svy — The survey prefix command

Description

svy fits statistical models for complex survey data by adjusting the results of a command for survey settings identified by svyset. Any Stata estimation command listed in [SVY] svy estimation may be used with svy. User-written programs that meet the requirements in [P] program properties may also be used.

Quick start

Data for a two-stage design with sampling weight wvar1, strata defined by levels of svar, sampling units are identified by su1, and second-stage clustering is defined by su2

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

Adjust linear regression for complex survey design settings specified in svyset

svy: regress ...

As above, but restrict estimation to the subpopulation where group equals 4

svy, subpop(if group==4): regress ...

Same as above, but use new binary variable insample to indicate the subpopulation

generate insample = (group==4)

svy, subpop(insample): regress ...

Specify that the design degrees of freedom is 135 instead of the difference between the number of unique values of su1 and the number of levels of svar

svy, dof(135): regress ...

Note: Any estimation command meeting the requirements specified in the Description may be substituted for regress in the examples above.
Syntax

```
svy [vcetype] [ , svy_options eform_option ] : command
```

svctype | Description
---|---
SE
linearized | Taylor-linearized variance estimation
bootstrap | bootstrap variance estimation; see [SVY] svy bootstrap
brr | BRR variance estimation; see [SVY] svy brr
jackknife | jackknife variance estimation; see [SVY] svy jackknife
sdr | SDR variance estimation; see [SVY] svy sdr

Specifying a vcetype overrides the default from svyset.

types

svy_options | Description
---|---
if/in
subpop([varname] [if]) | identify a subpopulation
SE
dof(#) | design degrees of freedom
bootstrap_options | more options allowed with bootstrap variance estimation;
| see [SVY] bootstrap_options
brr_options | more options allowed with BRR variance estimation;
| see [SVY] brr_options
jackknife_options | more options allowed with jackknife variance estimation;
| see [SVY] jackknife_options
sdr_options | more options allowed with SDR variance estimation;
| see [SVY] sdr_options
Report
level(#) | set confidence level; default is level(95)
noconsreport | do not display constraints
display_options | control columns and column formats, row spacing, line width,
| display of omitted variables and base and empty cells, and
| factor-variable labeling
noheader | suppress table header
nolegend | suppress table legend
noadjust | do not adjust model Wald statistic
noisily | display any output from command
trace | trace command
coeflegend | display legend instead of statistics
svy requires that the survey design variables be identified using `svyset`; see [SVY] svyset.

`command` defines the estimation command to be executed. The `by` prefix cannot be part of `command`.

`mi estimate` may be used with `svy linearized` if the estimation command allows `mi estimate`; it may not be used with `svy bootstrap`, `svy brr`, `svy jackknife`, or `svy sdr`.

`noheader`, `nolegend`, `noadjust`, `noisily`, `trace`, and `coeflegend` are not shown in the dialog boxes for estimation commands.

Warning: Using `if` or `in` restrictions will often not produce correct variance estimates for subpopulations. To compute estimates for subpopulations, use the `subpop()` option.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

### Options

`if/in` specifies that estimates be computed for the single subpopulation identified by `subpop`, which is

\[
\text{[varname] [if]}
\]

Thus the subpopulation is defined by the observations for which `varname ≠ 0` that also meet the `if` conditions. Typically, `varname = 1` defines the subpopulation, and `varname = 0` indicates observations not belonging to the subpopulation. For observations whose subpopulation status is uncertain, `varname` should be set to a missing value; such observations are dropped from the estimation sample.

See [SVY] Subpopulation estimation and [SVY] estat.

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

`bootstrap_options` are other options that are allowed with bootstrap variance estimation specified by `svy bootstrap` or specified as `svyset` using the `vce(bootstrap)` option; see [SVY] bootstrap_options.

`brr_options` are other options that are allowed with BRR variance estimation specified by `svy brr` or specified as `svyset` using the `vce(brr)` option; see [SVY] brr_options.

`jackknife_options` are other options that are allowed with jackknife variance estimation specified by `svy jackknife` or specified as `svyset` using the `vce(jackknife)` option; see [SVY] jackknife_options.

`sdr_options` are other options that are allowed with SDR variance estimation specified by `svy sdr` or specified as `svyset` using the `vce(sdr)` option; see [SVY] sdr_options.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] 20.8 Specifying the width of confidence intervals.

`nocnsreport`; see [R] Estimation options.

`display_options`: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#), fvwrapon(style), cformat(%fmt), pformat(%fmt), sformat(%fmt),` and `nolstretch`; see [R] Estimation options.

The following options are available with `svy` but are not shown in the dialog boxes:

`noheader` prevents the table header from being displayed. This option implies `nolegend`.

---

**svy — The survey prefix command**

---

**svyset** requires that the survey design variables be identified using `svyset`; see [SVY] svyset.

`command` defines the estimation command to be executed. The `by` prefix cannot be part of `command`.

`mi estimate` may be used with `svy linearized` if the estimation command allows `mi estimate`; it may not be used with `svy bootstrap`, `svy brr`, `svy jackknife`, or `svy sdr`.

`noheader`, `nolegend`, `noadjust`, `noisily`, `trace`, and `coeflegend` are not shown in the dialog boxes for estimation commands.

Warning: Using `if` or `in` restrictions will often not produce correct variance estimates for subpopulations. To compute estimates for subpopulations, use the `subpop()` option.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

### Options

`if/in` specifies that estimates be computed for the single subpopulation identified by `subpop`, which is

\[
\text{[varname] [if]}
\]

Thus the subpopulation is defined by the observations for which `varname ≠ 0` that also meet the `if` conditions. Typically, `varname = 1` defines the subpopulation, and `varname = 0` indicates observations not belonging to the subpopulation. For observations whose subpopulation status is uncertain, `varname` should be set to a missing value; such observations are dropped from the estimation sample.

See [SVY] Subpopulation estimation and [SVY] estat.

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

`bootstrap_options` are other options that are allowed with bootstrap variance estimation specified by `svy bootstrap` or specified as `svyset` using the `vce(bootstrap)` option; see [SVY] bootstrap_options.

`brr_options` are other options that are allowed with BRR variance estimation specified by `svy brr` or specified as `svyset` using the `vce(brr)` option; see [SVY] brr_options.

`jackknife_options` are other options that are allowed with jackknife variance estimation specified by `svy jackknife` or specified as `svyset` using the `vce(jackknife)` option; see [SVY] jackknife_options.

`sdr_options` are other options that are allowed with SDR variance estimation specified by `svy sdr` or specified as `svyset` using the `vce(sdr)` option; see [SVY] sdr_options.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] 20.8 Specifying the width of confidence intervals.

`nocnsreport`; see [R] Estimation options.

`display_options`: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#), fvwrapon(style), cformat(%fmt), pformat(%fmt), sformat(%fmt),` and `nolstretch`; see [R] Estimation options.

The following options are available with `svy` but are not shown in the dialog boxes:

`noheader` prevents the table header from being displayed. This option implies `nolegend`. 
nolegend prevents the table legend identifying the subpopulations from being displayed.

noadjust specifies that the model Wald test be carried out as \( W/k \sim F(k, d) \), where \( W \) is the Wald test statistic, \( k \) is the number of terms in the model excluding the constant term, \( d \) is the total number of sampled PSUs minus the total number of strata, and \( F(k, d) \) is an \( F \) distribution with \( k \) numerator degrees of freedom and \( d \) denominator degrees of freedom. By default, an adjusted Wald test is conducted: \((d - k + 1)W/(kd) \sim F(k, d - k + 1)\).

See Korn and Graubard (1990) for a discussion of the Wald test and the adjustments thereof. Using the noadjust option is not recommended.

noisily requests that any output from command be displayed.

trace causes a trace of the execution of command to be displayed.

coeflegend; see [R] Estimation options.

The following option is usually available with svy at the time of estimation or on replay but is not shown in all dialog boxes:

eform_option; see [R] eform_option.

Remarks and examples

The svy prefix is designed for use with complex survey data. Typical survey design characteristics include sampling weights, one or more stages of clustered sampling, and stratification. For a general discussion of various aspects of survey designs, including multistage designs, see [SVY] svyset.

Below we present an example of the effects of weights, clustering, and stratification. This is a typical case, but drawing general rules from any one example is still dangerous. You could find particular analyses from other surveys that are counterexamples for each of the trends for standard errors exhibited here.

Example 1: The effects of weights, clustering, and stratification

We use data from the Second National Health and Nutrition Examination Survey (NHANES II) (McDowell et al. 1981) as our example. This is a national survey, and the dataset has sampling weights, strata, and clustering. In this example, we will consider the estimation of the mean serum zinc level of all adults in the United States.

First, consider a proper design-based analysis, which accounts for weighting, clustering, and stratification. Before we issue our svy estimation command, we set the weight, strata, and PSU identifier variables:

```
. use https://www.stata-press.com/data/r16/nhanes2f
. svyset psuid [pweight=finalwgt], strata(stratid)
```

pweight: finalwgt
VCE: linearized
Single unit: missing
Strata 1: stratid
SU 1: psuid
FPC 1: <zero>
We now estimate the mean by using the proper design-based analysis:

```
.svy: mean zinc
```

```
Survey: Mean estimation
Number of strata =    31    Number of obs =    9,189
Number of PSUs =      62    Population size = 104,176,071
  Design df =            31

<table>
<thead>
<tr>
<th></th>
<th>Linearized</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std. Err.</td>
<td>[95% Conf. Interval]</td>
</tr>
<tr>
<td>zinc</td>
<td>87.18207</td>
<td>.4944827</td>
<td>86.17356 - 88.19057</td>
</tr>
</tbody>
</table>
```

If we ignore the survey design and use `mean` to estimate the mean, we get

```
.mean zinc
```

```
Mean estimation
Number of obs =    9,189

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std. Err.</td>
</tr>
<tr>
<td>zinc</td>
<td>86.51518</td>
<td>.1510744</td>
</tr>
</tbody>
</table>
```

The point estimate from the unweighted analysis is smaller by more than one standard error than the proper design-based estimate. Also, design-based analysis produced a standard error that is 3.27 times larger than the standard error produced by our incorrect analysis.

Example 2: Halfway is not enough—the importance of stratification and clustering

When some people analyze survey data, they say, “I know I have to use my survey weights, but I will just ignore the stratification and clustering information.” If we follow this strategy, we will obtain the proper design-based point estimates, but our standard errors, confidence intervals, and test statistics will usually be wrong.

To illustrate this effect, suppose that we used the `svy: mean` procedure with `pweights` only.

```
.svyset [pweight=finalwgt]

.svy: mean zinc
```

```
Survey: Mean estimation
Number of strata =         1    Number of obs =         9,189
Number of PSUs =          9,189    Population size = 104,176,071
  Design df =            9,188

<table>
<thead>
<tr>
<th></th>
<th>Linearized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>zinc</td>
<td>87.18207</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

\[\]
This approach gives us the same point estimate as our design-based analysis, but the reported standard error is less than one-half the design-based standard error. If we accounted only for clustering and weights and ignored stratification in NHANES II, we would obtain the following analysis:

```
.svysset psuid [pweight=finalwgt]
pweight: finalwgt
  VCE: linearized
Single unit: missing
  Strata 1: <one>
    SU 1: psuid
    FPC 1: <zero>
```

```
.svy: mean zinc
(running mean on estimation sample)
```

Survey: Mean estimation

<table>
<thead>
<tr>
<th></th>
<th>Linearized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td>zinc</td>
<td>87.18207</td>
</tr>
</tbody>
</table>

Here our standard error is about 50% larger than what we obtained in our proper design-based analysis.

### Example 3

Let’s look at a regression. We model zinc on the basis of age, weight, sex, race, and rural or urban residence. We compare a proper design-based analysis with an ordinary regression (which assumes independent and identically distributed error).
Here is our design-based analysis:

```
svyset psuid [pweight=finalwgt], strata(stratid)
pweight: finalwgt
VCE: linearized
```

```bash
.run
Single unit: missing
Strata 1: stratid
SU 1: psuid
FPC 1: <zero>
```

```
svy: regress zinc age c.age#c.age weight female black orace rural
```

Survey: Linear regression

<table>
<thead>
<tr>
<th>Number of strata</th>
<th>Number of obs</th>
<th>Number of PSUs</th>
<th>Design df</th>
<th>F( 7, 25)</th>
<th>Prob &gt; F</th>
<th>R-squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>9,189</td>
<td>62</td>
<td>31</td>
<td>62.50</td>
<td>0.0000</td>
<td>0.0698</td>
</tr>
</tbody>
</table>

If we had improperly ignored our survey weights, stratification, and clustering (that is, if we had used the usual Stata `regress` command), we would have obtained the following results:

```
.regress zinc age c.age#c.age weight female black orace rural
```

```
Source | SS    | df  | MS     | Number of obs | F(7, 9181) | Prob > F | Adj R-squared | Root MSE |
-------|-------|-----|--------|---------------|------------|----------|---------------|---------|
Model   | 110417.827 | 7 | 15773.9753 | 9,189 | 79.72 |
Residual | 1816535.3 | 9,181 | 197.85811 | | 0.0573 |
Total   | 1926953.13 | 9,188 | 209.724982 | | 14.066 |
```

```
zinc | Coef. | Std. Err. | t     | P>|t| | [95% Conf. Interval] |
age  | -0.1701161 | 0.0844192 | -2.02 | 0.053 | -0.3422901 , 0.002058 |
c.age#c.age | 0.0008744 | 0.0008655 | 1.01 | 0.320 | -0.0008907 , 0.0026396 |
weight | 0.0536225 | 0.139115 | 3.85 | 0.001 | 0.251499 , 0.0818951 |
female | -0.134161 | 0.4403625 | -13.93 | 0.000 | -0.7032286 , -0.5236035 |
black | -2.881813 | 1.075958 | -2.68 | 0.012 | -5.076244 , -.687381 |
orace | -4.118051 | 1.621121 | -2.54 | 0.016 | -7.424349 , -.8117528 |
rural | -.5386327 | .6171836 | -0.87 | 0.390 | -1.797387 , .7201216 |
_cons | 92.47495 | 2.228263 | 41.50 | 0.000 | 87.93038 , 97.01952 |
```

```
zinc | Coef. | Std. Err. | t     | P>|t| | [95% Conf. Interval] |
age  | -0.090298 | 0.0638452 | -1.41 | 0.157 | -0.2154488 , 0.0348528 |
c.age#c.age | -0.000324 | 0.0006788 | -0.05 | 0.962 | -0.0013631 , 0.0012983 |
weight | 0.0606481 | 0.0105986 | 5.72 | 0.000 | 0.0398725 , 0.0814237 |
female | -0.021949 | 0.3194705 | -15.72 | 0.000 | -0.5648182 , -0.4395716 |
black | -2.311753 | 0.5073536 | -4.56 | 0.000 | -3.306279 , -1.317227 |
orace | -3.390879 | 1.060981 | -3.20 | 0.001 | -5.470637 , -1.311121 |
rural | -.0996462 | 0.3098948 | -0.31 | 0.755 | -.7041089 , .5108166 |
_cons | 89.49465 | 1.477528 | 60.57 | 0.000 | 86.59836 , 92.39093 |
The point estimates differ by 3%–100%, and the standard errors for the proper designed-based analysis are 30%–110% larger. The differences are not as dramatic as we saw with the estimation of the mean, but they are still substantial.

### Stored results

`svy` stores the following in `e()`:

**Scalars**

- `e(N)` number of observations
- `e(N_sub)` subpopulation observations
- `e(N_strata)` number of strata
- `e(N_strata_omit)` number of strata omitted
- `e(singleton)` 1 if singleton strata, 0 otherwise
- `e(census)` 1 if census data, 0 otherwise
- `e(F)` model $F$ statistic
- `e(df_m)` model degrees of freedom
- `e(df_r)` variance degrees of freedom
- `e(N_pop)` estimate of population size
- `e(N_subpop)` estimate of subpopulation size
- `e(N_psu)` number of sampled PSUs
- `e(stages)` number of sampling stages
- `e(k_eq)` number of equations in `e(b)`
- `e(k_aux)` number of ancillary parameters
- `e(p)` $p$-value
- `e(rank)` rank of `e(V)`

**Macros**

- `e(prefix)` svy
- `e(cmdname)` command name from `command`
- `e(cmd)` same as `e(cmdname)` or `e(vce)`
- `e(command)` command
- `e(cmdline)` command as typed
- `e(wtype)` weight type
- `e(wexp)` weight expression
- `e(weight#)` variable identifying weight for stage #
- `e(wvar)` weight variable name
- `e(singleunit)` `singleunit()` setting
- `e(strata)` `strata()` variable
- `e(strata#)` variable identifying strata for stage #
- `e(psu)` `psu()` variable
- `e(su#)` variable identifying sampling units for stage #
- `e(fpc)` `fpc()` variable
- `e(fpc#)` FPC for stage #
- `e(title)` title in estimation output
- `e(poststrata)` `poststrata()` variable
- `e(postweight)` `postweight()` variable
- `e(vce)` `vcetype` specified in `vce()`
- `e(vcetype)` title used to label Std. Err.
- `e(mse)` `mse`, if specified
- `e(subpop)` `subpop` from `subpop()`
- `e(adjust)` `noadjust`, if specified
- `e(properties)` `b V`
- `e(estat_cmd)` program used to implement `estat`
- `e(predict)` program used to implement `predict`
- `e(marginsnotok)` predictions disallowed by `margins`
- `e(marginswtype)` weight type for `margins`

**Matrices**

- `e(b)` estimates
- `e(V)` design-based variance
- `e(V_srs)` simple-random-sampling-without-replacement variance, $\hat{V}_{srswor}$
svy — The survey prefix command

\[ e(V_{\text{srssub}}) \]
subpopulation simple-random-sampling-without-replacement variance, \( \hat{V}_{\text{srswor}} \)
(created only when subpop() is specified)

\[ e(V_{\text{srswr}}) \]
simple-random-sampling-with-replacement variance, \( \hat{V}_{\text{srswr}} \)
(created only when fpc() option is svyset)

\[ e(V_{\text{srssubwr}}) \]
subpopulation simple-random-sampling-with-replacement variance, \( \hat{V}_{\text{srswr}} \)
(created only when subpop() is specified)

\[ e(V_{\text{modelbased}}) \]
model-based variance

\[ e(V_{\text{msp}}) \]
variance from misspecified model fit, \( \hat{V}_{\text{msp}} \)

\[ e(N_{\text{strata single}}) \]
number of strata with one sampling unit

\[ e(N_{\text{strata certain}}) \]
number of certainty strata

\[ e(N_{\text{strata}}) \]
number of strata

\[ e(N_{\text{subp}}) \]
estimate of subpopulation sizes within over() groups

Functions

\[ e(\text{sample}) \]
marks estimation sample

svy also carries forward most of the results already in \texttt{e()} from \texttt{command}.

In addition to the above, the following is stored in \texttt{r()}:

Matrices

\[ r(\text{table}) \]
matrix containing the coefficients with their standard errors, test statistics, \( p \)-values, and confidence intervals

Note that results stored in \texttt{r()} are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

See [SVY] Variance estimation for all the details behind the point estimate and variance calculations made by svy.

References


Also see

[SVY] svy estimation — Estimation commands for survey data
[SVY] svy postestimation — Postestimation tools for svy
[SVY] svy bootstrap — Bootstrap for survey data
[SVY] svy brr — Balanced repeated replication for survey data
[SVY] svy jackknife — Jackknife estimation for survey data
[SVY] svy sdr — Successive difference replication for survey data
[SVY] svyset — Declare survey design for dataset
[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
[P] program properties — Properties of user-defined programs
[P] _robust — Robust variance estimates
[U] 20 Estimation and postestimation commands