# 20 Estimation and postestimation commands

**Contents**

## 20.1 All estimation commands work the same way

All Stata commands that fit statistical models—commands such as `regress`, `logit`, and `sureg`—work similarly. Most single-equation estimation commands have the syntax

*command* *varlist* [*if*] [*in*] [*weight*] [, *options*]

and most multiple-equation estimation commands have the syntax

*command* (*varlist*) (*varlist*) ... (*varlist*) [*if*] [*in*] [*weight*] [, *options*]

Adopt a loose definition of single and multiple equation in interpreting this. For instance, `heckman` is a two-equation system, mathematically speaking, yet we categorize it, syntactically, with single-equation commands because most researchers think of it as a linear regression with an adjustment for the censoring. The important thing is that most estimation commands have one or the other of these two syntaxes.

In single-equation commands, the first variable in the *varlist* is the dependent variable, and the remaining variables are the independent variables, with some exceptions. For instance, `mixed` allows special variable prefixes to identify random factors.

Prefix commands may be specified in front of an estimation command to modify or extend what it does. The syntax is

*prefix*`:` *command* ...

See [U] **11.1.10 Prefix commands** for the full list of prefix commands. To find out which prefix commands are available for an estimation command, see the command's syntax section.

Also, all estimation commands—whether single or multiple equation—share the following features:

1. You can use the standard features of Stata's syntax—`if` *exp* and `in` *range*—to specify the estimation subsample; you do not have to make a special dataset.
2. You can retype the estimation command without arguments to redisplay the most recent estimation results. For instance, after fitting a model with `regress`, you can see the estimates again by typing `regress` by itself. You do not have to do this immediately—any number of commands can occur between the estimation and the replaying, and, in fact, you can even replay the last estimates after the data have changed or you have dropped the data altogether. Stata never forgets (unless you type `discard`; see [P] **discard**).
3. You can specify the `level()` option at the time of estimation, or when you redisplay results if that makes sense, to specify the width of the confidence intervals for the coefficients. The default is `level(95)`, meaning 95% confidence intervals. You can reset the default with `set level`; see [R] **level**.
4. You can use the postestimation command `margins` to display model results in terms of marginal effects ($dy/dx$ or even $df(y)/dx$), which can be displayed as either derivatives or elasticities; see [R] **margins**.
5. You can use the postestimation command `margins` to obtain tables of estimated marginal means, adjusted predictions, and predictive margins; see [U] **20.17 Obtaining conditional and average marginal effects** and [R] **margins**.
6. You can use the postestimation command `pwcompare` to obtain pairwise comparisons across levels of factor variables. You can compare estimated cell means, marginal means, intercepts, marginal intercepts, slopes, or marginal slopes—collectively called margins. See [U] **20.18 Obtaining pairwise comparisons**, [R] **margins**, and [R] **margins, pwcompare**.

7. You can use the postestimation command `contrast` to obtain contrasts, which is to say, to compare levels of factor variables and their interactions. This command can also produce ANOVA-style tests of main effects, interactions effects, simple effects, and nested effects; and it can be used after most estimation commands. See **[U] 20.19 Obtaining contrasts, tests of interactions, and main effects**, [R] **contrast**, and [R] **margins, contrast**.

8. You can use the postestimation command `marginsplot` to graph any of the results produced by `margins`. And because `margins` can replicate any result produced by `pwcompare` and `contrast`, you can graph any result produced by them, too. See [R] **marginsplot**.

9. You can use the postestimation command `estat` to obtain common statistics associated with the model. The available statistics are documented in the postestimation section following the documentation of the estimation command, for instance, in [R] **regress postestimation** following [R] **regress**.

   You can always use the postestimation command `estat vce` to obtain the variance–covariance matrix of the estimators (VCE), presented as either a correlation matrix or a covariance matrix. (You can also obtain the estimated coefficients and covariance matrix as vectors and matrices and manipulate them with Stata's matrix capabilities; see **[U] 14.5 Accessing matrices created by Stata commands**.)

10. You can use the postestimation command `predict` to obtain predictions, residuals, influence statistics, and the like, either for the data on which you just estimated or for some other data. You can use postestimation command `predictnl` to obtain point estimates, standard errors, etc., for customized predictions. See [R] **predict** and [R] **predictnl**.

11. You can use the postestimation command `forecast` to perform dynamic and static forecasts, with optional forecast confidence intervals. This includes the ability to produce forecasts from multiple estimation commands, even when estimates imply simultaneous systems. An example of a simultaneous system is when `y2` predicts `y1` in estimation 1 and `y1` predicts `y2` in estimation 2. `forecast` provides many facilities for creating comparative forecast scenarios. See [TS] **forecast**.

12. You can refer to the values of coefficients and standard errors in expressions (such as with `generate`) by using standard notation; see **[U] 13.5 Accessing coefficients and standard errors**. You can refer in expressions to the values of other estimation-related statistics by using `e(`*resultname*`)`. For instance, all commands define `e(N)` recording the number of observations in the estimation subsample. After estimation, type `ereturn list` to see a list of all that is available. See the *Stored results* section in the estimation command's documentation for their definitions.

    An especially useful `e()` result is `e(sample)`: it returns 1 if an observation was used in the estimation and 0 otherwise, so you can add `if e(sample)` to the end of other commands to restrict them to the estimation subsample. You could type, for instance, `summarize if e(sample)`.

13. You can use the postestimation command `test` to perform tests on the estimated parameters (Wald tests of linear hypotheses), `testnl` to perform Wald tests of nonlinear hypotheses, and `lrtest` to perform likelihood-ratio tests. You can use the postestimation command `lincom` to obtain point estimates and confidence intervals for linear combinations of the estimated parameters and the postestimation command `nlcom` to obtain nonlinear combinations.

14. You can specify the `coeflegend` option at the time of estimation or when you redisplay results to see how to type your coefficients in postestimation commands, such as `test` and `lincom` (see [R] **test** and [R] **lincom**), and in expressions.

15. You can use the `statsby` prefix command (see [D] **statsby**) to fit models over each category in a categorical variable and collect the results in a Stata dataset.

16. You can use the `collect` suite of commands to collect estimation results and create customized tables from those results. See [TABLES] **Intro**.

17. You can use the postestimation command `etable` to easily create a table of estimation results from one or multiple estimation commands. See [R] **etable**.

18. You can use the postestimation command `estimates` to store estimation results by name for later retrieval or for displaying/comparing multiple models by using `estimates`, or to save estimation results in a file; see [R] **estimates**.

19. You can use the postestimation command `_estimates` to hold estimates, perform other estimation commands, and then restore the prior estimates. This is of particular interest to programmers. See [P] **_estimates**.

20. You can use the postestimation command `suest` to obtain the joint parameter vector and variance–covariance matrix for coefficients from two different models by using seemingly unrelated estimation. This is especially useful for testing the equality, say, of coefficients across models. See [R] **suest**.

21. You can use the postestimation command `hausman` to perform Hausman model-specification tests by using `hausman`; see [R] **hausman**.

22. With some exceptions, you can specify the `vce(robust)` option at the time of estimation to obtain the Huber/White/robust alternate estimate of variance, or you can specify the `vce(cluster` *clustvar*`)` option to relax the assumption of independence of the observations; see [R] ***vce_option***.

    Most estimation commands also allow a `vce(`*vcetype*`)` option to specify other alternative variance estimators—the allowed alternative variance estimators are documented with the estimator—and usually `vce(opg)`, `vce(bootstrap)`, and `vce(jackknife)` are available.

    Where `vce(bootstrap)` and `vce(jackknife)` are available, we recommend using them instead of the prefix commands `bootstrap` and `jackknife`.

As a rule, the points discussed briefly above and in more detail later in this entry do not apply to the Bayesian analysis or the Bayesian model averaging commands. For more information about Bayesian analysis commands, see the *Stata Bayesian Analysis Reference Manual*. For more information about Bayesian model averaging commands, see the *Stata Bayesian Model Averaging Reference Manual*.

## 20.2 Standard syntax

You can combine Stata's `if` *exp* and `in` *range* with any estimation command. Estimation commands also allow `by` *varlist*`:`, where it would be sensible.

### ▷ Example 1

We have data on 74 automobiles that record the mileage rating (`mpg`), weight (`weight`), and whether the car is domestic or foreign produced (`foreign`). We can fit a linear regression model of `mpg` on `weight` and the square of `weight`, using just the foreign-made automobiles, by typing

```
. use https://www.stata-press.com/data/r18/auto2
(1978 automobile data)
. regress mpg weight c.weight#c.weight if foreign
```

| Source | SS | df | MS | Number of obs | = | 22 |
|---|---|---|---|---|---|---|
| | | | | F(2, 19) | = | 8.31 |
| Model | 428.256889 | 2 | 214.128444 | Prob > F | = | 0.0026 |
| Residual | 489.606747 | 19 | 25.7687762 | R-squared | = | 0.4666 |
| | | | | Adj R-squared | = | 0.4104 |
| Total | 917.863636 | 21 | 43.7077922 | Root MSE | = | 5.0763 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0132182 | .0275711 | -0.48 | 0.637 | -.0709252 | .0444888 |
| c.weight# c.weight | 5.50e-07 | 5.41e-06 | 0.10 | 0.920 | -.0000108 | .0000119 |
| _cons | 52.33775 | 34.1539 | 1.53 | 0.142 | -19.14719 | 123.8227 |

We use the factor-variable notation `c.weight#c.weight` to add the square of `weight` to our regression; see **[U] 11.4.3 Factor variables**.

We can run separate regressions for the domestic and foreign-produced automobiles with the `by` *varlist*`:` prefix:

```
. by foreign: regress mpg weight c.weight#c.weight

-> foreign = Domestic
      Source |       SS           df       MS      Number of obs   =        52
-------------+----------------------------------   F(2, 49)        =     91.64
       Model |  905.395466         2  452.697733   Prob > F        =    0.0000
    Residual |  242.046842        49  4.93973146   R-squared       =    0.7891
-------------+----------------------------------   Adj R-squared   =    0.7804
       Total |  1147.44231        51  22.4988688   Root MSE        =    2.2226

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0131718   .0032307    -4.08   0.000    -.0196642   -.0066794
             |
    c.weight#|
    c.weight |   1.11e-06   4.95e-07     2.25   0.029     1.19e-07    2.11e-06
             |
       _cons |   50.74551   5.162014     9.83   0.000     40.37205    61.11896
------------------------------------------------------------------------------

-> foreign = Foreign
      Source |       SS           df       MS      Number of obs   =        22
-------------+----------------------------------   F(2, 19)        =      8.31
       Model |  428.256889         2  214.128444   Prob > F        =    0.0026
    Residual |  489.606747        19  25.7687762   R-squared       =    0.4666
-------------+----------------------------------   Adj R-squared   =    0.4104
       Total |  917.863636        21  43.7077922   Root MSE        =    5.0763

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0132182   .0275711    -0.48   0.637    -.0709252    .0444888
             |
    c.weight#|
    c.weight |   5.50e-07   5.41e-06     0.10   0.920    -.0000108    .0000119
             |
       _cons |   52.33775    34.1539     1.53   0.142    -19.14719    123.8227
------------------------------------------------------------------------------
```

Although all estimation commands allow `if` *exp* and `in` *range*, only some allow the `by` *varlist*`:` prefix. For `by()`, the duration of Stata's memory is limited: it remembers the last set of estimates only. This means that, if we were to use any of the other features described below, they would use the last regression estimated, which right now is `mpg` on `weight` and square of `weight` for the `Foreign` subsample.

We can instead collect the statistics from each of the by-groups by using the `statsby` prefix; see [D] **statsby**.

```
. statsby, by(foreign): regress mpg weight c.weight#c.weight
(running regress on estimation sample)

      Command: regress mpg weight c.weight#c.weight
           By: foreign

Statsby groups:
..
```

`statsby` runs the regression first on domestic cars and then on foreign cars, and it saves the coefficients by overwriting our dataset. Do not worry; if the dataset has not been previously saved, `statsby` will refuse to run unless we also specify the `clear` option.

Here is what we now have in memory.

```
. list
```

| | foreign | _b_weight | _stat_2 | _b_cons |
|---|---|---|---|---|
| 1. | Domestic | -.0131718 | 1.11e-06 | 50.74551 |
| 2. | Foreign | -.0132182 | 5.50e-07 | 52.33775 |

These are the coefficients from the two regressions above. `statsby` does not know how to name the coefficient for `c.weight#c.weight`, so it labels the coefficient with the generic name `_stat_2`. We can also save the standard errors and other statistics from the regressions; see [D] **statsby**.

◁

## 20.3 Replaying prior results

When you type an estimation command without arguments, it redisplays prior results.

### ▷ Example 2

To perform a regression of `mpg` on the variables `weight` and `displacement`, we could type

```
. use https://www.stata-press.com/data/r18/auto2, clear
(1978 automobile data)
. regress mpg weight displacement
```

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(2, 71) | = | 66.79 |
| Model | 1595.40969 | 2 | 797.704846 | Prob > F | = | 0.0000 |
| Residual | 848.049768 | 71 | 11.9443629 | R-squared | = | 0.6529 |
| | | | | Adj R-squared | = | 0.6432 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.4561 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0065671 | .0011662 | -5.63 | 0.000 | -.0088925 | -.0042417 |
| displacement | .0052808 | .0098696 | 0.54 | 0.594 | -.0143986 | .0249602 |
| _cons | 40.08452 | 2.02011 | 19.84 | 0.000 | 36.05654 | 44.11251 |

We now go on to do other things—summarizing data, listing observations, performing hypothesis tests, or anything else. If we decide that we want to see the last set of estimates again, we type the estimation command without arguments.

```
. regress
```

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(2, 71) | = | 66.79 |
| Model | 1595.40969 | 2 | 797.704846 | Prob > F | = | 0.0000 |
| Residual | 848.049768 | 71 | 11.9443629 | R-squared | = | 0.6529 |
| | | | | Adj R-squared | = | 0.6432 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.4561 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0065671 | .0011662 | -5.63 | 0.000 | -.0088925 | -.0042417 |
| displacement | .0052808 | .0098696 | 0.54 | 0.594 | -.0143986 | .0249602 |
| _cons | 40.08452 | 2.02011 | 19.84 | 0.000 | 36.05654 | 44.11251 |

We can also specify most reporting options on replay. For example, if we want to see a legend of terms with which to refer to the estimated coefficients in subsequent commands, we can type

```
. regress, coeflegend
(output omitted)
```

See [U] **20.12 Accessing estimated coefficients** for an example using legend terms.

These features work with every estimation command, so we could just as well have used, say, `stcox` or `logit`. ◁

## 20.4 Cataloging estimation results

Stata keeps only the results of the most recently fit model in active memory. You can use Stata's `estimates` command, however, to temporarily store estimation results for displaying, comparing, cross-model testing, etc., during the same session. You can also save estimation results to disk, but that will be the subject of the next section. You may temporarily store up to 300 sets of estimation results.

### ▷ Example 3

Continuing with our automobile data, we fit four models, give each one a title, and then store them. We fit the models `quietly` to minimize output.

```
. quietly regress mpg weight displ
. estimates title: Linear regression, base model
. estimates store r_base
. quietly regress mpg weight displ foreign
. estimates title: Linear regression, alternate model
. estimates store r_alt
. quietly qreg mpg weight displ
. estimates title: Quantile regression, base model
. estimates store q_base
. quietly qreg mpg weight displ foreign
. estimates title: Quantile regression, alternate model
. estimates store q_alt
```

We saved the four models under the names `r_base`, `r_alt`, `q_base`, and `q_alt`, but if we forget, we can ask to see a directory of what is stored:

```
. estimates dir
```

| Name | Command | Dependent variable | Number of param. | Title |
|---|---|---|---|---|
| r_base | regress | mpg | 3 | *Linear regression, base model* |
| r_alt | regress | mpg | 4 | *Linear regression, alternate model* |
| q_base | qreg | mpg | 3 | *Quantile regression, base model* |
| q_alt | qreg | mpg | 4 | *Quantile regression, alternate model* |

We can ask Stata to replay any of the previous models:

```
. estimates replay r_base
```

Model **r_base** (*Linear regression, base model*)

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(2, 71) | = | 66.79 |
| Model | 1595.40969 | 2 | 797.704846 | Prob > F | = | 0.0000 |
| Residual | 848.049768 | 71 | 11.9443629 | R-squared | = | 0.6529 |
| | | | | Adj R-squared | = | 0.6432 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.4561 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0065671 | .0011662 | -5.63 | 0.000 | -.0088925 | -.0042417 |
| displacement | .0052808 | .0098696 | 0.54 | 0.594 | -.0143986 | .0249602 |
| _cons | 40.08452 | 2.02011 | 19.84 | 0.000 | 36.05654 | 44.11251 |

Or we can ask to see all the models in a combined table:

```
. estimates table _all
```

| Variable | r_base | r_alt | q_base | q_alt |
|---|---|---|---|---|
| weight | -.00656711 | -.00677449 | -.00581172 | -.00595056 |
| displacement | .00528078 | .00192865 | .0042841 | .00018552 |
| foreign | | -1.6006312 | | -2.1326005 |
| _cons | 40.084522 | 41.847949 | 37.559865 | 39.213348 |

`estimates` displayed just the coefficients, but we could ask for other statistics.

We can also select one of the stored estimates to be made active, making it as if we had just fit the model:

```
. estimates restore r_alt
(results r_alt are active now)
. regress
```

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(3, 70) | = | 45.88 |
| Model | 1619.71935 | 3 | 539.906448 | Prob > F | = | 0.0000 |
| Residual | 823.740114 | 70 | 11.7677159 | R-squared | = | 0.6629 |
| | | | | Adj R-squared | = | 0.6484 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.4304 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0067745 | .0011665 | -5.81 | 0.000 | -.0091011 | -.0044479 |
| displacement | .0019286 | .0100701 | 0.19 | 0.849 | -.0181556 | .0220129 |
| foreign | -1.600631 | 1.113648 | -1.44 | 0.155 | -3.821732 | .6204699 |
| _cons | 41.84795 | 2.350704 | 17.80 | 0.000 | 37.15962 | 46.53628 |

◁

You can do a lot more with `estimates`; see [R] **estimates**. In particular, `estimates` makes it easy to perform cross-model tests, such as the Hausman specification test.

## 20.5 Saving estimation results

`estimates` can also save estimation results into a file.

```
. estimates save alt
file alt.ster saved
```

That saved the active estimation results, meaning the ones we just estimated or, in our case, the ones we just restored. Later, even in another Stata session, we could reload our estimates:

```
. estimates use alt
. regress
```

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(3, 70) | = | 45.88 |
| Model | 1619.71935 | 3 | 539.906448 | Prob > F | = | 0.0000 |
| Residual | 823.740114 | 70 | 11.7677159 | R-squared | = | 0.6629 |
| | | | | Adj R-squared | = | 0.6484 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.4304 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0067745 | .0011665 | -5.81 | 0.000 | -.0091011 | -.0044479 |
| displacement | .0019286 | .0100701 | 0.19 | 0.849 | -.0181556 | .0220129 |
| foreign | -1.600631 | 1.113648 | -1.44 | 0.155 | -3.821732 | .6204699 |
| _cons | 41.84795 | 2.350704 | 17.80 | 0.000 | 37.15962 | 46.53628 |

There is one important difference between storing results in memory and saving them in a file: `e(sample)` is lost. We have not discussed `e(sample)` yet, but it allows us to identify the observations among those currently in memory that were used in the estimation. For instance, after estimation, we could type

```
. summarize mpg weight displ foreign if e(sample)
```

and see the summary statistics of the relevant data. We could do that after `estimates restore`, too. But we cannot do it after `estimates use`. Part of the reason is that we might not even have the relevant data in memory. Even if we do, however, here is what will happen:

```
. summarize mpg weight displ foreign if e(sample)
```

| Variable | Obs | Mean | Std. dev. | Min | Max |
|---|---|---|---|---|---|
| mpg | 0 | | | | |
| weight | 0 | | | | |
| displacement | 0 | | | | |
| foreign | 0 | | | | |

Stata will just assume that none of the data in memory played a role in obtaining the estimation results.

There is more worth knowing. You could, for instance, type `estimates describe` to see the command line that produced the estimates. See [R] **estimates**.

## 20.6 Specification search tools

Stata's lasso commands select covariates and fit models for continuous, binary, and count outcomes. See [LASSO] **Lasso intro** for an overview of lasso features.

The commands `stepwise`, `fp`, and `mfp` are not really estimation commands but are combined with estimation commands to assist in specification searches.

`stepwise`, one of Stata's prefix commands, provides stepwise estimation. You can use the `stepwise` prefix with some, but not all, estimation commands. See [R] **stepwise** for a list of supported estimation commands.

`fp` and `mfp` are commands to assist you in performing fractional-polynomial functional specification searches. See [R] **fp** and [R] **mfp** for additional information.

## 20.7 Specifying the estimation subsample

You specify the estimation subsample—the sample to be used in estimation—by specifying the `if` *exp* and `in` *range* qualifiers with the estimation command.

Once an estimation command has been run or previous estimates restored, Stata remembers the estimation subsample, and you can use the qualifier `if e(sample)` on the end of other Stata commands. The term estimation subsample refers to the set of observations used to produce the active estimation results. That might turn out to be all the observations (as it was in the above example) or only some of the observations:

```
. regress mpg weight 5.rep78 if foreign
      Source |       SS           df       MS      Number of obs   =        21
-------------+----------------------------------   F(2, 18)        =     10.21
       Model |  423.317154         2  211.658577   Prob > F        =    0.0011
    Residual |   372.96856        18  20.7204756   R-squared       =    0.5316
-------------+----------------------------------   Adj R-squared   =    0.4796
       Total |  796.285714        20  39.8142857   Root MSE        =     4.552

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0131402   .0029684    -4.43   0.000    -.0193765   -.0069038
             |
       rep78 |
   Excellent |   5.052676    2.13492     2.37   0.029     .5673764    9.537977
       _cons |   52.86088   6.540147     8.08   0.000     39.12054    66.60122
------------------------------------------------------------------------------

. summarize mpg weight 5.rep78 if e(sample)
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         mpg |         21    25.28571    6.309856         17         41
      weight |         21    2263.333    364.7099       1760       3170
             |
       rep78 |
   Excellent |         21    .4285714    .5070926          0          1
```

Twenty-one observations were used in the above regression, and we subsequently obtained the means for those same 21 observations by typing `summarize ... if e(sample)`. Observations were dropped for two reasons: we specified `if foreign` when we ran the regression, and there were observations for which `5.rep78` was missing. The reason does not matter; `e(sample)` is true if the observation was used and is false otherwise.

You can use `if e(sample)` on the end of any Stata command that allows `if` *exp*.

Here, Stata has a shorthand command that produces the same results as `summarize ... if e(sample)`:

```
. estat summarize, label
Estimation sample regress                         Number of obs =         21
```

| Variable | Mean | Std. dev. | Min | Max | Label |
|---|---|---|---|---|---|
| mpg | 25.28571 | 6.309856 | 17 | 41 | Mileage (mpg) |
| weight | 2263.333 | 364.7099 | 1760 | 3170 | Weight (lbs.) |
| rep78 | | | | | Repair record 1978 |
| Excellent | .4285714 | .5070926 | 0 | 1 | |

See [R] **estat summarize**.

## 20.8 Specifying the width of confidence intervals

You can specify the width of the confidence intervals for the coefficients by using the `level()` option at estimation or when you play back the results.

### ▷ Example 4

To obtain narrower, 90% confidence intervals when we fit the model, we type

```
. regress mpg weight displ, level(90)
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     66.79
       Model |  1595.40969         2  797.704846   Prob > F        =    0.0000
    Residual |  848.049768        71  11.9443629   R-squared       =    0.6529
-------------+----------------------------------   Adj R-squared   =    0.6432
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4561

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [90% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065671   .0011662    -5.63   0.000    -.0085108   -.0046234
displacement |   .0052808   .0098696     0.54   0.594    -.0111679    .0217294
       _cons |   40.08452    2.02011    19.84   0.000     36.71781    43.45124
------------------------------------------------------------------------------
```

If we subsequently typed `regress` without arguments, 95% confidence intervals would be reported because that is the default. If we initially fit the model with 95% confidence intervals, we could later type `regress, level(90)` to redisplay results with 90% confidence intervals.

Also, we could type `set level 90` to make 90% intervals our default; see [R] **level**.

Stata allows noninteger confidence intervals between 10.00 and 99.99, with a maximum of two digits following the decimal point. For instance, we could type

```
. regress mpg weight displ, level(92.5)
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     66.79
       Model |  1595.40969         2  797.704846   Prob > F        =    0.0000
    Residual |  848.049768        71  11.9443629   R-squared       =    0.6529
-------------+----------------------------------   Adj R-squared   =    0.6432
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4561

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [92.5% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065671   .0011662    -5.63   0.000    -.0086745   -.0044597
displacement |   .0052808   .0098696     0.54   0.594    -.0125535    .023115
       _cons |   40.08452   2.02011     19.84   0.000     36.43419    43.73485
------------------------------------------------------------------------------
```

## 20.9 Formatting the coefficient table

You can change the formatting of the coefficient table with the `sformat()`, `pformat()`, and `cformat()` options. The `sformat()` option changes the output format of test statistics; `pformat()` changes $p$-values; and `cformat()` changes coefficients, standard errors, and confidence limits. We can reduce the number of decimal places by specifying `%f` fixed-width formats:

```
. regress mpg weight displ, cformat(%6.3f) sformat(%4.1f) pformat(%4.2f)
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     66.79
       Model |  1595.40969         2  797.704846   Prob > F        =    0.0000
    Residual |  848.049768        71  11.9443629   R-squared       =    0.6529
-------------+----------------------------------   Adj R-squared   =    0.6432
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4561

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |    -0.007      0.001     -5.6   0.00       -0.009      -0.004
displacement |     0.005      0.010      0.5   0.59       -0.014       0.025
       _cons |    40.085      2.020     19.8   0.00        36.057      44.113
------------------------------------------------------------------------------
```

The option `cformat(%6.3f)`, for example, fixes a width of six characters with three digits to the right of the decimal point. For more information on formats, see [U] **12.5.1 Numeric formats**.

The formatting options may also be specified when replaying results, so you can try different formats without refitting the model:

```
. regress, cformat(%7.4f)
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     66.79
       Model |  1595.40969         2  797.704846   Prob > F        =    0.0000
    Residual |  848.049768        71  11.9443629   R-squared       =    0.6529
-------------+----------------------------------   Adj R-squared   =    0.6432
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4561

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |   -0.0066     0.0012     -5.63   0.000      -0.0089     -0.0042
displacement |    0.0053     0.0099      0.54   0.594      -0.0144      0.0250
       _cons |   40.0845     2.0201     19.84   0.000       36.0565     44.1125
------------------------------------------------------------------------------
```

## 20.10 Obtaining the variance–covariance matrix

Typing `estat vce` displays the variance–covariance matrix of the estimators in active memory.

▷ Example 5

In example 2, we typed `regress mpg weight displacement`. The full variance–covariance matrix of the estimators can be displayed at any time after estimation:

```
. estat vce
Covariance matrix of coefficients of regress model
        e(V) |      weight  displace~t       _cons
-------------+------------------------------------
      weight |   1.360e-06
displacement |   -.0000103    .00009741
       _cons |  -.00207455    .01188356   4.0808455
```

Typing `estat vce` with the `corr` option presents this matrix as a correlation matrix:

```
. estat vce, corr
Correlation matrix of coefficients of regress model
        e(V) |    weight  displa~t     _cons
-------------+------------------------------
      weight |    1.0000
displacement |   -0.8949    1.0000
       _cons |   -0.8806    0.5960    1.0000
```

See [R] **estat vce**.

Also, Stata's matrix commands understand that `e(V)` refers to the matrix:

```
. matrix list e(V)
symmetric e(V)[3,3]
                  weight  displacement         _cons
      weight     1.360e-06
displacement     -.0000103     .00009741
       _cons    -.00207455     .01188356     4.0808455
. matrix Vinv = invsym(e(V))
. matrix list Vinv
symmetric Vinv[3,3]
                  weight  displacement         _cons
      weight      60175851
displacement     4081161.2     292709.46
       _cons     18706.732     1222.3339     6.1953911
```

See [U] **14.5 Accessing matrices created by Stata commands**.

◁

## 20.11 Obtaining predicted values

Our discussion below, although cast in terms of predicted values, applies equally to the other statistics generated by `predict`; see [R] **predict**.

When Stata fits a model, whether it is regression or anything else, it internally stores the results, including the estimated coefficients and the variable names. The `predict` command allows you to use that information.

## Example 6

Let's perform a linear regression of `mpg` on `weight` and the square of `weight`:

```
. regress mpg weight c.weight#c.weight
```

| Source | SS | df | MS | | |
|---|---|---|---|---|---|
| | | | | Number of obs | = 74 |
| | | | | F(2, 71) | = 72.80 |
| Model | 1642.52197 | 2 | 821.260986 | Prob > F | = 0.0000 |
| Residual | 800.937487 | 71 | 11.2808097 | R-squared | = 0.6722 |
| | | | | Adj R-squared | = 0.6630 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = 3.3587 |

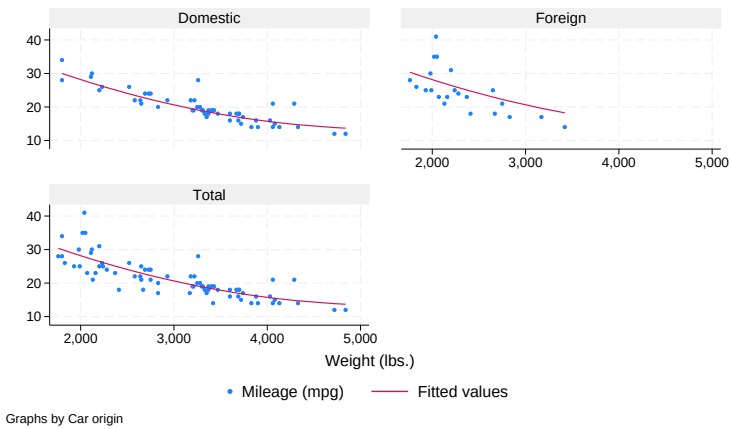
| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0141581 | .0038835 | -3.65 | 0.001 | -.0219016 | -.0064145 |
| c.weight# c.weight | 1.32e-06 | 6.26e-07 | 2.12 | 0.038 | 7.67e-08 | 2.57e-06 |
| _cons | 51.18308 | 5.767884 | 8.87 | 0.000 | 39.68225 | 62.68392 |

After the regression, `predict` is defined to be

$$-0.0141581\texttt{weight} + 1.32 \times 10^{-6}\texttt{weight}^2 + 51.18308$$

(Actually, it is more precise because the coefficients are internally stored at much higher precision than shown in the output.) Thus, we can create a new variable—let's call it `fitted`—equal to the prediction by typing `predict fitted` and then use `scatter` to display the fitted and actual values separately for domestic and foreign automobiles:

```
. predict fitted
(option xb assumed; fitted values)
. scatter mpg fitted weight, by(foreign, total style(altleg)) c(. l) m(o i) sort
```




`predict` can calculate much more than just predicted values. For `predict` after linear regression, `predict` can calculate residuals, standardized residuals, Studentized residuals, influence statistics, and more. In any case, we specify what is to be calculated via an option, so if we wanted the residuals stored in new variable `r`, we would type

```
. predict r, resid
```

The options that may be specified following `predict` vary according to the estimation command previously used; the `predict` options are documented along with the estimation command. For instance, to discover all the things `predict` can do following `regress`, see [R] **regress**.

◁

### 20.11.1 Using predict

The use of `predict` is not limited to linear regression. `predict` can be used after any estimation command.

▷ Example 7

You fit a logistic regression model of whether a car is manufactured outside the United States on the basis of its weight and mileage rating using either the `logistic` or the `logit` command; see [R] **logistic** and [R] **logit**. We will use `logit`.

```
. use https://www.stata-press.com/data/r18/auto2, clear
(1978 automobile data)
. logit foreign weight mpg
Iteration 0:  Log likelihood =  -45.03321
Iteration 1:  Log likelihood = -29.238536
Iteration 2:  Log likelihood = -27.244139
Iteration 3:  Log likelihood = -27.175277
Iteration 4:  Log likelihood = -27.175156
Iteration 5:  Log likelihood = -27.175156
Logistic regression                                     Number of obs =     74
                                                        LR chi2(2)    =  35.72
                                                        Prob > chi2   = 0.0000
Log likelihood = -27.175156                             Pseudo R2     = 0.3966
```

| foreign | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0039067 | .0010116 | -3.86 | 0.000 | -.0058894 | -.001924 |
| mpg | -.1685869 | .0919175 | -1.83 | 0.067 | -.3487418 | .011568 |
| _cons | 13.70837 | 4.518709 | 3.03 | 0.002 | 4.851859 | 22.56487 |

After `logit`, `predict` without options calculates the probability of a positive outcome (we learned that by looking at [R] **logit**). To obtain the predicted probabilities that each car is manufactured outside the United States, we type

```
. predict probhat
(option pr assumed; Pr(foreign))
. summarize probhat
```

| Variable | Obs | Mean | Std. dev. | Min | Max |
|---|---|---|---|---|---|
| probhat | 74 | .2972973 | .3052979 | .000729 | .8980594 |

```
. list make mpg weight foreign probhat in 1/5
```

| | make | mpg | weight | foreign | probhat |
|---|---|---|---|---|---|
| 1. | AMC Concord | 22 | 2,930 | Domestic | .1904363 |
| 2. | AMC Pacer | 17 | 3,350 | Domestic | .0957767 |
| 3. | AMC Spirit | 22 | 2,640 | Domestic | .4220815 |
| 4. | Buick Century | 20 | 3,250 | Domestic | .0862625 |
| 5. | Buick Electra | 15 | 4,080 | Domestic | .0084948 |

◁

### 20.11.2 Making in-sample predictions

`predict` does not retrieve a vector of prerecorded values—it calculates the predictions on the basis of the recorded coefficients and the data currently in memory. In the above examples, when we typed things like

```
. predict probhat
```

`predict` filled in the prediction everywhere that it could be calculated.

We sometimes have more data in memory than were used by the estimation command, either because we explicitly ignored some of the observations by specifying an `if` *exp* with the estimation command or because there are missing values. In such cases, if we want to restrict the calculation to the estimation subsample, we would do that in the usual way by adding `if e(sample)` to the end of the command:

```
. predict probhat if e(sample)
```

### 20.11.3 Making out-of-sample predictions

Because `predict` makes its calculations on the basis of the recorded coefficients and the data in memory, `predict` can do more than calculate predicted values for the data on which the estimation took place—it can make out-of-sample predictions, as well.

If you fit your model on a subset of the observations, you could then predict the outcome for all the observations:

```
. logit foreign weight mpg if rep78 > 3
. predict pall
```

If you do not specify `if e(sample)` at the end of the `predict` command, `predict` calculates the predictions for all observations possible.

In fact, because `predict` works from the active estimation results, you can use `predict` with *any* dataset that contains the necessary variables.

▷ Example 8

Continuing with our previous `logit` example, assume that we have a second dataset containing the `mpg` and `weight` of a different sample of cars. We have just fit your model and now continue:

```
. use otherdat, clear
(Different cars)
. predict probhat                    Stata remembers the previous model
(option pr assumed; Pr(foreign))
. summarize probhat foreign
    Variable |       Obs        Mean    Std. dev.       Min        Max
-------------+--------------------------------------------------------
     probhat |        12    .2505068    .3187104   .0084948   .8920776
     foreign |        12    .1666667    .3892495          0          1
```

◁

▷ Example 9

We can obtain out-of-sample predictions in many ways. Above, we estimated on one dataset and then used another. If our first dataset had contained both sets of cars, marked, say, by the variable `difcars` being 0 if from the first sample and 1 if from the second, we could type

```
. logit foreign weight mpg if difcars==0
same output as above appears
. predict probhat
(option pr assumed; Pr(foreign))
. summarize probhat foreign if difcars==1
same output as directly above appears
```

If we just had a few additional cars, we could even input them after estimation. Assume that our data once again contain only the first sample of cars, and assume that we are interested in an additional sample of only two cars; we could type

```
. use https://www.stata-press.com/data/r18/auto2
(1978 automobile data)
. keep make mpg weight foreign
. logit foreign weight mpg
same output as above appears
. input
                    make       mpg     weight   foreign
 75. "Merc. Zephyr" 20 2830 0                  we type in our new data
 76. "VW Dasher" 23 2160 1
 77. end
. predict probhat                              obtain all the predictions
(option pr assumed; Pr(foreign))
. list in -2/l
```

| | make | mpg | weight | foreign | probhat |
|---|---|---|---|---|---|
| 75. | Merc. Zephyr | 20 | 2,830 | Domestic | .3275397 |
| 76. | VW Dasher | 23 | 2,160 | Foreign | .8009743 |

◁

## 20.11.4 Obtaining standard errors, tests, and confidence intervals for predictions

When you use `predict`, you create, for each observation in the prediction sample, a statistic that is a function of the data and the estimated model parameters. You also could have generated your own customized predictions by using `generate`. In either case, to get standard errors, Wald tests, and confidence intervals for your predictions, use `predictnl`. For example, if we want the standard errors for our predicted probabilities, we could type

```
. drop probhat
. predictnl probhat = predict(), se(phat_se)
. list in 1/5
```

| | make | mpg | weight | foreign | probhat | phat_se |
|---|---|---|---|---|---|---|
| 1. | AMC Concord | 22 | 2,930 | Domestic | .1904363 | .0658387 |
| 2. | AMC Pacer | 17 | 3,350 | Domestic | .0957767 | .0536297 |
| 3. | AMC Spirit | 22 | 2,640 | Domestic | .4220815 | .0892845 |
| 4. | Buick Century | 20 | 3,250 | Domestic | .0862625 | .0461928 |
| 5. | Buick Electra | 15 | 4,080 | Domestic | .0084948 | .0093079 |

Comparing this output with our previous listing of the first five predicted probabilities, you will notice that the output is identical except that we now have an additional variable, `phat_se`, which contains the estimated standard error for each predicted probability.

We first had to `drop probhat` because `predictnl` will regenerate it. Note also the use of `predict()` within `predictnl`—it specified that we wanted to generate a point estimate (and standard error) for the default prediction after `logit`; see [R] **predictnl** for more details.

## 20.12 Accessing estimated coefficients

You can access coefficients and standard errors after estimation by referring to `_b[`*name*`]` and `_se[`*name*`]`; see [U] **13.5 Accessing coefficients and standard errors**.

### ▷ Example 10

Let's return to linear regression. We are doing a study of earnings of men and women at a particular company. In addition to each person's earnings, we have information on their educational attainment and tenure with the company. We type the following:

```
. regress lnearn ed tenure i.female female#(c.ed c.tenure)
```
(*output omitted*)

If you are not familiar with the `#` notation, see [U] **11.4.3 Factor variables**.

We now wish to predict everyone's income as if they were male and then compare these as-if earnings with the actual earnings:

```
. generate asif = _b[_cons] + _b[ed]*ed + _b[tenure]*tenure
```

◁

### ▷ Example 11

We are analyzing the mileage of automobiles and are using a slightly more sophisticated model than any we have used so far. As we have previously, we will fit a linear regression model of `mpg` on `weight` and the square of `weight`, but we also add the interaction of `foreign` with `weight`, the car's gear ratio (`gear_ratio`), and `foreign` interacted with `gear_ratio`. We will use factor-variable notation to create the squared term and the interactions; see [U] **11.4.3 Factor variables**.

```
. use https://www.stata-press.com/data/r18/auto2, clear
(1978 automobile data)
. regress mpg weight c.weight#c.weight i.foreign#c.weight gear_ratio
> i.foreign#c.gear_ratio
```

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(5, 68) | = | 33.44 |
| Model | 1737.05293 | 5 | 347.410585 | Prob > F | = | 0.0000 |
| Residual | 706.406534 | 68 | 10.3883314 | R-squared | = | 0.7109 |
| | | | | Adj R-squared | = | 0.6896 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.2231 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0118517 | .0045136 | -2.63 | 0.011 | -.0208584 | -.002845 |
| c.weight# c.weight | 9.81e-07 | 7.04e-07 | 1.39 | 0.168 | -4.25e-07 | 2.39e-06 |
| foreign# c.weight Foreign | -.0032241 | .0015577 | -2.07 | 0.042 | -.0063326 | -.0001157 |
| gear_ratio | 1.159741 | 1.553418 | 0.75 | 0.458 | -1.940057 | 4.259539 |
| foreign# c.gear_ratio Foreign | 1.597462 | 1.205313 | 1.33 | 0.189 | -.8077036 | 4.002627 |
| _cons | 44.61644 | 8.387943 | 5.32 | 0.000 | 27.87856 | 61.35432 |

If you are not experienced in both regression technology and automobile technology, you may find it difficult to interpret this regression. Putting aside issues of statistical significance, we find that mileage decreases with a car's weight but increases with the square of weight; decreases even more rapidly with weight for foreign cars; increases with higher gear ratio; and increases even more rapidly with higher gear ratio in foreign cars.

Thus, do foreign cars yield better or worse gas mileage? Results are mixed. As the foreign cars' weight increases, they do more poorly in relation to domestic cars, but they do better at higher gear ratios. One way to compare the results is to predict what mileage foreign cars would have if they were manufactured domestically. The regression provides all the information necessary for making that calculation. Mileage for domestic cars is estimated to be

$$-0.012\,\texttt{weight} + 9.81 \times 10^{-7}\,\texttt{weight}^2 + 1.160\,\texttt{gear\_ratio} + 44.6$$

We can use that equation to predict the mileage of foreign cars and then compare it with the true outcome. The `_b[]` function simplifies reference to the estimated coefficients. We can type

```
. generate asif=_b[weight]*weight + _b[c.weight#c.weight]*c.weight#c.weight +
>  _b[gear_ratio]*gear_ratio + _b[_cons]
```

`_b[weight]` refers to the estimated coefficient on `weight`, `_b[c.weight#c.weight]` to the estimated coefficient on `c.weight#c.weight`, and so on.

We might now ask how the actual mileage of a Honda compares with the `asif` prediction:

```
. list make asif mpg if strpos(make,"Honda")
```

| | make | asif | mpg |
|---|---|---|---|
| 61. | Honda Accord | 26.52597 | 25 |
| 62. | Honda Civic | 30.62202 | 28 |

Notice the way we constructed our `if` clause to select Hondas. `strpos()` is the string function that returns the location in the first string where the second string is found or, if the second string does not occur in the first, returns 0. Thus any recorded `make` that contains the string "`Honda`" anywhere in it would be listed; see [FN] **String functions**.

We find that both Honda models yield slightly lower gas mileage than the `asif` domestic car–based prediction. (We do not endorse this model as a complete model of the determinants of mileage, nor do we single out Honda for any special scorn. In fact, please note that the observed values are within the root mean squared error of the average prediction.)

We might wish to compare the overall average `mpg` and the `asif` prediction over all foreign cars in the data:

```
. summarize mpg asif if foreign
```

| Variable | Obs | Mean | Std. dev. | Min | Max |
|---|---|---|---|---|---|
| mpg | 22 | 24.77273 | 6.611187 | 14 | 41 |
| asif | 22 | 26.67124 | 3.142912 | 19.70466 | 30.62202 |

We find that, on average, foreign cars yield slightly lower mileage than our `asif` prediction. This might lead us to ask if any foreign cars do better than the `asif` prediction:

```
. list make asif mpg if foreign & mpg>asif, sep(0)
```

| | make | asif | mpg |
|---|---|---|---|
| 55. | BMW 320i | 24.31697 | 25 |
| 57. | Datsun 210 | 28.96818 | 35 |
| 63. | Mazda GLC | 29.32015 | 30 |
| 66. | Subaru | 28.85993 | 35 |
| 68. | Toyota Corolla | 27.01144 | 31 |
| 71. | VW Diesel | 28.90355 | 41 |

We find six such automobiles.

◁

# 20.13 Performing hypothesis tests on the coefficients

## 20.13.1 Linear tests

After estimation, `test` is used to perform tests of linear hypotheses on the basis of the variance–covariance matrix of the estimators (Wald tests).

### ▷ Example 12

Using the automobile data, we perform the following regression:

```
. use https://www.stata-press.com/data/r18/auto2, clear
(1978 automobile data)
. generate weightsq=weight^2
. regress mpg weight weightsq foreign
```

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(3, 70) | = | 52.25 |
| Model | 1689.15372 | 3 | 563.05124 | Prob > F | = | 0.0000 |
| Residual | 754.30574 | 70 | 10.7757963 | R-squared | = | 0.6913 |
| | | | | Adj R-squared | = | 0.6781 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.2827 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0165729 | .0039692 | -4.18 | 0.000 | -.0244892 | -.0086567 |
| weightsq | 1.59e-06 | 6.25e-07 | 2.55 | 0.013 | 3.45e-07 | 2.84e-06 |
| foreign | -2.2035 | 1.059246 | -2.08 | 0.041 | -4.3161 | -.0909002 |
| _cons | 56.53884 | 6.197383 | 9.12 | 0.000 | 44.17855 | 68.89913 |

(Note: `test` has many syntaxes and features, so do not use this example as an excuse for not reading [R] **test**.) We can use the `test` command to calculate the joint significance of `weight` and `weightsq`:

```
. test weight weightsq
 ( 1)  weight = 0
 ( 2)  weightsq = 0
       F(  2,    70) =   60.83
            Prob > F =    0.0000
```

We are not limited to testing whether the coefficients are 0. We can test whether the coefficient on `foreign` is $-2$ by typing

```
. test foreign = -2
 ( 1)  foreign = -2
       F(  1,    70) =    0.04
            Prob > F =    0.8482
```

We can even test more complicated hypotheses because `test` can perform basic algebra. Here is an absurd hypothesis:

```
. test 2*(weight+weightsq)=-3*(foreign-(weight-weightsq))
 ( 1)  - weight + 5*weightsq + 3*foreign = 0
       F(  1,    70) =    4.31
            Prob > F =    0.0416
```

`test` simplified the algebra of our hypothesis and then presented the test results. We can also use `test`'s `accumulate` option to combine this test with another test:

```
. test foreign+weight=0, accum
 ( 1)  - weight + 5*weightsq + 3*foreign = 0
 ( 2)  weight + foreign = 0
       F(  2,    70) =    9.12
            Prob > F =    0.0003
```

There are limitations. `test` can test only linear hypotheses. If we attempt to test a nonlinear hypothesis, `test` will tell us that it is not possible:

```
. test weight/foreign=0
not possible with test
r(131);
```

Testing nonlinear hypotheses is discussed in [U] **20.13.4 Nonlinear Wald tests** below. ◁

## 20.13.2 Using test

`test` bases its results on the estimated variance–covariance matrix of the estimators (that is, it performs a Wald test), so it can be used after any estimation command. For maximum likelihood estimation, `test`'s results for a single variable are generally equivalent to the asymptotic $z$ statistic presented in the coefficient table for that variable because `test` bases its results on the information matrix.

### ▷ Example 13

Let's examine the repair records of the cars in our automobile data as rated by *Consumer Reports*:

```
. tabulate rep78 foreign
```

| Repair record 1978 | Car origin: Domestic | Foreign | Total |
|---|---|---|---|
| Poor | 2 | 0 | 2 |
| Fair | 8 | 0 | 8 |
| Average | 27 | 3 | 30 |
| Good | 9 | 9 | 18 |
| Excellent | 2 | 9 | 11 |
| Total | 48 | 21 | 69 |

The values are coded 1–5, corresponding to Poor, Fair, Average, Good, and Excellent. We will fit this variable by using a maximum-likelihood ordered logit model (the `nolog` option suppresses the iteration log, saving some space):

```
. ologit rep78 price foreign weight weightsq displ, nolog
Ordered logistic regression                       Number of obs =     69
                                                  LR chi2(5)    =  33.12
                                                  Prob > chi2   = 0.0000
Log likelihood = -77.133082                       Pseudo R2     = 0.1767
```

| rep78 | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| price | -.000034 | .0001188 | -0.29 | 0.775 | -.0002669 | .000199 |
| foreign | 2.685647 | .9320404 | 2.88 | 0.004 | .8588817 | 4.512413 |
| weight | -.0037447 | .0025609 | -1.46 | 0.144 | -.0087639 | .0012745 |
| weightsq | 7.87e-07 | 4.50e-07 | 1.75 | 0.080 | -9.43e-08 | 1.67e-06 |
| displacement | -.0108919 | .0076805 | -1.42 | 0.156 | -.0259455 | .0041617 |
| /cut1 | -9.417196 | 4.298202 | | | -17.84152 | -.992874 |
| /cut2 | -7.581864 | 4.234091 | | | -15.88053 | .7168028 |
| /cut3 | -4.82209 | 4.14768 | | | -12.95139 | 3.307214 |
| /cut4 | -2.793441 | 4.156221 | | | -10.93948 | 5.352602 |

We now wonder whether all our variables other than `foreign` are jointly significant. We test the hypothesis just as we would after linear regression:

```
. test weight weightsq displ price
 ( 1)  [rep78]weight = 0
 ( 2)  [rep78]weightsq = 0
 ( 3)  [rep78]displacement = 0
 ( 4)  [rep78]price = 0
           chi2(  4) =    3.63
         Prob > chi2 =    0.4590
```

You will have to decide whether you want to perform tests on the basis of the information matrix instead of constraining the equation, reestimating it, and then calculating the likelihood-ratio test. To compare this with the results performed by a likelihood-ratio test, see **[U] 20.13.3 Likelihood-ratio tests** below. Results will differ little.

◁

## 20.13.3 Likelihood-ratio tests

After maximum likelihood estimation, you can obtain likelihood-ratio tests by fitting both the unconstrained and the constrained models, storing the results using `estimates store`, and then running `lrtest`. See [R] **lrtest** for the full details.

### ▷ Example 14

In **[U] 20.13.2 Using test** above, we fit an ordered logit on `rep78` and then tested the significance of all the explanatory variables except `foreign`.

To obtain the likelihood-ratio test, sometime after fitting the full model, we type `estimates store` *full_model_name*, where *full_model_name* is just a label that we assign to these results.

```
. ologit rep78 price foreign weight weightsq displ
  (output omitted)
. estimates store myfullmodel
```

This command saves the current model results with the name `myfullmodel`.

Next, we fit the constrained model. After that, typing '`lrtest myfullmodel .`' compares the current model with the model we saved:

```
. ologit rep78 foreign
Iteration 0:  Log likelihood = -93.692061
Iteration 1:  Log likelihood = -79.696089
Iteration 2:  Log likelihood = -79.034005
Iteration 3:  Log likelihood = -79.029244
Iteration 4:  Log likelihood = -79.029243
Ordered logistic regression                       Number of obs =     69
                                                  LR chi2(1)    =  29.33
                                                  Prob > chi2   = 0.0000
Log likelihood = -79.029243                       Pseudo R2     = 0.1565
```

| rep78 | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| foreign | 2.98155 | .6203644 | 4.81 | 0.000 | 1.765658 | 4.197442 |
| /cut1 | -3.158382 | .7224269 | | | -4.574313 | -1.742452 |
| /cut2 | -1.362642 | .3557343 | | | -2.059868 | -.6654154 |
| /cut3 | 1.232161 | .3431227 | | | .5596532 | 1.90467 |
| /cut4 | 3.246209 | .5556657 | | | 2.157124 | 4.335293 |

```
. lrtest myfullmodel .
Likelihood-ratio test
Assumption: . nested within myfullmodel
 LR chi2(4) =   3.79
Prob > chi2 = 0.4348
```

When we tested the same constraint with `test` (which performed a Wald test), we obtained a $\chi^2$ of 3.63 and a significance level of 0.4590. We used `.` (the dot) to specify the results in active memory, although we could have stored them with `estimates store` and referred to them by name instead. Also, the order in which you specify the two models to `lrtest` doesn't matter; `lrtest` is smart enough to know the full model from the constrained model.

◁

Two other postestimation commands work in the same way as `lrtest`, meaning that they accept names of stored estimation results as their input: `hausman` for performing Hausman specification tests and `suest` for seemingly unrelated estimation. We do not cover these commands here; see [R] **hausman** and [R] **suest** for more details.

### 20.13.4 Nonlinear Wald tests

`testnl` can be used to test nonlinear hypotheses about the parameters of the active estimation results. `testnl`, like `test`, bases its results on the variance–covariance matrix of the estimators (that is, it performs a Wald test), so it can be used after any estimation command; see [R] **testnl**.

▷ Example 15

We fit the model

```
. regress price mpg weight foreign
  (output omitted)
```

and then type

```
. testnl (38*_b[mpg]^2 = _b[foreign]) (_b[mpg]/_b[weight]=4)
  (1)  38*_b[mpg]^2 = _b[foreign]
  (2)  _b[mpg]/_b[weight] = 4
               chi2(2) =        0.04
          Prob > chi2 =        0.9806
```

We performed this test on linear regression estimates, but tests of this type could be performed after any estimation command.

◁

A concept of a $p$-value is fundamental to classical hypothesis testing; see Wasserstein and Lazar (2016) for a useful discussion about its interpretation and use in practice. Also see [U] **27.34 Bayesian analysis** for an alternative to classical hypothesis testing.

## 20.14 Obtaining linear combinations of coefficients

`lincom` computes point estimates, standard errors, $t$ or $z$ statistics, $p$-values, and confidence intervals for a linear combination of coefficients after any estimation command. Results can optionally be displayed as odds ratios, incidence-rate ratios, or relative-risk ratios.

### ▷ Example 16

We fit a linear regression:

```
. use https://www.stata-press.com/data/r18/regress, clear
. regress y x1 x2 x3
```

| Source | SS | df | MS | Number of obs | = | 148 |
|---|---|---|---|---|---|---|
| | | | | F(3, 144) | = | 96.12 |
| Model | 3259.3561 | 3 | 1086.45203 | Prob > F | = | 0.0000 |
| Residual | 1627.56282 | 144 | 11.3025196 | R-squared | = | 0.6670 |
| | | | | Adj R-squared | = | 0.6600 |
| Total | 4886.91892 | 147 | 33.2443464 | Root MSE | = | 3.3619 |

| y | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| x1 | 1.457113 | 1.07461 | 1.36 | 0.177 | -.666934 | 3.581161 |
| x2 | 2.221682 | .8610358 | 2.58 | 0.011 | .5197797 | 3.923583 |
| x3 | -.006139 | .0005543 | -11.08 | 0.000 | -.0072345 | -.0050435 |
| _cons | 36.10135 | 4.382693 | 8.24 | 0.000 | 27.43863 | 44.76407 |

Suppose that we want to see the difference of the coefficients of `x2` and `x1`. We type

```
. lincom x2 - x1
 ( 1)  - x1 + x2 = 0
```

| y | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| (1) | .7645682 | .9950282 | 0.77 | 0.444 | -1.20218 | 2.731316 |

◁

`lincom` is handy for computing the odds ratio of one covariate group relative to another.

▷ Example 17

We estimate the parameters of a logistic model of low birthweight:

```
. use https://www.stata-press.com/data/r18/lbw3
(Hosmer & Lemeshow data)
. logit low age lwd i.race smoke ptd ht ui
Iteration 0:  Log likelihood =   -117.336
Iteration 1:  Log likelihood =   -99.3982
Iteration 2:  Log likelihood = -98.780418
Iteration 3:  Log likelihood = -98.777998
Iteration 4:  Log likelihood = -98.777998
Logistic regression                                     Number of obs =    189
                                                        LR chi2(8)    =  37.12
                                                        Prob > chi2   = 0.0000
Log likelihood = -98.777998                             Pseudo R2     = 0.1582
```

| low | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| age | -.0464796 | .0373888 | -1.24 | 0.214 | -.1197603 | .0268011 |
| lwd | .8420615 | .4055338 | 2.08 | 0.038 | .0472299 | 1.636893 |
| race | | | | | | |
| Black | 1.073456 | .5150753 | 2.08 | 0.037 | .0639273 | 2.082985 |
| Other | .815367 | .4452979 | 1.83 | 0.067 | -.0574008 | 1.688135 |
| smoke | .8071996 | .404446 | 2.00 | 0.046 | .0145001 | 1.599899 |
| ptd | 1.281678 | .4621157 | 2.77 | 0.006 | .3759478 | 2.187408 |
| ht | 1.435227 | .6482699 | 2.21 | 0.027 | .1646414 | 2.705813 |
| ui | .6576256 | .4666192 | 1.41 | 0.159 | -.2569313 | 1.572182 |
| _cons | -1.216781 | .9556797 | -1.27 | 0.203 | -3.089878 | .656317 |

Level 1 of `race` designates white, level 2 designates black, and level 3 designates other.

If we want to obtain the odds ratio for black smokers relative to white nonsmokers (the reference group), we type

```
. lincom 2.race + smoke, or
 ( 1)  [low]2.race + [low]smoke = 0
```

| low | Odds ratio | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| (1) | 6.557805 | 4.744692 | 2.60 | 0.009 | 1.588176 | 27.07811 |

`lincom` computed $\exp(\beta_{2.\text{race}} + \beta_{\text{smoke}}) = 6.56$.

◁

## 20.15 Obtaining nonlinear combinations of coefficients

`lincom` is limited to estimating linear combinations of coefficients, for example, `2.race + smoke`, or exponentiated linear combinations, as in the above. For general nonlinear combinations, use `nlcom`.

## ▷ Example 18

Continuing our previous example, suppose that we want the ratio of the coefficients (and standard errors, Wald test, confidence interval, etc.) of blacks and races other than white and black:

```
. nlcom _b[2.race]/_b[3.race]
       _nl_1: _b[2.race]/_b[3.race]
```

| low | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| _nl_1 | 1.316531 | .7359262 | 1.79 | 0.074 | -.1258574 | 2.75892 |

The Wald test given is that of the null hypothesis that the nonlinear combination is 0 versus the two-sided alternative—this is probably not informative for a ratio. If we would instead like to test whether this ratio is 1, we can rerun `nlcom`, this time subtracting 1 from our ratio estimate.

```
. nlcom _b[2.race]/_b[3.race] - 1
       _nl_1: _b[2.race]/_b[3.race] - 1
```

| low | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| _nl_1 | .3165314 | .7359262 | 0.43 | 0.667 | -1.125857 | 1.75892 |

We can interpret this as not much evidence that the ratio minus 1 is different from 0, meaning that we cannot reject the null hypothesis that the ratio equals 1.

When using `nlcom`, we needed to refer to the model coefficients by their "proper" names, for example, `_b[2.race]`, and not by the shorthand `2.race`, such as when using `lincom`. If we had typed

```
. nlcom 2.race/3.race
```

Stata would have reported an error.

If you have difficulty determining what to type for a coefficient when using `lincom` or `nlcom`, replay your results by using the `coeflegend` option. Here are the results for our current estimates:

```
. logit, coeflegend
Logistic regression                                     Number of obs =    189
                                                        LR chi2(8)    =  37.12
                                                        Prob > chi2   = 0.0000
Log likelihood = -98.777998                             Pseudo R2     = 0.1582
```

| low | Coefficient | Legend |
|---|---|---|
| age | -.0464796 | _b[age] |
| lwd | .8420615 | _b[lwd] |
| race | | |
| Black | 1.073456 | _b[2.race] |
| Other | .815367 | _b[3.race] |
| smoke | .8071996 | _b[smoke] |
| ptd | 1.281678 | _b[ptd] |
| ht | 1.435227 | _b[ht] |
| ui | .6576256 | _b[ui] |
| _cons | -1.216781 | _b[_cons] |

◁

## 20.16 Obtaining marginal means, adjusted predictions, and predictive margins

`predict` uses the current estimation results (the coefficients and the VCE) to estimate the value of statistics for observations in the data. `lincom` and `nlcom` use the current estimation results to estimate a specific linear or nonlinear expression of the coefficients. The `margins` command combines aspects of both and estimates margins of responses.

`margins` answers the question "What does my model have to say about such-and-such", where such-and-such might be

- my estimation sample or another sample
- a sample with the values of some covariates fixed
- a sample evaluated at each level of a treatment
- a population represented by a complex survey sample
- someone who looks like the fifth person in my sample
- someone who looks like the mean of the covariates in my sample
- someone who looks like the median of the covariates in my sample
- someone who looks like the 25th percentile of the covariates in my sample
- someone who looks like some other function of the covariates in my sample
- a standardized population
- a balanced experimental design
- any combination of the above
- any comparison of the above

`margins` answers these questions either conditionally on fixed values of all covariates or averaged over the observations in a sample. It answers these questions about almost any predictions or any other response that you can calculate as a function of your estimated parameters—linear responses, probabilities, hazards, survival times, odds ratios, risk differences, etc. You can even make multiple predictions at the same time when appropriate. For example, you may want the predicted probabilities and the linear prediction after `logit`.

`margins` answers these questions in terms of the response given covariate levels, or in terms of the change in the response for a change in levels (also known as marginal effects). It answers these questions providing standard errors, test statistics, and confidence intervals; and those statistics can take the covariates as given or adjust for sampling, also known as predictive margins and survey statistics.

A margin is a statistic based on a response for a fitted model calculated over a dataset in which some of or all the covariates are fixed at values different from what they really are.

Margins go by different names in different fields, and they can estimate many interesting statistics related to a fitted model. We discuss some common uses below; see [R] **margins** for more applications.

### 20.16.1 Obtaining estimated marginal means

A classic application of margins is to estimate the expected marginal means from a linear estimator as though the design for the covariates were balanced—assuming an equal number of observations for each unique combination of levels for the factor-variable covariates. These means have a long history in the study of ANOVA and MANOVA but are of limited use with nonexperimental data. For a

discussion, see *Obtaining margins as though the data were balanced* in [R] **margins** and example 4 in [R] **anova**.

Estimated marginal means are also called least-squares means.

Consider an analysis of variance of the change in systolic blood pressure as determined by one of four drug treatments and adjusting for the patient's disease (Afifi and Azen 1979).

```
. use https://www.stata-press.com/data/r18/systolic
(Systolic blood pressure data)
. tabulate drug disease
```

| Drug used | Patient's disease 1 | 2 | 3 | Total |
|---|---|---|---|---|
| 1 | 6 | 4 | 5 | 15 |
| 2 | 5 | 4 | 6 | 15 |
| 3 | 3 | 5 | 4 | 12 |
| 4 | 5 | 6 | 5 | 16 |
| Total | 19 | 19 | 20 | 58 |

```
. anova systolic drug##disease
```

| | | | | | |
|---|---|---|---|---|---|
| Number of obs = | 58 | R-squared | = | 0.4560 | |
| Root MSE = | 10.5096 | Adj R-squared | = | 0.3259 | |

| Source | Partial SS | df | MS | F | Prob>F |
|---|---|---|---|---|---|
| Model | 4259.3385 | 11 | 387.21259 | 3.51 | 0.0013 |
| drug | 2997.4719 | 3 | 999.15729 | 9.05 | 0.0001 |
| disease | 415.87305 | 2 | 207.93652 | 1.88 | 0.1637 |
| drug#disease | 707.26626 | 6 | 117.87771 | 1.07 | 0.3958 |
| Residual | 5080.8167 | 46 | 110.45254 | | |
| Total | 9340.1552 | 57 | 163.86237 | | |

Despite having randomized on drug, we see in the tabulation that our data are not balanced—for example, 12 patients were administered drug 3, whereas 16 were administered drug 4. The diseases are also not balanced across drugs. To estimate the marginal mean for each level of drug while treating the design as though it were balanced, we type

```
. margins drug, asbalanced
Adjusted predictions                                    Number of obs = 58
Expression: Linear prediction, predict()
At: drug      (asbalanced)
    disease   (asbalanced)
```

| | Margin | Delta-method std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| drug | | | | | | |
| 1 | 25.99444 | 2.751008 | 9.45 | 0.000 | 20.45695 | 31.53194 |
| 2 | 26.55556 | 2.751008 | 9.65 | 0.000 | 21.01806 | 32.09305 |
| 3 | 9.744444 | 3.100558 | 3.14 | 0.003 | 3.503344 | 15.98554 |
| 4 | 13.54444 | 2.637123 | 5.14 | 0.000 | 8.236191 | 18.8527 |

Assuming everyone in the sample were treated with drug 4 and that the diseases were equally distributed across the drug treatments, the expected mean change in pressure resulting from treatment with drug 4 is 13.54. Because we are treating the data as balanced, we could also say that 13.54 is

the expected mean change resulting from drug 4 for any sample where an equal number of patients has each of the three diseases.

If we want an estimate of the mean that uses the distribution of diseases observed in the sample, we would remove the `asbalanced` option:

```
. margins drug
Predictive margins                                      Number of obs = 58
Expression: Linear prediction, predict()
```

| | Margin | Delta-method std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| drug | | | | | | |
| 1 | 25.89799 | 2.750533 | 9.42 | 0.000 | 20.36145 | 31.43452 |
| 2 | 26.41092 | 2.742762 | 9.63 | 0.000 | 20.89003 | 31.93181 |
| 3 | 9.722989 | 3.099185 | 3.14 | 0.003 | 3.484652 | 15.96132 |
| 4 | 13.55575 | 2.640602 | 5.13 | 0.000 | 8.24049 | 18.871 |

We can now say that a pressure change of 13.56 is expected if everyone in the sample is given drug 4 and the distribution of diseases is as observed in the sample.

The second set of margins are not usually called estimated marginal means because they do not impose a balanced design when estimating the mean. They are adjusted predictions that just happen to be means because the response is linear.

Neither of these values is the average pressure change for those taking drug 4 in our sample because `margins` treats everyone in the sample as having taken drug 4. Treating everyone as though they have taken each drug is what makes the means comparable. We are essentially standardizing on the values of all the other covariates in our model (in this example, just `disease`).

To obtain the observed mean for those taking drug 4, we must tell `margins` to treat drug 4 as its sample, which we do with the `over()` option.

```
. summarize systolic if drug==4
```

| Variable | Obs | Mean | Std. dev. | Min | Max |
|---|---|---|---|---|---|
| systolic | 16 | 13.5 | 9.323805 | -5 | 27 |

```
. margins, over(drug)
Predictive margins                                      Number of obs = 58
Expression: Linear prediction, predict()
Over:       drug
```

| | Margin | Delta-method std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| drug | | | | | | |
| 1 | 26.06667 | 2.713577 | 9.61 | 0.000 | 20.60452 | 31.52881 |
| 2 | 25.53333 | 2.713577 | 9.41 | 0.000 | 20.07119 | 30.99548 |
| 3 | 8.75 | 3.033872 | 2.88 | 0.006 | 2.643133 | 14.85687 |
| 4 | 13.5 | 2.62741 | 5.14 | 0.000 | 8.211298 | 18.7887 |

The margin in the last line of the table matches the mean from `summarize`.

For many questions, we prefer one of the first two estimates of `margins` to the last one. If we compare drugs 3 and 4 from the last results, the 8.75 and 13.5 include both the effect from the drug and the differing distribution of diseases among patients taking drug 3 and drug 4 in our sample.

Our first set of margins, those from `margins drug, asbalanced`, assumed that for both drug 3 and drug 4, we had an equal number of patients with each disease. Our second set of margins, those from `margins drug`, assumed that for both drug 3 and drug 4, we wanted the observed distribution of patients from the whole sample. By assuming a common distribution of diseases across the drugs, our first two sets of margins remove the effect of disease when we compare across drugs.

### 20.16.2 Obtaining adjusted predictions

We will use the term adjusted predictions to refer to margins that are evaluated at fixed values for all covariates. The `margins` command has a great deal of flexibility in letting you choose what those fixed values are. Consider a model of high blood pressure as a function of sex, age group, and body mass index (BMI, a common measure of weight relative to height; variable `bmi`). We will allow the effect of age to differ for males and females by interacting the age group and sex variables. We will also allow the effect of BMI to differ across all combinations of age group and sex by specifying a full factorial model.

```
. use https://www.stata-press.com/data/r18/nhanes2
. logistic highbp sex##agegrp##c.bmi
Logistic regression                                     Number of obs =  10,351
                                                        LR chi2(23)   = 2521.83
                                                        Prob > chi2   =  0.0000
Log likelihood = -5789.851                              Pseudo R2     =  0.1788
```

| highbp | Odds ratio | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| sex | | | | | | |
| Female | .4012124 | .2695666 | -1.36 | 0.174 | .107515 | 1.497199 |
| agegrp | | | | | | |
| 30-39 | .8124869 | .6162489 | -0.27 | 0.784 | .1837399 | 3.592768 |
| 40-49 | 1.346976 | 1.101181 | 0.36 | 0.716 | .2713222 | 6.687051 |
| 50-59 | 5.415758 | 4.254136 | 2.15 | 0.032 | 1.161532 | 25.2515 |
| 60-69 | 16.31623 | 10.09529 | 4.51 | 0.000 | 4.852423 | 54.86321 |
| 70+ | 161.2491 | 130.7332 | 6.27 | 0.000 | 32.9142 | 789.9717 |
| sex#agegrp | | | | | | |
| Female#30-39 | 1.441256 | 1.44721 | 0.36 | 0.716 | .2013834 | 10.31475 |
| Female#40-49 | 6.29497 | 6.575021 | 1.76 | 0.078 | .8126879 | 48.75998 |
| Female#50-59 | 4.377185 | 4.43183 | 1.46 | 0.145 | .6016818 | 31.84366 |
| Female#60-69 | 1.790026 | 1.502447 | 0.69 | 0.488 | .3454684 | 9.27492 |
| Female#70+ | .1958758 | .2165763 | -1.47 | 0.140 | .0224297 | 1.710562 |
| bmi | 1.18539 | .0221872 | 9.09 | 0.000 | 1.142692 | 1.229684 |
| sex#c.bmi | | | | | | |
| Female | .9809543 | .0250973 | -0.75 | 0.452 | .9329775 | 1.031398 |
| agegrp#c.bmi | | | | | | |
| 30-39 | 1.021812 | .0299468 | 0.74 | 0.462 | .9647712 | 1.082225 |
| 40-49 | 1.00982 | .0315328 | 0.31 | 0.754 | .9498702 | 1.073554 |
| 50-59 | .979291 | .0298836 | -0.69 | 0.493 | .9224373 | 1.039649 |
| 60-69 | .9413883 | .0228342 | -2.49 | 0.013 | .8976813 | .9872234 |
| 70+ | .8738056 | .0278416 | -4.23 | 0.000 | .8209061 | .930114 |
| sex#agegrp#c.bmi | | | | | | |
| Female#30-39 | 1.000676 | .0377954 | 0.02 | 0.986 | .9292736 | 1.077564 |
| Female#40-49 | .9702656 | .0382854 | -0.76 | 0.444 | .8980559 | 1.048281 |
| Female#50-59 | .9852929 | .0380345 | -0.38 | 0.701 | .9134969 | 1.062732 |
| Female#60-69 | 1.028896 | .0330473 | 0.89 | 0.375 | .9661212 | 1.09575 |
| Female#70+ | 1.12236 | .0480541 | 2.70 | 0.007 | 1.032019 | 1.220609 |
| _cons | .0052191 | .0024787 | -11.07 | 0.000 | .0020575 | .0132388 |

Note: **_cons** estimates baseline odds.

We can evaluate the probability of having high blood pressure for each age group while holding the proportion of males and females and the value of bmi to its average by specifying the covariate agegrp to margins and including the option atmeans:

```
. margins agegrp, atmeans
Adjusted predictions                                    Number of obs = 10,351
Model VCE: OIM

Expression: Pr(highbp), predict()
At: 1.sex    = .4748333 (mean)
    2.sex    = .5251667 (mean)
    1.agegrp = .2241329 (mean)
    2.agegrp = .1566998 (mean)
    3.agegrp = .1228867 (mean)
    4.agegrp = .1247222 (mean)
    5.agegrp = .2763018 (mean)
    6.agegrp = .0952565 (mean)
    bmi      =  25.5376 (mean)
```

| | Margin | Delta-method std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| agegrp | | | | | | |
| 20-29 | .1611491 | .0091135 | 17.68 | 0.000 | .1432869 | .1790113 |
| 30-39 | .2487466 | .0121649 | 20.45 | 0.000 | .2249038 | .2725893 |
| 40-49 | .3679695 | .0144456 | 25.47 | 0.000 | .3396567 | .3962823 |
| 50-59 | .5204507 | .0146489 | 35.53 | 0.000 | .4917394 | .549162 |
| 60-69 | .5714605 | .0095866 | 59.61 | 0.000 | .5526711 | .5902499 |
| 70+ | .6637982 | .0154566 | 42.95 | 0.000 | .6335038 | .6940927 |

The header of the table showed us the mean values of each covariate. These are the values at which the probabilities were evaluated. The mean values for the levels of agegrp appear in the header even though they were not used. agegrp assumed the values 1, 2, 3, 4, 5, and 6, as shown in the table. The means of the levels of agegrp are shown because we might have asked for more margins in the table, for example, margins sex agegrp.

The modeled probability is just below 25% for those under 40 years of age, and it then increases fairly rapidly to 52% in the 50–59 age group. Above age 59, the probability remains under 67%. It is often easier for nonstatisticians to interpret the statistics computed by margins than it is to interpret the coefficients of a fitted model.

## 20.16.3 Obtaining predictive margins

Rather than evaluate the probability of having high blood pressure at one fixed point (the means), as we did above, we can evaluate the probability at the covariate values for each observation in our data and average those probabilities. Here is the modeled probability averaged over our sample:

```
. margins
Predictive margins                                      Number of obs = 10,351
Model VCE: OIM

Expression: Pr(highbp), predict()
```

| | Margin | Delta-method std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| _cons | .4227611 | .0042939 | 98.46 | 0.000 | .4143451 | .4311771 |

If we fix the level of agegrp to 1, compute the probability for each observation, and then average those probabilities, the result is the predictive margin for level 1 of agegrp. margins, by default, computes these margins for each level of each variable specified on the command line. Let's compute the predictive margins for agegrp:

```
. margins agegrp
Predictive margins                                      Number of obs = 10,351
Model VCE: OIM
Expression: Pr(highbp), predict()
```

| | Margin | Delta-method std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| agegrp | | | | | | |
| 20-29 | .2030932 | .0087166 | 23.30 | 0.000 | .1860089 | .2201774 |
| 30-39 | .2829091 | .010318 | 27.42 | 0.000 | .2626862 | .3031319 |
| 40-49 | .3769536 | .0128744 | 29.28 | 0.000 | .3517202 | .4021871 |
| 50-59 | .5153439 | .0136201 | 37.84 | 0.000 | .4886491 | .5420387 |
| 60-69 | .5641065 | .009136 | 61.75 | 0.000 | .5462003 | .5820127 |
| 70+ | .6535679 | .0151371 | 43.18 | 0.000 | .6238997 | .683236 |

One way of looking at predictive margins is that they answer the question "What would the average response (probability) be in my sample if everyone were in one age group?" Another way of looking at predictive margins is that they standardize the effect of being in an age group with the distribution of other covariate values in our sample. The margins above are comparable because only the level of agegrp is changing across the margins. They represent our sample because all the other covariates take on their values in the sample when the margins are evaluated.

The predictive margins in this table differ from the adjusted predictions we estimated in [U] **20.16.2 Obtaining adjusted predictions** because the probability is a nonlinear function of the coefficients in a logistic model; see *Example 3: Average response versus response at average* in [R] **margins** for details.

Our analysis so far has been a bit naïve. The dataset we are using is from the Second National Health and Nutrition Examination Survey (NHANES II). It has weights to make it representative of the population from which it was drawn as well as other survey characteristics—strata and primary sampling units. The data have already been svyset; see [SVY] **svyset**. We should take note of these characteristics and use the svy prefix when fitting our model.

```
. svy: logistic highbp sex##agegrp##c.bmi
(output omitted)
```

If we were to repeat the command margins agegrp, we would see that our point estimates differ only a little, but our standard errors are generally larger.

We are not restricted to margining over a single factor variable. Let's see if the pattern of high blood pressure over age groups differs for men and women. We do that by specifying the interaction of sex and agegrp to margins. We add the vce(unconditional) option to accommodate the survey design.

```
. margins sex#agegrp, vce(unconditional)
Predictive margins
Number of strata = 31                          Number of obs    =      10,351
Number of PSUs   = 62                          Population size = 117,157,513
                                               Design df        =          31
Expression: Pr(highbp), predict()
```

| | Margin | Linearized std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| sex#agegrp | | | | | | |
| Male#20-29 | .2931664 | .0204899 | 14.31 | 0.000 | .251377 | .3349557 |
| Male#30-39 | .3664032 | .0241677 | 15.16 | 0.000 | .3171128 | .4156936 |
| Male#40-49 | .3945619 | .0240343 | 16.42 | 0.000 | .3455435 | .4435802 |
| Male#50-59 | .5376423 | .0295377 | 18.20 | 0.000 | .4773997 | .5978849 |
| Male#60-69 | .5780997 | .0224681 | 25.73 | 0.000 | .5322756 | .6239237 |
| Male#70+ | .6507023 | .0209322 | 31.09 | 0.000 | .6080109 | .6933938 |
| Female#20-29 | .1069761 | .0135978 | 7.87 | 0.000 | .0792432 | .1347091 |
| Female#30-39 | .1898006 | .0143975 | 13.18 | 0.000 | .1604367 | .2191646 |
| Female#40-49 | .3250246 | .0236775 | 13.73 | 0.000 | .276734 | .3733152 |
| Female#50-59 | .4855339 | .03364 | 14.43 | 0.000 | .4169247 | .5541431 |
| Female#60-69 | .5441773 | .0186243 | 29.22 | 0.000 | .5061928 | .5821618 |
| Female#70+ | .6195342 | .0275568 | 22.48 | 0.000 | .5633317 | .6757367 |

Each line in the table corresponds to holding both `sex` and `agegrp` to fixed values while using the observed level of `bmi` to evaluate the probability and then averaging over the observations in the sample. To calculate the results in the first line of the table, `margins` fixed `sex` $= 1$ and `agegrp` $= 1$, evaluated the probability for each observation, and then averaged the probabilities. All of these margins reflect the observed distribution of `bmi` in the sample.

The first six lines represent males, and the second six lines represent females. Comparing males with females for the same age groups, males are almost three times as likely to have high blood pressure in the first age group $(0.293/0.107 = 2.7)$; they are almost twice as likely in the second age group; and while the relative gap narrows, it is not until above age 70 that the probability for males drops below the probability for females.

Can the pattern of probabilities be affected by controlling one's `bmi`? Let's reevaluate the probabilities while holding `bmi` to two levels—20 (which is well within the normal range) and 30 (which is at the boundary between overweight and obese). We add the option `at(bmi=(20 30))` to set `bmi` first to 20 and then to 30.

```
. margins sex#agegrp, at(bmi=(20 30)) vce(unconditional)
Adjusted predictions
Number of strata = 31                          Number of obs    =      10,351
Number of PSUs   = 62                          Population size = 117,157,513
                                               Design df        =          31
Expression: Pr(highbp), predict()
1._at: bmi = 20
2._at: bmi = 30
```

| | Margin | Linearized std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| _at#sex# agegrp | | | | | | |
| 1#Male#20-29 | .1392353 | .0217328 | 6.41 | 0.000 | .094911 | .1835596 |
| 1#Male#30-39 | .1714727 | .0241469 | 7.10 | 0.000 | .1222249 | .2207205 |
| 1#Male#40-49 | .1914061 | .0366133 | 5.23 | 0.000 | .1167329 | .2660794 |
| 1#Male#50-59 | .3380778 | .0380474 | 8.89 | 0.000 | .2604797 | .4156759 |
| 1#Male#60-69 | .4311378 | .0371582 | 11.60 | 0.000 | .3553532 | .5069225 |
| 1#Male#70+ | .6131166 | .0521657 | 11.75 | 0.000 | .506724 | .7195092 |
| 1 # Female # 20-29 | .0439911 | .0061833 | 7.11 | 0.000 | .0313802 | .056602 |
| 1 # Female # 30-39 | .075806 | .0134771 | 5.62 | 0.000 | .0483193 | .1032926 |
| 1 # Female # 40-49 | .1941274 | .0231872 | 8.37 | 0.000 | .1468367 | .2414181 |
| 1 # Female # 50-59 | .3493224 | .0405082 | 8.62 | 0.000 | .2667055 | .4319394 |
| 1 # Female # 60-69 | .3897998 | .0226443 | 17.21 | 0.000 | .3436165 | .4359831 |
| 1#Female#70+ | .4599175 | .0338926 | 13.57 | 0.000 | .3907931 | .5290419 |
| 2#Male#20-29 | .4506376 | .0370654 | 12.16 | 0.000 | .3750422 | .526233 |
| 2#Male#30-39 | .569466 | .04663 | 12.21 | 0.000 | .4743635 | .6645686 |
| 2#Male#40-49 | .6042078 | .039777 | 15.19 | 0.000 | .5230821 | .6853334 |
| 2#Male#50-59 | .7268547 | .0339618 | 21.40 | 0.000 | .657589 | .7961203 |
| 2#Male#60-69 | .7131811 | .0271145 | 26.30 | 0.000 | .6578807 | .7684816 |
| 2#Male#70+ | .6843337 | .0357432 | 19.15 | 0.000 | .611435 | .7572323 |
| 2 # Female # 20-29 | .1638185 | .024609 | 6.66 | 0.000 | .1136282 | .2140088 |
| 2 # Female # 30-39 | .3038899 | .0271211 | 11.20 | 0.000 | .2485761 | .3592037 |
| 2 # Female # 40-49 | .4523337 | .0364949 | 12.39 | 0.000 | .3779019 | .5267655 |
| 2 # Female # 50-59 | .6132219 | .0376898 | 16.27 | 0.000 | .536353 | .6900908 |
| 2 # Female # 60-69 | .68786 | .0274712 | 25.04 | 0.000 | .631832 | .7438879 |
| 2#Female#70+ | .7643662 | .0343399 | 22.26 | 0.000 | .6943296 | .8344029 |

That is a lot of margins, but they are in sets of six age groups. The first six margins are men with a BMI of 20, the second six are women with a BMI of 20, the third six are men with a BMI of 30, and the last six are women with a BMI of 30. These margins tell a more complete story. The probability of high blood pressure is much lower for both men and women who maintain a BMI of 20. More interesting is that the relationship between men and women differs depending on BMI. While young men who maintain a BMI of 20 are still twice as likely as young women to have high blood pressure (0.139/0.044) and youngish men are over 50% more likely (0.171/0.076), the gap narrows substantially for men in the four older groups. The story is worse for those with a BMI of 30. Both men and women with a high BMI have a substantially increased risk of high blood pressure, with men ages 50–69 almost 10 percentage points higher than women. Before you dismiss these differences as caused by the usual attenuation of the logistic curve in the tails, recall that when we fit the model, we allowed the effect of `bmi` to be different for each combination of `sex` and `agegrp`.

You may have noticed that the header of the prior results says "Adjusted predictions" rather than "Predictive margins". That is because our model has only three covariates, and we have fixed the values of each. `margins` is no longer averaging over the data, but is instead evaluating the margins at fixed points that we have requested. It lets us know that by changing the header.

We could post the results of `margins` and form linear combinations or perform tests about any of the assertions above; see *Example 10: Testing margins—contrasts of margins* in [R] **margins**.

There is much more to know about margins and the `margins` command. See *Remarks and examples* in [R] **margins** for more details.

## 20.17 Obtaining conditional and average marginal effects

Marginal effects measure the change in a response given a change in a covariate, which is to say that marginal effects are derivatives. As used here, marginal effects can also be the discrete change in a response as an indicator goes from 0 to 1. Some authors reserve the term marginal effect for the continuous change and use the term partial effect for the discrete change. We will not make that distinction. Regardless, marginal effects are most often used to make it easier to interpret how changes in covariates affect a nonlinear response from a fitted model—a probability, a censored dependent variable, a survival time, a hazard, etc.

Marginal effects can either be evaluated at a specified point for all the covariates in our model (conditional marginal effects) or be evaluated at the observed values of the covariates in a dataset and then averaged (average marginal effects).

To Stata, marginal effects are just margins whose response happens to be the derivative of another response. Those interested in marginal effects will be interested in all or most of [R] **margins**.

### 20.17.1 Obtaining conditional marginal effects

We call a marginal effect conditional when we fix the values of all the covariates and then take the derivative of the response with respect to a covariate. The mean of all covariates is often used as the fixed point, and this is sometimes called the marginal effect at the means.

Consider a simple probit model of union membership for women as a function of having graduated from college (`collgrad`), living in the South (`south`), tenure on the job (`tenure`), and the interaction of `south` and `tenure`. We are interested in how being in the South affects union membership. We fit the model by using an extract from 1988 of the U.S. National Longitudinal Survey of Labor Market Experience (see [XT] **xt**).

```
. use https://www.stata-press.com/data/r18/nlsw88b, clear
(NLSW, 1988 extract)
. probit union i.collgrad i.south tenure south#c.tenure
Iteration 0:  Log likelihood = -1042.6816
Iteration 1:  Log likelihood = -997.71809
Iteration 2:  Log likelihood = -997.60984
Iteration 3:  Log likelihood = -997.60983
Probit regression                                     Number of obs =  1,868
                                                      LR chi2(4)    =  90.14
                                                      Prob > chi2   = 0.0000
Log likelihood = -997.60983                           Pseudo R2     = 0.0432
```

| union | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| collgrad | | | | | | |
| not grad | .2783278 | .0726167 | 3.83 | 0.000 | .1360018 | .4206539 |
| 1.south | -.2534964 | .1050552 | -2.41 | 0.016 | -.4594008 | -.0475921 |
| tenure | .0362944 | .0068205 | 5.32 | 0.000 | .0229264 | .0496624 |
| south# c.tenure | | | | | | |
| 1 | -.0239785 | .0119533 | -2.01 | 0.045 | -.0474065 | -.0005504 |
| _cons | -.8497418 | .0664524 | -12.79 | 0.000 | -.9799862 | -.7194974 |

Clearly, being located in the South decreases union membership. Using the `dydx()` and `atmeans` options of `margins`, we can ask how much it decreases membership by evaluating the marginal effect of being southern at the means of all covariates:

```
. margins, dydx(south) atmeans
Conditional marginal effects                          Number of obs = 1,868
Model VCE: OIM
Expression: Pr(union), predict()
dy/dx wrt:  1.south
At: 0.collgrad = .7521413 (mean)
    1.collgrad = .2478587 (mean)
    0.south    = .5744111 (mean)
    1.south    = .4255889 (mean)
    tenure     = 6.571065 (mean)
```

| | Delta-method dy/dx | std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| 1.south | -.1236055 | .019431 | -6.36 | 0.000 | -.1616896 | -.0855215 |

```
Note: dy/dx for factor levels is the discrete change from the base level.
```

At the means of all the covariates, southern women are 12 percentage points less likely to be members of a union. This marginal effect includes both the direct effect of `i.south` and the interaction `south#c.tenure`.

As `margins` reports below the table, this change in the response is for the discrete change of going from not southern (0) to southern (1).

The header of `margins` tells us where the marginal effect was estimated. This margin fixes `tenure` to be 6.6 years. There is nothing special about this point. We could also evaluate the marginal effect at the median of `tenure`:

```
. margins, dydx(south) atmeans at((medians) _continuous)
Conditional marginal effects                              Number of obs = 1,868
Model VCE: OIM

Expression: Pr(union), predict()
dy/dx wrt:  1.south
At: 0.collgrad = .7521413 (mean)
    1.collgrad = .2478587 (mean)
    0.south    = .5744111 (mean)
    1.south    = .4255889 (mean)
    tenure     = 4.666667 (median)
```

| | dy/dx | Delta-method std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| 1.south | -.1061338 | .0201722 | -5.26 | 0.000 | -.1456706 | -.066597 |

Note: dy/dx for factor levels is the discrete change from the base level.

With `tenure` at its median of 4.67, the marginal effect is about 2 percentage points less than it was at the mean of 6.6.

When examining conditional marginal effects, it is often useful to evaluate them at a range of values for the covariates. We can do that by asking both for values of the indicator covariate `collgrad` and for a range of values for `tenure`:

```
. margins collgrad, dydx(south) at(tenure=(0(5)25))
Conditional marginal effects                              Number of obs = 1,868
Model VCE: OIM

Expression: Pr(union), predict()
dy/dx wrt:  1.south
1._at: tenure =  0
2._at: tenure =  5
3._at: tenure = 10
4._at: tenure = 15
5._at: tenure = 20
6._at: tenure = 25
```

| | dy/dx | Delta-method std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| 0.south | (base outcome) | | | | | |
| 1.south _at#collgrad | | | | | | |
| 1#grad | -.0627725 | .0254161 | -2.47 | 0.014 | -.112587 | -.0129579 |
| 1#not grad | -.0791483 | .0321151 | -2.46 | 0.014 | -.1420928 | -.0162038 |
| 2#grad | -.1031957 | .0189184 | -5.45 | 0.000 | -.140275 | -.0661164 |
| 2#not grad | -.1256566 | .0232385 | -5.41 | 0.000 | -.1712031 | -.0801101 |
| 3#grad | -.1496772 | .022226 | -6.73 | 0.000 | -.1932392 | -.1061151 |
| 3#not grad | -.1760137 | .0266874 | -6.60 | 0.000 | -.2283202 | -.1237073 |
| 4#grad | -.2008801 | .036154 | -5.56 | 0.000 | -.2717407 | -.1300196 |
| 4#not grad | -.2282 | .0419237 | -5.44 | 0.000 | -.310369 | -.146031 |
| 5#grad | -.2549707 | .0546355 | -4.67 | 0.000 | -.3620543 | -.1478872 |
| 5#not grad | -.2799495 | .0613127 | -4.57 | 0.000 | -.4001201 | -.1597789 |
| 6#grad | -.3097656 | .0747494 | -4.14 | 0.000 | -.4562717 | -.1632594 |
| 6#not grad | -.3289702 | .0816342 | -4.03 | 0.000 | -.4889703 | -.1689701 |

Note: dy/dx for factor levels is the discrete change from the base level.

We now have a more complete picture of the effect that being in the South has on union participation. For those with no tenure and without a college degree (the first line in the table), being in the South

decreases union participation by only 6 percentage points. For those with 25 years of tenure and with a college degree (the last line in the table), being in the South decreases participation by almost 33 percentage points. We can read the effect for any combination of tenure and college graduation status from the other lines in the table.

### 20.17.2 Obtaining average marginal effects

To compute average marginal effects, the marginal effect is first computed for each observation in the dataset and then averaged. If the sample over which we compute the average marginal effect represents a population, then we have estimated the marginal effect for the population.

We continue with our example of labor union participation.

```
. use https://www.stata-press.com/data/r18/nlsw88b
(NLSW, 1988 extract)
. probit union i.collgrad i.south tenure south#c.tenure
  (output omitted)
```

To estimate the average marginal effect for each of our regressors, we type

```
. margins, dydx(*)
Average marginal effects                                  Number of obs = 1,868
Model VCE: OIM
Expression: Pr(union), predict()
dy/dx wrt:  1.collgrad 1.south tenure
```

| | dy/dx | Delta-method std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| collgrad | | | | | | |
| not grad | .0878847 | .0238065 | 3.69 | 0.000 | .0412248 | .1345447 |
| 1.south | -.126164 | .0191504 | -6.59 | 0.000 | -.1636981 | -.0886299 |
| tenure | .0083571 | .0016521 | 5.06 | 0.000 | .005119 | .0115951 |

```
Note: dy/dx for factor levels is the discrete change from the base level.
```

For this sample, the average marginal effect is very close to the marginal effect at the mean that we computed earlier. That is not always true; it depends on the distribution of the other covariates. The results also tell us that on average, for populations like the one from which our sample was drawn, union participation increases 0.8 percentage points for every year of tenure on the job. College graduates are, on average, 8.8 percentage points more likely to participate.

In the examples above, we treated the covariates in the sample as fixed and known. We could have accounted for the fact that this sample was drawn from a population and the covariates represent just one sample from that population. We do that by adding the `vce(robust)` or `vce(cluster` *clustvar*`)` option when fitting the model and the `vce(unconditional)` option when estimating the margins; see *Obtaining margins with survey data and representative samples* in [R] **margins**. It makes little difference in the examples above.

## 20.18 Obtaining pairwise comparisons

`pwcompare` performs pairwise comparisons across the levels of factor variables. `pwcompare` can compare estimated cell means, marginal means, intercepts, marginal intercepts, slopes, or marginal slopes—collectively called margins. `pwcompare` reports comparisons as contrasts (differences) of margins along with significance tests or confidence intervals for the contrasts. The tests and confidence intervals can be adjusted for multiple comparisons.

`pwcompare` is for use after an estimation command in which you have used factor variables in specifying the model. You could not use `pwcompare` after typing

```
. regress yield fertilizer1-fertilizer5
```

but you could use `pwcompare` after typing

```
. regress yield i.fertilizer
```

Below, we fit a linear regression of wheat yield on type of fertilizer, and then we compare the mean yields for each pair of fertilizers and obtain $p$-values and confidence intervals adjusted for multiple comparisons by using Tukey's honestly significant difference.

```
. use https://www.stata-press.com/data/r18/yield
(Artificial wheat yield dataset)
. regress yield i.fertilizer
```

| Source | SS | df | MS | Number of obs | = | 200 |
|---|---|---|---|---|---|---|
| | | | | F(4, 195) | = | 5.33 |
| Model | 1078.84207 | 4 | 269.710517 | Prob > F | = | 0.0004 |
| Residual | 9859.55334 | 195 | 50.561812 | R-squared | = | 0.0986 |
| | | | | Adj R-squared | = | 0.0801 |
| Total | 10938.3954 | 199 | 54.9668111 | Root MSE | = | 7.1107 |

| yield | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| fertilizer | | | | | | |
| 10-08-22 | 3.62272 | 1.589997 | 2.28 | 0.024 | .4869212 | 6.758518 |
| 16-04-08 | .4906299 | 1.589997 | 0.31 | 0.758 | -2.645169 | 3.626428 |
| 18-24-06 | 4.922803 | 1.589997 | 3.10 | 0.002 | 1.787005 | 8.058602 |
| 29-03-04 | -1.238328 | 1.589997 | -0.78 | 0.437 | -4.374127 | 1.89747 |
| _cons | 41.36243 | 1.124298 | 36.79 | 0.000 | 39.14509 | 43.57977 |

```
. pwcompare fertilizer, effects mcompare(tukey)
Pairwise comparisons of marginal linear predictions
Margins: asbalanced
```

| | Number of comparisons |
|---|---|
| fertilizer | 10 |

| | Contrast | Std. err. | Tukey t | Tukey P>\|t\| | Tukey [95% conf. | interval] |
|---|---|---|---|---|---|---|
| fertilizer | | | | | | |
| 10-08-22 vs 10-10-10 | 3.62272 | 1.589997 | 2.28 | 0.156 | -.7552913 | 8.000731 |
| 16-04-08 vs 10-10-10 | .4906299 | 1.589997 | 0.31 | 0.998 | -3.887381 | 4.868641 |
| *(output omitted)* | | | | | | |
| 29-03-04 vs 18-24-06 | -6.161132 | 1.589997 | -3.87 | 0.001 | -10.53914 | -1.78312 |

See [R] **pwcompare** and [R] **margins, pwcompare**.

## 20.19 Obtaining contrasts, tests of interactions, and main effects

`contrast` estimates and tests contrasts—comparisons of levels of factor variables. It also performs joint tests of these contrasts and can produce ANOVA-style tests of main effects, interaction effects, simple effects, and nested effects. It can be used after most estimation commands.

`contrast` provides a set of contrast operators such as `r.`, `ar.`, and `p.`. These operators are prefixed onto variable names—for example, `r.`*varname*—to specify the contrasts to be performed. The operators can be used with the `contrast` and `margins` commands.

Below, we fit a regression of cholesterol level on age group category.

```
. regress chol i.agegrp
```

The reported coefficients on `i.agegrp` will themselves be contrasts, namely, contrasts on the reference category. After estimation, if we wanted to compare the cell mean of each age group with that of the previous group, we would perform a reverse-adjacent contrast by typing

```
. contrast ar.agegrp
```

That is exactly what we will do:

```
. use https://www.stata-press.com/data/r18/cholesterol
(Artificial cholesterol data)
. regress chol i.agegrp
```

| Source | SS | df | MS | Number of obs | = | 75 |
|---|---|---|---|---|---|---|
| | | | | F(4, 70) | = | 35.02 |
| Model | 14943.3997 | 4 | 3735.84993 | Prob > F | = | 0.0000 |
| Residual | 7468.21971 | 70 | 106.688853 | R-squared | = | 0.6668 |
| | | | | Adj R-squared | = | 0.6477 |
| Total | 22411.6194 | 74 | 302.859722 | Root MSE | = | 10.329 |

| chol | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| agegrp | | | | | | |
| 20-29 | 8.203575 | 3.771628 | 2.18 | 0.033 | .6812991 | 15.72585 |
| 30-39 | 21.54105 | 3.771628 | 5.71 | 0.000 | 14.01878 | 29.06333 |
| 40-59 | 30.15067 | 3.771628 | 7.99 | 0.000 | 22.6284 | 37.67295 |
| 60-79 | 38.76221 | 3.771628 | 10.28 | 0.000 | 31.23993 | 46.28448 |
| _cons | 180.5198 | 2.666944 | 67.69 | 0.000 | 175.2007 | 185.8388 |

```
. contrast ar.agegrp
Contrasts of marginal linear predictions
Margins: asbalanced
```

| | df | F | P>F |
|---|---|---|---|
| agegrp | | | |
| (20-29 vs 10-19) | 1 | 4.73 | 0.0330 |
| (30-39 vs 20-29) | 1 | 12.51 | 0.0007 |
| (40-59 vs 30-39) | 1 | 5.21 | 0.0255 |
| (60-79 vs 40-59) | 1 | 5.21 | 0.0255 |
| Joint | 4 | 35.02 | 0.0000 |
| Denominator | 70 | | |

| | Contrast | Std. err. | [95% conf. | interval] |
|---|---|---|---|---|
| agegrp | | | | |
| (20-29 vs 10-19) | 8.203575 | 3.771628 | .6812991 | 15.72585 |
| (30-39 vs 20-29) | 13.33748 | 3.771628 | 5.815204 | 20.85976 |
| (40-59 vs 30-39) | 8.60962 | 3.771628 | 1.087345 | 16.1319 |
| (60-79 vs 40-59) | 8.611533 | 3.771628 | 1.089257 | 16.13381 |

We could use orthogonal polynomial contrasts to test whether there is a linear, quadratic, or even higher-order trend in the estimated cell means.

```
. contrast p.agegrp, noeffects
Contrasts of marginal linear predictions
Margins: asbalanced
```

| | df | F | P>F |
|---|---|---|---|
| agegrp | | | |
| (linear) | 1 | 139.11 | 0.0000 |
| (quadratic) | 1 | 0.15 | 0.6962 |
| (cubic) | 1 | 0.37 | 0.5448 |
| (quartic) | 1 | 0.43 | 0.5153 |
| Joint | 4 | 35.02 | 0.0000 |
| Denominator | 70 | | |

You are not limited to using `contrast` in one-way models. Had we fit

```
. regress chol agegrp##race
```

we could `contrast` to obtain tests of the main effects and interaction effects.

```
. contrast agegrp##race
```

These results would be the same as would be reported by `anova`. We mention this because you can use `contrast` after any estimation command that allows factor variables and works with `margins`. You could type

```
. logistic highbp agegrp##race
. contrast agegrp##race
```

See [R] **contrast** and [R] **margins, contrast**.

## 20.20 Graphing margins, marginal effects, and contrasts

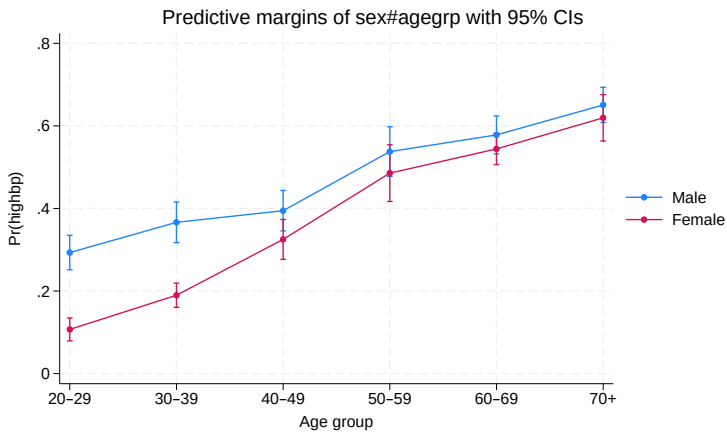
Using `marginsplot`, you can graph any of the results produced by `margins`, and because `margins` can replicate any of the results produced by `pwcompare` and `contrast`, you can graph any of the results produced by them, too.

In [U] **20.16.3 Obtaining predictive margins**, we did the following:

```
. use https://www.stata-press.com/data/r18/nhanes2
. svy: logistic highbp sex##agegrp##c.bmi
. margins sex#agegrp, vce(unconditional)
```

We can now graph those results by typing

```
. marginsplot, xdimension(agegrp)
Variables that uniquely identify margins: sex agegrp
```




See [R] **marginsplot**. Mitchell (2021) shows how to make similar graphs for a variety of predictions and models.

## 20.21 Dynamic forecasts and simulations

The `forecast` suite of commands lets you obtain forecasts from forecast models, collections of equations that jointly determine the outcomes of one or more endogenous variables. You fit stochastic equations using estimation commands such as `regress` or `var`, and then you add those results to your forecast model. You can also specify identities that define variables in terms of other variables, and you can also specify exogenous variables whose values are already known or otherwise determined by factors outside your model. `forecast` then solves the resulting system of equations to obtain forecasts.

`forecast` works with time-series and panel datasets, and you can obtain either dynamic or static forecasts. Dynamic forecasts use previous periods' forecast values wherever lags appear in the model's equations and thus allow you to obtain forecasts for multiple periods in the future. Static forecasts use previous periods' actual values wherever lags appear in the model's equations, so if you use lags, you cannot make predictions much beyond the end of the time horizon in your dataset. However, static forecasts are useful during model development.

You can incorporate outside information into your forecasts, and you can specify a future path for some of the model's variables and obtain forecasts for the other variables conditional on that path.

These features allow you to produce forecasts under different scenarios, and they allow you to explore how different policy interventions would affect your forecasts.

`forecast` also has the capability to produce confidence intervals around the forecasts. You can have `forecast` account for the sampling variance of the estimated parameters in the stochastic equations. There are two ways to account for an additive stochastic error term in the stochastic equations. You can request either that `forecast` assume the error terms are normally distributed and take draws from a random-number generator or that `forecast` take random samples from the pool of static-forecast residuals.

See [TS] **forecast**.

## 20.22 Obtaining robust variance estimates

Many Stata estimation commands provide robust and cluster-robust variance estimates. To obtain these estimates, you simply specify option `vce(robust)` to obtain robust standard errors or `vce(cluster` *clustvar*`)` to obtain cluster-robust standard errors. Below, we provide a general discussion of why you might specify one of these options, how to interpret standard errors with and without `vce(robust)` specified, and an overview of important concepts relating to cluster-robust standard errors.

Estimates of variance refer to estimated standard errors or, more completely, the estimated variance–covariance matrix of the estimators of which the standard errors are a subset, being the square root of the diagonal elements. Call this matrix the variance. All estimation commands produce an estimate of variance and, using that, produce confidence intervals and significance tests.

In addition to the conventional estimator of variance, there is another estimator that has been called by various names because it has been derived independently in different ways by different authors. Two popular names associated with the calculation are Huber and White, but it is also known as the sandwich estimator of variance (because of how the calculation formula physically appears) and the robust estimator of variance (because of claims made about it). Also, this estimator has an independent and long tradition in the survey literature.

The conventional estimator of variance is derived by starting with a model. Let's start with the regression model

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

although it is not important for the discussion that we are using regression. Under the model-based approach, we assume that the model is true and thereby derive an estimator for $\boldsymbol{\beta}$ and its variance.

The estimator of the standard error of $\widehat{\boldsymbol{\beta}}$ we develop is based on the assumption that the model is true in every detail. $y_i$ is not exactly equal to $\mathbf{x}_i\boldsymbol{\beta}$ (so that we would only need to solve an equation to obtain precisely that value of $\boldsymbol{\beta}$) because the observed $y_i$ has noise $\epsilon_i$ added to it, the noise is Gaussian, and it has constant variance. That noise leads to the uncertainty about $\boldsymbol{\beta}$, and it is from the characteristics of that noise that we are able to calculate a sampling distribution for $\widehat{\boldsymbol{\beta}}$.

The key thought here is that the standard error of $\widehat{\boldsymbol{\beta}}$ arises because of $\epsilon$ and is valid only because the model is absolutely, without question, true; we just do not happen to know the particular values of $\boldsymbol{\beta}$ and $\sigma^2$ that make the model true. The implication is that, in an infinite-sized sample, the estimator $\widehat{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$ would converge to the true value of $\boldsymbol{\beta}$ and that its variance would go to 0.

Now here is another interpretation of the estimation problem: We are going to fit the model

$$y_i = \mathbf{x}_i\mathbf{b} + e_i$$

and, to obtain estimates of $\mathbf{b}$, we are going to use the calculation formula

$$\widehat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

We have made no claims that the model is true or any claims about $e_i$ or its distribution. We shifted our notation from $\boldsymbol{\beta}$ and $\epsilon_i$ to $\mathbf{b}$ and $e_i$ to emphasize this. All we have stated are the physical actions we intend to carry out on the data. Interestingly, it is possible to calculate a standard error for $\widehat{\mathbf{b}}$ here. At least, it is possible if you will agree with us on what the standard error measures are.

We are going to define the standard error as measuring the standard error of the calculated $\widehat{\mathbf{b}}$ if we were to repeat the data collection followed by estimation over and over again.

This is a different concept of the standard error from the conventional, model-based ideas, but it is related. Both measure uncertainty about $\mathbf{b}$ (or $\boldsymbol{\beta}$). The regression model–based derivation states from where the variation arises and so can make grander statements about the applicability of the measured standard error. The weaker second interpretation makes fewer assumptions and so produces a standard error suitable for one purpose.

There is a subtle difference in interpretation of these identically calculated point estimates. $\widehat{\boldsymbol{\beta}}$ is the estimate of $\boldsymbol{\beta}$ under the assumption that the model is true. $\widehat{\mathbf{b}}$ is the estimate of $\mathbf{b}$, which is merely what the estimator would converge to if we collected more and more data.

Is the estimate of $\mathbf{b}$ unbiased? If we mean, "Does $\mathbf{b} = \boldsymbol{\beta}$?" that depends on whether the model is true. $\widehat{\mathbf{b}}$ is, however, an unbiased estimate of $\mathbf{b}$, which admittedly is not saying much.

What if $\mathbf{x}$ and $e$ are correlated? Don't we have a problem then? We may have an interpretation problem—$\mathbf{b}$ may not measure what we want to measure, namely, $\boldsymbol{\beta}$—but we measure $\widehat{\mathbf{b}}$ to be such-and-such and expect, if the experiment and estimation were repeated, that we would observe results in the range we have reported.

So, we have two different understandings of what the parameters mean and how the variance in their estimators arises. However, both interpretations must confront the issue of how to make valid statistical inference about the coefficient estimates when the data do not come from a simple random sample or the distribution of $(\mathbf{x}_i, \epsilon_i)$ is not independent and identically distributed (i.i.d.). In essence, we need an estimator of the standard errors that is robust to this deviation from the standard case.

Hence, the name *the robust estimate of variance*; its associated authors are Huber (1967) and White (1980, 1982) (who developed it independently), although many others have extended its development, including Gail, Tan, and Piantadosi (1988); Kent (1982); Royall (1986); and Lin and Wei (1989). In the survey literature, this same estimator has been developed; see Kish and Frankel (1974), Fuller (1975), and Binder (1983). Most of Stata's estimation commands can produce this alternative estimate of variance and do so via the `vce(robust)` option.

### 20.22.1 Interpreting standard errors

Without `vce(robust)`, we get one measure of variance:

```
. use https://www.stata-press.com/data/r18/auto7
(1978 automobile data)
. regress mpg weight foreign
```

| Source | SS | df | MS | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(2, 71) | = | 69.75 |
| Model | 1619.2877 | 2 | 809.643849 | Prob > F | = | 0.0000 |
| Residual | 824.171761 | 71 | 11.608053 | R-squared | = | 0.6627 |
| | | | | Adj R-squared | = | 0.6532 |
| Total | 2443.45946 | 73 | 33.4720474 | Root MSE | = | 3.4071 |

| mpg | Coefficient | Std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0065879 | .0006371 | -10.34 | 0.000 | -.0078583 | -.0053175 |
| foreign | -1.650029 | 1.075994 | -1.53 | 0.130 | -3.7955 | .4954422 |
| _cons | 41.6797 | 2.165547 | 19.25 | 0.000 | 37.36172 | 45.99768 |

With `vce(robust)`, we get another:

```
. regress mpg weight foreign, vce(robust)
```

| Linear regression | | | | Number of obs | = | 74 |
|---|---|---|---|---|---|---|
| | | | | F(2, 71) | = | 73.81 |
| | | | | Prob > F | = | 0.0000 |
| | | | | R-squared | = | 0.6627 |
| | | | | Root MSE | = | 3.4071 |

| mpg | Coefficient | Robust std. err. | t | P>\|t\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| weight | -.0065879 | .0005462 | -12.06 | 0.000 | -.007677 | -.0054988 |
| foreign | -1.650029 | 1.132566 | -1.46 | 0.150 | -3.908301 | .6082424 |
| _cons | 41.6797 | 1.797553 | 23.19 | 0.000 | 38.09548 | 45.26392 |

Either way, the point estimates are the same. (See [R] **regress** for an example where specifying `vce(robust)` produces strikingly different standard errors.)

How do we interpret these results? Let's consider the model-based interpretation. Suppose that

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i$$

where $(\mathbf{x}_i, \epsilon_i)$ are i.i.d. with variance $\sigma^2$. For the model-based interpretation, we also must assume that $\mathbf{x}_i$ and $\epsilon_i$ are uncorrelated. With these assumptions and a few technical regularity conditions, our first regression gives us consistent parameter estimates and standard errors that we can use for valid statistical inference about the coefficients. Now suppose that we weaken our assumptions so that $(\mathbf{x}_i, \epsilon_i)$ are independent and—but not necessarily—identically distributed. Our parameter estimates are still consistent, but the standard errors from the first regression can no longer be used to make valid inference. We need estimates of the standard errors that are robust to the fact that the error term is not identically distributed. The standard errors in our second regression are just what we need. We can use them to make valid statistical inference about our coefficients, even though our data are not identically distributed.

Now consider a non–model-based interpretation. If our data come from a survey design that ensures that $(\mathbf{x}_i, e_i)$ are i.i.d., then we can use the nonrobust standard errors for valid statistical inference about the population parameters $\mathbf{b}$. For this interpretation, we do not need to assume that $\mathbf{x}_i$ and $e_i$

are uncorrelated. If they are uncorrelated, the population parameters $\mathbf{b}$ and the model parameters $\beta$ are the same. However, if they are correlated, then the population parameters $\mathbf{b}$ that we are estimating are not the same as the model-based $\beta$. So, what we are estimating is different, but we still need standard errors that allow us to make valid statistical inference. If the process that we used to collect the data caused $(\mathbf{x}_i, e_i)$ to be independent but not identically distributed, then we need to use the robust standard errors to make valid statistical inference about the population parameters $\mathbf{b}$.

### 20.22.2 Correlated errors: Cluster–robust standard errors

The robust estimator of variance has one feature that the conventional estimator does not have: the ability to relax the assumption of independence of the observations. That is, if you specify the `vce(cluster` *clustvar*`)` option, it can produce "correct" standard errors (in the measurement sense), even if the observations are correlated.

For the automobile data, it is difficult to believe that the models of the various manufacturers are truly independent. Manufacturers, after all, use common technology, engines, and drive trains across their model lines. The VW Dasher in the above regression has a measured residual of −2.80. Having been told that, do you really believe that the residual for the VW Rabbit is as likely to be above 0 as below? (The residual is −2.32.) Similarly, the measured residual for the Chevrolet Malibu is 1.27. Does that provide information about the expected value of the residual of the Chevrolet Monte Carlo (which turns out to be 1.53)?

We need to be careful about picking examples from data; we have not told you about the Datsun 210 and 510 (residuals +8.28 and −1.01) or the Cadillac Eldorado and Seville (residuals −1.99 and +7.58), but you should at least question the assumption of independence. It may be believable that the measured `mpg` given the weight of one manufacturer's vehicles is independent of other manufacturers' vehicles, but it is at least questionable whether a manufacturer's vehicles are independent of one another.

In commands with the `vce(robust)` option, another option—`vce(cluster` *clustvar*`)`—relaxes the independence assumption and requires only that the observations be independent across the clusters:

```
. regress mpg weight foreign, vce(cluster manufacturer)
Linear regression                               Number of obs     =         74
                                                F(2, 22)          =      90.93
                                                Prob > F          =     0.0000
                                                R-squared         =     0.6627
                                                Root MSE          =     3.4071

                                  (Std. err. adjusted for 23 clusters in manufacturer)
------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0065879   .0005339   -12.34   0.000    -.0076952   -.0054806
     foreign |  -1.650029   1.039033    -1.59   0.127    -3.804852    .5047939
       _cons |   41.6797   1.844559    22.60   0.000     37.85432    45.50508
------------------------------------------------------------------------------
```

It turns out that, in these data, whether or not we specify `vce(cluster` *clustvar*`)` makes little difference. The VW and Chevrolet examples above were not representative; had they been, the confidence intervals would have widened. (In the above, `manuf` is a variable that takes on values such as "Chev." or "VW", recording the manufacturer of the vehicle. This variable was created from variable `make`, which contains values such as "Chev. Malibu" or "VW Rabbit", by extracting the first word.)

As a demonstration of how well clustering can work, in [R] **regress** we fit a random-effects model with `regress, vce(robust)` and then compared the results with ordinary least squares and the generalized least squares (GLS) random-effects estimator. Here we will simply summarize the results.

We start with a dataset on 4,711 women aged 14–46 years. Subjects appear an average of 6.057 times in the data; there are a total of 28,534 observations. The model we use is log wage on age, age-squared, and job tenure. The focus of the example is the estimated coefficient on tenure. We obtain the following results:

| Estimator | Point estimate | Confidence interval |
|---|---|---|
| (Inappropriate) least squares | 0.039 | [ 0.038, 0.041 ] |
| Robust clustered | 0.039 | [ 0.036, 0.042 ] |
| GLS random effects | 0.026 | [ 0.025, 0.027 ] |

Notice how well the robust clustered estimate does compared with the GLS random-effects model. We then run a Hausman specification test, obtaining $\chi^2(3) = 336.62$, which casts grave doubt on the assumptions justifying the use of the GLS estimator and hence on the GLS results. At this point, we will simply quote our comments:

> Meanwhile, our robust regression results still stand, as long as we are careful about the interpretation. The correct interpretation is that if the data collection were repeated (on women sampled the same way as in the original sample) and if we were to refit the model, then 95% of the time we would expect the estimated coefficient on tenure to be in the range $[\,0.036, 0.042\,]$.
>
> Even with robust regression, we must be careful about going beyond that statement. Here the Hausman test is probably picking up something that differs within- and between-person, which would cast doubt on our robust regression model in terms of interpreting $[\,0.036, 0.042\,]$ to contain the rate of return for keeping a job, economywide, for all women, without exception.

The formula for the robust estimator of variance is

$$\widehat{\mathcal{V}} = \widehat{\mathbf{V}}\Big(\sum_{j=1}^{N} \mathbf{u}_j'\mathbf{u}_j\Big)\widehat{\mathbf{V}}$$

where $\widehat{\mathbf{V}} = (-\partial^2 \ln L/\partial\boldsymbol{\beta}^2)^{-1}$ (the conventional estimator of variance) and $\mathbf{u}_j$ (a row vector) is the contribution from the $j$th observation to $\partial \ln L/\partial\boldsymbol{\beta}$.

In the example above, observations are assumed to be independent. Assume for a moment that the observations denoted by $j$ are not independent but that they can be divided into $M$ groups $G_1$, $G_2$, ..., $G_M$ that are independent. The robust estimator of variance is

$$\widehat{\mathcal{V}} = \widehat{\mathbf{V}}\Big(\sum_{k=1}^{M} \mathbf{u}_k^{(G)\prime}\mathbf{u}_k^{(G)}\Big)\widehat{\mathbf{V}}$$

where $\mathbf{u}_k^{(G)}$ is the contribution of the $k$th group to $\partial \ln L/\partial\boldsymbol{\beta}$. That is, application of the robust variance formula merely involves using a different decomposition of $\partial \ln L/\partial\boldsymbol{\beta}$, namely, $\mathbf{u}_k^{(G)}$, $k = 1, \ldots, M$, rather than $\mathbf{u}_j$, $j = 1, \ldots, N$. Moreover, if the log-likelihood function is additive in the observations denoted by $j$,

$$\ln L = \sum_{j=1}^{N} \ln L_j$$

then $\mathbf{u}_j = \partial \ln L_j / \partial \boldsymbol{\beta}$, so

$$\mathbf{u}_k^{(G)} = \sum_{j \in G_k} \mathbf{u}_j$$

That is what the `vce(cluster` *clustvar*`)` option does. (This point was first made in writing by Rogers [1993], although he considered the point an obvious generalization of Huber [1967] and the calculation—implemented by Rogers—had appeared in Stata a year earlier.)

## ❑ Technical note

What is written above is asymptotically correct but ignores a finite-sample adjustment to $\widehat{\mathcal{V}}$. For maximum likelihood estimators, when you specify `vce(robust)` but not `vce(cluster` *clustvar*`)`, a better estimate of variance is $\widehat{\mathcal{V}}^* = \{N/(N-1)\}\widehat{\mathcal{V}}$. When you also specify the `vce(cluster` *clustvar*`)` option, this becomes $\widehat{\mathcal{V}}^* = \{M/(M-1)\}\widehat{\mathcal{V}}$.

For linear regression, the finite-sample adjustment is $N/(N-k)$ without `vce(cluster` *clustvar*`)`—where $k$ is the number of regressors—and is $\{M/(M-1)\}\{(N-1)/(N-k)\}$ with `vce(cluster` *clustvar*`)`. Also, two data-dependent modifications to the calculation for $\widehat{\mathcal{V}}^*$, suggested by MacKinnon and White (1985), are provided by `regress`; see [R] **regress**. Angrist and Pischke (2009, chap. 8) is devoted to robust covariance matrix estimation and offers practical guidance on the use of `vce(robust)` and `vce(cluster` *clustvar*`)` in both cross-sectional and panel-data applications.

❑

Halbert Lynn White Jr. (1950–2012) was born in Kansas City. After receiving economics degrees at Princeton and MIT, he taught and researched econometrics at the University of Rochester and, from 1979, at the University of California in San Diego. He also co-founded an economics and legal consulting firm known for its rigorous use of econometrics methods. His 1980 paper on heteroskedasticity introduced the use of robust covariance matrices to economists and passed 16,000 citations in Google Scholar in 2012. His 1982 paper on maximum likelihood estimation of misspecified models helped develop the now-common use of quasimaximum likelihood estimation techniques. Later in his career, he explored the use of neural networks, nonparametric models, and time-series modeling of financial markets.

Among his many awards and distinctions, White was made a fellow of the American Academy of Arts and Sciences and the Econometric Society, and he won a fellowship from the John Simon Guggenheim Memorial Foundation. Had he not died prematurely, many scholars believe he would have eventually been awarded the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel.

Aside from his academic work, White was an avid jazz musician who played with well-known jazz trombonist and fellow University of California at San Diego teacher Jimmy Cheatam.

Peter Jost Huber (1934– ) was born in Wohlen (Aargau, Switzerland). He gained mathematics degrees from ETH Zürich, including a PhD thesis on homotopy theory, and then studied statistics at Berkeley on postdoctoral fellowships. This visit yielded a celebrated 1964 paper on robust estimation, and Huber's later monographs on robust statistics were crucial in directing that field. Thereafter, his career took him back and forth across the Atlantic, with periods at Cornell, ETH Zürich, Harvard, MIT, and Bayreuth. His work has touched several other major parts of statistics, theoretical and applied, including regression, exploratory multivariate analysis, large datasets, and statistical computing. Huber also has a major long-standing interest in Babylonian astronomy.

## 20.23 Obtaining scores

Many of the estimation commands that provide the `vce(robust)` option also provide the ability to generate equation-level score variables via the `predict` command. With the `score` option, `predict` returns an important ingredient into the robust variance calculation that is sometimes useful in its own right. As explained above in [U] **20.22 Obtaining robust variance estimates**, ignoring the finite-sample corrections, the robust estimate of variance is

$$\widehat{\mathcal{V}} = \widehat{\mathbf{V}}\Big(\sum_{j=1}^{N} \mathbf{u}_j'\mathbf{u}_j\Big)\widehat{\mathbf{V}}$$

where $\widehat{\mathbf{V}} = (-\partial^2 \ln L/\partial\boldsymbol{\beta}^2)^{-1}$ is the conventional estimator of variance. If we consider likelihood functions that are additive in the observations

$$\ln L = \sum_{j=1}^{N} \ln L_j$$

then $\mathbf{u}_j = \partial \ln L_j/\partial\boldsymbol{\beta}$. In general, function $L_j$ is a function of $\mathbf{x}_j$ and $\boldsymbol{\beta}$, $L_j(\boldsymbol{\beta};\mathbf{x}_j)$. For many likelihood functions, however, it is only the linear form $\mathbf{x}_j\boldsymbol{\beta}$ that enters the function. In those cases,

$$\frac{\partial \ln L_j(\mathbf{x}_j\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \frac{\partial \ln L_j(\mathbf{x}_j\boldsymbol{\beta})}{\partial(\mathbf{x}_j\boldsymbol{\beta})}\frac{\partial(\mathbf{x}_j\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \frac{\partial \ln L_j(\mathbf{x}_j\boldsymbol{\beta})}{\partial(\mathbf{x}_j\boldsymbol{\beta})}\mathbf{x}_j$$

By writing $u_j = \partial \ln L_j(\mathbf{x}_j\boldsymbol{\beta})/\partial(\mathbf{x}_j\boldsymbol{\beta})$, this becomes simply $u_j\mathbf{x}_j$. Thus the formula for the robust estimate of variance can be rewritten as

$$\widehat{\mathcal{V}} = \widehat{\mathbf{V}}\Big(\sum_{j=1}^{N} u_j^2\mathbf{x}_j'\mathbf{x}_j\Big)\widehat{\mathbf{V}}$$

We refer to $u_j$ as the equation-level score (in the singular), and it is $u_j$ that is returned when you use `predict` with the `score` option. $u_j$ is like a residual in that

1. $\sum_j u_j = 0$ and
2. correlation of $u_j$ and $\mathbf{x}_j$, calculated over $j = 1, \ldots, N$, is 0.

In fact, for linear regression, $u_j$ is the residual, normalized,

$$\begin{aligned}\frac{\partial \ln L_j}{\partial(\mathbf{x}_j\boldsymbol{\beta})} &= \frac{\partial}{\partial(\mathbf{x}_j\boldsymbol{\beta})} \ln f\Big\{(y_j - \mathbf{x}_j\boldsymbol{\beta})/\sigma\Big\} \\ &= (y_j - \mathbf{x}_j\boldsymbol{\beta})/\sigma\end{aligned}$$

where $f(\cdot)$ is the standard normal density.

## ▷ Example 19

`probit` provides the `vce(robust)` option and `predict, score`. Equation-level scores play an important role in calculating the robust estimate of variance, but we can use `predict, score` regardless of whether we specify `vce(robust)`:

```
. use https://www.stata-press.com/data/r18/auto2
(1978 automobile data)
. probit foreign mpg weight
Iteration 0:  Log likelihood =  -45.03321
Iteration 1:  Log likelihood = -27.914626
Iteration 2:  Log likelihood = -26.858074
Iteration 3:  Log likelihood = -26.844197
Iteration 4:  Log likelihood = -26.844189
Iteration 5:  Log likelihood = -26.844189
Probit regression                                 Number of obs =     74
                                                  LR chi2(2)    =  36.38
                                                  Prob > chi2   = 0.0000
Log likelihood = -26.844189                       Pseudo R2     = 0.4039
------------------------------------------------------------------------------
     foreign | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         mpg |  -.1039503   .0515689    -2.02   0.044    -.2050235   -.0028772
      weight |  -.0023355   .0005661    -4.13   0.000     -.003445   -.0012261
       _cons |   8.275464   2.554142     3.24   0.001     3.269437    13.28149
------------------------------------------------------------------------------
. predict double u, score
. summarize u
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
           u |         74   -6.64e-14    .5988325  -1.655439   1.660787
. correlate u mpg weight
(obs=74)
             |        u      mpg   weight
-------------+---------------------------
           u |   1.0000
         mpg |   0.0000   1.0000
      weight |  -0.0000  -0.8072   1.0000
. list make foreign mpg weight u if abs(u)>1.65
     +-------------------------------------------------------+
     | make           foreign   mpg   weight           u     |
     |-------------------------------------------------------|
24.  | Ford Fiesta   Domestic    28    1,800   -1.6554395     |
64.  | Peugeot 604    Foreign    14    3,420    1.6607871     |
     +-------------------------------------------------------+
```

The light, high-mileage Ford Fiesta is surprisingly domestic, whereas the heavy, low-mileage Peugeot 604 is surprisingly foreign.

◁

## ❑ Technical note

For some estimation commands, one score is not enough. Consider a likelihood that can be written as $L_j(\mathbf{x}_j\boldsymbol{\beta}_1, \mathbf{z}_j\boldsymbol{\beta}_2)$, a function of two linear forms (or linear equations). Then $\partial \ln L_j/\partial\boldsymbol{\beta}$ can be written as $(\partial \ln L_j/\partial\boldsymbol{\beta}_1, \partial \ln L_j/\partial\boldsymbol{\beta}_2)$. Each of the components can in turn be written as $[\partial \ln L_j/\partial(\beta_1\mathbf{x})]\mathbf{x} = u_1\mathbf{x}$ and $[\partial \ln L_j/\partial(\beta_2\mathbf{z})]\mathbf{z} = u_2\mathbf{z}$. There are then two equation-level scores, $u_1$ and $u_2$, and, in general, there could be more.

Stata's `streg, distribution(weibull)` command is an example of this: it estimates $\boldsymbol{\beta}$ and a shape parameter, $\ln p$, the latter of which can be thought of as a degenerate linear form $(\ln p)\mathbf{z}$ with $\mathbf{z} = \mathbf{1}$. After this command, `predict, scores` requires that you specify two new variable names, or you can specify *stub*`*`, which will generate new variables *stub*`1` and *stub*`2`; the first will be defined containing $u_1$—the score associated with $\boldsymbol{\beta}$—and the second will be defined containing $u_2$—the score associated with $\ln p$.

❑

## ❑ Technical note

Using Stata's matrix commands—see [P] **matrix**—we can make the robust variance calculation for ourselves and then compare it with that made by Stata.

```
. use https://www.stata-press.com/data/r18/auto2, clear
(1978 automobile data)
. quietly probit foreign mpg weight
. predict double u, score
. matrix accum S = mpg weight [iweight=u^2*74/73]
(obs=26.53642547)
. matrix rV = e(V)*S*e(V)
. matrix list rV
symmetric rV[3,3]
                  foreign:    foreign:    foreign:
                      mpg      weight       _cons
   foreign:mpg   .00352299
foreign:weight   .00002216   2.434e-07
 foreign:_cons  -.14090346  -.00117031   6.4474174
. quietly probit foreign mpg weight, vce(robust)
. matrix list e(V)
symmetric e(V)[3,3]
                  foreign:    foreign:    foreign:
                      mpg      weight       _cons
   foreign:mpg   .00352299
foreign:weight   .00002216   2.434e-07
 foreign:_cons  -.14090346  -.00117031   6.4474174
```

The results are the same.

There is an important lesson here for programmers. Given the scores, conventional variance estimates can be easily transformed to robust estimates. If we were writing a new estimation command, it would not be difficult to include a `vce(robust)` option.

It is, in fact, easy if we ignore clustering. With clustering, it is more work because the calculation involves forming sums within clusters. For programmers interested in implementing robust variance calculations, Stata provides a `_robust` command to ease the task. This is documented in [P] **_robust**.

To use `_robust`, you first produce conventional results (a vector of coefficients and covariance matrix) along with a variable containing the scores $u_j$ (or variables if the likelihood function has more than one stub). You then call `_robust`, and it will transform your conventional variance estimate into the robust estimate. `_robust` will handle the work associated with clustering and the details of the finite-sample adjustment, and it will even label your output so that the word *Robust* appears above the standard error when the results are displayed.

Of course, this is all even easier if you write your commands with Stata's `ml` maximum likelihood optimization, in which case you merely pass the `vce(robust)` option on to `ml`. Then, `ml` will call `_robust` itself and do all the work for you.

❑

❑ Technical note

For some estimation commands, `predict, score` computes parameter-level scores $\partial L_j/\partial\beta$ instead of equation-level scores $\partial L_j/\partial\mathbf{x}_j\beta$. Those estimation commands, such as `cmclogit`, `stcox`, and the multilevel mixed-effects commands, share the characteristic that there are multiple observations per independent event.

In making the robust variance calculation, parameter-level scores $\partial L_j/\partial\beta$ are really needed, and so you may be asking yourself why `predict, score` does not always produce parameter-level scores. In the usual case, we can obtain them from equation-level scores via the chain rule, and fewer variables are required if we adopt this approach. In the cases above, however, the likelihood is calculated at the group level and is not split into contributions from the individual observations. Thus, the chain rule cannot be used, and we must use the parameter level scores directly.

`_robust` can be tricked into using them if each parameter appears to be in its own equation as a constant. This requires resetting the row and column stripes on the covariance matrix before `_robust` is called. The equation names for each row and column must be unique, and the variable names must all be `_cons`.

❑

## 20.24 Weighted estimation

The syntax for weights was introduced in [U] **11.1.6 weight**. Stata provides four kinds of weights: `fweights`, or frequency weights; `pweights`, or sampling weights; `aweights`, or analytic weights; and `iweights`, or importance weights. The syntax for using each is the same. Type

```
. regress y x1 x2
```

and you obtain unweighted estimates; type

```
. regress y x1 x2 [pweight=pop]
```

and you obtain (in this example) `pweighted` estimates.

The sections below explain how each type of weight is used in estimation.

### 20.24.1 Frequency weights

Frequency weights—`fweights`—are integers and are nothing more than replication counts. The weight is statistically uninteresting, but from a data-processing perspective it is important. Consider the following data,

| y | x1 | x2 |
|---|---|---|
| 22 | 1 | 0 |
| 22 | 1 | 0 |
| 22 | 1 | 1 |
| 23 | 0 | 1 |
| 23 | 0 | 1 |
| 23 | 0 | 1 |

and the estimation command

```
. regress y x1 x2
```

Equivalent is the following, more compressed data,

| y | x1 | x2 | pop |
|---|---|---|---|
| 22 | 1 | 0 | 2 |
| 22 | 1 | 1 | 1 |
| 23 | 0 | 1 | 3 |

and the corresponding estimation command

```
. regress y x1 x2 [fweight=pop]
```

When you specify frequency weights, you are treating each observation as one or more real observations.

❑ Technical note

You might occasionally run across a command that does not allow weights at all, especially among community-contributed commands. You can use `expand` (see [D] **expand**) with such commands to obtain frequency-weighted results. The `expand` command duplicates observations so that the data become self-weighting. Suppose that you want to run the command `usercmd`, which does something or other, and you would like to type `usercmd y x1 x2 [fw=pop]`. Unfortunately, `usercmd` does not allow weights. Instead, you type

```
. expand pop
. usercmd y x1 x2
```

to obtain your result. Moreover, there is an important principle here: the results of running any command with frequency weights should be the same as running the command on the unweighted, expanded data. Unweighted, duplicated data and frequency-weighted data are merely two ways of recording identical information.

❑

### 20.24.2 Analytic weights

Analytic weights—analytic is a term we made up—statistically arise in one particular problem: linear regression on data that are themselves observed means. That is, think of the model

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

and now think about fitting this model on data $(\overline{y}_j, \overline{\mathbf{x}}_j)$ that are themselves observed averages. For instance, a piece of the underlying data for $(y_i, \mathbf{x}_i)$ might be $(3, 1)$, $(4, 2)$, and $(2, 2)$, but you do not know that. Instead, you have one observation $\{(3 + 4 + 2)/3, (1 + 2 + 2)/3\} = (3, 1.67)$ and know only that the $(3, 1.67)$ arose as the average of three underlying observations. All your data are like that.

`regress` with `aweights` is the solution to that problem:

```
. regress y x [aweight=pop]
```

There is a history of misusing such weights. A researcher does not have cell-mean data but instead has a probability-weighted random sample. Long before Stata existed, some researchers were using `aweights` to produce estimates from such samples. We will come back to this point in [U] **20.24.3 Sampling weights** below.

Anyway, the statistical problem that `aweights` resolve can be written as

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2/w_i)$$

where the $w_i$ are the analytic weights. The details of the solution are to make linear regression calculations using the weights as if they were `fweights` but to normalize them to sum to $N$ before doing that.

Most commands that allow aweights handle them in this manner. That is, if you specify aweights, they are

1. normalized to sum to $N$ and then
2. inserted in the calculation formulas in the same way as fweights.

While we focus on the use of aweights in linear regression above, aweights are allowed by commands other than regress. These weights can be used more generally to account for observations that have different variances or different precisions. In that sense, we could also refer to analytic weights as precision weights.

### 20.24.3 Sampling weights

Sampling weights—probability weights or pweights—refer to probability-weighted random samples. Actually, what you specify in [pweight=...] is a variable recording the number of subjects in the full population that the sampled observation in your data represents. That is, an observation that had probability $1/3$ of being included in your sample has pweight 3.

Some researchers have used aweights with these kinds of data. If they do, they are probably making a mistake. Consider the regression model

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

Begin by considering the exact nature of the problem of fitting this model on cell-mean data—for which aweights are the solution: heteroskedasticity arising from the grouping. The error term $\epsilon_i$ is homoskedastic (meaning that it has constant variance $\sigma^2$). Say that the first observation in the data is the mean of three underlying observations. Then,

$$\begin{aligned} y_1 &= \mathbf{x}_1\boldsymbol{\beta} + \epsilon_1, \qquad \epsilon_1 \sim N(0, \sigma^2) \\ y_2 &= \mathbf{x}_2\boldsymbol{\beta} + \epsilon_2, \qquad \epsilon_2 \sim N(0, \sigma^2) \\ y_3 &= \mathbf{x}_3\boldsymbol{\beta} + \epsilon_3, \qquad \epsilon_3 \sim N(0, \sigma^2) \end{aligned}$$

and taking the mean,

$$(y_1 + y_2 + y_3)/3 = \{(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3)/3\}\boldsymbol{\beta} + (\epsilon_1 + \epsilon_2 + \epsilon_3)/3$$

For another observation in the data—which may be the result of summing a different number of observations—the variance will be different. Hence, the model for the data is

$$\overline{y}_j = \overline{x}_j\boldsymbol{\beta} + \overline{\epsilon}_j, \qquad \overline{\epsilon}_j \sim N(0, \sigma^2/N_j)$$

This makes intuitive sense. Consider two observations, one recording means over 2 subjects and the other recording means over 100,000 subjects. You would expect the variance of the residual to be less in the 100,000-subject observation; that is, there is more information in the 100,000-subject observation than in the 2-subject observation.

Now instead say that you are fitting the same model, $y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i$, $\epsilon_i \sim N(0, \sigma^2)$, on probability-weighted data. Each observation in your data is one subject, but the different subjects have different chances of being included in your sample. Therefore, for each subject in your data,

$$y_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i, \qquad \epsilon_i \sim N(0, \sigma^2)$$

That is, there is no heteroskedasticity problem. The use of the `aweight`ed estimator cannot be justified on these grounds.

As a matter of fact, from the argument just given, you do not need to adjust for the weights at all, although the argument does not justify not making an adjustment. If you do not adjust, you are holding tightly to the assumed truth of your model. Two issues arise when considering adjustment for sampling weights:

1. the efficiency of the point estimate $\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ and
2. the reported standard errors (and, more generally, the variance matrix of $\widehat{\boldsymbol{\beta}}$).

Efficiency argues in favor of adjustment, and that, by the way, is why many researchers have used `aweights` with `pweight`ed data. The adjustment implied by `pweights` to the point estimates is the same as the adjustment implied by `aweights`.

With regard to the second issue, the use of `aweights` produces incorrect results because it interprets larger weights as designating more accurately measured points. For `pweights`, however, the point is no more accurately measured—it is still just one observation with one residual $\epsilon_j$ and variance $\sigma^2$. In [U] **20.22 Obtaining robust variance estimates** above, we introduced another estimator of variance that measures the variation that would be observed if the data collection followed by the estimation were repeated. Those same formulas provide the solution to `pweights`, and they have the added advantage that they are not conditioned on the model being true. If we have any hopes of measuring the variation that would be observed were the data collection followed by estimation repeated, we must include the probability of the observations being sampled in the calculation.

In Stata, when you type

```
. regress y x1 x2 [pw=pop]
```

the results are the same as if you had typed

```
. regress y x1 x2 [pw=pop], vce(robust)
```

That is, specifying `pweights` implies the `vce(robust)` option and, hence, the robust variance calculation (but weighted). In this example, we use `regress` simply for illustration. The same is true of `probit` and all of Stata's estimation commands. Estimation commands that do not have a `vce(robust)` option (there are a few) do not allow `pweights`.

`pweights` are adequate for handling random samples where the probability of being sampled varies. `pweights` may be all you need. If, however, the observations are not sampled independently but are sampled in groups—called clusters in the jargon—you should specify the estimator's `vce(cluster` *clustvar*`)` option as well:

```
. regress y x1 x2 [pw=pop], vce(cluster block)
```

There are two ways of thinking about this:

1. The robust estimator answers the question of which variation would be observed were the data collection followed by the estimation repeated; if that question is to be answered, the estimator must account for the clustered nature of how observations are selected. If observations 1 and 2 are in the same cluster, then you cannot select observation 1 without selecting observation 2 (and, by extension, you cannot select observations like 1 without selecting observations like 2).
2. If you prefer, you can think about potential correlations. Observations in the same cluster may not really be independent—that is an empirical question to be answered by the data. For instance, if the clusters are neighborhoods, it would not be surprising that the individual neighbors are similar in their incomes, their tastes, and their attitudes, and even more similar

than two randomly drawn persons from the area at large with similar characteristics, such as age and sex.

Either way of thinking leads to the same (robust) estimator of variance.

Sampling weights usually arise from complex sampling designs, which often involve not only unequal probability sampling and cluster sampling but also stratified sampling. There is a family of commands in Stata designed to work with the features of complex survey data, and those are the commands that begin with `svy`. To fit a linear regression model with stratification, for example, you would use the `svy: regress` command.

Non-`svy` commands that allow `pweights` and clustering give essentially identical results to the `svy` commands. If the sampling design is simple enough that it can be accommodated by the non-`svy` command, that is a fine way to perform the analysis. The `svy` commands differ in that they have more features, and they do all the little details correctly for real survey data. See [SVY] **Survey** for a brief discussion of some of the issues involved in the analysis of survey data and for a list of all the differences between the `svy` and non-`svy` commands.

Not all model estimation commands in Stata allow `pweights`. This is often because they are computationally or statistically difficult to implement.

### 20.24.4 Importance weights

Stata's `iweights`—importance weights—are the emergency exit. These weights are for those who want to take control and create special effects. For example, programmers have used `regress` with `iweights` to compute iteratively reweighted least-squares solutions for various problems.

`iweights` are treated much like `aweights`, except that they are not normalized. Stata's iweight rule is that

1. the weights are not normalized and
2. they are generally inserted into calculation formulas in the same way as `fweights`. There are exceptions; see the *Methods and formulas* for the particular command.

`iweights` are used mostly by programmers who are often on the way to implementing one of the other kinds of weights.

## 20.25 A list of postestimation commands

The following commands can be used after estimation:

| | |
|---|---|
| `contrast` | contrasts and ANOVA-style joint tests of estimates |
| `estat ic` | Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC) |
| `estat summarize` | summary statistics for the estimation sample |
| `estat vce` | variance–covariance matrix of the estimators (VCE) |
| `estimates` | cataloging estimation results |
| `etable` | table of estimation results |
| `forecast` | dynamic forecasts and simulations |
| `hausman` | Hausman's specification test |
| `lincom` | point estimates, standard errors, testing, and inference for linear combinations of coefficients |
| `linktest` | link test for model specification |
| * `lrtest` | likelihood-ratio test |
| `margins` | marginal means, predictive margins, and marginal effects |
| `marginsplot` | graph the results from margins (profile plots, interaction plots, etc.) |
| `nlcom` | point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients |
| `predict` | predictions, residuals, influence statistics, and other diagnostic measures |
| `predictnl` | point estimates, standard errors, testing, and inference for generalized predictions |
| `pwcompare` | pairwise comparisons of estimates |
| `suest` | seemingly unrelated estimation |
| `test` | Wald tests of simple and composite linear hypotheses |
| `testnl` | Wald tests of nonlinear hypotheses |

Also see [U] **13.5 Accessing coefficients and standard errors** for accessing coefficients and standard errors.

The commands above are general-purpose postestimation commands that can be used after almost all estimation commands. Many estimation commands provide other estimator-specific postestimation commands. You can see the full list of postestimation commands available for an estimator by looking in the entry titled *estimator* **postestimation** that immediately follows each estimator's entry in the reference manuals.

You can also see which postestimation commands are available by launching the Postestimation Selector; select **Statistics** > **Postestimation**. You will see a list of all postestimation features that are available for the active estimation results. This list is automatically updated when a new estimation command is run or estimates are restored from memory or disk. See [R] **postest** for more details.

## 20.26 References


Afifi, A. A., and S. P. Azen. 1979. *Statistical Analysis: A Computer Oriented Approach*. 2nd ed. New York: Academic Press.

Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.

Binder, D. A. 1983. On the variances of asymptotically normal estimators from complex surveys. *International Statistical Review* 51: 279–292. https://doi.org/10.2307/1402588.

Buja, A., and H. R. Künsch. 2008. A conversation with Peter Huber. *Statistical Science* 23: 120–135. https://doi.org/10.1214/07-STS251.

Daniels, L., and N. Minot. 2020. *An Introduction to Statistics and Data Analysis Using Stata*. Thousand Oaks, CA: Sage.

Deaton, A. S. 1997. *The Analysis of Household Surveys: A Microeconometric Approach to Development Policy*. Baltimore, MD: Johns Hopkins University Press.

Fuller, W. A. 1975. Regression analysis for sample survey. *Sankhyā, Series C* 37: 117–132.

Gail, M. H., W. Y. Tan, and S. Piantadosi. 1988. Tests for no treatment effect in randomized clinical trials. *Biometrika* 75: 57–64. https://doi.org/10.2307/2336434.

Hampel, F. R. 1992. Introduction to Huber (1964) "Robust estimation of a location parameter". In *Breakthroughs in Statistics. Volume II: Methodology and Distribution*, ed. S. Kotz and N. L. Johnson, 479–491. New York: Springer.

Huber, P. J. 1967. The behavior of maximum likelihood estimates under nonstandard conditions. In Vol. 1 of *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 221–233. Berkeley: University of California Press.

——. 2011. *Data Analysis: What Can Be Learned from the Past 50 Years*. Hoboken, NJ: Wiley.

Kaufman, R. L. 2013. *Heteroskedasticity in Regression: Detection and Correction*. Thousand Oaks, CA: Sage.

Kent, J. T. 1982. Robust properties of likelihood ratio tests. *Biometrika* 69: 19–27. https://doi.org/10.2307/2335849.

Kish, L., and M. R. Frankel. 1974. Inference from complex samples. *Journal of the Royal Statistical Society, Series B* 36: 1–37. https://doi.org/10.1111/j.2517-6161.1974.tb00981.x.

Lin, D. Y., and L. J. Wei. 1989. The robust inference for the Cox proportional hazards model. *Journal of the American Statistical Association* 84: 1074–1078. https://doi.org/10.2307/2290085.

Lumley, T. S. 2020. Weights in statistics. *Biased and Inefficient*. https://notstatschat.rbind.io/2020/08/04/weights-in-statistics/.

MacKinnon, J. G., and H. L. White, Jr. 1985. Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics* 29: 305–325. https://doi.org/10.1016/0304-4076(85)90158-7.

McAleer, M., and T. Pérez-Amaral. 2012. Professor Halbert L. White, 1950–2012. *Journal of Economic Surveys* 26: 551–554. https://doi.org/10.1111/j.1467-6419.2012.00735.x.

Mitchell, M. N. 2021. *Interpreting and Visualizing Regression Models Using Stata*. 2nd ed. College Station, TX: Stata Press.

Pedace, R. 2013. *Econometrics for Dummies*. Hoboken, NJ: Wiley.

Rogers, W. H. 1993. sg17: Regression standard errors in clustered samples. *Stata Technical Bulletin* 13: 19–23. Reprinted in *Stata Technical Bulletin Reprints*, vol. 3, pp. 88–94. College Station, TX: Stata Press.

Royall, R. M. 1986. Model robust confidence intervals using maximum likelihood estimators. *International Statistical Review* 54: 221–226. https://doi.org/10.2307/1403146.

Wasserstein, R. L., and N. A. Lazar. 2016. The ASA statement on $p$-values: Context, process, and purpose. *American Statistician* 70: 129–133. https://doi.org/10.1080/00031305.2016.1154108.

White, H. L., Jr. 1980. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica* 48: 817–838. https://doi.org/10.2307/1912934.

——. 1982. Maximum likelihood estimation of misspecified models. *Econometrica* 50: 1–25. https://doi.org/10.2307/1912526.

Williams, R. 2012. Using the margins command to estimate and interpret adjusted predictions and marginal effects. *Stata Journal* 12: 308–331.