Package 'ordinal'

March 9, 2019

Type Package

Title Regression Models for Ordinal Data

Version 2019.3-9

Date 2019-03-09

LazyData true

ByteCompile yes

Depends R (>= 2.13.0), stats, methods

Imports ucminf, MASS, Matrix, numDeriv

Suggests lme4, nnet, xtable, testthat (>= 0.8), tools

Description Implementation of cumulative link (mixed) models also known as ordered regression models, proportional odds models, proportional hazards models for grouped survival times and ordered logit/probit/... models. Estimation is via maximum likelihood and mixed models are fitted with the Laplace approximation and adaptive Gauss-Hermite quadrature. Multiple random effect terms are allowed and they may be nested, crossed or partially nested/crossed. Restrictions of symmetry and equidistance can be imposed on the thresholds (cut-points/intercepts). Standard model methods are available (summary, anova, drop-methods, step, confint, predict etc.) in addition to profile methods and slice methods for visualizing the likelihood function and checking convergence.

License GPL (>= 2)

NeedsCompilation yes

URL https://github.com/runehaubo/ordinal

BugReports https://github.com/runehaubo/ordinal/issues

Author Rune Haubo Bojesen Christensen [aut, cre]

Maintainer Rune Haubo Bojesen Christensen <rune.haubo@gmail.com>

Repository CRAN

Date/Publication 2019-03-09 12:40:03 UTC

R topics documented:

`ordinal-package` *Regression Models for Ordinal Data via Cumulative Link (Mixed) Models*

Description

This package facilitates analysis of ordinal (ordered categorical data) via cumulative link models (CLMs) and cumulative link mixed models (CLMMs). Robust and efficient computational methods gives speedy and accurate estimation. A wide range of methods for model fits aids the data analysis.

Details

Package:	ordinal
Type:	Package
License:	GPL (>= 2)
LazyLoad:	yes

This package implements cumualtive link models and cumulative link models with normally distributed random effects, denoted cumulative link mixed (effects) models. Cumulative link models are also known as ordered regression models, proportional odds models, proportional hazards models for grouped survival times and ordered logit/probit/... models.

Cumulative link models are fitted with `clm` and the main features are:

- A range of standard link functions are available.
- In addition to the standard location (additive) effects, scale (multiplicative) effects are also allowed.
- nominal effects are allowed for any subset of the predictors — these effects are also known as partial proportional odds effects when using the logit link.
- Restrictions can be imposed on the thresholds/cut-points, e.g., symmetry or equidistance.
- A (modified) Newton-Raphson algorithm provides the maximum likelihood estimates of the parameters. The estimation scheme is robust, fast and accurate.
- Rank-deficient designs are identified and unidentified coefficients exposed in `print` and `summary` methods as with `glm`.
- A suite of standard methods are available including `anova`, `add/drop`-methods, `step`, `profile`, `confint`.
- A `slice` method facilitates illustration of the likelihood function and a `convergence` method summarizes the accuracy of the model estimation.
- The `predict` method can predict probabilities, response class-predictions and cumulative probabilities, and it provides standard errors and confidence intervals for the predictions.

Cumulative link mixed models are fitted with `clmm` and the main features are:

- Any number of random effect terms can be included.
- The syntax for the model formula resembles that of `lmer` from the `lme4` package.
- Nested random effects, crossed random effects and partially nested/crossed random effects are allowed.
- Estimation is via maximum likelihood using the Laplace approximation or adaptive Gauss-Hermite quadrature (one random effect).
- Vector-valued and correlated random effects such as random slopes (random coefficient models) are fitted with the Laplace approximation.
- Estimation employs sparse matrix methods from the `Matrix` package.
- During model fitting a Newton-Raphson algorithm updates the conditional modes of the random effects a large number of times. The likelihood function is optimized with a general purpose optimizer.

A major update of the package in August 2011 introduced new and improved implementations of `clm` and `clmm`. The old implementations are available with `clm2` and `clmm2`. At the time of writing there is functionality in `clm2` and `clmm2` not yet available in `clm` and `clmm`. This includes flexible link functions (log-gamma and Aranda-Ordaz links) and a profile method for random effect variance parameters in CLMMs. The new implementations are expected to take over the old implementations at some point, hence the latter will eventually be `deprecated` and `defunct`.

Author(s)

Rune Haubo B Christensen

Maintainer: Rune Haubo B Christensen <rune.haubo@gmail.com>

Examples

```
## A simple cumulative link model:
fm1 <- clm(rating ~ contact + temp, data=wine)
summary(fm1)

## A simple cumulative link mixed model:
fmm1 <- clmm(rating ~ contact + temp + (1|judge), data=wine)
summary(fmm1)
```

---

anova.clm *ANODE Tables and Likelihood ratio test of cumulative link models*

Description

Type I, II, and III analysis of deviance (ANODE) tables for cumulative link models and comparison of cumulative link models with likelihood ratio tests. Models may differ by terms in location, scale and nominal formulae, in link, threshold function.

Usage

```
## S3 method for class 'clm'
anova(object, ..., type = c("I", "II", "III", "1", "2", "3"))
```

Arguments

object	a clm object.
...	optionally one or more additional clm objects.
type	the type of hypothesis test if anova is called with a single model; ignored if more than one model is passed to the method.

Details

The ANODE table returned when anova is called with a single model apply only to terms in formula, that is, terms in nominal and scale are ignored.

Value

An analysis of deviance table based on Wald chi-square test if called with a single model and a comparison of models with likelihood ratio tests if called with more than one model.

Author(s)

Rune Haubo B Christensen

See Also

clm

Examples

```
## Analysis of deviance tables with Wald chi-square tests:
fm <- clm(rating ~ temp * contact, scale=~contact, data=wine)
anova(fm, type="I")
anova(fm, type="II")
anova(fm, type="III")

options(contrasts = c("contr.treatment", "contr.poly"))
m1 <- clm2(SURENESS ~ PROD, scale = ~PROD, data = soup,
          link = "logistic")

## anova
anova(m1, update(m1, scale = ~.-PROD))
mN1 <- clm2(SURENESS ~ 1, nominal = ~PROD, data = soup,
           link = "logistic")
anova(m1, mN1)
anova(m1, update(m1, scale = ~.-PROD), mN1)

## Fit model from polr example:
if(require(MASS)) {
    fm1 <- clm2(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
    anova(fm1, update(fm1, scale =~ Cont))
}
```

---

clm *Cumulative Link Models*

Description

Fits cumulative link models (CLMs) such as the propotional odds model. The model allows for various link functions and structured thresholds that restricts the thresholds or cut-points to be e.g., equidistant or symmetrically arranged around the central threshold(s). Nominal effects (partial proportional odds with the logit link) are also allowed. A modified Newton algorithm is used to optimize the likelihood function.

Usage

```
clm(formula, scale, nominal, data, weights, start, subset, doFit = TRUE,
  na.action, contrasts, model = TRUE, control=list(),
  link = c("logit", "probit", "cloglog", "loglog", "cauchit",
           "Aranda-Ordaz", "log-gamma"),
  threshold = c("flexible", "symmetric", "symmetric2", "equidistant"), ...)
```

Arguments

formula	a formula expression as for regression models, of the form `response ~ predictors`. The response should be a factor (preferably an ordered factor), which will be interpreted as an ordinal response with levels ordered as in the factor. The model must have an intercept: attempts to remove one will lead to a warning and will be ignored. An offset may be used. See the documentation of `formula` for other details.
scale	an optional formula expression, of the form `~ predictors`, i.e. with an empty left hand side. An offset may be used. Variables included here will have multiplicative effects and can be interpreted as effects on the scale (or dispersion) of a latent distribution.
nominal	an optional formula of the form `~ predictors`, i.e. with an empty left hand side. The effects of the predictors in this formula are assumed to be nominal rather than ordinal - this corresponds to the so-called partial proportional odds (with the logit link).
data	an optional data frame in which to interpret the variables occurring in the formulas.
weights	optional case weights in fitting. Defaults to 1. Negative weights are not allowed.
start	initial values for the parameters in the format `c(alpha, beta, zeta)`, where `alpha` are the threshold parameters (adjusted for potential nominal effects), `beta` are the regression parameters and `zeta` are the scale parameters.
subset	expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.
doFit	logical for whether the model should be fitted or the model environment should be returned.
na.action	a function to filter missing data. Applies to terms in all three formulae.
contrasts	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
model	logical for whether the model frame should be part of the returned object.
control	a list of control parameters passed on to `clm.control`.
link	link function, i.e., the type of location-scale distribution assumed for the latent distribution. The default `"logit"` link gives the proportional odds model.
threshold	specifies a potential structure for the thresholds (cut-points). `"flexible"` provides the standard unstructured thresholds, `"symmetric"` restricts the distance between the thresholds to be symmetric around the central one or two thresholds for odd or equal numbers or thresholds respectively, `"symmetric2"` restricts the

latent mean in the reference group to zero; this means that the central threshold (even no. response levels) is zero or that the two central thresholds are equal apart from their sign (uneven no. response levels), and "equidistant" restricts the distance between consecutive thresholds to be of the same size.

... additional arguments are passed on to clm.control.

Details

This is a new (as of August 2011) improved implementation of CLMs. The old implementation is available in clm2, but will probably be removed at some point.

There are methods for the standard model-fitting functions, including summary, anova, model.frame, model.matrix, drop1, dropterm, step, stepAIC, extractAIC, AIC, coef, nobs, profile, confint, vcov and slice.

Value

If doFit = FALSE the result is an environment representing the model ready to be optimized. If doFit = TRUE the result is an object of class "clm" with the components listed below.

Note that some components are only present if scale and nominal are used.

aliased	list of length 3 or less with components alpha, beta and zeta each being logical vectors containing alias information for the parameters of the same names.
alpha	a vector of threshold parameters.
alpha.mat	(where relevant) a table (data.frame) of threshold parameters where each row corresponds to an effect in the nominal formula.
beta	(where relevant) a vector of regression parameters.
call	the mathed call.
coefficients	a vector of coefficients of the form c(alpha, beta, zeta)
cond.H	condition number of the Hessian matrix at the optimum (i.e. the ratio of the largest to the smallest eigenvalue).
contrasts	(where relevant) the contrasts used for the formula part of the model.
control	list of control parameters as generated by clm.control.
convergence	convergence code where 0 indicates successful convergence and negative values indicate convergence failure; 1 indicates successful convergence to a non-unique optimum.
edf	the estimated degrees of freedom, i.e., the number of parameters in the model fit.
fitted.values	the fitted probabilities.
gradient	a vector of gradients for the coefficients at the estimated optimum.
Hessian	the Hessian matrix for the parameters at the estimated optimum.
info	a table of basic model information for printing.
link	character, the link function used.
logLik	the value of the log-likelihood at the estimated optimum.

maxGradient	the maximum absolute gradient, i.e., max(abs(gradient)).
model	if requested (the default), the model.frame containing variables from formula, scale and nominal parts.
n	the number of observations counted as nrow(X), where X is the design matrix.
na.action	(where relevant) information returned by model.frame on the special handling of NAs.
nobs	the number of observations counted as sum(weights).
nom.contrasts	(where relevant) the contrasts used for the nominal part of the model.
nom.terms	(where relevant) the terms object for the nominal part.
nom.xlevels	(where relevant) a record of the levels of the factors used in fitting for the nominal part.
start	the parameter values at which the optimization has started. An attribute start.iter gives the number of iterations to obtain starting values for models where scale is specified or where the cauchit link is chosen.
S.contrasts	(where relevant) the contrasts used for the scale part of the model.
S.terms	(where relevant) the terms object for the scale part.
S.xlevels	(where relevant) a record of the levels of the factors used in fitting for the scale part.
terms	the terms object for the formula part.
Theta	(where relevant) a table (data.frame) of thresholds for all combinations of levels of factors in the nominal formula.
threshold	character, the threshold structure used.
tJac	the transpose of the Jacobian for the threshold structure.
xlevels	(where relevant) a record of the levels of the factors used in fitting for the formula part.
y.levels	the levels of the response variable after removing levels for which all weights are zero.
zeta	(where relevant) a vector of scale regression parameters.

Author(s)

Rune Haubo B Christensen

Examples

```
fm1 <- clm(rating ~ temp * contact, data = wine)
fm1 ## print method
summary(fm1)
fm2 <- update(fm1, ~.-temp:contact)
anova(fm1, fm2)

drop1(fm1, test = "Chi")
add1(fm1, ~.+judge, test = "Chi")
```

```
fm2 <- step(fm1)
summary(fm2)

coef(fm1)
vcov(fm1)
AIC(fm1)
extractAIC(fm1)
logLik(fm1)
fitted(fm1)

confint(fm1) ## type = "profile"
confint(fm1, type = "Wald")
pr1 <- profile(fm1)
confint(pr1)

## plotting the profiles:
par(mfrow = c(2, 2))
plot(pr1, root = TRUE) ## check for linearity
par(mfrow = c(2, 2))
plot(pr1)
par(mfrow = c(2, 2))
plot(pr1, approx = TRUE)
par(mfrow = c(2, 2))
plot(pr1, Log = TRUE)
par(mfrow = c(2, 2))
plot(pr1, Log = TRUE, relative = FALSE)

## other link functions:
fm4.lgt <- update(fm1, link = "logit") ## default
fm4.prt <- update(fm1, link = "probit")
fm4.ll <- update(fm1, link = "loglog")
fm4.cll <- update(fm1, link = "cloglog")
fm4.cct <- update(fm1, link = "cauchit")
anova(fm4.lgt, fm4.prt, fm4.ll, fm4.cll, fm4.cct)

## structured thresholds:
fm5 <- update(fm1, threshold = "symmetric")
fm6 <- update(fm1, threshold = "equidistant")
anova(fm1, fm5, fm6)

## the slice methods:
slice.fm1 <- slice(fm1)
par(mfrow = c(3, 3))
plot(slice.fm1)
## see more at '?slice.clm'

## Another example:
fm.soup <- clm(SURENESS ~ PRODID, data = soup)
summary(fm.soup)

if(require(MASS)) { ## dropterm, addterm, stepAIC, housing
    fm1 <- clm(rating ~ temp * contact, data = wine)
```

```
    dropterm(fm1, test = "Chi")
    addterm(fm1, ~.+judge, test = "Chi")
    fm3 <- stepAIC(fm1)
    summary(fm3)

    ## Example from MASS::polr:
    fm1 <- clm(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
    summary(fm1)
}
```

`clm.control` *Set control parameters for cumulative link models*

Description

Set control parameters for cumulative link models

Usage

```
clm.control(method = c("Newton", "model.frame", "design", "ucminf", "nlminb",
   "optim"),
   sign.location = c("negative", "positive"),
   sign.nominal = c("positive", "negative"),
   ..., trace = 0L,
   maxIter = 100L, gradTol = 1e-06, maxLineIter = 15L, relTol = 1e-6,
   tol = sqrt(.Machine$double.eps), maxModIter = 5L,
   convergence = c("warn", "silent", "stop", "message"))
```

Arguments

`method`	`"Newton"` fits the model by maximum likelihood and `"model.frame"` cause `clm` to return the `model.frame`, `"design"` causes `clm` to return a list of design matrices etc. that can be used with `clm.fit`. `ucminf`, `nlminb` and `optim` refer to general purpose optimizers.
`sign.location`	change sign of the location part of the model.
`sign.nominal`	change sign of the nominal part of the model.
`trace`	numerical, if `> 0` information is printed about and during the optimization process. Defaults to `0`.
`maxIter`	the maximum number of Newton-Raphson iterations. Defaults to `100`.
`gradTol`	the maximum absolute gradient; defaults to `1e-6`.
`maxLineIter`	the maximum number of step halfings allowed if a Newton(-Raphson) step over shoots. Defaults to `15`.
`relTol`	relative convergence tolerence: relative change in the parameter estimates between Newton iterations. Defaults to `1e-6`.

tol	numerical tolerence on eigenvalues to determine negative-definiteness of Hessian. If the Hessian of a model fit is negative definite, the fitting algorithm did not converge. If the Hessian is singular, the fitting algorithm did converge albeit not to a *unique* optimum, so one or more parameters are not uniquely determined even though the log-likelihood value is.
maxModIter	the maximum allowable number of consecutive iterations where the Newton step needs to be modified to be a decent direction. Defaults to 5.
convergence	action to take if the fitting algorithm did not converge.
...	control arguments parsed on to ucminf, nlminb or optim.

Value

a list of control parameters.

Author(s)

Rune Haubo B Christensen

See Also

clm

`clm.fit` *Fit Cumulative Link Models*

Description

A direct fitter of cumulative link models.

Usage

```
clm.fit(y, ...)

## Default S3 method:
clm.fit(y, ...)

## S3 method for class 'factor'
clm.fit(y, X, S, N, weights = rep(1, nrow(X)),
     offset = rep(0, nrow(X)), S.offset = rep(0, nrow(X)),
     control = list(), start, doFit=TRUE,
     link = c("logit", "probit", "cloglog", "loglog", "cauchit",
              "Aranda-Ordaz", "log-gamma"),
     threshold = c("flexible", "symmetric", "symmetric2", "equidistant"),
     ...)
```

Arguments

y	for the default method a list of model components. For the factor method the response variable; a factor, preferably and ordered factor.
X, S, N	optional design matrices for the regression parameters, scale parameters and nominal parameters respectively.
weights	optional case weights.
offset	an optional offset.
S.offset	an optional offset for the scale part of the model.
control	a list of control parameters, optionally a call to clm.control.
start	an optional list of starting values of the form c(alpha, beta, zeta) for the thresholds and nominal effects (alpha), regression parameters (beta) and scale parameters (zeta).
doFit	logical for whether the model should be fit or the model environment should be returned.
link	the link function.
threshold	the threshold structure, see further at clm.
...	currently not used.

Details

This function does almost the same thing that clm does: it fits a cumulative link model. The main differences are that clm.fit does not setup design matrices from formulae and only does minimal post processing after parameter estimation.

Compared to clm, clm.fit does little to warn the user of any problems with data or model. However, clm.fit will attempt to identify column rank defecient designs. Any unidentified parameters are indicated in the aliased component of the fit.

clm.fit.factor is not able to check if all thresholds are increasing when nominal effects are specified since it needs access to the terms object for the nominal model. If the terms object for the nominal model (nom.terms) is included in y, the default method is able to chech if all thresholds are increasing.

Value

A list with the following components: aliased, alpha, coefficients, cond.H, convergence, df.residual, edf, and optionally beta, zeta These components are documented in clm.

Author(s)

Rune Haubo B Christensen

See Also

clm

Examples

```
## A simple example:
fm1 <- clm(rating ~ contact + temp, data=wine)
summary(fm1)
## get the model frame containing y and X:
mf1 <- update(fm1, method="design")
names(mf1)
res <- clm.fit(mf1$y, mf1$X) ## invoking the factor method
stopifnot(all.equal(coef(res), coef(fm1)))
names(res)

## Fitting with the default method:
mf1$control$method <- "Newton"
res2 <- clm.fit(mf1)
stopifnot(all.equal(coef(res2), coef(fm1)))
```

`clm2` *Cumulative link models*

Description

A new improved implementation of CLMs is available in `clm`.

Fits cumulative link models with an additive model for the location and a multiplicative model for the scale. The function allows for structured thresholds. A popular special case of a CLM is the proportional odds model. In addition to the standard link functions, two flexible link functions, "Arandar-Ordaz" and "log-gamma" are available, where an extra link function parameter provides additional flexibility. A subset of the predictors can be allowed to have nominal rather than ordinal effects. This has been termed "partial proportional odds" when the link is the logistic.

Usage

```
clm2(location, scale, nominal, data, weights, start, subset,
    na.action, contrasts, Hess = TRUE, model,
    link = c("logistic", "probit", "cloglog", "loglog",
    "cauchit", "Aranda-Ordaz", "log-gamma"), lambda,
    doFit = TRUE, control,
    threshold = c("flexible", "symmetric", "equidistant"), ...)
```

Arguments

`location`	a formula expression as for regression models, of the form `response ~ predictors`. The response should be a factor (preferably an ordered factor), which will be interpreted as an ordinal response with levels ordered as in the factor. The model must have an intercept: attempts to remove one will lead to a warning and will be ignored. An offset may be used. See the documentation of `formula` for other details.

scale	a optional formula expression as for the location part, of the form ~ predictors, i.e. with an empty left hand side. An offset may be used. See the documentation of formula for other details.
nominal	an optional formula of the form ~ predictors, i.e. with an empty left hand side. The effects of the predictors in this formula are assumed to nominal.
data	an optional data frame in which to interpret the variables occurring in the formulas.
weights	optional case weights in fitting. Defaults to 1.
start	initial values for the parameters in the format c(alpha, beta, log(zeta), lambda).
subset	expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.
na.action	a function to filter missing data. Applies to terms in all three formulae.
contrasts	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
Hess	logical for whether the Hessian (the inverse of the observed information matrix) should be computed. Use Hess = TRUE if you intend to call summary or vcov on the fit and Hess = FALSE in all other instances to save computing time. The argument is ignored if method = "Newton" where the Hessian is always computed and returned. Defaults to TRUE.
model	logical for whether the model frames should be part of the returned object.
link	link function, i.e. the type of location-scale distribution assumed for the latent distribution. The Aranda-Ordaz and log-gamma links add additional flexibility with a link function parameter, lambda. The Aranda-Ordaz link (Aranda-Ordaz, 1983) equals the logistic link, when lambda = 1 and approaches the loglog link when lambda approaches zero. The log-gamma link (Genter and Farewell, 1985) equals the loglog link when lambda = 1, the probit link when lambda = 0 and the cloglog link when lambda = -1.
lambda	numerical scalar: the link function parameter. Used in combination with link Aranda-Ordaz or log-gamma and otherwise ignored. If lambda is specified, the model is estimated with lambda fixed at this value and otherwise lambda is estimated by ML. For Aranda-Ordaz lambda has to be positive; > 1e-5 for numerical reasons.
doFit	logical for whether the model should be fit or the model environment should be returned.
control	a call to clm2.control.
threshold	specifies a potential structure for the thresholds (cut-points). "flexible" provides the standard unstructured thresholds, "symmetric" restricts the distance between the thresholds to be symmetric around the central one or two thresholds for odd or equal numbers or thresholds respectively, and "equidistant" restricts the distance between consecutive thresholds to the same value.
...	additional arguments are passed on to clm2.control and possibly further on to the optimizer, which can lead to surprising error or warning messages when mistyping arguments etc.

Details

There are methods for the standard model-fitting functions, including summary, vcov, predict, anova, logLik, profile, plot.profile, confint, update, dropterm, addterm, and an extractAIC method.

The design of the implementation is inspired by an idea proposed by Douglas Bates in the talk "Exploiting sparsity in model matrices" presented at the DSC conference in Copenhagen, July 14 2009. Basically an environment is set up with all the information needed to optimize the likelihood function. Extractor functions are then used to get the value of likelihood at current or given parameter values and to extract current values of the parameters. All computations are performed inside the environment and relevant variables are updated during the fitting process. After optimizer termination relevant variables are extracted from the environment and the remaining are discarded.

Some aspects of clm2, for instance, how starting values are obtained, and of the associated methods are inspired by polr from package MASS.

Value

If doFit = FALSE the result is an environment representing the model ready to be optimized. If doFit = TRUE the result is an object of class "clm2" with the following components:

beta	the parameter estimates of the location part.
zeta	the parameter estimates of the scale part on the log scale; the scale parameter estimates on the original scale are given by exp(zeta).
Alpha	vector or matrix of the threshold parameters.
Theta	vector or matrix of the thresholds.
xi	vector of threshold parameters, which, given a threshold function (e.g. "equidistant"), and possible nominal effects define the class boundaries, Theta.
lambda	the value of lambda if lambda is supplied or estimated, otherwise missing.
coefficients	the coefficients of the intercepts (theta), the location (beta), the scale (zeta), and the link function parameter (lambda).
df.residual	the number of residual degrees of freedoms, calculated using the weights.
fitted.values	vector of fitted values for each observation. An observation here is each of the scalar elements of the multinomial table and not a multinomial vector.
convergence	TRUE if the gradient based convergence criterion is met and FALSE otherwise.
gradient	vector of gradients for all the parameters at termination of the optimizer.
optRes	list with results from the optimizer. The contents of the list depends on the choice of optimizer.
logLik	the log likelihood of the model at optimizer termination.
Hessian	if the model was fitted with Hess = TRUE, this is the Hessian matrix of the parameters at the optimum.
scale	model.frame for the scale model.
location	model.frame for the location model.
nominal	model.frame for the nominal model.

edf	the (effective) number of degrees of freedom used by the model.
start	the starting values.
convTol	convergence tolerance for the maximum absolute gradient of the parameters at termination of the optimizer.
method	character, the optimizer.
y	the response variable.
lev	the names of the levels of the response variable.
nobs	the (effective) number of observations, calculated as the sum of the weights.
threshold	character, the threshold function used in the model.
estimLambda	1 if lambda is estimated in one of the flexible link functions and 0 otherwise.
link	character, the link function used in the model.
call	the matched call.
contrasts	contrasts applied to terms in location and scale models.
na.action	the function used to filter missing data.

Author(s)

Rune Haubo B Christensen

References

Agresti, A. (2002) *Categorical Data Analysis.* Second edition. Wiley.

Aranda-Ordaz, F. J. (1983) An Extension of the Proportional-Hazards Model for Grouped Data. *Biometrics*, 39, 109-117.

Genter, F. C. and Farewell, V. T. (1985) Goodness-of-link testing in ordinal regression models. *The Canadian Journal of Statistics*, 13(1), 37-44.

Christensen, R. H. B., Cleaver, G. and Brockhoff, P. B. (2011) Statistical and Thurstonian models for the A-not A protocol with and without sureness. *Food Quality and Preference, 22*, pp. 542-549.

Examples

```
options(contrasts = c("contr.treatment", "contr.poly"))

## A tabular data set:
(tab26 <- with(soup, table("Product" = PROD, "Response" = SURENESS)))
dimnames(tab26)[[2]] <- c("Sure", "Not Sure", "Guess", "Guess", "Not Sure", "Sure")
dat26 <- expand.grid(sureness = as.factor(1:6), prod = c("Ref", "Test"))
dat26$wghts <- c(t(tab26))

m1 <- clm2(sureness ~ prod, scale = ~prod, data = dat26,
          weights = wghts, link = "logistic")

## print, summary, vcov, logLik, AIC:
m1
summary(m1)
vcov(m1)
```

```
logLik(m1)
AIC(m1)
coef(m1)
coef(summary(m1))

## link functions:
m2 <- update(m1, link = "probit")
m3 <- update(m1, link = "cloglog")
m4 <- update(m1, link = "loglog")
m5 <- update(m1, link = "cauchit", start = coef(m1))
m6 <- update(m1, link = "Aranda-Ordaz", lambda = 1)
m7 <- update(m1, link = "Aranda-Ordaz")
m8 <- update(m1, link = "log-gamma", lambda = 1)
m9 <- update(m1, link = "log-gamma")

## nominal effects:
mN1 <-  clm2(sureness ~ 1, nominal = ~ prod, data = dat26,
            weights = wghts, link = "logistic")
anova(m1, mN1)

## optimizer / method:
update(m1, scale = ~ 1, method = "Newton")
update(m1, scale = ~ 1, method = "nlminb")
update(m1, scale = ~ 1, method = "optim")


## threshold functions
mT1 <- update(m1, threshold = "symmetric")
mT2 <- update(m1, threshold = "equidistant")
anova(m1, mT1, mT2)

## Extend example from polr in package MASS:
## Fit model from polr example:
if(require(MASS)) {
    fm1 <- clm2(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
    fm1
    summary(fm1)
    ## With probit link:
    summary(update(fm1, link = "probit"))
    ## Allow scale to depend on Cont-variable
    summary(fm2 <- update(fm1, scale =~ Cont))
    anova(fm1, fm2)
    ## which seems to improve the fit
}

##################################
## It is possible to fit multinomial models (i.e. with nominal
## effects) as the following example shows:
if(require(nnet)) {
    (hous1.mu <- multinom(Sat ~ 1, weights = Freq, data = housing))
    (hous1.clm <- clm2(Sat ~ 1, weights = Freq, data = housing))

    ## It is the same likelihood:
```

```
    all.equal(logLik(hous1.mu), logLik(hous1.clm))

    ## and the same fitted values:
    fitHous.mu <-
        t(fitted(hous1.mu))[t(col(fitted(hous1.mu)) == unclass(housing$Sat))]
    all.equal(fitted(hous1.clm), fitHous.mu)

    ## The coefficients of multinom can be retrieved from the clm2-object
    ## by:
    Pi <- diff(c(0, plogis(hous1.clm$xi), 1))
    log(Pi[2:3]/Pi[1])

    ## A larger model with explanatory variables:
    (hous.mu <- multinom(Sat ~ Infl + Type + Cont, weights = Freq, data = housing))
    (hous.clm <- clm2(Sat ~ 1, nominal = ~ Infl + Type + Cont, weights = Freq,
                      data = housing))

    ## Almost the same likelihood:
    all.equal(logLik(hous.mu), logLik(hous.clm))

    ## And almost the same fitted values:
    fitHous.mu <-
        t(fitted(hous.mu))[t(col(fitted(hous.mu)) == unclass(housing$Sat))]
    all.equal(fitted(hous.clm), fitHous.mu)
    all.equal(round(fitted(hous.clm), 5), round(fitHous.mu), 5)
}
```

`clm2.control` *Set control parameters for cumulative link models*

Description

Set control parameters for cumulative link models

Usage

```
clm2.control(method = c("ucminf", "Newton", "nlminb", "optim",
             "model.frame"), ..., convTol = 1e-4,
             trace = 0, maxIter = 100, gradTol = 1e-5,
             maxLineIter = 10)
```

Arguments

`method`	the optimizer used to maximize the likelihood function. `"Newton"` only works for models without `scale`, structured thresholds and flexible link functions, but is considerably faster than the other optimizers when applicable. `model.frame` simply returns a list of model frames with the location, scale and nominal model frames. `"optim"` uses the `"BFGS"` method.

...	control arguments passed on to the chosen optimizer; see ucminf, optim, and nlminb for details.
convTol	convergence criterion on the size of the maximum absolute gradient.
trace	numerical, if > 0 information is printed about and during the optimization process. Defaults to 0.
maxIter	the maximum number of Newton-Raphson iterations. Defaults to 100.
gradTol	the maximum absolute gradient. This is the termination criterion and defaults to 1e-5.
maxLineIter	the maximum number of step halfings allowed if a Newton(-Raphson) step over shoots. Defaults to 10.

Value

a list of control parameters.

Author(s)

Rune Haubo B Christensen

See Also

clm2

---

clmm *Cumulative Link Mixed Models*

Description

Fits Cumulative Link Mixed Models with one or more random effects via the Laplace approximation or quadrature methods

Usage

```
clmm(formula, data, weights, start, subset, na.action, contrasts, Hess =
TRUE, model = TRUE, link = c("logit", "probit", "cloglog", "loglog",
"cauchit"), doFit = TRUE, control = list(), nAGQ = 1L,
threshold = c("flexible", "symmetric", "symmetric2", "equidistant"), ...)
```

Arguments

formula	a two-sided linear formula object describing the fixed-effects part of the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right. The vertical bar character "|" separates an expression for a model matrix and a grouping factor.
data	an optional data frame in which to interpret the variables occurring in the formula.
weights	optional case weights in fitting. Defaults to 1.
start	optional initial values for the parameters in the format `c(alpha, beta, tau)`, where `alpha` are the threshold parameters, `beta` are the fixed regression parameters and `tau` are variance parameters for the random effects on the log scale.
subset	expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.
na.action	a function to filter missing data.
contrasts	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
Hess	logical for whether the Hessian (the inverse of the observed information matrix) should be computed. Use `Hess = TRUE` if you intend to call `summary` or `vcov` on the fit and `Hess = FALSE` in all other instances to save computing time.
model	logical for whether the model frames should be part of the returned object.
link	link function, i.e. the type of location-scale distribution assumed for the latent distribution. The default `"logit"` link gives the proportional odds mixed model.
doFit	logical for whether the model should be fit or the model environment should be returned.
control	a call to `clmm.control`
nAGQ	integer; the number of quadrature points to use in the adaptive Gauss-Hermite quadrature approximation to the likelihood function. The default (1) gives the Laplace approximation. Higher values generally provide higher precision at the expense of longer computation times, and values between 5 and 10 generally provide accurate maximum likelihood estimates. Negative values give the non-adaptive Gauss-Hermite quadrature approximation, which is generally faster but less accurate than the adaptive version. See the references for further details. Quadrature methods are only available with a single random effects term; the Laplace approximation is always available.
threshold	specifies a potential structure for the thresholds (cut-points). `"flexible"` provides the standard unstructured thresholds, `"symmetric"` restricts the distance between the thresholds to be symmetric around the central one or two thresholds for odd or equal numbers or thresholds respectively, `"symmetric2"` restricts the latent mean in the reference group to zero; this means that the central threshold (even no. response levels) is zero or that the two central thresholds are equal apart from their sign (uneven no. response levels), and `"equidistant"` restricts the distance between consecutive thresholds to be of the same size.
...	additional arguments are passed on to `clm.control`.

Details

This is a new (as of August 2011) improved implementation of CLMMs. The old implementation is available in clmm2. Some features are not yet available in clmm; for instance scale effects, nominal effects and flexible link functions are currently only available in clmm2. clmm is expected to take over clmm2 at some point.

There are standard print, summary and anova methods implemented for "clmm" objects.

Value

a list containing

alpha	threshold parameters.
beta	fixed effect regression parameters.
stDev	standard deviation of the random effect terms.
tau	log(stDev) - the scale at which the log-likelihood function is optimized.
coefficients	the estimated model parameters = c(alpha, beta, tau).
control	List of control parameters as generated by clm.control.
Hessian	Hessian of the model coefficients.
edf	the estimated degrees of freedom used by the model = length(coefficients).
nobs	sum(weights).
n	length(y).
fitted.values	fitted values evaluated with the random effects at their conditional modes.
df.residual	residual degrees of freedom; length(y) - sum(weights)
tJac	Jacobian of the threshold function corresponding to the mapping from standard flexible thresholds to those used in the model.
terms	the terms object for the fixed effects.
contrasts	contrasts applied to the fixed model terms.
na.action	the function used to filter missing data.
call	the matched call.
logLik	value of the log-likelihood function for the model at the optimum.
Niter	number of Newton iterations in the inner loop update of the conditional modes of the random effects.
optRes	list of results from the optimizer.
ranef	list of the conditional modes of the random effects.
condVar	list of the conditional variance of the random effects at their conditional modes.

Author(s)

Rune Haubo B Christensen

Examples

```
## Cumulative link model with one random term:
fmm1 <- clmm(rating ~ temp + contact + (1|judge), data = wine)
summary(fmm1)

## Not run:
## May take a couple of seconds to run this.

## Cumulative link mixed model with two random terms:
mm1 <- clmm(SURENESS ~ PROD + (1|RESP) + (1|RESP:PROD), data = soup,
            link = "probit", threshold = "equidistant")
mm1
summary(mm1)

## test random effect:
mm2 <- clmm(SURENESS ~ PROD + (1|RESP), data = soup,
            link = "probit", threshold = "equidistant")
anova(mm1, mm2)

## End(Not run)
```

`clmm.control` *Set control parameters for cumulative link mixed models*

Description

Set control parameters for cumulative link mixed models

Usage

```
clmm.control(method = c("nlminb", "ucminf", "model.frame"), ..., trace = 0,
maxIter = 50, gradTol = 1e-4, maxLineIter = 50,  useMatrix = FALSE,
innerCtrl = c("warnOnly", "noWarn", "giveError"),
checkRanef = c("warn", "error", "message"))
```

Arguments

`method`	the optimizer used to maximize the marginal likelihood function.
`...`	control arguments passed on to the optimizer; see `ucminf` for details. `ucminf` for details.
`trace`	numerical, if > 0 information is printed about and during the outer optimization process, if < 0 information is also printed about the inner optimization process. Defaults to `0`.
`maxIter`	the maximum number of Newton updates of the inner optimization. `50`.
`gradTol`	the maximum absolute gradient of the inner optimization.

maxLineIter	the maximum number of step halfings allowed if a Newton(-Raphson) step over shoots during the inner optimization.
useMatrix	if TRUE, a general implementation of the Laplace approximation using the Matrix package is used, while if FALSE (default), a C implementation of the Laplace approximation valid only for models with a single random effects term is used when possible. TRUE is not valid for models fitted with quadrature methods.
innerCtrl	the use of warnings/errors if the inner optimization fails to converge.
checkRanef	the use of message/warning/error if there are more random effects than observations.

Value

a list of control parameters

Author(s)

Rune Haubo B Christensen

See Also

clmm

---

clmm2 *Cumulative link mixed models*

Description

Fits cumulative link mixed models, i.e. cumulative link models with random effects via the Laplace approximation or the standard and the adaptive Gauss-Hermite quadrature approximation. The functionality in clm2 is also implemented here. Currently only a single random term is allowed in the location-part of the model.

A new implementation is available in clmm that allows for more than one random effect.

Usage

```
clmm2(location, scale, nominal, random, data, weights, start, subset,
     na.action, contrasts, Hess = FALSE, model = TRUE, sdFixed,
     link = c("logistic", "probit", "cloglog", "loglog",
     "cauchit", "Aranda-Ordaz", "log-gamma"), lambda,
     doFit = TRUE, control, nAGQ = 1,
     threshold = c("flexible", "symmetric", "equidistant"), ...)
```

Arguments

location	as in clm2.
scale	as in clm2.
nominal	as in clm2.
random	a factor for the random effects in the location-part of the model.
data	as in clm2.
weights	as in clm2.
start	initial values for the parameters in the format c(alpha, beta, log(zeta), lambda, log(stDev)) where stDev is the standard deviation of the random effects.
subset	as in clm2.
na.action	as in clm2.
contrasts	as in clm2.
Hess	logical for whether the Hessian (the inverse of the observed information matrix) should be computed. Use Hess = TRUE if you intend to call summary or vcov on the fit and Hess = FALSE in all other instances to save computing time.
model	as in clm2.
sdFixed	If sdFixed is specified (a positive scalar), a model is fitted where the standard deviation for the random term is fixed at the value of sdFixed. If sdFixed is left unspecified, the standard deviation of the random term is estimated from data.
link	as in clm2.
lambda	as in clm2.
doFit	as in clm2 although it can also be one of c("no", "R" "C"), where "R" use the R-implementation for fitting, "C" (default) use C-implementation for fitting and "no" behaves as FALSE and returns the environment.
control	a call to clmm2.control.
threshold	as in clm2.
nAGQ	the number of quadrature points to be used in the adaptive Gauss-Hermite quadrature approximation to the marginal likelihood. Defaults to 1 which leads to the Laplace approximation. An odd number of quadrature points is encouraged and 3, 5 or 7 are usually enough to achive high precision. Negative values give the standard, i.e. non-adaptive Gauss-Hermite quadrature.
...	additional arguments are passed on to clm2.control and possibly further on to the optimizer, which can lead to surprising error or warning messages when mistyping arguments etc.

Details

There are methods for the standard model-fitting functions, including summary, vcov, profile, plot.profile, confint, anova, logLik, predict and an extractAIC method.

A Newton scheme is used to obtain the conditional modes of the random effects for Laplace and AGQ approximations, and a non-linear optimization is performed over the fixed parameter set to get the maximum likelihood estimates. The Newton scheme uses the observed Hessian rather than

the expected as is done in e.g. glmer, so results from the Laplace approximation for binomial fits should in general be more precise - particularly for other links than the "logistic".

Core parts of the function are implemented in C-code for speed.

The function calls clm2 to up an environment and to get starting values.

Value

If doFit = FALSE the result is an environment representing the model ready to be optimized. If doFit = TRUE the result is an object of class "clmm2" with the following components:

stDev	the standard deviation of the random effects.
Niter	the total number of iterations in the Newton updates of the conditional modes of the random effects.
grFac	the grouping factor defining the random effects.
nAGQ	the number of quadrature points used in the adaptive Gauss-Hermite Quadrature approximation to the marginal likelihood.
ranef	the conditional modes of the random effects, sometimes referred to as "random effect estimates".
condVar	the conditional variances of the random effects at their conditional modes.
beta	the parameter estimates of the location part.
zeta	the parameter estimates of the scale part on the log scale; the scale parameter estimates on the original scale are given by exp(zeta).
Alpha	vector or matrix of the threshold parameters.
Theta	vector or matrix of the thresholds.
xi	vector of threshold parameters, which, given a threshold function (e.g. "equidistant"), and possible nominal effects define the class boundaries, Theta.
lambda	the value of lambda if lambda is supplied or estimated, otherwise missing.
coefficients	the coefficients of the intercepts (theta), the location (beta), the scale (zeta), and the link function parameter (lambda).
df.residual	the number of residual degrees of freedoms, calculated using the weights.
fitted.values	vector of fitted values conditional on the values of the random effects. Use predict to get the fitted values for a random effect of zero. An observation here is taken to be each of the scalar elements of the multinomial table and not a multinomial vector.
convergence	TRUE if the optimizer terminates wihtout error and FALSE otherwise.
gradient	vector of gradients for the unit-variance random effects at their conditional modes.
optRes	list with results from the optimizer. The contents of the list depends on the choice of optimizer.
logLik	the log likelihood of the model at optimizer termination.
Hessian	if the model was fitted with Hess = TRUE, this is the Hessian matrix of the parameters at the optimum.
scale	model.frame for the scale model.

location	model.frame for the location model.
nominal	model.frame for the nominal model.
edf	the (effective) number of degrees of freedom used by the model.
start	the starting values.
method	character, the optimizer.
y	the response variable.
lev	the names of the levels of the response variable.
nobs	the (effective) number of observations, calculated as the sum of the weights.
threshold	character, the threshold function used in the model.
estimLambda	1 if lambda is estimated in one of the flexible link functions and 0 otherwise.
link	character, the link function used in the model.
call	the matched call.
contrasts	contrasts applied to terms in location and scale models.
na.action	the function used to filter missing data.

Author(s)

Rune Haubo B Christensen

References

Agresti, A. (2002) *Categorical Data Analysis.* Second edition. Wiley.

Examples

```
options(contrasts = c("contr.treatment", "contr.poly"))

## More manageable data set:
dat <- subset(soup, as.numeric(as.character(RESP)) <=  24)
dat$RESP <- dat$RESP[drop=TRUE]

m1 <- clmm2(SURENESS ~ PROD, random = RESP, data = dat, link="probit",
           Hess = TRUE, method="ucminf", threshold = "symmetric")

m1
summary(m1)
logLik(m1)
vcov(m1)
extractAIC(m1)
anova(m1, update(m1, location = SURENESS ~ 1, Hess = FALSE))
anova(m1, update(m1, random = NULL))

## Use adaptive Gauss-Hermite quadrature rather than the Laplace
## approximation:
update(m1, Hess = FALSE, nAGQ = 3)

## Use standard Gauss-Hermite quadrature:
```

```
update(m1, Hess = FALSE, nAGQ = -7)

######################################################################
## Binomial example with the cbpp data from the lme4-package:
if(require(lme4)) {
    cbpp2 <- rbind(cbpp[,-(2:3)], cbpp[,-(2:3)])
    cbpp2 <- within(cbpp2, {
        incidence <- as.factor(rep(0:1, each=nrow(cbpp)))
        freq <- with(cbpp, c(incidence, size - incidence))
    })

    ## Fit with Laplace approximation:
    fm1 <- clmm2(incidence ~ period, random = herd, weights = freq,
                 data = cbpp2, Hess = 1)
    summary(fm1)

    ## Fit with the adaptive Gauss-Hermite quadrature approximation:
    fm2 <- clmm2(incidence ~ period, random = herd, weights = freq,
                 data = cbpp2, Hess = 1, nAGQ = 7)
    summary(fm2)
}
```

---

clmm2.control *Set control parameters for cumulative link mixed models*

Description

Set control parameters for cumulative link mixed models

Usage

```
clmm2.control(method = c("ucminf", "nlminb", "model.frame"), ...,
              trace = 0, maxIter = 50, gradTol = 1e-4,
              maxLineIter = 50,
              innerCtrl = c("warnOnly", "noWarn", "giveError"))
```

Arguments

method	the optimizer used to maximize the marginal likelihood function.
...	control arguments passed on to the chosen optimizer; see ucminf, optim, and nlminb for details.
trace	numerical, if > 0 information is printed about and during the outer optimization process, if < 0 information is also printed about the inner optimization process. Defaults to 0.
maxIter	the maximum number of Newton updates of the inner optimization. 50.
gradTol	the maximum absolute gradient of the inner optimization.

maxLineIter	the maximum number of step halfings allowed if a Newton(-Raphson) step over shoots during the inner optimization.
innerCtrl	the use of warnings/errors if the inner optimization fails to converge.

Details

When the default optimizer, ucminf is used, the default values of that optimizers control options are changed to grtol = 1e-5 and grad = "central".

Value

a list of control parameters.

Author(s)

Rune Haubo B Christensen

See Also

clmm2

---

confint *Confidence intervals and profile likelihoods for parameters in cumulative link models*

Description

Computes confidence intervals from the profiled likelihood for one or more parameters in a cumulative link model, or plots the profile likelihood.

Usage

```
## S3 method for class 'clm'
confint(object, parm, level = 0.95,
        type = c("profile", "Wald"), trace = FALSE, ...)

## S3 method for class 'profile.clm'
confint(object, parm = seq_len(nprofiles),
        level = 0.95, ...)

## S3 method for class 'clm'
profile(fitted, which.beta = seq_len(nbeta),
        which.zeta = seq_len(nzeta), alpha = 0.001,
        max.steps = 50, nsteps = 8, trace = FALSE, step.warn = 5,
        control = list(), ...)

## S3 method for class 'profile.clm'
```

```
plot(x, which.par = seq_len(nprofiles),
        level = c(0.95, 0.99), Log = FALSE, relative = TRUE, root =
        FALSE, fig = TRUE, approx = root, n = 1e3,
        ask = prod(par("mfcol")) < length(which.par) && dev.interactive(),
        ..., ylim = NULL)
```

Arguments

object, fitted, x	a fitted clm object or a profile.clm object.
parm, which.par, which.beta, which.zeta	a numeric or character vector indicating which regression coefficients should be profiled. By default all coefficients are profiled. Ignored for confint.clm where all parameters are considered.
level	the confidence level. For the plot method a vector of levels for which horizontal lines should be drawn.
type	the type of confidence interval.
trace	if trace is TRUE or positive, information about progress is printed.
Log	should the profile likelihood be plotted on the log-scale?
relative	should the relative or the absolute likelihood be plotted?
root	should the (approximately linear) likelihood root statistic be plotted?
approx	should the Gaussian or quadratic approximation to the (log) likelihood be included?
fig	should the profile likelihood be plotted?
ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
n	the no. points used in the spline interpolation of the profile likelihood.
ylim	overrules default y-limits on the plot of the profile likelihood.
alpha	the likelihood is profiled in the 100*(1-alpha)% confidence region as determined by the profile likelihood.
control	a list of control parameters for clm. Possibly use clm.control to set these.
max.steps	the maximum number of profiling steps in each direction for each parameter.
nsteps	the (approximate) number of steps to take in each direction of the profile for each parameter. The step length is determined accordingly assuming a quadratic approximation to the log-likelihood function. The actual number of steps will often be close to nsteps, but will deviate when the log-likelihood functions is irregular.
step.warn	a warning is issued if the number of steps in each direction (up or down) for a parameter is less than step.warn. If few steps are taken, the profile will be unreliable and derived confidence intervals will be inaccurate.
...	additional arguments to be parsed on to methods.

Details

These `confint` methods call the appropriate profile method, then finds the confidence intervals by interpolation of the profile traces. If the profile object is already available, this should be used as the main argument rather than the fitted model object itself.

Value

`confint`: A matrix with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

`plot.profile.clm` invisibly returns the profile object, i.e., a list of `data.frame`s with an `lroot` component for the likelihood root statistic and a matrix `par.vals` with values of the parameters.

Author(s)

Rune Haubo B Christensen

See Also

`profile` and `confint`

Examples

```
## Accurate profile likelihood confidence intervals compared to the
## conventional Wald intervals:
fm1 <- clm(rating ~ temp * contact, data = wine)
confint(fm1) ## type = "profile"
confint(fm1, type = "Wald")
pr1 <- profile(fm1)
confint(pr1)

## plotting the profiles:
par(mfrow = c(2, 2))
plot(pr1, root = TRUE) ## check for linearity
par(mfrow = c(2, 2))
plot(pr1)
par(mfrow = c(2, 2))
plot(pr1, approx = TRUE)
par(mfrow = c(2, 2))
plot(pr1, Log = TRUE)
par(mfrow = c(2, 2))
plot(pr1, Log = TRUE, relative = FALSE)
## Not likely to be useful but allowed for completeness:
par(mfrow = c(2, 2))
plot(pr1, Log = FALSE, relative = FALSE)

## Example from polr in package MASS:
## Fit model from polr example:
if(require(MASS)) {
    fm1 <- clm(Sat ~ Infl + Type + Cont, weights = Freq,
               data = housing)
```

```
    pr1 <- profile(fm1)
    confint(pr1)
    par(mfrow=c(2,2))
    plot(pr1)
}
```

---

convergence *Check convergence of cumulative link models*

Description

Check the accuracy of the parameter estimates of cumulative link models. The number of correct decimals and number of significant digits is given for the maximum likelihood estimates of the parameters in a cumulative link model fitted with clm.

Usage

```
convergence(object, ...)

## S3 method for class 'clm'
convergence(object, digits = max(3, getOption("digits") - 3),
   tol = sqrt(.Machine$double.eps), ...)
```

Arguments

object	for the clm method an object of class "clm", i.e., the result of a call to clm.
digits	the number of digits in the printed table.
tol	numerical tolerence to judge if the Hessian is positive definite from its smallest eigenvalue.
...	arguments to a from methods. Not used by the clm method.

Details

The number of correct decimals is defined as...

The number of significant digits is defined as ...

The number of correct decimals and the number of significant digits are determined from the numerical errors in the parameter estimates. The numerical errors are determined from the Method Independent Error Theorem (Elden et al, 2004) and is based on the Newton step evaluated at convergence.

Value

Convergence information. In particular a table where the `Error` column gives the numerical error in the parameter estimates. These numbers express how far the parameter estimates in the fitted model are from the true maximum likelihood estimates for this model. The `Cor.Dec` gives the number of correct decimals with which the the parameters are determined and the `Sig.Dig` gives the number of significant digits with which the parameters are determined.

The number denoted `logLik.error` is the error in the value of log-likelihood in the fitted model at the parameter values of that fit. An accurate determination of the log-likelihood is essential for accurate likelihood ratio tests in model comparison.

Author(s)

Rune Haubo B Christensen

References

Elden, L., Wittmeyer-Koch, L. and Nielsen, H. B. (2004) *Introduction to Numerical Computation — analysis and Matlab illustrations.* Studentliteratur.

Examples

```
## Simple model:
fm1 <- clm(rating ~ contact + temp, data=wine)
summary(fm1)
convergence(fm1)
```

`drop.coef` *Ensure Full Rank Design Matrix*

Description

Coefficients (columns) are dropped from a design matrix to ensure that it has full rank.

Usage

```
drop.coef(X, silent = FALSE)
```

Arguments

`X`	a design matrix, e.g., the result of `model.matrix` possibly of less than full column rank, i.e., with redundant parameters. Works for `ncol(X) >= 0` and `nrow(X) >= 0`.
`silent`	should a message not be issued if X is column rank deficient?

Details

Redundant columns of the design matrix are identified with the LINPACK implementation of the qr decomposition and removed. The returned design matrix will have qr(X)$rank columns.

Value

The design matrix X without redundant columns.

Author(s)

Rune Haubo B Christensen

See Also

qr and lm

Examples

```
X <- model.matrix( ~ PRODID * DAY, data = soup)
ncol(X)
newX <- drop.coef(X)
ncol(newX)

## Essentially this is being computed:
qr.X <- qr(X, tol = 1e-7, LAPACK = FALSE)
newX <- X[, qr.X$pivot[1:qr.X$rank], drop = FALSE]
## is newX of full column rank?
ncol(newX) == qr(newX)$rank
## the number of columns being dropped:
ncol(X) - ncol(newX)
```

---

gfun *Gradients of common densities*

Description

Gradients of common density functions in their standard forms, i.e., with zero location (mean) and unit scale. These are implemented in C for speed and care is taken that the correct results are provided for the argument being NA, NaN, Inf, -Inf or just extremely small or large.

Usage

```
gnorm(x)

glogis(x)
```

```
gcauchy(x)
```

Arguments

x	numeric vector of quantiles.

Details

The gradients are given by:

- gnorm: If $f(x)$ is the normal density with mean 0 and spread 1, then the gradient is

$$f'(x) = -xf(x)$$

- glogis: If $f(x)$ is the logistic density with mean 0 and scale 1, then the gradient is

$$f'(x) = 2\exp(-x)^2(1+\exp(-x))^{-3} - \exp(-x)(1+\exp(-x))^{-2}$$

- pcauchy: If $f(x) = [\pi(1+x^2)^2]^{-1}$ is the cauchy density with mean 0 and scale 1, then the gradient is

$$f'(x) = -2x[\pi(1+x^2)^2]^{-1}$$

These gradients are used in the Newton-Raphson algorithms in fitting cumulative link models with `clm` and cumulative link mixed models with `clmm`.

Value

a numeric vector of gradients.

Author(s)

Rune Haubo B Christensen

See Also

Gradients of densities are also implemented for the extreme value distribtion (`gumbel`) and the the log-gamma distribution (`log-gamma`).

Examples

```
x <- -5:5
gnorm(x)
glogis(x)
gcauchy(x)
```

gumbel *The Gumbel Distribution*

Description

Density, distribution function, quantile function, random generation, and gradient of density of the extreme value (maximum and minimum) distributions. The Gumbel distribution is also known as the extreme value maximum distribution, the double-exponential distribution and the log-Weibull distribution.

Usage

```
dgumbel(x, location = 0, scale = 1, log = FALSE, max = TRUE)

pgumbel(q, location = 0, scale = 1, lower.tail = TRUE, max = TRUE)

qgumbel(p, location = 0, scale = 1, lower.tail = TRUE, max = TRUE)

rgumbel(n, location = 0, scale = 1, max = TRUE)

ggumbel(x, max = TRUE)
```

Arguments

x,q	numeric vector of quantiles.
p	vector of probabilities.
n	number of observations.
location	numeric scalar.
scale	numeric scalar.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
log	logical; if TRUE, probabilities p are given as log(p).
max	distribution for extreme maxima (default) or minima? The default corresponds to the standard right-skew Gumbel distribution.

Details

dgumbel, pgumbel and ggumbel are implemented in C for speed and care is taken that 'correct' results are provided for values of NA, NaN, Inf, -Inf or just extremely small or large.

The distribution functions, densities and gradients are used in the Newton-Raphson algorithms in fitting cumulative link models with clm and cumulative link mixed models with clmm.

Value

pgumbel gives the distribution function, dgumbel gives the density, ggumbel gives the gradient of the density, qgumbel is the quantile function, and rgumbel generates random deviates.

Author(s)

Rune Haubo B Christensen

References

wikipedia.org/wiki/Gumbel_distribution

See Also

Gradients of densities are also implemented for the normal, logistic, cauchy, cf. gfun and the log-gamma distribution, cf. lgamma.

Examples

```
## Illustrating the symmetry of the distribution functions:
pgumbel(5) == 1 - pgumbel(-5, max=FALSE) ## TRUE
dgumbel(5) == dgumbel(-5, max=FALSE) ## TRUE
ggumbel(5) == -ggumbel(-5, max=FALSE) ## TRUE

## More examples:
x <- -5:5

(pp <- pgumbel(x))
qgumbel(pp)
dgumbel(x)
ggumbel(x)

(ppp <- pgumbel(x, max=FALSE))
## Observe that probabilities close to 0 are more accurately determined than
## probabilities close to 1:
qgumbel(ppp, max=FALSE)
dgumbel(x, max=FALSE)
ggumbel(x, max=FALSE)

## random deviates:
set.seed(1)
(r1 <- rgumbel(10))
set.seed(1)
r2 <- -rgumbel(10, max = FALSE)
all(r1 == r2) ## TRUE
```

income *Income distribution (percentages) in the Northeast US*

Description

Income distribution (percentages) in the Northeast US in 1960 and 1970 adopted from McCullagh (1980).

Usage

```
income
```

Format

`year` year.

`pct` percentage of population in income class per year.

`income` income groups. The unit is thousands of constant (1973) US dollars.

Source

Data are adopted from McCullagh (1980).

References

McCullagh, P. (1980) Regression Models for Ordinal Data. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 42, No. 2., pp. 109-142.

Examples

```
print(income)

## Convenient table:
(tab <- xtabs(pct ~ year + income, income))

## small rounding error in 1970:
rowSums(tab)

## compare link functions via the log-likelihood:
links <- c("logit", "probit", "cloglog", "loglog", "cauchit")
sapply(links, function(link) {
  clm(income ~ year, data=income, weights=pct, link=link)$logLik })
## a heavy tailed (cauchy) or left skew (cloglog) latent distribution
## is fitting best.

## The data are defined as:
income.levels <- c(0, 3, 5, 7, 10, 12, 15)
income <- paste(income.levels, c(rep("-", 6), "+"),
                c(income.levels[-1], ""), sep = "")
```

```
income <-
  data.frame(year=factor(rep(c("1960", "1970"), each = 7)),
             pct = c(6.5, 8.2, 11.3, 23.5, 15.6, 12.7, 22.2,
               4.3, 6, 7.7, 13.2, 10.5, 16.3, 42.1),
             income=factor(rep(income, 2), ordered=TRUE,
               levels=income))
```

`lgamma` *The log-gamma distribution*

Description

Density, distribution function and gradient of density for the log-gamma distribution. These are implemented in C for speed and care is taken that the correct results are provided for values of `NA`, `NaN`, `Inf`, `-Inf` or just extremely small or large values.

The log-gamma is a flexible location-scale distribution on the real line with an extra parameter, λ. For $\lambda = 0$ the distribution equals the normal or Gaussian distribution, and for λ equal to 1 and -1, the Gumbel minimum and maximum distributions are obtained.

Usage

```
plgamma(q, lambda, lower.tail = TRUE)

dlgamma(x, lambda, log = FALSE)

glgamma(x, lambda)
```

Arguments

`x,q`	numeric vector of quantiles.
`lambda`	numerical scalar
`lower.tail`	logical; if `TRUE` (default), probabilities are $P[X \leq x]$ otherwise, $P[X > x]$.
`log`	logical; if `TRUE`, probabilities p are given as log(p).

Details

If $\lambda < 0$ the distribution is right skew, if $\lambda = 0$ the distribution is symmetric (and equals the normal distribution), and if $\lambda > 0$ the distribution is left skew.

The log-gamma distribution function is defined as . . . pending.

The density and gradient of the density are defined as. . . pending.

These distribution functions, densities and gradients are used in the Newton-Raphson algorithms in fitting cumulative link models with `clm2` and cumulative link mixed models with `clmm2` using the log-gamma link.

Value

plgamma gives the distribution function, dlgamma gives the density and glgamma gives the gradient of the density.

Author(s)

Rune Haubo B Christensen

References

Genter, F. C. and Farewell, V. T. (1985) Goodness-of-link testing in ordinal regression models. *The Canadian Journal of Statistics*, 13(1), 37-44.

See Also

Gradients of densities are also implemented for the normal, logistic, cauchy, cf. gfun and the Gumbel distribution, cf. gumbel.

Examples

```
## Illustrating the link to other distribution functions:
x <- -5:5
plgamma(x, lambda = 0) == pnorm(x)
all.equal(plgamma(x, lambda = -1), pgumbel(x)) ## TRUE, but:
plgamma(x, lambda = -1) == pgumbel(x)
plgamma(x, lambda = 1) == pgumbel(x, max = FALSE)

dlgamma(x, lambda = 0) == dnorm(x)
dlgamma(x, lambda = -1) == dgumbel(x)
dlgamma(x, lambda = 1) == dgumbel(x, max = FALSE)

glgamma(x, lambda = 0) == gnorm(x)
all.equal(glgamma(x, lambda = -1), ggumbel(x)) ## TRUE, but:
glgamma(x, lambda = -1) == ggumbel(x)
all.equal(glgamma(x, lambda = 1), ggumbel(x, max = FALSE)) ## TRUE, but:
glgamma(x, lambda = 1) == ggumbel(x, max = FALSE)
## There is a loss of accuracy, but the difference is very small:
glgamma(x, lambda = 1) - ggumbel(x, max = FALSE)

## More examples:
x <- -5:5
plgamma(x, lambda = .5)
dlgamma(x, lambda = .5)
glgamma(x, lambda = .5)
```

nominal_test *Likelihood ratio tests of model terms in scale and nominal formulae*

Description

Add all model terms to scale and nominal formulae and perform likelihood ratio tests. These tests can be viewed as goodness-of-fit tests. With the logit link, `nominal_test` provides likelihood ratio tests of the proportional odds assumption. The `scale_test` tests can be given a similar interpretation.

Usage

```
nominal_test(object, ...)

## S3 method for class 'clm'
nominal_test(object, scope, trace=FALSE, ...)

scale_test(object, ...)

## S3 method for class 'clm'
scale_test(object, scope, trace=FALSE, ...)
```

Arguments

`object`	for the `clm` method an object of class `"clm"`, i.e., the result of a call to `clm`.
`scope`	a formula or character vector specifying the terms to add to scale or nominal. In `nominal_test` terms in scope already in `nominal` are ignored. In `scale_test` terms in scope already in `scale` are ignored. In `nominal_test` the default is to add all terms from `formula` (location part) and `scale` that are not also in `nominal`. In `scale_test` the default is to add all terms from `formula` (location part) that are not also in `scale`.
`trace`	if `TRUE` additional information may be given on the fits as they are tried.
`...`	arguments passed to or from other methods.

Details

The definition of AIC is only up to an additive constant because the likelihood function is only defined up to an additive constant.

Value

A table of class `"anova"` containing columns for the change in degrees of freedom, AIC, the likelihood ratio statistic and a p-value based on the asymptotic chi-square distribtion of the likelihood ratio statistic under the null hypothesis.

Author(s)

Rune Haubo B Christensen

Examples

```
## Fit cumulative link model:
fm <- clm(rating ~ temp + contact, data=wine)
summary(fm)
## test partial proportional odds assumption for temp and contact:
nominal_test(fm)
## no evidence of non-proportional odds.
## test if there are signs of scale effects:
scale_test(fm)
## no evidence of scale effects.

## tests of scale and nominal effects for the housing data from MASS:
if(require(MASS)) {
    fm1 <- clm(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
    scale_test(fm1)
    nominal_test(fm1)
    ## Evidence of multiplicative/scale effect of 'Cont'. This is a breach
    ## of the proportional odds assumption.
}
```

---

predict.clm *Predict Method for CLM fits*

Description

Obtains predictions from a cumulative link model.

Usage

```
## S3 method for class 'clm'
predict(object, newdata, se.fit = FALSE, interval = FALSE,
           level = 0.95,
           type = c("prob", "class", "cum.prob", "linear.predictor"),
           na.action = na.pass, ...)
```

Arguments

object	a fitted object of class inheriting from clm.

newdata	optionally, a data frame in which to look for variables with which to predict. Note that all predictor variables should be present having the same names as the variables used to fit the model. If the response variable is present in newdata predictions are obtained for the levels of the response as given by newdata. If the response variable is omitted from newdata predictions are obtained for all levels of the response variable for each of the rows of newdata.
se.fit	should standard errors of the predictions be provided? Not applicable and ignored when type = "class".
interval	should confidence intervals for the predictions be provided? Not applicable and ignored when type = "class".
level	the confidence level.
type	the type of predictions. "prob" gives probabilities, "class" gives predicted response class membership defined as highest probability prediction, "cum.prob" gives cumulative probabilities (see details) and "linear.predictor" gives predictions on the scale of the linear predictor including the boundary categories.
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	further arguments passed to or from other methods.

Details

If newdata is omitted and type = "prob" a vector of fitted probabilities are returned identical to the result from fitted.

If newdata is supplied and the response variable is omitted, then predictions, standard errors and intervals are matrices rather than vectors with the same number of rows as newdata and with one column for each response class. If type = "class" predictions are always a vector.

If newdata is omitted, the way missing values in the original fit are handled is determined by the na.action argument of that fit. If na.action = na.omit omitted cases will not appear in the residuals, whereas if na.action = na.exclude they will appear (in predictions, standard errors or interval limits), with residual value NA. See also napredict.

If type = "cum.prob" or type = "linear.predictor" there will be two sets of predictions, standard errors and intervals; one for j and one for j-1 (in the usual notation) where j = 1, ..., J index the response classes.

If newdata is supplied and the response variable is omitted, then predict.clm returns much the same thing as predict.polr (matrices of predictions). Similarly, if type = "class".

If the fit is rank-deficient, some of the columns of the design matrix will have been dropped. Prediction from such a fit only makes sense if newdata is contained in the same subspace as the original data. That cannot be checked accurately, so a warning is issued (cf. predict.lm).

If a flexible link function is used (Aranda-Ordaz or log-gamma) standard errors and confidence intervals of predictions do not take the uncertainty in the link-parameter into account.

Value

A list containing the following components

fit	predictions or fitted values if newdata is not supplied.

`se.fit`	if `se.fit=TRUE` standard errors of the predictions otherwise `NULL`.
`upr, lwr`	if `interval=TRUE` lower and upper confidence limits.

Author(s)

Rune Haubo B Christensen

See Also

`clm`, `clmm`.

Examples

```
## simple model:
fm1 <- clm(rating ~ contact + temp, data=wine)
summary(fm1)

## Fitted values with standard errors and confidence intervals:
predict(fm1, se.fit=TRUE, interval=TRUE) # type="prob"
## class predictions for the observations:
predict(fm1, type="class")

newData <- expand.grid(temp = c("cold", "warm"),
                       contact = c("no", "yes"))

## Predicted probabilities in all five response categories for each of
## the four cases in newData:
predict(fm1, newdata=newData, type="prob")
## now include standard errors and intervals:
predict(fm1, newdata=newData, se.fit=TRUE, interval=TRUE, type="prob")
```

---

profile.clmm2 *Confidence intervals and profile likelihoods for the standard deviation for the random term in cumulative link mixed models*

Description

Computes confidence intervals from the profiled likelihood for the standard devation for the random term in a fitted cumulative link mixed model, or plots the associated profile likelihood function.

Usage

```
## S3 method for class 'profile.clmm2'
confint(object, parm = seq_along(Pnames), level = 0.95, ...)

## S3 method for class 'clmm2'
```

```
profile(fitted, alpha = 0.01, range, nSteps = 20, trace = 1, ...)

## S3 method for class 'profile.clmm2'
plot(x, parm = seq_along(Pnames), level = c(0.95, 0.99),
        Log = FALSE, relative = TRUE, fig = TRUE, n = 1e3, ..., ylim = NULL)
```

Arguments

object	a fitted profile.clmm2 object.
fitted	a fitted clmm2 object.
x	a profile.clmm2 object.
parm	For confint.profile.clmm2: a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered. Currently only "stDev" or 1 are supported. For plot.profile.clmm2: a specification of which parameters the profile likelihood are to be plotted for, either a vector of numbers or a vector of names. If missing, all parameters are considered. Currently only "stDev" or 1 are supported.
level	the confidence level required. Observe that the model has to be profiled in the appropriate region; otherwise the limits are NA.
trace	logical. Should profiling be traced? Defaults to TRUE due to the time consuming nature of the computation.
alpha	Determines the range of profiling. By default the likelihood is profiled approximately in the 99% confidence interval region as determined by the Wald approximation. This is usually sufficient for 95% profile likelihood confidence limits.
range	if range is specified, this overrules the range computation based on alpha. range should be all positive and stDev is profiled in range(range).
nSteps	the number of points at which to profile the likelihood function. This determines the resolution and accuracy of the profile likelihood function; higher values gives a higher resolution, but also longer computation times.
Log	should the profile likelihood be plotted on the log-scale?
relative	should the relative or the absolute likelihood be plotted?
fig	should the profile likelihood be plotted?
n	the no. points used in the spline interpolation of the profile likelihood for plotting.
ylim	overrules default y-limits on the plot of the profile likelihood.
...	additional argument(s), e.g. graphical parameters for the plot method.

Details

A confint.clmm2 method deliberately does not exist due to the time consuming nature of the computations. The user is required to compute the profile object first and then call confint on the profile object to obtain profile likelihood confidence intervals.

In plot.profile.clm2: at least one of Log and relative arguments have to be TRUE.

Value

confint: A matrix with columns giving lower and upper confidence limits. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

plot.profile.clm2 invisibly returns the profile object.

Author(s)

Rune Haubo B Christensen

See Also

profile and confint

Examples

```
options(contrasts = c("contr.treatment", "contr.poly"))

if(require(lme4)) { ## access cbpp data
    cbpp2 <- rbind(cbpp[,-(2:3)], cbpp[,-(2:3)])
    cbpp2 <- within(cbpp2, {
        incidence <- as.factor(rep(0:1, each=nrow(cbpp)))
        freq <- with(cbpp, c(incidence, size - incidence))
    })

    ## Fit with Laplace approximation:
    fm1 <- clmm2(incidence ~ period, random = herd, weights = freq,
                 data = cbpp2, Hess = 1)

    pr.fm1 <- profile(fm1)
    confint(pr.fm1)

    par(mfrow = c(2,2))
    plot(pr.fm1)
    plot(pr.fm1, Log=TRUE, relative = TRUE)
    plot(pr.fm1, Log=TRUE, relative = FALSE)
}
```

---

ranef *Extract conditional modes and conditional variances from clmm objects*

Description

The ranef function extracts the conditional modes of the random effects from a clmm object. That is, the modes of the distributions for the random effects given the observed data and estimated model parameters. In a Bayesian language they are posterior modes.

The conditional variances are computed from the second order derivatives of the conditional distribution of the random effects. Note that these variances are computed at a fixed value of the model parameters and thus do not take the uncertainty of the latter into account.

Usage

```
ranef(object, ...)

condVar(object, ...)

## S3 method for class 'clmm'
ranef(object, condVar=FALSE, ...)

## S3 method for class 'clmm'
condVar(object, ...)
```

Arguments

object	a clmm object.
condVar	an optional logical argument indicating of conditional variances should be added as attributes to the conditional modes.
...	currently not used by the clmm methods.

Details

The ranef method returns a list of data.frames; one for each distinct grouping factor. Each data.frame has as many rows as there are levels for that grouping factor and as many columns as there are random effects for each level. For example a model can contain a random intercept (one column) or a random intercept and a random slope (two columns) for the same grouping factor.

If conditional variances are requested, they are returned in the same structure as the conditional modes (random effect estimates/predictions).

Value

The ranef method returns a list of data.frames with the random effects predictions/estimates computed as conditional modes. If condVar = TRUE a data.frame with the conditional variances is stored as an attribute on each data.frame with conditional modes.

The condVar method returns a list of data.frames with the conditional variances. It is a convenience function that simply computes the conditional modes and variances, then extracts and returns only the latter.

Author(s)

Rune Haubo B Christensen

Examples

```
fm1 <- clmm(rating ~ contact + temp + (1|judge), data=wine)

## Extract random effect estimates/conditional modes:
re <- ranef(fm1, condVar=TRUE)

## Get conditional variances:
attr(re$judge, "condVar")
## Alternatively:
condVar(fm1)
```

`slice` *Slice the likelihood of a clm*

Description

Slice likelihood and plot the slice. This is usefull for illustrating the likelihood surface around the MLE (maximum likelihood estimate) and provides graphics to substantiate (non-)convergence of a model fit. Also, the closeness of a quadratic approximation to the log-likelihood function can be inspected for relevant parameters. A slice is considerably less computationally demanding than a profile.

Usage

```
slice(object, ...)

## S3 method for class 'clm'
slice(object, parm = seq_along(par), lambda = 3,
     grid = 100, quad.approx = TRUE, ...)

## S3 method for class 'slice.clm'
plot(x, parm = seq_along(x),
    type = c("quadratic", "linear"), plot.mle = TRUE,
    ask = prod(par("mfcol")) < length(parm) && dev.interactive(), ...)
```

Arguments

`object`	for the `clm` method an object of class `"clm"`, i.e., the result of a call to `clm`.
`x`	a `slice.clm` object, i.e., the result of `slice(clm.object)`.
`parm`	for `slice.clm` a numeric or character vector indexing parameters, for `plot.slice.clm` only a numeric vector is accepted. By default all parameters are selected.
`lambda`	the number of curvature units on each side of the MLE the slice should cover.
`grid`	the number of values at which to compute the log-likelihood for each parameter.

quad.approx	compute and include the quadratic approximation to the log-likelihood function?
type	"quadratic" plots the log-likelihood function which is approximately quadratic, and "linear" plots the signed square root of the log-likelihood function which is approximately linear.
plot.mle	include a vertical line at the MLE (maximum likelihood estimate) when type = "quadratic"? Ignored for type = "linear".
ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
...	further arguments to plot.default for the plot method. Not used in the slice method.

Value

The slice method returns a list of data.frames with one data.frame for each parameter slice. Each data.frame contains in the first column the values of the parameter and in the second column the values of the (positive) log-likelihood "logLik". A third column is present if quad.approx = TRUE and contains the corresponding quadratic approximation to the log-likelihood. The original model fit is included as the attribute "original.fit".

The plot method produces a plot of the likelihood slice for each parameter.

Author(s)

Rune Haubo B Christensen

Examples

```
## fit model:
fm1 <- clm(rating ~ contact + temp, data = wine)
## slice the likelihood:
sl1 <- slice(fm1)

## three different ways to plot the slices:
par(mfrow = c(2,3))
plot(sl1)
plot(sl1, type = "quadratic", plot.mle = FALSE)
plot(sl1, type = "linear")

## Verify convergence to the optimum:
sl2 <- slice(fm1, lambda = 1e-5, quad.approx = FALSE)
plot(sl2)
```

soup *Discrimination study of packet soup*

Description

The soup data frame has 1847 rows and 13 variables. 185 respondents participated in an A-not A discrimination test with sureness. Before experimentation the respondents were familiarized with the reference product and during experimentation, the respondents were asked to rate samples on an ordered scale with six categories given by combinations of (reference, not reference) and (sure, not sure, guess) from 'referene, sure' = 1 to 'not reference, sure' = 6.

Usage

```
soup
```

Format

RESP factor with 185 levels: the respondents in the study.

PROD factor with 2 levels: index reference and test products.

PRODID factor with 6 levels: index reference and the five test product variants.

SURENESS ordered factor with 6 levels: the respondents ratings of soup samples.

DAY factor with two levels: experimentation was split over two days.

SOUPTYPE factor with three levels: the type of soup regularly consumed by the respondent.

SOUPFREQ factor with 3 levels: the frequency with which the respondent consumes soup.

COLD factor with two levels: does the respondent have a cold?

EASY factor with ten levels: How easy did the respondent find the discrimation test? 1 = difficult, 10 = easy.

GENDER factor with two levels: gender of the respondent.

AGEGROUP factor with four levels: the age of the respondent.

LOCATION factor with three levels: three different locations where experimentation took place.

Source

Data are produced by Unilever Research. Permission to publish the data is granted.

References

Christensen, R. H. B., Cleaver, G. and Brockhoff, P. B.(2011) Statistical and Thurstonian models for the A-not A protocol with and without sureness. *Food Quality and Preference, 22*, pp. 542-549.

VarCorr *Extract variance and correlation parameters*

Description

The VarCorr function extracts the variance and (if present) correlation parameters for random effect terms in a cumulative link mixed model (CLMM) fitted with `clmm`.

Usage

```
VarCorr(x, ...)

## S3 method for class 'clmm'
VarCorr(x, ...)
```

Arguments

x	a `clmm` object.
...	currently not used by the `clmm` method.

Details

The `VarCorr` method returns a list of `data.frames`; one for each distinct grouping factor. Each `data.frame` has as many rows as there are levels for that grouping factor and as many columns as there are random effects for each level. For example a model can contain a random intercept (one column) or a random intercept and a random slope (two columns) for the same grouping factor.

If conditional variances are requested, they are returned in the same structure as the conditional modes (random effect estimates/predictions).

Value

A list of matrices with variances in the diagonal and correlation parameters in the off-diagonal — one matrix for each random effects term in the model. Standard deviations are provided as attributes to the matrices.

Author(s)

Rune Haubo B Christensen

Examples

```
fm1 <- clmm(rating ~ contact + temp + (1|judge), data=wine)
VarCorr(fm1)
```

wine *Bitterness of wine*

Description

The wine data set is adopted from Randall(1989) and from a factorial experiment on factors determining the bitterness of wine. Two treatment factors (temperature and contact) each have two levels. Temperature and contact between juice and skins can be controlled when cruching grapes during wine production. Nine judges each assessed wine from two bottles from each of the four treatment conditions, hence there are 72 observations in all.

Usage

```
wine
```

Format

response scorings of wine bitterness on a 0—100 continuous scale.

rating ordered factor with 5 levels; a grouped version of response.

temp temperature: factor with two levels.

contact factor with two levels ("no" and "yes").

bottle factor with eight levels.

judge factor with nine levels.

Source

Data are adopted from Randall (1989).

References

Randall, J (1989). The analysis of sensory data by generalised linear model. *Biometrical journal 7*, pp. 781–793.

Tutz, G. and W. Hennevogl (1996). Random effects in ordinal regression models. *Computational Statistics & Data Analysis 22*, pp. 537–557.

Examples

```
head(wine)
str(wine)

## Variables 'rating' and 'response' are related in the following way:
(intervals <- seq(0,100, by = 20))
all(wine$rating == findInterval(wine$response, intervals)) ## ok

## A few illustrative tabulations:
```

```
## Table matching Table 5 in Randall (1989):
temp.contact.bottle <- with(wine, temp:contact:bottle)[drop=TRUE]
xtabs(response ~ temp.contact.bottle + judge, data = wine)

## Table matching Table 6 in Randall (1989):
with(wine, {
  tcb <- temp:contact:bottle
  tcb <- tcb[drop=TRUE]
  table(tcb, rating)
})
## or simply: with(wine, table(bottle, rating))

## Table matching Table 1 in Tutz & Hennevogl (1996):
tab <- xtabs(as.numeric(rating) ~ judge + temp.contact.bottle,
             data = wine)
colnames(tab) <-
  paste(rep(c("c","w"), each = 4), rep(c("n", "n", "y", "y"), 2),
        1:8, sep=".")
tab


## A simple model:
m1 <- clm(rating ~ temp * contact, data = wine)
summary(m1)
```

Index