Package ‘cold’

August 25, 2021

Type Package

Title Count Longitudinal Data

Version 2.0-3

Author M. Helena Goncalves and M. Salome Cabral,
apart from a set of Fortran-77 subroutines written by R. Piessens
and E. de Doncker, belonging to the suite ``Quadpack''.

Maintainer M. Helena Goncalves `<mhgoncal@ualg.pt>`

Description Performs regression analysis for longitudinal count data, allowing for serial dependence among observations from a given individual and two dimensional random effects on the linear predictor. Estimation is via maximization of the exact likelihood of a suitably defined model. Missing values and unbalanced data are allowed. Details can be found in the accompanying scientific papers: Goncalves & Cabral (2021, Journal of Statistical Software, <doi:10.18637/jss.v099.i03>) and Goncalves et al. (2007, Computational Statistics & Data Analysis, <doi:10.1016/j.csda.2007.03.002>).

Encoding UTF-8

NeedsCompilation yes

License GPL (>= 2)

LazyLoad yes

LazyData yes

Depends R (>= 3.5.3), methods, stats, graphics, grDevices, utils, cubature, MASS

Repository CRAN

Date/Publication 2021-08-25 10:00:02 UTC

R topics documented:

`cold-package` *Count Longitudinal Data*

Description

Performs Poisson regression analysis for longitudinal count data, allowing for serial dependence among observations from a given individual and two random effects. Estimation is via maximization of the exact likelihood of a suitably defined model. Missing values and unbalanced data are allowed.

Details

This package contains functions to perform the fit of parametric models via likelihood method for count longitudinal data using "S4" classes and methods as implemented in the `methods` package.

Author(s)

M. Helena Gonçalves and M. Salomé Cabral

References

Azzalini, A. (1994). Logistic regression and other discrete data models for serially correlated observations. *J. Ital. Stat. Society*, 3 (2), 169-179. doi: 10.1007/bf02589225.

Gonçalves, M. Helena (2002). *Likelihood methods for discrete longitudinal data*. PhD thesis, Faculty of Sciences, University of Lisbon.

Gonçalves, M. Helena, Cabral, M. Salomé, Ruiz de Villa, M. Carme, Escrich, Eduardo and Solanas, Montse. (2007). Likelihood approach for count data in longitudinal experiments. *Computational Statistics and Data Analysis*, 51, 12, 6511-6520. doi: 10.1016/j.csda.2007.03.002.

Gonçalves, M. Helena and Cabral, M. Salomé. (2021). `cold`: An `R` Package for the Analysis of Count Longitudinal Data. *Journal of Statistical Software*, 99, 3, 1–24. doi: 10.18637/jss.v099.i03.

See Also

`cold-class`, `cold`, `Methods`

`anova-methods` *Methods for function* `anova`

Description

Computes an analysis deviance table for two nested fitted model objects of class `cold`.

Usage

```
## S4 method for signature 'cold'
anova(object, ...)
```

Arguments

`object`	an object of class `cold`.
`...`	an object of class `cold`.

Warning

The comparison between two models by anova will only be valid if they are fitted to the same dataset.

Methods

`signature(object = "ANY")`: Generic function.

`signature(object="cold")`: Anova for `cold` object.

Note

It uses the naive solution of Pinheiro et al. (2000) to calculate the p-value when the difference between the models is the number of random effects.

References

Pinheiro, J.C. and Bates, D.M. (2000). Mixed-Effects Models in S and S-PLUS. *Springer-Verlag*.

Examples

```
#####  data = seizure

seiz1 <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

seiz2 <-cold(y ~ lage + lbase + v4 + trt, data = seizure, start = NULL,
dependence = "AR1")

anova(seiz1, seiz2)

#####  data = datacold

mod0 <- cold(z ~ Time * Treatment, data = datacold, time = "Time",
id = "Subject", dependence = "ind")

mod0R <- cold(z  ~ Time * Treatment, random = ~ 1, data = datacold,
time = "Time", id = "Subject", dependence = "indR")
summary(mod0R)

anova(mod0, mod0R)
```

`bolus` *Bolus data*

Description

The dataset has the number of requests per interval in 12 successive four-hourly intervals following abdominal surgery for 65 patients in a clinical trial to compare two groups (bolus/lock-out combinations).

Usage

`data("bolus")`

Format

A data frame with 780 observations on the following 4 variables.

`id` identifies de number of the individual profile. This vector contains observations of 65 individual profiles.

`group` a factor with levels `1mg` and `2mg`.

`time` a numeric vector that identifies the number of the time points observed.

`y` a numeric vector with the number of analgesic doses taken by hospital patients in 12 successive four-hourly intervals.

Details

The group `2mg` has 30 patients and the group `1mg` has 35 patients.

Source

Weiss, Robert E. (2005). Modeling Longitudinal Data. Springer

https://robweiss.faculty.biostat.ucla.edu/book-data-sets

References

Henderson, R. and Shimakura, S. (2003). A Serially Correlated Gamma Frailty Model for Longitudinal Count Data. *Biometrika*, vol. 90, No. 2, 355–366

Examples

```
data(bolus)

## change the reference class
contrasts(bolus$group)
bolus$group<-relevel(factor(bolus$group), ref = "2mg")
contrasts(bolus$group)

## Weiss, Robert E. (2005) pp 353-356, compare with Table 11.2

bol0R <- cold(y ~ time + group, random = ~ 1, data = bolus,
dependence = "indR")
summary (bol0R)


## reparametrization of time
bolus$time1 <- bolus$time - 6


bol0R1 <- cold(y ~ time1 + group, random = ~ 1,data = bolus,
dependence = "indR")
summary (bol0R1)

bol1R1 <- cold(y ~ time1 + group, random = ~ 1, data = bolus,
time = "time1", dependence = "AR1R")
```

```
summary (bol1R1)

anova(bol0R1, bol1R1)

plot(bol1R1, which = 1, factor = group, ylab = "Bolus count")
```

`coeftest` *Extract summary statistics*

Description

Extract information from poisson regression model objects of class `cold`.

Usage

```
coeftest(object)
```

Arguments

`object`	an object of class `cold`.

Value

Extract a list of summary statistics from poisson regression model corresponding to the fixed effects coefficients.

`coeftest-methods` *Methods for function* `coeftest`

Description

Extract information from poisson regression model objects of class `cold`.

Usage

```
## S4 method for signature 'cold'
coeftest(object)
```

Arguments

`object`	an object of class `cold`.

Value

Extract a list of summary statistics from poisson regression model corresponding to the fixed effects coefficients.

Methods

`signature(object="cold")`: list of summary statistics of fixed effects coefficients for `cold` object.

Examples

```
#####  data = seizure

seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

coeftest(seiz1M)
```

`cold` *Fit of parametric models via likelihood method*

Description

Performs the fit of parametric models via likelihood method. Serial dependence and two random effects are allowed according to the stochastic model chosen. Missing values are automatically accounted for computing the likelihood function.

Usage

```
cold(formula, random, data, id="id", time="time",subSET,
dependence ="ind", start=NULL, method="BFGS", integration="QUADPACK",
M="6000", control=coldControl(), integrate=coldIntegrate(),
cublim=coldcublim(), trace=FALSE)
```

Arguments

`formula`	a description of the model to be fitted of the form response~predictors.
`random`	the predictos that includes random effects of the form response~predictors.
`data`	a `data` frame containing the variables in the formula. NA values are allowed. If data is missing, an error message is produced. See "Details".
`id`	a string that matches the name of the `id` variable in `data`, i.e., the subject identification variable. By default, the program expects a variable named `id` to be present in the `data.frame`, otherwise the name of the variable playing the role of `id` must be declared by assigning `id` here.

`time`	a string that matches the name of the `time` variable in data. By default, the program expects a variable named `time` to be present in the `data.frame`, otherwise the name of the variable playing the role of time must be declared by assigning `time` here.
`subSET`	an optional expression indicating the subset of the rows of `data` that should be used in the fit. All observations are included by default.
`dependence`	expression stating which `dependence` structure should be used in the fit. The default is "ind". According to the stochastic model chosen serial dependence and random effects are allowed. There are six options: "`ind`" (independence), "`AR1`" (first order autoregressive), "`indR`" (independence with random intercept), "`AR1R`" (first order autoregressive with random intercept), "`indR2`" (independence with two random effects) or "`AR1R2`" (first order autoregressive with two random effects).
`start`	a vector of initial values for the nuisance parameters of the likelihood. The dimension of the vector is according to the structure of the dependence model.
`method`	The `method` used in the optimization process: "`BFGS`", "`CG`", "`L-BFGS-B`" and "`SANN`". The default is "`BFGS`". See `optim` for details.
`integration`	The `integration` code allows the user choose the integration method to solve the integrals: "`QUADPACK`" (fortran routines, only for random intercept models), "`cubature`" (uses cubature package to compute integrals when the dependence structure includes one or two random effects), "`MC`" (uses Monte Carlo methods to compute integrals when the dependence structure includes one or two random effects). The default is "`QUADPACK`".
`M`	Number of random points considered to evaluate the integral when the user choose Monte Carlo methods ("`integration`=MC"). The default is set to 6000.
`control`	a list of algorithmic constants for the optimizer `optim`. See R documentation of `optim.control` for details and possible control options. By default, `cold` sets the maximum number of iterations (`maxit`) equal to 100, the absolute convergence tolerance (`abstol`) and the relative convergence tolerance (`rel.tol`) equal to 1e-6 and uses the `optim` standard default values for the remaining options.
`integrate`	a list of algorithmic constants for the computation of a definite integral using a Fortran-77 subroutine. See "Details".
`cublim`	a list of algorithmic constants for the computation of a definite integral when the integration argument is set to cubature.
`trace`	logical flag: if TRUE, details of the nonlinear optimization are printed. By default the flag is set to FALSE.

Details

`data` are contained in a `data.frame`. Each element of the `data` argument must be identifiable by a name. The simplest situation occurs when all subjects are observed at the same time points. If there are missing values in the response variable `NA` values must be inserted. The response variable represent the individual profiles of each subject, it is expected a variable in the `data.frame` that identifies the correspondence of each component of the response variable to the subject that it belongs, by default is named `id` variable. It is expected a variable named `time` to be present in the `data.frame`. If the `time` component has been given a different name, this should be declared. The `time` variable should identify the time points that each individual profile has been observed.

`subSET` is an optional expression indicating the subset of `data` that should be used in the fit. This is a logical statement of the type `variable 1` == "a" & `variable 2` > x which identifies the observations to be selected. All observations are included by default.

For the models with random intercept `indR` and `AR1R`, by default `cold` compute integrals based on a Fortran-77 subroutine package `QUADPACK`. For some data sets, when the `dependence` structure has a random intercept term, the user could have the need to do a specification of the `integrate` argument list changing the integration limits in the `coldIntegrate` function. The `coldIntegrate` is an auxiliary function for controlling `cold` fitting. There are more two options to fit models with a random intercept by setting `integration="cubature"` or `integration="MC"`. For the models with two random effects `indR2` and `AR1R2`, the user has two define the `integration` method by setting `integration="cubature"` or `integration="MC"`. The second random effect is considered to be included in the `time` argument that plays the role of the time variable in the `data.frame`. For the two random effects models we have a random intercept and a random slope.

Value

An object of class `cold`.

Background

Assume that each subject of a given set has been observed at number of successive time points. For each subject and for each time point, a count response variable, and a set of covariates are recorded.

Individual random effects, b_0, can be incorporated in the form of a random intercept term of the linear predictor of the logarithmic regression, assuming a normal distribution of mean 0 and variance σ^2, parameterized as $\omega = \log(\sigma^2)$. The combination of serial Markov `dependence` with a random intercept corresponds here to the `dependence` structures `indR`, `AR1R`.

Two dimensional randoms effects can also be incorporated the linear predictor of the logarithmic regression. Consider a two-dimensional vector of random effects $b = (b_0, b_1)$ where we assumed to be a random sample from the bivariate normal distribution, $b \sim N(0, D)$ with $var(b_0) = \sigma^2_{b_0}$, $var(b_1) = \sigma^2_{b_1}$ and $cov(b_0, b_1) = 0$.

The combination of serial Markov `dependence` with two random effects corresponds here to the `dependence` structures `indR2`, `AR1R2`.

Original sources of the above formulation are given by Azzalini (1994), as for the AR1, and by Gonçalves (2002) and Gonçalves and Azzalini (2008) for the its extensions.

Author(s)

M. Helena Gonçalves and M. Salomé Cabral

References

Azzalini, A. (1994). Logistic regression and other discrete data models for serially correlated observations. *J. Ital. Stat. Society*, 3 (2), 169-179. doi: 10.1007/bf02589225.

Gonçalves, M. Helena (2002). *Likelihood methods for discrete longitudinal data*. PhD thesis, Faculty of Sciences, University of Lisbon.

Gonçalves, M. Helena, Cabral, M. Salomé, Ruiz de Villa, M. Carme, Escrich, Eduardo and Solanas, Montse. (2007). Likelihood approach for count data in longitudinal experiments. *Computational statistics and Data Analysis*, 51, 12, 6511-6520. doi: 10.1016/j.csda.2007.03.002.

Gonçalves, M. Helena and Cabral, M. Salomé. (2021). `cold`: An `R` Package for the Analysis of Count Longitudinal Data. *Journal of Statistical Software*, 99, 3, 1–24. doi: 10.18637/jss.v099.i03.

See Also

`cold-class`, `coldControl`, `coldIntegrate`, `optim`

Examples

```
#####  data = seizure
str(seizure)

### AR1
seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

summary(seiz1M)
getAIC(seiz1M)
getLogLik(seiz1M)

### independence
seiz0M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "ind")

summary(seiz0M)
getAIC(seiz0M)
getLogLik(seiz0M)


#####  data= datacold
str(datacold)

### AR1R with the default integration method
mod1R <- cold(z ~ Time * Treatment, random = ~ 1, data = datacold,
time = "Time", id = "Subject", dependence = "AR1R")

summary (mod1R)
vareff(mod1R)
randeff(mod1R)

### AR1R with integration="cubature"

mod1R.c <- cold(z ~ Time*Treatment, random = ~ 1, data = datacold,
time = "Time", id = "Subject", dependence = "AR1R", integration = "cubature")
summary (mod1R.c)
```

`cold-class` *Class* `cold` *of a maximum likelihood estimation*

Description

This class encapsulates results of a maximum likelihood procedure.

Objects from the Class

Objects can be created by calls of the form `new("cold",...)`, but most often as the result of a call to `cold`.

Slots

`coefficients`**:** Object of class `"matrix"`. Estimated parameters.

`se`**:** Object of class `"matrix"`. Standard errors of estimated parameters.

`covariance`**:** Object of class `"matrix"`. Covariance of estimated parameters.

`correlation`**:** Object of class `"matrix"`. Correlation of estimated parameters.

`log.likelihood`**:** Object of class `"numeric"`. The value of the log likelihood.

`message`**:** Object of class `"integer"`. A character string giving any additional information returned by the optimizer, or NULL. See `optim` for details.

`n.cases`**:** Object of class `"numeric"`. Number of individual profiles used in the optimization procedure.

`ni.cases`**:** Object of class `"numeric"`. Number of individual profiles in the dataset.

`aic`**:** Object of class `"numeric"`. The Akaike information criterion for a fitted model object.

`Fitted`**:** Object of class `"numeric"`. The fitted values for the estimated parameters.

`bi.estimate`**:** Object of class `"numeric"`. The estimated values for the individual random effects.

`Fitted.av`**:** Object of class `"numeric"`.

`Time`**:** Object of class `"numeric"`. Vector of time points.

`model.matrix`**:** Object of class `"matrix"`. The model matrix.

`y.matrix`**:** Object of class `"matrix"`. The matrix of response values.

`random.matrix`**:** Object of class `"matrix"`. The model matrix of random effects.

`subset.data`**:** Object of class `"data.frame"`. The data subset if considered.

`final.data`**:** Object of class `"data.frame"`. The data set considered.

`y.av`**:** Object of class `"numeric"`. The average of the response value over an individual profile.

`data.id`**:** Object of class `"numeric"`. Vector of individual observations.

`call`**:** Object of class `"language"`. The call to `"cold"`.

Methods

- **anova** `signature(object="cold")`: Anova table.
- **coeftest** `signature(object="cold")`: A list of summary statistics of the fixed effects coefficients.
- **fitted** `signature(object="cold")`: The fitted values of a fitted model.
- **fixeff** `signature(object="cold")`: The values corresponding to the fixed effects of a fitted model.
- **getAIC** `signature(object="cold")`: A numeric value corresponding to the AIC of the fitted model.
- **getcoef** `signature(object="cold")`: The values corresponding to the coefficient estimates of the fitted model.
- **getLogLik** `signature(object="cold")`: A numeric value corresponding to the log-Likelihood of the fitted model.
- **getvcov** `signature(object="cold")`: The variance-covariance matrix of the fitted model.
- **model.mat** `signature(object="cold")`: The fixed effects model matrix of the fitted model.
- **plot** `signature(x="cold",y="missing")`: Plots three type of plots.
- **randeff** `signature(object="cold")`: A data frame corresponding to the conditional random effects of the fitted model.
- **show** `signature(object="cold")`: Display object briefly.
- **summary** `signature(object="cold")`: Generate object summary.
- **vareff** `signature(object="cold")`: Numeric value(s) corresponding to the estimated random effect(s) variance of the fitted model.

`cold-internal` *Internal functions*

Description

Functions for internal usage only.

`coldControl` *Auxiliary for controlling "cold" fitting*

Description

Auxiliary function as user interface for `cold` fitting.

Usage

```
coldControl(maxit=100,abstol=-Inf,reltol=sqrt(.Machine$double.eps))
```

Arguments

`maxit`	maximum number of iterations.
`abstol`	absolute convergence tolerance.
`reltol`	relative convergence tolerance.

Details

See R documentation of `optim` for details of standard default values for the remaining options not considered in `coldControl`.

Value

A list with the arguments as components.

See Also

`cold-class`, `optim`

`coldcublim` *Auxiliary for controlling "cold" fitting*

Description

Auxiliary function as user interface for `cold` fitting.

Usage

```
coldcublim (l1i=-4,l2i=-4,l1s=4,l2s=4, tol=1e-4, maxEval=100)
```

Arguments

`l1i`	lower limit of integration for the log-likelihood.
`l1s`	upper limit of integration for the log-likelihood.
`l2i`	lower limit of integration for the log-likelihood.
`l2s`	upper limit of integration for the log-likelihood.
`tol`	The maximum tolerance.
`maxEval`	The maximum number of function evaluations needed.

Value

A list with the arguments as components.

`coldIntegrate` *Auxiliary for controlling "cold" fitting*

Description

Auxiliary function as user interface for `cold` fitting.

Usage

```
coldIntegrate(li=-4,ls=4, epsabs=.Machine$double.eps^.25,
epsrel=.Machine$double.eps^.25,limit=100,key=6,lig=-4,lsg=4)
```

Arguments

`li`	lower limit of integration for the log-likelihood.
`ls`	upper limit of integration for the log-likelihood.
`epsabs`	absolute accuracy requested.
`epsrel`	relative accuracy requested.
`key`	integer from 1 to 6 for choice of local integration rule for number of Gauss-Kronrod quadrature points. A gauss-kronrod pair is used with: 7 - 15 points if key = 1, 10 - 21 points if key = 2, 15 - 31 points if key = 3, 20 - 41 points if key = 4, 25 - 51 points if key = 5 and 30 - 61 points if key = 6.
`limit`	integer that gives an upperbound on the number of subintervals in the partition of (`li`,`ls`), limit.ge.1.
`lig`	lower limit of integration for the gradient.
`lsg`	upper limit of integration for the gradient.

Details

`coldIntegrate` returns a list of constants that are used to compute integrals based on a Fortran-77 subroutine `dqage` from a Fortran-77 subroutine package `QUADPACK` for the numerical computation of definite one-dimensional integrals. The subroutine `dqage` is a simple globally adaptive integrator in which it is possible to choose between 6 pairs of Gauss-Kronrod quadrature formulae for the rule evaluation component. The source code `dqage` was modified and re-named `dqager`, the change was the introduction of an extra variable that allow, in our Fortran-77 subroutines when have a call to `dqager`, to control for which parameter the integral is computed.

For given values of `li` and `ls`, the above-described numerical integration is performed over the interval (`li`*σ, `ls`*σ), where $\sigma = \exp(\omega)/2$ is associated to the current parameter value ω examined by the `optim` function. In some cases, this integration may generate an error, and the user must suitably adjust the values of `li` and `ls`. In case different choices of these quantities all lead to a successful run, it is recommended to retain the one with largest value of the log-likelihood. Integration of the gradient is regulated similarly by `lig` and `lsg`.

For datasets where the individual profiles have a high number of observed time points (say, more than 30), use `coldIntegrate` function to set the integration limits for the likelihood and for the gradient to small values than the default ones.

When the fitting procedure is complete but the computation of the information matrix produces NaNs, changing in `coldIntegrate` function the default values for the gradient integration limits (`lig` and `lsg`) might solve this problem.

Value

A list with the arguments as components.

See Also

`cold-class`

Examples

```
#####  data= seizure

Integ <- coldIntegrate(li = -3.5, ls = 3.5, lig = -3.5, lsg = 3.5)

### AR1R without patient 207

seizure207 <- seizure[seizure$id != 207, ]

seiz1R1.207 <- cold(y~ lage + lbase + v4 + trt  + trt:lbase,
random = ~ 1, data = seizure207, dependence = "AR1R", integrate = Integ)
summary (seiz1R1.207)
```

`datacold` *Data*

Description

This example is an artificial data.

Usage

```
data(datacold)
```

Format

A data frame with 390 observations on the following 4 variables.

`Subject` identifies de number of the individual profile. This vector contains observations of 30 individual profiles.

`Treatment` a factor with levels `0` and `1`.

`Time` a numeric vector that identifies the number of the time points observed.

`z` a numeric vector representing the response variable.

Examples

```
data(datacold)

mod0<- cold(z~Time*Treatment, data=datacold, time="Time",
id="Subject", dependence="ind")
summary (mod0)

modI<- cold(z~Time*Treatment, data=datacold, time="Time",
id="Subject",  dependence="AR1")
summary (modI)

anova(mod0,modI)

plot(modI,which=1,factor=Treatment,xlab="Time (weeks)",
ylab="Count", main="Model AR1")


### independent with random intercept
mod0R <- cold(z ~ Time * Treatment, random = ~ 1, data = datacold,
time = "Time", id = "Subject", dependence = "indR")
summary(mod0R)

### independent with random intercept (dependence="indR")
### using cubature (integration = "cubature")

mod0R.C <- cold(z ~ Time * Treatment, random = ~ 1, data = datacold,
time = "Time", id = "Subject", dependence = "indR", integration = "cubature")
summary(mod0R.C)

randeff(mod0R.C)




### dependence="indR2"
## It takes a long time to run

## Using Monte Carlo method (integration = "MC")
mod0R2MC <- cold(z ~ Time * Treatment, ~ 1 + Time, datacold, time = "Time",
id = "Subject", dependence = "indR2", integration = "MC")
```

```
summary (mod0R2MC)

randeff(mod0R2MC)

## Using cubature (integration = "cubature")
mod0R2C<-cold(z ~ Time * Treatment, random = ~ 1 + Time, data = datacold,
time = "Time", id = "Subject", dependence = "indR2", integration = "cubature")
summary (mod0R2C)
```

`datacoldM` *Data with missing values*

Description

This example is an artificial data with missing values.

Usage

```
data("datacoldM")
```

Format

A data frame with 390 observations on the following 4 variables.

`Subject` identifies de number of the individual profile.This vector contains observations of 30 individual profiles.

`Treatment` a factor with levels `0` and `1`.

`Time` a numeric vector that identifies the number of the time points observed.

`z` a numeric vector representing the response variable.

Examples

```
data(datacoldM)
 str(datacoldM)

mod0.M<- cold(z~Time*Treatment, data=datacoldM, time="Time",
id="Subject", dependence="ind")
summary (mod0.M)

mod1.M<- cold(z~Time*Treatment, data=datacoldM, time="Time",
id="Subject", dependence="AR1")
summary (mod1.M)

anova(mod0.M,mod1.M)
```

`fitted-methods` *Methods for function* `fitted`

Description

Methods for function `fitted` extracting fitted values of a fitted model object from class `cold`.

Usage

```
## S4 method for signature 'cold'
fitted(object)
```

Arguments

`object`	an object of class `cold`.

Methods

`signature(object="cold")`**:** fitted for `cold` object.

Examples

```
#####  data = seizure

seiz1M<-cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

fitted(seiz1M)[1:16]
```

`fixeff` *Extract fixed effects estimates*

Description

Methods for function `fixeff` extracting fixed effects estimates of a fitted model object from class `cold`.

Usage

```
fixeff(object)
```

Arguments

`object`	an object of class `cold`.

Value

Extract fixed effects estimates.

`fixeff-methods` *Methods for function* `fixeff`

Description

Methods for function `fixeff` extracting fixed effects estimates of a fitted model object from class `cold`.

Methods

`signature(object="cold")`: fixed effects estimates of a fitted model object.

Examples

```
#####  data = seizure

### AR1
seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

fixeff(seiz1M)

### indR
seiz0R<-cold(y ~ lage + lbase + trt + trt:lbase + v4, random = ~ 1,
        data = seizure, dependence = "indR")

fixeff(seiz0R)
```

`getAIC` *Extract the Akaike Information Criterion*

Description

Methods for function `getAIC` extracting the Akaike information criterion of the fitted model object from class `cold`.

Usage

```
getAIC(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns a numeric value corresponding to the AIC of the fitted model.

---

getAIC-methods *Methods for function* `getAIC`

Description

Methods for function `getAIC` extracting the Akaike information criterion of the fitted model object from class `cold`.

Usage

```
## S4 method for signature 'cold'
getAIC(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns a numeric value corresponding to the AIC of the fitted model.

Methods

getAIC `signature(object="cold")`: Returns a numeric value corresponding to the AIC of the fitted model.

Examples

```
#####  data = seizure

### AR1
seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

getAIC(seiz1M)

### indR
seiz0R <- cold(y ~ lage + lbase + trt + trt:lbase + v4, random = ~ 1,
        data = seizure, dependence = "indR")

getAIC(seiz0R)
```

`getcoef` *Extract the coefficient estimates*

Description

Methods for function `getcoef` extracting the coefficient estimates of the fitted model object from class `cold`.

Usage

```
getcoef(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns the coefficient estimates of the fitted model.

`getcoef-methods` *Methods for function* `getcoef`

Description

Methods for function `getcoef` extracting the coefficient estimates of the fitted model object from class `cold`.

Methods

`signature(object="cold")`: Returns the coefficient estimates of the fitted model.

Examples

```
#####  data = seizure

### AR1
seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

getcoef(seiz1M)

### indR
seiz0R <- cold(y ~ lage + lbase + trt + trt:lbase + v4, random = ~ 1,
        data = seizure, dependence = "indR")

getcoef(seiz0R)
```

`getLogLik` *Extract Log-Likelihood*

Description

Extract the Log-Likelihood of a fitted model object from class `cold`.

Usage

```
getLogLik(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns a numeric value corresponding to the log-Likelihood of the fitted model.

`getLogLik-methods` *Methods for function* `getLogLik`

Description

Extract the Log-Likelihood of a fitted model object from class `cold`.

Usage

```
## S4 method for signature 'cold'
getLogLik(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns a numeric value corresponding to the log-Likelihood of the fitted model.

Methods

`signature(object="cold")`: Returns a numeric value corresponding to the log-Likelihood of the fitted model.

Examples

```
#####  data = seizure

### AR1R
seiz1M<-cold(y~lage+lbase+v4+trt+trt:lbase, data=seizure, start=NULL,
dependence="AR1")

getLogLik(seiz1M)

### indR
seiz0R<-cold(y ~ lage + lbase + trt + trt:lbase + v4, random = ~ 1,
        data = seizure, dependence = "indR")

getLogLik(seiz0R)
```

`getvcov` *Extract variance-covariance matrix*

Description

Extract the variance-covariance matrix of a fitted model object from class `cold`.

Usage

```
getvcov(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns a numeric value corresponding to the variance-covariance matrix.

`getvcov-methods` *Methods for function* `getvcov`

Description

Extract the variance-covariance matrix of a fitted model object from class `cold`.

Methods

`signature(object="cold")`**:** Returns a numeric value corresponding to the variance-covariance matrix of the fixed effect estimates.

Examples

```
#####  data = seizure

### AR1
seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

getvcov(seiz1M)
```

`model.mat` *Extract the fixed effects model matrix*

Description

Methods for function `model.mat` extracting the fixed effects model matrix for a fitted model object from class `cold`.

Usage

```
model.mat(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns a numeric value corresponding to the fixed effects model matrix.

`model.mat-methods` *Methods for function* `model.mat`

Description

Methods for function `model.mat` extracting the fixed effects model matrix for a fitted model object from class `cold`.

Methods

`signature(object="cold")`: Returns the fixed effects model matrix of the fitted model.

Examples

```
#####  data = seizure

### AR1
seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

model.mat(seiz1M)[1:20,]
```

`plot-methods` *Methods for function* `plot`

Description

Three plots (selectable by `which`) are currently available: a plot for the fitted model (`which`=1), a plot for the individual mean profile (`which`=2) and a plot for the observed data and the corresponded mean profile (`which`=3).

Usage

```
## S4 method for signature 'cold,missing'
plot(x,which=c(1:3),xlab=NULL,ylab=NULL,
main=NULL,factor,subSET,ident=FALSE,
caption=c("Individual mean profiles"),cex.caption=1)
```

Arguments

`x`	an object of class `cold`.
`which`	if a subset of the plots is required, specify a subset of the numbers 1:3. Default is (`which`=1).
`xlab`	label to plots.
`ylab`	label to plots.
`factor`	identify the factor for `which`=1.
`main`	title to plots in addition to the caption.
`subSET`	logical expression indicating elements to keep in individual mean profile plots: missing values are taken as FALSE for `which`=2.
`ident`	logical expression indicating whether or not to add the number of the subject to individual mean profile plots. The `ident` argument is for option `which`=2.
`caption`	captions to appear above the plots.
`cex.caption`	controls the size of caption.

Methods

`signature(x="ANY", y="ANY")`: Generic function.

`signature(x="cold", y="missing")`: Plot diagnostics for `cold` object.

Examples

```
#####  data= datacold
### AR1R
mod1R <- cold(z ~ Time * Treatment, data = datacold, time = "Time",
id = "Subject", dependence = "AR1R")

plot(mod1R, which = 1, xlab = "Time(weeks)", ylab = "Count",
factor = Treatment, main = "Model AR1R")

plot(mod1R, which = 2, xlab = "Time(weeks)", ylab = "Count",
main = "Model AR1R")

par(mfrow = c(1, 2))
plot(mod1R, which = 2, ident = TRUE, subSET = Treatment == "0",
ylab = "Count", main = "0")
plot(mod1R, which = 2, ident = TRUE, subSET = Treatment == "1",
ylab = "Count", main = "1")
par(mfrow = c(1, 1))

par(mfrow = c(2, 2))
plot(mod1R, which = 3, subSET = (id == c(2)), xlab = "Time (weeks)",
ylab = "count", main = "0_Subject2")
plot(mod1R, which = 3, subSET = (id == c(10)), xlab = "Time (weeks)",
ylab = "Count", main = "0_Subject10")
plot(mod1R, which = 3, subSET = (id == c(26)), xlab = "Time (weeks)",
ylab = "Count", main = "1_Subject26")
plot(mod1R, which=3, subSET=(id == c(30)), xlab = "Time (weeks)",
ylab = "Count", main = "1_Subject32")
par(mfrow = c(1, 1))
```

`randeff` *Extract conditional random effects*

Description

Methods for function `randeff` extracting conditional random effects of a fitted model object from class `cold`.

Usage

```
randeff(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns a data.frame corresponding to the conditional random effects of the fitted model.

`randeff-methods` *Methods for function* `randeff`

Description

Methods for function `randeff` extracting conditional random effects of a fitted model object from class `cold`.

Methods

`signature(object="cold")`: `randeff` for `cold` object.

Examples

```
#####  data = seizure

### indR
seiz0R <- cold(y ~ lage + lbase + trt + trt:lbase + v4, random = ~ 1,
        data = seizure, dependence = "indR")

randeff(seiz0R)
```

`resid-methods` *Methods for function* `residd`

Description

Methods for function `resid` extracting residual values of a fitted model object from class `cold`.

Usage

```
## S4 method for signature 'cold'
resid(object, type = c( "pearson","response","null"),...)
```

Arguments

`object`	an object of class `cold`.
`type`	an optional character string specifying the type of residuals to be used. Two types are allowed: pearson and response. Defaults to "pearson".
`...`	other arguments.

Methods

`signature(object="cold")`: residuals for `cold` object.

Examples

```
#####  data = seizure

seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
start = NULL, dependence = "AR1")

resid(seiz1M)[1:16]
```

`seizure` *Epileptic Seizure*

Description

The dataset has the number of epileptic seizures in each of four two-week intervals, and in a baseline eight-week interval, for treatment and control groups with a total of 59 individuals.

Usage

`data(seizure)`

Format

A data frame with 236 observations on the following 9 variables.

- `id` identifies de number of the individual profile. This vector contains observations of 59 individual profiles.
- `y` a numeric vector with the number of epileptic seizures in the four two-weeks intervals observed.
- `v4` a numeric vector indicating the fourth visit.
- `time` a numeric vector that identifies the number of the time points observed.
- `trt` a numeric vector indicator of treatment, whether the patient is treated with placebo (`trt=0`) or progabide (`trt=1`).
- `base` the number of epileptic seizures in a baseline 8-week interval.
- `age` a numeric vector of subject `age`.
- `lbase` recode the variable `base` by log(`base`/4).
- `lage` recode the variable `age` by log(`age`).

Source

Thall, P.F., and Vail, S.C. (1990). Some covariance models for longitudinal count data with overdispersion. *Biometrics*, 46, 657–671.

References

Diggle, P.J., Heagerty, P., Liang, K.Y., and Zeger, S.L. (2002). Analysis of Longitudinal Data. 2nd edition. Oxford University Press.

Examples

```
#####  data = seizure
str(seizure)

### independence
seiz0M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
dependence = "ind")
summary(seiz0M)

### AR1
seiz1M <- cold(y ~ lage + lbase + v4 + trt + trt:lbase, data = seizure,
dependence = "AR1")
summary(seiz1M)

anova(seiz0M, seiz1M)
```

`show-methods` *Methods for function* `show`

Description

Show objects of class `cold` and `summary.cold`.

Methods

`signature(object = "cold")` Print simple summary of a `cold` object, just the call, the number of profiles in the fit, the number of coefficients, the value of the log-likelihood and a message giving additional information returned by the optimizer.

`signature(object = "summary.cold")` Shows call, the number of profiles in the fit, table of coefficients, standard errors and p-values, the log-likelihood, the AIC coefficient, and a message giving additional information returned by the optimizer.

`summary-methods` *Methods for function* `summary`

Description

Summarize objects.

Usage

```
## S4 method for signature 'cold'
summary(object, cov=FALSE, cor=FALSE)
```

Arguments

`object`	an object of class `cold`.
`cov`	if set to TRUE prints the matrix of covariances between parameters estimates. The default is FALSE.
`cor`	if set to TRUE prints the matrix of correlations between parameters estimates. The default is FALSE.

Details

Computes and returns a list of summary statistics of the fitted model given a `cold` object, using the components (list elements) "call" and "terms" from its argument, plus depending on the structure of the dependence model chosen. In the table for coefficient estimates will appear `rho` if the dependence structure of the process corresponds to an `AR1`, `AR1R` or `AR1R2`. If the structure of the dependence model chosen includes the random intercept (models `"indR"` and `"AR1R"`), the variance estimate for random intercept is given. If the structure of the dependence model chosen includes two random effects (models `"indR2"` and `"AR1R2"`) the variance estimates for random intercept and for the slope (time) are given.

Methods

`signature(object = "ANY")`: Generic function.

`signature(object = "cold")`: Prints a summary as an object of class `summary.cold`, containing information about the matched call to `cold`, the number of profiles in the data, the number of profiles used in the fit, the log-likelihood, the AIC, a table with estimates, asymptotic SE, t-values and p-values, the estimated correlation and variance-covariance matrix for the estimated parameters if the user wishes, and a message giving additional information returned by the optimizer.

`summary.cold-class` *Class* `summary.cold`*, summary of* `cold` *objects*

Description

Extract of `cold` object.

Objects from the class

Objects can be created by calls of the form `new("summary.cold",...)`, but most often by invoking `summary` on an `cold` object. They contain values meant for printing by `show`.

Slots

`coefficients`**:** Object of class `"matrix"`. Estimated parameters.

`se`**:** Object of class `"matrix"`. Standard errors of estimated parameters.

`covariance`**:** Object of class `"matrix"`. Covariance of estimated parameters.

`correlation`**:** Object of class `"matrix"`. Correlation of estimated parameters.

`log.likelihood`**:** Object of class `"numeric"`. The value of the log likelihood.

`message`**:** Object of class `"integer"`. A character string giving any additional information returned by the optimizer, or NULL. See `optim` for details.

`n.cases`**:** Object of class `"numeric"`. Number of individual profiles used in the optimization procedure.

`ni.cases`**:** Object of class `"numeric"`. Number of individual profiles in the dataset.

`aic`**:** Object of class `"numeric"`. The Akaike information criterion for a fitted model object.

`call`**:** Object of class `"language"`. The `call` that generated `cold` object.

Extends

Class `"cold"`, directly.

Methods

show `signature(object = "summary.cold")`: Pretty-prints object.

See Also

`cold`, `cold-class`

`vareff` *Extract variance of random effects*

Description

Methods for function `vareff` extracting the variance of random effects of a fitted model object.

Usage

```
vareff(object)
```

Arguments

`object`	an object of class `cold`.

Value

Returns the variance estimates of random effects of a fitted model.

`vareff-methods` *Methods for function* `vareff`

Description

Methods for function `vareff` extracting the variance estimates of random effects of a fitted model object.

Methods

`signature(object="cold")`: `vareff` for `cold` object.

Examples

```
#####  data = seizure

### indR
seiz0R <- cold(y ~ lage + lbase + trt + trt:lbase + v4, random = ~ 1,
        data = seizure, dependence = "indR")

vareff(seiz0R)
```

Index