Package ‘ptmixed’

June 13, 2020

Title Poisson-Tweedie Generalized Linear Mixed Model

Version 0.5.4

Description Fits the Poisson-Tweedie generalized linear mixed
model described in Signorelli et al. (2020, arXiv preprint:
<arXiv:2004.11193>). Likelihood approximation based on adaptive Gauss
Hermite quadrature rule.

License GPL-3

URL http://mirkosignorelli.wixsite.com/home/software

Depends R (>= 3.5.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports aod, GLMMadaptive, graphics, lme4, matrixcalc, moments,
mvtnorm, numDeriv, tweeDEseq, rstudioapi

Suggests knitr, rmarkdown

NeedsCompilation no

Author Mirko Signorelli [aut, cre],
Pietro Spitali [aut],
Roula Tsonaka [aut]

Maintainer Mirko Signorelli <msignorelli.rpackages@gmail.com>

Repository CRAN

Date/Publication 2020-06-13 07:10:07 UTC

R topics documented:

`df1` *Example dataset with longitudinal counts*

Description

An example dataset of a study with longitudinal counts, used to illustrate how `ptmixed()` and `nbmixed()` work.

Usage

```
data(df1)
```

Format

A data frame with 18 rows and 5 variables

Author(s)

Mirko Signorelli

See Also

examples in the `nbmixed` and `ptmixed` help pages `loglik.pt.1re` *Loglikelihood of Poisson-Tweedie generalized linear mixed model with random intercept*

Description

Evaluates the loglikelihood of a Poisson-Tweedie generalized linear mixed model with random intercept, using the adaptive Gauss-Hermite quadrature rule.

Usage

```
loglik.pt.1re(
  beta,
  D,
  a,
  Sigma,
  y,
  X,
  Z,
  id,
  offset = NULL,
  GHk = 5,
  tol = 9.88131291682493e-324,
  GHs = NULL
)
```

Arguments

`beta`	Vector of regression coefficients
`D`	Dispersion parameter (must be > 1)
`a`	Power parameter (must be < 1)
`Sigma`	A matrix with the variance of the random intercept
`y`	Response vector (discrete)
`X`	Design matrix for the fixed effects
`Z`	Design matrix for the random effects
`id`	Id indicator (it should be numeric)
`offset`	Offset term to be added to the linear predictor
`GHk`	Number of quadrature points (default is 5)
`tol`	Tolerance value for the evaluation of the probability mass function of the Poisson-Tweedie distribution
`GHs`	Quadrature points at which to evaluate the loglikelihood. If `NULL` (default value), the GH quadrature points are computed internally

Value

The loglikelihood value obtained using a Gauss-Hermite quadrature approximation with `GHk` quadrature points.

Author(s)

Mirko Signorelli

See Also

`ptmixed` and the examples therein

`make.spaghetti` *Generate a spaghetti plot to visualize longitudinal data*

Description

A spaghetti plot, or trajectory plot, is a plot that allows to compare across individuals or groups the trajectories of a longitudinal outcome

Usage

```
make.spaghetti(
  x,
  y,
  id,
  group = NULL,
  data,
  col = NULL,
  pch = 16,
  lty = 1,
  lwd = 1,
  title = "",
  xlab = NA,
  ylab = NA,
  legend.title = "",
  ylim = NULL,
  cex.axis = 1,
  cex.title = 1,
  cex.lab = 1,
  cex.leg = 1,
  legend.inset = -0.3
)
```

Arguments

`x`	the time variable (numeric vector)
`y`	the longitudinal outcome (numeric vector)
`id`	the subject indicator
`group`	the group that each subject belongs to (optional, do not specify if not relevant)
`data`	a data frame containing x, y, id and optionally group
`col`	a vector of colors (optional)
`pch`	dot type
`lty`	line type
`lwd`	line width
`title`	plot title
`xlab`	label for the x axis
`ylab`	label for the y axis
`legend.title`	legend title
`ylim`	limits for the y axis
`cex.axis`	font size for the axes
`cex.title`	title font size
`cex.lab`	font size for axis labels
`cex.leg`	font size for the legend
`legend.inset`	moves legend more to the left / right

Author(s)

Mirko Signorelli

Examples

```
# generate example data
set.seed(123)
n = 12; t = 6
id = rep(1:n, each = t)
rand.int = rep(rnorm(n, sd = 0.5), each = t)
group = rep(c(0,1), each = n*t/2)
time = rep(0:(t-1), n)
offset = rnorm(n*t, sd = 0.3)
beta = c(3, 0, 0.1, 0.3)
X = model.matrix(~group + time + group*time)
mu = 2^(X %*% beta + rand.int + offset)
y = rpois(n*t, lambda = mu)
group = ifelse(group == 0, 'control', 'treatment')
data.long = data.frame(y, group, time, id, offset)
rm(list = setdiff(ls(), 'data.long'))
```

```
# create plot
make.spaghetti(x = time, y, id, group,
data = data.long, title = 'spaghetti plot')
```

`nbglm` *Negative binomial generalized linear model*

Description

Estimates a negative binomial generalized linear model.

Usage

```
nbglm(
  formula,
  offset = NULL,
  data,
  maxit = c(500, 1e+05),
  trace = T,
  theta.start = NULL
)
```

Arguments

`formula`	A formula for the fixed effects part of the model. It should be in the form `y ~ x1 + x2`
`offset`	An offset to be added to the linear predictor. Default is `NULL`.
`data`	A data frame containing the variables declared in `formula`.
`maxit`	Vector containing the maximum number of iterations used in optim by the BFGS method and, if this fails, by the Nelder-Mead method
`trace`	Logical value. If `TRUE`, additional information is printed during the optimization. Default is `TRUE`.
`theta.start`	Numeric vector comprising initial parameter values for the vector of regression coefficients and the dispersion parameter

Details

Maximum likelihood estimation of a negative binomial GLM (the NB distribution is obtained as special case of the Poisson-Tweedie distribution when a = 0).

Value

A list containing the following elements: function's call (`call`); maximum likelihood estimate (`mle`); value of the loglikelihood at the mle (`logl`); `convergence` value (if 0, the optimization converged); the observed Fisher information (`fisher.info`) and the starting values used in the optimization (`theta.init`)

Author(s)

Mirko Signorelli

See Also

ptmixed for the Poisson-Tweedie GLMM

Examples

```
data(df1, package = 'ptmixed')

# estimate the model
fit1 = nbglm(formula = y ~ group*time, data = df1)

# view model summary
summary(fit1)
```

---

nbmixed *Negative binomial generalized linear mixed model*

Description

Estimates the negative binomial generalized linear mixed model with random intercept (here, the NB distribution is obtained as special case of the Poisson-Tweedie distribution when a = 0). Likelihood approximation for the model is based on the adaptive Gauss-Hermite quadrature rule.

Usage

```
nbmixed(
  fixef.formula,
  id,
  offset = NULL,
  data,
  npoints = 5,
  hessian = T,
  trace = T,
  theta.start = NULL,
  reltol = 1e-08,
  maxit = c(10000, 100),
  freq.updates = 200,
  min.var.init = 0.001
)
```

Arguments

`fixef.formula`	A formula for the fixed effects part of the model. It should be in the form `y ~ x1 + x2`
`id`	A variable to distinguish observations from the same subject.
`offset`	An offset to be added to the linear predictor. Default is `NULL`.
`data`	A data frame containing the variables declared in `fixef.formula`.
`npoints`	Number of quadrature points employed in the adaptive quadrature. Default is 5.
`hessian`	Logical value. If `TRUE`, the hessian matrix is evaluated at the MLE to derive the observed Fisher information matrix. Default is `TRUE`.
`trace`	Logical value. If `TRUE`, additional information is printed during the optimization. Default is `TRUE`.
`theta.start`	Numeric vector comprising initial parameter values for the vector of regression coefficients, the dispersion parameter (using the same parametrization of `ptmixed`) and the variance of the random intercept. Default is `NULL`: initial parameter estimates are computed automatically by the function.
`reltol`	Relative tolerance to be used in optim. Default to 1e-8
`maxit`	Vector containing the maximum number of iterations used in optim by the Nelder-Mead method and, if this fails, by the BFGS method
`freq.updates`	Number of iterations after which the quadrature points are updated when the Nelder-Mead algorithm is used for the optimization. Default value is 200. To update the quadrature points at every iteration (note that this may make the computation about 10x slower), set `freq.updates = 1` or `freq.updates = NA`. The function first tries to optimize the loglikelihood using the Nelder-Mead algorithm, updating the quadrature points every `freq.updates` iterations. If this fails to converge, a second attempt is made using the BFGS algorithm, for which the quadrature points are updated at every iteration.
`min.var.init`	If the initial estimate of the variance of the random intercept is smaller than this value, estimation is stopped and the user is advided to use the simpler Poisson-Tweedie GLM is used. Default is 1e-3.

Value

A list containing the following elements: function's call (`call`); maximum likelihood estimate (`mle`); value of the loglikelihood at the mle (`logl`); `convergence` value (if 0, the optimization converged); the observed Fisher information (`fisher.info`), if `hessian = T`; the number of quadrature points used (`quad.points`) and the starting value used in the optimization (`theta.init`); relevant warnings (`warnings`).

Author(s)

Mirko Signorelli

See Also

`summary.ptglmm`, `ranef`

Examples

```
data(df1, package = 'ptmixed')
head(df1)

# 1) Quick example (hessian and SEs not computed)

# estimate the model
fit1 = nbmixed(fixef.formula = y ~ group + time, id = id,
              offset = offset, data = df1, npoints = 5,
              freq.updates = 200, hessian = FALSE, trace = TRUE)

# print summary:
summary(fit1, wald = FALSE)


# 2) Full computation, including computation of SEs

# estimate the model
fit2 = nbmixed(fixef.formula = y ~ group + time, id = id,
              offset = offset, data = df1, npoints = 5,
              freq.updates = 200, hessian = TRUE, trace = TRUE)

# print summary:
summary(fit2)

# extract summary:
results = summary(fit2)
ls(results)
results$coefficients
```

`pmf` *Probability mass function plot for a discrete variable*

Description

This function produces a simple plot of the probability mass function of a discrete variable

Usage

```
pmf(
  x,
  absolute = T,
  xlim = NULL,
  lwd = 1,
  col = "black",
  title = NULL,
  xlab = NULL,
  bty = "l",
  cex.title = NULL,
  cex.axis = NULL
)
```

Arguments

`x`	the (discrete) variable of interest
`absolute`	logical. If `TRUE` (default) absolute frequencies are plotted, if `FALSE` relative frequencies
`xlim`	limits for the x axis
`lwd`	line width
`col`	color used for the vertical frequency bars
`title`	plot title
`xlab`	label for the x axis
`bty`	box type (default is `bty="l"`)
`cex.title`	title font size
`cex.axis`	font size for the axes

Author(s)

Mirko Signorelli

Examples

```
pmf(cars$speed)
pmf(cars$speed, absolute = FALSE)
pmf(cars$speed, lwd = 2, col = 'blue')
```

---

ptglm *Poisson-Tweedie generalized linear model*

Description

Estimates a Poisson-Tweedie generalized linear model.

Usage

```
ptglm(
  formula,
  offset = NULL,
  data,
  maxit = c(500, 1e+05),
  trace = T,
  theta.start = NULL
)
```

Arguments

`formula`	A formula for the fixed effects part of the model. It should be in the form `y ~ x1 + x2`
`offset`	An offset to be added to the linear predictor. Default is `NULL`.
`data`	A data frame containing the variables declared in `formula`.
`maxit`	Vector containing the maximum number of iterations used in optim by the BFGS method and, if this fails, by the Nelder-Mead method
`trace`	Logical value. If `TRUE`, additional information is printed during the optimization. Default is `TRUE`.
`theta.start`	Numeric vector comprising initial parameter values for the vector of regression coefficients, the dispersion parameter and the power parameter (to be specified exactlyin this order!).

Value

A list containing the following elements: function's call (`call`); maximum likelihood estimate (`mle`); value of the loglikelihood at the mle (`logl`); `convergence` value (if 0, the optimization converged); the observed Fisher information (`fisher.info`) and the starting values used in the optimization (`theta.init`)

Author(s)

Mirko Signorelli

See Also

`ptmixed` for the Poisson-Tweedie GLMM

Examples

```
data(df1, package = 'ptmixed')

# estimate the model
fit1 = ptglm(formula = y ~ group*time, data = df1)

# view model summary:
summary(fit1)
```

`ptmixed` *Poisson-Tweedie generalized linear mixed model*

Description

Estimates the Poisson-Tweedie generalized linear mixed model with random intercept. Likelihood approximation for the model is based on the adaptive Gauss-Hermite quadrature rule.

Usage

```
ptmixed(
  fixef.formula,
  id,
  offset = NULL,
  data,
  npoints = 5,
  hessian = T,
  trace = T,
  theta.start = NULL,
  reltol = 1e-08,
  maxit = c(10000, 100),
  freq.updates = 200,
  min.var.init = 0.001
)
```

Arguments

`fixef.formula`	A formula for the fixed effects part of the model. It should be in the form `y ~ x1 + x2`
`id`	A variable to distinguish observations from the same subject.
`offset`	An offset to be added to the linear predictor. Default is `NULL`.
`data`	A data frame containing the variables declared in `fixef.formula`.
`npoints`	Number of quadrature points employed in the adaptive quadrature. Default is 5.
`hessian`	Logical value. If `TRUE`, the hessian matrix is evaluated at the MLE to derive the observed Fisher information matrix. Default is `TRUE`.
`trace`	Logical value. If `TRUE`, additional information is printed during the optimization. Default is `TRUE`.
`theta.start`	Numeric vector comprising initial parameter values for the vector of regression coefficients, the dispersion parameter, the power parameter and the variance of the random intercept (to be specified exactlyin this order!). Default is `NULL`: initial parameter estimates are computed automatically by the function.
`reltol`	Relative tolerance to be used in optim. Default to 1e-8
`maxit`	Vector containing the maximum number of iterations used in optim by the Nelder-Mead method and, if this fails, by the BFGS method
`freq.updates`	Number of iterations after which the quadrature points are updated when the Nelder-Mead algorithm is used for the optimization. Default value is 200. To update the quadrature points at every iteration (note that this may make the computation about 10x slower), set `freq.updates = 1` or `freq.updates = NA`. The function first tries to optimize the loglikelihood using the Nelder-Mead algorithm, updating the quadrature points every `freq.updates` iterations. If this fails to converge, a second attempt is made using the BFGS algorithm, for which the quadrature points are updated at every iteration.
`min.var.init`	If the initial estimate of the variance of the random intercept is smaller than this value, estimation is stopped and the user is advided to use the simpler Poisson-Tweedie GLM is used. Default is 1e-3.

Value

A list containing the following elements: function's call (`call`); maximum likelihood estimate (`mle`); value of the loglikelihood at the mle (`logl`); `convergence` value (if 0, the optimization converged); the observed Fisher information (`fisher.info`), if `hessian = T`; the number of quadrature points used (`quad.points`) and the starting value used in the optimization (`theta.init`); relevant warnings (`warnings`).

Author(s)

Mirko Signorelli

See Also

`summary.ptglmm`, `ranef`

Examples

```
data(df1, package = 'ptmixed')
head(df1)

# 1) Quick example (just 1 quadrature point, hessian and SEs
# not computed - NB: we recommend to increase the number of
# quadrature points to obtain much more accurate estimates,
# as shown in example 2 below where we use 5 quadrature points)

# estimate the model
fit1 = ptmixed(fixef.formula = y ~ group + time, id = id,
              offset = offset, data = df1, npoints = 1,
              freq.updates = 200, hessian = FALSE, trace = TRUE)

# print summary:
summary(fit1, wald = FALSE)


# 2) Full computation that uses more quadrature points
# for the likelihood approximation and includes numeric
# evaluation of the hessian matrix

# estimate the model:
fit2 = ptmixed(fixef.formula = y ~ group + time, id = id,
              offset = offset, data = df1, npoints = 5,
              freq.updates = 200, hessian = TRUE, trace = TRUE)

# print summary:
summary(fit2)

# extract summary:
results = summary(fit2)
ls(results)
results$coefficients
```

ranef — *Compute random effects for Poisson-Tweedie and negative binomial mixed model*

Description

Compute the BLUP (best linear unbiased predictor) of the random effects for the Poisson-Tweedie and negative binomial generalized linear mixed models (fitted through `ptmixed` and `nbmixed` respectively)

Usage

```
ranef(obj)
```

Arguments

`obj`	an object of class `ptglmm` (obtained from `ptmixed` or `nbmixed`).

Value

A vector with the EB estimates of the random effects

Author(s)

Mirko Signorelli

See Also

`ptmixed`, `nbmixed`

Examples

```
data(df1, package = 'ptmixed')

# estimate a Poisson-Tweedie or negative binomial GLMM (using
# ptmixed() or nbmixed())
fit0 = nbmixed(fixef.formula = y ~ group + time, id = id,
              offset = offset, data = df1, npoints = 5,
              freq.updates = 200, hessian = FALSE, trace = TRUE)

# obtain random effect estimates
ranef(obj = fit0)
```

`simulate_ptglmm` *Simulate data from the Poisson-Tweedie generalized linear mixed model*

Description

Simulates a dataset comprising t repeated measurements for n subjects from a Poisson-Tweedie GLMM. Subjects are assumed to belong to two different groups. The linear predictor comprises an intercept, main effects of group and of time, and the interaction between time and group; a random intercept; and, optionally, a normally-distributed offset term.

Usage

```
simulate_ptglmm(
  n = 20,
  t = 5,
  seed = 1,
  beta = c(3, 0, 0, 0.4),
  D = 1.5,
  a = -1,
  sigma2 = 0.8^2,
  offset = F
)
```

Arguments

`n`	number of subjects
`t`	number of time points (0, 1, ..., t-1)
`seed`	seed for random number generation
`beta`	vector of regression coefficients, to be specified in this order: intercept, group main effect, time main effect, group*time interaction
`D`	dispersion parameter of the Poisson-Tweedie distribution (D > 1)
`a`	power parameter of the Poisson-Tweedie distribution (a < 1)
`sigma2`	Variance of the subject-specific random intercept
`offset`	Logical value. If `TRUE`, a normally-distributed offset is added to the linear predictor.

Value

A list containing the following elements: a dataframe (`data`) containing the response y, the subject id, the group indicator and time; a vector with the true random intercept values (`true.randint`).

Author(s)

Mirko Signorelli

Examples

```
# simulate a simple, small dataset
example1 = simulate_ptglmm(n = 5, t = 2)
example1$data
# the function allows to set several different parameters
example2 = simulate_ptglmm(n = 20, t = 5, seed = 1,
beta = c(2.2, 1.2, 0.3, -0.5), D = 1.8, a = 0.5,
sigma2 = 0.7, offset = TRUE)
# view the distribution of the response variable:
pmf(example2$data$y)
# visualize the data with a trajectory plot:
make.spaghetti(x = time, y = y, id = id,
group = group, data = example2$data)
```

`summary.ptglm` *Summarizing Poisson-Tweedie and negative binomial GLM estimation results*

Description

Provides parameter estimates, standard errors and univariate Wald test for the Poisson-Tweedie and the negative binomial generalized linear models (fitted through `ptglm` and `nbglm` respectively)

Usage

```
## S3 method for class 'ptglm'
summary(object, silent = F, ...)
```

Arguments

`object`	an object of class `ptglm` (obtained from `ptglm` or `nbglm`). #' @param silent logical. If `TRUE`, information on parameter estimates and tests is not printed on screen. Default is `FALSE` (info is printed)
`silent`	logical. If `TRUE`, information on parameter estimates and tests is not printed on screen. Default is `FALSE` (info is printed)
`...`	Further arguments passed to or from other methods.

Value

A list with the following elements: `logl`, `coefficients`, `D`, `a`

Author(s)

Mirko Signorelli

See Also

`ptglm`, `nbglm` and the examples therein

summary.ptglmm	*Summarizing Poisson-Tweedie and negative binomial mixed model estimation results*

Description

Provides parameter estimates, standard errors and univariate Wald test for the Poisson-Tweedie and negative binomial generalized linear mixed models (fitted through `ptmixed` and `nbmixed` respectively)

Usage

```
## S3 method for class 'ptglmm'
summary(object, wald = T, silent = F, ...)
```

Arguments

`object`	an object of class `ptglmm` (obtained from `ptmixed` or `nbmixed`).
`wald`	logical. If `TRUE`, standard errors and univariate Wald test are computed. Default is `TRUE`.
`silent`	logical. If `TRUE`, information on parameter estimates and tests is not printed on screen. Default is `FALSE` (info is printed)
`...`	Further arguments passed to or from other methods.

Value

A list with the following elements: value of the loglikelihood at the MLE (`logl`), table with maximum likelihood estimates of the regression coefficients, SEs and Wald tests `coefficients`, and maximum likelihood estimates of the other parameters (`D`, `a` and `sigma2`)

Author(s)

Mirko Signorelli

See Also

`ptmixed`, `nbmixed` and the examples therein

wald.test *Wald test for regression coefficients*

Description

Compute a multivariate Wald test for one of the following models: Poisson-Tweedie GLMM, negative binomial GLMM, Poisson-Tweedie GLM, negative binomial GLM. The null hypothesis has to be specified in the (matrix) form \$L b = k\$, where \$b\$ is the vector of regression coefficients and \$L\$ and \$k\$ are defined below

Usage

```
wald.test(obj, L, k = NULL)
```

Arguments

`obj`	an object of class `ptglmm` (obtained from `ptmixed` or `nbmixed`) or `ptglm` (obtained from `ptglm` or `nbglm`)
`L`	a matrix used to define the hypothesis to test, in the form \$L b = k\$
`k`	a vector used to define the hypothesis to test, in the form \$L b = k\$. Default is a null vector (\$L b = 0\$)

Value

A data frame with the result of the test

Author(s)

Mirko Signorelli

Examples

```
# generate data
data(df1, package = 'ptmixed')

# estimate one of the following models: a Poisson-Tweedie or
# negative binomial GLMM (using ptmixed() or nbmixed()), or
# a Poisson-Tweedie or negative binomial GLM (using ptglm()
# or nbgml())
fit1 = nbglm(formula = y ~ group*time, data = df1)

# define L for beta2 = beta4 = 0
L = matrix(0, nrow = 2, ncol = 4)
L[1, 2] = L[2, 4] = 1

# compute multivariate Wald test
wald.test(obj = fit1, L = L, k = NULL)
```

Index