Sampling II

LPO 9951 | Fall 2015

PURPOSE In the last lecture, we discussed a number of ways to properly estimate the means and variances of complex survey designs. In this lecture, we'll discuss how to use Stata's internal `svy` commands and various variance estimation methods to more easily and correctly estimate what we want.

Complex survey designs: Cluster sampling and stratification

In the NCES surveys you'll be using this semester, the designers combined a design that includes multistage cluster sampling with stratification. In ECLS, for example, the designers designated counties as *PSUs*. They next stratified the sample by creating strata that combined census region with msa status, percent minority, and per capita income. They then randomly selected schools within each *PSU* (schools were the *SSUs*) and then randomly selected kindergarteners within each school (students were the *TSUs*). They then created two strata for each school with Asian and Pacific Islander students in one stratum and all other students in the other. Students were randomly sampled within this second stratum. The target number of children per school was 24.

Weights in complex survey designs such as the one employed with ECLS are calculated via the same that we discussed in the last lecture. Nothing changes except for the layers of complexity. The good news, however, is that we a researchers don't have to compute the weights ourselves. Instead, we can use information provided by the survey makers.

The *PSUs* that are provided by NCES are what is known as "analysis *PSUs*". They aren't the identifier for the actual school or student. Instead, they are allocated within strata (many times 2 *PSU* per strata). Strata themselves may be analysis strata, that is, not the same strata that were used to run the survey. Oftentimes, this is done in service of further protecting the anonimity of participants. As far your analyses go, the end result is the same, but sometimes this can be a source of confusion.

Variance estimation in complex survey designs

There are four common options for estimating variance in complex survey designs:

1. Taylor series linearized estimates
2. Balanced repeated replication (BRR) estimates
3. Jackknife estimates
4. Bootstrap estimates

Remember that these are all estimates: you cannot directly compute the variance of quantities of interest from complex surveys. Instead, you must use one of these techniques, with trade-offs for each. We'll be using a couple of datasets for this lesson:

- *nhanes*, which is a health survey conducted using a complex survey design that comes with a variety of weights

- *nmihs_bs*, which is a survey of births that comes with bootstrap replicate weights

Let's start with the *nhanes* dataset from which we'd like to get average height weight and age for the US population. First, let's get the naive estimate:

```
. webuse nhanes2f, clear

. // naive mean
. mean age height weight

Mean estimation                    Number of obs   =     10,337

--------------------------------------------------------------
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |    47.5637   .1693381      47.23177    47.89564
      height |   167.6512   .0950124       167.465    167.8375
      weight |   71.90088   .1510277      71.60484    72.19692
--------------------------------------------------------------
```

We can also take a look at the sampling design, particularly the designation of strata and *PSUs*:

```
. tab stratid psuid

   stratum |   primary sampling
identifier |     unit, 1 or 2
    , 1-32 |         1          2 |     Total
-----------+----------------------+----------
         1 |       215        165 |       380 
         2 |       118         67 |       185 
         3 |       199        149 |       348 
         4 |       231        229 |       460 
         5 |       147        105 |       252 
         6 |       167        131 |       298 
         7 |       270        206 |       476 
         8 |       179        158 |       337 
         9 |       143        100 |       243 
        10 |       143        119 |       262 
        11 |       120        155 |       275 
        12 |       170        144 |       314 
        13 |       154        188 |       342 
        14 |       205        200 |       405 
        15 |       189        191 |       380 
        16 |       177        159 |       336 
        17 |       180        213 |       393 
        18 |       144        215 |       359 
        20 |       158        125 |       283 
        21 |       102        111 |       213 
        22 |       173        128 |       301 
        23 |       182        158 |       340 
        24 |       202        232 |       434 
        25 |       139        115 |       254 
        26 |       132        129 |       261 
        27 |       144        139 |       283 
        28 |       135        163 |       298 
        29 |       287        215 |       502 
        30 |       166        199 |       365 
        31 |       143        165 |       308 
        32 |       239        211 |       450 
```

```
-----------+----------------------+----------
     Total |     5,353      4,984 |    10,337
```

We can use the weights supplied with *nhanes* to get accurate estimates of the means, but the variance estimates will be off:

```
. mean age height weight [pw = finalwgt]

Mean estimation                   Number of obs   =     10,337

--------------------------------------------------------------
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |   42.23732   .1617236      41.92031    42.55433
      height |   168.4625   .1139787      168.2391     168.686
      weight |   71.90869   .1802768      71.55532    72.26207
--------------------------------------------------------------
```

svyset and svy: <command>

To aid in the analysis of complex survey data, Stata has incorporated the `svyset` command and the `svy:` prefix, with its suite of commands. With `svyset`, you can set the *PSU* (and *SSU* and *TSU* if applicable), the weights, and the type of variance estimation along with the variance weights (if applicable). Once set, most Stata estimation commands such as `mean` can be combined with `svy:` in order to produce correct estimates.

Variance estimators

Taylor series linearized estimates

Taylor series linearized estimates are based on the general strategy of Taylor series estimation, which is used to linearize a non-linear function in order to describe the function in question. In this case, a Taylor series is used to approximate the function, and the variance of the result is the estimate of the variance.

The basic intuition behind a linearized estimate is that the variance in a complex survey will be a nonlinear function of the set of variances calculated within each stratum. We can calculate these, then use the first derivative of the function that would calculate the actual variance as a first order approximation of the actual variance. This works well enough in practice. To do this, you absolutely must have multiple *PSUs* in each stratum so you can calculate variance within each stratum.

This is the most common method and is used as the default by Stata. You must, however, have within-stratum variance among *PSUs* for this to work, which means that you must have at least two *PSUs* per stratum. This lonely PSU problem is common and difficult to deal with. We'll return the lonely PSU later.

To set up a dataset to use linearized estimates in Stata, we use the `svyset` command:

```
. // set survey characteristics with svyset
. svyset psuid [pweight = finalwgt], strata(stratid)

      pweight: finalwgt
          VCE: linearized
  Single unit: missing
     Strata 1: stratid
         SU 1: psuid
        FPC 1: <zero>
```

Now that we've set the data, every time we want estimates that reflect the sampling design, we use the `svy: <command>` format:

```
. svy: mean age height weight
(running mean on estimation sample)

Survey: Mean estimation

Number of strata =      31      Number of obs   =       10,337
Number of PSUs   =      62      Population size =  117,023,659
                                Design df       =           31

--------------------------------------------------------------
             |             Linearized
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |   42.23732   .3034412      41.61844    42.85619
      height |   168.4625   .1471709      168.1624    168.7627
      weight |   71.90869   .1672315      71.56762    72.24976
--------------------------------------------------------------
```

As you can see, the parameter estimates (means) are exactly the same as using the weighted sample, but the standard errors are quite different: nearly twice as large for age, but actually smaller for weight.

Balanced repeated replication (BRR) estimates

In a balanced repeated replication (BRR) design, the quantity of interests is estimated repeatedly by using half the sample at a time. In a survey which is designed with BRR in mind, each sampling stratum contains two *PSUs*. BRR proceeds by estimating the quantity of interest from one of the *PSUs* within each stratum. For H strata, 2^H replications are done, and the variance of the quantity of interest across these strata forms the basis for the estimate.

BRR weights are usually supplied with a survey. These weights result in appropriate half samples being formed across strata. BRR weights should generally be used when the sample was designed with them in mind, and not elsewhere. This can be a serious complication when survey data are subset.

To get variance estimates using BRR in stata, you either need to have a set of replicate weights set up or you need to create a set of balanced replicates yourself. If the data has BRR weights it's simple:

```
. webuse nhanes2brr, clear

. // svyset automagically
. svyset

      pweight: finalwgt
          VCE: brr
          MSE: off
    brrweight: brr_1 brr_2 brr_3 brr_4 brr_5 brr_6 brr_7 brr_8 brr_9 brr_10 brr_11
               brr_12 brr_13 brr_14 brr_15 brr_16 brr_17 brr_18 brr_19 brr_20 brr_21
               brr_22 brr_23 brr_24 brr_25 brr_26 brr_27 brr_28 brr_29 brr_30 brr_31
               brr_32
  Single unit: missing
     Strata 1: <one>
         SU 1: <observations>
```

```
        FPC 1: <zero>

. // compute mean using svy pre-command and brr weights
. svy: mean age height weight
(running mean on estimation sample)

BRR replications (32)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
................................

Survey: Mean estimation          Number of obs   =      10,351
                                 Population size =  117,157,513
                                 Replications    =          32
                                 Design df       =          31

--------------------------------------------------------------
             |                 BRR
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |   42.25264   .3013406      41.63805    42.86723
      height |   168.4599     .14663      168.1608    168.7589
      weight |   71.90064   .1656452       71.5628    72.23847
--------------------------------------------------------------
```

If you don't have the data set up this way, you need to create a Hadamard with dimensions equal to the number of strata. Hadamard matrices are special in that they are square matrices comprised of 1s and -1s that arranged in such a way that each row and column sums to zero (equal numbers of ones and negative ones) and adjacent rows/columns are orthogonal (correlation of zero).

```
. webuse nhanes2, clear

. // create Hadamard matrix in Mata
. mata: h2 = (1, 1 \ 1, -1)

. mata: h4 = h2 # h2

. mata: h8 = h2 # h4

. mata: h16 = h2 # h8

. mata: h32 = h2 # h16

. // check row and column sums
. mata: rowsum(h32)
         1
     +------+
   1 |  32  |
   2 |   0  |
   3 |   0  |
   4 |   0  |
   5 |   0  |
   6 |   0  |
   7 |   0  |
   8 |   0  |
```

```
   9 |   0  |
  10 |   0  |
  11 |   0  |
  12 |   0  |
  13 |   0  |
  14 |   0  |
  15 |   0  |
  16 |   0  |
  17 |   0  |
  18 |   0  |
  19 |   0  |
  20 |   0  |
  21 |   0  |
  22 |   0  |
  23 |   0  |
  24 |   0  |
  25 |   0  |
  26 |   0  |
  27 |   0  |
  28 |   0  |
  29 |   0  |
  30 |   0  |
  31 |   0  |
  32 |   0  |
     +------+

. mata: colsum(h32)
        1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
    +---------------------------------------------------------------------------------
  1 |  32    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    +---------------------------------------------------------------------------------
       17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
     ---------------------------------------------------------------------------------+
  1     0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 |
     ---------------------------------------------------------------------------------+

. // save Mata matrix in Stata matrix form
. mata: st_matrix("h32", h32)
```

Now that we've made our matrix, we can use it with the BRR command to get our estimates:

```
. svy brr, hadamard(h32): mean age height weight
(running mean on estimation sample)

BRR replications (32)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
................................

Survey: Mean estimation

Number of strata =      31      Number of obs   =      10,351
Number of PSUs   =      62      Population size = 117,157,513
                                Replications    =          32
                                Design df       =          31
```

```
-------------------------------------------------------------
             |                BRR
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+-----------------------------------------------
         age |   42.25264   .2779063      41.68585    42.81944
      height |   168.4599   .1411963      168.1719    168.7479
      weight |   71.90064   .1620071      71.57022    72.23105
-------------------------------------------------------------
```

Jackknife estimates

The Jackknife is a general strategy for variance estimation, so named by Tukey because of its general usefulness. The strategy for creating a jackknifed estimate is to delete every observation save one, then estimate the quantity of interest. This is repeated for every single observation in the dataset. The variance of every estimate computed provides an estimate of the variance for the quantity of interest.

In a complex sample, this is done by *PSUs*, deleting each *PSU* one at a time and re-weighting the observations within the stratum, then calculating the parameter of interest. The variance of these parameters estimates is the within-stratum variance estimate. The within stratum variances calculated this way are then averaged across strata to give the final variance estimate.

The jackknife is best used when Taylor series estimation cannot be done, for instance in the case of lonely *PSUs*.

In Stata, the command is:

```
. webuse nhanes2jknife, clear

. // set svyset using jackknife weigts
. svyset [pweight = finalwgt], jkrweight(jkw_*) vce(jackknife)

      pweight: finalwgt
          VCE: jackknife
          MSE: off
    jkrweight: jkw_1 jkw_2 jkw_3 jkw_4 jkw_5 jkw_6 jkw_7 jkw_8 jkw_9 jkw_10 jkw_11
               jkw_12 jkw_13 jkw_14 jkw_15 jkw_16 jkw_17 jkw_18 jkw_19 jkw_20 jkw_21
               jkw_22 jkw_23 jkw_24 jkw_25 jkw_26 jkw_27 jkw_28 jkw_29 jkw_30 jkw_31
               jkw_32 jkw_33 jkw_34 jkw_35 jkw_36 jkw_37 jkw_38 jkw_39 jkw_40 jkw_41
               jkw_42 jkw_43 jkw_44 jkw_45 jkw_46 jkw_47 jkw_48 jkw_49 jkw_50 jkw_51
               jkw_52 jkw_53 jkw_54 jkw_55 jkw_56 jkw_57 jkw_58 jkw_59 jkw_60 jkw_61
               jkw_62
  Single unit: missing
     Strata 1: <one>
         SU 1: <observations>
        FPC 1: <zero>
```

Now we can compare the naive estimates with the `svyset` estimates:

```
. mean age weight height

Mean estimation                   Number of obs   =     10,351

-------------------------------------------------------------
```

```
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |   47.57965   .1692044      47.24798    47.91133
      weight |   71.89752   .1509381      71.60165    72.19339
      height |   167.6509   .0949079      167.4648    167.8369
--------------------------------------------------------------

. // compute mean with jackknife weights
. svy: mean age weight height
(running mean on estimation sample)

Jackknife replications (62)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
............

Survey: Mean estimation

Number of strata =      31      Number of obs   =       10,351
                                Population size =  117,157,513
                                Replications    =           62
                                Design df       =           31

--------------------------------------------------------------
             |             Jackknife
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |   42.25264   .3026765      41.63533    42.86995
      weight |   71.90064   .1654453      71.56321    72.23806
      height |   168.4599   .1466141      168.1609    168.7589
--------------------------------------------------------------
```

Bootstrap estimates

The bootstrap is a more general method than the jackknife. Bootstrapping involves repeatedly resampling within the sample itself and generating estimates of the quantity of interest. The variance of these replications (usually many, many replications) provides an estimate of the total variance. In NCES surveys, within stratum bootstrapping can be used, with the sum of the variances obtained used as an estimate of the population variance. Bootstrapping is an accurate, but computationally intense method of variance estimation.

As with the jackknife, bootstrapping must be accomplished by deleting each *PSU* within the stratum one at a time, re-weighting, calculating the estimate, than calculating the bootstrap variance estimate from the compiled samples.

```
. webuse nmihs_bs, clear

. // svyset
. svyset idnum [pweight = finwgt], vce(bootstrap) bsrweight(bsrw*)

      pweight: finwgt
          VCE: bootstrap
          MSE: off
    bsrweight: bsrw1 bsrw2 bsrw3 bsrw4 bsrw5 bsrw6 bsrw7 bsrw8 bsrw9 bsrw10 bsrw11
               <.................................................................>
```

```
              bsrw993 bsrw994 bsrw995 bsrw996 bsrw997 bsrw998 bsrw999 bsrw1000
  Single unit: missing
     Strata 1: <one>
         SU 1: idnum
        FPC 1: <zero>

. // convert birth weight grams to lbs for the Americans
. gen birthwgtlbs = birthwgt * 0.0022046
(7 missing values generated)

. // compute naive mean birthweight
. mean birthwgtlbs

Mean estimation                   Number of obs   =      9,946

--------------------------------------------------------------
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
 birthwgtlbs |   6.272294   .0217405      6.229678     6.31491
--------------------------------------------------------------

. // compute mean with svy bootstrap
. svy: mean birthwgtlbs
(running mean on estimation sample)

Bootstrap replications (1000)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
..................................................   100
..................................................   150
..................................................   200
..................................................   250
..................................................   300
..................................................   350
..................................................   400
..................................................   450
..................................................   500
..................................................   550
..................................................   600
..................................................   650
..................................................   700
..................................................   750
..................................................   800
..................................................   850
..................................................   900
..................................................   950
..................................................  1000

Survey: Mean estimation           Number of obs   =      9,946
                                  Population size =  3,895,562
                                  Replications    =      1,000

--------------------------------------------------------------
             |   Observed   Bootstrap         Normal-based
```

```
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
 birthwgtlbs |    7.39743   .0143754      7.369255    7.425606
--------------------------------------------------------------
```

Lonely *PSUs*

The most common problem that students have with complex surveys is what is known as “lonely *PSUs*.” When you subset the data, you may very well end up with a sample that does not have mutliple *PSUs* per stratum. There are several options for what do in this case:

- Eliminate the offending data by dropping strata with singleton *PSUs*. This is a terrible idea.
- Reassign the *PSU* to a neighboring stratum. This is okay, but you must have a reason why you’re doing this.
- Assign a variance to the stratum with a singleton *PSU*. This could be the average of the variance across the other strata. This process is also known as “scaling” and generally is okat, but you should take a look at how different this stratum is from the others before proceeding.

The svyset command includes three possible options for dealing with loney *PSUs*. Based on the above, I recommend you use the `singleunit(scaled)` command, but with caution and full knowledge of the implications for your estimates.

Init: 23 August 2015; Updated: 24 August 2015