Package ‘msaeOB’

March 14, 2022

Type Package

Title Optimum Benchmarking for Multivariate Small Area Estimation

Version 0.1.0

Author Muhammad Yasqi Imanda [aut, cre], Zenda Oka Briantiko [aut], Azka Ubaidillah [aut]

Maintainer Muhammad Yasqi Imanda <221810403@stis.ac.id>

Description Implements multivariate optimum benchmarking small area estimation. This package provides optimum benchmarking estimation for univariate and multivariate small area estimation and its MSE. In fact, MSE estimators for optimum benchmark are not readily available, so resampling method that called parametric bootstrap is applied. The optimum benchmark model and parametric bootstrap in this package are based on the model proposed in small area estimation. J.N.K Rao and Isabel Molina (2015, ISBN: 978-1-118-73578-7).

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

URL https://github.com/yas-q/msaeOB

BugReports https://github.com/yas-q/msaeOB/issues

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Imports magic, abind, Matrix, MASS, stats

Depends R (>= 3.5.0)

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-14 19:50:02 UTC
**R topics documented:**

- datamsaeOB
- datamsaeOBns
- est_mseOB
- est_mseOBns
- est_saeOB
- est_saeOBns
- mse_mseOB
- mse_mseOBns
- mse_saeOB
- mse_saeOBns

**Description**

Dataset to simulate optimum benchmarking of Multivariate Fay-Herriot model

This data is generated based on multivariate Fay-Herriot model by these following steps:

1. Generate explanatory variables $X_1$ and $X_2$. $X_1 \sim U(4, 6)$ and $X_2 \sim N(5, 0.5)$.
   
   Sampling error $e$ is generated with the following $\sigma_e_{11} = 0.05$, $\sigma_e_{22} = 0.1$, $\sigma_e_{33} = 0.15$, and $\rho_e = 1/2$.
   
   For random effect $u$, we set $\sigma_u_{11} = 0.1$, $\sigma_u_{22} = 0.2$, and $\sigma_u_{33} = 0.3$.
   
   For the weight, we generate $w_1, w_2, w_3$ by set $w_1, w_2, w_3 \sim U(5, 15)$
   
   Set beta, $\beta_{01} = 10$, $\beta_{02} = 9$, $\beta_{03} = 8$, $\beta_{11} = 0.15$, $\beta_{12} = -0.45$, $\beta_{13} = 0.3$, $\beta_{21} = -0.5$, $\beta_{22} = 0.25$, and $\beta_{23} = -0.75$.

   Calculate direct estimation $Y_1, Y_2, Y_3$ where $Y_i = X \cdot \beta + u_i + e_i$

2. Then combine the direct estimations $Y_1, Y_2, Y_3$, explanatory variables $X_1, X_2$, weight $w_1, w_2, w_3$, and sampling varians covarians $v_1, v_{12}, v_{13}, v_2, v_{23}, v_3$ in a dataframe then named as datamsaeOB

**Usage**

- datamsaeOB

**Format**

A data frame with 40 rows and 14 variables:

- Y1 Direct Estimation of Y1
- Y2 Direct Estimation of Y2
- Y3 Direct Estimation of Y3
**datamsaeOBns**

X1  Auxiliary variable of X1  
X2  Auxiliary variable of X2  
w1  Known proportion of units in small areas of Y1  
w2  Known proportion of units in small areas of Y2  
w3  Known proportion of units in small areas of Y3  
v1  Sampling Variance of Y1  
v12  Sampling Covariance of Y1 and Y2  
v13  Sampling Covariance of Y1 and Y3  
v2  Sampling Variance of Y2  
v23  Sampling Covariance of Y2 and Y3  
v3  Sampling Variance of Y3  

---

**datamsaeOBns**  
*Sample Data for Multivariate Non Sampled Area in Small Area Estimation with Optimum Benchmarking*

---

**Description**

Dataset to simulate optimum benchmarking of Multivariate non sampled area in Fay-Herriot model

This data is generated based on multivariate Fay-Herriot model by these following steps:

1. Generate explanatory variables X1 and X2. X1 ~ U(4, 6) and X2 ~ N(5, 0.5).  
   Cluster is generated discrete uniform distribution with a = 1 and b = 2.  
   Sampling error e is generated with the following $\sigma_{e11} = 0.05, \sigma_{e22} = 0.1, \sigma_{e33} = 0.15$, and $\rho_e = 1/2$.  
   For random effect u, we set $\sigma_{u11} = 0.1, \sigma_{u22} = 0.2, \text{and} \sigma_{u33} = 0.3$.  
   For the weight, we generate w1, w2, w3 by set w1, w2, w3 ~ U(5, 15)  
   Set beta, $\beta_{01} = 10, \beta_{02} = 9, \beta_{03} = 8, \beta_{11} = 0.15, \beta_{12} = -0.45, \beta_{13} = 0.3, \beta_{21} = -0.5, \beta_{22} = 0.25, \text{and} \beta_{23} = -0.75$.  
   Calculate direct estimation Y1 Y2 Y3 where $Y_i = X \ast \beta + u_i + e_i$

2. Then combine the direct estimations Y1 Y2 Y3, explanatory variables X1 X2, weight w1 w2 w3,  
   and sampling variances covariances v1 v12 v13 v2 v23 v3 in a dataframe then named as datamsaeOBns

**Usage**

datamsaeOBns
Format

A data frame with 40 rows and 17 variables:

- **Y1** Direct Estimation of Y1
- **Y2** Direct Estimation of Y2
- **Y3** Direct Estimation of Y3
- **X1** Auxiliary variable of X1
- **X2** Auxiliary variable of X2
- **w1** Known proportion of units in small areas of Y1
- **w2** Known proportion of units in small areas of Y2
- **w3** Known proportion of units in small areas of Y3
- **v1** Sampling Variance of Y1
- **v12** Sampling Covariance of Y1 and Y2
- **v13** Sampling Covariance of Y1 and Y3
- **v2** Sampling Variance of Y2
- **v23** Sampling Covariance of Y2 and Y3
- **v3** Sampling Variance of Y3
- **c1** Cluster for Y1
- **c2** Cluster for Y2
- **c3** Cluster for Y3

---

| est_msaeOB     | EBLUPs Optimum Benchmarking based on a Multivariate Fay Herriot (Model 1) |

Description

This function gives EBLUPs optimum benchmarking based on multivariate Fay-Herriot (Model 1)

Usage

```r
est_msaeOB(
  formula,
  vardir,
  weight,
  samevar = FALSE,
  MAXITER = 100,
  PRECISION = 1e-04,
  data
)
```
est_msaeOB

Arguments

- **formula**: an object of class list of formula describe the fitted models
- **vardir**: matrix containing sampling variances of direct estimators. The order is: \(v_{12}, \ldots, v_{1r}, v_{2r}, \ldots, v_{(r-1)r}, v_{rr}\)
- **weight**: matrix containing proportion of units in small areas. The order is: \(w_{1}, w_{2}, \ldots, w_{(r)}\)
- **samevar**: logical. If TRUE, the variances is same. Default is FALSE
- **MAXITER**: maximum number of iterations for Fisher-scoring. Default is 100
- **PRECISION**: coverage tolerance limit for the Fisher Scoring algorithm. Default value is \(1e^{-4}\)
- **data**: dataframe containing the variables named in formula, vardir, and weight

Value

This function returns a list with following objects:

- **eblup**: a list containing a value of estimators
  - **est.eblup**: a dataframe containing EBLUP estimators
  - **est.eblupOB**: a dataframe containing optimum benchmark estimators

- **fit**: a list containing following objects:
  - **method**: fitting method, named "REML"
  - **convergence**: logical value of convergence of Fisher Scoring
  - **iterations**: number of iterations of Fisher Scoring algorithm
  - **estcoef**: a data frame containing estimated model coefficients (\(\beta\), std. error, \(t\) value, \(p\)-value)
  - **refvar**: estimated random effect variance

- **random.effect**: a data frame containing values of random effect estimators
- **agregation**: a data frame containing agregation of direct, EBLUP, and optimum benchmark estimation

Examples

```r
# load dataset
data(datamsaeOB)

# Compute EBLUP & Optimum Benchmark using auxiliary variables X1 and X2 for each dependent variable

# Using parameter 'data'
Fo = list(f1 = Y1 ~ X1 + X2,
          f2 = Y2 ~ X1 + X2,
          f3 = Y3 ~ X1 + X2)
vardir = c("v1", "v12", "v13", "v2", "v23", "v3")
weight = c("w1", "w2", "w3")
est_msae = est_msaeOB(Fo, vardir, weight, data = datamsaeOB)

# Without parameter 'data'
```
est_msaeOBns

**Description**

This function gives EBLUPs optimum benchmarking for non sampled area based on multivariate Fay-Herriot (Model 1)

**Usage**

```r
est_msaeOBns(formula, vardir, weight, 
cluster, 
samevar = FALSE, 
MAXITER = 100, 
PRECISION = 1e-04, 
data)
```

**Arguments**

- `formula`: an object of class list of formula describe the fitted models
- `vardir`: matrix containing sampling variances of direct estimators. The order is: `var1, cov12, ... , cov1r, var2, cov23, ... , covr(r)`
- `weight`: matrix containing proportion of units in small areas. The order is: `w1, w2, ... , w(r)`
- `cluster`: matrix containing cluster of auxiliary variables. The order is: `c1, c2, ... , c(r)`
- `samevar`: logical. If TRUE, the variances is same. Default is FALSE
- `MAXITER`: maximum number of iterations for Fisher-scoring. Default is 100
- `PRECISION`: coverage tolerance limit for the Fisher Scoring algorithm. Default value is 1e-4
- `data`: dataframe containing the variables named in formula, vardir, and weight
**Value**

This function returns a list with following objects:

- **eblup**: a list containing a value of estimators
  - `est.eblup`: a dataframe containing EBLUP estimators
  - `est.eblupOB`: a dataframe containing optimum benchmark estimators

- **fit**: a list containing following objects:
  - `method`: fitting method, named “REML”
  - `convergence`: logical value of convergence of Fisher Scoring
  - `iterations`: number of iterations of Fisher Scoring algorithm
  - `estcoef`: a data frame containing estimated model coefficients (\(\beta, \text{std. error}, t\text{ value}, p\text{-value}\))
  - `refvar`: estimated random effect variance

- **random.effect**: a data frame containing values of random effect estimators

- **agregation**: a data frame containing agregation of direct, EBLUP, and optimum benchmark estimation

**Examples**

```r
## load dataset
data(datamsaeOBns)

## Compute EBLUP & Optimum Benchmark using auxiliary variables X1 and X2 for each dependent variable

## Using parameter 'data'
## Not run:
Fo = list(f1 = Y1 ~ X1 + X2,
          f2 = Y2 ~ X1 + X2,
          f3 = Y3 ~ X1 + X2)
vardir = c("v1", "v12", "v13", "v2", "v23", "v3")
weight = c("w1", "w2", "w3")
cluster = c("c1", "c2", "c3")
est_msae = est_msaeOBns(Fo, vardir, weight, cluster, data = datamsaeOBns)

## Without parameter 'data'
Fo = list(f1 = datamsaeOBns$Y1 ~ datamsaeOBns$X1 + datamsaeOBns$X2,
          f2 = datamsaeOBns$Y2 ~ datamsaeOBns$X1 + datamsaeOBns$X2,
          f3 = datamsaeOBns$Y3 ~ datamsaeOBns$X1 + datamsaeOBns$X2)
vardir = datamsaeOBns[, c("v1", "v12", "v13", "v2", "v23", "v3")]
weight = datamsaeOBns[, c("w1", "w2", "w3")]
cluster = datamsaeOBns[, c("c1", "c2", "c3")]
est_msae = est_msaeOBns(Fo, vardir, weight, cluster)

## Return
est_msae$eblup$est.eblupOB # to see the Optimum Benchmark estimators
```
### Description

This function gives EBLUPs optimum benchmarking based on univariate Fay-Herriot (Model 1)

### Usage

```r
est_saeOB(
  formula,
  vardir,
  weight,
  samevar = FALSE,
  MAXITER = 100,
  PRECISION = 1e-04,
  data
)
```

### Arguments

- **formula**: an object of class list of formula describe the fitted model.
- **vardir**: vector containing sampling variances of direct estimators.
- **weight**: vector containing proportion of units in small areas.
- **samevar**: logical. If TRUE, the varians is same. Default is FALSE.
- **MAXITER**: maximum number of iterations for Fisher-scoring. Default is 100.
- **PRECISION**: coverage tolerance limit for the Fisher Scoring algorithm. Default value is 1e-4.
- **data**: dataframe containing the variables named in formula, vardir, and weight.

### Value

This function returns a list with following objects:

- **eblup**: a list containing a value of estimators.
  - `est.eblup`: a dataframe containing EBLUP estimators
  - `est.eblupOB`: a dataframe containing optimum benchmark estimators

- **fit**: a list containing following objects:
  - `method`: fitting method, named "REML"
  - `convergence`: logical value of convergence of Fisher Scoring
est_saeOBns

- iterations: number of iterations of Fisher Scoring algorithm
- estcoef: a data frame containing estimated model coefficients (beta, std. error, t value, p-value)
- refvar: estimated random effect variance

random.effect: a data frame containing values of random effect estimators
agregation: a data frame containing aggregation of direct, EBLUP, and optimum benchmark estimation

Examples

```r
## load dataset
data(datamsaeOB)

# Compute EBLUP & Optimum Benchmark using auxiliary variables X1 and X2 for each dependent variable

## Using parameter 'data'
est_sae = est_saeOB(Y1 ~ X1 + X2, v1, w1, data = datamsaeOB)

## Without parameter 'data'
est_sae = est_saeOB(datamsaeOB$Y1 ~ datamsaeOB$X1 + datamsaeOB$X2, datamsaeOB$v1, datamsaeOB$w1)

## Return
est_sae$eblup$est.eblupOB # to see the Optimum Benchmark estimators
```

---

**est_saeOBns**

EBLUPs Optimum Benchmarking for Non Sampled Area based on a Univariate Fay-Herriot (Model 1)

---

**Description**

This function gives EBLUPs optimum benchmarking for non sampled area based on univariate Fay-Herriot (model 1)

**Usage**

```r
est_saeOBns(
  formula,
  vardir,
  weight,
  cluster,
  samevar = FALSE,
  MAXITER = 100,
  PRECISION = 1e-04,
  data
)
```
Arguments

- **formula**: an object of class list of formula describe the fitted model
- **vardir**: vector containing sampling variances of direct estimators
- **weight**: vector containing proportion of units in small areas
- **cluster**: vector containing cluster of auxiliary variable
- **samevar**: logical. If TRUE, the variances is same. Default is FALSE
- **MAXITER**: maximum number of iterations for Fisher-scoring. Default is 100
- **PRECISION**: coverage tolerance limit for the Fisher Scoring algorithm. Default value is 1e-4
- **data**: dataframe containing the variables named in formula, vardir, and weight

Value

This function returns a list with following objects:

- **eblup**: a list containing a value of estimators
  - **est.eblup**: a dataframe containing EBLUP estimators
  - **est.eblupOB**: a dataframe containing optimum benchmark estimators

- **fit**: a list containing following objects:
  - **method**: fitting method, named "REML"
  - **convergence**: logical value of convergence of Fisher Scoring
  - **iterations**: number of iterations of Fisher Scoring algorithm
  - **estcoef**: a data frame containing estimated model coefficients (beta, std. error, t value, p-value)
  - **refvar**: estimated random effect variance

- **random.effect**: a dataframe containing values of random effect estimators
- **agregation**: a dataframe containing aggregation of direct, EBLUP, and optimum benchmark estimation

Examples

```r
## load dataset
data(datamsaeOBns)

# Compute EBLUP & Optimum Benchmark using auxiliary variables X1 and X2 for each dependent variable
## Using parameter 'data'
est_sae = est_saeOBns(Y1 ~ X1 + X2, v1, w1, c1, data = datamsaeOBns)

## Without parameter 'data'
est_sae = est_saeOBns(datamsaeOBns$Y1 ~ datamsaeOBns$X1 + datamsaeOBns$X2, datamsaeOBns$v1, datamsaeOBns$w1, datamsaeOBns$c1)

## Return
est_sae$eblup$est.eblupOB # to see the Optimum Benchmark estimators
```
mse_msaeb

---

### mse_mseOB

**Parametric Bootstrap Mean Squared Error Estimators of Optimum Benchmarking for Multivariate Small Area Estimation**

#### Description

Calculates the parametric bootstrap mean squared error estimates of optimum benchmarking for multivariate small area estimation

#### Usage

```r
mse_mseOB(
  formula,
  vardir,
  weight,
  samevar = FALSE,
  B = 100,
  MAXITER = 100,
  PRECISION = 1e-04,
  data
)
```

#### Arguments

- **formula**: an object of class list of formula describe the fitted models
- **vardir**: matrix containing sampling variances of direct estimators. The order is: var1, cov12, ..., cov1r, var2, cov23, ..., cov2r, ..., cov(r-1)(r), var(r)
- **weight**: matrix containing proportion of units in small areas. The order is: w1, w2, ..., w(r)
- **samevar**: logical. If TRUE, the variances are same. Default is FALSE
- **B**: number of bootstrap. Default is 1000
- **MAXITER**: maximum number of iterations for Fisher-scoring. Default is 100
- **PRECISION**: coverage tolerance limit for the Fisher Scoring algorithm. Default value is 1e-4
- **data**: dataframe containing the variables named in formula, vardir, and weight

#### Value

- **mse.eblup**: estimated mean squared errors of the EBLUPs for the small domains based on Prasad Rao
- **pbmse.eblupOB**: parametric bootstrap mean squared error estimates of the optimum benchmark
- **running.time**: time for running function
Examples

```r
## load dataset
data(datamsaeOB)

# Compute MSE EBLUP and Optimum Benchmark
# This is the long running example
## Using parameter 'data'
Fo = list(f1 = Y1 ~ X1 + X2,
          f2 = Y2 ~ X1 + X2,
          f3 = Y3 ~ X1 + X2)
vardir = c("v1", "v12", "v13", "v2", "v23", "v3")
weight = c("w1", "w2", "w3")

t = 5

mse = mse_msaeOB(Fo, vardir, weight, data = datamsaeOB)

## Without parameter 'data'
Fo = list(f1 = datamsaeOB$Y1 ~ datamsaeOB$X1 + datamsaeOB$X2,
          f2 = datamsaeOB$Y2 ~ datamsaeOB$X1 + datamsaeOB$X2,
          f3 = datamsaeOB$Y3 ~ datamsaeOB$X1 + datamsaeOB$X2)
vardir = datamsaeOB[, c("v1", "v12", "v13", "v2", "v23", "v3")]
weight = datamsaeOB[, c("w1", "w2", "w3")]

mse = mse_msaeOB(Fo, vardir, weight)

## Return
mse$pbmse.eblupOB # to see the MSE of Optimum Benchmark
```

---

**mse_msaeOBns**

*Parametric Bootstrap Mean Squared Error Estimators of Optimum Benchmarking for Multivariate Non Sampled Area in Small Area Estimation*

**Description**

Calculates the parametric bootstrap mean squared error estimates of optimum benchmarking for multivariate non sampled area in small area estimation.

**Usage**

```r
mse_msaeOBns(
  formula,
  vardir,
  weight,
  cluster,
  samevar = FALSE,
  B = 100,
  MAXITER = 100,
)"
Arguments

formula: an object of class list of formula describe the fitted models
vardir: matrix containing sampling variances of direct estimators. The order is: var1, cov12, ..., var1r, cov2r, ..., var(r)
weight: matrix containing proportion of units in small areas. The order is: w1, w2, ..., w(r)
cluster: matrix containing cluster of auxiliary variables. The order is: c1, c2, ..., c(r)
samevar: logical. If TRUE, the variances are same. Default is FALSE
B: number of bootstrap. Default is 1000
MAXITER: maximum number of iterations for Fisher-scoring. Default is 100
PRECISION: coverage tolerance limit for the Fisher Scoring algorithm. Default value is 1e-4
data: dataframe containing the variables named in formula, vardir, and weight

Value

mse.eblup: estimated mean squared errors of the EBLUPs for the small domains based on Prasad Rao
pbmse.eblupOB: parametric bootstrap mean squared error estimates of the optimum benchmark
running.time: time for running function

Examples

## load dataset
data(datamsaeOBns)

# Compute MSE EBLUP and Optimum Benchmark
# This is the long running example
## Using parameter 'data'
Fo = list(f1 = Y1 ~ X1 + X2,
          f2 = Y2 ~ X1 + X2,
          f3 = Y3 ~ X1 + X2)
vardir = c("v1", "v12", "v13", "v2", "v23", "v3")
weight = c("w1", "w2", "w3")
cluster = c("c1", "c2", "c3")
mse.mse = mse.mseOBns(Fo, vardir, weight, cluster, data = datamsaeOBns)

## Without parameter 'data'
Fo = list(f1 = datamsaeOBns$Y1 ~ datamsaeOBns$X1 + datamsaeOBns$X2,
          f2 = datamsaeOBns$Y2 ~ datamsaeOBns$X1 + datamsaeOBns$X2,
          f3 = datamsaeOBns$Y3 ~ datamsaeOBns$X1 + datamsaeOBns$X2)
vardir = datamsaeOBns[, c("v1", "v12", "v13", "v2", "v23", "v3")]
weight = datamsaeOBns[, c("w1", "w2", "w3")]
cluster = datamsaeOBns[, c("c1", "c2", "c3")]

mse.mse = mse.mseOBns(Fo, vardir, weight, cluster, data = datamsaeOBns)
mse_saeOB = mse_saeOBns(Fo, vardir, weight, cluster)

## Return
mse_saeOB$pbmse.eblupOB # to see the MSE of Optimum Benchmark

---

mse_saeOB

**Parametric Bootstrap Mean Squared Error Estimators of Optimum Benchmarking for Univariate Small Area Estimation**

**Description**

Calculates the parametric bootstrap mean squared error estimates of optimum benchmarking for univariate small area estimation

**Usage**

```r
mse_saeOB(
  formula,
  vardir,
  weight,
  samevar = FALSE,
  B = 100,
  MAXITER = 100,
  PRECISION = 1e-04,
  data
)
```

**Arguments**

- `formula`: an object of class list of formula describe the fitted model
- `vardir`: vector containing sampling variances of direct estimators
- `weight`: vector containing proportion of units in small areas
- `samevar`: logical. If TRUE, the variances is same. Default is FALSE
- `B`: number of bootstrap. Default is 1000
- `MAXITER`: maximum number of iterations for Fisher-scoring. Default is 100
- `PRECISION`: coverage tolerance limit for the Fisher Scoring algorithm. Default value is 1e-4
- `data`: dataframe containing the variables named in formula, vardir, and weight

**Value**

- `mse.eblup`: estimated mean squared errors of the EBLUPs for the small domains based on Prasad Rao
- `pbmse.eblupOB`: parametric bootstrap mean squared error estimates of the optimum benchmark
- `running.time`: time for running function
Examples

```r
## load dataset
data(datamsaeOB)

# Compute MSE EBLUP and Optimum Benchmark

## Using parameter 'data'
mse_sae = mse_saeOB(Y1 ~ X1 + X2, v1, w1, data = datamsaeOB)

## Without parameter 'data'
mse_sae = mse_saeOB(datamsaeOB$Y1 ~ datamsaeOB$X1 + datamsaeOB$X2, datamsaeOB$v1, datamsaeOB$w1)

## Return
mse_sae$pbmse.eblupOB # to see the MSE Optimum Benchmark estimators
```

### Description

Calculates the parametric bootstrap mean squared error estimates of optimum benchmarking for univariate non sampled area in small area estimation

#### Usage

```r
mse_saeOBns(formula, vardir, weight, cluster, samevar = FALSE, B = 100, MAXITER = 100, PRECISION = 1e-04, data)
```

#### Arguments

- `formula`: an object of class list of formula describe the fitted model
- `vardir`: vector containing sampling variances of direct estimators
- `weight`: vector containing proportion of units in small areas
- `cluster`: vector containing cluster of auxiliary variable

---

**mse_saeOBns**  
*Parametric Bootstrap Mean Squared Error Estimators of Optimum Benchmarking for Univariate Non Sampled Area in Small Area Estimation*
samevar logical. If TRUE, the varians is same. Default is FALSE
B number of bootstrap. Default is 1000
MAXITER maximum number of iterations for Fisher-scoring. Default is 100
PRECISION coverage tolerance limit for the Fisher Scoring algorithm. Default value is 1e-4
data dataframe containing the variables named in formula, vardir, and weight

Value
mse.eblup estimated mean squared errors of the EBLUPs for the small domains based on Prasad Rao
pbmse.eblupOB parametric bootstrap mean squared error estimates of the optimum benchmark
running.time time for running function

Examples

## load dataset
data(datamsaeOBns)

# Compute MSE EBLUP and Optimum Benchmark

## Using parameter 'data'
 mse.sae = mse_saeOBns(Y1 ~ X1 + X2, v1, w1, c1, data = datamsaeOBns)

## Without parameter 'data'
 mse.sae = mse_saeOBns(datamsaeOBns$Y1 ~ datamsaeOBns$X1 + datamsaeOBns$X2, datamsaeOBns$v1, datamsaeOBns$w1, datamsaeOBns$c1)

## Return
mse.sae$pbmse.eblupOB # to see the MSE Optimum Benchmark estimators
Index

* datasets
  datamsaeOB, 2
  datamsaeOBns, 3
  datamsaeOB, 2
  datamsaeOBns, 3
  est_msaeb, 4
  est_msaebns, 6
  est_saeB, 8
  est_saeBns, 9
  mse_msaeb, 11
  mse_msaebns, 12
  mse_saeB, 14
  mse_saeBns, 15