Package: rsae (via r-universe)

September 8, 2024

Type Package

Title Robust Small Area Estimation

Version 0.3

Description Empirical best linear unbiased prediction (EBLUP) and robust prediction of the area-level means under the basic unit-level model. The model can be fitted by maximum likelihood or a (robust) M-estimator. Mean square prediction error is computed by a parametric bootstrap.

License GPL-3

URL https://github.com/tobiasschoch/rsae

BugReports https://github.com/tobiasschoch/rsae/issues

Encoding UTF-8

NeedsCompilation yes

LazyData true

Depends R (>= 3.5.0)

Imports stats, graphics

Suggests knitr, rmarkdown, robustbase

VignetteBuilder knitr, rmarkdown

Repository https://tobiasschoch.r-universe.dev

RemoteUrl https://github.com/tobiasschoch/rsae

RemoteRef HEAD

RemoteSha 8053e7d64c4451b9c2ee067b67a1fecede77be8e

Contents

`rsae-package` *Robust Small Area Estimation*

Description

The package implements methods to fit the basic unit-level model (also known as model type "B" in Rao, 2003, or nested-error regression model in Battese et al., 1988), to predict area-specific means by the empirical best linear unbiased predictor (EBLUP) or a robust prediction method (Copt and Victoria-Feser, 2009; Heritier et al., 2011), and to compute the mean square prediction error of the predicted area-level means by a parametric bootstrap (Sinha and Rao, 2009; see also Hall and Maiti, 2006; Lahiri, 2003).

The methods are discussed in Schoch (2012).

Details

Implemented methods:

- maximum likelihood estimator
- Huber-type M-estimator (RML II of Richardson and Welsh, 1995, not the method proposed in Sinha and Rao, 2009); see Schoch (2012) for details

How to: Data analysis involves the following steps:

1. prepare the data/ specify the model for estimation; see `saemodel()`
2. fit the model by various (robust) methods; see `fitsaemodel()`
3. (robustly) predict the random effects and the area means; see `robpredict()`

References

Battese, G. E., Harter, R. M., and W.A. Fuller (1988). An error component model for prediction of county crop areas using. *Journal of the American Statistical Association* **83**, 28–36. doi:10.2307/2288915

Copt, S. and M.-P. Victoria-Feser (2009). *Robust Predictions in Mixed Linear Models*, Research Report, University of Geneva.

Lahiri, P. (2003). On the impact of bootstrap in survey sampling and small area estimation. *Statistical Science* **18**, 199–210. doi:10.1214/ss/1063994975

Hall, P. and T. Maiti (2006). On parametric bootstrap methods for small area prediction. *Journal of the Royal Statistical Society. Series B* **68**, 221–238. doi:10.1111/j.14679868.2006.00541.x

Heritier, S., Cantoni, E., Copt, S., and M.-P. Victoria-Feser (2009). *Robust methods in Biostatistics*, New York: John Wiley and Sons.

Rao, J.N.K. (2003). *Small Area Estimation*, New York: John Wiley and Sons.

Richardson, A.M. and A.H. Welsh (1995). Robust restricted maximum likelihood in mixed linear model. *Biometrics* **51**, 1429–1439. doi:10.2307/2533273

Schoch, T. (2012). Robust Unit-Level Small Area Estimation: A Fast Algorithm for Large Datasets. *Austrian Journal of Statistics* **41**, 243–265. doi:10.17713/ajs.v41i4.1548

Sinha, S.K. and J.N.K. Rao (2009). Robust small area estimation. *Canadian Journal of Statistics* **37**, 381–399. doi:10.1002/cjs.10029

`fitsaemodel` *Fitting SAE Models*

Description

`fitsaemodel` fits SAE models that have been specified by `saemodel()` (or synthetic data generated by `makedata()`) for various (robust) estimation methods.

Usage

```
fitsaemodel(method, model, ...)
convergence(object)

## S3 method for class 'fit_model_b'
print(x, digits = max(3L, getOption("digits") - 3L),
    ...)
## S3 method for class 'summary_fit_model_b'
print(x, digits = max(3L, getOption("digits")
    - 3L), ...)
## S3 method for class 'fit_model_b'
summary(object, ...)
## S3 method for class 'fit_model_b'
coef(object, type = "both", ...)
```

Arguments

`method`	`[character]` estimation method; `method = "ml"` for (non-robust) maximum likelihood or `method = "huberm"` for Huber-type M-estimation.
`model`	an object of class `"saemodel"`, i.e., a SAE model; see `saemodel()`.
`digits`	`[integer]` number of digits printed by the `print()` and `summary()` methods.
`object`	an object of class `"fit_model_b"`.
`x`	an object of class `"fit_model_b"`.
`type`	`[character]` name of the effects to be extracted by the `coef` method; it can take one of the following possibilities: `"both"` (extracts fixed and random effects; default), `"ranef"` (only random effects), or `"fixef"` (only fixed effects).
`...`	additional arguments passed on to `fitsaemodel.control()`.

Details

Function `fitsaemodel()` computes the following estimators:

- maximum likelihood (ML): `method = "ml"`,
- Huber-type M-estimation: `method = "huberm"`; this method is called RML II by Richardson and Welsh (1995); see Schoch (2012)

Maximum likelihood: The ML is **not** robust against outliers.

Huber-type M-estimation: The call for the Huber-type M-estimator (with Huber psi-function) is: `fitsaemodel(method = "huberm", model, k)`, where `k` is the robustness tuning constant of the Huber psi-function, $k \in (0, \infty]$.

By default, the computation of the M-estimator is initialized by a robust estimate that derives from a fixed-effects model (centered by the median instead of the mean); see Schoch (2012) for the details.

If the data are supposed to be heavily contaminated (**or** if the **default algorithm did not converge**), one may try to initialize the algorithm underlying `fitsaemodel()` by a high breakdown-point estimate. The package offers two initialization methods: **NOTE:** the **robustbase** package (Maechler et al., 2022) must be installed to use this functionality.

- `init = "lts"`: least trimmed squares (LTS) regression from **robustbase**; see ltsReg() and Rousseeuw and Van Driessen (2006),
- `init = "s"`: regression S-estimator from **robustbase**; see lmrob() and Maronna et al. (2019).

For small and medium size datasets, both methods are equivalent in terms of computation time. For large data, the S-estimator is considerably faster.

Implementation: The methods are computed by (nested) iteratively re-weighted least squares and a derivative of Richard Brent's `zeroin` algorithm; see Brent (2013, Chapter 4). The functions depend on the subroutines in BLAS (Blackford et al., 2002) and LAPACK (Anderson et al., 2000); see Schoch (2012).

Value

An instance of the class `"fitmodel"`

References

Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J., et al. (2000). *LAPACK users' guide* (3rd ed.). Philadelphia: Society for Industrial and Applied Mathematics (SIAM).

Blackford, L.S., Petitet, A., Pozo, R., Remington, K., Whaley, R.C., Demmel, J., et al. (2002). An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software* **28**, 135–151. doi:10.1145/567806.567807

Brent, R.P. (2013). *Algorithms for minimization without derivatives*. Mineola (NY): Dover Publications Inc. (This publication is an unabridged republication of the work originally published by Prentice-Hall Inc., Englewood Cliffs, NJ, in 1973).

Maechler, M., Rousseeuw, P., Croux, C., Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Verbeke, T., Koller, M., Conceicao, E.L.T. and M. Anna di Palma (2022). robustbase: Basic Robust Statistics R package version 0.95-0. https://CRAN.R-project.org/package=robustbase

Maronna, R.A., Martin, D., V.J. Yohai and M. Salibian-Barrera (2019): *Robust statistics: Theory and methods*. Chichester: John Wiley and Sons, 2nd ed.

Richardson, A.M. and A.H. Welsh (1995). Robust restricted maximum likelihood in mixed linear model. *Biometrics* **51**, 1429–1439. doi:10.2307/2533273

Rousseeuw, P. J. and K. Van Driessen (2006). Computing LTS regression for large data sets. *Data Mining and Knowledge Discovery* **12**, 29–45. doi:10.1007/s1061800500244

Schoch, T. (2012). Robust Unit-Level Small Area Estimation: A Fast Algorithm for Large Datasets. *Austrian Journal of Statistics* **41**, 243–265. doi:10.17713/ajs.v41i4.1548

See Also

`fitsaemodel.control()`

Examples

```
# use the landsat data
head(landsat)

# define the saemodel using the landsat data
model <- saemodel(formula = HACorn ~ PixelsCorn + PixelsSoybeans,
    area = ~CountyName,
    data = subset(landsat, subset = (outlier == FALSE)))

# summary of the model
summary(model)

# maximum likelihood estimates
fitsaemodel("ml", model)

# Huber M-estimate with robustness tuning constant k = 2
m <- fitsaemodel("huberm", model, k = 2)
m

# summary of the fitted model/ estimates
summary(m)

# obtain more information about convergence
convergence(m)

# extract the fixed effects
coef(m, "fixef")

# extract the random effects
coef(m, "ranef")

# extract both
coef(m)
```

`fitsaemodel.control` *Tuning Parameters of* `fitsaemodel`

Description

This function is used to define global settings and parameters that are used by `fitsaemodel()`.

Usage

```
fitsaemodel.control(niter = 40, iter = c(200, 200), acc = 1e-05,
    dec = 0, decorr = 0, init = "default", k_Inf = 20000, ...)
```

Arguments

`niter`	`[integer]` the maximum number of outer-loop iterations (default: `niter = 40`).
`iter`	`[integer]` the maximum number of inner-loop iterations. It can be a vector of size 2. The first element of the vector refers to the estimation of the regression coefficients β; the second element refers to the estimation of the variance of the unit-level errors, v; the maximum number of iterations used to compute the ratio of variances, d, cannot be modified (default: `iter = c(200, 200)`).
`acc`	`[numeric]` numeric tolerance used in the termination rule of the iterative updating algorithms. It can be a vector of size 4. The positions `1:4` of the vector `acc` refer to `1`: (overall) outer-loop, `2`: regression coefficients, β, `3`: variance component, v, `4`: ratio of variances d; default: `acc = 1e-05`.
`dec`	`[integer]` type of matrix square root (decomposition); `dec = 0` for eigenvalue decomposition (default) or `dec = 1` for Cholesky decomposition.
`decorr`	`[integer]` type of decorrelation of the residuals; `decorr = 0`: no robust decorrelation (default); `decorr = 1`: means are replaced by medians.
`init`	`[character]` method by which the main algorithm is initialized. By default, `init = "default"` the algorithm is initialized by a robust fixed-effects estimator; alternatively, (and provided that the **robustbase** package is installed) one may choose one of the high-breakdown-point initial estimators: "lts" (fast least-trimmed squares, LTS, regression) or "s" (regression S-estimator). For more details on the initialization methods, see documentation of `fitsaemodel()`.
`k_Inf`	`[numeric]` tuning constant of the robust estimator that represents infinity (default: `k_Inf = 20000`).
`...`	additional arguments (not used).

Details

Changing the default values of the parameters may result in failure of convergence or loss of convergence speed.

Value

A list with entries

- niter
- iter
- acc
- k_Inf
- init
- dec
- decorr
- add

See Also

fitsaemodel()

Examples

```
# use the landsat data
head(landsat)

# define the saemodel using the landsat data
model <- saemodel(formula = HACorn ~ PixelsCorn + PixelsSoybeans,
    area = ~CountyName,
    data = subset(landsat, subset = (outlier == FALSE)))

# summary of the model
summary(model)

# obtain the maximum likelihood estimates with, for instance, 'niter = 50'
# number of outer-loop iterations (by default: niter = 40). Here, we use
# 'niter = 50' for the sake of demonstration, not because it is needed.
fitsaemodel("ml", model, niter = 50)
```

---

landsat *LANDSAT Data: Prediction of County Crop Areas Using Survey and Satellite Data*

Description

The landsat data is a compilation of survey and satellite data from Battese et al. (1988). It consists of data on segments (primary sampling unit; 1 segement approx. 250 hectares) under corn and soybeans for 12 counties in north-central Iowa.

Usage

```
data(landsat)
```

Format

A data frame with 37 observations on the following 10 variables.

`SegmentsInCounty` number of segments per county.

`SegementID` sample segment identifier (per county).

`HACorn` hectares of corn for each sample segment (as reported in the June 1978 Enumerative Survey).

`HASoybeans` hectares of soybeans for each sample segment (as reported in the June 1978 Enumerative Survey).

`PixelsCorn` number of pixels classified as corn for each sample segment (LANDSAT readings).

`PixelsSoybeans` number of pixels classified as soybeans for each sample segment (LANDSAT readings).

`MeanPixelsCorn` county-specific mean number of pixels classified as corn.

`MeanPixelsSoybeans` county-specific mean number of pixels classified as soybeans.

`outlier` outlier indicator; observation number 33 is flagged as outlier.

`CountyName` county names (factor variable): Cerro Gordo, Hamilton, Worth, Humboldt, Franklin, Pocahontas, Winnebago, Wright, Webster, Hancock, Kossuth, Hardin.

Details

The `landsat` data in Battese et al. (1988) is a compilation of the LANDSAT satellite data from the U.S. Department of Agriculture (USDA) and the 1978 June Enumerative Survey.

Survey data: The survey data on the areas under corn and soybeans (reported in hectares) in the 37 segments of the 12 counties (north-central Iowa) have been determined by USDA Statistical Reporting Service staff, who interviewed farm operators. A segment is about 250 hectares.

Satellite data: For the LANDSAT satellite data, information is recorded as "pixels". A pixel is about 0.45 hectares. The USDA has been engaged in research toward transforming satellite information into good estimates of crop areas at the individual pixel and segments level. The satellite (LANDSAT) readings were obtained during August and September 1978.

Data for more than one sample segment are available for several counties (i.e., unbalanced data).

Observations No. 33 has been flaged as outlier; see Battese et al. (1988, p. 28).

Source

The `landsat` data is from Table 1 of Battese et al. (1988, p. 29).

References

Battese, G. E., Harter, R. M., and W.A. Fuller (1988). An error component model for prediction of county crop areas using. *Journal of the American Statistical Association* **83**, 28–36. doi:10.2307/2288915

See Also

landsat_means

Examples

```
head(landsat)
```

---

landsat_means *Means of the LANDSAT Data for Corn and Soybeans*

Description

The landsat data is a compilation of survey and satellite data from Battese et al. (1988). The county-specific population means of pixels of the segments under corn and soybeans, respectively, are available in the data.frame landsat_means.

Usage

```
data(landsat_means)
```

Format

A data frame with 12 observations (counties) on the following 3 variables.

(intercept) all ones.

MeanPixelsCorn county-specific mean of number of pixels classified as corn (LANDSAT readings).

MeanPixelsSoybeans county-specific number of pixels classified as soybeans (LANDSAT readings).

Details

The data.frame landsat_means is an aggregation of the data.frame landsat.

References

Battese, G. E., Harter, R. M., and W.A. Fuller (1988). An error component model for prediction of county crop areas using. *Journal of the American Statistical Association* **83**, 28–36. doi:10.2307/2288915

Examples

```
head(landsat_means)
```

makedata *Synthetic Data Generation for the Basic Unit-Level SAE Model*

Description

This function generates synthetic data (possibly contaminated by outliers) for the basic unit-level SAE model.

Usage

```
makedata(seed = 1024, intercept = 1, beta = 1, n = 4, g = 20, areaID = NULL,
         ve = 1, ve.contam = 41, ve.epsilon = 0, vu = 1, vu.contam = 41,
         vu.epsilon = 0)
```

Arguments

seed	[integer] seed value used in set.seed (default seed = 1024).
intercept	[numeric] or [NULL] value of the intercept of the fixed-effects model or NULL for a model without intercept (default: intercept = 1).
beta	[numeric vector] value of the fixed-effect coefficients (without intercept; default: beta = 1). For each given coefficient, a vector of realizations is drawn from the standard normal distribution.
n	[integer] number of units per area in balanced-data setups (default: n = 4).
g	[integer] number of areas (default: g = 20).
areaID	[integer vector] or [NULL]. If one attempts to generate synthetic unbalanced data, one calls makedata with a vector, the elements of which area identifiers. This vector should contain a series of (integer valued) area IDs. The number of areas is set equal to the number unique IDs.
ve	[numeric] nonnegative value of model/ residual variance.
ve.contam	[numeric] nonnegative value of model variance of the outlier part in a mixture distribution (Tukey-Huber-type contamination model) $e = (1-h)N(0, ve) + hN(0, ve.contam)$.
ve.epsilon	[numeric] value in $[0, 1]$ that defines the relative number of outliers (i.e., epsilon or h in the contamination mixture distribution). Typically, it takes values between 0 and 0.5 (but it is not restricted to this interval).
vu	[numeric] value of the (area-level) random-effect variance.
vu.contam	[numeric] nonnegative value of the (area-level) random-effect variance of the outlier part in the contamination mixture distribution.
vu.epsilon	[numeric] value in $[0, 1]$ that defines the relative number of outliers in the contamination mixture distribution of the (area-level) random effects.

Details

Let y_i denote an area-specific n_i-vector of the response variable for the areas $i = 1, ..., g$. Define a $(n_i \times p)$-matrix X_i of realizations from the std. normal distribution, $N(0, 1)$, and let β denote a p-vector of regression coefficients. Now, the y_i are drawn using the law $y_i \sim N(X_i\beta, v_e I_i + v_u J_i)$ with v_e and v_u the variances of the model error and random-effect variance, respectively, and I_i and J_i denoting the identity matrix and matrix of ones, respectively.

In addition, we allow the distribution of the model/residual and area-level random effect to be contaminated (cf. Stahel and Welsh, 1997). Notably, the laws of $e_{i,j}$ and u_i are replaced by the Tukey-Huber contamination mixture:

- $e_{i,j} \sim (1 - \epsilon^{ve})N(0, v_e) + \epsilon^{ve} N(0, v_e^{\epsilon})$
- $u_i \sim (1 - \epsilon^{vu})N(0, v_u) + \epsilon^{vu} N(0, v_u^{\epsilon})$

where ϵ^{ve} and ϵ^{vu} regulate the degree of contamination; v_e^{ϵ} and v_u^{ϵ} define the variance of the contamination part of the mixture distribution.

Four different contamination setups are possible:

- no contamination (i.e., `ve.epsilon = vu.epsilon = 0`),
- contaminated model error (i.e., `ve.epsilon != 0` and `vu.epsilon = 0`),
- contaminated random effect (i.e., `ve.epsilon = 0` and `vu.epsilon != 0`),
- both are conaminated (i.e., `ve.epsilon != 0` and `vu.epsilon != 0`).

Value

An instance of the class `saemodel`.

References

Schoch, T. (2012). Robust Unit-Level Small Area Estimation: A Fast Algorithm for Large Datasets. *Austrian Journal of Statistics* **41**, 243–265. doi:10.17713/ajs.v41i4.1548

Stahel, W. A. and A. Welsh (1997). Approaches to robust estimation in the simplest variance components model. *Journal of Statistical Planning and Inference* **57**, 295–319. doi:10.1016/S0378-3758(96)00050X

See Also

`saemodel()`, `fitsaemodel()`

Examples

```
# generate a model with synthetic data
model <- makedata()
model

# summary of the model
summary(model)
```

robpredict *Robust Prediction of Random Effects, Fixed Effects, and Area-Specific Means*

Description

Function robpredict() predicts the area-level means by (1) the empirical best linear unbiased predictor (EBLUP) or (2) a robust prediction method which is due to Copt and Victoria-Feser (2009). In addition, the function computes the mean square prediction error (MSPE) of the predicted area-level means by a parametric bootstrap method.

Usage

```
robpredict(fit, areameans = NULL, k = NULL, reps = NULL, seed = 1024,
    progress_bar = TRUE)

## S3 method for class 'pred_model_b'
print(x, digits = max(3L, getOption("digits") - 3L),
    ...)
## S3 method for class 'pred_model_b'
plot(x, type = "e", sort = NULL, ...)
## S3 method for class 'pred_model_b'
residuals(object, ...)
## S3 method for class 'pred_model_b'
as.matrix(x, ...)
## S3 method for class 'pred_model_b'
head(x, n = 6L, ...)
## S3 method for class 'pred_model_b'
tail(x, n = 6L, keepnums = TRUE, addrownums, ...)
```

Arguments

fit	an object of class fit_model_b; a fitted SAE model.
areameans	[matrix] or [NULL] area-level means of dimension (g, p), where g and p denote, respectively, the number of areas and number of fixed-effects terms in the regression model (incl. intercept). By default, areadata = NULL which implies that the predictions are base on the data used in fitting the model (not new data).
k	[numeric] or [NULL] robustness tuning constant (of the Huber psi-function) for robust prediction. Note that k does not necessarily have to be the same as the k that has been used in fitsaemodel(). By default, k = NULL which implies that the tuning constant specified in fitsaemodel() is used.
reps	[integer] or [NULL] number of bootstrap replicates for the computation of the mean squared prediction error (MSPE). If reps = NULL the MSPE is not computed.
seed	[integer] a positive integer used as argument seed in set.seed() to specify the random seed.

progress_bar	[logical] whether a progress bar is displayed for the parametric bootstrap; see **NOTE** below.
x	an object of class "pred_model_b".
digits	[integer] number of digits to be printed by.
type	[character] type of plot method: "e" (error bars; default) or "l" (lines).
sort	[character] or [NULL] if sort = "means", the predicted means are plotted in ascending order (default: sort = NULL); similarly, with sort = "fixef" and sort = "ranef" the predicted means are sorted along the fixed effects or the random effects, respectively.
object	an object of class fit_model_b.
n	[integer] vector of length up to dim(x), i.e., number of areas.
keepnums	in each dimension, if no names in that dimension are present, create them using the indices included in that dimension. Ignored if dim(x) is NULL or its length 1.
addrownums	deprecated - keepnums should be used instead.
...	additional arguments (not used).

Details

Function robpredict() computes predictions of the area-level means and—if required—an estimate of the area-specific mean square prediction error (MSPE).

Prediction of area-level means
- Case 1: If areameans is a matrix with area-level means of the explanatory variables, then the computation of the fixed effects effects are based on areameans.
- Case 2: If areameans = NULL, then the predictions are based on the sample data that have been used to fit the model.

Mean square prediction error
- If reps = NULL, the number of bootstrap replicates is not specified; hence, MSPE is not computed.
- If reps is a positive integer and areameans is not NULL (see Case 1 above), then a (robust) parametric bootstrap estimate of MSPE is computed as proposed by Sinha and Rao (2009); see also Lahiri (2003) and Hall.

Robustness
- The EBLUP obtains if k = NULL, i.e., if the robustness tuning constant k is unspecified.
- Robust predictions of the area-level means are computed if k is a nonnegative real number. Small values of k imply that outliers are heavily downweighted; formally, the EBLUP corresponds to choosing the tuning constant k equal to infinity. The value of the tuning constant k specified in robpredict() can be different from the tuning constant k used in fitting the model. The robust prediction method is due to Copt and Victoria-Feser (2009); see also Heritier et al. (2009, 113-114) and differs from the method in Sinha and Rao (2009).

Value

An instance of the S3 class pred_model_b

NOTE

Users of `Rgui.exe` on Windows are recommended to call `robpredict()` with argument `progress_bar = FALSE` because `Rgui.exe` does not handle calls to `txtProgressBar()` well (the execution time of the same job increases and it tends to stall the execution of R). Users of `R-Studio` and `Rterm.exe` are **not** affected.

References

Copt, S. and M.-P. Victoria-Feser (2009). *Robust Predictions in Mixed Linear Models*, Research Report, University of Geneva.

Lahiri, P. (2003). On the impact of bootstrap in survey sampling and small area estimation. *Statistical Science* **18**, 199–210. doi:10.1214/ss/1063994975

Hall, P. and T. Maiti (2006). On parametric bootstrap methods for small area prediction. *Journal of the Royal Statistical Society. Series B* **68**, 221–238. doi:10.1111/j.14679868.2006.00541.x

Heritier, S., Cantoni, E., Copt, S., and M.-P. Victoria-Feser (2009). *Robust methods in biostatistics*. New York: John Wiley and Sons.

Schoch, T. (2012). Robust Unit-Level Small Area Estimation: A Fast Algorithm for Large Datasets. *Austrian Journal of Statistics* **41**, 243–265. doi:10.17713/ajs.v41i4.1548

Sinha, S.K. and J.N.K. Rao (2009). Robust small area estimation. *Canadian Journal of Statistics* **37**, 381–399. doi:10.1002/cjs.10029

See Also

`saemodel()`, `makedata()`, `fitsaemodel()`

Examples

```
# use the landsat data
head(landsat)

# set up the model
model <- saemodel(formula = HACorn ~ PixelsCorn + PixelsSoybeans,
    area = ~CountyName,
    data = subset(landsat, subset = (outlier == FALSE)))

# summary of the model
summary(model)

# Huber M-estimate with robustness tuning constant k = 2
m <- fitsaemodel("huberm", model, k = 2)
m

# summary of the fitted model/ estimates
summary(m)

# robust prediction of the random effects and the area-level means (robust
# EBLUP) using the counts-specific means (landsat_means)
head(landsat_means)
```

```
# for robust prediction, we use the robustness tuning constant 'k = 1.8'
m_predicted <- robpredict(m, landsat_means, k = 1.8)
head(m_predicted)

# extract prediction as matrix
as.matrix(m_predicted)

# extract residuals from the predictions
head(residuals(m_predicted))

# prediction incl. MSPE; parametric bootstrap with only 'reps = 10'
# replications (for demonstration purposes; in practice, 'reps' should be
# considerably larger)
m_predicted_mspe <- robpredict(m, landsat_means, k = 1.8, reps = 10,
                               progress_bar = FALSE)
head(m_predicted_mspe)
```

`saemodel` *Setting Up a SAE Model*

Description

Function `saemodel()` is used to specify a model. Once a model has been specified, it can be fitted using `fitsaemodel()` by different estimation methods.

Usage

```
saemodel(formula, area, data, type = "b", na.omit = FALSE)

## S3 method for class 'saemodel'
print(x, ...)
## S3 method for class 'saemodel'
summary(object, ...)
## S3 method for class 'saemodel'
as.matrix(x, ...)
```

Arguments

`formula`	a `formula` object of describing the fixed-effects part of the model, with the response on the RHS of the ~ operator and the terms or regressors, separated by + operators, on the LHS of the formula.
`area`	a one-sided `formula` object. A ~ operator followed by only one single term defining the area-specific random-effect part.
`data`	data.frame.
`type`	`[character]` "a" or "b" refering to J.N.K. Rao's definition of model type A (area-level model) or B (unit-level model); default is "b".
`na.omit`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds. Note that none of the algorithms can cope with missing values.

x	an object of class "saemodel".
object	an object of the class "saemodel".
...	additional arguments (not used).

Details

Function saemodel() is used to specify a model.

- model is a symbolic description (formula of the fixed-effects model to be fitted.
  A typical model has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response (explanatory variables); see formula.
  A formula has an implied intercept term. To remove this use either y ~ x - 1 or y ~ 0 + x; see formula for more details of allowed formulae.
- area is a symbolic description (formula) of the random effects (nested error structure). It must be right-hand side only formula consisting of one term, e.g., ~ areaDefinition.

The data must no contain missing values.

The design matrix (i.e., matrix of the explanatory variables defined the right-hand side of model) must have full column rank; otherwise execution is terminated by an error.

Once a model has been specified, it can be fitted by fitsaemodel().

Value

An instance of the S3 class "saemodel"

References

Rao, J.N.K. (2003). *Small Area Estimation*, New York: John Wiley and Sons.

See Also

makedata(), fitsaemodel()

Examples

```
# use the landsat data
head(landsat)

# set up the model
model <- saemodel(formula = HACorn ~ PixelsCorn + PixelsSoybeans,
    area = ~CountyName,
    data = subset(landsat, subset = (outlier == FALSE)))

# summar of the model
summary(model)
```

Index