Package 'lmeNB'

July 2, 2014

Type Package

Title Fit negative binomial mixed-effect regression model.

Version 1.2

Date 2014-01-23

Author Zhao, Y. and Kondo, Y.

Maintainer Yumi Kondo <y.kondo@stat.ubc.ca>

Description Fit negative binomial mixed-effect regression model. For the distribution of subject-specific random intercept, a gamma or log-normal distributions or semi-parametric procedure are allowed. For the safety monitoring, subject-specific conditional probability index can be computed. For details, see Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Depends numDeriv,statmod,lmeNBBayes

NeedsCompilation no

Date/Publication 2014-01-24 22:13:26

R topics documented:

CP.ar1.se 2
CP.se 5
index.batch 7
lmeNB 10
mle.a3.fun 13
mle.ar1.fun 18
mle.ar1.non3 24
mle.fun 30
rNBME.R 35

Index 39

CP.ar1.se *Compute a conditional probability of observing a set of counts as extreme as the new observations of a subjectvisit given the previous observations of the same subject based on the negative binomial mixed-effect AR(1) model.*

Description

Given the parameter estimates of `c( alpha,theta,delta, beta0,beta1,...)` of the negative binomial mixed effect AR(1) model, these functions compute the following conditional probability:

Pr(q(Y_i,new)>=q(y_i,new)| Y_i,pre=y_i,pre)

where *y_i,new* and *y_i,pre* are vectors of previous and new observations from subject *i* and q() is a function which provides a scalar summary of the new observations.

These functions are subroutines of `index.batch`.

When input the maximum likelihood estimates of the parameters, `CP.ar1.se` returns the estimate of the conditional probability and its asymptotic standard error of a subject based on AR(1) model. The computation for the probability is done by its subroutine `jCP.ar1`, which has two subroutines `CP1.ar1` and `MCCP.ar1`. `CP1.ar1` computes the probability via the adaptive quadrature while `MCCP.ar1` computes the probability via the Monte Carlo integration.

Usage

```
CP.ar1.se(tpar, ypre, ynew, y2m = NULL, XM, stp,
  dist = "G", V, mc = FALSE, qfun = "sum")

jCP.ar1(tpar, ypre, ynew,
y2m=NULL, XM, stp,
        mod = "G", LG = FALSE, MC = FALSE, N = 40000, qfun = "sum", oth = NULL)

CP1.ar1(ypre, ynew, y2m = NULL,
stp, u, th, a,
dt, mod = "G", gh, qfun = "sum")

MCCP.ar1(ypre, ynew, stp, u, th, a, dt, mod = "G", Ns = 1000, gh, qfun = "sum")
```

Arguments

tpar	A vector of length 4 + # covariates, containing the estimates of the model in the order that `log(a),log(theta),logit(delta),beta0,beta1....` If random effects are assumed to be from the log-normal distribution, then `theta` is a variance estimate of the random effect. If random effects are assumed to be from the gamma distribution, then `theta` is a shape parameter estimate. If the semi-parametric approach is taken, then `theta` is a place holder and can be any value.

ypre	A vector of counts of the length the number of previous observations.
y2m	Internal use only. Set as y2m=NULL
ynew	A vector of counts of the length the number of new observations.
XM	A *n_i* by # covariates matrix containing the covariate values of subject *i*, whre *n_i* is the total number of previous and new observations. If there is no covariate, i.e., the model only has an intercept term, then set XM=NULL.
stp	A vector of length the total number of previous and new observations. The first entry must be zero. For example, if there is no missing scans and there are five repeated measures, then stp=c(0,1,1,1,1). If the third scan is missing and there are four repeated measures, then stp=c(0,1,2,1).
mod	If mod="G", then the conditional probability is computed by assuming the random effect is from the gamma distribution. If mod="N", then the conditional probability is computed by assuming the random effect is from the log-normal distribution. If mod="NoN", then the conditional probability is computed based on the semi-parametric model. Note that this option is NOT accepted in CP.ar1.se.
LG	If LG=TRUE then the logit of the conditional probability is returned.
MC	If TRUE then the function MCCP.ar1 is called and the Monte carlo integration is performed. Fast but could be unreliable; not recommended for computing the confidence intervals. If FALSE then the function CP1.ar1 is called and the adaptive quadrature is performed. Slow but reliable.
N	The number of the Monte carlo integration. Necessary if MC=TRUE
qfun	The summary statistics q.
oth	If mod="NoN", othr must be the frequency table of the random effects, which can be obtained based on dist=obj$gi where obj is the output of mle.ar1.non3. If else, it can be unspecified.
dist	Same as mod.
V	The estimated variance covariance matrix of the parameters log(a),log(theta),log(delta),beta0,be
th	The estimated *theta*.
a	The estimated *alpha*.
dt	The estimated *delta*.
Ns	Same as N.
mc	Same as MC.
u	A vector of length the number of repeated measures, containing the estimated mean *(mu_ij;j=1,...,n_i)*. If the mean of *Y_ij* is modeled linearly on beta with the log-link function, then u=exp(beta0+ XM[,1]*beta1+XM[,2]*beta2+...)).
gh	Same as oth.

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The functions to fit the relevant models: mle.fun, mle.ar1.fun, mle.a3.fun, mle.ar1.non3,

The other subroutines of index.batch to compute the conditional probability index:

CP.se, jCP,

The functions to generate simulated datasets: rNBME.R.

Examples

```
## Not run:
ilgt <- function (x)
{
    tem = exp(x)
    res = tem/(1 + tem)
    return(res)
}
lgt <- function (p)
{
    log(p/(1 - p))
}
## the vector of a parameter estimates if log(a),log(theta),logit(delta),beta0.
tpar  <- c(log(2),log(0.5),lgt(0.5),2)
ypre <- c(0, 1)
ynew <- c(1, 0, 0)
## No covariate
XM <- NULL
## no missing visit
stp <- c(0,1,1,1,1)
dist = "G"
## The estimate of the variance covariance matrix
V <-
matrix(
c( 0.17720309, -0.240418504,  0.093562548,  0.009141980,
  -0.24041850,  0.605132808, -0.160454773, -0.003978118,
   0.09356255, -0.160454773,  0.095101658,  0.005661923,
   0.00914198, -0.003978118,  0.005661923,  0.007574769),
nrow=4)

## the estimate of the conditional probability based on the sum summary statistics and its SE
CP.ar1.se(tpar = tpar, ypre = ypre, ynew = ynew,
  XM =XM, stp = stp,
  dist = dist, V = V, mc = FALSE, qfun = "sum")

## the estimate of the conditional probability based on the max summary statistics and its SE
CP.ar1.se(tpar = tpar, ypre = ypre, ynew = ynew,
  XM =XM, stp = stp,
  dist = dist, V = V, mc = FALSE, qfun = "max")


## CP.ar1.se calls for jCP.ar1 to compute the estimate of the conditional probability
## the estimate of the conditional probability based on the sum summary statistics
```

```
jCP.ar1(tpar = tpar, ypre = ypre, ynew = ynew,
y2m=NULL,  XM =XM, stp = stp,
        mod = dist, LG = FALSE, MC = FALSE, N = 40000, qfun = "sum", oth = NULL)

## jCP.ar1 calls for CP.ar1 to compute the estimate of the conditional probability
## via the adaptive quadrature (MC=F)
## the estimate of the conditional probability

u <- rep(exp(tpar[4]),length(ypre)+length(ynew))

CP1.ar1(ypre = ypre, ynew =ynew, y2m = NULL,
stp =stp, u = u, th = exp(tpar[2]), a = exp(tpar[1]),
dt= ilgt(tpar[3]), mod = dist, qfun = "sum")


## jCP.ar1 calls for CP.ar1 to compute the estimate of the conditional probability
## via the Monte Carlo method (MC=T)
## the estimate of the conditional probability
MCCP.ar1(ypre = ypre, ynew =ynew, stp = stp,
 u = u, th = exp(tpar[2]), a = exp(tpar[1]),  dt = ilgt(tpar[3]),
 mod =dist, Ns = 1000, qfun = "sum")

## End(Not run)
```

`CP.se` *Compute a conditional probability of observing a set of counts as extreme as the new observations of a subject given the previous observations from the same subject based on the negative binomial mixed effect independent model.*

Description

Given the parameter estimates of `c( alpha,theta,beta0,beta1,...)` of the negative binomial mixed effect independent model, these functions compute the following conditional probability:

Pr(q(Y_new) >= q(y_new) | Y_pre = y_pre)

where *y_new* and *y_pre* are vectors of previous and new observations from a same subject and `q()` is a function which provides a scalar summary of the new observations.

These functions are subroutines of `index.batch`.

When input the maximum likelihood estimates of the paramters, `CP.se` returns the estimate of the conditional probability of single subject and the asymptotic standard error of the logit of the estimate of the conditional probability based on the independent model. The computation for of the probability is done by its subroutine `jCP`.

Usage

```
CP.se(tpar, Y1, Y2, sn1, sn2, XM = NULL, dist = "G", V, pty = "sum")
jCP(tpar, Y1, Y2, sn1, sn2, XM = NULL, dist = "G", LG = FALSE, oth = NULL, type = "sum")
```

Arguments

tpar	A vector of length 3 + # covariates, containing the estimates of the model in the order of log(a),log(theta),beta0,beta1... If random effects are assumed to be from the log-normal distribution, then theta is a variance estimate of the random effect. If random effects are assumed to be from the gamma distribution, then theta is a shape parameter estimate. If the semi-parametric approach is taken, then theta is a place holder and can be any number.
Y1	A scalar containing the sum of the previously observed response counts of a subject.
Y2	A scalar containing the summary statistics of the newly observed response counts of a subject *q(y_new)*
sn1	The number of previous observations.
sn2	The number of new observations.
XM	A *n_i* by # covariates matrix containing the covariate values of subject *i*, where whre *n_i* is the total number of previous and new observations. If there is no covariate, i.e., the model only has an intercept term, then set XM=NULL.
dist	If dist=="G", then the conditional probability is computed by assuming the random effect is from the gamma distribution. If dist=="N", then the conditional probability is computed by assuming the random effect is from the log-normal distribution. Note that the semiparametric model is NOT accepted in CP.se. For jCP, if dist="NoN", then the conditional probability is computed by assuming the random effect is from a distribution represented by the argument oth..
V	The variance covariance matrix of the parameter estimates.
pty	The summary statistics for the new scans. *q()*. The current options are "sum" and "max".
oth	The argument only for jCP. If dist= "G" or "N", oth=NULL. If dist="NoN", othr must be the frequency table of the random effects. i.e.,dist=obj$gtb where obj is the output of mle.a3.fun.
LG	If LG=TRUE then the logit of the conditional probability is returned.
type	Same as pty.

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The functions to fit the relevant models:

mle.fun, mle.ar1.fun, mle.a3.fun, mle.ar1.non3,

The other subroutines of index.batch to compute the conditional probability index:

MCCP.ar1, CP.ar1.se, CP.se, jCP,

The functions to generate simulated datasets: rNBME.R.

Examples

```
## Not run:
## tpar contains: log(a),log(theta),beta0
tpar <-  c(0.5, -0.5, 1.3)
## A scalar containing the sum of the response counts in pre scans
Y1 <- 0
## A scalar containing the summary statistics of the response counts in new scans q(y_new)
Y2 <- 1
## The number of scans in the pre scans.
sn1 <- 2
## The number of scans in the new scans.
sn2 <- 3
## the covariate matrix
XM <- NULL
dist <- "G"
## the variance covariance matrix:
V <- matrix(
  c(0.0490673003, -0.0004481864, 0.013279476,
   -0.0004481864,  0.0245814022, 0.001231522,
    0.0132794760,  0.0012315221, 0.023888065),nrow=3)

## the estimate of the conditional probability based on the sum summary statistics and its SE
CP.se(tpar = tpar, Y1=Y1 ,Y2= Y2, sn1 = sn1, sn2 = sn2, XM = XM, dist = dist, V = V, pty = "sum")

## the estimate of the conditional probability based on the max summary statistics and its SE
CP.se(tpar = tpar, Y1=Y1 ,Y2= Y2, sn1 = sn1, sn2 = sn2, XM = XM, dist = dist, V = V, pty = "max")

## jCP calls for CP.se to compute the estimate of the conditional probability
jCP(tpar = tpar, Y1 = Y1, Y2 = Y2, sn1 = sn1, sn2 = sn2,
    XM = XM, dist = dist, LG = FALSE, oth = NULL, type = "sum")

## End(Not run)
```

---

index.batch *Compute the point estimate and its 95 of the conditional probability* Pr(q(Y_i,new)>=q(y_i,new)| Y_i,pre=y_i,pre)

Description

Given an output of mle.fun, mle.ar1.fun, mle.a3.fun, mle.ar1.non3 or lmeNB. This function computes the probability of observing the response counts as large as those new observations of subject *i y_i,new=(y_i,m_i+1,...,y_i,ni)* conditional on the subject's previous observations *y_i,pre(y_i,1,...,y_i,m_i)*.

That is, this function returns a point estimate and its asymptotic 95

Pr(q(Y_i,new)>=q(y_i,new)| Y_i,pre=y_i,pre).

The standard error is not produced when the semi-parametric approach is employed.

A scalar statistic to summarize the new response counts can be either the total count, *q(Y_i,new)=sum_j=m_i+1^ni Y_ij*, or the maximum, *q(Y_i,new)=max { Y_ij;j=m_i+1,...,n_i }* .

See Zhao et al., for more details.

Usage

```
index.batch(data,labelnp, ID, Vcode,olmeNB,subset=NULL,
            qfun = "sum", IPRT = TRUE,  i.se = TRUE, iMC = FALSE)
```

Arguments

data	A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. The each row must contains the data corresponding to the repeated measure j of subjects and the rows *(i,j)*s must be ordered as *(1,1),...,(1,n_1),(2,1),...,(2,n_2),...,(N,n_N)*. This dataset does not have to be the same as the one used in the computations of negative binomial mixed effect regression (mle.fun,mle.ar1.fun,mle.a3.fun,mle.ar1.non3 or lmeNB).
labelnp	A vector of length the total number of repeated measures, indicating new measures by TRUE. For examples, if patient *i* has a *n_i* repeated measures and the last *n_i-m_i+1* measures are new, then then labelnp=c(rep(rep(FALSE,m_1),rep(TRUE,n_1-m_1+1)),rep
ID	A vector of length *sum_i^N n_i* , containing the subject IDs. i.e., c(rep(1,n_1),rep(2,n_2),...,rep(N
Vcode	A vector of length the total number of repeated measures, containing the indices of time point. For example, there are three subjectss and two subjects do not have missing visits and completed five visits while the other subjects missed a visit at the third time and there are four visits in total, then Vcode=c(1,2,3,4,5,1,2,3,4,5,1,2,4,5). Necessary only if the olmeNB is an output of AR(1) models from mle.ar1.fun,mle.ar1.non3.
olmeNB	Output of mle.fun,mle.ar1.fun,mle.a3.fun,mle.ar1.non3 or lmeNB.
subset	An optional expression indicating the subset of the subjects of that the index should be computed.
qfun	If qfun="sum", a scalar statistic to summarize the new response counts is the total count. If qfun="max", a scalar statistic to summarize the new response counts is the maximum.
IPRT	print control.

i.se	If i.se=TRUE then the standard errors of the estimator of the conditional probability are returned for the output of mle.fun or mle.ar1.fun. The semi-parametric approach, mle.a3.fun or mle.ar1.non3, do not return the standard errors.
iMC	Necessary for the AR(1) outputs,mle.ar1.fun and mle.ar1.non3. If iMC=TRUE then the function MCCP.ar1 is called and the Monte carlo integration is performed. Fast but could be unreliable; not recommended for computing the confidence intervals If iMC=FALSE then the function CP1.ar1 is called and the adaptive quadrature is performed. Slow but reliable.

Details

The standard error of the point estimate on the logit scale is constructed using the delta method for the parametric model, where distributional assumption was made for random effects.

Value

The N by 4 (3, if hide the SE) numeric matrix, containing the point estimate of the conditional probability, and the lower and the upper bounds of the 95

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

A wrapper function to fit negative binomial mixed effect model: lmeNB

The functions to fit the relevant models: mle.fun, mle.ar1.fun, mle.a3.fun, mle.ar1.non3,

The subroutines of index.batch: jCP.ar1, CP1.ar1, MCCP.ar1, CP.ar1.se, CP.se, jCP,

The functions to generate simulated datasets: rNBME.R.

Examples

```
## See the examples in linked R-functions.
```

`lmeNB` *Performs the maximum likelihood estimation for the negative binomial mixed-effect model. This function is a wrapper for* `mle.fun`, `mle.ar1.fun`, `mle.a3.fun` *and* `mle.ar1.non3`.

Description

The model assumes that given the random effect $G_i=g_i$, the count responses Y_ijs of subject i, ($i = 1, ..., N$), at time points $j (=1,...,n_i)$ follow the negative binomial distribution:

$Y_ij \mid G_i = g_i \sim NB(r_ij, p_i),$

where $r_ij = exp(X_ij^T beta)/a$ and $beta$ is fixed effect coefficients. The failure probability p_i of subject i, is parametrized as:

$p_i = 1/(g_i*a+1),$

and $a > 0$.

The model assumes $E(G_i) = 1$ so that $E(Y_ij|G_i=g_i)=r_ij*g_i*a$ and $E(Y_ij)= r_ij*g_i = exp(X_ij^T beta)$.

This assumption allows the interpretation of the latent random variable G_i as the subject i's activity rate relative to the overall cohort.

Regarding the dependence structures of Y_ij and Y_ij' conditional on the random effect, we consider two scenarios.

[1]: Y_ij and Y_ij' are independent conditionally on Gi. This assumption leads us to

$cov(Y_ij,Y_ij'|G_i=g_i)=0$ $cov(Y_ij,Y_ij')=E(Y_ij)^2*Var(Y_ij)+E(Y_ij)*(1+(Var(Y_ij)+1)*a)$

[2]: AR(1) structures for Y_ij and Y_ij' conditionally on Gi.

Given emph G_i=g_i, Y_ij depends on $Y_i(j-1)$ through the beta binomial thinning and is conditionally independent on Y_ij' given $Y_i(j-1)$ for all $j' < j-1$.

The beta binomial thinning operator depends on a parameter d which indicates the strength of the positive AR(1) association between repeated measures of a subject: the larger d, the stronger the positive correlations between the repeated measures of the same subject are.

This interpretation depends on the result:

$cov(Y_ij,Y_ij'|G_i=g_i)=d^{j-j'} E(Y_ij')*(a*g_i^2+g_i)$.

Regarding the random effect G_is distribution, `lmNB` allows three scenarios:

(1) The log-normal distribution with $E(G_i)=1$ and $Var(G_i)=theta$

(2) The gamma distribution with $E(G_i)=1$ and $Var(G_i)=theta$.

(3) No distributional assumption and the random effect distribution is approximated by the estimated values of the quantity: $gamma_i = w_i (y_i+/ mu_i+) + (1-w_i) , i=1,...,N$,

where $y_i+ = sum_j=1^n_i y_ij$, $mu_i+ = sum_j=1^n_i mu_ij$ and, $w_i = sqrt(Var(G_i)/Var(Y_i+/mu_i+))$.

See Zhao et al. for more details.

Usage

```
lmeNB(formula,data,ID,p.ini=NULL,IPRT=FALSE,AR=FALSE,RE=c("G","N","semipara"),
      deps=0.001,Vcode,i.tol=1e-75,o.tol=1.e-3,maxit=100)
```

Arguments

formula	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The formula must contain an intercept term.
data	A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. Each row must contain the data corresponding to the repeated measure j of a subject and the rows *(i,j)*s must be ordered as *(1,1),...,(1,n_1),(2,1),...,(2,n_2),...,(N,n_N)*. Missing values are not accepted.
ID	A vector of length *sum_i^N n_i* , containing the patient IDs. i.e., c(rep(ID_1,n_1),rep(ID_2,n_2),... The length must be the same as the number of rows of data. Missing values are NOT accepted.
p.ini	The initial values of the parameters: If AR=0 and (RE="G" or RE="N"),p.ini=*(log(a), log(var(G)), beta0, beta1, ...)*. If AR=0 and RE="semipara", p.ini=*(log(a), log(var(G)),beta0, beta1, ...)*. If AR=1 and (RE="G" or RE="N"),p.ini=*(log(a), log(var(G)),logit(d),beta0,beta1,...)*. If AR=1 and RE="semipara", p.ini= *(log(a), log(var(G)),logit(d), b0, b1, ...)*. NULL is accepted.
IPRT	A logical, passed to Iprt of function optim. If TRUE then print iterations.
AR	A logical, if TRUE, then the AR(1) structure is assumed among the responses.
RE	The distribution of random effects *G_i*. If model="G" then the random effects are assumed to be from the gamma distribution. If *model="N"* then they are assumed to be form the log-normal distribution.
deps	Passed to mle.a3.fun and mle.ar1.non3.
Vcode	A vector of length the total number of repeated measures, containing the indices of time point. For example, there are three subjects and the first two subjects do not have missing visits and completed five visits while the last subject missed the third visit and have four visits in total, then Vcode=c(1,2,3,4,5,1,2,3,4,5,1,2,4,5). Necessary only if the AR(1) model is fit.
i.tol	A real number to determine the tolerance for integrate. Necessary only for semiparametric random effect model.
o.tol	A real number to determine the tolerance for optim. Necessary only for semiparametric random effect model.
maxit	The maximum number of iterations. Necessary only for semiparametric random effect model.

Details

mle.fun calls optim to minimize the negative log-likelihood of the negative binomial model with respect to the model parameters: c(log(*a*), log(*theta*), *beta0*, *beta1*, ...).

The Nelder-Mead algorithm is employed.

The log-likelihood is obtained by marginalizing out the random effects.

The numerical integration is carried out using adaptive quadrature.

The missing count responses, if assumed to be missing at random, can be ignored.

Other types of missing data are currently not accepted.

When the estimated over-dispersion parameter (a) is close to zero, the negative binomial model reduces to the poisson model, suggesting that the negative binomial mixed-effect model might not be appropriate.

When `AR=1` and the estimated auto-correlation parameter (d) is close to zero, the model is suggesting that there is no AR(1) structure among the sequentially collected responses. Hence user might use `AR=0` setting which assume no AR(1) structure.

We note that the results could be sensitive to initial values.

Value

`opt`	The values returned by `optim`.
`nlk`	The value of the negative log-likelihood corresponding to `opt$par`
`V`	The approximated asymptotic covariance matrix of the maximum likelihood estimators. `V=solve(opt$hessian)`
`est`	A (3 + # covariates) by 2 matrix. The first column contains the estimates of the model parameters, log(a), log(*theta*),*beta0*,*beta1*,...) The second column contains the approximated standard deviations of the estimators, i.e., `sqrt(diag(V))`
`mod`	If `model="G"` then `mod="G"`. If `model="N"` then `mod="N"`.
`idat`	A dataframe, containing ID, CEL, x.1, x.2, ... The column labeled as CEL contains the response counts.
`cor`	`"ind"`, indicating that the model assumes independent structure of the count responses given the random effects.

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The subroutines of this function is: `mle.fun`, `mle.ar1.fun`, `mle.a3.fun`, `mle.ar1.non3`,

The subroutines of `index.batch` to compute the conditional probability index: `jCP.ar1`, `CP1.ar1`, `MCCP.ar1`, `CP.ar1.se`, `CP.se`, `jCP`,

The functions to generate simulated datasets: `rNBME.R`.

Examples

```
## See the examples in help files of rNBME.R.
```

`mle.a3.fun` *Fit the semi-parametric negative binomial mixed-effect independent model.*

Description

This function fits the semi-parametric negative binomial mixed-effect independent model to repeated count responses (Zhao et al).

The negative binomial mixed-effect independent model assumes that given the random effect $G_i=g_i$, the count responses Y_ijs of subject i, $(i=1, ..., N)$, at time points $j=1,...,n_i$ are independent andfollow the negative binomial distribution:

$$Y_{ij} \mid G_i = g_i \sim NB(r_{ij}, p_i),$$

where p_i , the failure probability of subject i at each time point j is parametrized as:

$$p_i = 1/(g_i * a + 1),$$

and $a > 0$.

The model assumes $E(G_i) = 1$ so that

$E(Y_{ij} | G_i = g_i) = r_{ij} * g_i * a$ and $E(Y_{ij}) = r_{ij} * g_i$.

This assumption allows the interpretation of the latent random variable G_i as the subject i's activity rate relative to the overall cohort.

The marginal mean $mu_ij = E(Y_ij)$ is modeled with fixed effect coefficients, *beta*: $mu_ij = exp(X_ij^T beta)$.

The distribution of random effect terms are not assumed and approximated by the estimated values of the quantity:

$$gamma_i = w_i \, (y_{i+} / mu_{i+}) + (1 - w_i) \, , \; i = 1, ..., N \, ,$$

where $y_i+ = sum_j=1^n_i \, y_ij$, $mu_i+ = sum_j=1^n_i \, mu_ij$ and, $w_i = sqrt(Var(G_i)/Var(Y_i+/mu_i+))$. See Zhao et al. for more details.

The missing count responses, if assumed to be missing at random, can be igored. Other types of missing data are currently not accepted.

Usage

```
mle.a3.fun(formula,data,ID, p.ini = NULL, IPRT = TRUE, deps = 1e-04,maxit = 100)
```

Arguments

formula	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The formula must contain an intercept term.
data	A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. Each row must contain the data corresponding to the repeated measure j of a subject and the rows *(i,j)*s must be ordered as *(1,1),...,(1,n_1),(2,1),...,(2,n_2),...,(N,n_N)*. Missing values are not accepted.
ID	A vector of length *sum_i^N n_i* , containing the subject IDs. i.e., c(rep(ID_1,n_1),rep(ID_2,n_2),...
p.ini	A vector of length 3 + # covariates, containing the initial values for the parameters *(log(a), log(var(G)), beta0, beta1, ...)*. NULL is accepted.
IPRT	A logical, passed to Iprt of function optim. If TRUE then print iterations.
deps	A small positive real number for the stopping criterion: the algorithm stops iteration when max(p.new - p.old) < deps
maxit	The maximum number of iterations.

Details

The algorithm repeats the following four steps until a stoping criterion is satisfied:

Step 1) Estimate the coefficients of covariates by the method of generalized Least Squares.

Step 2) Approximate the distribution of the random effect *G_i* by *gamma_i*.

Step 3) Estimate *a* by minimizing the negative psudo-profile likelihood. The numerical minimization is carried out using optimize and the numerical integration is carried out using adaptive quadrature.

Step 4) Estimate *var(G)* by the medhod of moment and update the weights.

Value

opt	The values returned by optimize to minimize the negative of the psudo-profile log-likelihood with respect to a.
V	If the number of covariate is nonzero, vcm returns a naive estimate of the variance covariance matrix of b0, b1,...
est	A vector of length # covariates + 2 containing the estimates of the parameters *(log(a), log(var(G)), b0, b1, ...)*
gi	A vector of length *N*, containing the approximation of *g_i*s
gtb	The relative frequency table of *g_i*, (*i=1,...,N*). gh1 contains the unique values in ascending order and ghw contains relative frequencies.
mod	"NoN", denoting that the fitted model is a semi-parametric mixed-effect model.
cor	"ind", indicating that the model assumes independent structure of the count responses given the random effects.
maxit	The maximum number of iterations.

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The functions to fit the other models: mle.fun, mle.ar1.fun,

mle.ar1.non3,

The subroutines of index.batch to compute the conditional probability index: jCP.ar1, CP1.ar1, MCCP.ar1, CP.ar1.se, CP.se, jCP,

The functions to generate simulated datasets: rNBME.R.

Examples

```
## Not run:
## generate a simulated dataset from the negative binomial
## mixed-effect independent model:
## Y_ij | G_i = g_i ~ NB(r_ij,p_i) where r_ij = exp(X^T beta)/a , p_i =1/(a*g_i+1)
## with G_i is from unknown distribution
## For the simulation purpose, G_i's are from the mixture of
$$ the gamma and the log-normal distributions.


sn = 5 ## the number of repeated measures of each subject
n = 80 ## the number of subjects
logtheta = 1.3
th = exp(logtheta) ## the parameter for the gamma distribution of g_i
loga = -0.5
## the parameter for the failure probability of the negative binomial distribution
a = exp(loga)
b0 = 0.5
u1 = rep(exp(b0),sn) ## the mean vector for group 1 at time point 1,...,sn
u2 = rep(exp(b0),sn) ## the mean vector for group 2 at time point 1,...,sn

DT3= rNBME.R(gdist="GN", n=n, a=a, th=th, u1=u1, u2=u2, sn=sn,
    othrp=list(p.mx=0.1,u.n=3,s.n=1,sh.mx = NA) ## 0  < p.mx < 1
    )

ID = DT3$id
dt3=data.frame(CEL=DT3$y)

Vcode=rep(-1:(sn-2),n) # scan number -1, 0, 1, 2, 3
new=Vcode>0         # new scans: 1,2,3


## 1) Fit the negative binomial mixed-effect AR(1) model
```

```
## where random effects is from the gamma distribution

logitd = -0.2
re.gamma.ar1=mle.ar1.fun(formula=CEL~1,data=dt3,ID=ID,
      Vcode=Vcode,
      p.ini=c(loga,logtheta,logitd,b0),
      ## log(a), log(theta), logit(d), b0
       model="G",
      IPRT=TRUE)


## compute the estimates of the conditional probabilities
## with sum of the new repeated measure as a summary statistics
Psum=index.batch(olmeNB=re.gamma.ar1,data=dt3,ID=ID, Vcode=Vcode,
                 labelnp=new,qfun="sum", IPRT=TRUE,i.se=FALSE)



## 2) Fit the negative binomial mixed-effect AR(1) model
## where random effects is from the log-normal distribution


re.logn.ar1=mle.ar1.fun(formula=CEL~1,data=dt3,ID=ID,
      Vcode=Vcode, model="N", IPRT=TRUE)

## REQUIRES SOME TIME..
Psum=index.batch(olmeNB=re.logn.ar1, data=dt3,ID=ID,Vcode=Vcode,
         labelnp=new,qfun="sum", IPRT=TRUE)#,i.se=FALSE)



## 3) Fit the negative binomial independent model
## where random effects is from the gamma distribution
re.gamma.ind=mle.fun(formula=CEL~1,data=dt3,ID=ID,
                  model="G",
   IPRT=TRUE)

Psum=index.batch(olmeNB=re.gamma.ind, data=dt3,ID=ID,Vcode=Vcode,
                 labelnp=new,qfun="sum", IPRT=TRUE,i.se=TRUE)



## 4) Fit the negative binomial independent model
## where random effects is from the lognormal distribution
re.logn.ind=mle.fun(formula=CEL~1,data=dt3,ID=ID,
                  model="N",
   p.ini=c(loga,logtheta,b0),
   IPRT=TRUE)

Psum=index.batch(olmeNB=re.logn.ind,data=dt3,ID=ID,labelnp=new,qfun="sum", IPRT=TRUE)

## 5) Fit the semi-parametric negative binomial AR(1) model
```

```
logvarG = -0.4

re.semi.ar1=mle.ar1.non3(formula=CEL~1,data=dt3,ID=ID,Vcode=Vcode)
Psum=index.batch(olmeNB=re.semi.ar1,data=dt3,ID=ID,Vcode=Vcode,
                 labelnp=new,qfun="sum", IPRT=TRUE,iMC=TRUE,i.se=FALSE)


## 6) Fit the semi-parametric negative binomial independent model
## This is closest to the true model
re.semi.ind=mle.a3.fun(formula=CEL~1,data=dt3,ID=ID, p.ini=c(loga, logvarG, b0))

## compute the estimates of the conditional probabilities
## with sum of the new repeated measure as a summary statistics
Psum=index.batch(olmeNB=re.semi.ind,data=dt3,ID=ID, labelnp=new,
                 qfun="sum", IPRT=TRUE,i.se=FALSE)
## compute the estimates of the conditional probabilities
## with max of the new repeated measure as a summary statistics
Pmax=index.batch(olmeNB=re.semi.ind, data=dt3,ID=ID,labelnp=new, qfun="max",
                 IPRT=TRUE,i.se=FALSE)


## Which patient's estimated probabilities based on the sum and max
## statistics disagrees the most?
( IDBigDif <- which(rank(abs(Pmax$condProbSummary[,1]-Psum$condProbSummary[,1]))==80) )
## Show the patient's CEL counts
dt3$CEL[ID==IDBigDif]
## Show the estimated conditional probabilities based on the sum summary statistics
Psum$condProbSummary[IDBigDif,]
## Show the estimated conditional probabilities based on the max summary statistics
Pmax$condProbSummary[IDBigDif,]


## ========================================================================== ##
## == Which model performed the best in terms of the estimation of beta0 == ##
## ========================================================================== ##

getpoints <- function(y,estb0,sdb0=NULL,crit=qnorm(0.975))
{
points(estb0,y,col="blue",pch=16)
if (!is.null(sdb0))
{
points(c(estb0-crit*sdb0,estb0+crit*sdb0),rep(y,2),col="red",type="l")
}
}
ordermethod <- c("gamma.ar1","logn.ar1","gamma.ind","logn.ind","semi.ar1","semi.ind")

estb0s <- c(
re.gamma.ar1$est[4,1],
re.logn.ar1$est[4,1],
re.gamma.ind$est[3,1],
re.logn.ind$est[3,1],
re.semi.ar1$est[4],
re.semi.ind$est[3]
```

```
)

## The true beta0 is:
b0
c <- 1.1
plot(0,0,type="n",xlim=c(min(estb0s)-0.5,max(estb0s)*c),ylim=c(0,7),yaxt="n",
main="Simulated from the independent model \n with random effect ~ mixture of normal and gamma")

legend("topright",
legend=ordermethod)
abline(v=b0,lty=3)

## 1) gamma.ar1
sdb0 <- re.gamma.ar1$est[4,2]
getpoints(6,estb0s[1],sdb0)

## 2)logn.ar1
sdb0 <- re.logn.ar1$est[4,2]
getpoints(5,estb0s[2],sdb0)

## 3) gamma.ind
sdb0 <- re.gamma.ind$est[3,2]
getpoints(4,estb0s[3],sdb0)

## 4) logn.ind
sdb0 <- re.logn.ind$est[3,2]
getpoints(3,estb0s[4],sdb0)

## 5) semi.ar1
getpoints(2,estb0s[5])

## 6) semi.ind
getpoints(1,estb0s[6])

## End(Not run)
```

`mle.ar1.fun` *Performs the maximum likelihood estimation for the negative binomial mixed-effect AR(1) model*

Description

This function fits a negative binomial mixed-effect AR(1) model in the formulation described Zhao et al.

Under this model, the count response measure *Y_ij* from a subject *i* at time point *j* (*i=1,...,N,j=1,...,n_i*) is from the negative binomial distribution:

$Y_{ij} \mid G_i = g_i \sim NB(r_{ij}, p_i)$,

and the parametrization of r_{ij}, p_i and the distributional assumption on G_i are the same as the independence model (`mle.fun`).

Given emph G_i=g_i, *Y_ij* depends on *Y_i(j-1)* through the beta binomial thinning and is conditionally independent on *Y_ij'* given *Y_i(j-1)* for all *j' < j-1* .

The beta binomial thinning operator depends on a parameter *d* which indicates the strength of the positive AR(1) association between repeated measures of a subject: the larger *d*, the stronger the positive correlations between the repeated measures of the same subject are.

This interpretation depends on the result:

cov(Y_ij,Y_ij'|G_i=g_i)=d^j-j' E(Y_ij')(a*g_i^2+g_i).*

See Zhao et al., for more details.

Usage

```
mle.ar1.fun(formula, data, ID, Vcode,
            p.ini=NULL, IPRT = FALSE, model = "G",
            i.tol = 1e-75, o.tol = 0.001)
```

Arguments

`formula`	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The formula must contain an intercept term.
`data`	A data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. Each row must contain the data corresponding to the repeated measure j of subject and the rows *(i,j)*s must be ordered as *(1,1),...,(1,n_1),(2,1),...,(2,n_2),...,(N,n_N)*. Missing data are currently not accepted.
`ID`	A vector of length *sum_i^N n_i* , containing the subject IDs. i.e., `c(rep(ID_1,n_1),rep(ID_2,n_2),...`
`Vcode`	A vector of length the total number of repeated measures, containing the indices of time point. For example, there are three subjects and the first two subjects do not have missing visits and completed five visits while the last subject missed the third visit and have four visits in total, then `Vcode=c(1,2,3,4,5,1,2,3,4,5,1,2,4,5)`
`p.ini`	Initial values for the parameters (log(*a*), log(*theta*),*logit(d) beta0*, *beta1*, ...)
`IPRT`	A logical, passed to Iprt of function `optim`. If `TRUE` then print iterations.
`model`	The distribution of random effects *G_i*. If `model="G"` then the random effects are assumed to be from the gamma distribution. If *model="N"* then they are assumed to be form the log-normal distribution.
`i.tol`	A real number to determine the tolerance for `integrate`.
`o.tol`	A real number to determine the tolerance for `optim`.

Details

`mle.ar1.fun` calls `optim` to minimize the negative log-likelihood of the negative binomial model with respect to the model parameters: c(log(*a*), log(*theta*), logit(*d*), *beta0*, *beta1*, ...).

The Nelder-Mead algorithm is employed.

The log-likelihood is obtained by marginalizing out the random effects.

The numerical integration is carried out using adaptive quadrature.

When missing visits are present, an approximation of the likelihood is used (see Web Appendix II in Zhao et al for details.)

Value

`opt`	The values returned by `optim`.
`nlk`	The value of the negative log-likelihood corresponding to `opt$par`
`V`	The approximated asymptotic covariance matrix of the maximum likelihood estimators. `V=solve(opt$hessian)`
`est`	The (4 + # covariates) by 2 matrix. The first column contains the estimates of the fixed effects, (log(*a*), log(*theta*),logit(*d*), *beta0*,*beta1*,...) The second column contains the approximated standard deviations of the estimators, i.e., `sqrt(diag(V))`
`mod`	If `model="G"` then `mod="G"`. If `model="N"` then `mod="N"`.
`Vcode`	A vector containing the input indices of time point.
`cor`	`"ar1"`, indicating that the model assumes AR1 correlation structure of the count responses given the random effects.

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The functions to fit the other models: `mle.fun`,

`mle.a3.fun`, `mle.ar1.non3`,

The subroutines of `index.batch` to compute the conditional probability index: `jCP.ar1`, `CP1.ar1`, `MCCP.ar1`, `CP.ar1.se`, `CP.se`, `jCP`,

The functions to generate simulated datasets: `rNBME.R`.

Examples

```
## Not run:

## ============================================================================
## generate a data based on the negative binomial mixed-effect AR(1) model.
## Under this model, the response counts follows the negative binomial:
## Y_ij | G_i = g_i ~ NB(r_ij,p_i) where r_ij = exp(X^T beta)/a , p_i =1/(a*g_i+1)
## with G_i ~ Gamma(scale=th,shape=1/th)
##
## The adjacent repeated measures of the same subject are correlated
```

```
## with correlation structure:
## cov(Y_ij,Y_ij'|G_i=g_i)=d^{j-j'} E(Y_ij')*(a*g_i^2+g_i)

# log(a) = -0.5, log(th)=1.3, logit(delta) = -0.2
# b0 =  0.5, no covariates;
loga = -0.5
logtheta= 1.3
logitd = -0.2
b0 = 0.5
# 80 subjects each with 5 scans
n = 80
sn = 5


DT2 =  rNBME.R(gdist = "G",
               n = n, ##  the total number of subjectss
       sn = sn,
       th=exp(logtheta),
               u1 = rep(exp(b0),sn),
       u2 = rep(exp(b0),sn),
       a = exp(loga),
       d = exp(logitd)/(1+exp(logitd))
      )
Vcode=rep(-1:(sn-2),n) # scan number -1, 0, 1, 2, 3
ID = DT2$id
new = Vcode > 0
dt2=data.frame(CEL=DT2$y)

## ============================================================================

## 1) Fit the negative binomial mixed-effect AR(1) model
## where the random effects are from the gamma distribution
## This is the true model

re.gamma.ar1=mle.ar1.fun(formula=CEL~1,data=dt2,ID=ID,
      Vcode=Vcode,
      p.ini=c(loga,logtheta,logitd,b0),
      ## log(a), log(theta), logit(d), b0
       model="G",
      IPRT=TRUE)



## compute the estimates of the conditional probabilities
## with sum of the new repeated measure as a summary statistics
Psum=index.batch(olmeNB=re.gamma.ar1,data=dt2,ID=ID,Vcode=Vcode,
 labelnp=new,qfun="sum", IPRT=TRUE,i.se=FALSE)


## compute the estimates of the conditional probabilities
## with max of the new repeated measure as a summary statistics
Pmax=index.batch(olmeNB=re.gamma.ar1,data=dt2,ID=ID,Vcode=Vcode,
                 labelnp=new,qfun="max", IPRT=TRUE)
```

```
## Which patient's estimated probabilities based on the sum and max
## statistics disagrees the most?
( IDBigDif <- which(rank(abs(Pmax$condProbSummary[,1]-Psum$condProbSummary[,1]))==80) )
## Show the patient's CEL counts
dt2$CEL[ID==IDBigDif]
## Show the estimated conditional probabilities based on the sum summary statistics
Psum$condProbSummary[IDBigDif,]
## Show the estimated conditional probabilities based on the max summary statistics
Pmax$condProbSummary[IDBigDif,]


## 2) Fit the negative binomial mixed-effect AR(1) model
## where random effects is from the log-normal distribution

re.logn.ar1=mle.ar1.fun(formula=CEL~1,data=dt2,ID=ID,
      Vcode=Vcode,
      p.ini=c(loga,logtheta,logitd,b0),
      ## log(a), log(theta), logit(d), b0
       model="N",
      IPRT=TRUE)

## Require some time
Psum=index.batch(olmeNB=re.logn.ar1,data=dt2,ID=ID,Vcode=Vcode,
                 labelnp=new,qfun="sum", IPRT=TRUE,i.se=TRUE)
re.logn.ar1$Psum=Psum




## 3) Fit the negative binomial independent model
## where random effects are from the gamma distribution
re.gamma.ind=mle.fun(formula=CEL~1,data=dt2,ID=ID,
                   model="G",
   p.ini=c(loga,logtheta,b0),
   IPRT=TRUE)

Psum=index.batch(olmeNB=re.gamma.ind,data=dt2,ID=ID,
                 labelnp=new,qfun="sum", IPRT=TRUE,i.se=TRUE)




## 4) Fit the negative binomial independent model
## where random effects are from the lognormal distribution
re.logn.ind=mle.fun(formula=CEL~1,data=dt2,ID=ID,
                   model="N",
   p.ini=c(loga,logtheta,b0),
   IPRT=TRUE)

Psum=index.batch(olmeNB=re.logn.ind, data=dt2,ID=ID,
                 labelnp=new,qfun="sum", IPRT=TRUE,i.se=TRUE)
```

```
## 5) Fit the semi-parametric negative binomial AR(1) model

logvarG = -0.5

re.semi.ar1=mle.ar1.non3(formula=CEL~1,data=dt2,ID=ID,
                         p.ini=c(loga, logvarG, logitd,b0),Vcode=Vcode)

Psum=index.batch(olmeNB=re.semi.ar1,data=dt2,ID=ID, Vcode=Vcode,
 labelnp=new,qfun="sum", IPRT=TRUE,i.se=FALSE)
## No option of i.se=TRUE



## 6) Fit the semi-parametric negative binomial independent model
re.semi.ind=mle.a3.fun(formula=CEL~1,data=dt2,ID=ID, p.ini=c(loga, logvarG, b0))
Psum=index.batch(olmeNB=re.semi.ind,data=dt2,ID=ID,
                 labelnp=new, qfun="sum", IPRT=TRUE,i.se=FALSE)
## No option of i.se=TRUE



## ========================================================================== ##
## == Which model performed the best in terms of the estimation of beta0 == ##
## ========================================================================== ##

getpoints <- function(y,estb0,sdb0=NULL,crit=qnorm(0.975))
{
points(estb0,y,col="blue",pch=16)
if (!is.null(sdb0))
{
points(c(estb0-crit*sdb0,estb0+crit*sdb0),rep(y,2),col="red",type="l")
}
}
ordermethod <- c("gamma.ar1","logn.ar1","gamma.ind","logn.ind","semi.ar1","semi.ind")

estb0s <- c(
re.gamma.ar1$est[4,1],
re.logn.ar1$est[4,1],
re.gamma.ind$est[3,1],
re.logn.ind$est[3,1],
re.semi.ar1$est[4],
re.semi.ind$est[3]
)

## The true beta0 is:
b0
c <- 1.1
plot(0,0,type="n",xlim=c(min(estb0s)-0.5,max(estb0s)*c),ylim=c(0,7),yaxt="n",
main="Simulated from the AR(1) model \n with random effect ~ gamma")

legend("topright",
legend=ordermethod)
```

```
abline(v=b0,lty=3)

## 1) gamma.ar1
sdb0 <- re.gamma.ar1$est[4,2]
getpoints(6,estb0s[1],sdb0)

## 2)logn.ar1
sdb0 <- re.logn.ar1$est[4,2]
getpoints(5,estb0s[2],sdb0)

## 3) gamma.ind
sdb0 <- re.gamma.ind$est[3,2]
getpoints(4,estb0s[3],sdb0)

## 4) logn.ind
sdb0 <- re.logn.ind$est[3,2]
getpoints(3,estb0s[4],sdb0)

## 5) semi.ar1
getpoints(2,estb0s[5])

## 6) semi.ind
getpoints(1,estb0s[6])



## End(Not run)
```

`mle.ar1.non3` *Fit the semi-parametric negative binomial mixed-effect AR(1) model.*

Description

This function fits the semi-parametric negative binomial mixed-effect AR(1) model in the formulation described Zhao et al.

Under this model, the count response of subjects *i* at time point *j* $Y\_ij\ (i=1,...,N, j=1,...,n\_i)$ is from the negative binomial distribution:

$Y\_ij \mid G\_i = g\_i \sim NB(r\_ij, p\_i),$

and the parametrization of r_ij, p_i and the distributional assumption on G_i are the same as the independence model (`mle.fun`).

Given emph G_i=g_i, Y_ij depends on $Y\_i(j-1)$ through the beta binomial thinning and conditionally independent on Y_ij' given $Y\_i(j-1)$ for all $j' < j-1$.

The beta binomial thinning operator depends on a parameter *d* which indicate the strength of the positive AR(1) dependence between repeated measures of a subjects: the larger *d*, the stronger the positive correlations between the repeated measures of the same subjects are.

This interpretation depends on the result:

cov(Y_ij,Y_ij'|G_i=g_i)=d^j-j' E(Y_ij')(a*g_i^2+g_i).*

The distribution of random effect terms are not assumed and approximated by the estimated values of the quantity:

gamma_i = w_i (y_i+/ mu_i+) + (1-w_i) , i=1,...,N ,

where *y_i+ = sum_j=1^n_i y_ij ,*

mu_i+ = sum_j=1^n_i mu_ij with *mu_ij=E(Y_ij)* and, *w_i = sqrt(Var(G_i)/Var(Y_i+/mu_i+)) .* See Zhao et al. for more details.

Usage

```
mle.ar1.non3(formula, data, ID, Vcode,p.ini = NULL, IPRT = TRUE, deps = 0.001, maxit=100)
```

Arguments

`formula`	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The formula must contain an intercept term.
`data`	A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. The each row must contains the data corresponding to the repeated measure j of subjects and the rows *(i,j)*s must be ordered as *(1,1),...,(1,n_1),(2,1),...,(2,n_2),...,(N,n_N)*
`ID`	A vector of length *sum_i^N n_i* , containing the subject IDs. i.e., `c(rep(1,n_1),rep(2,n_2),...,rep(N`
`Vcode`	A vector of length the total number of repeated measures, containing the indices of time point. For example, there are three subjectss and two subjects do not have missing visits and completed five visits while the other subjects missed a visit at the third time and there are four visits in total, then `Vcode=c(1,2,3,4,5,1,2,3,4,5,1,2,4,5)`
`p.ini`	A vector of length 3 + # covariates, containing the initial values for the parameters *(log(a), log(var(G)),logit(d), b0, b1, ...)* `NULL` is accepted.
`IPRT`	A logical. If `TRUE` then print iterations.
`deps`	A small positive real number for the stopping criterion: the algorithm stops iteration when `max(p.new - p.old) < deps`
`maxit`	The maximum number of iterations in optimization.

Details

The algorithm repeats the following four steps until a stoping criterion is satisfied:

Step 1) Estimate the coefficients of covariates by the method of generalized least squares.

Step 2) Approximate the distribution of the random effect *G_i* by *gamma_i*.

Step 3) Estimate *a* and *d* using the psudo-profile likelihood. This step calls `optim` to minimize the negative psudo log-likelihood with respect to log(*a*) and logit(*d*). The numerical integration is carried out using adaptive quadrature. When missing visits are present, an approximation of the likelihood is used (see Web Appendix II in Zhao et al for details.)

Step 4) Estimate *var(G)* by the medhod of moment and update the weights.

Value

opt	The values returned by optim to minimize the negative of the psudo-profile log-likelihood with respect to log(a) and logit(d).
V	If the number of covariate is nonzero, vcm returns an naive estimate of the variance covariance matrix of b0, b1,...
est	A vector of length # covariates + 4 containing the estimates of the parameters *(log(a), log(var(G)), logit(d), b0, b1, ...)*
gi	A vector of length *N*, containing the approximation of *g_i*s
mod	"NoN", denoting that the fitted model is a semi-parametric mixed-effect model.
Vcode	A vector containing the input indices of time point.
cor	"ar1", indicating that the model assumes AR1 correlation structure of the count responses given the random effects.

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The functions to fit the other models: mle.fun, mle.ar1.fun, mle.a3.fun,

The subroutines of index.batch to compute the conditional probability index: jCP.ar1, CP1.ar1, MCCP.ar1, CP.ar1.se, CP.se, jCP,

The functions to generate simulated datasets: rNBME.R.

Examples

```
## Not run:
## ============================================================================== ##
## generate a data based on the semi-parametric negative binomial
## mixed-effect AR(1) model.
## Under this model, the response counts follows the negative binomial:
## Y_ij | G_i = g_i ~ NB(r_ij,p_i) where r_ij = exp(X^T beta)/a , p_i =1/(a*g_i+1)
## G_i is from unknown distribution.
## For simulation purpose, we generate the sample of gi from
## the mixture of three gamma distribuions.

## The adjacent repeated measures of the same subjects are correlated
## with correlation structure:
## cov(Y_ij,Y_ij'|G_i=g_i)=d^{j-j'} E(Y_ij')*(a*g_i^2+g_i)

# log(a) = -0.5, log(th)=1.3, logit(delta) = -0.2
```

```
# b0 =  0.5, no covariates;
loga = -0.5
logtheta = 1.3
logitd = -0.2
b0 = 0.5
# 80 subjects each with 5 scans
n = 80
sn = 5

## generate a sample of size B from the mixture of three gamma distribution:
p1 <- 0.5
p2 <- 0.3
B <- 1000
sampledG<- c(
rgamma(n=p1*B,scale=1,shape=10),
rgamma(n=p2*B,scale=3,shape=5),
rgamma(n=(1-p1-p2)*B,scale=5,shape=5)
)


## mean is set to 1;
sampledG <- sampledG/mean(sampledG)
logvarG <- log(var(sampledG))
## hist(sampledG)

DT4 =  rNBME.R(gdist = "NoN",
                n = n, ##  the total number of subjectss
       sn = sn,
                u1 = rep(exp(b0),sn),
       u2 = rep(exp(b0),sn),
       a = exp(loga),
       d = exp(logitd)/(1+exp(logitd)),
       othrp = sampledG
      )
Vcode=rep(-1:(sn-2),n) # scan number -1, 0, 1, 2, 3
ID = DT4$id
new = Vcode > 0
dt4=data.frame(CEL=DT4$y)
## ============================================================================== ##

## [1] Fit the negative binomial mixed-effect AR(1) model
## where random effects is from the gamma distribution


re.gamma.ar1=mle.ar1.fun(formula=CEL~1,data=dt4,ID=ID,
      Vcode=Vcode,
      p.ini=c(loga,logtheta,logitd,b0),
      ## log(a), log(theta), logit(d), b0
       model="G",
      IPRT=TRUE)

Psum=index.batch(olmeNB=re.gamma.ar1, data=dt4,ID=ID,Vcode=Vcode,
         labelnp=new,qfun="sum", IPRT=TRUE,i.se=FALSE)
```

```
## [2] Fit the negative binomial mixed-effect AR(1) model
## where random effects is from the log-normal distribution


re.logn.ar1=mle.ar1.fun(formula=CEL~1,data=dt4,ID=ID,
      Vcode=Vcode,
      p.ini=c(loga,logtheta,logitd,b0),
      ## log(a), log(theta), logit(d), b0
       model="N",
      IPRT=TRUE)
\dontrun{
## Requires some time
Psum=index.batch(olmeNB=re.logn.ar1,data=dt4,ID=ID,Vcode=Vcode,
         labelnp=new,qfun="sum", IPRT=TRUE)
}


## [3] Fit the negative binomial independent model
## where random effects is from the lognormal distribution
re.logn.ind=mle.fun(formula=CEL~1,data=dt4,ID=ID,
                    model="N",
   p.ini=c(loga,logtheta,b0),
   IPRT=TRUE)

Psum=index.batch(olmeNB=re.logn.ind,data=dt4,ID=ID,
                 labelnp=new,qfun="sum", IPRT=TRUE)


## [4] Fit the semi-parametric negative binomial AR(1) model
## This model is closest to the true model

logvarG = log(var(sampledG))

re.semi.ar1=mle.ar1.non3(formula=CEL~1,data=dt4,ID=ID,
                         p.ini=c(loga, logvarG, logitd,b0),Vcode=Vcode)

## compute the estimates of the conditional probabilities
## with sum of the new repeated measure as a summary statistics
Psum=index.batch(olmeNB=re.semi.ar1, labelnp=new,data=dt4,ID=ID,Vcode=Vcode,
                 qfun="sum", IPRT=TRUE,i.se=TRUE)

## compute the estimates of the conditional probabilities
## with max of the new repeated measure as a summary statistics
Pmax=index.batch(olmeNB=re.semi.ar1, labelnp=new,qfun="max",data=dt4,ID=ID,Vcode=Vcode,
                 IPRT=TRUE,i.se=TRUE)

## Which patient's estimated probabilities
## based on the sum and max statistics disagrees the most?
( IDBigDif <- which(rank(abs(Pmax$condProbSummary[,1]-Psum$condProbSummary[,1]))==80) )
## Show the patient's CEL counts
```

```
dt4$CEL[ID==IDBigDif]
## Show the estimated conditional probabilities based on the sum summary statistics
Psum$condProbSummary[IDBigDif,1]
## Show the estimated conditional probabilities based on the max summary statistics
Pmax$condProbSummary[IDBigDif,1]


## [5] Fit the semi-parametric negative binomial independent model


re.semi.ind=mle.a3.fun(formula=CEL~1,data=dt4,ID=ID, p.ini=c(loga, logvarG, b0))
Psum=index.batch(olmeNB=re.semi.ind, labelnp=new,
                 data=dt4,ID=ID, qfun="sum", IPRT=TRUE,i.se=TRUE)



## ========================================================================== ##
## == Which model performed the best in terms of the estimation of beta0 == ##
## ========================================================================== ##

getpoints <- function(y,estb0,sdb0=NULL,crit=qnorm(0.975))
{
points(estb0,y,col="blue",pch=16)
if (!is.null(sdb0))
{
points(c(estb0-crit*sdb0,estb0+crit*sdb0),rep(y,2),col="red",type="l")
}
}
ordermethod <- c("gamma.ar1","logn.ar1","logn.ind","semi.ar1","semi.ind")

estb0s <- c(
re.gamma.ar1$est[4,1],
re.logn.ar1$est[4,1],
re.logn.ind$est[3,1],
re.semi.ar1$est[4],
re.semi.ind$est[3]
)

## The true beta0 is:
b0
c <- 1.1
plot(0,0,type="n",xlim=c(min(estb0s)-0.5,max(estb0s)*c),ylim=c(0,7),yaxt="n",
main="Simulated from the AR(1) model \n with random effect ~ a semi-parametric distribution")

legend("topright",
legend=ordermethod)
abline(v=b0,lty=3)

## [1] gamma.ar1
sdb0 <- re.gamma.ar1$est[4,2]
getpoints(6,estb0s[1],sdb0)

## [2] logn.ar1
```

```
sdb0 <- re.logn.ar1$est[4,2]
getpoints(5,estb0s[2],sdb0)

## [3] logn.ind
sdb0 <- re.logn.ind$est[3,2]
getpoints(4,estb0s[3],sdb0)

## [4] semi.ar1
getpoints(3,estb0s[4])

## [5] semi.ind
getpoints(2,estb0s[5])


## End(Not run)
```

`mle.fun` *Performs the maximum likelihood estimation for the negative binomial mixed-effect independent model*

Description

This function fits a negative binomial mixed-effect independent model to repeated count measures (Zhao et al).

The model assumes that given the random effect $G\_i=g\_i$, the count responses Y_ijs of subject i, ($i = 1, ..., N$), at time points $j =1,...,n\_i)$ are independent and follow the negative binomial distribution:

$Y\_ij \mid G\_i =g\_i \sim NB(r\_ij,p\_i),$

where p_i , the failure probability of subject i at every time point j is parametrized as:

$p\_i= 1/(g\_i*a+1),$

and $a > 0$.

The model assumes $E(G\_i) = 1$

so that $E(Y\_ij|G\_i=g\_i)=r\_ij*g\_i*a$ and $E(Y\_ij)= r\_ij*g\_i$.

This assumption allows the interpretation of the latent random variable G_i as the subject i's activity rate relative to the overall cohort.

The random effect terms G_is are assumed to be either from the log-normal distribution with $E(G\_i)=1$ and $var(G\_i)=theta$ or the gamma distribution with `scale` = $theta$ and `shape`=1/$theta$.

The marginal mean $mu\_ij=E(Y\_ij)$ is modeled with fixed effect coefficients, $beta$: $mu\_ij= exp(X\_ij\^T\ beta)$.

Usage

```
mle.fun(formula, data, ID, p.ini=NULL, IPRT = FALSE,
        model = "G",i.tol=1e-75,o.tol=1.e-3,COV=TRUE)
```

Arguments

formula	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The formula must contain an intercept term.
data	A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. Each row must contain the data corresponding to the repeated measure j of a subject and the rows *(i,j)*s must be ordered as *(1,1),...,(1,n_1),(2,1),...,(2,n_2),...,(N,n_N)*. Missing values are not accepted.
ID	A vector of length *sum_i^N n_i* , containing the patient IDs. i.e., c(rep(ID_1,n_1),rep(ID_2,n_2),...
p.ini	The initial values of the parameters (log(*a*), log(*theta*), *beta0*, *beta1*, ...). NULL is accepted.
IPRT	A logical, passed to Iprt of function optim. If TRUE then print iterations.
model	The distribution of random effects *G_i*. If model="G" then the random effects are assumed to be from the gamma distribution. If *model="N"* then they are assumed to be form the log-normal distribution.
i.tol	A real number to determine the absolute tolerance for integrate.
o.tol	A real number to determine the relative tolerance for optim.
COV	Internal use only

Details

mle.fun calls optim to minimize the negative log-likelihood of the negative binomial model with respect to the model parameters: c(log(*a*), log(*theta*), *beta0*, *beta1*, ...).

The Nelder-Mead algorithm is employed.

The log-likelihood is obtained by marginalizing out the random effects.

The numerical integration is carried out using adaptive quadrature.

The missing count responses, if assumed to be missing at random, can be igored.Other types of missing data are currently not accepted.

Value

opt	The values returned by optim.
nlk	The value of the negative log-likelihood corresponding to opt$par
V	The approximated asymptotic covariance matrix of the maximum likelihood estimators. V=solve(opt$hessian)
est	A (3 + # covariates) by 2 matrix. The first column contains the estimates of the model parameters, log(*a*), log(*theta*),*beta0*,*beta1*,...) The second column contains the approximated standard deviations of the estimators, i.e., sqrt(diag(V))
mod	If model="G" then mod="G". If model="N" then mod="N".
cor	"ind", indicating that the model assumes independent structure of the count responses given the random effects.

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The wrapper function for all the negative binomial mixed effect regression: lmeNB.

The functions to fit the other negative binomial mixed effect models:

mle.ar1.fun, mle.a3.fun, mle.ar1.non3,

The subroutines of index.batch to compute the conditional probability index: jCP.ar1, CP1.ar1, MCCP.ar1, CP.ar1.se, CP.se, jCP,

The functions to generate simulated datasets: rNBME.R.

Examples

```
## Not run:

## ========================================================================== ##
## generate a simulated dataset from the negative binomial mixed-effect independent model:
## Y_ij | G_i = g_i ~ NB(r_ij,p_i) where r_ij = exp(X^T beta)/a , p_i =1/(a*g_i+1)
## with G_i ~ Gamma(scale=th,shape=1/th)
set.seed(1)
sn = 5 ## the number of repeated measures of each patient
n = 80 ## the number of patients
loga = - 0.5
a = exp(loga) ## the parameter for the failure probability of the negative binomial distribution
logtheta <- 1.3
th = exp(logtheta) ## the parameter for the gamma distribution of g_i

## No difference between the means of groups
## The model only has an intercept term beta0 = 0.5
b0 = 0.5
u1 = rep(exp(b0),sn) ## the mean vector for group 1 at time point 1,...,sn
u2 = rep(exp(b0),sn) ## the mean vector for group 2 at time point 1,...,sn


DT0=rNBME.R(gdist="G", n=n, a=a, th=th, u1=u1, u2=u2, sn=sn)
ID = DT0$id
Vcode=rep(-1:(sn-2),n) # scan number -1, 0, 1, 2, 3
new = Vcode > 0
dt1=data.frame(CEL=DT0$y)
logitd = -0.5

## ========================================================================== ##
```

```
## [1]: Fit the negative binomial independent model
## where the random effects are from the gamma distribution. This is the true model.
## This is the true model

re.gamma.ind=mle.fun(formula=CEL~1,data=dt1,ID=ID,model="G",
             p.ini=c(loga,logtheta,b0),IPRT=TRUE)
## compute the estimates of the conditional probabilities
## with sum of the new repeated measure as a summary statistics
Psum=index.batch(olmeNB=re.gamma.ind, ID=ID,data=dt1,
                 labelnp=new,qfun="sum", IPRT=TRUE)

## compute the estimates of the conditional probabilities
## with max of the new repeated measure as a summary statistics
Pmax=index.batch(olmeNB=re.gamma.ind, ID=ID,data=dt1,
                 labelnp=new,qfun="max", IPRT=TRUE)

## Which patient's estimated probabilities
## based on the sum and max statistics disagrees the most?
( IDBigDif <- which(rank(abs(Pmax$condProbSummary[,1]-Psum$condProbSummary[,1]))==80) )
## Show the patient's CEL counts
dt1$CEL[ID==IDBigDif]
## Show the estimated conditional probabilities based on the sum summary statistics
Psum$condProbSummary[IDBigDif,]
## Show the estimated conditional probabilities based on the max summary statistics
Pmax$condProbSummary[IDBigDif,]



## [2]: Fit the negative binomial independent model
## where the random effects are from the lognormal distribution.
re.logn.ind=mle.fun(formula=CEL~1,data=dt1,ID=ID,
                    model="N",
   p.ini=c(loga,logtheta,b0),
   IPRT=TRUE)

Psum=index.batch(olmeNB=re.logn.ind, ID=ID,data=dt1,
                 labelnp=new,qfun="sum", IPRT=TRUE)



## [3]: Fit the semi-parametric negative binomial independent model


re.semi.ind=mle.a3.fun(formula=CEL~1,data=dt1,ID=ID)

Psum=index.batch(olmeNB=re.semi.ind,ID=ID,data=dt1,
                 labelnp=new, qfun="sum", IPRT=TRUE)



## [4]: Fit the negative binomial mixed-effect AR(1) model
```

```
## where random effects are from the gamma distribution


re.gamma.ar1=mle.ar1.fun(formula=CEL~1,data=dt1,ID=ID,
      p.ini=c(loga,logtheta,logitd,b0),
      ## log(a), log(theta), logit(d), b0
       model="G", Vcode=Vcode,
      IPRT=TRUE)

Psum=index.batch(olmeNB=re.gamma.ar1, ID=ID,data=dt1, labelnp=new,Vcode=Vcode,
                 qfun="sum", IPRT=TRUE,i.se=FALSE) ## i.se=T requires more time...




## ========================================================================== ##
## == Which model performed the best in terms of the estimation of beta0 == ##
## ========================================================================== ##

getpoints <- function(y,estb0,sdb0=NULL,crit=qnorm(0.975))
{
points(estb0,y,col="blue",pch=16)
if (!is.null(sdb0))
{
points(c(estb0-crit*sdb0,estb0+crit*sdb0),rep(y,2),col="red",type="l")
}
}

ordermethod <- c("gamma.ind","logn.ind","semi.ind","gamma.ar1")

estb0s <- c(
re.gamma.ind$est[3,1],
re.logn.ind$est[3,1],
re.semi.ind$est[3],
re.gamma.ar1$est[4,1]
)

## The true beta0 is:
b0
c <- 1.1
plot(0,0,type="n",xlim=c(min(estb0s)-0.5,max(estb0s)*c),
ylim=c(0,5),yaxt="n",
main="Simulated from the independent model \n with random effect ~ gamma")

legend("topright",
col="red",
legend=ordermethod)
abline(v=b0,lty=3)

## [1] gamma.ind
sdb0 <- re.gamma.ind$est[3,2]
getpoints(4,estb0s[1],sdb0)
```

```
## [2] logn.ind
sdb0 <- re.logn.ind$est[3,2]
getpoints(3,estb0s[2],sdb0)

## [3] semi.ind
getpoints(2,estb0s[3])

## [4] gamma.ar1
sdb0 <- re.gamma.ar1$est[4,2]
getpoints(1,estb0s[4],sdb0)


## End(Not run)
```

`rNBME.R` *Simulate a dataset from the negative binomial mixed-effect independent/AR(1) model*

Description

This function simulates a dataset based on the negative binomial mixed-effect independent/AR(1) model with two treatment groups described in Zhao et al. The group mean can be different at each time point, but no other covariates are allowed.

See `mle.fun`, `mle.ar1.fun` for details of the model explanations.

Usage

```
rNBME.R(
gdist = "G", n = 200, sn = 5, th = exp(1.3),
u1 = rep(1.5, 5), u2 = rep(1.5, 5),
a = exp(-0.5),d=NULL,  othrp = list(u.n = 3, s.n = 0.5, p.mx = 0.05, sh.mx = NA)
)
```

Arguments

`gdist`	The distribution of the random effect term *G_i*. If `gdist="G"`, *G_i* is from the gamma distribution. If `gdist="N"`, *G_i* is from the log normal distribution. If `gdist="U"`, *G_i* (on the log scale) is from the uniform distribution. If `gdist="GN"`, *G_i* is from the mixture of the gamma distribution and the normal distribution. If the generated values are negative, they are truncated to zero. If `gdist="NoN"`, *G_i* is sampled from the pre-specified vector `othrp` with replacement.
`n`	The number of patients. It must be an even number.

sn	The number of repeated measures per patient. Generated datasets are balanced design.
th	If gdist="G", th is a scale parameter of the gamma distribution. If gdist="N" or gdist=="U", th is *var(G_i)*. If gdist="GN", see details. If gdist="NoN", this parameter is not used.
u1	A vector of length *sn*, specifying the mean of the treatment group 1 *E(Y_ij)* = u1[j].
u2	A vector of length *sn*, specifying the mean of the treatment group 2 *E(Y_ij)* = u2[j].
a	The parameter *a* of the negative binomial mixed-effect independent model. See *mle.fun*.
d	If d=NULL, generate data from the independent model. If d is a scalar between 0 and 1, then d is *delta* in the AR(1) model, and generate datasets from the AR(1) model.
othrp	If gdist="GN", parameters for the GN option. See details. If gdist="NoN", othrp is a vector, containing a sample of *G_i*, which is treated as a population and *G_i* is resampled.

Details

The generated datasets have equal number of scans per person.

The number of patients in the two groups are the same.

If gdist=="GN", datasets are generated from:

othrp$p.mx*N(mean=othrp$u.n,s.d=othrp$s.n) + (1-othrp$p.mx)*gamma(scale=th,shape),

where shape of the gamma distribution is chosen to ensure *E(G_i)=1*.

Value

id	The vector of length n*sn containing patient IDs: rep(1:n,each=sn)
vn	The vector of length n*sn containing the indicies of time points: rep(1:sn, n)
gp	The vector of length n*sn containing the indicies of the treatment groups
y	The vector of length n*sn containing generated response counts
g	The vector of length n*sn containing generated random effect terms
Gpara	The record of the distribution and parameter specifications used to generate the dataset

Author(s)

Zhao, Y. and Kondo, Y.

References

Zhao, Y., Li, D.K.B., Petkau, J.A., Riddehough, A. & Traboulsee, A. Detection of unusual increases in MRI lesion counts in multiple sclerosis patients.

See Also

The functions to fit related models: mle.fun, mle.ar1.fun, mle.a3.fun, mle.ar1.non3,

The subroutines of index.batch to compute the conditional probability index: jCP.ar1, CP1.ar1, MCCP.ar1, CP.ar1.se, CP.se, jCP,

Examples

```
## Not run:

## ============================================================================== ##
## generate a simulated dataset from the negative binomial mixed-effect
## independent model:
## Y_ij | G_i = g_i ~ NB(r_ij,p_i) where r_ij = exp(X^T beta)/a , p_i =1/(a*g_i+1)
## with G_i ~ log-normal(E(G_i)=1,var(G_i)=th)
set.seed(1)
sn = 5 # the number of repeated measures of each patient
n = 80 ## the number of patients
loga = - 0.5
a = exp(loga) ## dispersion parameter
logtheta <- 1.3
th = exp(logtheta) ## the variance of the gamma distributed random effect g_i


## No difference between the means of groups
## The model only has an intercept term beta0 = 0.5
b0 = 0.5
u1 = rep(exp(b0),sn)  ## the mean vector for group 1 at time point 1,...,sn
u2 = rep(exp(b0),sn) ## the mean vector for group 2 at time point 1,...,sn

## Data 0 generated from the IND model
DT.ind= rNBME.R(gdist="N", n=n, a=a, th=th, u1=u1, u2=u2, sn=sn)
## Data 1
DT.ar= rNBME.R(gdist="N", n=n, a=a, th=th, u1=u1, u2=u2, sn=sn, d=0.4)

dt.ind=data.frame(CEL=DT.ind$y,Vcode=DT.ind$vn-2,ID=DT.ind$id)
dt.ar=data.frame(CEL=DT.ar$y,Vcode=DT.ar$vn-2,ID=DT.ar$id)
## ============================================================================== ##

#### Fit IND models
tst1=lmeNB(CEL~1, data=dt.ind, ID=dt.ind$ID, IPRT=T)
tst2=lmeNB(CEL~1, data=dt.ar, ID=dt.ar$ID, IPRT=T)
tst3=lmeNB(CEL~1, data=dt.ind, ID=dt.ind$ID, IPRT=T, RE="N")
tst4=lmeNB(CEL~1, data=dt.ar, ID=dt.ar$ID, IPRT=T, RE="N") #not printing
tst5=lmeNB(CEL~Vcode, data=dt.ind, ID=dt.ind$ID, IPRT=T, RE="N")
## conditional probability index
Psum5=index.batch(olmeNB=tst5, labelnp=dt.ind$Vcode >= 1, data=dt.ind, ID=dt.ind$ID)



## Fit the semi-parametric model
tst6=lmeNB(CEL~1, data=dt.ind, ID=dt.ind$ID, IPRT=T, RE="semipara")
```

```
tst7=lmeNB(CEL~Vcode, data=dt.ind, ID=dt.ind$ID, IPRT=T, RE="semipara")
## conditional probability index
Psum7=index.batch(olmeNB=tst7,labelnp=dt.ind$Vcode >= 1, data=dt.ind,
                 ID=dt.ind$ID, Vcode=Vcode)


## Fit the AR1 models
tst8=lmeNB(CEL~1, data=dt.ind, ID=dt.ind$ID, IPRT=T,AR=T,Vcode=dt.ind$Vcode)
tst9=lmeNB(CEL~1, data=dt.ar, ID=dt.ar$ID, IPRT=T,AR=T,Vcode=dt.ar$Vcode)
## conditional probability index
Psum9=index.batch(olmeNB=tst9, labelnp=dt.ind$Vcode >= 1, data=dt.ind,
                  ID=dt.ind$ID,Vcode=dt.ind$Vcode)



## End(Not run)
```

Index

∗Topic **\textasciitildekwd1**
CP.ar1.se, 2
CP.se, 5
index.batch, 7
lmeNB, 10
mle.a3.fun, 13
mle.ar1.fun, 18
mle.ar1.non3, 24
mle.fun, 30
rNBME.R, 35
∗Topic **\textasciitildekwd2**
CP.ar1.se, 2
CP.se, 5
index.batch, 7
lmeNB, 10
mle.a3.fun, 13
mle.ar1.fun, 18
mle.ar1.non3, 24
mle.fun, 30
rNBME.R, 35

CP.ar1.se, 2, *7*, *9*, *12*, *15*, *20*, *26*, *32*, *37*
CP.se, *4*, 5, *7*, *9*, *12*, *15*, *20*, *26*, *32*, *37*
CP1.ar1, *9*, *12*, *15*, *20*, *26*, *32*, *37*
CP1.ar1 *(*CP.ar1.se*)*, 2

formulaToDat *(*mle.fun*)*, 30

index.batch, 2, *4*, *5*, *7*, 7, *9*, *12*, *15*, *20*, *26*, *32*, *37*
integrate, *11*, *19*, *31*

jCP, *4*, *7*, *9*, *12*, *15*, *20*, *26*, *32*, *37*
jCP *(*CP.se*)*, 5
jCP.ar1, *9*, *12*, *15*, *20*, *26*, *32*, *37*
jCP.ar1 *(*CP.ar1.se*)*, 2

lmeNB, *8*, *9*, 10, *32*

MCCP.ar1, *7*, *9*, *12*, *15*, *20*, *26*, *32*, *37*
MCCP.ar1 *(*CP.ar1.se*)*, 2

mle.a3.fun, *4*, *6–10*, *12*, 13, *20*, *26*, *32*, *37*
mle.ar1.fun, *4*, *7–10*, *12*, *15*, 18, *26*, *32*, *35*, *37*
mle.ar1.non3, *3*, *4*, *7–10*, *12*, *15*, *20*, 24, *32*, *37*
mle.fun, *4*, *7–10*, *12*, *15*, *18*, *20*, *24*, *26*, 30, *35–37*

optim, *11*, *19*, *31*

rNBME.R, *4*, *7*, *9*, *12*, *15*, *20*, *26*, *32*, 35