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 ...

Same 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
```

<table>
<thead>
<tr>
<th>vcetype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>Taylor-linearized variance estimation</td>
</tr>
<tr>
<td></td>
<td>bootstrap variance estimation; see [SVY] svy bootstrap</td>
</tr>
<tr>
<td></td>
<td>BRR variance estimation; see [SVY] svy brr</td>
</tr>
<tr>
<td></td>
<td>jackknife variance estimation; see [SVY] svy jackknife</td>
</tr>
<tr>
<td></td>
<td>SDR variance estimation; see [SVY] svy sdr</td>
</tr>
</tbody>
</table>

Specifying a vcetype overrides the default from svyset.

```
svy_options Description
```

<table>
<thead>
<tr>
<th>if/in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subpop([varname] [if])</td>
<td>identify a subpopulation</td>
</tr>
<tr>
<td>SE</td>
<td>design degrees of freedom</td>
</tr>
<tr>
<td>bootstrap_options</td>
<td>more options allowed with bootstrap variance estimation; see [SVY] bootstrap_options</td>
</tr>
<tr>
<td>brr_options</td>
<td>more options allowed with BRR variance estimation; see [SVY] brr_options</td>
</tr>
<tr>
<td>jackknife_options</td>
<td>more options allowed with jackknife variance estimation; see [SVY] jackknife_options</td>
</tr>
<tr>
<td>sdr_options</td>
<td>more options allowed with SDR variance estimation; see [SVY] sdr_options</td>
</tr>
</tbody>
</table>

Reporting

```
level(#) set confidence level; default is level(95)
nocnsr report 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. svy command defines the estimation command to be executed. The by prefix cannot be part of command.

collect is allowed; see [U] 11.1.10 Prefix commands. 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

subpop(subpop) specifies that estimates be computed for the single subpopulation identified by subpop, which is

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

Thus the subpopulation is defined by the observations for which varname \( \neq 0 \) 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, \( df = N_{\text{psu}} - N_{\text{strata}} \).

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.

cnsreport; see [R] Estimation options.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nolabel, 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/r18/nhanes2f
svyset psuid [pweight=finalwgt], strata(stratid)
```

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

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

Survey: Mean estimation

| Number of strata = 31 | Number of obs = 9,189 |
| Number of PSUs = 62 | Population size = 104,176,071 |
| Design df = 31 | |

| Linearized | Mean std. err. [95% conf. interval] |
| zinc | 87.18207 .4944827 86.17356 88.19057 |

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

```
.mean zinc
```

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

| Mean Std. err. [95% conf. interval] |
| zinc | 86.51518 .1510744 86.21904 86.81132 |

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]
```

```
Sampling weights: finalwgt
VCE: linearized
Single unit: missing
Strata 1: <one>
Sampling unit 1: <observations>
FPC 1: <zero>
```

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

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 | |

| Linearized | Mean std. err. [95% conf. interval] |
| zinc | 87.18207 .1828747 86.82359 87.54054 |
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:

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

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

Survey: Mean estimation

<table>
<thead>
<tr>
<th>Linearized</th>
<th></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>.7426221</td>
<td>77.74616 96.61798</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)
Sampling weights: finalwgt
VCE: linearized
Single unit: missing
Strata 1: stratid
Sampling unit 1: psuid
FPC 1: <zero>
```

```
.svy: regress zinc age c.age#c.age weight female black orace rural
(running regress on estimation sample)
```

Survey: Linear regression

|                | Linearized |                 | P>|t| | [95% conf. interval] |
|----------------|------------|-----------------|------|----------------------|
|                | Coefficient| std. err.       | t    |                      |
| zinc           | age        | -.1701161       | .0844192 | -2.02   | 0.053 | -.3422901 | .002058 |
|                | c.age#c.age| .0008744        | .0008655 | 1.01    | 0.320 | -.0008907 | .0026396 |
|                | weight     | .0535225        | .0139115 | 3.85    | 0.001 | .0251499  | .0818951 |
|                | female     | -6.134161       | .4403625 | -13.93  | 0.000 | -7.032286 | -5.236035 |
|                | 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 |

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
```

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 9,189</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>110417.827</td>
<td>7</td>
<td>15773.9753</td>
<td>Prob &gt; F = 0.0000</td>
</tr>
<tr>
<td>Residual</td>
<td>1816535.3</td>
<td>9,181</td>
<td>197.85811</td>
<td>R-squared = 0.0573</td>
</tr>
<tr>
<td>Total</td>
<td>1926953.13</td>
<td>9,188</td>
<td>209.724982</td>
<td>Root MSE = 14.066</td>
</tr>
</tbody>
</table>

```
zinc | Coefficient| Std. err. | t | P>|t| | [95% conf. interval] |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>-.090298</td>
<td>.0638452</td>
<td>-1.41</td>
<td>0.157</td>
<td>-.2154488</td>
</tr>
<tr>
<td>c.age#c.age</td>
<td>-.0000324</td>
<td>.0006788</td>
<td>-0.05</td>
<td>0.962</td>
<td>-.0013631</td>
</tr>
<tr>
<td>weight</td>
<td>.0606481</td>
<td>.0105986</td>
<td>5.72</td>
<td>0.000</td>
<td>.0398725</td>
</tr>
<tr>
<td>female</td>
<td>-5.021949</td>
<td>.3194705</td>
<td>-15.72</td>
<td>0.000</td>
<td>-5.648182</td>
</tr>
<tr>
<td>black</td>
<td>-2.311753</td>
<td>.5073536</td>
<td>-4.56</td>
<td>0.000</td>
<td>-3.306279</td>
</tr>
<tr>
<td>orace</td>
<td>-3.390879</td>
<td>1.060981</td>
<td>-3.20</td>
<td>0.001</td>
<td>-5.470637</td>
</tr>
<tr>
<td>rural</td>
<td>-.0966462</td>
<td>.3098948</td>
<td>-0.31</td>
<td>0.755</td>
<td>-.7041089</td>
</tr>
<tr>
<td>_cons</td>
<td>89.49465</td>
<td>1.477528</td>
<td>60.57</td>
<td>0.000</td>
<td>86.59836</td>
</tr>
</tbody>
</table>
```
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

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

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

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

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

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

\[ e(\_N\_strata\_single) \text{ number of strata with one sampling unit} \]

\[ e(\_N\_strata\_certain) \text{ number of certainty strata} \]

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

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

Functions

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

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

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

Matrices

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

Note that results stored in \( r() \) 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