Package ‘mlt’

May 12, 2020

Title Most Likely Transformations

Version 1.2-0

Date 2020-05-11

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>.

Depends basefun (>= 1.0-5), variables (>= 1.0-2)

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

Suggests MASS, nnet, TH.data, multcomp

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

License GPL-2

Encoding UTF-8

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2020-05-12 12:50:02 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. (2018).

An introduction to the package is available in the mlt package vignette from package mlt.docreg (Hothorn, 2018).

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

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 (2018), Most Likely Transformations: The mlt Package, *Journal of Statistical Software*, forthcoming. URL: https://cran.r-project.org/package=mlt.docreg.

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, data = NULL,
    todistr = c("Normal", "Logistic", "MinExtrVal", "MaxExtrVal", "Exponential"),
     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`

`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

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) 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.

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"),
              ...)
## 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 weights
`offset`	an optional vector of offset values
`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
`optim`	a list of functions implementing suitable optimisers
`...`	additional arguments, currently ignored

Details

This function fits a conditional transformation model by searching for the most likely transformation as described in Hothorn et al. (2017).

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.

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(object, parm = coef(object, fixed = FALSE),
       w = NULL, newdata, ...)
## 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),
       ...)
## S3 method for class 'mlt'
as.mlt(object)
```

Arguments

`object`	a fitted conditional transformation model as returned by `mlt`
`fixed`	a logical indicating if only estimated coefficients (`fixed = FALSE`) should be returned
`value`	coefficients to be assigned to the model
`parm`	model parameters
`w`	model weights
`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 gradient (sum of the score contributions) and `estfun` the 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 = NULL, 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 optimiser. This is still experimental and thus switched off (defaulting to NULL).
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.

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", "survivor", "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",
        "survivor", "density", "logdensity", "hazard", "loghazard", "cumhazard",
         "logcumhazard", "odds", "logodds", "quantile"),
         terms = c("bresponse", "binteracting", "bshifting"),
         q = NULL, prob = NULL, K = 50, interpolate = TRUE, ...)
## S3 method for class 'mlt'
predict(object, newdata = object$data, ...)
## S3 method for class 'ctm'
simulate(object, nsim = 1, seed = NULL, newdata, K = 50, q = NULL,
         interpolate = TRUE, bysim = TRUE, ...)
## S3 method for class 'mlt'
simulate(object, nsim = 1, seed = NULL, newdata = object$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 linearly

`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`). 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`.

R *Response Variable*

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), ...)
## 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.Surv(object)
## S3 method for class 'response'
as.Surv(object)
```

Arguments

`object` A vector of (conceptually) exact measurements or an object of class `response` (for `as.Surv`) or a list.

`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
`...`	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.

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

Examples

```
### ordered factor
R(gl(3, 3, labels = LETTERS[1:3]))
```

Index