svyset — Declare survey design for dataset

Description

svyset manages the survey analysis settings of a dataset. You use svyset to designate variables that contain information about the survey design, such as the sampling units and weights. svyset is also used to specify other design characteristics, such as the number of sampling stages and the sampling method, and analysis defaults, such as the method for variance estimation. You must svyset your data before using any svy command; see [SVY] **svy estimation**.

svyset without arguments reports the current settings. svyset, clear removes the current survey settings.

Quick start

One-stage design with sampling weight wvar1, strata defined by levels of svar, and sampling units identified by su1

```
svyset su1 [pweight=wvar1], strata(svar)
```

Two-stage design with finite population correction fpc and _n indicating second-stage sampling units are the sampled individuals

```
svyset su1 [pweight=wvar1], strata(svar) fpc(fpc) || _n
```

Two-stage design with second-stage clustering defined by su2

```
svyset su1 [pweight=wvar1], strata(svar) fpc(fpc) || su2
```

Jackknife variance estimation as the default for svy commands, with sampling weight wvar2, and replicate-weight variables rwvar*

```
svyset [pweight=wvar2], vce(jackknife) jkrweight(rwvar*)
```

Same as above, but use the MSE formula

```
svyset [pweight=wvar2], vce(jackknife) jkrweight(rwvar*) mse
```

Display current survey settings

```
svyset
```

Clear current survey settings

```
svyset, clear
```

Menu

Statistics > Survey data analysis > Setup and utilities > Declare survey design for dataset

Syntax

Single-stage design

`svyset` [*psu*] [*weight*] [, *design_options options*]

Multiple-stage design

`svyset` *psu* [*weight*] [, *design_options*] [`||` *ssu* , *design_options*] ... [*options*]

Clear the current settings

`svyset, clear`

Report the current settings

`svyset`

psu identifies the primary sampling units and may be `_n` or *varname*. In the single-stage syntax, *psu* is optional and defaults to `_n`.

`_n` indicates that individuals were randomly sampled if the design does not involve clustered sampling.

varname contains identifiers for the clusters in a clustered sampling design.

ssu is `_n` or *varname* containing identifiers for sampling units (clusters) in subsequent stages of the survey design.

`_n` indicates that individuals were randomly sampled within the last sampling stage.

design_options	Description
Main	
`strata(`*varname*`)`	variable identifying strata
`fpc(`*varname*`)`	finite population correction
`weight(`*varname*`)`	stage-level sampling weight

options	Description
Weights	
brrweight(*varlist*)	balanced repeated replicate (BRR) weights
fay(#)	Fay's adjustment
bsrweight(*varlist*)	bootstrap replicate weights
bsn(#)	bootstrap mean-weight adjustment
jkrweight(*varlist*, *jkropts*)	jackknife replicate weights
sdrweight(*varlist*, *sdropts*)	successive difference replicate (SDR) weights
SE	
vce(linearized)	Taylor linearized variance estimation
vce(bootstrap)	bootstrap variance estimation
vce(brr)	BRR variance estimation
vce(jackknife)	jackknife variance estimation
vce(sdr)	SDR variance estimation
dof(#)	design degrees of freedom
mse	use the MSE formula with vce(bootstrap), vce(brr), vce(jackknife), or vce(sdr)
singleunit(*method*)	strata with a single sampling unit; *method* may be missing, certainty, scaled, or centered
Poststratification	
poststrata(*varname*)	variable identifying poststrata
postweight(*varname*)	poststratum population sizes
Calibration	
rake(*varlist*, *calopts*)	adjust weights using the raking-ratio method
regress(*varlist*, *calopts*)	adjust weights using linear regression calibration
clear	clear all settings from the data
noclear	change some of the settings without clearing the others
clear(*opnames*)	clear specified settings without clearing all others; *opnames* may be one or more of weight, vce, dof, mse, bsrweight, brrweight, jkrweight, sdrweight, poststrata, rake, or regress

collect is allowed; see **[U] 11.1.10 Prefix commands**.
pweights and iweights are allowed; see **[U] 11.1.6 weight**.
clear, noclear, and clear() are not shown in the dialog box.

jkropts	Description
stratum(# [# ...])	stratum identifier for each jackknife replicate weight
fpc(# [# ...])	finite population correction for each jackknife replicate weight
multiplier(# [# ...])	variance multiplier for each jackknife replicate weight
reset	reset characteristics for each jackknife replicate weight

sdropts	Description
fpc(# [# ...])	finite population correction for each SDR weight

calopts	Description
* totals(*spec*)	population totals
noconstant	suppress constant term
ll(#)	lower limit for weight ratios
ul(#)	upper limit for weight ratios
iterate(#)	maximum number of iterations
tolerance(#)	convergence tolerance
force	allow calibration adjustments that failed to converge

*totals() is required.

Options

Main

strata(*varname*) specifies the name of a variable (numeric or string) that contains stratum identifiers.

fpc(*varname*) requests a finite population correction for the variance estimates. If *varname* has values less than or equal to 1, it is interpreted as a stratum sampling rate $f_h = n_h/N_h$, where n_h = number of units sampled from stratum h and N_h = total number of units in the population belonging to stratum h. If *varname* has values greater than or equal to n_h, it is interpreted as containing N_h. It is an error for *varname* to have values between 1 and n_h or to have a mixture of sampling rates and stratum sizes.

weight(*varname*) specifies a stage-level sampling weight variable. For most models, stage-level sampling weights are multiplied together to create a single observation-level sampling weight variable used for weighted estimation. For commands such as gsem and meglm, each stage-level weight variable is assumed to correspond with a hierarchical group level in the model and is used to compute the pseudolikelihood at that associated group level. Stage-level sampling weights are required to be constant within their corresponding group level. For examples of fitting a multilevel model with stage-level sampling weights, see example 5 and example 6 in [ME] **meglm**.

Weights

brrweight(*varlist*) specifies the replicate-weight variables to be used with vce(brr) or with svy brr.

fay(#) specifies Fay's adjustment (Judkins 1990). The value specified in fay(#) is used to adjust the BRR weights and is present in the BRR variance formulas.

The sampling weight of the selected PSUs for a given replicate is multiplied by 2-#, where the sampling weight for the unselected PSUs is multiplied by #. When brrweight(*varlist*) is specified, the replicate-weight variables in *varlist* are assumed to be adjusted using #.

fay(0) is the default and is equivalent to the original BRR method. # must be between 0 and 2, inclusive, and excluding 1. fay(1) is not allowed because this results in unadjusted weights.

bsrweight(*varlist*) specifies the replicate-weight variables to be used with vce(bootstrap) or with svy bootstrap.

bsn(#) specifies that # bootstrap replicate-weight variables were used to generate each bootstrap mean-weight variable specified in the bsrweight() option. The default is bsn(1). The value specified in bsn(#) is used to adjust the variance estimate to account for mean bootstrap weights.

`jkrweight(`*varlist*`,` *jkropts*`)` specifies the replicate-weight variables to be used with `vce(jackknife)` or with `svy jackknife`.

The following *jkropts* set characteristics on the jackknife replicate-weight variables. If one value is specified, all the specified jackknife replicate-weight variables will be supplied with the same characteristic. If multiple values are specified, each replicate-weight variable will be supplied with the corresponding value according to the order specified. *jkropts* are not shown in the dialog box.

`stratum(`# [# ...]`)` specifies an identifier for the stratum in which the sampling weights have been adjusted.

`fpc(`# [# ...]`)` specifies the FPC value to be added as a characteristic of the jackknife replicate-weight variables. The values set by this suboption have the same interpretation as the `fpc(`*varname*`)` option.

`multiplier(`# [# ...]`)` specifies the value of a jackknife multiplier to be added as a characteristic of the jackknife replicate-weight variables.

`reset` indicates that the characteristics for the replicate-weight variables may be overwritten or reset to the default, if they exist.

`sdrweight(`*varlist*`,` *sdropts*`)` specifies the replicate-weight variables to be used with `vce(sdr)` or with `svy sdr`. The following *sdropts* is available:

`fpc(`#`)` specifies the FPC value associated with the SDR weights. The value set by this suboption has the same interpretation as the `fpc(`*varname*`)` option. This option is not shown in the dialog box.

SE

`vce(`*vcetype*`)` specifies the default method for variance estimation; see [SVY] **Variance estimation**.

`vce(linearized)` sets the default to Taylor linearization.

`vce(bootstrap)` sets the default to the bootstrap; also see [SVY] **svy bootstrap**.

`vce(brr)` sets the default to BRR; also see [SVY] **svy brr**.

`vce(jackknife)` sets the default to the jackknife; also see [SVY] **svy jackknife**.

`vce(sdr)` sets the default to the SDR; also see [SVY] **svy sdr**.

`dof(`#`)` specifies the design degrees of freedom, overriding the default calculation, $\mathrm{df} = N_{\mathrm{psu}} - N_{\mathrm{strata}}$.

`mse` specifies that the MSE formula be used when `vce(bootstrap)`, `vce(brr)`, `vce(jackknife)`, or `vce(sdr)` is specified. This option requires `vce(bootstrap)`, `vce(brr)`, `vce(jackknife)`, or `vce(sdr)`.

`singleunit(`*method*`)` specifies how to handle strata with one sampling unit.

`singleunit(missing)` results in missing values for the standard errors and is the default.

`singleunit(certainty)` causes strata with single sampling units to be treated as certainty units. Certainty units contribute nothing to the standard error.

`singleunit(scaled)` results in a scaled version of `singleunit(certainty)`. The scaling factor comes from using the average of the variances from the strata with multiple sampling units for each stratum with one sampling unit.

`singleunit(centered)` specifies that strata with one sampling unit are centered at the grand mean instead of the stratum mean.

Poststratification

`poststrata(`*varname*`)` specifies the name of the variable (numeric or string) that contains poststratum identifiers. See [SVY] **Poststratification** for more information.

`postweight(`*varname*`)` specifies the name of the numeric variable that contains poststratum population totals (or sizes), that is, the number of elementary sampling units in the population within each poststratum. See [SVY] **Poststratification** for more information.

Calibration

`rake(`*varlist*`,` *calopts*`)` and `regress(`*varlist*`,` *calopts*`)` specify that the sampling weights be adjusted using a calibration adjustment. See [SVY] **Calibration** for more information.

`rake()` specifies that the weights be adjusted by the raking-ratio method.

`regress()` specifies that the weights be adjusted by linear regression.

The following *calopts* are available:

`totals(`*spec*`)` is required. It specifies the population totals corresponding to the variables specified in *varlist*. *spec* is one of

matname [`, skip copy`]

{ [*eqname*`:`]*name* = # | `/`*eqname* = # } [. . .]

# [# . . .]`, copy`

That is, *spec* may be a matrix name, for example, `totals(poptotals)`; a list of variable names in *varlist* with their population totals, for example, `totals(_cons=1300 dogs=850 cats=450)`; or a list of values, for example, `totals(850 450 1300)`.

`skip` specifies that any parameters found in the specified totals vector that are not also found in the model be ignored. The default action is to issue an error message.

`copy` specifies that the list of values or the totals vector be copied into the population-totals vector by position rather than by name.

`noconstant` suppresses the intercept in the linear regression adjustment.

`ll(`#`)` specifies a lower limit for the weight ratios for truncated linear calibration.

`ul(`#`)` specifies an upper limit for the weight ratios for truncated linear calibration.

`iterate(`#`)` specifies the maximum number of iterations. When the number of iterations equals `iterate()`, the calibration adjustment stops and presents a note. The default is `iterate(1000)`.

`tolerance(`#`)` specifies the tolerance for the Lagrange multiplier in the calibration equations. Convergence is achieved when the relative change in the Lagrange multiplier from one iteration to the next is less than or equal to `tolerance()`. The default is `tolerance(1e-7)`.

`force` prevents `svy` estimation from exiting with an error if the calibration adjustment fails to converge.

The following options are available with svyset but are not shown in the dialog box:

clear clears all the settings from the data. Typing

```
. svyset, clear
```

clears the survey design characteristics from the data in memory. Although this option may be specified with some of the other svyset options, it is redundant because svyset automatically clears the previous settings before setting new survey design characteristics.

noclear allows some of the options in *options* to be changed without clearing all the other settings. This option is not allowed with *psu*, *ssu*, *design_options*, or clear.

clear(*opnames*) allows some of the options in *options* to be cleared without clearing all the other settings. *opnames* refers to an option name and may be one or more of the following: weight, vce, dof, mse, brrweight, bsrweight, jkrweight, sdrweight, poststrata, rake, or regress.

This option implies the noclear option.

Remarks and examples

Remarks are presented under the following headings:

Introduction to survey design characteristics

Stata's suite of commands for survey data analysis relies on properly identified survey design characteristics for point estimation, model fitting, and variance estimation. In fact, the svy prefix will report an error if no survey design characteristics have been identified using svyset. Settings made by svyset are saved with a dataset. So, if a dataset is saved after it has been svyset, it does not have to be set again.

Typical survey design characteristics include sampling weights, one or more stages of clustered sampling, and stratification. O'Donnell et al. (2008, 26–27) show four survey sample designs with the corresponding svyset specification. Use svyset to declare your dataset to be complex survey data by specifying the survey design variables. We will use the following contrived dataset for the examples in this section.

```
. use https://www.stata-press.com/data/r19/stage5a
```

▷ Example 1: Simple random sampling with replacement

Use _n for *psu* to specify that the primary sampling units (PSUs) are the sampled individuals.

```
. svyset _n
Sampling weights: <none>
             VCE: linearized
     Single unit: missing
        Strata 1: <one>
 Sampling unit 1: <observations>
           FPC 1: <zero>
```

The output from svyset states that there are no sampling weights (each observation is given a sampling weight of 1), there is only one stratum (which is the same as no stratification), and the PSUs are the observed individuals.

◁

▷ Example 2: One-stage clustered design with stratification

The most commonly specified design, one-stage clustered design with stratification, can be used to approximate multiple-stage designs when only the first-stage information is available. In this design, the population is partitioned into strata and the PSUs are sampled independently within each stratum. A dataset from this design will have a variable that identifies the strata, another variable that identifies the PSUs, and a variable containing the sampling weights. Let's assume that these variables are, respectively, strata, su1, and pw.

```
. svyset su1 [pweight=pw], strata(strata)

Sampling weights: pw
             VCE: linearized
     Single unit: missing
        Strata 1: strata
 Sampling unit 1: su1
           FPC 1: <zero>
```

◁

▷ Example 3: Two-stage designs

In two-stage designs, the PSUs are sampled without replacement and then collections of individuals are sampled within the selected PSUs. svyset uses || (double "or" bars) to separate the stage-specific design specifications. The first-stage information is specified before ||, and the second-stage information is specified afterward. We will assume that the variables containing the finite population correction (FPC) information for the two stages are named fpc1 and fpc2; see *Finite population correction (FPC)* for a discussion about the FPC.

Use _n for *ssu* to specify that the second-stage sampling units are the sampled individuals.

```
. svyset su1 [pweight=pw], fpc(fpc1) || _n, fpc(fpc2)

Sampling weights: pw
             VCE: linearized
     Single unit: missing
        Strata 1: <one>
 Sampling unit 1: su1
           FPC 1: fpc1
        Strata 2: <one>
 Sampling unit 2: <observations>
           FPC 2: fpc2
```

Suppose that su2 identifies the clusters of individuals sampled in the second stage.

```
. svyset su1 [pweight=pw], fpc(fpc1) || su2, fpc(fpc2)

Sampling weights: pw
             VCE: linearized
     Single unit: missing
        Strata 1: <one>
 Sampling unit 1: su1
           FPC 1: fpc1
        Strata 2: <one>
 Sampling unit 2: su2
           FPC 2: fpc2
```

Stratification can take place in one or both of the sampling stages. Suppose that `strata` identifies the second-stage strata and the first stage was not stratified.

```
. svyset su1 [pweight=pw], fpc(fpc1) || su2, fpc(fpc2) strata(strata)

Sampling weights: pw
             VCE: linearized
     Single unit: missing
        Strata 1: <one>
 Sampling unit 1: su1
           FPC 1: fpc1
        Strata 2: strata
 Sampling unit 2: su2
           FPC 2: fpc2
```

◁

▷ Example 4: Multiple-stage designs

Specifying designs with three or more stages is not much more difficult than specifying two-stage designs. Each stage will have its own variables for identifying strata, sampling units, and the FPC. Not all stages will be stratified and some will be sampled with replacement; thus some stages may not have a variable for identifying strata or the FPC.

Suppose that we have a three-stage design with variables `su`*#* and `fpc`*#* for the sampling unit and FPC information in stage *#*. Also assume that the design called for stratification in the first stage only.

```
. svyset su1 [pweight=pw], fpc(fpc1) strata(strata)
>         || su2, fpc(fpc2)
>         || su3, fpc(fpc3)

Sampling weights: pw
             VCE: linearized
     Single unit: missing
        Strata 1: strata
 Sampling unit 1: su1
           FPC 1: fpc1
        Strata 2: <one>
 Sampling unit 2: su2
           FPC 2: fpc2
        Strata 3: <one>
 Sampling unit 3: su3
           FPC 3: fpc3
```

Use _n for *ssu* in the last stage if the individuals are sampled within the third stage of clustered sampling.

```
. svyset su1 [pweight=pw], fpc(fpc1) strata(strata)
>         || su2, fpc(fpc2)
>         || su3, fpc(fpc3)
>         || _n

Sampling weights: pw
             VCE: linearized
     Single unit: missing
        Strata 1: strata
 Sampling unit 1: su1
           FPC 1: fpc1
        Strata 2: <one>
 Sampling unit 2: su2
           FPC 2: fpc2
        Strata 3: <one>
 Sampling unit 3: su3
           FPC 3: fpc3
        Strata 4: <one>
 Sampling unit 4: <observations>
           FPC 4: <zero>
```

◁

Finite population correction (FPC)

An FPC accounts for the reduction in variance that occurs when sampling *without* replacement from a finite population compared to sampling *with* replacement from the same population. Specifying an FPC variable for stage i indicates that the sampling units in that stage were sampled without replacement. See Cochran (1977) for an introduction to variance estimation and sampling without replacement.

▷ Example 5

Consider the following dataset:

```
. use https://www.stata-press.com/data/r19/fpc

. list

     +------------------------------------------+
     | stratid   psuid   weight   nh   Nh     x |
     |------------------------------------------|
  1. |       1       1        3    5   15   2.8 |
  2. |       1       2        3    5   15   4.1 |
  3. |       1       3        3    5   15   6.8 |
  4. |       1       4        3    5   15   6.8 |
  5. |       1       5        3    5   15   9.2 |
     |------------------------------------------|
  6. |       2       1        4    3   12   3.7 |
  7. |       2       2        4    3   12   6.6 |
  8. |       2       3        4    3   12   4.2 |
     +------------------------------------------+
```

Here the variable nh is the number of PSUs per stratum that were sampled, Nh is the total number of PSUs per stratum in the sampling frame (that is, the population), and x is our survey item of interest.

If we wish to use a finite population correction in our computations, we must svyset an FPC variable when we specify the variables for sampling weights, PSUs, and strata. The FPC variable typically contains the number of sampling units per stratum in the population; Nh is our FPC variable. Here we estimate the population mean of x assuming sampling without replacement.

```
. svyset psuid [pweight=weight], strata(stratid) fpc(Nh)

Sampling weights: weight
             VCE: linearized
     Single unit: missing
        Strata 1: stratid
 Sampling unit 1: psuid
           FPC 1: Nh

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

Survey: Mean estimation

Number of strata = 2                      Number of obs   =  8
Number of PSUs   = 8                      Population size = 27
                                          Design df       =  6

--------------------------------------------------------------
             |             Linearized
             |       Mean   std. err.     [95% conf. interval]
-------------+------------------------------------------------
           x |   5.448148   .6160407      3.940751    6.955545
--------------------------------------------------------------
```

We must respecify the survey design before estimating the population mean of x assuming sampling with replacement.

```
. svyset psuid [pweight=weight], strata(stratid)

Sampling weights: weight
             VCE: linearized
     Single unit: missing
        Strata 1: stratid
 Sampling unit 1: psuid
           FPC 1: <zero>

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

Survey: Mean estimation

Number of strata = 2                      Number of obs   =  8
Number of PSUs   = 8                      Population size = 27
                                          Design df       =  6

--------------------------------------------------------------
             |             Linearized
             |       Mean   std. err.     [95% conf. interval]
-------------+------------------------------------------------
           x |   5.448148   .7412683       3.63433    7.261966
--------------------------------------------------------------
```

Including an FPC always reduces the variance estimate. However, the reduction in the variance estimates will be small when the N_h are large relative to the n_h.

Rather than having a variable that represents the total number of PSUs per stratum in the sampling frame, we sometimes have a variable that represents a sampling rate $f_h = n_h/N_h$. The syntax for `svyset` is the same whether the FPC variable contains N_h or f_h. The survey variance-estimation routines in Stata are smart enough to identify what type of FPC information has been specified. If the FPC variable is less than or equal to 1, it is interpreted as a sampling rate; if it is greater than or equal to n_h, it is interpreted as containing N_h. It is an error for the FPC variable to have values between 1 and n_h or to have a mixture of sampling rates and stratum sizes.

◁

Multiple-stage designs and with-replacement sampling

Although survey data are seldom collected using with-replacement sampling, dropping the FPC information when the sampling fractions are small is common. In either case, svyset ignores the design variables specified in later sampling stages because this information is not necessary for variance estimation. In the following, we describe why this is true.

▷ Example 6

Consider the two-stage design where PSUs are sampled with replacement and individuals are sampled without replacement within the selected PSUs. Sampling the individuals with replacement would change some of the details in the following discussion, but the result would be the same.

Our population contains 100 PSUs, with five individuals in each, so our population size is 500. We will sample 10 PSUs with replacement and then sample two individuals without replacement from within each selected PSU. This results in a dataset with 10 PSUs, each with 2 observations, for a total of 20 observations. If our dataset contained the PSU information in variable su1 and the second-stage FPC information in variable fpc2, our svyset command would be as follows.

```
. use https://www.stata-press.com/data/r19/svyset_wr

. svyset su1 || _n, fpc(fpc2)
note: stage 1 is sampled with replacement; further stages will be ignored for
      variance estimation.

Sampling weights: <none>
             VCE: linearized
     Single unit: missing
        Strata 1: <one>
 Sampling unit 1: su1
           FPC 1: <zero>
```

As expected, svyset tells us that it is ignoring the second-stage information because the first-stage units were sampled with replacement. Because we do not have an FPC variable for the first stage, we can regard the sampling of PSUs as a series of independently and identically distributed draws. The second-sampled PSU is drawn independently from the first and has the same sampling distribution because the first-sampled PSU is eligible to be sampled again.

Consider the following alternative scenario. Because there are 10 ways to pick two people of five, let's expand the 100 PSUs to form $100 \times 10 = 1{,}000$ "new PSUs" (NPSUs), each of size 2, representing all possible two-person groups that can be sampled from the original 100 groups of five people. We now have a population of $1{,}000 \times 2 = 2{,}000$ "new people"; each original person was replicated four times. We can select 10 NPSUs with replacement to end up with a dataset consisting of 10 groups of two to form samples of 20 people. If our "new" dataset contained the PSU information in variable nsu1, our svyset command would be as follows:

```
. svyset nsu1

Sampling weights: <none>
             VCE: linearized
     Single unit: missing
        Strata 1: <one>
 Sampling unit 1: nsu1
           FPC 1: <zero>
```

There is nothing from a sampling standpoint to distinguish between our two scenarios. The information contained in the variables su1 and nsu1 is equivalent; thus svyset can behave as if our dataset came from the second scenario.

The following questions may spring to mind after reading the above:

- The population in the first scenario has 500 people; the second has 2,000. Does that not invalidate the comparison between the two scenarios?

  Although the populations are different, the sampling schemes described for each scenario result in the same sampling space. By construction, each possible sample from the first scenario is also a possible sample from the second scenario. For the first scenario, the number of possible samples of 10 of 100 PSUs sampled with replacement, where two of five individuals are sampled without replacement, is

  $$100^{10} \times \binom{5}{2}^{10} = 10^{30}$$

  For the second scenario, the number of possible samples of 10 of 1,000 NPSUs sampled with replacement, where each NPSU is sampled as a whole, is

  $$1{,}000^{10} = 10^{30}$$

- Does the probability of being in the sample not depend on what happens in the first sampling stage?

  Not when the first stage is sampled with replacement. Sampling with replacement means that all PSUs have the same chance of being selected even after one of the PSUs has been selected. Thus each of the two-person groups that can possibly be sampled has the same chance of being sampled even after a specific two-person group has been selected.

- Is it valid to have replicated people in the population like the one in the second scenario?

  Yes, because each person in the population can be sampled more than once. Sampling with replacement allows us to construct the replicated people.

◁

Replication-weight variables

Many groups that collect survey data for public use have taken steps to protect the privacy of the survey participants. This may result in datasets that have replicate-weight variables instead of variables that identify the strata and sampling units from the sampling stages. These datasets require replication methods for variance estimation.

The `brrweight()`, `jkrweight()`, `bsrweight()`, and `sdrweight()` options allow `svyset` to identify the set of replication weights for use with BRR, jackknife, bootstrap, and SDR variance estimation (`svy brr`, `svy jackknife`, `svy bootstrap`, and `svy sdr`), respectively. In addition to the weight variables, `svyset` also allows you to change the default variance estimation method from linearization to BRR, jackknife, bootstrap, or SDR.

▷ Example 7

Here are two simple examples using jackknife replication weights.

1. Data containing only sampling weights and jackknife replication weights, and we set the default variance estimator to the jackknife:

```
. use https://www.stata-press.com/data/r19/stage5a_jkw
. svyset [pweight=pw], jkrweight(jkw_*) vce(jackknife)
 Sampling weights: pw
              VCE: jackknife
              MSE: off
Jackknife weights: jkw_1 .. jkw_9
      Single unit: missing
         Strata 1: <one>
  Sampling unit 1: <observations>
            FPC 1: <zero>
```

2. Data containing only sampling weights and jackknife replication weights, and we set the default variance estimator to the jackknife by using the MSE formula:

```
. svyset [pweight=pw], jkrweight(jkw_*) vce(jackknife) mse
 Sampling weights: pw
              VCE: jackknife
              MSE: on
Jackknife weights: jkw_1 .. jkw_9
      Single unit: missing
         Strata 1: <one>
  Sampling unit 1: <observations>
            FPC 1: <zero>
```

◁

▷ Example 8: Characteristics for jackknife replicate-weight variables

The `jkrweight()` option has suboptions that allow you to identify certain characteristics of the jackknife replicate-weight variables. These characteristics include the following:

- An identifier for the stratum in which the sampling weights have been adjusted because one of its PSUs was dropped. We use the `stratum()` suboption to set these values. The default is one stratum for all the replicate-weight variables.
- The FPC value. We use the `fpc()` suboption to set these values. The default value is zero.

  This characteristic is ignored when the `mse` option is supplied to `svy jackknife`.
- A jackknife multiplier used in the formula for variance estimation. The multiplier for the standard leave-one-out jackknife method is

$$\frac{n_h - 1}{n_h}$$

  where n_h is the number of PSUs sampled from stratum h. We use the `multiplier()` suboption to set these values. The default is derived from the above formula, assuming that n_h is equal to the number of replicate-weight variables for stratum h.

Because of privacy concerns, public survey datasets may not contain stratum-specific information. However, the population size and an overall jackknife multiplier will probably be provided. You must then supply this information to `svyset` for the jackknife replicate-weight variables. We will use the 1999–2000 NHANES data to illustrate how to set these characteristics.

The NHANES datasets for years 1999–2000 are available for download from the Centers for Disease Control and Prevention (CDC) website, https://www.cdc.gov. This particular release of the NHANES data contains jackknife replication weights in addition to the usual PSU and stratum information. These variables are contained in the demographic dataset. In our web browser, we saved the demographic data from the CDC website https://wwwn.cdc.gov/Nchs/Nhanes/1999-2000/DEMO.XPT. We suggest that you rename the data to `demo.xpt`.

The 1999–2000 NHANES datasets are distributed in SAS Transport format, so we use Stata's `import sasxport8` command to read the data into memory. Because of the nature of the survey design, the demographic dataset `demo.xpt` has two sampling-weight variables. `wtint2yr` contains the sampling weights appropriate for the interview data, and `wtmec2yr` contains the sampling weights appropriate for the Mobile Examination Center (MEC) exam data. Consequently, there are two sets of jackknife replicate-weight variables. The jackknife replicate-weight variables for the interview data are named `wtirep01`, `wtirep02`, ..., `wtirep52`. The jackknife replicate-weight variables for the MEC exam data are named `wtmrep01`, `wtmrep02`, ..., `wtmrep52`. The documentation published with the NHANES data gives guidance on which weight variables to use.

```
. import sasxport5 demo.xpt

. describe wtint2yr wtmec2yr wtirep01 wtmrep01

Variable      Storage   Display    Value
    name         type    format    label      Variable label
--------------------------------------------------------------------------------

wtint2yr        double  %10.0g                Full Sample 2 Year Interview
                                                Weight
wtmec2yr        double  %10.0g                Full Sample 2 Year MEC Exam Weight
wtirep01        double  %10.0g                Interview Weight Jack Knife
                                                Replicate 01
wtmrep01        double  %10.0g                MEC Exam Weight Jack Knife
                                                Replicate 01
```

The number of PSUs in the NHANES population is not apparent, so we will not set an FPC value, but we can set the standard jackknife multiplier for the 52 replicate-weight variables and save the results as a Stata dataset for future use. Also the NHANES datasets all contain a variable called `seqn`. This variable has a respondent sequence number that allows the dataset users to merge the demographic dataset with other 1999–2000 NHANES datasets, so we sort on `seqn` before saving `demo99_00.dta`.

```
. local mult = 51/52

. svyset, jkrweight(wtmrep*, multiplier(`mult'))
  (output omitted)

. svyset, jkrweight(wtirep*, multiplier(`mult'))
  (output omitted)

. svyset, clear

. sort seqn

. save demo99_00
file demo99_00.dta saved
```

To complete this example, we will perform a simple analysis using the blood pressure data; however, before we can perform any analysis, we have to merge the blood pressure dataset, `bpx.xpt`, with our demographic dataset, `demo99_00.dta`. In our web browser, we saved the blood pressure data from the CDC website https://wwwn.cdc.gov/Nchs/Nhanes/1999-2000/BPX.XPT. We suggest that you rename the data to `bpx.xpt`.

We can then use import sasxport8 to read in the blood pressure data, sort on seqn, and save the resulting dataset to bpx99_00.dta. We read in our copy of the demographic data, drop the irrelevant weight variables, and merge in the blood pressure data from bpx99_00.dta. A quick call to tabulate on the _merge variable generated by merge indicates that 683 observations in the demographic data are not present in the blood pressure data. We do not drop these observations; otherwise, the estimate of the population size will be incorrect. Finally, we set the appropriate sampling and replicate-weight variables with svyset before replacing bpx99_00.dta with a more complete copy of the blood pressure data.

```
. import sasxport5 bpx.xpt, clear
. sort seqn
. save bpx99_00
file bpx99_00.dta saved
. use demo99_00
. drop wtint?yr wtirep*
. merge 1:1 seqn using bpx99_00
    Result                      Number of obs
    -----------------------------------------
    Not matched                           683
        from master                       683  (_merge==1)
        from using                          0  (_merge==2)

    Matched                             9,282  (_merge==3)
    -----------------------------------------
. drop _merge
. svyset [pw=wtmec2yr], jkrweight(wtmrep*) vce(jackknife)
  (output omitted)
. save bpx99_00, replace
file bpx99_00.dta saved
```

Having saved our merged dataset (with svysettings), we estimate the mean systolic blood pressure for the population, using the MEC exam replication weights for jackknife variance estimation.

```
. svy: mean bpxsar
(running mean on estimation sample)
Jackknife replications (52): .........10.........20.........30.........40.......
> ..50.. done
Survey: Mean estimation
Number of strata = 1             Number of obs   =       7,898
                                 Population size = 231,756,417
                                 Replications    =          52
                                 Design df       =          51

--------------------------------------------------------------
             |             Jackknife
             |       Mean   std. err.     [95% conf. interval]
-------------+------------------------------------------------
      bpxsar |   119.7056   .5109122      118.6799    120.7313
--------------------------------------------------------------
```

◁

Combining datasets from multiple surveys

The 2001–2002 NHANES datasets are also available from the CDC website, https://www.cdc.gov. The guidelines that are published with these datasets recommend that the 1999–2000 and 2001–2002 NHANES datasets be combined to increase the accuracy of results. Combining datasets from multiple surveys is a complicated process, and Stata has no specific tools for this task. However, the distributors of the

NHANES datasets provide sampling-weight variables for the 1999–2002 combined data in the respective demographic datasets. They also provide some simple instructions on how to combine the datasets from these two surveys.

In the previous example, we worked with the 1999–2000 NHANES data. The 2001–2002 NHANES demographics data are contained in `demo_b.xpt`, and the blood pressure data are contained in `bpx_b.xpt`. We follow the same steps as in the previous example to merge the blood pressure data with the demographic data for 2001–2002.

Visit the following CDC websites and save the data:

https://wwwn.cdc.gov/Nchs/Nhanes/2001-2002/BPX_B.XPT

https://wwwn.cdc.gov/Nchs/Nhanes/2001-2002/DEMO_B.XPT

We suggest that you rename the data to `bpx_b.xpt` and `demo_b.xpt`. We can then continue with our example:

```
. import sasxport5 bpx_b.xpt, clear
. sort seqn
. save bpx01_02
file bpx01_02.dta saved
. import sasxport5 demo_b.xpt, clear
. drop wtint?yr
. sort seqn
. merge 1:1 seqn using bpx01_02
    Result                      Number of obs
    -----------------------------------------
    Not matched                           562
        from master                       562  (_merge==1)
        from using                          0  (_merge==2)

    Matched                            10,477  (_merge==3)
    -----------------------------------------

. drop _merge
. svyset sdmvpsu [pw=wtmec2yr], strata(sdmvstra)
Sampling weights: wtmec2yr
             VCE: linearized
     Single unit: missing
        Strata 1: sdmvstra
 Sampling unit 1: sdmvpsu
           FPC 1: <zero>
. save bpx01_02, replace
file bpx01_02.dta saved
```

The demographic dataset for 2001–2002 does not contain replicate-weight variables, but there are variables that provide information on PSUs and strata for variance estimation. The PSU information is contained in `sdmvpsu`, and the stratum information is in `sdmvstra`. See the documentation that comes with the NHANES datasets for the details regarding these variables.

This new blood pressure dataset (`bpx01_02.dta`) is all we need if we are interested in analyzing blood pressure data only for 2001–2002. However, we want to use the 1999–2002 combined data, so we will follow the advice in the guidelines and just combine the datasets from the two surveys.

For those concerned about overlapping stratum identifiers between the two survey datasets, it is a simple exercise to check that `sdmvstra` ranges from 1 to 13 for 1999–2000 but ranges from 14 to 28 for 2001–2002. Thus the stratum identifiers do not overlap, so we can simply append the data.

The 2001–2002 NHANES demographic dataset has no jackknife replicate-weight variables, so we drop the replicate-weight variables from the 1999–2000 dataset. The sampling-weight variable `wtmec2yr` is no longer appropriate for use with the combined data because its values are based on the survey designs individually, so we drop it from the combined dataset. Finally, we use `svyset` to identify the design variables for the combined surveys. `wtmec4yr` is the sampling-weight variable for the MEC exam data developed by the data producers for the combined 1999–2002 NHANES data.

```
. use bpx99_00

. drop wt?rep*

. append using bpx01_02

. drop wtmec2yr

. svyset sdmvpsu [pw=wtmec4yr], strata(sdmvstra)

Sampling weights: wtmec4yr
             VCE: linearized
     Single unit: missing
        Strata 1: sdmvstra
 Sampling unit 1: sdmvpsu
           FPC 1: <zero>

. save bpx99_02
file bpx99_02.dta saved
```

Now we can estimate the mean systolic blood pressure for our population by using the combined surveys and jackknife variance estimation.

```
. svy jackknife: mean bpxsar
(running mean on estimation sample)

Jackknife replications (57): .........10.........20.........30.........40.......
> ..50....... done

Survey: Mean estimation

Number of strata = 28          Number of obs   =      16,297
Number of PSUs   = 57          Population size = 237,466,080
                               Replications    =          57
                               Design df       =          29

--------------------------------------------------------------
             |             Jackknife
             |       Mean   std. err.     [95% conf. interval]
-------------+------------------------------------------------
      bpxsar |   119.8914   .3828434      119.1084    120.6744
--------------------------------------------------------------
```

Video example

Specifying the design of your survey data to Stata

Stored results

`svyset` stores the following in `r()`:

Scalars	
`r(stages)`	number of sampling stages
`r(stages_wt)`	last stage containing stage-level weights
`r(bsn)`	bootstrap mean-weight adjustment
`r(fay)`	Fay's adjustment
`r(dof)`	`dof()` value
Macros	
`r(wtype)`	weight type
`r(wexp)`	weight expression
`r(wvar)`	weight variable name
`r(weight#)`	variable identifying weight for stage #
`r(su#)`	variable identifying sampling units for stage #
`r(strata#)`	variable identifying strata for stage #
`r(fpc#)`	FPC for stage #
`r(bsrweight)`	`bsrweight()` variable list
`r(brrweight)`	`brrweight()` variable list
`r(jkrweight)`	`jkrweight()` variable list
`r(sdrweight)`	`sdrweight()` variable list
`r(sdrfpc)`	`fpc()` value from within `sdrweight()`
`r(vce)`	*vcetype* specified in `vce()`
`r(mse)`	`mse`, if specified
`r(poststrata)`	`poststrata()` variable
`r(postweight)`	`postweight()` variable
`r(rake)`	`rake()` specification
`r(regress)`	`regress()` specification
`r(settings)`	`svyset` arguments to reproduce the current settings
`r(singleunit)`	`singleunit()` setting

References

Cochran, W. G. 1977. *Sampling Techniques*. 3rd ed. New York: Wiley.

Judkins, D. R. 1990. Fay's method for variance estimation. *Journal of Official Statistics* 6: 223–239.

O'Donnell, O., E. van Doorslaer, A. Wagstaff, and M. Lindelow. 2008. *Analyzing Health Equity Using Household Survey Data: A Guide to Techniques and Their Implementation*. Washington, DC: World Bank.

Also see

[SVY] **Survey** — Introduction to survey commands

[SVY] **svy** — The survey prefix command

[SVY] **svydescribe** — Describe survey data

[SVY] **Calibration** — Calibration for survey data

[SVY] **Poststratification** — Poststratification for survey data

[SVY] **Subpopulation estimation** — Subpopulation estimation for survey data

[SVY] **Variance estimation** — Variance estimation for survey data