Package 'scf'

October 22, 2025

Title Analyzing the Survey of Consumer Finances

Version 1.0.4

Description Analyze public-use micro data from the Survey of Consumer Finances. Provides tools to download prepared data files, construct replicate-weighted multiply imputed survey designs, compute descriptive statistics and model estimates, and produce plots and tables. Methods follow design-based inference for complex surveys and pooling across multiple imputations. See the package website and the code book for background.

License MIT + file LICENSE

URL https://github.com/jncohen/scf

BugReports https://github.com/jncohen/scf/issues

Depends R (>= 3.6)

Imports ggplot2, haven, httr, rlang, stats, survey, utils

Suggests dplyr, hexbin, kableExtra, knitr, mitools, rmarkdown, markdown, testthat (>= 3.0.0)

VignetteBuilder knitr, markdown, rmarkdown

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Joseph Cohen [aut, cre]

Maintainer Joseph Cohen <joseph.cohen@qc.cuny.edu>

Repository CRAN

Date/Publication 2025-10-22 19:00:02 UTC

Contents

`scf` *Analyzing Survey of Consumer Finances Public-Use Microdata*

Description

This package provides functions to analyze the U.S. Federal Reserve's Survey of Consumer Finances (SCF) public-use microdata. It encapsulates the SCF's multiply-imputed, replicate-weighted structure in a custom object class (`scf_mi_survey`) and supports estimation of population-level statistics, including univariate and bivariate distributions, hypothesis tests, data visualizations, and regression models.

Designed for generalist analysts, the package assumes familiarity with standard statistical methods but not with the complexities of multiply-imputed or survey-weighted data. All functions prioritize transparency, reproducibility, and pedagogical clarity.

Methodological Background

The SCF is one of the most detailed and methodologically rigorous sources of data on U.S. household finances. It is nationally representative, includes an oversample of high-wealth households and households in predominantly Black communities, and provides multiply-imputed estimates for item nonresponse. These features increase the analytical value of the data set but also introduce methodological complexity. Valid inference requires attention to:

- **Survey Weights:** The SCF employs a dual-frame, stratified, and clustered probability sample. Analysts must apply the provided sampling weights to produce population-representative estimates.
- **Replicate Weights:** Each observation is associated with 999 replicate weights, generated using a custom replication method developed by the Federal Reserve. These are used to estimate sampling variance.
- **Multiple Imputation:** The SCF uses multiple imputation to address item nonresponse, providing five implicates per household. Estimates must be pooled across implicates to obtain valid point estimates and standard errors.

The `scf` package provides a structured, user-friendly interface for handling these design complexities, enabling applied researchers and generalist analysts to conduct principled and reproducible analysis of SCF microdata using familiar statistical workflows.

Package Architecture and Workflow

This section recommends a sequence of operations enacted through the package's functions. For an in-depth discussion of the methodological considerations involved in these functions' formulation, see Cohen (2025).

1. **Data Acquisition**: Download the data from Federal Reserve servers to your working directory using `scf_download()`.
2. **Data Loading**: Load the data into R using `scf_load()`. This function returns an `scf_mi_survey` object (described below).
3. **Data Wrangling**: Use `scf_update()` to modify the data, or `scf_subset()` to filter it. These functions return new `scf_mi_survey` objects.
4. **Descriptive Statistics**: Compute univariate and bivariate statistics using functions like `scf_mean()`, `scf_median()`, `scf_percentile()`, `scf_freq()`, `scf_xtab()`, and `scf_corr()`.
5. **Basic Inferential Tests**: Conduct hypothesis tests using `scf_ttest()` for means and `scf_prop_test()` for proportions.
6. **Regression Modeling**: Fit regression models using `scf_ols()` for linear regression, `scf_logit()` for logistic regression, and `scf_glm()` for generalized linear models.
7. **Data Visualization**: Create informative visualizations using `scf_plot_dist()` for distributions, `scf_plot_cbar()` and `scf_plot_bbar()` for categorical data, `scf_plot_smooth()` for smoothers, and `scf_plot_hex()` for hexbin plots.
8. **Diagnostics and Infrastructure**: Use `scf_MIcombine()` to pool results across implicates.

Core Data Object and Its Structure

This suite of functions operate from a custom object class, `scf_mi_survey`, which is created by `scf_design()` via `scf_load()`. Specifically, the object is a structured list containing the elements:

- `mi_design`: A list of five `survey::svrepdesign()` objects (one per implicate)
- `year`: Year of survey
- `n_households`: Estimated number of U.S. households in that year, per data from the Federal Reserve Economic Data (FRED) series TTLHH, accessed 6/17/2025.

Imputed Missing Data

The SCF addresses item nonresponse using multiple imputation (see Kennickell 1998). This procedure generates five completed data sets, each containing distinct but plausible values for the missing entries. The method applies a predictive model to the observed data, simulates variation in both model parameters and residuals, and generates five independent estimates for each missing value. These completed data sets—called *implicates*—reflect both observed relationships and the uncertainty in estimating them. See `scf_MIcombine()` for details.

Mock Data for Testing

A mock SCF dataset (`scf2022_mock_raw.rds`) is bundled in `inst/extdata/` for internal testing purposes. It is a structurally valid `scf_mi_survey` object created by retaining only the first ~200 rows per implicate and only variables used in examples and tests.

This object is intended solely for package development and documentation rendering. It is **not suitable for analytical use or valid statistical inference.**

Theming and Visual Style

All built-in graphics follow a common aesthetic set by `scf_theme()`. Users may modify the default theme by calling `scf_theme()` explicitly within their scripts. See `scf_theme()` documentation for customization options.

Pedagogical Design

The package is designed to support instruction in advanced methods courses on complex survey analysis and missing data. It promotes pedagogical transparency through several features:

- Each implicate's design object is accessible via `scf_mi_survey$mi_design[[i]]`
- Raw implicate-level data can be viewed directly through `scf_mi_survey$mi_design[[i]]$variables`
- Users can execute analyses on individual implicates or combine them using Rubin's Rules
- Key functions implement design-based estimation strategies explicitly, such as replicate-weight variance estimation
- Minimal abstraction is used, so each step remains visible and tractable

These features allow instructors to demonstrate how survey weights, replicate designs, and multiple imputation contribute to final results. Students can follow the full analytic path from raw inputs to pooled estimates using transparent, inspectable code and data structures.

Author(s)

Joseph N. Cohen, CUNY Queens College

References

Barnard J, Rubin DB. Small-sample degrees of freedom with multiple imputation. doi:10.1093/biomet/86.4.948.

Bricker J, Henriques AM, Moore KB. Updates to the sampling of wealthy families in the Survey of Consumer Finances. Finance and Economics Discussion Series 2017-114. https://www.federalreserve.gov/econres/scfindex.h

Kennickell AB, McManus DA, Woodburn RL. Weighting design for the 1992 Survey of Consumer Finances. U.S. Federal Reserve. https://www.federalreserve.gov/Pubs/OSS/oss2/papers/weight92.pdf

Kennickell AB. Multiple imputation in the Survey of Consumer Finances. Statistical Journal of the IAOS 33(1):143-151. doi:10.3233/SJI160278.

Little RJA, Rubin DB. Statistical analysis with missing data. ISBN: 9780470526798.

Lumley T. survey: Analysis of complex survey samples. R package version 4.1-1. https://CRAN.R-project.org/package=survey

Lumley T. Analysis of complex survey samples. doi:10.18637/jss.v009.i08.

Lumley T. Complex surveys: A guide to analysis using R. ISBN: 9781118210932.

U.S. Federal Reserve. Codebook for 2022 Survey of Consumer Finances. https://www.federalreserve.gov/econres/scfindex.ht

See Also

Useful links:

- https://github.com/jncohen/scf
- Report bugs at https://github.com/jncohen/scf/issues

---

scf_activate_theme *Activate SCF Plot Theme*

Description

Sets the default `ggplot2` theme to `scf_theme()`. Call this function manually in your session or script to apply the style globally.

Usage

```
scf_activate_theme()
```

Value

No return value, called for side effects.

`scf_corr` *Estimate Correlation Between Two Continuous Variables in SCF Microdata*

Description

This function estimates the linear association between two continuous variables using Pearson's correlation. Estimates are computed within each implicate and then pooled across implicates to account for imputation uncertainty.

Usage

```
scf_corr(scf, var1, var2)
```

Arguments

`scf`	An `scf_mi_survey` object, created by `scf_load()`
`var1`	One-sided formula specifying the first variable
`var2`	One-sided formula specifying the second variable

Details

Computes the Pearson correlation coefficient between two continuous variables using multiply-imputed, replicate-weighted SCF data. Returns pooled estimates and standard errors using Rubin's Rules.

Value

An object of class `scf_corr`, containing:

results Data frame with pooled correlation estimate, standard error, t-statistic, degrees of freedom, p-value, and minimum/maximum values across implicates.

imps Named vector of implicate-level correlations.

aux Variable names used in the estimation.

Implementation

- Inputs: an `scf_mi_survey` object and two one-sided formulas (e.g., `~income`)
- Correlation computed using `cor(..., use = "complete.obs")` within each implicate
- Rubin's Rules applied to pool results across implicates

Interpretation

Pearson's `$r$` ranges from -1 to +1 and reflects the strength and direction of a linear bivariate association between two continuous variables. Values near 0 indicate weak linear association. Note that the operation is sensitive to outliers and does not capture nonlinear relationships nor adjust for covariates.

Statistical Notes

Correlation is computed within each implicate using complete cases. Rubin's Rules are applied manually to pool estimates and calculate total variance. Degrees of freedom are adjusted using the Barnard-Rubin method. This function does not use `scf_MIcombine()`, which is intended for vector-valued estimates; direct pooling is more appropriate for scalar statistics like correlation coefficients.

Note

Degrees of freedom are approximated using a simplified Barnard–Rubin adjustment, since correlation is a scalar quantity. Interpret cautiously with few implicates.

See Also

`scf_plot_hex()`, `scf_ols()`

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Estimate correlation between income and net worth
corr <- scf_corr(scf2022, ~income, ~networth)
print(corr)
summary(corr)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_design *Construct SCF Core Data Object*

Description

Stores SCF microdata as five implicate-specific designs created by `survey::svrepdesign()`. Raw implicate data frames are not retained.

Usage

```
scf_design(design, year, n_households)
```

Arguments

design	A list of five survey::svrepdesign() objects (one per implicate).
year	Numeric SCF survey year (e.g., 2022).
n_households	Numeric total U.S. households represented in year.

Details

Wrap a list of replicate-weighted survey designs into an "scf_mi_survey". Typically called by scf_load().

Value

An object of class "scf_mi_survey" with:

mi_design List of replicate-weighted designs (one per implicate).

year SCF survey year.

n_households Estimated number of U.S. households.

See Also

scf_load(), scf_update()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Construct scf_mi_survey object
obj <- scf_design(
  design = scf2022$mi_design,
  year = 2022,
  n_households = attr(scf2022, "n_households")
)
class(obj)
length(obj$mi_design)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

scf_download *Download and Prepare SCF Microdata for Local Analysis*

Description

Downloads SCF public-use microdata from official servers. For each year, this function retrieves five implicates, merges them with replicate weights and official summary variables, and saves them as `.rds` files ready for use with `scf_load()`.

Usage

```
scf_download(years = seq(1989, 2022, 3), overwrite = FALSE, verbose = TRUE)
```

Arguments

`years`	Integer vector of SCF years to download (e.g., `c(2016, 2019)`). Must be triennial from 1989 to 2022.
`overwrite`	Logical. If `TRUE`, re-download and overwrite existing `.rds` files. Default is `FALSE`.
`verbose`	Logical. If `TRUE`, display progress messages. Default is `TRUE`.

Value

These files are designed to be loaded using scf_load(), which wraps them into replicate-weighted designs.

Implementation

This function downloads from official servers three types of files for each year:

- five versions of the dataset (one per implicate), each stored as a separate data frame in a list
- a table of replicate weights, and
- a data table with official derivative variables

These tables are collected to a list and saved to an `.rds` format file in the working directory. By default, the function downloads all available years.

Details

The SCF employs multiply-imputed data sets to address unit-level missing data. Each household appears in one of five implicates. This function ensures all implicates are downloaded, merged, and prepared for downstream analysis using `scf_load()`, `scf_design()`, and the `scf` workflow.

References

U.S. Federal Reserve. Codebook for 2022 Survey of Consumer Finances. https://www.federalreserve.gov/econres/scfindex.ht

See Also

scf_load(), scf_design(), scf_update()

Examples

```
# Download and prepare SCF data for 2022
scf_download(2022)

# Load into a survey design object
scf2022 <- scf_load(2022)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_freq *Tabulate a Discrete Variable from SCF Microdata*

Description

This function estimates the relative frequency (proportion) of each category in a discrete variable from the SCF public-use microdata. Use this function to discern the univariate distribution of a discrete variable.

Usage

```
scf_freq(scf, var, by = NULL, percent = TRUE)
```

Arguments

scf	A scf_mi_survey object created by scf_load(). Must contain five replicate-weighted implicates.
var	A one-sided formula specifying a categorical variable (e.g., ~racecl).
by	Optional one-sided formula specifying a discrete grouping variable (e.g., ~own).
percent	Logical. If TRUE (default), scales results and standard errors to percentages.

Details

Computes weighted proportions and standard errors for a categorical variable in multiply-imputed SCF data, optionally stratified by a grouping variable. Proportions and standard errors are computed separately within each implicate using svymean(), then averaged across implicates using SCF-recommended pooling logic. Group-wise frequencies are supported, but users may find the features of scf_xtab() to be more useful.

Value

A list of class "scf_freq" with:

results Pooled category proportions and standard errors, by group if specified.

imps A named list of implicate-level proportion estimates.

aux Metadata about the variable and grouping structure.

Details

Proportions are estimated within each implicate using survey::svymean(), then pooled using the standard MI formula for proportions. When a grouping variable is provided via by, estimates are produced separately for each group-category combination. Results may be scaled to percentages using the percent argument.

Estimates are pooled using the standard formula:

- The mean of implicate-level proportions is the point estimate
- The standard error reflects both within-implicate variance and across-implicate variation

Unlike means or model parameters, category proportions do not use Rubin's full combination rules (e.g., degrees of freedom).

See Also

scf_xtab(), scf_plot_dist()]

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Proportions of homeownership
scf_freq(scf2022, ~own)

# Example for real analysis: ross-tabulate education by homeownership
scf_freq(scf2022, ~own, by = ~edcl)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

scf_glm *Estimate Generalized Linear Model from SCF Microdata*

Description

Estimates generalized linear models (GLMs) with SCF public-use microdata. Use this function when modeling outcomes that follow non-Gaussian distributions (e.g., binary or count data). Rubin's Rules are used to combine implicate-level coefficient and variance estimates.

GLMs are performed across SCF implicates using `svyglm()` and returns pooled coefficients, standard errors, z-values, p-values, and fit diagnostics including AIC and pseudo-R-Squared when applicable.

Usage

```
scf_glm(object, formula, family = binomial())
```

Arguments

`object`	A `scf_mi_survey` object, typically created using `scf_load()` and `scf_design()`.
`formula`	A valid model formula, e.g., `rich ~ age + factor(edcl)`.
`family`	A GLM family object such as `binomial()`, `poisson()`, or `gaussian()`. Defaults to `binomial()`.

Value

An object of class `"scf_glm"` and `"scf_model_result"` with:

results A data frame of pooled coefficients, standard errors, z-values, p-values, and significance stars.

fit A list of fit diagnostics including mean and SD of AIC; for binomial models, pseudo-R2 and its SD.

models A list of implicate-level `svyglm` model objects.

call The matched function call.

Implementation

This function fits a GLM to each implicate in a `scf_mi_survey` object using `survey::svyglm()`. The user specifies a model formula and a valid GLM family (e.g., `binomial()`, `poisson()`, `gaussian()`). Coefficients and variance-covariance matrices are extracted from each implicate and pooled using Rubin's Rules.

Details

Generalized linear models (GLMs) extend linear regression to accommodate non-Gaussian outcome distributions. The choice of `family` determines the link function and error distribution. For example:

- `binomial()` fits logistic regression for binary outcomes
- `poisson()` models count data
- `gaussian()` recovers standard OLS behavior

Model estimation is performed independently on each implicate using `svyglm()` with replicate weights. Rubin's Rules are used to pool coefficient estimates and variance matrices. For the pooling procedure, see `scf_MIcombine()`.

Internal Suppression

For CRAN compliance and to prevent diagnostic overload during package checks, this function internally wraps each implicate-level model call in `suppressWarnings()`. This suppresses the known benign warning:

`"non-integer #successes in a binomial glm!"`

which arises from using replicate weights with `family = binomial()`. This suppression does not affect model validity or inference. Users wishing to inspect warnings can run `survey::svyglm()` directly on individual implicates via `model$models[[i]]`.

For further background, see: https://stackoverflow.com/questions/12953045/warning-non-integer-successes-in-a-binomial-glm-survey-packages

See Also

`scf_ols()`, `scf_logit()`, `scf_regtable()`

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Run logistic regression
model <- scf_glm(scf2022, own ~ age + factor(edcl), family = binomial())
summary(model)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

scf_implicates *Extract Implicate-Level Estimates from SCF Results*

Description

Returns implicate-level outputs from SCF result objects produced by functions in the `scf` suite. Supports result objects containing implicate-level data frames, `svystat` summaries, or `svyglm` model fits.

Usage

```
scf_implicates(x, long = FALSE)
```

Arguments

`x`	A result object containing implicate-level estimates (e.g., from scf_mean, scf_ols).
`long`	Logical. If TRUE, returns stacked data frame. If FALSE, returns list.

Value

A list of implicate-level data frames, or a single stacked data frame if `long = TRUE`.

Usage

This function allows users to inspect how estimates vary across the SCF's five implicates, which is important for diagnostics, robustness checks, and transparent reporting.

For example:

```
scf_implicates(scf_mean(scf2022, ~income))
scf_implicates(scf_ols(scf2022, networth ~ age + income), long = TRUE)
```

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Extract implicate-level results
out <- scf_freq(scf2022, ~own)
scf_implicates(out, long = TRUE)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

scf_load *Load SCF Data as Multiply-Imputed Survey Designs*

Description

Converts SCF .rds files prepared by scf_download() into scf_mi_survey objects. Each object wraps five implicates per year in replicate-weighted, multiply-imputed survey designs suitable for use with scf_ functions.

Converts SCF .rds files prepared by scf_download() into scf_mi_survey objects. Each object wraps five implicates per year in replicate-weighted, multiply-imputed survey designs suitable for use with scf_ functions.

Usage

```
scf_load(min_year, max_year = min_year, data_directory = ".")
```

Arguments

min_year	Integer. First SCF year to load (1989–2022, divisible by 3).
max_year	Integer. Last SCF year to load. Defaults to min_year.
data_directory	Character. Directory containing .rds files or a full path to a single .rds file. Defaults to the current working directory ".". For examples and tests, use tempdir() to avoid leaving files behind.

Value

Invisibly returns a scf_mi_survey (or named list if multiple years). Attributes: mock (logical), year, n_households.

Invisibly returns a scf_mi_survey (or named list if multiple years). Attributes: mock (logical), year, n_households.

Implementation

Provide a year or range and either (1) a directory containing scf<year>.rds files, or (2) a full path to a single .rds file. Files must contain five implicate data frames with columns wgt and wt1b1..wt1bK (typically K=999).

Provide a year or range and either (1) a directory containing scf<year>.rds files, or (2) a full path to a single .rds file. Files must contain five implicate data frames with columns wgt and wt1b1..wt1bK (typically K=999).

See Also

scf_download(), scf_design(), scf_update(), survey::svrepdesign()

Load SCF Data as Multiply-Imputed Survey Designs

scf_download(), scf_design(), scf_update(), survey::svrepdesign()

Examples

```
# Using with CRAN-compliant mock data:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_logit *Estimate Logistic Regression Model using SCF Microdata*

Description

Fits a replicate-weighted logistic regression model to multiply-imputed SCF data, returning pooled coefficients or odds ratios with model diagnostics. Use this function to model a binary variable as a function of predictors.

Usage

```
scf_logit(object, formula, odds = TRUE, ...)
```

Arguments

`object`	A `scf_mi_survey` object created with `scf_load()` and `scf_design()`.
`formula`	A model formula specifying a binary outcome and predictors, e.g., `rich ~ age + factor(edcl)`.
`odds`	Logical. If `TRUE` (default), exponentiates coefficient estimates to produce odds ratios for interpretability.
`...`	Additional arguments passed to `scf_glm()`.

Value

An object of class `"scf_logit"` and `"scf_model_result"` with:

results A data frame of pooled estimates (log-odds or odds ratios), standard errors, and test statistics.

fit Model diagnostics including AIC and pseudo-R-Squared (for binomial family).

models List of implicate-level `svyglm` model objects.

call The matched function call.

Details

This function internally calls `scf_glm()` with `family = binomial()` and optionally exponentiates pooled log-odds to odds ratios.

Logistic regression models the probability of a binary outcome using the logit link.

Coefficients reflect the change in log-odds associated with a one-unit change in the predictor.

When `odds = TRUE`, the coefficient estimates and standard errors are transformed from log-odds to odds ratios and approximate SEs.

Warning

When modeling binary outcomes using survey-weighted logistic regression, users may encounter the warning:

`"non-integer #successes in a binomial glm!"`

This message is benign. It results from replicate-weighted survey designs where the implied number of "successes" is non-integer. The model is estimated correctly. Coefficients are valid and consistent with maximum likelihood.

For background, see: https://stackoverflow.com/questions/12953045/warning-non-integer-successes-in-a-binomial-glm-survey-packages

See Also

`scf_glm()`, `scf_ols()`, `scf_MIcombine()`

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Run logistic regression
model <- scf_logit(scf2022, own ~ age)
summary(model)

# Do not implement these lines in real analysis: Cleanup for package check
unlink(file.path(td, "scf2022.rds"), force = TRUE)
```

scf_mean *Estimate Mean in Multiply-Imputed SCF Data*

Description

Returns the population-level estimate of a continuous variable's weighted mean across the Survey's five implicates. Use this operation to derive an estimate of a population's 'typical' or 'average' score on a continuous variable.

Usage

```
scf_mean(scf, var, by = NULL, verbose = FALSE)
```

Arguments

scf	A scf_mi_survey object created with scf_load(). Must contain five replicate-weighted implicates.
var	A one-sided formula identifying the continuous variable to summarize (e.g., ~networth).
by	Optional one-sided formula specifying a discrete grouping variable for stratified means.
verbose	Logical. If TRUE, include implicate-level results in print output. Default is FALSE.

Value

A list of class "scf_mean" with:

results Pooled estimates with standard errors and range across implicates. One row per group, or one row total.

imps A named list of implicate-level estimates.

aux Variable and group metadata.

Details

The mean is a measure of central tendency that represents the arithmetic average of a distribution. It is most appropriate when the distribution is symmetric and not heavily skewed. Unlike the median, the mean is sensitive to extreme values, which may distort interpretation in the presence of outliers. Use this function to describe the "typical" value of a continuous variable in the population or within subgroups.

See Also

scf_median(), scf_percentile(), scf_xtab(), scf_plot_dist()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Estimate means
scf_mean(scf2022, ~networth)
scf_mean(scf2022, ~networth, by = ~edcl)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_median *Estimate the Population Median of a Continuous SCF Variable*

Description

Estimates the median (50th percentile) of a continuous SCF variable. Use this operation to characterize a typical or average value. In contrast to `scf_mean()`, this function is both uninfluenced by, and insensitive to, outliers.

Usage

```
scf_median(scf, var, by = NULL)
```

Arguments

`scf`	A `scf_mi_survey` object created by `scf_load()`. Must contain five implicates.
`var`	A one-sided formula specifying the continuous variable of interest (e.g., `~networth`).
`by`	Optional one-sided formula for a categorical grouping variable.

Value

A list of class `"scf_median"` with:

results A data frame with pooled medians, standard errors, and range across implicates.

imps A list of implicate-level results.

aux Variable and grouping metadata.

Implementation

This function wraps scf_percentile() with q = 0.5. The user provides a scf_mi_survey object and a one-sided formula indicating the variable of interest. An optional grouping variable can be specified with a second formula. Output includes pooled medians, standard errors, min/max across implicates, and implicate-level values.

Statistical Notes

Median estimates are not pooled using Rubin's Rules. Following SCF protocol, the function calculates the median within each implicate and averages across implicates. See the data set's official codebook.

See Also

scf_percentile(), scf_mean()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Estimate medians
scf_median(scf2022, ~networth)
scf_median(scf2022, ~networth, by = ~edcl)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_MIcombine *Combine Estimates Across SCF Implicates Using Rubin's Rules*

Description

This function is the **canonical implementation** of Rubin's Rules in the scf package. It defines how point estimates, standard errors, and degrees of freedom are pooled across the SCF's multiply-imputed replicate-weighted implicates.

Usage

```
scf_MIcombine(results, variances, call = sys.call(), df.complete = Inf)
```

Arguments

`results`	A list of implicate-level model outputs. Each element must be a named numeric vector or an object with methods for `coef()` and `vcov()`. Typically generated internally by modeling functions.
`variances`	Optional list of variance-covariance matrices. If omitted, extracted using `vcov()`.
`call`	Optional. The originating function call. Defaults to `sys.call()`.
`df.complete`	Optional degrees of freedom for the complete-data model. Used for small-sample corrections. Defaults to `Inf`, assuming large-sample asymptotics.

Details

Most `scf` functions that compute descriptive statistics or model estimates internally call this function (or apply its logic). As such, this documentation serves as the authoritative explanation of the pooling procedure and its assumptions for all SCF workflows in the package.

Value

An object of class `"scf_MIresult"` with components:

coefficients Pooled point estimates across implicates.

variance Pooled variance-covariance matrix.

df Degrees of freedom for each parameter, adjusted using Barnard-Rubin formula.

missinfo Estimated fraction of missing information for each parameter.

nimp Number of implicates used in pooling.

call Function call recorded for reproducibility.

Supports `coef()`, `SE()`, `confint()`, and `summary()` methods.

Implementation

`scf_MIcombine()` pools a set of implicate-level estimates and their associated variance-covariance matrices using Rubin's Rules.

This includes:

- Calculation of pooled point estimates
- Total variance from within- and between-imputation components
- Degrees of freedom via Barnard-Rubin method
- Fraction of missing information

Inputs are typically produced by functions like `scf_mean()`, `scf_ols()`, or `scf_percentile()`.

This function is primarily used internally, but may be called directly by advanced users constructing custom estimation routines from implicate-level results.

Details

The SCF provides five implicates per survey wave, each a plausible version of the population under a specific missing-data model. Analysts conduct the same statistical procedure on each implicate, producing a set of five estimates $Q_1, Q_2, ..., Q_5$. These are then combined using Rubin's Rules, a procedure to combine results across these implicates with an attempt to account for:

- **Within-imputation variance**: Uncertainty from complex sample design
- **Between-imputation variance**: Uncertainty due to missing data

For a scalar quantity Q, the pooled estimate and total variance are calculated as:

$$\bar{Q} = \frac{1}{M} \sum Q_m$$

$$\bar{U} = \frac{1}{M} \sum U_m$$

$$B = \frac{1}{M-1} \sum (Q_m - \bar{Q})^2$$

$$T = \bar{U} + \left(1 + \frac{1}{M}\right) B$$

Where:

- M is the number of implicates (typically 5 for SCF)
- Q_m is the estimate from implicate m
- U_m is the sampling variance of Q_m, accounting for replicate weights and design

The total variance T reflects both within-imputation uncertainty (sampling error) and between-imputation uncertainty (missing-data imputation).

The standard error of the pooled estimate is $\sqrt{T}$. Degrees of freedom are adjusted using the Barnard-Rubin method:

$$\nu = (M-1)\left(1 + \frac{\bar{U}}{(1+\frac{1}{M})B}\right)^2$$

The fraction of missing information (FMI) is also reported: it reflects the proportion of total variance attributable to imputation uncertainty.

See `scf_MIcombine()` for full implementation details.

References

Barnard J, Rubin DB. Small-sample degrees of freedom with multiple imputation. doi:10.1093/biomet/86.4.948.

Little RJA, Rubin DB. Statistical analysis with missing data. ISBN: 9780470526798.

U.S. Federal Reserve. Codebook for 2022 Survey of Consumer Finances. https://www.federalreserve.gov/econres/scfindex.ht

See Also

`scf_mean()`, `scf_ols()`, `scf_glm()`, `scf_logit()`

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Pool simple survey mean for mock data
outlist <- lapply(scf2022$mi_design, function(d) survey::svymean(~I(age >= 65), d))
pooled  <- scf_MIcombine(outlist)     # vcov/coef extracted automatically
SE(pooled); coef(pooled)

unlink("scf2022.rds", force = TRUE)
```

`scf_ols` *Estimate an Ordinary Least Squares Regression on SCF Microdata*

Description

Computes an OLS regression on SCF data using `svyglm()` across the SCF's five implicates. Returns coefficient estimates, standard errors, test statistics, and model diagnostics.

Usage

```
scf_ols(object, formula)
```

Arguments

`object`	A `scf_mi_survey` object created with `scf_load()` and `scf_design()`. Must contain five implicates with replicate weights.
`formula`	A model formula specifying a continuous outcome and predictor variables (e.g., `networth ~ income + age`).

Details

Fits a replicate-weighted linear regression model to each implicate of multiply-imputed SCF data and pools coefficients and standard errors using Rubin's Rules.

Value

An object of class `"scf_ols"` and `"scf_model_result"` with:

results A data frame of pooled coefficients, standard errors, t-values, p-values, and significance stars.

fit A list of model diagnostics including mean AIC, standard deviation of AIC, mean R-squared, and its standard deviation.

imps A list of implicate-level `svyglm` model objects.

call The matched call used to produce the model.

Implementation

Ordinary least squares (OLS) regression estimates the linear relationship between a continuous outcome and one or more predictor variables. Each coefficient represents the expected change in the outcome for a one-unit increase in the corresponding predictor, holding all other predictors constant.

Use this function to model associations between SCF variables while accounting for complex survey design and multiple imputation.

This function takes a `scf_mi_survey` object and a model formula. Internally, it fits a weighted linear regression to each implicate using `survey::svyglm()`, extracts coefficients and variance-covariance matrices, and pools them via `scf_MIcombine()`.

See Also

`scf_glm()`, `scf_logit()`, `scf_MIcombine()`

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Run OLS model
model <- scf_ols(scf2022, networth ~ income + age)
print(model)
summary(model)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_percentile *Estimate Percentile in a Continuous Variable in SCF Microdata*

Description

Calculates the percentile score of a continuous variable in the SCF microdata. Use this function to either (1) identify where a continuous variable's value stands in relation to all observed values, or (2) to discern value below which a user-specified percentage of households fall on that metric.

Usage

```
scf_percentile(scf, var, q = 0.5, by = NULL, verbose = FALSE)
```

Arguments

scf	A scf_mi_survey object created with scf_load(). Must contain five implicates.
var	A one-sided formula identifying the continuous variable to summarize (e.g., ~networth).
q	A quantile to estimate (between 0 and 1). Defaults to 0.5 (median).
by	Optional one-sided formula specifying a discrete grouping variable for stratified percentiles.
verbose	Logical. If TRUE, include implicate-level results in print output. Default is FALSE.

Value

A list of class "scf_percentile" with:

results Pooled percentile estimates with standard errors and range across implicates. One row per group, or one row total.

imps A named list of implicate-level estimates.

aux Variable, group, and quantile metadata.

Details

The percentile is a value below which a given percentage of observations fall. This function estimates the desired percentile score within each implicate of the SCF's multiply-imputed dataset, and then averages them to generate a population estimate.

When a grouping variable is supplied, the percentile is estimated separately within each group in each implicate. Group-level results are then pooled across implicates.

Unlike scf_mean(), this function does not pool results using Rubin's Rules. Instead, it follows the Federal Reserve's practice for reporting percentiles in official SCF publications: compute the desired percentile separately within each implicate, then average the resulting values to obtain a pooled estimate.

Standard errors are approximated using the sample standard deviation of the five implicate-level estimates. This method is consistent with the SCF's official percentile macro (see Kennickell 1998; per Federal Reserve Board's 2022 SCF's official SAS script)

References

Kennickell AB, McManus DA, Woodburn RL. Weighting design for the 1992 Survey of Consumer Finances. U.S. Federal Reserve. https://www.federalreserve.gov/Pubs/OSS/oss2/papers/weight92.pdf

U.S. Federal Reserve. Codebook for 2022 Survey of Consumer Finances. https://www.federalreserve.gov/econres/scfindex.ht

See Also

scf_median()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Estimate percentiles
scf_percentile(scf2022, ~networth, q = 0.5)
scf_percentile(scf2022, ~networth, q = 0.9, by = ~edcl)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_plot_bbar *Stacked Bar Chart of Two Discrete Variables in SCF Data*

Description

Visualizes a discrete-discrete bivariate distribution using stacked bars based on pooled cross-tabulations from `scf_xtab()`. Use this function to visualize the relationship between two discrete variables.

Usage

```
scf_plot_bbar(
  design,
  rowvar,
  colvar,
  scale = c("percent", "count"),
  percent_by = c("total", "row", "col"),
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  fill_colors = NULL,
  row_labels = NULL,
  col_labels = NULL
)
```

Arguments

`design`	A `scf_mi_survey` object created by `scf_load()`. Must contain five implicates with replicate weights.
`rowvar`	A one-sided formula for the x-axis grouping variable (e.g., `~edcl`).
`colvar`	A one-sided formula for the stacked fill variable (e.g., `~racecl`).
`scale`	Character. One of `"percent"` (default) or `"count"`.

percent_by	Character. One of "total" (default), "row", or "col" — determines normalization base when scale = "percent".
title	Optional character string for the plot title.
xlab	Optional character string for the x-axis label.
ylab	Optional character string for the y-axis label.
fill_colors	Optional vector of fill colors to pass to ggplot2::scale_fill_manual().
row_labels	Optional named vector to relabel row categories (x-axis).
col_labels	Optional named vector to relabel col categories (legend).

Value

A ggplot2 object.

Implementation

This function calls scf_xtab() to estimate the joint distribution of two categorical variables across multiply-imputed SCF data. The result is translated into a ggplot2 stacked bar chart using pooled counts or normalized percentages.

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Stacked bar chart: education by ownership
scf_plot_bbar(scf2022, ~own, ~edcl)

# Example for real analysis: Column percentages instead of total percent
scf_plot_bbar(scf2022, ~own, ~edcl, percent_by = "col")

# Example for real analysis: Raw counts (estimated number of households)
scf_plot_bbar(scf2022, ~own, ~edcl, scale = "count")

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_plot_cbar *Bar Plot of Summary Statistics by Grouping Variable in SCF Data*

Description

Computes and plots a grouped summary statistic (either a mean, median, or quantile) for a continuous variable across a discrete factor. Estimates are pooled across implicates using `scf_mean()`, `scf_median()`, or `scf_percentile()`. Use this function to visualize the bivariate relationship between a discrete and a continuous variable.

Usage

```
scf_plot_cbar(
  design,
  yvar,
  xvar,
  stat = "mean",
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  fill = "#0072B2",
  angle = 30,
  label_map = NULL
)
```

Arguments

`design`	A `scf_mi_survey` object from `scf_load()`.
`yvar`	One-sided formula for the continuous variable (e.g., `~networth`).
`xvar`	One-sided formula for the grouping variable (e.g., `~racecl`).
`stat`	`"mean"` (default), `"median"`, or a quantile (numeric between 0 and 1).
`title`	Plot title (optional).
`xlab`	X-axis label (optional).
`ylab`	Y-axis label (optional).
`fill`	Bar fill color. Default is `"#0072B2"`.
`angle`	Angle of x-axis labels. Default is 30.
`label_map`	Optional named vector to relabel x-axis category labels.

Value

A `ggplot2` object.

Implementation

The user specifies a continuous outcome (`yvar`) and a discrete grouping variable (`xvar`) via one-sided formulas. Group means are plotted by default. Medians or other percentiles can be specified via the `stat` argument.

Results are plotted using `ggplot2::geom_col()`, styled with `scf_theme()`, and optionally customized with additional arguments (e.g., axis labels, color, angles).

See Also

scf_mean(), scf_median(), scf_percentile(), scf_theme()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Plot mean net worth by education level
scf_plot_cbar(scf2022, ~networth, ~edcl, stat = "mean")

# Example for real analysis: Visualize 90th percentile of income by education
scf_plot_cbar(scf2022, ~income, ~edcl, stat = 0.9, fill = "#D55E00")

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_plot_dbar *Plot Bar Chart of a Discrete Variable from SCF Data*

Description

Creates a bar chart that visualizes the distribution of a discrete variable.

Usage

```
scf_plot_dbar(
  design,
  variable,
  title = NULL,
  xlab = NULL,
  ylab = "Percent",
  angle = 30,
  fill = "#0072B2",
  label_map = NULL
)
```

Arguments

design	A scf_mi_survey object created by scf_load(). Must contain valid implicates.
variable	A one-sided formula specifying a categorical variable (e.g., ~racecl).
title	Optional character string for the plot title. Default: "Distribution of <variable>".

xlab	Optional x-axis label. Default: variable name.
ylab	Optional y-axis label. Default: "Percent".
angle	Integer. Rotation angle for x-axis labels. Default is 30.
fill	Fill color for bars. Default is "#0072B2".
label_map	Optional named vector to relabel x-axis category labels.

Value

A ggplot2 object representing the pooled bar chart.

Implementation

This function internally calls scf_freq() to compute population proportion estimates, which are then plotted using ggplot2::geom_col(). The default output is scaled to percent and can be customized via title, axis labels, angle, and color.

Details

Produces a bar chart of category proportions from a one-way tabulation, pooled across SCF implicates using scf_freq(). This function summarizes weighted sample composition and communicates categorical distributions effectively in descriptive analysis.

Dependencies

Requires the ggplot2 package.

See Also

scf_freq(), scf_plot_bbar(), scf_xtab()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Bar chart of education categories
scf_plot_dbar(scf2022, ~edcl)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

scf_plot_dist *Plot a Univariate Distribution of an SCF Variable*

Description

This function provides a unified plotting interface for visualizing the distribution of a single variable from multiply-imputed SCF data. Discrete variables produce bar charts of pooled proportions; continuous variables produce binned histograms. Use this function to visualize the univariate distribution of an SCF variable.

Usage

```
scf_plot_dist(
  design,
  variable,
  bins = 30,
  title = NULL,
  xlab = NULL,
  ylab = "Percent",
  angle = 30,
  fill = "#0072B2",
  labels = NULL
)
```

Arguments

design	A scf_mi_survey object created by scf_load().
variable	A one-sided formula specifying the variable to plot.
bins	Number of bins for continuous variables. Default is 30.
title	Optional plot title.
xlab	Optional x-axis label.
ylab	Optional y-axis label. Default is "Percent".
angle	Angle for x-axis tick labels. Default is 30.
fill	Fill color for bars. Default is "#0072B2".
labels	Optional named vector of custom axis labels (for discrete variables only).

Value

A ggplot2 object.

Implementation

For discrete variables (factor or numeric with <= 25 unique values), the function uses scf_freq() to calculate category proportions and produces a bar chart. For continuous variables, it bins values across implicates and estimates Rubin-pooled frequencies for each bin.

Users may supply a named vector of custom axis labels using the labels argument.

See Also

scf_theme()

Examples

```
# Mock workflow for CRAN (demo only — not real SCF data)
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

scf_plot_dist(scf2022, ~own)
scf_plot_dist(scf2022, ~age, bins = 10)

unlink("scf2022.rds", force = TRUE)


# Real workflow
scf_download(2022)
scf2022 <- scf_load(2022)
scf_plot_dist(scf2022, ~own)
scf_plot_dist(scf2022, ~age, bins = 10)

# Clean up
unlink("scf2022.rds", force = TRUE)
```

---

scf_plot_hex *Hexbin Plot of Two Continuous SCF Variables*

Description

Visualizes the bivariate relationship between two continuous SCF variables using hexagonal bins.

Usage

```
scf_plot_hex(design, x, y, bins = 50, title = NULL, xlab = NULL, ylab = NULL)
```

Arguments

design	A scf_mi_survey object created by scf_load().
x	A one-sided formula for the x-axis variable (e.g., ~income).
y	A one-sided formula for the y-axis variable (e.g., ~networth).
bins	Integer. Number of hexagonal bins along the x-axis. Default is 50.
title	Optional character string for the plot title.
xlab	Optional x-axis label. Defaults to the variable name.
ylab	Optional y-axis label. Defaults to the variable name.

Value

A ggplot2 object displaying a Rubin-pooled hexbin plot.

Implementation

The function stacks all implicates into one data frame, retains replicate weights, and uses ggplot2::geom_hex() to produce a density-style scatterplot. The color intensity of each hexagon reflects the Rubin-pooled weighted count of households in that cell. Missing values are excluded.

This plot is especially useful for visualizing joint distributions with large samples and skewed marginals, such as net worth vs. income.

Aesthetic Guidance

This plot uses a log-scale fill and viridis palette to highlight variation in density. To adjust the visual style globally, use scf_theme() or set it explicitly with ggplot2::theme_set(scf_theme()). For mobile-friendly or publication-ready appearance, export the plot at 5.5 x 5.5 inches, 300 dpi.

Dependencies

Requires the ggplot2 package. The fill scale uses scale_fill_viridis_c() from ggplot2. Requires the **hexbin** package. The function will stop with an error if it is not installed.

See Also

scf_corr(), scf_plot_smooth(), scf_theme()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Plot hexbin of income vs. net worth
scf_plot_hex(scf2022, ~income, ~networth)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

scf_plot_hist *Histogram of a Continuous Variable in Multiply-Imputed SCF Data*

Description

Produces a histogram of a continuous SCF variable by binning across implicates, pooling weighted bin counts using `scf_freq()`, and plotting the result. Values outside `xlim` are clamped into the nearest endpoint to ensure all observations are included and replicate-weighted bins remain stable.

Usage

```
scf_plot_hist(
  design,
  variable,
  bins = 30,
  xlim = NULL,
  title = NULL,
  xlab = NULL,
  ylab = "Weighted Count",
  fill = "#0072B2"
)
```

Arguments

`design`	A `scf_mi_survey` object from `scf_load()`.
`variable`	A one-sided formula indicating the numeric variable to plot.
`bins`	Number of bins (default: 30).
`xlim`	Optional numeric range. Values outside will be included in edge bins.
`title`	Optional plot title.
`xlab`	Optional x-axis label. Defaults to the variable name.
`ylab`	Optional y-axis label. Defaults to "Weighted Count".
`fill`	Fill color for bars (default: `"#0072B2"`).

Value

A `ggplot2` object representing the Rubin-pooled histogram.

Implementation

This function bins a continuous variable (after clamping to `xlim` if supplied), applies the same `cut()` breaks across implicates using `scf_update_by_implicate()`, and computes Rubin-pooled frequencies with `scf_freq()`. Results are filtered to remove bins with undefined proportions and then plotted using `ggplot2::geom_col()`.

The logic here is specific to operations where the bin assignment must be computed **within** each implicate, not after pooling. This approach ensures consistent binning and stable pooled estimation in the presence of multiply-imputed microdata.

See Also

scf_freq(), scf_plot_dbar(), scf_plot_smooth(), scf_update_by_implicate()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Plot histogram of age
scf_plot_hist(scf2022, ~age, bins = 10)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_plot_smooth *Smoothed Distribution Plot of a Continuous Variable in SCF Data*

Description

Draws a smoothed distribution plot of a continuous variable in the SCF. Use this function to visualize a single continuous variable's distribution.

Usage

```
scf_plot_smooth(
  design,
  variable,
  binwidth = NULL,
  xlim = NULL,
  method = "loess",
  span = 0.2,
  color = "blue",
  xlab = NULL,
  ylab = "Percent of Households",
  title = NULL
)
```

Arguments

design	A scf_mi_survey object created by scf_load().
variable	A one-sided formula specifying a continuous variable (e.g., ~networth).
binwidth	Optional bin width. Default uses Freedman–Diaconis rule.

xlim	Optional numeric vector of length 2 to truncate axis.
method	Character. Smoothing method: "loess" (default) or "lm".
span	Numeric LOESS span. Default is 0.2. Ignored if method = "lm".
color	Line color. Default is "blue".
xlab	Optional label for x-axis. Defaults to the variable name.
ylab	Optional label for y-axis. Defaults to "Percent of Households".
title	Optional plot title.

Value

A ggplot2 object.

Implementation

Visualizes the weighted distribution of a continuous SCF variable by stacking implicates, binning observations, and smoothing pooled proportions. This function is useful for examining distribution shape, skew, or modality in variables like income or wealth.

All implicates are stacked and weighted, binned across a data-driven or user-specified bin width. Each bin's weight share is calculated, and a smoothing curve is fit to the resulting pseudo-density.

See Also

scf_theme(), scf_plot_dist()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Plot smoothed distribution
scf_plot_smooth(scf2022, ~networth, xlim = c(0, 2e6),
                method = "loess", span = 0.25)

# Do not implement these lines in real analysis: Cleanup for package check
 unlink("scf2022.rds", force = TRUE)
```

`scf_prop_test` *Test a Proportion in SCF Data*

Description

Tests a binary variable's proportion against a null hypothesis (one-sample), or compares proportions across two groups (two-sample). Supports two-sided, less-than, or greater-than alternatives.

Usage

```
scf_prop_test(
  design,
  var,
  group = NULL,
  p = 0.5,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

`design`	A `scf_mi_survey` object created by `scf_load()`. Must contain replicate-weighted implicates.
`var`	A one-sided formula indicating a binary variable (e.g., `~rich`).
`group`	Optional one-sided formula indicating a binary grouping variable (e.g., `~female`). If omitted, a one-sample test is performed.
`p`	Null hypothesis value. Defaults to `0.5` for one-sample, `0` for two-sample tests.
`alternative`	Character. One of `"two.sided"` (default), `"less"`, or `"greater"`.
`conf.level`	Confidence level for the confidence interval. Default is `0.95`.

Value

An object of class `"scf_prop_test"` with:

results A data frame with the pooled estimate, standard error, z-statistic, p-value, confidence interval, and significance stars.

proportions (Only in two-sample tests) A data frame of pooled proportions by group.

fit A list describing the method, null value, alternative hypothesis, and confidence level.

Statistical Notes

Proportions are computed in each implicate using weighted means, and variances are approximated under the binomial model. Rubin's Rules are applied to pool point estimates and standard errors. For pooling details, see `scf_MIcombine()`.

See Also

scf_ttest(), scf_mean(), scf_MIcombine()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Wrangle data for example
scf2022 <- scf_update(scf2022,
  rich   = networth > 1e6,
  female = factor(hhsex, levels = 1:2, labels = c("Male","Female")),
  over50 = age > 50
)

# Example for real analysis: One-sample test
scf_prop_test(scf2022, ~rich, p = 0.10)

# Example for real analysis: Two-sample test
scf_prop_test(scf2022, ~rich, ~female, alternative = "less")

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_regtable *Format and Display Regression Results from Multiply-Imputed SCF Models*

Description

This function formats and aligns coefficient estimates, standard errors, and significance stars from one or more SCF regression model objects (e.g., from scf_ols(), scf_logit(), or scf_glm()).

Usage

```
scf_regtable(
  ...,
  model.names = NULL,
  digits = 0,
  auto_digits = FALSE,
  labels = NULL,
  output = c("console", "markdown", "csv"),
  file = NULL
)
```

Arguments

...	One or more SCF regression model objects, or a single list of such models.
model.names	Optional character vector naming the models. Defaults to "Model 1", "Model 2", etc.
digits	Integer specifying decimal places for numeric formatting when auto_digits = FALSE. Default is 0.
auto_digits	Logical; if TRUE, uses adaptive decimal places: 0 digits for large numbers (>= 1000), 2 digits for moderate (>= 1), and 3 digits for smaller values.
labels	Optional named character vector or labeling function to replace term names with descriptive labels.
output	Output format: one of "console" (print to console), "markdown" (print Markdown table for R Markdown), or "csv" (write CSV file).
file	File path for CSV output; required if output = "csv".

Details

It compiles a side-by-side table with terms matched across models, appends model fit statistics (sample size N, R-squared or pseudo-R-squared, and AIC), and outputs the results as console text, Markdown for R Markdown documents, or a CSV file.

The function aligns all unique coefficient terms across provided models, formats coefficients with significance stars and standard errors, appends model fit statistics as additional rows, and renders output in the specified format. It avoids external dependencies by using base R formatting and simple text or Markdown output.

Value

Invisibly returns a data frame with formatted regression results and fit statistics.

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Wrangle data for example:  Perform OLS regression
m1 <- scf_ols(scf2022, income ~ age)

# Example for real analysis: Print regression results as a console table
scf_regtable(m1, digits = 2)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

scf_subset *Subset an* scf_mi_survey *Object*

Description

Subsetting refers to the process of retaining only those observations that satisfy a logical (TRUE/FALSE) condition. This function applies such a filter independently to each implicate in an scf_mi_survey object created by scf_design() via scf_load(). The result is a new multiply-imputed, replicate-weighted survey object with appropriately restricted designs.

Usage

```
scf_subset(scf, expr)
```

Arguments

scf	A scf_mi_survey object, typically created by scf_load().
expr	A logical expression used to filter rows, evaluated separately in each implicate's variable frame (e.g., age < 65 & own == 1).

Value

A new scf_mi_survey object (see scf_design())

Implementation

Use scf_subset() to focus analysis on analytically meaningful sub-populations. For example, to analyze only households headed by seniors:

"r scf2022_seniors <- scf_subset(scf2022, age >= 65)

This is especially useful when analyzing populations such as renters, homeowners, specific age brackets, or any group defined by logical expressions over SCF variables.

Details

Filtering is conducted separately in each implicate. This preserves valid design structure but means that the same household may fall into or out of the subset depending on imputed values. For example, a household with five different age imputations—say, 64, 66, 63, 65, and 67—would be classified as a senior in only three of five implicates if subsetting on age >= 65.

Empty subsets in any implicate can cause downstream analysis to fail. Always check subgroup sizes after subsetting.

See Also

scf_load(), scf_update()

Examples

```
# Mock workflow for CRAN (demo only — not real SCF data)
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Filter for working-age households with positive net worth
scf_sub <- scf_subset(scf2022, age < 65 & networth > 0)
scf_mean(scf_sub, ~income)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

`scf_theme` *Default Plot Theme for SCF Visualizations*

Description

The theme is designed to:

- Render cleanly in **print** (single-column or wrapped layout)
- Scale well on **HD desktop** monitors without visual clutter
- Remain **legible on mobile** with clear fonts and sufficient contrast

The default figure dimensions assumed for export are **5.5 inches by 5.5 inches** at **300 dpi**, which balances compactness with accessibility across media.

All theme settings are exposed via comments to enable easy brand customization.

Usage

```
scf_theme(base_size = 13, base_family = "sans", grid = TRUE, axis = TRUE, ...)
```

Arguments

`base_size`	Base font size. Defaults to 13.
`base_family`	Font family. Defaults to "sans".
`grid`	Logical. Show gridlines? Defaults to TRUE.
`axis`	Logical. Include axis ticks and lines? Defaults to TRUE.
`...`	Additional arguments passed to `ggplot2::theme_minimal()`.

Details

Defines the SCF package's default `ggplot2` theme, optimized for legibility, clarity, and aesthetic coherence across print, desktop, and mobile platforms.

Value

A ggplot2 theme object applied by all scf_plot_*() functions.

See Also

ggplot2::theme(), scf_plot_dist()

Examples

```
library(ggplot2)
ggplot(mtcars, aes(factor(cyl))) +
  geom_bar(fill = "#0072B2") +
  scf_theme()
```

---

scf_ttest *T-Test of Means using SCF Microdata*

Description

Tests whether the mean of a continuous variable differs from a specified value (one-sample), or whether group means differ across a binary factor (two-sample). Estimates and standard errors are computed using svymean() within each implicate, then pooled using Rubin's Rules. Use this function to test hypotheses about means in the SCF microdata.

Usage

```
scf_ttest(
  design,
  var,
  group = NULL,
  mu = 0,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

Arguments

design	A scf_mi_survey object created by scf_load().
var	A one-sided formula specifying a numeric variable (e.g., ~income).
group	Optional one-sided formula specifying a binary grouping variable (e.g., ~female).
mu	Numeric. Null hypothesis value. Default is 0.
alternative	Character. One of "two.sided" (default), "less", or "greater".
conf.level	Confidence level for the confidence interval. Default is 0.95.

Value

An object of class "scf_ttest" with:

results A data frame with pooled estimate, standard error, t-statistic, degrees of freedom, p-value, and confidence interval.

means Group-specific means (for two-sample tests only).

fit List describing the test type, null hypothesis, confidence level, and alternative.

See Also

scf_prop_test(), scf_mean(), scf_MIcombine()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Wrangle data for example: Derive analysis vars
scf2022 <- scf_update(scf2022,
  female = factor(hhsex, levels = 1:2, labels = c("Male","Female")),
  over50 = age > 50
)

# Example for real analysis:  One-sample t-test
scf_ttest(scf2022, ~income, mu = 75000)

# Example for real analysis:  Two-sample t-test
scf_ttest(scf2022, ~income, group = ~female)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_update *Create or Alter SCF Variables*

Description

Use this function to create or alter SCF variables once the raw data set has been loaded into memory using the scf_load() function. This function updates an scf_mi_survey object by evaluating transformations within each implicate, and then returning a new object with the new or amended variables.

Most of the time, you can use scf_update() to define variables based on simple logical conditions, arithmetic transformations, or categorical binning. These rules are evaluated separately in

each implicate, using the same formula. However, if the transformation you want to apply depends on the distribution of the data within each implicate, such as computing an average percentile or ranking households across all implicates, this function will not suffice. In those cases, use `scf_update_by_implicate()` to write a custom function that operates on each implicate individually.

Usage

```
scf_update(object, ...)
```

Arguments

`object`	A `scf_mi_survey` object, typically created by `scf_load()`.
`...`	Named expressions assigning new or modified variables using `=` syntax. Each expression must return a vector of the same length as the implicate data frame.

Value

A new `scf_mi_survey` object with:

implicates A list of updated data frames (one per implicate).

mi_design A list of updated `svyrep.design` survey objects.

data (If present in the original object) unchanged pooled data.

Usage

Use `scf_update()` during data wrangling to clean, create, or alter variables before calculating statistics or running models. The function is useful when the analyst wishes to:

- Recode missing values that are coded as numeric data
- Recast variables that are not in the desired format (e.g., converting a numeric variable to a factor)
- Create new variables based on existing ones (e.g., calculating ratios, differences, or indicators)

See Also

`scf_load()`, `scf_update_by_implicate()`, `survey::svrepdesign()`

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Create a binary indicator for being over age 50
scf2022 <- scf_update(scf2022,
  over50 = age > 50
```

```
)

# Example: Create a log-transformed income variable
scf2022 <- scf_update(scf2022,
  log_income = log(income + 1)
)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

`scf_update_by_implicate` *Modify Each Implicate Individually in SCF Data*

Description

Each household in SCF data is represented by five *implicates*, which reflect uncertainty from the imputation process. Most transformations — such as computing log income or assigning categorical bins — can be applied uniformly across implicates using `scf_update()`. However, some operations depend on the *internal distribution* of variables within each implicate. For those, you need to modify each one separately.

This function extracts each implicate from the replicate-weighted survey design, applies your transformation, and rebuilds the survey design objects accordingly.

Usage

```
scf_update_by_implicate(object, f)
```

Arguments

`object`	A `scf_mi_survey` object from `scf_load()`.
`f`	A function that takes a data frame as input and returns a modified data frame. This function will be applied independently to each implicate.

Details

Applies a user-defined transformation to each implicate's data frame separately. This is useful when you need to compute values that depend on the distribution within each implicate — such as ranks, percentiles, or groupwise comparisons — which cannot be computed reliably using `scf_update()`.

Value

A modified `scf_mi_survey` object with updated implicate-level designs.

Use this When

- You need implicate-specific quantiles (e.g., flag households in the top 10% of wealth)
- You want to assign percentile ranks (e.g., income percentile by implicate)
- You are computing statistics within groups (e.g., groupwise z-scores)
- You need to derive a variable based on implicate-specific thresholds or bins

See Also

scf_update()

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: compute implicate-specific z-scores of income
scf2022 <- scf_update_by_implicate(scf2022, function(df) {
  mu <- mean(df$income, na.rm = TRUE)
  sigma <- sd(df$income, na.rm = TRUE)
  df$z_income <- (df$income - mu) / sigma
  df
})

# Verify new variable exists
head(scf2022$mi_design[[1]]$variables$z_income)

# Do not implement these lines in real analysis: Cleanup for package check
unlink("scf2022.rds", force = TRUE)
```

---

scf_xtab *Cross-Tabulate Two Discrete Variables in Multiply-Imputed SCF Data*

Description

Computes replicate-weighted two-way cross-tabulations of two discrete variables using multiply-imputed SCF data. Estimates cell proportions and standard errors, with optional scaling of proportions by cell, row, or column. Results are pooled across implicates using Rubin's Rules.

Usage

```
scf_xtab(scf, rowvar, colvar, scale = "cell")
```

Arguments

scf	A scf_mi_survey object, typically created by scf_load(). Must include five implicates with replicate weights.
rowvar	A one-sided formula specifying the row variable (e.g., ~edcl).
colvar	A one-sided formula specifying the column variable (e.g., ~racecl).
scale	Character. Proportion basis: "cell" (default), "row", or "col".

Value

A list of class "scf_xtab" with:

results Data frame with one row per cell. Columns: row, col, prop, se, row_share, col_share, rowvar, and colvar.

matrices List of matrices: cell (default proportions), row, col, and se.

imps List of implicate-level cell count tables.

aux List with rowvar and colvar names.

Statistical Notes

Implicate-level tables are created using svytable() on replicate-weighted designs. Proportions are calculated as shares of total population estimates. Variance across implicates is used to estimate uncertainty. Rubin's Rules are applied in simplified form.

For technical details on pooling logic, see scf_MIcombine() or the SCF package manual.

Examples

```
# Do not implement these lines in real analysis:
# Use functions `scf_download()` and `scf_load()`
td  <- tempdir()
src <- system.file("extdata", "scf2022_mock_raw.rds", package = "scf")
file.copy(src, file.path(td, "scf2022.rds"), overwrite = TRUE)
scf2022 <- scf_load(2022, data_directory = td)

# Example for real analysis: Cross-tabulate ownership by education
scf_xtab(scf2022, ~own, ~edcl, scale = "row")

# Do not implement these lines in real analysis: Cleanup for package check
unlink(file.path(td, "scf2022.rds"), force = TRUE)
rm(scf2022)
```

Index