Package: robsurvey (via r-universe)

October 22, 2024

Type Package

Title Robust Survey Statistics Estimation

Version 0.7

Description Robust (outlier-resistant) estimators of finite population characteristics like of means, totals, ratios, regression, etc. Available methods are M- and GM-estimators of regression, weight reduction, trimming, and winsorization. The package extends the 'survey' <https://CRAN.R-project.org/package=survey> package.

License GPL (>= 2)

Classification/MSC-2010 62D05, 62F35

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

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

Encoding UTF-8

NeedsCompilation yes

LazyData true

Depends R (>= 3.5.0)

Imports grDevices, KernSmooth, stats, survey (>= 3.35-1), utils

Suggests hexbin, knitr, MASS, rmarkdown, wbacon

VignetteBuilder knitr, rmarkdown

Author Beat Hulliger [aut] (<https://orcid.org/0000-0001-5252-8606>), Tobias Schoch [aut, cre] (<https://orcid.org/0000-0002-1640-3395>), Martin Sterchi [ctr, com], R-core [ctb, cph] (for zeroin2.c)

Maintainer Tobias Schoch <tobias.schoch@fhnw.ch>

Repository CRAN

Date/Publication 2024-08-22 07:10:02 UTC

Contents

robsurvey-package *Package Overview*

Description

A key *design pattern* of the package is that the majority of the estimating methods is available in two "flavors":

- bare-bone methods
- survey methods

Bare-bone methods are stripped-down versions of the survey methods in terms of functionality and informativeness. These functions may serve users and package developers as building blocks. In particular, bare-bone functions *cannot compute* variances.

The survey methods are much more capable and depend, for variance estimation, on the **survey** package.

Basic Robust Estimators

Trimming:

- Bare-bone methods: `weighted_mean_trimmed` and `weighted_total_trimmed`
- Survey methods: `svymean_trimmed` and `svytotal_trimmed`

Winsorization:

- Bare-bone methods:
  - `weighted_mean_winsorized` and `weighted_total_winsorized`
  - `weighted_mean_k_winsorized` and `weighted_total_k_winsorized`
- Survey methods:
  - `svymean_winsorized` and `svytotal_winsorized`
  - `svymean_k_winsorized` and `svytotal_k_winsorized`

Dalen's estimators (weight reduction methods):

- Bare-bone methods: `weighted_mean_dalen` and `weighted_total_dalen`
- Survey methods: `svymean_dalen` and `svytotal_dalen`

M-estimators:

- Bare-bone methods:
  - `weighted_mean_huber` and `weighted_total_huber`
  - `weighted_mean_tukey` and `weighted_total_tukey`
  - `huber2` (weighted Huber Proposal 2 estimator)
- Survey methods:
  - `svymean_huber` and `svytotal_huber`
  - `svymean_tukey` and `svytotal_tukey`
  - `mer` (minimum estimated risk estimator)

Survey Regression (weighted least squares)

svyreg

Robust Regression and Ratio Estimation (weighted)

- Regression M-estimators: svyreg_huberM and svyreg_tukeyM
- Regression GM-estimators (Mallows and Schweppe): svyreg_huberGM and svyreg_tukeyGM
- Ratio M-estimators: svyratio_huber and svyratio_tukey

Note: The functions svyreg_huber and svyreg_tukey are deprecated, use instead svyreg_huberM and svyreg_tukeyM, respectively; see also robsurvey-deprecated.

Robust Generalized Regression (GREG) and Ratio Prediction of the Population Mean and Total

- Regression predictors: svymean_reg and svytotal_reg
- Ratio predictors: svymean_ratio and svytotal_ratio

Utility functions

- weighted_quantile and weighted_median
- weighted_mad and weighted_IQR
- weighted_mean and weighted_total
- weighted_line, weighted_median_line, and weighted_median_ratio

---

class_svyreg_rob *Utility Functions for Objects of Class svyreg_rob*

Description

Methods and utility functions for objects of class svyreg_rob.

Usage

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

## S3 method for class 'svyreg_rob'
summary(object, mode = c("design", "model", "compound"),
        digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'svyreg_rob'
coef(object, ...)

## S3 method for class 'svyreg_rob'
vcov(object, mode = c("design", "model", "compound"), ...)
```

```
## S3 method for class 'svyreg_rob'
SE(object, mode = c("design", "model", "compound"), ...)

## S3 method for class 'svyreg_rob'
residuals(object, ...)

## S3 method for class 'svyreg_rob'
fitted(object, ...)

## S3 method for class 'svyreg_rob'
robweights(object)

## S3 method for class 'svyreg_rob'
plot(x, which = 1L:4L,
     hex = FALSE, caption = c("Standardized residuals vs. Fitted Values",
     "Normal Q-Q", "Response vs. Fitted values",
     "Sqrt of abs(Residuals) vs. Fitted Values"),
  panel = if (add.smooth) function(x, y, ...) panel.smooth(x, y,
     iter = iter.smooth, ...) else points, sub.caption = NULL, main = "",
  ask = prod(par("mfcol")) < length(which) && dev.interactive(), ...,
  id.n = 3, labels.id = names(residuals(x)), cex.id = 0.75, qqline = TRUE,
  add.smooth = getOption("add.smooth"), iter.smooth = 3,
  label.pos = c(4, 2), cex.caption = 1, cex.oma.main = 1.25)
```

Arguments

`x`	object of class `svyreg_rob`.
`digits`	`[integer]` minimal number of significant digits.
`...`	additional arguments passed to the method.
`object`	object of class `svyreg_rob`.
`mode`	`[character]` mode of variance estimator: `"design"`, `"model"` or `"compound"` (default: `"design"`).
`which`	`[integer]` indicating which plots to be drawn; if a subset of the plots is required, you can specify a subset of the numbers `1:4`.
`hex`	`[logical]` if `TRUE`, a hexagonally binned plot is shown in place of a scatterplot.
`caption`	`[character]` captions to appear above the plots; vector of valid graphics annotations. It can be set to `""` or `NA` to suppress all captions.
`panel`	panel function. The useful alternative to `points`, `panel.smooth` can be chosen by `add.smooth = TRUE`.
`sub.caption`	`[character]` common title—above the figures if there are more than one; used as sub (s.`title`) otherwise. If `NULL`, as by default, a possible abbreviated version of `deparse(x$call)` is used.
`main`	`[character]` title to each plot—in addition to `caption`.
`ask`	`[logical]`; if `TRUE`, the user is *ask*ed before each plot, see `par(ask=.)`.
`id.n`	`[integer]` number of points to be labelled in each plot, starting with the most extreme.

labels.id	[character] vector of labels from which the labels for extreme points will be chosen. NULL uses observation numbers.
cex.id	[numeric] magnification of point labels.
qqline	[logical] indicating if a qqline should be added to the normal Q-Q plot.
add.smooth	[logical] indicating if a smoother should be added to most plots; see also panel above.
iter.smooth	[integer] the number of robustness iterations, the argument iter in panel.smooth.
label.pos	[numeric] positioning of labels, for the left half and right half of the graph respectively.
cex.caption	[numeric] controls the size of caption.
cex.oma.main	[numeric] controls the size of the sub.caption only if that is *above* the figures when there is more than one.

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

Variance For variance estimation (summary, vcov, and SE) three modes are available:

- "design": design-based variance estimator using linearization; see Binder (1983)
- "model": model-based weighted variance estimator (the sampling design is ignored)
- "compound": design-model-based variance estimator; see Rubin-Bleuer and Schiopu-Kratina (2005) and Binder and Roberts (2009)

Utility functions The following utility functions are available:

- summary gives a summary of the estimation properties
- plot shows diagnostic plots for the estimated regression model
- robweights extracts the robustness weights (if available)
- coef extracts the estimated regression coefficients
- vcov extracts the (estimated) covariance matrix
- residuals extracts the residuals
- fitted extracts the fitted values

References

Binder, D. A. (1983). On the Variances of Asymptotically Normal Estimators from Complex Surveys. *International Statistical Review* **51**, 279–292. doi:10.2307/1402588

Binder, D. A. and Roberts, G. (2009). Design- and Model-Based Inference for Model Parameters. In: *Sample Surveys: Inference and Analysis* ed. by Pfeffermann, D. and Rao, C. R. Volume 29B of *Handbook of Statistics*, Amsterdam: Elsevier, Chap. 24, 33–54 doi:10.1016/S01697161(09)002247

Rubin-Bleuer, S. and Schiopu-Kratina, I. (2005). On the Two-phase framework for joint model and design-based inference. *The Annals of Statistics* **33**, 2789–2810. doi:10.1214/009053605000000651

See Also

Weighted least squares: svyreg; robust weighted regression svyreg_huberM, svyreg_huberGM, svyreg_tukeyM and svyreg_tukeyGM

Examples

```
head(workplace)

library(survey)
# Survey design for simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Compute regression M-estimate with Huber psi-function
m <- svyreg_huberM(payroll ~ employment, dn, k = 14)

# Diagnostic plots (e.g., standardized residuals against fitted values)
plot(m, which = 1L)

# Plot of the robustness weights of the M-estimate against its residuals
plot(residuals(m), robweights(m))

# Utility functions
summary(m)
coef(m)
SE(m)
vcov(m)
residuals(m)
fitted(m)
robweights(m)
```

`class_svystat_rob` *Utility Functions for Objects of Class svystat_rob*

Description

Methods and utility functions for objects of class `svystat_rob`.

Usage

```
mse(object, ...)
## S3 method for class 'svystat_rob'
mse(object, ...)
## S3 method for class 'svystat'
mse(object, ...)
## S3 method for class 'svystat_rob'
summary(object, digits = max(3L,
```

```
        getOption("digits") - 3L), ...)
## S3 method for class 'svystat_rob'
coef(object, ...)
## S3 method for class 'svystat_rob'
SE(object, ...)
## S3 method for class 'svystat_rob'
vcov(object, ...)
## S3 method for class 'svystat_rob'
scale(x, ...)
## S3 method for class 'svystat_rob'
residuals(object, ...)
## S3 method for class 'svystat_rob'
fitted(object, ...)
robweights(object)
## S3 method for class 'svystat_rob'
robweights(object)
## S3 method for class 'svystat_rob'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`object`	object of class `svystat_rob`.
`digits`	`[integer]` minimal number of significant digits.
`...`	additional arguments passed to the method.
`x`	object of class `svystat_rob`.

Details

Package **survey** must be attached to the search path in order to use the functions (see `library` or `require`).

Utility functions:

- `mse` computes the estimated risk (mean square error) in presence of representative outliers; see also `mer`
- `summary` gives a summary of the estimation properties
- `robweights` extracts the robustness weights
- `coef` extracts the estimate of location
- `SE` extracts the (estimated) standard error
- `vcov` extracts the (estimated) covariance matrix
- `residuals` extracts the residuals
- `fitted` extracts the fitted values

See Also

`svymean_dalen`, `svymean_huber`, `svymean_ratio`, `svymean_reg`, `svymean_tukey`, `svymean_trimmed`, `svymean_winsorized`

`svytotal_dalen`, `svytotal_huber`, `svytotal_ratio`, `svytotal_reg`, `svytotal_tukey`, `svytotal_trimmed`, `svytotal_winsorized`

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Estimated one-sided k winsorized population total (i.e., k = 2 observations
# are winsorized at the top of the distribution)
wtot <- svytotal_k_winsorized(~employment, dn, k = 2)

# Show summary statistic of the estimated total
summary(wtot)

# Estimated mean square error (MSE)
mse(wtot)

# Estimate, std. err., variance, and the residuals
coef(wtot)
SE(wtot)
vcov(wtot)
residuals(wtot)

# M-estimate of the total (Huber psi-function; tuning constant k = 3)
mtot <- svytotal_huber(~employment, dn, k = 45)

# Plot of the robustness weights of the M-estimate against its residuals
plot(residuals(mtot), robweights(mtot))
```

`counties` *Data on a Simple Random Sample of 100 Counties in the U.S.*

Description

Data from a simple random sample (without replacement) of 100 of the 3141 counties in the United Stated (U.S. Bureau of the Census, 1994).

Usage

```
data(counties)
```

Format

A `data.frame` with 100 observations on the following variables:

`state` state, `[character]`.

`county` county, `[character]`.

`landarea` land area, 1990 (square miles), `[double]`.

`totpop` population total, 1992, `[double]`.

`unemp` number of unemployed persons, 1991, `[double]`.

`farmpop` farm population, 1990, `[double]`.

`numfarm` number of farms, 1987, `[double]`.

`farmacre` acreage in farms, 1987, `[double]`.

`weights` sampling weight, `[double]`.

`fpc` finite population corretion, `[double]`.

Details

The data (and 10 additional variables) are published in Lohr (1999, Appendix C).

Source

Lohr, S. L. (1999). *Sampling: Design and Analysis*, Pacific Grove (CA): Duxbury Press.

Examples

```
head(counties)

library(survey)
# Survey design for simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~1, fpc = ~fpc, weights = ~weights, data = counties,
                  calibrate.formula = ~1)
    } else {
        # legacy mode
        svydesign(ids = ~1, fpc = ~fpc, weights = ~weights, data = counties)
    }
```

`flour` *Measurement of Copper Content in Wholemeal Flour*

Description

Measurement of copper content in wholemeal flour (measured in parts per million).

Usage

```
data(flour)
```

Format

A `data.frame` with 24 observations (sorted in ascending order) on the following variables:

`copper` copper content `[double]`.

`weight` weight `[double]`.

Details

The data are published in Maronna et al. (2019, p. 2).

Source

Maronna, R. A., Martin, R. D., Yohai, V. J. and Salibián-Barrera, M. (2019). *Robust Statistics: Theory and Methods (with R)*, Hoboken (NJ): John Wiley and Sons, 2nd edition. doi:10.1002/9781119214656

Examples

```
head(flour)
```

`huber2` *Weighted Huber Proposal 2 Estimator*

Description

Weighted Huber Proposal 2 estimator of location and scatter.

Usage

```
huber2(x, w, k = 1.5, na.rm = FALSE, maxit = 50, tol = 1e-04, info = FALSE,
       k_Inf = 1e6, df_cor = TRUE)
```

Arguments

`x`	`[numeric vector]` data.
`w`	`[numeric vector]` weights (same length as `x`).
`k`	`[double]` robustness tuning constant ($0 < k \leq \infty$).
`na.rm`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds (default: `FALSE`).
`maxit`	`[integer]` maximum number of iterations to use (default: `50`).
`tol`	`[double]` numerical tolerance criterion to stop the iterations (default: `1e-04`).

info	[logical] indicating whether additional information should be returned (default: FALSE).
k_Inf	[integer] numerical value that represents Inf (default: 1e+06).
df_cor	[logical] if TRUE, the degrees of freedom of the estimate of scale is adjusted (default: TRUE).

Details

Function huber2 computes the weighted Huber (1964) Proposal 2 estimates of location and scale.

The method is initialized by the weighted median (location) and the weighted interquartile range (scale).

Value

The return value depends on info:

info = FALSE: estimate of mean or total [double]

info = TRUE: a [list] with items:

- characteristic [character],
- estimator [character],
- estimate [double],
- variance (default: NA),
- robust [list],
- residuals [numeric vector],
- model [list],
- design (default: NA),
- [call]

Comparison

The huber2 estimator is initialized by the weighted median and the weighted (scaled) interquartile range. For unweighted data, this estimator *differs* from hubers in **MASS**, which is initialized by mad.

The difference between the estimators is usually negligible (for sufficiently small values of tol). See examples.

References

Huber, P. J. (1964). Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics* **35**, 73–101. doi:10.1214/aoms/1177703732

Examples

```
head(workplace)

# Weighted "Proposal 2" estimator of the mean
huber2(workplace$employment, workplace$weight, k = 8)
```

```
# More information on the estimate, i.e., info = TRUE
m <- huber2(workplace$employment, workplace$weight, k = 8, info = TRUE)

# Estimate of scale
m$scale

# Comparison with MASS::hubers (without weights). We make a copy of MASS::hubers
library(MASS)
hubers_mod <- hubers

# Then we replace mad by the (scaled) IQR as initial scale estimator
body(hubers_mod)[[7]][[3]][[2]] <- substitute(s0 <- IQR(y, type = 2) * 0.7413)

# Define the numerical tolerance
TOLERANCE <- 1e-8

# Comparison
m1 <- huber2(workplace$payroll, rep(1, 142), tol = TOLERANCE)
m2 <- hubers_mod(workplace$payroll, tol = TOLERANCE)$mu
m1 / m2 - 1

# The absolute relative difference is < 4.0-09 (smaller than TOLERANCE)
```

---

losdata *Length-of-Stay (LOS) Hospital Data*

Description

A simple random sample of 70 patients in inpatient hospital treatment.

Usage

```
data(losdata)
```

Format

A `data.frame` with data on the following variables:

`los` length of stay (days) `[integer]`.

`weight` sampling weight `[double]`.

`fpc` finite population correction `[double]`.

Details

The `losdata` are a simple random sample without replacement (SRSWOR) of size $n = 70$ patients from the (fictive) population of $N = 2479$ patients in inpatient hospital treatment. We have constructed the `losdata` as a showcase; though, the LOS measurements are real data that we have taken from the 201 observations in Ruffieux et al. (2000). The original LOS data of Ruffieux et al.

(2000) are available in the R package **robustbase**; see `robustbase::data(los)`. Our `losdata` are a SRSWOR of size $n = 70$ from the 201 original observations.

Ruffieux et al. (2000) and data.frame `los` in the R package **robustbase**.

Source

Ruffieux, C., Paccaud, F. and Marazzi, A. (2000). Comparing rules for truncating hospital length of stay. *Casemix Quarterly* **2**.

Examples

```
head(losdata)

library(survey)
# Survey design for simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~1, fpc = ~fpc, weights = ~weight, data = losdata,
                  calibrate.formula = ~1)
    } else {
        # legacy mode
        svydesign(ids = ~1, fpc = ~fpc, weights = ~weight, data = losdata)
    }
```

`mer` *Minimum Estimated Risk (MER) M-Estimator*

Description

`mer` is an adaptive M-estimator of the weighted mean or total. It is defined as the estimator that minimizes the estimated mean square error, `mse`, of the estimator under consideration.

Usage

```
mer(object, verbose = TRUE, max_k = 10, init = 1, method = "Brent",
    optim_args = list())
```

Arguments

`object`	an object of class `svystat_rob`.
`verbose`	`[logical]` indicating whether additional information is printed to the console (default: `TRUE`).
`init`	`[numeric]` determines the left boundary value of the search interval and the initial value of the search; we must have `init` < `max_k`.
`method`	`[character]` the method of `optim` to be used.
`max_k`	`[numeric vector]` defines the right boundary value of the search interval (default: `max_k = 1000`)
`optim_args`	`[list]`: arguments passed on to `optim`.

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

MER-estimators are available for the methods svymean_huber, svytotal_huber, svymean_tukey and svytotal_tukey.

Value

Object of class svystat_rob

References

Hulliger, B. (1995). Outlier Robust Horvitz-Thompson Estimators. *Survey Methodology* **21**, 79–87.

See Also

Overview (of all implemented functions)

Examples

```
head(losdata)

library(survey)
# Survey design for simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~1, fpc = ~fpc, weights = ~weight, data = losdata,
                  calibrate.formula = ~1)
    } else {
        # legacy mode
        svydesign(ids = ~1, fpc = ~fpc, weights = ~weight, data = losdata)
    }

# M-estimator of the total with tuning constant k = 8
m <- svymean_huber(~los, dn, type = "rhj", k = 8)

# MER estimator
mer(m)
```

MU284pps *PPS Sample From the MU284 Population*

Description

Probability-proportional-to-size sample (PPS) without replacement of municipalities from the MU284 population in Särndal et al. (1992). The sample inclusion probabilities are proportional to the population size in 1975 (variable P75).

Usage

```
data(MU284pps)
```

Format

A `data.frame` with 60 observations on the following variables:

`LABEL` identifier variable, `[integer]`.

`P85` 1985 population size (in thousands), `[double]`.

`P75` 1975 population size (in thousands), `[double]`.

`RMT85` Revenues from the 1985 municipal taxation (in millions of kronor), `[double]`.

`CS82` number of Conservative seats in municipal council, `[double]`.

`SS82` number of Social-Democrat seats in municipal council (1982), `[double]`.

`S82` total number of seats in municipal council (1982), `[double]`.

`ME84` number of municipal employees in 1984, `[double]`.

`REV84` real estate values according to 1984 assessment (in millions of kronor), `[double]`.

`REG` geographic region indicator, `[integer]`.

`CL` cluster indicator (a cluster consists of a set of neighbouring municipalities), `[integer]`.

`weights` sampling weights, `[double]`.

`pi` sample inclusion probability, `[double]`.

Details

The MU284 population of Särndal et al. (1992, Appendix B) is a dataset with observations on the 284 municipalities in Sweden in the late 1970s and early 1980s. The `MU284` *population* data are available in the **sampling** package of Tillé and Matei (2021).

The data frame `MU284pps` is a probability-proportional-to-size sample (PPS) without replacement from the MU284 population. The sample inclusion probabilities are proportional to the population size in 1975 (variable P75). The sample has been selected by Brewer's method; see Tillé (2006, Chap. 7). The sampling weight (inclusion probabilities) are calibrated to the population size and the population total of P75.

Source

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, New York: Springer-Verlag.

Tillé, Y. and Matei, A. (2021). *sampling: Survey Sampling*. R package version 2.9. https://CRAN.R-project.org/package=sampling

Tillé, Y. (2006). *Sampling Algorithms*. New York: Springer-Verlag.

See Also

`MU284strat`

Examples

```
head(MU284pps)

library(survey)
# Survey design with inclusion probabilities proportional to size
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~LABEL, fpc = ~pi, data = MU284pps, pps = "brewer",
                  calibrate.formula = ~1)
    } else {
        # legacy mode
        svydesign(ids = ~LABEL, fpc = ~pi, data = MU284pps, pps = "brewer")
    }
```

`MU284strat` *Stratified Sample from the MU284 Population*

Description

Stratified simple random sample (without replacement) of municipalities from the MU284 population in Särndal et al. (1992). Stratification is by geographic region and a take-all stratum (by 1975 population size), which includes the big cities Stockholm, Göteborg, and Malmö.

Usage

```
data(MU284strat)
```

Format

A `data.frame` with 60 observations on the following variables:

`LABEL` identifier variable, `[integer]`.

`P85` 1985 population size (in thousands), `[double]`.

`P75` 1975 population size (in thousands), `[double]`.

`RMT85` Revenues from the 1985 municipal taxation (in millions of kronor), `[double]`.

`CS82` number of Conservative seats in municipal council, `[double]`.

`SS82` number of Social-Democrat seats in municipal council (1982), `[double]`.

`S82` total number of seats in municipal council (1982), `[double]`.

`ME84` number of municipal employees in 1984, `[double]`.

`REV84` real estate values according to 1984 assessment (in millions of kronor), `[double]`.

`CL` cluster indicator (a cluster consists of a set of neighbouring municipalities), `[integer]`.

`REG` geographic region indicator, `[integer]`.

`Stratum` stratum indicator, `[integer]`.

`weights` sampling weights, `[double]`.

`fpc` finite population correction, `[double]`.

Details

The MU284 population of Särndal et al. (1992, Appendix B) is a dataset with observations on the 284 municipalities in Sweden in the late 1970s and early 1980s. The MU284 *population* data are available in the **sampling** package of Tillé and Matei (2021).

The population is divided into two parts based on 1975 population size (P75):

- the MU281 population, which consists of the 281 smallest municipalities;
- the MU3 population of the three biggest municipalities/ cities in Sweden (Stockholm, Göteborg, and Malmö).

The three biggest cities take exceedingly large values (representative outliers) on almost all of the variables. To account for this, a stratified sample has been drawn from the MU284 population using a take-all stratum. The sample data, MU284strat, (of size $n = 60$) consists of

- a stratified simple random sample (without replacement) from the MU281 population, where stratification is by geographic region (REG) with proportional sample size allocation;
- a take-all stratum that includes the three biggest cities/ municipalities (population M3).

Source

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, New York: Springer-Verlag.

Tillé, Y. and Matei, A. (2021). *sampling: Survey Sampling*. R package version 2.9. https://CRAN.R-project.org/package=sampling

See Also

MU284pps

Examples

```
head(MU284strat)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~LABEL, strata = ~Stratum, fpc = ~fpc,
                  weights = ~weights, data = MU284strat,
                  calibrate.formula = ~-1 + Stratum)
    } else {
        # legacy mode
        svydesign(ids = ~LABEL, strata = ~Stratum, fpc = ~fpc,
                  weights = ~weights, data = MU284strat)
    }
```

pps *Sampling with probability proportional to size (pps without replacement)*

Description

Methods to compute the first-order sample inclusion probabilities (given a measure of size) and sampling mechanisms to draw samples with probabilities proportional to size (pps).

Usage

```
pps_probabilities(size, n)
pps_draw(x, method = "brewer", sort = TRUE)

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

Arguments

size	[numeric vector] measure of size.
n	[integer] sample size.
x	object of class prob_pps.
method	[character] currently only method "brewer" is implemented.
sort	[logical] indicating whether the sampled indices are sorted in ascending order (default: TRUE).
...	additional arguments.

Details

Function pps_probabilities computes the first-order sample inclusion probabilities for a given sample size n; see e.g., Särndal et al., 1992 (p. 90). The probabilities (and additional attributes) are returned as a vector, more precisely as an object of class prob_pps.

For an object of class prob_pps (inclusion probabilities and additional attributes), function pps_draw draws a pps sample without replacement and returns the indexes of the population elements. Only the method of Brewer (1963, 1975) is currently implemented.

Value

Function pps_probabilities returns the probabilities (an object of class (prob_pps).

Function pps_draw returns a pps sample of indexes from the population elements.

References

Brewer, K. W. R. (1963). A Model of Systematic Sampling with Unequal Probabilities. *Australian Journal of Statistics* **5**, 93–105. doi:10.1111/j.1467842X.1963.tb00132.x

Brewer, K. W. R. (1975). A simple procedure for πpswor, *Australian Journal of Statistics* **17**, 166–172. doi:10.1111/j.1467842X.1975.tb00954.x

Särndal, C.-E., Swensson, B., Wretman, J. (1992). *Model Assisted Survey Sampling*, New York: Springer-Verlag.

Examples

```
# We are going to pretend that the workplace sample is our population.
head(workplace)

# The population size is N = 142. We want to draw a pps sample (without
# replacement) of size n = 10, where the variable employment is the measure of
# size. The first-order sample inclusion probabilities are calculated as
# follows

p <- pps_probabilities(workplace$employment, n = 10)

# Now, we draw a pps sample using Brewer's method.
pps_draw(p, method = "brewer")
```

---

robsurvey-deprecated *Deprecated Functions in Package 'robsurvey'*

Description

These functions are provided for compatibility with older versions of the package only, and may be defunct as soon as the next release.

- svyreg_huber
- svyreg_tukey

Use instead:

- svyreg_huberM
- svyreg_tukeyM

See Also

robsurvey-package

robsvyreg *Internal Function for the Regression GM-Estimator*

Description

Internal function to call the robust survey regression *GM*-estimator; this function is **only** intended for internal use. The function does **not** check or validate the arguments. In particular, missing values in the data may make the function crash.

Usage

```
robsvyreg(x, y, w, k, psi, type, xwgt, var = NULL, verbose = TRUE, ...)
svyreg_control(tol = 1e-5, maxit = 100, k_Inf = 1e6, init = NULL,
               mad_center = TRUE, ...)
```

Arguments

x	[numeric matrix] design matrix (NA values not allowed).
y	[numeric vector] dependent variable (NA values not allowed).
w	[numeric vector] weights (no NA's allowed).
k	[double] robustness tuning constant ($0 < k \leq \infty$).
psi	[integer] psi-functions: 0: Huber, 1: asymmetric Huber, and 2: Tukey biweight.
type	[integer] type of estimator; 0: M-estimator; 1: Mallows and 2: Schweppe type GM-estimator.
xwgt	[numeric vector] weights for design space used in GM-estimators (default: NULL, NA values not allowed).
var	[numeric vector] heteroscedastic variance (default: NULL).
verbose	[logical] indicating whether additional information is printed to the console (default: TRUE).
tol	[double] numerical tolerance criterion to stop the iterations (default: 1e-05).
maxit	[integer] maximum number of iterations to use (default: 100).
k_Inf	[integer] numerical value that represents Inf (default: 1e+06).
init	either NULL or [numeric vector], if init = NULL the regression estimator is initialized by weighted least squares; otherwise, init can be specified as the estimate (i.e., *p*-vector) to initialize the iteratively re-weighted least squares method (default: NULL).
mad_center	[logical] if TRUE, the weighted MAD is centered about the (weighted) median, otherwise the weighted MAD is centered about zero (default: TRUE).
...	additional arguments passed to the method (see svyreg_control).

Details

Not documented

Value

[list]

---

svymean-m-estimator *Weighted Huber and Tukey Mean and Total (M-Estimator) – Robust Horvitz-Thompson Estimator*

Description

Weighted Huber and Tukey M-estimator of the population mean and total (robust Horvitz-Thompson estimator)

Usage

```
svymean_huber(x, design, k, type = "rwm", asym = FALSE, na.rm = FALSE,
              verbose = TRUE, ...)
svytotal_huber(x, design, k, type = "rwm", asym = FALSE, na.rm = FALSE,
               verbose = TRUE, ...)
svymean_tukey(x, design, k, type = "rwm", na.rm = FALSE, verbose = TRUE, ...)
svytotal_tukey(x, design, k, type = "rwm", na.rm = FALSE, verbose = TRUE, ...)
```

Arguments

x	a one-sided [formula], e.g., ~myVariable.
design	an object of class survey.design; see svydesign.
k	[double] robustness tuning constant ($0 < k \leq \infty$).
type	[character] type of method: "rwm" or "rht".
asym	[logical] if TRUE, an asymmetric Huber psi-function is used (default: FALSE).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
verbose	[logical] indicating whether additional information is printed to the console (default: TRUE).
...	additional arguments passed to the method (e.g., maxit: maxit number of iterations, etc.; see svyreg_control).

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

Methods/ types type = "rht" or type = "rwm"; see weighted_mean_huber or weighted_mean_tukey for more details.

Variance estimation. Taylor linearization (residual variance estimator).

Utility functions summary, coef, SE, vcov, residuals, fitted, robweights.

Bare-bone functions See weighted_mean_huber weighted_mean_tukey, weighted_total_huber, and weighted_total_tukey.

Value

Object of class svystat_rob

Failure of convergence

By default, the method assumes a maximum number of maxit = 100 iterations and a numerical tolerance criterion to stop the iterations of tol = 1e-05. If the algorithm fails to converge, you may consider changing the default values; see svyreg_control.

References

Hulliger, B. (1995). Outlier Robust Horvitz-Thompson Estimators. *Survey Methodology* **21**, 79–87.

See Also

Overview (of all implemented functions)

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Robust Horvitz-Thompson M-estimator of the population total
svytotal_huber(~employment, dn, k = 9, type = "rht")

# Robust weighted M-estimator of the population mean
m <- svymean_huber(~employment, dn, k = 12, type = "rwm")

# Summary statistic
summary(m)

# Plot of the robustness weights of the M-estimate against its residuals
plot(residuals(m), robweights(m))

# Extract estimate
coef(m)

# Extract estimate of scale
scale(m)

# Extract estimated standard error
```

```
SE(m)
```

---

svymean_dalen *Dalen's Estimators of the Population Mean and Total*

Description

Dalen's estimators Z2 and Z3 of the population mean and total; see weighted_mean_dalen for further details.

Usage

```
svymean_dalen(x, design, censoring, type = "Z2", na.rm = FALSE,
              verbose = TRUE, ...)
svytotal_dalen(x, design, censoring, type = "Z2", na.rm = FALSE,
               verbose = TRUE, ...)
```

Arguments

x	a one-sided [formula], e.g., ~myVariable.
design	an object of class survey.design; see svydesign.
censoring	[double] cutoff threshold above which the observations are censored.
type	[character] type of estimator; either "Z2" or "Z3" (default: "Z2").
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
verbose	[logical] indicating whether additional information is printed to the console (default: TRUE).
...	additional arguments (currently not used).

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

Methods/ types type = "Z2" or type = "Z3"; see weighted_mean_dalen for more details.

Utility functions summary, coef, SE, vcov, residuals, fitted, robweights.

Bare-bone functions See weighted_mean_dalen and weighted_total_dalen.

Value

Object of class svystat_rob

References

Dalén, J. (1987). Practical Estimators of a Population Total Which Reduce the Impact of Large Observations. R & D Report U/STM 1987:32, Statistics Sweden, Stockholm.

See Also

Overview (of all implemented functions)

svymean_trimmed, svytotal_trimmed, svymean_winsorized, svytotal_winsorized, svymean_huber and svytotal_huber

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Dalen's estimator Z3 of the population total
svytotal_dalen(~employment, dn, censoring = 20000, type = "Z3")

# Dalen's estimator Z3 of the population mean
m <- svymean_dalen(~employment, dn, censoring = 20000, type = "Z3")

# Summarize
summary(m)

# Extract estimate
coef(m)

# Extract estimated standard error
SE(m)
```

---

svymean_ratio *Robust Ratio Predictor of the Mean and Total*

Description

Robust ratio predictor (M-estimator) of the population mean and total with Huber and Tukey biweight (bisquare) psi-function.

Usage

```
svytotal_ratio(object, total, variance = "wu", keep_object = TRUE, ...)
svymean_ratio(object, total, N = NULL, variance = "wu",
              keep_object = TRUE, N_unknown = FALSE, ...)
```

Arguments

object	an object of class [ratio], e.g., result of the Huber ratio M-estimator svyratio_huber.
total	[numeric] vector of population totals of the auxiliary variables.
N	[numeric] population size (see also N_unknown.
variance	[character] type of variance estimator (default: "wu"); see Details Section.
keep_object	[logical] if TRUE, object is returned as an additional slot of the return value (default: TRUE).
N_unknown	[logical] if TRUE, it is assumed that the population size is unknown; thus, it is estimated (default: FALSE).
...	additional arguments (currently not used).

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

The (robust) ratio predictor of the population total or mean is computed in two steps.

- Step 1: Fit the ratio model associated with the predictor by one of the functions svyratio_huber or svyratio_tukey. The fitted model is called object.
- Step 2: Based on the fitted model obtained in the first step, we predict the population total and mean, respectively, by the predictors svytotal_ratio and svymean_ratio, where object is the fitted ratio model.

Auxiliary data Two types of auxiliary variables are distinguished: (1) population size N and (2) the population total of the auxiliary variable (denominator) used in the ratio model.

The option N_unknown = TRUE can be used in the predictor of the population mean if N is unknown.

Variance estimation Three variance estimators are implemented (argument variance): "base", "wu", and "hajek". These estimators correspond to the estimators v0, v1, and v2 in Wu (1982).

Utility functions The return value is an object of class svystat_rob. Thus, the utility functions summary, coef, SE, vcov, residuals, fitted, and robweights are available.

Value

Object of class svystat_rob

References

Wu, C.-F. (1982). Estimation of Variance of the Ratio Estimator. *Biometrika* **69**, 183–189.

See Also

Overview (of all implemented functions)

svymean_reg and svytotal_reg for (robust) GREG regression predictors

svyreg_huberM, svyreg_huberGM, svyreg_tukeyM and svyreg_tukeyGM for robust regression M- and GM-estimators

svymean_huber, svytotal_huber, svymean_tukey and svytotal_tukey for M-estimators

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Robust ratio M-estimator with Huber psi-function
rat <- svyratio_huber(~payroll, ~ employment, dn, k = 5)

# Summary of the ratio estimate
summary(rat)

# Diagnostic plots of the ration/regression M-estimate (e.g.,
# standardized residuals against fitted values)
plot(rat, which = 1L)

# Plot of the robustness weights of the ratio/regression M-estimate
# against its residuals
plot(residuals(rat), robweights(rat))

# Robust ratio predictor of the population mean
m <- svymean_ratio(rat, total = 1001233, N = 90840)
m

# Summary of the ratio estimate of the population mean
summary(m)

# Extract estimate
coef(m)

# Extract estimate of scale
scale(m)

# Extract estimated standard error
```

```
SE(m)
```

---

svymean_reg — *Robust Generalized Regression Predictor (GREG) of the Mean and Total*

Description

Generalized regression estimator (GREG) predictor of the mean and total, and robust GREG M-estimator predictor

Usage

```
svytotal_reg(object, totals, N = NULL, type, k = NULL, check.names = TRUE,
             keep_object = TRUE, ...)
svymean_reg(object, totals, N = NULL, type, k = NULL, check.names = TRUE,
            keep_object = TRUE, N_unknown = FALSE, ...)
```

Arguments

`object`	an object of class `[svyreg_rob]`, e.g., result of the Huber regression M-estimator `svyreg_huberM`.
`totals`	`[numeric]` vector of population totals of the auxiliary variables.
`N`	`[numeric]` population size (see also `N_unknown`.
`type`	`[character]` type of predictor; see Details Section.
`k`	`[numeric]` robustness tuning constant of the `psi`-function used in the bias-correction term of the GREG. The definition of `k` depends on the `type` of predictor and is discussed in the Details Section.
`check.names`	`[logical]` if `TRUE`, the names of `auxiliary` are checked against the names of the independent variables of the fitted model `object` (default: `TRUE`).
`keep_object`	`[logical]` if `TRUE`, `object` is returned as an additional slot of the return value (default: `TRUE`).
`N_unknown`	`[logical]` if `TRUE`, it is assumed that the population size is unknown; thus, it is estimated (default: `FALSE`).
`...`	additional arguments (currently not used).

Details

Package **survey** must be attached to the search path in order to use the functions (see `library` or `require`).

The (robust) GREG predictor of the population total or mean is computed in two steps.

- Step 1: Fit the regression model associated with the GREG predictor by one of the functions `svyreg`, `svyreg_huberM`, `svyreg_huberGM`, `svyreg_tukeyM` or `svyreg_tukeyGM`. The fitted model is called `object`.

- Step 2: Based on the fitted model obtained in the first step, we predict the population total and mean, respectively, by the predictors `svytotal_reg` and `svymean_reg`, where `object` is the fitted regression model.

The following GREG predictors are available:

GREG (not robust, `k = NULL`**)** The following *non-robust* GREG predictors are available:

- `type = "projective"` ignores the bias correction term of the GREG predictor; see Särndal and Wright (1984).
- `type = "ADU"` is the "standard" GREG, which is an asymptotically design unbiased (ADU) predictor; see Särndal et al.(1992, Chapter 6).

If the fitted regression model (`object`) does include a regression intercept, the predictor types `"projective"` and `"ADU"` are identical because the bias correction of the GREG is zero by design.

Robust GREG The following *robust* GREG predictors are available:

- `type = "huber"` and `type = "tukey"` are, respectively, the robust GREG predictors with Huber and Tukey bisquare (biweight) psi-function. The tuning constant must satisfy `0 < k <= Inf`. We can use the Huber-type GREG predictor although the model has been fitted by the regression estimator with Tukey psi-function (and vice versa).
- `type = "BR"` is the bias-corrected robust GREG predictor of Beaumont and Rivest (2009), which is inspired by the bias-corrected robust predictor of Chambers (1986). The tuning constant must satisfy `0 < k <= Inf`.
- `type = "lee"` is the bias-corrected predictor of Lee (1991; 1992). Tthe tuning constant `k` must satisfy `0 <= k <= 1`.
- `type = "duchesne"` is the bias-corrected, calibration-type estimator/ predictor of Duchesne (1999). The tuning constant `k` must be specified as a vector `k = c(a, b)`, where `a` and `b` are the tuning constants of Duchesne's modified Huber psi-function (default values: `a = 9` and `b = 0.25`).

Auxiliary data Two types of auxiliary variables are distinguished: (1) population size N and (2) population totals of the auxiliary variables used in the regression model (i.e., non-constant explanatory variables).

The option `N_unknown = TRUE` can be used in the predictor of the population mean if N is unknown.

The names of the entries of `totals` are checked against the names of the regression fit (`object`), unless we specify `check.names = FALSE`.

Utility functions The return value is an object of class `svystat_rob`. Thus, the utility functions `summary`, `coef`, `SE`, `vcov`, `residuals`, `fitted`, and `robweights` are available.

Value

Object of class `svystat_rob`

References

Beaumont, J.-F. and Rivest, L.-P. (2009). Dealing with outliers in survey data. In: *Sample Surveys: Theory, Methods and Inference* ed. by Pfeffermann, D. and Rao, C. R. Volume 29A of *Handbook of Statistics*, Amsterdam: Elsevier, Chap. 11, 247–280. doi:10.1016/S01697161(08)000114

Chambers, R. (1986). Outlier Robust Finite Population Estimation. *Journal of the American Statistical Association* **81**, 1063–1069. doi:10.1080/01621459.1986.10478374

Duchesne, P. (1999). Robust calibration estimators, *Survey Methodology* **25**, 43–56.

Gwet, J.-P. and Rivest, L.-P. (1992). Outlier Resistant Alternatives to the Ratio Estimator. *Journal of the American Statistical Association* **87**, 1174–1182. doi:10.1080/01621459.1992.10476275

Lee, H. (1991). Model-Based Estimators That Are Robust to Outliers, in *Proceedings of the 1991 Annual Research Conference*, Bureau of the Census, 178–202. Washington, DC, Department of Commerce.

Lee, H. (1995). Outliers in business surveys. In: *Business survey methods* ed. by Cox, B. G., Binder, D. A., Chinnappa, B. N., Christianson, A., Colledge, M. J. and Kott, P. S. New York: John Wiley and Sons, Chap. 26, 503–526. doi:10.1002/9781118150504.ch26

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, New York: Springer.

Särndal, C.-E. and Wright, R. L. (1984). Cosmetic Form of Estimators in Survey Sampling. *Scandinavian Journal of Statistics* **11**, 146–156.

See Also

Overview (of all implemented functions)

`svymean_ratio` and `svytotal_ratio` for (robust) ratio predictors

`svymean_huber`, `svytotal_huber`, `svymean_tukey` and `svytotal_tukey` for M-estimators

`svyreg`, `svyreg_huberM`, `svyreg_huberGM`, `svyreg_tukeyM` and `svyreg_tukeyGM` for robust regression M- and GM-estimators

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Robust regression M-estimator with Huber psi-function
reg <- svyreg_huberM(payroll ~ employment, dn, k = 3)

# Summary of the regression M-estimate
summary(reg)

# Diagnostic plots of the regression M-estimate (e.g., standardized
# residuals against fitted values)
```

```
plot(reg, which = 1L)

# Plot of the robustness weights of the regression M-estimate against
# its residuals
plot(residuals(reg), robweights(reg))

# ADU (asymptotically design unbiased) estimator
m <- svytotal_reg(reg, totals = 1001233, 90840, type = "ADU")
m

# Robust GREG estimator of the mean; the population means of the auxiliary
# variables are from a register
m <- svymean_reg(reg, totals = 1001233, 90840, type = "huber", k = 20)
m

# Summary of the robust GREG estimate
summary(m)

# Extract estimate
coef(m)

# Extract estimated standard error
SE(m)

# Approximation of the estimated mean square error
mse(m)
```

---

svymean_trimmed *Weighted Trimmed Mean and Total*

Description

Weighted trimmed population mean and total.

Usage

```
svymean_trimmed(x, design, LB = 0.05, UB = 1 - LB, na.rm = FALSE, ...)
svytotal_trimmed(x, design, LB = 0.05, UB = 1 - LB, na.rm = FALSE, ...)
```

Arguments

x	a one-sided [formula], e.g., ~myVariable.
design	an object of class survey.design; see svydesign.
LB	[double] lower bound of trimming such that $0 \leq \text{LB} < \text{UB} \leq 1$.
UB	[double] upper bound of trimming such that $0 \leq \text{LB} < \text{UB} \leq 1$.
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
...	additional arguments (currently not used).

Details

Package **survey** must be attached to the search path in order to use the functions (see `library` or `require`).

Characteristic. Population mean or total. Let μ denote the estimated trimmed population mean; then, the estimated trimmed total is given by $\hat{N}\mu$ with $\hat{N} = \sum w_i$, where summation is over all observations in the sample.

Trimming. The methods trims the `LB` $\cdot 100\%$ of the smallest observations and the (1 - `UB`) $\cdot 100\%$ of the largest observations from the data.

Variance estimation. Large-sample approximation based on the influence function; see Huber and Ronchetti (2009, Chap. 3.3) and Shao (1994).

Utility functions. `summary`, `coef`, `SE`, `vcov`, `residuals`, `fitted`, `robweights`.

Bare-bone functions. See `weighted_mean_trimmed` and `weighted_total_trimmed`.

Value

Object of class `svystat_rob`

References

Huber, P. J. and Ronchetti, E. (2009). *Robust Statistics*, New York: John Wiley and Sons, 2nd edition. doi:10.1002/9780470434697

Shao, J. (1994). L-Statistics in Complex Survey Problems. *The Annals of Statistics* **22**, 976–967. doi:10.1214/aos/1176325505

See Also

Overview (of all implemented functions)

`weighted_mean_trimmed` and `weighted_total_trimmed`

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Estimated trimmed population total (5% symmetric trimming)
svytotal_trimmed(~employment, dn, LB = 0.05, UB = 0.95)
```

```
# Estimated trimmed population mean (5% trimming at the top of the distr.)
svymean_trimmed(~employment, dn, UB = 0.95)
```

---

svymean_winsorized *Weighted Winsorized Mean and Total*

Description

Weighted winsorized mean and total

Usage

```
svymean_winsorized(x, design, LB = 0.05, UB = 1 - LB, na.rm = FALSE,
                   trim_var = FALSE, ...)
svymean_k_winsorized(x, design, k, na.rm = FALSE, trim_var = FALSE, ...)
svytotal_winsorized(x, design, LB = 0.05, UB = 1 - LB, na.rm = FALSE,
                    trim_var = FALSE, ...)
svytotal_k_winsorized(x, design, k, na.rm = FALSE, trim_var = FALSE, ...)
```

Arguments

x	a one-sided [formula], e.g., ~myVariable.
design	an object of class survey.design; see svydesign.
LB	[double] lower bound of winsorization such that $0 \leq$ LB $<$ UB ≤ 1.
UB	[double] upper bound of winsorization such that $0 \leq$ LB $<$ UB ≤ 1.
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
trim_var	[logical] indicating whether the variance should be approximated by the variance estimator of the trimmed mean/ total (default: FALSE).
k	[integer] number of observations to be winsorized at the top of the distribution.
...	additional arguments (currently not used).

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

Characteristic. Population mean or total. Let μ denote the estimated winsorized population mean; then, the estimated winsorized total is given by $\hat{N}\mu$ with $\hat{N} = \sum w_i$, where summation is over all observations in the sample.

Modes of winsorization. The amount of winsorization can be specified in relative or absolute terms:

- *Relative:* By specifying LB and UB, the method winsorizes the LB $\cdot 100\%$ of the smallest observations and the (1 - UB) $\cdot 100\%$ of the largest observations from the data.
- *Absolute:* By specifying argument k in the functions with the "infix" _k_ in their name (e.g., svymean_k_winsorized), the largest k observations are winsorized, $0 < k < n$, where n denotes the sample size. E.g., k = 2 implies that the largest and the second largest observation are winsorized.

Variance estimation. Large-sample approximation based on the influence function; see Huber and Ronchetti (2009, Chap. 3.3) and Shao (1994). Two estimators are available:

simple_var = FALSE Variance estimator of the winsorized mean/ total. The estimator depends on the estimated probability density function evaluated at the winsorization thresholds, which can be – depending on the context – numerically unstable. As a remedy, a simplified variance estimator is available by setting simple_var = TRUE.

simple_var = TRUE Variance is approximated using the variance estimator of the trimmed mean/ total.

Utility functions. summary, coef, SE, vcov, residuals, fitted and robweights.

Bare-bone functions. See:

- weighted_mean_winsorized,
- weighted_mean_k_winsorized,
- weighted_total_winsorized,
- weighted_total_k_winsorized.

Value

Object of class svystat_rob

References

Huber, P. J. and Ronchetti, E. (2009). *Robust Statistics*, New York: John Wiley and Sons, 2nd edition. doi:10.1002/9780470434697

Shao, J. (1994). L-Statistics in Complex Survey Problems. *The Annals of Statistics* **22**, 976–967. doi:10.1214/aos/1176325505

See Also

Overview (of all implemented functions)

weighted_mean_winsorized, weighted_mean_k_winsorized, weighted_total_winsorized and weighted_total_k_winsorized

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
```

```
                 data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Estimated winsorized population mean (5% symmetric winsorization)
svymean_winsorized(~employment, dn, LB = 0.05)

# Estimated one-sided k winsorized population total (2 observations are
# winsorized at the top of the distribution)
svytotal_k_winsorized(~employment, dn, k = 2)
```

`svyratio_huber` *Robust Survey Ratio M-Estimator*

Description

svyratio_huber and svyratio_tukey compute the robust M-estimator of the ratio of two variables with, respectively, Huber and Tukey biweight (bisquare) psi-function.

Usage

```
svyratio_huber(numerator, denominator, design, k, var = denominator,
               na.rm = FALSE, asym = FALSE, verbose = TRUE, ...)
svyratio_tukey(numerator, denominator, design, k, var = denominator,
               na.rm = FALSE, verbose = TRUE, ...)
```

Arguments

numerator	a one-sided [formula] object (i.e., symbolic description, e.g., ~payroll).
denominator	a one-sided [formula] object (i.e., symbolic description, e.g., ~employment).
design	an object of class survey.design; see svydesign.
k	[double] robustness tuning constant ($0 < k \leq \infty$).
var	a [formula] object that defines the heteroscedastic variance (default: numerator).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
asym	[logical] toggle for asymmetric Huber psi-function (default: FALSE).
verbose	[logical] indicating whether additional information is printed to the console (default: TRUE).
...	additional arguments passed to the method (e.g., maxit: maxit number of iterations, etc.).

Details

Package **survey** must be attached to the search path in order to use the functions (see `library` or `require`).

The functions `svyratio_huber` and `svyratio_tukey` are implemented as wrapper functions of the regression estimators `svyreg_huberM` and `svyreg_tukeyM`. See the help files of these functions (e.g., on how additional parameters can be passed via `...` or on the usage of the `var` argument).

Value

Object of class `svyreg.rob` and `ratio`

See Also

Overview (of all implemented functions)

`summary`, `coef`, `residuals`, `fitted`, `SE` and `vcov`

`plot` for regression diagnostic plot methods

`svyreg_huberM`, `svyreg_huberGM`, `svyreg_tukeyM` and `svyreg_tukeyGM` for robust regression estimators

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Compute regression M-estimate with Huber psi-function
m <- svyratio_huber(~payroll, ~employment, dn, k = 8)

# Regression inference
summary(m)

# Extract the coefficients
coef(m)

# Extract estimated standard error
SE(m)

# Extract variance/ covariance matrix
vcov(m)
```

```
# Diagnostic plots (e.g., standardized residuals against fitted values)
plot(m, which = 1L)

# Plot of the robustness weights of the M-estimate against its residuals
plot(residuals(m), robweights(m))
```

svyreg *Survey Regression Estimator – Weighted Least Squares*

Description

Weighted least squares estimator of regression

Usage

```
svyreg(formula, design, var = NULL, na.rm = FALSE, ...)
```

Arguments

formula	a [formula] object (i.e., symbolic description of the model)
design	an object of class survey.design; see svydesign.
var	a one-sided [formula] object or variable name ([character]) that defines the heteroscedastic variance or [NULL] indicating homoscedastic variance (default: NULL).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
...	additional arguments (currently not used).

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

svyreg computes the regression coefficients by weighted least squares.

Models for svyreg_rob are specified symbolically. 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; see formula and lm.

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.

Value

Object of class svyreg_rob.

See Also

Overview (of all implemented functions)

summary, coef, residuals, fitted, SE and vcov

plot for regression diagnostic plot methods

Robust estimating methods svyreg_huberM, svyreg_huberGM, svyreg_tukeyM and svyreg_tukeyGM.

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Compute the regression estimate (weighted least squares)
m <- svyreg(payroll ~ employment, dn)

# Regression inference
summary(m)

# Extract the coefficients
coef(m)

# Extract variance/ covariance matrix
vcov(m)

# Diagnostic plots (e.g., Normal Q-Q-plot)
plot(m, which = 2L)
```

---

svyreg_huber-deprecated   *Deprecated Huber Robust Survey Regression M-Estimator*

Description

The function svyreg_huber is **deprecated**; use instead svyreg_huberM.

Usage

```
svyreg_huber(formula, design, k, var = NULL, na.rm = FALSE, asym = FALSE,
             verbose = TRUE, ...)
```

Arguments

formula	a [formula] object (i.e., symbolic description of the model)
design	an object of class survey.design; see svydesign.
k	[double] robustness tuning constant ($0 < k \leq \infty$).
var	a one-sided [formula] object or variable name ([character]) that defines the heteroscedastic variance or [NULL] indicating homoscedastic variance (default: NULL).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
asym	[logical] toggle for asymmetric Huber psi-function (default: FALSE).
verbose	[logical] indicating whether additional information is printed to the console (default: TRUE).
...	additional arguments passed to the method (e.g., maxit: maxit number of iterations, etc.).

Details

See svyreg_huberM.

Value

Object of class svyreg.rob

See Also

robsurvey-deprecated

---

svyreg_huberM *Huber Robust Survey Regression M- and GM-Estimator*

Description

svyreg_huberM and svyreg_huberGM compute, respectively, a survey weighted *M*- and *GM*-estimator of regression using the Huber psi-function.

Usage

```
svyreg_huberM(formula, design, k, var = NULL, na.rm = FALSE, asym = FALSE,
              verbose = TRUE, ...)
svyreg_huberGM(formula, design, k, type = c("Mallows", "Schweppe"),
               xwgt, var = NULL, na.rm = FALSE, asym = FALSE, verbose = TRUE,
               ...)
```

Arguments

- formula — a [formula] object (i.e., symbolic description of the model)
- design — an object of class survey.design; see svydesign.
- k — [double] robustness tuning constant ($0 < k \leq \infty$).
- var — a one-sided [formula] object or variable name ([character]) that defines the heteroscedastic variance or [NULL] indicating homoscedastic variance (default: NULL).
- na.rm — [logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
- asym — [logical] toggle for asymmetric Huber psi-function (default: FALSE).
- verbose — [logical] indicating whether additional information is printed to the console (default: TRUE).
- type — [character] "Mallows" or "Schweppe".
- xwgt — [numerical vector] or [NULL] of weights in the design space (default: NULL); xwgt is only relevant for type = "Mallows" or type = "Schweppe".
- ... — additional arguments passed to the method (e.g., maxit: maxit number of iterations, etc.).

Details

Package **survey** must be attached to the search path in order to use the functions (see library or require).

svyreg_huberM and svyreg_huberGM compute, respectively, *M*- and *GM*-estimates of regression by iteratively re-weighted least squares (IRWLS). The estimate of regression scale is (by default) computed as the (normalized) weighted median of absolute deviations from the weighted median (MAD; see weighted_mad) for each IRWLS iteration. If the weighted MAD is zero (or nearly so), the scale is computed as the (normalized) weighted interquartile range (IQR).

M-estimator The regression M-estimator svyreg_huberM is robust against residual outliers (granted that the tuning constant k is chosen appropriately).

GM-estimator Function svyreg_huberGM implements the Mallows and Schweppe regression GM-estimator (see argument type). The regression GM-estimators are robust against residual outliers *and* outliers in the model's design space (leverage observations; see argument xwgt).

Numerical optimization See svyreg_control.

Models Models for svyreg_rob are specified symbolically. 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; see formula and lm.

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.

Value

Object of class svyreg.rob

Failure of convergence

By default, the method assumes a maximum number of `maxit = 100` iterations and a numerical tolerance criterion to stop the iterations of `tol = 1e-05`. If the algorithm fails to converge, you may consider changing the default values; see `svyreg_control`.

See Also

Overview (of all implemented functions)

Overview (of all implemented functions)

`summary`, `coef`, `residuals`, `fitted`, `SE` and `vcov`

`plot` for regression diagnostic plot methods

Other robust estimating methods `svyreg_tukeyM` and `svyreg_tukeyGM`

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Compute regression M-estimate with Huber psi-function
m <- svyreg_huberM(payroll ~ employment, dn, k = 8)

# Regression inference
summary(m)

# Extract the coefficients
coef(m)

# Extract variance/ covariance matrix
vcov(m)

# Diagnostic plots (e.g., standardized residuals against fitted values)
plot(m, which = 1L)

# Plot of the robustness weights of the M-estimate against its residuals
plot(residuals(m), robweights(m))
```

svyreg_tukey-deprecated

Deprecated Tukey Biweight Robust Survey Regression M-Estimator

Description

The function svyreg_tukey is **deprecated**; use instead svyreg_tukeyM.

Usage

```
svyreg_tukey(formula, design, k, var = NULL, na.rm = FALSE, verbose = TRUE,
             ...)
```

Arguments

formula	a [formula] object (i.e., symbolic description of the model)
design	an object of class survey.design; see svydesign.
k	[double] robustness tuning constant ($0 < k \leq \infty$).
var	a one-sided [formula] object or variable name ([character]) that defines the heteroscedastic variance or [NULL] indicating homoscedastic variance (default: NULL).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
verbose	[logical] indicating whether additional information is printed to the console (default: TRUE).
...	additional arguments passed to the method (e.g., maxit: maxit number of iterations, etc.).

Details

See svyreg_tukeyM.

Value

Object of class svyreg.rob

See Also

robsurvey-deprecated

`svyreg_tukeyM` *Tukey Biweight Robust Survey Regression M- and GM-Estimator*

Description

`svyreg_tukeyM` and `svyreg_tukeyGM` compute, respectively, a survey weighted *M*- and *GM*-estimator of regression using the biweight Tukey psi-function.

Usage

```
svyreg_tukeyM(formula, design, k, var = NULL, na.rm = FALSE, verbose = TRUE,
              ...)
svyreg_tukeyGM(formula, design, k, type = c("Mallows", "Schweppe"),
               xwgt, var = NULL, na.rm = FALSE, verbose = TRUE, ...)
```

Arguments

`formula`	a `[formula]` object (i.e., symbolic description of the model)
`design`	an object of class `survey.design`; see `svydesign`.
`k`	`[double]` robustness tuning constant ($0 < k \leq \infty$).
`var`	a one-sided `[formula]` object or variable name (`[character]`) that defines the heteroscedastic variance or `[NULL]` indicating homoscedastic variance (default: `NULL`).
`na.rm`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds (default: `FALSE`).
`verbose`	`[logical]` indicating whether additional information is printed to the console (default: `TRUE`).
`type`	`[character]` `"Mallows"` or `"Schweppe"`.
`xwgt`	`[numerical vector]` or `[NULL]` of weights in the design space (default: `NULL`); `xwgt` is only relevant for `type = "Mallows"` or `type = "Schweppe"`.
`...`	additional arguments passed to the method (e.g., `maxit`: maxit number of iterations, etc.).

Details

Package **survey** must be attached to the search path in order to use the functions (see `library` or `require`).

`svyreg_tukeyM` and `svyreg_tukeyGM` compute, respectively, M- and GM-estimates of regression by iteratively re-weighted least squares (IRWLS). The estimate of regression scale is (by default) computed as the (normalized) weighted median of absolute deviations from the weighted median (MAD; see `weighted_mad`) for each IRWLS iteration. If the weighted MAD is zero (or nearly so), the scale is computed as the (normalized) weighted interquartile range (IQR).

M-estimator The regression M-estimator `svyreg_tukeyM` is robust against residual outliers (granted that the tuning constant `k` is chosen appropriately).

GM-estimator Function svyreg_huberGM implements the Mallows and Schweppe regression GM-estimator (see argument type). The regression GM-estimators are robust against residual outliers *and* outliers in the model's design space (leverage observations; see argument xwgt).

Numerical optimization See svyreg_control.

Models Models for svyreg_rob are specified symbolically. 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; see formula and lm.

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.

Value

Object of class svyreg.rob

Failure of convergence

By default, the method assumes a maximum number of maxit = 100 iterations and a numerical tolerance criterion to stop the iterations of tol = 1e-05. If the algorithm fails to converge, you may consider changing the default values; see svyreg_control.

See Also

Overview (of all implemented functions)

summary, coef, residuals, fitted, SE and vcov

plot for regression diagnostic plot methods.

Other robust estimating methods svyreg_huberM and svyreg_huberGM

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

# Compute regression M-estimate with Tukey bisquare psi-function
m <- svyreg_tukeyM(payroll ~ employment, dn, k = 8)

# Regression inference
summary(m)
```

```
# Extract the coefficients
coef(m)

# Extract variance/ covariance matrix
vcov(m)

# Diagnostic plots (e.g., standardized residuals against fitted values)
plot(m, which = 1L)

# Plot of the robustness weights of the M-estimate against its residuals
plot(residuals(m), robweights(m))
```

---

svysummary *Weighted Five-Number Summary of a Variable*

Description

Weighted five-number summary used for `survey.design` and `survey.design2` objects (similar to `base::summary` for `[numeric vectors]`).

Usage

```
svysummary(object, design, na.rm = FALSE, ...)
```

Arguments

`object`	one-sided `[formula]` for which a summary is desired, e.g., `~payroll`.
`design`	an object of class `survey.design`; see `svydesign`.
`na.rm`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds (default: `FALSE`).
`...`	additional arguments.

Value

A weighted five-number summary (numeric variable) or a frequency table (factor variable).

Examples

```
head(workplace)

library(survey)
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
```

```
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }

svysummary(~payroll, dn)
```

---

weighted-m-estimator *Weighted Huber and Tukey Mean and Total (bare-bone functions)*

Description

Weighted Huber and Tukey *M*-estimator of the mean and total (bare-bone function with limited functionality; see svymean_huber, svymean_tukey, svytotal_huber, and svytotal_tukey for more capable methods)

Usage

```
weighted_mean_huber(x, w, k, type = "rwm", asym = FALSE, info = FALSE,
                    na.rm = FALSE, verbose = TRUE, ...)
weighted_total_huber(x, w, k, type = "rwm", asym = FALSE, info = FALSE,
                     na.rm = FALSE, verbose = TRUE, ...)
weighted_mean_tukey(x, w, k, type = "rwm", info = FALSE, na.rm = FALSE,
                    verbose = TRUE, ...)
weighted_total_tukey(x, w, k, type = "rwm", info = FALSE, na.rm = FALSE,
                     verbose = TRUE, ...)
```

Arguments

x	[numeric vector] data.
w	[numeric vector] weights (same length as x).
k	[double] robustness tuning constant ($0 < k \leq \infty$).
type	[character] type of method: "rwm" or "rht"; see below (default: "rwm").
asym	[logical] toggle for asymmetric Huber psi-function (default: FALSE).
info	[logical] indicating whether additional information should be returned (default: FALSE).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
verbose	[logical] indicating whether additional information is printed to the console (default: TRUE).
...	additional arguments passed to the method (e.g., maxit: maxit number of iterations, etc.).

Details

Characteristic. Population mean or total. Let μ denote the estimated population mean; then, the estimated total is given by $\hat{N}\mu$ with $\hat{N} = \sum w_i$, where summation is over all observations in the sample.

Type. Two methods/types are available for estimating the location μ:

type = "rwm" (default): robust weighted *M*-estimator of the population mean and total, respectively. This estimator is recommended for sampling designs whose inclusion probabilities are *not* proportional to some measure of size. [Legacy note: In an earlier version, the method type = "rwm" was called "rhj"; the type "rhj" is now silently converted to "rwm"]

type = "rht": robust Horvitz-Thompson *M*-estimator of the population mean and total, respectively. This estimator is recommended for proportional-to-size sampling designs.

Variance estimation. See the related but more capable functions:

- svymean_huber and svymean_tukey,
- svytotal_huber and svytotal_tukey.

Psi-function. By default, the Huber or Tukey psi-function are used in the specification of the M-estimators. For the Huber estimator, an asymmetric version of the Huber psi-function can be used by setting the argument asym = TRUE in the function call.

Value

The return value depends on info:

info = FALSE: estimate of mean or total [double]

info = TRUE: a [list] with items:

- characteristic [character],
- estimator [character],
- estimate [double],
- variance (default: NA),
- robust [list],
- residuals [numeric vector],
- model [list],
- design (default: NA),
- [call]

Failure of convergence

By default, the method assumes a maximum number of maxit = 100 iterations and a numerical tolerance criterion to stop the iterations of tol = 1e-05. If the algorithm fails to converge, you may consider changing the default values; see svyreg_control.

References

Hulliger, B. (1995). Outlier Robust Horvitz-Thompson Estimators. *Survey Methodology* **21**, 79–87.

See Also

Overview (of all implemented functions)

Examples

```
head(workplace)

# Robust Horvitz-Thompson M-estimator of the population total
weighted_total_huber(workplace$employment, workplace$weight, k = 9,
    type = "rht")

# Robust weighted M-estimator of the population mean
weighted_mean_huber(workplace$employment, workplace$weight, k = 12,
    type = "rwm")
```

---

weighted_IQR *Weighted Interquartile Range (IQR)*

Description

Weighted (normalized) interquartile range

Usage

```
weighted_IQR(x, w, na.rm = FALSE, constant = 0.7413)
```

Arguments

x	[numeric vector] data.
w	[numeric vector] weights (same length as x).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
constant	[double] constant scaling factor to make the weighted IQR a consistent estimator of the scale (default: 0.7413).

Details

By default, the weighted IQR is normalized to be an unbiased estimate of scale at the Gaussian core model. If normalization is not wanted, put constant = 1.

Value

Weighted IQR

See Also

Overview (of all implemented functions)

Examples

```
head(workplace)

# normalized weighted IQR (default)
weighted_IQR(workplace$employment, workplace$weight)

# weighted IQR (without normalization)
weighted_IQR(workplace$employment, workplace$weight, constant = 1)
```

---

weighted_line *Weighted Robust Line Fitting*

Description

`weighted_line` fits a robust line and allows weights.

Usage

```
weighted_line(x, y = NULL, w, na.rm = FALSE, iter = 1)

## S3 method for class 'medline'
print(x, ...)
## S3 method for class 'medline'
coef(object, ...)
## S3 method for class 'medline'
residuals(object, ...)
## S3 method for class 'medline'
fitted(object, ...)
```

Arguments

`x`	`[numeric vector]` explanatory variable.
`y`	`[numeric vector]` response variable (default: `NULL`).
`w`	`[numeric vector]` weights (same length as `x`).
`na.rm`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds (default: `FALSE`).
`iter`	`[integer]` number of iterations to use (default: `1`).
`object`	object of class `medline`.
`...`	additional arguments passed to the method.

Details

`weighted_line` uses different quantiles for splitting the sample than `stats::line()`.

Value

intercept and slope of the fitted line

See Also

Overview (of all implemented functions)

line

Examples

```
head(cars)

# compute weighted line
weighted_line(cars$speed, cars$dist, w = rep(1, length(cars$speed)))
m <- weighted_line(cars$speed, cars$dist, w = rep(1:10, each = 5))
m
coef(m)
residuals(m)
fitted(m)
```

---

weighted_mad *Weighted Median Absolute Deviation from the Median (MAD)*

Description

Weighted median of the absolute deviations from the weighted median

Usage

```
weighted_mad(x, w, na.rm = FALSE, constant = 1.482602)
```

Arguments

x	[numeric vector] data.
w	[numeric vector] weights (same length as x).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
constant	[double] constant scaling factor to make the MAD a consistent estimator of the scale (default: 1.4826).

Details

The weighted MAD is computed as the (normalized) weighted median of the absolute deviation from the weighted median; see weighted_median. The weighted MAD is normalized to be an unbiased estimate of scale at the Gaussian core model. If normalization is not wanted, put constant = 1.

Value

Weighted median absolute deviation from the (weighted) median

See Also

Overview (of all implemented functions)

Examples

```
head(workplace)

# normalized weighted MAD (default)
weighted_mad(workplace$employment, workplace$weight)

# weighted MAD (without normalization)
weighted_mad(workplace$employment, workplace$weight, constant = 1)
```

---

weighted_mean — *Weighted Total and Mean (Horvitz-Thompson and Hajek Estimators)*

Description

Weighted total and mean (Horvitz-Thompson and Hajek estimators)

Usage

```
weighted_mean(x, w, na.rm = FALSE)
weighted_total(x, w, na.rm = FALSE)
```

Arguments

x	[numeric vector] data.
w	[numeric vector] weights (same length as x).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).

Details

weighted_total and weighted_mean compute, respectively, the Horvitz-Thompson estimator of the population total and the Hajek estimator of the population mean.

Value

Estimated population mean or total

See Also

Overview (of all implemented functions)

Examples

```
head(workplace)

# Horvitz-Thompson estimator of the total
weighted_total(workplace$employment, workplace$weight)

# Hajek estimator of the mean
weighted_mean(workplace$employment, workplace$weight)
```

---

weighted_mean_dalen *Dalen Estimators of the Mean and Total*

Description

Dalén's estimators of the population mean and the population total (bare-bone functions with limited functionality).

Usage

```
weighted_mean_dalen(x, w, censoring, type = "Z2", info = FALSE,
                    na.rm = FALSE, verbose = TRUE)
weighted_total_dalen(x, w, censoring, type = "Z2", info = FALSE,
                     na.rm = FALSE, verbose = TRUE)
```

Arguments

x	[numeric vector] data.
w	[numeric vector] weights (same length as x).
censoring	[double] cutoff threshold above which the observations are censored.
type	[character] type of estimator; either "Z2" or "Z3" (default: "Z2").
info	[logical] indicating whether additional information should be returned (default: FALSE).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).
verbose	[logical] indicating whether additional information should be printed to the console (default: FALSE).

Details

Let $\sum_{i \in s} w_i x_i$ denote the expansion estimator of the x-total (summation is over all elements i in sample s). The estimators Z2 and Z3 of Dalén (1987) are defined as follows.

Estimator Z2 The estimator Z2 of the population total sums over $\min(c, w_i x_i)$; hence, it censors the products $w_i x_i$ to the censoring constant c (censoring). The estimator of the population x-mean is is defined as the total divided by the population size.

Estimator Z3 The estimator Z3 of the population total is defined as the sum over the elements z_i, which is equal to $z_i = w_i x_i$ if $w_i y_i \leq c$ and $z_i = c + (y_i - c/w_i)$ otherwise.

Value

The return value depends on info:

info = FALSE: estimate of mean or total [double]

info = TRUE: a [list] with items:

- characteristic [character],
- estimator [character],
- estimate [double],
- variance (default: NA),
- robust [list],
- residuals [numeric vector],
- model [list],
- design (default: NA),
- [call]

References

Dalén, J. (1987). Practical Estimators of a Population Total Which Reduce the Impact of Large Observations. R & D Report U/STM 1987:32, Statistics Sweden, Stockholm.

See Also

Overview (of all implemented functions)

Examples

```
head(workplace)

# Dalen's estimator of the total (with censoring threshold: 100000)
weighted_total_dalen(workplace$employment, workplace$weight, 100000)
```

---

weighted_mean_trimmed *Weighted Trimmed Mean and Total (bare-bone functions)*

Description

Weighted trimmed mean and total (bare-bone functions with limited functionality; see svymean_trimmed and svytotal_trimmed for more capable methods)

Usage

```
weighted_mean_trimmed(x, w, LB = 0.05, UB = 1 - LB, info = FALSE,
                      na.rm = FALSE)
weighted_total_trimmed(x, w, LB = 0.05, UB = 1 - LB, info = FALSE,
                       na.rm = FALSE)
```

Arguments

x	[numeric vector] data.
w	[numeric vector] weights (same length as x).
LB	[double] lower bound of trimming such that $0 \leq$ LB $<$ UB ≤ 1.
UB	[double] upper bound of trimming such that $0 \leq$ LB $<$ UB ≤ 1.
info	[logical] indicating whether additional information should be returned (default: FALSE).
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).

Details

Characteristic. Population mean or total. Let μ denote the estimated trimmed population mean; then, the estimated trimmed population total is given by $\hat{N}\mu$ with $\hat{N} = \sum w_i$, where summation is over all observations in the sample.

Trimming. The methods trims the LB $\cdot 100\%$ of the smallest observations and the (1 - UB) $\cdot 100\%$ of the largest observations from the data.

Variance estimation. See survey methods:

- svymean_trimmed,
- svytotal_trimmed.

Value

The return value depends on info:

info = FALSE**:** estimate of mean or total [double]

info = TRUE**:** a [list] with items:

- characteristic [character],
- estimator [character],
- estimate [double],
- variance (default: NA),
- robust [list],
- residuals [numeric vector],
- model [list],
- design (default: NA),
- [call]

See Also

Overview (of all implemented functions)

svymean_trimmed and svytotal_trimmed

Examples

```
head(workplace)

# Estimated trimmed population total (5% symmetric trimming)
weighted_total_trimmed(workplace$employment, workplace$weight, LB = 0.05,
    UB = 0.95)

# Estimated trimmed population mean (5% trimming at the top of the distr.)
weighted_mean_trimmed(workplace$employment, workplace$weight, UB = 0.95)
```

---

weighted_mean_winsorized

Weighted Winsorized Mean and Total (bare-bone functions)

Description

Weighted winsorized mean and total (bare-bone functions with limited functionality; see `svymean_winsorized` and `svytotal_winsorized` for more capable methods)

Usage

```
weighted_mean_winsorized(x, w, LB = 0.05, UB = 1 - LB, info = FALSE,
                         na.rm = FALSE)
weighted_mean_k_winsorized(x, w, k, info = FALSE, na.rm = FALSE)
weighted_total_winsorized(x, w, LB = 0.05, UB = 1 - LB, info = FALSE,
                          na.rm = FALSE)
weighted_total_k_winsorized(x, w, k, info = FALSE, na.rm = FALSE)
```

Arguments

`x`	`[numeric vector]` data.
`w`	`[numeric vector]` weights (same length as `x`).
`LB`	`[double]` lower bound of winsorization such that $0 \leq \texttt{LB} < \texttt{UB} \leq 1$.
`UB`	`[double]` upper bound of winsorization such that $0 \leq \texttt{LB} < \texttt{UB} \leq 1$.
`info`	`[logical]` indicating whether additional information should be returned (default: `FALSE`).
`na.rm`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds (default: `FALSE`).
`k`	`[integer]` number of observations to be winsorized at the top of the distribution.

Details

Characteristic. Population mean or total. Let μ denote the estimated winsorized population mean; then, the estimated population total is given by $\hat{N}\mu$ with $\hat{N} = \sum w_i$, where summation is over all observations in the sample.

Modes of winsorization. The amount of winsorization can be specified in relative or absolute terms:

- *Relative:* By specifying `LB` and `UB`, the methods winsorizes the `LB` $\cdot 100\%$ of the smallest observations and the (1 - `UB`) $\cdot 100\%$ of the largest observations from the data.
- *Absolute:* By specifying argument `k` in the functions with the "infix" `_k_` in their name, the largest k observations are winsorized, $0 < k < n$, where n denotes the sample size. E.g., `k = 2` implies that the largest and the second largest observation are winsorized.

Variance estimation. See survey methods:

- `svymean_winsorized`,
- `svytotal_winsorized`,
- `svymean_k_winsorized`,
- `svytotal_k_winsorized`.

Value

The return value depends on `info`:

`info = FALSE`: estimate of mean or total `[double]`

`info = TRUE`: a `[list]` with items:

- `characteristic [character]`,
- `estimator [character]`,
- `estimate [double]`,
- `variance` (default: `NA`),
- `robust [list]`,
- `residuals [numeric vector]`,
- `model [list]`,
- `design` (default: `NA`),
- `[call]`

See Also

Overview (of all implemented functions)

`svymean_winsorized`, `svymean_k_winsorized`, `svytotal_winsorized` and `svytotal_k_winsorized`

Examples

```
head(workplace)

# Estimated winsorized population mean (5% symmetric winsorization)
weighted_mean_winsorized(workplace$employment, workplace$weight, LB = 0.05)
```

```
# Estimated one-sided k winsorized population total (2 observations are
# winsorized at the top of the distribution)
weighted_total_k_winsorized(workplace$employment, workplace$weight, k = 2)
```

`weighted_median` *Weighted Median*

Description

Weighted population median.

Usage

```
weighted_median(x, w, na.rm = FALSE)
```

Arguments

`x`	`[numeric vector]` data.
`w`	`[numeric vector]` weights (same length as `x`).
`na.rm`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds (default: `FALSE`).

Details

Weighted sample median; see `weighted_quantile` for more information.

Value

Weighted estimate of the population median

See Also

Overview (of all implemented functions)

`weighted_quantile`

Examples

```
head(workplace)

weighted_median(workplace$employment, workplace$weight)
```

`weighted_median_line` *Robust Simple Linear Regression Based on Medians*

Description

Robust simple linear regression based on medians: two methods are available: `"slopes"` and `"product"`.

Usage

```
weighted_median_line(x, y = NULL, w, type = "slopes", na.rm = FALSE)
```

Arguments

`x`	`[numeric vector]` explanatory variable.
`y`	`[numeric vector]` response variable (default: `NULL`).
`w`	`[numeric vector]` weights (same length as `x`).
`type`	`[character]` `"slopes"` or `"products"` (default: `"slopes"`).
`na.rm`	`[logical]` indicating whether `NA` values should be removed before the computation proceeds (default: `FALSE`).

Details

Overview. Robust simple linear regression based on medians

Type. Two methods/ types are available. Let $m(x, w)$ denote the weighted median of variable `x` with weights `w`:

`type = "slopes"`**:** The slope is computed as

$$b1 = m\left(\frac{y - m(y, w)}{x - m(x, w)}, w\right).$$

`type = "products"`**:** The slope is computed as

$$b1 = \frac{m\big([y - m(y, w)][x - m(x, w)], w\big)}{m\big([x - m(x, w)]^2, w\big)}.$$

Value

A vector with two components: intercept and slope

See Also

Overview (of all implemented functions)

`line`, `weighted_line` and `weighted_median_ratio`

Examples

```
x <- c(1, 2, 4, 5)
y <- c(3, 2, 7, 4)
weighted_line(y ~ x, w = rep(1, length(x)))
weighted_median_line(y ~ x, w = rep(1, length(x)))
m <- weighted_median_line(y ~ x, w = rep(1, length(x)), type = "prod")
m
coef(m)
fitted(m)
residuals(m)

# cars data
head(cars)
with(cars, weighted_median_line(dist ~ speed, w = rep(1, length(dist))))
with(cars, weighted_median_line(dist ~ speed, w = rep(1, length(dist)),
type = "prod"))

# weighted
w <- c(rep(1,20), rep(2,20), rep(5, 10))
with(cars, weighted_median_line(dist ~ speed, w = w))
with(cars, weighted_median_line(dist ~ speed, w = w, type = "prod"))

# outlier in y
cars$dist[49] <- 360
with(cars, weighted_median_line(dist ~ speed, w = w))
with(cars, weighted_median_line(dist ~ speed, w = w, type = "prod"))

# outlier in x
data(cars)
cars$speed[49] <- 72
with(cars, weighted_median_line(dist ~ speed, w = w))
with(cars, weighted_median_line(dist ~ speed, w = w, type = "prod"))
```

---

weighted_median_ratio *Weighted Robust Ratio Estimator Based on Median*

Description

A weighted median of the ratios y/x determines the slope of a regression through the origin.

Usage

```
weighted_median_ratio(x, y = NULL, w, na.rm = FALSE)
```

Arguments

x	[numeric vector] explanatory variable.
y	[numeric vector] response variable (default: NULL).
w	[numeric vector] weights (same length as x).

na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).

Value

A vector with two components: intercept and slope

See Also

Overview (of all implemented functions)

line, weighted_line and weighted_median_line

Examples

```
x <- c(1,2,4,5)
y <- c(1,0,5,2)
m <- weighted_median_ratio(y ~ x, w = rep(1, length(y)))
m
coef(m)
fitted(m)
residuals(m)
```

---

weighted_quantile *Weighted Quantile*

Description

Weighted population quantile.

Usage

```
weighted_quantile(x, w, probs, na.rm = FALSE)
```

Arguments

x	[numeric vector] data.
w	[numeric vector] weights (same length as x).
probs	[numeric vector] vector of probabilities with values in [0,1].
na.rm	[logical] indicating whether NA values should be removed before the computation proceeds (default: FALSE).

Details

Overview. `weighted_quantile` computes the weighted sample quantiles; argument `probs` allows vector inputs.

Implementation. The function is based on a weighted version of the quickselect/Find algorithm with the Bentley and McIlroy (1993) 3-way partitioning scheme. For very small arrays, we use insertion sort.

Compatibility. For equal weighting, i.e., when all elements in `w` are equal, `weighted_quantile` is identical with `type = 2` of `stats::quantile`; see also Hyndman and Fan (1996).

Value

Weighted estimate of the population quantiles

References

Bentley, J. L. and McIlroy, D. M. (1993). Engineering a Sort Function, *Software - Practice and Experience* **23**, 1249–1265. doi:10.1002/spe.4380231105

Hyndman, R.J. and Fan, Y. (1996). Sample Quantiles in Statistical Packages, *The American Statistician* **50**, 361–365. doi:10.1080/00031305.1996.10473566

See Also

Overview (of all implemented functions)

`weighted_median`

Examples

```
head(workplace)

# Weighted 25% quantile (1st quartile)
weighted_quantile(workplace$employment, workplace$weight, 0.25)
```

`wgt_functions` *Weight Functions (for the M- and GM-Estimators)*

Description

Weight functions associated with the Huber and the Tukey biweight psi-functions; and the weight function of Simpson et al. (1992) for GM-estimators.

Usage

```
huberWgt(x, k = 1.345)
tukeyWgt(x, k = 4.685)
simpsonWgt(x, a, b)
```

Arguments

x [numeric vector] data.

k [double] robustness tuning constant ($0 < k \leq \infty$).

a [double] robustness tuning constant ($0 \leq a \leq \infty$); see details below.

b [double] robustness tuning constant ($0 < b \leq \infty$; see details below.

Details

The functions huberWgt and tukeyWgt return the weights associated with the respective psi-function.

The function simpsonWgt is used (in regression GM-estimators) to downweight leverage observations (i.e., outliers in the model's design space). Let d_i denote the (robust) squared Mahalanobis distance of the i-th observation. The Simpson et al. (1992) type of weight is defined as $\min\{1, (b/d_i)^{a/2}\}$, where a and b are tuning constants.

- By default, a = 1; this choice implies that the weights are computed on the basis of the robust Mahalanobis distances. Alternative: a = Inf implies a weight of zero for all observations whose (robust) squared Mahalanobis is larger than b.
- The tuning constants b is a threshold on the distances.

Value

Numerical vector of weights

References

Simpson, D. G., Ruppert, D. and Carroll, R.J. (1992). On One-Step GM Estimates and Stability of Inferences in Linear Regression. *Journal of the American Statistical Association* **87**, 439–450. doi:10.2307/2290275

See Also

Overview (of all implemented functions)

svyreg_huberM, svyreg_huberGM, svyreg_tukeyM and svyreg_tukeyGM

Examples

```
head(flour)

# standardized distance from median (copper content in wholemeal flour)
x <- flour$copper
z <- abs(x - median(x)) / mad(x)

# plot of weight functions vs. distance
plot(z, huberWgt(z, k = 3), ylim = c(0, 1), xlab = "distance",
     ylab = "weight")
points(z, tukeyWgt(z, k = 6), pch = 2, col = 2)
points(z, simpsonWgt(z, a = Inf, b = 3), pch = 3, col = 4)
legend("topright", c("huberWgt(k = 3)", "tukeyWgt(k = 6)",
       "simpsonWgt(a = Inf, b = 3)"), pch = 1:3, col = c(1, 2, 4))
```

`workplace` *(Modified) Canadian Workplace and Employee Survey*

Description

The `workplace` data are from Fuller (2009, pp. 366–367).

Usage

```
data(workplace)
```

Format

A `data.frame` with a sample of 142 workplaces on the following variables

`ID` identifier variable `[integer]`.

`weight` sampling weight `[double]`.

`employment` employment total `[double]`.

`payroll` payroll total (1000 dollars)`[double]`.

`strat` stratum identifier`[integer]`.

`fpc` finite population correction `[integer]`.

Details

The `workplace` data represent a sample of workplaces in the retail sector in a Canadian province. The data are *not* those collected by Statistics Canada, but have been generated by Fuller (2009, Example 3.1.1) to display similar characteristics to the original 1999 Canadian Workplace and Employee Survey (WES).

Sampling design of the 1999 WES: The WES target population is defined as all workplaces operating in Canada with paid employees. The sampling frame is stratified by industry, geographic region, and size (size is defined using estimated employment). A sample of workplaces has been drawn independently in each stratum using simple random sample without replacement (the stratum-specific sample sizes are determined by Neyman allocation). Several strata containing very large workplaces were sampled exhaustively; see Patak et al (1998). The original sampling weights were adjusted for nonresponse.

Remarks by Fuller (2009, p. 365): The original weights of WES were about 2200 for the stratum of small workplaces, about 750 for medium-sized, and about 35 for large workspaces.

Source

The data `workplace` is from Table 6.3 in Fuller (2009, pp. 366–367).

References

Fuller, W. A. (2009). *Sampling Statistics*, Hoboken (NJ): John Wiley and Sons. doi:10.1002/9780470523551

Patak, Z., Hidiroglou, M. and Lavallée, P. (1998). The methodology of the Workplace and Employee Survey. *Proceedings of the Survey Research Methods Section, American Statistical Association*, 83–91.

Examples

```
head(workplace)

library("survey")
# Survey design for stratified simple random sampling without replacement
dn <- if (packageVersion("survey") >= "4.2") {
        # survey design with pre-calibrated weights
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace, calibrate.formula = ~-1 + strat)
    } else {
        # legacy mode
        svydesign(ids = ~ID, strata = ~strat, fpc = ~fpc, weights = ~weight,
                  data = workplace)
    }
```

Index