Package: gustave (via r-universe)

September 11, 2024

Type Package

Title A User-Oriented Statistical Toolkit for Analytical Variance Estimation

Depends R(>= 3.2.5)

Imports methods, utils, stats, Matrix

Suggests testthat, sampling, magrittr, tibble, dplyr, data.table

Version 1.0.0

URL https://github.com/InseeFr/gustave

BugReports https://github.com/InseeFr/gustave/issues

Description Provides a toolkit for analytical variance estimation in survey sampling. Apart from the implementation of standard variance estimators, its main feature is to help the sampling expert produce easy-to-use variance estimation ``wrappers'', where systematic operations (linearization, domain estimation) are handled in a consistent and transparent way.

License GPL-3

Collate 'data.R' 'utils.R' 'define_variance_wrapper.R' 'variance_function.R' 'define_statistic_wrapper.R' 'standard_statistic_wrapper.R' 'qvar.R'

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/inseefr/gustave

RemoteRef HEAD

RemoteSha b9b156e84631d2cbf873d4d3fb5038dc8064763c

Contents

`add_zero` *Expand a matrix or a data.frame with zeros based on rownames matching*

Description

For a given two-dimensional object with rownames and a character vector, `add_zero` produces a corresponding object whose rownames match the character vector, with zeros on the additional rows.

This function is an easy-to-use and reliable way to reintroduce non-responding units in the variance estimation process (after the non-response phase is taken into account).

Usage

```
add_zero(y, rownames, remove = TRUE)
```

Arguments

`y`	A (sparse) matrix or a data.frame. The object to add zeros to.
`rownames`	A character vector (other types are coerced to character). The character vector giving the rows of the produced object.
`remove`	Should rows of `y` whose name do not appear in the rownames argument be removed ? TRUE by default, a warning is shown when rows are removed.

Value

A (sparse) matrix or data.frame depending on the type of `y`.

Author(s)

Martin Chevalier

Examples

```
# Data generation
set.seed(1)
n <- 10
p <- 2
y <- matrix(1:(n*p), ncol = p, dimnames = list(sample(letters, n)))
y[c(3, 8, 12)] <- NA
rownames <- letters

# Standard use
add_zero(y, rownames)

# Use when rownames in y do not match
# any element in the rownames argument
rownames(y)[1:3] <- toupper(rownames(y)[1:3])
add_zero(y, rownames)
add_zero(y, rownames, remove = FALSE)
```

---

define_statistic_wrapper

Define a statistic wrapper

Description

`define_statistic_wrapper` defines statistic *wrappers* to be used together with `variance estimation wrappers`. A statistic wrapper produces both the point estimator and the linearized variable associated with a given statistic to estimate variance on (Deville, 1999). `define_statistic_wrapper` is intended for **advanced use only**, standard statistic wrappers are included in the gustave package (see `standard statistic wrappers`)

Usage

```
define_statistic_wrapper(
  statistic_function,
  arg_type,
  arg_not_affected_by_domain = NULL,
  display_function = standard_display
)
```

Arguments

`statistic_function`
: An R function specific to the statistic to calculate. It should produce at least the point estimator and the linearized variable associated with the statistic (see Details).

`arg_type`
: A named list with three character vectors describing the type of each argument of `statistic_function` (see Details).

`arg_not_affected_by_domain`
: A character vector indicating the arguments which should not be affected by domain-splitting. Such parameters may appear in some complex linearization formula, for instance when the At-Risk of Poverty Rate (ARPR) is estimated by region but with a poverty line calculated at the national level.

`display_function`
: An R function which produces, for each variance estimation, the data.frame to be displayed by the variance estimation wrapper. The default display function (`standard_display`) uses standard metadata to display usual variance indicator (point estimate, variance, standard deviation, coefficient of variation, confidence interval) broken down by statistic wrapper, domain (if any) and level (if the variable is a factor).

Details

When the statistic to estimate is not a total, the application of analytical variance estimation formulae developed for the estimator of a total is not straightforward (Deville, 1999). An asymptotically unbiased variance estimator can nonetheless be obtained if the estimation of variance is performed on a variable obtained from the original data through a linearization step.

`define_statistic_wrapper` is the function used to create, for a given statistic, an easy-to-use function which calculates both the point estimator and the linearized variable associated with the statistic. These operations are implemented by the `statistic_function`, which can have any needed input (for example `num` and `denom` for a ratio estimator) and should output a list with at least two named elements:

- `point`: the point estimator of the statistic
- `lin`: the linearized variable to be passed on to the variance estimation formula. If several variables are to be associated with the statistics, `lin` can be a list itself.

All other named elements in the output of `define_statistic_wrapper` are treated as metadata (that may be used later on by `display_function`).

`arg_type` is a named list of three elements that describes the nature of the argument of `statistic_function`:

- `data`: data argument(s), numerical vector(s) to be used to calculate the point estimate and the linearized variable associated with the statistic
- `weight`: weight argument, numerical vector to be used as row weights
- `param`: parameters, non-data arguments to be used to control some aspect of the computation

Statistic wrappers are quite flexible tools to apply a variance function to an estimator requiring a linearization step (e.g. all estimators except the estimator of a total) with virtually no additional complexity for the end-user.

standard statistic wrappers are included within the gustave package and automatically added to the variance estimation wrappers. New statistic wrappers can be defined using the define_statistic_wrapper and then explicitly added to the variance estimation wrappers using the objects_to_include argument.

Note: To some extent, statistic wrappers can be seen as ggplot2 geom_ and stat_ functions: they help the end-user in writing down what he or she wants without having to go too deep into the details of the corresponding layers.

Value

A function to be used within a variance estimation wrapper to estimate a specific statistic (see examples). Its formals are the ones of statistic_function with the addition of by and where (for domain estimation, set to NULL by default).

Author(s)

Martin Chevalier

References

Deville J.-C. (1999), "Variance estimation for complex statistics and estimators: linearization and residual techniques", *Survey Methodology*, 25:193–203

See Also

standard statistic wrappers, define_variance_wrapper

Examples

```
### Example from the Information and communication technologies (ICT) survey

# Let's define a variance wrapper asfor the ICT survey
# as in the examples of the qvar function:
precision_ict <- qvar(
  data = ict_sample,
  dissemination_dummy = "dissemination",
  dissemination_weight = "w_calib",
  id = "firm_id",
  scope_dummy = "scope",
  sampling_weight = "w_sample",
  strata = "strata",
  nrc_weight = "w_nrc",
  response_dummy = "resp",
  hrg = "hrg",
  calibration_weight = "w_calib",
  calibration_var = c(paste0("N_", 58:63), paste0("turnover_", 58:63)),
  define = TRUE
)
precision_ict(ict_survey, mean(speed_quanti))

# Let's now redefine the mean statistic wrapper
```

```
mean2 <- define_statistic_wrapper(
  statistic_function = function(y, weight){
    point <- sum(y * weight) / sum(weight)
    lin <- (y - point) / sum(weight)
    list(point = point, lin = lin, metadata = list(n = length(y)))
  },
  arg_type = list(data = "y", weight = "weight")
)

# mean2 can now be used inside precision_ict (and yields
# the same results as the mean statistic wrapper)
precision_ict(ict_survey, mean(speed_quanti), mean2(speed_quanti))
```

`define_variance_wrapper`

Define a variance estimation wrapper

Description

Given a variance estimation *function* (specific to a survey), `define_variance_wrapper` defines a variance estimation *wrapper* easier to use (e.g. automatic domain estimation, linearization).

Usage

```
define_variance_wrapper(
  variance_function,
  reference_id,
  reference_weight,
  default_id = NULL,
  technical_data = NULL,
  technical_param = NULL,
  objects_to_include = NULL
)
```

Arguments

`variance_function`
: An R function. It is the methodological workhorse of the variance estimation: from a set of arguments including the variables of interest (see below), it should return a vector of estimated variances. See Details.

`reference_id`
: A vector containing the ids of all the responding units of the survey. It can also be an unevaluated expression (enclosed in `quote()`) to be evaluated within the execution environment of the wrapper. It is compared with `default$id` (see below) to check whether some observations are missing in the survey file. The matrix of variables of interest passed on to `variance_function` has `reference_id` as rownames and is ordered according to its values.

`reference_weight`
: A vector containing the reference weight of the survey. It can also be an unevaluated expression (enclosed in `quote()`) to be evaluated within the execution environment of the wrapper.

`default_id`
: A character vector of length 1, the name of the default identifying variable in the survey file. It can also be an unevaluated expression (enclosed in `quote()`) to be evaluated within the survey file.

`technical_data`
: A named list of technical data needed to perform the variance estimation (e.g. sampling strata, first- or second-order probabilities of inclusion, estimated response probabilities, calibration variables). Its names should match the names of the corresponding arguments in `variance_function`.

`technical_param`
: A named list of technical parameters used to control some aspect of the variance estimation process (e.g. alternative methodology). Its names should match the names of the corresponding arguments in `variance_function`.

`objects_to_include`
: (Advanced use) A character vector indicating the name of additional R objects to include within the variance wrapper.

Details

Defining variance estimation wrappers is the **key feature** of the `gustave` package. It is the workhorse of the ready-to-use `qvar` function and should be used directly to handle more complex cases (e.g. surveys with several stages or balanced sampling).

Analytical variance estimation is often difficult to carry out by non-specialists owing to the complexity of the underlying sampling and estimation methodology. This complexity yields complex *variance estimation functions* which are most often only used by the sampling expert who actually wrote them. A *variance estimation wrapper* is an intermediate function that is "wrapped around" the (complex) variance estimation function in order to provide the non-specialist with user-friendly features (see examples):

- calculation of complex statistics (see `standard statistic wrappers`)
- domain estimation
- handy evaluation and factor discretization

`define_variance_wrapper` allows the sampling expert to define a variance estimation wrapper around a given variance estimation function and set its default parameters. The produced variance estimation wrapper is standalone in the sense that it contains all technical data necessary to carry out the estimation (see `technical_data`).

The arguments of the `variance_function` fall into three types:

- the data argument (mandatory, only one allowed): the numerical matrix of variables of interest to apply the variance estimation formula on
- technical data arguments (optional, one or more allowed): technical and methodological information used by the variance estimation function (e.g. sampling strata, first- or second-order probabilities of inclusion, estimated response probabilities, calibration variables)
- technical parameters (optional, one or more allowed): non-data arguments to be used to control some aspect of the variance estimation (e.g. alternative methodology)

technical_data and technical_param are used to determine which arguments of variance_function relate to technical information, the only remaining argument is considered as the data argument.

Value

An R function that makes the estimation of variance based on the provided variance function easier. Its parameters are:

- data: one or more calls to a statistic wrapper (e.g. total(), mean(), ratio()). See examples and standard statistic wrappers) and standard statistic wrappers)
- where: a logical vector indicating a domain on which the variance estimation is to be performed
- by: q qualitative variable whose levels are used to define domains on which the variance estimation is performed
- alpha: a numeric vector of length 1 indicating the threshold for confidence interval derivation (0.05 by default)
- display: a logical verctor of length 1 indicating whether the result of the estimation should be displayed or not
- id: a character vector of size 1 containing the name of the identifying variable in the survey file. Its default value depends on the value of default_id in define_variance_wrapper
- envir: an environment containing a binding to data

Author(s)

Martin Chevalier

References

Rao, J.N.K (1975), "Unbiased variance estimation for multistage designs", *Sankhya*, C n°37

See Also

qvar, standard statistic wrappers, varDT

Examples

```
### Example from the Labour force survey (LFS)

# The (simulated) Labour force survey (LFS) has the following characteristics:
# - first sampling stage: balanced sampling of 4 areas (each corresponding to
#   about 120 dwellings) on first-order probability of inclusion (proportional to
#   the number of dwellings in the area) and total annual income in the area.
# - second sampling stage: in each sampled area, simple random sampling of 20
#   dwellings
# - neither non-response nor calibration

# As this is a multi-stage sampling design with balanced sampling at the first
# stage, the qvar function does not apply. A variance wrapper can nonetheless
# be defined using the core define_variance_wrapper function.
```

```
# Step 1 : Definition of the variance function and the corresponding technical data

# In this context, the variance estimation function specific to the LFS
# survey can be defined as follows:

var_lfs <- function(y, ind, dwel, area){

  variance <- list()

  # Variance associated with the sampling of the dwellings
  y <- sum_by(y, ind$id_dwel)
  variance[["dwel"]] <- var_srs(
    y = y, pik = dwel$pik_dwel, strata = dwel$id_area,
    w = (1 / dwel$pik_area^2 - dwel$q_area)
  )

  # Variance associated with the sampling of the areas
  y <- sum_by(y = y, by = dwel$id_area, w = 1 / dwel$pik_dwel)
  variance[["area"]] <- varDT(y = y, precalc = area)

  Reduce(`+`, variance)

}

# where y is the matrix of variables of interest and ind, dwel and area the technical data:

technical_data_lfs <- list()

# Technical data at the area level
# The varDT function allows for the pre-calculation of
# most of the methodological quantities needed.
technical_data_lfs$area <- varDT(
  y = NULL,
  pik = lfs_samp_area$pik_area,
  x = as.matrix(lfs_samp_area[c("pik_area", "income")]),
  id = lfs_samp_area$id_area
)

# Technical data at the dwelling level
# In order to implement Rao (1975) formula for two-stage samples,
# we associate each dwelling with the diagonal term corresponding
# to its area in the first-stage variance estimator:
lfs_samp_dwel$q_area <- with(technical_data_lfs$area, setNames(diago, id))[lfs_samp_dwel$id_area]
technical_data_lfs$dwel <- lfs_samp_dwel[c("id_dwel", "pik_dwel", "id_area", "pik_area", "q_area")]

# Technical data at the individual level
technical_data_lfs$ind <- lfs_samp_ind[c("id_ind", "id_dwel", "sampling_weight")]

# Test of the variance function var_lfs
y <- matrix(as.numeric(lfs_samp_ind$unemp), ncol = 1, dimnames = list(lfs_samp_ind$id_ind))
with(technical_data_lfs, var_lfs(y = y, ind = ind, dwel = dwel, area = area))
```

```
# Step 2 : Definition of the variance wrapper

# Call of define_variance_wrapper
precision_lfs <- define_variance_wrapper(
  variance_function = var_lfs,
  technical_data = technical_data_lfs,
  reference_id = technical_data_lfs$ind$id_ind,
  reference_weight = technical_data_lfs$ind$sampling_weight,
  default_id = "id_ind"
)

# Test
precision_lfs(lfs_samp_ind, unemp)

# The variance wrapper precision_lfs has the same features
# as variance wrappers produced by the qvar function (see
# qvar examples for more details).
```

`ict_pop` *Sampling frame of the Information and communication technologies (ICT) survey*

Description

A (simulated) dataset containing basic identification information and auxiliary variables for the sampling of the Information and communication technologies (ICT) survey in the information and communication sector (NACE rev 2 J section).

Usage

```
ict_pop
```

Format

A data frame with 7670 observations and 5 variables:

firm_id identifier of the firm

division identifier of the economic sub-sector

employees number of employees

turnover firm turnover, in thousand euros

strata stratification variable

See Also

`qvar`, `ict_sample`, `ict_survey`

`ict_sample` *Sample of the Information and communication technologies (ICT) survey*

Description

A (simulated) dataset containing sampling information about the sample of the Information and communication technologies (ICT) survey in the information and communication sector (NACE rev 2 J section)

Usage

```
ict_sample
```

Format

A data frame with 650 observations and 8 variables:

firm_id identifier of the firm

division identifier of the economic sub-sector

employees number of employees

turnover firm turnover, in euros

strata stratification variable

w_sample sampling weight

scope boolean indicating whether the firm did belong to the scope of the survey or not

resp boolean indicating whether the firm did respond to the survey or not

nrc boolean indicating whether the firm did take part in the non-response correction process or not

hrg homogeneous response group used for the non-response correction

response_prob_est response probability of the unit estimated using homogeneous response groups

w_nrc weight after unit non-response correction

calib boolean indicating whether the firm was integrated in the calibration process or not (`TRUE` for all responding units)

N_58, N_59, N_60, N_61, N_62, N_63, turnover_58, turnover_59, turnover_60, turnover_61, turnover_62, turnover_6 calibration variables (number of firms and turnover broken down by economic sub-sector)

w_calib calibrated weight

dissemination boolean indicating whether the unit appears in the dissemination file

See Also

`qvar`, `ict_pop`, `ict_survey`

ict_survey — *Survey data of the Information and communication technologies (ICT) survey*

Description

A (simulated) dataset containing calibration and survey variables of the respondents to the Information and communication technologies (ICT) survey in the information and communication sector (NACE rev 2 J section)

Usage

```
ict_survey
```

Format

A data frame with 612 observations and 11 variables:

firm_id identifier of the firm

division identifier of the economic sub-sector

employees number of employees

turnover firm turnover, in euros

w_calib calibrated weight

speed_quanti, speed_quanti_NA internet connection speed of the firm in Mbps, without or with missing values

speed_quali, speed_quali_NA internet connection speed of the firm recoded in classes, without or with missing values

big_data, big_data_NA use of big data analytics within the firm, without or with missing values

See Also

qvar, ict_pop, ict_sample

lfs_samp_area — *Sample of areas in the Labour force survey*

Description

A (simulated) dataset containing information about 4 geographical areas (about 120 dwellings each) sampled for the labour force survey.

Usage

```
lfs_samp_area
```

Format

A data frame with 4 observations and 3 variables:

id_area identifier of the area

income total annual income of the area in thousand euros (from income registry)

pik_area first-order inclusion probability of the area (proportional to the number of dwellings in the area)

See Also

define_variance_wrapper, lfs_samp_dwel, lfs_samp_ind

---

lfs_samp_dwel *Sample of dwellings in the Labour force survey*

Description

A (simulated) dataset containing information about 80 dwellings sampled for the Labour force survey (in the 4 areas described in lfs_samp_area).

Usage

```
lfs_samp_dwel
```

Format

A data frame with 80 observations and 6 variables:

id_dwel identifier of the dwelling

id_area identifier of the area

income total annual income of the dwelling in thousand euros (from income registry)

pik_area first-order inclusion probability of the area

pik_dwel first-order inclusion probability of the dwelling within the area (20 dwelling sampled per area)

pik first-order inclusion probability of the dwelling

See Also

define_variance_wrapper, lfs_samp_area, lfs_samp_ind

lfs_samp_ind *Sample of individuals in the Labour force survey*

Description

A (simulated) dataset containing information about 157 individuals sampled for the Labour force survey (all members of the 80 dwellings described in `lfs_samp_dwel`). It also contains the unemployment status extracted from the survey file (no non-response).

Usage

```
lfs_samp_ind
```

Format

A data frame with 157 observations and 5 variables:

id_ind identifier of the individual

id_dwel identifier of the dwelling

income total annual income of the individual in thousand euros (from income registry)

unemp unemployment status

sampling_weight sampling weight of the individual (inverse of the first-order inclusion probability of the dwelling)

See Also

`define_variance_wrapper`, `lfs_samp_area`, `lfs_samp_dwel`

qvar *Quickly perform a variance estimation in common cases*

Description

`qvar` (for "quick variance estimation") is a function performing analytical variance estimation in most common cases, that is:

- stratified simple random sampling
- non-response correction (if any) through reweighting
- calibration (if any)

Used with `define = TRUE`, it defines a so-called variance wrapper, that is a standalone ready-to-use function that can be applied to the survey dataset without having to specify the methodological characteristics of the survey (see `define_variance_wrapper`).

Usage

```
qvar(
  data,
  ...,
  by = NULL,
  where = NULL,
  alpha = 0.05,
  display = TRUE,
  id,
  dissemination_dummy,
  dissemination_weight,
  sampling_weight,
  strata = NULL,
  scope_dummy = NULL,
  nrc_weight = NULL,
  response_dummy = NULL,
  nrc_dummy = NULL,
  calibration_weight = NULL,
  calibration_dummy = NULL,
  calibration_var = NULL,
  define = FALSE,
  envir = parent.frame()
)
```

Arguments

data	The data.frame containing all the technical information required to prepare the variance estimation process (see other arguments below). Note that this file should contain all the units sampled, including the out-of-scope and non-responding units. If a variance estimation is to be performed right away (when define = FALSE), it should also contain the variables of interest.
...	One or more calls to a statistic wrapper (e.g. total(), mean(), ratio()). See examples and standard statistic wrappers
by	A qualitative variable whose levels are used to define domains on which the variance estimation is performed.
where	A logical vector indicating a domain on which the variance estimation is to be performed.
alpha	A numeric vector of length 1 indicating the threshold for confidence interval derivation (0.05 by default).
display	A logical verctor of length 1 indicating whether the result of the estimation should be displayed or not.
id	The identification variable of the units in data. It should be unique for each row in data and not contain any missing values.
dissemination_dummy	A character vector of length 1, the name of the logical variable in data indicating whether the unit does appear in the disseminated file and should be used for point estimates. It should not contain any missing values.

dissemination_weight
: A character vector of length 1, the name of the numerical variable in data corresponding to the dissemination weight of the survey. It should not contain any missing values.

sampling_weight
: A character vector of length 1, the name of the numeric variable in data corresponding to the sampling weights of the survey. It should not contain any missing values.

strata
: A character vector of length 1, the name of the factor variable in data whose level match the stratification used in the survey. Character variables are coerced to factor. If defined, it should not contain any missing value. If NULL, the variance estimation process does not take any stratification into account.

scope_dummy
: A character vector of length 1, the name of the logical variable in data indicating whether the unit belongs to the scope of the survey or not. Numerical variables are coerced to logical. If defined, it should not contain any missing value. If NULL, all units are supposed to be within the scope of the survey.

nrc_weight
: A character vector of length 1, the name of the numerical variable in data corresponding to the weights after non-response correction. If defined, all responding units should have a non-missing value. If NULL, all units are supposed to be responding and the variance estimation process does not include a second phase in order to take non-response into account.

response_dummy
: A character vector of length 1, the name of of the logical variable in data indicating whether the unit is a responding unit or not. Numerical variables are coerced to logical. response_dummy should be defined as long as a nrc_weight is provided. All units in the scope of the survey should have a non-missing value.

nrc_dummy
: A character vector of length 1, the name of the logical variable in data indicating whether the units did take part in the non-response correction process. All units in the scope of the survey should have a non-missing value.

calibration_weight
: A character vector of length 1, the name of the numerical variable in data corresponding to the calibrated weights. If defined, all responding units should have a non-missing value. If NULL, the variance estimation process does not take any calibration step into account.

calibration_dummy
: A character vector of length 1, the name of of the logical variable in data indicating whether the unit did take part in the calibration process or not. Numerical variables are coerced to logical. If defined, all responding units should have a non-missing value. If NULL, calibration is supposed to have been performed on all responding units.

calibration_var
: A character vector, the name of the variable(s) used in the calibration process. Logical variables are coerced to numeric. Character and factor variables are automatically discretized. calibration_var should be defined as long as a calibration_weight is provided. All units taking part in the calibration process should have only non-missing values for all variables in calibration_var.

define
: Logical vector of lentgh 1. Should a variance wrapper be defined instead of performing a variance estimation (see details and examples)?

envir An environment containing a binding to data.

Details

qvar performs not only technical but also methodological checks in order to ensure that the standard variance estimation methodology does apply (e.g. equal probability of inclusion within strata, number of units per stratum).

Used with define = TRUE, the function returns a variance estimation *wrapper*, that is a ready-to-use function that implements the described variance estimation methodology and contains all necessary data to do so (see examples).

Note: To some extent, qvar is analogous to the qplot function in the ggplot2 package, as it is an easier-to-use function for common cases. More complex cases are to be handled by using the core functions of the gustave package, e.g. define_variance_wrapper.

See Also

define_variance_wrapper, standard_statistic_wrapper

Examples

```
### Example from the Information and communication technologies (ICT) survey

# The (simulated) Information and communication technologies (ICT) survey
# has the following characteristics:
# - stratified one-stage sampling design
# - non-response correction through reweighting in homogeneous response groups
# - calibration on margins.

# The ict_survey data.frame is a (simulated) subset of the ICT
# survey file containing the variables of interest for the 612
# responding firms.

# The ict_sample data.frame is the (simulated) sample of 650
# firms corresponding to the ict_survey file. It contains all
# technical information necessary to estimate a variance with
# the qvar() function.

## Methodological description of the survey

# Direct call of qvar()
qvar(

  # Sample file
  data = ict_sample,

  # Dissemination and identification information
  dissemination_dummy = "dissemination",
  dissemination_weight = "w_calib",
  id = "firm_id",

  # Scope
```

```
  scope_dummy = "scope",

  # Sampling design
  sampling_weight = "w_sample",
  strata = "strata",

  # Non-response correction
  nrc_weight = "w_nrc",
  response_dummy = "resp",
  hrg = "hrg",

  # Calibration
  calibration_weight = "w_calib",
  calibration_var = c(paste0("N_", 58:63), paste0("turnover_", 58:63)),

  # Statistic(s) and variable(s) of interest
  mean(employees)

)

# Definition of a variance estimation wrapper
precision_ict <- qvar(

  # As before
  data = ict_sample,
  dissemination_dummy = "dissemination",
  dissemination_weight = "w_calib",
  id = "firm_id",
  scope_dummy = "scope",
  sampling_weight = "w_sample",
  strata = "strata",
  nrc_weight = "w_nrc",
  response_dummy = "resp",
  hrg = "hrg",
  calibration_weight = "w_calib",
  calibration_var = c(paste0("N_", 58:63), paste0("turnover_", 58:63)),

  # Replacing the variables of interest by define = TRUE
  define = TRUE

)

# Use of the variance estimation wrapper
precision_ict(ict_sample, mean(employees))

# The variance estimation wrapper can also be used on the survey file
precision_ict(ict_survey, mean(speed_quanti))

## Features of the variance estimation wrapper

# Several statistics in one call (with optional labels)
precision_ict(ict_survey,
  "Mean internet speed in Mbps" = mean(speed_quanti),
```

```
  "Turnover per employee" = ratio(turnover, employees)
)

# Domain estimation with where and by arguments
precision_ict(ict_survey,
  mean(speed_quanti),
  where = employees >= 50
)
precision_ict(ict_survey,
  mean(speed_quanti),
  by = division
)

# Domain may differ from one estimator to another
precision_ict(ict_survey,
 "Mean turnover, firms with 50 employees or more" = mean(turnover, where = employees >= 50),
 "Mean turnover, firms with 100 employees or more" = mean(turnover, where = employees >= 100)
)

# On-the-fly evaluation (e.g. discretization)
precision_ict(ict_survey, mean(speed_quanti > 100))

# Automatic discretization for qualitative (character or factor) variables
precision_ict(ict_survey, mean(speed_quali))

# Standard evaluation capabilities
variables_of_interest <- c("speed_quanti", "speed_quali")
precision_ict(ict_survey, mean(variables_of_interest))

# Integration with %>% and dplyr
library(magrittr)
library(dplyr)
ict_survey %>%
  precision_ict("Internet speed above 100 Mbps" = mean(speed_quanti > 100)) %>%
  select(label, est, lower, upper)
```

`res_cal` *Linear Regression Residuals Calculation*

Description

`res_cal` calculates linear regression residuals in an efficient way : handling several dependent variables at a time, using Matrix::TsparseMatrix capabilities and allowing for pre-calculation of the matrix inverse.

Usage

```
res_cal(y = NULL, x, w = NULL, by = NULL, precalc = NULL, id = NULL)
```

Arguments

y	A (sparse) numerical matrix of dependent variable(s).
x	A (sparse) numerical matrix of independent variable(s).
w	An optional numerical vector of row weights.
by	An optional categorical vector (factor or character) when residuals calculation is to be conducted within by-groups (see Details).
precalc	A list of pre-calculated results (see Details).
id	A vector of identifiers of the units used in the calculation. Useful when `precalc = TRUE` in order to assess whether the ordering of the y data matrix matches the one used at the precalculation step.

Details

In the context of the gustave package, linear regression residual calculation is solely used to take into account the effect of calibration on variance estimation. Independent variables are therefore most likely to be the same from one variance estimation to another, hence the inversion of the matrix `t(x) %*% Diagonal(x = w) %*% x` can be done once and for all at a pre-calculation step.

The parameters y and precalc determine whether a list of pre-calculated data should be used in order to speed up the regression residuals computation at execution time:

- if y not NULL and precalc NULL : on-the-fly calculation of the matrix inverse and the regression residuals (no pre-calculation).
- if y NULL and precalc NULL : pre-calculation of the matrix inverse which is stored in a list of pre-calculated data.
- if y not NULL and precalc not NULL : calculation of the regression residuals using the list of pre-calculated data.

The by parameter allows for calculation within by-groups : all calculation are made separately for each by-group (when calibration was conducted separately on several subsamples), but in an efficient way using Matrix::TsparseMatrix capabilities (especially when the matrix inverse is pre-calculated).

Value

- if y is not NULL (calculation step) : a numerical matrix with same structure (regular base::matrix or Matrix::TsparseMatrix) and dimensions as y.
- if y is NULL (pre-calculation step) : a list containing pre-calculated data.

Author(s)

Martin Chevalier

Examples

```
# Generating random data
set.seed(1)
n <- 100
H <- 5
y <- matrix(rnorm(2*n), nrow = n)
x <- matrix(rnorm(10*n), nrow = n)
by <- letters[sample(1:H, n, replace = TRUE)]

# Direct calculation
res_cal(y, x)

# Calculation with pre-calculated data
precalc <- res_cal(y = NULL, x)
res_cal(y, precalc = precalc)
identical(res_cal(y, x), res_cal(y, precalc = precalc))

# Matrix::TsparseMatrix capability
require(Matrix)
X <- as(x, "TsparseMatrix")
Y <- as(y, "TsparseMatrix")
identical(res_cal(y, x), as.matrix(res_cal(Y, X)))

# by parameter for within by-groups calculation
res_cal(Y, X, by = by)
all.equal(
 res_cal(Y, X, by = by)[by == "a", ],
  res_cal(Y[by == "a", ], X[by == "a", ])
)
```

---

standard_statistic_wrapper

Standard statistic wrappers

Description

Functions to be used within variance estimation wrappers in order to specify which statistic is to be estimated.

Usage

```
total(y, by = NULL, where = NULL)

ratio(num, denom, by = NULL, where = NULL)

mean(y, by = NULL, where = NULL)

diff_of_ratio(num1, denom1, num2, denom2, by = NULL, where = NULL)
```

```
ratio_of_ratio(num1, denom1, num2, denom2, by = NULL, where = NULL)
```

Arguments

y	A vector corresponding to the variable to calculate the statitic on. If not numeric (character or factor), it is automatically discretized.
by	Factor vector (character vectors are coerced to factors) whose levels are used to break down the estimation by domains.
where	Logical vector indicating the domain to perform variance estimation on.
num, num1, num2	Numerical vector(s) corresponding to the numerator(s) to be used in the estimation.
denom, denom1, denom2	Numerical vector(s) corresponding to the denominator(s) to be used in the estimation.

Details

When the estimator is not the estimator of a total, the application of analytical variance estimation formulae developed for the estimator of a total is not straightforward (Deville, 1999). An asymptotically unbiased variance estimator can nonetheless be obtained if the estimation of variance is performed on a variable obtained from the original data through a linerization step.

The `ratio`, `mean`, `diff_of_ratio` and `ratio_of_ratio` functions produce the point estimate of the statistic and derive the corresponding linearized variable which is later on passed on to the variance estimation function within the variance estimation wrapper.

Note: The `total` function does not perform any linearization (as none is needed for the estimator of a total) and solely produces the corresponding point estimator.

Author(s)

Martin Chevalier

References

Caron N. (1998), "Le logiciel Poulpe : aspects méthodologiques", *Actes des Journées de méthodologie statistique* http://jms-insee.fr/jms1998s03_1/

Deville J.-C. (1999), "Variance estimation for complex statistics and estimators: linearization and residual techniques", *Survey Methodology*, 25:193–203

See Also

`define_statistic_wrapper`, `define_variance_wrapper`

Examples

```
# See qvar examples
```

sum_by *Efficient by-group (weighted) summation*

Description

sum_by performs an efficient and optionally weighted by-group summation by using linear algebra and the Matrix package capabilities. The by-group summation is performed through matrix cross-product of the y parameter (coerced to a matrix if needed) with a (very) sparse matrix built up using the by and the (optional) w parameters.

Compared to base R, dplyr or data.table alternatives, this implementation aims at being easier to use in a matrix-oriented context and can yield efficiency gains when the number of columns becomes high.

Usage

```
sum_by(y, by, w = NULL, na_rm = TRUE, keep_sparse = FALSE)
```

Arguments

y	A (sparse) vector, a (sparse) matrix or a data.frame. The object to perform by-group summation on.
by	The factor variable defining the by-groups. Character variables are coerced to factors.
w	The optional row weights to be used in the summation.
na_rm	Should NA values in y be removed (ie treated as 0 in the summation) ? Similar to na.rm argument in sum, but TRUE by default. If FALSE, NA values in y produce NA values in the result.
keep_sparse	When y is a sparse vector or a sparse matrix, should the result also be sparse ? FALSE by default. As sparseVector-class does not have a name attribute, when y is a sparseVector the result does not have any name (and a warning is cast).

Value

A vector, a matrix or a data.frame depending on the type of y. If y is sparse and keep_sparse = TRUE, then the result is also sparse (without names when it is a sparse vector, see keep_sparse argument for details).

Author(s)

Martin Chevalier

Examples

```
# Data generation
set.seed(1)
n <- 100
p <- 10
H <- 3
y <- matrix(rnorm(n*p), ncol = p, dimnames = list(NULL, paste0("var", 1:10)))
y[1, 1] <- NA
by <- letters[sample.int(H, n, replace = TRUE)]
w <- rep(1, n)
w[by == "a"] <- 2

# Standard use
sum_by(y, by)

# Keeping the NAs
sum_by(y, by, na_rm = FALSE)

# With a weight
sum_by(y, by, w = w)
```

---

varDT *Variance approximation with Deville-Tillé (2005) formula*

Description

varDT estimates the variance of the estimator of a total in the case of a balanced sampling design with equal or unequal probabilities using Deville-Tillé (2005) formula. Without balancing variables, it falls back to Deville's (1993) classical approximation. Without balancing variables and with equal probabilities, it falls back to the classical Horvitz-Thompson variance estimator for the total in the case of simple random sampling. Stratification is natively supported.

var_srs is a convenience wrapper for the (stratified) simple random sampling case.

Usage

```
varDT(
  y = NULL,
  pik,
  x = NULL,
  strata = NULL,
  w = NULL,
  precalc = NULL,
  id = NULL
)

var_srs(y, pik, strata = NULL, w = NULL, precalc = NULL)
```

Arguments

y	A (sparse) numerical matrix of the variable(s) whose variance of their total is to be estimated.
pik	A numerical vector of first-order inclusion probabilities.
x	An optional (sparse) numerical matrix of balancing variable(s).
strata	An optional categorical vector (factor or character) when variance estimation is to be conducted within strata.
w	An optional numerical vector of row weights (see Details).
precalc	A list of pre-calculated results (see Details).
id	A vector of identifiers of the units used in the calculation. Useful when precalc = TRUE in order to assess whether the ordering of the y data matrix matches the one used at the pre-calculation step.

Details

varDT aims at being the workhorse of most variance estimation conducted with the gustave package. It may be used to estimate the variance of the estimator of a total in the case of (stratified) simple random sampling, (stratified) unequal probability sampling and (stratified) balanced sampling. The native integration of stratification based on Matrix::TsparseMatrix allows for significant performance gains compared to higher level vectorizations (*apply especially).

Several time-consuming operations (e.g. collinearity-check, matrix inversion) can be pre-calculated in order to speed up the estimation at execution time. This is determined by the value of the parameters y and precalc:

- if y not NULL and precalc NULL : on-the-fly calculation (no pre-calculation).
- if y NULL and precalc NULL : pre-calculation whose results are stored in a list of pre-calculated data.
- if y not NULL and precalc not NULL : calculation using the list of pre-calculated data.

w is a row weight used at the final summation step. It is useful when varDT or var_srs are used on the second stage of a two-stage sampling design applying the Rao (1975) formula.

Value

- if y is not NULL (calculation step) : the estimated variances as a numerical vector of size the number of columns of y.
- if y is NULL (pre-calculation step) : a list containing pre-calculated data.

Difference with varest from package sampling

varDT differs from sampling::varest in several ways:

- The formula implemented in varDT is more general and encompasses balanced sampling.
- Even in its reduced form (without balancing variables), the formula implemented in varDT slightly differs from the one implemented in sampling::varest. Caron (1998, pp. 178-179) compares the two estimators (sampling::varest implements V_2, varDT implements V_1).

- varDT introduces several optimizations:
  - matrixwise operations allow to estimate variance on several interest variables at once
  - Matrix::TsparseMatrix capability and the native integration of stratification yield significant performance gains.
  - the ability to pre-calculate some time-consuming operations speeds up the estimation at execution time.
- varDT does not natively implements the calibration estimator (i.e. the sampling variance estimator that takes into account the effect of calibration). In the context of the gustave package, res_cal should be called before varDT in order to achieve the same result.

Author(s)

Martin Chevalier

References

Caron N. (1998), "Le logiciel Poulpe : aspects méthodologiques", *Actes des Journées de méthodologie statistique* http://jms-insee.fr/jms1998s03_1/ Deville, J.-C. (1993), *Estimation de la variance pour les enquêtes en deux phases*, Manuscript, INSEE, Paris.

Deville, J.-C., Tillé, Y. (2005), "Variance approximation under balanced sampling", *Journal of Statistical Planning and Inference*, 128, issue 2 569-591

Rao, J.N.K (1975), "Unbiased variance estimation for multistage designs", *Sankhya*, C n°37

See Also

res_cal

Examples

```
library(sampling)
set.seed(1)

# Simple random sampling case
N <- 1000
n <- 100
y <- rnorm(N)[as.logical(srswor(n, N))]
pik <- rep(n/N, n)
varDT(y, pik)
sampling::varest(y, pik = pik)
N^2 * (1 - n/N) * var(y) / n

# Unequal probability sampling case
N <- 1000
n <- 100
pik <- runif(N)
s <- as.logical(UPsystematic(pik))
y <- rnorm(N)[s]
pik <- pik[s]
varDT(y, pik)
varest(y, pik = pik)
```

```
# The small difference is expected (see Details).

# Balanced sampling case
N <- 1000
n <- 100
pik <- runif(N)
x <- matrix(rnorm(N*3), ncol = 3)
s <- as.logical(samplecube(x, pik))
y <- rnorm(N)[s]
pik <- pik[s]
x <- x[s, ]
varDT(y, pik, x)

# Balanced sampling case (variable of interest
# among the balancing variables)
N <- 1000
n <- 100
pik <- runif(N)
y <- rnorm(N)
x <- cbind(matrix(rnorm(N*3), ncol = 3), y)
s <- as.logical(samplecube(x, pik))
y <- y[s]
pik <- pik[s]
x <- x[s, ]
varDT(y, pik, x)
# As expected, the total of the variable of interest is perfectly estimated.

# strata argument
n <- 100
H <- 2
pik <- runif(n)
y <- rnorm(n)
strata <- letters[sample.int(H, n, replace = TRUE)]
all.equal(
 varDT(y, pik, strata = strata),
 varDT(y[strata == "a"], pik[strata == "a"]) + varDT(y[strata == "b"], pik[strata == "b"])
)

# precalc argument
n <- 1000
H <- 50
pik <- runif(n)
y <- rnorm(n)
strata <- sample.int(H, n, replace = TRUE)
precalc <- varDT(y = NULL, pik, strata = strata)
identical(
 varDT(y, precalc = precalc),
 varDT(y, pik, strata = strata)
)
```

varSYG *Sen-Yates-Grundy variance estimator*

Description

varSYG computes the Sen-Yates-Grundy variance estimator.

Usage

```
varSYG(y = NULL, pikl, precalc = NULL)
```

Arguments

y	A (sparse) numerical matrix of the variable(s) whose variance of their total is to be estimated.
pikl	A numerical matrix of second-order inclusion probabilities.
precalc	A list of pre-calculated results (analogous to the one used by varDT).

Details

varSYG aims at being an efficient implementation of the Sen-Yates-Grundy variance estimator for sampling designs with fixed sample size. It should be especially useful when several variance estimations are to be conducted, as it relies on (sparse) matrix linear algebra.

Moreover, in order to be consistent with varDT, varSYG has a precalc argument allowing for the re-use of intermediary results calculated once and for all in a pre-calculation step (see varDT for details).

Value

- if y is not NULL (calculation step) : a numerical vector of size the number of columns of y.
- if y is NULL (pre-calculation step) : a list containing pre-calculated data (analogous to the one used by varDT).

Difference with varHT from package sampling

varSYG differs from sampling::varHT in several ways:

- The formula implemented in varSYG is solely the Sen-Yates-Grundy estimator, which is the one calculated by varHT when method = 2.
- varSYG introduces several optimizations:
  - matrixwise operations allow to estimate variance on several interest variables at once
  - Matrix::TsparseMatrix capability yields significant performance gains.

Author(s)

Martin Chevalier

Examples

```
library(sampling)
set.seed(1)

# Simple random sampling case
N <- 1000
n <- 100
y <- rnorm(N)[as.logical(srswor(n, N))]
pikl <- matrix(rep((n*(n-1))/(N*(N-1)), n*n), nrow = n)
diag(pikl) <- rep(n/N, n)
varSYG(y, pikl)
sampling::varHT(y = y, pikl = pikl, method = 2)
```

---

var_pois *Variance estimator for a Poisson sampling design*

Description

`var_pois` estimates the variance of the estimator of a total for a Poisson sampling design.

Usage

```
var_pois(y, pik, w = rep(1, length(pik)))
```

Arguments

`y`	A (sparse) numerical matrix of the variable(s) whose variance of their total is to be estimated.
`pik`	A numerical vector of first-order inclusion probabilities.
`w`	An optional numerical vector of row weights (see Details).

Details

`w` is a row weight used at the final summation step. It is useful when `var_pois` is used on the second stage of a two-stage sampling design applying the Rao (1975) formula.

Value

The estimated variances as a numerical vector of size the number of columns of `y`.

Author(s)

Martin Chevalier

References

Rao, J.N.K (1975), "Unbiased variance estimation for multistage designs", *Sankhya*, C n°37

Index