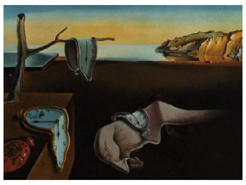

Journal of Statistical Software

May 2011, Volume 41, Issue 10. *http://www.jstatsoft.org/*

State Space Methods in Stata

David M. Drukker
Stata

Richard B. Gates
Stata

Abstract

We illustrate how to estimate parameters of linear state-space models using the Stata program `sspace`. We provide examples of how to use `sspace` to estimate the parameters of unobserved-component models, vector autoregressive moving-average models, and dynamic-factor models. We also show how to compute one-step, filtered, and smoothed estimates of the series and the states; dynamic forecasts and their confidence intervals; and residuals.

Keywords: state-space, unobserved-components models, local-level model, local-linear-trend model, basic structural model, dynamic-factor model, vector autoregressive moving-average model, `sspace`.

1. Introduction

Stata is a general purpose package for statistics, graphics, data management, and matrix language programming. Stata's coverage of statistical areas is one of the most complete available, with many commands for regression analysis (StataCorp 2009k,l,m), multivariate statistics (StataCorp 2009i), panel-data analysis (StataCorp 2009h), survey data analysis (StataCorp 2009n), survival analysis and epidemiology statistics (StataCorp 2009o), and time-series analysis (StataCorp 2009p). It is used for data management (Mitchell 2010), health research (Juul and Frydenberg 2010; Cleves, Gould, Gutierrez, and Marchenko 2010), as well as in economic analysis (Cameron and Trivedi 2009; Baum 2006). Stata is also a programming language used by researchers to implement and disseminate their methods; see any of the more than 40 issues of *The Stata Journal* for examples of peer-reviewed user-written programs and see StataCorp (2009j,f,g) for Stata's programming capabilities.

The Stata command `sspace`, released in version 11, estimates the parameters of linear state-space models by maximum likelihood (StataCorp 2009e). As demonstrated by Harvey (1989) and Commandeur, Koopman, and Ooms (2011), linear state-space models are very flexible, and many linear time-series models can be written as linear state-space models. In this article, we show how to use `sspace` to estimate the parameters of linear state-space models. We also note that Stata has some additional commands, such as `dfactor`, which provide simpler syntaxes for estimating the parameters of particular linear state-space models.

Because of this flexibility, `sspace` has two syntaxes; we call them the covariance-form syntax and the error-form syntax. They are illustrated by estimating the parameters of a local-linear-trend model with a seasonal component and a vector autoregressive moving-average (VARMA) model, respectively. In each syntax, the user must specify one or more state equations, one or more observation equations, and the stochastic components.

2. Case 1: The local-level model

The local-level model is described by Commandeur *et al.* (2011, Section 2.1) and we briefly review it here. The observation and state equations of this model are

$$\begin{aligned} y_t &= \mu_t + \epsilon_t, \\ \mu_t &= \mu_{t-1} + \xi_t, \end{aligned} \tag{1}$$

respectively, where $\epsilon_t \sim N(0, \sigma^2_\epsilon)$ and $\xi_t \sim N(0, \sigma^2_\xi)$ and both are independent. We express the level component at time t, μ_t, as a function of that at time $t-1$. This notation is a subtle change from that in Commandeur *et al.* (2011), but it is more consistent with the syntax of Stata's `sspace` for describing the model and how `sspace` executes the state-space recursions by starting with index 0 instead of 1. The parameters in this model are σ^2_ϵ, σ^2_ξ, and μ_0.

2.1. Covariance-form syntax

The covariance-form syntax of `sspace` is as follows:

```
sspace state_eq [state_eq ...  state_eq]
       obs_eq [obs_eq ...  obs_eq] [if] [in] [, options]
```

where `state_eq` are state equations of the form

```
(statevar [lagged_statevars] [indepvars], state [noerror noconstant
                                  covstate(covform)])
```

and `obs_eq` are observation equations of the form

```
(depvar [statevars] [indepvars] [, noerror noconstant
                                  covobserved(covform)])
```

A list of state equations, observation equations, and options specifies an `sspace` model. The square brackets indicate optional arguments, so the syntax diagram indicates that at least one state equation and one observation equation are required. Each equation must be enclosed in parentheses. In Stata parlance, a comma in the command toggles the parser from model specification mode to options specification mode. Options included within an equation are applied to that equation. Options specified outside the individual equations are applied to the model as a whole.

Each state equation specifies the name of a latent variable and must have the `state` option specified. A state equation optionally contains a list of lagged state variables and a list exogenous covariates. By default, a constant is included in the equation unless the `noconstant` option is specified. By default, an error term is included in the equation unless the `noerror` option is specified. The option `covstate()` allows you to specify the covariance structure of the state equations. The `covform` in the syntax diagram may be `identity`, `dscalar`, `diagonal`, or `unstructured`. The default is `diagonal`. The option `dscalar` states that the covariance is diagonal and that all the variance terms are equal.

Each observation equation specifies the name of an observed dependent variable. An observation equation optionally contains a list of contemporaneous state variables and a list exogenous covariates. By default, a constant is included in the equation unless the `noconstant` option is specified. By default, an error term is included in the equation unless the `noerror` option is specified. The option `covobserved()` allows you to specify the covariance structure of the observation equations. The covariance forms are the same as the option `covstate()`.

The `[if]` and the `[in]` specifications allow you to estimate the parameters using a subsample of the observations.

The `options` in the main syntax diagram include model, optimization, and display options. An important model option is `constraints()`, parameter constraints that identify the model. A popular optimization option is the `technique()` option. Two good techniques for `sspace` are `technique(BHHH)`, or the Berndt-Hall-Hall-Hausman technique; and the `technique(NR)`, for Newton-Raphson. Optimization techniques may be mixed; such is the default, `technique (BHHH 5 NR)`, which specifies the BHHH method for the first 5 iterations and NR for the remaining iterations. An example of a display option is `level()`, which allows you to set the confidence level to something other than the default of 95%.

We clarify this syntax in the following example.

2.2. Estimating the variances of a local-level model using `sspace`

Here we illustrate the `sspace` syntax by estimating the parameters of the local-level model on the well-known Nile dataset containing observations on the annual Nile River flow volume at Aswan, Egypt, from 1870 to 1970. The Stata command `use` loads the dataset into memory and the command `describe` describes it.

```
. use http://www.stata.com/ddrukker/nile.dta

(Nile river annual flow volume at Aswan from 1870 to 1970)
```

The `describe` command will display a dataset's size, its variables, their storage type and format, any labels associated with the variables, sorting information, and any descriptive information that you have added to document your data.

```
. describe

Contains data from data/nile.dta
  obs:           100                          Nile river annual flow volume
                                                at Aswan from 1870 to 1970
 vars:             2                          16 Jun 2008 10:49
 size:         1,200 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
AFV             long   %12.0g                 Annual Flow Volume
year            long   %ty
-------------------------------------------------------------------------------
Sorted by:  year
```

Stata computes time-series operators of variables using a time variable specified by the `tsset` command. Below we specify `year` to be our time variable; we `tsset` the data, in Stata parlance.

```
. tsset year


        time variable:  year, 1871 to 1970
                delta:  1 year
```

We could now use `sspace` to estimate the parameters using the code

```
        constraint define 1 [level]L.level = 1
        constraint define 2 [AFV]level     = 1
        sspace  (level L.level, state noconstant) ///
                (AFV level, noconstant),          ///
                constraints(1 2)
```

While this code is transparent to Stata users, we discuss it in some detail for readers who are unaccustomed to Stata.

The first two lines define constraints on the model parameters, as discussed below. The third line begins with the command `sspace` and is followed by the definition of the state equation

```
        (level L.level, state noconstant)
```

which is best understood from right to left. The option `noconstant` specifies that there is no constant term in the equation; the option `state` specifies the equation as a state equation; and the comma separates the options from equation specification. By specifying the equation as `level L.level`, we specify `level` as the name for the unobserved state and we specify that the state equation is

$$\texttt{level}_t = \alpha\texttt{level}_{t-1}$$

We use Stata's lag operator, `L.` in this example, to model `level` as a linear function of the lagged `level`.

At the end of third line, the three slashes, `///`, denote a line continuation in Stata. In this example, we see that lines 3, 4, and 5 compose a single Stata command.

The fourth line specifies that the observation equation in the model is

$$\texttt{AFV}_t = \beta\texttt{level}_t + \epsilon_t$$

where the ϵ_t are independent and identically distributed (IID) normal errors. As in the state equation above, we used the `noconstant` option to suppress the constant term.

The model in Equation (1) requires that $\alpha = \beta = 1$. Lines 1 and 2 declare these constraints; on line 4, the option `constraints(1 2)` applies them to this model.

Repeating the code, we proceed with estimation:

```
. constraint define 1 [level]L.level = 1
. constraint define 2 [AFV]level     = 1
. sspace  (level L.level, state noconstant) ///
>         (AFV level, noconstant),          ///
>         constraints(1 2)
```

```
searching for initial values ...
(setting technique to bhhh)
Iteration 0:   log likelihood = -635.14379
Iteration 1:   log likelihood =  -633.9615
Iteration 2:   log likelihood = -633.60088
Iteration 3:   log likelihood = -633.57318
Iteration 4:   log likelihood = -633.54533
(switching technique to nr)
Iteration 5:   log likelihood = -633.51888
Iteration 6:   log likelihood = -633.46465
Iteration 7:   log likelihood = -633.46456
Iteration 8:   log likelihood = -633.46456
Refining estimates:
Iteration 0:   log likelihood = -633.46456
Iteration 1:   log likelihood = -633.46456

State-space model

Sample: 1871 - 1970                             Number of obs   =        100
Log likelihood = -633.46456
 ( 1)  [level]L.level = 1
 ( 2)  [AFV]level = 1
------------------------------------------------------------------------------
             |                 OIM
         AFV |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
level        |
       level |
         L1. |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
AFV          |
       level |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
var(level)   |   1469.176   1280.375     1.15   0.251    -1040.313    3978.666
var(AFV)     |   15098.52   3145.548     4.80   0.000     8933.358    21263.68
------------------------------------------------------------------------------
Note: Model is not stationary.
Note: Tests of variances against zero are conservative and are provided only
      for reference.
```

`e()` result name	Commandeur *et al.* notation
`e(A)`	$\mathbf{T}$
`e(B)`	
`e(C)`	$\mathbf{R}$
`e(chol_Q)`	$\mathbf{Q}^{1/2}$
`e(D)`	$\mathbf{Z}$
`e(F)`	
`e(G)`	
`e(chol_R)`	$\mathbf{H}^{1/2}$

Table 1: Kalman filter matrices in Stata's `e()` results and their Commandeur *et al.* (2011) equivalents.

The output table reports that `sspace` estimates σ^2_ξ to be 1,469.2 and σ^2_ϵ to be 15,098.5.

Having provided a simple example of how to use `sspace`, we now provide some technical details about its implementation. `sspace` uses the Mata optimizer `optimize()` (StataCorp 2009c). `sspace` uses analytic first derivatives, from which it numerically computes the second order derivatives necessary for Newton-Raphson optimization. If you are using the multiprocessor version of Stata (Stata MP), the numerical second derivatives are computed in parallel.

`optimize()` will not declare convergence until the length of the scaled gradient is smaller than 10^{-6}. That is when $\mathbf{g}_k^T \widehat{\boldsymbol{\mathcal{H}}}_k^{-1} \mathbf{g}_k < 10^{-6}$, where $\mathbf{g}_k$ is the gradient on the k-th step and $\widehat{\boldsymbol{\mathcal{H}}}_k$ is the approximated negative Hessian. The requirement that $\widehat{\boldsymbol{\mathcal{H}}}_k$ be nonsingular prevents `sspace` from declaring convergence when the parameters are not identified, as discussed in Drukker and Wiggins (2004).

The standard errors are computed from the negative Hessian unless the variance-covariance option, `vce()`, specifies otherwise. The `OIM` in the table header for the standard errors indicates that the standard errors are computed from the observed information matrix. If non-normal errors are suspected, use `vce(robust)` to obtain the Huber-White robust standard errors (StataCorp 2009q, _robust).

Stata estimation commands store their results in a memory region called `ereturn`. The results may be accessed by the user and are used by other Stata commands, which are referred to as *postestimation commands* in Stata parlance. Typing

```
. ereturn list
```

lists the results saved in `e()`. You may view or access any `e()` result by identifying the object as `e(`*name*`)`, where *name* is the name of the object.

The matrices saved off by `sspace` are listed in Table 1 along with the Commandeur *et al.* (2011, Equations 1 and 2) equivalents.

Mixing both notations, a linear state-space model is

$$\begin{aligned}\boldsymbol{\alpha}_\mathbf{t} &= \mathbf{T}\boldsymbol{\alpha}_\mathbf{t-1} + \mathbf{B}\mathbf{x}_\mathbf{t} + \mathbf{R}\boldsymbol{\eta}_\mathbf{t} \\ \mathbf{y}_t &= \mathbf{Z}\boldsymbol{\alpha}_\mathbf{t} + \mathbf{F}\mathbf{w}_\mathbf{t} + \mathbf{G}\boldsymbol{\epsilon}_\mathbf{t},\end{aligned}$$

where $\mathbf{x}_t$ and $\mathbf{w}_t$ are column vectors of covariates. The vector $\mathbf{w}_t$ may contain lagged independent variables specified on the left-hand side of observation equations. Commandeur *et al.* (2011) incorporate the regression coefficent matrices $\mathbf{B}$ and $\mathbf{F}$ into the state transition matrix $\mathbf{T}$ and the observation equation matrix $\mathbf{Z}$, respectively.

The Kalman filter recursions are initialized with $\boldsymbol{\alpha}_1 = \mathbf{T}\boldsymbol{\alpha}_0 + \mathbf{B}\mathbf{x}_1$.

In this example the matrices are all 1×1, and we have `e(A)` $= 1$, `e(D)` $= 1$, `e(chol_Q)` $= \sqrt{\text{var(level)}}$, and `e(chol_R)` $= \sqrt{\text{var(AFV)}}$. The remaining matrices do not exist for this model.

Stata's `sspace` uses the square-root filter to numerically implement the Kalman filter recursions (DeJong 1991b; Durbin and Koopman 2001, Section 6.3). Moreover, when the model is not stationary, as is the case here, the filter is augmented as described by DeJong (1991a), DeJong and Chu-Chun-Lin (1994), and Durbin and Koopman (2001, Section 5.7). The two techniques are used together to evaluate the likelihood (DeJong 1988) and to provide maximum likelihood (ML) estimates of the parameters of the state-space model. The techniques also provide an estimate of the initial state. The initial state, $\alpha_0 = \mu_0$ is diffuse and is modeled as $\text{var}(\mu_0) \to \infty$ and $\text{E}[\mu_0] = \delta$. The ML estimate of δ is 1120.0. This quantity is not reported by `sspace`, but is stored as `e(d)`.

We can obtain predictions using the `predict` command, after estimating the parameters. All the standard objects and their standard errors can be predicted using `predict` after `sspace`. These objects and the syntax for `predict` after `sspace` are discussed in StataCorp (2009d).

2.3. Case 1 postestimation

With the local-level model estimates still in memory we predict the smoothed trend of the Nile annual flow volume using the DeJong (1989) diffuse Kalman filter. Here we use the `rmse` option to obtain the smoothed trend root-mean-square error (RMSE) that is subsequently used to compute 90% confidence intervals. A second call to `predict` obtains the standardized residuals. We graph the series, trend, and trend confidence intervals in one graph and the standardized residuals in a second graph. We then combine the two graphs into one and allow it to render. This graph is displayed in Figure 1.

```
. predict trend, state equation(level) smethod(smooth) rmse(rmse)
.
. scalar z = invnormal(.95)
. gen lb = trend - z*rmse
. gen ub = trend + z*rmse
.
. predict res, rstandard
.
. twoway (tsline AFV trend) (tsrline lb ub), tlabel(1870(50)1970) ///
>    ytitle(Annual Flow Volume) name(AFV) nodraw legend(off)
.
. tsline res, yline(3 -3) yline(0) tlabel(1870(50)1970) name(RES) nodraw
.
. graph combine AFV RES, name(AFVR) rows(2)
```

Next, we demonstrate forecasting. First we use the `preserve` command to save the original dataset. We then extend the data by 10 years using the `tsappend` command. We compute

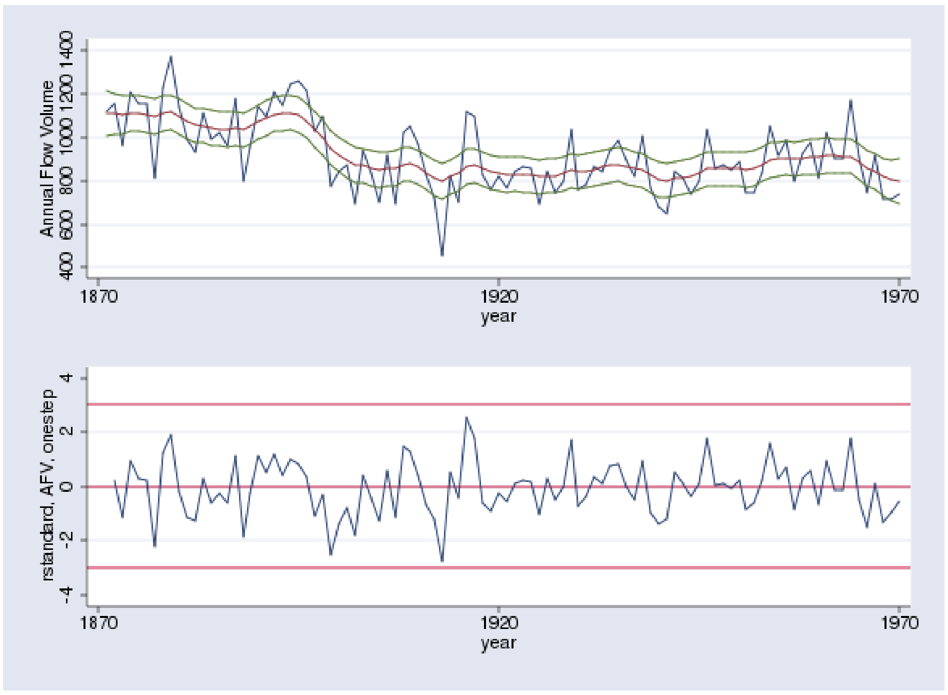

Figure 1: In the upper panel we display the Nile annual flow volume time-series (blue) with smoothed trend estimates (red) and trend 90% confidence intervals. The lower panel displays the standardized residuals.

the one-step predictions, compute dynamic forecasts from 1971 to 1980, and compute the RMSE's for the predictions and forecast predictions. We then compute the 50% confidence intervals for the forecasts and graph the results. Finally, we restore the original dataset. The graph is shown in Figure 2.

```
. preserve

. tsappend, add(10)

. predict flow, dynamic(1971) rmse(rflow)

. scalar z = invnormal(.75)

. gen lb = flow - z*rflow
(1 missing value generated)

. gen ub = flow + z*rflow
(1 missing value generated)

. twoway (tsline AFV flow) (tsrline lb ub if year>=1970),                  ///
>    tlabel(1870(10)1980) ytitle(Annual Flow Volume) name(FOR1) xline(1970) ///
>    legend(label(1 "AFV") label(2 "predicted/forecast") label(3 "50% CI"))

. restore
```

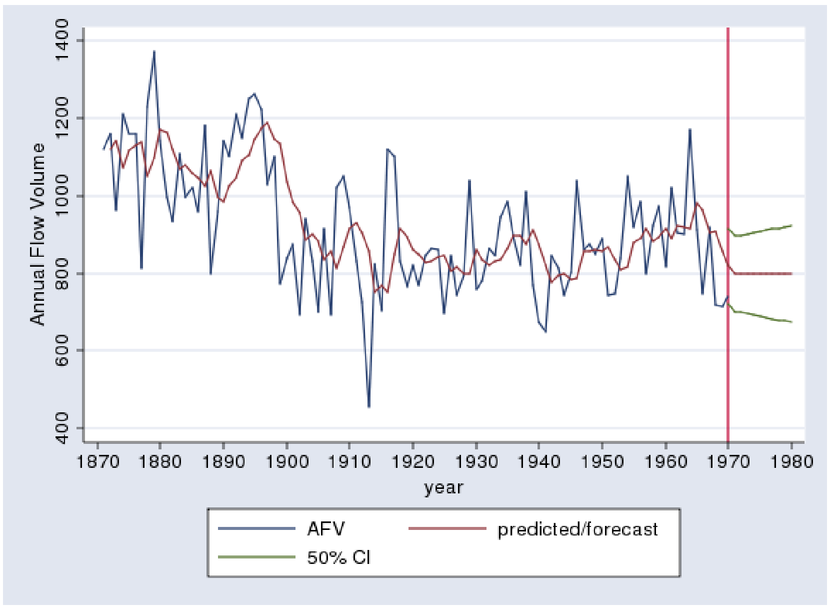

Figure 2: The Nile river annual flow volume (blue), one-step predictions and dynamic forecasts (red), and forecast 50% confidence intervals.

3. Case 2: A local-linear-trend model

In this section we review the structure of a local-linear-trend model with an autoregressive component, AR(1), and a seasonal component. The state-space form of a time-domain seasonal component is described in Commandeur *et al.* (2011, Section 2.1). Our state-space model is

$$\mu_t = \mu_{t-1} + \nu_{t-1} + \xi_t, \tag{2}$$

$$\nu_t = \nu_{t-1}, \tag{3}$$

$$\eta_t = \phi \cdot \eta_{t-1} + \zeta_t, \tag{4}$$

$$\gamma_{1,t} = -\gamma_{1,t-1} - \gamma_{2,t-1} - \gamma_{3,t-1} + \omega_t, \tag{5}$$

$$\gamma_{2,t} = \gamma_{1,t}, \tag{6}$$

$$\gamma_{3,t} = \gamma_{2,t}, \tag{7}$$

$$y_t = \mu_t + \eta_t + \gamma_{1,t}, \tag{8}$$

where $\zeta_t \sim NID(0, \sigma^2_\zeta)$, $\xi_t \sim NID(0, \sigma^2_\xi)$, and $\omega_t \sim NID(0, \sigma^2_\omega)$.

Equation (8) is the observation equation and it depends on the states μ (the linear trend), η (the AR(1) term), and γ_1 (the seasonal component). The observation equation has no error term. The model has six state equations: two for the linear trend, one for the AR(1) component and three for the seasonal component.

3.1. Estimating parameters of the local-linear-trend model using sspace

We now use `sspace` to estimate the parameters of a local-linear-trend model with an AR(1) component and a seasonal component. We fit this model to quarterly data on the food and tobacco production (FTP) in the United States for the years 1947 to 2000. Cox (2009) uses the dataset to demonstrate graphing seasonal time-series data in Stata.
First we read the dataset into memory and `describe` it:

```
. use http://www.stata.com/ddrukker/ftp.dta

(Food and tobacco production in the United States for 1947-2000)

. describe

Contains data from data/ftp.dta
  obs:           216                           Food and tobacco production in
                                                 the United States for
                                                 1947-2000
 vars:             2                           11 Jan 2010 10:02
 size:         2,592 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
ftp             float  %8.0g                  food and tobacco production
date            float  %tq
-------------------------------------------------------------------------------
Sorted by:  date
```

As before we `tsset` the data:

```
. tsset date

        time variable:  date, 1947q1 to 2000q4
                delta:  1 quarter
```

The code to estimate the parameters of the model is:

```
constraint 1 [trend]L.trend = 1
constraint 2 [trend]L.slope = 1
constraint 3 [slope]L.slope = 1
constraint 4 [season]L.season = -1
constraint 5 [season]L.s2 = -1
constraint 6 [season]L.s3 = -1
constraint 7 [s2]L.season = 1
constraint 8 [s3]L.s2 = 1
constraint 9 [ftp]ar  = 1
constraint 10 [ftp]trend  = 1
constraint 11 [ftp]season = 1
sspace  (trend L.trend L.slope, state noconstant)       ///
        (slope L.slope, state noerror noconstant)       ///
        (ar L.ar, state noconstant)                     ///
        (season L.season L.s2 L.s3, state noconstant)   ///
        (s2 L.season, noerror state noconstant)         ///
        (s3 L.s2, noerror state noconstant)             ///
        (ftp ar trend season, noerror noconstant),      ///
        constraints(1/11) covstate(diagonal)
```

The basic structure is the same as in the previous example. After defining some constraints, we issue the `sspace` command. The structure of the `sspace` command is also similar to the previous example. After specifying the state equations, we specify the observation equation, and then we specify the model-level options. The syntaxes for the state equations for the observation equation are similar to those in the previous example. The model-level option `covstate(diagonal)` is new; it specifies that covariance matrix of the state-errors have a diagonal structure. Each error has its own variance, but the errors are independent of one another.

The 6 state equations in the code above correspond the state equations (2)–(7). The algebraic version of the observation equation in the code above is given in Equation (8).

Repeating and running the code yields

```
. constraint 1 [trend]L.trend = 1
. constraint 2 [trend]L.slope = 1
. constraint 3 [slope]L.slope = 1
. constraint 4 [season]L.season = -1
. constraint 5 [season]L.s2 = -1
. constraint 6 [season]L.s3 = -1
. constraint 7 [s2]L.season = 1
. constraint 8 [s3]L.s2 = 1
. constraint 9 [ftp]ar= 1
. constraint 10 [ftp]trend= 1
. constraint 11 [ftp]season= 1
. sspace  (trend L.trend L.slope, state noconstant)       ///
>         (slope L.slope, state noerror noconstant)       ///
>         (ar L.ar, state noconstant)                     ///
>         (season L.season L.s2 L.s3, state noconstant)   ///
>         (s2 L.season, noerror state noconstant)         ///
>         (s3 L.s2, noerror state noconstant)             ///
>         (ftp ar trend season, noerror noconstant),      ///
>         constraints(1/11) covstate(diagonal)

searching for initial values ..
(setting technique to bhhh)
Iteration 0:   log likelihood = -405.11164
Iteration 1:   log likelihood = -366.97349
Iteration 2:   log likelihood = -347.30821
Iteration 3:   log likelihood = -347.08995
Iteration 4:   log likelihood =  -346.9888
(switching technique to nr)
Iteration 5:   log likelihood = -346.96929
Iteration 6:   log likelihood = -327.72965  (not concave)
Iteration 7:   log likelihood = -306.45684  (not concave)
Iteration 8:   log likelihood = -295.90364
Iteration 9:   log likelihood = -294.77578
Iteration 10:  log likelihood = -294.67933
Iteration 11:  log likelihood = -294.59382
Iteration 12:  log likelihood = -294.59331
Iteration 13:  log likelihood = -294.59331
Refining estimates:
Iteration 0:   log likelihood = -294.59331
Iteration 1:   log likelihood = -294.59331  (backed up)

State-space model

Sample: 1947q1 - 2000q4                           Number of obs    =        216
                                                  Wald chi2(1)     =       0.11
Log likelihood = -294.59331                       Prob > chi2      =     0.7363
 ( 1)  [trend]L.trend = 1
 ( 2)  [trend]L.slope = 1
 ( 3)  [slope]L.slope = 1
 ( 4)  [season]L.season = -1
 ( 5)  [season]L.s2 = -1
 ( 6)  [season]L.s3 = -1
 ( 7)  [s2]L.season = 1
 ( 8)  [s3]L.s2 = 1
 ( 9)  [ftp]ar = 1
 (10)  [ftp]trend = 1
 (11)  [ftp]season = 1
------------------------------------------------------------------------------
             |                 OIM
         ftp |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
trend        |
       trend |
         L1. |          1          .        .       .            .           .
             |
       slope |
         L1. |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
slope        |
       slope |
         L1. |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
ar           |
          ar |
         L1. |   .1522196   .4519697     0.34   0.736    -.7336248    1.038064
-------------+----------------------------------------------------------------
season       |
      season |
         L1. |         -1          .        .       .            .           .
             |
          s2 |
         L1. |         -1          .        .       .            .           .
             |
          s3 |
         L1. |         -1          .        .       .            .           .
```

```
-------------+----------------------------------------------------------------
s2           |
      season |
         L1. |          1          .         .        .            .           .
-------------+----------------------------------------------------------------
s3           |
          s2 |
         L1. |          1          .         .        .            .           .
-------------+----------------------------------------------------------------
ftp          |
          ar |          1          .         .        .            .           .
       trend |          1          .         .        .            .           .
      season |          1          .         .        .            .           .
-------------+----------------------------------------------------------------
var(trend)   |    .385335   .0958063     4.02   0.000     .1975581    .5731119
var(ar)      |   .0987783    .095571     1.03   0.301    -.0885374     .286094
var(season)  |   .0356305   .0136982     2.60   0.009     .0087825    .0624785
------------------------------------------------------------------------------
Note: Model is not stationary.
Note: Tests of variances against zero are conservative and are provided only
      for reference.
```

The coefficient table lists 15 estimates, only 4 of which are unconstrained: `ar.L1` = 0.152, `var(trend)` = 0.385, `var(ar)` = 0.0988, and `var(season)` = 0.0356. These are estimates of ϕ, σ^2_ξ, σ^2_ζ, and σ^2_ω, respectively. We specified that the covariance for the state equations be diagonal; this is the default and was added for clarity.

3.2. Case 2 postestimation

After estimation, we can use the `predict` command to compute estimates of the observables or unobservables using the one-step, filter, or smoothed methods (Durbin and Koopman 2001, Chapter 4; DeJong 1989). The observation equation residuals or standardized residuals may be computed using the one-step or smoothed methods.

Below we compute the one-step estimates of the food and tobacco production:

```
. predict ftp1

 (option xb assumed; fitted values)
```

Now we predict the one-step trend:

```
. predict trend, state equation(trend)
```

Finally, we compute the residuals:

```
. predict res, residuals
```

Now we perform some computations to produce more informative graphs. In the code below, we store the index that marks the last quarter of the sample in a local macro and generate a new variable `q` containing the quarter per annum of each observation.

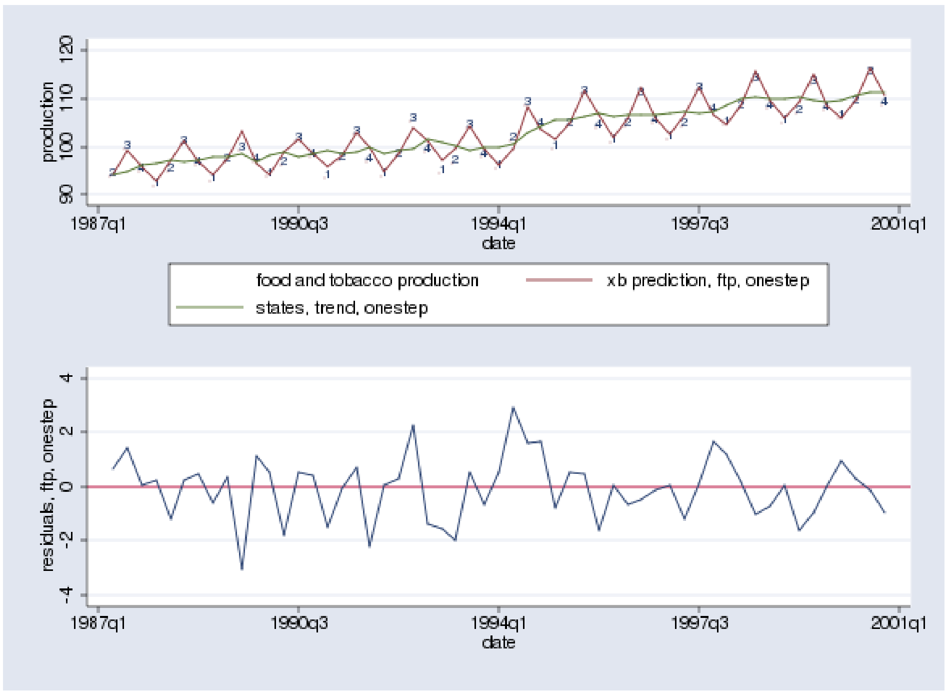

Figure 3: Quarterly data on food and tobacco production in the United States with smoothed series and the filtered trend in the top panel. The one-step residuals are in the bottom panel.

```
. local n = floor(3*_N/4)
. generate int q = quarter(dofq(date))
```

The next block produces the graphs shown in Figure 3. Figure 3 shows the time-series plots using plotting tips by Cox (2009) with the smoothed series and filtered trend. We only graph the last quarter of the sample. (The growth in the series covers up the seasonal detail when we graph the series over the entire sample.)

```
. twoway (scatter ftp date in `n'/L, msymbol(none) mlabel(q) ///
>        mlabposition(0) ytitle(production) ylabel(#3))      ///
>        (tsline ftp1 trend in `n'/L), nodraw name(FTP1)

. tsline res in `n'/L, nodraw name(RES) yline(0)
. graph combine FTP1 RES, name(FTP2) rows(2)
. graph drop FTP1 RES
```

Next we illustrate how to forecast estimates. We begin by extending the data, adding two years starting at Q1 of year 2001.

```
. tsappend, add(8)
```

The next code block predicts `ftp`, specifying that dynamic forecasts should begin on quarter Q1 of 2001. The function `tq(2001q1)` translates the string “2001q1” to the appropriate

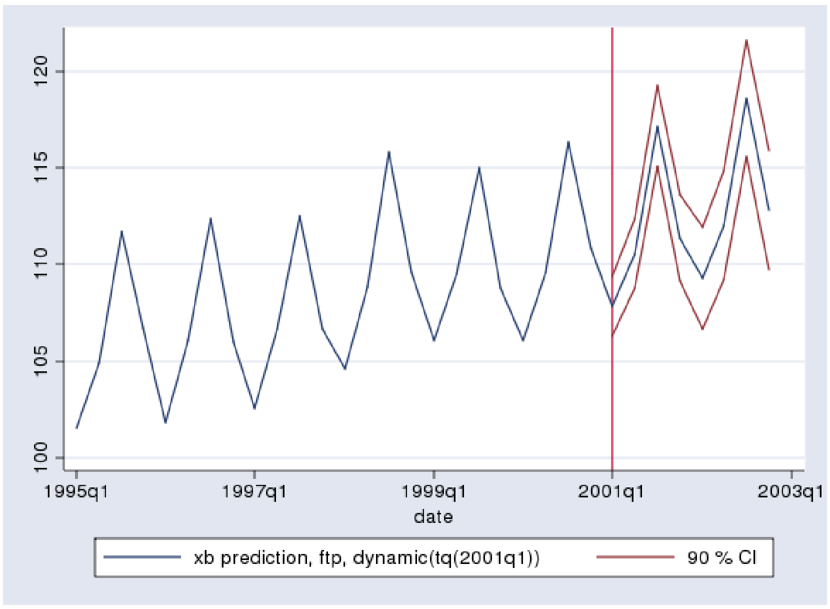

Figure 4: One-step predictions of US food and tobacco production with dynamic predictions starting at Q1 of year 2001. Approximate 95% confidence bounds are also given.

numeric value: the number of quarters since Q1 1960. We also request the root-mean-square error (RMSE) of the residuals. We use the RMSE estimates to compute an approximate 90% confidence intervals about the forecast values. These are displayed in Figure 4.

```
. predict ftp1, dynamic(tq(2001q1)) rmse(rftp)
. scalar z = invnormal(0.95)

. gen lb = ftp1 + z*rftp if date>=tq(2001q1)
. gen ub = ftp1 - z*rftp if date>=tq(2001q1)

. tsline ftp1 if date>=tq(1995q1) || tsrline lb ub if date>=tq(1995q1), ///
                    xline(`=tq(2001q1)') legend(label(2 "90 \% CI")) name(DYN)
```

4. Case 3: The vector autoregressive moving-average model

We used `freduse` (see Drukker 2006) to obtain Federal Reserve data on the capacity utilization rate, `caputil`, and manufacturing hours, `hours`, for the US economy (`http://research.stlouisfed.org/fred2`). Here we model the differenced series, `D.caputil` and `D.hours`, as a first-order vector autoregressive moving-average (VARMA(1,1)) process. In this model, we allow the lag of `D.caputil` to affect `D.hours`, but we do not allow the lag of `D.hours` to affect the lag of `D.caputil`, as was done in StataCorp (2009e, Example 4).

$$\begin{pmatrix}\Delta c_t\\ \Delta h_t\end{pmatrix} = \begin{pmatrix}\phi_1 & 0\\ \phi_2 & \phi_3\end{pmatrix}\begin{pmatrix}\Delta c_{t-1}\\ \Delta h_{t-1}\end{pmatrix} + \begin{pmatrix}\theta_1 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}\eta_{t-1,1}\\ \eta_{t-1,2}\end{pmatrix} + \begin{pmatrix}\eta_{t,1}\\ \eta_{t,2}\end{pmatrix} \tag{9}$$

The state equations and the observation equations for the state-space form of this VARMA(1,1) model may be written, respectively, in vector form as

$$\begin{pmatrix}\alpha_{t,1}\\ \alpha_{t,2}\\ \alpha_{t,3}\end{pmatrix} = \begin{pmatrix}\phi_1 & 1 & 0\\ 0 & 0 & 0\\ \phi_2 & 0 & \phi_3\end{pmatrix}\begin{pmatrix}\alpha_{t-1,1}\\ \alpha_{t-1,2}\\ \alpha_{t-1,3}\end{pmatrix} + \begin{pmatrix}1 & 0\\ \theta_1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}\eta_{t,1}\\ \eta_{t,2}\end{pmatrix}, \tag{10}$$

$$\begin{pmatrix}\Delta c_t\\ \Delta h_t\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\alpha_{t,1}\\ \alpha_{t,2}\\ \alpha_{t,3}\end{pmatrix} \tag{11}$$

where $\alpha_{t,1} = \Delta c_t$, $\alpha_{t,2} = \theta_1\eta_{t,1}$, $\alpha_{t,3} = \Delta h_t$, and the 2×2 covariance matrix $\text{cov}(\boldsymbol{\eta}_t)$ is diagonal.

Next we use the `sspace` error-form syntax to estimate the parameters of this model.

4.1. Error-form syntax

The error-form syntax of `sspace` has the same overall structure as the covariance form, but it has an extra component in the state equation.

```
        (statevar [lagged_statevars] [indepvars] [state_errors], state [noconstant])
```

The optional `[state_errors]` lists state-equation errors that enter a state equation. Each state error has the form `e.statevar`, where `statevar` is the name of a state in the model. The `state_errors` define the covariance structure so the option `covstate()` is not necessary. Also, the `noerror` option has no meaning in this style of syntax.

4.2. Estimation of the VARMA(1,1)

We now use the error-form syntax of `sspace` to estimate the parameters of the VARMA(1,1) model whose state-space form is given in Equations (10) and (11).

The code for estimating the model parameters is given below:

```
constraint 1 [u1]L.u2 = 1
constraint 2 [u1]e.u1 = 1
constraint 3 [u3]e.u3 = 1
constraint 4 [D.caputil]u1 = 1
constraint 5 [D.hours]u3   = 1
sspace  (u1 L.u1 L.u2 e.u1, state noconstant)   ///
        (u2 e.u1, state noconstant)             ///
        (u3 L.u1 L.u3 e.u3, state noconstant)   ///
        (D.caputil u1, noconstant)              ///
        (D.hours u3, noconstant),               ///
        constraints(1/5) covstate(diagonal) vce(robust)
```

The code has the same structure as the previous examples. After defining the contraints, we use them in the `sspace` command. The `sspace` command itself has two parts: First come the equations that define the state-space form of the model. Second we specify model-level options.

The code specifies the five equations that define the state-space form of the model. The first three equations are the state equations whose algebraic counterparts are in Equation (10). The only difference in the two versions is that the states are named α_1, α_2, and α_3 in the algebra and named `u1`, `u2` and `u3` in code. The last two equations are the observation equations whose algebraic equivalent is given in Equation (11).

We have already discussed the model-level options `constraints()` and `covstate()`. The model-level option `vce(robust)` specifies that the standard errors should be estimated using the Huber-White robust estimator which is robust to nonnormal errors in this case.

Below we read in the dataset and run the code.

```
. webuse manufac.dta

(St. Louis Fed (FRED) manufacturing data)

. constraint 1 [u1]L.u2 = 1
. constraint 2 [u1]e.u1 = 1
. constraint 3 [u3]e.u3 = 1
. constraint 4 [D.caputil]u1 = 1
. constraint 5 [D.hours]u3 = 1
. sspace  (u1 L.u1 L.u2 e.u1, state noconstant)   ///
>         (u2 e.u1, state noconstant)             ///
>         (u3 L.u1 L.u3 e.u3, state noconstant)   ///
>         (D.caputil u1, noconstant)              ///
>         (D.hours u3, noconstant),               ///
>         constraints(1/5) covstate(diagonal) vce(robust)

Iteration 0:   log pseudolikelihood = -468.09528
Iteration 1:   log pseudolikelihood = -436.36371
Iteration 2:   log pseudolikelihood = -417.72583
Iteration 3:   log pseudolikelihood = -414.07834
Iteration 4:   log pseudolikelihood = -411.97958
(switching technique to nr)
Iteration 5:   log pseudolikelihood = -410.90058
Iteration 6:   log pseudolikelihood = -408.46772
Iteration 7:   log pseudolikelihood = -408.44012
Iteration 8:   log pseudolikelihood = -408.44012
Refining estimates:
Iteration 0:   log pseudolikelihood = -408.44012
Iteration 1:   log pseudolikelihood = -408.44012

State-space model

Sample: 1972m2 - 2008m12                          Number of obs    =        443
                                                  Wald chi2(4)     =     281.15
Log likelihood = -408.44012                       Prob > chi2      =     0.0000
 ( 1)  [u1]L.u2 = 1
 ( 2)  [u1]e.u1 = 1
 ( 3)  [u3]e.u3 = 1
 ( 4)  [D.caputil]u1 = 1
 ( 5)  [D.hours]u3 = 1
------------------------------------------------------------------------------
```

```
             |               Robust
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
u1           |
          u1 |
         L1. |   .8041549   .0586271    13.72   0.000     .6892478     .919062
             |
          u2 |
         L1. |          1          .        .       .            .           .
        e.u1 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
u2           |
        e.u1 |  -.5236703   .0807037    -6.49   0.000    -.6818466    -.365494
-------------+----------------------------------------------------------------
u3           |
          u1 |
         L1. |   .0861277   .0247206     3.48   0.000     .0376762    .1345791
             |
          u3 |
         L1. |  -.4734121   .1275157    -3.71   0.000    -.7233384   -.2234859
        e.u3 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
D.caputil    |
          u1 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
D.hours      |
          u3 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
var(u1)      |   .3564469   .0407754     8.74   0.000     .2765287    .4363651
var(u3)      |    .060721   .0120762     5.03   0.000     .0370521    .0843898
------------------------------------------------------------------------------
Note: Tests of variances against zero are conservative and are provided only
      for reference.
```

The output table gives us the parameter estimates, the estimated standard errors, confidence intervals, and tests against zero. Using the notation of Equation (9), the estimates of the AR parameters are $\hat{\phi}_1 = 0.804$, $\hat{\phi}_2 = 0.0861$, and $\hat{\phi}_3 = -0.473$. The estimated MA parameter is $\hat{\theta}_1 = -0.524$. The variance estimates are $\widehat{\text{var}}(\eta_{t,1}) = 0.356$ and $\widehat{\text{var}}(\eta_{t,2}) = 0.0607$.

4.3. Case 3 postestimation

We now predict the differenced capital utilization using the one-step predictions and the standardized residuals:

```
. predict pcaputil, equation(D.caputil)

 (option xb assumed; fitted values)

. predict stdres, rstandard equation(D.caputil)
```

`predict` computes predicted values for both `D.caputil` and `D.hours` by default because there are two observation equations in our model. To override the default behavior, we specify the

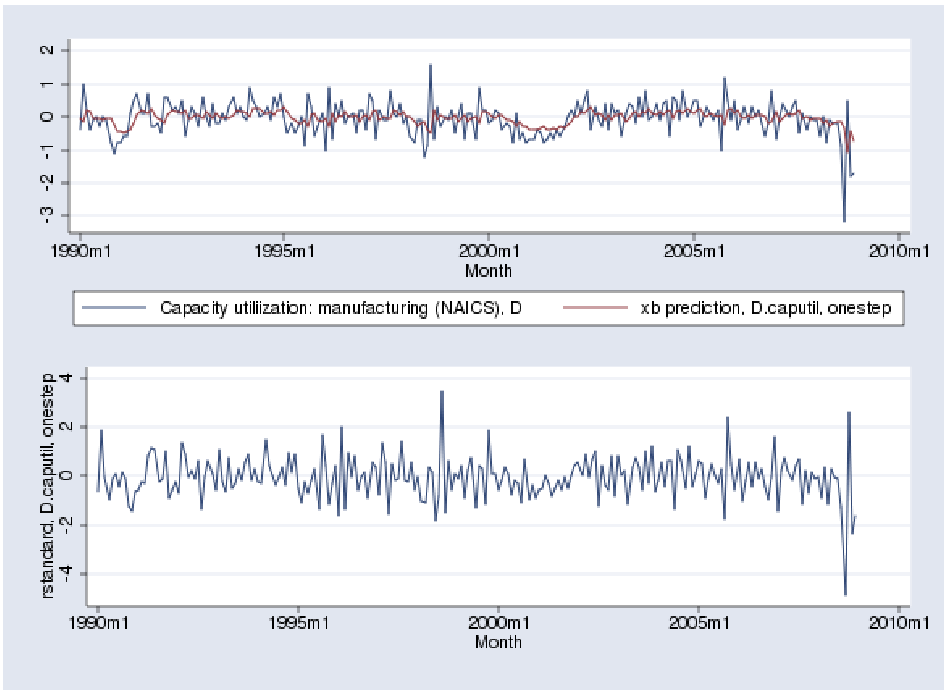

Figure 5: The upper panel displays the one-step predictions of manufacturing capacity utilization starting at January 1990. The standardized residuals are displayed in the lower panel.

option `equation(D.caputil)`, which instructs `predict` to perform the computation only for the `D.caputil` equation.

```
. tsline D.caputil pcaputil if month>=tm(1990m1), name(CAP) nodraw
. tsline stdres if month>=tm(1990m1), name(RES) nodraw
. graph combine CAP RES, rows(2) name(CH2)
. graph drop CAP RES
```

Figure 5 displays the capacity utilization time series and the one-step predictions in the upper panel. The lower panel displays the standardized residuals. Only the latter half of the data are shown.

5. A dynamic-factor example

State-space models have been used to formulate estimators for popular models such as ARMA and VARMA models and to formulate estimators for new models suggested by the state-space framework. The unobserved-components (UC) model discussed in Harvey (1989) and the dynamic-factor model are two of the most important models that naturally fit into a state-space framework. Above we considered UC models and a VARMA model; now we consider a dynamic-factor model.

Dynamic-factor models are VAR models augmented by unobserved factors that may also have an autoregressive structure. Dynamic-factor models have been applied in macroeconomics, see Geweke (1977), Sargent and Sims (1977), Stock and Watson (1989), Stock and Watson (1991) and Watson and Engle (1983). Lütkepohl (2005) provides a good introduction to dynamic-factor models and their state-space formulation. StataCorp (2009a) provides a quick introduction to these models and has several examples including the one discussed below.

In this example, we consider a dynamic-factor model without exogenous variables in which the dynamic factor follows an AR(2) process, and the disturbances in the equations for the observable variables follow AR(1) processes. This example illustrates how to specify a dynamic-factor model and how to specify an AR(2) process. The `dfactor` command is an easy-to-use alternative to `sspace` for dynamic-factor models.

The state-space form of the model we consider is

$$f_t = \phi_1 f_{t-1} + \phi_2 f_{t-2} + \eta_t \tag{12}$$
$$f_{t-1} = f_{t-1} \tag{13}$$
$$\boldsymbol{\mu}_t = \boldsymbol{\Psi}\boldsymbol{\mu}_{t-1} + \boldsymbol{\epsilon}_t \tag{14}$$
$$\mathbf{y}_t = \mathbf{b} f_t + \boldsymbol{\mu}_t \tag{15}$$

where f_t is an unobserved factor that follows an AR(2) process; $\boldsymbol{\mu}_t$ is a 4×1 vector of errors, each of which follows an AR(1) process; and $\mathbf{y}_t$ is a 4×1 vector of dependent variables. Equations (12), (13), and (14) are the state equations. Equation (15) is the vector of observation equations.

The system is driven by η_t (a scalar IID error) and by $\boldsymbol{\epsilon}_t$ (a 4×1 vector of IID errors). By restricting $\boldsymbol{\Psi}$ to be 4×4 diagonal matrix, we specify that the unobserved factor is the only source of correlation between the dependent variables. ϕ_1 and ϕ_2 are the coefficients of the AR(2) process for the dynamic factor. $\mathbf{b}$ is a 4×1 vector of coefficients.

We downloaded some US macroeconomic data from the FRED database of the St. Louis Federal Reserve using the `freduse` command discussed in Drukker (2006). Specifically, we have data on the $\mathbf{y}_t$ variables industrial production index, `ipman`; real disposable income, `income`; an aggregate weekly hours index, `hours`; and aggregate unemployment, `unemp`. These data were used in the Stata manuals, so we use the `webuse` command to download the dataset and read it into memory.

In the code below, we use `sspace` to estimate the parameters of this model:

```
. webuse dfex, clear
(St. Louis Fed (FRED) macro data)
. constraint define 1 [Lf]L.f = 1
. constraint define 2 [D.ipman]u1 = 1
. constraint define 3 [D.income]u2 = 1
. constraint define 4 [D.hours]u3 = 1
. constraint define 5 [D.unemp]u4 = 1
. constraint define 6 [var(f)]_cons = 1

. sspace  (f L.f L.Lf, state noconstant)                    ///
>         (Lf L.f, state noconstant noerror)                ///
>         (u1 L.u1, state noconstant)                       ///
>         (u2 L.u2, state noconstant)                       ///
>         (u3 L.u3, state noconstant)                     ///
>         (u4 L.u4, state noconstant)                     ///
>         (D.ipman f u1, noconstant noerror)              ///
>         (D.income f u2, noconstant noerror)             ///
>         (D.hours f u3, noconstant noerror)              ///
>         (D.unemp f u4, noconstant noerror),             ///
>         covstate(diagonal) constraints(1/6)
searching for initial values ..............
(setting technique to bhhh)
Iteration 0:   log likelihood = -667.60855
Iteration 1:   log likelihood = -631.10186
Iteration 2:   log likelihood = -618.21015
Iteration 3:   log likelihood = -615.08888
Iteration 4:   log likelihood = -613.63357
(switching technique to nr)
Iteration 5:   log likelihood = -612.55257
Iteration 6:   log likelihood = -610.31321
Iteration 7:   log likelihood = -610.28847
Iteration 8:   log likelihood = -610.28846
Refining estimates:
Iteration 0:   log likelihood = -610.28846
Iteration 1:   log likelihood = -610.28846

State-space model

Sample: 1972m2 - 2008m11                          Number of obs   =        442
                                                  Wald chi2(10)   =     990.91
Log likelihood = -610.28846                       Prob > chi2     =     0.0000
 ( 1)  [Lf]L.f = 1
 ( 2)  [D.ipman]u1 = 1
 ( 3)  [D.income]u2 = 1
 ( 4)  [D.hours]u3 = 1
 ( 5)  [D.unemp]u4 = 1
 ( 6)  [var(f)]_cons = 1
------------------------------------------------------------------------------
             |                 OIM
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
f            |
           f |
         L1. |   .4058457   .0906183     4.48   0.000     .2282371    .5834544
             |
          Lf |
         L1. |   .3663499   .0849584     4.31   0.000     .1998344    .5328654
-------------+----------------------------------------------------------------
Lf           |
           f |
         L1. |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
u1           |
          u1 |
         L1. |  -.2772149    .068808    -4.03   0.000    -.4120761   -.1423538
-------------+----------------------------------------------------------------
```

```
u2           |
          u2 |
         L1. |  -.2213824   .0470578    -4.70   0.000    -.3136141   -.1291508
-------------+----------------------------------------------------------------
u3           |
          u3 |
         L1. |  -.3969317   .0504256    -7.87   0.000     -.495764   -.2980994
-------------+----------------------------------------------------------------
u4           |
          u4 |
         L1. |  -.1736835   .0532071    -3.26   0.001    -.2779675   -.0693995
-------------+----------------------------------------------------------------
D.ipman      |
           f |   .3214972    .027982    11.49   0.000     .2666535    .3763408
          u1 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
D.income     |
           f |   .0760412   .0173844     4.37   0.000     .0419684     .110114
          u2 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
D.hours      |
           f |   .1933165   .0172969    11.18   0.000     .1594151    .2272179
          u3 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
D.unemp      |
           f |  -.0711994   .0066553   -10.70   0.000    -.0842435   -.0581553
          u4 |          1          .        .       .            .           .
-------------+----------------------------------------------------------------
var(f)       |          1          .        .       .            .           .
var(u1)      |   .1387909   .0154558     8.98   0.000     .1084981    .1690837
var(u2)      |   .2636239   .0179043    14.72   0.000     .2285322    .2987157
var(u3)      |   .0822919   .0071096    11.57   0.000     .0683574    .0962265
var(u4)      |   .0218056   .0016658    13.09   0.000     .0185407    .0250704
------------------------------------------------------------------------------
Note: Tests of variances against zero are conservative and are provided only
      for reference.
```

The code is similar to what we have seen in previous examples. Equation (13), which is the second equation specified in the `sspace` command, is the new element in this example. This method of including lags as additional trivial state equations is a standard trick in state-space modeling, see Lütkepohl (2005, Chapter 18.2) and Hamilton (1994, Chapter 13.1).

The Stata command `dfactor` provides an easy-to-use syntax for estimating the parameters of dynamic-factor models. For example, the command

```
. dfactor (D.(ipman income hours unemp), noconstant ar(1)) (f = , ar(1/2))
```

produces the same parameter estimates as the above `sspace` command.

`predict` after `sspace` and after `dfactor` provide all the standard options to forecast the observed dependent variables or to extract unobserved factors. We have already illustrated the use of `predict` after `sspace`, see StataCorp (2009b) for further examples and a detailed discussion of the underlying mathematics.

6. Conclusion

We have illustrated how to estimate the parameters of UC models, VARMA models, and dynamic-factor models using Stata's `sspace` command. Stata's `sspace` command can estimate the parameters of many other linear state-space models.

Using Stata's ADO programming language (StataCorp 2009q), the `sspace` command could be used as a computational engine for new estimation commands. The `dfactor` command is an example. These commands would be easy-to-use versions of `sspace`, presenting a simplified syntax unique to the target model. Because Stata is such a popular platform among applied researchers, `sspace` provides an opportunity for theoretical researchers to easily make their methods available to a huge audience of applied researchers. More complicated estimators could combine Stata's byte-compiled matrix language Mata, see StataCorp (2009f) and StataCorp (2009g), and `sspace` to implement new estimators.

References

Baum CF (2006). *An Introduction to Modern Econometrics Using Stata*. Stata Press, College Station, Texas.

Cameron AC, Trivedi PK (2009). *Microeconometrics Using Stata*. Stata Press, College Station, Texas.

Cleves MA, Gould WW, Gutierrez RG, Marchenko YU (2010). *An Introduction to Survival Analysis Using Stata*. 3rd edition. Stata Press, College Station, Texas.

Commandeur JJF, Koopman SJ, Ooms M (2011). "Statistical Software for State Space Methods." *Journal of Statistical Software*, **41**(1), 1–18. URL `http://www.jstatsoft.org/v41/i01/`.

Cox NJ (2009). "Stata Tip 76: Separating Seasonal Time Series." *The Stata Journal*, **9**, 321–326.

DeJong P (1988). "The Likelihood for a State Space Model." *Biometrika*, **75**, 165–169.

DeJong P (1989). "Smoothing and Interpolation with the State Space Model." *Journal of the American Statistical Association*, **84**, 1085–1088.

DeJong P (1991a). "The Diffuse Kalman Filter." *The Annals of Statistics*, **19**, 1073–1083.

DeJong P (1991b). "Stable Algorithms for the State Space Model." *Journal of Time Series Analysis*, **12**, 143–145.

DeJong P, Chu-Chun-Lin S (1994). "Stationary and Non-Stationary State Space Models." *Journal of Time Series Analysis*, **15**, 151–166.

Drukker DM (2006). "Importing Federal Reserve Economic Data." *The Stata Journal*, **6**(3), 384–386.

Drukker DM, Wiggins V (2004). "Verifying the Solution from a Nonlinear Solver: A Case Study: Comment." *American Economic Review*, **94**(1), 397–399.

Durbin J, Koopman SJ (2001). *Time Series Analysis by State Space Methods.* Oxford University Press, Oxford.

Geweke J (1977). "The Dynamic Factor Analysis of Economic Time Series Models." In DJ Aigner, AS Goldberger (eds.), *Latent Variables in Socioeconomic Models*, pp. 365–383. North-Holland, Amsterdam.

Hamilton JD (1994). *Time Series Analysis.* Princeton University Press, Princeton.

Harvey AC (1989). *Forecasting, Structural Time Series Models and the Kalman Filter.* Cambridge University Press, Cambridge.

Juul S, Frydenberg M (2010). *An Introduction to Stata for Health Researchers.* 3rd edition. Stata Press, College Station, Texas.

Lütkepohl H (2005). *New Introduction to Multiple Time Series Analysis.* Springer-Verlag, New York.

Mitchell MN (2010). *Data Management Using Stata – A Practical Handbook.* Stata Press, College Station, Texas.

Sargent TJ, Sims CA (1977). "Business Cycle Modeling without Pretending to Have Too Much A Priori Economic Theory." In CA Sims (ed.), *New Methods in Business Cycle Research: Proceedings from a Conference*, pp. 45–109. Federal Reserve Bank of Minneapolis, Minneapolis.

StataCorp (2009a). "`dfactor`: Dynamic-Factor Models." In *Stata Time-Series Reference Manual, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009b). "`dfactor` Postestimation: Postestimation Tools for `dfactor`." In *Stata Time-Series Reference Manual, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009c). "`optimize()`: Function Optimization." In *Mata Matrix Programming Reference Manual, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009d). "`sspace` Postestimation: Postestimation Tools for `sspace`." In *Stata Time-Series Reference Manual, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009e). "`sspace`: State-Space Models." In *Stata Time-Series Reference Manual, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009f). *Mata Matrix Programming 0–3, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009g). *Mata Matrix Programming 4–6, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009h). *Stata Longintudinal Data/Panel Data Reference Manual, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009i). *Stata Multivariate Statitics Reference Manual, Release 11.* Stata Press, College Station, Texas.

StataCorp (2009j). *Stata Progamming Reference Manual, Release 11*. Stata Press, College Station, Texas.

StataCorp (2009k). *Stata Reference Manual, A–H Release 11*. Stata Press, College Station, Texas.

StataCorp (2009l). *Stata Reference Manual, I–P Release 11*. Stata Press, College Station, Texas.

StataCorp (2009m). *Stata Reference Manual, Q–Z Release 11*. Stata Press, College Station, Texas.

StataCorp (2009n). *Stata Survey Data Reference Manual, Release 11*. Stata Press, College Station, Texas.

StataCorp (2009o). *Stata Survival Analysis and Epidemiological Tables Reference Manual, Release 11*. Stata Press, College Station, Texas.

StataCorp (2009p). *Stata Time-Series Reference Manual, Release 11*. Stata Press, College Station, Texas.

StataCorp (2009q). "Programming." In *Stata Progamming Reference Manual, Release 11*. Stata Press, College Station, Texas.

Stock JH, Watson MW (1989). "New Indexes of Coincident and Leading Economic Indicators." In OJ Blanchard, S Fischer (eds.), *NBER Macroeconomics Annual 1989*, volume 4, pp. 351–394. MIT Press, Cambridge, MA.

Stock JH, Watson MW (1991). "A Probability Model of the Coincident Economic Indicators." In K Lahiri, GH Moore (eds.), *Leading Economic Indicators: New Approaches and Forecasting Records*, pp. 63–89. Cambridge University Press, Cambridge.

Watson MW, Engle RF (1983). "Alternative Algorithms for the Estimation of Dymanic Factor, MIMIC and Varying Coefficient Regression Models." *Journal of Econometrics*, **23**, 385–400.

Affiliation:

David M. Drukker
Stata
4905 Lakeway Drive
College Station, TX, United States of America
E-mail: ddrukker@stata.com
URL: http://stata.com/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/

Volume 41, Issue 10 *Submitted:* 2010-01-15
May 2011 *Accepted:* 2010-09-29