Package: qape (via r-universe)

October 14, 2024

Type Package

Title Quantile of Absolute Prediction Errors

Version 2.1

Date 2023-08-21

Author Alicja Wolny-Dominiak, Tomasz Zadlo

Maintainer Alicja Wolny-Dominiak
`<alicja.wolny-dominiak@ue.katowice.pl>`

Imports lme4, Matrix, mvtnorm, plyr, dplyr, matrixcalc, future.apply, reshape2

Depends R (>= 3.5.0)

Description Estimates QAPE using bootstrap procedures. The residual, parametric and double bootstrap is used. The test of normality using Cholesky decomposition is added. Y pop is defined.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-21 14:10:12 UTC

Contents

`bootPar` *Parametric bootstrap estimators of prediction accuracy*

Description

The function computes values of parametric bootstrap estimators of RMSE and QAPE prediction accuracy measures.

Usage

```
bootPar(predictor, B, p)
```

Arguments

`predictor`	one of objects: EBLUP, ebpLMMne or plugInLMM.
`B`	number of iterations in the bootstrap procedure.
`p`	orders of quantiles in the QAPE.

Details

We use bootstrap model presented by Chatterjee, Lahiri and Li (2008) p. 1229 but assumed for all population elements. Vectors of random effects and random components are generated from the multivariate normal distribution where REML estimates of model parameters are used. Random effects are generated for all population elements even for subsets with zero sample sizes (for which random effects are not estimated). We use the MSE estimator defined as the mean of squared bootstrap errors considered by Rao and Molina (2015) p. 141 and given by equation (6.2.22). The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. It is estimated as a quantile of absolute bootstrap errors as proposed by Zadlo (2017) in Section 2.

Value

`estQAPE`	estimated value/s of QAPE - number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).
`estRMSE`	estimated value/s of RMSE (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`summary`	estimated accuracy measures for the predictor of characteristics defined in thetaFun.
`predictorSim`	bootstrapped values of the predictor/s.
`thetaSim`	bootstrapped values of the predicted population or subpopulation characteristic/s.
`Ysim`	simulated values of the (possibly tranformed) variable of interest.
`error`	differences between bootstrapped values of the predictor/s and bootstrapped values of the predicted characteristic/s.
`positiveDefiniteEstG`	logical indicating if the estimated covariance matrix of random effects, used to generate bootstrap realizations of the dependent variable, is positive definite.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Butar, B. F., Lahiri, P. (2003) On measures of uncertainty of empirical Bayes small-area estimators, Journal of Statistical Planning and Inference, 112, 63-76.

2. Chatterjee, S., Lahiri, P. Li, H. (2008) Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models, Annals of Statistics, Vol. 36 (3), 1221-1245.

3. Rao, J.N.K. and Molina, I. (2015) Small Area Estimation. Second edition, John Wiley & Sons, New Jersey.

4. Zadlo T. (2017), On asymmetry of prediction errors in small area estimation, Statistics in Transition, 18 (3), 413-432.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)
library(matrixcalc)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}
set.seed(123456)

predictor <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictor$thetaP

### Estimation of prediction accuracy
est_accuracy <- bootPar(predictor, 10, c(0.75,0.9))

# Estimation of prediction RMSE
est_accuracy$estRMSE

# Estimation of prediction QAPE
est_accuracy$estQAPE

#        [,1]     [,2]
# 75% 2888.291 115.6076
# 90% 5472.738 127.0623

####### Interpretations in case of prediction of investments
####### for population element no. 379:
```

```
### It is estimated that at least 75% of absolute prediction errors are
# smaller or equal 2888.291 milion Polish zloty
# and at least 25% of absolute prediction errors are
# greater or equal 2888.291 milion Polish zloty.
### It is estimated that at least 90% of absolute prediction errors are
# smaller or equal 5472.738 milion Polish zloty
# and at least 10% of absolute prediction errors are
# greater or equal 5472.738 milion Polish zloty.

detach(invData2018)
```

`bootParFuture` *Parametric bootstrap estimators of prediction accuracy - parallel computing.*

Description

The function computes values of parametric bootstrap estimators of RMSE and QAPE prediction accuracy measures using parallel computing

Usage

```
bootParFuture(predictor, B, p)
```

Arguments

`predictor`	one of objects: EBLUP, ebpLMMne or plugInLMM.
`B`	number of iterations in the bootstrap procedure.
`p`	orders of quantiles in the QAPE.

Details

We use bootstrap model presented by Chatterjee, Lahiri and Li (2008) p. 1229 but assumed for all population elements. Vectors of random effects and random components are generated from the multivariate normal distribution where REML estimates of model parameters are used. Random effects are generated for all population elements even for subsets with zero sample sizes (for which random effects are not estimated). We use the MSE estimator defined as the mean of squared bootstrap errors considered by Rao and Molina (2015) p. 141 and given by equation (6.2.22). The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. It is estimated as a quantile of absolute bootstrap errors as proposed by Zadlo (2017) in Section 2. The parallel processing is performed via the *future.apply* package.

Value

`estQAPE`	estimated value/s of QAPE - number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).
`estRMSE`	estimated value/s of RMSE (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`summary`	estimated accuracy measures for the predictor of characteristics defined in thetaFun.
`predictorSim`	bootstrapped values of the predictor/s.
`thetaSim`	bootstrapped values of the predicted population or subpopulation characteristic/s.
`Ysim`	simulated values of the (possibly tranformed) variable of interest.
`error`	differences between bootstrapped values of the predictor/s and bootstrapped values of the predicted characteristic/s.
`positiveDefiniteEstG`	logical indicating if the estimated covariance matrix of random effects, used to generate bootstrap realizations of the dependent variable, is positive definite.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Butar, B. F., Lahiri, P. (2003) On measures of uncertainty of empirical Bayes small-area estimators, Journal of Statistical Planning and Inference, Vol. 112, pp. 63-76.

2. Chatterjee, S., Lahiri, P. Li, H. (2008) Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models, Annals of Statistics, Vol. 36 (3), pp. 1221?1245.

3. Rao, J.N.K. and Molina, I. (2015) Small Area Estimation. Second edition, John Wiley & Sons, New Jersey.

4. Zadlo T. (2017), On asymmetry of prediction errors in small area estimation, Statistics in Transition, 18 (3), 413-432.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)
library(matrixcalc)
library(future.apply)
```

```
data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}
set.seed(123)

predictor <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictor$thetaP

### Estimation of prediction accuracy
est_accuracy <- bootParFuture(predictor, 10, c(0.75,0.9))

# Estimation of prediction RMSE
est_accuracy$estRMSE

# Estimation of prediction QAPE
est_accuracy$estQAPE

#        [,1]     [,2]
# 75% 1370.823 180.0514
# 90% 1477.444 249.7517

####### Interpretations in case of prediction of investments
####### for population element no. 379:
### It is estimated that at least 75% of absolute prediction errors are
# smaller or equal 1370.823 milion Polish zloty
# and at least 25% of absolute prediction errors are
# greater or equal 1370.823 milion Polish zloty.
### It is estimated that at least 90% of absolute prediction errors are
# smaller or equal 1477.444 milion Polish zloty
# and at least 10% of absolute prediction errors are
# greater or equal 1477.444 milion Polish zloty.

detach(invData2018)
```

`bootParFutureCor` *Parametric bootstrap estimators of prediction accuracy - parallel computing using corrected covariance matrices*

Description

The function computes values of parametric bootstrap estimators of RMSE and QAPE prediction accuracy measures using parallel computing under the misspecified model. The model misspecification is obtained by the modification of the covariance matrices of random effects and random components estimated based on sample data. The correction is made by the division of the diagonal elements of random effects and random components estimated based on sample data by values defined by users and then, the corrected covariance matrices are used to generate bootstrap realizations of the dependent variables.

Usage

```
bootParFutureCor(predictor, B, p, ratioR, ratioG)
```

Arguments

`predictor`	one of objects: EBLUP, ebpLMMne or plugInLMM.
`B`	number of iterations in the bootstrap procedure.
`p`	orders of quantiles in the QAPE.
`ratioR`	the value by which the diagonal elements of the covariance matrix of random components estimated based on sample data are divided. Then, the corrected covariance matrix is used to generate bootstrap realizations of random components.
`ratioG`	the value by which the diagonal elements of the covariance matrix of random effects estimated based on sample data are divided. Then, the corrected covariance matrix, assuming that it is positive definite, is used to generate bootstrap realizations of random effects. If it is not positive definite, the alert is printed and the dependent variable is generated based on the model without random effects.

Details

We use bootstrap model presented by Chatterjee, Lahiri and Li (2008) p. 1229 but assumed for all population elements. Vectors of random effects and random components are generated from the multivariate normal distribution, where REML estimates of model parameters are used. Random effects are generated for all population elements, even for subsets with zero sample sizes (for which random effects are not estimated). We use the MSE estimator defined as the mean of squared bootstrap errors considered by Rao and Molina (2015) p. 141 and given by equation (6.2.22). The QAPE is a quantile of absolute prediction error, which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. It is estimated as a quantile of absolute bootstrap errors, as proposed by Zadlo (2017) in Section 2. The parallel processing is performed via the *future.apply* package. The dependent variable is generated based on the modified (misspecified) model with corrected covariance matrices of random effects and random components. The correction is made by the division of the diagonal elements of the covariance matrix of random components estimated based on sample data by *ratioR*, and by the division of the diagonal elements of the covariance matrix of random effects estimated based on sample data by *ratioG*. If the estimated covariance matrix of random effect after the correction is not positive definite, the alert is printed and the bootstrap realizations of dependent variable are generated based on the model without random effects.

Value

`estQAPE`	estimated value/s of QAPE - number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).
`estRMSE`	estimated value/s of RMSE (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`summary`	estimated accuracy measures for the predictor of characteristics defined in thetaFun.
`predictorSim`	bootstrapped values of the predictor/s.
`thetaSim`	bootstrapped values of the predicted population or subpopulation characteristic/s.
`Ysim`	simulated values of the (possibly tranformed) variable of interest.
`error`	differences between bootstrapped values of the predictor/s and bootstrapped values of the predicted characteristic/s.
`positiveDefiniteEstG`	logical indicating if the estimated covariance matrix of random effects, used to generate bootstrap realizations of the dependent variable, is positive definite.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Butar, B. F., Lahiri, P. (2003) On measures of uncertainty of empirical Bayes small-area estimators, Journal of Statistical Planning and Inference, Vol. 112, pp. 63-76.

2. Chatterjee, S., Lahiri, P. Li, H. (2008) Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models, Annals of Statistics, Vol. 36 (3), pp. 1221?1245.

3. Rao, J.N.K. and Molina, I. (2015) Small Area Estimation. Second edition, John Wiley & Sons, New Jersey.

4. Zadlo T. (2017), On asymmetry of prediction errors in small area estimation, Statistics in Transition, 18 (3), 413-432.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)
library(matrixcalc)
library(future.apply)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}
set.seed(123)

predictor <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictor$thetaP

### Estimation of prediction accuracy under the misspecified model
est_accuracy <- bootParFutureCor(predictor, 10, c(0.75,0.9), 2, 0.01)

# Estimation of prediction RMSE under the misspecified model
est_accuracy$estRMSE

# Estimation of prediction QAPE under the misspecified model
est_accuracy$estQAPE

detach(invData2018)
```

`bootParMis` *Parametric bootstrap estimators of prediction accuracy under the misspecified model*

Description

The function computes values of parametric bootstrap estimators of RMSE and QAPE prediction accuracy measures of two predictors under the model assumed for one of them.

Usage

```
bootParMis(predictorLMM, predictorLMMmis, B, p)
```

Arguments

`predictorLMM`	plugInLMM object, the first predictor used to define the bootstrap model.
`predictorLMMmis`	plugInLMM object, the second predictor.
`B`	number of iterations in the bootstrap procedure.
`p`	orders of quantiles in the QAPE.

Details

We use bootstrap model presented by Chatterjee, Lahiri and Li (2008) p. 1229 but assumed for all population elements. We use model specification used in *predictorLMM*. Vectors of random effects and random components are generated from the multivariate normal distribution where REML estimates of model parameters are used. Random effects are generated for all population elements even for subsets with zero sample sizes (for which random effects are not estimated). We use the MSE estimator defined as the mean of squared bootstrap errors considered by Rao and Molina (2015) p. 141 and given by equation (6.2.22). The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. It is estimated as a quantile of absolute bootstrap errors as proposed by Zadlo (2017) in Section 2. The prediction accuracy of two predictors *predictorLMM* and *predictorLMMmis* is estimated under the model specified in *predictorLMM*.

Value

`estQAPElmm`	estimated value/s of QAPE of *predictorLMM* - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in *thetaFun*).
`estRMSElmm`	estimated value/s of RMSE of *predictorLMM* (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`estQAPElmmMis`	estimated value/s of QAPE of *predictorLMMmis* - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in *thetaFun*).
`estRMSElmmMis`	estimated value/s of RMSE of *predictorLMMmis* (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`predictorLMMSim`	bootstrapped values of *predictorLMM*.
`predictorLMMmisSim`	bootstrapped values of *predictorLMMmis*.

`thetaSim`	bootstrapped values of the predicted population or subpopulation characteristic/s.
`Ysim`	simulated values of the (possibly tranformed) variable of interest.
`errorLMM`	differences between bootstrapped values of *predictorLMM* and bootstrapped values of the predicted characteristic/s.
`errorLMMmis`	differences between bootstrapped values of *predictorLMMmis* and bootstrapped values of the predicted characteristic/s.
`positiveDefiniteEstG`	logical indicating if the estimated covariance matrix of random effects, used to generate bootstrap realizations of the dependent variable, is positive definite.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Butar, B. F., Lahiri, P. (2003) On measures of uncertainty of empirical Bayes small-area estimators, Journal of Statistical Planning and Inference, Vol. 112, pp. 63-76.

2. Chatterjee, S., Lahiri, P. Li, H. (2008) Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models, Annals of Statistics, Vol. 36 (3), pp. 1221?1245.

3. Rao, J.N.K. and Molina, I. (2015) Small Area Estimation. Second edition, John Wiley & Sons, New Jersey.

4. Zadlo T. (2017), On asymmetry of prediction errors in small area estimation, Statistics in Transition, 18 (3), 413-432.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)
library(matrixcalc)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed
```

```
YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'
random.part.mis <- '(1|NUTS4type)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}


predictorLMM<-plugInLMM(YS, fixed.part, random.part, reg, con, weights,backTrans,thetaFun)
predictorLMM$thetaP
predictorLMMmis <- plugInLMM(YS, fixed.part, random.part.mis, reg, con,weights,backTrans,thetaFun)
predictorLMMmis$thetaP

set.seed(123456)
### Estimation of prediction accuracy under the model used to define predictorLMM
est_accuracy <- bootParMis(predictorLMM, predictorLMMmis, 10, c(0.75,0.9))

# Estimation of prediction RMSE of predictorLMM
est_accuracy$estRMSElmm

# Estimation of prediction RMSE of predictorLMMmis
est_accuracy$estRMSElmmMis

# Estimation of prediction QAPE of predictorLMM
est_accuracy$estQAPElmm

# Estimation of prediction QAPE of predictorLMMmis
est_accuracy$estQAPElmmMis

detach(invData2018)
```

`bootRes` *Residual bootstrap estimators of prediction accuracy*

Description

The function computes values of residual bootstrap estimators of RMSE and QAPE prediction accuracy measures.

Usage

```
bootRes(predictor, B, p, correction)
```

Arguments

`predictor`	one of objects: EBLUP, ebpLMMne or plugInLMM.
`B`	number of iterations in the bootstrap procedure.
`p`	orders of quantiles in the QAPE.
`correction`	logical. If TRUE, both bootstrapped random effects and random components are tranformed to avoid the problem of underdispersion of residual bootstrap distributions (see Details).

Details

Residual bootstrap considered by Carpener, Goldstein and Rasbash (2003), Chambers and Chandra (2013) and Thai et al. (2013) is used. To generate one bootstrap realization of the population vector of the variable of interest: (i) from the sample vector of predicted random components the simple random sample with replacement of population size is drawn at random, (ii) from the vector of predicted random effects the simple random sample with replacement of size equal to the number of random effects in the whole population is drawn at random. If *correction* is *TRUE*, then predicted random effects are transformed as described in Carpener, Goldstein and Rasbash (2003) in Section 3.2 and predicted random components as presented in Chambers and Chandra (2013) in Section 2.2. We use the MSE estimator defined as the mean of squared bootstrap errors considered by Rao and Molina (2015) p. 141 given by equation (6.2.22). The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. It is estimated as a quantile of absolute bootstrap errors as proposed by Zadlo (2017) in Section 2.

Value

`estQAPE`	estimated value/s of QAPE - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*).
`estRMSE`	estimated value/s of RMSE (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`summary`	estimated accuracy measures for the predictor of characteristics defined in thetaFun.
`summary`	estimated accuracy measures for the predictor of characteristics defined in thetaFun.
`predictorSim`	bootstrapped values of the predictor/s.
`thetaSim`	bootstrapped values of the predicted population or subpopulation characteristic/s.
`Ysim`	simulated values of the (possibly tranformed) variable of interest.
`error`	differences between bootstrapped values of the predictor/s and bootstrapped values of the predicted characteristic/s.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

2. Chambers, R. and Chandra, H. (2013) A Random Effect Block Bootstrap for Clustered Data, Journal of Computational and Graphical Statistics, 22(2), 452-470.

3. Thai, H.-T., Mentre, F., Holford, N.H., Veyrat-Follet, C. and Comets, E. (2013), A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. Pharmaceutical Statistics, 12, 129-140.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379:380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379:380)]}
set.seed(123456)

predictor <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictor$thetaP

### Estimation of prediction accuracy
est_accuracy <- bootRes(predictor, 10, c(0.5,0.8), correction = TRUE)

# Estimation of prediction RMSE
est_accuracy$estRMSE
```

```
# Estimation of prediction QAPE
est_accuracy$estQAPE

#         [,1]      [,2]
#50%  612.6089  67.45543
#80% 1886.9269 120.16246
####### Interpretations in case of prediction of investments
####### for population element no. 379:
### It is estimated that at least 50% of absolute prediction errors are
# smaller or equal 612.6089 milion Polish zloty
# and at least 50% of absolute prediction errors are
# greater or equal 612.6089 milion Polish zloty.
### It is estimated that at least 80% of absolute prediction errors are
# smaller or equal 1886.9269 milion Polish zloty
# and at least 20% of absolute prediction errors are
# greater or equal 1886.9269 milion Polish zloty.

detach(invData2018)
```

`bootResFuture` *Residual bootstrap estimators of prediction accuracy - parallel computing*

Description

The function computes values of residual bootstrap estimators of RMSE and QAPE prediction accuracy measures using parallel computing.

Usage

```
bootResFuture(predictor, B, p, correction)
```

Arguments

`predictor`	one of objects: EBLUP, ebpLMMne or plugInLMM.
`B`	number of iterations in the bootstrap procedure.
`p`	orders of quantiles in the QAPE.
`correction`	logical. If TRUE, both bootstrapped random effects and random components are tranformed to avoid the problem of underdispersion of residual bootstrap distributions (see Details).

Details

Residual bootstrap considered by Carpener, Goldstein and Rasbash (2003), Chambers and Chandra (2013) and Thai et al. (2013) is used. To generate one bootstrap realization of the population vector of the variable of interest: (i) from the sample vector of predicted random components the simple random sample with replacement of population size is drawn at random, (ii) from the vector of predicted random effects the simple random sample with replacement of size equal the number of random effects in the whole population is drawn at random. If *correction* is *TRUE*, then predicted random effects are transformed as described in Carpener, Goldstein and Rasbash (2003) in Section 3.2 and predicted random components as presented in Chambers and Chandra (2013) in Section 2.2. We use the MSE estimator defined as the mean of squared bootstrap errors considered by Rao and Molina (2015) p. 141 given by equation (6.2.22). The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. It is estimated as a quantile of absolute bootstrap errors as proposed by Zadlo (2017) in Section 2. The parallel processing is performed via the *future.apply* package.

Value

`estQAPE`	estimated value/s of QAPE - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estRMSE`	estimated value/s of RMSE (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`summary`	estimated accuracy measures for the predictor of characteristics defined in thetaFun.
`predictorSim`	bootstrapped values of the predictor/s.
`thetaSim`	bootstrapped values of the predicted population or subpopulation characteristic/s.
`Ysim`	simulated values of the (possibly tranformed) variable of interest.
`error`	differences between bootstrapped values of the predictor/s and bootstrapped values of the predicted characteristic/s.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

2. Chambers, R. and Chandra, H. (2013) A Random Effect Block Bootstrap for Clustered Data, Journal of Computational and Graphical Statistics, 22(2), 452-470.

3. Thai, H.-T., Mentre, F., Holford, N.H., Veyrat-Follet, C. and Comets, E. (2013), A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. Pharmaceutical Statistics, 12, 129-140.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379:380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379:380)]}
set.seed(123456)

predictor <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictor$thetaP

### Estimation of prediction accuracy
est_accuracy <- bootResFuture(predictor, 10, c(0.5,0.8), correction = TRUE)

# Estimation of prediction RMSE
est_accuracy$estRMSE

# Estimation of prediction QAPE
est_accuracy$estQAPE

#         [,1]      [,2]
#50%  612.6089  67.45543
#80% 1886.9269 120.16246
####### Interpretations in case of prediction of investments
####### for population element no. 379:
### It is estimated that at least 50% of absolute prediction errors are
# smaller or equal 612.6089 milion Polish zloty
# and at least 50% of absolute prediction errors are
# greater or equal 612.6089 milion Polish zloty.
### It is estimated that at least 80% of absolute prediction errors are
# smaller or equal 1886.9269 milion Polish zloty
```

```
# and at least 20% of absolute prediction errors are
# greater or equal 1886.9269 milion Polish zloty.

detach(invData2018)
```

`bootResMis` *Residual bootstrap estimators of prediction accuracy under the misspecified model*

Description

The function computes values of residual bootstrap estimators of RMSE and QAPE prediction accuracy measures of two predictors under the model assumed for one of them.

Usage

```
bootResMis(predictorLMM, predictorLMMmis, B, p, correction)
```

Arguments

`predictorLMM`	plugInLMM object, the first predictor used to define the bootstrap model.
`predictorLMMmis`	plugInLMM object, the second predictor.
`B`	number of iterations in the bootstrap procedure.
`p`	orders of quantiles in the QAPE.
`correction`	logical. If TRUE, both bootstrapped random effects and random components are tranformed to avoid the problem of underdispersion of residual bootstrap distributions (see Details).

Details

Residual bootstrap considered by Carpener, Goldstein and Rasbash (2003), Chambers and Chandra (2013) and Thai et al. (2013) is used. We use model specification used in *predictorLMM*. To generate one bootstrap realization of the population vector of the variable of interest: (i) from the sample vector of predicted random components the simple random sample with replacement of population size is drawn at random, (ii) from the vector of predicted random effects the simple random sample with replacement of size equal the number of random effects in the whole population is drawn at random. If *correction* is *TRUE*, then predicted random effects are transformed as described in Carpener, Goldstein and Rasbash (2003) in Section 3.2 and predicted random components as presented in Chambers and Chandra (2013) in Section 2.2. We use the MSE estimator defined as the mean of squared bootstrap errors considered by Rao and Molina (2015) p. 141 given by equation (6.2.22). The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. It is estimated as a quantile of absolute bootstrap errors as proposed by Zadlo (2017) in Section 2. The prediction accuracy of two predictors *predictorLMM* and *predictorLMMmis* is estimated under the model specified in *predictorLMM*.

Value

`estQAPElmm`	estimated value/s of QAPE of *predictorLMM* - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in *thetaFun*).
`estRMSElmm`	estimated value/s of RMSE of *predictorLMM* (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`estQAPElmmMis`	estimated value/s of QAPE of *predictorLMMmis* - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in *thetaFun*).
`estRMSElmmMis`	estimated value/s of RMSE of *predictorLMMmis* (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`predictorLMMSim`	bootstrapped values of *predictorLMM*.
`predictorLMMmisSim`	bootstrapped values of *predictorLMMmis*.
`thetaSim`	bootstrapped values of the predicted population or subpopulation characteristic/s.
`Ysim`	simulated values of the (possibly tranformed) variable of interest.
`errorLMM`	differences between bootstrapped values of *predictorLMM* and bootstrapped values of the predicted characteristic/s.
`errorLMMmis`	differences between bootstrapped values of *predictorLMMmis* and bootstrapped values of the predicted characteristic/s.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

2. Chambers, R. and Chandra, H. (2013) A Random Effect Block Bootstrap for Clustered Data, Journal of Computational and Graphical Statistics, 22(2), 452-470.

3. Thai, H.-T., Mentre, F., Holford, N.H., Veyrat-Follet, C. and Comets, E. (2013), A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. Pharmaceutical Statistics, 12, 129-140.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)
```

```
data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379:380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'
random.part.mis <- '(1|NUTS4type)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379:380)]}

predictorLMM <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictorLMM$thetaP

predictorLMMmis <- plugInLMM(YS, fixed.part, random.part.mis, reg,con,weights,backTrans,thetaFun)
predictorLMMmis$thetaP

set.seed(123456)
### Estimation of prediction accuracy
est_accuracy <- bootResMis(predictorLMM, predictorLMMmis, 10, c(0.5,0.8), correction = TRUE)

# Estimation of prediction RMSE of predictorLMM
est_accuracy$estRMSElmm

# Estimation of prediction RMSE of predictorLMMmis
est_accuracy$estRMSElmmMis

# Estimation of prediction QAPE of predictorLMM
est_accuracy$estQAPElmm

# Estimation of prediction QAPE of predictorLMMmis
est_accuracy$estQAPElmmMis

detach(invData2018)
```

`correction` *Correction term for predicted random effects*

Description

The function computes the list of matrices used to correct predicted random effects as presented in Carpenter, Goldstein and Rasbash (2003) in Section 3.2 to avoid the problem of underdispersion of residual bootstrap distributions.

Usage

```
correction(model)
```

Arguments

`model`	*lmer* object.

Value

a list of square matrices used to correct predicted random effects. The length of the list is equal the number of grouping variables used in case of random effects. Each matrix is of order equal the number of random effects at the considered level of grouping.

Author(s)

Tomasz Zadlo

References

Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

Examples

```
library(lme4)
data(invData)
attach(invData)
model <- lmer(investments ~ newly_registered + ((1|NUTS2) +
((newly_registered - 1)|NUTS2) + ((newly_registered)|NUTS4)))
correction(model)

detach(invData)
```

`corrRancomp` *Correction of predicted random components*

Description

The function computes the corrected predicted random components as presented in Chambers and Chandra (2013) in Section 2.2 to avoid the problem of underdispersion of residual bootstrap distributions.

Usage

```
corrRancomp(model)
```

Arguments

`model`	*lmer* object.

Value

the vector of corrected predicted random components.

Author(s)

Tomasz Zadlo

References

Chambers, R. and Chandra, H. (2013) A Random Effect Block Bootstrap for Clustered Data, Journal of Computational and Graphical Statistics, 22(2), 452-470.

Examples

```
library(lme4)
data(invData)
attach(invData)
model <- lmer(investments ~ newly_registered + ((1|NUTS2) +
((newly_registered - 1)|NUTS2) + ((newly_registered)|NUTS4)))
corrRancomp(model)
detach(invData)
```

`corrRanef` *Correction of predicted random effects*

Description

The function computes the corrected predicted random effects as presented in Carpenter, Goldstein and Rasbash (2003) in Section 3.2 to avoid the problem of underdispersion of residual bootstrap distributions.

Usage

```
corrRanef(model)
```

Arguments

`model`	*lmer* object.

Value

a list of corrected predicted random effects (of the same form as *ranef(model)*).

Author(s)

Tomasz Zadlo

References

Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

Examples

```
library(lme4)
data(invData)
attach(invData)
model <- lmer(investments ~ newly_registered + ((1|NUTS2) +
((newly_registered - 1)|NUTS2) + ((newly_registered)|NUTS4)))
corrRanef(model)
detach(invData)
```

`doubleBoot` *Double bootstrap estimators of prediction accuracy*

Description

The function computes values of double bootstrap estimators of the MSE and the QAPE prediction accuracy measures.

Usage

```
doubleBoot(predictor, B1, B2, p, q)
```

Arguments

`predictor`	one of objects: EBLUP, ebpLMMne or plugInLMM.
`B1`	number of first-level bootstrap iterations.
`B2`	number of second-level bootstrap iterations.
`p`	orders of quantiles in the QAPE.
`q`	estimator bounds assumed for *estMSE_db_1_EF* and *estMSE_db_telesc_EF* (which are corrected versions of *estMSE_db_1* and *estMSE_db_telesc*, respectively).

Details

Double-bootstrap method considered by Hall and Maiti (2006) and Erciulescu and Fuller (2013) is used. Vectors of random effects and random components are generated from the multivariate normal distribution and REML estimates of model parameters are used. Random effects are generated for all population elements even for subsets with zero sample sizes (for which random effects are not estimated). Double-bootstrap MSE estimator presented in Hall and Maiti (2006) and Erciulescu and Fuller (2013) are taken into account. The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE.

Value

`estMSE_param`	value/s of the parametric bootstrap MSE estimator. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2`	value/s of the double bootstrap MSE estimator computed as the difference of doubled value of *estMSE_param* and the second-level MSE estimator based on B2 iterations. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2_WDZ`	value/s of the double bootstrap MSE estimator computed as the mean of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). More than one value is computed if in *thetaFun* more than one population characteristic is defined.

`estMSE_db_B2_HM`	value/s of the double bootstrap MSE estimator proposed by Hall and Maiti (2006) equation (2.17). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1`	value/s of the double bootstrap MSE estimator computed as the difference of doubled value of *estMSE_param* and the second-level MSE estimator based on B2=1 iteration. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1_WDZ`	value/s of the double bootstrap MSE estimator computed as the mean of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1_EF`	value/s of the double bootstrap MSE estimator proposed by Erciulescu and Fuller (2014) given by equation (13) with correction (17), where the bound for the correction is declared as *q*. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_telesc`	value/s of the telescoping double bootstrap MSE estimator proposed by Erciulescu and Fuller (2014) given by equation (15). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_telesc_WDZ`	value/s of the double bootstrap MSE estimator computed as the mean of the sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_telesc_EF`	value/s of the telescoping double bootstrap MSE estimator proposed by Erciulescu and Fuller (2014) given by equation (15) with correction (17), where the bound for the correction is declared as *q*. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estQAPE_param`	value/s of parametric bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of absolute parametric bootstrap errors. Number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_B2`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of square roots of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). Number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).

`estQAPE_db_1`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of square roots of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). Number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_telesc`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of square roots of the sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error). Number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*).
`error1`	the matrix of first-level bootstrap errors. Number of rows is equal to the number of predicted characteristics (declared in in *thetaFun*), number of columns is equal to *B1*.
`error2`	the list of matrices of second-level bootstrap errors. The length of list is equal to the number of predicted characteristics (declared in in *thetaFun*), the number of rows of each matrix is equal to *B1*, the number of columns is equal to *B2*.
`corSquaredError1_db_B2`	the matrix of corrected squared first-level bootstrap errors defined as doubled squared first-level bootstrap errors minus the mean of squared second-level bootstrap errors (computed for the approriate first-level bootstrap iterations). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_1`	the matrix of corrected squared first-level bootstrap errors defined as doubled squared first-level bootstrap errors minus the squared second-level bootstrap error (computed once for each first-level bootstrap iteration). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_telesc`	the matrix of corrected squared first-level bootstrap errors defined by elements from which the average given by equation (15) in Erciulescu and Fuller (2014) is counted. Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_B2_WDZ`	the matrix of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.

`corSquaredError1_db_1_WDZ`
: the matrix of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.

`corSquaredError1_db_telesc_WDZ`
: the matrix of corrected squared first-level bootstrap errors defined by sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error).Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.

`positiveDefiniteEstGlev1`
: logical indicating if the estimated covariance matrix of random effects, used to generate bootstrap realizations of the dependent variable at the first level of the double bootstrap, is positive definite.

`positiveDefiniteEstGlev2`
: number of cases ouf of B1 with positive definite estimated covariance matrix of random effects used to generate bootstrap realizations of the dependent variable at the second level of the double bootstrap.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Erciulescu, A. L. and Fuller, W. A. (2013) Parametric Bootstrap Procedures for Small Area Prediction Variance. JSM 2014 - Survey Research Methods Section, 3307-3318.

2. Hall, P. and Maiti, T. (2006) On Parametric Bootstrap Methods for Small Area Prediction. Journal of the Royal Statistical Society. Series B, 68(2), 221-238.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)
```

```
N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

### Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}
set.seed(123456)

predictor <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictor$thetaP


### Estimation of prediction accuracy
# in the first column
# for the predictor of the value of the variable for population element no. 379,
# in the second column
# for the predictor of the value of the variable for population element no. 380:
doubleBoot(predictor, 3, 3, c(0.5,0.9), 0.77)
#q=0.77 assumed as in Erciulescu and FUller (2014) eq. (17)

detach(invData2018)
```

`doubleBootFuture` *Double bootstrap estimators of prediction accuracy - parallel computing*

Description

The function computes values of double bootstrap estimators of the MSE and the QAPE prediction accuracy measures using parallel computing.

Usage

```
doubleBootFuture(predictor, B1, B2, p, q)
```

Arguments

`predictor`	one of objects: EBLUP, ebpLMMne or plugInLMM.
`B1`	number of first-level bootstrap iterations.

`B2`	number of second-level bootstrap iterations.
`p`	orders of quantiles in the QAPE.
`q`	estimator bounds assumed for *estMSE_db_1_EF* and *estMSE_db_telesc_EF* (which are corrected versions of *estMSE_db_1* and *estMSE_db_telesc*, respectively).

Details

Double-bootstrap method considered by Hall and Maiti (2006) and Erciulescu and Fuller (2013) is used. Vectors of random effects and random components are generated from the multivariate normal distribution and REML estimates of model parameters are used. Random effects are generated for all population elements even for subsets with zero sample sizes (for which random effects are not estimated). Double-bootstrap MSE estimator presented in Hall and Maiti (2006) and Erciulescu and Fuller (2013) are taken into account. The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. The parallel processing is performed via the *future.apply* package.

Value

`estMSE_param`	value/s of the parametric bootstrap MSE estimator. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2`	value/s of the double bootstrap MSE estimator computed as the difference of doubled value of *estMSE_param* and the second-level MSE estimator based on B2 iterations. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2_WDZ`	value/s of the double bootstrap MSE estimator computed as the mean of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2_HM`	value/s of the double bootstrap MSE estimator proposed by Hall and Maiti (2006) equation (2.17). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1`	value/s of the double bootstrap MSE estimator computed as the difference of doubled value of *estMSE_param* and the second-level MSE estimator based on B2=1 iteration. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1_WDZ`	value/s of the double bootstrap MSE estimator computed as the mean of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1_EF`	value/s of the double bootstrap MSE estimator proposed by Erciulescu and Fuller (2014) given by equation (13) with correction (17), where the bound for the correction is declared as *q*. More than one value is computed if in *thetaFun* more than one population characteristic is defined.

`estMSE_db_telesc`	value/s of the telescoping double bootstrap MSE estimator proposed by Erciulescu and Fuller (2014) given by equation (15). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_telesc_WDZ`	value/s of the double bootstrap MSE estimator computed as the mean of the sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_telesc_EF`	value/s of the telescoping double bootstrap MSE estimator proposed by Erciulescu and Fuller (2014) given by equation (15) with correction (17), where the bound for the correction is declared as q. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estQAPE_param`	value/s of parametric bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of absolute parametric bootstrap errors. Number of rows is equal the number of orders of quantiles to be considered (declared in p), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_B2`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of square roots of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). Number of rows is equal the number of orders of quantiles to be considered (declared in p), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_1`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of square roots of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). Number of rows is equal the number of orders of quantiles to be considered (declared in p), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_telesc`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) given by a quantile of square roots of the sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error). Number of rows is equal to the number of orders of quantiles to be considered (declared in p), number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*).
`error1`	the matrix of first-level bootstrap errors. Number of rows is equal to the number of predicted characteristics (declared in in *thetaFun*), number of columns is equal to *B1*.

`error2`	the list of matrices of second-level bootstrap errors. The length of list is equal to the number of predicted characteristics (declared in in *thetaFun*), the number of rows of each matrix is equal to *B1*, the number of columns is equal to *B2*.
`corSquaredError1_db_B2`	the matrix of corrected squared first-level bootstrap errors defined as doubled squared first-level bootstrap errors minus the mean of squared second-level bootstrap errors (computed for the approriate first-level bootstrap iterations). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_1`	the matrix of corrected squared first-level bootstrap errors defined as doubled squared first-level bootstrap errors minus the squared second-level bootstrap error (computed once for each first-level bootstrap iteration). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_telesc`	the matrix of corrected squared first-level bootstrap errors defined by elements from which the average given by equation (15) in Erciulescu and Fuller (2014) is counted. Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_B2_WDZ`	the matrix of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.
`corSquaredError1_db_1_WDZ`	the matrix of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.
`corSquaredError1_db_telesc_WDZ`	the matrix of corrected squared first-level bootstrap errors defined by sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error).Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.
`positiveDefiniteEstGlev1`	logical indicating if the estimated covariance matrix of random effects, used to generate bootstrap realizations of the dependent variable at the first level of the double bootstrap, is positive definite.
`positiveDefiniteEstGlev2`	number of cases ouf of B1 with positive definite estimated covariance matrix of

random effects used to generate bootstrap realizations of the dependent variable at the second level of the double bootstrap.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Erciulescu, A. L. and Fuller, W. A. (2013) Parametric Bootstrap Procedures for Small Area Prediction Variance. JSM 2014 - Survey Research Methods Section, 3307-3318.

2. Hall, P. and Maiti, T. (2006) On Parametric Bootstrap Methods for Small Area Prediction. Journal of the Royal Statistical Society. Series B, 68(2), 221-238.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

### Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}
set.seed(123456)

predictor <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictor$thetaP


### Estimation of prediction accuracy
# in the first column
```

```
# for the predictor of the value of the variable for population element no. 379,
# in the second column
# for the predictor of the value of the variable for population element no. 380:
doubleBootFuture(predictor, 3, 3, c(0.5,0.9), 0.77)
#q=0.77 assumed as in Erciulescu and FUller (2014) eq. (17)

detach(invData2018)
```

`doubleBootMis` *Double bootstrap estimators of prediction accuracy under the misspecified model*

Description

The function computes values of double bootstrap estimators of the MSE and the QAPE prediction accuracy measures of two predictors under the model assumed for one of them.

Usage

```
doubleBootMis(predictorLMM, predictorLMMmis, B1, B2, p, q)
```

Arguments

`predictorLMM`	plugInLMM object, the first predictor used to define the bootstrap model.
`predictorLMMmis`	plugInLMM object, the second predictor.
`B1`	the number of first-level bootstrap iterations.
`B2`	the number of second-level bootstrap iterations.
`p`	orders of quantiles in the QAPE.
`q`	estimator bounds assumed for *estMSE_db_1_EF* and *estMSE_db_telesc_EF* (which are corrected versions of *estMSE_db_1* and *estMSE_db_telesc*, respectively).

Details

Double-bootstrap method considered by Hall and Maiti (2006) and Erciulescu and Fuller (2013) is used. We use model specification used in *predictorLMM*. Vectors of random effects and random components are generated from the multivariate normal distribution and REML estimates of model parameters are used. Random effects are generated for all population elements even for subsets with zero sample sizes (for which random effects are not estimated). Double-bootstrap MSE estimator presented in Hall and Maiti (2006) and Erciulescu and Fuller (2013) are taken into account. The QAPE is a quantile of absolute prediction error which means that at least p100% of realizations of absolute prediction errors are smaller or equal to QAPE. The prediction accuracy of two predictors *predictorLMM* and *predictorLMMmis* is estimated under the model specified in *predictorLMM*.

Value

`estMSE_param_LMMmis`	value/s of the parametric bootstrap MSE estimator of *predictorLMMmis*. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2_LMMmis`	value/s of the double bootstrap MSE estimator of *predictorLMMmis* computed as the difference of doubled value of *estMSE_param* and the second-level MSE estimator based on B2 iterations. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2_WDZ_LMMmis`	value/s of the double bootstrap MSE estimator of *predictorLMMmis* computed as the mean of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_B2_HM_LMMmis`	value/s of the double bootstrap MSE estimator of *predictorLMMmis* proposed by Hall and Maiti (2006) equation (2.17). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1_LMMmis`	value/s of the double bootstrap MSE estimator of *predictorLMMmis* computed as the difference of doubled value of *estMSE_param* and the second-level MSE estimator based on B2=1 iteration. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1_WDZ_LMMmis`	value/s of the double bootstrap MSE estimator of *predictorLMMmis* computed as the mean of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_1_EF_LMMmis`	value/s of the double bootstrap MSE estimator of *predictorLMMmis* proposed by Erciulescu and Fuller (2014) given by equation (13) with correction (17), where the bound for the correction is declared as *q*. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_telesc_LMMmis`	value/s of the telescoping double bootstrap MSE estimator of *predictorLMMmis* proposed by Erciulescu and Fuller (2014) given by equation (15). More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estMSE_db_telesc_WDZ_LMMmis`	value/s of the double bootstrap MSE estimator of *predictorLMMmis* computed as the mean of the sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error). More than one value is computed if in *thetaFun* more than one population characteristic is defined.

`estMSE_db_telesc_EF_LMMmis`	value/s of the telescoping double bootstrap MSE estimator of *predictorLMMmis* proposed by Erciulescu and Fuller (2014) given by equation (15) with correction (17), where the bound for the correction is declared as *q*. More than one value is computed if in *thetaFun* more than one population characteristic is defined.
`estQAPE_param_LMMmis`	value/s of parametric bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) of *predictorLMMmis* given by a quantile of absolute parametric bootstrap errors. Number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_B2_LMMmis`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) of *predictorLMMmis* given by a quantile of square roots of squared first-level bootstraped errors, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). Number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_1_LMMmis`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) of *predictorLMMmis* given by a quantile of square roots of squared first-level bootstraped errors, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). Number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in in *thetaFun*).
`estQAPE_db_telesc_LMMmis`	value/s of double-bootstrap estimator of QAPE (Quantile of Absolute Prediction Error) of *predictorLMMmis* given by a quantile of square roots of the sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error). Number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*).
`error1_LMMmis`	the matrix of first-level bootstrap errors of *predictorLMMmis*. Number of rows is equal to the number of predicted characteristics (declared in in *thetaFun*), number of columns is equal to *B1*.
`error2_LMMmis`	the list of matrices of second-level bootstrap errors of *predictorLMMmis*. The length of list is equal to the number of predicted characteristics (declared in in *thetaFun*), the number of rows of each matrix is equal to *B1*, the number of columns is equal to *B2*.
`corSquaredError1_db_B2_LMMmis`	the matrix of corrected squared first-level bootstrap errors of *predictorLMMmis* defined as doubled squared first-level bootstrap errors minus the mean of squared second-level bootstrap errors (computed for the approriate first-level

	bootstrap iterations). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_1_LMMmis`	the matrix of corrected squared first-level bootstrap errors of *predictorLMMmis* defined as doubled squared first-level bootstrap errors minus the squared second-level bootstrap error (computed once for each first-level bootstrap iteration). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_telesc_LMMmis`	the matrix of corrected squared first-level bootstrap errors of *predictorLMMmis* defined by elements from which the average given by equation (15) in Erciulescu and Fuller (2014) is counted. Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values can be negative.
`corSquaredError1_db_B2_WDZ_LMMmis`	the matrix of squared first-level bootstraped errors of *predictorLMMmis*, each corrected by the mean of squared second-level bootstraped errors based on B2 iterations (where correction is made only if their difference is non-negative). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.
`corSquaredError1_db_1_WDZ_LMMmis`	the matrix of squared first-level bootstraped errors of *predictorLMMmis*, each corrected by the squared second-level bootstraped error based on 1 iteration (where correction is made only if their difference is non-negative). Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.
`corSquaredError1_db_telesc_WDZ_LMMmis`	the matrix of corrected squared first-level bootstrap errors of *predictorLMMmis* defined by sums of the following elements: squared first-level bootstraped error, squared first-level bootstrap error for the next iteration and the opposite of second-level bootstraped error based on 1 iteration (but negative sums are replaced by squared first-level bootstraped error).Number of rows is equal to *B1*, the number of columns is equal to the number of predicted characteristics (declared in in *thetaFun*). Values are non-negative.
`positiveDefiniteEstGlev1`	logical indicating if the estimated covariance matrix of random effects, used to generate bootstrap realizations of the dependent variable at the first level of the double bootstrap, is positive definite.
`positiveDefiniteEstGlev2`	number of cases ouf of B1 with positive definite estimated covariance matrix of random effects used to generate bootstrap realizations of the dependent variable at the second level of the double bootstrap

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Erciulescu, A. L. and Fuller, W. A. (2013) Parametric Bootstrap Procedures for Small Area Prediction Variance. JSM 2014 - Survey Research Methods Section, 3307-3318.

2. Hall, P. and Maiti, T. (2006) On Parametric Bootstrap Methods for Small Area Prediction. Journal of the Royal Statistical Society. Series B, 68(2), 221-238.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)


data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part.mis <- '(1|NUTS4type)'
random.part <- '(1|NUTS2)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

### Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}
set.seed(123456)

predictorLMM <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictorLMM$thetaP

predictorLMMmis <- plugInLMM(YS, fixed.part, random.part.mis, reg,con,weights,backTrans,thetaFun)
predictorLMMmis$thetaP

### Estimation of prediction accuracy
# in the first column
# for the predictor of the value of the variable for population element no. 379,
# in the second column
# for the predictor of the value of the variable for population element no. 380:
doubleBootMis(predictorLMM, predictorLMMmis, 3, 3, c(0.5,0.9), 0.77)
```

```
#q=0.77 assumed as in Erciulescu and FUller (2014) eq. (17)

detach(invData2018)
```

`EBLUP` *Empirical Best Linear Unbiased Predictor*

Description

The function computes the value of the EBLUP of the linear combination of the variable of interest under the linear mixed model estimated using REML.

Usage

```
EBLUP(YS, fixed.part, random.part, reg, con, gamma, weights, estMSE)
```

Arguments

`YS`	values of the variable of interest observed in the sample.
`fixed.part`	fixed-effects terms declared as in *lmer* object.
`random.part`	random-effects terms declared as in *lmer* object.
`reg`	the population matrix of auxiliary variables named in *fixed.part* and *random.part*.
`con`	the population 0-1 vector with 1s for elements in the sample and 0s for elements which are not in the sample.
`gamma`	the population vector which transpose multiplied by the population vector of the variable of interest gives the predicted characteristic. For example, if *gamma* is the population vector of 1s, the sum of the values of the variable of interest in the whole dataset is predicted.
`weights`	the population vector of weights, defined as in *lmer* object, allowing to include heteroscedasticity of random components in the mixed linear model.
`estMSE`	logical. If TRUE, the naive MSE estimator and its components are computed.

Details

The function computes the value of the EBLUP of the linear combination of the variable of interest based on the formula (21) in Zadlo (2017) (see Remark 5.1 in the paper for further explanations). Predicted values for unsampled population elements in subsets for which random effects are not observed in the sample are computed based only on fixed effects. The naive MSE estimator of the EBLUP, which is the sum of two components given by equations (31) and (32) in Zadlo (2017) p. 8094, where unknown parameters are replaced by their REML estimates, is also computed. The naive MSE estimator ignores the variability of EBLUP resulting from the estimation of variance components.

Value

The function returns a list with the following objects:

`fixed.part`	the fixed part of the formula of model.
`random.part`	the random part of the formula of model.
`thetaP`	the value of the predictor.
`beta`	the estimated vector of fixed effects.
`Xbeta`	the product of two matrices: the population model matrix of auxiliary variables X and the estimated vector of fixed effects.
`sigma2R`	the estimated variance parameter of the distribution of random components.
`R`	the estimated covariance matrix of random components for sampled elements.
`G`	the estimated covariance matrix of random effects.
`model`	the formula of the model (as in *lmer* object).
`mEst`	*lmer* object with the estimated model.
`YS`	the sample vector of the variable of interest.
`reg`	the population matrix of auxiliary variables named in *fixed.part* and *random.part*.
`con`	the population 0-1 vector with 1s for elements in the sample and 0s for elements which are not in the sample.
`regS`	the sample matrix of auxiliary variables named in *fixed.part* and *random.part*.
`regR`	the matrix of auxiliary variables named in *fixed.part* and *random.part* for population elements which are not observed in the sample.
`gamma`	the population vector which transpose multiplied by the population vector of the variable of interest gives the predicted characteristic.
`gammaS`	the subvector of *gamma* for sampled elements.
`gammaR`	the subvector of *gamma* for population elements which are not observed in the sample.
`weights`	the population vector of weights, defined as in *lmer* object, allowing to include the heteroscedasticity of random components in the mixed linear model.
`Z`	the population model matrix of auxiliary variables associated with random effects.
`ZBlockNames`	labels of blocks of random effects in Z matrix.
`X`	the population model matrix of auxiliary variables associated with fixed effects.
`ZS`	the submatrix of Z matrix where the number of rows equals the number of sampled elements and the number of columns equals the number of estimated random effects.
`XR`	the submatrix of X matrix (with the same number of columns) for population elements which are not observed in the sample.
`ZR`	the submatrix of Z matrix where the number of rows equals the number of population elements which are not observed in the sample and the number of columns equals the number of estimated random effects.
`eS`	the sample vector of estimated random components.

`vS`	the estimated vector of random effects.
`g1`	the first component of the naive MSE estimator (computed if *estMSE = TRUE*).
`g2`	the second component of the naive MSE estimator (computed if *estMSE = TRUE*).
`neMSE`	the naive MSE estimator (computed if *estMSE = TRUE*).

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Henderson, C.R. (1950) Estimation of Genetic Parameters (Abstract). Annals of Mathematical Statistics 21, 309-310.
2. Royall, R.M. (1976) The Linear Least Squares Prediction Approach to Two-Stage Sampling. Journal of the American Statistical Association 71, 657-473.
3. Zadlo, T. (2017) On prediction of population and subpopulation characteristics for future periods, Communications in Statistics - Simulation and Computation 461(10), 8086-8104.

Examples

```
library(lme4)
library(Matrix)


### Prediction of the subpopulation mean based on the cross-sectional data

data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size
n <- 100 # sample size
# subpopulation of interest: NUTS4type==2
Nd <- sum(NUTS4type == 2) # subpopulation size

set.seed(123456)
sampled_elements <- sample(N,n)
con <- rep(0,N)
con[sampled_elements] <- 1 # elements in the sample
YS <- investments[sampled_elements]
fixed.part <- 'newly_registered'
random.part <- '(1| NUTS2)'
reg = invData2018[, -which(names(invData2018) == 'investments')]

gamma <- rep(0,N)
gamma[NUTS4type == 2] <- 1/Nd

weights <- rep(1,N) # homoscedastic random components
```

```
estMSE <- TRUE

# Predicted value of the mean in the following subpopulation: NUTS4type==2
EBLUP(YS, fixed.part, random.part, reg, con, gamma, weights, estMSE)$thetaP

# All results
EBLUP(YS, fixed.part, random.part, reg, con, gamma, weights, estMSE)

detach(invData2018)

##########################################################

### Prediction of the subpopulation total based on the longitudinal data

data(invData)
attach(invData)

N <- nrow(invData[(year == 2013),]) # population size in the first period
n <- 38 # sample size in the first period
# subpopulation and time period of interest: NUTS2 == '02' & year == 2018
# subpopulation size in the period of interest:
Ndt <- sum(NUTS2 == '02' & year == 2018)

set.seed(123456)
sampled_elements_in_2013 <- sample(N,n)
con2013 <- rep(0,N)
con2013[sampled_elements_in_2013] <- 1 # elements in the sample in 2013

# balanced panel sample - the same elements in all 6 periods:
con <- rep(con2013,6)

YS <- investments[con == 1]
fixed.part <- 'newly_registered'
random.part <- '(newly_registered  | NUTS4)'
reg <- invData[, -which(names(invData) == 'investments')]

gamma <- rep(0,nrow(invData))
gamma[NUTS2 == '02' & year == 2018] <- 1

weights <- rep(1,nrow(invData)) # homoscedastic random components
estMSE <- TRUE

# Predicted value of the total
# in the following subpopulation: NUTS4type == 2
# in the following time period: year == 2018
EBLUP(YS, fixed.part, random.part, reg, con, gamma, weights, estMSE)$thetaP

# All results
EBLUP(YS, fixed.part, random.part, reg, con, gamma, weights, estMSE)

detach(invData)
```

`ebpLMMne` *Empirical Best Predictor based on the nested error linear mixed model*

Description

The function computes the value of the EBP under the nested error linear mixed model estimated using REML assumed for possibly transformed variable of interest.

Usage

```
ebpLMMne(YS, fixed.part, division, reg, con, backTrans, thetaFun, L)
```

Arguments

`YS`	values of the variable of interest (already transformed if necessary) observed in the sample and used in the model as the dependent variable.
`fixed.part`	fixed-effects terms declared as in *lmer* object.
`division`	the variable dividing the population dataset into subsets (the nested error linear mixed model with 'division'-specific random components is estimated).
`reg`	the population matrix of auxiliary variables named in *fixed.part* and *division*.
`con`	the population 0-1 vector with 1s for elements in the sample and 0s for elements which are not in the sample.
`backTrans`	back-transformation function of the variable of interest (e.g. if YS is log-tranformed, then backTrans <- function(x) exp(x)).
`thetaFun`	the predictor function (e.g. mean or sd)
`L`	the number of iterations used to compute the value of the predictor.

Details

The function computes the value of the EBP based on the algorithm described in Molina and Rao (2010) in Section 4.

Value

The function returns a list with the following objects:

`thetaP`	the value/s of the predictor (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`fixed.part`	the fixed part of the formula of model.
`random.part`	the random part of the formula of model.
`division`	the variable dividing the population dataset into subsets (the nested error linear mixed model with 'division'-specific random components is estimated).
`thetaFun`	the function of the population values of the variable of interest (on the original scale) which defines at least one population or subpopulation characteristic to be predicted.

`backTrans`	back-transformation function of the variable of interest (e.g. if YS is log-tranformed, then backTrans <- function(x) exp(x).
`L`	the number of iterations used to compute the value of the predictor.
`beta`	the estimated vector of fixed effects.
`Xbeta`	the product of two matrices: the population model matrix of auxiliary variables X and the estimated vector of fixed effects.
`sigma2R`	the estimated variance parameter of the distribution of random components.
`R`	the estimated covariance matrix of random components for sampled elements.
`G`	the estimated covariance matrix of random effects.
`model`	the formula of the model (as in *lmer* object).
`mEst`	*lmer* object with the estimated model.
`YS`	values of the variable of interest (already transformed if necessary) observed in the sample and used in the model as the dependent variable.
`reg`	the population matrix of auxiliary variables named in *fixed.part* and *random.part*.
`con`	the population 0-1 vector with 1s for elements in the sample and 0s for elements which are not in the sample.
`regS`	the sample matrix of auxiliary variables named in *fixed.part* and *random.part*.
`regR`	the matrix of auxiliary variables named in *fixed.part* and *random.part* for unsampled population elements.
`weights`	the population vector of weigts, defined as in *lmer* object, allowing to include the heteroscedasticity of random components in the mixed linear model.
`Z`	the population model matrix of auxiliary variables associated with random effects.
`ZBlockNames`	labels of blocks of random effects in Z matrix.
`X`	the population model matrix of auxiliary variables associated with fixed effects.
`ZS`	the submatrix of Z matrix where the number of rows equals the number of sampled elements and the number of columns equals the number of estimated random effects.
`XR`	the submatrix of X matrix (with the same number of columns) for unsampled population elements.
`ZR`	the submatrix of Z matrix where the number of rows equals the number of unsampled population elements and the number of columns equals the number of estimated random effects.
`eS`	the sample vector of estimated random components.
`vS`	the estimated vector of random effects.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Chwila, A., Zadlo, T. (2022) On properties of empirical best predictors. Communications in Statistics - Simulation and Computation, 51(1), 220-253, https://doi.org/10.1080/03610918.2019.1649422
2. Molina, I., Rao, J.N.K. (2010) Small area estimation of poverty indicators. Canadian Journal of Statistics 38(3), 369-385.
3. Zadlo, T. (2017). On prediction of population and subpopulation characteristics for future periods, Communications in Statistics - Simulation and Computation 461(10), 8086-8104.

Examples

```
library(lme4)
library(Matrix)


### Prediction of the subpopulation median
### and the subpopulation standard deviation
### based on the cross-sectional data

data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size
n <- 100 # sample size

set.seed(123456)
sampled_elements <- sample(N,n)
con <- rep(0,N)
con[sampled_elements] <- 1 # elements in the sample
YS <- log(investments[sampled_elements]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
division <- 'NUTS2' # NUTS2-specific random effects are taken into account
reg <- invData2018[, -which(names(invData2018) == 'investments')]


# Characteristics to be predicted - the median and the standard deviation
# in the subpopulation of interest: NUTS4type==2
thetaFun <- function(x) {c(median(x[NUTS4type == 2]), sd(x[NUTS4type == 2]))}

L <- 5

# Predicted values of the median and the standard deviation
# in the following subpopulation: NUTS4type==2
set.seed(123456)
ebpLMMne(YS, fixed.part, division, reg, con, backTrans, thetaFun, L)$thetaP

set.seed(123456)
ebpLMMne(YS, fixed.part, division, reg, con, backTrans, thetaFun, L)
```

```
# All results
set.seed(123456)
str(ebpLMMne(YS, fixed.part, division, reg, con, backTrans, thetaFun, L))

detach(invData2018)

###########################################################

### Prediction of the subpopulation quartiles based on longitudinal data

data(invData)
attach(invData)

N <- nrow(invData[(year == 2013),]) # population size in the first period
n <- 38 # sample size in the first period

set.seed(123456)
sampled_elements_in_2013 <- sample(N,n)
con2013 <- rep(0,N)
con2013[sampled_elements_in_2013] <- 1 # elements in the sample in 2013

# balanced panel sample - the same elements in all 6 periods:
con <- rep(con2013,6)

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
division <- 'NUTS4' # NUTS4-specific random effects are taken into account
reg <- invData[, -which(names(invData) == 'investments')]
thetaFun <- function(x) {quantile(x[NUTS2 == '02' & year == 2018],probs = c(0.25,0.5,0.75))}

L <- 5

# Predicted values of quartiles
# in the following subpopulation: NUTS4type==2
# in the following time period: year==2018
set.seed(123456)
ebpLMMne(YS, fixed.part, division, reg, con, backTrans, thetaFun, L)$thetaP

set.seed(123456)
ebpLMMne(YS, fixed.part, division, reg, con, backTrans, thetaFun, L)

# All results
str(ebpLMMne(YS, fixed.part, division, reg, con, backTrans, thetaFun, L))


detach(invData)
```

EmpCM *Empirical covariance matrix of predicted random effects*

Description

A list of empirical covariance matrices of predicted random effects, where the length of the list equals the number of grouping variables used to define random effects as described in Carpenter, Goldstein and Rasbash (2003) in Section 3.2 and in Thai et al. (2013) in Section 2.3.3.

Usage

```
EmpCM(model)
```

Arguments

`model`	*lmer* object.

Value

a list of empirical covariance matrices of predicted random effects.

Author(s)

Tomasz Zadlo

References

1. Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

2. Thai, H.-T., Mentre, F., Holford, N.H., Veyrat-Follet, C. and Comets, E. (2013), A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. Pharmaceutical Statistics, 12, 129-140.

Examples

```
library(lme4)
data(invData)
attach(invData)
model <- lmer(investments ~ newly_registered + ((1|NUTS2) +
((newly_registered - 1)|NUTS2) + ((newly_registered)|NUTS4)))
EmpCM(model)
detach(invData)
```

EstCM *Estimated covariance matrix of predicted random effects*

Description

A list of estimated covariance matrices of predicted random effects, where the length of the list equals the number of grouping variables used to define random effects as described in Carpenter, Goldstein and Rasbash (2003) in Section 3.2 and in Thai et al. (2013) in Section 2.3.3.

Usage

```
EstCM(model)
```

Arguments

`model`	*lmer* object.

Value

a list of estimated covariance matrices of predicted random effects.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

2. Thai, H.-T., Mentre, F., Holford, N.H., Veyrat-Follet, C. and Comets, E. (2013), A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. Pharmaceutical Statistics, 12, 129-140.

Examples

```
library(lme4)
data(invData)
attach(invData)
model <- lmer(investments ~ newly_registered + ((1|NUTS2) +
((newly_registered - 1)|NUTS2) + ((newly_registered)|NUTS4)))
EstCM(model)
detach(invData)
```

`invData` *Population data - investments in Poland at NUTS 4 level*

Description

A data frame with 2280 observations on 6 variables presented below.

Arguments

`year`	year.
`NUTS4`	NUTS 4 code (powiats).
`NUTS2`	NUTS 2 code (voivodships).
`NUTS4type`	type of NUTS 4 (1 - land counties, 2 - city counties/cities with powiat status).
`investments`	investment outlays in millions PLN, in current prices; data concern Polish economic entities, including independent health care facilities and cultural institutions with legal personalities in which the number of employed persons exceeds 9 (source of data: Annual survey of the economic activity of enterprises conducted by Statistics Poland).
`newly_registered`	newly registered entities of the national economy recorded in the REGON register (in thousands).

Source

Statistics Poland, https://bdl.stat.gov.pl/eng

Examples

```
data(invData)
hist(invData$newly_registered[invData$year==2018])
boxplot(invData$investments~invData$year)
boxplot(invData$investments[invData$year==2018]~invData$NUTS2[invData$year==2018])
boxplot(invData$investments[invData$year==2018]~invData$NUTS4type[invData$year==2018])
```

`mcBootMis` *Monte Carlo simulation study of accuracy of estimators of accuracy measures*

Description

The function computes in the Monte Carlo simulation study values of accuracy measures of estimators of accuracy measures of two predictors under the model defined by the first of them.

Usage

```
mcBootMis(Ypop, predictorLMM, predictorLMMmis, K, B1, B2, p, q)
```

Arguments

`Ypop`	population values of the variable of interest (already transformed if necessary) which are used as the dependent variable in the population model.
`predictorLMM`	plugInLMM object, the predictor used to define the model assumed in the simulation study.
`predictorLMMmis`	plugInLMM object, the second predictor, the properties of which are assessed under the misspecified model used in *predictorLMM*.
`K`	the number of Monte Carlo iterations.
`B1`	the number of first-level bootstrap iterations.
`B2`	the number of second-level bootstrap iterations.
`p`	orders of quantiles in the QAPE.
`q`	estimator bounds assumed for *estMSE_db_1_EF* and *estMSE_db_telesc_EF* (which are corrected versions of *estMSE_db_1* and *estMSE_db_telesc*, respectively).

Details

In the model-based simulation study population values of the dependent variable are generated based on the (possibly transformed) Linear Mixed Model used in *predictorLMM* and the accuracy of predictors *predictorLMM* and *predictorLMMmis* is assessed. What is more, the the accuracy of parametric, residual and double bootstrap estimators of accuracy measures is studied under the model used in *predictorLMM*. Values of some MSE estimators can be negative, the number of negative values of MSE estimators obtained in the simulation study are presented in objects *neg_estMSE_LMM* and *neg_estMSE_LMMmis*. Hence, some RMSE estimators computed as square roots of MSE estimators can produce NaNs - see warnings.

Value

`QAPElmm`	value/s of the QAPE of *predictorLMM* assessed in the Monte Carlo study - the number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).
`RMSElmm`	value/s of the RMSE of *predictorLMM* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSElmm`	value/s of the rRMSE (in percentages) of *predictorLMM* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rBlmm`	value/s of the relative bias (in percentages) of *predictorLMM* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).

`QAPElmmMis`	value/s of the QAPE of *predictorLMM2* assessed in the Monte Carlo study - the number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).
`RMSElmmMis`	value/s of the RMSE of *predictorLMM2* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSElmmMis`	value/s of the rRMSE (in percentages) of *predictorLMM2* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rBlmmMis`	value/s of the relative bias (in percentages) of *predictorLMMmis* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estRMSE_rbF_LMM`	relative bias (in percentages) of estimated value/s of RMSE of *predictorLMM* without correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSE.estRMSE_rbF_LMM`	relative RMSE (in percentages) of estimated value/s of RMSE of *predictorLMM* without correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estRMSE_rbF_LMMmis`	relative bias (in percentages) of estimated value/s of RMSE of *predictorLMMmis* without correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSE.estRMSE_rbF_LMMmis`	relative RMSE (in percentages) of estimated value/s of RMSE of *predictorLMMmis* without correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estMSE_rbF_LMM`	relative bias (in percentages) of estimated value/s of MSE of *predictorLMM* without correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSE.estMSE_rbF_LMM`	relative RMSE (in percentages) of estimated value/s of MSE of *predictorLMM* without correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estMSE_rbF_LMMmis`	relative bias (in percentages) of estimated value/s of MSE of *predictorLMMmis* without correction to avoid the problem of underdispersion of residual bootstrap

distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).

`rRMSE.estMSE_rbF_LMMmis`
relative RMSE (in percentages) of estimated value/s of MSE of *predictorLMMmis* without correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).

`rB.estQAPE_rbF_LMM`
relative bias (in percentages) of estimated value/s of QAPE of *predictorLMM* without correction to avoid the problem of underdispersion of residual bootstrap distributions, the number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).

`rRMSE.estQAPE_rbF_LMM`
relative RMSE (in percentages) of estimated value/s of QAPE of *predictorLMM* without correction to avoid the problem of underdispersion of residual bootstrap distributions, the number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).

`rB.estQAPE_rbF_LMMmis`
relative bias (in percentages) of estimated value/s of QAPE of *predictorLMMmis* without correction to avoid the problem of underdispersion of residual bootstrap distributions, the number of rows is equal to the number of orders of quantiles to be considered (declared in *p*), the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).

`rRMSE.estQAPE_rbF_LMMmis`
relative RMSE (in percentages) of estimated value/s of QAPE of *predictorLMMmis* without correction to avoid the problem of underdispersion of residual bootstrap distributions, the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*) .

`rB.estRMSE_rbT_LMM`
relative bias (in percentages) of estimated value/s of RMSE of *predictorLMM* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).

`rRMSE.estRMSE_rbT_LMM`
relative RMSE (in percentages) of estimated value/s of RMSE of *predictorLMM* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).

`rB.estRMSE_rbT_LMMmis`
relative bias (in percentages) of estimated value/s of RMSE of *predictorLMMmis* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).

`rRMSE.estRMSE_rbT_LMMmis`
relative RMSE (in percentages) of estimated value/s of RMSE of *predictorLMMmis* with correction to avoid the problem of underdispersion of residual boot-

	strap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estMSE_rbT_LMM`	relative bias (in percentages) of estimated value/s of MSE of *predictorLMM* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSE.estMSE_rbT_LMM`	relative RMSE (in percentages) of estimated value/s of MSE of *predictorLMM* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estMSE_rbT_LMMmis`	relative bias (in percentages) of estimated value/s of MSE of *predictorLMMmis* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSE.estMSE_rbT_LMMmis`	relative RMSE (in percentages) of estimated value/s of MSE of *predictorLMMmis* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estQAPE_rbT_LMM`	relative bias (in percentages) of estimated value/s of QAPE of *predictorLMM* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSE.estQAPE_rbT_LMM`	relative RMSE (in percentages) of estimated value/s of QAPE of *predictorLMM* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rB.estQAPE_rbT_LMMmis`	relative bias (in percentages) of estimated value/s of QAPE of *predictorLMMmis* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSE.estQAPE_rbT_LMMmis`	relative RMSE (in percentages) of estimated value/s of QAPE of *predictorLMMmis* with correction to avoid the problem of underdispersion of residual bootstrap distributions (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`neg_estMSE_LMM`	the number of negative values of MSE estimators of *predictorLMM* obtained in the simulaton study out of K iterations, the number of rows is equal to 10 - the number of considered parametric and double bootstrap MSE estimators, the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).

`neg_estMSE_LMMmis`
the number of negative values of MSE estimators of *predictorLMMmis* obtained in the simulaton study out of K iterations, the number of rows is equal to 10 - the number of considered parametric and double bootstrap MSE estimators, the number of columns is equal to the number of predicted characteristics (declared in *thetaFun*).

`rB.estMSE_param_LMMmis`
relative bias (in percentages) of *estMSE_param* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estMSE_param_LMMmis`
relative RMSE (in percentages) of *estMSE_param* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rB.estMSE_db_B2_LMMmis`
relative bias (in percentages) of *estMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estMSE_db_B2_LMMmis`
relative RMSE (in percentages) of *estMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rB.estMSE_db_B2_WDZ_LMMmis`
relative bias (in percentages) of *estMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estMSE_db_B2_WDZ_LMMmis`
relative RMSE (in percentages) of *estMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rB.estMSE_db_B2_HM_LMMmis`
relative bias (in percentages) of *estMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estMSE_db_B2_HM_LMMmis`
relative RMSE (in percentages) of *estMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rB.estMSE_db_1_LMMmis`
relative bias (in percentages) of *estMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estMSE_db_1_LMMmis`
relative RMSE (in percentages) of *estMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rB.estMSE_db_1_WDZ_LMMmis`
relative bias (in percentages) of *estMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estMSE_db_1_WDZ_LMMmis`
relative RMSE (in percentages) of *estMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rB.estMSE_db_1_EF_LMMmis`
relative bias (in percentages) of *estMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estMSE_db_1_EF_LMMmis`	relative RMSE (in percentages) of *estMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estMSE_db_telesc_LMMmis`	relative bias (in percentages) of *estMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estMSE_db_telesc_LMMmis`	relative RMSE (in percentages) of *estMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estMSE_db_telesc_WDZ_LMMmis`	relative bias (in percentages) of *estMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estMSE_db_telesc_WDZ_LMMmis`	relative RMSE (in percentages) of *estMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estMSE_db_telesc_EF_LMMmis`	relative bias (in percentages) of *estMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estMSE_db_telesc_EF_LMMmis`	relative RMSE (in percentages) of *estMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_param_LMMmis`	relative bias (in percentages) of *estRMSE_param* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_param_LMMmis`	relative RMSE (in percentages) of *estRMSE_param* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_B2_LMMmis`	relative bias (in percentages) of *estRMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_B2_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_B2_WDZ_LMMmis`	relative bias (in percentages) of *estRMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_B2_WDZ_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_B2_HM_LMMmis`	relative bias (in percentages) of *estRMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_B2_HM_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rB.estRMSE_db_1_LMMmis`	relative bias (in percentages) of *estRMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_1_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_1_WDZ_LMMmis`	relative bias (in percentages) of *estRMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_1_WDZ_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_1_EF_LMMmis`	relative bias (in percentages) of *estRMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_1_EF_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_telesc_LMMmis`	relative bias (in percentages) of *estRMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_telesc_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_telesc_WDZ_LMMmis`	relative bias (in percentages) of *estRMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_telesc_WDZ_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estRMSE_db_telesc_EF_LMMmis`	relative bias (in percentages) of *estRMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estRMSE_db_telesc_EF_LMMmis`	relative RMSE (in percentages) of *estRMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estQAPE_param_LMMmis`	relative bias (in percentages) of *estQAPE_param* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estQAPE_param_LMMmis`	relative RMSE (in percentages) of *estQAPE_param* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estQAPE_db_B2_LMMmis`	relative bias (in percentages) of *estQAPE_db_B2* estimator (see *doubleBoot* function) of *predictorLMMmis*.

`rRMSE.estQAPE_db_B2_LMMmis`	relative RMSE (in percentages) of *estQAPE_db_B2* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estQAPE_db_1_LMMmis`	relative bias (in percentages) of *estQAPE_db_1* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estQAPE_db_1_LMMmis`	relative RMSE (in percentages) of *estQAPE_db_1* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estQAPE_db_telesc_LMMmis`	relative bias (in percentages) of *estQAPE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rRMSE.estQAPE_db_telesc_LMMmis`	relative RMSE (in percentages) of *estQAPE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMMmis*.
`rB.estMSE_param_LMM`	relative bias (in percentages) of *estMSE_param* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estMSE_param_LMM`	relative RMSE (in percentages) of *estMSE_param* estimator (see *doubleBoot* function) of *predictorLMM*.
`rB.estMSE_db_B2_LMM`	relative bias (in percentages) of *estMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estMSE_db_B2_LMM`	relative RMSE (in percentages) of *estMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMM*.
`rB.estMSE_db_B2_WDZ_LMM`	relative bias (in percentages) of *estMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estMSE_db_B2_WDZ_LMM`	relative RMSE (in percentages) of *estMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.
`rB.estMSE_db_B2_HM_LMM`	relative bias (in percentages) of *estMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estMSE_db_B2_HM_LMM`	relative RMSE (in percentages) of *estMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMM*.
`rB.estMSE_db_1_LMM`	relative bias (in percentages) of *estMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estMSE_db_1_LMM`	relative RMSE (in percentages) of *estMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estMSE_db_1_WDZ_LMM`
relative bias (in percentages) of *estMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estMSE_db_1_WDZ_LMM`
relative RMSE (in percentages) of *estMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estMSE_db_1_EF_LMM`
relative bias (in percentages) of *estMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estMSE_db_1_EF_LMM`
relative RMSE (in percentages) of *estMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estMSE_db_telesc_LMM`
relative bias (in percentages) of *estMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estMSE_db_telesc_LMM`
relative RMSE (in percentages) of *estMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estMSE_db_telesc_WDZ_LMM`
relative bias (in percentages) of *estMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estMSE_db_telesc_WDZ_LMM`
relative RMSE (in percentages) of *estMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estMSE_db_telesc_EF_LMM`
relative bias (in percentages) of *estMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estMSE_db_telesc_EF_LMM`
relative RMSE (in percentages) of *estMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_param_LMM`
relative bias (in percentages) of *estRMSE_param* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_param_LMM`
relative RMSE (in percentages) of *estRMSE_param* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_B2_LMM`
relative bias (in percentages) of *estRMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_B2_LMM`
relative RMSE (in percentages) of *estRMSE_db_B2* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_B2_WDZ_LMM`
relative bias (in percentages) of *estRMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_B2_WDZ_LMM`
relative RMSE (in percentages) of *estRMSE_db_B2_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_B2_HM_LMM`
relative bias (in percentages) of *estRMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_B2_HM_LMM`
relative RMSE (in percentages) of *estRMSE_db_B2_HM* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_1_LMM`
relative bias (in percentages) of *estRMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_1_LMM`
relative RMSE (in percentages) of *estRMSE_db_1* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_1_WDZ_LMM`
relative bias (in percentages) of *estRMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_1_WDZ_LMM`
relative RMSE (in percentages) of *estRMSE_db_1_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_1_EF_LMM`
relative bias (in percentages) of *estRMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_1_EF_LMM`
relative RMSE (in percentages) of *estRMSE_db_1_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_telesc_LMM`
relative bias (in percentages) of *estRMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_telesc_LMM`
relative RMSE (in percentages) of *estRMSE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_telesc_WDZ_LMM`
relative bias (in percentages) of *estRMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_telesc_WDZ_LMM`
relative RMSE (in percentages) of *estRMSE_db_telesc_WDZ* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estRMSE_db_telesc_EF_LMM`
relative bias (in percentages) of *estRMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rRMSE.estRMSE_db_telesc_EF_LMM`
relative RMSE (in percentages) of *estRMSE_db_telesc_EF* estimator (see *doubleBoot* function) of *predictorLMM*.

`rB.estQAPE_param_LMM`	relative bias (in percentages) of *estQAPE_param* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estQAPE_param_LMM`	relative RMSE (in percentages) of *estQAPE_param* estimator (see *doubleBoot* function) of *predictorLMM*.
`rB.estQAPE_db_B2_LMM`	relative bias (in percentages) of *estQAPE_db_B2* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estQAPE_db_B2_LMM`	relative RMSE (in percentages) of *estQAPE_db_B2* estimator (see *doubleBoot* function) of *predictorLMM*.
`rB.estQAPE_db_1_LMM`	relative bias (in percentages) of *estQAPE_db_1* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estQAPE_db_1_LMM`	relative RMSE (in percentages) of *estQAPE_db_1* estimator (see *doubleBoot* function) of *predictorLMM*.
`rB.estQAPE_db_telesc_LMM`	relative bias (in percentages) of *estQAPE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMM*.
`rRMSE.estQAPE_db_telesc_LMM`	relative RMSE (in percentages) of *estQAPE_db_telesc* estimator (see *doubleBoot* function) of *predictorLMM*.
`MCpositiveDefiniteEstGlev1`	number of cases ouf of K with postive definite estimated covariance matrix of random effects used to generate bootstrap realizations of the dependent variable at the first level of the double bootstrap.
`MCpositiveDefiniteEstGlev2`	number of cases ouf of K*B1 with positive definite estimated covariance matrix of random effects used to generate bootstrap realizations of the dependent variable at the second level of the double bootstrap.

Author(s)

Tomasz Zadlo

References

1. Chatterjee, S., Lahiri, P. Li, H. (2008) Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models, Annals of Statistics, Vol. 36 (3), pp. 1221?1245.

2. Rao, J.N.K. and Molina, I. (2015) Small Area Estimation. Second edition, John Wiley & Sons, New Jersey.

3. Zadlo T. (2017), On asymmetry of prediction errors in small area estimation, Statistics in Transition, 18 (3), 413-432.

Examples

```
library(lme4)
library(Matrix)
library(mvtnorm)
library(matrixcalc)
library(qape)

data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size

con <- rep(1,N)
con[c(379,380)] <- 0 # last two population elements are not observed

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'
random.part.mis <- '(1|NUTS4type)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted:
# values of the variable for last two population elements
thetaFun <- function(x) {x[c(379,380)]}

predictorLMM <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictorLMMmis <- plugInLMM(YS, fixed.part, random.part.mis, reg, con,weights,backTrans,thetaFun)

Ypop <- log(invData2018$investments)

# Monte Carlo simulation study under the model defined in predictorLMM
# correctly specified for predictorLMM and misspecified for predictorLMMmis
set.seed(211)
mcBootMis(Ypop, predictorLMM, predictorLMMmis, 2, 2, 2, c(0.5, 0.9), 0.77)

detach(invData2018)
```

`mcLMMmis` *Monte Carlo simuation study of accuracy of predictors under the misspecified model*

Description

The function computes in the Monte Carlo simulation study values of accuracy measures of three predictors under the model assumed for one of them with possible modifications of covariance matrices of random effects and random components.

Usage

```
mcLMMmis(Ypop, predictorLMMmis, predictorLMM, predictorLMM2, K, p, ratioR, ratioG)
```

Arguments

`Ypop`	population values of the variable of interest (already transformed if necessary) which are used as the dependent variable in the population model.
`predictorLMMmis`	plugInLMM object, the predictor used to define the model assumed in the simulation study.
`predictorLMM`	plugInLMM object, the first predictor, the accuracy of which is assessed in the simulation study.
`predictorLMM2`	plugInLMM object, the second predictor, the accuracy of which is assessed in the simulation study.
`K`	the number of Monte Carlo iterations.
`p`	orders of quantiles in the QAPE.
`ratioR`	the value by which the diagonal elements of the covariance matrix of random components of the model based on the whole population data and formulation used in *predictorLMMmis* are divided. Then, the corrected covariance matrix is used to generate bootstrap realizations of random components.
`ratioG`	the value by which the diagonal elements of the covariance matrix of random effects of the model based on the whole population data and formulation used in *predictorLMMmis* are divided. Then, the corrected covariance matrix, assuming that it is positive definite, is used to generate bootstrap realizations of random effects. If it is not positive definite, the alert is printed and the dependent variable is generated based on the model without random effects.

Details

In the model-based simulation study population values of the dependent variable are generated based on the (possibly transformed) Linear Mixed Model used in *predictorLMMmis* with possibly modified covariance matrices of random effects and random components by the usage of *ratioR* and *ratioG* arguments. In the simulation study accuracy of predictors *predictorLMM* and *predictorLMM2* is assessed. Although, all the predictors are *plugInLMM* objects, it should be noted that under the non-transformed Linear Mixed Model and in the case of the prediction of the linear combination of the dependent variable (e.g. the mean, the total, and one realization of the variable), the predictors are Empirical Best Linear Unbiased Predictors. What is more, if *predictorLMMmis* is defined as *predictorLMM*, the Monte Carlo simulation study of accuracy of *predictorLMM* under correctly specified model and of *predictorLMM2* under misspecified model is conducted.

Value

`errorLMM`	Monte Carlo prediction errors of *predictorLMM* - number of rows is equal to the number of predicted characteristics (declared in *thetaFun*), number of columns is equal to *K*.
`errorLMM2`	Monte Carlo prediction errors of *predictorLMM2* - number of rows is equal to the number of predicted characteristics (declared in *thetaFun*), number of columns is equal to *K*.
`QAPElmm`	value/s of the QAPE of *predictorLMM* assessed in the Monte Carlo study - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in *thetaFun*).
`RMSElmm`	value/s of the RMSE of *predictorLMM* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSElmm`	value/s of the rRMSE (in percentages) of *predictorLMM* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rBlmm`	value/s of the relative bias (in percentages) of *predictorLMM* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`QAPElmm2`	value/s of the QAPE of *predictorLMM2* assessed in the Monte Carlo study - number of rows is equal the number of orders of quantiles to be considered (declared in *p*), number of columns is equal the number of predicted characteristics (declared in *thetaFun*).
`RMSElmm2`	value/s of the RMSE of *predictorLMM2* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rRMSElmm2`	value/s of the rRMSE (in percentages) of *predictorLMM2* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`rBlmm2`	value/s of the relative bias (in percentages) of *predictorLMM2* assessed in the Monte Carlo study (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`positiveDefiniteEstG`	logical indicating if the estimated covariance matrix of random effects, used to generate Monte Carlo realizations of the dependent variable, is positive definite.

Author(s)

Tomasz Zadlo

References

1. Chatterjee, S., Lahiri, P. Li, H. (2008) Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models, Annals of Statistics, Vol. 36 (3),

1221-1245.

2. Rao, J.N.K. and Molina, I. (2015) Small Area Estimation. Second edition, John Wiley & Sons, New Jersey.

3. Zadlo T. (2017), On asymmetry of prediction errors in small area estimation, Statistics in Transition, 18 (3), 413-432.

Examples

```
data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]

invData2018$investments <- invData2018$investments/1000
attach(invData2018)
N <- nrow(invData2018) # population size

con <- rep(0,N)
set.seed(123456)
con[sample(N,50)] <- 1 # sample size equals 50

YS <- log((investments[con == 1])) # log-transformed values
backTrans <- function(x) {exp(x)} # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(1|NUTS2)'
random.part2 <- '(1|NUTS4type)'

reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components
weights.mis <- sqrt(newly_registered)

# Characteristics to be predicted:
# the population mean and the population total
thetaFun <- function(x) {c(mean(x), median(x))}

predictorLMMmis <- plugInLMM(YS, fixed.part, random.part, reg,con,weights.mis,backTrans,thetaFun)
predictorLMMmis$thetaP

predictorLMM <- plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
predictorLMM$thetaP

predictorLMM2 <- plugInLMM(YS, fixed.part, random.part2, reg, con, weights, backTrans, thetaFun)
predictorLMM2$thetaP

Ypop <- log(invData2018$investments)

# Monte Carlo simulation study under the misspecified model defined in predictorLMMmis
# with modified covariance matrices R and G
set.seed(123456)
mcLMMmis(Ypop, predictorLMMmis, predictorLMM, predictorLMM2, 5, c(0.75,0.9), 2, 0.1)
```

```
# Monte Carlo simulation study under the model defined in predictorLMM
# correctly specified for predictorLMM and misspecified for predictorLMM2
set.seed(123456)
mcLMMmis(Ypop, predictorLMM, predictorLMM, predictorLMM2, 5, c(0.75,0.9), 1, 1)

detach(invData2018)
```

`modifyDataset` *Modification of the values of the variables in the dataset*

Description

The function modifies the values of the declared variables used in the random part of the model if they are not unique. Unique values of the variables are required to build correct Z matrix for unsampled population elements.

Usage

```
modifyDataset(data, names)
```

Arguments

`data`	the population dataset.
`names`	the vector of names of the dataset columns which values should be modified (names of the variables used to define the random part of the model).

Value

The dataset with modified values of the declared variables.

Author(s)

Tomasz Zadlo

Examples

```
data(realestData)
# some values of "NUTS2" and "NUTS4type" are the same - we will modify them:
modifyDataset(realestData, c("NUTS2", "NUTS4type"))
```

`normCholTest` *Test of normality of the dependent variable*

Description

The function conducts a test of normality of the depenedent variable based on residuals transformed using Cholesky decomposition of the inverse of the estimated variance-covariance matrix of the variable.

Usage

```
normCholTest(model, normTest)
```

Arguments

`model`	lmer object.
`normTest`	function which implements a normality test e.g. shapiro.test (takes a vector of the values of the variable as an argument and conducts a test of normality of the variable).

Value

`testResults`	output of the normTest function chosen by the user.

Author(s)

Tomasz Zadlo

Examples

```
library(lme4)
mod <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
normCholTest(mod, shapiro.test)
```

`plugInLMM` *PLUG-IN predictor based on the linear mixed model*

Description

The function computes the value of the plug-in predictor under the linear mixed model estimated using REML assumed for possibly transformed variable of interest.

Usage

```
plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)
```

Arguments

`YS`	values of the variable of interest (already transformed if necessary) observed in the sample and used in the model as the dependent variable.
`fixed.part`	fixed-effects terms declared as in *lmer* object.
`random.part`	random-effects terms declared as in *lmer* object.
`reg`	the population matrix of auxiliary variables named in *fixed.part* and *random.part*.
`con`	the population 0-1 vector with 1s for elements in the sample and 0s for elements which are not in the sample.
`weights`	the population vector of weights, defined as in *lmer* object, allowing to include the heteroscedasticity of random components in the mixed linear model.
`backTrans`	back-transformation function of the variable of interest (e.g. if YS is log-tranformed, then backTrans <- function(x) exp(x)).
`thetaFun`	the predictor function (e.g. mean or sd).

Details

The function computes the value of the plug-in estimator in two steps as presented by Chwila and Zadlo (2019) p. 20. Firstly, we build the population vector consisting of real values of the variable of interest for sampled elements and (possibly back-transformed) fitted values of the variable of interest based on the estimated model. Secondly, the value/s of *thetaFun* based on the population vector built in the first step is/are computed. Predicted values for unsampled population elements in subsets for which random effects are not observed in the sample are computed based only on fixed effects.

Value

The function returns a list with the following objects:

`thetaP`	the value/s of the predictor (more than one value is computed if in *thetaFun* more than one population characteristic is defined).
`fixed.part`	the fixed part of the formula of model.
`random.part`	the random part of the formula of model.
`thetaFun`	the function of the population values of the variable of interest (on the original scale) which defines at least one population or subpopulation characteristic to be predicted.
`backTrans`	back-transformation function of the variable of interest (e.g. if YS used in the model is log-tranformed, then backTrans <- function(x) exp(x)).
`YP`	predicted values of the variable of interest for unsampled elements (without back-tranformation).
`YbackTrans`	population vector of the values of the variable of interest on the orignal scale for sampled elements and back-transformed predicted values of the variable of interest for unsampled elements.
`YPbackTrans`	back-transformed predicted values of the variable of interest for unsampled elements.

`beta`	the estimated vector of fixed effects.
`Xbeta`	the product of two matrices: the population model matrix of auxiliary variables X and the estimated vector of fixed effects.
`sigma2R`	the estimated variance parameter of the distribution of random components.
`R`	the estimated covariance matrix of random components for sampled elements.
`G`	the estimated covariance matrix of random effects.
`model`	the formula of the model (as in *lmer* object).
`mEst`	*lmer* object with the estimated model.
`YS`	values of the variable of interest (already transformed if necessary) observed in the sample and used in the model as the dependent variable.
`reg`	the population matrix of auxiliary variables named in *fixed.part* and *random.part*.
`con`	the population 0-1 vector with 1s for elements in the sample and 0s for elements which are not in the sample.
`regS`	the sample matrix of auxiliary variables named in *fixed.part* and *random.part*.
`regR`	the matrix of auxiliary variables named in *fixed.part* and *random.part* for unsampled population elements.
`weights`	the population vector of weigts, defined as in *lmer* object, allowing to include the heteroscedasticity of random components in the mixed linear model.
`Z`	the population model matrix of auxiliary variables associated with random effects.
`ZBlockNames`	labels of blocks of random effects in Z matrix.
`ZS`	the submatrix of Z matrix where the number of rows equals the number of sampled elements and the number of columns equals the number of estimated random effects.
`XR`	the submatrix of X matrix (with the same number of columns) for unsampled population elements.
`ZR`	the submatrix of Z matrix where the number of rows equals the number of unsampled population elements and the number of columns equals the number of estimated random effects.
`eS`	the sample vector of estimated random components.
`vS`	the estimated vector of random effects.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

Chwila, A., Zadlo, T. (2022) On properties of empirical best predictors. Communications in Statistics - Simulation and Computation, 51(1), 220-253, https://doi.org/10.1080/03610918.2019.1649422

Examples

```
library(lme4)
library(Matrix)


### Prediction of the subpopulation median
### and the subpopulation standard deviation
### based on the cross-sectional data

data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)

N <- nrow(invData2018) # population size
n <- 100 # sample size

set.seed(123456)
sampled_elements <- sample(N,n)
con <- rep(0,N)
con[sampled_elements] <- 1 # elements in the sample
YS <- log(investments[sampled_elements]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(log(newly_registered)  | NUTS2)'
reg <- invData2018[, -which(names(invData2018) == 'investments')]
weights <- rep(1,N) # homoscedastic random components

# Characteristics to be predicted - the median and the standard deviation
# in following subpopulation: NUTS4type == 2
thetaFun <- function(x) {c(median(x[NUTS4type == 2]),sd(x[NUTS4type == 2]))}

# Predicted values of the median and the standard deviation
# in the following subpopulation: NUTS4type == 2

plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)$thetaP
plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)

# All results
str(plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun))

detach(invData2018)

##########################################################

### Prediction of the subpopulation quartiles based on longitudinal data

data(invData)
attach(invData)

N <- nrow(invData[(year == 2013),]) # population size in the first period
n <- 38 # sample size in the first period
```

```
# subpopulation and time period of interest: NUTS2 == '02' & year == 2018
Ndt=sum(NUTS2=='02' & year==2018) # subpopulation size in the period of interest

set.seed(123456)
sampled_elements_in_2013 <- sample(N,n)
con2013 <- rep(0,N)
con2013[sampled_elements_in_2013] <- 1 # elements in the sample in 2013

# balanced panel sample - the same elements in all 6 periods:
con <- rep(con2013,6)

YS <- log(investments[con == 1]) # log-transformed values
backTrans <- function(x) exp(x) # back-transformation of the variable of interest
fixed.part <- 'log(newly_registered)'
random.part <- '(0 + log(newly_registered) | NUTS4)'
reg <- invData[, -which(names(invData) == 'investments')]
weights <- rep(1,nrow(invData)) # homoscedastic random components

# Characteristics to be predicted - quartiles in 2018
# in the following subpopulation: NUTS4type == 2
thetaFun <- function(x) {quantile(x[NUTS2 == '02' & year == 2018],probs = c(0.25,0.5,0.75))}

# Predicted values of quartiles
# in the following subpopulation: NUTS4type == 2
# in the following time period: year == 2018
plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)$thetaP
plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun)

# All results
str(plugInLMM(YS, fixed.part, random.part, reg, con, weights, backTrans, thetaFun))

detach(invData)
```

`print.EBLUP` *print the value of EBLUP predictor*

Description

Print the value of EBLUP predictor.

Usage

```
## S3 method for class 'EBLUP'
print(x, ...)
```

Arguments

`x`	the object of class 'EBLUP'.
`...`	not used.

Author(s)

Alicja Wolny-Dominiak

`print.ebpLMMne` *print the value of ebpLMMne predictor*

Description

Print the value of ebpLMMne predictor.

Usage

```
## S3 method for class 'ebpLMMne'
print(x, ...)
```

Arguments

`x`	the object of class 'ebpLMMne'.
`...`	not used.

Author(s)

Alicja Wolny-Dominiak

`print.plugInLMM` *print the value of plugInLMM predictor*

Description

Print the value of plugInLMM predictor.

Usage

```
## S3 method for class 'plugInLMM'
print(x, ...)
```

Arguments

`x`	the object of class 'plugInLMM'.
`...`	not used.

Author(s)

Alicja Wolny-Dominiak

`quantileNaN` *quantile NaN*

Description

The function returns NaN when one of its arguments is NaN (instead of the error returned in this case by the classic quantile function)

Usage

```
quantileNaN(x, probs)
```

Arguments

`x`	numeric vector whose sample quantiles are wanted.
`probs`	numeric vector of probabilities with values in [0,1].

Author(s)

Tomasz Zadlo

`realestData` *Population data - real estate in Poland at NUTS 4 level*

Description

A data frame with 1504 observations on the following 7 variables (NUTS 4 units with masked values of the variables due to Statistical confidentiality has been removed).

Arguments

`year`	year.
`NUTS4`	NUTS 4 code (powiats).
`NUTS2`	NUTS 2 code (voivodships).
`NUTS4type`	type of NUTS 4 (1 - land counties, 2 - city counties/cities with powiat status).
`premises`	number of residential premises sold in market transactions (in thousands).
`area`	usable floor area of residential premises sold in market transactions (in millions of square meters).
`price`	sum of prices of residential premises sold (in billions of Polish zloty).

Source

Statitics Poland, https://bdl.stat.gov.pl/eng

Examples

```
data(realestData)
hist(realestData$price[realestData$year==2018])
boxplot(realestData$price~realestData$year)
boxplot(realestData$price[realestData$year==2018]~realestData$NUTS2[realestData$year==2018])
boxplot(realestData$price[realestData$year==2018]~realestData$NUTS4type[realestData$year==2018])


library(lme4)
attach(realestData)

N <- nrow(realestData[(year == 2015),]) # population size in the first period
n <- 75 # sample size in the first period
set.seed(123456)
sampled_elements_in_2015 <- sample(N,n)
con2015 <- rep(0,N)
con2015[sampled_elements_in_2015] <- 1

sampled_elements_in_2016 <- sample(N,n)
con2016 <- rep(0,N)
con2016[sampled_elements_in_2016] <- 1

sampled_elements_in_2017 <- sample(N,n)
con2017 <- rep(0,N)
con2017[sampled_elements_in_2017] <- 1

sampled_elements_in_2018 <- sample(N,n)
con2018 <- rep(0,N)
con2018[sampled_elements_in_2018] <- 1

con=as.logical(con2015, con2016, con2017, con2018)

model1 <- lmer(price ~ premises + area + (1|NUTS2)+(0+premises|NUTS2) +
(1|NUTS4type)+(0+area|NUTS4type), subset=con)
AIC(model1)
model2 <- lmer(price ~ premises + area + (0+premises|NUTS2) + (0+area|NUTS4type), subset = con)
AIC(model2)
```

---

srswrRe *Bootstrap sample of predicted random effects*

Description

The function draws at random a simple random sample with replacement from predicted random effects, where the sample size is equal the number of random effects in the whole population.

Usage

```
srswrRe(listRanef, reg)
```

Arguments

`listRanef`	*ranef(model)* object where *model* is an *lmer* object.
`reg`	the population matrix of auxiliary variables named in *fixed.part* and *random.part*.

Value

`tablsrswrRe`	a vector of a simple random sample with replacement from predicted random effects, where the sample size is equal the number of random effects in the whole population.
`lsrswrRe`	a list of length equal the number of grouping variables taken into account in the random part of the model. Each list consists of 4 sublists: $raneftotal - a vector of a simple random sample with replacement from all predicted random effects under the cosidered grouping variable, $ranefname - a name of the grouping variable, $k - the number of random effects under the considered grouping variable, $df - a data frame of predicted random effects under the considered grouping variable, $dfsamp - a data frame of a simple random sample with replacement from predicted random effects under the considered grouping variable.

Author(s)

Alicja Wolny-Dominiak, Tomasz Zadlo

References

1. Carpenter, J.R., Goldstein, H. and Rasbash, J. (2003), A novel bootstrap procedure for assessing the relationship between class size and achievement. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 431-443.

2. Chambers, R. and Chandra, H. (2013) A Random Effect Block Bootstrap for Clustered Data, Journal of Computational and Graphical Statistics, 22(2), 452-470.

3. Thai, H.-T., Mentre, F., Holford, N.H., Veyrat-Follet, C. and Comets, E. (2013), A comparison of bootstrap approaches for estimating uncertainty of parameters in linear mixed-effects models. Pharmaceutical Statistics, 12, 129-140.

Examples

```
library(lme4)
data(invData)
# data from one period are considered:
invData2018 <- invData[invData$year == 2018,]
attach(invData2018)
N <- nrow(invData2018) # population size
n <- 100 # sample size

set.seed(12345)
```

```
sampled_elements <- sample(N,n)
reg <- invData2018[, -which(names(invData2018) == 'investments')]

detach(invData2018)

invData2018sample <- invData2018[sampled_elements,]
attach(invData2018sample)
model <- lmer(investments ~ newly_registered + (1|NUTS2) + (1|NUTS4type))
srswrRe(ranef(model),reg)$tablsrswrRe
srswrRe(ranef(model),reg)$lsrswrRe

detach(invData2018sample)
```

`summary.EBLUP` *Summary of EBLUP prediction*

Description

Print the summary of EBLUP prediction and LMM model.

Usage

```
## S3 method for class 'EBLUP'
summary(object, ...)
```

Arguments

`object`	the object of class 'EBLUP'.
`...`	not used.

Author(s)

Alicja Wolny-Dominiak

`summary.ebpLMMne` *Summary of ebpLMMne prediction*

Description

Print the summary of ebpLMMne prediction and LMM.

Usage

```
## S3 method for class 'ebpLMMne'
summary(object, ...)
```

Arguments

`object`	the object of class ’ebpLMMne’.
`...`	not used.

Author(s)

Alicja Wolny-Dominiak

`summary.plugInLMM` *Summary of plugInLMM prediction*

Description

Print the summary of ebpLMMne prediction and LMM model.

Usage

```
## S3 method for class 'plugInLMM'
summary(object, ...)
```

Arguments

`object`	the object of class ’plugInLMM’.
`...`	not used.

Author(s)

Alicja Wolny-Dominiak

`Zfun` *Matrix Z creator*

Description

The function creates the Z matrix of auxiliary variables asscociatied with random effects.

Usage

```
Zfun(model, data)
```

Arguments

`model`	formula of model (use *formula()* function).
`data`	data.

Value

`Z`	Z matrix.
`vNames`	labels of random effects.
`ZBlockNames`	labels of blocks of random effects.

Author(s)

Alicja Wolny-Dominiak

Examples

```
data(invData)
modelFormula <- formula(investments~newly_registered + (newly_registered | NUTS2))
reg <- invData

Zfun(modelFormula, reg)
```

Index