

Package ‘prodlim’

April 19, 2009

Title Product Limit Estimation

Version 1.0.5

Author Thomas A. Gerds

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

Depends R (>= 1.9.1), KernSmooth, survival

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

License GPL-2

Repository CRAN

Date/Publication 2008-07-21 14:41:37

R topics documented:

Hist	2
neighborhood	4
NN	5
plot.Hist	6
plot.prodlim	11
plot.SimSurv	13
predict.prodlim	14
predictSurvIndividual	16
print.prodlim	17
prodlim	17
row.match	21
SimSurv	22
sindex	25
summary.Hist	26
summary.prodlim	27
Index	29

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, id = NULL, cens.code = "0")
```

Arguments

<code>time</code>	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 <i>Details</i> .
<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 data.frame for the longitudinal form of storing the data of a multi state model – see <i>Details</i> .
<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 <i>Details</i> .
<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.

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:

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

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
MultiStateFrame2 <- data.frame(time=rep(1,10),
                              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,id))

```

neighborhood

Symmetric 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

<code>x</code>	Numeric vector – typically the observations of a continuous random variate.
<code>bandwidth</code>	Controls the distance between neighbors in a neighborhood. It can be a decimal, i.e. the bandwidth, or the string "smooth", in which case the fourth root of the sample size is used, or <code>NULL</code> in which case the <code>dpik</code> function of the package <code>KernSmooth</code> is used to find the optimal bandwidth.
<code>kernel</code>	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 is described by the first neighbor (`first.nbh`) of the usually very long vector `neighbors` and the size of the neighborhood (`size.nbh`). Further, the arguments `bandwidth`, `kernel`, the total sample size `n` and the number of unique values `nu` is included.

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](#), [prodlim](#)

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

[prodlim](#)

Examples

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

plot.Hist

Box-arrow diagrams for multi-state models

Description

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

Usage

```
## S3 method for class 'Hist':
plot(x,
      layout,
      xbox.rule=.3,
      ybox.rule=1.5,
      state.lab,
      state.cex=2,
      rect.args,
      args.state.lab,
      arrow.lab,
      arrow.lab.offset,
```

```

arrow.lab.side=NULL,
arrow.lab.cex=2,
arrow.lab.style,
arrows.args,
arrow.lab.args=NULL,
arrow.head.offset=3,
arrow.double.dist=1,
arrow.fix.code=NULL,
enumerate.bboxes=FALSE,
box.numbers,
cex.bboxlabs=1.28,
margin,
verbose=FALSE,
reverse,
...)
```

Arguments

x	An object of class Hist.
layout	A list saying how to arrange the boxes. The list should contain the number of graphic columns (<code>ncol</code>) the number of graphic rows (<code>nrow</code>) the box positions (<code>box.pos</code>). See the examples below.
xbox.rule	Numeric value specifying the distance between the text and the box in x-direction.
ybox.rule	Numeric value specifying the distance between the text and the box in y-direction.
state.lab	Vector of text that appears in the boxes. Defaults to <code>attr(x,"state.names")</code> .
state.cex	Cex for the <code>state.lab</code>
rect.args	List of arguments that are passed to the call of the function <code>rect</code> .
args.state.lab	List of arguments that are passed to the call of the function <code>text</code> for labeling the inside of the boxes.
arrow.lab	Vector of labels for the arrows.
arrow.lab.offset	List of vectors of length two: the horizontal and the vertical offset of the arrow label from the arrow mid. Length should be equal to the number of arrows. Can be a single numeric value which is then used repeatedly. If missing the <code>strwidth</code> and <code>strheight</code> of the current label is used.
arrow.lab.side	Vector of integers one for each arrow label. Can either be one or minus one. Minus one indicates that the label should appear on the other side of the arrow.
arrow.lab.cex	Cex for arrow labels.
arrow.lab.style	Choice between "count", "symbolic", "character", "no". "count": the arrows are labeled using the number of transitions "symbolic": the expression $\alpha_{[hj]}(t)$ is used to label the arrow between state h and state j

	"character": expects labels from argument <code>arrow.lab</code>
	"no": omit arrow labels
<code>arrows.args</code>	A list with further arguments that are passed to the function <code>arrows</code> .
<code>arrow.lab.args</code>	A list with further arguments that are passed to the function <code>text</code> for labeling the arrows.
<code>arrow.head.offset</code>	Distance from the arrow heads to the boxes.
<code>arrow.double.dist</code>	Distance between arrows that go in both directions.
<code>arrow.fix.code</code>	Vector of integers. If for some reason the direction of one or several arrows is wrong the <code>nth</code> value of <code>arrow.fix.code</code> will be used for the <code>nth</code> arrow. See argument <code>code</code> of <code>arrows</code> .
<code>enumerate.boxes</code>	Logical. If TRUE the boxes are numbered in the upper left corner.
<code>box.numbers</code>	Vector of strings that appear in the upper left corner of the boxes, if <code>enumerate.boxes==TRUE</code> . Defaults to <code>0, ..., number.of.boxes</code> .
<code>cex.boxlabs</code>	Cex for the numbers in the upper left corners.
<code>margin</code>	Set the figure margin via <code>par(mar=margin)</code>
<code>verbose</code>	If TRUE echo various things.
<code>reverse</code>	If true the order of the boxes is reversed. For competing risk models only the order of the absorbing states.
<code>...</code>	for compatibility with other plot functions

Details

The default layout works only for two-state, competing risk and illness-death models. All the `cex` parameters and `xy.rules` depend on the device.

Value

The first argument but in the invisible cloak.

Note

A cool thing would be to use the functionality of the unix program 'dot' <http://www.graphviz.org/About.php> to obtain more complex graphs.

Author(s)

Thomas Gerds (tag@biostat.ku.dk)

See Also

[Hist](#)

Examples

```

## A simple survival model
## Not run:
library(survival)
data(pbc)
SurvFrame <- pbc
## 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,
      state.lab=c("Randomization", "Death"),
      arrow.lab=expression(lambda(t)),
      layout=list(nrow=2, ncol=2, box.pos=list(c(1, 1), c(2, 2))))

## 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,
      cex=2,
      state.lab=c("Alive", "Dead\n cause 1", "Dead\n cause 2"),
      style="character", ybox.rule=1,
      arrow.lab=c(expression(gamma[1](t), gamma[2](t))),
      enumerate.bboxes=TRUE,
      cex.bboxlabs = 1.28)

## change the layout
plot(CRHist,
      cex=2,
      state.lab=c("Alive", "Dead\n cause 1", "Dead\n cause 2"),
      style="character", ybox.rule=1,
      arrow.lab=c(expression(gamma[1](t), gamma[2](t))),
      arrow.lab.side=c(1, -1),
      enumerate.bboxes=TRUE,
      cex.bboxlabs = 1.28,
      layout=list(nrow=2, ncol=3, box.pos=list(c(1, 2), c(2, 1), c(2, 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,
      cex=2,
      state.lab=c("Alive", "Dead\n cause 1", "Dead\n cause 2", "Dead\n cause 3"),
      arrow.lab=c(expression(alpha[1](t), alpha[2](t), alpha[3](t))),
      enumerate.bboxes=TRUE,
      ybox.rule=.8,
      verbose=FALSE,
      cex.bboxlabs = 1.28)

## change the layout
plot(CRHist2,
      cex=2,

```

```

state.lab=c("Alive","Dead\n cause 1","Dead\n cause 2","Dead\n cause 3"),
arrow.lab=c(expression(alpha[1](t),alpha[2](t),alpha[3](t))),
enumerate.bboxes=TRUE,
ybox.rule=.8,
verbose=FALSE,
cex.boxlabs = 1.28,
layout=list(nrow=3,ncol=3,box.pos=list(c(2,1),c(1,2),c(3,2),c(2,3))),
reverse=FALSE,
arrow.lab.side=c(-1,1,-1),
arrow.lab.offset=c(7,9,8))

## illness-death models
illness.death.frame <-
  data.frame(time=1:4,
             from=c("Disease-free","Disease-free",
                   "Diseased","Disease-free"),
             to=c("0","Diseased","Dead","Dead"))
IDHist <- with(illness.death.frame,
              Hist(time,event=list(from,to)))
plot(IDHist,
     ybox.rule=4,
     xbox.rule=.3,
     state.cex=1.3,
     enum=TRUE,
     arrow.lab.side=c(-1,-1,1))

## illness-death with recovery
illness.death.frame2 <- data.frame(time=1:5,
                                   from=c("Disease\nfree","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,
     ybox.rule=1.3,
     xbox.rule=.3,
     state.cex=2,
     arrow.lab.offset=c(13,13,8,10),
     enum=TRUE,
     verbose=FALSE)
## change the layout

plot(IDHist2,
     ybox.rule=1.3,
     xbox.rule=.3,
     state.cex=2,
     enum=TRUE,
     verbose=FALSE,
     layout=list(ncol=3,nrow=2,box.pos=list(c(1,1),c(2,2),c(1,3))),
     arrow.lab.side=c(-1,1,-1,1),
     arrow.lab.offset=c(15,15,10,10))
## 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, what, cause = 1, newdata, add = FALSE, col, lty, lwd,
      ylim, xlim, xlab = "Time", ylab, legend = TRUE,
      legend.args= NULL, mark.time = FALSE, cex.mark.time = 1.5,
      conf.int = TRUE, conf.int.args = list(col = "gray"),
      atrisk, atrisk.args, timeOrigin,...)
```

Arguments

x	an object of class 'prodlim' as returned by the <code>prodlim</code> function.
what	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. When there are strata defined by <code>newdata</code> only one cause is possible.
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 be plotted automatically
legend.args	list of arguments that is passed to the function <code>legend</code>
mark.time	if TRUE the curves are tick-marked at right censoring times
cex.mark.time	cex for the tick-marks if <code>mark.time</code> is set TRUE
conf.int	if TRUE pointwise confidence intervals are plotted

<code>conf.int.args</code>	list with extra arguments like <code>col</code> , <code>lty</code> , etc. used for the pointwise confidence intervals. Extra argument <code>time</code> determines time points at which confidence intervals are plotted.
<code>atrisk</code>	if TRUE display numbers of subjects at risk. if data are cluster correlated also the number of clusters with at least one subject at risk is given.
<code>atrisk.args</code>	list of extra arguments for plotting of the numbers at risk. Recognized extra arguments are: <code>times</code> : at which time points numbers are plotted. <code>labels</code> vector of labels, e.g. "No. Patients".
<code>timeOrigin</code>	Start of the time axis
<code>...</code>	graphical parameters that are passed to function <code>plot</code>

Details

See examples.

Value

NULL

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

`prodlim`, `plot.Hist`, `summary.prodlim`, `neighborhood`

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))
with(dat, plot(Hist(time, status)))

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

### Kaplan-Meier in discrete strata
kmfitX <- prodlim(Hist(time, status) ~ X, data = dat)
plot(kmfitX)

### 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.args=list(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, 1))
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)

```

plot.SimSurv

Plot simulated survival data

Description

Survival data are simulated with `SimSurv`. Then this function applies the Kaplan-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)

```

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 <code>times</code> carried forward
predictors	The relevant part of the argument <code>newdata</code> .
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 <code>newdata</code> . Together time and strata show where to find the predicted probabilities.
dimensions	a list with the following components:
time	The length of <code>times</code>
strata	The number of rows in <code>newdata</code>
names.strata	Labels for the covariate values.

Author(s)

Thomas Gerds <tag@biostat.ku.dk>

See Also

[predictSurvIndividual](#)

Examples

```
## Not run:
library(survival)
data(pbc)
fit <- prodlim(Hist(time,status)~1,data=pbc)

## 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 (ie overlapping strata)
mfit <- prodlim(Hist(time,status)~edema+age,data=pbc)
predict(mfit,times=c(-1,0,10,100,1000,10000),newdata=pbc[18:21,])
```

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

## and even transposed
predict(mfit,times=c(-1,0,10,100,1000,10000),newdata=pbcc[18:21,],mode="matrix",bytime=TRUE)
## End(Not run)
```

```
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,
        ml=FALSE)
```

Arguments

<code>formula</code>	A formula whose left hand side is a <code>Hist</code> object. In some special cases it can also be a <code>Surv</code> 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.
<code>data</code>	A <code>data.frame</code> in which all the variables of <code>formula</code> can be interpreted.
<code>subset</code>	Expression identifying a subset of the rows of the data for the fitting process.
<code>na.action</code>	A missing-data filter function, applied to the <code>model.frame</code> , after any subset argument has been used. Default is <code>options()\$na.action</code> .
<code>reverse</code>	For right censored data, if <code>reverse=TRUE</code> then the censoring distribution is estimated.
<code>conf.int</code>	The level (between 0 and 1) for two-sided pointwise confidence intervals. Defaults to 0.95.
<code>bandwidth</code>	Smoothing parameter for symmetric nearest neighborhoods based on the values of a continuous covariate. See function <code>neighborhood</code> for details.
<code>discrete.level</code>	Numeric covariates are treated as factors when their number of unique values exceeds not <code>discrete.level</code> . Otherwise the product limit method is applied, in overlapping neighborhoods according to the bandwidth.
<code>maxiter</code>	Maximal number of iterations to obtain the nonparametric maximum likelihood estimate. Defaults to 1000.
<code>grid</code>	When <code>ml=FALSE</code> grid for one-step product limit estimate. Defaults to sorted list of unique left and right endpoints of the observed intervals.
<code>tol</code>	Numeric value whose negative exponential is used as convergence criterion for finding the nonparametric maximum likelihood estimate. Defaults to 7 meaning $\exp(-7)$.
<code>ml</code>	If <code>TRUE</code> (not the default!) use the usual Turnbull algorithm, else the product limit version of the self-consistent estimate.

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 <- SimSurv(100, cova=list(patnr=list("rbinom", 30, .5)), surv.baseline=1/1000000)
fit <- prodlim(Hist(time, status)~cluster(patnr), data=dat)
print(fit)
plot(fit)
summary(fit)

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

pfit <- prodlim(Surv(time, status)~1, data=abc)
pfit <- prodlim(Hist(time, status)~1, data=abc) ## same thing
sfit <- survfit(Surv(time, status)~1, data=abc, 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(abc$time)))

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

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

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

pfit.edema <- prodlim(Surv(time, status)~edema, data=abc)
summary(pfit.edema)
summary(pfit.edema, intervals=TRUE)
plot(pfit.edema)

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

prodlim(Surv(time, status)~age, data=abc)
```

```
##-----both discrete and continuous covariates-----
prodlim(Surv(time, status)~edema+age, data=pbcc)

##-----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))

nqml.fitm1 <- prodlim(Hist(time=list(L, R), event=1)~1, data=dat, ml=TRUE)

##-----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)
```

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)
```

SimSurv

Simulating survival data

Description

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

Usage

```
SimSurv(N,
        surv = list(dist="rweibull",
                    args=list(shape=1),
                    baseline=1,
                    link="exp",
                    coef=1,
                    transform=NULL,
                    method="simulation"),
        cens = list(dist="rexp",
                    args=NULL,
                    baseline=1/100,
                    link="exp",
                    max=NULL,
                    type="right",
                    coef=0,
                    transform=NULL,
                    method="ttransform"),
        cova = list(X1=list("rnorm", mean=0, sd=2),
                    X2=list("rbinom", size=1, prob=.5)),
        verbose=1,
        ...)
```

Arguments

N	Sample size
surv	A list with the following arguments

dist	name of the function to draw the survival distribution
args	list of extra arguments to 'dist'
baseline	the baseline risk
link	names of the link to the covariates
coef	numeric vector of regression coefficients. the ith is used for the ith entry of <code>cova</code>
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. the second
cens	A list with the following elements
dist	name of the function to draw the censoring distribution
args	list of extra arguments to 'dist'
baseline	the baseline risk
link	names of the link to the covariates
max	maximal value where all event times are right censored
type	"right" for right censored data, "interval" for interval censored data
coef	numeric vector of regression coefficients. the ith is used for the ith entry of <code>cova</code>
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.
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

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 = "rnbinom", 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
formula	the uncensored event times

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)))

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

```

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

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

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

summary.Hist	<i>Summary of event histories</i>
--------------	-----------------------------------

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 *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

```
## S3 method for class 'prodlim':
summary(object,
        times,
        newdata,
        verbose = TRUE,
        max.tables = 20,
        incidence = FALSE,
        cause=1,
        intervals = FALSE,
        proz = FALSE,
        ...)
```

Arguments

object	An object with class 'prodlim' derived with <code>prodlim</code>
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 <code>object\$model.matrix</code> .
verbose	If FALSE suppress all warnings and the printing of the summary.
max.tables	Integer. If <code>newdata</code> is not given the value of <code>max.tables</code> decides about the maximal number of tables to be shown. Defaults to 20.
incidence	Logical. If TRUE report event probabilities instead of survival probabilities. Only when <code>object\$model=="survival"</code> .
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 <code>times</code> .
proz	Logical. If TRUE all estimated values are multiplied by 100 and thus interpretable on a percent scale.
...	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), proz=TRUE, intervals=TRUE, digits=3)
```

Index

- *Topic **cluster**
 - prodlim, 17
- *Topic **misc**
 - row.match, 21
 - sindex, 25
- *Topic **nonparametric**
 - NN, 5
 - prodlim, 17
- *Topic **smooth**
 - neighborhood, 4
- *Topic **survival**
 - Hist, 1
 - plot.Hist, 6
 - plot.prodlim, 10
 - plot.SimSurv, 13
 - predict.prodlim, 13
 - predictSurvIndividual, 16
 - print.prodlim, 17
 - prodlim, 17
 - SimSurv, 22
 - summary.Hist, 26
 - summary.prodlim, 27

arrows, 7
attributes, 3

cluster, 19, 20

dpik, 4, 5

Hist, 1, 8, 19, 20, 26

neighborhood, 4, 12, 20
NN, 5, 20

plot.Hist, 3, 6, 12, 26
plot.prodlim, 10, 13
plot.SimSurv, 13
plot.survfit, 12
predict.prodlim, 13, 16, 17
predictCuminc, 20

predictCuminc (*predict.prodlim*),
13

predictSurv, 16, 20
predictSurv (*predict.prodlim*), 13
predictSurvIndividual, 15, 16, 20
print.Hist (*print.prodlim*), 17
print.neighborhood
(*print.prodlim*), 17
print.prodlim, 17
prodlim, 1, 3, 5, 6, 12, 17, 24, 27, 28

rect, 7
row.match, 21

SimSurv, 13, 22
sindex, 25
strata, 20
summary.Hist, 3, 26, 28
summary.prodlim, 12, 17, 27
Surv, 1, 19, 20
survfit, 20

text, 7