The eha Package

July 19, 2006

Version 0.96-4

Date 2006-07-13

Title Event History Analysis.

Description A package for survival and event history analysis.

License GPL version 2 or newer.

Author GÃűran BrostrÃűm

Depends R (>= 2.3.1), survival, graphics

Maintainer GÃűran BrostrÃűm <gb@stat.umu.se>

R topics documented:

`age.window` *Age cut of survival data*

Description

For a given age interval, each spell is cut to fit into the given age interval.

Usage

```
age.window(dat, window, surv=c("enter", "exit", "event"))
```

Arguments

`dat`	Input data frame. Must contain survival data.
`window`	Vector of length two; the age interval.
`surv`	Vector of length three giving the names of the central variables in 'dat'.

Details

The `window` must be in the order `(begin, end)`

Value

A data frame of the same form as the input data frame, but 'cut' as desired. Intervals exceeding `window[2]` will be given `event = 0`

Author(s)

GÃűran BrostrÃűm

See Also

`cal.window`, `coxreg`, `mlreg`

Examples

```
dat <- data.frame(enter = 0, exit = 5.731, event = 1, x = 2)
window <- c(2, 5.3)
dat.trim <- age.window(dat, window)
```

`cal.window` *Calendar time cut of survival data*

Description

For a given time interval, each spell is cut so that it fully lies in the given time interval

Usage

```
cal.window(dat, window, surv=c("enter", "exit", "event", "birthdate"))
```

Arguments

`dat`	Input data frame. Must contain survival data and a birth date.
`window`	Vector of length two; the time interval
`surv`	Vector of length four giving the names of the central variables in 'dat'.

Details

The `window` must be in the order `(begin, end)`

Value

A data frame of the same form as the input data frame, but 'cut' as desired. Intervals exceeding `window[2]` will be given `event = 0`

Author(s)

Göran Broström

See Also

`age.window`, `coxreg`, `mlreg`

Examples

```
dat <- data.frame(enter = 0, exit = 5.731, event = 1,
birthdate = 1962.505, x = 2)
window <- c(1963, 1965)
dat.trim <- cal.window(dat, window)
```

`check.surv` *Check the integrity of survival data.*

Description

Check that exit occurs after enter, that spells from an individual do not overlap, and that each individual experiences at most one event.

Usage

```
check.surv(enter, exit, event, id = NULL, eps = 1e-08)
```

Arguments

`enter`	Left truncation time.
`exit`	Time of exit.
`event`	Indicator of event. Zero means 'no event'.
`id`	Identification of individuals.
`eps`	The smallest allowed spell length or overlap.

Details

Interval lengths must be strictly positive.

Value

A vector of id's for the insane individuals. Of zero length if no errors.

Note

Author(s)

Göran Broström

References

See Also

`join.spells`, `coxreg`, `mlreg`

Examples

```
xx <- data.frame(enter = c(0, 1), exit = c(1.5, 3), event = c(0, 1), id =
c(1,1))
check.surv(xx$enter, xx$exit, xx$event, xx$id)
```

`coxreg` *Cox regression*

Description

Performs Cox regression with some special attractions, especially *sampling of risksets* and *the weird bootstrap*.

Usage

```
coxreg(formula = formula(data), data = parent.frame(),
na.action = getOption("na.action"), init, method = c("efron", "breslow"),
control = list(eps = 1e-08, maxiter = 10, trace = FALSE),
singular.ok = TRUE, model = FALSE,
x = FALSE, y = TRUE, boot = FALSE, rs, max.survs)
```

Arguments

`formula`	a formula object, with the response on the left of a operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
`data`	a data.frame in which to interpret the variables named in the formula.
`na.action`	a missing-data filter function, applied to the model.frame, after any subset argument has been used. Default is `options()$na.action`.
`init`	vector of initial values of the iteration. Default initial value is zero for all variables.
`method`	Method of treating ties, "efron" (default) or "breslow".
`control`	a list with components `eps` (convergence criterion), `maxiter` (maximum number of iterations), and `silent` (logical, controlling amount of output). You can change any component without mention the other(s).
`singular.ok`	Not used
`model`	Not used
`x`	Return the design matrix in the model object?
`y`	return the response in the model object?
`rs`	Risk set?
`boot`	Number of boot replicates. Defaults to FALSE, no boot samples.
`max.survs`	Sampling of risk sets? If given, it should (the upper limit on) the number of survivors in each risk set.

Details

The default method, `efron`, and the alternative, `breslow`, are both the same as in `coxph` in package `survival`.

Value

A list of class `c("coxreg", "coxph")` with components

`coefficients`	Fitted parameter estimates.
`var`	Covariance matrix of the estimates.
`loglik`	Vector of length two; first component is the value at the initial parameter values, the second componet is the maximized value.
`score`	The score test statistic (at the initial value).
`linear.predictors`	The estimated linear predictors.
`residuals`	The martingale residuals.
`hazard`	The estimated baseline hazard.
`means`	Means of the columns of the design matrix.
`w.means`	Weighted (against exposure time) means of covariates; weighted relative frequencies of levels of factors.
`n`	Number of spells in indata (possibly after removal of cases with NA's).
`events`	Number of events in data.
`terms`	Used by extractor functions.
`assign`	Used by extractor functions.
`wald.test`	The Walt test statistic (at the initial value).
`y`	The Surv vector.
`isF`	Logical vector indicating the covariates that are factors.
`covars`	The covariates.
`ttr`	Total Time at Risk.
`levels`	List of levels of factors.
`formula`	The calling formula.
`bootstrap`	The (matrix of) bootstrap replicates, if requested on input. It is up to the user to do whatever desirable with this sample.
`boot.sd`	The estimated standard errors of the bootstrap replicates.
`call`	The call.
`method`	The method.
`convergence`	Did the optimization converge?
`fail`	Did the optimization fail? (Is `NULL` if not).

Warning

The use of `rs` is dangerous, see note. It can however speed up computing time considerably for huge data sets.

Note

This function starts by creating risksets, if no riskset is supplied via `rs`, with the aid of `risksets`. Supplying output from `risksets` via `rs` fails if there are any NA's in the data! Note also that it depends on stratification, so `rs` contains information about stratification. Giving another strata variable in the formula is an error. The same is ok, for instance to supply stratum interactions.

Author(s)

Göran Broström

References

See Also

coxph, risksets

Examples

```
dat <- data.frame(time=  c(4, 3,1,1,2,2,3),
                  status=c(1,1,1,0,1,1,0),
                  x=     c(0, 2,1,1,1,0,0),
                  sex=   c(0, 0,0,0,1,1,1))
coxreg( Surv(time, status) ~ x + strata(sex), data = dat) #stratified model
# Same as:
rs <- risksets(Surv(dat$time, dat$status), strata = dat$sex)
coxreg( Surv(time, status) ~ x, data = dat, rs = rs) #stratified model
```

---

coxreg.fit *Cox regression*

Description

Called by coxreg, but a user can call it directly.

Usage

```
coxreg.fit(X, Y, rs, strats, offset, init, max.survs,
method = "breslow", boot = FALSE, control)
```

Arguments

X	The design matrix.
Y	The survival object.
rs	The risk set composition. If absent, calculated.
strats	The stratum variable. Can be absent.
offset	Offset. Can be absent.
init	Start values. If absent, equal to zero.
max.survs	Sampling of risk sets? If so, gives the maximum number of survivors in each risk set.
method	Either "efron" (default) or "breslow".
boot	Number of bootstrap replicates. Defaults to FALSE, no bootstrapping.
control	See coxreg

Details

`rs` is dangerous to use when NA's are present.

Value

A list with components

`coefficients`	Estimated regression parameters.
`var`	Covariance matrix of estimated coefficients.
`loglik`	First component is value at `init`, second at maximum.
`score`	Score test statistic, at initial value.
`linear.predictors`	Linear predictors.
`residuals`	Martingale residuals.
`hazard`	Estimated baseline hazard. At value zero of 'design' variables.
`means`	Means of the columns of the design matrix.
`bootstrap`	The bootstrap replicates, if requested on input.
`conver`	`TRUE` if convergence.
`fail`	`TRUE` if failure.
`iter`	Number of performed iterations.

Note

It is the user's responsibility to check that indata is sane.

Author(s)

Göran Broström

See Also

`coxreg`, `risksets`

Examples

```
X <- as.matrix(data.frame(
               x=     c(0, 2,1,4,1,0,3),
               sex=   c(1, 0,0,0,1,1,1)))
time <- c(1,2,3,4,5,6,7)
status <- c(1,1,1,0,1,1,0)
stratum <- rep(1, length(time))

coxreg.fit(X, Surv(time, status), strats = stratum, max.survs = 6,
    control = list(eps=1.e-4, maxiter = 10, trace = FALSE))
```

`cro` *Creates a minimal representation of a data frame.*

Description

Given a data frame with a defined response variable, this function creates a unique representation of the covariates in the data frame, vector (matrix) of responses, and a pointer vector, connecting the responses with the corresponding covariates.

Usage

```
cro(dat, response=1)
```

Arguments

`dat`	A data frame
`response`	The column(s) where the response resides.

Details

The rows in the data frame are converted to text strings with `paste` and compared with `match`.

Value

A list with components

`y`	The response.
`covar`	A data frame with unique rows of covariates.
`keys`	Pointers from `y` to `covar`, connecting each response with its covariate vector.

Note

This function is based on suggestions by Anne York and Brian Ripley.

Author(s)

Göran Broström

See Also

`match`, `paste`

Examples

```
dat <- data.frame(y = c(1.1, 2.3, 0.7), x1 = c(1, 0, 1), x2 = c(0, 1, 0))
cro(dat)
```

`Hweibull` *Cumulative hazard function for the Weibull distribution*

Description

Calculates the cumulative hazard function of a Weibull distribution

Usage

```
Hweibull(x, shape, scale = 1, log = FALSE)
```

Arguments

`x`	vector of quantiles.
`shape`	shape parameter.
`scale`	scale parameter, defaulting to 1.
`log`	logical: if TRUE, the answer is given as log(H).

Details

See pweibull

Value

Gives the cumulative hazard function at x.

Author(s)

GÃűran brostrÃűm

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(x, shape, scale = 1, log = FALSE){
    res <- (x / scale)^shape
    if (log) res <- logb(res, base = exp(1))
    res
  }
```

`frail.fit` *Fits a frailty proportional hazards model*

Description

This function is called from `mlreg`. Can be called directly by a user who knows what (s)he is doing!

Usage

```
frail.fit(X, Y, rs, strats, offset, init, max.survs, frailty, control)
```

Arguments

`X`	Design matrix
`Y`	Survival object
`rs`	Risk set pointers
`strats`	Stratum variable
`offset`	Offset variable
`init`	Initial regression parameter values
`max.survs`	Sampling of risk sets?
`frailty`	The frailty grouping variable
`control`	Values for the optimization

Details

Frailty is analyzed in a marginal framework, with numerical integration. This is very slow and somewhat unstable.

Value

See the code!

Warning

Not very much tested. Can fail, and is very slow.

Author(s)

Göran Broström

See Also

`mlreg`

Examples

```
fam.size <- rep(20, 5)
n <- sum(fam.size)
fam <- rep(1:length(fam.size), fam.size)
beta <- 1
sigma <- 1
frail <- rep(rnorm(length(fam.size), 0, sigma), fam.size)
x <- runif(n)
exit <- rexp(n, exp(x * beta + frail))
event <- rep(1, n)
enter <- numeric(n)
mlreg(Surv(enter, exit, event) ~ x, frailty = frail)
```

---

geome.fit *Constant intensity discrete time proportional hazards*

Description

This function is called from `mlreg`. A user may call it directly.

Usage

```
geome.fit(X, Y, rs, strats, offset, init, max.survs, method = "ML",
boot = FALSE, control)
```

Arguments

`X`	The design matrix
`Y`	Survival object
`rs`	risk set produced by `risksets`
`strats`	Stratum indicator
`offset`	Offset
`init`	Initial values
`max.survs`	Maximal survivors
`method`	"ML" or "MPPL"
`boot`	should we bootstrap?
`control`	See mlreg

Value

See the code.

Note

Nothing special

Author(s)

Göran Broström

References

See mlreg

See Also

mlreg

Examples

`hweibull` *The hazard function of a Weibull distribution*

Description

Calculates the hazard function of a Weibull distribution.

Usage

```
hweibull(x, shape, scale = 1, log = FALSE)
```

Arguments

`x`	vector of quantiles
`shape`	The shape parameter
`scale`	The scale parameter, defaults to 1.
`log`	logical; if TRUE, the log of the hazard function is given.

Details

See dweibull.

Value

The hazard function, evaluated at x.

Author(s)

GÃűran BostrÃűm

See Also

pweibull

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(x, shape, scale = 1, log = FALSE){
    if (shape <= 0 || scale <= 0)
      error("scale and shape must be positive")
    (x / scale)^(shape - 1) / scale
  }
```

---

join.spells *Straighten up a survival data frame*

Description

Unnecessary cut spells are glued together, overlapping spells are "polished", etc.

Usage

```
join.spells(dat, eps = 1.e-8)
```

Arguments

dat	A data frame with names enter, exit, event, id.
eps	Tolerance for equality of two event times.

Details

In case of overlapping intervals (i.e., a data error), the appropriate id's are returned.

Value

A data frame with the same variables as the input, but individual spells are joined, if possible (identical covariate values, and adjacent time intervals).

Author(s)

Göran Broström

References

Therneau, T.M. and Grambsch, P.M. (2000). *Modeling Survival Data: Extending the Cox model.* Springer.

See Also

coxreg, mlreg, check.surv

`make.communal` *Put communals in "fixed" data frame*

Description

Given an ordinary data frame suitable for survival analysis, and a data frame with "communal" time series, this function includes the communal covariates as fixed, by the "cutting spells" method.

Usage

```
make.communal(dat, com.dat, communal = TRUE, start, period = 1, lag = 0,
surv=c("enter", "exit", "event", "birthdate"), tol=1e-04, fortran=TRUE)
```

Arguments

`dat`	A data frame containing interval specified survival data and covariates, of which one must give a "birth date", the connection between duration and calendat time
`com.dat`	Data frame with communal covariates. They must have the same start year and periodicity, given by `com.ins`
`communal`	Boolean; if TRUE, then it is a true communal (default), otherwise a fixed. The first component is the first year (start date in decimal form), and the second component is the period length. The third is `lag` and the fourth is `scale`.
`start`	Start date in decimal form.
`period`	Period length. Defaults to one.
`lag`	The lag of the effect. Defaults to zero.
`surv`	Character vector of length 4 giving the names of interval start, interval end, event indicator, birth date, in that order. These names must correspond to names in `dat`
`tol`	Largest length of an interval considered to be of zero length. The cutting sometimes produces zero length intervals, which we want to discard.
`fortran`	If `TRUE`, then a Fortran implementation of the function is used. This is the default. This possibility is only for debugging purposes. You should of course get identical results with the two methods.

Details

The main purpose of this function is to prepare a data file for use with `coxreg`, `mlreg`, `weibreg` and `coxph`.

Value

The return value is a data frame with the same variables as in the combination of `dat` and `com.dat`. Therefore it is an error to have common name(s) in the two data frames.

Note

Not very vigorously tested.

Author(s)

Göran Broström

See Also

coxreg, mlreg, coxph, cal.window

Examples

```
dat <- data.frame(enter = 0, exit = 5.731, event = 1,
birthdate = 1962.505, x = 2)
## Birth date: July 2, 1962 (approximately).
com.dat <- data.frame(price = c(12, 3, -5, 6, -8, -9, 1, 7))
dat.com <- make.communal(dat, com.dat, start = 1962.000)
```

---

mlreg *ML proportional hazards regression*

Description

Maximum Likelihood estimation of proportional hazards models.

Usage

```
mlreg(formula = formula(data), data = parent.frame(),
na.action = getOption("na.action"), init, method = c("ML", "MPPL"),
control = list(eps = 1e-08, maxiter = 10, n.points = 12, trace = FALSE),
singular.ok = TRUE, model = FALSE,
x = FALSE, y = TRUE, boot = FALSE, geometric = FALSE,
rs, frailty = NULL, max.survs)
```

Arguments

formula	a formula object, with the response on the left of a operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
data	a data.frame in which to interpret the variables named in the formula.
na.action	a missing-data filter function, applied to the model.frame, after any subset argument has been used. Default is options()$na.action.
init	vector of initial values of the iteration. Default initial value is zero for all variables.
method	Method of treating ties, "ML", the default, means pure maximum likelihood, i.e, data are treated as discrete. The choice "MPPL" implies that risk sets with no tied events are treated as in ordinary Cox regression. This is a cameleont that adapts to data, part discrete and part continuous.
control	a list with components eps (convergence criterion), maxiter (maximum number of iterations), and silent (logical, controlling amount of output). You can change any component without mention the other(s).
singular.ok	Not used.

model	Not used.
x	Return the design matrix in the model object?
y	return the response in the model object?
boot	No. of bootstrap replicates. Defaults to FALSE, i.e., no bootstrapping.
geometric	If TRUE, the intensity is assumed constant within strata.
rs	Risk set? If present, speeds up calculations considerably.
frailty	A grouping variable for frailty analysis. Full name is needed.
max.survs	Sampling of risk sets?

Details

Method ML performs a true discrete analysis, i.e., one parameter per observed event time. Method MPPL is a compromize between the discrete and continuous time approaches; one parameter per observed event time with multiple events. With no ties in data, an ordinary Cox regression (as with coxreg) is performed.

Value

A list of class c("mlreg", "coxreg", "coxph") with components

coefficients	Fitted parameter estimates.
var	Covariance matrix of the estimates.
loglik	Vector of length two; first component is the value at the initial parameter values, the second componet is the maximized value.
score	The score test statistic (at the initial value).
linear.predictors	The estimated linear predictors.
residuals	The martingale residuals.
hazard	The estimated baseline hazard.
means	Means of the columns of the design matrix.
w.means	Weighted (against exposure time) means of covariates; weighted relative frequencies of levels of factors.
n	Number of spells in indata (possibly after removal of cases with NA's).
events	Number of events in data.
terms	Used by extractor functions.
assign	Used by extractor functions.
wald.test	The Walt test statistic (at the initial value).
y	The Surv vector.
isF	Logical vector indicating the covariates that are factors.
covars	The covariates.
ttr	Total Time at Risk.
levels	List of levels of factors.
formula	The calling formula.
call	The call.
bootstrap	The bootstrap sample, if requested on input.

sigma	Present if a frailty model is fitted. Equals the estimated frailty standard deviation.
sigma.sd	The standard error of the estimated frailty standard deviation.
method	The method.
convergence	Did the optimization converge?
fail	Did the optimization fail? (Is NULL if not).

Warning

The use of rs is dangerous, see note above. It can however speed up computing time.

Note

This function starts by creating risksets, if no riskset is supplied via rs, with the aid of risksets. This latter mechanism fails if there are any NA's in the data! Note also that it depends on stratification, so rs contains information about stratification. Giving another strata variable in the formula is an error. The same is ok, for instance to supply stratum interactions.

Author(s)

Göran Broström

References

Broström, G. (2002). Cox regression; Ties without tears. *Communications in Statistics: Theory and Methods* **31**, 285–297.

See Also

coxph, risksets

Examples

```
dat <- data.frame(time=  c(4, 3,1,1,2,2,3),
                  status=c(1,1,1,0,1,1,0),
                  x=     c(0, 2,1,1,1,0,0),
                  sex=   c(0, 0,0,0,1,1,1))
mlreg( Surv(time, status) ~ x + strata(sex), data = dat) #stratified model
# Same as:
rs <- risksets(Surv(dat$time, dat$status), strata = dat$sex)
mlreg( Surv(time, status) ~ x, data = dat, rs = rs) #stratified model
```

`mlreg.fit` *ML proportional hazards regression*

Description

Called by `mlreg`, but a user can call it directly.

Usage

```
mlreg.fit(X, Y, rs, strats, offset, init, max.survs,
method = "ML", boot = FALSE, control)
```

Arguments

`X`	The design matrix.
`Y`	The survival object.
`rs`	The risk set composition. If absent, calculated.
`strats`	The stratum variable. Can be absent.
`offset`	Offset. Can be absent.
`init`	Start values. If absent, equal to zero.
`max.survs`	Sampling of risk sets? If so, gives the maximum number of survivors in each risk set.
`method`	Either "ML" (default) or "MPPL".
`boot`	No. of bootstrap replicates. Defaults to FALSE, i.e., no bootstrapping.
`control`	See `coxreg`

Details

See `mlreg` for details.

Value

A list with components

`coefficients`	Estimated regression parameters.
`var`	Covariance matrix of estimated coefficients.
`loglik`	First component is value at `init`, second at maximum.
`score`	Score test statistic, at initial value.
`linear.predictors`	Linear predictors.
`residuals`	Martingale residuals.
`hazard`	Estimated baseline hazard. At value zero of 'design' variables.
`means`	Means of the columns of the design matrix.
`bootstrap`	The bootstrap sample, if requested on input.
`conver`	`TRUE` if convergence.
`fail`	`TRUE` if failure.
`iter`	Number of performed iterations.

Note

`rs` is dangerous to use when NA's are present. It is the user's responsibility to ensure that indata is sane.

Author(s)

Göran Broström

References

put references to the literature/web site here

See Also

`coxreg`, `risksets`

Examples

```
X <- as.matrix(data.frame(
                x=     c(0, 2,1,4,1,0,3),
                sex=   c(1, 0,0,0,1,1,1)))
time <- c(1,2,3,4,5,6,7)
status <- c(1,1,1,0,1,1,0)
stratum <- rep(1, length(time))

mlreg.fit(X, Surv(time, status), strats = stratum, max.survs = 6,
    control = list(eps=1.e-4, maxiter = 10, trace = TRUE))
```

`perstat` *Period statistics*

Description

Calculates occurrence / exposure rates for time periods given by `period` and for ages given by `age`.

Usage

```
perstat(surv, period, age = c(0, 200))
```

Arguments

`surv`	An (extended) `surv` object (4 columns with `enter`, `exit`, `event`, `birthdate`)
`period`	A vector of dates (in decimal form)
`age`	A vector of length 2; lowest and highest age

Details

Value

A list with components

`events`	No. of events in eavh time period.
`exposure`	Exposure times in each period.
`intensity`	`events / exposure`

Author(s)

Göran Broström

See Also

`piecewise`

`piecewise` *Piecewise hazards*

Description

Calculate piecewise hazards, no. of events, and exposure times in each interval indicated by cutpoints.

Usage

```
piecewise(enter, exit, event, cutpoints)
```

Arguments

`enter`	Left interval endpoint
`exit`	Right interval endpoint
`event`	Indicator of event
`cutpoints`	Vector of cutpoints

Details

Exact calculation.

Value

A list with components

`deaths`	Vector of number of deaths
`exposure`	Vector of total exposure time
`hazard`	Vector of hazards, `hazard` == `deaths` / `exposure`

Author(s)

Göran Broström

See Also

`perstat`

`plot.Surv` *Plots of survivor functions.*

Description

Kaplan-Meier estimates. If only one curve, confidence limits according to Greenwood's formula are drawn.

Usage

```
plot.Surv(x, strata=NULL, fn = c("cum", "surv", "log", "loglog"),
limits=TRUE, conf=0.95, main=NULL, xlab=NULL, ylab=NULL,
xlim=NULL, ylim=NULL, lty.con=NULL, col.con = NULL, x.axis = TRUE, ...)
```

Arguments

`x`	A `Surv` object.
`strata`	Defines a partition of the data. One survivor function for each level of `strata` is drawn.
`fn`	Which type of plot?
`limits`	If TRUE, and if the number of curves is one, confidence limits are drawn.
`conf`	The confidence level for the confidence limits.
`main`	A heading for the plot.
`xlab`	Label on the x axis.
`ylab`	Label on the y-axis.
`xlim`	Horizontal plot limits. If NULL, calculated by the function.
`ylim`	Vertical plot limits. If NULL, set to `c(0, 1)`
`lty.con`	Line type of confidence bands.
`col.con`	Color of confidence bands.
`x.axis`	Should `abline(h=0)` be drawn?
`...`	Anything that `plot` likes...

Details

Left truncation is allowed. Note, though, that this fact may result in strange estimated curves due to lack of data in certain (low) ages.

Value

No value is returned.

Author(s)

Göran Broström

Examples

```
time0 <- numeric(50)
group <- c(rep(0, 25), rep(1, 25))
time1 <- rexp( 50, exp(group) )
event <- rep(1, 50)
plot.Surv(Surv(time0, time1, event), strata = group)
```

---

plot.weibreg *Plots output from a Weibull regression*

Description

Just a simple plot of the hazard functions for each stratum.

Usage

```
plot.weibreg(x, what = c("haz", "cum", "den", "sur"), main = NULL,
xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,
new.data = x$means, ...)
```

Arguments

x	A weibreg object
what	Which functions shoud be plotted! Default is all. They will scroll by, so you have to take care explicitely what you want to be produced. See, eg, par(mfrow = ...)
main	Header for the plot
xlim	x limits
ylim	y limits
xlab	x label
ylab	y label
new.data	At which covariate values?
...	Extra parameters passed to 'plot'

Details

The plot is drawn at the mean values of the covariates.

Value

No return value

Author(s)

Göran Broström

See Also

weibreg

Examples

```
y <- rweibull(4, shape = 1, scale = 1)
x <- c(1,1,2,2)
fit <- weibreg(Surv(y, c(1,1,1,1)) ~ x)
plot(fit)
```

---

print.coxreg *Prints coxreg objects*

Description

More "pretty-printing" than `print.coxph`, which is a fall-back for 'difficult' objects.

Usage

```
print.coxreg(x, digits = max(options()$digits - 4, 3), ...)
```

Arguments

`x`	A `coxreg` object, typically the result of running `coxreg`
`digits`	Output format.
`...`	Other arguments.

Details

Doesn't work with three-way and higher interactions, in which case `print.coxph` is used. Prints also output from `mlreg`.

Value

No value is returned.

Author(s)

Göran Broström

See Also

`coxreg`, `print.coxph`

print.weibreg *Prints weibreg objects*

Description

Usage

```
print.weibreg(x, digits = max(options()$digits - 4, 3), ...)
```

Arguments

x	A weibreg object
digits	Precision in printing
...	Not used.

Value

No value is returned.

Note

Doesn't work for threeway or higher order interactions. Use print.coxph in that case.

Author(s)

Göran Broström

See Also

weibreg, print.coxph

risksets *Finds the compositions and sizes of risk sets*

Description

Focus is on the risk set composition just prior to a failure.

Usage

```
risksets(x, strata, max.survs)
```

Arguments

x	A Surv object.
strata	Stratum indicator.
max.survs	Maximum number of survivors in each risk set. If smaller than the 'natural number', survivors are sampled from the present ones.

Details

If the input argument max.survs is left alone, all survivors are accounted for in all risk sets.

Value

A list with components

`antrs`	No. of risk sets in each stratum. The number of strata is given by `length(antrs)`.
`risktimes`	Ordered distinct failure time points.
`eventset`	vector of pointers to events in each risk set.
`riskset`	vector of pointers to the members of the risk sets, in order. The 'events' first are the events.
`size`	The sizes of the risk sets.
`n.events`	The number of events in each risk set.

Note

can be used to "sample the risk sets".

Author(s)

Göran Broström

See Also

`table.events`, `coxreg`, `mlreg`

Examples

```
enter <- c(0, 1, 0, 0)
exit <- c(1, 2, 3, 4)
event <- c(1, 1, 1, 0)
risksets(Surv(enter, exit, event))
```

`summary.coxreg` *Prints coxreg objects*

Description

This is the same as `print.coxreg`

Usage

```
summary.coxreg(object, ...)
```

Arguments

`object`	A `coxreg` object
`...`	Additional ...

Author(s)

Göran Broström

See Also

print.coxreg

Examples

```
## The function is currently defined as
function (object, ...)
print(object)
```

---

summary.weibreg *Prints a weibreg object*

Description

This is the same as print.weibreg

Usage

```
summary.weibreg(object, ...)
```

Arguments

object	A weibreg object
...	Additional ...

Author(s)

Göran Broström

See Also

print.weibreg

Examples

```
## The function is currently defined as
function (object, ...)
print(object)
```

`table.events` *Calculating failure times, risk set sizes and No. of events in each risk set*

Description

From input data of the 'interval' type, with an event indicator, summary statistics for each risk set (at an event time point) are calculated.

Usage

```
table.events(enter=rep(0, length(exit)), exit, event, strict=TRUE )
```

Arguments

`enter`	Left truncation time point.
`exit`	End time point, an event or a right censoring.
`event`	Event indicator.
`strict`	If TRUE, then tabulating is not done after a time point where all individuals in a riskset failed.

Value

A list with components

`times`	Ordered distinct event time points.
`events`	Number of events at each event time point.
`riskset.sizes`	Number at risk at each event time point.

Author(s)

Göran Broström

See Also

`risksets`

Examples

```
exit = c(1,2,3,4,5)
event = c(1,1,0,1,1)
table.events(exit = exit, event = event)
```

`toBinary` *Transforms a "survival" data frame into a data frame suitable for binary (logistic) regression*

Description

The result of the transformation can be used to do survival analysis via logistic regression. If the `cloglog` link is used, this corresponds to a discrete time analogue to Cox's proportional hazards model.

Usage

```
toBinary(dat, surv = c("enter", "exit", "event"),
strats, max.survs = NROW(dat))
```

Arguments

`dat`	A data frame with three variables representing the survival response. The default is that they are named `enter`, `exit`, and `event`
`surv`	A character string with the names of the three variables representing survival.
`strats`	An eventual stratification variable.
`max.survs`	Maximal numger of survivors per risk set. If set to a (small) number, survivors are sampled from the risk sets.

Details

toBinary calls `risksets` in the `eha` package.

Value

Returns a data frame expanded risk set by risk set. The three "survival variables" are replaced by a variable named `event` (which overwrites an eventual variable by that name in the input). Two more variables are created, `riskset` and `orig.row`.

`event`	Indicates an event in the corresponding risk set.
`riskset`	Factor (with levels 1, 2, ...) indicating risk set.
`orig.row`	The row number for this item in the original data frame.

Note

The survival variables must be three. If you only have *exit* and *event*, create a third containing all zeros.

Author(s)

Göran Broström

References

put references to the literature/web site here

See Also

mlreg

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(dat,
                     surv = c("enter", "exit", "event"),
                     strats,
                     max.survs = NROW(dat))
{
    if (!is.data.frame(dat))
      stop("dat must be a data frame")
    if (length(surv) != 3)
      stop("surv must have length 3")
    fixed.names <- names(dat)
    surv.indices <- match(surv, fixed.names)
    if (length(which(is.na(surv.indices)))) {
        x <- which(is.na(surv.indices))
        stop(paste(surv[x], " is not a name in the data frame."))
    }
    enter <- dat[, surv.indices[1]]
    exit <- dat[, surv.indices[2]]
    event <- dat[, surv.indices[3]]

    covars <- dat[, -surv.indices, drop = FALSE]

    nn <- NROW(dat)

    if (missing(strats) || is.null(strats)) strats <- rep(1, nn)
    rs <- risksets(Surv(enter, exit, event), strata = strats, max.survs)

    weg <- (abs(rs$size - rs$n.events) > 0.01)
    rs$riskset <- rs$riskset[rep(weg, rs$size)]
    rs$eventset <- rs$eventset[rep(weg, rs$n.events)]
    rs$n.events <- rs$n.events[weg]
    rs$size <- rs$size[weg]

    n.rs <- length(rs$size)
    ev <- numeric(sum(rs$size))
    start <- 1
    for (i in 1:n.rs) {
        ev[start:(start + rs$n.events[i] - 1)] <- 1
        start <- start + rs$size[i]
    }
    rs$ev <- ev

    out <- data.frame(event = rs$ev,
                      riskset = factor(rep(1:length(rs$size), rs$size))
                      )

    out <- cbind(out, covars[rs$riskset, , drop = FALSE])
```

```
    out$orig.row <- (1:nn)[rs$riskset]
    out
  }
```

`toDate` *Convert time in years since "0000-01-01" to a date.*

Description

This function uses `as.Date` and a simple linear transformation.

Usage

```
toDate(times)
```

Arguments

`times`	a vector of durations

Details

Value

A vector of dates as character strings of the type "1897-05-21".

Author(s)

Göran Broström

See Also

`toTime`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.
toDate(1897.357)
```

`toTime` *Calculate duration in years from "0000-01-01" to a given date*

Description

Given a vector of dates, the output is a vector of durations in years since "0000-01-01".

Usage

```
toTime(dates)
```

Arguments

`dates`	A vector of dates in character form or of class `Date`

Details

Value

A vector of durations, as decribed above.

Author(s)

Göran Broström

See Also

`toDate`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
toTime(c("1897-05-16", "1901-11-21"))
```

`weibreg` *Weibull regression*

Description

Proportional hazards model with baseline hazard(s) from the Weibull family of distributions. Allows for stratification with different scale and shape in each stratum, and left truncated and right censored data.

Usage

```
weibreg(formula = formula(data), data = parent.frame(),
na.action = getOption("na.action"), init, shape = 0,
control = list(eps = 1e-04, maxiter = 10, trace = FALSE),
singular.ok = TRUE, model = FALSE, x = FALSE, y = TRUE, center = TRUE)
```

Arguments

`formula`	a formula object, with the response on the left of a operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
`data`	a data.frame in which to interpret the variables named in the formula.
`na.action`	a missing-data filter function, applied to the model.frame, after any subset argument has been used. Default is `options()$na.action`.
`init`	vector of initial values of the iteration. Default initial value is zero for all variables.
`shape`	If positive, the shape parameter is fixed at that value (in each stratum). If zero or negative, the shape parameter is estimated. If more than one stratum is present in data, each stratum gets its own estimate.
`control`	a list with components `eps` (convergence criterion), `maxiter` (maximum number of iterations), and `silent` (logical, controlling amount of output). You can change any component without mention the other(s).
`singular.ok`	Not used.
`model`	Not used.
`x`	Return the design matrix in the model object?
`y`	Return the response in the model object?
`center`	Should covariates be centered? This affects the estimate of scale only, but will often stabilize the numerical calculations.

Details

The parameterization is the same as in `coxreg`, `mlreg`, and `coxph`, but different from the one used by `survreg`. The model is

$$h(t; a, b, \beta, z) = (a/b)(t/b)^{a-1} \exp(z\beta)$$

This is in correspondence with `Weibull`. To compare regression coefficients with those from `survreg` you need to divide by estimated shape ($\hat{a}$) and change sign. The p-values and test statistics are however the same, with one exception; the score test is done at maximized scale and shape in `weibreg`.

This model is a Weibull distribution with shape parameter a and scale parameter $b\Gamma(1+1/a)\exp(-z\beta/a)$

Value

A list of class `c("weibreg", "coxreg", "coxph")` with components

`coefficients`	Fitted parameter estimates.
`var`	Covariance matrix of the estimates.
`loglik`	Vector of length two; first component is the value at the initial parameter values, the second componet is the maximized value.
`score`	The score test statistic (at the initial value).
`linear.predictors`	The estimated linear predictors.
`means`	Means of the columns of the design matrix.
`w.means`	Weighted (against exposure time) means of covariates; weighted relative frequencies of levels of factors.
`n`	Number of spells in indata (possibly after removal of cases with NA's).
`events`	Number of events in data.
`terms`	Used by extractor functions.
`assign`	Used by extractor functions.
`y`	The Surv vector.
`isF`	Logical vector indicating the covariates that are factors.
`covars`	The covariates.
`ttr`	Total Time at Risk.
`levels`	List of levels of factors.
`formula`	The calling formula.
`call`	The call.
`method`	The method.
`convergence`	Did the optimization converge?
`fail`	Did the optimization fail? (Is `NULL` if not).
`pfixed`	TRUE if shape was fixed in the estimation.

Warning

The print method `print.weibreg` doesn't work if threeway or higher order interactions are present. Use `print.coxph` in that case.

Note further that covariates are internally centered, if `center = TRUE`, by this function, and this is not corrected for in the output. This affects the estimate of `log(scale)`, but nothing else. If you don't like this, set `center = FALSE`.

Author(s)

Göran Broström

See Also

`coxreg`, `mlreg`, `print.weibreg`

Examples

```
dat <- data.frame(time = c(4, 3, 1, 1, 2, 2, 3),
                  status = c(1, 1, 1, 0, 1, 1, 0),
                  x = c(0, 2, 1, 1, 1, 0, 0),
                  sex = c(0, 0, 0, 0, 1, 1, 1))
weibreg( Surv(time, status) ~ x + strata(sex), data = dat) #stratified model
```

---

weibreg.fit *Weibull regression*

Description

This function is called by weibreg, but it can also be directly called by a user.

Usage

```
weibreg.fit(X, Y, strata, offset, init, shape, control, center = TRUE)
```

Arguments

X	The design (covariate) matrix.
Y	A survival object, the response.
strata	A stratum variable.
offset	Offset.
init	Initial regression parameter values.
shape	If positive, a fixed value of the shape parameter in the Weibull distribution. Otherwise, the shape is estimated.
control	Controls convergence and output.
center	Should covariates be centered?

Details

See weibreg for more detail.

Value

coefficients	Estimated regression coefficients plus estimated scale and shape coefficients, sorted by strata, if present.
var	
loglik	Vector of length 2. The first component is the maximized loglihood with only scale and shape in the model, the second the final maximum.
score	Score test statistic at initial values
linear.predictors	Linear predictors for each interval.
means	Means of the covariates
conver	TRUE if convergence
fail	TRUE if failure
iter	Number of Newton-Raphson iterates.
n.strata	The number of strata in the data.

Author(s)

Göran Broström

See Also

weibreg

`wfunk` *Loglihood function of a Weibull regression*

Description

Calculates minus the log likelihood function and its first and second order derivatives for data from a Weibull regression model. Is called by weibreg.

Usage

```
wfunk(beta = NULL, lambda, p, X = NULL, Y, offset = rep(0, length(Y)),
ord = 2, pfixed = FALSE, trace = TRUE)
```

Arguments

beta	Regression parameters
lambda	The scale paramater
p	The shape parameter
X	The design (covariate) matrix.
Y	The response, a survival object.
offset	Offset.
ord	ord = 0 means only loglihood, 1 means score vector as well, 2 loglihood, score and hessian.
pfixed	Logical, if TRUE the shape parameter is regarded as a known constant in the calculations.
trace	If TRUE, more verbose output.

Details

Note that the function returns log likelihood, score vector and minus hessian, i.e. the observed information. The model is

$$h(t; p, \lambda, \beta, z) = p/\lambda(t/\lambda)^{(p-1)} \exp\left(-(t/\lambda)^p\right) \exp(z\beta)$$

This is in correspondence with dweibull.

Value

A list with components

f	The log likelihood. Present if ord >= 0
fp	The score vector. Present if ord >= 1
fpp	The negative of the hessian. Present if ord >= 2

Author(s)

Göran Broström

See Also

`weibreg`

Index