Analyzing the Survey of Consumer Finances with scf

Joseph N. Cohen
City University of New York, Queens College
65-30 Kissena Boulevard
Queens, New York 11367
joseph.cohen@qc.cuny.edu
https://jncohen.commons.gc.cuny.edu/
ORCID: 0000-0002-6197-4453

January 15, 2026

Abstract

The R package scf streamlines analysis of the Survey of Consumer Finances (SCF) public-use microdata. The SCF requires non-standard handling due to its complex sampling design and multiple imputation. The package scf provides a user-friendly interface for acquiring, preparing, analyzing, and visualizating, using survey-tailored procedures through easy-to-use code. This article reviews the SCF structure, details design and logic of the package, and demonstrates utility through applied examples that replicate official estimates.

1 Introduction

This article introduces the R package \pkg{{scf} (Cohen 2025), which simplifies the analysis of the *Survey of Consumer Finances* (SCF) public-use microdata (Federal Reserve Board 2023b). The SCF offers much untapped potential for research on the economic fortunes, strategies, and organization of U.S. households. However, two of the data's particularly powerful features also make it more difficult to analyze: a complex sampling design and its imputation of missing data. Such data requires specialized operations that are unfamiliar to many generalist analysts. Standard statistical functions will almost certainly produce biased estimates. The complexity of handling the data using multiple tools, like \pkg{{survey} (Lumley et al. 2024) and \pkg{{mitools} (Lumley and Scott 2019) can unnecessarily delay or discourage competent generalists from engaging the data altogether. The package `scf` addresses this barrier by scripting the required handling routines in a set of easy-to-use functions to lower the technical barriers to valid SCF analysis and broaden the community of analysts able to work with the data.

This package streamlines the analysis of SCF public-use microdata by scripting the specialized operations required to analyze these data into a user-friendly suite of functions. It builds on `survey` package, wrapping each of the SCF's five imputed sets into replicate-weighted design objects stored in R. Users can then choose among easy functions that script the application of survey functions to each of the implicates, and then combine them via Rubin's Rules (Rubin 1987) or similar appropriate functions to render population estimates. The package includes functions for downloading, wrangling, estimating, and visualizing population statistics. Its workflow and grammar are designed to be simple enough to be used by intermediate-level data analysts without expertise in complex survey analysis or missing data handling.

This paper proceeds as follows: Section 2 briefly describes the SCF and its uses. Section 3 describes the methodological challenges involved in analyzing SCF microdata. Section 4 describes the `scf` package. Section 5 reviews the `scf` package's suite of commands and replicates officially published estimates. Section 6 reviews the strengths of the `scf` universe of functions versus alternatives. Section 7 ends with some technical notes, installation instructions, and a summary.

Note: Upper-case "SCF" refers to the dataset; lower-case monospaced `scf` denotes the R package.

2 The survey

The *Survey of Consumer Finances* (SCF) is a triennial survey of U.S. household finances conducted by the Federal Reserve. Each wave includes several thousand households (over 4,400 in 2022) and collects detailed information on assets, debts, income, expenditures, demographics, and financial behaviors. The survey uses a dual-frame design combining a nationally representative area-probability sample (Davern et al. 2024)[1] with a special oversample of high-wealth households identified through IRS records. These features allow the SCF to support high-quality population estimates and detailed study of financially atypical but economically consequential households.

The same features that make the SCF uniquely valuable also complicate its analysis. Its complex structure, involving stratification, clustering, and replicate-weight variance estimation, requires analysts to use specialized tools (Rust and Rao 1996; Lumley 2004, 2011). In addition, the SCF handles item nonresponse through multiple imputation, distributing five implicates that represent different stochastic draws from the imputation model (Kennickell 2017; Little and Rubin 2019). Valid analysis therefore requires estimating statistics separately within each implicate, combining results appropriately, and incorporating replicate weights to compute correct standard errors.

In practice, these requirements impose substantial burdens on applied researchers. Correct workflows typically involve coordinating several packages like `survey` and `mitools`, along with dozens of lines of custom code to specify replicate-weighted designs, iterate over implicates, and pool estimates. Small implementation errors, such as misspecifying replicate weights or pooling variances incorrectly, can produce biased or irreproducible results. As a consequence, capable analysts with strong substantive expertise often face barriers to engaging the SCF microdata.

The `scf` package is designed to eliminate these barriers. It programmatically wraps each imputed dataset into a correctly specified replicate-weighted design object and automates implicate-level estimation and pooling. This ensures that point estimates and standard errors produced by `scf` functions adhere to the SCF's design without requiring manual specification or iteration from the user. By formalizing best practices into a transparent and reproducible workflow, `scf` lowers the technical threshold for analyzing the SCF responsibly and makes the survey accessible to a broader community of researchers in the R ecosystem.

[1] With an overweight for geographic areas with relatively high proportions of Black-headed households; see Thompson and Suarez (2017).

3 Package design

This section outlines the structure and logic of the `scf` package, including its design principles, core data object, estimation flow, and software dependencies.

3.1 Design philosophy

The `scf` package supports valid, reproducible, and transparent analysis of *Survey of Consumer Finances* (SCF) microdata. It enables users to analyze SCF data without specialized skills. The package aims for:

- **Rigor:** Estimators conform to best practices for multiply imputed, replicate-weighted complex survey data.
- **Accessibility:** The package enables responsible use of the SCF even for users who are still learning the statistical underpinnings of complex survey analysis and missing data handling. Its grammar targets intermediate-level R analysts.
- **Transparency:** The structure of the analysis is not hidden. Users can easily extract implicate-level design objects, raw data, or results. This feature supports the use of the package and data in teaching survey analysis and missing data handling, as well as household finance topics.

By formalizing a reproducible, didactic workflow, `scf` reduces the risk of misanalysis and lowers the entry cost of working with SCF data.

3.2 Core object

All `scf` operations revolve around a central object class, `scf_mi_survey`, which represents the SCF microdata for a given survey year. The object is a multilevel list that contains:

- `mi_design` A list of five `survey::svrepdesign` objects, one per implicate, encoding the SCF's replicate-weighted survey structure.
- `year` The SCF survey year (e.g., 2022).
- `n_households` The number of U.S. households represented by the SCF sample that year.

This structure enables both functional automation (via wrapper functions) and user-side transparency. Users can always access individual implicates directly via `object$mi_design[[i]]` and apply any survey-compatible function for advanced diagnostics or custom modeling.

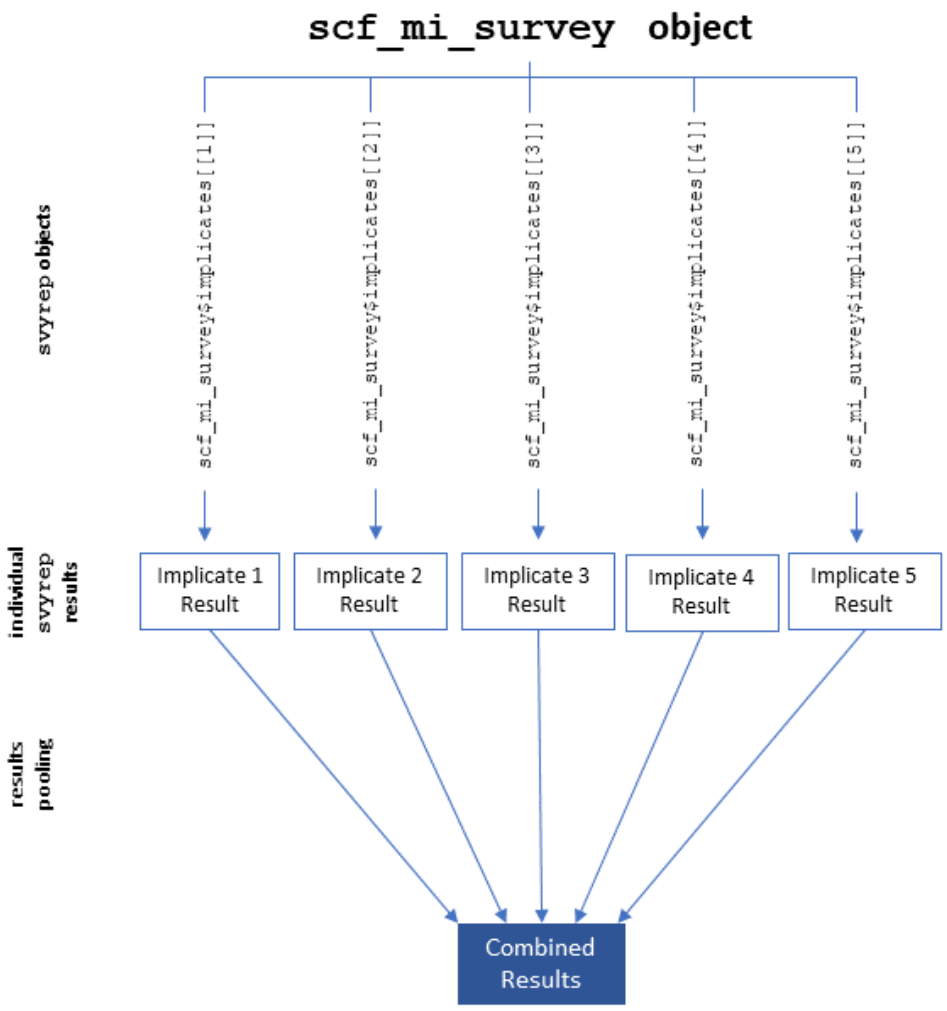

Figure 1: Architecture of the 'scf' Package Data Object and Workflow

3.3 Estimation and pooling logic

The `scf` package's estimation and pooling logic is summarized in Figure 1:

The `scf_mi_survey` is a complex data list that includes one encoded survey design object for each of the SCF's imputed data sets, stored as `scf2022$mi_design[[i]]`. The `scf` package encodes these survey design objects at the outset of the recommended workflow, using `survey::svrepdesign()` wrapped in the functions `scf_design()` via `scf_load()`. For any given `scf` function, an operation in the `survey` package is applied to each of the five implicates.[2] The five separate results are generally then recombined into a single

[2]For example, to calculate the means of a variable, `scf_mean()` applies the `survey::svymean()` function to each implicate, using the replicate weights and the main sampling weight. Likewise, `scf_ols()` uses `survey::svyglm()` to fit a linear regression model to each implicate.

combined parameter estimate using Rubin's Rules (see above) or averaged directly (for percentiles and proportions). The final result is returned as a list containing the pooled parameter estimate, its standard error, and any additional information needed for inference (e.g., confidence intervals).

3.4 Dependencies

The `scf` package depends primarily on `survey` package (Lumley et al. 2024), which provides the underlying infrastructure for complex survey analysis in R. All estimation and design logic in `scf` is built on `survey::svrepdesign()` and related methods. The package avoids additional dependencies such as `mice` opting instead to implement MI logic internally for clarity and compatibility.

4 Functionality overview

The `scf` package provides a suite of functions to support end-to-end analysis of SCF microdata. Its API is organized into logical categories—data infrastructure, wrangling, description, inference, modeling, visualization, and utility—each tailored to address a core analytical need. Functions abstract away the complexities of multiple imputation and replicate weighting while maintaining access to lower-level components for users who wish to customize or extend the analysis.

Table 1 lists the exported functions in the `scf` package, grouped by usage domain.

Table 1: Functions in the `scf` package

Function	Category	Description
scf_download()	Infrastructure	Download and prepare raw SCF data
scf_load()	Infrastructure	Load SCF data and construct MI survey object
scf_design()	Infrastructure	Build scf_mi_survey object from implicate designs
scf_update()	Wrangling	Create and change variables in SCF data object
scf_subset()	Wrangling	Filter SCF data using logical expressions
scf_mean()	Descriptive	Estimate population means with MI pooling
scf_median()	Descriptive	Estimate medians via implicate averaging
scf_percentile()	Descriptive	Estimate user-specified percentiles
scf_freq()	Descriptive	Tabulate category proportions, optionally by group
scf_xtab()	Descriptive	Two-way cross-tabulations for discrete variables
scf_corr()	Descriptive	Pearson correlation with pooled estimates
scf_ttest()	Inference	T-test for group or population mean differences
scf_prop_test()	Inference	Test population proportions (one- or two-sample)
scf_ols()	Modeling	Ordinary least squares regression with Rubin pooling
scf_logit()	Modeling	Logistic regression with optional odds ratio output
scf_glm()	Modeling	Generalized linear models with a user-specified family
scf_MIcombine()	Support	Apply Rubin's Rules for pooled estimates
scf_implicates()	Support	Extract implicate-level results from scf_* outputs
scf_regtable()	Output	Format and display regression output for publication
scf_plot_dbar()	Visualization	Bar chart for discrete variables
scf_plot_cbar()	Visualization	Bar chart for continuous-by-discrete relationships
scf_plot_bbar()	Visualization	Stacked bar chart of two discrete variables
scf_plot_dist()	Visualization	Distribution plot for discrete or continuous variables
scf_plot_hist()	Visualization	Histogram for continuous variables
scf_plot_hex()	Visualization	Hexbin plot for two continuous variables
scf_plot_smooth()	Visualization	Smoothed density plot for a continuous variable
scf_theme()	Visualization	Default plot theme for SCF graphics

Each function handles the SCF's complex design internally, ensuring valid inference by default. Infrastructure and wrangling tools support data preparation; estimation functions return pooled results using Rubin's Rules or implicate-level averaging where appropriate; plotting and output tools provide ready-to-publish summaries. All functions operate on `scf_mi_survey` objects unless otherwise noted.

5 Examples and validation

This section demonstrates the package's workflow and functionality by replicating official population statistic estimates published by the U.S. Federal Reserve. The Federal Reserve generally uses a non-public version of its dataset in creating statistical reports, but it releases official estimates derived from the public-use microdata in an Excel work-

book distributed on the SCF website (Federal Reserve Board 2023a). Below, official estimates from that workbook are cited, mentioning the table and cell range to which the `scf` package-generated results match.

5.1 Data wrangling

5.1.1 Data acquisition and loading

Users can download the SCF data and load it into R using the `scf_download()` and `scf_load()` functions. Once the data are loaded, users can apply various functions to analyze the data. The following syntax downloads and loads the object associated with the 2022 data:

```
scf_download(2022)
scf2022 <- scf_load(2022)
```

5.1.2 Variable creation or transformation

To create or change variable values, use the `scf_update()` function. Below, a logical variable named "senior" is created, which is TRUE if the household head is aged 65 or older:

```
scf2022 <- scf_update(scf2022, senior = age >= 65)
```

5.2 Describing estimated population distributions

The suite contains a range of functions designed to extract descriptors of the estimated population distributions suggested by the microdata.

5.2.1 Univariate continuous distributions

Below, the estimated population mean household income for 2022 is reproduced.[3] Users can also try `scf_mean()` and `scf_percentile()`.

```
scf_mean(scf2022, ~income)
```

```
## Multiply-Imputed, Replicate-Weighted Mean Estimate
##
##  variable estimate      se      min      max
##    income 141389.9 5006.27 138806.8 143615.9
```

5.2.2 Univariate discrete distributions

Univariate discrete distributions can be described as frequency tables using `scf_freq()`.[4]

```
# Wrangling raw data
scf2022 <- scf_update(scf2022,
            edu1 = ifelse(x5931 == 0, NA,
                    ifelse(x5931 %in% c(-1, 1:7), 1,
                     ifelse(x5931 %in% c(8), 2,
                      ifelse(x5931 %in% c(9:11), 3,
                       ifelse(x5931 %in% c(12:15),4, NA)))))
           )
scf2022 <- scf_update(scf2022,
                      edu1 = factor(edu1,
                                    levels = 1:4,
                                    labels = c("<HS", "HS",
                                               "Some College", "College+")))

# Producing a frequency table of population estimates
scf_freq(scf2022, ~edu1)
```

```
## SCF Frequency Table (Pooled Results)
```

[3] The result reproduces official estimates of mean household income in 2022 reported in the sheet "Table 1 01-22", Cell AE7.

[4] Result reproduces official estimates of the distribution of household head's educational attainment levels from Sheet "Table 01 01-22", cell range AG34:AG37. The ordinal educational scale is constructed from the raw education variable `x5931`.

```
##
##  group     category proportion se_proportion
##     NA          <HS    9.16622     0.4699786
##     NA           HS   23.65321     0.6682255
##     NA Some College   27.08191     0.6808270
##     NA     College+   40.09866     0.9710413
```

5.2.3 Bivariate discrete-discrete

The `scf_xtab()` function produces cross-tabulations of two discrete variables, allowing users to see the distribution of one variable across the categories of another.[5]

```
# Bound and log net worth
scf2022 <- scf_update(scf2022,
                      under35 = factor(ifelse(age < 35, 1, 0),
                                       levels = 0:1,
                                       labels = c("Over 35", "Under 35")))

# Car ownership and under 35 year-old head
scf_xtab(scf2022, ~own, ~under35, scale = "col")
```

```
## SCF Cross-Tabulation
## Row Variable: own | Column Variable: under35
## Displayed as: col proportions (percent)
##
##        under35: Over 35 under35: Under 35
## own: 0            12.30             19.37
## own: 1            87.70             80.63
```

[5]This result nearly replicates "Table 9 22 %s & medians" cell: B19. Official report is 80.6932, and our estimate is 80.6332.

5.2.4 Bivariate discrete-continuous

Use the "by" argument in functions like `scf_mean()` to estimate the mean population score across the categories of a discrete variable.[6]

```
scf_mean(scf2022, ~networth, by = ~under35)
```

```
## Multiply-Imputed, Replicate-Weighted Mean Estimate
##
##     group variable  estimate       se       min       max
##   Over 35 networth 1278931.3 41136.74 1258541.6 1299278.7
##  Under 35 networth  183375.9 22800.59  176700.9  186269.5
```

5.2.5 Bivariate continuous-continuous

The `scf_corr()` function estimates the Pearson correlation between two continuous variables, returning a pooled estimate with standard error. There are no correlations reported in our replication data, but we can use this combination to demonstrate the package's visualization capabilities.

5.3 Data visualization

The above-cited workbook from which we are replicating official statistics does not include population-level correlation estimates, which would be estimated with the function `scf_corr()`. However, for the purpose of including both a visualization and completing the typology of basic statistics, we include `scf_plot_hex()`. Below, the function produces a hexbin plo t that visualizes the relationship between net worth (logged) and age of the household head.

[6]The result replicates official estimates of mean net worth of under-35 headed households in sheet "Table 4", cell Y18.

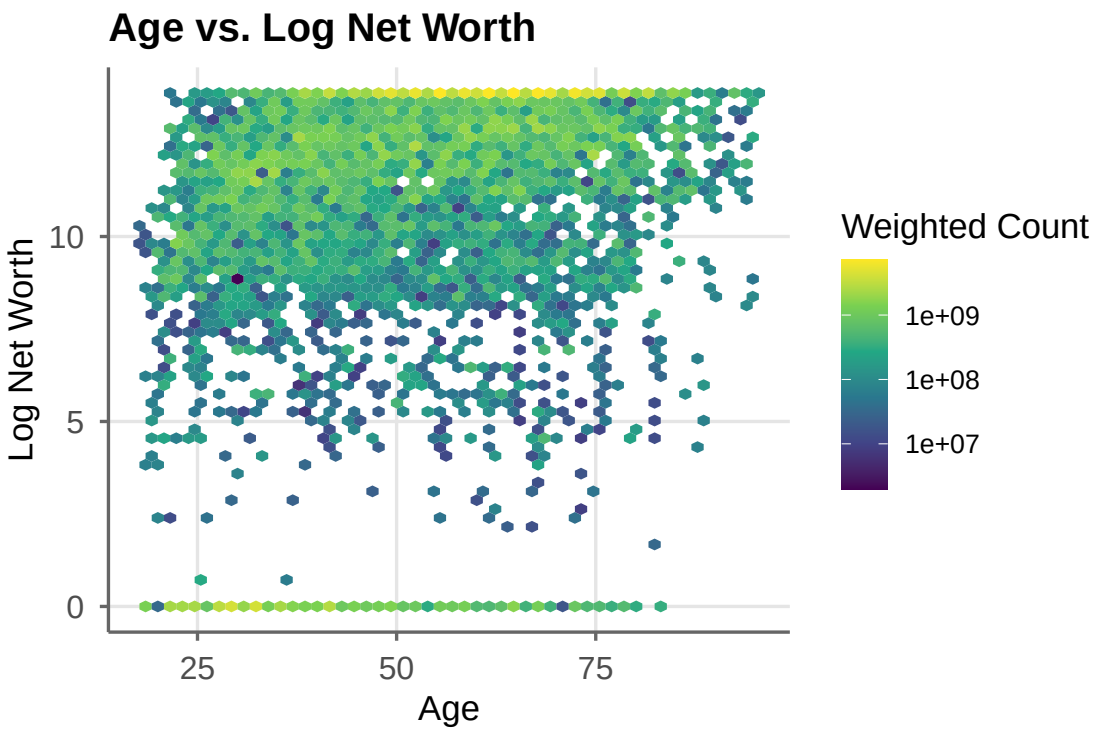

Figure 2: Hexbin Plot of Household Age and Net Worth

```
scf2022 <- scf_update(scf2022,
                      networth_bound = pmax(1, pmin(networth, 1e6)),
                      log_networth   = log(networth_bound)
)
```

```
scf_plot_hex(
  scf2022, ~age, ~log_networth,
  title = "Age vs. Log Net Worth",
  xlab = "Age",
  ylab = "Log Net Worth"
)
```

5.4 Regression analysis

Below, we calculate an OLS model predicting logged net worth on a binary variable demarcating households with heads over 65 years *versus* under 65. Note the prior operations to top- and bottom-code, then log-transform, the outcome. Note that `scf_update()`

processes operations given earlier in the same command.

```
model1 <- scf_ols(scf2022, log_networth ~ senior)
```

The function `scf_regtable()` formats regression results for publication. Below, we display the results of the above model with three decimal places and output to console, as opposed to in LaTeX format or as a CSV file.

```
# Show results using function scf_regtable()
scf_regtable(model1, digits = 3, output = ("console"))
```

```
## (Intercept)   10.547*** (0.056)
## seniorTRUE    1.298*** (0.093)
## N             4595
## R2            0.024
## AIC           8363
```

Non-seniors have estimated mean net worth $38,052.54; seniors $139,411. Model fit registers a mean R-Squared 0.02.

6 Comparison to existing tools

A standard method for analyzing SCF data in R involves using Lumley *et al.*'s (2024) `survey` package, implemented in a workflow defined by Damico (2025). The `survey` package is the industry standard in complex survey analysis. For years, Damico's code repository had been recommended in Federal Reserve technical notes as guidance for R users who wanted to use the data. Both are invaluable resources for using and teaching people how to use a wide range of federal survey data, including the SCF.

The SCF's combination of replicate weights and multiple imputation, however, requires more elaborate scripting than most federal surveys. This complexity can induce coding errors and discourages generalists from using the data. The `scf` package streamlines the Damico-style workflow by automating these steps in a small set of functions that integrate design construction, implicate iteration, and pooling. This lowers the technical barrier to valid analysis while preserving the underlying methodology.

A comparison of the workflow required to get the median household income in 2007 using both methods illustrates the difference. Both workflows yield identical statistics and standard errors; the difference lies in reduced coding surface and error exposure, though at some processing time costs.

6.0.1 Traditional method

The traditional approach, as defined in Damico (2025)), requires dozens of lines of code to make the same calculation. Damico's code is replicated over more than 100 lines in Appendix A. Note that this process would be longer for data visualizations.

The operation took 74.88 seconds in total on the test run performed in this paper.

6.0.2 Via `scf` package

Both approaches yield identical medians and standard errors. The `scf` package performs this internally, reducing dozens of lines of code to a simple call.

```
# Download Data
scf_download(2007)
```

```
## [1] "scf2007.rds"
```

```
# Load Data into R
scf2007 <- scf_load(2007)

# Calculate Median of Income
scf_median(scf2007, ~income)
```

```
## Multiply-Imputed Median Estimate
##
##  variable quantile estimate       se      min      max
##    income      0.5 67401.63 1371.688 66224.31 67695.96
```

The operation took 98.08 seconds in total. This is 1.31 times slower than the manual method. While this slower percentage-wise, the absolute time difference is likely minor for most users, and the reduced cognitive load and error risk may ultimately result in less total analysis time.

Table 2: Benchmarking comparison of traditional SCF workflow and the `scf` package.

Method	Lines_of_Code	Runtime_Seconds	Estimates_Agree
Traditional SCF workflow	78	74.88	Yes
scf package workflow	4	98.08	Yes

7 Additional notes

7.1 Technical notes

The `scf` package is implemented entirely in R and depends only on established CRAN infrastructure. It builds on the `survey` package (Lumley et al. 2024) for complexsurvey estimation, including replicateweight support and designbased inference. Optional graphics use `ggplot2` (Wickham et al. 2025); these are not required for estimation. Pooling and variance calculations for multiply imputed, replicateweighted data follow the public SCF technical documentation and are implemented directly within the package.

7.2 Reproducibility and availability

The `scf` package is publicly available on CRAN. To install:

```
install.packages("scf")
```

The package includes all code used to generate the applied examples in Section 6 of this article. The examples are based on the 2022 wave of the Survey of Consumer Finances public-use microdata, available directly from the Federal Reserve's website. A basic package vignette is included to illustrate core usage patterns; a more detailed vignette reproducing official SCF statistics may be added in a future release.

7.3 Summary

The `scf` package makes Survey of Consumer Finances microdata accessible to generalist quantitative analysts. In so doing, it helps give household finance researchers easier access to a powerful and high-quality data set. The package is designed to encode best practices for working with complex survey data, including replicate-weighted estimation, multiple imputation, and robust inference. It is built on the `survey` package, which is the industry standard tool for complex survey analysis in R.

8 Appendix: Manual Implementation of SCF analysis without scf package

```
scf_dta_import <-
    function(this_url){
        this_tf <- tempfile()
        download.file(this_url , this_tf , mode = 'wb')
        this_tbl <- read_dta(this_tf)
        this_df <- data.frame(this_tbl)
        file.remove(this_tf)
        names(this_df) <- tolower(names(this_df))
        this_df
    }

scf_df <- scf_dta_import(
  "https://www.federalreserve.gov/econres/files/scf2007s.zip")
ext_df <- scf_dta_import(
  "https://www.federalreserve.gov/econres/files/scfp2007s.zip")
scf_rw_df <- scf_dta_import(
  "https://www.federalreserve.gov/econres/files/scf2007rw1s.zip")

# Checks of data dimensions, per Damico (2025)
stopifnot(nrow(scf_df) == nrow(scf_rw_df) * 5)
stopifnot(nrow(scf_df) == nrow(ext_df))
stopifnot(all(sort(intersect(names(scf_df),
                             names(ext_df))) == c('y1' , 'yy1')))
stopifnot(all(sort(intersect(names(scf_df),
                             names(scf_rw_df))) == c('y1' , 'yy1')))
stopifnot(all(sort(intersect(names(ext_df),
                             names(scf_rw_df))) == c('y1' , 'yy1')))

# Removing implicate identifier and adding column of 5s for weighting
scf_rw_df[ , 'y1' ] <- NULL
scf_df[ , 'five' ] <- 5
```

```r
# Construct multiply-imputed complex survey design, per Damico (2025)
s1_df <- scf_df[ str_sub(scf_df[ , 'y1' ] , -1 , -1) == 1 , ]
s2_df <- scf_df[ str_sub(scf_df[ , 'y1' ] , -1 , -1) == 2 , ]
s3_df <- scf_df[ str_sub(scf_df[ , 'y1' ] , -1 , -1) == 3 , ]
s4_df <- scf_df[ str_sub(scf_df[ , 'y1' ] , -1 , -1) == 4 , ]
s5_df <- scf_df[ str_sub(scf_df[ , 'y1' ] , -1 , -1) == 5 , ]

# Combine to list
scf_imp <- list(s1_df , s2_df , s3_df , s4_df , s5_df)
scf_list <- lapply(scf_imp , merge , ext_df)

# Replace missing values in replicate weights with zero
scf_rw_df[ is.na(scf_rw_df) ] <- 0
scf_rw_df[ , paste0('wgt' , 1:999) ] <-
    scf_rw_df[ , paste0('wt1b' , 1:999) ] * scf_rw_df[ , paste0('mm' , 1:999) ]
scf_rw_df <- scf_rw_df[ , c('yy1' , paste0('wgt' , 1:999)) ]

# Sort table by unique identifiers, per Damico
scf_list <- lapply(scf_list , function(w) w[ order(w[ , 'yy1' ]) , ])
scf_rw_df <- scf_rw_df[ order(scf_rw_df[ , 'yy1' ]) , ]

# Define function to combine results: From Damico (2025)
scf_MIcombine <-
    function (results, variances, call = sys.call(), df.complete = Inf, ...) {
        m <- length(results)
        oldcall <- attr(results, "call")
        if (missing(variances)) {
            variances <- suppressWarnings(lapply(results, vcov))
            results <- lapply(results, coef)
        }
        vbar <- variances[[1]]
        cbar <- results[[1]]
        for (i in 2:m) {
            cbar <- cbar + results[[i]]
            # MODIFICATION:
            # vbar <- vbar + variances[[i]]
```

```
        }
        cbar <- cbar/m
        # MODIFICATION:
        # vbar <- vbar/m
        evar <- var(do.call("rbind", results))
        r <- (1 + 1/m) * evar/vbar
        df <- (m - 1) * (1 + 1/r)^2
        if (is.matrix(df)) df <- diag(df)
        if (is.finite(df.complete)) {
            dfobs <- ((df.complete + 1)/(df.complete + 3)) * df.complete *
            vbar/(vbar + evar)
            if (is.matrix(dfobs)) dfobs <- diag(dfobs)
            df <- 1/(1/dfobs + 1/df)
        }
        if (is.matrix(r)) r <- diag(r)
        rval <- list(coefficients = cbar, variance = vbar + evar *
        (m + 1)/m, call = c(oldcall, call), nimp = m, df = df,
        missinfo = (r + 2/(df + 3))/(r + 1))
        class(rval) <- "MIresult"
        rval
    }

# Define Lumley object: From Damico (2025)
scf_design <-
    svrepdesign(
        weights = ~wgt ,
        repweights = scf_rw_df[ , -1 ] ,
        data = imputationList(scf_list) ,
        scale = 1 ,
        rscales = rep(1 / 998 , 999) ,
        mse = FALSE ,
        type = "other" ,
        combined.weights = TRUE
   )

# Calculate median income
```

```
scf_MIcombine(with(scf_design ,
    svyquantile(
        ~ income ,
        0.5 , se = TRUE , interval.type = 'quantile'
)))
```

The operation took 74.881 seconds in total.

9 Computational details

```
## R version 4.5.1 (2025-06-13 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 11 x64 (build 26200)
##
## Matrix products: default
##   LAPACK version 3.12.1
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] mitools_2.4    survey_4.4-8   survival_3.8-3 Matrix_1.7-4   haven_2.5.5
##  [6] tibble_3.3.0   stringr_1.6.0  dplyr_1.1.4    scales_1.4.0   ggplot2_4.0.1
## [11] knitr_1.51     scf_1.0.5
##
```

```
## loaded via a namespace (and not attached):
##  [1] generics_0.1.4     stringi_1.8.7      lattice_0.22-7     hms_1.1.4
##  [5] digest_0.6.39      magrittr_2.0.4     evaluate_1.0.5     RColorBrewer_1.1-3
##  [9] fastmap_1.2.0      DBI_1.2.3          httr_1.4.7         viridisLite_0.4.2
## [13] cli_3.6.5          rlang_1.1.6        splines_4.5.1      withr_3.0.2
## [17] yaml_2.3.12        otel_0.2.0         tools_4.5.1        tzdb_0.5.0
## [21] forcats_1.0.1      curl_7.0.0         vctrs_0.6.5        R6_2.6.1
## [25] lifecycle_1.0.4    pkgconfig_2.0.3    pillar_1.11.1      hexbin_1.28.5
## [29] gtable_0.3.6       glue_1.8.0         Rcpp_1.1.0         xfun_0.55
## [33] tidyselect_1.2.1   rstudioapi_0.17.1  farver_2.1.2       htmltools_0.5.9
## [37] rmarkdown_2.30     labeling_0.4.3     readr_2.1.6        compiler_4.5.1
## [41] S7_0.2.1
```

10 Bibliography

Cohen, Joseph N. 2025. *scf: Survey of Consumer Finances Tools*. https://CRAN.R-project.org/package=scf.

Damico, Anthony. 2025. *Ajdamico/Asdfree*. GitHub repository. https://github.com/ajdamico/asdfree.

Davern, Michael, Rene Bautista, Jeremy Freese, Pamela Herd, and Stephen L. Morgan. 2024. *General Social Survey, 1972–2024*. Edited by NORC at the University of Chicago. NORC at the University of Chicago. https://gssdataexplorer.norc.org.

Federal Reserve Board. 2023a. *Historic Tables: Estimates in Nominal Dollars*. https://www.federalreserve.gov/econres/files/scf2022_tables_public_nominal_historical.xlsx.

Federal Reserve Board. 2023b. *Survey of Consumer Finances*. Survey data. https://doi.org/10.17016/8799.

Kennickell, Arthur B. 2017. "Multiple Imputation in the Survey of Consumer Finances." *Statistical Journal of the IAOS* 33 (1): 143–51. https://doi.org/10.3233/SJI-160278.

Little, Roderick J. A., and Donald B. Rubin. 2019. *Statistical Analysis with Missing Data*. 3rd ed. John Wiley & Sons. https://doi.org/10.1002/9781119482260.

Lumley, Thomas. 2004. "Analysis of Complex Survey Samples." *Journal of Statistical Software* 9: 1–19. https://doi.org/10.18637/jss.v009.i08.

Lumley, Thomas. 2011. *Complex Surveys: A Guide to Analysis Using R*. John Wiley & Sons. https://doi.org/10.1002/9780470580066.

Lumley, Thomas, Pei Gao, and Brian Schneider. 2024. *survey: Analysis of Complex Survey Samples*. https://CRAN.R-project.org/package=survey.

Lumley, Thomas, and Allen Scott. 2019. *mitools: Tools for Multiple Imputation of Missing Data*. https://CRAN.R-project.org/package=mitools.

Rubin, Donald B. 1987. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons. https://doi.org/10.1002/9780470316696.

Rust, Keith F., and J. N. K. Rao. 1996. "Variance Estimation for Complex Surveys Using Replication Techniques." *Statistical Methods in Medical Research* 5 (3): 283–310. https://doi.org/10.1177/096228029600500305.

Thompson, Jeffrey P., and Gustavo A. Suarez. 2017. *Updating the Racial Wealth Gap*. http://dx.doi.org/10.17016/FEDS.2015.076r1.

Wickham, Hadley, Winston Chang, Lionel Henry, et al. 2025. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggplot2.