Package 'crch'

January 27, 2018

Title Censored Regression with Conditional Heteroscedasticity

Version 1.0-1

Date 2018-01-19

Depends R (>= 2.10.0)

Imports stats, Formula, ordinal, sandwich, scoringRules

Suggests glmx, lmtest, memisc

Description Different approaches to censored or truncated regression with conditional heteroscedasticity are provided. First, continuous distributions can be used for the (right and/or left censored or truncated) response with separate linear predictors for the mean and variance. Second, cumulative link models for ordinal data (obtained by interval-censoring continuous data) can be employed for heteroscedastic extended logistic regression (HXLR). In the latter type of models, the intercepts depend on the thresholds that define the intervals.

License GPL-2 | GPL-3

NeedsCompilation yes

Author Jakob Messner [aut, cre] (<https://orcid.org/0000-0002-1027-3673>), Achim Zeileis [aut] (<https://orcid.org/0000-0003-0918-3766>), Reto Stauffer [aut] (<https://orcid.org/0000-0002-3798-5507>)

Maintainer Jakob Messner <jwmm@elektro.dtu.dk>

Repository CRAN

Date/Publication 2018-01-23 13:41:57

R topics documented:

`clogis` *The Censored Logistic Distribution*

Description

Density, distribution function, quantile function, and random generation for the left and/or right censored logistic distribution.

Usage

```
dclogis(x, location = 0, scale = 1, left = -Inf, right = Inf, log = FALSE)

pclogis(q, location = 0, scale = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)

rclogis(n, location = 0, scale = 1, left = -Inf, right = Inf)

qclogis(p, location = 0, scale = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)
```

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.
`location`	location parameter.
`scale`	scale parameter.
`left`	left censoring point.
`right`	right censoring point.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].

Details

If `location` or `scale` are not specified they assume the default values of `0` and `1`, respectively. `left` and `right` have the defaults `-Inf` and `Inf` respectively.

The censored logistic distribution has density $f(x)$:

$$\begin{array}{ll} \Lambda((left-\mu)/\sigma) & \text{if } x \le left \\ 1-\Lambda((right-\mu)/\sigma) & \text{if } x \ge right \\ \lambda((x-\mu)/\sigma)/\sigma & \text{if } left < x < right \end{array}$$

where Λ and λ are the cumulative distribution function and probability density function of the standard logistic distribution respectively, μ is the location of the distribution, and σ the scale.

Value

`dclogis` gives the density, `pclogis` gives the distribution function, `qclogis` gives the quantile function, and `rclogis` generates random deviates.

See Also

`dlogis`

`cnorm` *The Censored Normal Distribution*

Description

Density, distribution function, quantile function, and random generation for the left and/or right censored normal distribution.

Usage

```
dcnorm(x, mean = 0, sd = 1, left = -Inf, right = Inf, log = FALSE)

pcnorm(q, mean = 0, sd = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)

rcnorm(n, mean = 0, sd = 1, left = -Inf, right = Inf)

qcnorm(p, mean = 0, sd = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)
```

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.
mean	vector of means.
sd	vector of standard deviations.
left	left censoring point.
right	right censoring point.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].

Details

If mean or sd are not specified they assume the default values of 0 and 1, respectively. left and right have the defaults -Inf and Inf respectively.

The censored normal distribution has density $f(x)$:

$$\begin{array}{ll} \Phi((left-\mu)/\sigma) & \text{if } x \leq left \\ 1-\Phi((right-\mu)/\sigma) & \text{if } x \geq right \\ \phi((x-\mu)/\sigma)/\sigma & \text{if } left < x < right \end{array}$$

where Φ and ϕ are the cumulative distribution function and probability density function of the standard normal distribution respectively, μ is the mean of the distribution, and σ the standard deviation.

Value

dcnorm gives the density, pcnorm gives the distribution function, qcnorm gives the quantile function, and rcnorm generates random deviates.

See Also

dnorm

---

coef.crch *Methods for CRCH Objects*

Description

Methods for extracting information from fitted crch objects.

Usage

```
## S3 method for class 'crch'
coef(object, model = c("full", "location", "scale", "df"), ...)
## S3 method for class 'crch'
vcov(object, model = c("full", "location", "scale", "df"), ...)
```

```
## S3 method for class 'crch'
terms(x, model = c("location", "scale", "full"), ...)
## S3 method for class 'crch'
fitted(object, type = c("location", "scale"), ...)
```

Arguments

object, x	an object of class "crch".
model	model for which coefficients shall be returned.
type	type of fitted values.
...	further arguments passed to or from other methods.

Details

In addition to the methods above, a set of standard extractor functions for "crch" objects is available, including methods to the generic functions print, summary, logLik, and residuals.

See Also

crch

---

coef.crch.boost *Methods for boosted CRCH Objects*

Description

Methods for extracting information from fitted crch.boost objects.

Usage

```
## S3 method for class 'crch.boost'
coef(object, model = c("full", "location", "scale", "df"),
  mstop = NULL, zero.coefficients = FALSE, standardize = FALSE, ...)
## S3 method for class 'crch.boost'
print(x, digits = max(3, getOption("digits") - 3),
  mstop = NULL, zero.coefficients = FALSE, ...)
## S3 method for class 'crch.boost'
summary(object, mstop = NULL, zero.coefficients = FALSE, ...)
## S3 method for class 'crch.boost'
logLik(object, mstop = NULL, ...)
```

Arguments

object, x	an object of class "crch.boost".
model	model for which coefficients shall be returned.
mstop	stopping iteration for which coefficients shall be returned. Can be either a character ("max", "aic", "bic", "cv") or a numeric value.
zero.coefficients	logical whether zero coefficients are returned.
standardize	logical whether coefficients shall be standardized.
digits	the number of significant digits to use when printing.
...	further arguments passed to or from other methods.

Details

In addition to the methods above, the "crch" methods terms, model.frame, model.matrix, residuals, and fitted can be used also for "crch.boost" objects .

See Also

crch.boost,coef.crch

---

coef.hxlr *Methods for HXLR Objects*

Description

Methods for extracting information from fitted hxlr objects.

Usage

```
## S3 method for class 'hxlr'
coef(object, model = c("full", "intercept", "location", "scale"),
  type = c("CLM", "latent"), ...)
## S3 method for class 'hxlr'
vcov(object, model = c("full", "intercept", "location", "scale"),
  type = c("CLM", "latent"), ...)
## S3 method for class 'hxlr'
terms(x, model = c("full", "location", "scale"), ...)
```

Arguments

object, x	an object of class "hxlr".
model	model for which coefficients shall be returned.
type	type of coefficients. Default are CLM type coefficients. For type "latent" coefficients are converted in coefficients for location and scale of the latent distribution (analog to crch models).
...	further arguments passed to or from other methods.

Details

In addition to the methods above, a set of standard extractor functions for "hxlr" objects is available, including methods to the generic functions print, summary, and logLik.

See Also

hxlr

---

crch *Censored Regression with Conditional Heteroscedasticy*

Description

Fitting censored (tobit) or truncated regression models with conditional heteroscedasticy.

Usage

```
crch(formula, data, subset, na.action, weights, offset,
  link.scale = c("log", "identity", "quadratic"),
  dist = c("gaussian", "logistic", "student"), df = NULL,
  left = -Inf, right = Inf, truncated = FALSE,
  type = c("ml", "crps"), control = crch.control(...),
  model = TRUE, x = FALSE, y = FALSE, ...)

trch(formula, data, subset, na.action, weights, offset,
  link.scale = c("log", "identity", "quadratic"),
  dist = c("gaussian", "logistic", "student"), df = NULL,
  left = -Inf, right = Inf, truncated = TRUE,
  type = c("ml", "crps"), control = crch.control(...),
  model = TRUE, x = FALSE, y = FALSE, ...)

crch.fit(x, z, y, left, right, truncated = FALSE, dist = "gaussian",
  df = NULL, link.scale = "log", type = "ml", weights = NULL, offset = NULL,
  control = crch.control())
```

Arguments

formula	a formula expression of the form y ~ x | z where y is the response and x and z are regressor variables for the location and the scale of the fitted distribution respectively.
data	an optional data frame containing the variables occurring in the formulas.
subset	an optional vector specifying a subset of observations to be used for fitting.
na.action	a function which indicates what should happen when the data contain NAs.
weights	optional case weights in fitting.

offset	optional numeric vector with *a priori* known component to be included in the linear predictor for the location. For crch.fit, offset can also be a list of 2 offsets used for the location and scale respectively.
link.scale	character specification of the link function in the scale model. Currently, "identity", "log", "quadratic" are supported. The default is "log". Alternatively, an object of class "link-glm" can be supplied.
dist	assumed distribution for the dependent variable y.
df	optional degrees of freedom for dist="student". If omitted the degrees of freedom are estimated.
left	left limit for the censored dependent variable y. If set to -Inf, y is assumed not to be left-censored.
right	right limit for the censored dependent variable y. If set to Inf, y is assumed not to be right-censored.
truncated	logical. If TRUE truncated model is fitted with left and right interpreted as truncation points, If FALSE censored model is fitted. Default is FALSE
type	loss function to be optimized. Can be either "ml" for maximum likelihood (default) or "crps" for minimum continuous ranked probability score (CRPS).
control	a list of control parameters passed to optim or to the internal boosting algorithm if control=crch.boost(). Default is crch.control().
model	logical. If TRUE *model frame* is included as a component of the returned value.
x, y	for crch: logical. If TRUE the model matrix and response vector used for fitting are returned as components of the returned value. for crch.fit: x is a design matrix with regressors for the location and y is a vector of observations.
z	a design matrix with regressors for the scale.
...	arguments to be used to form the default control argument if it is not supplied directly.

Details

crch fits censored (tobit) or truncated regression models with conditional heteroscedasticy with maximum likelihood estimation. Student-t, Gaussian, and logistic distributions can be fitted to left- and/or right censored or truncated responses. Different regressors can be used to model the location and the scale of this distribution. If control=crch.boost() optimization is performed by boosting.

trch is a wrapper function for crch with default truncated = TRUE.

crch.fit is the lower level function where the actual fitting takes place.

Value

An object of class "crch" or "crch.boost", i.e., a list with the following elements.

coefficients	list of coefficients for location, scale, and df. Scale and df coefficients are in log-scale.
df	if dist = "student": degrees of freedom of student-t distribution. else NULL.
residuals	the residuals, that is response minus fitted values.

fitted.values	list of fitted location and scale parameters.
dist	assumed distribution for the dependent variable y.
cens	list of censoring points.
optim	output from optimization from optim.
method	optimization method used for optim.
type	used loss function (maximum likelihood or minimum CRPS).
control	list of control parameters passed to optim
start	starting values of coefficients used in the optimization.
weights	case weights used for fitting.
offset	list of offsets for location and scale.
n	number of observations.
nobs	number of observations with non-zero weights.
loglik	log-likelihood.
vcov	covariance matrix.
link	a list with element "scale" containing the link objects for the scale model.
truncated	logical indicating wheter a truncated model has been fitted.
converged	logical variable whether optimization has converged or not.
iterations	number of iterations in optimization.
call	function call.
formula	the formula supplied.
terms	the terms objects used.
levels	list of levels of the factors used in fitting for location and scale respectively.
contrasts	(where relevant) the contrasts used.
y	if requested, the response used.
x	if requested, the model matrix used.
model	if requested, the model frame used.
stepsize, mstop, mstopopt, standardize	return values of boosting optimization. See crch.boost for details.

References

Messner JW, Mayr GJ, Zeileis A (2016). Heteroscedastic Censored and Truncated Regression with crch. *The R Journal*, **3**(1), 173–181. https://journal.R-project.org/archive/2016-1/messner-mayr-zeileis.pdf.

Messner JW, Zeileis A, Broecker J, Mayr GJ (2014). Probabilistic Wind Power Forecasts with an Inverse Power Curve Transformation and Censored Regression. *Wind Energy*, **17**(11), 1753–1766. doi: 10.1002/we.1666.

See Also

predict.crch, crch.control, crch.boost

Examples

```
data("RainIbk")
## mean and standard deviation of square root transformed ensemble forecasts
RainIbk$sqrtensmean <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]), 1, mean)
RainIbk$sqrtenssd <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]), 1, sd)

## fit linear regression model with Gaussian distribution
CRCH <- crch(sqrt(rain) ~ sqrtensmean, data = RainIbk, dist = "gaussian")
## same as lm?
all.equal(coef(lm(sqrt(rain) ~ sqrtensmean, data = RainIbk)),
  head(coef(CRCH), -1))

## print
CRCH
## summary
summary(CRCH)

## left censored regression model with censoring point 0:
CRCH2 <- crch(sqrt(rain) ~ sqrtensmean, data = RainIbk,
  dist = "gaussian", left = 0)

## left censored regression model with censoring point 0 and
## conditional heteroscedasticy:
CRCH3 <- crch(sqrt(rain) ~ sqrtensmean|sqrtenssd, data = RainIbk,
  dist = "gaussian",  left = 0)

## left censored regression model with censoring point 0 and
## conditional heteroscedasticy with logistic distribution:
CRCH4 <- crch(sqrt(rain) ~ sqrtensmean|sqrtenssd, data = RainIbk,
  dist = "logistic", left = 0)

## compare AIC
AIC(CRCH, CRCH2, CRCH3, CRCH4)
```

---

crch.boost *Auxiliary functions to fit* crch *models via boosting.*

Description

Auxiliary functions to fit crch models via boosting

Usage

```
crch.boost(maxit = 100, nu = 0.1, start = NULL, dot = "separate",
  mstop = c("max", "aic", "bic", "cv"),  nfolds = 10, foldid = NULL,
  maxvar = NULL)
```

```
crch.boost.fit(x, z, y, left, right, truncated = FALSE, dist = "gaussian",
  df = NULL, link.scale = "log", type = "ml", weights = NULL, offset = NULL,
  control = crch.boost())
```

Arguments

maxit	the maximum number of boosting iterations.
nu	boosting step size. Default is 0.1.
start	a previously boosted but not converged "crch.boost" object to continue.
dot	character specifying how to process formula parts with a dot (.) on the right-hand side. This can either be "separate" so that each formula part is expanded separately or "sequential" so that the parts are expanded sequentially conditional on all prior parts. Default is "separate"
mstop	method to find optimum stopping iteration. Default is "max" which is maxit. Alternatives are "aic" and "bic" for AIC and BIC optimization and "cv" for cross validation. mstop can also be a positive integer to set the number of boosting iterations. Then maxit is set to mstop and mstop="max".
nfolds	if mstopopt = "cv", number of folds in cross validation.
foldid	if mstopopt = "cv", an optional vector of values between 1 and nfold identifying the fold each observation is in. If supplied, nfolds can be missing.
maxvar	Positive numeric. Maximum number of parameters to be selected during each iteration (not including intercepts). Used for stability selection.
x, z, y, left, right, truncated, dist, df, link.scale, type, weights, offset, control	see crch.fit for details.

Details

crch.boost extends crch to fit censored (tobit) or truncated regression models with conditional heteroscedasticy by boosting. If crch.boost() is supplied as control in crch then crch.boost.fit is used as lower level fitting function. Note that crch.control() with method=boosting is equivalent to crch.boost(). Thus, boosting can more conveniently be called with crch(..., method = "boosting").

Value

For crch.boost: A list with components named as the arguments. For crch.boost.fit: An object of class "crch.boost", i.e., a list with the following elements.

coefficients	list of coefficients for location and scale. Scale coefficients are in log-scale. Coefficients are of optimum stopping stopping iteration specified by mstop.
df	if dist = "student": degrees of freedom of student-t distribution. else NULL.
residuals	the residuals, that is response minus fitted values.
fitted.values	list of fitted location and scale parameters at optimum stopping iteration specified by mstop.
dist	assumed distribution for the dependent variable y.
cens	list of censoring points.

control	list of control parameters.
weights	case weights used for fitting.
offset	list of offsets for location and scale.
n	number of observations.
nobs	number of observations with non-zero weights.
loglik	log-likelihood.
link	a list with element "scale" containing the link objects for the scale model.
truncated	logical indicating wheter a truncated model has been fitted.
iterations	number of boosting iterations.
stepsize	boosting stepsize nu.
mstop	criterion used to find optimum stopping iteration.
mstopopt	optimum stopping iterations for different criteria.
standardize	list of center and scale values used to standardize response and regressors.

References

Messner JW, Mayr GJ, Zeileis A (2016). Non-Homogeneous Boosting for Predictor Selection in Ensemble Post-Processing. *Working Papers*, Faculty of Economics and Statistics, University of Innsbruck, url:http://econpapers.repec.org/paper/innwpaper/2016-04.htm.

See Also

crch, crch.control

Examples

```
# generate data
set.seed(5)
x <- matrix(rnorm(1000*20),1000,20)
y <- rnorm(1000, 1 + x[,1] - 1.5 * x[,2], exp(-1 + 0.3*x[,3]))
y <- pmax(0, y)
data <- data.frame(cbind(y, x))

# fit model with maximum likelihood
CRCH <- crch(y ~ .|., data = data, dist = "gaussian", left = 0)

# fit model with boosting
boost <- crch(y ~ .|.,  data = data, dist = "gaussian", left = 0,
  control = crch.boost(mstop = "aic"))

# more conveniently, the same model can also be fit through
# boost <- crch(y ~ .|.,  data = data, dist = "gaussian", left = 0,
#   method = "boosting", mstop = "aic")

# AIC comparison
AIC(CRCH, boost)
```

```
# summary
summary(boost)

# plot
plot(boost)
```

---

crch.control *Auxiliary Function for Controlling crch Fitting*

Description

Auxiliary function for crch fitting. Specifies a list of values passed to optim.

Usage

```
crch.control(method = "BFGS", maxit = NULL, hessian = NULL,
  trace = FALSE, start = NULL, dot = "separate",
  lower = -Inf, upper = Inf, ...)
```

Arguments

method	optimization method passed to optim
maxit	the maximum number of iterations. Default is 5000 except for method="boosting" where the default is 100.
hessian	logical or NULL. If TRUE the numerical Hessian matrix from the optim output is used for estimation of the covariance matrix. If FALSE no covariance matrix is computed. If NULL (the default) the Hessian matrix is computed analytically for dist="gaussian", dist="logistic", and dist="student" with predefined df. For dist="student" without prespecified df, no analytical solution is available and a numerical Hessian matrix is forced.
trace	non-negative integer. If positive, tracing information on the progress of the optimization is produced.
start	initial values for the parameters to be optimized over.
dot	character specifying how to process formula parts with a dot (.) on the right-hand side. This can either be "separate" so that each formula part is expanded separately or "sequential" so that the parts are expanded sequentially conditional on all prior parts. Default is "separate"
lower, upper	bounds on the variables for the "L-BFGS-B" method, or bounds in which to search for method "Brent".
...	additional parameters passed to optim.

Value

A list with components named as the arguments.

See Also

crch, optim

crch.stabsel *Auxiliary functions to perform stability selection using boosting.*

Description

Auxilirary function which allows to do stability selection on heteroscedastic crch models based on crch.boost.

Usage

```
crch.stabsel(formula, data, ..., nu = 0.1, q, B = 100, thr = 0.9,
  maxit = 2000, data_percentage = 0.5)
```

Arguments

formula	a formula expression of the form y ~ x | z where y is the response and x and z are regressor variables for the location and the scale of the fitted distribution respectively.
data	an optional data frame containing the variables occurring in the formulas.
...	Additional attributes to control the crch model. Note that control is *not* allowed; crch.stabsel uses crch.boost by default.
nu	Boosting step size (see crch.boost) default is 0.1 as for crch.boost while lower values might yield better results frequently and should be considered.
q	Positive numeric. Maximum number of parameters to be selected during each iteration (not including intercepts).
B	numeric, total number of iterations.
thr	numeric threshold ((0.5-1.0)). Used to generate the new formula and the computation of the per-family error rate.
maxit	Positive numeric value. Maximum number for the boosting algorithm. If q is not reached before maxit the algorithm will stop.
data_percentage	Percentage of data which should be sampled in each of the iterations. Default (and suggested) is 0.5.

Details

crch.boost allows to perform gradient boosting on heteroscedastic additive models. crch.stabsel is a wrapper around the core crch.boost algorithm to perform stability selection (see references).

Half of the data set (data) is sampled B times to perform boosting (based on crch.boost). Rather than perform the boosting iterations until a certain stopping criterion is reached (e.g., maximum number of iterations maxit) the algorithm stops as soon as q parameters have been selected. The number of parameters is computed across both parameters location and scale. Intercepts are not counted.

Value

Returns an object of class "stabsel.crch" containing the stability selection summary and the new formula based on the stability selection.

table	A table object containing the parameters which have been selected and the corresponding frequency of selection.
formula.org	Original formula used to perform the stability selection.
formula.new	New formula based including the coefficients selected during stability selection.
family	A list object which contains the distribution-specification from the crch.stabsel call including: dist, cens, and truncated.
parameter	List with the parameters used to perform the stability selection including q, B, thr, p, and PFER (per-family error rate).

References

Meinhausen N, Buehlmann P (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **72**(4), 417–473. doi: 10.1111/j.14679868.2010.00740.x.

See Also

crch, crch.boost

Examples

```
# generate data
set.seed(5)
x <- matrix(rnorm(1000*20),1000,20)
y <- rnorm(1000, 1 + x[,1] - 1.5 * x[,2], exp(-1 + 0.3*x[,3]))
y <- pmax(0, y)
data <- data.frame(cbind(y, x))

# fit model with maximum likelihood
CRCH1 <- crch(y ~ .|., data = data, dist = "gaussian", left = 0)

# Perform stability selection
stabsel <- crch.stabsel(y ~ .|.,  data = data, dist = "gaussian", left = 0,
           q = 8, B = 5)

# Show stability selection summary
print(stabsel); plot(stabsel)

CRCH2 <- crch(stabsel$formula.new, data = data, dist = "gaussian", left = 0 )
BOOST <- crch(stabsel$formula.new, data = data, dist = "gaussian", left = 0,
              control = crch.boost() )

### AIC comparison
sapply( list(CRCH1,CRCH2,BOOST), logLik )
```

ct *The Censored Student-t Distribution*

Description

Density, distribution function, quantile function, and random generation for the left and/or right censored student-t distribution with `df` degrees of freedom.

Usage

```
dct(x, location = 0, scale = 1, df, left = -Inf, right = Inf, log = FALSE)

pct(q, location = 0, scale = 1, df, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)

rct(n, location = 0, scale = 1, df, left = -Inf, right = Inf)

qct(p, location = 0, scale = 1, df, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)
```

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.
`location`	location parameter.
`scale`	scale parameter.
`df`	degrees of freedom (> 0, maybe non-integer). `df = Inf` is allowed.
`left`	left censoring point.
`right`	right censoring point.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].

Details

If `location` or `scale` are not specified they assume the default values of `0` and `1`, respectively. `left` and `right` have the defaults `-Inf` and `Inf` respectively.

The censored student-t distribution has density $f(x)$:

$$\begin{array}{ll} T((left-\mu)/\sigma) & \text{if } x \le left \\ 1-T((right-\mu)/\sigma) & \text{if } x \ge right \\ \tau((x-\mu)/\sigma)/\sigma & \text{if } left < x < right \end{array}$$

where T and τ are the cumulative distribution function and probability density function of the student-t distribution with `df` degrees of freedom respectively, μ is the location of the distribution, and σ the scale.

Value

`dct` gives the density, `pct` gives the distribution function, `qct` gives the quantile function, and `rct` generates random deviates.

See Also

`dt`

`hxlr` *Heteroscedastic Extended Logistic Regression*

Description

This is a wrapper function for `clm` (from package **ordinal**) to fit (heteroscedastic) extended logistic regression (HXLR) models (Messner et al. 2013).

Usage

```
hxlr(formula, data, subset, na.action, weights, thresholds, link, control, ...)
```

Arguments

`formula`	a formula expression of the form `y ~ x | z` where `y` is the response and `x` and `z` are regressor variables for the location and the scale of the latend distribution respectively. Response can either be a continuous variable or a factor.
`data`	an optional data frame containing the variables occurring in the formulas.
`subset`	an optional vector specifying a subset of observations to be used for fitting.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Default is na.omit
`weights`	optional case weights in fitting.
`thresholds`	vector of (transformed) thresholds that are used to cut the continuous response into categories. Data frames or matrices with multiple columns are allowed as well. Then each column is used as separate predictor variable for the intercept model.
`link`	link function, i.e., the type of location-scale distribution assumed for the latent distribution. Default is `logit`.
`control`	a list of control parameters passed to `optim`. Default is `hxlr.control`
`...`	arguments to be used to form the default `control` argument if it is not supplied directly.

Details

Extended logistic regression (Wilks 2009) extends binary logistic regression to multi-category responses by including the thresholds, that are used to cut a continuous variable into categories, in the regression equation. Heteroscedastic extended logistic regression (Messner et al. 2013) extends this model further and allows to add additional predictor variables that are used to predict the scale of the latent logistic distribution.

Value

An object of class `"hxlr"`, i.e., a list with the following elements.

`coefficients`	list of CLM coefficients for intercept, location, and scale model.
`fitted.values`	list of fitted location and scale parameters.
`optim`	output from optimization from `optim`.
`method`	Optimization method used for `optim`.
`control`	list of control parameters passed to `optim`
`start`	starting values of coefficients used in the optimization.
`weights`	case weights used for fitting.
`n`	number of observations.
`nobs`	number of observations with non-zero weights.
`loglik`	log-likelihood.
`vcov`	covariance matrix.
`converged`	logical variable whether optimization has converged or not.
`iterations`	number of iterations in optimization.
`call`	function call.
`scale`	the formula supplied.
`terms`	the `terms` objects used.
`levels`	list of levels of the factors used in fitting for location and scale respectively.
`thresholds`	the thresholds supplied.

References

Messner JW, Mayr GJ, Zeileis A, Wilks DS (2014). Extending Extended Logistic Regression to Effectively Utilize the Ensemble Spread. *Monthly Weather Review*, **142**, 448–456. doi: 10.1175/MWRD1300271.1.

Wilks DS (2009). Extending Logistic Regression to Provide Full-Probability-Distribution MOS Forecasts. *Meteorological Applications*, **368**, 361–368.

See Also

`predict.hxlr`, `clm`

Examples

```
data("RainIbk")
## mean and standard deviation of square root transformed ensemble forecasts
RainIbk$sqrtensmean <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]), 1, mean)
RainIbk$sqrtenssd <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]), 1, sd)

## climatological deciles
q <- unique(quantile(RainIbk$rain, seq(0.1, 0.9, 0.1)))

## fit ordinary extended logistic regression with ensemble mean as
## predictor variable
XLR <- hxlr(sqrt(rain) ~ sqrtensmean, data = RainIbk, thresholds = sqrt(q))
## print
XLR
## summary
summary(XLR)


## fit ordinary extended logistic regression with ensemble mean
## and standard deviation as predictor variables
XLRS <- hxlr(sqrt(rain) ~ sqrtensmean + sqrtenssd, data = RainIbk,
  thresholds = sqrt(q))
## fit heteroscedastic extended logistic regression with ensemble
## standard deviation as predictor for the scale
HXLR <- hxlr(sqrt(rain) ~ sqrtensmean | sqrtenssd, data = RainIbk,
  thresholds = sqrt(q))

## compare AIC of different models
AIC(XLR, XLRS, HXLR)

## XLRS and HXLR are nested in XLR -> likelihood-ratio-tests
if(require("lmtest")) {
  lrtest(XLR, XLRS)
  lrtest(XLR, HXLR)
}

## Not run:
######################################################################
## Cross-validation and bootstrapping RPS for different models
## (like in Messner 2013).
N <- NROW(RainIbk)
## function that returns model fits
fits <- function(data, weights = rep(1, N)) {
  list(
    "XLR"    = hxlr(sqrt(rain) ~ sqrtensmean, data = data,
      weights = weights, thresholds = sqrt(q)),
    "XLR:S"  = hxlr(sqrt(rain) ~ sqrtensmean + sqrtenssd, data = data,
      weights = weights, thresholds = sqrt(q)),
    "XLR:SM" = hxlr(sqrt(rain) ~ sqrtensmean + I(sqrtensmean*sqrtenssd),
      data = data, weights = weights, thresholds = sqrt(q)),
```

```
    "HXLR"   = hxlr(sqrt(rain) ~ sqrtensmean | sqrtenssd, data = data,
      weights = weights, thresholds = sqrt(q)),
    "HXLR:S" = hxlr(sqrt(rain) ~ sqrtensmean + sqrtenssd | sqrtenssd,
      data = data, weights = weights, thresholds = sqrt(q))
  )
}


## cross validation
id <- sample(1:10, N, replace = TRUE)
obs <- NULL
pred <- list(NULL)
for(i in 1:10) {
  ## splitting into test and training data set
  trainIndex <- which(id != i)
  testIndex <- which(id == i)
  ## weights that are used for fitting the models
  weights <- as.numeric(table(factor(trainIndex, levels = c(1:N))))
  ## testdata
  testdata <- RainIbk[testIndex,]
  ## observations
  obs <- c(obs, RainIbk$rain[testIndex])
  ## estimation
  modelfits <- fits(RainIbk, weights)
  ## Prediction
  pred2 <- lapply(modelfits, predict, newdata = testdata, type = "cumprob")
  pred <- mapply(rbind, pred, pred2, SIMPLIFY = FALSE)
}
names(pred) <- c(names(modelfits))

## function to compute RPS
rps <- function(pred, obs) {
  OBS <- NULL
  for(i in 1:N)
    OBS <- rbind(OBS, rep(0:1, c(obs[i] - 1, length(q) - obs[i] + 1)))
  apply((OBS-pred)^2, 1, sum)
}

## compute rps
RPS <- lapply(pred, rps, obs = as.numeric(cut(obs, c(-Inf, q, Inf))))

## bootstrapping mean rps
rpsall <- NULL
for(i in 1:250) {
  index <- sample(length(obs), replace = TRUE)
  rpsall <- rbind(rpsall, sapply(RPS, function(x) mean(x[index])))
}

rpssall <- 1 - rpsall/rpsall[,1]
boxplot(rpssall[,-1], ylab = "RPSS", main = "RPSS relative to XLR")
abline(h = 0, lty = 2)

## End(Not run)
```

hxlr.control *Auxiliary Function for Controlling HXLR Fitting*

Description

Auxiliary function for `hxlr` fitting. Specifies a list of values passed to `optim`.

Usage

```
hxlr.control(method = "BFGS", maxit = 5000, hessian = TRUE,
  trace = FALSE, start = NULL, ...)
```

Arguments

`method`	optimization method used in `optim`
`maxit`	the maximum number of iterations.
`hessian`	logical. Should a numerically differentiated Hessian matrix be returned?
`trace`	non-negative integer. If positive, tracing information on the progress of the optimization is produced.
`start`	initial values for the parameters to be optimized over.
`...`	Additional parameters passed to `optim`.

Value

A list with components named as the arguments.

See Also

`hxlr`, `optim`

plot.crch.boost *Plot coefficient paths of boosted CRCH objects.*

Description

Plot paths of coefficients or log-likelihood contributions for `crch.boost` models.

Usage

```
## S3 method for class 'crch.boost'
plot(x, loglik = FALSE,
  standardize = TRUE, which = c("both", "location", "scale"),
  mstop = NULL, coef.label = TRUE, col = NULL, ...)
```

Arguments

x	an object of class "crch.boost".
loglik	logical whether log-likelihood contribution shall be plotted instead of coefficient value.
standardize	logical whether coefficients shall be standardized. Not used if loglik = TRUE
which	which coefficients: "location" and "scale" plots only the coefficients for the location and scale part of the model respectively. "both" plots the coefficient paths of both parts in one graph.
mstop	Stopping iteration for which a vertical line is plotted. Possible choices are "max", "aic", "bic", "cv", "all", or "no". Default is the stopping iteration used for fitting.
coef.label	logical whether paths shall be labeled.
col	Color(s) for the paths. If which="both" a vector of two colors where the paths for the location are plotted in the first color and for the scale in the second color.
...	further arguments passed to plot.ts.

See Also

crch.boost,plot.ts

---

predict.crch *Predicted/Fitted Values for CRCH Fits*

Description

Obtains various types of predictions for crch models.

Usage

```
## S3 method for class 'crch'
predict(object, newdata = NULL, type = c("location", "scale",
  "response", "parameter", "density", "probability", "quantile", "crps"),
  na.action = na.pass, at = 0.5, left = NULL, right = NULL, ...)
```

Arguments

object	an object of class "crch".
newdata	an optional data frame in which to look for variables with which to predict.
type	type of prediction: "location" returns the location of the predicted distribution. "scale" returns the scale of the predicted distribution. "response" returns the expected value of the predicted distribution (not equal to location for censored and truncated distributions). "parameter" returns a data frame with predicted location and scale parameters. "density" evaluates the predictive density at at. "probability" evaluates the predictive CDF at at. "quantile" returns a matrix of predicted quantiles with quantile probabilities at. "crps" returns the CRPS of the predictive distributions at at.

na.action	a function which indicates what should happen when the data contain NAs. Default is na.pass
at	a vector of values to evaluate the predictive density (type = "density"), probability (type = "probability"), or CRPS (type = "crps") or a vector of quantile probabilities used for type = "quantile". Alternatively, with at = "function" a function is returned that takes at as an argument.
left	left censoring or truncation point. Only used for type = "quantile". If NULL, censoring or truncation point is obtained from object.
right	right censoring or truncation point. Only used for type = "quantile". If NULL, censoring or truncation point is obtained from object.
...	further arguments passed to or from other methods.

Value

For type "response", "location", or "scale" a vector with either the location or the scale of the predicted distribution.

For type "quantile" a matrix of predicted quantiles each column corresponding to an element of at.

See Also

crch

---

predict.crch.boost *Predicted/Fitted Values for boosted CRCH Fits*

Description

Obtains various types of predictions for crch.boost models.

Usage

```
## S3 method for class 'crch.boost'
predict(object, newdata = NULL, mstop = NULL, ...)
```

Arguments

object	an object of class "crch.boost".
newdata	an optional data frame in which to look for variables with which to predict.
mstop	stopping iteration. Can be either a character ("max", "aic", "bic", "cv") or a numeric value. If not NULL, newdata has to be supplied.
...	further arguments passed to or from other methods.

Value

For type "response", "location", or "scale" a vector with either the location or the scale of the predicted distribution.

For type "quantile" a matrix of predicted quantiles each column corresponding to an element of at.

See Also

crch.boost,predict.crch

---

predict.hxlr *Predict/Fitted Values for HXLR Fits*

Description

Obtains various types of predictions/fitted values for heteroscedastic extended logistic regression (HXLR) models.

Usage

```
## S3 method for class 'hxlr'
predict(object, newdata = NULL, type = c("class", "probability",
  "cumprob", "location", "scale"), thresholds = object$thresholds,
  na.action = na.pass, ...)
## S3 method for class 'hxlr'
fitted(object, type = c("class", "probability",
  "cumprob", "location", "scale"), ...)
```

Arguments

object	an object of class "hxlr".
newdata	an optional data frame in which to look for variables with which to predict.
type	type of prediction: "probability" returns a data frame with category probabilities, "cumprob" returns cumulative probabilities, "location" and "scale" return the location and scale of the predicted latent distribution respectively, and "class" returns the category with the highest probability. Default is "class".
thresholds	optional thresholds used for defining the thresholds for types "probability", "cumprob", and "class". Can differ from thresholds used for fitting. If omitted, the same thresholds as for fitting are used.
na.action	A function which indicates what should happen when the data contain NAs. Default is na.pass
...	further arguments passed to or from other methods.

Value

For type "prob" a matrix with number of intervals (= number of thresholds + 1) columns is produced. Each row corresponds to a row in newdata and contains the predicted probabilities to fall in the corresponding interval.

For type "cumprob" a matrix with number of thresholds columns is produced. Each row corresponds to a row in newdata and contains the predicted probabilities to fall below the corresponding threshold.

For types "class", "location", and "scale" a vector is returned respectively with either the most probable categories ("class") or the location ("location") or scale (scale) of the latent distribution.

See Also

hxlr

RainIbk *Precipitation Observations and Forecasts for Innsbruck*

Description

Accumulated 5-8 days precipitation amount for Innsbruck. Data includes GEFS reforecasts (Hamill et al. 2013) and observations from SYNOP station Innsbruck Airport (11120) from 2000-01-01 to 2013-09-17.

Usage

```
data("RainIbk")
```

Format

A data frame with 4977 rows. The first column (rain) are 3 days accumulated precipitation amount observations, Columns 2-12 (rainfc) are 5-8 days accumulated precipitation amount forecasts from the individual ensemble members.

Source

Observations: http://www.ogimet.com/synops.phtml.en

Reforecasts: http://www.esrl.noaa.gov/psd/forecasts/reforecast2/

References

Hamill TM, Bates GT, Whitaker JS, Murray DR, Fiorino M, Galarneau Jr TJ, Zhu Y, Lapenta W (2013). NOAA's Second-Generation Global Medium-Range Ensemble Reforecast Data Set. *Bulletin of the American Meteorological Society*, 94(10), 1553-1565.

Examples

```
## Spread skill relationship ##

## load and prepare data
data(RainIbk)

## mean and standard deviation of square root transformed ensemble forecasts
RainIbk$sqrtensmean <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]), 1, mean)
RainIbk$sqrtenssd <-
  apply(sqrt(RainIbk[,grep('^rainfc',names(RainIbk))]),  1, sd)

## quintiles of sqrtenssd
sdcat <- cut(RainIbk$sqrtenssd, c(-Inf, quantile(RainIbk$sqrtenssd,
  seq(0.2,0.8,0.2)), Inf), labels = c(1:5))

## mean forecast errors for each quintile
m <- NULL
for(i in levels(sdcat)) {
  m <- c(m, mean((sqrt(RainIbk$rain)[sdcat == i] -
  RainIbk$sqrtensmean[sdcat == i])^2, na.rm = TRUE))
}

## plot
boxplot((sqrt(rain) - sqrtensmean)^2~sdcat, RainIbk,
  xlab = "Quintile of ensemble standard deviation",
  ylab = "mean squared error", main = "Spread skill relationship")
```

---

tlogis *The Truncated Logistic Distribution*

Description

Density, distribution function, quantile function, and random generation for the left and/or right truncated logistic distribution.

Usage

```
dtlogis(x, location = 0, scale = 1, left = -Inf, right = Inf, log = FALSE)

ptlogis(q, location = 0, scale = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)

rtlogis(n, location = 0, scale = 1, left = -Inf, right = Inf)

qtlogis(p, location = 0, scale = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)
```

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.
location	location parameter.
scale	scale parameter.
left	left truncation point.
right	right truncation point.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].

Details

If location or scale are not specified they assume the default values of 0 and 1, respectively. left and right have the defaults -Inf and Inf respectively.

The truncated logistic distribution has density

$$f(x) = 1/\sigma\lambda((x-\mu)/\sigma)/(\Lambda((right-\mu)/\sigma) - \Lambda((left-\mu)/\sigma))$$

for $left \leq x \leq right$, and 0 otherwise.

Λ and λ are the cumulative distribution function and probability density function of the standard logistic distribution respectively, μ is the location of the distribution, and σ the scale.

Value

dtlogis gives the density, ptlogis gives the distribution function, qtlogis gives the quantile function, and rtlogis generates random deviates.

See Also

dlogis

---

tnorm *The Truncated Normal Distribution*

Description

Density, distribution function, quantile function, and random generation for the left and/or right truncated normal distribution.

Usage

```
dtnorm(x, mean = 0, sd = 1, left = -Inf, right = Inf, log = FALSE)

ptnorm(q, mean = 0, sd = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)

rtnorm(n, mean = 0, sd = 1, left = -Inf, right = Inf)

qtnorm(p, mean = 0, sd = 1, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)
```

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.
mean	vector of means.
sd	vector of standard deviations.
left	left censoring point.
right	right censoring point.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].

Details

If mean or sd are not specified they assume the default values of 0 and 1, respectively. left and right have the defaults -Inf and Inf respectively.

The truncated normal distribution has density

$$f(x) = 1/\sigma\phi((x-\mu)/\sigma)/(\Phi((right-\mu)/\sigma) - \Phi((left-\mu)/\sigma))$$

for $left \leq x \leq right$, and 0 otherwise.

Φ and ϕ are the cumulative distribution function and probability density function of the standard normal distribution respectively, μ is the mean of the distribution, and σ the standard deviation.

Value

dtnorm gives the density, ptnorm gives the distribution function, qtnorm gives the quantile function, and rtnorm generates random deviates.

See Also

dnorm

tt *The Truncated Student-t Distribution*

Description

Density, distribution function, quantile function, and random generation for the left and/or right truncated student-t distribution with `df` degrees of freedom.

Usage

```
dtt(x, location = 0, scale = 1, df, left = -Inf, right = Inf, log = FALSE)

ptt(q, location = 0, scale = 1, df, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)

rtt(n, location = 0, scale = 1, df, left = -Inf, right = Inf)

qtt(p, location = 0, scale = 1, df, left = -Inf, right = Inf,
  lower.tail = TRUE, log.p = FALSE)
```

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.
`location`	location parameter.
`scale`	scale parameter.
`df`	degrees of freedom (> 0, maybe non-integer). `df = Inf` is allowed.
`left`	left censoring point.
`right`	right censoring point.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are P[X <= x] otherwise, P[X > x].

Details

If `location` or `scale` are not specified they assume the default values of `0` and `1`, respectively. `left` and `right` have the defaults `-Inf` and `Inf` respectively.

The truncated student-t distribution has density

$$f(x) = 1/\sigma\tau((x-\mu)/\sigma)/(T((right-\mu)/\sigma) - T((left-\mu)/\sigma))$$

for $left \leq x \leq right$, and 0 otherwise.

where T and τ are the cumulative distribution function and probability density function of the student-t distribution with `df` degrees of freedom respectively, μ is the location of the distribution, and σ the scale.

Value

`dtt` gives the density, `ptt` gives the distribution function, `qtt` gives the quantile function, and `rtt` generates random deviates.

See Also

`dt`

Index