Package 'mlt'

December 14, 2023

Title Most Likely Transformations

Version 1.5-0

Date 2023-12-13

Description Likelihood-based estimation of conditional transformation models via the most likely transformation approach described in Hothorn et al. (2018) <DOI:10.1111/sjos.12291> and Hothorn (2020) <DOI:10.18637/jss.v092.i01>.

Depends basefun (>= 1.1-2), variables (>= 1.1-0)

Imports BB, alabama, stats, coneproj, graphics, methods, grDevices, sandwich, numDeriv, survival, Matrix, nloptr

Suggests MASS, nnet, TH.data, multcomp

URL http://ctm.R-forge.R-project.org

License GPL-2

Encoding UTF-8

NeedsCompilation yes

Author Torsten Hothorn [aut, cre] (<https://orcid.org/0000-0001-8301-0471>)

Maintainer Torsten Hothorn <Torsten.Hothorn@R-project.org>

Repository CRAN

Date/Publication 2023-12-14 12:20:02 UTC

R topics documented:

`mlt-package` *General Information on the* **mlt** *Package*

Description

The **mlt** package implements maximum likelihood estimation in conditional transformation models as introduced by Hothorn et al. (2020).

An introduction to the package is available in the `mlt` package vignette from package `mlt.docreg` (Hothorn, 2020).

A short talk on most likely transformations is available from https://channel9.msdn.com/Events/useR-international-R-User-conference/useR2016/Most-Likely-Transformations.

Novice users might find the high(er) level interfaces offered by package **tram** more convenient.

Author(s)

This package is authored by Torsten Hothorn <Torsten.Hothorn@R-project.org>.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Torsten Hothorn (2020), Most Likely Transformations: The mlt Package, *Journal of Statistical Software*, **92**(1), 1–68, doi:10.18637/jss.v092.i01

`confband` *Confidence Bands*

Description

Confidence bands for transformation, distribution, survivor or cumulative hazard functions

Usage

```
confband(object, newdata, level = 0.95, ...)
## S3 method for class 'mlt'
confband(object, newdata, level = 0.95,
       type = c("trafo", "distribution", "survivor", "cumhazard"),
       K = 20, cheat = K, ...)
```

Arguments

object	an object of class mlt
newdata	a data frame of observations
level	the confidence level
type	the function to compute the confidence band for
K	number of grid points the function is evaluated at
cheat	number of grid points the function is evaluated at when using the quantile obtained for K grid points
...	additional arguments to confint.glht

Details

The function is evaluated at K grid points and simultaneous confidence intervals are then interpolated in order to construct the band.

A smoother band can be obtained by setting cheat to something larger than K: The quantile is obtained for K grid points but the number of evaluated grid points cheat can be much larger at no additional cost. Technically, the nominal level is not maintained in this case but the deviation will be small for reasonably large K.

Value

For each row in newdata the function and corresponding confidence band evaluated at the K (or cheat) grid points is returned.

---

ctm *Conditional Transformation Models*

Description

Specification of conditional transformation models

Usage

```
ctm(response, interacting = NULL, shifting = NULL, scaling = NULL,
    scale_shift = FALSE, data = NULL,
    todistr = c("Normal", "Logistic", "MinExtrVal", "MaxExtrVal",
                "Exponential", "Laplace", "Cauchy"),
    sumconstr = inherits(interacting, c("formula", "formula_basis")), ...)
```

Arguments

response	a basis function, ie, an object of class basis
interacting	a basis function, ie, an object of class basis
shifting	a basis function, ie, an object of class basis
scaling	a basis function, ie, an object of class basis
scale_shift	a logical choosing between two different model types in the presence of a scaling term
data	either a data.frame containing the model variables or a formal description of these variables in an object of class vars
todistr	a character vector describing the distribution to be transformed
sumconstr	a logical indicating if sum constraints shall be applied
...	arguments to as.basis when shifting is a formula

Details

Specification of a transformation model of the form

$$P(Y \leq y \mid X = x) = F_Z(\sqrt{\exp(s(x)^\top \gamma)}[(a(y) \otimes b(x))^\top \vartheta] + d(x)^\top \beta)$$

(scale_shift = FALSE) or

$$P(Y \leq y \mid X = x) = F_Z(\sqrt{\exp(s(x)^\top \gamma)}[(a(y) \otimes b(x))^\top \vartheta + d(x)^\top \beta])$$

(scale_shift = TRUE) with bases $a(y)$ (response), $b(x)$ (interacting), $d(x)$ (shifting), and $s(x)$ (scaling). All except response can be missing (in this case an unconditional distribution is estimated).

This function only specifies the model which can then be fitted using mlt. The shift term is positive by default.

Possible choices of the distributions the model transforms to (the inverse link functions F_Z) include the standard normal ("Normal"), the standard logistic ("Logistic"), the standard minimum extreme value ("MinExtrVal", also known as Gompertz distribution), and the standard maximum extreme value ("MaxExtrVal", also known as Gumbel distribution) distributions. The exponential distribution ("Exponential") can be used to fit Aalen additive hazard models. Laplace and Cauchy distributions are also available.

Value

An object of class ctm.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

`ctm-methods` *Methods for ctm Objects*

Description

Methods for objects of class ctm

Usage

```
## S3 method for class 'ctm'
variable.names(object,
               which = c("all", "response", "interacting",
                         "shifting", "scaling"),
               ...)
## S3 method for class 'ctm'
coef(object, ...)
```

Arguments

`object`	an unfitted conditional transformation model as returned by `ctm`
`which`	a character specifying which names shall be returned
`...`	additional arguments

Details

`coef` can be used to get and set model parameters.

`mlt` *Most Likely Transformations*

Description

Likelihood-based model estimation in conditional transformation models

Usage

```
mlt(model, data, weights = NULL, offset = NULL, fixed = NULL, theta = NULL,
    pstart = NULL, scale = FALSE, dofit = TRUE, optim = mltoptim())
```

Arguments

model	a conditional transformation model as specified by ctm
data	a data.frame containing all variables specified in model
weights	an optional vector of case weights
offset	an optional vector of offset values; offsets are not added to an optional scaling term (see link{ctm})
fixed	a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix
theta	optional starting values for the model parameters
pstart	optional starting values for the distribution function evaluated at the data
scale	a logical indicating if (internal) scaling shall be applied to the model coefficients
dofit	a logical indicating if the model shall be fitted to the data (TRUE) or not. If theta is given, a model of class mlt (a full "fitted" model) featuring these parameters is returned. Otherwise, an unfitted model of class ctm is returned
optim	a list of functions implementing suitable optimisers

Details

This function fits a conditional transformation model by searching for the most likely transformation as described in Hothorn et al. (2018) and Hothorn (2020).

Value

An object of class mlt with corresponding methods.

References

Torsten Hothorn, Lisa Moest, Peter Buehlmann (2018), Most Likely Transformations, *Scandinavian Journal of Statistics*, **45**(1), 110–134, doi:10.1111/sjos.12291.

Torsten Hothorn (2020), Most Likely Transformations: The mlt Package, *Journal of Statistical Software*, **92**(1), 1–68, doi:10.18637/jss.v092.i01

Examples

```
### set-up conditional transformation model for conditional
### distribution of dist given speed
dist <- numeric_var("dist", support = c(2.0, 100), bounds = c(0, Inf))
speed <- numeric_var("speed", support = c(5.0, 23), bounds = c(0, Inf))
ctmm <- ctm(response = Bernstein_basis(dist, order = 4, ui = "increasing"),
            interacting = Bernstein_basis(speed, order = 3))

### fit model
mltm <- mlt(ctmm, data = cars)

### plot data
plot(cars)
```

```
### predict quantiles and overlay data with model via a "quantile sheet"
q <- predict(mltm, newdata = data.frame(speed = 0:24), type = "quantile",
             p = 2:8 / 10, K = 500)
tmp <- apply(q, 1, function(x) lines(0:24, x, type = "l"))
```

`mlt-methods` *Methods for mlt Objects*

Description

Methods for objects of class mlt

Usage

```
## S3 method for class 'mlt'
coef(object, fixed = TRUE, ...)
coef(object) <- value
## S3 method for class 'mlt'
weights(object, ...)
## S3 method for class 'mlt'
logLik(object, parm = coef(object, fixed = FALSE), w = NULL, newdata, ...)
## S3 method for class 'mlt'
vcov(object, parm = coef(object, fixed = FALSE), complete = FALSE, ...)
Hessian(object, ...)
## S3 method for class 'mlt'
Hessian(object, parm = coef(object, fixed = FALSE), ...)
Gradient(object, ...)
## S3 method for class 'mlt'
Gradient(object, parm = coef(object, fixed = FALSE), ...)
## S3 method for class 'mlt'
estfun(x, parm = coef(x, fixed = FALSE),
       w = NULL, newdata, ...)
## S3 method for class 'mlt'
residuals(object, parm = coef(object, fixed = FALSE),
       w = NULL, newdata, what = c("shifting", "scaling"), ...)
## S3 method for class 'mlt'
mkgrid(object, n, ...)
## S3 method for class 'mlt'
bounds(object)
## S3 method for class 'mlt'
variable.names(object, ...)
## S3 method for class 'mlt_fit'
update(object, weights = stats::weights(object),
      subset = NULL, offset = object$offset, theta = coef(object, fixed = FALSE),
       fixed = NULL, ...)
## S3 method for class 'mlt'
as.mlt(object)
```

Arguments

object, x	a fitted conditional transformation model as returned by mlt
fixed	a logical indicating if only estimated coefficients (fixed = FALSE) should be returned OR (for update) a named vector of fixed regression coefficients; the names need to correspond to column names of the design matrix
value	coefficients to be assigned to the model
parm	model parameters
w	model weights
what	type of residual: shifting means score with respect to a constant intercept for the shift term and scaling means score with respect to a constant intercept in the scaling term. This works whether or not such terms are actually present in the model
weights	model weights
newdata	an optional data frame of new observations. Allows evaluation of the log-likelihood for a given model object on these new observations. The parameters parm and w are ignored in this situation.
n	number of grid points
subset	an optional integer vector indicating the subset of observations to be used for fitting.
offset	an optional vector of offset values
theta	optional starting values for the model parameters
complete	currently ignored
...	additional arguments

Details

coef can be used to get and set model parameters, weights and logLik extract weights and evaluate the log-likelihood (also for parameters other than the maximum likelihood estimate). Hessian returns the Hessian and vcov the inverse thereof. Gradient gives the negative gradient (sum of the score contributions) and estfun the negative score contribution by each observation. mkgrid generates a grid of all variables (as returned by variable.names) in the model. update allows refitting the model with alternative weights and potentially different starting values. bounds gets bounds for bounded variables in the model.

---

mltoptim *Control Optimisation*

Description

Define optimisers and their control parameters

Usage

```
mltoptim(auglag = list(maxtry = 5, kkt2.check = FALSE),
         spg = list(maxit = 10000, quiet = TRUE, checkGrad = FALSE),
         nloptr = list(algorithm = "NLOPT_LD_MMA", xtol_rel = 1.0e-8),
         trace = FALSE)
```

Arguments

auglag	A list with control parameters for the auglag optimiser. maxtry is the number of times the algorithm is started on random starting values in case it failed with the precomputed ones.
spg	A list with control parameters for the BBoptim optimiser (calling spg internally).
nloptr	A list with control parameters for the nloptr family of optimisers.
trace	A logical switching trace reports by the optimisers off.

Details

This function sets-up functions to be called in mlt internally.

Value

A list of functions with arguments theta (starting values), f (log-likelihood), g (scores), ui and ci (linear inequality constraints). Adding further such functions is a way to add more optimisers to mlt. The first one in this list converging defines the resulting model.

Examples

```
### set-up linear transformation model for conditional
### distribution of dist given speed
dist <- numeric_var("dist", support = c(2.0, 100), bounds = c(0, Inf))
ctmm <- ctm(response = Bernstein_basis(dist, order = 4, ui = "increasing"),
            shifting = ~ speed, data = cars)

### use auglag with kkt2.check = TRUE => the numerically determined
### hessian is returned as "optim_hessian" slot
op <- mltoptim(auglag = list(maxtry = 5, kkt2.check = TRUE))[1]
mltm <- mlt(ctmm, data = cars, scale = FALSE, optim = op)

### compare analytical and numerical hessian
all.equal(c(Hessian(mltm)), c(mltm$optim_hessian), tol = 1e-4)
```

`plot-predict-simulate` *Plots, Predictions and Samples from mlt Objects*

Description

Plot, predict and sample from objects of class mlt

Usage

```
## S3 method for class 'ctm'
plot(x, newdata, type = c(
         "distribution", "logdistribution",
         "survivor", "logsurvivor",
         "density", "logdensity",
         "hazard", "loghazard",
         "cumhazard", "logcumhazard",
         "odds", "logodds",
         "quantile", "trafo"),
     q = NULL, prob = 1:(K - 1) / K, K = 50, col = rgb(.1, .1, .1, .1), lty = 1,
     add = FALSE, ...)
## S3 method for class 'mlt'
plot(x, ...)
## S3 method for class 'ctm'
predict(object, newdata, type = c("trafo",
         "distribution", "logdistribution",
         "survivor", "logsurvivor",
         "density", "logdensity",
         "hazard", "loghazard",
         "cumhazard", "logcumhazard",
         "odds", "logodds",
         "quantile"),
         terms = c("bresponse", "binteracting", "bshifting"),
         q = NULL, prob = NULL, K = 50, interpolate = FALSE, ...)
## S3 method for class 'mlt'
predict(object, newdata = object$data, ...)
## S3 method for class 'ctm'
simulate(object, nsim = 1, seed = NULL, newdata, K = 50, q = NULL,
         interpolate = FALSE, bysim = TRUE, ...)
## S3 method for class 'mlt'
simulate(object, nsim = 1, seed = NULL, newdata = object$data, bysim = TRUE, ...)
```

Arguments

`object`	a fitted conditional transformation model as returned by `mlt` or an unfitted conditional transformation model as returned by `ctm`
`x`	a fitted conditional transformation model as returned by `mlt`

newdata	an optional data frame of observations
type	type of prediction or plot to generate
q	quantiles at which to evaluate the model
prob	probabilities for the evaluation of the quantile function (type = "quantile")
terms	terms to evaluate for the predictions, corresponds to the argument response, interacting and shifting in ctm
K	number of grid points to generate (in the absence of q)
col	color for the lines to plot
lty	line type for the lines to plot
add	logical indicating if a new plot shall be generated (the default)
interpolate	logical indicating if quantiles shall be interpolated linearily. This unnecessary option is no longer implemented (starting with 1.2-1).
nsim	number of samples to generate
seed	optional seed for the random number generator
bysim	logical, if TRUE a list with nsim elements is returned, each element is of length nrow(newdata) and contains one sample from the conditional distribution for each row of newdata. If FALSE, a list of length nrow(newdata) is returned, its ith element of length nsim contains nsim samples from the conditional distribution given newdata[i,].
...	additional arguments

Details

plot evaluates the transformation function over a grid of q values for all observations in newdata and plots these functions (according to type). predict evaluates the transformation function over a grid of q values for all observations in newdata and returns the result as a matrix (where _columns_ correspond to _rows_ in newdata, see examples). Lack of type = "mean" is a feature and not a bug.

Argument type defines the scale of the plots or predictions: type = "distribution" means the cumulative distribution function, type = "survivor" is the survivor function (one minus distribution function), type = "density" the absolute continuous or discrete density (depending on the response), type = "hazard", type = "cumhazard", and type = "odds" refers to the hazard (absolute continuous or discrete), cumulative hazard (defined as minus log-survivor function in both the absolute continuous and discrete cases), and odds (distribution divided by survivor) functions. The quantile function can be evaluated for probabilities prob by type = "quantile".

Note that the predict method for ctm objects requires all model coefficients to be specified in this unfitted model. simulate draws samples from object by numerical inversion of the quantile function.

Note that offsets are ALWAYS IGNORED when computing predictions. If you want the methods to pay attention to offsets, specify them as a variable in the model with fixed regression coefficient using the fixed argument in mlt.

More examples can be found in Hothorn (2018).

References

Torsten Hothorn (2020), Most Likely Transformations: The mlt Package, *Journal of Statistical Software*, **92**(1), 1–68, doi:10.18637/jss.v092.i01

Examples

```
library("survival")
op <- options(digits = 2)

### GBSG2 dataset
data("GBSG2", package = "TH.data")

### right-censored response
GBSG2$y <- with(GBSG2, Surv(time, cens))

### define Bernstein(log(time)) parameterisation
### of transformation function. The response
### is bounded (log(0) doesn't work, so we use log(1))
### support defines the support of the Bernstein polynomial
### and add can be used to make the grid wider (see below)
rvar <- numeric_var("y", bounds = c(0, Inf),
                    support = c(100, 2000))
rb <- Bernstein_basis(rvar, order = 6, ui = "increasing")
### dummy coding of menopausal status
hb <- as.basis(~ 0 + menostat, data = GBSG2)
### treatment contrast of hormonal treatment
xb <- as.basis(~ horTh, data = GBSG2, remove_intercept = TRUE)

### set-up and fit Cox model, stratified by menopausal status
m <- ctm(rb, interacting = hb, shifting = xb, todistr = "MinExtrVal")
fm <- mlt(m, data = GBSG2)

### generate grid for all three variables
### note that the response grid ranges between 1 (bounds[1])
### and 2000 (support[2])
(d <- mkgrid(m, n = 10))
### data.frame of menopausal status and treatment
nd <- do.call("expand.grid", d[-1])

### plot model on different scales, for all four combinations
### of menopausal status and hormonal treatment
typ <- c("distribution", "survivor", "density", "hazard",
         "cumhazard", "odds")
layout(matrix(1:6, nrow = 2))
nl <- sapply(typ, function(tp)
    ### K = 500 makes densities and hazards smooth
    plot(fm, newdata = nd, type = tp, col = 1:nrow(nd), K = 500))
legend("topleft", lty = 1, col = 1:nrow(nd),
       legend = do.call("paste", nd), bty = "n")

### plot calls predict, which generates a grid with K = 50
```

```
### response values
### note that a K x nrow(newdata) matrix is returned
### (for reasons explained in the next example)
predict(fm, newdata = nd, type = "survivor")

### newdata can take a list, and evaluates the survivor
### function on the grid defined by newdata
### using a linear array model formulation and is
### extremely efficient (wrt computing time and memory)
### d[1] (the response grid) varies fastest
### => the first dimension of predict() is always the response,
### not the dimension of the predictor variables (like one
### might expect)
predict(fm, newdata = d, type = "survivor")

### owing to this structure, the result can be quickly stored in
### a data frame as follows
cd <- do.call("expand.grid", d)
cd$surv <- c(S <- predict(fm, newdata = d, type = "survivor"))

### works for distribution functions
all.equal(1 - S, predict(fm, newdata = d, type = "distribution"))
### cumulative hazard functions
all.equal(-log(S), predict(fm, newdata = d, type = "cumhazard"))
### log-cumulative hazard functions (= trafo, for Cox models)
all.equal(log(-log(S)), predict(fm, newdata = d, type = "logcumhazard"))
all.equal(log(-log(S)), predict(fm, newdata = d, type = "trafo"))
### densities, hazards, or odds functions
predict(fm, newdata = d, type = "density")
predict(fm, newdata = d, type = "hazard")
predict(fm, newdata = d, type = "odds")
### and quantiles (10 and 20%)
predict(fm, newdata = d[-1], type = "quantile", prob = 1:2 / 10)

### note that some quantiles are only defined as intervals
### (> 2000, in this case). Intervals are returned as an "response"
### object, see ?R. Unfortunately, these can't be stored as array, so
### a data.frame is returned where the quantile varies first
p <- c(list(prob = 1:9/10), d[-1])
np <- do.call("expand.grid", p)
(Q <- predict(fm, newdata = d[-1], type = "quantile", prob = 1:9 / 10))
np$Q <- Q
np

### simulating from the model works by inverting the distribution
### function; some obs are right-censored at 2000
(s <- simulate(fm, newdata = nd, nsim = 3))
### convert to Surv
sapply(s, as.Surv)

### generate 3 parametric bootstrap samples from the model
tmp <- GBSG2[, c("menostat", "horTh")]
s <- simulate(fm, newdata = tmp, nsim = 3)
```

```
### refit the model using the simulated response
lapply(s, function(y) {
  tmp$y <- y
  coef(mlt(m, data = tmp))
})

options(op)
```

R *Response Variables*

Description

Represent a possibly censored or truncated response variable

Usage

```
R(object, ...)
## S3 method for class 'numeric'
R(object = NA, cleft = NA, cright = NA,
   tleft = NA, tright = NA, tol = sqrt(.Machine$double.eps),
   as.R.ordered = FALSE, as.R.interval = FALSE, ...)
## S3 method for class 'ordered'
R(object, cleft = NA, cright = NA, ...)
## S3 method for class 'integer'
R(object, cleft = NA, cright = NA, bounds = c(min(object), Inf), ...)
## S3 method for class 'factor'
R(object, ...)
## S3 method for class 'Surv'
R(object, as.R.ordered = FALSE, as.R.interval = FALSE, ...)
as.Surv(object)
## S3 method for class 'response'
as.Surv(object)
## S3 method for class 'response'
as.double(x, ...)
```

Arguments

object	A vector of (conceptually) exact measurements or an object of class response (for as.Surv) or a list.
x	same as object.
cleft	A vector of left borders of censored measurements
cright	A vector of right borders of censored measurements
tleft	A vector of left truncations
tright	A vector of right truncations

tol	Tolerance for checking if cleft < cright
bounds	Range of possible values for integers
as.R.ordered	logical, should numeric responses or right-censored (and possible left-truncated survival) times be coded as ordered factor? This leads to a parameterisation allowing to maximise the nonparametric maximum likelihood
as.R.interval	logical, should numeric responses be coded for the nonparametric maximum likelihood
...	other arguments, ignored except for tleft and tright to R.ordered and R.integer

Details

R is basically an extention of Surv for the representation of arbitrarily censored or truncated measurements at any scale. The storage.mode of object determines if models are fitted by the discrete likelihood (integers or factors) or the continuous likelihood (log-density for numeric objects). Interval-censoring is given by intervals (cleft, cright], also for integers and factors (see example below). Left- and right-truncation can be defined by the tleft and tright arguments. Existing Surv objects can be converted using R(Surv(...))$ and, in some cases, an as.Surv() method exists for representing response objects as Surv objects.

R applied to a list calls R for each of the list elements and returns a joint object.

More examples can be found in Hothorn (2018) and in vignette("tram", package = "tram").

References

Torsten Hothorn (2020), Most Likely Transformations: The mlt Package, *Journal of Statistical Software*, **92**(1), 1–68, doi:10.18637/jss.v092.i01

Examples

```
library("survival")

### randomly right-censored continuous observations
time <- as.double(1:9)
event <- rep(c(FALSE, TRUE), length = length(time))

Surv(time, event)
R(Surv(time, event))

### right-censoring, left-truncation
ltm <- 1:9 / 10
Surv(ltm, time, event)
R(Surv(ltm, time, event))

### interval-censoring
Surv(ltm, time, type = "interval2")
R(Surv(ltm, time, type = "interval2"))

### interval-censoring, left/right-truncation
lc <- as.double(1:4)
```

```
lt <- c(NA, NA, 7, 8)
rt <- c(NA, 9, NA, 10)
x <- c(3, NA, NA, NA)
rc <- as.double(11:14)
R(x, cleft = lt, cright = rt)
as.Surv(R(x, cleft = lt, cright = rt))
R(x, tleft = 1, cleft = lt, cright = rt)
R(x, tleft = 1, cleft = lt, cright = rt, tright = 15)
R(x, tleft = lc, cleft = lt, cright = rt, tright = rc)

### discrete observations: counts
x <- 0:9
R(x)
### partially interval-censored counts
rx <- c(rep(NA, 6), rep(15L, 4))
R(x, cright = rx)

### ordered factor
x <- gl(5, 2, labels = LETTERS[1:5], ordered = TRUE)
R(x)
### interval-censoring (ie, observations can have multiple levels)
lx <- ordered(c("A", "A", "B", "C", "D", "E"),
              levels = LETTERS[1:5], labels = LETTERS[1:5])
rx <- ordered(c("B", "D", "E", "D", "D", "E"),
              levels = LETTERS[1:5], labels = LETTERS[1:5])
R(rx, cleft = lx, cright = rx)

### facilitate nonparametric maximum likelihood
(y <- round(runif(10), 1))
R(y, as.R.ordered = TRUE)

R(Surv(time, event), as.R.ordered = TRUE)
R(Surv(ltm, time, event), as.R.ordered = TRUE)
```

Index