Package ‘hbsae’

February 20, 2015

Version 1.0

Author Harm Jan Boonstra

Maintainer Harm Jan Boonstra <hjboonstra@gmail.com>

License GPL-3

Title Hierarchical Bayesian Small Area Estimation

Type Package

LazyLoad yes

Description Functions to compute small area estimates based on a basic area or unit-level model. The model is fit using restricted maximum likelihood, or in a hierarchical Bayesian way. In the latter case numerical integration is used to average over the posterior density for the between-area variance. The output includes the model fit, small area estimates and corresponding MSEs, as well as some model selection measures. Additional functions provide means to compute aggregate estimates and MSEs, to minimally adjust the small area estimates to benchmarks at a higher aggregation level, and to graphically compare different sets of small area estimates.

Date 2012-09-05

Depends R (>= 2.10)

Imports Matrix, arm

Collate 'fakedata.R' 'hbsae.R'

Repository CRAN

Date/Publication 2012-09-05 13:03:20

NeedsCompilation no

R topics documented:

`hbsae-package` *A package for hierarchical Bayesian small area estimation.*

Description

Package hbsae provides functions to compute small area estimates based on the basic unit-level and area-level models. The models are fit and small area estimates are computed in a hierarchical Bayesian way, using numerical integration.

Details

The main function that does most of the computational work is `fSAE.Unit`. Function `fSAE` is provided as a more convenient interface to `fSurvReg`, `fSAE.Area` and `fSAE.Unit`.

Author(s)

Harm Jan Boonstra <hjboonstra@gmail.com>

`aggr` *Compute aggregates of small area estimates and MSEs.*

Description

Compute aggregates of small area estimates and MSEs.

Usage

```
aggr(x, R)
```

Arguments

x	sae object.
R	aggregation matrix, r x M matrix where M is the number of areas and r the number of aggregate areas; default is aggregation over all areas.

Value

Object of class sae with aggregated small area estimates and MSEs.

See Also

sae-class

Examples

```
d <- generateFakeData()

# compute small area estimates
sae <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop)

# by default aggregate over all areas
global <- aggr(sae)
EST(global); SE(global)

# aggregation to broad area
# first build aggregation matrix
M <- d$Xpop[, c("area22", "area23", "area24")] / d$Xpop[, "(Intercept)"]
M <- cbind(1 - rowSums(M), M); colnames(M)[1] <- "area21"
est.area2 <- aggr(sae, M)
EST(est.area2); SE(est.area2)
COV(est.area2)  # covariance matrix
```

---

bench *Benchmark small area estimates.*

Description

Benchmark small area estimates to agree with given totals at aggregate levels.

Usage

```
bench(x, R, rhs, mseMethod = "no")
```

Arguments

`x`	sae object to be benchmarked.
`R`	restriction matrix, r x M matrix where r is the number of restrictions and M the number of areas; default is a single constraint on the population total. Note that R acts on the vector of area population totals, not the vector of means.
`rhs`	r-vector of benchmark totals corresponding to the restrictions in the rows of R.
`mseMethod`	if "no", MSEs are not updated, if "exact", constraints are treated as exact identities, and if "model", the squared differences between original and benchmarked estimates are added to the MSEs.

Value

An object of class `sae` with adjusted estimates.

References

Y. You, J.N.K. Rao and P. Dick (2004). Benchmarking Hierarchical Bayes Small Area Estimators in the Canadian Census Undercoverage Estimation. Statistics in Transition 6(5), 631-640.

See Also

`sae-class`

Examples

```
d <- generateFakeData()

# compute small area estimates
sae <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop)

# calibrate to overall population total
sae.c <- bench(sae, rhs=sum(d$mY0*sae$Narea))
plot(sae, sae.c)
```

`CVarea` *Compute area-level cross-validation measure for sae objects.*

Description

This function computes a cross-validation measure defined at the area level. It can be used, for example, to compare the predictive ability of area and unit-level models. The code is based in part on that of `cv.glm` from package **boot**.

Usage

```
CVarea(sae, weight = TRUE,
  cost = function(y, yhat, w) sum(w * (y - yhat)^2)/sum(w),
  K = 10, method = "hybrid", seed)
```

Arguments

sae	object of class sae, resulting from a call to `fSAE`, `fSAE.Area`, or `fSAE.Unit`.
weight	if `TRUE`, use weights inversely proportional to the MSEs of y - yhat in the cost function.
cost	cost function to be used. Defaults to a quadratic cost function.
K	K in K-fold cross-validation. Specifies in how many parts the dataset should be divided.
method	method used to refit the model. One of "HB", "hybrid" (default) or "REML", in the order of slow to fast.
seed	random seed used in selecting groups of areas to leave out in K-fold cross-validation.

Value

The computed area-level cross-validation measure.

Examples

```
d <- generateFakeData()

# compute small area estimates based on area-level and unit-level models
saeArea <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, type="area",
  silent=TRUE, keep.data=TRUE)
saeUnit <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, type="unit",
  silent=TRUE, keep.data=TRUE)

# compare area and unit-level models based on area-level cross-validation criterion
CVarea(saeArea, K=10, seed=1)  # 10-fold CV for area-level model
CVarea(saeUnit, K=10, seed=1)  # 10-fold CV for unit-level model
```

---

fSAE — *Fit a linear model with random area effects and compute small area estimates.*

Description

This function prepares the (unit-level) input data and calls one of the lower level functions `fSurvReg`, `fSAE.Area` or `fSAE.Unit` to compute survey regression, area-level model or unit-level model small area estimates. Area-level model estimates are computed by first computing survey regression estimates and using these as input for `fSAE.Area`.

Usage

```
fSAE(formula, data, area = NULL, popdata = NULL,
  type = "unit", model.direct = NULL,
  formula.area = NULL, contrasts.arg = NULL,
  remove.redundant = TRUE, redundancy.tol = 1e-07,
  sparse = FALSE, ...)
```

Arguments

formula	model formula, indicating response variable and covariates.
data	unit-level data frame containing all variables used in formula, area and formula.area arguments. These variables should not contain missing values.
area	name of area indicator variable in data; if NULL, no random effects are used in the model.
popdata	data frame or matrix containing area population totals for all covariates. The rows should correspond to areas for which estimates are required. Column names should include those produced by model.matrix(formula, data, contrasts.arg), up to permutations of the names in interactions. A column named '(Intercept)' is required and should contain the area population sizes. If popdata is NULL, only the model fit will be returned.
type	type of small area estimates: "direct" for survey regression, "area" for area-level model, "unit" for unit-level model estimates. If type is "data" then only the data including the model matrix and population means are returned.
model.direct	if type="area", this argument can be used to specify by means of a formula the covariates to use for the computation of the initial survey regression estimates. If unspecified, the covariates specified by formula are used both at the unit-level (for the initial estimates) and at the area-level (for the area-level model estimates).
formula.area	if type="unit", this is an optional formula specifying covariates that should be used at the area level. These covariates should be available in popdata.
contrasts.arg	list for specification of contrasts for factor variables. Passed to model.matrix.
remove.redundant	if TRUE redundant columns in the design matrix are removed. A warning is issued if the same redundancy does not show also in the corresponding population totals. In the case of the area level model there may still be redundancy at the area level.
redundancy.tol	tolerance for detecting linear dependencies among the columns of the design matrix. Also used as tolerance in the check whether the design matrix redundancy is shared by the population totals.
sparse	if TRUE sparse.model.matrix (package Matrix) is used to compute the covariate design matrix. This can be efficient for large datasets and a model containing categorical variables with many categories.
...	additional arguments passed to fSAE.Unit or fSurvReg.

Value

An object of class sae containing the small area estimates, their MSEs, and the model fit. If type is "data" a list containing the model matrix, response vector, area indicator, area population sizes and matrix of population means is returned.

See Also

sae-class

Examples

```
d <- generateFakeData()

# model fitting only
(fit <- fSAE(y0 ~ x + area2, data=d$sam, area="area"))

# model fitting and small area estimation, unit-level model
saeHB <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, silent=TRUE)
saeHB  # print a summary
EST(saeHB)  # small area estimates
SE(saeHB)  # error estimates
str(saeHB)
plot(saeHB, list(est=d$mY0), CI=2)  # compare to true population means

# unit-level model with REML model-fit instead of Bayesian approach
saeREML <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, method="REML", silent=TRUE)
plot(saeHB, saeREML)  # compare

# basic area-level model
saeA <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, type="area")
plot(saeHB, saeA)

# SAE estimates based on a linear unit-level model without area effects
saeL <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, method="synthetic")
plot(saeHB, saeL)

# model-based estimation of overall population mean without area effects
est.global <- fSAE(y0 ~ x + area2, data=d$sam, area=NULL, popdata=colSums(d$Xpop), method="synthetic")
EST(est.global); MSE(est.global)

# no model fitting or estimation, but return design matrix, variable of interest,
#   area indicator, area population sizes and matrix of population means
dat <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, type="data")
str(dat)
```

`fSAE.Area` *Compute small area estimates based on the basic area-level model.*

Description

This function returns small area estimates based on the basic area-level model, also known as the Fay-Herriot model. It calls `fSAE.Unit` to carry out the computations.

Usage

```
fSAE.Area(est.init, var.init, X, x = X, ...)
```

Arguments

`est.init`	m-vector of initial estimates, where m is the number of sampled areas.
`var.init`	m-vector of corresponding variance estimates.
`X`	M x p matrix of area population means, where M is the number of areas for which estimates are required.
`x`	defaults to X, but in case auxiliary information is available at the unit level, it may be set to the corresponding matrix of covariate sample means.
`...`	additional arguments passed to `fSAE.Unit`.

Value

An object of class `sae` containing the small area estimates and MSEs, the model fit, and model selection measures.

References

R.E. Fay and R.A. Herriot (1979). Estimates of Income for Small Places: An Application of James-Stein Procedures to Census Data. Journal of the American Statistical Association 74(366), 269-277.

J.N.K. Rao (2003). Small Area Estimation. Wiley.

See Also

sae-class

Examples

```
d <- generateFakeData()

# first compute input estimates without "borrowing strength" over areas
sae0 <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, type="direct", keep.data=TRUE)

# compute small area estimates based on the basic area-level model
#   using the above survey regression estimates as input
sae <- fSAE.Area(EST(sae0), MSE(sae0), X=sae0$Xp)
EST(sae)  # estimates
SE(sae)  # standard errors
```

`fSAE.Unit` *Compute small area estimates based on the basic unit-level model.*

Description

This is the function that carries out most of the computational work. It computes small area estimates based on the basic unit-level model, also known as the Battese-Harter-Fuller model, although it is also called by `fSurvReg` and `fSAE.Area` to compute survey regression or area-level model small area estimates. By default, Hierarchical Bayes estimates are computed, using fast one-dimensional numerical integration to average over the posterior density for the ratio of between and within area variance. This way, the small area estimates and MSEs account for the uncertainty about this parameter. Besides hierarchical Bayes, REML and hybrid methods are supported. These methods use the REML estimate or posterior mean of the variance ratio, respectively, as a plug-in estimate. Both methods do not account for uncertainty about this parameter. Synthetic estimates are computed by setting the variance ratio to zero.

Usage

```
fSAE.Unit(y, X, area, Narea = NULL, Xpop = NULL,
  fpc = TRUE, v = NULL, vpop = NULL, w = NULL,
  wpop = NULL, method = "HB", beta0 = rep(0, ncol(X)),
  Omega0 = Diagonal(n = ncol(X), x = 0), nu0 = 0,
  s20 = 0, prior = function(x) rep.int(1L, length(x)),
  CV = TRUE, CVweights = NULL, silent = FALSE,
  keep.data = FALSE, full.cov = nrow(Xpop) < 1000,
  lambda0 = NULL, rel.int.tol = 0.01, ...)
```

Arguments

`y`	response vector of length n.
`X`	n x p model matrix.
`area`	n-vector of area codes, typically a factor variable with m levels, where m is the number of sampled areas.
`Narea`	M-vector of area population sizes, where M is the number of areas for which estimates are required. There should be a ono-to-one correspondence with the rows of `Xpop`. This argument is required unless `Xpop=NULL` or `fpc=FALSE`.
`Xpop`	M x p matrix of population means. If `Xpop` is not provided, only the model fit is returned.
`fpc`	whether a finite population correction should be used. Default is `TRUE`.
`v`	unit-level variance structure, n-vector. Defaults to a vector of 1s. In some cases it might be useful to take v proportional to the sampling probabilities.
`vpop`	population area means of v, M-vector. Defaults to a vector of 1s. Not used when `fpc` is `FALSE`.
`w`	area-level variance structure, m-vector. Defaults to a vector of 1s.
`wpop`	area-level variance structure, M-vector. Defaults to a vector of 1s. Only components of `wpop` corresponding to out-of-sample areas are actually used.
`method`	one of "HB", "hybrid", "REML", "synthetic", "survreg", "BLUP" where "HB" (default) does the full hierarchical Bayes computation, i.e. numerical integration over the posterior density for the between area variance parameter, "hybrid"

computes the Best Linear Unbiased Predictor (BLUP) with the posterior mean for the variance parameter plugged in, "REML" computes the BLUP with the restricted maximum likelihood estimate of the variance parameter plugged in, "synthetic" computes synthetic estimates where the between area variance is set to 0, and "survreg" computes survey regression estimates where the between area variance approaches infinity. "BLUP" computes BLUP estimates with the value provided for `lambda0` as a fixed plug-in value for the ratio of between and within area variance. Only method "HB" takes uncertainty about the between-area variance into account.

- `beta0` mean vector of normal prior for coefficient vector.
- `Omega0` inverse covariance matrix of normal prior for coefficient vector. Default prior corresponds to the (improper) uniform distribution.
- `nu0` degrees of freedom parameter for inverse gamma prior for residual (within-area) variance. Default is 0.
- `s20` scale parameter for inverse gamma prior for residual (within-area) variance. Default is 0.
- `prior` prior density for the ratio lambda = between-area-variance / within-area variance. This should be a (vectorized) function that takes a vector lambda and returns a vector of prior density values at lambda. The density does not have to be normalized. The default is the (improper) uniform prior. The within-area variance and lambda are assumed independent a priori.
- `CV` whether leave-one-out cross-validation and other model selection measures should be computed. Default `TRUE`.
- `CVweights` n-vector of weights to use for CV computation.
- `silent` if `FALSE`, plot the posterior density for the variance ratio.
- `keep.data` if `TRUE` return the input data (y,X,area,Xpop). This is required input for the cross-validation function `CVArea`.
- `full.cov` if `TRUE` compute the full covariance matrix for the small area estimates. The computed correlations do not account for uncertainty about the variance ratio.
- `lambda0` optional starting value for the ratio of between and within-area variance used in the numerical routines. If `method="BLUP"` then this value will instead be used as a fixed plug-in value.
- `rel.int.tol` tolerance for the estimated relative integration error (default is 1 percent). A warning is issued if the estimated relative error exceeds this value.
- `...` additional control parameters passed to function `integrate`.

Details

The default Hierarchical Bayes method uses numerical integration (as provided by function `integrate`) to compute small area estimates and MSEs. The model parameters returned, such as fixed and random effects, are currently not averaged over the posterior distribution for the variance ratio. They are evaluated at the posterior mean of the variance ratio.

Value

An object of class `sae` containing the small area estimates and MSEs, the model fit, and model selection measures.

References

G.E. Battese, R.M. Harter and W.A. Fuller (1988). An Error-Components Model for Prediction of County Crop Areas Using Survey and Satellite Data. Journal of the American Statistical Association, 83(401), 28-36.

G.S. Datta and M. Ghosh (1991). Bayesian Prediction in Linear Models: Applications to Small Area Estimation. The Annals of Statistics 19(4), 1748-1770.

J.N.K. Rao (2003). Small Area Estimation. Wiley.

See Also

sae-class

Examples

```
d <- generateFakeData()

# generate design matrix, variable of interest, area indicator and population data
dat <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, type="data")

# compute small area estimates based on the basic unit-level model
sae <- fSAE.Unit(dat$y, dat$X, dat$area, dat$Narea, dat$PopMeans)
EST(sae)  # estimates
SE(sae)  # standard errors
```

`fSurvReg` *Compute small area estimates based on the survey regression estimator.*

Description

This function computes survey regression estimates as a special case of unit-level model small area estimates with a (relatively) very large value for the between-area variance but without including area effects in the model fit. The model assumes a single overall variance parameter, so that the resulting estimated variances are not area-specific but smoothed. Varying inclusion probabilities may be taken into account by including them in the model, e.g. as an additional covariate, and/or as model variance structure by specifying arguments v and vpop, see fSAE.Unit. The resulting estimates may be used as input estimates for area-level model small area estimation.

Usage

```
fSurvReg(y, X, area, Narea, Xpop, removeEmpty = TRUE,
  ...)
```

Arguments

`y`	response vector of length n.
`X`	n x p model matrix.
`area`	n-vector of area codes, typically a factor variable with m levels, where m is the number of sampled areas.
`Narea`	M-vector of area population sizes.
`Xpop`	M x p matrix of population means.
`removeEmpty`	whether out-of-sample areas should be removed from the results. If `FALSE` these areas are retained in the vectors of estimates, but they will have (relatively) very large standard errors.
`...`	optional arguments v and vpop passed to `fSAE.Unit`.

Value

An object of class `sae` containing the survey regression small area estimates and their estimated variances.

References

G.E. Battese, R.M. Harter and W.A. Fuller (1988). An Error-Components Model for Prediction of County Crop Areas Using Survey and Satellite Data. Journal of the American Statistical Association, 83(401), 28-36.

J.N.K. Rao (2003). Small Area Estimation. Wiley.

See Also

`sae-class`

Examples

```
d <- generateFakeData()

# generate design matrix, variable of interest, area indicator and population data
dat <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, type="data")

sae <- fSurvReg(dat$y, dat$X, dat$area, dat$Narea, dat$PopMeans)
EST(sae)  # estimates
SE(sae)  # standard errors
```

generateFakeData	*Generate artificial dataset for demonstration and testing purposes.*

Description

Generate artificial dataset for demonstration and testing purposes.

Usage

```
generateFakeData(M = 50, meanNarea = 1000, sW = 100,
  sB = 50, sBx = 0.5, samplingFraction = 0.1)
```

Arguments

`M`	number of areas.
`meanNarea`	mean number of population units per area.
`sW`	within area standard deviation.
`sB`	between area standard deviation.
`sBx`	random slope standard deviation.
`samplingFraction`	sampling fraction used to draw a random sample from the population units.

Value

List containing sample (sam), population totals (Xpop), and true population means for four target variables (mY0, mY1, mY2, mY3).

plot.sae	*Plot method for objects of class sae.*

Description

This function calls `coefplot` from package **arm** to plot small area estimates and standard errors. Multiple sets of estimates can be compared using this plot. The default ordering of the estimates is by their area population sizes.

Usage

```
## S3 method for class 'sae'
plot(..., sort.by = NULL,
  decreasing = FALSE, index = NULL, maxrows = 50,
  maxcols = 6, labels = NULL, type = "sae",
  ylab = if (type == "coef") "coef" else "area",
  main = switch(type, coef = "coefficients", raneff = "random effects", "SAE estimates"),
  offset = 0.2, cex.var = 0.9, mar = c(0, 1, 5, 0),
  CI = 1)
```

Arguments

`...`	list of sae objects (or lists with at least a component "est" (and optionally "mse")). Other components not matching other arguments are passed to the lower-level plotting functions.
`sort.by`	vector by which to sort the vectors to be plotted; defaults to population size vector of the first sae object.
`decreasing`	if `TRUE`, sort in decreasing order (default).
`index`	vector of indices of the selected areas to be plotted.
`maxrows`	maximum number of rows in a column.
`maxcols`	maximum number of columns of estimates on a page.
`labels`	optional vector of lables; default is the vector of area names of the first sae object.
`type`	"sae" for small area estimates (default), "coef" for coefficients, "raneff" for random effects.
`ylab`	label for y-axis.
`main`	main title.
`offset`	space used between plots of multiple estimates for the same area.
`cex.var`	the fontsize of the variable names, default=0.9.
`mar`	a numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(0,1,5,0).
`CI`	confidence interval, default is 1, which will plot plus and minus 1 standard errors. If CI=2, plots plus and minus 2 standard errors instead.

`plot.weights` *Plot method for objects of class* `weights`.

Description

Plot method for objects of class `weights`.

Usage

```
## S3 method for class 'weights'
plot(x, log = FALSE,
   main = "Distribution of weights",
   xlab = if (log) "log(weight)" else "weight",
   ylab = "frequency", col = "cyan", ...)
```

Arguments

x	object of class weights as returned by function uweights.
log	whether to log-tranform the weights.
main	main title of plot.
xlab	x-axis label.
ylab	y-axis label.
col	colour.
...	additional arguments passed to hist.

See Also

uweights, summary.weights

---

print.sae — *Print method for objects of class sae.*

Description

Print method for objects of class sae.

Usage

```
## S3 method for class 'sae'
print(x,
   digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	object of class sae.
digits	number of digits to display.
...	additional arguments passed to print.default.

sae-class *S3 class for the fitted model and SAE outcomes.*

Description

Functions fSAE, fSurvReg, fSAE.Area and fSAE.Unit return an object of class sae. It contains information on the model fit as well as the small area estimates, error estimates and a few model selection measures. The functions listed below extract the main components from an object of class sae.

`EST(x, type="sae")` return the vector of small area estimates of sae object x. Alternatively, with type "coef" or "raneff" fixed or random effect estimates are returned.

`MSE(x, type="sae")` return the vector of mean squared errors of sae object x. Alternatively, with type "coef" or "raneff" MSEs of fixed or random effects are returned.

`SE(x, type="sae")` extract standard errors, i.e. square roots of the MSEs.

`relSE(x, type="sae")` extract relative standard errors.

`COV(x)` extract the covariance matrix for the small area estimates.

`COR(x)` extract the correlation matrix for the small area estimates.

`coef(x)` coef method for sae objects; returns vector of fixed effects.

`vcov(x)` vcov method for sae objects; returns covariance matrix for fixed effects.

`raneff(x, pop)` return vector of random effects. If pop=TRUE returns a vector for predicted areas (zero for out-of-sample areas), otherwise a vector for sampled areas.

`raneff.se(x, pop)` return vector of standard errors for random effects.

`residuals(x)` residuals method for sae objects; returns a vector of residuals.

`fitted(x)` fitted method for sae objects; returns a vector of fitted values.

`se2(x)` extracts within-area variance estimate.

`sv2(x)` extracts between-area variance estimate.

`wDirect(x, pop)` extract vector of weights of the survey regression components in the small area estimates. If pop=TRUE returns a vector for predicted areas (zero for out-of-sample areas), otherwise a vector for sampled areas.

`synthetic(x)` extract vector of synthetic estimates.

`CV(x)` extract leave-one-out cross-validation measure.

`cAIC(x)` extract conditional AIC measure.

`R2(x)` extract unit-level R-squared goodness-of-fit measure.

Other components include

`relErrM,relErrV` relative numerical integration errors in estimates and MSEs, for method "HB".

Examples

```
d <- generateFakeData()

# compute small area estimates
sae <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop)

coef(sae)  # fixed effects
raneff(sae)  # random effects
sv2(sae)  # between-area variance
se2(sae)  # within-area variance
cAIC(sae)  # conditional AIC
```

---

summary.weights — *Summary method for objects of class* `weights`.

Description

Summary method for objects of class `weights`.

Usage

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

Arguments

`object`	object of class `weights` as returned by function `uweights`.
`...`	not used.

See Also

`uweights`, `plot.weights`

---

uweights — *Compute unit weights underlying the small area estimates or their aggregate.*

Description

The small area estimates can be interpreted as weighted sums of the response variable. This function computes the weights corresponding to the aggregated small area estimates or the weights corresponding to a specific small area estimate. The weights applied to the response variable need not exactly reproduce the Hierarchical Bayes estimate since the latter is averaged over the posterior distribution for the variance ratio whereas the weights are evaluated at the posterior mean. Under the default prior for the fixed effects, the weights applied to the design matrix reproduce the corresponding population numbers.

Usage

```
uweights(x, areaID = NULL, forTotal = FALSE)
```

Arguments

x	sae object.
areaID	if left unspecified (NULL), weights corresponding to the overall (aggregated) estimate are returned. Otherwise weights that reproduce the estimate for a specific area are returned.
forTotal	if FALSE weights will be divided by the population size.

Value

An object of class weights.

See Also

summary.weights, plot.weights

Examples

```
d <- generateFakeData()

# compute small area estimates
sae <- fSAE(y0 ~ x + area2, data=d$sam, area="area", popdata=d$Xpop, method="hybrid", keep.data=TRUE)

# compute unit weights
w <- uweights(sae, forTotal=TRUE)
summary(w)  # summary statistics
plot(w)  # histogram of weights
# checks
all.equal(sum(w * sae$y), sum(EST(sae) * sae$Narea))
all.equal(colSums(w * as.matrix(sae$X)), colSums(sae$Xp * sae$Narea))
```

Index