Package 'rsae'

February 20, 2015

Type Package

Title Robust Small Area Estimation

Version 0.1-5

Date 2014-02-12

Author Tobias Schoch

Maintainer Tobias Schoch <tobias.schoch@gmail.com>

Description Robust Small Area Estimation. Robust Basic Unit- and Area-Level Models

Suggests robustbase, nlme

License GPL (>= 2) | FreeBSD

LazyLoad yes

Repository CRAN

Repository/R-Forge/Project rsae

Repository/R-Forge/Revision 4

Repository/R-Forge/DateTimeStamp 2014-02-13 15:06:52

Date/Publication 2014-02-13 18:17:05

NeedsCompilation yes

R topics documented:

rsae-package *Robust Small Area Estimation*

Description

Computes robust basic unit- and area-level and predicts area-specific means

Details

Package:	rsae
Type:	Package
Version:	0.1-5
Date:	2014-02-12
Suggests:	robustbase, nlme
License:	GPL (>=2) | FreeBSD
LazyLoad:	yes

Implemented methods:

- maximum likelihood (as reference)
- Huber-type M-estimation

How to:

Data analysis with rsae involves the following steps:

1. prepare the data/ set up 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

Author(s)

Tobias Schoch

Maintainer: Tobias Schoch <tobias.schoch@gmail.com>

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).

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.

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.

Brent, R.P. (1973): *Algorithms for minimization without derivatives*. Englewood Cliffs, NJ: Prentice-Hall.

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

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.

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

Maronna, R.A., Martin, D., and V.J. Yohai (2006): *Robust statistics: Theory and methods*. Chichester: John Wiley.

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.

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

Schoch, T. (2012 under revision): Robust Unit-Level Small Area Estimation: A Fast Algorithm for Large Datasets, *Austrian Journal of Statistics*.

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

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.

`fitsaemodel` *Fit SAE model using various methods*

Description

`fitsaemodel` is the workhorse function. It estimates SAE models that have been set up by `saemodel` (or synthetic data generated by `makedata`) by various (robust) estimation methods.

Usage

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

convergence(object)

## S3 method for class 'fitsaemodel'
print(x, digits=6, ...)
## S3 method for class 'fitsaemodel'
summary(object, digits=6, ...)
## S3 method for class 'fitsaemodel'
coef(object, type="both", ...)
```

Arguments

method	character string defining the method to be used; currently, either method="ml" for (non-robust) maximum likelihood or method="huberm" for Huber-type M-estimation
model	a "saemodel" object (i.e., a SAE model; see saemodel)
x	used by the print method
digits	used by the print and summary methods; number of decimal places to be shown
object	an object of the class "fitsaemodel"; i.e., a fitted model
type	character string use in the coef method; it can take one of the following possibilities: "both", "ranef", or "fixef". The first reports both, random and fixed effects (default).
...	additional arguments delivered to either fitsaemodel.control

Details

The function fitsaemodel is a wrapper function that calls the algorithm associated with a particular method. Two methods are currently implemented

- maximum likelihood (method="ml"),
- Huber-type M-estimation (method="huberm"; cf. RML II of Richardson and Welsh, 1995).

Maximum likelihood: The call for ML is straightforward: fitsaemodel(method="ml", model), where model is a SAE model generated by saemodel. Note that ML is not a robust fitting method.

Huber-type M-estimation: The call for Huber-type M-estimaton (with Huber psi-function) is: fitsaemodel(method="huberm", model, k), where model is a SAE model generated by saemodel, and k is the robustness tuning constant of the Huber psi-function.

By default, the "huberm" method is initialized by means of pre-determined robust estimates of a fixed-effects model (centered by the median instead of the mean); see Schoch (2012) for the details.

If your data are supposed to be heavily contaminated (**or** if the **default algorithm did not converge**), you may initialize the fitsaemodel alogrithm with a high-breakdown-point estimate. The **rsae** package offers two methods to initialize the algorithm, "lts" and "s"; see below. **NOTE**, you have to install the **robustbase** package in order to use these methods. The initialization methods are called in the fitsaemodel device (as additional argument), using

- init="lts", for fast-LTS regression form **robustbase**; see also Rousseeuw and Van Driessen (2006),
- init="s", for a regression S-estimator from **robustbase**; see also Maronna et al. (2006).

For more details on the methods, you are refered to the documentation of **robustbase**. In general, for small to medium datasets, both methods are equivalent. For data with more than 50,000 observations, the S-estimator is considerably faster. (If the "ml" does not converge, you may initialize it analogously–though, it may be rather inefficient.)

Implementation: For both method="ml" and method="huberm", the estimates are obtained by means of a nested loop of IRWLS approaches and Brent's zeroin method (Brent, 1973). All the functions/subroutines are optimized to be rich in BLAS level-one operations (Blackford et al., 2002) and draw heavily on LAPACK (Anderson et al., 2000).

Value

An instance of the class `"fitmodel"`

Author(s)

Tobias Schoch

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.

Brent, R.P. (1973): *Algorithms for minimization without derivatives*. Englewood Cliffs, NJ: Prentice-Hall.

Maronna, R.A., Martin, D., and V.J. Yohai (2006): *Robust statistics: Theory and methods*. Chichester: John Wiley.

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

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

Schoch, T. (2012 under revision) Robust Unit-Level Small Area Estimation: A Fast Algorithm for Large Datasets, *Austrian Journal of Statistics*.

See Also

`fitsaemodel.control`

Examples

```
#generate the synthetic data/model
mymodel <- makedata()
#compute Huber M-estimation type estimates of the model "mymodel"
#robustness tuning constant k = 2
myfittedmodel <- fitsaemodel("huberm", mymodel, k=2)
myfittedmodel
#get a summary of the model
summary(myfittedmodel)
```

`fitsaemodel.control` *Tuning parameters of fitsaemodel*

Description

This function carries global settings and parameter definitions that are used by fitsaemodel (and its derivatives). Modifications of the parameters can be delivered as additional arguments in the fitsaemodel call.

Usage

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

Arguments

`niter`	integer, defining the maximum number of outer-loop iterations (default: `niter=40`)
`iter`	integer or vector of size 2, defining the maximum loops of the inner loops (default: `iter=c(200, 200)`; element 1 refers to beta; element 2 refers to v; note that d has an implicitly defined maxiter of 100 and cannot be modified)
`acc`	scalar or vector of size 4, defining the numeric tolerance used in the termination rule of the loops (default: `acc=1e-05`; the positions of elements in the vector of size 4 are: 1=acc outer-loop; 2=acc inner-loop beta; 3=acc inner-loop v; 4=acc inner-loop d).
`dec`	type of matrix square root (decomposition); `dec=0` for SVD (default) or `dec=1` for Cholesky decomposition
`decorr`	in order to robustly decorrelate the residuals, one has to chose `decorr=1` (default `decorr=0`)
`init`	a character string; specifies by what method the main algorithm is initialized; by default: `init="default"`; alternatively, (and provided that the **robustbase** package is installed) one may choose a high-breakdown-point initial estimate: either "lts" (fast LTS regression) or "s" (S-estimate of regression). For datasets with more than 100,000 observations, the former is rather slow. For more details on the initializing methods see the documentation of **robustbase** (`"ltsReg"` and `"lmrob.S"`).
`...`	(will be used in the future)

Details

Caution! Modifying the default values of the parameters may result in convergence failure and/or loss of convergence speed.

Value

(an object used by the robust methods)

Author(s)

Tobias Schoch

See Also

fitsaemodel

---

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

Description

The landsat data.frame is a compilation (by Battese et al., 1988) of survey and satellite data. 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; see Details, below.

Usage

```
data(landsat)
```

Format

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

SegmentsInCounty a numeric vector; no. of segments per county

SegementID a numeric vector; sample segment identifier (per county)

HACorn a numeric vector; hectares of corn for each sample segment (as reported in the June 1978 Enumerative Survey)

HASoybeans a numeric vector; hectares of soybeans for each sample segment (as reported in the June 1978 Enumerative Survey)

PixelsCorn a numeric vector; no. of pixels classified as corn for each sample segment (LANDSAT readings)

PixelsSoybeans a numeric vector; no. of pixels classified as soybeans for each sample segment (LANDSAT readings)

MeanPixelsCorn a numeric vector; county mean number of pixels classified as corn

MeanPixelsSoybeans a numeric vector; county mean number of pixels classified as soybeans

outlier a logical vector; flags observation no. 33 as outlier

CountyName a factor with levels (i.e., county names) Cerro Gordo Hamilton Worth Humboldt Franklin Pocahontas Winnebago Wright Webster Hancock Kossuth Hardin

Details

The `landsat` data is a compilation (by Battese et al., 1988) 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". The USDA has been engaged in research toward transforming satellite information into good estimates of crop areas at the individual pixel and segments level. A pixel is about 0.45 hectares. 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 (cf., Battese et al. (1988, p. 28).

Source

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

References

Battese, G.E, 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, pp. 28–36.

Examples

```
data(landsat)
```

`makedata` *Synthetic data generation for the basic unit-level SAE model (incl. outlier contamination)*

Description

This function serves for synthetically generating data with area-level variation. It has been written to test several estimating methods. In addition, one may introduce contamination to the laws of the model- and/or random effects (see Details, below).

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	an integer, defining the set.seed (default seed=1024)
intercept	either a scalar as intercept of the fixed-effects model or NULL (default: intercept=1)
beta	scalar or vector defining the fixed-effect coefficients (default: beta=1). For each given coefficient, a vector of realizations is drawn from the standard normal distribution.
n	integer, defining the number of units per area in balanced-data setups (default: n=4)
g	integer, defining the number of areas (default: g=20)
areaID	by default areaID=NULL. If one attempts to generate synthetic unbalanced data, one may call 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; see the rsae Vignette for more details.
ve	scalar, defining the model/ residual variance
ve.contam	scalar, defining the model variance of the outlier part in a mixture distribution (Tuckey-Huber-type contamination model). e = (1-h)*N(0, ve) + h*N(0, ve.contam)
ve.epsilon	scalar, defining 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	scalar, defining the (area-level) random-effect variance
vu.contam	scalar, defining the (area-level) random-effect variance of the outlier part in the contamination mixture distribution (cf., ve.contam)
vu.epsilon	scalar, defining the relative number of outliers in the contamination mixture distribution of the (area-level) random effects (cf., ve.epsilon)

Details

The function makedata generates synthetic datasets that may be used to study the behavior of different estimating methods. 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_e^{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

Instance of the class saemodel.

Author(s)

Tobas Schoch

References

Stahel, W.A. and A. Welsh (1997): Approaches to robust estimation in the simplest variance components model, *Journal of Inference and Statistical Planning* 57, pp. 295-319.

Examples

```
#generate synthetic data
mymodel <- makedata()
```

---

robpredict — *Robust prediction of random effects, fixed effects, and area-specific means*

Description

The function robpredict robustly predicts the random effects, fixed effects, and area-specific means under the model. As concerned with robustly predicting the realizations of the random effects, we rely on the method of Copt and Victoria-Feser (cf. Heritier et al., 2009, 113–114); not the method of Sinha and Rao (2009).

Usage

```
robpredict(fit, areameans=NULL, k=NULL, reps=NULL)

## S3 method for class 'meanssaemodel'
print(x, digits=4, ...)
## S3 method for class 'meanssaemodel'
plot(x, y=NULL, type="e", sort=NULL, ...)
## S3 method for class 'meanssaemodel'
residuals(object, ...)
```

Arguments

fit	a fitted SAE model; object of class fitsaemodel
areameans	numeric matrix (typically, with area-level means); the no. of rows must be equal to the no. of areas; the no. of columns must be equal to the no. of fixed-effects coefficients (incl. intercept). By default: areadata=NULL, i.e., predictions are based on those data that have been used to estimate the model.
k	robustness tuning constant (of the Huber psi-function) for robust prediction. Notice that k does not necessarily be the same as the k that has been used in fitsaemodel. By default, k is equal to the tuning constant used in estimating the model parameters.
reps	number (integer) of bootstrap replicates for mean squared prediction error; default: reps=NULL
x	object of the class "meanssaemodel"; this argument is only used in the print method.
digits	integer, defining the number of decimal places to be shown in the print method (default: digits=4)
y	has no meaning, yet! (default: y=NULL; needs to included in the args list, because it is part of plot's generic arg definition)
type	character specifying the plot method; either "e" (error bars; default) or "l" (lines).
sort	only used in the plot method; if sort="means", the predicted means are ploted 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	object of the class fitsaemodel; a fitted model used in the residuals method.
...	not used

Details

The robpredict function enables the following modes of prediction:

- if areameans=NULL, then the predictions are exclusively based on the sample values,
- if robpredict is called with areameans (i.e., matrix with area-specific means of the auxiliary data of conformable size), then the fixed-effect predictions and thus also the predictions of the area-specific means are based on the auxiliary data,
- if, in addition to specifying areameans, one specifies also the number of bootstrap replications (i.e., reps; some positive integer), the function computes area-specific mean square prediction error (MSPE) estimates for the area-level means. The MSPE is obtained, in line with Sinha and Rao (2009), from a (robust) parametric bootstrap; see Lahiri (2003) and Hall and Maiti (2006) for more details.

The tuning constant k regulates the degree of robustness (i.e., degree of winsorization of the Huber psi-function) when predicting the random effects. If k is sufficiently large (ideally, if k is equal to infinity), the predictions correspond to the EBLUP.

Value

Instance of the S3 class `meanssaemodel`

Author(s)

Tobias Schoch

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.

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.

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

Sinha, S.K. and J.N.K. Rao (2009): Robust Small Area Estimation. *The Canadian Journal of Statistics* 37, 381–399.

Examples

```
#generate the synthetic data/model
mymodel <- makedata()
#compute Huber M-estimation type estimates of the model "mymodel"
#robustness tuning constant k = 2
myfittedmodel <- fitsaemodel("huberm", mymodel, k=2)
myfittedmodel
#get a summary of the model
summary(myfittedmodel)
#robustly predict the random effects and the area-level means.
#Here, we choose the robustness tuning constant k equal to 1.8
mypredictions <- robpredict(myfittedmodel, k=1.8)
mypredictions
```

`sae-internal` *Internal functions*

Description

These functions are for internal use only.

Author(s)

Tobias Schoch

See Also

rsae-package

---

saemodel *Set up a SAE model*

Description

saemodel is the workhorse function to set up a model (i.e., an instance of the "saemodel" class). It is the starting point of every model fitting exercise. Once a model has been initilized/ set up, we consider estimating its parameter.

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 two-sided linear formula object describing the fixed-effects part, 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	either "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 type="b"
na.omit	a logical indicating whether NA should be removed (default is FALSE). Note that none of the algorithms can cope with missing values.
x	an object of the class "saemodel" (this argument is implicitly used by the print and as.matrix methods)
object	an object of the class "saemodel" (this argument is implicitly used by the summary method)
...	not used

Details

The step of setting up a SAE model is the starting point of any (robust) SAE modeling exercise. (Use the makedata to generate a synthetic dataset; see also, below). Here, we have to define the fixed-effects- and random-effects part of the model, and to tell R what data it shall use.

Once a model has been initilized/ set up, we consider estimating its parameter; see `fitsaemodel`.

Value

Instance of the S3 class `"saemodel"`.

Author(s)

Tobias Schoch

References

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

See Also

`makedata`

Index