forecast estimates — Add estimation results to a forecast model

Description Quick start Menu Syntax Options Remarks and examples
References Also see

Description

`forecast estimates` adds estimation results to the forecast model currently in memory. You must first create a new model using `forecast create` before you can add estimation results with `forecast estimates`. After estimating the parameters of an equation or set of equations, you must use `estimates store` to store the estimation results in memory or use `estimates save` to save them on disk before adding them to the model.

Quick start

Add estimation results stored in `myestimates` to the forecast model in memory

```
forecast estimates myestimates
```

Same as above, but specify the prediction produced by `predict, pr outcome(#1)`

```
forecast estimates myestimates, predict("pr outcome(#1)")
```

Add estimates from `var` estimation stored in memory as `varest`

```
forecast estimates varest
```

Also add the second estimation result saved on disk as `notcurrent.ster` to the forecast model

```
forecast estimates using notcurrent, number(2)
```

Menu

Statistics > Time series > Forecasting

Syntax

Add estimation result currently in memory to model

`forecast estimates name [ , options ]`

name is the name of a stored estimation result; see [R] **estimates store**.

Add estimation result currently saved on disk to model

`forecast estimates using filename [ , number(#) options ]`

filename is an estimation results file created by `estimates save`; see [R] **estimates save**. If no file extension is specified, `.ster` is assumed.

options	Description
`predict(`*p_options*`)`	call `predict` using *p_options*
`names(`*namelist*[, `replace`]`)`	use *namelist* for names of left-hand-side variables
`advise`	advise whether estimation results can be dropped from memory

Options

`predict(`*p_options*`)` specifies the `predict` options to use when predicting the dependent variables. For a single-equation estimation command, you simply specify the appropriate options to pass to `predict`. If multiple options are required, enclose them in quotation marks:

```
. forecast estimates ..., predict("pr outcome(#1)")
```

For a multiple-equation estimation command, you can either specify one set of options that will be applied to all equations or specify *p* options, where *p* is the number of endogenous variables being added. If multiple options are required for each equation, enclose each equation's options in quotes:

```
. forecast estimates ..., predict("pr eq(#1)" "pr eq(#2)")
```

If you do not specify the `eq()` option for any of the equations, `forecast` automatically includes it for you.

If you are adding results from a linear estimation command that `forecast` recognizes as one whose predictions can be calculated as $\mathbf{x}_t'\boldsymbol{\beta}$, do not specify the `predict()` option, because this will slow `forecast`'s computation time substantially. Use the `advise` option to determine whether `forecast` needs to call `predict`.

If you do not specify any `predict` options, `forecast` uses the default type of prediction for the command whose results are being added.

`names(`*namelist*[, `replace`]`)` instructs `forecast estimates` to use *namelist* as the names of the left-hand-side variables in the estimation result being added. You must use this option if any of the left-hand-side variables contains time-series operators. By default, `forecast estimates` uses the names stored in the `e(depvar)` macro of the results being added.

`forecast estimates` creates a new variable in the dataset for each element of *namelist*. If a variable of the same name already exists in your dataset, `forecast estimates` exits with an error unless you specify the `replace` option, in which case existing variables are overwritten.

`advise` requests that `forecast estimates` report a message indicating whether the estimation results being added can be removed from memory. This option is useful if you expect your model to contain more than 300 sets of estimation results, the maximum number that Stata allows you to store in memory; see [R] **Limits**. This option also provides an indication of the speed with which the model can be solved: `forecast` executes much more slowly with estimation results that must remain in memory.

`number(#)`, for use with `forecast estimates using`, specifies that the #th set of estimation results from *filename* be loaded. This assumes that multiple sets of estimation results have been saved in *filename*. The default is `number(1)`. See [R] **estimates save** for more information on saving multiple sets of estimation results in a single file.

Remarks and examples

For an overview of the `forecast` commands, see [TS] **forecast**. This manual entry assumes you have already read that manual entry. `forecast estimates` adds stochastic equations previously fit by Stata estimation commands to a forecast model.

Remarks are presented under the following headings:

Introduction
The advise option
Using saved estimation results
The predict option
Forecasting with ARIMA models

Introduction

After you fit an equation that will become a part of your model, you must use either `estimates store` to store the estimation results in memory or `estimates save` to save the estimation results to disk. Then you can use `forecast estimates` to add that equation to your model.

We usually refer to "equation" in the singular, but of course, you can also use a multiple-equation estimation command to fit several equations at once and add them to the model. When we discuss adding a stochastic equation to a model, we really mean adding a single estimation result.

In this discussion, we also need to make a distinction between making a forecast and obtaining a prediction. We use the word "predict" to refer to the process of obtaining a fitted value for a single equation, just as you can use the `predict` command to obtain fitted values, residuals, or other statistics after fitting a model with an estimation command. We use the word "forecast" to mean finding a solution to the complete set of equations that compose the forecast model. The iterative techniques we use to solve the model and produce forecasts require that we be able to obtain predictions from each of the equations in the model.

▷ Example 1: A simple example

Here we illustrate how to add estimation results from a regression model in which none of the left-hand-side variables contains time-series operators or mathematical transformations. We use `quietly` with the estimation command because the output is not relevant here. We type

```
. use https://www.stata-press.com/data/r19/klein2
. quietly reg3 (c p L.p w) (i p L.p L.k) (wp y L.y yr), endog(w p y) exog(t wg g)
. estimates store klein
. forecast create kleinmodel
  Forecast model kleinmodel started.
. forecast estimates klein
  Added estimation results from reg3.
  Forecast model kleinmodel now contains 3 endogenous variables.
```

`forecast estimates` indicated that three endogenous variables were added to the forecast model. That is because we specified three equations in our call to `reg3`. As we mentioned in example 1 in [TS] **forecast**, the `endog()` option of `reg3` has no bearing on `forecast`. All that matters are the three left-hand-side variables.

◁

❑ Technical note

When you add an estimation result to your forecast model, `forecast` looks at the macro `e(depvar)` to determine the endogenous variables being added. If that macro is empty, `forecast` tries a few other macros to account for nonstandard commands. The number of endogenous variables being added to the model is based on the number of words found in the macro containing the dependent variables.

❑

You can fit equations with the `D.` and `S.` first- and seasonal-difference time-series operators adorning the left-hand-side variables, but in those cases, when you add the equations to the model, you must use the `names()` option of `forecast estimates`. When you specify `names(`*namelist*`)`, `forecast estimates` uses *namelist* as the names of the newly declared endogenous variables and ignores what is in `e(depvar)`. Moreover, `forecast` does not automatically "undo" the operators on left-hand-side variables. For example, you might fit a regression with `D.x` as the regressand and then add it to the model using `forecast estimates ..., name(Dx)`. In that case, `forecast` will solve the model in terms of `Dx`. You must add an identity to convert `Dx` to the corresponding level variable `x`, as the next example illustrates.

Of course, you are free to use the `D.`, `S.`, and `L.` time-series operators on endogenous variables when they appear on the right-hand sides of equations. It is only when `D.` or `S.` appears on the left-hand side that you must use the `names()` option to provide alternative names for them. You cannot add equations to models for which the `L.` operator appears on left-hand-side variables. You cannot use the `F.` forward operator anywhere in forecast models.

▷ Example 2: Differenced and log-transformed dependent variables

Consider the following model:

$$\texttt{D.logC} = \beta_{10} + \beta_{11}\texttt{D.logW} + \beta_{12}\texttt{D.logY} + u_{1t} \qquad (1)$$

$$\texttt{logW} = \beta_{20} + \beta_{21}\texttt{L.logW} + \beta_{22}\texttt{M} + \beta_{23}\texttt{logY} + \beta_{24}\texttt{logC} + u_{2t} \qquad (2)$$

Here `logY` and `M` are exogenous variables, so we will assume they are filled in over the forecast horizon before solving the model. Ultimately, we are interested in forecasting `C` and `W`. However, the first equation is specified in terms of changes in the logarithm of `C`, and the second equation is specified in terms of the logarithm of `W`.

We will refer to variables and transformations like logC, D.logC, and C as "related" variables because they are related to one another by simple mathematical functions. Including the related variables, we in fact have a five-equation model with two stochastic equations and three identities:

$$\begin{aligned}
\texttt{dlogC} &= \beta_{10} + \beta_{11}\texttt{D.logW} + \beta_{12}\texttt{D.logY} + u_{1t} \\
\texttt{logC} &= \texttt{L.logC + dlogC} \\
\texttt{C} &= \exp(\texttt{logC}) \\
\texttt{logW} &= \beta_{20} + \beta_{21}\texttt{L.logW} + \beta_{22}\texttt{M} + \beta_{23}\texttt{logY} + \beta_{24}\texttt{logC} + u_{2t} \\
\texttt{W} &= \exp(\texttt{logW})
\end{aligned}$$

To fit (1) and (2) in Stata and create a forecast model, we type

```
. use https://www.stata-press.com/data/r19/fcestimates, clear
. quietly regress D.logC D.logW D.logY
. estimates store dlogceq
. quietly regress logW L.logW M logY logC
. estimates store logweq
. forecast create cwmodel, replace
  (Forecast model kleinmodel ended.)
  Forecast model cwmodel started.
. forecast estimates dlogceq, names(dlogC)
  Added estimation results from regress.
  Forecast model cwmodel now contains 1 endogenous variable.
. forecast identity logC = L.logC + dlogC
  Forecast model cwmodel now contains 2 endogenous variables.
. forecast identity C = exp(logC)
  Forecast model cwmodel now contains 3 endogenous variables.
. forecast estimates logweq
  Added estimation results from regress.
  Forecast model cwmodel now contains 4 endogenous variables.
. forecast identity W = exp(logW)
  Forecast model cwmodel now contains 5 endogenous variables.
```

Because the left-hand-side variable in (1) contains a time-series operator, we had to use the names() option of forecast estimates when adding that equation's estimation results to our forecast model. Here we named this endogenous variable dlogC. We then added the other four equations to our model. In general, when we have a set of related variables, we prefer to specify the identities right after we add the stochastic equation so that we do not forget about them.

◁

❑ Technical note

In the previous example, we "undid" the log-transformations by simply exponentiating the logarithmic variable. However, that is only an approximation that does not work well in many applications. Suppose we fit the linear regression model

$$\ln y_t = \mathbf{x}_t'\boldsymbol{\beta} + u_t$$

where u_t is a zero-mean regression error term. Then $E(y_t|\mathbf{x}_t) = \exp(\mathbf{x}_t'\boldsymbol{\beta}) \times E\{\exp(u_t)\}$. Although $E(u_t) = 0$, Jensen's inequality suggests that $E\{\exp(u_t)\} \neq 1$, implying that we cannot predict y_t by simply taking the exponential of the linear prediction $\mathbf{x}_t'\boldsymbol{\beta}$.

If we assume that $u_t \sim N(0, \sigma^2)$, then $E\{\exp(u_t)\} = \exp(\sigma^2/2)$. Moreover, many estimation commands like `regress` provide an estimate $\hat{\sigma}^2$ of σ^2, so for regression models that contain a logarithmic dependent variable, we can obtain better forecasts for the dependent variable in levels if we approximate $E\{\exp(u_t)\}$ as $\exp(\hat{\sigma}^2/2)$. Suppose we run the regression

```
    . regress lny x1 x2 x3
    . estimates store myreg
```

then we could add `lny` and `y` as endogenous variables like this:

```
    . forecast estimates lny
    . forecast identity y = exp(lny)*`=e(rmse)^2 / 2'
```

In the second command, Stata will first evaluate the expression `` `=e(rmse)^2 / 2' `` and replace it with its numerical value. After `regress`, the macro `e(rmse)` contains the square root of the estimate of $\hat{\sigma}^2$, so the value of this expression will be our estimate of $E\{\exp(u_t)\}$. Then `forecast` will forecast `y` as the product of this number and exp(`lny`). Here we had to use a macro expression including an equals sign to force Stata to evaluate the expression immediately and obtain the expression's value. Identities are not associated with estimation results, so as soon as we used another estimation command or restored some other estimation results (perhaps unknowingly by invoking `forecast solve`), our reference to `e(rmse)` would no longer be meaningful. See **[U] 18.3.8 Macro expressions** for more information on macro evaluation.

Another alternative would be to use Duan's (1983) smearing technique. Stata code for this is provided in Cameron and Trivedi (2022).

A third alternative is to use the generalized linear model (GLM) as implemented by the `glm` command with a log-link function. In a GLM framework, we would be modeling $\ln\{E(y_t)\}$ rather than $E\{\ln(y_t)\}$ because we would be using `regress`, but oftentimes, the two quantities are similar. Moreover, obtaining predicted values for y_t in the GLM does not present the transformation problem as happens with linear regression. The `forecast` commands contain special code to handle estimation results obtained by using `glm` with the `link(log)` option, and you do not need to specify an identity to obtain `y` as a function of `lny`. All you would need to do is

```
    . glm y x1 x2 x3, link(log)
    . estimates store myglm
    . forecast estimates myglm
```

❑

The advise option

To produce forecasts from your model, `forecast` must be able to obtain predictions for each estimation result that you have added. For many of the most commonly used estimation commands such as `regress`, `ivregress`, and `var`, `forecast` includes special code to quickly obtain these predictions. For estimation commands that either require more involved computations to obtain predictions or are not widely used in forecasting, `forecast` instead relies on the `predict` command to obtain predictions.

The `advise` option of `forecast estimates` advises you as to whether `forecast` includes the special code to obtain fast predictions for the command whose estimation results are being added to the model. For example, here we use `advise` with `forecast estimates` when building the Klein (1950) model.

▷ Example 3: Using the advise option

```
. use https://www.stata-press.com/data/r19/klein2, clear
. quietly reg3 (c p L.p w) (i p L.p L.k) (wp y L.y yr), endog(w p y) exog(t wg g)
. estimates store klein
. forecast create kleinmodel, replace
  (Forecast model cwmodel ended.)
  Forecast model kleinmodel started.
. forecast estimates klein, advise
  (These estimation results are no longer needed; you can drop them.)
  Added estimation results from reg3.
  Forecast model kleinmodel now contains 3 endogenous variables.
```

After we typed `forecast estimates`, Stata advised us that "[t]hese estimation results are no longer needed; you can drop them". That means `forecast` includes code to obtain predictions from `reg3` without having to call `predict`. `forecast` has recorded all the information it needs about the estimation results stored in `klein`, and we could type

```
. estimates drop klein
```

to remove those estimates from memory.

◁

For relatively small models, there is no need to use `estimates drop` to remove estimation results from memory. However, Stata allows no more than 300 sets of estimation results to be in memory at once, and `forecast solve` requires estimation results to be in memory (and not merely saved on disk) before it can produce forecasts. For very large models in which that limit may bind, you can use the `advise` option to determine which estimation results are needed to solve the model and which can be dropped.

Suppose we had estimation results from a command for which `forecast` must call `predict` to obtain predictions. Then instead of obtaining the note saying the estimation results were no longer needed, we would obtain a note stating

```
. forecast estimates IUsePredict
  (These estimation results are needed to solve the model.)
```

In that case, the estimation results would need to be in memory before calling `forecast solve`.

The `advise` option also provides an indication of how quickly forecasts can be produced from the model. Models for which `forecast` never needs to call `predict` can be solved much more quickly than models that include equations for which `forecast` must restore estimation results and call `predict` to obtain predictions.

Using saved estimation results

Stata's `estimates` commands allow you to save estimation results to disk so that they are available in subsequent Stata sessions. You can use the `using` option of `forecast estimates` to use estimation results saved on disk without having to first call `estimates use`. In fact, `estimates use` can even retrieve estimation results stored on a website, as the next example demonstrates.

▷ Example 4: Adding saved estimation results

The file `klein.ster` contains the estimation results produced by `reg3` for the three stochastic equations of Klein's (1950) model. That file is stored on the Stata Press website in the same location as the example datasets. Here we create a forecast model and add those results:

```
. use https://www.stata-press.com/data/r19/klein2
. forecast create example4, replace
  (Forecast model kleinmodel ended.)
  Forecast model example4 started.
. forecast estimates using https://www.stata-press.com/data/r19/klein
  Added estimation results from reg3.
  Forecast model example4 now contains 3 endogenous variables.
```

If you do not specify a file extension, `forecast estimates` assumes the file ends in `.ster`. You are more likely to save your estimation results on your computer's disk drive rather than a web server, but in either case, this example shows that you can fit equations in one session of Stata, save the results to disk, and then build your forecast model later.

◁

The `estimates save` command allows you to save multiple estimation results to the same file and numbers them sequentially starting at 1. You can use the `number()` option of `forecast estimates using` to specify which set of estimation results from the specified file you wish to add to the forecast model. If you do not specify `number()`, `forecast estimates using` uses the first set of results.

When you use `forecast estimates using`, `forecast` loads the estimation results from disk and stores them in memory using a temporary name. Later, when you proceed to solve your model, `forecast` checks to see whether those estimation results are still in memory. If not, it will attempt to reload them from the file you had specified. You should therefore not move or rename estimation result files between the time you add them to your model and the time you solve the model.

The predict option

As we mentioned while discussing the `advise` option, the `forecast` commands include code to quickly obtain predictions from some of the most commonly used commands, while they use `predict` to obtain predictions from other estimation commands. When you add estimation results that require `forecast` to use `predict`, by default, `forecast` assumes that it can pass the option `xb` on to `predict` to obtain the appropriate predicted values. You use the `predict()` option of `forecast estimates` to specify the option that `predict` must use to obtain predicted values from the estimates being added.

For example, suppose you used `tobit` to fit an equation whose dependent variable is left-censored at zero and then stored the estimation results under the name `tobitreg`. When solving the model, you want to use the predicted values of the left-truncated mean, the expected value of the dependent variable conditional on its being greater than zero. Looking at the *Syntax for predict* in [R] **tobit postestimation**, we see that the appropriate option we must pass to `predict` is `e(0,.)`. To add this estimation result to an existing forecast model, we would therefore type

```
. forecast estimates tobitreg, predict(e(0,.))
```

Now, whenever `forecast` calls `predict` with those estimation results, it will pass the option `e(0,.)` so that we obtain the appropriate predictions. If you are adding results from a multiple-equation estimation command with k dependent variables, then you must specify k `predict` options within the `predict()` option, separated by spaces.

Forecasting with ARIMA models

Practitioners often use ARIMA models to forecast some of the variables in their models, and you can certainly use estimation results produced by commands such as `arima` with `forecast`. There are just two rules to follow when using commands that use the Kalman filter to obtain predictions. First, do not specify the `predict()` option with `forecast estimates`. The `forecast` commands know how to handle these estimators automatically. Second, as we stated earlier, the `forecast` commands do not "undo" any time-series operators that may adorn the left-hand-side variables of estimation results, so you must use `forecast identity` to specify identities to recover the underlying variables in levels.

▷ Example 5: An ARIMA model with first- and seasonal-differencing

`wpi1.dta` contains quarterly observations on the variable `wpi`. First, let's fit a multiplicative seasonal ARIMA model with both first- and seasonal-difference operators applied to the dependent variable and store the estimation results:

```
. use https://www.stata-press.com/data/r19/wpi1
. arima wpi, arima(1, 1, 1) sarima(1, 1, 1, 4)
  (output omitted)
. estimates store arima
```

(For details on fitting seasonal ARIMA models, see [TS] **arima**).

With the difference operators used here, when `forecast` calls `predict`, it will obtain predictions in terms of `DS4.wpi`. Using the definitions of time-series operators in [TS] **tsset**, we have

$$\texttt{DS4.wpi}_t = (\texttt{wpi}_t - \texttt{wpi}_{t-4}) - (\texttt{wpi}_{t-1} - \texttt{wpi}_{t-5})$$

so that

$$\texttt{wpi}_t = \texttt{DS4.wpi}_t + \texttt{wpi}_{t-4} + (\texttt{wpi}_{t-1} - \texttt{wpi}_{t-5})$$

Because our `arima` results include a dependent variable with time-series operators, we must use the `name()` option of `forecast estimates` to specify an alternative variable name. We will name ours `ds4wpi`. Then we can specify an identity by using the previous equation to recover our forecasts in terms of `wpi`. We type

```
. forecast create arimaexample, replace
  (Forecast model example4 ended.)
  Forecast model arimaexample started.
. forecast estimates arima, name(ds4wpi)
  Added estimation results from arima.
  Forecast model arimaexample now contains 1 endogenous variable.
. forecast identity wpi = ds4wpi + L4.wpi + (L.wpi - L5.wpi)
  Forecast model arimaexample now contains 2 endogenous variables.
. forecast solve, begin(tq(1988q1))
Computing dynamic forecasts for model arimaexample.
───────────────────────────────────────────────────
Starting period:  1988q1
Ending period:    1990q4
Forecast prefix:  f_

1988q1:  .............
1988q2:  ...............
1988q3:  ...............
  (output omitted)
1990q4:  ............

Forecast 2 variables spanning 12 periods.
─────────────────────────────────────────
```

Because our entire forecast model consists of a single equation fit by `arima`, we can also call `predict` to obtain forecasts:

```
. predict a_wpi, y dynamic(tq(1988q1))
(5 missing values generated)

. list t f_wpi a_wpi in -5/l

      +------------------------------+
      |      t      f_wpi      a_wpi |
      |------------------------------|
120.  | 1989q4   110.2182   110.2182 |
121.  | 1990q1   111.6782   111.6782 |
122.  | 1990q2   112.9945   112.9945 |
123.  | 1990q3   114.3281   114.3281 |
124.  | 1990q4   115.5142   115.5142 |
      +------------------------------+
```

Looking at the last few observations in the dataset, we see that the forecasts produced by `forecast` (`f_wpi`) match those produced by `predict` (`a_wpi`). Of course, the advantage of `forecast` is that we can combine multiple sets of estimation results and obtain forecasts for an entire system of equations.

◁

❑ Technical note

Do not add estimation results to your forecast model that you have stored after calling an estimation command with the `by:` prefix. The stored estimation results will contain information from only the last group on which the estimation command was executed. `forecast` will then use those results for all observations in the forecast horizon regardless of the value of the group variable you specified with `by:`.

❑

References

Cameron, A. C., and P. K. Trivedi. 2022. *Microeconometrics Using Stata*. 2nd ed. College Station, TX: Stata Press.

Duan, N. 1983. Smearing estimate: A nonparametric retransformation method. *Journal of the American Statistical Association* 78: 605–610. https://doi.org/10.1080/01621459.1983.10478017.

Klein, L. R. 1950. *Economic Fluctuations in the United States 1921–1941*. New York: Wiley.

Also see

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

[R] **estimates** — Save and manipulate estimation results

[R] **predict** — Obtain predictions, residuals, etc., after estimation