

Package ‘eha’

October 26, 2009

Encoding latin1

Version 1.2-13

Date 2009-10-26

Title Event History Analysis

Description A package for survival and event history analysis

License GPL (>= 3)

Author Göran Broström

Depends R (>= 2.2.0), survival, graphics

Maintainer Göran Broström <gb@stat.umu.se>

Repository CRAN

Date/Publication 2009-10-26 21:50:25

R topics documented:

aftreg	2
aftreg.fit	5
age.window	6
cal.window	7
check.dist	8
check.surv	9
coxreg	10
coxreg.fit	13
cro	15
EV	16
geome.fit	17
Gompertz	18
hweibull	19
join.spells	20
Loglogistic	21

Lognormal	22
make.communal	22
Makeham	24
male.mortality	25
mlreg	26
perstat	28
phfunc	29
phreg	30
phreg.fit	33
piecewise	34
plot.aftreg	35
plot.coxreg	36
plot.hazdata	37
plot.phreg	38
plot.Surv	39
plot.weibreg	40
print.aftreg	41
print.coxreg	42
print.phreg	43
print.weibreg	43
risksets	44
summary.coxreg	45
summary.weibreg	46
table.events	47
toBinary	48
toDate	49
toTime	50
weibreg	51
weibreg.fit	53
wfunk	54
Index	56

aftreg	<i>Accelerated Failure Time Regression</i>
--------	--

Description

The accelerated failure time model with parametric baseline hazard(s). Allows for stratification with different scale and shape in each stratum, and left truncated and right censored data.

Usage

```
aftreg(formula = formula(data), data = parent.frame(),
na.action = getOption("na.action"), dist = "weibull", init, shape = 0,
id, control = list(eps = 1e-08, maxiter = 20, trace = FALSE),
singular.ok = TRUE, model = FALSE, x = FALSE, y = TRUE)
```

Arguments

<code>formula</code>	a formula object, with the response on the left of a <code>~</code> operator, and the terms on the right. The response must be a survival object as returned by the <code>Surv</code> function.
<code>data</code>	a <code>data.frame</code> in which to interpret the variables named in the formula.
<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>dist</code>	Which distributin? Default is "weibull", with the alternatives "gompertz", "ev", "loglogistic" and "lognormal". A special case like the <code>exponential</code> can be obtained by choosing "weibull" in combination with <code>shape = 1</code> .
<code>init</code>	vector of initial values of the iteration. Default initial value is zero for all variables.
<code>shape</code>	If positive, the shape parameter is fixed at that value. If zero or negative, the shape parameter is estimated. Stratification is not meaningful if shape is fixed.
<code>id</code>	If there are more than one spell per individual, it is essential to keep spells together by the <code>id</code> argument. This allows for time-varying covariates.
<code>control</code>	a list with components <code>eps</code> (convergence criterion), <code>maxiter</code> (maximum number of iterations), and <code>silent</code> (logical, controlling amount of output). You can change any component without mention the other(s).
<code>singular.ok</code>	Not used.
<code>model</code>	Not used.
<code>x</code>	Return the design matrix in the model object?
<code>y</code>	Return the response in the model object?

Details

The parameterization is different from the one used by `survreg`. The model is

$$S(t; a, b, \beta, z) = S_0((t / \exp(b - z\beta))^{\exp(a)})$$

where S_0 is some standardized survivor function.

Value

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

<code>coefficients</code>	Fitted parameter estimates.
<code>var</code>	Covariance matrix of the estimates.
<code>loglik</code>	Vector of length two; first component is the value at the initial parameter values, the second componet is the maximized value.
<code>score</code>	The score test statistic (at the initial value).
<code>linear.predictors</code>	The estimated linear predictors.
<code>means</code>	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.aftreg` doesn't work if threeway or higher order interactions are present. Use `print.coxph` in that case.

Author(s)

Göran Broström

See Also

`coxreg`, `phreg`

Examples

```
data(male.mortality)
aftreg(Surv(enter, exit, event) ~ ses, data = male.mortality, dist = "gompertz")
phreg(Surv(enter, exit, event) ~ ses, data = male.mortality, dist = "gompertz")
coxreg(Surv(enter, exit, event) ~ ses, data = male.mortality)
```

aftreg.fit	<i>Parametric proportional hazards regression</i>
------------	---

Description

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

Usage

```
aftreg.fit(X, Y, dist, strata, offset, init, shape, id, control)
```

Arguments

<code>X</code>	The design (covariate) matrix.
<code>Y</code>	A survival object, the response.
<code>dist</code>	Which baseline distribution?
<code>strata</code>	A stratum variable.
<code>offset</code>	Offset.
<code>init</code>	Initial regression parameter values.
<code>shape</code>	If positive, a fixed value of the shape parameter in the distribution. Otherwise, the shape is estimated.
<code>id</code>	See corresponding argument to <code>aftreg</code> .
<code>control</code>	Controls convergence and output.

Details

See `aftreg` for more detail.

Value

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

Author(s)

Göran Broström

See Also[aftreg](#)

`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

<code>dat</code>	Input data frame. Must contain survival data.
<code>window</code>	Vector of length two; the age interval.
<code>surv</code>	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)

Göran Broström

See Also[cal.window](#), [coxreg](#), [aftreg](#)**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](#), [aftreg](#)

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.dist`*Graphical goodness-of-fit test*

Description

Comparison of the cumulative hazards functions for a semi-parametric and a parametric model.

Usage

```
check.dist(sp, pp, new.data = sp$means)
```

Arguments

<code>sp</code>	An object of type "cotxreg", typically output from coxreg
<code>pp</code>	An object of type "phreg", typically output from phreg
<code>new.data</code>	At what covariate values should the baseline hazards for the coxreg model be calculated?

Details

For the moment only a graphical comparison.

Value

No return value.

Author(s)

Göran Broström

References

~put references to the literature/web site here ~

See Also

[coxreg](#) and [phreg](#).

Examples

```
data(male.mortality)
oldpar <- par(mfrow = c(2, 2))
fit.cr <- coxreg(Surv(enter, exit, event) ~ ses, data = male.mortality)
fit.w <- phreg(Surv(enter, exit, event) ~ ses, data = male.mortality)
fit.g <- phreg(Surv(enter, exit, event) ~ ses, data = male.mortality,
  dist = "gompertz")
fit.ln <- phreg(Surv(enter, exit, event) ~ ses, data = male.mortality,
  dist = "lognormal")
```

```
fit.ev <- phreg(Surv(enter, exit, event) ~ ses, data = male.mortality,
dist = "ev")
check.dist(fit.cr, fit.w)
check.dist(fit.cr, fit.g)
check.dist(fit.cr, fit.ln)
check.dist(fit.cr, fit.ev)
par(oldpar)
```

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.

Author(s)

Göran Broström

See Also

[join.spells](#), [coxreg](#), [aftreg](#)

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(), weights, t.offset,
na.action = getOption("na.action"), init = NULL,
method = c("efron", "breslow", "mnppl", "ml"),
control = list(eps = 1e-08, maxiter = 25, trace = FALSE),
singular.ok = TRUE, model = FALSE,
center = TRUE,
x = FALSE, y = TRUE, boot = FALSE, efrac = 0,
geometric = FALSE, rs = NULL,
frailty = NULL, max.survs = NULL)
```

Arguments

<code>formula</code>	a formula object, with the response on the left of a <code>~</code> operator, and the terms on the right. The response must be a survival object as returned by the <code>Surv</code> function.
<code>data</code>	a <code>data.frame</code> in which to interpret the variables named in the formula.
<code>weights</code>	Case weights; time-fixed or time-varying.
<code>t.offset</code>	Case offsets; time-varying.
<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>init</code>	vector of initial values of the iteration. Default initial value is zero for all variables.
<code>method</code>	Method of treating ties, "efron" (default), "breslow", "mnppl" (maximum partial partial likelihood), or "ml" (maximum likelihood).
<code>control</code>	a list with components <code>eps</code> (convergence criterion), <code>maxiter</code> (maximum number of iterations), and <code>silent</code> (logical, controlling amount of output). You can change any component without mention the other(s).
<code>singular.ok</code>	Not used
<code>model</code>	Not used
<code>center</code>	If TRUE, the hazards are calculated at the means of the covariates. If FALSE, at zero.
<code>x</code>	Return the design matrix in the model object?
<code>y</code>	return the response in the model object?

<code>rs</code>	Risk set?
<code>boot</code>	Number of boot replicates. Defaults to FALSE, no boot samples.
<code>efrac</code>	Upper limit of fraction failures in 'mppl'.
<code>geometric</code>	If TRUE, forces an 'ml' model with constant riskset probability. Default is FALSE.
<code>frailty</code>	Grouping variable for frailty analysis. Not in use yet.
<code>max.survs</code>	Sampling of risk sets? If given, it should be (the upper limit of) 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`. The methods `mppl` and `ml` are maximum likelihood based.

Value

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

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

<code>bootstrap</code>	The (matrix of) bootstrap replicates, if requested on input. It is up to the user to do whatever desirable with this sample.
<code>boot.sd</code>	The estimated standard errors of the bootstrap replicates.
<code>call</code>	The call.
<code>method</code>	The method.
<code>convergence</code>	Did the optimization converge?
<code>fail</code>	Did the optimization fail? (Is <code>NULL</code> 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

Broström, G. and Lindkvist, M. (2008). Partial partial likelihood. *Communications in Statistics: Simulation and Computation* 37:4, 679-686.

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, weights, t.offset = NULL,
           strats, offset, init, max.survs,
           method = "breslow", center = TRUE,
           boot = FALSE, efrac = 0, calc.hazards = TRUE,
           calc.martres = TRUE, control, verbose = TRUE)
```

Arguments

X	The design matrix.
Y	The survival object.
rs	The risk set composition. If absent, calculated.
weights	Case weights; time-fixed or time-varying.
t.offset	Case offset; time-varying.
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".
center	See <code>coxreg</code> .
boot	Number of bootstrap replicates. Defaults to FALSE, no bootstrapping.
efrac	Upper limit of fraction failures in 'mppl'.
calc.hazards	Should estimates of baseline hazards be calculated?
calc.martres	Should martingale residuals be calculated?
control	See <code>coxreg</code>
verbose	Should Warnings about convergence be printed?

Details

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

Value

A list with components

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

<code>dat</code>	A data frame
<code>response</code>	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

<code>y</code>	The response.
<code>covar</code>	A data frame with unique rows of covariates.
<code>keys</code>	Pointers from <code>y</code> to <code>covar</code> , 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)
```

Description

Density, distribution function, quantile function, hazard function, cumulative hazard function, and random generation for the EV distribution with parameters `shape` and `scale`.

Usage

```
dEV(x, shape = 1, scale = 1, log = FALSE)
pEV(q, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
qEV(p, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
hEV(x, shape = 1, scale = 1, log = FALSE)
HEV(x, shape = 1, scale = 1, log.p = FALSE)
rEV(n, shape = 1, scale = 1)
```

Arguments

`x`, `q` vector of quantiles.
`p` vector of probabilities.
`n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
`shape`, `scale` shape and scale parameters, both defaulting to 1.
`log`, `log.p` logical; if TRUE, probabilities `p` are given as `log(p)`.
`lower.tail` logical; if TRUE (default), probabilities are $P(X \leq x)$, otherwise, $P(X > x)$.

Details

The EV distribution with `scale` parameter a and `shape` parameter σ has hazard function given by

$$h(x) = (b/\sigma)(x/\sigma)^{b-1} \exp(-(x/\sigma)^b)$$

for $x \geq 0$.

Value

`dEV` gives the density, `pEV` gives the distribution function, `qEV` gives the quantile function, `hEV` gives the hazard function, `HEV` gives the cumulative hazard function, and `rEV` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

`geome.fit`*Constant intensity discrete time proportional hazards*

Description

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

Usage

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

Arguments

<code>X</code>	The design matrix
<code>Y</code>	Survival object
<code>rs</code>	risk set produced by <code>risksets</code>
<code>strats</code>	Stratum indicator
<code>offset</code>	Offset
<code>init</code>	Initial values
<code>max.survs</code>	Maximal survivors
<code>method</code>	"ml", always, i.e., this argument is ignored.
<code>boot</code>	should we bootstrap?
<code>control</code>	See <code>coxreg</code> .

Value

See the code.

Note

Nothing special

Author(s)

Göran Broström

References

See [coxreg](#).

See Also

[coxreg](#)

Description

Density, distribution function, quantile function, hazard function, cumulative hazard function, and random generation for the Gompertz distribution with parameters `shape` and `scale`.

Usage

```

dgompertz(x, shape = 1, scale = 1, log = FALSE)
pgompertz(q, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
qgompertz(p, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
hgompertz(x, shape = 1, scale = 1, log = FALSE)
Hgompertz(x, shape = 1, scale = 1, log.p = FALSE)
rgompertz(n, shape = 1, scale = 1)

```

Arguments

<code>x</code> , <code>q</code>	vector of quantiles.
<code>p</code>	vector of probabilities.
<code>n</code>	number of observations. If <code>length(n) > 1</code> , the length is taken to be the number required.
<code>shape</code> , <code>scale</code>	Parameters: <code>shape</code> , defaulting to 1, and <code>scale</code> , defaulting to 1.
<code>log</code> , <code>log.p</code>	logical; if TRUE, probabilities <code>p</code> are given as <code>log(p)</code> .
<code>lower.tail</code>	logical; if TRUE (default), probabilities are $P(X \leq x)$, otherwise, $P(X > x)$.

Details

The Gompertz distribution with `shape` parameter a and `scale` parameter σ has hazard given by

$$h(x) = a \exp(x/\sigma)$$

for $x \geq 0$.

Value

`dgompertz` gives the density, `pgompertz` gives the distribution function, `qgompertz` gives the quantile function, `hgompertz` gives the hazard function, `Hgompertz` gives the cumulative hazard function, and `rgompertz` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

`hweibull`*The (Cumulative) Hazard Function of a Weibull Distribution*

Description

`hweibull` calculates the hazard function of a Weibull distribution, and `Hweibull` calculates the corresponding cumulative hazard function.

Usage

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

Arguments

<code>x</code>	Vector of quantiles.
<code>shape</code>	The shape parameter.
<code>scale</code>	The scale parameter, defaults to 1.
<code>log</code>	logical; if TRUE, the log of the hazard function is given.

Details

See [dweibull](#).

Value

The (cumulative) hazard function, evaluated at `x`.

Author(s)

Göran Broström

See Also

[pweibull](#)

Examples

```
hweibull(3, 2, 1)
dweibull(3, 2, 1) / pweibull(3, 2, 1, lower.tail = FALSE)
Hweibull(3, 2, 1)
-pweibull(3, 2, 1, log.p = TRUE, lower.tail = FALSE)
```

`join.spells`*Straighten up a survival data frame*

Description

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

Usage

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

Arguments

<code>dat</code>	A data frame with names enter, exit, event, id.
<code>strict</code>	If TRUE, nothing is changed if errors in spells (non-positive length, overlapping intervals, etc.) are detected. Otherwise (the default), bad spells are removed, with "earlier life" having higher priority.
<code>eps</code>	Tolerance for equality of two event times. Should be kept small.

Details

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

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](#), [aftreg](#), [check.surv](#)

Description

Density, distribution function, quantile function, hazard function, cumulative hazard function, and random generation for the Loglogistic distribution with parameters `shape` and `scale`.

Usage

```
dllogis(x, shape = 1, scale = 1, log = FALSE)
pllogis(q, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
qllogis(p, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
hllogis(x, shape = 1, scale = 1, log = FALSE)
Hllogis(x, shape = 1, scale = 1, log.p = FALSE)
rllogis(n, shape = 1, scale = 1)
```

Arguments

`x`, `q` vector of quantiles.
`p` vector of probabilities.
`n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
`shape`, `scale` shape and scale parameters, both defaulting to 1.
`log`, `log.p` logical; if TRUE, probabilities `p` are given as `log(p)`.
`lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Details

The Loglogistic distribution with `shape` parameter a and `scale` parameter σ has density given by

$$f(x) = (a/\sigma)(x/\sigma)^{a-1}/(1 + (x/\sigma)^a)^2$$

for $x \geq 0$. The cumulative distribution function is $F(x) = 1 - 1/(1 + (x/\sigma)^a)$ on $x \geq 0$.

Value

`dllogis` gives the density, `pllogis` gives the distribution function, `qllogis` gives the quantile function, `hllogis` gives the hazard function, `Hllogis` gives the cumulative hazard function, and `rllogis` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

 Lognormal

The Lognormal Distribution

Description

Hazard function and cumulative hazard function for the Lognormal distribution with parameters `shape` and `scale`.

Usage

```
hlnorm(x, meanlog = 0, sdlog = 1,
       shape = 1 / sdlog, scale = exp(meanlog), log = FALSE)
Hlnorm(x, meanlog = 0, sdlog = 1,
       shape = 1 / sdlog, scale = exp(meanlog), log.p = FALSE)
```

Arguments

`x` vector of quantiles.
`meanlog, sdlog` Mean and standard deviation in the distribution of the logarithm of a lognormal random variable.
`shape, scale` shape and scale parameters, both defaulting to 1.
`log, log.p` logical; if TRUE, probabilities `p` are given as `log(p)`.

Details

This is a complement to the Lognormal distribution, see [Lognormal](#) for further details.

Value

`hlnorm` gives the hazard function, and `Hlnorm` gives the cumulative hazard function.
 Invalid arguments will result in return value `NaN`, with a warning.

 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

<code>dat</code>	A data frame containing interval specified survival data and covariates, of which one must give a "birth date", the connection between duration and calendar time
<code>com.dat</code>	Data frame with communal covariates. They must have the same start year and periodicity, given by <code>com.ins</code>
<code>communal</code>	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 <code>lag</code> and the fourth is <code>scale</code> .
<code>start</code>	Start date in decimal form.
<code>period</code>	Period length. Defaults to one.
<code>lag</code>	The lag of the effect. Defaults to zero.
<code>surv</code>	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 <code>dat</code>
<code>tol</code>	Largest length of an interval considered to be of zero length. The cutting sometimes produces zero length intervals, which we want to discard.
<code>fortran</code>	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`, `aftreg`, 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`, `aftreg`, `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)

```

 Makeham

The Gompertz-Makeham Distribution

Description

Density, distribution function, quantile function, hazard function, cumulative hazard function, and random generation for the Gompertz-Makeham distribution with parameters `shape1`, `shape2` and `scale`.

Usage

```

dmakeham(x, shape = c(1, 1), scale = 1, log = FALSE)
pmakeham(q, shape = c(1, 1), scale = 1, lower.tail = TRUE, log.p = FALSE)
qmakeham(p, shape = c(1, 1), scale = 1, lower.tail = TRUE, log.p = FALSE)
hmakeham(x, shape = c(1, 1), scale = 1, log = FALSE)
Hmakeham(x, shape = c(1, 1), scale = 1, log.p = FALSE)
rmakeham(n, shape = c(1, 1), scale = 1)

```

Arguments

<code>x</code> , <code>q</code>	vector of quantiles.
<code>p</code>	vector of probabilities.
<code>n</code>	number of observations. If <code>length(n) > 1</code> , the length is taken to be the number required.
<code>shape</code> , <code>scale</code>	Parameters: <code>shape</code> , a vector of length 2, defaulting to <code>c(1, 1)</code> , and <code>scale</code> , defaulting to 1.
<code>log</code> , <code>log.p</code>	logical; if TRUE, probabilities <code>p</code> are given as <code>log(p)</code> .
<code>lower.tail</code>	logical; if TRUE (default), probabilities are $P(X \leq x)$, otherwise, $P(X > x)$.

Details

The Gompertz-Makeham distribution with `shape` parameters a_1 and a_2 and `scale` parameter σ has hazard function given by

$$h(x) = a_2 + a_1 \exp(x/\sigma)$$

for $x \geq 0$.

Value

`dmakeham` gives the density, `pmakeham` gives the distribution function, `qmakeham` gives the quantile function, but is not yet implemented, `hmakeham` gives the hazard function, `Hmakeham` gives the cumulative hazard function, and `rmakeham` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

<code>male.mortality</code>	<i>Male mortality in ages 40-60, nineteenth century</i>
-----------------------------	---

Description

Males born in the years 1800-1820 and surviving at least 40 years in the parish Skellefteå in northern Sweden are followed from their fortieth birthday until death or the sixtieth birthday, whichever comes first.

Usage

```
data(male.mortality)
```

Format

A data frame with 2058 observations on the following 6 variables.

id Personal identification number.

enter Start of duration. Measured in years since the fortieth birthday.

exit End of duration. Measured in years since the fortieth birthday.

event a logical vector indicating death at end of interval.

birthdate The birthdate in decimal form.

ses Socio-economic status, a factor with levels `lower` `upper`

Details

The interesting explanatory covariate is `ses` (socioeconomic status), which is a time-varying covariate. This explains why several individuals are represented by more than one record each. Left truncation and right censoring are introduced this way.

Source

Data is coming from The Demographic Data Base, Umeå University, Umeå, Sweden.

References

<http://www.ddb.umu.se>

Examples

```
data(male.mortality)
coxreg(Surv(enter, exit, event) ~ ses, data = male.mortality)
```

mlreg

*ML proportional hazards regression***Description**

Maximum Likelihood estimation of proportional hazards models. Is deprecated, use `coxreg` instead.

Usage

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

Arguments

<code>formula</code>	a formula object, with the response on the left of a <code>~</code> operator, and the terms on the right. The response must be a survival object as returned by the <code>Surv</code> function.
<code>data</code>	a <code>data.frame</code> in which to interpret the variables named in the formula.
<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>init</code>	vector of initial values of the iteration. Default initial value is zero for all variables.
<code>method</code>	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.
<code>control</code>	a list with components <code>eps</code> (convergence criterion), <code>maxiter</code> (maximum number of iterations), and <code>silent</code> (logical, controlling amount of output). You can change any component without mention the other(s).
<code>singular.ok</code>	Not used.
<code>model</code>	Not used.
<code>center</code>	Should covariates be centered? Default is TRUE
<code>x</code>	Return the design matrix in the model object?
<code>y</code>	return the response in the model object?
<code>boot</code>	No. of bootstrap replicates. Defaults to FALSE, i.e., no bootstrapping.
<code>geometric</code>	If TRUE, the intensity is assumed constant within strata.
<code>rs</code>	Risk set? If present, speeds up calculations considerably.
<code>frailty</code>	A grouping variable for frailty analysis. Full name is needed.
<code>max.survs</code>	Sampling of risk sets?

Details

Method ML performs a true discrete analysis, i.e., one parameter per observed event time. Method MPPL is a compromise 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

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

Note futher that `mlreg` is deprecated. `coxreg` should be used instead.

Author(s)

Göran Broström

References

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

See Also

[coxreg](#), [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
```

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

Value

A list with components	
events	No. of events in each time period.
exposure	Exposure times in each period.
intensity	events / exposure

Author(s)

Göran Broström

See Also

[piecewise](#)

phfunc

Loglikelihood function of a proportional hazards regression

Description

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

Usage

```
phfunc(beta = NULL, lambda, p, X = NULL, Y, offset = rep(0, length(Y)),
ord = 2, pfixed = FALSE, dist = "weibull")
```

Arguments

beta	Regression parameters
lambda	The scale parameter
p	The shape parameter
X	The design (covariate) matrix.
Y	The response, a survival object.
offset	Offset.
ord	ord = 0 means only loglikelihood, 1 means score vector as well, 2 loglikelihood, score and hessian.

<code>pfixed</code>	Logical, if TRUE the shape parameter is regarded as a known constant in the calculations, meaning that it is not considered in the partial derivatives.
<code>dist</code>	Which distribution? The default is "weibull", with the alternatives "loglogistic" and "lognormal".

Details

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

Value

A list with components

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

Author(s)

Göran Broström

See Also

[phreg](#)

phreg

Parametric Proportional Hazards Regression

Description

Proportional hazards model with parametric baseline hazard(s). Allows for stratification with different scale and shape in each stratum, and left truncated and right censored data.

Usage

```
phreg(formula = formula(data), data = parent.frame(),
na.action = getOption("na.action"), dist = "weibull", init, shape = 0,
control = list(eps = 1e-08, maxiter = 20, trace = FALSE),
singular.ok = TRUE, model = FALSE, x = FALSE, y = TRUE, center = TRUE)
```

Arguments

<code>formula</code>	a formula object, with the response on the left of a <code>~</code> operator, and the terms on the right. The response must be a survival object as returned by the <code>Surv</code> function.
<code>data</code>	a <code>data.frame</code> in which to interpret the variables named in the formula.
<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>dist</code>	Which distributin? Default is "weibull", with the alternatives "loglogistic" and "lognormal".
<code>init</code>	vector of initial values of the iteration. Default initial value is zero for all variables.
<code>shape</code>	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.
<code>control</code>	a list with components <code>eps</code> (convergence criterion), <code>maxiter</code> (maximum number of iterations), and <code>silent</code> (logical, controlling amount of output). You can change any component without mention the other(s).
<code>singular.ok</code>	Not used.
<code>model</code>	Not used.
<code>x</code>	Return the design matrix in the model object?
<code>y</code>	Return the response in the model object?
<code>center</code>	Should the covariates be centered? Default is TRUE. No correction in parameter estimates is made if centering.

Details

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

$$S(t; a, b, \beta, z) = S_0((t/b)^a)^{\exp(z\beta)}$$

where S_0 is some standardized survivor function.

Value

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

<code>coefficients</code>	Fitted parameter estimates.
<code>var</code>	Covariance matrix of the estimates.
<code>loglik</code>	Vector of length two; first component is the value at the initial parameter values, the second componet is the maximized value.
<code>score</code>	The score test statistic (at the initial value).
<code>linear.predictors</code>	The estimated linear predictors.
<code>means</code>	Means of the columns of the design matrix.

<code>w.means</code>	Weighted (against exposure time) means of covariates; weighted relative frequencies of levels of factors.
<code>n</code>	Number of spells in indata (possibly after removal of cases with NA's).
<code>events</code>	Number of events in data.
<code>terms</code>	Used by extractor functions.
<code>assign</code>	Used by extractor functions.
<code>y</code>	The Surv vector.
<code>isF</code>	Logical vector indicating the covariates that are factors.
<code>covars</code>	The covariates.
<code>ttr</code>	Total Time at Risk.
<code>levels</code>	List of levels of factors.
<code>formula</code>	The calling formula.
<code>call</code>	The call.
<code>method</code>	The method.
<code>convergence</code>	Did the optimization converge?
<code>fail</code>	Did the optimization fail? (Is NULL if not).
<code>pfixed</code>	TRUE if shape was fixed in the estimation.

Warning

The print method `print.phreg` 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 $\log(\text{scale})$, but nothing else. If you don't like this, set `center = FALSE`.

Author(s)

Göran Broström

See Also

`coxreg`, `check.dist`, `link{aftreg}`.

Examples

```
data(male.mortality)
fit <- phreg(Surv(enter, exit, event) ~ ses, data = male.mortality)
fit
plot(fit)
fit.cr <- coxreg(Surv(enter, exit, event) ~ ses, data = male.mortality)
check.dist(fit.cr, fit)
```

phreg.fit	<i>Parametric proportional hazards regression</i>
-----------	---

Description

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

Usage

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

Arguments

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

Details

See `phreg` for more detail.

Value

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

Author(s)

Göran Broström

See Also[phreg](#)

`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

<code>enter</code>	Left interval endpoint
<code>exit</code>	Right interval endpoint
<code>event</code>	Indicator of event
<code>cutpoints</code>	Vector of cutpoints

Details

Exact calculation.

Value

A list with components

<code>events</code>	Vector of number of events
<code>exposure</code>	Vector of total exposure time
<code>intensity</code>	Vector of hazards, <code>intensity == events / exposure</code>

Author(s)

Göran Broström

See Also[perstat](#)

`plot.aftreg`*Plots output from a Weibull regression*

Description

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

Usage

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

Arguments

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

Details

The plot is drawn at the mean values of the covariates, by default.

Value

No return value.

Author(s)

Göran Broström

See Also

[aftreg](#)

Examples

```

y <- rlllogis(40, shape = 1, scale = 1)
x <- rep(c(1,1,2,2), 10)
fit <- aftreg(Surv(y, rep(1, 40)) ~ x, dist = "loglogistic")
plot(fit)

```

plot.coxreg *Plots of survivor functions.*

Description

Baseline hazards estimates.

Usage

```

plot.coxreg(x, fn = c("cum", "surv", "log", "loglog"), fig = TRUE,
xlim=NULL, ylim=NULL, main=NULL, xlab="Duration", ylab="",
new.data = x$means, ...)

```

Arguments

x	A coxreg object, typically the output from link{coxreg}.
fn	Which type of plot?
fig	Should a plot actually be produced? Default is TRUE.
xlim	Horizontal plot limits. If NULL, calculated by the function.
ylim	Vertical plot limits. If NULL, set to c(0, 1) for a survival plot, otherwise adaptive, data dependent.
main	A heading for the plot.
xlab	Label on the x axis.
ylab	Label on the y-axis.
new.data	At what covariate values should the calculations be done? Default is the mean values of the covariates.
...	Anything that <code>plot.default</code> likes...

Details

This function is a wrapper for `plot.hazdata`.

Value

A list where the elements are two-column matrices, one for each stratum in the model. The first column contains risktimes, and the second the y coordinates for the requested curve.

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)
fit <- coxreg(Surv(time0, time1, event) ~ strata(group))
plot.coxreg(fit)
```

plot.hazdata *Plots of survivor functions.*

Description

Baseline hazards estimates.

Usage

```
plot.hazdata(x, fn = c("cum", "surv", "log", "loglog"), fig = TRUE,
            xlim=NULL, ylim=NULL, main=NULL, xlab=NULL, ylab=NULL, ...)
```

Arguments

x	A hazdata object, typically the 'hazards' element in the output from <code>link{coxreg}</code> .
fn	Which type of plot?
fig	Should a plot actually be produced? Default is TRUE.
xlim	Horizontal plot limits. If NULL, calculated by the function.
ylim	Vertical plot limits. If NULL, set to <code>c(0, 1)</code>
main	A heading for the plot.
xlab	Label on the x axis.
ylab	Label on the y-axis.
...	Anything that <code>plot.default</code> likes...

Details

It is also possible to have as first argument an object of type "coxreg", given that it contains a component of type "hazdata".

Value

A list where the elements are two-column matrices, one for each stratum in the model. The first column contains risktimes, and the second the y coordinates for the requested curve(s).

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)
fit <- coxreg(Surv(time0, time1, event) ~ strata(group))
plot(fit$hazards)
```

plot.phreg

Plots output from a Weibull regression

Description

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

Usage

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

Arguments

x	A phreg object
fn	Which functions should be plotted! Default is all. They will scroll by, so you have to take care explicitly what you want to be produced. See, eg, <code>par(mfrow = ...)</code>
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, by default.

Value

No return value.

Author(s)

Göran Broström

See Also[phreg](#)**Examples**

```

y <- rlllogis(40, shape = 1, scale = 1)
x <- rep(c(1,1,2,2), 10)
fit <- phreg(Surv(y, rep(1, 40)) ~ x, dist = "loglogistic")
plot(fit)

```

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

Arguments

x	A weibreg object
fn	Which functions should be plotted! Default is all. They will scroll by, so you have to take care explicitly what you want to be produced. See, eg, <code>par(mfrow = ...)</code>
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.aftreg *Prints aftreg objects*

Description

The hazard, the cumulative hazard, the density, and the survivor baseline functions are plotted.

Usage

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

Arguments

x	A aftreg 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[phreg](#), [print.coxph](#)

`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

<code>x</code>	A <code>coxreg</code> object, typically the result of running <code>coxreg</code>
<code>digits</code>	Output format.
<code>...</code>	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.phreg *Prints phreg objects*

Description

The hazard, the cumulative hazard, the density, and the survivor baseline functions are plotted.

Usage

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

Arguments

x	A phreg 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

[phreg](#), [print.coxph](#)

print.weibreg *Prints weibreg objects*

Description

The hazard, the cumulative hazard, the density, and the survivor baseline functions are plotted.

Usage

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

Arguments

<code>x</code>	A <code>weibreg</code> object
<code>digits</code>	Precision in printing
<code>...</code>	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

<code>x</code>	A <code>Surv</code> object.
<code>strata</code>	Stratum indicator.
<code>max.survs</code>	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

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

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

<code>object</code>	A coxreg object
<code>...</code>	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	<i>Calculating failure times, risk set sizes and No. of events in each risk set</i>
--------------	---

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	<i>Transforms a "survival" data frame into a data frame suitable for binary (logistic) regression</i>
----------	---

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

<code>dat</code>	A data frame with three variables representing the survival response. The default is that they are named <code>enter</code> , <code>exit</code> , and <code>event</code>
<code>surv</code>	A character string with the names of the three variables representing survival.
<code>strats</code>	An eventual stratification variable.
<code>max.survs</code>	Maximal number 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`.

<code>event</code>	Indicates an event in the corresponding risk set.
<code>riskset</code>	Factor (with levels 1, 2, ...) indicating risk set.
<code>risktime</code>	The 'risktime' (age) in the corresponding riskset.
<code>orig.row</code>	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

[coxreg](#), [glm](#).

Examples

```
enter <- rep(0, 4)
exit <- 1:4
event <- rep(1, 4)
z <- rep(c(-1, 1), 2)
dat <- data.frame(enter, exit, event, z)
binDat <- toBinary(dat)
dat
binDat
coxreg(Surv(enter, exit, event) ~ z, method = "ml", data = dat)
## Same as:
summary(glm(event ~ z + riskset, data = binDat, family = binomial(link = cloglog)))
```

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

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	<i>Calculate duration in years from "0000-01-01" to a given date</i>
--------	--

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

Value

A vector of durations, as described 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

<code>formula</code>	a formula object, with the response on the left of a <code>~</code> operator, and the terms on the right. The response must be a survival object as returned by the <code>Surv</code> function.
<code>data</code>	a <code>data.frame</code> in which to interpret the variables named in the formula.
<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>init</code>	vector of initial values of the iteration. Default initial value is zero for all variables.
<code>shape</code>	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.
<code>control</code>	a list with components <code>eps</code> (convergence criterion), <code>maxiter</code> (maximum number of iterations), and <code>silent</code> (logical, controlling amount of output). You can change any component without mention the other(s).
<code>singular.ok</code>	Not used.
<code>model</code>	Not used.
<code>x</code>	Return the design matrix in the model object?
<code>y</code>	Return the response in the model object?
<code>center</code>	Deprecated, and not used. Will be removed in the future.

Details

The parameterization is the same as in `coxreg` 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 $b \exp(-z\beta/a)$

Value

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

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

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

Author(s)

Göran Broström

See Also[weibreg](#)

wfunk

*Loglikelihood 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)
```

Arguments

<code>beta</code>	Regression parameters
<code>lambda</code>	The scale parameter
<code>p</code>	The shape parameter
<code>X</code>	The design (covariate) matrix.
<code>Y</code>	The response, a survival object.

<code>offset</code>	Offset.
<code>ord</code>	<code>ord = 0</code> means only loglikelihood, <code>1</code> means score vector as well, <code>2</code> loglikelihood, score and hessian.
<code>pfixed</code>	Logical, if <code>TRUE</code> the shape parameter is regarded as a known constant in the calculations, meaning that it is not considered in the partial derivatives.

Details

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

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

This is in correspondence with [dweibull](#).

Value

A list with components

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

Author(s)

Göran Broström

See Also

[weibreg](#)

Index

- *Topic **cluster**
 - toBinary, 47
- *Topic **datasets**
 - male.mortality, 24
- *Topic **distribution**
 - check.dist, 7
 - EV, 15
 - Gompertz, 17
 - Loglogistic, 20
 - Lognormal, 21
 - Makeham, 23
 - phfunc, 28
 - wfunk, 53
- *Topic **dplot**
 - plot.aftreg, 34
 - plot.phreg, 37
 - plot.weibreg, 39
- *Topic **manip**
 - check.surv, 8
 - cro, 14
 - join.spells, 19
- *Topic **nonparametric**
 - perstat, 27
 - piecewise, 33
- *Topic **print**
 - summary.coxreg, 44
 - summary.weibreg, 45
- *Topic **regression**
 - aftreg, 1
 - aftreg.fit, 4
 - coxreg, 9
 - coxreg.fit, 12
 - mlreg, 25
 - phreg, 29
 - phreg.fit, 32
 - print.aftreg, 40
 - print.phreg, 42
 - print.weibreg, 42
 - weibreg, 50
 - weibreg.fit, 52
- *Topic **survival**
 - aftreg, 1
 - aftreg.fit, 4
 - age.window, 5
 - cal.window, 6
 - check.surv, 8
 - coxreg, 9
 - coxreg.fit, 12
 - geome.fit, 16
 - hweibull, 18
 - join.spells, 19
 - make.communal, 21
 - mlreg, 25
 - perstat, 27
 - phfunc, 28
 - phreg, 29
 - phreg.fit, 32
 - piecewise, 33
 - plot.aftreg, 34
 - plot.coxreg, 35
 - plot.hazdata, 36
 - plot.phreg, 37
 - plot.Surv, 38
 - plot.weibreg, 39
 - print.aftreg, 40
 - print.coxreg, 41
 - print.phreg, 42
 - print.weibreg, 42
 - risksets, 43
 - summary.coxreg, 44
 - summary.weibreg, 45
 - table.events, 46
 - toBinary, 47
 - toDate, 48
 - toTime, 49
 - weibreg, 50
 - weibreg.fit, 52
 - wfunk, 53

- aftreg, 1, 4–6, 8, 19, 22, 34
- aftreg.fit, 4
- age.window, 5, 6
- cal.window, 5, 6, 22
- check.dist, 7, 31
- check.surv, 8, 19
- coxph, 10, 11, 22, 30, 50
- coxreg, 3, 5–8, 9, 12, 13, 16, 19, 22, 26, 27, 30, 31, 41, 44, 48, 50, 52
- coxreg.fit, 12
- cro, 14
- dEV (EV), 15
- dgompertz (Gompertz), 17
- dllogis (Loglogistic), 20
- dmakeham (Makeham), 23
- dweibull, 18, 54
- EV, 15
- geome.fit, 16
- glm, 48
- Gompertz, 17
- HEV (EV), 15
- hEV (EV), 15
- Hgompertz (Gompertz), 17
- hgompertz (Gompertz), 17
- Hllogis (Loglogistic), 20
- hllogis (Loglogistic), 20
- Hlnorm (Lognormal), 21
- hlnorm (Lognormal), 21
- Hmakeham (Makeham), 23
- hmakeham (Makeham), 23
- Hweibull (hweibull), 18
- hweibull, 18
- join.spells, 8, 19
- Llogis (Loglogistic), 20
- Lnorm (Lognormal), 21
- Loglogistic, 20
- Lognormal, 21, 21
- make.communal, 21
- Makeham, 23
- male.mortality, 24
- match, 14
- mlreg, 25, 41, 52
- paste, 14
- perstat, 27, 33
- pEV (EV), 15
- pgompertz (Gompertz), 17
- phfunc, 28
- phreg, 3, 7, 29, 29, 32, 33, 38, 41, 42
- phreg.fit, 32
- piecewise, 28, 33
- pllogis (Loglogistic), 20
- plot.aftreg, 34
- plot.coxreg, 35
- plot.default, 35, 36
- plot.hazdata, 35, 36
- plot.phreg, 37
- plot.Surv, 38
- plot.weibreg, 39
- pmakeham (Makeham), 23
- print.aftreg, 3, 40
- print.coxph, 3, 31, 40–43, 51
- print.coxreg, 41, 44, 45
- print.phreg, 31, 42
- print.weibreg, 42, 45, 51, 52
- pweibull, 18
- qEV (EV), 15
- qgompertz (Gompertz), 17
- qllogis (Loglogistic), 20
- qmakeham (Makeham), 23
- rEV (EV), 15
- rgompertz (Gompertz), 17
- risksets, 11, 13, 27, 43, 46
- rllogis (Loglogistic), 20
- rmakeham (Makeham), 23
- summary.coxreg, 44
- summary.weibreg, 45
- survreg, 2, 30, 50
- table.events, 44, 46
- toBinary, 47
- toDate, 48, 49
- toTime, 48, 49
- weibreg, 40, 43, 50, 52–54
- weibreg.fit, 52
- Weibull, 51
- wfunk, 53