Book and Software Review

The survey Package for R, 15 Years on

Thomas Lumley[1]

[1] University of Auckland, t.lumley@auckland.ac.nz

Abstract

In 2008, issue 57 of *The Survey Statistician* published an article about the **survey** package for R, which at that point had been under development for about five years. The current version of the package is 4.2; the version in 2008 was 3.6-12. In this article I will discuss the changes since 2008, and major changes planned for the near future.

Keywords: software, two-phase designs, domain estimation, regression, plausible values.

1 Introduction

The **survey** package has three main goals:

- Integrate standard design-based analyses into R
- Allow non-specialist users to display, analyse, and model complex surveys with only the necessary changes from their usual data analysis practice
- Provide an implementation platform for novel analysis methods.

Since 2008 there has been substantial progress on all these goals. The package is now quite widely used in teaching, research, and at some official statistics agencies. The main download site reports an average of about 2000 downloads per day, and 131 other packages list survey as a dependency.

It's probably no longer necessary in 2023 to introduce readers to R, the free statistical computing environment. It is worth quoting from 2008:

The flexibility of R comes at a price in performance: it is often slower and usually requires more memory than competing systems. This disadvantage is becoming progressively less important as computers improve.

The trend has continued; it is entirely possible to handle national survey data sets on commodity servers, and moderate-sized surveys such as NHANES (see https://www.cdc.gov/nchs/nhanes/index.htm) can be analysed interactively on standard laptops. Speed has not been a priority for implementation, however I am happy to learn about real examples where the package is genuinely too slow and have worked to optimise the code in such cases. A notable example early in development was designs where a large stratum is sampled with certainty, as in some business surveys.

While the package, like R, remains based on command-line code, there is now a basic graphical user interface from the iNZight project (https://inzight.nz). A full description is outside the scope of this article, but iNZight was developed for teaching introductory statistics and for supporting data exploration and analysis. The system now allows survey designs to be specified for data files. When a design is specified, appropriate methods for analysis and graphics are transparently used with no extra effort from the user. The visual interface allows for only two stages of sampling, with stratification at the first stage, and sampling with or without replacement.

The survey package does not provide much in the way of output formatting. This is deliberate; R has other packages for constructing and formatting tables, notably the new **gt** package (Iannone et al, 2023).

Basic structure of the package

Analysis functions in the **survey** package take two standard inputs: a model formula describing the variables to be used, and a survey design object containing the data and meta-data. The design objects are created by the functions `svydesign` (when strata, clusters, etc are supplied) and `svrepdesign` (when replicate weights are supplied). Replicate weights can also be created for a `svydesign` object. Arbitrarily many stages of stratified cluster sampling are possible, with or without replacement. There is also some support for PPS sampling and for arbitrary designs specified via a pairwise probability matrix.

The design objects are not simply data sets: subsetting a design object gives a design subpopulation object that will produce valid domain estimates. Raking, calibration, and trimming of weights are also available for design objects.

When data are inconveniently large to reside in memory, they can now be kept in a database. The `svydesign` and `svrepdesign` functions accept a database table as a data source and produce survey design objects that load only the necessary data into memory for each analysis.

2. New features since 2008

Comparisons between domains

The package has always supported domain estimation, but did not initially provide variance estimates for contrasts *between* domains (except through regression or ratio estimation). When using replicate weights it is easy to estimate these variances by carrying the replicate estimates through any calculation, and this was added not long after 2008. With linearisation it is somewhat more complicated, but since version 4.0 of the package, analysis functions can return a matrix of influence functions that will be carried through subsequent calculations. This approach means new functions – built into the package or written by users – can automatically take advantage of the between-domain comparisons.

Here are two examples comparing different levels of school, in a built-in dataset on California standardised assessments. The first compares the mean size of high schools and elementary schools, the second compares the agreement between two yes/no progress indicators, measured with Cohen’s kappa.

```
> library(survey)
> data(api)
> dclus1<-svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
> means<- svyby(~enroll, by=~stype, design=dclus1, svymean, covmat=TRUE)
> means
  stype    enroll        se
E     E  432.8542  16.51599
H     H 1130.2857 357.12197
M     M  897.7200  99.53188
> svycontrast(means, c(H=1, E=-1))
         contrast     SE
contrast   697.43 346.47
> kappas<- svyby(~comp.imp+sch.wide, by=~stype, design=dclus1, svykappa, 
covmat=TRUE)
```

```
> kappas
  stype     kappa  se.kappa
E     E 0.4827586 0.0716370
H     H 0.4285714 0.1638083
M     M 0.7491639 0.1235138
> svycontrast(kappas, c(M=1,E=-1))
         contrast     SE
contrast  0.26641 0.1575
```

Rank tests

My initial view on rank tests for survey data was that they were not well motivated: they could not be exact and would rely on the same asymptotics as a t-test. Users, however, were interested in having rank tests available, and expressed some surprise that they had not been implemented, so Alastair Scott and I worked out how to define them (Lumley & Scott, 2013). The resulting tests use the values of the estimated population cumulative distribution function as scaled ranks, and the population or superpopulation null hypothesis of the test (though not, of course, the power) is the same regardless of the sampling design. The most useful of these are probably the Wilcoxon test and the tests for specific quantiles, but users can define their own rank transformations. As is well known, the Wilcoxon test can also be derived as a score test in a proportional odds model, but this approach requires more computational effort and seems more difficult to generalise.

Weighted version of the $G^{\rho,\gamma}$ family of logrank tests for survival data have also been implemented, based on ideas of Rader (2014) but with some speed and memory optimisations.

Two-phase samples

Sampling from existing databases has become increasingly important in health research, either when new variables are being measured on existing cohorts, or when subsamples are taken for validation from electronic health record databases. Shepherd et al (2022) gives a modern example of a multi-wave twophase validation study. Two-phase sample objects can also be used to implement calibration for non-response in a single-phase sample.

The `twophase` function defines a two-phase sample object, by providing the sampling design at each phase. The resulting object can be used in all analyses, and supports calibration of phase 2 to phase 1 as well as calibration of phase 1 to the population. This example shows a two-phase case-control design as described by Breslow and Chatterjee (1999). The first phase is actually a pair of clinical trials, treated here as a simple random sample; the second phase is stratified on outcome and tumour histology. In the second analysis, the design is post-stratified on tumour stage, improving precision for a number of the parameters.

```
> data(nwtco, package="survival")
> nwtco$incc2<-
as.logical(with(nwtco, ifelse(rel | instit==2,1,rbinom(nrow(nwtco),1,.1))))
> dccs2<-
twophase(id=list(~seqno,~seqno),strata=list(NULL,~interaction(rel,instit)),
    data=nwtco, subset=~incc2)
> summary(svyglm(rel~factor(stage)*factor(histol),design=dccs2,
family=quasibinomial()))
```

```
Call:
svyglm(formula = rel ~ factor(stage) * factor(histol), design = dccs2,
    family = quasibinomial())

Survey design:
twophase2(id = id, strata = strata, probs = probs, fpc = fpc,
    subset = subset, data = data)

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -2.6802     0.1264 -21.206  < 2e-16 ***
factor(stage)2                   0.6994     0.2004   3.490 0.000502 ***
factor(stage)3                   1.0275     0.2208   4.653 3.67e-06 ***
factor(stage)4                   0.7804     0.2449   3.187 0.001477 **
factor(histol)2                  1.2147     0.3246   3.743 0.000192 ***
factor(stage)2:factor(histol)2   0.2073     0.4589   0.452 0.651547
factor(stage)3:factor(histol)2   0.4942     0.4343   1.138 0.255383
factor(stage)4:factor(histol)2   1.0384     0.6137   1.692 0.090918 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasibinomial family taken to be 1.000901)

Number of Fisher Scoring iterations: 4
> gccs8<-calibrate(dccs2, phase=2, formula=~interaction(rel,stage,instit))
> summary(svyglm(rel~factor(stage)*factor(histol),design=gccs8,
family=quasibinomial()))

Call:
svyglm(formula = rel ~ factor(stage) * factor(histol), design = gccs8,
    family = quasibinomial())

Survey design:
calibrate(dccs2, phase = 2, formula = ~interaction(rel, stage,
    instit))

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -2.6836     0.1089 -24.647  < 2e-16 ***
factor(stage)2                   0.7706     0.1478   5.213 2.22e-07 ***
factor(stage)3                   0.7707     0.1543   4.995 6.85e-07 ***
factor(stage)4                   1.0658     0.1768   6.028 2.27e-09 ***
factor(histol)2                  1.2167     0.3226   3.771 0.000171 ***
factor(stage)2:factor(histol)2   0.1647     0.4474   0.368 0.712865
factor(stage)3:factor(histol)2   0.7037     0.4346   1.619 0.105717
factor(stage)4:factor(histol)2   0.9331     0.5760   1.620 0.105501
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasibinomial family taken to be 1.000901)

Number of Fisher Scoring iterations: 4
```

Regression models

Even in 2008, **survey** supported generalised linear models and the Cox proportional hazards model. Additions since then include loglinear models for contingency tables, the proportional odds model and other cumulative link models, and accelerated failure models for survival data. A companion package, **svyVGAM** provides an interface to the wide range of models in the **VGAM** package (Yee, 2023; Lumley 2020), including multinomial regression, negative binomial, and zero-inflated and zero-truncated Poisson.

```
> library(svyVGAM)
> dclus2<-svydesign(id=~dnum+snum, fpc=~fpc1+fpc2, data=apiclus2)
> dclus2<-update(dclus2, mealcat=cut(meals,c(0,25,50,75,100)))
> svy_vglm(mealcat~avg.ed+mobility+ell, design=dclus2,
family=multinomial(refLevel=1))
2 - level Cluster Sampling design
With (38, 116) clusters.
update(dclus2, mealcat = cut(meals, c(0, 25, 50, 75, 100)))

Call:
vglm(formula = formula, family = family, data = surveydata, weights =
.survey.prob.weights)


Coefficients:
(Intercept):1 (Intercept):2 (Intercept):3      avg.ed:1      avg.ed:2
avg.ed:3
  16.94881271   17.10446320   13.33436684   -5.80250194   -7.26575425   -
9.08907835
   mobility:1    mobility:2    mobility:3         ell:1         ell:2
ell:3
  -0.00363934    0.16130315    0.21750911    0.10998773    0.17127858
0.38340792

Degrees of Freedom: 348 Total; 336 Residual
Residual deviance: 4368.787
Log-likelihood: -2184.394

This is a multinomial logit model with 4 levels
> svy_vglm(ordered(mealcat)~avg.ed+mobility+ell, design=dclus2,
family=propodds())
2 - level Cluster Sampling design
With (38, 116) clusters.
update(dclus2, mealcat = cut(meals, c(0, 25, 50, 75, 100)))

Call:
vglm(formula = formula, family = family, data = surveydata, weights =
.survey.prob.weights)


Coefficients:
(Intercept):1 (Intercept):2 (Intercept):3        avg.ed      mobility
ell
    7.6020176     5.0890255     1.4022044    -3.3569363     0.1117306
0.1446080

Degrees of Freedom: 348 Total; 342 Residual
Residual deviance: 5091.878
Log-likelihood: -2545.939
```

The package implements research on ‘working likelihood’ analyses of generalised linear models and the Cox model. This comes in two parts. First, the familiar Rao-Scott tests for multiway tables have been extended to these regression models. Second, there are now design-based analogues of AIC and BIC for model selection (Lumley & Scott, 2015) in generalised linear models.

```
> model0<-svyglm(I(sch.wide=="Yes")~ell+meals+mobility, design=dclus2,
family=quasibinomial())
> model1<-svyglm(I(sch.wide=="Yes")~ell+meals+mobility+as.numeric(stype),
+      design=dclus2, family=quasibinomial())
> model2<-svyglm(I(sch.wide=="Yes")~ell+meals+mobility+stype, design=dclus2,
family=quasibinomial())
> anova(model2)
```

```
Anova table:  (Rao-Scott LRT)
svyglm(formula = I(sch.wide == "Yes") ~ ell, design = dclus2,
    family = quasibinomial())
            stats     DEff       df ddf        p
ell       1.1259  0.76870  1.00000  38 0.236339
meals     4.8189  1.24181  1.00000  37 0.058326 .
mobility  0.3712  1.42335  1.00000  36 0.608418
stype    52.4054  2.43494  2.00000  34 0.001341 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova(model0,model2)
Working (Rao-Scott+F) LRT for stype
 in svyglm(formula = I(sch.wide == "Yes") ~ ell + meals + mobility +
    stype, design = dclus2, family = quasibinomial())
Working 2logLR =  21.52228 p= 0.0013407
(scale factors:  1.7 0.3 );  denominator df= 34
> anova(model1, model2)
Working (Rao-Scott+F) LRT for stype - as.numeric(stype)
 in svyglm(formula = I(sch.wide == "Yes") ~ ell + meals + mobility +
    stype, design = dclus2, family = quasibinomial())
Working 2logLR =  25.10744 p= 1.816e-05
df=1;  denominator df= 34
```

Predictive margins (Korn & Graubard, 1999) are also available for generalised linear models. In this example an extract from NHANES is used to estimate the prevalence of high cholesterol by race/ethnicity, standardised for age and sex. These are then transformed to prevalence ratios (relative risks) by svycontrast.

```
> data(nhanes)
> nhanes_design <- svydesign(id=~SDMVPSU, strata=~SDMVSTRA, weights=~WTMEC2YR,
nest=TRUE,data=nhanes)
> agesexmodel<-svyglm(HI_CHOL~agecat+RIAGENDR,
design=nhanes_design,family=quasibinomial)
> means<-svypredmeans(adjustmodel=agesexmodel, groupfactor= ~factor(race))
> means
      mean     SE
2 0.114596 0.0065
3 0.084718 0.0106
1 0.123081 0.0058
4 0.108770 0.0304
> ## relative risks compared to non-Hispanic white
> svycontrast(means,quote(`1`/`2`))
         nlcon     SE
contrast 1.074 0.0722
> svycontrast(means,quote(`3`/`2`))
           nlcon     SE
contrast 0.73928 0.0923
```

Multiple imputation and plausible values

Constructing multiple imputations is a specialised task for which there is other software, but with the accompanying **mitools** package the **survey** package does support the analysis of multiply-imputed data and of data with ‘plausible values’, as in some educational surveys. From a computational viewpoint the distinction between the two is that imputation provides multiple complete datasets whereas plausible values provide multiple alternative columns in a single dataset. In both settings, the results of multiple analyses are combined using Rubin’s rules.

For plausible values, the inputs consist of an *action* to be performed on each plausible value, and a set of formulas specifying which columns go into which variables in the action. Here is an example using the New Zealand subset of the PISA 2012 educational survey (OECD, 2013).

```
> library(mitools)
> data(pisamaths, package="mitools")
> des<-svydesign(id=~SCHOOLID+STIDSTD, strata=~STRATUM, nest=TRUE,
+      weights=~W_FSCHWT+condwt, data=pisamaths)
> results<-withPV(list(maths~PV1MATH+PV2MATH+PV3MATH+PV4MATH+PV5MATH),
+    data=des,
+    action=quote(svyglm(maths~ST04Q01*(PCGIRLS+SMRATIO)+MATHEFF+OPENPS,
design=des))
+    )
> MIcombine(results)
Multiple imputation results:
      withPV.survey.design(list(maths ~ PV1MATH + PV2MATH + PV3MATH +
    PV4MATH + PV5MATH), data = des, action = quote(svyglm(maths ~
    ST04Q01 * (PCGIRLS + SMRATIO) + MATHEFF + OPENPS, design = des)))
      MIcombine.default(results)
                          results         se
(Intercept)          4.729436e+02 20.7907335
ST04Q01Male          5.268461e+01 24.1976184
PCGIRLS              5.974293e+01 17.6683218
SMRATIO              3.552268e-02  0.1233799
MATHEFF              4.736517e+01  3.0904746
OPENPS               1.317289e+01  3.0565353
ST04Q01Male:PCGIRLS -1.109811e+02 32.8851005
ST04Q01Male:SMRATIO  4.391909e-03  0.1358470
```

the action is to fit a design-weighted linear model for maths performance with predictors gender, percentage of girls in the school, student:teacher ratio, and two attitude questions. The response variable maths is not a variable in the data set; instead, the first argument of withPV says that the action should be performed five times, with each of the actual maths score variables substituted for maths.

For multiple imputation, the function `imputationList` wraps a set of imputed complete data sets so they can be used to create a packaged list of design objects that can then be used for analyses. Again, `MIcombine` is used to combine the sets of results.

Your own extensions

Users will inevitably need some estimators that are not already implemented. Two tools for extending the package are `withReplicates` and `svycontrast`.

When the estimator can be expressed as an explicit function of quantities that can already be estimated, svycontrast will compute standard errors using either the delta method (with symbolic differentiation) or replicate weights. A couple of examples have already been seen above. Here is a slightly more complicated example. For each school in a sample, we have the number of enrolled students, and we wish to calculate the fraction of students in elementary, middle, and high schools. The code first computes the estimated population total, then divides the domain totals by it.

```
> totals<-svyby(~enroll,~stype,design=dclus1,svytotal,covmat=TRUE)
> totals<-svycontrast(totals, list(total=quote(E+M+H)), add=TRUE)
> svycontrast(totals, list(quote(E/total), quote(M/total), quote(H/total)))
      nlcon     SE
[1,] 0.6196 0.0657
[2,] 0.2231 0.0412
[3,] 0.1573 0.0431
```

The `withReplicates` function is useful when you already have code to compute point estimates, but not standard errors. This could be because you have written the code, or it could be because there is existing code in some other package that accepts precision weights or frequency weights and gives correct point estimates. The `withReplicates` function provides a simple interface to run this existing code using replicate weights and compute a variance estimate. For example, here we fit a median regression using code that expects precision weights, and extract the coefficient estimates. Re-running this code with 100 sets of bootstrap replicate weights allows for design-based standard error computations.

```
> library(quantreg)
> data(api)
> ## one-stage cluster sample
> dclus1<-svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
> ## convert to bootstrap
> bclus1<-as.svrepdesign(dclus1,type="bootstrap", replicates=100)
> ## median regression
> withReplicates(bclus1, quote(coef(rq(api00~api99, tau=0.5,
weights=.weights))))
               theta      SE
(Intercept) 87.78505 16.6070
api99        0.91589  0.0237
```

To provide more examples of how to add new estimators to the package, I wrote a post about different ways to implement zero-inflated Poisson regression (Lumley, 2015).

The future

There are two major upgrades planned in the short term. The first is to speed up the basic multistage variance computation using C++ code contributed by Ben Schneider (2022). The relative speed gains are greatest for small data sets, but the package will be faster for surveys of all sizes. The second major upgrade is to add an interface to small-area estimation code developed by Richard Li, Jon Wakefield and co-workers (Li et al, 2022).

I also plan to add explicit support for multi-phase and multi-frame samples. Multi-phase sampling is of increasing interest in health research, with validation subsampling of large existing databases, and there is no real barrier to extending from the current two phases to three or more. Multi-frame sampling has well-developed methods and the main task in implementation is developing an appropriate user interface.

Mixed models are an area of interest for future development. It is surprisingly difficult to estimate these models under complex sampling, especially when the model structure is not aligned with the sampling structure, but I plan to implement methods based on pairwise likelihood (Yi et al 2016; Huang, 2019).

Other developments in the future will depend in part on what users ask for. Many changes to the package over the years have been responses to user feedback. Sometimes this simply meant fixing bugs; other times it needed new coding or even new methods research.

I will end by noting that the **survey** package does not have any dedicated external or internal funding for either development or maintenance, though at some points it has been supported indirectly by statistical methods research funding (from the Marsden Fund of the Royal Society of New Zealand). It is not clear that this situation is sustainable indefinitely.

References

Breslow N.E. and Chatterjee N. (1999), Design and analysis of two-phase studies with binary outcome applied to Wilms tumour prognosis, *Applied Statistics,* 48:457-68.

Graubard B., Korn E. (1999), Predictive Margins with Survey Data, *Biometrics,* 55:652-659.

Huang X. (2019), *Mixed Models for Complex Survey Data* PhD thesis, University of Auckland.

Iannone R., Cheng J., Schloerke B., Hughes E., Lauer A. and Seo J. (2023), *gt: Easily Create Presentation-Ready Display Tables*. R package version 0.9.0. https://CRAN.R-project.org/package=gt

Li Z.R., Martin B.D., Hsiao Y., Godwin J., Paige J., Gao P., Wakefield J., Clark S.J., Fuglstad G-A, and Riebler A. (2022), *SUMMER: Small-Area-Estimation Unit/Area Models and Methods for Estimation in R*. R package version 1.3.0. https://CRAN.R-project.org/package=SUMMER

Lumley T. (2015), *Zero-inflated Poisson from complex samples.* https://notstatschat.rbind.io/2015/05/26/zero-inflated-poisson-from-complex-samples/

Lumley T. (2020), *MOAR survey regression models,* https://notstatschat.rbind.io/2020/09/24/moar-survey-regression-models/

Lumley T., and Scott A.J. (2013), Two-sample rank tests under complex sampling, *Biometrika*, 100 (4), 831-842.

Lumley T., and Scott A.J. (2015), AIC and BIC for modelling with complex survey data, *Journal of Survey Statistics and Methodology,* 3 (1): 1-18.

OECD (2013), *PISA 2012 Assessment and Analytical Framework: Mathematics, Reading, Science, Problem Solving and Financial Literacy*. OECD Publishing.

Rader KA (2014), *Methods for Analyzing Survival and Binary Data in Complex Surveys.* Doctoral dissertation, Harvard University. http://nrs.harvard.edu/urn-3:HUL.InstRepos:12274283

Schneider B (2022), *Practical Significance: More on Speeding up the Survey Package: Adding the New C++ Functions to the Package.* https://www.practicalsignificance.com/posts/adding-rcpp-to-the-survey-package/

Shepherd B.E., Han K., Chen T., Bian A., Pugh S., Duda S.N., Lumley T., Heerman W.J., and Shaw P.A. (2022), Multiwave validation sampling for error-prone electronic health records, *Biometrics*. doi: 10.1111/biom.13713. Epub ahead of print. PMID: 35775996

Yee, T. (2023), *VGAM: Vector Generalized Linear and Additive Models*. R package version 1.1-8. https://CRAN.R-project.org/package=VGAM

Yi G.Y., Rao J.N.K., and Li H. (2016), A weighted composite likelihood approach for analysis of survey data under two-level models, *Statistica Sinica,* 26(2) 569-587.