Analysis of Complex Survey Data in Stata

Isabel Cañette

Senior Statistician

StataCorp LP

2010 Mexican Stata Users Group meeting

April 29, 2010

Introduction

- Surveys are aimed to collect information to study characteristics on a fixed population
- Surveys (as opposed to census) are usually performed to cut costs and time resources
- For the same reasons, researchers may opt for complex survey designs, as opposed to simple random samples (SRS)
- In some situations, SRS may be impossible due to lack of a list with all the individuals.

Stata approach to survey data

In Stata, we separate the stage of the declaration of the design from the estimation stage. Once the design is declared (`svyset`), it will be automatically taken into account every time we use the `svy` prefix.

If for i.i.d data we write:

```
regress
proportion
```

Then, for survey data we write:

```
svy: regress
svy: proportion
```

Survey data characteristics

Syntax for a one-stage design:

svyset [*psu*] [*weight*] [, strata(*varname*) fpc(*varname*)]

These optional arguments refer to Stata variables containing:

- sampling units, or clusters, are the actual units we sample.
- sampling weight is the inverse of the probability of an observation being sampled (in other words, the number of observations in the population represented by each observation in the sample)
- Stratification consists of dividing the population into two or more sections, and taking independent samples within each section (strata).
- fpc (finite population correction) is the proportion of psu sampled within each stratum (only for sampling without replacement)

Using svyset for a SRS

If we have a SRS (without replacement), we will usually consider it as “standard” data. There are, however, cases in which we may want to use `svyset`. We’ll use this setting to illustrate the use of `fpc` and `weight`.

Using svyset for a SRS

```
. use srs1, clear

. qui summarize x

. display  "N =" ,`r(N)´, " mean ="`r(mean)´
N = 100  mean =10.075665

. sample 50, count
(50 observations deleted)

. gen fpcvar = 50/100

. gen pwvar = 100/50

. mean x

Mean estimation                    Number of obs   =       50

--------------------------------------------------------------
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
           x |   10.12268   .1299423      9.861551    10.38381
--------------------------------------------------------------
```

```
. svyset [pw=pwvar], fpc(fpcvar)

      pweight: pwvar
          VCE: linearized
  Single unit: missing
     Strata 1: <one>
         SU 1: <observations>
        FPC 1: fpcvar

. svy: mean x
(running mean on estimation sample)

Survey: Mean estimation

Number of strata =       1          Number of obs    =      50
Number of PSUs   =      50          Population size  =     100
                                    Design df        =      49

--------------------------------------------------------------
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
           x |   10.04064   .1075293      9.824552    10.25673
--------------------------------------------------------------
```

Simple design with two strata

We want to estimate the mean for the age of a population; we have a stratified sample, with two strata (region), and a SRS within each region

Region	Population mean	N	n
1	40	500	30
2	50	200	50

true mean: (1/700)*(500*40 + 200*50) = 42.857

```
. use age1, clear

. gen fpcvar = cond(region ==1, 30/500, 50/200)

. gen pwvar = cond(region ==1, 500/30, 200/50)
```

```
. svyset [pw=pwvar], strata(region) fpc(fpcvar)

      pweight: pwvar
          VCE: linearized
  Single unit: missing
     Strata 1: region
         SU 1: <observations>
        FPC 1: fpcvar

. svy: mean age
(running mean on estimation sample)

Survey: Mean estimation

Number of strata =       2          Number of obs     =       80
Number of PSUs   =      80          Population size   =      700
                                    Design df         =       78

--------------------------------------------------------------
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |   43.12323   .2530591      42.61942    43.62703
--------------------------------------------------------------
```

Adding PSUs to the design

We want to know the income per household in a certain city, and we don't have a list of households. Instead of trying to create a list of households, it would be more practical to sample blocks. Each block would be considered a sampling unit. Our setting would be:

```
svyset block [pw = pwvar], fpc(fpcvar)
```

If instead of sampling clusters from the city, we first divided the city into regions and then, within each region, we sampled blocks (eventually with different criteria among regions), our setting would be:

```
svyset block [pw = pwvar], strata(region)  ///
            fpc(fpcvar)
```

Multistage designs

We want to perform a survey on the eating habits of children attending elementary schools.
A possible design would be: perform samples independently on each state. For each state, perform a random sample of counties. Within each county, perform a random sample of schools, and interview each student for the selected schools.

```
svyset  county  [pw = pwvar], strata(state) fpc(fpcvar) || ///
            school, fpc(fpcvar2)
```

If within each school we stratify per grade and sample students independently on each grade, then we need to add another level:

```
svyset  county  [pw = pwvar], strata(state) fpc(fpcvar)  ///
                  || school, fpc(fpcvar2) ///
                  || student, fpc(fpcvar3) strata(grade)
```

Estimation

After declaring your survey design with `svyset`, you only need to use the `svy` prefix for your supported estimation command.

```
. webuse nhanes2, clear

. svyset psu [pw=finalwgt], strata(strata)
(output omitted)
. svy: probit highbp weight i.region
(running probit on estimation sample)
Survey: Probit regression
Number of strata   =        31              Number of obs      =      10351
Number of PSUs     =        62              Population size    =  117157513
                                            Design df          =         31
                                            F(   4,     28)    =      51.58
                                            Prob > F           =     0.0000
```

highbp	Coef.	Linearized Std. Err.	t	P>|t|	[95% Conf.	Interval]
weight	.0229814	.0016002	14.36	0.000	.0197178	.0262449
region						
2	-.1367888	.1321226	-1.04	0.309	-.4062547	.1326772
3	-.0568238	.1284056	-0.44	0.661	-.3187087	.2050611
4	-.1563827	.1255029	-1.25	0.222	-.4123475	.0995822
_cons	-2.889106	.1712036	-16.88	0.000	-3.238278	-2.539934

Variance for totals

Total estimator: one-stage design.

- L strata $h = 1, .., L$
- $i = 1, ..., n_h$ PSU's are sampled from stratum h.
- Cluster i from stratum h is composed of $j = 1, \ldots, m_{hi}$ elements.

$$\hat{Y} = \sum_{h=1}^{L} \sum_{i=1}^{n_h} \sum_{j=1}^{m_{hi}} w_{hij} y_{hij}$$

$$\hat{V}(\hat{Y}) = \sum_{h=1}^{L} (1 - f_h) \frac{n_h}{n_h - 1} \sum_{i=1}^{n_h} (y_{hi} - \bar{y}_h)^2$$

where

- y_{hi} is the weighted PSU total for cluster hi
- $\bar{y}_h$ is the mean of PSU totals in stratum h

Variance for totals (cont)

Total estimation: multistage design

$$\hat{V}(\hat{Y}) = \sum_{h=1}^{L}(1 - f_h)\frac{n_h}{n_h - 1}\sum_{i=1}^{n_h}(y_{hi} - \bar{y}_h)^2 + \sum_{h=1}^{L} f_h * (\text{contribution from further stages})$$

Notice that:

- y_{hi} are estimated totals per PSU, not actual totals
- If we don't use f_h when we `svyset`, then we will be using only information on the primary stage
- If f_h are too small, further stages will contribute very little to the variance estimator.

Variance estimation: linearized for regression models

We use Taylor expansion for models that are fitted via estimation equations:

$$\hat{G}(\beta) = \sum_j w_j S(\beta; y_j, x_j) = 0$$

For example, for OLS, G are the normal equations; for (pseudo) `ml` estimation, G are the scores.

$$\hat{V}(\hat{\beta}) = D\hat{V}(\hat{G}(\beta))|_{\beta=\hat{\beta}} D'$$

where D is the inverse of the derivative of G with respect to β. We use the formulas for the total to estimate the variance of $\hat{G}(\beta)$).

Variance estimation: linearized for regression models

If we `svyset` with only `weights` and PSU, this is equivalent to using the "plain" command with `weights` and `vce(cluster)`.

```
. sysuse auto, clear
. svyset rep [pw=trunk]
(output omitted)
. svy: logit for mpg
(output omitted)
```

foreign	Coef.	Linearized Std. Err.	t	P>|t|	[95% Conf.	Interval]
mpg	.2167174	.1159963	1.87	0.135	-.10534	.5387749
_cons	-5.726014	2.838586	-2.02	0.114	-13.60719	2.155165

```
. logit for mpg [pw=trunk], vce(cluster rep)
(output omitted)
```

foreign	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
mpg	.2167174	.1159963	1.87	0.062	-.0106312	.4440661
_cons	-5.726014	2.838586	-2.02	0.044	-11.28954	-.1624873

Replication-based methods

- Allow us to compute variance estimates without having the whole design information
- They use a set of variables containing weights: estimation is performed using each variable weight, and all those results are used to compute the variance
- Different methods differ on how weights are computed, and therefore on the formula used to compute the variance.
- Currently, we have two replication-based methods: `jackknife` and `brr` (balanced repeated replications).
- In the near future, we will also have `bootstrap` and `sdr` (successive difference replications).

The bootstrap

By default, the bootstrap variance estimator is computed as:

$$\hat{V}(\hat{\theta}) = \frac{b}{r} \sum_{i=1}^{r} \left(\hat{\theta}_{(i)} - \bar{\theta}_{(.)}\right) \left(\hat{\theta}_{(i)} - \bar{\theta}_{(.)}\right)'$$

where

- $\hat{\theta}_{(i)}$ is the point estimate for the ith replication
- $\bar{\theta}_{(.)}$ is the mean of $\{\theta_{(1)}, \dots, \theta_{(r)}\}$

Computation of `bootstrap` vce for survey data requires that weights be supplied by user.
Available in the near future. In the meantime, you can use the user-written command `bs4rw` (use `findit` to locate it).

Bootstrap example

```
. use nmihs_bs, clear

. svyset [pw=finwgt], bsrweight(bsrw*)  vce(bootstrap)

      pweight: finwgt
          VCE: bootstrap
          MSE: off
    bsrweight: bsrw1 bsrw2 bsrw3 bsrw4 bsrw5 bsrw6 bsrw7 bsrw8 bsrw9 bsrw10
 (output omitted)              bsrw996 bsrw997 bsrw998 bsrw999 bsrw1000
  Single unit: missing
     Strata 1: <one>
         SU 1: <observations>
        FPC 1: <zero>

. svy, nodots: logit lowbw highbp

Survey: Logistic regression                      Number of obs     =      9949
                                                 Population size   = 1419516.8
                                                 Replications      =      1000
                                                 Wald chi2(1)      =      0.02
                                                 Prob > chi2       =    0.8894

------------------------------------------------------------------------------
             |   Observed   Bootstrap                         Normal-based
       lowbw |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      highbp |   .0731004   .5256499     0.14   0.889    -.9571545    1.103355
       _cons |  -2.751879   .1138499   -24.17   0.000    -2.975021   -2.528737
------------------------------------------------------------------------------
```

Bootstrap example (2)

```
. use nmihs_mbs, clear
. svyset [pw=finwgt], bsrweight(mbsrw*) bsn(5)  vce(bootstrap)
      pweight: finwgt
          VCE: bootstrap
          MSE: off
    bsrweight: mbsrw1 mbsrw2 mbsrw3 mbsrw4 mbsrw5 mbsrw6 mbsrw7 mbsrw8 mbsrw9
(output omitted)
               mbsrw199 mbsrw200
          bsn: 5
  Single unit: missing
     Strata 1: <one>
         SU 1: <observations>
        FPC 1: <zero>
. svy, nodots: logit lowbw highbp
Survey: Logistic regression                    Number of obs      =       9946
                                               Population size    = 3895561.7
                                               Replications       =        200
                                               Wald chi2(1)       =      49.16
                                               Prob > chi2        =     0.0000
```

lowbw	Observed Coef.	Bootstrap Std. Err.	z	P>|z|	Normal-based [95% Conf.	Interval]
highbp	.805297	.1148604	7.01	0.000	.5801747	1.030419
_cons	-2.655877	.0094716	-280.41	0.000	-2.674441	-2.637313

Other potential uses for svy + vce(bootstrap)

Besides survey data, this feature can be used with non-survey data:

- If you have a huge dataset, the "standard" `bootstrap` may take time because it needs to access the disk to `preserve` and `restore` the data for each iteration. It may be convenient to generate weights and use them with `svy, vce(bootstrap)` instead.
- The "standard" `bootstrap` can't be used with `weights`. You can use this feature to do that. Your replication weights needs to be adjusted by your sample weights. Our documentation shows a way to do it.
- This feature can be used to perform `bootstrap` with clusters, a way that would be more efficient than the "naive" one (i.e. using option `vce(cluster)` for `bootstrap` prefix). Our documentation also will show you a way to do that.

Estimation on subpopulations: subpop() vs if

```
. webuse nmihs, clear

. svy: mean birthwgt if agegrp ==1
(running mean on estimation sample)

Survey: Mean estimation

Number of strata =       6          Number of obs     =     1652
Number of PSUs   =    1652          Population size   =   477969
                                    Design df         =     1646

--------------------------------------------------------------
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
    birthwgt |   3205.137    16.9503      3171.891    3238.384
--------------------------------------------------------------

. gen u = agegrp == 1

. svy, subpop(u): mean birthwgt
(running mean on estimation sample)

Survey: Mean estimation

Number of strata =       6         Number of obs     =      9952
Number of PSUs   =    9952         Population size   =   3898763
                                   Subpop. no. obs   =      1652
                                   Subpop. size      = 477968.6
                                   Design df         =      9946

--------------------------------------------------------------
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
    birthwgt |   3205.137    18.5948      3168.688    3241.587
--------------------------------------------------------------
```

Estimation on subpopulations: Remarks

Unless you have a very good reason to do the opposite, always use `subpop()` (not `if/in`) when working with `svy` data.

- `if/in` restricts the data to the subset of observations that satisfy the condition, and performs the estimation as if it were the whole sample from a known "population".
- It underestimates the variance, because it ignores the fact that our sample was taken from the whole population (not from the subset with the `if` condition in the population), and then we restricted the analysis to the observations that happened to be in the subset.

Postestimation commands

Most post-estimation commands are also available for survey data: `lincom`, `nlcom`, `predict`, `predictnl`, `test`, `testnl`, work the same way as for "standard" estimations.

When performing a Wald test after `svy` estimation, degrees of feedom for `test` are adjusted according to the survey design.

The command `test` is particularly important for survey data, where likelihood-based commands (like `lrtest`) are not valid.

Performing a Wald test

```
. webuse nhanes2, clear

. svy: probit highbp weight i.region
(output omitted)

. test 2.region = 3.region

Adjusted Wald test

 ( 1)  [highbp]2.region - [highbp]3.region = 0

       F(  1,    31) =    0.61
            Prob > F =    0.4400
```

margins

We can use `margins` to compute marginal means and marginal effects. Here we compute the mean predicted probability of positive outcome per region.

```
. margins region, vce(unconditional)
Predictive margins                          Number of obs     =     10351
Expression   : Pr(highbp), predict()
```

	Margin	Linearized Std. Err.	t	P>|t|	[95% Conf.	Interval]
region						
1	.1220135	.0201861	6.04	0.000	.0808436	.1631833
2	.0980896	.0125024	7.85	0.000	.0725908	.1235885
3	.1116153	.0122231	9.13	0.000	.0866861	.1365446
4	.0949684	.0098451	9.65	0.000	.0748892	.1150475