Package 'svyweight'

May 3, 2022

Title Quick and Flexible Survey Weighting

Version 0.1.0

Description Quickly and flexibly calculates weights for survey data, in order to correct for survey non-response or other sampling issues. Uses rake weighting, a common technique also know as rim weighting or iterative proportional fitting. This technique allows for weighting on multiple variables, even when the interlocked distribution of the two variables is not known. Interacts with Thomas Lumley's 'survey' package, as described in Lumley, Thomas (2011, ISBN:978-1-118-21093-2). Adds additional functionality, more adaptable syntax, and error-checking to the base weighting functionality in 'survey.'

License GPL-3

Encoding UTF-8

LazyData true

Imports survey, gdata, stats

Depends R (>= 3.5.00)

Suggests dplyr, srvyr, testthat, mice

RoxygenNote 7.1.2

NeedsCompilation no

Author Ben Mainwaring [aut, cre]

Maintainer Ben Mainwaring <mainwaringb@gmail.com>

Repository CRAN

Date/Publication 2022-05-03 10:00:02 UTC

R topics documented:

`as.w8margin` *Weight Margin Objects*

Description

Creates an object of class `w8margin`. Represents the desired target distribution of a categorical variable, after weighting (as a *counts*, not percentage). w8margin objects are in the format required by the 'survey' package's `survey::rake()`and `survey::postStratify()`, and are intended mostly for use with these functions. Methods exist for numeric vectors, matrices, and data frames (see details).

Usage

```
as.w8margin(
  target,
  varname,
  levels = NULL,
  samplesize = NULL,
  na.allow = FALSE,
  rebase.tol = 0.01,
  ...
)

## S3 method for class 'data.frame'
as.w8margin(
  target,
  varname,
  levels = NULL,
  samplesize = NULL,
  na.allow = FALSE,
  rebase.tol = 0.01,
  ...
)

## S3 method for class 'numeric'
as.w8margin(
  target,
  varname,
  levels = NULL,
  samplesize = NULL,
  na.allow = FALSE,
  rebase.tol = 0.01,
  ...
)

## S3 method for class 'matrix'
```

```
as.w8margin(
  target,
  varname,
  levels = NULL,
  samplesize = NULL,
  na.allow = FALSE,
  rebase.tol = 0.01,
  byrow = TRUE,
  ...
)
```

Arguments

`target`	Numbers specifying the desired target distribution of a categorical variable, after rake weighting. Can be a numeric vector, numeric matrix, or data frame with one (and only one) numeric column. Unless `levels` is specified, vectors and matrices must be named, and data frames must have a character or factor column specifying names. See details.
`varname`	Character vector specifying the name of the observed variable that the `w8margin` object should match. Can take a NULL value for data frames, in which case the original column name is used.
`levels`	Optional character vector, specifying which numeric elements of `target` match with each factor level in the observed data. Overrides names specified in `target`.
`samplesize`	Numeric with the desired target sample size for the w8margin object. Defaults to the sum of `target`.
`na.allow`	Logical specifying whether NA values should be allowed in w8margin objects. If TRUE, w8margin objects must be imputed (such as with `impute_w8margin()` before they can be used for weighting.
`rebase.tol`	Numeric between 0 and 1. If targets are rebased, and the rebased sample sizes differs from the original sample size by more than this percentage, generates a warning.
`...`	Other method-specific arguments, currently not used
`byrow`	Logical, specifying how matrix targets should be converted to vectors. If FALSE, the elements within columns will be adjacent in the resulting w8margin object, otherwise elements within rows will be adjacent.

Details

w8margin objects are inputs to the `survey::rake()`and `survey::postStratify()`. These functions require a specific, highly-structured input format. For flexibility, `as.w8margin()` can be used to convert a variety of common inputs into the format needed by these functions.

`as.w8margin()` has methods for numeric vectors, numeric matrices, and data frames. Each method has multiple ways of determining how to match numeric elements of `target` with factor levels in the observed data. For numeric vector and matrix inputs, the default is to match based on the name of each element (for vectors) or the interaction of row and column names of each element (for matrices). These names can be overridden by specifying the `levels` parameter.

Data frame inputs must have either one or two columns. Two-column data frames must have one numeric column and one character column. The numeric column specifies the target distribution, while the character column specifies how to match numeric elements with factor levels in the observed data. If `varname` is NULL, a default value will be taken from the name of the non-numeric column.

One-column data frames must have a numeric column. Row names are converted to a character column in order to match numeric elements with factor levels in the observed data. One-column data frames must specify a `varname` parameter, and (unless `levels` is specified) must have non-default row names. The `levels` parameter can be used with both one- and two-column data frames.

Technically, `w8margin` objects are data frames with two columns. The first column specifies levels in the observed factor variable, and the *name* of the first column indicates the name of the observed factor variable. The second column is named "Freq" and indicates the desired post-raking frequency of each category (as a *count* rather than percentage). The structure is designed for compatibility with the 'survey' package. Because frequency is specified as a count, `rakesvy()` and `rakew8()` re-call `as.w8margin()` whenever weighting a data set to a new observed sample size. Weight margins must be manually re-calculated for new sample sizes when using `survey::postStratify()` or `rake`.

Value

An object of class w8margin, with specified variable name and sample size.

Examples

```
# Convert vector of percentages to w8margin
turnout2013_w8margin <- as.w8margin(
    c(voted = .715, `did not vote` = .285, ineligible = NA),
    varname = "turnout2013",
    na.allow = TRUE,
    samplesize = 1500)

# Convert matrix of percentages to w8margin
gender_educ_mat <- matrix(
    c(.15, .17, .17, .01, .19, .16, .14, .01),
    nrow = 2,
    byrow = TRUE,
    dimnames = list(c("Male", "Female"), c("Low", "Medium", "High", NA)))
gender_educ_w8margin <- as.w8margin(gender_educ_mat,
    varname = "gender_educ", samplesize = 2179)

# Convert data frame of counts to w8margin
# Keep default values for samplesize and varname
region_df <- data.frame(
    eastwest = c("east", "west"), Freq = c(425, 1754))
region_w8margin <- as.w8margin(region_df,
                               levels = c("East Germany", "West Germany"),
                               varname = NULL)

# Calculate rake weights using w8margin objects (without NAs)
require(survey)
gles17_dweighted <- svydesign(ids = gles17$vpoint, weights = gles17$dweight,
```

```
    strata = gles17$eastwest, data = gles17, nest = TRUE)
rake(design = gles17_dweighted,
     sample.margins = list(~gender_educ, ~eastwest),
     population.margins = list(gender_educ_w8margin, region_w8margin))
```

`eff_n` *Effective Sample Size and Weighting Efficiency*

Description

Computes Kish's effective sample size or weighting efficiency for a `survey.design` object.

Usage

```
eff_n(design)

weight_eff(design)
```

Arguments

`design`	An `svydesign` object, presumably with design or post-stratification weights.

Details

Kish's effective sample size is a frequently-used, general metric to indicate how much uncertainty and error increase due to weighting. Effective sample size is calculated as `sum(weights(design))^2 / sum(weights(design)^2)`. Weighting efficiency is `eff_n(design) / sum(weights(design))`.

While weighting efficency and effective sample size are frequently use, they are less valid than the standard errors produced by `survey::svymean()` and related functions from the survey package. In particular, they ignore clustering and stratification in sample designs, and covariance between weighting variables and outcome variables. As such, these metrics should be used with caution

Value

A numeric value, indicating effective sample size (for `eff_n()`) or weighting efficiency (for `weight_eff()`)

References

Kish, Leslie. 1965. *Survey Sampling* New York: Wiley.

Examples

```
gles17_weighted <- rakesvy(design = gles17,
    gender ~ c("Male" = .495, "Female" = .505),
    eastwest ~ c("East Germany" = .195, "West Germany" = .805)
)

eff_n(gles17_weighted)
weight_eff(gles17_weighted)
```

---

gles17 *Partial Data from the 2017 German Election Survey*

Description

Partial data from the pre-election 2017 wave of the German Longitudinal Election Study (GLES). Includes variables for vote in the 2013 German federal election to the Bundestag (lower house of parliament) - specifically the 'second vote'. Also includes other demographics that might be used for weighting, such as gender, birth year, and state. Each row in the dataset is a unique respondent who completed the survey.

Usage

```
gles17
```

Format

A data frame with 2179 rows and 11 columns:

gender gender

educ educational attainment, based on kind of secondary school from which respondent graduated

gender_educ interaction of gender and education attainment

birthyear four-digit birth year

votingage eligibility to vote in the (upcoming) 2017 German federal elections

agecat approximate age category in 2017, estimated from birth year

state state the respondent lives in

eastwest whether the respondent lives in East or West Germany

vote2013 respondent's reported vote in 2013 (specifically the 'second vote')

turnout2013 whether the respondent actually voted in 2013

votecurrent party the respondent plans to vote for in the upcoming (2017) election

intnum unique code for the interviewer who conducted an interview

vpoint unique code anonymously identifying census block where an interview was conducted

hhsize number of people in the household

dweight nationally-representative design weight supplied by the GLES study authors ...

Source

GLES data and documentation is available at https://gles-en.eu/download-data/vor-und-nachwahlquerschnitt-20
Data is taken from the pre-election wave, file ZA6800, for a limited number of variables. Note that most documentation is available in English, but some may be in German only.

`impute_w8margin` *Impute NAs in w8margin Object*

Description

Imputes NA values in a weight target (in `w8margin` form), based on the observed distribution of the variable in a dataset.

Usage

```
impute_w8margin(w8margin, observed, weights = NULL, rebase = TRUE)
```

Arguments

`w8margin`	w8margin object, with NA values that should be imputed based on observed data.
`observed`	factor or character vector, containing observed data used for imputing targets.
`weights`	numeric vector of weights, the same length as `observed`, to be used when computing the distribution of the observed variable. NULL is equivalent to a vector where all elements are 1, and indicates the data is unweighted.
`rebase`	logical, indicating whether non-NA weight targets should be adjusted so that the total target sample size is unchanged (`rebase = TRUE`), or whether non-NA weight targets should remain the same and total target sample size increases.

Details

Any NA target frequencies in `w8margin` are imputed using the percentage distribution in `observed`, from `svytable(~observed, Ntotal = 1, ...)`. The percentage is multiplied by the desired target sample size. For example, if has a target of NA and a desired total sample of 1500, and the observed frequency of the weighting variable is 0%, the imputed target will be (10% * 1500). If a `weights` argument is provided, then weighted percentage distributions are used; this may be useful when design weights are present, or when first raking on variables with complete targets.

If `rebase == TRUE` (the default), targets for non-NA categories are scaled down so that the total target frequency (`sum(w8margin$Freq, na.rm = TRUE)`) remains constant, after imputing new category targets. If `rebase == FALSE`, targets for non-NA categories remain constant, and the total target frequency will increase.

There is an important theoretical distinction between missing *targets* for conceptually valid categories, versus missing observed data due to non-response or refusal. It is only conceptually appropriate to impute targets if the targets themselves are missing. When handling missing observed data, multiple imputation techniques (such as `mice::mice()`) will often produce better results, except when missingness is closely related to weighting variable (technically referred to as "missing not at random").

Value

A w8margin object, where NA target frequencies have been replaced using the observed distribution of the weighting variable.

Examples

```
turnout_w8margin <- as.w8margin(
    c(voted = .715, `did not vote` = .285, ineligible = NA),
    varname = "turnout2013",
    na.allow = TRUE,
    samplesize = 1500)
impute_w8margin(turnout_w8margin, observed = gles17$turnout2013)
```

---

rakesvy *Flexibly Calculate Rake Weights*

Description

Calculate rake weights on a data frame, or on a `survey.design` object from `survey::svydesign()`. Targets may be counts or percentages, in vector, matrix, data frame, or w8margin form. Before weighting, targets are converted to w8margins, checked for validity, and matched to variables in observed data, `rakesvy` returns a weighted `svydesign` object, while `rakew8` returns a vector of weights.

Usage

```
rakesvy(
  design,
  ...,
  samplesize = "from.data",
  match.levels.by = "name",
  na.targets = "fail",
  rebase.tol = 0.01,
  control = list(maxit = 10, epsilon = 1, verbose = FALSE)
)

rakew8(
  design,
  ...,
  samplesize = "from.data",
  match.levels.by = "name",
  na.targets = "fail",
  rebase.tol = 0.01,
  control = list(maxit = 10, epsilon = 1, verbose = FALSE)
)
```

Arguments

design	A survey.design object from survey::svydesign(), or a data frame that can be coerced to one. When a data frame is coerced, the coercion assuming no clustering or design weighting.
...	Formulas specifying weight targets, with an object that can be coerced to class w8margin (see as.w8margin()) on the right-hand side, and (optionally) a matching variable or transformation of it on the left-hand side. Objects that can be coerced to w8margin include named numeric vectors and matrices, and data frames in the format accepted by rake.
samplesize	Either a number specifying the desired post-raking sample size, or a character string "from.data" or "from.targets" specifying how to calculate the desired sample size (see details).
match.levels.by	A character string that specifies how to match levels in the target with the observed data, either "name" (the default) or "order" (see details).
na.targets	A characters string that specifies how to handle NAs in *targets*, either "fail" (the default) or "observed" (see details).
rebase.tol	Numeric between 0 and 1. If targets are rebased, and the rebased sample sizes differs from the original sample size by more than this percentage, generates a warning.
control	Parameters passed to the control argument of survey::rake(), to control the details of convergence in weighting.

Details

rakesvy and rakew8 wrangles observed data and targets into compatible formats, before using survey::rake() to make underlying weighting calculations. The function matches weight targets to observed variables, cleans both targets and observed variables, and then checks the validity of weight targets (partially by calling w8margin_matched()) before raking. It also allows a weight target of zero, and assigns an automatic weight of zero to cases on this target level.

Weight target levels can be matched with observed variable levels in two ways, specified via the match.levels.by parameter. "name" (the default) matches based on name, disregarding order (so a "male" level in the weight target will be matched with a "male" level in the observed data). "order" matches based on order, disregarding name (so the first level or row of the target will match with the first level of the observed factor variable).

By default, with parameter na.targets = "fail"), an NA in weight targets will cause an error. With na.targets = "observed", rakesvy() and rakew8() will instead compute a target that matches the observed data. The category with an NA target will therefore have a similar incidence rate in the pre-raking and post-raking dataset. This is accomplished by calling impute_w8margin() before raking; see the impute_w8margin documentation for more details. Note that NAs in *observed* data (as opposed to targets) will always cause failure, and are not affected by this parameter.

The desired sample size (in other words, the desired sum of weights after raking) is specified via the samplesize parameter. This can be a numeric value. Alternatively, "from.data" specifies that the observed sample size before weighting (taken from sum(weights(design)) if applicable, or nrow() if not); "from.targets" specifies that the total sample sizes in target objects should be followed, and should only be used if all targets specify the same sample size.

Value

rakesvy() returns a survey.design object with rake weights applied. Any changes made to the variables in design in order to call rake, such as dropping empty factor levels, are temporary and *not* returned in the output object.

rakew8() returns a vector of weights. This avoids creating duplicated survey.design objects, which can be useful when calculating multiple sets of weights for the same data.

Examples

```
# Computing only rake weights
# EG, for a survey conducted with simple random sampling
gles17$simple_weight <- rakew8(design = gles17,
    gender ~ c("Male" = .495, "Female" = .505),
    eastwest ~ c("East Germany" = .195, "West Germany" = .805)
)

# Specifying a recode of variable in observed dataset
require(dplyr)
gles17_raked <- rakesvy(design = gles17,
    gender ~ c("Male" = .495, "Female" = .505),
    dplyr::recode(agecat, `<=29` = "<=39", `30-39` = "<=39") ~
        c("<=39" = .31, "40-49" = .15, "50-59" = .19, "60-69" = .15, ">=70" = .21)
)

# Computing rake weights after design weights
# EG, for a survey with complex sampling design
require(survey)
gles17_dweighted <- svydesign(ids = gles17$vpoint, weights = gles17$dweight,
    strata = gles17$eastwest, data = gles17, nest = TRUE)
gles17_raked <- rakesvy(design = gles17_dweighted,
    gender ~ c("Male" = .495, "Female" = .505),
    agecat ~ c("<=29" = .16, "30-39" = .15, "40-49" = .15,
        "50-59" = .19, "60-69" = .15, ">=70" = .21)
)

# With unnamed target levels, using match.levels.by = "order"
rakew8(design = gles17,
    gender ~ c(.495, .505),
    eastwest ~ c(.195, .805),
    match.levels.by = "order"
)
```

---

svyweight *svyweight: Quick and Flexible Rake Weighting*

Description

svyweight is a package for quickly and flexibly calculating rake weights (also know as rim weights). It is designed to interact with survey.design objects generated via survey::svydesign(), and other to otherwise build on functionalities from Thomas Lumley's 'survey' package.

Rake weighting concepts

Post-stratification weights are commonly used in survey research to ensure that sample is representative of the population it is drawn from, in cases where some people selected for inclusion in a sample might decline to participate. To calculate post-stratification weights, observed categorical variables in a survey dataset (usually demographic variables) must be matched with "targets" (typically known population demographics from census data). Survey respondents from underrepresented categories are upweighted, while respondents from overrepresented categories are downweighted.

svyweight implements "rake" or "rim" weighting (sometimes known as iterative proportional fitting). This is a widely-used method for simultaneously calculating weights on multiple variables, when no join distribution for these variables is known. For example, population data on past vote (from election results) and age (from the census) are generally known. However, as the joint distribution of past vote and age is *not* generally known, a technique such as rake weighting must be used to apply weights on both variables simultaneously.

Package features

The core function in svyweight is `rakesvy()` (and the related `rakew8()`. This takes calculates post-stratification weights given A) data frame or a `survey.design` object generated by `svydesign()`, and B) a set of weighting targets The command is designed to make weighting as simple as possible, with the following features:

- Weighting to either counts or percentage targets
- Allowing specification of targets as vectors, matrices, or data frames
- Accepting targets of 0 (equivalent to dropping cases from analysis)
- Allowing targets to be quickly rebased a specified sample size
- Flexibly matching targets to the correct variables in a dataset
- Dynamically specifying weight targets based on recodes of variables in observed data

The package does this in part by introducing the `w8margin` object class. A w8margin is a desired raw *count* of categories for a variable, in the format required for actually computing weights. However, this format is somewhat cumbersome to specify manually. The package includes methods for converting named vectors, matrices, and data frames to w8margin object; `[rakesvy()]` and `rakew8()` call these methods automatically.

At present, the core weighting calculations are actually performed via the 'survey' package's `survey::rake()` function. This might change with future releases, although the basic approach to iterative weighting is not expected to change.

The package is under development. Contributions to the package, or suggestions for additional features, are gratefully accepted via email or GitHub.

Author(s)

Ben Mainwaring (<mainwaringb@gmail.com>, https://www.linkedin.com/in/mainwaringb)

References

Lumley, Thomas. 2011. *Complex Surveys: A Guide to Analysis Using R*. New York: Wiley.

See Also

Package GitHub repository: https://github.com/mainwaringb/svyweight

---

`w8margin_matched` *Check if w8margin Matches Observed Data*

Description

Checks whether specified `w8margin` object and variable in observed data are compatible, and are expected to produce valid call to `rake`. Returns a logical true/false, and generates warning messages to specify likely issues. Intended to help quickly diagnose incompatibilities between w8margins and observed data.

Usage

```
w8margin_matched(w8margin, observed, refactor = FALSE,
  na.targets.allow = FALSE, zero.targets.allow = FALSE)
```

Arguments

`w8margin`	w8margin object, or other object type that can be coerced to w8margin with a temporary variable name.
`observed`	factor vector (or, if `refactor = TRUE`, a vector that can be coerced to factor).
`refactor`	logical, specifying whether to factor observed variable before checking match.
`na.targets.allow`	logical, indicating whether NA values in target should produce error (`FALSE`, the default) or be allowed. NA values are never allowed in observed data.
`zero.targets.allow`	logical, indicating whether zero values in target should produce error (`FALSE`, the default) or be allowed.

Details

With default parameters (`na.targets.allow = FALSE`, `zero.targets.allow = FALSE`, and `refactor = FALSE`), the function checks whether a `w8margin` object is in the strict format required by `rake`; this format will also be accepted by `rakesvy()` and `rakew8()`. Changing the default parameters relaxes some checks. With the parameters altered, the function will only assess whether `w8margin` objects are usable by `rakesvy()` and `rakew8()`, which accept a more flexible range of target formats.

It should not generally be necessary to call `w8margin_matched()` manually when using `rakesvy()` and `rakew8()` to compute weights. However, may be useful to call directly, when manually calling underlying weighting functions from the `survey` package, or for diagnostic purposes.

Value

A logical, indicating whether w8margin is compatible with observed.

Examples

```
gender_w8margin <- as.w8margin(
    c(Male = .49, Female = .51),
    varname = "gender",
    samplesize = 2179)
# Returns TRUE
w8margin_matched(gender_w8margin, gles17$gender)

gender_w8margin_alt <- as.w8margin(
    c(man = .49, woman = .51),
    varname = "gender",
    samplesize = 2179)
# Returns FALSE - level names in gles17$gender do not match level names in gender_w8margin_alt
w8margin_matched(gender_w8margin_alt, gles17$gender)

agecat_50plus_w8margin <- as.w8margin(
    c("50-59" = .35, "60-69" = .27, ">=70" = .38),
    varname = "educ",
    samplesize = 2179
)
gles17_50plus <- gles17[gles17$agecat %in% c("50-59", "60-69", ">=70"),]
# Returns FALSE - gles17$agecat has empty factor levels for <=29, 30-39, 40-49
w8margin_matched(agecat_50plus_w8margin, gles17_50plus$agecat)
# Returns TRUE - gles17$agecat is refactored to drop empty levels
w8margin_matched(agecat_50plus_w8margin, gles17_50plus$agecat, refactor = TRUE)
```

Index