(crFit,cause=2)
plot(crFit,cause=2)

```

quantile.prodlim *Quantiles for Kaplan-Meier and Aalen-Johannsen estimates.*

Description

Quantiles for Kaplan-Meier and Aalen-Johannsen estimates.

Usage

```
## S3 method for class 'prodlim'
quantile(x, q, ...)
```

Arguments

x	Object of class "prodlim".
q	Quantiles. Vector of values between 0 and 1.
...	not used

Author(s)

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

Examples

```
## Not run:  
d=SimSurv(100)  
f=prodlm(Hist(time,status)~X2,data=d)  
# median  
quantile(f,.5)  
  
## End(Not run)
```

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 argument 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

```
tab <- 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)
```

sampleCompriskFrame	<i>Dummy data set used for illustrating the use of some functions in the package.</i>
---------------------	---

Description

Dummy data set used for illustrating the use of some functions in the package.

Usage

```
data(sampleCompriskFrame)
```

Format

A data frame with 10 observations on the following 4 variables.

id a numeric vector
time a numeric vector
event a numeric vector
X a numeric vector

SimSurv	<i>Simulating survival data</i>
---------	---------------------------------

Description

Censored event times are drawn from user specified conditional distributions given simulated covariates.

Usage

```
SimSurv(N,  
surv,  
cens,  
cova,  
verbose=1,  
...)
```

Arguments

N	Sample size
surv	Dummy argument. The survival distribution is determined by arguments of the form <code>surv.arg</code> . See Details
cens	If FALSE data are left uncensored. Otherwise the censoring distribution is specified by using arguments of the form <code>cens.arg</code> . See Details.
cova	either a matrix with N rows, or a named list to generate the covariates. each entry is a list where the first element is the names of the function used to draw the covariate values and the remaining elements are arguments passed to that function. For example <code>cova=list(X = list(dist = "rlnorm", meanlog = 2, sdlog = 0.4))</code> generates a single log-normally distributed covariate called X.
verbose	Set to FALSE to shut up in simulations
...	used for convenient argument specification, e.g. <code>surv.transform.X2=function(x)x^2</code> will overwrite a corresponding entry of <code>surv</code> for the transform of the covariate X2

Details

`surv.dist` name of the function to draw the survival distribution `surv.args` list of extra arguments to `surv.dist`. `surv.baseline` the baseline risk `surv.link` names of the link to the covariates `surv.coef` numeric vector of regression coefficients. the `ith` is used for the `ith` entry of `cova` `surv.transform` list of function names one for each covariate. the `ith` is applied to values of the `ith` covariate before it is linked to generate the survival times. `cens.dist` name of the function to draw the censoring distribution `cens.args` list of extra arguments to `dist` `cens.baseline` the baseline risk `cens.link` names of the link to the covariates `cens.max` maximal value where all event times are right censored `cens.type` "right" for right censored data, "interval" for interval censored data `cens.coef` numeric vector of regression coefficients. the `ith` is used for the `ith` entry of `cova` `cens.transform` list of function names one for each covariate, where the `ith` element is applied to values of the `ith` covariate before it is linked to generate the survival times.

Possible distributions to generate covariates:

`X.lognorm = list(dist = "rlnorm", meanlog = 2, sdlog = 0.4)`

`X.unif = list(dist = "runif", min = 0, max = 10),`

`X.exp = list(dist = "rexp", rate = 0.4)`

`X.bernoulli = list(dist = "rbinom", size = 1, prob = 0.3)`

`X.binom = list(dist = "rbinom", size = 4, prob = 0.8)`

`X.nbinom = list(dist = "rnbino", size = 3, mu = 2)`

`X.poisson = list(dist = "rpois", lambda = 1.3)`

Value

A data.frame:

`time` the right censored event times

status	the survival status
X	the values of the covariate X
f.X	the transformed values of the covariate X
time	the uncensored event times

Attributes of the data.frame formula a formula to evaluate the generated event history object

Author(s)

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

References

Ralf Bender, Thomas Augustin, and Maria Blettner. Generating survival times to simulate Cox proportional hazards models by Ralf Bender, Thomas Augustin and Maria Blettner, *Statistics in Medicine* 2005; 24:1713-1723. *Stat Med*, 25(11):1978-9, 2006.

See Also

[prodlim](#)

Examples

```

SimSurv(10)

SurvData=SimSurv(100,cens.baseline=1/10,surv.baseline=2)

Hist(SurvData$time,SurvData$status)

prodlim(Hist(time,status)~1,data=SurvData)

plot(prodlim(Hist(time,status)~1,data=SurvData))

plot(SurvData,atrisk=FALSE,legend=FALSE)

SurvData=SimSurv(100,cens.baseline=1/10,surv.baseline=2,surv.coef=c(-1,-2),
  cova=list( X.exp = list(dist = "rexp", rate = 0.4),
  X.bernoulli = list(dist = "rbinom", size = 1, prob = 0.3)))

SurvData=SimSurv(100,cens.baseline=1/10,surv.baseline=2,surv.coef=c(-1,-2),cens.coef=c(0,1),
  cova=list( X.exp = list(dist = "rexp", rate = 0.4),
  X.bernoulli = list(dist = "rbinom", size = 1, prob = 0.3)))

set.seed(8)
SurvFrame <- SimSurv(N=100,
  surv=list(dist = "rweibull",args = list(shape=1),baseline = 1/100,link="exp",coef=1),
  cens=list(dist = "rexp",args=NULL,baseline=1/1000,link="exp",coef=0,max = 150,type = "interval",lateness=1,un),
  cova=list(Age = list("rnorm", mean = 0, sd = 2),Sex = list("rbinom", size=1,prob=.5)),
  keep.uncensored=TRUE,
  method=c("simulation"),

```

```

        verbose=1)

## Not run:

library(survival)
coxph(Surv(time,status==0)~X.exp+X.bernoulli,data=SurvData)

coxph(Surv(time,status)~X.exp+X.bernoulli,data=SurvData)

# Simulate survival times based on a parametric survival fit

library(survival)
data(ovarian)
fitSurv <- survreg(Surv(futime,fustat)~age+rx,data=ovarian)
fitCens <- survreg(Surv(futime,1-fustat)~age+rx,data=ovarian)
# note: survreg fits the accelerated failure time model version of the
#       Weibull survival regression model.
#       The parameters need to be transformed in order to
#       simulate from a fitted model using SimSurv:
survCoef <- -fitSurv$coef[-1]/fitSurv$scale
survBaseline <- exp(-fitSurv$coef["(Intercept)"]/fitSurv$scale)
survShape <- 1/fitSurv$scale

censCoef <- -fitCens$coef[-1]/fitCens$scale
censBaseline <- exp(-fitCens$coef["(Intercept)"]/fitCens$scale)
censShape <- 1/fitCens$scale
X <- model.frame(~age+rx,data=ovarian)

simuData <- SimSurv(N=26,
  cova=X,
  surv.coef=survCoef,
  surv.baseline=survBaseline,
  surv.shape=survShape,
  cens.model="Cox-Weibull",
  cens.coef=censCoef,
  cens.baseline=censBaseline,
  cens.shape=censShape)

simuFit <- survreg(Surv(time,status)~age+rx,data=simuData)
cbind(coef(simuFit),coef(fit))

plot(prodlim(Surv(time,status)~rx,data=simuData),confint=F)
plot(prodlim(Surv(futime,fustat)~rx,data=ovarian),confint=F,add=TRUE,lty=2)

## another example

surv.coef=c(0.1, 0.03, -0.5, 0.5)
cens.coef=c(0.1, 0.03,-0.5, 0.5)
variables=list(
  x1=list(dist="rnorm", mean=26, sd=2),
  x2=list(dist="rnorm", mean=50, sd=7),

```

```

      x3=list(dist="rbinom", size=1, prob=0.2),
      x4=list(dist="rbinom", size=1, prob=0.5))
surv.base<-0.00001
cens.base<-0.00009
sim.dat<-SimSurv(N=500,
  cova=variables,
  surv.coef=surv.coef,
  cens.coef=cens.coef,
  surv.model="Cox-exponential",
  cens.model="Cox-exponential",
  surv.baseline=surv.base,
  cens.baseline=cens.base,
  verbose=F)

```

```
## End(Not run)
```

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
strict	If TRUE make the comparison of jump times and eval times strict
comp	If "greater" count the number of jump times that are greater (greater or equal when strict==FALSE) than the eval times

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 be 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

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

forced	A named list of forced arguments for the subroutines.
split	Regular expression used for splitting keys from arguments. Default is "\. ".
ignore.case	If TRUE then all matching and splitting is not case sensitive.
replaceDefaults	If TRUE default arguments are replaced by given arguments. Can also be a named list with entries for each subroutine.
verbose	If TRUE warning messages are given for arguments in call that are not ignored via argument ignore and that do not match any key.

Author(s)

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

See Also

[plot.prodlim](#)

Examples

```
dumPlot = function(...){
  ## set defaults
  plot.DefaultArgs=list(x=0,y=0,type="n")
  lines.DefaultArgs=list(x=1:10,lwd=3)
  ## apply smartcontrol
  x=SmartControl(call=list(...),
                 defaults=list("plot"=plot.DefaultArgs, "lines"=lines.DefaultArgs),
                 ignore.case=TRUE,keys=c("plot","axis2","lines"),
                 forced=list("plot"=list(axes=FALSE),"axis2"=list(side=2)))
  ## call subroutines
  do.call("plot",x$plot)
  do.call("lines",x$lines)
  do.call("axis",x$axis2)
}
dumPlot(plot.ylim=c(0,5),plot.xlim=c(0,20),lines.lty=3,axis2.At=c(0,3,4))
```

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.prodlim	<i>Summary method for prodlim objects.</i>
-----------------	--

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

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

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\$model.matrix.
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 FALSE 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	Passed to lifetable
...	Further arguments that are passed to the print function.

Details

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

Value

A data.frame with the relevant information.

Author(s)

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

See Also

[prodlim](#), [summary.Hist](#)

Examples

```
SurvFrame <- data.frame(time=1:10,status=rbinom(10,1,.5))
x <- prodlim(formula=Hist(time=time,status!=0)~1,data=SurvFrame)
summary(x)
summary(x,times=c(1,5,10,12),percent=TRUE,intervals=TRUE,digits=3)
```

Index

- *Topic **Graphics**
 - SmartControl, 38
- *Topic **cluster**
 - prodlim, 27
- *Topic **datasets**
 - clusterTestData, 4
 - sampleCompriskFrame, 33
- *Topic **misc**
 - row.match, 32
 - sindex, 37
- *Topic **nonparametric**
 - NN, 15
 - prodlim, 27
- *Topic **smooth**
 - neighborhood, 14
- *Topic **survival**
 - atRisk, 2
 - backGround, 3
 - confInt, 5
 - getEvent, 6
 - Hist, 7
 - jackknife, 10
 - markTime, 12
 - meanNeighbors, 13
 - PercentAxis, 16
 - plot.Hist, 16
 - plot.prodlim, 19
 - plot.SimSurv, 22
 - plotIllnessDeathModel, 23
 - predict.prodlim, 24
 - predictSurvIndividual, 26
 - print.prodlim, 27
 - prodlim, 27
 - quantile.prodlim, 31
 - SimSurv, 33
 - summary.Hist, 39
 - summary.prodlim, 40
- atRisk, 2, 5, 13, 21, 22
- attributes, 8
- axis, 21, 22
- backGround, 3, 21, 22
- cluster, 29, 30
- clusterTestData, 4
- confInt, 3, 5, 13, 21, 22
- dpik, 14
- getEvent, 6
- getStates (getEvent), 6
- Hist, 6, 7, 18, 29, 30, 40
- jackknife, 10
- leaveOneOut (jackknife), 10
- legend, 21, 22
- lines.prodlim (plot.prodlim), 19
- markTime, 3, 5, 12, 21, 22
- meanNeighbors, 13
- neighborhood, 13, 14, 22, 30
- NN, 15, 30
- PercentAxis, 16
- plot, 21, 22
- plot.Hist, 8, 16, 22, 40
- plot.prodlim, 3, 5, 13, 16, 19, 23, 39
- plot.SimSurv, 22
- plot.survfit, 21
- plotIllnessDeathModel, 23
- predict.prodlim, 24, 26, 27
- predictCuminc, 30
- predictCuminc (predict.prodlim), 24
- predictSurv, 26, 30
- predictSurv (predict.prodlim), 24
- predictSurvIndividual, 25, 26, 30
- print.Hist (print.prodlim), 27

`print.neighborhood (print.prodlim)`, 27
`print.prodlim`, 27
`prodlim`, 7, 8, 10, 14, 15, 22, 27, 35, 41
`quantile.prodlim`, 31
`row.match`, 32
`sampleCompriskFrame`, 33
`SimSurv`, 23, 33
`sindex`, 37
`SmartControl`, 18, 21, 38
`strata`, 30
`summary.Hist`, 8, 39, 41
`summary.prodlim`, 22, 27, 40
`Surv`, 7, 29, 30
`survfit`, 30