**stepwise — Stepwise estimation**

<table>
<thead>
<tr>
<th>Description</th>
<th>Quick start</th>
<th>Menu</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options</td>
<td>Remarks and examples</td>
<td>Stored results</td>
<td>Methods and formulas</td>
</tr>
<tr>
<td>References</td>
<td>Also see</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

`stepwise` performs stepwise estimation. Typing

```
.stepwise, pr(#) : command
```

performs backward-selection estimation for `command`. The stepwise selection method is determined by the following option combinations:

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr(#)</td>
<td>backward selection</td>
</tr>
<tr>
<td>pr(#) hierarchical</td>
<td>backward hierarchical selection</td>
</tr>
<tr>
<td>pr(#) pe(#)</td>
<td>backward stepwise</td>
</tr>
<tr>
<td>pe(#)</td>
<td>forward selection</td>
</tr>
<tr>
<td>pe(#) hierarchical</td>
<td>forward hierarchical selection</td>
</tr>
<tr>
<td>pr(#) pe(#) forward</td>
<td>forward stepwise</td>
</tr>
</tbody>
</table>

`command` defines the estimation command to be executed. The following Stata commands are supported by `stepwise`:

```
betareg, clogit, cloglog, glm, intreg, logistic, logit, nbreg, ologit, oprobit, poisson, probit, qreg, regress, scobit, stcox, stcrreg, stintreg, streg, tobit
```

`stepwise` expects `command` to have the following form:

```
command_name [ depvar ] term [ term ... ] [ if ] [ in ] [ weight ] [ , command_options ]
```

where `term` is either `varname` or `(varlist)` (a `varlist` in parentheses indicates that this group of variables is to be included or excluded together). `depvar` is not present when `command_name` is `stcox`, `stcrreg`, `stintreg`, or `streg`; otherwise, `depvar` is assumed to be present. For `intreg`, `depvar` is actually two dependent variable names (`depvar1` and `depvar2`).

`s` is a synonym for `stepwise`.

For model selection and estimation using lasso, see the Stata Lasso Reference Manual.
Quick start

Backward selection, removing terms with $p \geq 0.2$

```
stepwise, pr(.2): regress y x1 x2 x3 x4
```

As above, and select from the indicators for categorical variable a

```
stepwise, pr(.2): regress y x1 x2 x3 x4 i.a
```

As above, but force x1 to be included in model

```
stepwise, pr(.2) lockterm1: regress y x1 x2 x3 x4 i.a
```

Consider the indicators for a as a group for inclusion in model

```
stepwise, pr(.2): regress y x1 x2 x3 x4 (i.a)
```

Add d1, d2, and d3, and force them to be included in model

```
stepwise, pr(.2) lockterm1: regress y (d1 d2 d3) x1 x2 x3 x4 (i.a)
```

Forward selection, adding terms with $p < 0.1$

```
stepwise, pe(.1): regress y x1 x2 x3 x4
```

Backward stepwise selection, removing terms with $p \geq 0.2$ and adding those with $p < 0.1$

```
stepwise, pr(.2) pe(.1): regress y x1 x2 x3 x4
```

Forward stepwise selection, adding terms with $p < 0.1$ and removing those with $p \geq 0.2$

```
stepwise, pr(.2) pe(.1) forward: regress y x1 x2 x3 x4
```

Backward hierarchical selection

```
stepwise, pr(.2) hierarchical: regress y x1 x2 x3 x4
```

Forward hierarchical selection

```
stepwise, pe(.1) hierarchical: regress y x1 x2 x3 x4
```

Note: In the above examples, `regress` could be replaced with any estimation command allowing the `stepwise` prefix.
Syntax

```
stepwise [, options] : command
```

```
options   Description
```

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*pr(#)</td>
<td>significance level for removal from the model</td>
</tr>
<tr>
<td>*pe(#)</td>
<td>significance level for addition to the model</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model 2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward</td>
<td>perform forward-stepwise selection</td>
</tr>
<tr>
<td>hierarchical</td>
<td>perform hierarchical selection</td>
</tr>
<tr>
<td>lockterm1</td>
<td>keep the first term</td>
</tr>
<tr>
<td>lr</td>
<td>perform likelihood-ratio test instead of Wald test</td>
</tr>
</tbody>
</table>

### Reporting

| display_options | control columns and column formats and line width |

* At least one of `pr(#)` or `pe(#)` must be specified.

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

Weights are allowed if `command` allows them; see [U] 11.1.6 weight.

All postestimation commands behave as they would after `command` without the `stepwise` prefix; see the postestimation manual entry for `command`.

#### Options

**Model**

`pr(#)` specifies the significance level for removal from the model; terms with \( p \geq pr() \) are eligible for removal.

`pe(#)` specifies the significance level for addition to the model; terms with \( p < pe() \) are eligible for addition.

**Model 2**

`forward` specifies the forward-stepwise method and may be specified only when both `pr()` and `pe()` are also specified. Specifying both `pr()` and `pe()` without `forward` results in backward-stepwise selection. Specifying only `pr()` results in backward selection, and specifying only `pe()` results in forward selection.

`hierarchical` specifies hierarchical selection.

`lockterm1` specifies that the first term be included in the model and not be subjected to the selection criteria.

`lr` specifies that the test of term significance be the likelihood-ratio test. The default is the less computationally expensive Wald test; that is, the test is based on the estimated variance–covariance matrix of the estimators.

### Reporting

`display_options`: `noci`, `nopvalues`, `cformat(\%fmt)`, `pformat(\%fmt)`, `sformat(\%fmt)`, and `nolstretch`; see [R] Estimation options.
Remarks and examples

Remarks are presented under the following headings:

Introduction
Search logic for a step
Full search logic
Examples
Estimation sample considerations
Messages
Programming for stepwise

Introduction

Typing

```
.stepwise, pr(.10): regress y1 x1 x2 d1 d2 d3 x4 x5
```
performs a backward-selection search for the regression model y1 on x1, x2, d1, d2, d3, x4, and x5. In this search, each explanatory variable is said to be a term. Typing

```
.stepwise, pr(.10): regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```
performs a similar backward-selection search, but the variables d1, d2, and d3 are treated as one term, as are x4 and x5. That is, d1, d2, and d3 may or may not appear in the final model, but they appear or do not appear together.

Example 1

Using the automobile dataset, we fit a backward-selection model of mpg:

```
use https://www.stata-press.com/data/r16/auto
.stepwise, pr(.2): regress mpg c.weight##c.weight displ gear turn headroom
> i.foreign price
```

```
Source | SS df MS Number of obs = 74
------ | ------ ------ ------ ------
Model | 1736.31455 5 347.262911 F(5, 68) = 33.39
Residual | 707.144906 68 10.3991898 Prob > F = 0.0000
| 10.3991898 R-squared = 0.7106
| 33.4720474 Adj R-squared = 0.6893
| 3.2248 Root MSE = 3.2248
Total | 2443.45946 73 33.4720474

mpg | Coef. Std. Err. t P>|t| [95% Conf. Interval]
weight | -.0158002 .0039169 -4.03 0.000 -.0236162 -.0079842
c.weight##c.weight | 1.77e-06 6.20e-07 2.86 0.006 5.37e-07 3.01e-06
c.weight | 59.02133 9.3903 6.29 0.000 40.28327 77.75938
Foreign | -3.615107 1.260844 -2.87 0.006 -6.131082 -1.099131
gear_ratio | 2.011674 1.468831 1.37 0.175 -.9193321 4.94268
turn | -.3087038 .1763099 -1.75 0.084 -.6605248 .0431172
_cons | 59.02133 9.3903 6.29 0.000 40.28327 77.75938
```

This estimation treated each variable as its own term and thus considered each one separately. The engine displacement and gear ratio should really be considered together:
. stepwise, pr(.2): regress mpg c.weight#c.weight (displ gear) turn headroom
> i.foreign price
note: 0b.foreign dropped because of estimability
begin with full model
p = 0.7116 >= 0.2000 removing headroom
p = 0.3944 >= 0.2000 removing displacement gear_ratio
p = 0.2798 >= 0.2000 removing price

begin with full model
p = 0.7116 >= 0.2000 removing headroom
p = 0.3944 >= 0.2000 removing displacement gear_ratio
p = 0.2798 >= 0.2000 removing price

Search logic for a step

Before discussing the complete search logic, consider the logic for a step—the first step—in
detail. The other steps follow the same logic. If you type

. stepwise, pr(.20): regress y1 x1 x2 (d1 d2 d3) (x4 x5)

the logic is

1. Fit the model y on x1 x2 d1 d2 d3 x4 x5.
2. Consider dropping x1.
3. Consider dropping x2.
4. Consider dropping d1 d2 d3.
5. Consider dropping x4 x5.
6. Find the term above that is least significant. If its significance
   level is $\geq 0.20$, remove that term.

If you type

. stepwise, pr(.20) hierarchical: regress y1 x1 x2 (d1 d2 d3) (x4 x5)

the logic would be different because the hierarchical option states that the terms are ordered. The initial logic would become

1. Fit the model y on x1 x2 d1 d2 d3 x4 x5.
2. Consider dropping x4 x5—the last term.
3. If the significance of this last term is $\geq 0.20$, remove the term.

The process would then stop or continue. It would stop if x4 x5 were not dropped, and otherwise, stepwise would continue to consider the significance of the next-to-last term, d1 d2 d3.
Specifying `pe()` rather than `pr()` switches to forward estimation. If you type

```
    . stepwise, pe(.20): regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```

`stepwise` performs forward-selection search. The logic for the first step is

1. Fit a model of y on nothing (meaning a constant).
2. Consider adding x1.
3. Consider adding x2.
4. Consider adding d1 d2 d3.
5. Consider adding x4 x5.
6. Find the term above that is most significant. If its significance level is < 0.20, add that term.

As with backward estimation, if you specify `hierarchical`,

```
    . stepwise, pe(.20) hierarchical: regress y1 x1 x2 (d1 d2 d3) (x4 x5)
```

the search for the most significant term is restricted to the next term:

1. Fit a model of y on nothing (meaning a constant).
2. Consider adding x1—the first term.
3. If the significance is < 0.20, add the term.

If x1 were added, `stepwise` would next consider x2; otherwise, the search process would stop.

`stepwise` can also use a stepwise selection logic that alternates between adding and removing terms. The full logic for all the possibilities is given below.
### Full search logic

<table>
<thead>
<tr>
<th>Option</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr()</td>
<td>Fit the full model on all explanatory variables. While the least-significant term is “insignificant”, remove it and reestimate.</td>
</tr>
<tr>
<td>(backward selection)</td>
<td></td>
</tr>
<tr>
<td>pr() hierarchical</td>
<td>Fit full model on all explanatory variables. While the last term is “insignificant”, remove it and reestimate.</td>
</tr>
<tr>
<td>(backward hierarchical selection)</td>
<td></td>
</tr>
<tr>
<td>pr() pe()</td>
<td>Fit full model on all explanatory variables. If the least-significant term is “insignificant”, remove it and reestimate; otherwise, stop. Do that again: if the least-significant term is “insignificant”, remove it and reestimate; otherwise, stop. Repeatedly, if the most-significant excluded term is “significant”, add it and reestimate; if the least-significant included term is “insignificant”, remove it and reestimate; until neither is possible.</td>
</tr>
<tr>
<td>(backward stepwise)</td>
<td></td>
</tr>
<tr>
<td>pe()</td>
<td>Fit “empty” model. While the most-significant excluded term is “significant”, add it and reestimate.</td>
</tr>
<tr>
<td>(forward selection)</td>
<td></td>
</tr>
<tr>
<td>pe() hierarchical</td>
<td>Fit “empty” model. While the next term is “significant”, add it and reestimate.</td>
</tr>
<tr>
<td>(forward hierarchical selection)</td>
<td></td>
</tr>
<tr>
<td>pr() pe() forward</td>
<td>Fit “empty” model. If the most-significant excluded term is “significant”, add it and reestimate; otherwise, stop. Do that again: if the most-significant excluded term is “significant”, add it and reestimate; otherwise, stop. Repeatedly, if the least-significant included term is “insignificant”, remove it and reestimate; if the most-significant excluded term is “significant”, add it and reestimate; until neither is possible.</td>
</tr>
<tr>
<td>(forward stepwise)</td>
<td></td>
</tr>
</tbody>
</table>
Examples

The following two statements are equivalent; both include solely single-variable terms:

. stepwise, pr(.2): regress price mpg weight displ
. stepwise, pr(.2): regress price (mpg) (weight) (displ)

The following two statements are equivalent; the last term in each is r1, ..., r4:

. stepwise, pr(.2) hierarchical: regress price mpg weight displ (r1-r4)
. stepwise, pr(.2) hierarchical: regress price (mpg) (weight) (displ) (r1-r4)

To group variables weight and displ into one term, type

. stepwise, pr(.2) hierarchical: regress price mpg (weight displ) (r1-r4)

stepwise can be used with commands other than regress; for instance,

. stepwise, pr(.2): logit outcome (sex weight) treated1 treated2
. stepwise, pr(.2): logistic outcome (sex weight) treated1 treated2

Either statement would fit the same model because logistic and logit both perform logistic regression; they differ only in how they report results; see [R] logit and [R] logistic.

We use the lockterm1 option to force the first term to be included in the model. To keep treated1 and treated2 in the model no matter what, we type

. stepwise, pr(.2) lockterm1: logistic outcome (treated1 treated2) ...

After stepwise estimation, we can type stepwise without arguments to redisplay results,

. stepwise
(output from logistic appears)

or type the underlying estimation command:

. logistic
(output from logistic appears)

At estimation time, we can specify options unique to the command being stepped:

. stepwise, pr(.2): logit outcome (sex weight) treated1 treated2, or

or is logit’s option to report odds ratios rather than coefficients; see [R] logit.

Estimation sample considerations

Whether you use backward or forward estimation, stepwise forms an estimation sample by taking observations with nonmissing values of all the variables specified (except for depvar1 and depvar2 for intreg). The estimation sample is held constant throughout the stepping. Thus, if you type

. stepwise, pr(.2) hierarchical: regress amount sk edul sval

and variable sval is missing in half the data, that half of the data will not be used in the reported model, even if sval is not included in the final model.

The function e(sample) identifies the sample that was used. e(sample) contains 1 for observations used and 0 otherwise. For instance, if you type

. stepwise, pr(.2) pe(.10): logistic outcome x1 x2 (x3 x4) (x5 x6 x7)
and the final model is outcome on x1, x5, x6, and x7, you could re-create the final regression by typing

```
    . logistic outcome x1 x5 x6 x7 if e(sample)
```

You could obtain summary statistics within the estimation sample of the independent variables by typing

```
    . summarize x1 x5 x6 x7 if e(sample)
```

If you fit another model, e(sample) will automatically be redefined. Typing

```
    . stepwise, lock pr(.2): logistic outcome (x1 x2) (x3 x4) (x5 x6 x7)
```

would automatically drop e(sample) and re-create it.

**Messages**

**note: _______ dropped because of estimability**

This indicates that a variable was dropped because its coefficient could not be estimated. This message is most commonly issued because the variable is collinear with other variables in the model. For instance, say that you type

```
    . stepwise, pr(.2): regress y x1 x2 x3 x4
```

and x2 is collinear with x3, one of these variables will automatically be dropped. If you type

```
    . stepwise, pr(.2): regress y x1 x2 i.a
```

and include indicators for factor variable a in your model, the set of indicators for a are perfectly collinear, and one will be omitted with the note indicating that it was dropped because of estimability.

**note: _______ dropped because of estimability**

**note: _______ obs. dropped because of estimability**

You probably received this message in fitting a logistic or probit model. Regardless of estimation strategy, stepwise checks that the full model can be fit. The indicated variable had a 0 or infinite standard error.

For logistic, logit, and probit, this message is typically caused by one-way causation. Assume that you type

```
    . stepwise, pr(.2): logistic outcome (x1 x2 x3) d1
```

and assume that variable d1 is an indicator (dummy) variable. Further assume that whenever d1 = 1, outcome = 1 in the data. Then the coefficient on d1 is infinite. One (conservative) solution to this problem is to drop the d1 variable and the d1==1 observations. The underlying estimation commands probit, logit, and logistic report the details of the difficulty and solution; stepwise simply accumulates such problems and reports the above summary messages. Thus, if you see this message, you could type

```
    . logistic outcome x1 x2 x3 d1
```

to see the details. Although you should think carefully about such situations, Stata’s solution of dropping the offending variables and observations is, in general, appropriate.
Programming for stepwise

stepwise requires that command_name follow standard Stata syntax and allow the if qualifier; see [U] 11 Language syntax. Furthermore, command_name must have sw or swml as a program property; see [P] program properties. If command_name has swml as a property, command_name must store the log-likelihood value in e(ll) and model degrees of freedom in e(df_m).

Stored results

stepwise stores whatever is stored by the underlying estimation command. Also, stepwise stores stepwise in e(stepwise).

Methods and formulas

Some statisticians do not recommend stepwise procedures; see Sribney (1998) for a summary.

References


Also see

[R] nestreg — Nested model statistics

[LASSO] Lasso intro — Introduction to lasso