A survey analysis example

Thomas Lumley

April 3, 2020

This document provides a simple example analysis of a survey data set, a subsample from the California Academic Performance Index, an annual set of tests used to evaluate California schools. The API website, including the original data files are at http://api.cde.ca.gov. The subsample was generated as a teaching example by Academic Technology Services at UCLA and was obtained from http://www.ats.ucla.edu/stat/stata/library/svy_survey.htm.

We have a cluster sample in which 15 school districts were sampled and then all schools in each district. This is in the data frame apiclus1, loaded with data(api). The two-stage sample is defined by the sampling unit (dnum) and the population size(fpc). Sampling weights are computed from the population sizes, but could be provided separately.

```r
> library(survey)
> data(api)
> dclus1 <- svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)
```

The `svydesign` function returns an object containing the survey data and metadata.

```r
> summary(dclus1)
1 - level Cluster Sampling design
With (15) clusters.
svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)
Probabilities:
     Min.   1st Qu.    Median      Mean   3rd Qu.    Max.
0.02954 0.02954 0.02954 0.02954 0.02954 0.02954
Population size (PSUs): 757
Data variables:
 [1] "cds" "stype" "name" "sname" "snum" "dname"
 [7] "dnum" "cname" "cnum" "flag" "pcttest" "api00"
[13] "api99" "target" "growth" "sch.wide" "comp.imp" "both"
[19] "awards" "meals" "ell" "yr.rnd" "mobility" "acs.k3"
[25] "acs.46" "acs.core" "pct.resp" "not.hsg" "hsg" "some.col"
[31] "col.grad" "grad.sch" "avg.ed" "full" "emer" "enroll"
[37] "api.stu" "fpc" "pw"
```
We can compute summary statistics to estimate the mean, median, and quartiles of the Academic Performance Index in the year 2000, the number of elementary, middle, and high schools in the state, the total number of students, and the proportion who took the test. Each function takes a formula object describing the variables and a survey design object containing the data.

```r
> svymean(~api00, dclus1)

   mean    SE
api00  644.17 23.542

> svyquantile(~api00, dclus1, quantile=c(0.25,0.5,0.75), ci=TRUE)

$quantiles
   0.25  0.5  0.75
api00  551.75 652 717.5

$CIs
   , , api00

   0.25  0.5  0.75
(lower 490.5049 557.1892 691.0000
upper) 626.9903 714.0000 772.7397

> svytotal(~stype, dclus1)

   total    SE
stypeE  4873.97 1333.32
stypeH   473.86  158.70
stypeM  846.17  167.55

> svytotal(~enroll, dclus1)

   total    SE
enroll  3404940 932235

> svyratio(~api.stu,~enroll, dclus1)

Ratio estimator: svyratio.survey.design2(~api.stu, ~enroll, dclus1)
Ratios=

   enroll
api.stu 0.8497087
SEs=

   enroll
api.stu 0.008386297
```

The ordinary R subsetting functions [ and subset work correctly on these survey objects, carrying along the metadata needed for valid standard errors. Here we compute the proportion of high school students who took the test
> svyratio(~api.stu, ~enroll, design=subset(dclus1, stype=="H"))

Ratio estimator: svyratio.survey.design2(~api.stu, ~enroll, design = subset(dclus1, stype == "H"))

Ratios=

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>enroll</td>
<td>0.8300683</td>
</tr>
</tbody>
</table>

SEs=

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>enroll</td>
<td>0.01472607</td>
</tr>
</tbody>
</table>

The warnings referred to in the output occurred because several school districts have only one high school sampled, making the second stage standard error estimation unreliable.

Specifying a large number of variables is made easier by the *make.formula* function

> vars<-names(apiclus1)[c(12:13,16:23,27:37)]
> svymean(make.formula(vars),dclus1,na.rm=TRUE)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>SE</td>
<td></td>
</tr>
<tr>
<td>api00</td>
<td>643.203822</td>
<td>25.4936</td>
</tr>
<tr>
<td>api99</td>
<td>605.490446</td>
<td>25.4987</td>
</tr>
<tr>
<td>sch.wideNo</td>
<td>0.127389</td>
<td>0.0247</td>
</tr>
<tr>
<td>sch.wideYes</td>
<td>0.872611</td>
<td>0.0247</td>
</tr>
<tr>
<td>comp.impNo</td>
<td>0.273885</td>
<td>0.0365</td>
</tr>
<tr>
<td>comp.impYes</td>
<td>0.726115</td>
<td>0.0365</td>
</tr>
<tr>
<td>bothNo</td>
<td>0.273885</td>
<td>0.0365</td>
</tr>
<tr>
<td>bothYes</td>
<td>0.726115</td>
<td>0.0365</td>
</tr>
<tr>
<td>awardsNo</td>
<td>0.292994</td>
<td>0.0397</td>
</tr>
<tr>
<td>awardsYes</td>
<td>0.707006</td>
<td>0.0397</td>
</tr>
<tr>
<td>meals</td>
<td>50.636943</td>
<td>6.6588</td>
</tr>
<tr>
<td>ell</td>
<td>26.891720</td>
<td>2.1567</td>
</tr>
<tr>
<td>yr.rndNo</td>
<td>0.942675</td>
<td>0.0358</td>
</tr>
<tr>
<td>yr.rndYes</td>
<td>0.057325</td>
<td>0.0358</td>
</tr>
<tr>
<td>mobility</td>
<td>17.719745</td>
<td>1.4555</td>
</tr>
<tr>
<td>pct.resp</td>
<td>67.171975</td>
<td>9.6553</td>
</tr>
<tr>
<td>not.hsg</td>
<td>23.082803</td>
<td>3.1976</td>
</tr>
<tr>
<td>hsg</td>
<td>24.847134</td>
<td>1.1167</td>
</tr>
<tr>
<td>some.col</td>
<td>25.210191</td>
<td>1.4709</td>
</tr>
<tr>
<td>col.grad</td>
<td>20.611465</td>
<td>1.7305</td>
</tr>
<tr>
<td>grad.sch</td>
<td>6.229299</td>
<td>1.5361</td>
</tr>
<tr>
<td>avg.ed</td>
<td>2.621529</td>
<td>0.1054</td>
</tr>
<tr>
<td>full</td>
<td>87.127389</td>
<td>2.1624</td>
</tr>
<tr>
<td>emer</td>
<td>10.968153</td>
<td>1.7612</td>
</tr>
<tr>
<td>enroll</td>
<td>573.713376</td>
<td>46.5959</td>
</tr>
<tr>
<td>api.stu</td>
<td>487.318471</td>
<td>41.4182</td>
</tr>
</tbody>
</table>
Summary statistics for subsets can also be computed with `svyby`. Here we compute the average proportion of “English language learners” and of students eligible for subsidized school meals for elementary, middle, and high schools

```R
> svyby(~ell+meals, ~stype, design=dclus1, svymean)

          stype     ell meals se.ell se.meals
E E 29.69444 53.09028 1.411617 7.070399
H H 15.00000 37.57143 5.347065 5.912262
M M 22.68000 43.08000 2.952862 6.017110

Regression models show that these socioeconomic variables predict API score and whether the school achieved its API target

```R
> regmodel <- svyglm(api00~ell+meals,design=dclus1)
> logitmodel <- svyglm(I(sch.wide=="Yes")~ell+meals, design=dclus1, family=quasibinomial())
> summary(regmodel)
```

```
Call:
svyglm(formula = api00 ~ ell + meals, design = dclus1)
Survey design:
svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

Coefficients:
           Estimate Std. Error t value Pr(>|t|)
(Intercept)  817.182   18.671 43.768 1.32e-14 ***
el  -0.509    0.326  -1.561   0.144
meals  -3.146    0.302  -10.423 2.29e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1

(Dispersion parameter for gaussian family taken to be 3161.207)

Number of Fisher Scoring iterations: 2
```

```R
> summary(logitmodel)
```

```
Call:
svyglm(formula = I(sch.wide == "Yes") ~ ell + meals, design = dclus1, family = quasibinomial())
Survey design:
svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.8996 0.5099  3.725 0.00290 **
```

4
ell  0.039925  0.012443  3.209  0.00751 **  
meals -0.019115  0.008825 -2.166  0.05117 .  
---  
Signif. codes:  0 ‘’’’  0.001 ‘**’  0.01 ‘*’  0.05 ‘.’  0.1 ‘ ’  1  
(Dispersion parameter for quasibinomial family taken to be 0.9627734)  
Number of Fisher Scoring iterations: 5  

We can calibrate the sampling using the statewide total for the previous year's API  
> gclus1 <- calibrate(dclus1, formula=~api99, population=c(6194, 3914069))  
which improves estimation of some quantities  
> svymean(~api00, gclus1)  
   mean    SE  
api00 666.72 3.2959  
> svyquantile(~api00, gclus1, quantile=c(0.25,0.5,0.75), ci=TRUE)  
$quantiles      
api00  592.0652  681.181  736.5414  
$CIs  
, , api00  
      0.25 0.5 0.75  
(lower 551.6500 661.6547 719.8118  
upper) 622.4326 697.7199 757.2617  
> svytotal(~stype, gclus1)  
     total    SE  
stypeE 4881.77 302.15  
stypeH 463.35 183.03  
stypeM 848.88 194.76  
> svytotal(~enroll, gclus1)  
   total    SE  
enroll 3357372 243227  
> svyratio(~api.stu,~enroll, gclus1)
Ratio estimator: svyratio.survey.design2(~api.stu, ~enroll, gclus1)

Ratios =
   enroll
api.stu 0.8506941
SEs =
   enroll
api.stu 0.008674888