Advanced Data Analysis in R

Survey Analsis in R

Michael DeWitt

2018-02-09 (Updated 2019-03-11)

Survey Analysis in R

What makes survey analysis different?

Survey analysis is *design based*

Often we talk about *probability or random samples*

These concepts make inferences really nice

Properties of Design Based Surveys

A quick refresher[1]

1. Every individual in the population must have a non-zero probability of ending up in the sample (π_i)

[1] Lumley (2010)

Properties of Design Based Surveys

A quick refresher[1]

1. Every individual in the population must have a non-zero probability of ending up in the sample (π_i)
2. The probability of π_i must be known for every individual in who does end up in the sample

[1] Lumley (2010)

Properties of Design Based Surveys

A quick refresher[1]

1. Every individual in the population must have a non-zero probability of ending up in the sample (π_i)
2. The probability of π_i must be known for every individual in who does end up in the sample
3. Eevery pair of individuals in the sample must have a non-zero probability of both ending up in the sample ($\pi_{i,j}$ for the pair of individuals (i, j))

[1]Lumley (2010)

Properties of Design Based Surveys

A quick refresher[1]

1. Every individual in the population must have a non-zero probability of ending up in the sample (π_i)
2. The probability of π_i must be known for every individual in who does end up in the sample
3. Eevery pair of individuals in the sample must have a non-zero probability of both ending up in the sample ($\pi_{i,j}$ for the pair of individuals (i, j))
4. The probability $\pi_{i,j}$ must be known for every pair that does ebd up in the sample

[1] Lumley (2010)

Introducing the `survey` package

A little about `survey`

Thomas Lumley developed the `survey` package

Initially a port of STATA's `svy` functions following a similar syntax

Can perform typical types of design based analysis

- Simple Random
- Stratified
- Clusters
- Multi-stage
- Repeated Measures

A little about `survey`

Perform post-survey corrections
- post-stratification
- raking (iterative proportional fitting)
- calibration

And more. . . !

Diving into the software. . .

Describing your model

The primary argument in survey is the svydesign function

```
library(survey)

svydesign(ids = to specify clusters (~1 otherwise),
          probs = Sampling Probabilities if available,
          strata = Strata membership if available,
          fpc = Finite Population Values,
          data = Your Data Frame,
          nest = T/F if there is nesting within your st
          weights = Sampling Weights if available,
          pps = Probability Proportional to Size)
```

Quick Note On `survey`

Many of the functions in `survey` utilise R “formula notation”

Indicates the tilde “$\sim$” must be used (e.g. `~cluster`)

But Let's Try An Example

Let's try an example with the api dataset that is part of the survey package

This data set represents California Academic Performance Index

```r
library(survey)
library(dplyr)
data(api)
```

Let's Inspect the Data

```
head(apisrs)
##                     cds stype           name
## 1039 15739081534155     H  McFarland High
## 1124 19642126066716     E Stowers (Cecil
## 2868 30664493030640     H Brea-Olinda Hig
## 1273 19644516012744     E Alameda Element
## 4926 40688096043293     E Sunnyside Eleme
## 2463 19734456014278     E Los Molinos Ele
##                               sname snum
## 1039                 McFarland High 1039
## 1124 Stowers (Cecil B.) Elementary 1124
## 2868               Brea-Olinda High 2868
## 1273             Alameda Elementary 1273
## 4926           Sunnyside Elementary 4926
```

Specifying the Survey Object (SRS)

This is a simple random sample with finite population correct (since we know the population)

```
(svy_api_srs <- svydesign(ids = ~1,
                          fpc = ~fpc,
                          data = apisrs))
## Independent Sampling design
## svydesign(ids = ~1, fpc = ~fpc, data = apisrs)
```

Trying With A Different Survey Design (Stratified)

In this case we have a stratified random sample (different school types)

```
(svy_api_strat <- svydesign(ids= ~1,
                            strata = ~stype,
                            fpc = ~fpc,
                            data = apistrat))
## Stratified Independent Sampling design
## svydesign(ids = ~1, strata = ~stype, fpc = ~fpc, da
```

Trying With A Different Survey Design (Cluster)

Two stage cluster sampling 40 school disticts then five schools within each district

- Stage 1 district cluster with population `fpc1`
- Stage 2 district cluster with population `fpc2`

```
(svy_api_cluster <- svydesign(ids= ~dnum+snum,
                               fpc = ~fpc1+fpc2,
                               data = apiclus2))
## 2 - level Cluster Sampling design
## With (40, 126) clusters.
## svydesign(ids = ~dnum + snum, fpc = ~fpc1 + fpc2,
```

Analysis with svy objects

Correct Estimates

survey applies correct calculations given the survey design

```
svymean(~api00, svy_api_cluster)
##          mean     SE
## api00 670.81 30.099
```

vs

```
cbind(mean(apiclus2[["api00"]]),
      sd(apiclus2[["api00"]]))
##          [,1]     [,2]
## [1,] 703.8095 134.1507
```

Survey Functions

Functions in the survey package begin with the svy prefix

Utilise the formula notation

```
svyquantile(x = ~api99+api00,
            svy_api_srs,
            quantile= c(0.25,.75))
##       0.25 0.75
## api99  513  738
## api00  544  752
```

Calculating Contrasts

You can add contrasts with `svycontrast`

Say I wanted to look at the ratio of my high school score to my elementary school score

```
# Mean
mean_score <- svyby( ~api99, ~stype, svymean,
                      design = svy_api_cluster)
# Contrast ratio use `quote`
svycontrast(mean_score, quote(H/E))
##             nlcon     SE
## contrast 0.90614 0.0507
```

Adding Contrasts to the data

Use the `update` function to add new calculated fields to your survey design object

```
(svy_api_cluster <- update(svy_api_cluster,
                           score_imp = api00/api99))
## 2 - level Cluster Sampling design
## With (40, 126) clusters.
## update(svy_api_cluster, score_imp = api00/api99)
```

Adding Contrasts to the data

Now we can easily perform our analysis

```
svyby(~score_imp, ~stype, svymean,
       design = svy_api_cluster)
##   stype score_imp          se
## E     E  1.057525 0.006591223
## H     H  1.005193 0.003781072
## M     M  1.017354 0.012393586
```

Performing Regressions

```
svyglm(score_imp~ meals + avg.ed, svy_api_cluster)
## 2 - level Cluster Sampling design
## With (40, 126) clusters.
## update(svy_api_cluster, score_imp = api00/api99)
##
## Call:  svyglm(formula = score_imp ~ meals + avg.ed
##
## Coefficients:
## (Intercept)        meals       avg.ed
##   0.9742040    0.0007394    0.0103667
##
## Degrees of Freedom: 125 Total (i.e. Null);  37 Res
## Null Deviance:        0.2624
## Residual Deviance: 0.2118    AIC: -391
```

Post-survey corrections

References

Lumley, Thomas. 2010. *Complex Surveys: A Guide to Analysis Using R*. John Wiley & Sons.