Title

stata.com

forecast coefvector — Specify an equation via a coefficient vector

Syntax

```
forecast coefvector cname [ , options ]
```

cname is a Stata matrix with one row.

options	Description
`variance(`*vname*`)`	specify parameter variance matrix
`errorvariance(`*ename*`)`	specify additive error variance matrix
`names(`*namelist*`[ , replace ])`	use *namelist* for names of left-hand-side variables

Description

`forecast coefvector` adds equations that are stored as coefficient vectors to your forecast model. Typically, equations are added using `forecast estimates` and `forecast identity`. `forecast coefvector` is used in less-common situations where you have a vector of parameters that represent a linear equation.

Most users of the `forecast` commands will not need to use `forecast coefvector`. We recommend skipping this manual entry until you are familiar with the other features of `forecast`.

Options

`variance(`*vname*`)` specifies that Stata matrix *vname* contains the variance matrix of the estimated parameters. This option only has an effect if you specify the `simulate()` option when calling `forecast solve` and request *sim_technique*'s `betas` or `residuals`. See [TS] **forecast solve**.

`errorvariance(`*ename*`)` specifies that the equations being added include an additive error term with variance *ename*, where *ename* is the name of a Stata matrix. The number of rows and columns in *ename* must match the number of equations represented by coefficient vector *cname*. This option only has an effect if you specify the `simulate()` option when calling `forecast solve` and request *sim_technique*'s `errors` or `residuals`. See [TS] **forecast solve**.

`names(`*namelist*`[ , replace ])` instructs `forecast coefvector` to use *namelist* as the names of the left-hand-side variables in the coefficient vector being added. By default, `forecast coefvector` uses the equation names on the column stripe of *cname*. You must use this option if any of the equation names stored with *cname* contains time-series operators.

Remarks and examples

stata.com

For an overview of the `forecast` commands, see [TS] **forecast**. This manual entry assumes you have already read that manual entry. This manual entry also assumes that you are familiar with Stata's matrices and the concepts of row and column names that can be attached to them; see [P] **matrix**. You use `forecast coefvector` to add endogenous variables to your model that are defined by linear equations, where the linear equations are stored in a coefficient (parameter) vector.

Remarks are presented under the following headings:

Introduction

`forecast coefvector` can be used to add equations that you obtained elsewhere to your model. For example, you might see the estimated coefficients for an equation in an article and want to add that equation to your model. User-written estimators that do not implement a `predict` command can also be included in forecast models via `forecast coefvector`. `forecast coefvector` can also be useful in situations where you want to simulate time-series data, as the next example illustrates.

▷ Example 1: A shock to an autoregressive process

Consider the following autoregressive process:

$$y_t = 0.9y_{t-1} - 0.6y_{t-2} + 0.3y_{t-3}$$

Suppose y_t is initially equal to zero. How does y_t evolve in response to a one-unit shock at time $t = 5$? We can use `forecast coefvector` to find out. First, we create a small dataset with time variable `t` and set our target variable `y` equal to zero:

```
. set obs 20
obs was 0, now 20
. generate t = _n
. tsset t
        time variable:  t, 1 to 20
                delta:  1 unit
. generate y = 0
```

Now let's think about our coefficient vector. The only tricky part is in labeling the columns. We can represent the lagged values of y_t using time-series operators; there is just one equation, corresponding to variable `y`. We can use `matrix coleq` to apply both variable and equation names to the columns of our matrix. In Stata, we type

```
. matrix y = (.9, -.6, 0.3)
. matrix coleq y = y:L.y y:L2.y y:L3.y
. matrix list y
y[1,3]
      y:   y:   y:
      L.  L2.  L3.
       y    y    y
r1   .9  -.6   .3
```

`forecast coefvector` ignores the row name of the vector being added (`r1` here), so we can leave it as is. Next we create a forecast model and add `y`:

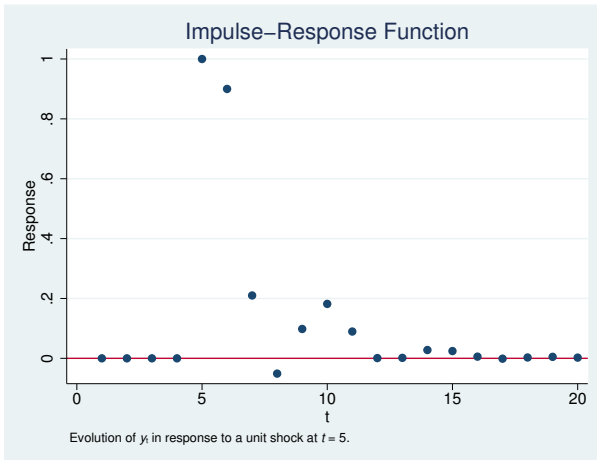
```
. forecast create
  Forecast model started.
. forecast coefvector y
  Forecast model now contains 1 endogenous variable.
```

To shock our system at $t = 5$, we can use `forecast adjust`:

```
. forecast adjust y = 1 in 5
  Endogenous variable y now has 1 adjustment.
```

Now we can solve our model. Because our `y` variable is filled in for the entire dataset, `forecast solve` will not be able to automatically determine when forecasting should commence. We have three lags in our process, so we will start at $t = 4$. To reduce the amount of output, we specify `log(off)`:

```
. forecast solve, begin(4) log(off)
Computing dynamic forecasts for current model.
------------------------------------------------------------
Starting period:  4
Ending period:    20
Forecast prefix:  f_
Forecast 1 variable spanning 17 periods.
------------------------------------------------------------
```



The graph shows our shock causing y to jump to 1 at $t = 5$. At $t = 6$, we can see that $y = 0.9$, and at $t = 7$, we can see that $y = 0.9 \times 0.9 - 0.6 \times 1 = 0.21$.

◁

The previous example used a coefficient vector representing a single equation. However, coefficient vectors can contain multiple equations. For example, say we read an article and saw the following results displayed:

$$x_t = 0.2 + 0.3x_{t-1} - 0.8z_t$$
$$z_t = 0.1 + 0.7z_{t-1} + 0.3x_t - 0.2x_{t-1}$$

We can add both equations at once to our forecast model. Again the key is in labeling the columns. `forecast coefvector` understands `_cons` to mean a constant term, and it looks at the equation names on the vector's columns to determine how many equations there are and to what endogenous variables they correspond:

```
. matrix eqvector = (0.2, 0.3, -0.8, 0.1, 0.7, 0.3, -0.2)
. matrix coleq eqvector = x:_cons x:L.x x:y y:_cons y:L.y y:x y:L.x
. matrix list eqvector
eqvector[1,7]
        x:     x:     x:     y:     y:     y:     y:
               L.                   L.           L.
    _cons      x      y  _cons      y      x      x
r1     .2     .3    -.8     .1     .7     .3    -.2
```

We could then type

```
. forecast coefvector y
```

to add our coefficient vector to a model.

Just like with estimation results whose left-hand-side variables contain time-series operators, if any of the equation names of the coefficient vector being added contains time-series operators, you must use the `names()` option of `forecast coefvector` to specify alternative names.

Simulations with coefficient vectors

The `forecast solve` command provides the option `simulate(`*sim_technique*`, ...)` to perform stochastic simulations and obtain measures of forecast uncertainty. How `forecast solve` handles coefficient vectors when performing these simulations depends on the options provided with `forecast coefvector`. There are four cases to consider:

1. You specify neither `variance()` nor `errorvariance()` with `forecast coefvector`. You have provided no measures of uncertainty with this coefficient vector. Therefore, `forecast solve` treats it like an identity. No random errors or residuals are added to this coefficient vector's linear combination, nor are the coefficients perturbed in any way.

2. You specify `variance()` but not `errorvariance()`. The `variance()` option provides the covariance matrix of the estimated parameters in the coefficient vector. Therefore, the coefficient vector is taken to be stochastic. If you request *sim_technique* `betas`, this coefficient vector is assumed to be distributed multivariate normal with a mean equal to the original value of the vector and covariance matrix as specified in the `variance()` option, and random draws are taken from this distribution. If you request *sim_technique* `residuals`, randomly chosen static residuals are added to this coefficient vector's linear combination. Because you did not specify a covariance matrix for the error terms with the `errorvariance()` option, *sim_technique* `errors` cannot draw random errors for this coefficient vector's linear combination, so *sim_technique* `errors` has no impact on the equations.

3. You specify `errorvariance()` but not `variance()`. Because you specified a covariance matrix for the assumed additive error term, the equations represented by this coefficient vector are stochastic. If you request *sim_technique* `residuals`, randomly chosen static residuals are added to this coefficient vector's linear combination. If you request *sim_technique* `errors`, multivariate normal errors with mean zero and covariance matrix as specified in the `errorvariance()` option are added during the simulations. However, specifying *sim_technique* `betas` does not affect the equations because there is no covariance matrix associated with the coefficients.

4. You specify both `variance()` and `errorvariance()`. The equations represented by this coefficient vector are stochastic, and `forecast solve` treats the coefficient vector just like an estimation result. *sim_technique*'s `betas`, `residuals`, and `errors` all work as expected.

Methods and formulas

Let β denote the $1 \times k$ coefficient vector being added. Then the matrix specified in the `variance()` option must be $k \times k$. Row and column names for that matrix are ignored.

Let m denote the number of equations represented by β. That is, if β is stored as Stata matrix `beta` and local macro `m` is to hold the number of equations, then in Stata parlance,

```
. local eqnames : coleq beta
. local eq : list uniq eqnames
. local m : list sizeof eq
```

Then the matrix specified in the `errorvariance` option must be $m \times m$. Row and column names for that matrix are ignored.

Also see

[TS] **forecast** — Econometric model forecasting

[TS] **forecast solve** — Obtain static and dynamic forecasts

[P] **matrix** — Introduction to matrix commands

[P] **matrix rownames** — Name rows and columns