Package ‘repeated’

July 14, 2019

Version 1.1.2

Title Non-Normal Repeated Measurements Models

Depends R (>= 1.4), rmutil

Description Various functions to fit models for non-normal repeated
measurements, such as Binary Random Effects Models with Two Levels of Nesting,
Bivariate Beta-binomial Regression Models, Marginal Bivariate Binomial Regression Models,
Cormack capture-recapture models, Continuous-time Hidden Markov Chain Models,
Discrete-time Hidden Markov Chain Models,
Changepoint Location Models using a Continuous-time Two-state Hidden Markov Chain,
generalized nonlinear autoregression models, multivariate Gaussian copula models,
generalized non-linear mixed models with one random effect,
generalized non-linear mixed models using h-likelihood for one random effect,
Repeated Measurements Models for Counts with Frailty or Serial Dependence,
Repeated Measurements Models for Continuous Variables with Frailty or Serial Dependence,
Ordinal Random Effects Models with Dropouts, marginal homogeneity models for square
contingency tables, correlated negative binomial models with Kalman update.
References include Lindey's text books,
JK Lindsey (2001) <isbn-10:0198508123> and JK Lindsey (1999) <isbn-10:0198505590>.

License GPL-2

URL http://www.commanster.eu/rcode.html

BugReports https://github.com/swihart/repeated/issues

Encoding UTF-8

LazyData true

LazyLoad true

RoxygenNote 6.1.1

NeedsCompilation yes

Author Bruce Swihart [cre, aut],
Jim Lindsey [aut] (Jim created this package, Bruce is maintaining the
CRAN version),
T.R. Ten Have [ctb, cph] (Wrote Logit_bin_nest.f90 (in binnest.f).),
Richard Cook [ctb, cph] (Wrote calcs.c, calcs.h, calcs1.c, calcs1.h,
calcs2.c, calcs2.h, calcs3.c, calcs3.h, calcs4.c, calcs4.h; defs.h;

gaps.c, gaps.h.),
Iain MacDonald [ctb, cph] (Wrote chidden.f, cphidden.f, hidden.f.),
Walter Zucchini [ctb, cph] (Wrote chidden.f, cphidden.f, hidden.f.),
Burton Garbow [ctb, cph] (Wrote eigen.f.),
Euginia Zharichenko [ctb, cph] (Wrote logitord.f.)

Maintainer Bruce Swihart <`bruce.swihart@gmail.com`>

Repository CRAN

Date/Publication 2019-07-13 23:30:03 UTC

R topics documented:

`binnest` *Binary Random Effects Models with Two Levels of Nesting*

Description

`binnest` is designed to handle binary and binomial data with two levels of nesting. The first level is the individual and the second will consist of clusters within individuals.

Usage

```
binnest(response, totals = NULL, nest = NULL, ccov = NULL,
  tvcov = NULL, mu = ~1, re1 = ~1, re2 = ~1, preg = NULL,
  pre1 = NULL, pre2 = NULL, binom.mix = c(10, 10),
  binom.prob = c(0.5, 0.5), fcalls = 900, eps = 0.01,
  print.level = 0)
```

Arguments

`response`	A list of three column matrices with counts, corresponding totals (not necessary if the response is binary), and (second-level) nesting indicator for each individual, one matrix or dataframe of such counts, or an object of class, response (created by `restovec`) or repeated (created by `rmna`).
`totals`	If `response` is a matrix or dataframe, a corresponding matrix or dataframe of totals (not necessary if the response is binary). Ignored otherwise.
`nest`	If `response` is a matrix or dataframe, a corresponding matrix or dataframe of nesting indicators. Ignored otherwise.
`ccov`	If `response` is a matrix, dataframe, list, or object of class, `response`, a matrix of time-constant covariates or an object of class, `tccov` (created by `tcctomat`). All of these covariates are used in the fixed effects part of the model. Ignored if response has class, `repeated`.
`tvcov`	If `response` is a matrix, dataframe, list, or object of class, `response`, an object of class, `tvcov` (created by `tvctomat`). All of these covariates are used in the fixed effects part of the model. Ignored if response has class, `repeated`.
`mu`	If `response` has class, `repeated`, a formula beginning with ~, specifying a linear regression function for the fixed effects, in the Wilkinson and Rogers notation, containing selected covariates in the response object. (A logit link is assumed.)
`re1`	If `response` has class, `repeated`, a formula beginning with ~, specifying a linear regression function for the variance of the first level of nesting, in the Wilkinson and Rogers notation, containing selected covariates in the response object. If NULL, a random effect is not fitted at this level. (A log link is assumed.)
`re2`	If `response` has class, `repeated`, a formula beginning with ~, specifying a linear regression function for the variance of the second level of nesting, in the Wilkinson and Rogers notation, containing selected covariates in the response object. If NULL, a random effect is not fitted at this level. (A log link is assumed.)
`preg`	Initial parameter estimates for the fixed effect regression model: either the model specified by `mu` or else the intercept plus one for each covariate in `ccov` and `tvcov`.
`pre1`	Initial parameter estimates for the first level of nesting variance model: either the model specified by `re1` or just the intercept. If NULL, a random effect is not fitted at this level.
`pre2`	Initial parameter estimates for the second level of nesting variance model: either the model specified by `re1` or just the intercept. If NULL, a random effect is not fitted at this level.

`binom.mix`	A vector of two values giving the totals for the binomial distributions used as the mixing distributions at the two levels of nesting.
`binom.prob`	A vector of two values giving the probabilities in the binomial distributions used as the mixing distributions at the two levels of nesting. If they are 0.5, the mixing distributions approximate normal mixing distributions; otherwise, they are skewed.
`fcalls`	Number of function calls allowed.
`eps`	Convergence criterion.
`print.level`	If 1, the iterations are printed out.

Details

The variance components at the two levels can only depend on the covariates if `response` has class, `repeated`.

Value

A list of classes `binnest` is returned.

Author(s)

T.R. Ten Have and J.K. Lindsey

References

Ten Have, T.R., Kunselman, A.R., and Tran, L. (1999) Statistics in Medicine 18, 947-960.

See Also

`gar`, `read.list`, `restovec`, `rmna`, `tcctomat`, `tvctomat`.

Examples

```
#y <- rbind(matrix(rbinom(20,1,0.6), ncol=4),
# matrix(rbinom(20,1,0.4), ncol=4))
y <- matrix(c(1,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1,1,0,0,1,0,1,1,0,1,0,
1,1,1,1,1,1,1,1,0,1,1,0),nrow=10,ncol=4,byrow=TRUE)
resp <- restovec(y, nest=1:4, times=FALSE)
ccov <- tcctomat(c(rep(0,5),rep(1,5)), name=”treatment”)
reps <- rmna(resp, ccov=ccov)

# two random effects
binnest(reps, mu=~treatment, preg=c(1,0), pre1=1, pre2=1)

# first level random effect only
binnest(reps, mu=~treatment, preg=c(1,-1), pre1=1)

# second level random effect only
binnest(reps, mu=~treatment, preg=c(1,-1), pre2=1)
```

`biv.betab` *Bivariate Beta-binomial Regression Models*

Description

`biv.betab` fits dependent (logit) linear regression models to a bivariate beta-binomial distribution.

Usage

```
biv.betab(freq, x = NULL, p, depend = TRUE, print.level = 0,
  typsize = abs(p), ndigit = 10, gradtol = 1e-05, stepmax = 10 *
  sqrt(p %*% p), steptol = 1e-05, iterlim = 100, fscale = 1)
```

Arguments

`freq`	A matrix containing four columns corresponding to 00, 01, 10, and 11 responses.
`x`	A matrix of explanatory variables, containing pairs of columns, one for each response, and the same number of rows as freq.
`p`	Initial parameter estimates: intercept, dependence (if depend is TRUE, and one for each pair of columns of x.
`depend`	If FALSE, the independence (logistic) model is fitted.
`print.level`	Arguments for nlm.
`typsize`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`stepmax`	Arguments for nlm.
`steptol`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`fscale`	Arguments for nlm.

Value

A list of class `bivbetab` is returned.

Author(s)

J.K. Lindsey

Examples

```
y <- matrix(  c( 2, 1, 1,13,
 4, 1, 3, 5,
 3, 3, 1, 4,
15, 8, 1, 6),ncol=4,byrow=TRUE)
first <- c(0,0,1,1)
second <- c(0,1,0,1)
self <- cbind(first,second)
other <- cbind(second,first)
biv.betab(y,cbind(self,other),p=c(-1,2,1,1))
# independence
biv.betab(y,cbind(self,other),p=c(-1,1,1),dep=FALSE)
```

`biv.binom` *Marginal Bivariate Binomial Regression Models*

Description

`biv.binom` fits (logit) linear regression models to a marginal bivariate binomial distribution. The covariates must be of length K, that is the number of 2x2 tables.

Usage

```
biv.binom(freq, marg1 = ~1, marg2 = ~1, interaction = ~1,
  pmarg1 = 1, pmarg2 = 1, pinteraction = 1, print.level = 0,
  typsize = abs(p), ndigit = 10, gradtol = 1e-05, stepmax = 10 *
  sqrt(p %*% p), steptol = 1e-05, iterlim = 100, fscale = 1)
```

Arguments

`freq`	A four-column matrix containing K 2x2 frequency tables.
`marg1`	The model formula for the first margin.
`marg2`	The model formula for the second margin.
`interaction`	The model formula for the interaction.
`pmarg1`	Initial parameter estimates for the first margin regression.
`pmarg2`	Initial parameter estimates for the second margin regression.
`pinteraction`	Initial parameter estimates for the interaction regression.
`print.level`	Arguments for nlm.
`typsize`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`stepmax`	Arguments for nlm.

`steptol`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`fscale`	Arguments for nlm.

Value

A list of class `bivbinom` is returned.

Author(s)

J.K. Lindsey

Examples

```
# 5 2x2 tables
Freq <- matrix(rpois(20,10),ncol=4)
x <- c(6,8,10,12,14)
print(z <- biv.binom(Freq,marg1=~x,marg2=~x,inter=~x,pmarg1=c(-2,0.08),
pmarg2=c(-2,0.1),pinter=c(3,0)))
```

`capture` *Capture-recapture Models*

Description

`capture` fits the Cormack capture-recapture model to `n` sample periods. Set `n` to the appropriate value and type `eval(setup)`.

Usage

```
capture(z, n)
```

Arguments

`z`	A Poisson generalized linear model object.
`n`	The number of repeated observations.

Details

`n <- periods` # number of periods

`eval(setup)`

This produces the following variables -

`p[i]`: logit capture probabilities,

`pbd`: constant capture probability,

`d[i]`: death parameters,

`b[i]`: birth parameters,

`pw`: prior weights.

Then set up a Poisson model for log linear models:

`z <- glm(y~model, family=poisson, weights=pw)`

and call the function, `capture`.

If there is constant effort, then all estimates are correct. Otherwise, `n[1]`, `p[1]`, `b[1]`, are correct only if there is no birth in period 1. `n[s]`, `p[s]`, are correct only if there is no death in the last period. `phi[s-1]` is correct only if effort is constant in `(s-1, s)`. `b[s-1]` is correct only if `n[s]` and `phi[s-1]` both are.

Value

`capture` returns a matrix containing the estimates.

Author(s)

J.K. Lindsey

Examples

```
y <- c(0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,14,1,1,0,2,1,2,1,16,0,2,0,11,
2,13,10,0)
n <- 5
eval(setup)
# closed population
print(z0 <- glm(y~p1+p2+p3+p4+p5, family=poisson, weights=pw))
# deaths and emigration only
print(z1 <- update(z0, .~.+d1+d2+d3))
# immigration only
print(z2 <- update(z1, .~.-d1-d2-d3+b2+b3+b4))
# deaths, emigration, and immigration
print(z3 <- update(z2, .~.+d1+d2+d3))
# add trap dependence
print(z4 <- update(z3, .~.+i2+i3))
# constant capture probability over the three middle periods
print(z5 <- glm(y~p1+pbd+p5+d1+d2+d3+b2+b3+b4, family=poisson, weights=pw))
# print out estimates
capture(z5, n)
```

---

catmiss *Marginal Probabilities for Categorical Repeated Measurements with Missing Data*

Description

`catmiss` calculates the marginal probabilities of repeated responses. If there are missing values, it gives both the complete data estimates and the estimates using all data. It is useful, for example, when a log linear model is fitted; the resulting fitted values can be supplied to `catmiss` to obtain the estimates of the marginal probabilities for the model. (Note however that the standard errors do not take into account the fitting of the model.)

Usage

```
catmiss(response, frequency, ccov = NULL)
```

Arguments

`response`	A matrix with one column for each of the repeated measures and one row for each possible combination of responses, including the missing values, indicated by NAs.
`frequency`	A vector containing the frequencies. Its length must be a multiple of the number of rows of `response`. Responses are arranged in blocks corresponding to the various possible combinations of values of the explanatory variables.
`ccov`	An optional matrix containing the explanatory variables (time-constant covariates) as columns, with one line per block of responses in `frequency`. Thus, the number of rows of response times the number of rows of `ccov` equals the length of `frequency`.

Value

A matrix with the probabilities and their standard errors is returned.

Author(s)

J.K. Lindsey

Examples

```
y <- rpois(27,15)
r1 <- gl(3,1,27)
r2 <- gl(3,3,27)
r3 <- gl(3,9)
# r1, r2, and r3 are factor variables with 3 indicating missing
# independence model with three binary repeated measures
# with missing values
print(z <- glm(y~r1+r2+r3, family=poisson))
```

```
# obtain marginal estimates (no observations with 3 missing values)
resp <- cbind(as.integer(r1), as.integer(r2), as.integer(r3))[1:26,]
resp <- ifelse(resp==3, NA, resp)
catmiss(resp, y[1:26])
```

`chidden` *Continuous-time Hidden Markov Chain Models*

Description

`chidden` fits a two or more state hidden Markov chain model with a variety of distributions in continuous time. All series on different individuals are assumed to start at the same time point. If the time points are equal, discrete steps, use `hidden`.

Usage

```
chidden(response = NULL, totals = NULL, times = NULL,
  distribution = "Bernoulli", mu = NULL, cmu = NULL, tvmu = NULL,
  pgamma, pmu = NULL, pcmu = NULL, ptvmu = NULL, pshape = NULL,
  pfamily = NULL, par = NULL, pintercept = NULL, delta = NULL,
  envir = parent.frame(), print.level = 0, ndigit = 10,
  gradtol = 1e-05, steptol = 1e-05, fscale = 1, iterlim = 100,
  typsize = abs(p), stepmax = 10 * sqrt(p %*% p))
```

Arguments

`response`	A list of two or three column matrices with counts or category indicators, times, and possibly totals (if the distribution is binomial), for each individual, one matrix or dataframe of counts, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here. If there is only one series, a vector of responses may be supplied instead. Multinomial and ordinal categories must be integers numbered from 0.
`totals`	If response is a matrix, a corresponding matrix of totals if the distribution is binomial. Ignored if response has class, `response` or `repeated`.
`times`	If `response` is a matrix, a vector of corresponding times, when they are the same for all individuals. Ignored if response has class, `response` or `repeated`.
`distribution`	Bernoulli, Poisson, multinomial, proportional odds, continuation ratio, binomial, exponential, beta binomial, negative binomial, normal, inverse Gauss, logistic, gamma, Weibull, Cauchy, Laplace, Levy, Pareto, gen(eralized) gamma, gen(eralized) logistic, Hjorth, Burr, gen(eralized) Weibull, gen(eralized) extreme value, gen(eralized) inverse Gauss, power exponential, skew Laplace, Student t, or (time-)discretized Poisson process. (For definitions of distributions, see the corresponding [dpqr]distribution help.)

`mu`	A general location function with two possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per observation; or (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that pmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial).
`cmu`	A time-constant location function with three possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per individual; (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that pcmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial); or (3) a function returning an array with one row for each individual, one column for each state of the hidden Markov chain, and, if multinomial, one layer for each category but the last. If used, this function or formula should contain the intercept. Ignored if `mu` is supplied.
`tvmu`	A time-varying location function with three possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per time point; (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that ptvmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial); or (3) a function returning an array with one row for each time point, one column for each state of the hidden Markov chain, and, if multinomial, one layer for each category but the last. This function or formula is usually a function of time; it is the same for all individuals. It only contains the intercept if `cmu` does not. Ignored if `mu` is supplied.
`pgamma`	A square `mxm` matrix of initial estimates of the continuous-time hidden Markov transition matrix, where `m` is the number of hidden states. Rows can either sum to zero or the diagonal elements can be zero, in which case they will be replaced by minus the sum of the other values on the rows. If the matrix contains zeroes off diagonal, these are fixed and not estimated.
`pmu`	Initial estimates of the unknown parameters in `mu`.
`pcmu`	Initial estimates of the unknown parameters in `cmu`.
`ptvmu`	Initial estimates of the unknown parameters in `tvmu`.
`pshape`	Initial estimate(s) of the dispersion parameter, for those distributions having one. This can be one value or a vector with a different value for each state.
`pfamily`	Initial estimate of the family parameter, for those distributions having one.
`par`	Initial estimate of the autoregression parameter.
`pintercept`	For multinomial, proportional odds, and continuation ratio models, `p-2` initial estimates for intercept contrasts from the first intercept, where `p` is the number of categories.

`delta`	Scalar or vector giving the unit of measurement (always one for discrete data) for each response value, set to unity by default. For example, if a response is measured to two decimals, delta=0.01. If the response is transformed, this must be multiplied by the Jacobian. For example, with a log transformation, `delta=1/response`. Ignored if response has class, `response` or `repeated`.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`fscale`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`typsize`	Arguments for nlm.
`stepmax`	Arguments for nlm.

Details

The time-discretized Poisson process is a continuous-time hidden Markov model for Poisson processes where time is then discretized and only presence or absence of one or more events is recorded in each, perhaps unequally-spaced, discrete interval.

For quantitative responses, specifying `par` allows an ‘observed’ autoregression to be fitted as well as the hidden Markov chain.

All functions and formulae for the location parameter are on the (generalized) logit scale for the Bernoulli, binomial, and multinomial distributions. Those for intensities of the discretized Poisson process are on the log scale.

If `cmu` and `tvmu` are used, these two mean functions are additive so that interactions between time-constant and time-varying variables are not possible.

The algorithm will run more quickly if the most frequently occurring time step is scaled to be equal to unity.

The object returned can be plotted to give the probabilities of being in each hidden state at each time point. See `hidden` for details. For distributions other than the multinomial, proportional odds, and continuation ratio, the (recursive) predicted values can be plotted using `mprofile` and `iprofile`.

Value

A list of classes `hidden` and `recursive` (unless multinomial, proportional odds, or continuation ratio) is returned that contains all of the relevant information calculated, including error codes.

Author(s)

J.K. Lindsey

References

MacDonald, I.L. and Zucchini, W. (1997) Hidden Markov and other Models for Discrete-valued Time Series. Chapman & Hall.

For time-discretized Poisson processes, see

Davison, A.C. and Ramesh, N.I. (1996) Some models for discretized series of events. JASA 91: 601-609.

Examples

```
# model for one randomly-generated binary series
y <- c(rbinom(10,1,0.1), rbinom(10,1,0.9))
mu <- function(p) array(p, c(1,2))
print(z <- chidden(y, times=1:20, dist="Bernoulli",
pgamma=matrix(c(-0.1,0.2,0.1,-0.2),ncol=2),
cmu=mu, pcmu=c(-2,2)))
# or equivalently
print(z <- chidden(y, times=1:20, dist="Bernoulli",
pgamma=matrix(c(-0.1,0.2,0.1,-0.2),ncol=2),
cmu=~1, pcmu=c(-2,2)))
# or
print(z <- chidden(y, times=1:20, dist="Bernoulli",
pgamma=matrix(c(-0.1,0.2,0.1,-0.2),ncol=2),
mu=~rep(a,20), pmu=c(-2,2)))
mexp(z$gamma)
par(mfcol=c(2,2))
plot(z)
plot(iprofile(z), lty=2)
plot(mprofile(z), add=TRUE)
print(z <- chidden(y, times=(1:20)*2, dist="Bernoulli",
pgamma=matrix(c(-0.05,0.1,0.05,-0.1),ncol=2),
cmu=~1, pcmu=c(-2,2)))
mexp(z$gamma) %*% mexp(z$gamma)
plot(z)
plot(iprofile(z), lty=2)
plot(mprofile(z), add=TRUE)
```

`cphidden` *Changepoint Location using a Continuous-time Two-state Hidden Markov Chain*

Description

`cphidden` fits a two-state hidden Markov chain model with a variety of distributions in continuous time in order to locate a changepoint in the chosen distribution. All series on different individuals are assumed to start at the same time point.

Usage

```
cphidden(response = NULL, totals = NULL, times = NULL,
  distribution = "Bernoulli", mu = NULL, cmu = NULL, tvmu = NULL,
  pgamma, pmu = NULL, pcmu = NULL, ptvmu = NULL, pshape = NULL,
  pfamily = NULL, par = NULL, pintercept = NULL, delta = NULL,
  envir = parent.frame(), print.level = 0, ndigit = 10,
  gradtol = 1e-05, steptol = 1e-05, fscale = 1, iterlim = 100,
  typsize = abs(p), stepmax = 10 * sqrt(p %*% p))
```

Arguments

`response`	A list of two or three column matrices with counts or category indicators, times, and possibly totals (if the distribution is binomial), for each individual, one matrix or dataframe of counts, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here. If there is only one series, a vector of responses may be supplied instead. Multinomial and ordinal categories must be integers numbered from 0.
`totals`	If response is a matrix, a corresponding matrix of totals if the distribution is binomial. Ignored if response has class, `response` or `repeated`.
`times`	If `response` is a matrix, a vector of corresponding times, when they are the same for all individuals. Ignored if response has class, `response` or `repeated`.
`distribution`	Bernoulli, Poisson, multinomial, proportional odds, continuation ratio, binomial, exponential, beta binomial, negative binomial, normal, inverse Gauss, logistic, gamma, Weibull, Cauchy, Laplace, Levy, Pareto, gen(eralized) gamma, gen(eralized) logistic, Hjorth, Burr, gen(eralized) Weibull, gen(eralized) extreme value, gen(eralized) inverse Gauss, power exponential, skew Laplace, or Student t. (For definitions of distributions, see the corresponding [dpqr]distribution help.)
`mu`	A general location function with two possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per observation; or (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that pmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial).
`cmu`	A time-constant location function with three possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per individual; (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that pcmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial); or (3) a function returning an array with one row for each individual, one column for each state of the hidden Markov chain, and, if multinomial, one layer for

	each category but the last. If used, this function or formula should contain the intercept. Ignored if `mu` is supplied.
`tvmu`	A time-varying location function with three possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per time point; (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that ptvmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial); or (3) a function returning an array with one row for each time point, one column for each state of the hidden Markov chain, and, if multinomial, one layer for each category but the last. This function or formula is usually a function of time; it is the same for all individuals. It only contains the intercept if `cmu` does not. Ignored if `mu` is supplied.
`pgamma`	An initial estimate of the transition intensity between the two states in the continuous-time hidden Markov chain.
`pmu`	Initial estimates of the unknown parameters in `mu`.
`pcmu`	Initial estimates of the unknown parameters in `cmu`.
`ptvmu`	Initial estimates of the unknown parameters in `tvmu`.
`pshape`	Initial estimate(s) of the dispersion parameter, for those distributions having one. This can be one value or a vector with a different value for each state.
`pfamily`	Initial estimate of the family parameter, for those distributions having one.
`par`	Initial estimate of the autoregression parameter.
`pintercept`	For multinomial, proportional odds, and continuation ratio models, `p-2` initial estimates for intercept contrasts from the first intercept, where `p` is the number of categories.
`delta`	Scalar or vector giving the unit of measurement (always one for discrete data) for each response value, set to unity by default. For example, if a response is measured to two decimals, delta=0.01. If the response is transformed, this must be multiplied by the Jacobian. For example, with a log transformation, `delta=1/response`. Ignored if response has class, `response` or `repeated`.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`fscale`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`typsize`	Arguments for nlm.
`stepmax`	Arguments for nlm.

Details

For quantitative responses, specifying `par` allows an ‘observed’ autoregression to be fitted as well as the hidden Markov chain.

All functions and formulae for the location parameter are on the (generalized) logit scale for the Bernoulli, binomial, and multinomial distributions.

If `cmu` and `tvmu` are used, these two mean functions are additive so that interactions between time-constant and time-varying variables are not possible.

The algorithm will run more quickly if the most frequently occurring time step is scaled to be equal to unity.

The object returned can be plotted to give the probabilities of being in each hidden state at each time point. See `hidden` for details. For distributions other than the multinomial, proportional odds, and continuation ratio, the (recursive) predicted values can be plotted using `mprofile` and `iprofile`.

Value

A list of classes `hidden` and `recursive` (unless multinomial, proportional odds, or continuation ratio) is returned that contains all of the relevant information calculated, including error codes.

Author(s)

J.K. Lindsey

Examples

```
# model for one randomly-generated binary series
y <- c(rbinom(10,1,0.1), rbinom(10,1,0.9))
mu <- function(p) array(p, c(1,2))
print(z <- cphidden(y, times=1:20, dist="Bernoulli",
pgamma=0.1,cmu=mu, pcmu=c(-2,2)))
# or equivalently
print(z <- cphidden(y, times=1:20, dist="Bernoulli",
pgamma=0.2,cmu=~1, pcmu=c(-2,2)))
# or
print(z <- cphidden(y, times=1:20, dist="Bernoulli",
pgamma=0.2,mu=~rep(a,20), pmu=c(-2,2)))
mexp(z$gamma)
par(mfcol=c(2,2))
plot(z)
plot(iprofile(z), lty=2)
print(z <- cphidden(y, times=(1:20)*2, dist="Bernoulli",
pgamma=0.1,cmu=~1, pcmu=c(-2,2)))
mexp(z$gamma) %*% mexp(z$gamma)
plot(z)
plot(iprofile(z), lty=2)
```

`gar` *Generalized Autoregression Models*

Description

`gar` fits a first- or second-order generalized autoregression, possibly with Kalman update over time (first-order only).

Usage

```
gar(response = NULL, distribution = "normal", times = NULL,
  totals = NULL, censor = NULL, delta = NULL, mu = NULL,
  shape = NULL, depend = NULL, shfn = FALSE, common = FALSE,
  preg = NULL, pshape = NULL, pdepend = NULL, parch = NULL,
  arch = "square", transform = "identity", link = "identity",
  autocorr = "exponential", order = 1, envir = parent.frame(),
  print.level = 0, ndigit = 10, gradtol = 1e-05, steptol = 1e-05,
  fscale = 1, iterlim = 100, typsize = abs(p), stepmax = 10 *
  sqrt(p %*% p))
```

Arguments

`response`	A list of two or three column matrices with responses, corresponding times, and possibly a censor indicator, for each individual, one matrix or dataframe of responses, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.
`distribution`	The distribution to be fitted: binomial, Poisson, exponential, negative binomial, mult Poisson, double Poisson, Consul generalized Poisson, beta binomial, mult binomial, double binomial, normal, inverse Gauss, logistic, gamma, Weibull, Cauchy, Laplace, Levy, Pareto, beta, simplex, two-sided power, gen(eralized) gamma, gen(eralized) logistic, Hjorth, Burr, gen(eralized) Weibull, gen(eralized) extreme value, gen(eralized) inverse Gauss, power exponential, power variance function Poisson, skew Laplace, or Student t. (For definitions of distributions, see the corresponding [dpqr]distribution help.)
`times`	When response is a matrix, a vector of possibly unequally spaced times when they are the same for all individuals or a matrix of times. Not necessary if equally spaced. Ignored if response has class, `response` or `repeated`.
`totals`	An appropriate scalar, vector, or matrix of binomial totals (only applicable for binomial, beta binomial, mult binomial, double binomial). Ignored if response has class, `response` or `repeated`.
`censor`	If response is a matrix, a matrix of the same size containing the censor indicator: 1=uncensored, 0=right-censored, -1=left-censored. Ignored if response has class, `response` or `repeated`.

`delta`	Scalar or vector giving the unit of measurement for each response value, set to unity by default. For example, if a response is measured to two decimals, `delta=0.01`. If the response has been pretransformed, this must be multiplied by the Jacobian. This transformation cannot contain unknown parameters. For example, with a log transformation, `delta=1/y`. (The delta values for the censored response are ignored.) The jacobian is calculated automatically for the transform option. Ignored if response has class, `response` or `repeated`.
`mu`	A user-specified function of `pmu` giving the regression equation for the location. It may also be a formula beginning with ~, specifying either a linear regression function for the location parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. It must yield a value for each observation on each individual.
`shape`	An optional user-specified shape regression function; this may depend on the location (function) through its second argument, in which case, `shfn` must be set to TRUE. It may also be a formula beginning with ~, specifying either a linear regression function for the shape parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. If it contains unknown parameters, the keyword `mu` may be used to specify a function of the location parameter.
`depend`	An optional user-specified regression function for the log dependence parameter. It may also be a formula beginning with ~, specifying either a linear regression function for the dependence parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. If used, `order` must be one.
`shfn`	If TRUE, the supplied shape function depends on the location function. The name of this location function must be the last argument of the shape function.
`common`	If TRUE, `mu` and `shape` must both be either functions with, as argument, a vector of parameters having some or all elements in common between them so that indexing is in common between them or formulae with unknowns. All parameter estimates must be supplied in `preg`. If FALSE, parameters are distinct between the two functions and indexing starts at one in each function.
`preg`	The initial parameter estimates for the location regression function. If `mu` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`pshape`	Zero to two estimates for the shape parameters, depending on the distribution, if `shape` is not a function; otherwise, estimates for the parameters in this function, with one extra at the end for three-parameter distributions. If `shape` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list (only for two-parameter distributions).
`pdepend`	One or two estimates of the dependence parameters for the Kalman update. With one, it is Markovian and, with two, it is nonstationary. For the latter, the `order` must be one. If `depend` is a function or formula, the corresponding number of estimates must be supplied. Either pdepend or parch or both must be supplied.
`parch`	Estimate for an ARCH model where the shape parameter depends on the square of the previous residual. Either pdepend or parch or both must be supplied.

`arch`	If `square`, then `shape+parch^diff*residual^2`; if `absolute value`, then `shape+parch^diff*|residual|`; if `exponential`, then `shape*exp(parch*residual^2*diff)`, where `diff` is the length of time since the previous observation and `residual` is the previous residual or innovation.
`transform`	Transformation of the response variable: `identity`, `exp`, `square`, `sqrt`, or `log`.
`link`	Link function for the mean: `identity`, `exp`, `square`, `sqrt`, `log`, `logit`, `cloglog` or `loglog` (last three only for binary/binomial-type data).
`autocorr`	The form of the (second if two) dependence function: `exponential` is the usual $\rho^{\lvert t_i-t_j\rvert}$; `gaussian` is $\rho^{(t_i-t_j)^2}$; `cauchy` is $1/(1+\rho(t_i-t_j)^2)$; `spherical` is $((\lvert t_i-t_j\rvert\rho)^3-3\lvert t_i-t_j\rvert\rho+2)/2$ for $\lvert t_i-t_j\rvert \leq 1/\rho$ and zero otherwise; `IOU` is the integrated Ornstein-Uhlenbeck process, $(2\rho\min(t_i,t_j)+\exp(-\rho t_i)+\exp(-\rho t_j)-1-\exp(\rho\lvert ti-t_j\rvert))/2\rho^3$.
`order`	First- or second-order stationary autoregression.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`fscale`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`typsize`	Arguments for nlm.
`stepmax`	Arguments for nlm.

Details

Nonlinear regression models can be supplied as formulae where parameters are unknowns in which case factor variables cannot be used and parameters must be scalars. (See `finterp`.)

Marginal and individual profiles can be plotted using `mprofile` and `iprofile` and residuals with `plot.residuals`.

When the dispersion parameter is not constant over time, `volatility` extracts the square root of the dispersion parameter for a fitted model.

Value

A list of classes `gar` and `recursive` is returned that contains all of the relevant information calculated, including error codes.

The volatility vector for models with a shape regression function and ARCH models contains the square root of the dispersion parameter at each time point.

Author(s)

J.K. Lindsey

References

Lindsey, J.K. (1997) Applying Generalized Linear Models. Springer, pp.\ 93–101

Lambert, P. (1996) Statistics in Medicine 15, 1695-1708

Examples

```
# first-order one-compartment model
# data objects for formulae
dose <- c(2,5)
dd <- tcctomat(dose)
times <- matrix(rep(1:20,2), nrow=2, byrow=TRUE)
tt <- tvctomat(times)
# vector covariates for functions
dose <- c(rep(2,20),rep(5,20))
times <- rep(1:20,2)
# functions
mu <- function(p) exp(p[1]-p[3])*(dose/(exp(p[1])-exp(p[2]))*
(exp(-exp(p[2])*times)-exp(-exp(p[1])*times)))
shape <- function(p) exp(p[1]-p[2])*times*dose*exp(-exp(p[1])*times)
# response
conc <- matrix(rgamma(40,shape(log(c(0.1,0.4))),
scale=mu(log(c(1,0.3,0.2))))/shape(log(c(0.1,0.4))),ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
ncol=20,byrow=TRUE)[,1:19])
conc <- restovec(ifelse(conc>0,conc,0.01),name="conc")
reps <- rmna(conc, ccov=dd, tvcov=tt)
# constant shape parameter
gar(conc, dist="gamma", times=1:20, mu=mu,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=1)
# or
gar(conc, dist="gamma", times=1:20, mu=~exp(absorption-volume)*
dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times)),
preg=list(absorption=0,elimination=log(0.4),volume=log(0.1)),
pdepend=0.5, pshape=1, envir=reps)
# generalized gamma distribution
gar(conc, dist="gen gamma", times=1:20, mu=mu,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=c(1,1))
# (if the covariates contained NAs, reps would have to be used as
# response instead of conc)
#
# time dependent shape parameter
gar(conc, dist="gamma", times=1:20, mu=mu, shape=shape,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=log(c(1,0.2)))
# or
gar(conc, dist="gamma", times=1:20, mu=~exp(absorption-volume)*
dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times)),
shape=~exp(b1-b2)*times*dose*exp(-exp(b1)*times),
preg=list(absorption=0,elimination=log(0.4),volume=log(0.1)),
pdepend=0.5, pshape=list(b1=0,b2=log(0.2)), envir=reps)
```

```
# generalized gamma distribution
gar(conc, dist="gen gamma", times=1:20, mu=mu, shape=shape,
preg=log(c(1,0.4,0.1)), pdepend=0.5,
pshape=c(log(c(1,0.2)),2))
#
# shape function depends on location parameter
shape <- function(p, mu) p[1]+p[2]*mu
gar(conc, dist="gamma", times=1:20, mu=mu, shape=shape, shfn=TRUE,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=c(1,0))
# or
gar(conc, dist="gamma", times=1:20, mu=mu, shape=~a+d*mu, shfn=TRUE,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=c(1,0))
```

`gausscop` *Multivariate Gaussian Copula with Arbitrary Marginals*

Description

gausscop fits multivariate repeated measurements models based on the Gaussian copula with a choice of marginal distributions. Dependence among responses is provided by the correlation matrix containing random effects and/or autoregression.

Usage

```
gausscop(response = NULL, distribution = "gamma", mu = NULL,
  shape = NULL, autocorr = "exponential", pmu = NULL,
  pshape = NULL, par = NULL, pre = NULL, delta = NULL,
  shfn = FALSE, common = FALSE, envir = parent.frame(),
  print.level = 0, ndigit = 10, gradtol = 1e-05, steptol = 1e-05,
  iterlim = 100, fscale = 1, stepmax = 10 * sqrt(theta %*% theta),
  typsize = abs(c(theta)))
```

Arguments

`response`	A list of two or three column matrices with response values, times, and possibly nesting categories, for each individual, one matrix or dataframe of response values, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.
`distribution`	The marginal distribution: exponential, gamma, Weibull, Pareto, inverse Gauss, logistic, Cauchy, Laplace, or Levy.
`mu`	The linear or nonlinear regression model to be fitted for the location parameter. For marginal distributions requiring positive response values, a log link is used. This model can be a function of the parameters or a formula beginning with ~, specifying either a linear regression function for the location parameter in

	the Wilkinson and Rogers notation or a general function with named unknown parameters that describes the location, returning a vector the same length as the number of observations.
`shape`	The linear or nonlinear regression model to be fitted for the log shape parameter. This can be a function of the parameters or a formula beginning with ~, specifying either a linear regression function for the location parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters that describes the location. If it contains unknown parameters, the keyword `mu` may be used to specify a function of the location parameter.
`autocorr`	The form of the autocorrelation function: `exponential` is the usual $\rho^{\lvert t_i-t_j\rvert}$; `gaussian` is $\rho^{(t_i-t_j)^2}$; `cauchy` is $1/(1+\rho(t_i-t_j)^2)$; `spherical` is $((\lvert t_i-t_j\rvert\rho)^3-3\lvert t_i-t_j\rvert\rho+2)/2$ for $\lvert t_i-t_j\rvert\leq 1/\rho$ and zero otherwise.
`pmu`	Initial parameter estimates for the location regression model.
`pshape`	Initial parameter estimate for the shape regression model.
`par`	If supplied, an initial estimate for the autocorrelation parameter.
`pre`	Zero, one or two parameter estimates for the variance components, depending on the number of levels of nesting.
`delta`	Scalar or vector giving the unit of measurement for each response value, set to unity by default. For example, if a response is measured to two decimals, `delta=0.01`. Ignored if response has class, `response` or `repeated`.
`shfn`	If TRUE, the supplied shape function depends on the location function. The name of this location function must be the last argument of the shape function.
`common`	If TRUE, `mu` and `shape` must both be functions with, as argument, a vector of parameters having some or all elements in common between them so that indexing is in common between them; all parameter estimates must be supplied in `pmu`. If FALSE, parameters are distinct between the two functions and indexing starts at one in each function.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`fscale`	Arguments for nlm.
`stepmax`	Arguments for nlm.
`typsize`	Arguments for nlm.

Details

With two levels of nesting, the first is the individual and the second will consist of clusters within individuals.

For clustered (non-longitudinal) data, where only random effects will be fitted, `times` are not necessary.

This function is designed to fit linear and nonlinear models with time-varying covariates observed at arbitrary time points. A continuous-time AR(1) and zero, one, or two levels of nesting can be handled.

Nonlinear regression models can be supplied as formulae where parameters are unknowns in which case factor variables cannot be used and parameters must be scalars. (See `finterp`.)

Value

A list of class gausscop is returned that contains all of the relevant information calculated, including error codes.

Author(s)

J.K. Lindsey

References

Song, P.X.K. (2000) Multivariate dispersion models generated from Gaussian copula. Scandinavian Journal of Statistics 27, 305-320.

Examples

```
# linear models
y <- matrix(rgamma(40,1,1),ncol=5)+rep(rgamma(8,0.5,1),5)
x1 <- c(rep(0,4),rep(1,4))
reps <- rmna(restovec(y),ccov=tcctomat(x1))
# independence with default gamma marginals
# compare with gnlm::gnlr(y, pmu=1, psh=0, dist="gamma", env=reps)
gausscop(y, pmu=1, pshape=0, env=reps)
gausscop(y, mu=~x1, pmu=c(1,0), pshape=0, env=reps)
# AR(1)
gausscop(y, pmu=1, pshape=0, par=0.1, env=reps)
## Not run:
# random effect
gausscop(y, pmu=1, pshape=0, pre=0.1, env=reps)
# try other marginal distributions
gausscop(y, pmu=1, pshape=0, pre=0.1, env=reps, dist="Weibull")
gausscop(y, pmu=1, pshape=0, pre=0.1, env=reps, dist="inverse Gauss",
stepmax=1)
gausscop(y, pmu=1, pshape=0, pre=0.1, env=reps, dist="Cauchy")
#
# first-order one-compartment model
# create data objects for formulae
dose <- c(2,5)
```

```
dd <- tcctomat(dose)
times <- matrix(rep(1:20,2), nrow=2, byrow=TRUE)
tt <- tvctomat(times)
# vector covariates for functions
dose <- c(rep(2,20),rep(5,20))
times <- rep(1:20,2)
# functions
mu <- function(p) exp(p[1]-p[3])*(dose/(exp(p[1])-exp(p[2]))*
(exp(-exp(p[2])*times)-exp(-exp(p[1])*times)))
shape <- function(p) exp(p[1]-p[2])*times*dose*exp(-exp(p[1])*times)
lmu <- function(p) p[1]-p[3]+log(dose/(exp(p[1])-exp(p[2]))*
(exp(-exp(p[2])*times)-exp(-exp(p[1])*times)))
lshape <- function(p) p[1]-p[2]+log(times*dose)-exp(p[1])*times
# response
#conc <- matrix(rgamma(40,shape(log(c(0.1,0.4))),
# scale=mu(log(c(1,0.3,0.2))))/shape(log(c(0.1,0.4))),ncol=20,byrow=TRUE)
#conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
# ncol=20,byrow=TRUE)[,1:19])
#conc <- restovec(ifelse(conc>0,conc,0.01),name="conc")
conc <- matrix(c(3.65586845,0.01000000,0.01000000,0.01731192,1.68707608,
0.01000000,4.67338974,4.79679942,1.86429851,1.82886732,1.54708795,
0.57592054,0.08014232,0.09436425,0.26106139,0.11125534,0.22685364,
0.22896015,0.04886441,0.01000000,33.59011263,16.89115866,19.99638316,
16.94021361,9.95440037,7.10473948,2.97769676,1.53785279,2.13059515,
0.72562344,1.27832563,1.33917155,0.99811111,0.23437424,0.42751355,
0.65702300,0.41126684,0.15406463,0.03092312,0.14672610),
ncol=20,byrow=TRUE)
conc <- restovec(conc)
reps <- rmna(conc, ccov=dd, tvcov=tt)
# constant shape parameter
gausscop(conc, mu=lmu, pmu=log(c(1,0.4,0.1)), par=0.5, pshape=0, envir=reps)
# or
gausscop(conc, mu=~absorption-volume+
log(dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times))),
pmu=list(absorption=0,elimination=log(0.4),volume=log(0.1)),
par=0.5, pshape=0, envir=reps)
# compare to gar autoregression
gar(conc, dist="gamma", times=1:20, mu=mu,
preg=log(c(1,0.4,0.1)), pdepend=0.5, pshape=1)
#
# time dependent shape parameter
gausscop(conc, mu=lmu, shape=lshape,
pmu=log(c(1,0.4,0.1)), par=0.5, pshape=c(-0.1,-0.1))
# or
gausscop(conc, mu=~absorption-volume+
log(dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times))),
shape=~b1-b2+log(times*dose)-exp(b1)*times,
pmu=list(absorption=0,elimination=log(0.4),volume=log(0.1)),
par=0.5, pshape=list(b1=-0.1,b2=-0.1), envir=reps)
#
# shape depends on location
```

```
lshape <- function(p, mu) p[1]*log(abs(mu))
gausscop(conc, mu=lmu, shape=lshape, shfn=TRUE, pmu=log(c(1,0.4,0.1)),
par=0.5, pshape=1)
# or
gausscop(conc, mu=~absorption-volume+
log(dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times))),
shape=~d*log(abs(mu)), shfn=TRUE,
pmu=list(absorption=0,elimination=log(0.4),volume=log(0.1)),
par=0.5, pshape=list(d=1), envir=reps)

## End(Not run)
```

---

glmm *Generalized Linear Mixed Models*

Description

`glmm` fits a generalized linear mixed model with a random intercept using a normal mixing distribution computed by Gauss-Hermite integration. For the normal, gamma, and inverse Gaussian distributions, the deviances supplied are -2 log likelihood, not the usual `glm` deviance; the degrees of freedom take into account estimation of the dispersion parameter.

Usage

```
glmm(formula, family = gaussian, data = list(), weights = NULL,
  offset = NULL, nest, delta = 1, maxiter = 20, points = 10,
  print.level = 0, control = glm.control(epsilon = 1e-04, maxit = 10,
  trace = FALSE))
```

Arguments

`formula`	A symbolic description of the model to be fitted. If it contains transformations of the data, including cbind for binomial data, a dataframe must be supplied.
`family`	A description of the error distribution and link function to be used in the model; see `family` for details.
`data`	A dataframe containing the variables in the model, that is optional in simple cases, but required in certain situations as specified elsewhere in this help page.
`weights`	An optional weight vector. If this is used, data must be supplied in a data.frame.
`offset`	The known component in the linear predictor. If this is used, data must be supplied in a data.frame. An offset cannot be specified in the model formula.
`nest`	The variable classifying observations by the unit (cluster) upon which they were observed.
`delta`	If the response variable has been transformed, this is the Jacobian of that transformation, so that AICs are comparable.
`maxiter`	The maximum number of iterations of the outer loop for numerical integration.

`points`	The number of points for Gauss-Hermite integration of the random effect.
`print.level`	If set equal to 2, the log probabilities are printed out when the underflow error is given.
`control`	A list of parameters for controlling the fitting process.

Details

If weights and/or offset are to be used or the formula transforms some variables, all of the data must be supplied in a dataframe. Because the `glm` function is such a hack, if this is not done, weird error messages will result.

na.omit is not allowed.

Value

`glmm` returns a list of class `glmm`

Author(s)

J.K. Lindsey

Examples

```
# Poisson counts
nest <- gl(5,4)
y <- rpois(20,5+2*as.integer(nest))
# overdispersion model
glmm(y~1, family=poisson, nest=gl(20,1), points=3)
# clustered model
glmm(y~1, family=poisson, nest=nest, points=3)
#
# binomial data with model for overdispersion
df <- data.frame(r=rbinom(10,10,0.5), n=rep(10,10), x=c(rep(0,5),
rep(1,5)), nest=1:10)
glmm(cbind(r,n-r)~x, family=binomial, nest=nest, data=df)
```

`gnlmix` *Generalized Nonlinear Regression with a Random Parameter*

Description

`gnlmix` fits user-specified nonlinear regression equations to one or both parameters of the common one and two parameter distributions. One parameter of the location regression is random with some specified mixing distribution.

Usage

```
gnlmix(y = NULL, distribution = "normal", mixture = "normal",
  random = NULL, nest = NULL, mu = NULL, shape = NULL,
  linear = NULL, pmu = NULL, pshape = NULL, pmix = NULL,
  delta = 1, common = FALSE, envir = parent.frame(),
  print.level = 0, typsize = abs(p), ndigit = 10, gradtol = 1e-05,
  stepmax = 10 * sqrt(p %*% p), steptol = 1e-05, iterlim = 100,
  fscale = 1, eps = 1e-04, points = 5, steps = 10)
```

Arguments

`y`	A response vector of uncensored data, a two column matrix for binomial data, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.
`distribution`	The distribution for the response: binomial, beta binomial, double binomial, mult(iplicative) binomial, Poisson, negative binomial, double Poisson, mult(iplicative) Poisson, gamma count, Consul generalized Poisson, logarithmic series, geometric, normal, inverse Gauss, logistic, exponential, gamma, Weibull, extreme value, Cauchy, Pareto, Laplace, Levy, beta, simplex, or two-sided power. (For definitions of distributions, see the corresponding [dpqr]distribution help.)
`mixture`	The mixing distribution for the random parameter: normal, Cauchy, logistic, Laplace, inverse Gauss, gamma, inverse gamma, Weibull, beta, simplex, or two-sided power. The first four have zero location parameter, the next three have unit location parameter, and the last two have location parameter set to 0.5.
`random`	The name of the random parameter in the `mu` formula.
`nest`	The variable classifying observations by the unit upon which they were observed. Ignored if `y` or `envir` has class, `response` or `repeated`.
`mu`	A user-specified formula containing named unknown parameters, giving the regression equation for the location parameter. This may contain the keyword, `linear` referring to a linear part.
`shape`	A user-specified formula containing named unknown parameters, giving the regression equation for the shape parameter. This may contain the keyword, `linear` referring to a linear part. If nothing is supplied, this parameter is taken to be constant. This parameter is the logarithm of the usual one.
`linear`	A formula beginning with ~ in W&R notation, specifying the linear part of the regression function for the location parameter or list of two such expressions for the location and/or shape parameters.
`pmu`	Vector of initial estimates for the location parameters. These must be supplied either in their order of appearance in the formula or in a named list.
`pshape`	Vector of initial estimates for the shape parameters. These must be supplied either in their order of appearance in the expression or in a named list.
`pmix`	Initial estimate for the logarithm of the dispersion parameter of the mixing distribution.

`delta`	Scalar or vector giving the unit of measurement (always one for discrete data) for each response value, set to unity by default. For example, if a response is measured to two decimals, `delta=0.01`. If the response is transformed, this must be multiplied by the Jacobian. The transformation cannot contain unknown parameters. For example, with a log transformation, `delta=1/y`. (The delta values for the censored response are ignored.)
`common`	If TRUE, the formulae with unknowns for the location and shape have names in common. All parameter estimates must be supplied in `pmu`.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `y`. If `y` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`typsize`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`stepmax`	Arguments for nlm.
`steptol`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`fscale`	Arguments for nlm.
`eps`	Precision of the Romberg integration.
`points`	For the Romberg integration, the number of extrapolation points so that 2*points is the order of integration, by default set to 5; points=2 is Simpson's rule.
`steps`	For the Romberg integration, the maximum number of steps, by default set to 10.

Details

It is recommended that initial estimates for `pmu` and `pshape` be obtained from `gnlr`.

These nonlinear regression models must be supplied as formulae where parameters are unknowns. (See `finterp`.)

Value

A list of class `gnlm` is returned that contains all of the relevant information calculated, including error codes.

Author(s)

J.K. Lindsey

Examples

```
dose <- c(9,12,4,9,11,10,2,11,12,9,9,9,4,9,11,9,14,7,9,8)
#y <- rgamma(20,shape=2+0.3*dose,scale=2)+rep(rnorm(4,0,4),rep(5,4))
y <- c(8.674419, 11.506066, 11.386742, 27.414532, 12.135699,  4.359469,
       1.900681, 17.425948,  4.503345,  2.691792,  5.731100, 10.534971,
      11.220260,  6.968932,  4.094357, 16.393806, 14.656584,  8.786133,
      20.972267, 17.178012)
resp <- restovec(matrix(y, nrow=4, byrow=TRUE), name="y")
reps <- rmna(resp, tvcov=tvctomat(matrix(dose, nrow=4, byrow=TRUE), name="dose"))

# same linear normal model with random normal intercept fitted four ways
# compare with growth::elliptic(reps, model=~dose, preg=c(0,0.6), pre=4)
glmm(y~dose, nest=individuals, data=reps)
gnlmm(reps, mu=~dose, pmu=c(8.7,0.25), psh=3.5, psd=3)
gnlmix(reps, mu=~a+b*dose+rand, random="rand", pmu=c(8.7,0.25),
pshape=3.44, pmix=2.3)
## Not run:
# gamma model with log link and random normal intercept fitted three ways
glmm(y~dose, family=Gamma(link=log), nest=individuals, data=reps, points=8)
gnlmm(reps, distribution="gamma", mu=~exp(a+b*dose), pmu=c(2,0.03),
psh=1, psd=0.3)
gnlmix(reps, distribution="gamma", mu=~exp(a+b*dose+rand), random="rand",
pmu=c(2,0.04), pshape=1, pmix=-2)

# gamma model with log link and random gamma mixtures
gnlmix(reps, distribution="gamma", mixture="gamma",
mu=~exp(a*rand+b*dose), random="rand", pmu=c(2,0.04),
pshape=1.24, pmix=3.5)
gnlmix(reps, distribution="gamma", mixture="gamma",
mu=~exp(a+b*dose)*rand, random="rand", pmu=c(2,0.04),
pshape=1.24, pmix=2.5)

## End(Not run)
```

`gnlmm` *Generalized Nonlinear Mixed Models*

Description

`gnlmm` fits user-specified nonlinear regression equations to one or both parameters of the common one and two parameter distributions. The intercept of the location regression has a normally-distributed random effect. This normal mixing distribution is computed by Gauss-Hermite integration.

Usage

```
gnlmm(y = NULL, distribution = "normal", mu = NULL, shape = NULL,
  linear = NULL, nest = NULL, pmu = NULL, pshape = NULL,
```

```
psd = NULL, exact = FALSE, wt = 1, delta = 1, shfn = FALSE,
scale = NULL, points = 10, common = FALSE,
envir = parent.frame(), print.level = 0, typsize = abs(p),
ndigit = 10, gradtol = 1e-05, stepmax = sqrt(p %*% p)/10,
steptol = 1e-05, iterlim = 100, fscale = 1)
```

Arguments

`y`	A response vector for uncensored data, a two column matrix for binomial data or censored data, with the second column being the censoring indicator (1: uncensored, 0: right censored, -1: left censored), or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna`) or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here. The beta, simplex, and two-sided power distributions for proportions do not allow censoring.
`distribution`	Either a character string containing the name of the distribution or a function giving the -log likelihood and calling the location and shape functions. Distributions are binomial, beta binomial, double binomial, mult(iplicative) binomial, Poisson, negative binomial, double Poisson, mult(iplicative) Poisson, gamma count, Consul generalized Poisson, logarithmic series, geometric, normal, inverse Gauss, logistic, exponential, gamma, Weibull, extreme value, Cauchy, Pareto, Laplace, and Levy, beta, simplex, and two-sided power. All but the binomial-based distributions and the beta, simplex, and two-sided power may be right and/or left censored. (For definitions of distributions, see the corresponding [dpqr]distribution help.)
`mu`	A user-specified function of `pmu`, and possibly `linear`, giving the regression equation for the location. This may contain a linear part as the second argument to the function. It may also be a formula beginning with ~, specifying a either linear regression function for the location parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. If it contains unknown parameters, the keyword `linear` may be used to specify a linear part. If nothing is supplied, the location is taken to be constant unless the linear argument is given.
`shape`	A user-specified function of `pshape`, and possibly `linear` and/or `mu`, giving the regression equation for the dispersion or shape parameter. This may contain a linear part as the second argument to the function and the location function as last argument (in which case `shfn` must be set to TRUE). It may also be a formula beginning with ~, specifying either a linear regression function for the shape parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. If it contains unknown parameters, the keyword `linear` may be used to specify a linear part and the keyword `mu` to specify a function of the location parameter. If nothing is supplied, this parameter is taken to be constant unless the linear argument is given. This parameter is the logarithm of the usual one.
`linear`	A formula beginning with ~ in W&R notation, specifying the linear part of the regression function for the location parameter or list of two such expressions for the location and/or shape parameters.

`nest`	The variable classifying observations by the unit upon which they were observed. Ignored if `y` or `envir` has class, response.
`pmu`	Vector of initial estimates for the location parameters. If `mu` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`pshape`	Vector of initial estimates for the shape parameters. If `shape` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`psd`	Initial estimate of the standard deviation of the normal mixing distribution.
`exact`	If TRUE, fits the exact likelihood function for continuous data by integration over intervals of observation, i.e. interval censoring.
`wt`	Weight vector.
`delta`	Scalar or vector giving the unit of measurement (always one for discrete data) for each response value, set to unity by default. Ignored if `y` has class, response. For example, if a response is measured to two decimals, `delta=0.01`. If the response is transformed, this must be multiplied by the Jacobian. The transformation cannot contain unknown parameters. For example, with a log transformation, `delta=1/y`. (The delta values for the censored response are ignored.)
`shfn`	If true, the supplied shape function depends on the location (function). The name of this location function must be the last argument of the shape function.
`scale`	The scale on which the random effect is applied: `identity`, `log`, `logit`, `reciprocal`, or `exp`.
`points`	The number of points for Gauss-Hermite integration of the random effect.
`common`	If TRUE, `mu` and `shape` must both be either functions with, as argument, a vector of parameters having some or all elements in common between them so that indexing is in common between them or formulae with unknowns. All parameter estimates must be supplied in `pmu`. If FALSE, parameters are distinct between the two functions and indexing starts at one in each function.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `y`. If `y` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`typsize`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`stepmax`	Arguments for nlm.
`steptol`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`fscale`	Arguments for nlm.

Details

The `scale` of the random effect is the link function to be applied. For example, if it is `log`, the supplied mean function, `mu`, is transformed as exp(log(mu)+sd), where `sd` is the random effect parameter.

It is recommended that initial estimates for `pmu` and `pshape` be obtained from `gnlr`.

Nonlinear regression models can be supplied as formulae where parameters are unknowns in which case factor variables cannot be used and parameters must be scalars. (See `finterp`.)

The printed output includes the -log likelihood (not the deviance), the corresponding AIC, the maximum likelihood estimates, standard errors, and correlations.

Value

A list of class `gnlm` is returned that contains all of the relevant information calculated, including error codes.

Author(s)

J.K. Lindsey

Examples

```
# data objects
sex <- c(0,1,1)
sx <- tcctomat(sex)
dose <- matrix(rpois(30,10),nrow=3)
dd <- tvctomat(dose)
# vectors for functions
dose <- as.vector(t(dose))
sex <- c(rep(0,10),rep(1,20))
nest <- rbind(rep(1,10),rep(2,10),rep(3,10))
#y <- rgamma(30,2,scale=exp(0.2+0.1*dose+0.1*sex+rep(rnorm(3),rep(10,3)))/2)
y <- c(0.6490851,0.9313931,0.4765569,0.4188045,2.8339637,2.8158090,
2.6059975,2.9958184,2.7351583,3.2884980,1.1180961,0.9443986,1.7915571,
9.0013379,2.3969570,3.4227356,0.5045518,0.7452521,1.8712467,3.6814198,
0.1489849,1.0327552,0.6102406,1.1536620,2.9145237,9.2847798,5.6454605,
1.9759672,1.5798008,5.1024496)
y <- restovec(matrix(y, nrow=3), nest=nest, name="y")
reps <- rmna(y, ccov=sx, tvcov=dd)
#
# log linear regression with gamma distribution
mu <- function(p) exp(p[1]+p[2]*sex+p[3]*dose)
## print(z <- gnlm::gnlr(y, dist="gamma", mu=mu, pmu=c(1,0,0), pshape=1))
## starting values for pmu and pshape from z$coef[1:3] and z$coef[4] respectively
gnlmm(y, dist="gamma", mu=mu, nest=nest, pmu=c(1.101,0.326,-0.045),
     pshape=0.391, psd=0.1, points=3)
# or equivalently
gnlmm(y, dist="gamma", mu=~exp(b0+b1*sex+b2*dose), nest=nest,
     pmu=c(1.101,0.326,-0.045), pshape=0.391, psd=0.1, points=3, envir=reps)
## Not run:
```

```
# or with identity link
print(z <- gnlm::gnlr(y, dist="gamma", mu=~sex+dose, pmu=c(0.1,0,0), pshape=1))
gnlmm(y, dist="gamma", mu=~sex+dose, nest=nest, pmu=z$coef[1:3],
pshape=z$coef[4], psd=0.1, points=3)
# or
gnlmm(y, dist="gamma", mu=~b0+b1*sex+b2*dose, nest=nest, pmu=z$coef[1:3],
pshape=z$coef[4], psd=0.1, points=3, envir=reps)
#
# nonlinear regression with gamma distribution
mu <- function(p) p[1]+exp(p[2]+p[3]*sex+p[4]*dose)
print(z <- gnlm::gnlr(y, dist="gamma", mu=mu, pmu=c(1,1,0,0), pshape=1))
gnlmm(y, dist="gamma", mu=mu, nest=nest, pmu=z$coef[1:4],
pshape=z$coef[5], psd=0.1, points=3)
# or
mu2 <- function(p, linear) p[1]+exp(linear)
gnlmm(y, dist="gamma", mu=mu2, linear=~sex+dose, nest=nest,
pmu=z$coef[1:4], pshape=1, psd=0.1, points=3)
# or
gnlmm(y, dist="gamma", mu=~a+exp(linear), linear=~sex+dose, nest=nest,
pmu=z$coef[1:4], pshape=1, psd=0.1, points=3)
# or
gnlmm(y, dist="gamma", mu=~b4+exp(b0+b1*sex+b2*dose), nest=nest,
pmu=z$coef[1:4], pshape=z$coef[5], psd=0.1,
points=3, envir=reps)
#
# include regression for the shape parameter with same mu function
shape <- function(p) p[1]+p[2]*sex
print(z <- gnlm::gnlr(y, dist="gamma", mu=mu, shape=shape, pmu=z$coef[1:4],
pshape=rep(1,2)))
gnlmm(y, dist="gamma", mu=mu, shape=shape, nest=nest,
pmu=z$coef[1:4], pshape=z$coef[5:6], psd=0.1, points=3)
# or
gnlmm(y, dist="gamma", mu=mu, shape=shape, nest=nest, pmu=z$coef[1:4],
pshape=z$coef[5:6], psd=0.1, points=3, envir=reps)
# or
gnlmm(y, dist="gamma", mu=~b4+exp(b0+b1*sex+b2*dose), shape=~a1+a2*sex,
nest=nest, pmu=z$coef[1:4], pshape=z$coef[5:6], psd=0.1,
points=3, envir=reps)

## End(Not run)
```

`gnlmm3` *Generalized Nonlinear Mixed Models for Three-parameter Distributions*

Description

`gnlmm3` fits user-specified nonlinear regression equations to one or more parameters of the common three parameter distributions. The intercept of the location regression has a normally-distributed random effect. This normal mixing distribution is computed by Gauss-Hermite integration.

Usage

```
gnlmm3(y = NULL, distribution = "normal", mu = NULL, shape = NULL,
  nest = NULL, family = NULL, linear = NULL, pmu = NULL,
  pshape = NULL, pfamily = NULL, psd = NULL, exact = FALSE,
  wt = 1, scale = NULL, points = 10, common = FALSE, delta = 1,
  envir = parent.frame(), print.level = 0, typsize = abs(p),
  ndigit = 10, gradtol = 1e-05, stepmax = 10 * sqrt(p %*% p),
  steptol = 1e-05, iterlim = 100, fscale = 1)
```

Arguments

`y`	A response vector for uncensored data, a two column matrix for binomial data or censored data, with the second column being the censoring indicator (1: uncensored, 0: right censored, -1: left censored), or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna`) or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.
`distribution`	Either a character string containing the name of the distribution or a function giving the -log likelihood and calling the location, shape, and family functions. Distributions are Box-Cox transformed normal, generalized inverse Gauss, generalized logistic, Hjorth, generalized gamma, Burr, generalized Weibull, power exponential, Student t, generalized extreme value, power variance function Poisson, and skew Laplace. (For definitions of distributions, see the corresponding [dpqr]distribution help.)
`mu`	A user-specified function of `pmu`, and possibly `linear`, giving the regression equation for the location. This may contain a linear part as the second argument to the function. It may also be a formula beginning with ~, specifying a either linear regression function for the location parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. If it contains unknown parameters, the keyword `linear` may be used to specify a linear part. If nothing is supplied, the location is taken to be constant unless the linear argument is given.
`shape`	A user-specified function of `pshape`, and possibly `linear` and/or `mu`, giving the regression equation for the dispersion or shape parameter. This may contain a linear part as the second argument to the function and the location function as last argument (in which case `shfn` must be set to TRUE). It may also be a formula beginning with ~, specifying either a linear regression function for the shape parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. If it contains unknown parameters, the keyword `linear` may be used to specify a linear part and the keyword `mu` to specify a function of the location parameter. If nothing is supplied, this parameter is taken to be constant unless the linear argument is given. This parameter is the logarithm of the usual one.
`nest`	The variable classifying observations by the unit upon which they were observed. Ignored if `y` or `envir` has class, response.
`family`	A user-specified function of `pfamily`, and possibly `linear`, for the regression equation of the third (family) parameter of the distribution. This may contain a

	linear part that is the second argument to the function. It may also be a formula beginning with ~, specifying either a linear regression function for the family parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters. If neither is supplied, this parameter is taken to be constant unless the linear argument is given. In most cases, this parameter is the logarithm of the usual one.
`linear`	A formula beginning with ~ in W&R notation, specifying the linear part of the regression function for the location parameter or list of two such expressions for the location and/or shape parameters.
`pmu`	Vector of initial estimates for the location parameters. If `mu` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`pshape`	Vector of initial estimates for the shape parameters. If `shape` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`pfamily`	Vector of initial estimates for the family parameters. If `family` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`psd`	Initial estimate of the standard deviation of the normal mixing distribution.
`exact`	If TRUE, fits the exact likelihood function for continuous data by integration over intervals of observation, i.e. interval censoring.
`wt`	Weight vector.
`scale`	The scale on which the random effect is applied: `identity`, `log`, `logit`, `reciprocal`, or `exp`.
`points`	The number of points for Gauss-Hermite integration of the random effect.
`common`	If TRUE, at least two of `mu`, `shape`, and `family` must both be either functions with, as argument, a vector of parameters having some or all elements in common between them so that indexing is in common between them or formulae with unknowns. All parameter estimates must be supplied in `pmu`. If FALSE, parameters are distinct between the two functions and indexing starts at one in each function.
`delta`	Scalar or vector giving the unit of measurement (always one for discrete data) for each response value, set to unity by default. Ignored if y has class, response. For example, if a response is measured to two decimals, `delta=0.01`. If the response is transformed, this must be multiplied by the Jacobian. The transformation cannot contain unknown parameters. For example, with a log transformation, `delta=1/y`. (The delta values for the censored response are ignored.)
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `y`. If `y` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`typsize`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.

`stepmax`	Arguments for nlm.
`steptol`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`fscale`	Arguments for nlm.

Details

The `scale` of the random effect is the link function to be applied. For example, if it is `log`, the supplied mean function, `mu`, is transformed as exp(log(mu)+sd), where sd is the random effect parameter.

It is recommended that initial estimates for `pmu`, `pshape`, and `pfamily` be obtained from `gnlr3`.

Nonlinear regression models can be supplied as formulae where parameters are unknowns in which case factor variables cannot be used and parameters must be scalars. (See `finterp`.)

The printed output includes the -log likelihood (not the deviance), the corresponding AIC, the maximum likelihood estimates, standard errors, and correlations.

Value

A list of class `gnlm` is returned that contains all of the relevant information calculated, including error codes.

Author(s)

J.K. Lindsey

Examples

```
# data objects
sex <- c(0,1,1)
sx <- tcctomat(sex)
#dose <- matrix(rpois(30,10),nrow=3)
dose <- matrix(c(8,9,11,9,11,11,7,8,7,12,8,8,9,10,15,10,9,9,20,14,4,7,
4,13,10,13,6,13,11,17),nrow=3)
dd <- tvctomat(dose)
# vectors for functions
dose <- as.vector(t(dose))
sex <- c(rep(0,10),rep(1,20))
nest <- rbind(rep(1,10),rep(2,10),rep(3,10))
#y <- (rt(30,5)+exp(0.2+0.3*dose+0.5*sex+rep(rnorm(3),rep(10,3))))*3
y <- c(62.39712552,196.94419614,2224.74940087,269.56691601,12.86079662,
14.96743546, 47.45765042,156.51381687,508.68804438,281.11065302,
92.32443655, 81.88000484, 40.26357733, 13.04433670, 15.58490237,
63.62154867, 23.69677549, 53.52885894, 88.02507682, 34.04302506,
44.28232323,116.80732423,106.72564484, 25.09749055, 12.61839145,
-0.04060996,153.32670123, 63.25866087, 17.79852591,930.52558064)
y <- restovec(matrix(y, nrow=3), nest=nest, name=”y”)
reps <- rmna(y, ccov=sx, tvcov=dd)
#
```

```
# log linear regression with Student t distribution
mu <- function(p) exp(p[1]+p[2]*sex+p[3]*dose)
## print(z <- gnlm::gnlr3(y, dist="Student", mu=mu, pmu=c(0,0,0), pshape=1, pfamily=1))
## starting values for pmu and pshape from z$coef[1:3] and z$coef[4] respectively
## starting value for pfamily in z$coef[5]
gnlmm3(y, dist="Student", mu=mu, nest=nest, pmu=c(3.69,-1.19, 0.039),
pshape=4.94, pfamily=-0.6887259, psd=50, points=3)
# or equivalently
gnlmm3(y, dist="Student", mu=~exp(b0+b1*sex+b2*dose), nest=nest,
pmu=c(3.69,-1.19, 0.039), pshape=4.94, pfamily=-0.6887259, psd=50,
points=3, envir=reps)
## Not run:
# or with identity link
print(z <- gnlm::gnlr3(y, dist="Student", mu=~sex+dose, pmu=c(0.1,0,0), pshape=1,
pfamily=1))
gnlmm3(y, dist="Student", mu=~sex+dose, nest=nest, pmu=z$coef[1:3],
pshape=z$coef[4], pfamily=z$coef[5], psd=50, points=3)
# or
gnlmm3(y, dist="Student", mu=~b0+b1*sex+b2*dose, nest=nest, pmu=z$coef[1:3],
pshape=z$coef[4], pfamily=z$coef[5], psd=50, points=3, envir=reps)
#
# nonlinear regression with Student t distribution
mu <- function(p) p[1]+exp(p[2]+p[3]*sex+p[4]*dose)
print(z <- gnlm::gnlr3(y, dist="Student", mu=mu, pmu=c(1,1,0,0), pshape=1,
pfamily=1))
gnlmm3(y, dist="Student", mu=mu, nest=nest, pmu=z$coef[1:4],
pshape=z$coef[5], pfamily=z$coef[6], psd=50, points=3)
# or
mu2 <- function(p, linear) p[1]+exp(linear)
gnlmm3(y, dist="Student", mu=mu2, linear=~sex+dose, nest=nest,
pmu=z$coef[1:4], pshape=z$coef[5], pfamily=z$coef[6], psd=50,
points=3)
# or
gnlmm3(y, dist="Student", mu=~a+exp(linear), linear=~sex+dose, nest=nest,
pmu=z$coef[1:4], pshape=z$coef[5], pfamily=z$coef[6], psd=50,
points=3)
# or
gnlmm3(y, dist="Student", mu=~b4+exp(b0+b1*sex+b2*dose), nest=nest,
pmu=z$coef[1:4], pshape=z$coef[5], pfamily=z$coef[6], psd=50,
points=3, envir=reps)
#
# include regression for the shape parameter with same mu function
shape <- function(p) p[1]+p[2]*sex
print(z <- gnlm::gnlr3(y, dist="Student", mu=mu, shape=shape, pmu=z$coef[1:4],
pshape=c(z$coef[5],0), pfamily=z$coef[6]))
gnlmm3(y, dist="Student", mu=mu, shape=shape, nest=nest,
pmu=z$coef[1:4], pshape=z$coef[5:6], pfamily=z$coef[7],
psd=5, points=3)
# or
gnlmm3(y, dist="Student", mu=mu, shape=shape, nest=nest, pmu=z$coef[1:4],
pshape=z$coef[5:6], pfamily=z$coef[7], psd=5, points=3,
envir=reps)
# or
```

```
gnlmm3(y, dist="Student", mu=~b4+exp(b0+b1*sex+b2*dose), shape=~a1+a2*sex,
nest=nest, pmu=z$coef[1:4], pshape=z$coef[5:6],
pfamily=z$coef[7], psd=5, points=3, envir=reps)

## End(Not run)
```

`hidden` *Discrete-time Hidden Markov Chain Models*

Description

`hidden` fits a two or more state hidden Markov chain model with a variety of distributions. All series on different individuals are assumed to start at the same time point. Time points are equal, discrete steps.

Usage

```
hidden(response = NULL, totals = NULL, distribution = "Bernoulli",
  mu = NULL, cmu = NULL, tvmu = NULL, pgamma, pmu = NULL,
  pcmu = NULL, ptvmu = NULL, pshape = NULL, pfamily = NULL,
  par = NULL, pintercept = NULL, delta = NULL,
  envir = parent.frame(), print.level = 0, ndigit = 10,
  gradtol = 1e-05, steptol = 1e-05, fscale = 1, iterlim = 100,
  typsize = abs(p), stepmax = 10 * sqrt(p %*% p))
```

Arguments

`response`	A list of two or three column matrices with counts or category indicators, times, and possibly totals (if the distribution is binomial), for each individual, one matrix or dataframe of counts, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here. If there is only one series, a vector of responses may be supplied instead. Multinomial and ordinal categories must be integers numbered from 0.
`totals`	If response is a matrix, a corresponding matrix of totals if the distribution is binomial. Ignored if response has class, `response` or `repeated`.
`distribution`	Bernoulli, Poisson, multinomial, proportional odds, continuation ratio, binomial, exponential, beta binomial, negative binomial, normal, inverse Gauss, logistic, gamma, Weibull, Cauchy, Laplace, Levy, Pareto, gen(eralized) gamma, gen(eralized) logistic, Hjorth, Burr, gen(eralized) Weibull, gen(eralized) extreme value, gen(eralized) inverse Gauss, power exponential, skew Laplace, or Student t. (For definitions of distributions, see the corresponding [dpqr]distribution help.)

`mu`	A general location function with two possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per observation; or (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that pmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial).
`cmu`	A time-constant location function with three possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per individual; (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that pcmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial); or (3) a function returning an array with one row for each individual, one column for each state of the hidden Markov chain, and, if multinomial, one layer for each category but the last. If used, this function or formula should contain the intercept. Ignored if `mu` is supplied.
`tvmu`	A time-varying location function with three possibilities: (1) a list of formulae (with parameters having different names) or functions (with one parameter vector numbering for all of them) each returning one value per time point; (2) a single formula or function which will be used for all states (and all categories if multinomial) but with different parameter values in each so that ptvmu must be a vector of length the number of unknowns in the function or formula times the number of states (times the number of categories minus one if multinomial); or (3) a function returning an array with one row for each time point, one column for each state of the hidden Markov chain, and, if multinomial, one layer for each category but the last. This function or formula is usually a function of time; it is the same for all individuals. It only contains the intercept if `cmu` does not. Ignored if `mu` is supplied.
`pgamma`	A square `mxm` matrix of initial estimates of the hidden Markov transition matrix, where `m` is the number of hidden states. Rows must sum to one. If the matrix contains zeroes or ones, these are fixed and not estimated. (Ones cannot appear on the diagonal.) If a `1x1` matrix or a scalar value of 1 is given, the independence model is fitted.
`pmu`	Initial estimates of the unknown parameters in `mu`.
`pcmu`	Initial estimates of the unknown parameters in `cmu`.
`ptvmu`	Initial estimates of the unknown parameters in `tvmu`.
`pshape`	Initial estimate(s) of the dispersion parameter, for those distributions having one. This can be one value or a vector with a different value for each state.
`pfamily`	Initial estimate of the family parameter, for those distributions having one.
`par`	Initial estimate of the autoregression parameter.
`pintercept`	For multinomial, proportional odds, and continuation ratio models, `p-2` initial estimates for intercept contrasts from the first intercept, where `p` is the number of categories.

`delta`	Scalar or vector giving the unit of measurement (always one for discrete data) for each response value, set to unity by default. For example, if a response is measured to two decimals, delta=0.01. If the response is transformed, this must be multiplied by the Jacobian. For example, with a log transformation, `delta=1/response`. Ignored if response has class, `response` or `repeated`.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`fscale`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`typsize`	Arguments for nlm.
`stepmax`	Arguments for nlm.

Details

To fit an ‘observed’ Markov chain, as well, with Bernoulli or multinomial responses, use the lagged response as a time-varying covariate. For quantitative responses, specifying `par` allows an ‘observed" autoregression to be fitted as well as the hidden Markov chain.

All functions and formulae for the location parameter are on the (generalized) logit scale for the Bernoulli, binomial, and multinomial distributions.

If `cmu` and `tvmu` are used, these two mean functions are additive so that interactions between time-constant and time-varying variables are not possible.

The object returned can be plotted to give the probabilities of being in each hidden state at each time point. For distributions other than the multinomial, proportional odds, and continuation ratio, the (recursive) predicted values can be plotted using `mprofile` and `iprofile`.

See MacDonald, I.L. and Zucchini, W. (1997) Hidden Markov and Other Models for Discrete-valued Time Series. Chapman and Hall.

Value

A list of classes `hidden` and `recursive` (unless multinomial, proportional odds, or continuation ratio) is returned that contains all of the relevant information calculated, including error codes.

Author(s)

J.K. Lindsey and P.J. Lindsey

References

MacDonald, I.L. and Zucchini, W. (1997) Hidden Markov and other Models for Discrete-valued Time Series. Chapman & Hall.

Examples

```
# generate two random Poisson sequences with change-points
y <- rbind(c(rpois(5,1), rpois(15,5)), c(rpois(15,1), rpois(5,5)))
print(z <- hidden(y,dist="Poisson", cmu=~1, pcmu=c(1,5),
pgamma=matrix(c(0.9,0.2,0.1,0.8),ncol=2)))
# or equivalently
mu <- function(p) array(rep(p[1:2],rep(2,2)), c(2,2))
print(z <- hidden(y,dist="Poisson", cmu=mu, pcmu=c(1,5),
pgamma=matrix(c(0.9,0.2,0.1,0.8),ncol=2)))
# or
# param nind For plotting: numbers of individuals to plot.
# param state For plotting: states to plot.
print(z <- hidden(y,dist="Poisson", mu=~rep(a,40), pmu=c(1,5),
pgamma=matrix(c(0.9,0.2,0.1,0.8),ncol=2)))
par(mfrow=c(3,2))
plot(z, nind=1:2)
plot(z, nind=1:2, smooth=TRUE)
plot(iprofile(z), lty=2)
plot(mprofile(z), add=TRUE)
plot(iprofile(z), nind=2, lty=2)
plot(mprofile(z), nind=2, add=TRUE)
```

`hnlmix` *Generalized Nonlinear Regression using h-likelihood for a Random Parameter*

Description

`hnlmix` fits user-specified nonlinear regression equations to one or both parameters of the common one and two parameter distributions. One parameter of the location regression is random with some specified mixing distribution.

Usage

```
hnlmix(y = NULL, distribution = "normal", mixture = "normal",
  random = NULL, nest = NULL, mu = NULL, shape = NULL,
  linear = NULL, pmu = NULL, pshape = NULL, pmix = NULL,
  prandom = NULL, delta = 1, common = FALSE,
  envir = parent.frame(), print.level = 0, typsize = abs(p),
  ndigit = 10, gradtol = 1e-05, stepmax = 10 * sqrt(p %*% p),
  steptol = 1e-05, iterlim = 100, fscale = 1, eps = 1e-04,
  points = 5)
```

Arguments

`y`	A response vector of uncensored data, a two column matrix for binomial data or censored data, with the second column being the censoring indicator (1: uncensored, 0: right censored, -1: left censored), or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.
`distribution`	The distribution for the response: binomial, beta binomial, double binomial, mult(iplicative) binomial, Poisson, negative binomial, double Poisson, mult(iplicative) Poisson, gamma count, Consul generalized Poisson, logarithmic series, geometric, normal, inverse Gauss, logistic, exponential, gamma, Weibull, extreme value, Cauchy, Pareto, Laplace, Levy, beta, simplex, or two-sided power. (For definitions of distributions, see the corresponding [dpqr]distribution help.)
`mixture`	The mixing distribution for the random parameter (whose initial values are supplied in `prandom`): normal, logistic, inverse Gauss, gamma, inverse gamma, Weibull, or beta. The first two have zero location parameter, the next three have unit location parameter, and the last one has location parameter set to 0.5.
`random`	The name of the random parameter in the `mu` formula.
`nest`	The cluster variable classifying observations by the unit upon which they were observed. Ignored if `y` or `envir` has class, `response` or `repeated`.
`mu`	A user-specified formula containing named unknown parameters, giving the regression equation for the location parameter. This may contain the keyword, `linear` referring to a linear part.
`shape`	A user-specified formula containing named unknown parameters, giving the regression equation for the shape parameter. This may contain the keyword, `linear` referring to a linear part. If nothing is supplied, this parameter is taken to be constant. This parameter is the logarithm of the usual one.
`linear`	A formula beginning with ~ in W&R notation, specifying the linear part of the regression function for the location parameter or list of two such expressions for the location and/or shape parameters.
`pmu`	Vector of initial estimates for the location parameters. These must be supplied either in their order of appearance in the formula or in a named list.
`pshape`	Vector of initial estimates for the shape parameters. These must be supplied either in their order of appearance in the expression or in a named list.
`pmix`	If NULL, this parameter is estimated from the variances. If a value is given, it is taken as fixed.
`prandom`	Either one estimate of the random effects or one for each cluster (see `nest`), in which case the last value is not used. If the location parameter of the mixing distribution is zero, the last value is recalculated so that their sum is zero; if it is unity, they must all be positive and the last value is recalculated so that the sum of their logarithms is zero; if it is 0.5, they must all lie in (0,1) and the last value is recalculated so that the sum of their logits is zero.
`delta`	Scalar or vector giving the unit of measurement (always one for discrete data) for each response value, set to unity by default. For example, if a response

	is measured to two decimals, `delta=0.01`. If the response is transformed, this must be multiplied by the Jacobian. The transformation cannot contain unknown parameters. For example, with a log transformation, `delta=1/y`. (The delta values for the censored response are ignored.)
`common`	If TRUE, the formulae with unknowns for the location and shape have names in common. All parameter estimates must be supplied in `pmu`.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `y`. If `y` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`typsize`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`stepmax`	Arguments for nlm.
`steptol`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`fscale`	Arguments for nlm.
`eps`	Arguments for nlm.
`points`	Arguments for nlm.

Details

It is recommended that initial estimates for `pmu` and `pshape` be obtained from `gnlr`.

These nonlinear regression models must be supplied as formulae where parameters are unknowns. (See `finterp`.)

Value

A list of class `hnlmix` is returned that contains all of the relevant information calculated, including error codes.

The two variances and shrinkage estimates of the random effects are provided.

Author(s)

J.K. Lindsey

Examples

```
dose <- c(9,12,4,9,11,10,2,11,12,9,9,9,4,9,11,9,14,7,9,8)
#y <- rgamma(20,2+0.3*dose,scale=2)+rep(rnorm(4,0,4),rep(5,4))
y <- c(8.674419, 11.506066, 11.386742, 27.414532, 12.135699,  4.359469,
        1.900681, 17.425948,  4.503345,  2.691792,  5.731100, 10.534971,
       11.220260,  6.968932,  4.094357, 16.393806, 14.656584,  8.786133,
       20.972267, 17.178012)
resp <- restovec(matrix(y, nrow=4, byrow=TRUE), name="y")
```

```
reps <- rmna(resp, tvcov=tvctomat(matrix(dose, nrow=4, byrow=TRUE), name="dose"))

# same linear normal model with random normal intercept fitted four ways
# compare with growth::elliptic(reps, model=~dose, preg=c(0,0.6), pre=4)
glmm(y~dose, nest=individuals, data=reps)
gnlmm(reps, mu=~dose, pmu=c(8.7,0.25), psh=3.5, psd=3)
hnlmix(reps, mu=~a+b*dose+rand, random="rand", pmu=c(8.7,0.25),
pshape=3.44, prandom=0)

# gamma model with log link and random normal intercept fitted three ways
glmm(y~dose, family=Gamma(link=log), nest=individuals, data=reps, points=8)
gnlmm(reps, distribution="gamma", mu=~exp(a+b*dose), pmu=c(2,0.03),
psh=1, psd=0.3)
hnlmix(reps, distribution="gamma", mu=~exp(a+b*dose+rand), random="rand",
pmu=c(2,0.04), pshape=1, prandom=0)

# gamma model with log link and random gamma mixtures
hnlmix(reps, distribution="gamma", mixture="gamma",
mu=~exp(a*rand+b*dose), random="rand", pmu=c(2,0.04),
pshape=1.24, prandom=1)
hnlmix(reps, distribution="gamma", mixture="gamma",
mu=~exp(a+b*dose)*rand, random="rand", pmu=c(2,0.04),
pshape=1.24, prandom=1)
```

`kalcount` *Repeated Measurements Models for Counts with Frailty or Serial Dependence*

Description

`kalcount` is designed to handle repeated measurements models with time-varying covariates. The distributions have two extra parameters as compared to the functions specified by `intensity` and are generally longer tailed than those distributions. Dependence among observations on a unit can be through gamma or power variance family frailties (a type of random effect), with or without autoregression, or serial dependence over time.

Usage

```
kalcount(response = NULL, times = NULL, origin = 0,
  intensity = "exponential", depend = "independence",
  update = "Markov", mu = NULL, shape = NULL, density = FALSE,
  ccov = NULL, tvcov = NULL, preg = NULL, ptvc = NULL,
  pbirth = NULL, pintercept = NULL, pshape = NULL, pinitial = 1,
  pdepend = NULL, pfamily = NULL, envir = parent.frame(),
  print.level = 0, ndigit = 10, gradtol = 1e-05, steptol = 1e-05,
  fscale = 1, iterlim = 100, typsize = abs(p), stepmax = 10 *
  sqrt(p %*% p))
```

Arguments

`response`	A list of two column matrices with counts and corresponding times for each individual, one matrix or dataframe of counts, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.
`times`	When response is a matrix, a vector of possibly unequally spaced times when they are the same for all individuals or a matrix of times. Not necessary if equally spaced. Ignored if response has class, `response` or `repeated`.
`origin`	If the time origin is to be before the start of observations, a positive constant to be added to all times.
`intensity`	The form of function to be put in the Pareto distribution. Choices are exponential, Weibull, gamma, log normal, log logistic, log Cauchy, log Student, and gen(eralized) logistic.
`depend`	Type of dependence. Choices are `independence`, `frailty`, and `serial`.
`update`	Type of for serial dependence. Choices are `Markov`, `serial`, `event`, `cumulated`, `count`, and `kalman`. With `frailty` dependence, weighting by length of observation time may be specified by setting update to `time`.
`mu`	A regression function for the location parameter or a formula beginning with ~, specifying either a linear regression function in the Wilkinson and Rogers notation (a log link is assumed) or a general function with named unknown parameters. Give the initial estimates in `preg` if there are no time-varying covariates and in `ptvc` if there are.
`shape`	A regression function for the shape parameter or a formula beginning with ~, specifying either a linear regression function in the Wilkinson and Rogers notation or a general function with named unknown parameters. It must yield one value per observation.
`density`	If TRUE, the density of the function specified in `intensity` is used instead of the intensity.
`ccov`	A vector or matrix containing time-constant baseline covariates with one row per individual, a model formula using vectors of the same size, or an object of class, `tccov` (created by `tcctomat`). If response has class, `repeated`, the covariates must be supplied as a Wilkinson and Rogers formula unless none are to be used or `mu` is given.
`tvcov`	A list of matrices with time-varying covariate values, observed in the time periods in `response`, for each individual (one column per variable), one matrix or dataframe of such covariate values, or an object of class, `tvcov` (created by `tvctomat`). If response has class, `repeated`, the covariates must be supplied as a Wilkinson and Rogers formula unless none are to be used or `mu` is given.
`preg`	Initial parameter estimates for the regression model: intercept plus one for each covariate in `ccov`. If `mu` is a formula or function, the parameter estimates must be given here only if there are no time-varying covariates. If `mu` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.

`ptvc`	Initial parameter estimates for the coefficients of the time-varying covariates, as many as in `tvcov`. If `mu` is a formula or function, the parameter estimates must be given here if there are time-varying covariates present.
`pbirth`	If supplied, this is the initial estimate for the coefficient of the birth model.
`pintercept`	The initial estimate of the intercept for the generalized logistic intensity.
`pshape`	An initial estimate for the shape parameter of the intensity function (except exponential intensity). If `shape` is a function or formula, the corresponding initial estimates. If `shape` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`pinitial`	An initial estimate for the initial parameter. With `frailty` dependence, this is the frailty parameter.
`pdepend`	An initial estimate for the serial dependence parameter. For `frailty` dependence, if a value is given here, an autoregression is fitted as well as the frailty.
`pfamily`	An optional initial estimate for the second parameter of a two-parameter power variance family mixture instead of the default gamma mixture. This yields a gamma mixture as `family -> 0`, an inverse Gauss mixture for `family = 0.5`, and a compound distribution of a Poisson-distributed number of gamma distributions for `-1 < family < 0`.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`fscale`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`typsize`	Arguments for nlm.
`stepmax`	Arguments for nlm.

Details

By default, a gamma mixture of the distribution specified in `intensity` is used, as the conditional distribution in the `serial` dependence models, and as a symmetric multivariate (random effect) model for `frailty` dependence.

Unless specified otherwise, the time origin is taken to be zero. The given times are the ends of the periods in which the counts occurred.

Here, the variance, with exponential intensity, is a quadratic function of the mean, whereas, for `nbkal`, it is proportional to the mean function.

If the counts on a unit are clustered, not longitudinal, use the failty dependence with the default exponential intensity, yielding a multivariate negative binomial distribution.

Nonlinear regression models can be supplied as formulae where parameters are unknowns in which case factor variables cannot be used and parameters must be scalars. (See `finterp`.)

Marginal and individual profiles can be plotted using `mprofile` and `iprofile` and residuals with `plot.residuals`.

Value

A list of classes `kalcount` and `recursive` is returned.

Author(s)

J.K. Lindsey

Examples

```
treat <- c(0,0,1,1)
tr <- tcctomat(treat)
dose <- # matrix(rpois(20,10),ncol=5)
matrix(c(9,13,16,7,12,6,9,10,11,9,10,10,7,9,9,9,8,10,15,4),
ncol=5,byrow=TRUE)
dd <- tvctomat(dose)
y <- # matrix(rpois(20,1+3*rep(treat,5)),ncol=5)
restovec(matrix(c(1,1,1,1,0,1,0,1,0,5,3,3,4,1,4,4,2,3,2,5),
ncol=5,byrow=TRUE))
reps <- rmna(y, ccov=tr, tvcov=dd)
#
# log normal intensity, independence model
kalcount(y, intensity="log normal", dep="independence", preg=1,
pshape=0.1)
# random effect
kalcount(y, intensity="log normal", dep="frailty", pinitial=0.1,
preg=1, psh=0.1)
# serial dependence
kalcount(y, intensity="log normal", dep="serial", pinitial=0.1,
preg=1, pdep=0.01, psh=0.1)
# random effect and autoregression
kalcount(y, intensity="log normal", dep="frailty", pinitial=0.1,
pdep=0.1, preg=1, psh=0.1)
# add time-constant variable
kalcount(y, intensity="log normal", pinitial=0.1, psh=0.1,
preg=c(1,0), ccov=treat)
# or equivalently
kalcount(y, intensity="log normal", mu=~treat, pinitial=0.1,
psh=0.1, preg=c(1,0))
# or
kalcount(y, intensity="log normal", mu=~b0+b1*treat,
pinitial=0.1, psh=0.1, preg=c(1,0), envir=reps)
# add time-varying variable
kalcount(y, intensity="log normal", pinitial=0.1, psh=0.1,
preg=c(1,0), ccov=treat, ptvc=0, tvc=dose)
# or equivalently, from the environment
```

```
dosev <- as.vector(t(dose))
kalcount(y, intensity="log normal", mu=~b0+b1*rep(treat,rep(5,4))+b2*dosev,
pinitial=0.1, psh=0.1, ptvc=c(1,0,0))
# or from the reps data object
kalcount(y, intensity="log normal", mu=~b0+b1*treat+b2*dose,
pinitial=0.1, psh=0.1, ptvc=c(1,0,0), envir=reps)
# try power variance family
kalcount(y, intensity="log normal", mu=~b0+b1*treat+b2*dose,
pinitial=0.001, psh=14, ptvc=c(5,-1,0.1), envir=reps,
pfamily=0.8)
```

`kalseries` *Repeated Measurements Models for Continuous Variables with Frailty or Serial Dependence*

Description

`kalseries` is designed to handle repeated measurements models with time-varying covariates. The distributions have two extra parameters as compared to the functions specified by `intensity` and are generally longer tailed than those distributions. Dependence among observations on a unit can be through gamma or power variance family frailties (a type of random effect), with or without autoregression, or one of two types of serial dependence over time.

Usage

```
kalseries(response = NULL, times = NULL, intensity = "exponential",
  depend = "independence", mu = NULL, shape = NULL,
  density = FALSE, ccov = NULL, tvcov = NULL, torder = 0,
  interaction = NULL, preg = NULL, ptvc = NULL, pintercept = NULL,
  pshape = NULL, pinitial = 1, pdepend = NULL, pfamily = NULL,
  delta = NULL, transform = "identity", link = "identity",
  envir = parent.frame(), print.level = 0, ndigit = 10,
  gradtol = 1e-05, steptol = 1e-05, fscale = 1, iterlim = 100,
  typsize = abs(p), stepmax = 10 * sqrt(p %*% p))
```

Arguments

`response`	A list of two column matrices with responses and corresponding times for each individual, one matrix or dataframe of response values, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.
`times`	When response is a matrix, a vector of possibly unequally spaced times when they are the same for all individuals or a matrix of times. Not necessary if equally spaced. Ignored if response has class, `response` or `repeated`.

`intensity`	The form of function to be put in the Pareto distribution. Choices are exponential, Weibull, gamma, normal, logistic, Cauchy, log normal, log logistic, log Cauchy, log Student, inverse Gauss, and gen(eralized) logistic. (For definitions of distributions, see the corresponding [dpqr]distribution help.)
`depend`	Type of dependence. Choices are `independence`, `Markov`, `serial`, and `frailty`.
`mu`	A regression function for the location parameter or a formula beginning with ~, specifying either a linear regression function in the Wilkinson and Rogers notation or a general function with named unknown parameters. Give the initial estimates in `preg` if there are no time-varying covariates and in `ptvc` if there are.
`shape`	A regression function for the shape parameter or a formula beginning with ~, specifying either a linear regression function in the Wilkinson and Rogers notation or a general function with named unknown parameters. It must yield one value per observation.
`density`	If TRUE, the density of the function specified in `intensity` is used instead of the intensity.
`ccov`	A vector or matrix containing time-constant baseline covariates with one row per individual, a model formula using vectors of the same size, or an object of class, `tccov` (created by `tcctomat`). If response has class, `repeated`, the covariates must be supplied as a Wilkinson and Rogers formula unless none are to be used or `mu` is given.
`tvcov`	A list of matrices with time-varying covariate values, observed at the event times in `response`, for each individual (one column per variable), one matrix or dataframe of such covariate values, or an object of class, `tvcov` (created by `tvctomat`). If a time-varying covariate is observed at arbitrary time, `gettvc` can be used to find the most recent values for each response and create a suitable list. If response has class, `repeated`, the covariates must be supplied as a Wilkinson and Rogers formula unless none are to be used or `mu` is given.
`torder`	The order of the polynomial in time to be fitted.
`interaction`	Vector of length equal to the number of time-constant covariates, giving the levels of interactions between them and the polynomial in time in the `linear model`.
`preg`	Initial parameter estimates for the regression model: intercept, one for each covariate in `ccov`, and `torder` plus sum(`interaction`). If `mu` is a formula or function, the parameter estimates must be given here only if there are no time-varying covariates. If `mu` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.
`ptvc`	Initial parameter estimates for the coefficients of the time-varying covariates, as many as in `tvcov`. If `mu` is a formula or function, the parameter estimates must be given here if there are time-varying covariates present.
`pintercept`	The initial estimate of the intercept for the generalized logistic intensity.
`pshape`	An initial estimate for the shape parameter of the intensity function (except exponential intensity). If `shape` is a function or formula, the corresponding initial estimates. If `shape` is a formula with unknown parameters, their estimates must be supplied either in their order of appearance in the expression or in a named list.

`pinitial`	An initial estimate for the initial parameter. With `frailty` dependence, this is the frailty parameter.
`pdepend`	An initial estimate for the serial dependence parameter. For `frailty` dependence, if a value is given here, an autoregression is fitted as well as the frailty.
`pfamily`	An optional initial estimate for the second parameter of a two-parameter power variance family mixture instead of the default gamma mixture. This yields a gamma mixture as `family -> 0`, an inverse Gauss mixture for `family = 0.5`, and a compound distribution of a Poisson-distributed number of gamma distributions for `-1 < family < 0`.
`delta`	Scalar or vector giving the unit of measurement for each response value, set to unity by default. For example, if a response is measured to two decimals, delta=0.01. If the response has been pretransformed, this must be multiplied by the Jacobian. This transformation cannot contain unknown parameters. For example, with a log transformation, `delta=1/y`. The jacobian is calculated automatically for the transform option. Ignored if response has class, `response` or `repeated`.
`transform`	Transformation of the response variable: `identity`, `exp`, `square`, `sqrt`, or `log`.
`link`	Link function for the mean: `identity`, `exp`, `square`, `sqrt`, or `log`.
`envir`	Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`fscale`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`typsize`	Arguments for nlm.
`stepmax`	Arguments for nlm.

Details

By default, a gamma mixture of the distribution specified in `intensity` is used, as the conditional distribution in the `Markov` and `serial` dependence models, and as a symmetric multivariate (random effect) model for `frailty` dependence. For example, with a Weibull `intensity` and `frailty` dependence, this yields a multivariate Burr distribution and with `Markov` or `serial` dependence, univariate Burr conditional distributions.

If a value for `pfamily` is used, the gamma mixture is replaced by a power variance family mixture.

Nonlinear regression models can be supplied as formulae where parameters are unknowns in which case factor variables cannot be used and parameters must be scalars. (See `finterp`.)

Marginal and individual profiles can be plotted using `mprofile` and `iprofile` and residuals with `plot.residuals`.

Value

A list of classes `kalseries` and `recursive` is returned.

Author(s)

J.K. Lindsey

Examples

```
treat <- c(0,0,1,1)
tr <- tcctomat(treat)
dose <- matrix(rpois(20,10), ncol=5)
dd <- tvctomat(dose)
y <- restovec(matrix(rnorm(20), ncol=5), name="y")
reps <- rmna(y, ccov=tr, tvcov=dd)
#
# normal intensity, independence model
kalseries(y, intensity="normal", dep="independence", preg=1, pshape=5)
## Not run:
# random effect
kalseries(y, intensity="normal", dep="frailty", preg=1, pinitial=1, psh=5)
# serial dependence
kalseries(y, intensity="normal", dep="serial", preg=1, pinitial=1,
pdep=0.1, psh=5)
# random effect and autoregression
kalseries(y, intensity="normal", dep="frailty", preg=1, pinitial=1,
pdep=0.1, psh=5)
#
# add time-constant variable
kalseries(y, intensity="normal", dep="serial", pinitial=1,
pdep=0.1, psh=5, preg=c(1,0), ccov=treat)
# or equivalently
kalseries(y, intensity="normal", mu=~treat, dep="serial", pinitial=1,
pdep=0.1, psh=5, preg=c(1,0))
# or
kalseries(y, intensity="normal", mu=~b0+b1*treat, dep="serial",
pinitial=1, pdep=0.1, psh=5, preg=c(1,0), envir=reps)
#
# add time-varying variable
kalseries(y, intensity="normal", dep="serial", pinitial=1, pdep=0.1,
psh=5, preg=c(1,0), ccov=treat, ptvc=0, tvc=dose)
# or equivalently, from the environment
dosev <- as.vector(t(dose))
kalseries(y, intensity="normal",
mu=~b0+b1*rep(treat,rep(5,4))+b2*dosev,
dep="serial", pinitial=1, pdep=0.1, psh=5, ptvc=c(1,0,0))
# or from the reps data object
kalseries(y, intensity="normal", mu=~b0+b1*treat+b2*dose,
dep="serial", pinitial=1, pdep=0.1, psh=5, ptvc=c(1,0,0),
envir=reps)
# try power variance family instead of gamma distribution for mixture
```

```
kalseries(y, intensity="normal", mu=~b0+b1*treat+b2*dose,
dep="serial", pinitial=1, pdep=0.1, psh=5, ptvc=c(1,0,0),
pfamily=0.1, envir=reps)
# first-order one-compartment model
# data objects for formulae
dose <- c(2,5)
dd <- tcctomat(dose)
times <- matrix(rep(1:20,2), nrow=2, byrow=TRUE)
tt <- tvctomat(times)
# vector covariates for functions
dose <- c(rep(2,20),rep(5,20))
times <- rep(1:20,2)
# functions
mu <- function(p) exp(p[1]-p[3])*(dose/(exp(p[1])-exp(p[2]))*
(exp(-exp(p[2])*times)-exp(-exp(p[1])*times)))
shape <- function(p) exp(p[1]-p[2])*times*dose*exp(-exp(p[1])*times)
# response
conc <- matrix(rgamma(40,shape(log(c(0.01,1))),
scale=mu(log(c(1,0.3,0.2))))/shape(log(c(0.1,0.4))),ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
ncol=20,byrow=TRUE)[,1:19])
conc <- restovec(ifelse(conc>0,conc,0.01))
reps <- rmna(conc, ccov=dd, tvcov=tt)
#
# constant shape parameter
kalseries(reps, intensity="gamma", dep="independence", mu=mu,
ptvc=c(-1,-1.1,-1), pshape=1.5)
# or
kalseries(reps, intensity="gamma", dep="independence",
mu=~exp(absorption-volume)*
dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times)),
ptvc=list(absorption=-1,elimination=-1.1,volume=-1),
pshape=1.2)
# add serial dependence
kalseries(reps, intensity="gamma", dep="serial", pdep=0.9,
mu=~exp(absorption-volume)*
dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times)),
ptvc=list(absorption=-1,elimination=-1.1,volume=-1),
pshape=0.2)
# time dependent shape parameter
kalseries(reps, intensity="gamma", dep="independence", mu=mu,
shape=shape, ptvc=c(-1,-1.1,-1), pshape=c(-3,0))
# or
kalseries(reps, intensity="gamma", dep="independence",
mu=~exp(absorption-volume)*
dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times)),
ptvc=list(absorption=-1,elimination=-1.1,volume=-1),
shape=~exp(b1-b2)*times*dose*exp(-exp(b1)*times),
pshape=list(b1=-3,b2=0))
# add serial dependence
```

```
kalseries(reps, intensity="gamma", dep="serial", pdep=0.5,
mu=~exp(absorption-volume)*
dose/(exp(absorption)-exp(elimination))*
(exp(-exp(elimination)*times)-exp(-exp(absorption)*times)),
ptvc=list(absorption=-1,elimination=-1.1,volume=-1),
shape=~exp(b1-b2)*times*dose*exp(-exp(b1)*times),
pshape=list(b1=-3,b2=0))

## End(Not run)
```

`logitord` *Ordinal Random Effects Models with Dropouts*

Description

`logitord` fits an longitudinal proportional odds model in discrete time to the ordinal outcomes and a logistic model to the probability of dropping out using a common random effect for the two.

Usage

```
logitord(y, id, out.ccov = NULL, drop.ccov = NULL, tvcov = NULL,
  out.tvcov = !is.null(tvcov), drop.tvcov = !is.null(tvcov), pout,
  pdrop, prand.out, prand.drop, random.out.int = TRUE,
  random.out.slope = !is.null(tvcov), random.drop.int = TRUE,
  random.drop.slope = !is.null(tvcov), binom.mix = 5, fcalls = 900,
  eps = 1e-04, print.level = 0)
```

Arguments

`y`	A vector of binary or ordinal responses with levels 1 to k and 0 indicating drop-out.
`id`	Identification number for each individual.
`out.ccov`	A vector, matrix, or model formula of time-constant covariates for the outcome regression, with variables having the same length as `y`.
`drop.ccov`	A vector, matrix, or model formula of time-constant covariates for the drop-out regression, with variables having the same length as `y`.
`tvcov`	One time-varying covariate vector.
`out.tvcov`	Include the time-varying covariate in the outcome regression.
`drop.tvcov`	Include the time-varying covariate in the drop-out regression.
`pout`	Initial estimates of the outcome regression coefficients, with length equal to the number of levels of the response plus the number of covariates minus one.
`pdrop`	Initial estimates of the drop-out regression coefficients, with length equal to one plus the number of covariates.
`prand.out`	Optional initial estimates of the outcome random parameters.
`prand.drop`	Optional initial estimates of the drop-out random parameters.

`random.out.int`	If TRUE, the outcome intercept is random.
`random.out.slope`	If TRUE, the slope of the time-varying covariate is random for the outcome regression (only possible if a time-varying covariate is supplied and if out.tvcov and random.out.int are TRUE).
`random.drop.int`	If TRUE, the drop-out intercept is random.
`random.drop.slope`	If TRUE, the slope of the time-varying covariate is random for the drop-out regression (only possible if a time-varying covariate is supplied and if drop.tvcov and random.drop.int are TRUE).
`binom.mix`	The total in the binomial distribution used to approximate the normal mixing distribution.
`fcalls`	Number of function calls allowed.
`eps`	Convergence criterion.
`print.level`	If 1, the iterations are printed out.

Value

A list of class `logitord` is returned.

Author(s)

T.R. Ten Have and J.K. Lindsey

References

Ten Have, T.R., Kunselman, A.R., Pulkstenis, E.P. and Landis, J.R. (1998) Biometrics 54, 367-383, for the binary case.

Examples

```
y <- trunc(runif(20,max=4))
id <- gl(4,5)
age <- rpois(20,20)
times <- rep(1:5,4)
logitord(y, id=id, out.ccov=~age, drop.ccov=age, pout=c(1,0,0),
pdrop=c(1,0))
logitord(y, id, tvcov=times, pout=c(1,0,0), pdrop=c(1,0))
```

`marg.hom` *Marginal Homogeneity Models*

Description

`marg.hom` fits a marginal homogeneity model to a contingency table that has two margins of equal size.

Usage

```
marg.hom(freq, marg1, marg2)
```

Arguments

`freq`	Vector of frequencies
`marg1`	Factor variable for the first margin
`marg2`	Factor variable for the second margin

Value

A list containing the call, the model, the deviance, the degrees of freedom, the aic, the fitted values, and the residuals is returned.

Author(s)

J.K. Lindsey

Examples

```
# 4x4x2 table in Freq, with margins indexed by Left and Right
Freq <- rpois(32,10)
Left <- gl(4,1,32)
Right <- gl(4,4,32)
marg.hom(Freq, Left, Right)
```

`nbkal` *Negative Binomial Models with Kalman Update*

Description

`nbkal` fits a negative binomial regression with Kalman update over time. The variance is proportional to the mean function, whereas, for `kalcount` with exponential intensity, it is a quadratic function of the mean.

Usage

```
nbkal(response, times, mu, preg, pdepend, kalman = TRUE,
  print.level = 0, ndigit = 10, gradtol = 1e-05, steptol = 1e-05,
  fscale = 1, iterlim = 100, typsize = abs(p), stepmax = 10 *
  sqrt(p %*% p))
```

Arguments

`response`	A list of two column matrices with counts and corresponding times for each individual, one matrix or dataframe of counts, or an object of class, response (created by `restovec`) or repeated (created by `rmna` or `lvna`).
`times`	When response is a matrix, a vector of possibly unequally spaced times when they are the same for all individuals or a matrix of times. Not necessary if equally spaced. Ignored if response has class, response or repeated.
`mu`	The mean function.
`preg`	The initial parameter estimates for the mean function.
`pdepend`	The estimates for the dependence parameters, either one or three.
`kalman`	If TRUE, fits the kalman update model, otherwise, a standard negative binomial distribution.
`print.level`	Arguments for nlm.
`ndigit`	Arguments for nlm.
`gradtol`	Arguments for nlm.
`steptol`	Arguments for nlm.
`fscale`	Arguments for nlm.
`iterlim`	Arguments for nlm.
`typsize`	Arguments for nlm.
`stepmax`	Arguments for nlm.

Details

Marginal and individual profiles can be plotted using `mprofile` and `iprofile` and residuals with `plot.residuals`.

Value

A list of classes `nbkal` and `recursive` is returned.

Author(s)

P. Lambert and J.K. Lindsey

References

Lambert, P. (1996) Applied Statistics 45, 31-38.

Lambert, P. (1996) Biometrics 52, 50-55.

Examples

```
y <- matrix(rnbinom(20,5,0.5), ncol=5)
times <- matrix(rep(seq(10,50,by=10),4), ncol=5, byrow=TRUE)
y0 <- matrix(rep(rnbinom(5,5,0.5),4), ncol=5, byrow=TRUE)
mu <- function(p) p[1]*log(y0)+(times<30)*p[2]*
(times-30)+(times>30)*p[3]*(times-30)
nbkal(y, preg=c(1.3,0.008,-0.05), times=times, pdep=1.2, mu=mu)
```

Index