# Package 'svrep'

March 30, 2022

**Type** Package

**Title** Tools for Creating, Updating, and Analyzing Survey Replicate Weights

**Version** 0.1.0

**Author** Ben Schneider

**Maintainer** Ben Schneider `<benjamin.julius.schneider@gmail.com>`

**Description** Provides tools for creating and working with survey replicate weights,
extending functionality of the 'survey' package from Lumley (2004) <doi:10.18637/jss.v009.i08>.
Methods are provided for applying nonresponse adjustments to
both full-sample and replicate weights as suggested by
Rust and Rao (1996) <doi:10.1177/096228029600500305> in order to account for
the impact of these adjustments on sampling variances.
Diagnostic functions are included to compare weights and weighted estimates
from different sets of replicate weights.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Imports** stats, survey (>= 4.1), utils

**Suggests** covr, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-03-30 17:50:02 UTC

## R topics documented:

## redistribute_weights *Redistribute weight from one group to another*

### Description

Redistributes weight from one group to another: for example, from non-respondents to respondents. Redistribution is conducted for the full-sample weights as well as each set of replicate weights. This can be done separately for each combination of a set of grouping variables, for example to implement a nonresponse weighting class adjustment.

### Usage

```
redistribute_weights(design, reduce_if, increase_if, by)
```

### Arguments

| | |
|---|---|
| `design` | A survey design object, created with either the `survey` or `srvyr` packages. |
| `reduce_if` | An expression indicating which cases should have their weights set to zero. Must evaluate to a logical vector with only values of TRUE or FALSE. |
| `increase_if` | An expression indicating which cases should have their weights increased. Must evaluate to a logical vector with only values of TRUE or FALSE. |
| `by` | (Optional) A character vector with the names of variables used to group the redistribution of weights. For example, if the data include variables named `"stratum"` and `"wt_class"`, one could specify by = `c("stratum","wt_class")`. |

### Value

The survey design object, but with updated full-sample weights and updated replicate weights. The resulting survey design object always has its value of `combined.weights` set to `TRUE`.

### Examples

```
# Load example data
suppressPackageStartupMessages(library(survey))
data(api)

dclus1 <- svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
dclus1$variables$response_status <- sample(x = c("Respondent", "Nonrespondent",
                                                 "Ineligible", "Unknown eligibility"),
                                           size = nrow(dclus1),
                                           replace = TRUE)
rep_design <- as.svrepdesign(dclus1)

# Adjust weights for cases with unknown eligibility
ue_adjusted_design <- redistribute_weights(
    design = rep_design,
    reduce_if = response_status %in% c("Unknown eligibility"),
    increase_if = !response_status %in% c("Unknown eligibility"),
    by = c("stype")
)

# Adjust weights for nonresponse
nr_adjusted_design <- redistribute_weights(
    design = ue_adjusted_design,
    reduce_if = response_status %in% c("Nonrespondent"),
    increase_if = response_status == "Respondent",
    by = c("stype")
)
```

---

`stack_replicate_designs` *Stack replicate designs, combining data and weights into a single object*

---

## Description

Stack replicate designs: combine rows of data, rows of replicate weights, and the respective full-sample weights. This can be useful when comparing estimates before and after a set of adjustments made to the weights. Another more delicate application is when combining sets of replicate weights from multiple years of data for a survey, although this must be done carefully based on guidance from a data provider.

## Usage

```
stack_replicate_designs(..., .id = "Design_Name")
```

## Arguments

| | |
|---|---|
| `...` | Replicate-weights survey design objects to combine. These can be supplied in one of two ways.<br>• Option 1 - A series of design objects, for example `'adjusted' = adjusted_design, 'orig' = orig_design`.<br>• Option 2 - A list object containing design objects, for example `list('nr' = nr_adjusted_design, 'ue' = ue_adjusted_design)`.<br>All objects must have the same specifications for `type`, `rho`, `mse`, `scales`, and `rscales`. |
| `.id` | A single character value, which becomes the name of a new column of identifiers created in the output data to link each row to the design from which it was taken. The labels used for the identifiers are taken from named arguments. |

## Value

A replicate-weights survey design object, with class `svyrep.design` and `svyrep.stacked`. The resulting survey design object always has its value of `combined.weights` set to `TRUE`.

## Examples

```
# Load example data, creating a replicate design object
suppressPackageStartupMessages(library(survey))
data(api)

dclus1 <- svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
dclus1$variables$response_status <- sample(x = c("Respondent", "Nonrespondent",
                                                 "Ineligible", "Unknown eligibility"),
                                           size = nrow(dclus1),
                                           replace = TRUE)
orig_rep_design <- as.svrepdesign(dclus1)

# Adjust weights for cases with unknown eligibility
ue_adjusted_design <- redistribute_weights(
    design = orig_rep_design,
    reduce_if = response_status %in% c("Unknown eligibility"),
    increase_if = !response_status %in% c("Unknown eligibility"),
    by = c("stype")
)

# Adjust weights for nonresponse
nr_adjusted_design <- redistribute_weights(
    design = ue_adjusted_design,
    reduce_if = response_status %in% c("Nonrespondent"),
    increase_if = response_status == "Respondent",
    by = c("stype")
)

# Stack the three designs, using any of the following syntax options
stacked_design <- stack_replicate_designs(orig_rep_design, ue_adjusted_design, nr_adjusted_design,
                                          .id = "which_design")
stacked_design <- stack_replicate_designs('original' = orig_rep_design,
                                     'unknown eligibility adjusted' = ue_adjusted_design,
                                          'nonresponse adjusted' = nr_adjusted_design,
                                          .id = "which_design")
list_of_designs <- list('original' = orig_rep_design,
                        'unknown eligibility adjusted' = ue_adjusted_design,
                        'nonresponse adjusted' = nr_adjusted_design)
stacked_design <- stack_replicate_designs(list_of_designs, .id = "which_design")
```

---

## summarize_rep_weights *Summarize the replicate weights*

---

### Description

Summarize the replicate weights of a design

### Usage

```
summarize_rep_weights(rep_design, type = "both")
```

## Arguments

| | |
|---|---|
| `rep_design` | A replicate design object, created with either the `survey` or `srvyr` packages. |
| `type` | Default is `"both"`. Use `type = "overall"`, for an overall summary of the replicate weights. Use `type = "specific"` for a summary of each column of replicate weights, with each column of replicate weights summarized in a given row of the summary.<br><br>Use `type = "both"` for a list containing both summaries, with the list containing the names `"overall"` and `"both"`. |

## Value

If `type = "both"` (the default), the result is a list of data frames with names `"overall"` and `"specific"`. If `type = "overall"`, the result is a data frame providing an overall summary of the replicate weights.

The contents of the `"overall"` summary are the following:

- "nrows": Number of rows for the weights
- "ncols": Number of columns of replicate weights
- "degf_svy_pkg": The degrees of freedom according to the survey package in R
- "rank": The matrix rank as determined by a QR decomposition
- "avg_wgt_sum": The average column sum
- "sd_wgt_sums": The standard deviation of the column sums
- "min_rep_wgt": The minimum value of any replicate weight
- "max_rep_wgt": The maximum value of any replicate weight

If `type = "specific"`, the result is a data frame providing a summary of each column of replicate weights, with each column of replicate weights described in a given row of the data frame. The contents of the `"specific"` summary are the following:

- "Rep_Column": The name of a given column of replicate weights. If columns are unnamed, the column number is used instead
- "N": The number of entries
- "N_NONZERO": The number of nonzero entries
- "SUM": The sum of the weights
- "MEAN": The average of the weights
- "CV": The coefficient of variation of the weights (standard deviation divided by mean)
- "MIN": The minimum weight
- "MAX": The maximum weight

## Examples

```
# Load example data
suppressPackageStartupMessages(library(survey))
data(api)

dclus1 <- svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
dclus1$variables$response_status <- sample(x = c("Respondent", "Nonrespondent",
                                                "Ineligible", "Unknown eligibility"),
                                          size = nrow(dclus1),
                                          replace = TRUE)
rep_design <- as.svrepdesign(dclus1)

# Adjust weights for cases with unknown eligibility
ue_adjusted_design <- redistribute_weights(
    design = rep_design,
    reduce_if = response_status %in% c("Unknown eligibility"),
    increase_if = !response_status %in% c("Unknown eligibility"),
    by = c("stype")
)

# Summarize replicate weights

summarize_rep_weights(rep_design, type = "both")

# Compare replicate weights

rep_wt_summaries <- lapply(list('original' = rep_design,
                                'adjusted' = ue_adjusted_design),
                           summarize_rep_weights,
                           type = "overall")
print(rep_wt_summaries)
```

---

`svyby_repwts` *Compare survey statistics calculated separately from different sets of replicate weights*

---

## Description

A modified version of the `svyby()` function from the `survey` package. Whereas `svyby()` calculates statistics separately for each subset formed by a specified grouping variable, `svyby_repwts()` calculates statistics separately for each replicate design, in addition to any additional user-specified grouping variables.

## Usage

```
svyby_repwts(
  rep_designs,
  formula,
  by,
  FUN,
  ...,
  deff = FALSE,
  keep.var = TRUE,
  keep.names = TRUE,
  verbose = FALSE,
  vartype = c("se", "ci", "ci", "cv", "cvpct", "var"),
  drop.empty.groups = TRUE,
  return.replicates = FALSE,
  na.rm.by = FALSE,
  na.rm.all = FALSE,
  multicore = getOption("survey.multicore")
)
```

## Arguments

| | |
|---|---|
| `rep_designs` | The replicate-weights survey designs to be compared. Supplied either as: <br>• A named list of replicate-weights survey design objects, for example `list('nr' = nr_adjusted_design, 'ue' = ue_adjusted_design)`. <br>• A 'stacked' replicate-weights survey design object created by `stack_replicate_designs()`. <br>The designs must all have the same number of columns of replicate weights, of the same type (bootstrap, JKn, etc.) |
| `formula` | A formula specifying the variables to pass to `FUN` |
| `by` | A formula specifying factors that define subsets |
| `FUN` | A function taking a formula and survey design object as its first two arguments. Usually a function from the `survey` package, such as `svytotal` or `svymean`. |
| `...` | Other arguments to `FUN` |
| `deff` | A value of `TRUE` or `FALSE`, indicating whether design effects should be estimated if possible. |
| `keep.var` | A value of `TRUE` or `FALSE`. If `FUN` returns a `svystat` object, indicates whether to extract standard errors from it. |
| `keep.names` | Define row names based on the subsets |
| `verbose` | If `TRUE`, print a label for each subset as it is processed. |
| `vartype` | Report variability as one or more of standard error, confidence interval, coefficient of variation, percent coefficient of variation, or variance |
| `drop.empty.groups` | If `FALSE`, report `NA` for empty groups, if `TRUE` drop them from the output |
| `return.replicates` | If `TRUE`, return all the replicates as the "replicates" attribute of the result. This can be useful if you want to produce custom summaries of the estimates from each replicate. |
| `na.rm.by` | If true, omit groups defined by `NA` values of the `by` variables |

| | |
|---|---|
| na.rm.all | If true, check for groups with no non-missing observations for variables defined by formula and treat these groups as empty |
| multicore | Use multicore package to distribute subsets over multiple processors? |

## Value

An object of class "svyby": a data frame showing the grouping factors and results of FUN for each combination of the grouping factors. The first grouping factor always consists of indicators for which replicate design was used for an estimate.

## Examples

```
suppressPackageStartupMessages(library(survey))
data(api)

dclus1 <- svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
dclus1$variables$response_status <- sample(x = c("Respondent", "Nonrespondent",
                                                 "Ineligible", "Unknown eligibility"),
                                           size = nrow(dclus1),
                                           replace = TRUE)
orig_rep_design <- as.svrepdesign(dclus1)

# Adjust weights for cases with unknown eligibility
ue_adjusted_design <- redistribute_weights(
    design = orig_rep_design,
    reduce_if = response_status %in% c("Unknown eligibility"),
    increase_if = !response_status %in% c("Unknown eligibility"),
    by = c("stype")
)

# Adjust weights for nonresponse
nr_adjusted_design <- redistribute_weights(
    design = ue_adjusted_design,
    reduce_if = response_status %in% c("Nonrespondent"),
    increase_if = response_status == "Respondent",
    by = c("stype")
)

# Compare estimates from the three sets of replicate weights

  list_of_designs <- list('original' = orig_rep_design,
                          'unknown eligibility adjusted' = ue_adjusted_design,
                          'nonresponse adjusted' = nr_adjusted_design)

  ##_ First compare overall means for two variables
  means_by_design <- svyby_repwts(formula = ~ api00 + api99,
                                  FUN = svymean,
                                  rep_design = list_of_designs)

  print(means_by_design)

  ##_ Next compare domain means for two variables
  domain_means_by_design <- svyby_repwts(formula = ~ api00 + api99,
                                         by = ~ stype,
                                         FUN = svymean,
                                         rep_design = list_of_designs)

  print(domain_means_by_design)

# Calculate confidence interval for difference between estimates

ests_by_design <- svyby_repwts(rep_designs = list('NR-adjusted' = nr_adjusted_design,
                                                  'Original' = orig_rep_design),
                               FUN = svymean, formula = ~ api00 + api99)

differences_in_estimates <- svycontrast(stat = ests_by_design, contrasts = list(
  'Mean of api00: NR-adjusted vs. Original' = c(1,-1,0,0),
  'Mean of api99: NR-adjusted vs. Original' = c(0,0,1,-1)
))

print(differences_in_estimates)

confint(differences_in_estimates, level = 0.95)
```

# Index