The Survey Statistician, 2025, Vol. 91, 47-55

Book and Software Review

Extensions of the *survey* package in R providing a new user interface and resampling methods for especially complex sample designs

Benjamin Schneider[1]

[1] Westat, BenjaminSchneider@westat.com

Abstract

This review covers two extensions of the **survey** package in R that can help analyse and process data from complex surveys. We first provide an overview of the extension **srvyr** that allows users to write code for survey data analysis in the style of the tidyverse, an influential suite of R packages for general-purpose data analysis. We then review a recent extension package **svrep** that allows R users to implement new and varied replication methods such as the generalized bootstrap. We conclude by highlighting features of these packages which are especially relevant for producers of official statistics.

Keywords: software, R, resampling, two-phase designs

1 Introduction

The **survey** package in R (Lumley, 2004) is an influential software package for the analysis of complex survey data with features such as sampling weights or stratification. Since Prof. Thomas Lumley first published the **survey** package twenty years ago, it has been the predominant open source software package for complex survey data analysis, serving as a free and transparent competitor to closed source software packages for survey data analysis such as WesVar, SUDAAN, or routines provided in Stata, SAS, and SPSS. As an open source package with general-purpose functionality for complex survey data analysis, the **survey** package provides a valuable foundation for other more specialized R packages that implement particular models or analyses. Extensions of the **survey** package are able to rely on the **survey** package as an interface for organizing metadata required for analysis (e.g., sampling weights and strata identifiers) and as a computational engine with tested and optimized routines for estimation. This review focuses on two recent general-purpose extensions of the **survey** package, with the first extension enhancing the **survey** package's interface and the second expanding upon its statistical routines to cover a broader range of sample designs and weighting methods.

2 srvyr: A Fresh User Interface Familiar to tidyverse Users

In a recent issue of the *The Survey Statistician*, the **survey** package's creator Thomas Lumley highlighted that one of the package's three main goals is to "allow non-specialist users to display, analyse, and model complex surveys with only the necessary changes from their usual data analysis practice" (Lumley, 2023). To that end, the package published in 2003 included a user interface that

adhered to the style and conventions of the data analysis functionality included in “base R” (that is, the functionality available in R without having to install any specific additional packages).

However, in the decades since the **survey** package was first published, R users have increasingly embraced the tidyverse suite of R packages, which provide an alternative R interface for data analysis instead of base R. The tidyverse suite of packages provides a consistent and powerful framework for data analysis that prioritizes readability and consistency (Wickham et al., 2019).

The **srvyr** package in R (Freedman Ellis and Schneider, 2016) provides a tidyverse-style user interface for the **survey** package, ensuring that R users can write analysis code in whichever interface they prefer to use, while relying on the **survey** package to handle the underlying statistical computations. As a result of the Tidyverse’s widespread adoption among R users, the **srvyr** interface has been increasingly emphasized in recent texts. For instance, this interface is emphasized in the recent introductory textbook “Exploring Complex Survey Data Analysis Using R: A Tidy Introduction with srvyr and survey” (Zimmer, 2024) and in the introductory textbook “Data Visualization”(Healy, 2019).

We’ll highlight a few of the key features that distinguish the base R interface and the **srvyr** interface. To illustrate these features, we’ll demonstrate an analysis of data from the 2023 U.S. National Health Interview Survey (NHIS) obtained from the IPUMS, an integrated collection of microdata and tabulations from official statistics programs of the U.S. and other countries (Blewett et al., 2024). The NHIS sample is obtained through a stratified, multistage sample design. With the **survey** and **srvyr** packages, the survey data as well as important design information (weights, strata identifiers, etc.) are all combined into a *survey design object* which can then be used for analysis. The interface for creating a survey design object is quite similar in both packages.

```
library(survey)
library(srvyr)
library(tidyverse)

nhis_svy <- as_survey_design(
  .data   = nhis_data,  # Dataset containing all variables
  weights = SAMPWEIGHT, # Sampling weights
  strata  = STRATA,     # Strata identifiers
  ids     = PSU,        # Sampling unit identifiers
  nest    = TRUE        # Sampling units are nested in strata
)
```

The survey design object can then be analysed using functions from either **survey** or **srvyr**. We can filter the data and apply transformations using either interface, and for simple analyses the two interfaces look quite similar. The example code below subsets the data to adults and derives a new variable for Body Mass Index (BMI) where missing values have been converted from a specific code (996) to an explicit missing value (*NA*).

```
nhis_adults_svy <- nhis_svy |> filter(AGE >= 18) |>
  mutate(BMI = ifelse(BMICALC == 996, NA, BMICALC))
```

In the example below, we use the **survey** package to estimate the mean BMI of U.S. adults based on self-reported height and weight data. The output from the **survey** package automatically includes a standard error estimate, and we can later add additional summaries such as a 95% confidence interval.

```
estimated_mean <- svymean(x = ~ BMI, design = nhis_adults_svy, na.rm = TR
UE)
print(estimated_mean)
```

```
      mean    SE
BMI 28.031 0.048
```

```
confint(estimated_mean, level = 0.95, df = degf(nhis_adults_svy))
```

```
       2.5 %   97.5 %
BMI 27.93664 28.12498
```

We can compute the same estimates using the **srvyr** interface. With **srvyr**, summaries are computed using a *summarise()* function. Inside the *summarise()* function, the user can request several summaries such as weighted means or totals, using functions with names such as *survey_mean()*, *survey_total()*, or *survey_quantile()*. These *survey_*()* functions can be used to obtain not only a point estimate, but a standard error, confidence interval, and coefficient of variation. A benefit of using the *summarise()* syntax is that multiple summaries can be produced all at once, for example in the following code which computes the mean and median BMI, as well as the number of sample observations. The output of the *summarise()* function is always a data frame (in R jargon, a dataset with rows of observations and columns of variables).

```
nhis_adults_svy |>
  summarise(
    MEAN_BMI   = survey_mean(BMI, na.rm = TRUE, vartype = c("se", "ci")),
    MEDIAN_BMI = survey_median(BMI, na.rm = TRUE)
  )
```

```
# A tibble: 1 × 6
  MEAN_BMI MEAN_BMI_se MEAN_BMI_low MEAN_BMI_upp MEDIAN_BMI MEDIAN_BMI_se
     <dbl>       <dbl>        <dbl>        <dbl>      <dbl>         <dbl>
1     28.0      0.0480         27.9         28.1       27.4        0.0255
```

The **srvyr** syntax is especially valuable when users want to produce grouped summaries. The user can simply take the code above and add a single line of code specifying grouping variables to use.

```
nhis_adults_svy |>
  filter(!is.na(BMI)) |>
  group_by(REGION, SEX) |>
  summarise(N_OBS      = unweighted(n()),
            MEAN_BMI   = survey_mean(BMI)) |>
  ungroup()
```

```
# A tibble: 8 × 5
  REGION                SEX    N_OBS MEAN_BMI MEAN_BMI_se
  <fct>                 <fct>  <int>    <dbl>       <dbl>
1 Northeast             Male    1953     27.9       0.137
2 Northeast             Female  2277     27.3       0.170
3 North Central/Midwest Male    2713     28.4       0.111
4 North Central/Midwest Female  3187     28.4       0.147
5 South                 Male    4584     28.3       0.101
6 South                 Female  5408     28.3       0.118
7 West                  Male    3161     27.9       0.113
8 West                  Female  3686     27.2       0.126
```

In contrast, the original **survey** interface requires more work to go from producing a single overall population estimate to producing grouped estimates and is limited to only analyzing one type of summary statistic at a time. In the example below, the **survey** package is used to estimate only the population mean by group. To compute the median and the number of unweighted observations as well, the user would have to repeat the call to *svyby()* again.

```
domain_estimates <- svyby(
  design  = nhis_adults_svy |> subset(!is.na(BMI)),
  by      = ~ REGION + SEX,
  formula = ~ BMI,
  FUN     = svymean,
  vartype = c("se", "ci"), covmat = TRUE
)

domain_estimates |> as_data_frame() |> print()
```

```
# A tibble: 8 × 6
  REGION                SEX      BMI    se  ci_l  ci_u
  <fct>                 <fct>  <dbl> <dbl> <dbl> <dbl>
1 Northeast             Male    27.9 0.137  27.7  28.2
2 North Central/Midwest Male    28.4 0.111  28.2  28.7
3 South                 Male    28.3 0.101  28.1  28.5
4 West                  Male    27.9 0.113  27.7  28.1
5 Northeast             Female  27.3 0.170  27.0  27.7
6 North Central/Midwest Female  28.4 0.147  28.1  28.7
7 South                 Female  28.3 0.118  28.1  28.5
8 West                  Female  27.2 0.126  26.9  27.4
```

The code examples above illustrate some of the key distinguishing features of the **srvyr** interface and the **survey** package's original interface for data analysis. The **srvyr** interface generally avoids the use of formulas: for example, to group the data, users can avoid typing *by = ~ REGION + SEX* and instead write *group_by(REGION, SEX)*. The **srvyr** interface generally borrows functions from the tidyverse (particularly the **dplyr** R package) whenever possible, so that the large number of R users who are already comfortable analyzing data with the Tidyverse can easily adapt their usual coding workflows to a complex survey context.

In short, while the **srvyr** package does not expand the kinds of analyses that can be done with the **survey** package in R, it does expand the kinds of workflows and the kinds of users that can easily benefit from using the **survey** package in their data analyses.

3 Using the svrep Package to analyse *Especially* Complex Samples

The **svrep** package (Schneider, 2022), on the other hand, extends the kinds of data analysis that can be done with the **survey** package, in particular when using replication methods for variance estimation. Standard replication methods implemented in the **survey** package and closed-source alternatives such as Stata currently allow analysts to implement some common forms of replication designed to handle the most common complex sample design features. For instance, R and Stata both allow users to analyse stratified cluster samples through delete-one jackknife replication or the Rao-Wu-Yue bootstrap (Rao, Wu, and Yue, 1992). But there are other complex design features which are not well supported by the standard replication methods, including general two-phase sampling, systematic sampling, fine stratification, unequal-probability sampling without replacement (PPSWOR), and the use of newer sampling techniques such as the cube method. The **svrep** package introduces replication methods which are specially designed for these complex features.

The following resampling methods are implemented in **svrep**:

- **The Rao-Wu-Yue-Beaumont (RWYB) bootstrap**, an extension of the Rao-Wu-Yue bootstrap method proposed by Beaumont and Émond (2022). The RWYB bootstrap can be used to analyze data from stratified, multistage samples with arbitrary sampling fractions and the use of PPSWOR sampling or Poisson sampling at one or more stages.
- **The random-groups jackknife**, a replication method that can be more efficient than the bootstrap in certain settings while being more robust than the standard delete-one jackknife when estimating variances for quantiles and other non-smooth statistics. The random groups and replicate weights are created using the approach developed by Valliant, Brick, and Dever (2008).
- **Fay's generalized replication method**, which is a highly general-purpose and efficient replication method proposed by Fay (1989). This method can be used for stratified multistage samples with PPSWOR sampling, as well as more complex sample designs such as samples selected using the cube method (Li et al., 2014).
- **The generalized survey bootstrap**, a bootstrap form of generalized replication which can be used to handle a much wider range of complex design features compared to standard bootstrap methods (Beaumont and Patak, 2012).

The two generalized replication methods listed above greatly expand the kinds of sample designs which can be conveniently analyzed in R.

The **svrep** package's user interface is designed largely as an extension of the **survey** package: this is why the package's name is derived from the shorthand for survey replicates used throughout the **survey** package (e.g., in the **survey** functions *svrepdesign()* and *as.svrepdesign()*). Users can describe their survey data and its complex design features using the familiar interface of the **survey** package (or **srvyr**), and then call functions from **svrep** to create replicate weights. The example R code below illustrates how this approach can be used to create bootstrap replicate weights using the RWYB method. This code uses the small example dataset "mu284" from the **survey** package, which is a two-stage sample with 15 observations, where the variables *id1 and id2 denote sampling unit identifiers at each stage, and the variables n1 and n2 denote population sizes at each stage which are used to compute finite population corrections. The variable y1 is an example variable included in the dataset for univariate analyses.*

```
library(svrep)
# Load an example dataset from a multistage sample, with two stages of SRSWOR
data("mu284", package = 'survey')

# Create a survey design object
multistage_srswor_design <- svydesign(data = mu284,
                                      ids  = ~ id1 + id2,
                                      fpc  = ~  n1  + n2)

# Create replicate weights
bootstrap_rep_design <- multistage_srswor_design |>
  as_bootstrap_design(replicates = 5000, type = "Rao-Wu-Yue-Beaumont")

# Estimate a mean and its variance
multistage_srswor_design |> svymean(x = ~ y1)
     mean     SE
y1 44.353 2.2737
bootstrap_rep_design     |> svymean(x = ~ y1)
     mean     SE
y1 44.353 2.3699
```

Fay's generalized replication method represents an efficient and flexible alternative to the bootstrap. Like the jackknife or balanced repeated replication, Fay's generalized replication method can produce stable variance estimates with a small number of replicates. The generalized replication method is essentially a general-purpose recipe for creating replicate weights based on a specified variance estimator. It works by representing a chosen variance estimator as a quadratic form matrix and decomposing that quadratic form matrix into a set of replicate weights. The **svrep** package lets users easily create generalized replicate weights for a survey in two steps: the user describes their survey design using the **survey** package, then specifies a variance estimator they want to use as the basis for the replicate weights. The example below shows the creation of generalized replicate weights based on the standard variance estimator for stratified multistage SRS designs.

```
gen_rep_design <- as_fays_gen_rep_design(
  design             = multistage_srswor_design,
  variance_estimator = "Stratified Multistage SRS"
)

print(gen_rep_design)
```

```
Call: as_fays_gen_rep_design(design = multistage_srswor_design, variance_
estimator = "Stratified Multistage SRS")
with 16 replicates and MSE variances.
```

```
gen_rep_design |> svymean(x = ~ y1)
```

```
     mean     SE
y1 44.353 2.3132
```

The generalized replication method is valuable for its ability to create replicate weights for especially complex designs. As an example, we can consider the election data included in the **survey** package, which includes a sample of U.S. counties selected using PPSWOR sampling. These example data consist of a sample dataset named *election_pps* with the variable named *p* denoting the sampling probability, along with a corresponding matrix of joint inclusion probabilities named *election_jointprob*. The variable *Bush* contains county-level totals of votes cast for President Bush in the 2004 U.S. presidential election, and the population total of that variable is the national vote total for that candidate. To estimate variances for the estimated population total, the analyst can use the Sen-Yates-Grundy estimator based on joint probabilities. The **survey** package allows users to analyze data using the Sen-Yates-Grundy estimator but does not provide replication methods that can yield comparable variance estimates.

```
data('election', package = 'survey')

# Create survey design object that uses the Yates-Grundy estimator based on joi
nt probabilities
pps_design_yg <- svydesign(
  data     = election_pps,
  id       = ~1,
  fpc      = ~p,
  pps      = ppsmat(election_jointprob),
  variance = "YG"
)

pps_design_yg |> svytotal(x = ~ Bush)
```

```
        total      SE
Bush 64518472 2406526
```

The example code below creates Fay's generalized replication weights and uses them to estimate the sampling variance of an estimated total. The example below demonstrates that the replication-based estimate standard error estimate for totals is exactly the same as the estimate from the Sen-Yates-Grundy estimator.

```
# Create generalized replicate weights and estimate a standard error
yg_gen_rep_design <- as_fays_gen_rep_design(
  design            = pps_design_yg,
  variance_estimator = "Yates-Grundy"
)
yg_gen_rep_design |> svytotal(x = ~ Bush)
```

```
        total      SE
Bush 64518472 2406526
```

The generalized survey bootstrap is a closely related replication method that introduces additional bootstrap randomness. In expectation, variance estimates for totals and other linear statistics estimated using the generalized survey bootstrap replicates will have the same value as the target variance estimator.

```
yg_gen_boot_design <- as_gen_boot_design(
  design            = pps_design_yg,
  variance_estimator = "Yates-Grundy",
  replicates        = 500
)
yg_gen_boot_design |> svytotal(x = ~ Bush)
```

```
        total      SE
Bush 64518472 2441291
```

There are in total ten different variance estimators which can be used in **svrep** for generalized replication. Some noteworthy variance estimators include the Deville-Tille estimator for samples selected using the cube method (Deville and Tillé, 2005), as well as the SD1 and SD2 successive-difference estimators which are useful for designs with systematic sampling and fine stratification (Ash, 2014).

For two-phase designs, analysts can choose an appropriate variance estimator for each phase and use these to create a combined two-phase variance estimator and resulting replicate weights. The **svrep** package also implements sample-based calibration methods described by Opsomer and Erciulescu (2021), wherein one sample's weights are calibrated based on estimates from another sample, for example when calibrating a second-phase sample to a first-phase sample or when calibrating one survey using estimates from a separate benchmark survey. These methods ensure that variance estimates based on the calibrated weights appropriately account for the variance of the estimated calibration targets.

In addition to implementing resampling and weight adjustment methods, **svrep** also provides general data manipulation and diagnostic tools to help survey statisticians work with replicate weights. Users can easily inspect the distribution of the replicate weights and save the data and weights to a CSV or other output file format. For bootstrap replicates specifically, there are also special tools to help analysts assess how the bootstrap variance estimates' precision changes as a function of the number of replicates, so that analysts can identify the number of replicates necessary to obtain a desired precision level.

4 Discussion

Both **srvyr** and **svrep** can help survey statisticians in different ways. The **srvyr** package helps analysts produce estimates using a provided dataset, while **svrep** primarily helps statisticians implement resampling-based methods for variance estimation and construct replicate weights that can be used to estimate variances for complex sample data, even when such data come from especially complex sampling methods or undergo complex weight adjustments. Both packages rely on **survey** as a foundation, relying on its strengths as a statistical computing engine as well as an expressive framework for organizing survey data and metadata.

There are two non-statistical aspects of **srvyr** and **svrep** which are particularly noteworthy for producers of official statistics. First, both packages include extensive suites of automated software tests that are executed daily by CRAN to verify that the software gives correct results, even after R or other dependencies undergo updates. The level of detail for testing in both packages is an important distinguishing feature relative to several open source software packages for complex survey analysis, which often do not even contain any automated tests of correctness and so may not be fit for purpose in the production of official statistics. Second, both packages include detailed documentation in their user manuals and in online documentation. **svrep** in particular contains detailed descriptions of the variance estimators and resampling algorithms it implements, including ample references to the relevant methodological literature. This emphasis on testing and documentation makes the package useful for bringing new resampling methods from academic research into applied production by survey organizations.

As the **survey** package and the R tidyverse continue to evolve, these packages will evolve too. Planned additions to the **survey** package related to multiphase sampling will affect both packages, which–like **survey**–currently only support multistage sampling or two-phase sample designs. For **svrep**, the list of variance estimators available for use with generalized replication will continue to grow, and there may eventually be implementations of additional methods such as successive-difference replication and pseudo-population bootstrap methods.

References

Ash, Stephen. 2014. “Using Successive Difference Replication for Estimating Variances.” *Survey Methodology, Statistics Canada* 40 (1): 47–59.

Beaumont, Jean-François, and Nelson Émond. 2022. “A Bootstrap Variance Estimation Method for Multistage Sampling and Two-Phase Sampling When Poisson Sampling Is Used at the Second Phase.” *Stats* 5 (2): 339–57. https://doi.org/10.3390/stats5020019.

Beaumont, Jean-François, and Zdenek Patak. 2012. “On the Generalized Bootstrap for Sample Surveys with Special Attention to Poisson Sampling.” *International Statistical Review* 80 (1): 127–48. https://doi.org/10.1111/j.1751-5823.2011.00166.x.

Blewett, Lynn A., Julia A. Rivera Drew, Miriam L. King, Kari C. W. Williams, Daniel Backman, Annie Chen, and Stephanie Richards. 2024. “IPUMS Health Surveys: National Health Interview Survey, Version 7.4 2023.” Minneapolis, MN: IPUMS. https://doi.org/10.18128/D070.V7.4.

Deville, Jean-Claude, and Yves Tillé. 2005. “Variance Approximation Under Balanced Sampling.” *Journal of Statistical Planning and Inference* 128 (2): 569–91. https://doi.org/10.1016/j.jspi.2003.11.011.

Fay, Robert. 1989. “Theory and Application of Replicate Weighting for Variance Calculations.” In *Proceedings of the Section on Survey Research Methods*, 212–17. Alexandria, VA: American Statistical Association.

Freedman Ellis, Greg, and Ben Schneider. 2016. “Srvyr: ’Dplyr’-Like Syntax for Summary Statistics of Survey Data.” https://doi.org/10.32614/CRAN.package.srvyr.

Healy, Kieran Joseph. 2019. *Data Visualization: A Practical Introduction*. Princeton, New Jersey ; Oxford, Oxfordshire: Princeton University Press.

Li, Jianzhu, Sixia Chen, Thomas Krenzke, and Leyla Mohadjer. 2014. “Replication Variance Estimation for Balanced Sampling: An Application to the PIAAC Study.” In *Proceedings of the Section on Survey Research Methods*. Alexandria, VA: American Statistical Association. http://www.asasrms.org/Proceedings/y2014/files/311399_87447.pdf.

Lumley, Thomas. 2004. “Analysis of Complex Survey Samples.” *Journal of Statistical Software* 9 (1): 1–19.

Lumley, Thomas. 2023. “The Survey Package for R, 15 Years On.” *The Survey Statistician* 88: 96–104. https://isi-iass.org/home/wp-content/uploads/Survey_Statistician_2023_July_N88_019.pdf.

Opsomer, J. D., and A. L. Erciulescu. 2021. “Replication Variance Estimation After Sample-Based Calibration.” *Survey Methodology, Statistics Canada* Vol. 47 (No. 2). https://www150.statcan.gc.ca/n1/pub/12-001-x/2021002/article/00006-eng.htm.

Rao, J. N. K., C. F. J. Wu, and K. Yue. 1992. “Some Recent Work on Resampling Methods for Complex Surveys.” *Survey Methodology* 18 (2): 209–17.

Schneider, Benjamin. 2022. “Svrep: Tools for Creating, Updating, and Analyzing Survey Replicate Weights.” https://doi.org/10.32614/CRAN.package.svrep.

Valliant, Richard, Michael Brick, and Jill Dever. 2008. “Weight Adjustments for the Grouped Jackknife Variance Estimator.” *Journal of Official Statistics* 24: 469–88.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the Tidyverse.” *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Zimmer, Stephanie, Powell, Rebecca, and Isabella Velásquez. 2024. *Exploring Complex Survey Data Analysis Using R: A Tidy Introduction with {srvyr} and {survey}*. Chapman & Hall: CRC Press.