Journal of Statistical Software

May 2011, Volume 41, Issue 9. http://www.jstatsoft.org/

State Space Methods in RATS

Thomas Doan
Estima

Abstract

This paper uses several examples to show how the econometrics program **RATS** can be used to analyze state space models. It demonstrates Kalman filtering and smoothing, estimation of hyperparameters, unconditional and conditional simulation. It also provides a more complicated example where a dynamic simultaneous equations model is transformed into a proper state space representation and its unknown parameters are estimated.

Keywords: ARMA model, Kalman filter, state space methods, unobserved components, software tools.

1. Introduction

RATS (Estima 2010) is a general-purpose econometrics and programming package, with a specialty in time series analysis. The instruction in **RATS** for handling state space models is called `DLM` (for dynamic linear models). This was introduced with version 5.00 in 2001. Since then, there have been many improvements. With version 8, the instruction has the following features:

- All component matrices can be fixed matrices, or time-varying formulas, or can be computed using functions of arbitrary complexity.
- Multiple observables (y_t with dimension > 1) are permitted, with proper handling of situations where some components are missing, but some are present.
- Non-stationary roots in the transition matrix are treated with the "exact" (limit) methods of Koopman (1997) and Durbin and Koopman (2001). The transition matrix is analyzed automatically for stationary and non-stationary roots.
- The ergodic variance for the stationary (linear combinations of) states is computed using the efficient Schur decomposition method described in Doan (2010).

- The calculations of the Kalman filter and smoother can switch (under user control) to use the last calculated values of the Kalman gain and predictive variances. This can save time in large models with time-invariant component matrices.

With the **RATS** distribution, we include the worked examples from several textbooks devoted to space-space models, including Durbin and Koopman (2001), Commandeur and Koopman (2007), West and Harrison (1997) and Harvey (1989). These are also posted on our web site at `http://www.estima.com/textbookindex.shtml`.

The aim of **RATS** is to combine the best features of statistical and "math" packages. It has almost all of the capabilities of the commonly used matrix languages[1], but also includes carefully written, highly-optimized instructions for estimating and analyzing the most important types of models. `DLM` is a particularly good example of this. While it is not hard to write in matrix form Kalman filtering and Kalman smoothing for small, well-behaved special cases, it is much harder to handle (correctly) missing data and initialization and to hook the calculations up with optimization code. `DLM` handles those issues and more.

In addition to the "built-in" instructions like `DLM` and `GARCH` (which handles a wide variety of univariate and multivariate GARCH models), **RATS** includes hundreds of procedures which are usually short sequences of special-purpose code. Because these are plain text files, they can be modified easily by the user. For instance, in our examples in this paper, we use the procedure `STAMPDIAGS` for diagnostics on state space models. If you want to change what this reports, you can just modify the procedure.

This paper is organized as follows. Section 2 describes the structure of state space models as handled through the `DLM` instruction. Section 3 introduces the model used through most of this paper. For that model, Section 3.1 demonstrates Kalman smoothing, given values for the variances. Section 3.2 shows the various ways to estimate the hyperparameters (variances) with Section 3.3 discussing several types of calculated or graphed diagnostics for those estimates. Section 3.4 shows how to forecast out-of-sample. Section 3.5 offers an example of unconditional simulation and 3.6 uses conditional simulations to do Gibbs sampling. We look at a more complicated model in Section 4. In all cases, we are providing only the segment of code needed to demonstrate a technique. The full running examples are available along with this manuscript and on our web site at `http://www.estima.com/resources_articles.shtml`.

2. The `DLM` instruction

Our state space structure takes a bit broader form than the one described in the introduction paper to this volume. Because the components are input to the `DLM` instruction using short alphabetical names based upon our own description of the state space model, we will use that from this point on in this article:[2]

$$\mathbf{Y}_t = \mu_t + \mathbf{C}_t^{\top}\mathbf{X}_t + \mathbf{v}_t \tag{1}$$

$$\mathbf{X}_t = \mathbf{A}_t\mathbf{X}_{t-1} + \mathbf{Z}_t + \mathbf{F}_t\mathbf{w}_t \tag{2}$$

Table 1 shows the translations between the notation in the introduction paper and ours. The addition of the μ_t term to the measurement equation is only a minor matter of convenience,

[1] Such as GAUSS, MATLAB and Ox.

[2] Note that **RATS** uses a different timing on the components in the state equation.

Introduction	**RATS**
α_t	$\mathbf{X}_t$
Z_t	$\mathbf{C}_t^\top$
ε_t	$\mathbf{v}_t$
H_t	$\mathbf{SV}_t$
T_t	$\mathbf{A}_{t+1}$
R_t	$\mathbf{F}_{t+1}$
η_t	$\mathbf{w}_{t+1}$
Q_t	$\mathbf{SW}_{t+1}$

Table 1: Notation for state space models.

since the identical model can be produced by subtracting μ_t from both sides of the equation. However, the enhanced form of the state equation with the $\mathbf{Z}_t$ state shift can not so easily be accommodated in the simpler form, particularly when the state shift component is time-varying.

Given a state space model, you can choose to:

- Kalman filter, computing the likelihood function assuming Gaussian errors; this also computes predictions for the states and the observables, the prediction errors and variances for the predictions and states.
- Kalman smooth, with calculations of smoothed states and their variances, disturbances and their variances.
- Simulate unconditionally, with random draws for the state and measurement shocks, producing simulated states and observables.
- Simulate conditional on observed data, producing simulated states and shocks.

There are as many as ten inputs to `DLM` when you count the pre-sample mean and variance, and even more potential outputs. However, most applications will not need all, or even most, of them. Most of the information is supplied to the instruction using *options*. Each option is given a name, which for `DLM` generally matches with the notation from (1) and (2), so the $\mathbf{A}$ matrix is put in using something like `A = 1.0` or `A = ADLM`. By relying on these options, we make it easier to do simpler models, since little-used inputs like μ_t and $\mathbf{Z}_t$ just default to zero and can be left out of the specification.

3. The local level model

The model that we will use for the example in this section is the local level model, applied to the Nile flow data, annual from 1871 to 1970. The model is (with *our* timing on the state equation)

$$y_t = \alpha_t + \varepsilon_t$$
$$\alpha_t = \alpha_{t-1} + \xi_t$$

where α_t is the unobservable local level. The model has time-invariant components $\mathbf{A} = \mathbf{C} = \mathbf{F} = 1$, $\mathbf{Z} = \mu = 0$. These are the default values for all but $\mathbf{C}$. The measurement error variance σ_ε^2 is input using the `SV` option, while the state shock variance σ_ξ^2 comes in through the `SW` option.

The data are read from a plain text file[3] into a time series called `NILE`, and the option `Y = NILE` is used to provide the y_t information for the measurement equation.

As with other unobserved components (UC) models, the state has non-stationary dynamics. To handle the initial conditions, we can use the option `PRESAMPLE = DIFFUSE`, which indicates that the initial condition for the state is fully diffuse. This is implemented using the "exact" method of Koopman (1997) and Durbin and Koopman (2001). The same outcome will be obtained using the more flexible `PRESAMPLE = ERGODIC`, which analyzes the transition matrix and determines its roots.

3.1. Kalman smoothing

For now, we will take the component variances as given, and discuss estimation in Section 3.2. We will peg them at $\sigma_\varepsilon^2 = 15099$ and $\sigma_\xi^2 = 1469.1$, which are the maximum likelihood values. The instruction for Kalman smoothing with the Nile data is:[4]

```
dlm(a = 1.0, c = 1.0, sv = 15099.0, sw = 1469.1, presample = diffuse,$
  y = nile, type = smooth) / xstates vstates
```

`TYPE = SMOOTH` chooses Kalman smoothing. The default analysis is Kalman filtering—the extra calculations for Kalman smoothing are not done unless requested. The `XSTATES` parameter gets the smoothed state estimates and `VSTATES` gets the smoothed state variances. Since the state vector is (in almost all cases) bigger than a single element, `XSTATES` is a time series of vectors and `VSTATES` is a time series of (symmetric) matrices. Code for generating 90% confidence intervals and graphing them is given next:

```
set a     = %scalar(xstates)
set p     = %scalar(vstates)
set lower = a + sqrt(p)*%invnormal(.05)
set upper = a + sqrt(p)*%invnormal(.95)
graph(footer = "Figure 1. Smoothed state and 90% confidence intervals") 4
# nile
# a
# lower / 3
# upper / 3
```

`SET` is the main **RATS** instruction for creating and transforming time series. The `%SCALAR` function selects the first element out of a vector or matrix, so the series `A` will be the time series of estimated states, and `P` the time series of estimated variances. `GRAPH` is the time series graphing instruction; the `/ 3` on the last two lines forces the upper and lower bounds to use the same color or pattern. The graph produced by this is Figure 1.

[3] **RATS** can also read data from **Excel** XLS and XLSX files, MATLAB, Stata and EViews native files, among others.

[4] **RATS** input is case-insensitive. For readability, we will use lower case in the examples, and upper case in the text. **RATS** uses `$` to show that an input line is continued to the next—no symbol is required to end a normal input line.

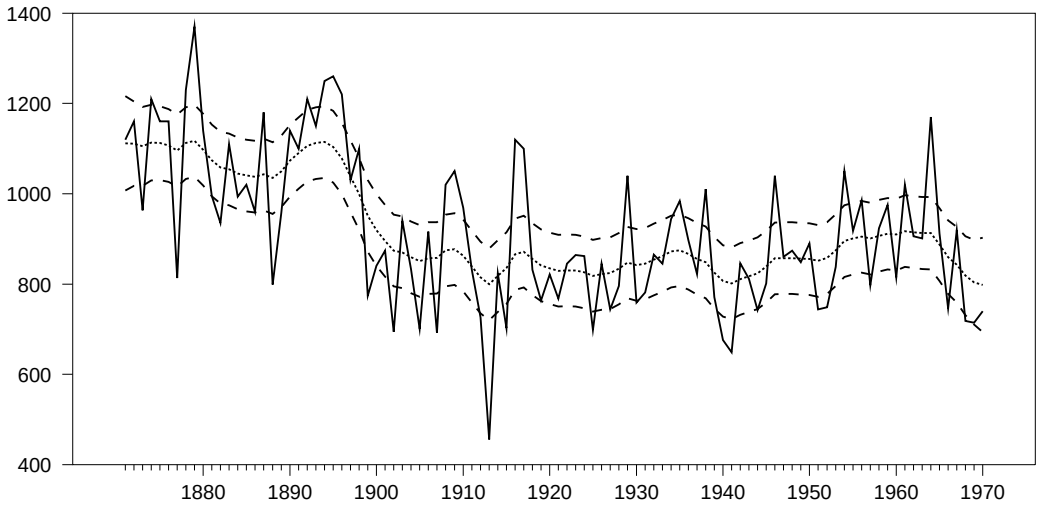

Figure 1: Smoothed state and 90% confidence interval.

3.2. Estimation of hyperparameters

The DLM instruction will always, as a side effect, compute the log likelihood of the model given the input variances. This can be maximized, with a wide range of choices for optimization, allowing for both derivative-based hill-climbing techniques, and slower but more flexible search methods. It also has the ability to (easily) incorporate equality or inequality constraints.

One way to estimate the two variances in the local level model is:

```
nonlin psi
compute psi = 0.0
dlm(a = 1.0, c = 1.0, sv = 1.0, sw = exp(psi), y=nile,$
  presample = diffuse, method = bfgs, var = concentrate)
```

The NONLIN instruction declares the set of free parameters to be estimated—here, that is $\psi = \log\left(\sigma_\xi^2/\sigma_\varepsilon^2\right)$. The measurement error variance is concentrated out, which can sometimes be helpful in improving the behavior of difficult estimation problems. The estimation method being used here is the hill-climbing method BFGS. The output is shown in Table 2.[5] Note that, while there are 100 data points, the likelihood is calculated using only the final 99 of them. This is done automatically here because of the diffuse initial conditions—the predictive variance for observation 1 is infinite, and so it is dropped from the calculation of the likelihood. DLM has an additional option CONDITION which can control the number of data points which are included in the filtering calculations, but omitted from the likelihood used for estimation. This is generally not needed, since DLM handles the diffuse states automatically, but is useful when the number of non-stationary states is not known *a priori*, if, for instance, autoregressive parameters are being estimated.

If we want to estimate both variances directly, we can do that with:

[5]The standard errors (and the covariance matrix more generally) are taken from the estimate of the inverse Hessian built up sequentially by the BFGS updating algorithm.

```
DLM - Estimation by BFGS
Convergence in 6 Iterations.  Final criterion was 0.0000002 <= 0.0000100
Annual Data From 1871:01 To 1970:01
Usable Observations              100
Rank of Observables               99
Log Likelihood             -632.5456
Concentrated Variance     15098.5298

    Variable               Coeff       Std Error     T-Stat      Signif
1.  PSI                 -2.329899732  1.012317234  -2.30155    0.02136051
```

Table 2: Estimation with concentrated variance.

```
DLM - Estimation by BFGS
Convergence in 9 Iterations.  Final criterion was 0.0000000 <= 0.0000100
Annual Data From 1871:01 To 1970:01
Usable Observations            100
Rank of Observables             99
Log Likelihood           -632.5456

    Variable               Coeff       Std Error     T-Stat      Signif
1.  SIGSQEPS          15098.510028  3126.130999   4.82978     0.00000137
2.  SIGSQXI            1469.172357  1266.235944   1.16027     0.24593993
```

Table 3: Estimation with both variances.

```
nonlin sigsqeps sigsqxi
stats(noprint) nile
compute sigsqeps = 0.5*%variance, sigsqxi = 0.1*sigsqeps
*
dlm(a = 1.0, c = 1.0, sv = sigsqeps, sw = sigsqxi, y = nile,$
  method = bfgs, presample = diffuse) 1871:1 1970:1
```

Direct estimation of the variances requires a bit more care with guess values. This uses scalings of the series sample variance, which should get the order of magnitude correct. The output is in Table 3.

While not important in this case, the NONLIN instruction can also handle various constraints on the parameters, either equality or inequality. With no change to the setup, we could estimate this with σ_ξ^2 pegged to zero (which here gives a model with a fixed mean) using

```
nonlin sigsqeps sigsqxi=0.0
dlm(a = 1.0, c = 1.0, sv = sigsqeps, sw = sigsqxi, y = nile,$
  method = bfgs, presample = diffuse) 1871:1 1970:1
```

In a more complex model, where there is some chance that a component variance might be zero, the NONLIN instruction can be used to set an inequality constraint:

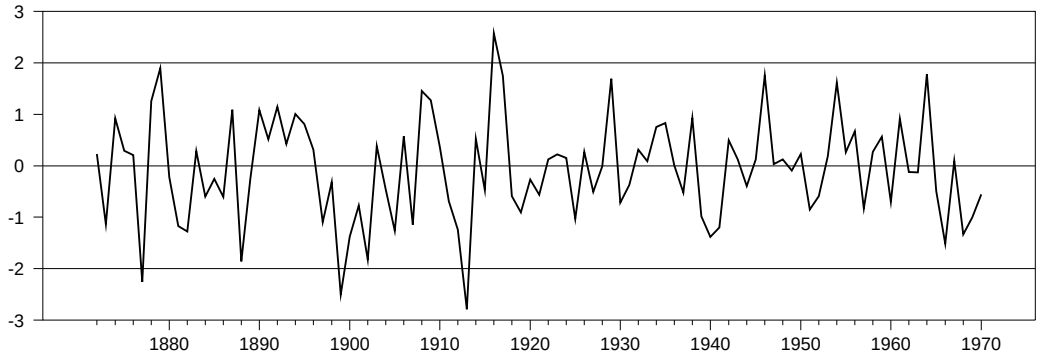

Figure 2: Standardized prediction errors.

```
nonlin sigsqeps sigsqxi>=0.0
dlm(a = 1.0, c = 1.0, sv = sigsqeps, sw = sigsqxi, y = nile,$
  method = bfgs, presample = diffuse) 1871:1 1970:1
```

This uses a penalty function variation on BFGS. Since it is quite a bit slower than standard BFGS, we generally do not recommend using it unless the simpler unconstrained estimates fail to provide values in range. The equality constraints from the previous case, on the other hand, are done by taking the constrained parameter out of the parameter set and using standard BFGS, so it actually runs faster than unconstrained BFGS.

3.3. Diagnostics

The most straightforward diagnostics come from the standardized residuals. These can be computed with the help of the `VHAT` and `SVHAT` options. `VHAT` is used to fetch the prediction errors and `SVHAT` the predictive variance.[6] Again, these will be in the form of a `VECTOR` (for `VHAT`) and a `SYMMETRIC` matrix (for `SVHAT`) to allow for the possibility of multiple observables. The following generates standardized predictive errors (into the series `EHAT`), graphs them (Figure 2) and does a standard set of diagnostics on the recursive residuals (output in Table 4):

```
dlm(a = 1.0, c = 1.0, sv = sigsqeps, sw = sigsqxi, y = nile,$
  method = bfgs, presample = diffuse,$
  vhat = vhat, svhat = svhat) 1871:1 1970:1
set ehat = %scalar(vhat)/sqrt(%scalar(svhat))
graph(footer = "Standardized residual", vgrid = ||-2.0, 2.0||)
# ehat
@STAMPDiags(ncorrs = 9) ehat
```

The `VGRID = ||-2.0, 2.0||` option on the `GRAPH` adds the horizontal lines at ± 2. Note that, because of the diffuse prior, the first standardized error is omitted. This is handled automatically in the code because the `SVHAT` for 1871:1 is a missing value.

The diagnostics in Table 4 include a Ljung-Box Q test for serial correlation, a Jarque-Bera normality test and a Goldfeld-Quandt style test for heteroscedasticity. The `STAMPDiags`

[6] Several of the possible outputs, such as `VHAT` and `SVHAT` have different meanings for different types of analysis. For instance, with Kalman smoothing, `VHAT` is the *smoothed* estimate of $\mathbf{v}_t$.

	Statistic	Sig. Level
Q(9-1)	8.84	0.3557
Normality	0.12	0.9441
H(33)	0.61	0.1650

Table 4: State space model diagnostics.

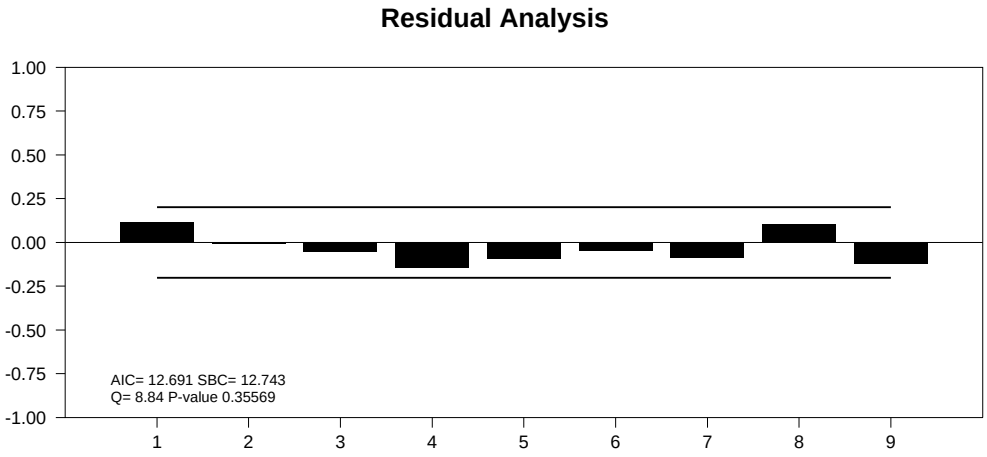

Figure 3: Autocorrelations of prediction errors.

procedure[7] also produces the graph of autocorrelations seen in Figure 3.

Durbin and Koopman (2001) recommend also computing auxiliary residuals, which are the Kalman smoothed estimates for the measurement errors and state disturbances. Large values for these can help identify outliers (in the measurement errors) or structural shifts (in the state disturbances). These can be obtained using the VHAT and WHAT options when Kalman smoothing. The results returned from those are standardized to mean zero, unit variance.

```
dlm(a = 1.0, c = 1.0, sv = sigsqeps, sw = sigsqxi, y = nile,$
  type = smooth, presample = diffuse,$
  vhat = vhat, what = what)
*
set outlier = %scalar(vhat)
diff(standardize) outlier
set break   = %scalar(what)
diff(standardize) break
```

The following graphs both of these. This uses SPGRAPH instructions to create a graph page with two panes. The result is Figure 4.

[7]The name of the procedure comes from **STAMP** (Koopman, Harvey, Doornik, and Shephard 2010), which uses these as standard diagnostics

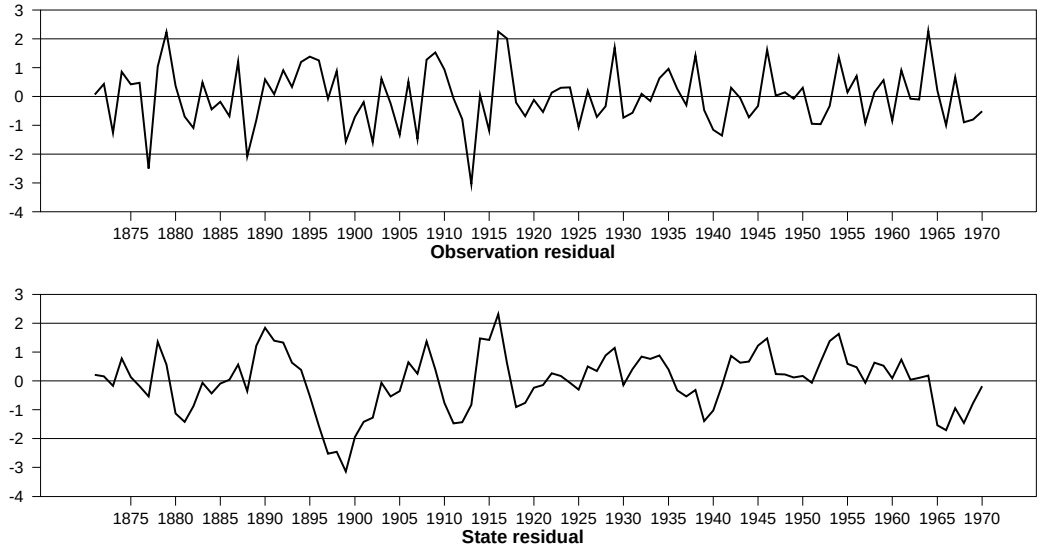

Figure 4: Diagnostic plots for auxiliary residuals.

```
spgraph(vfields = 2,$
  footer = "Diagnostic plots for auxiliary residuals")
graph(vgrid = ||-2.0, 2.0||, hlabel = "Observation residual")
# outlier
graph(vgrid = ||-2.0, 2.0||, hlabel = "State residual")
# break
spgraph(done)
```

3.4. Forecasts

Out-of-sample forecasts *can* be generated by simply running a Kalman filter past the end of the data set. When the Y value is missing, `DLM` does the Kalman "update" step but not the "correction". This is how embedded missing values are handled. For out-of-sample forecasts, however, it is generally more straightforward to Kalman filter through the observed data, then run a separate filter into the forecast range.

This next code segment uses the `X0` and `SX0` options to feed in the final estimated mean and variance for the states (from Kalman filtering over the sample) into the Kalman filter for the forecast range. The `YHAT` and `SVHAT` options are used to get the prediction and the predictive error variance for the dependent variable. You can also get the predicted value of the state and its predictive variance using the standard state parameters.

```
dlm(a = 1.0, c = 1.0, sv = 15099.0, sw = 1469.1, presample = diffuse,$
  y = nile, type = filter) / xstates vstates
dlm(a = 1.0, c = 1.0, sv = 15099.0, sw = 1469.1,$
  x0 = xstates(1970:1), sx0 = vstates(1970:1),$
  yhat = yhat, svhat = svhat) 1971:1 1980:1
```

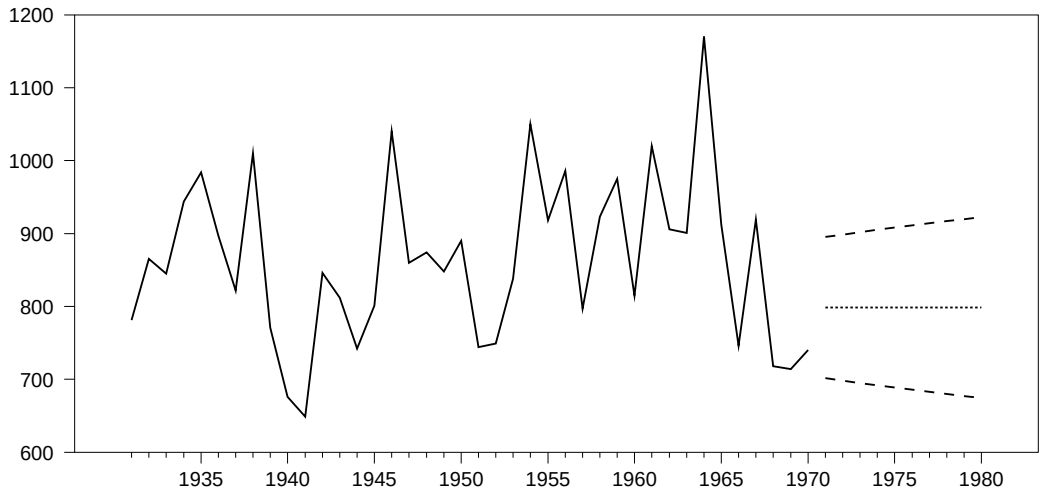

Figure 5: Out-of-sample forecasts with 50% CI.

```
set forecast 1971:1 1980:1 = %scalar(yhat)
set stderr   1971:1 1980:1 = sqrt(%scalar(svhat))
```

The following organizes a graph of the forecasts with their 50% confidence interval. Only forty years of actual data are included to give the forecast range enough space. This produces Figure 5.

```
set lower 1971:1 1980:1 = forecast + %invnormal(.25)*stderr
set upper 1971:1 1980:1 = forecast + %invnormal(.75)*stderr
graph(footer = "Out-of-sample Forecasts with 50% CI") 4
# nile     1931:1 1970:1
# forecast / 2
# lower    / 3
# upper    / 3
```

3.5. Simulations

There are two choices for random simulations of a model: `TYPE = SIMULATE` chooses unconditional simulation, where shocks for the states and measurements are drawn independently, and `TYPE = CSIMULATE`, where they are drawn subject to the requirement that the observed data are produced. `TYPE = SIMULATE` will generally be used in out-of-sample operations. In this section, we will demonstrate unconditional simulation, with conditional simulations described in Section 3.6.

To simulate out-of-sample, Kalman filtering is used through the observed range of the data to get the end-of-period estimates of the mean and variance of the state. Here, we will generate 10000 realizations for the process over the next fifty periods. The maximum flow for each realization is recorded. The percentiles are computed once the simulations are done. This could be used, for instance, to estimate the level for 50-year or 100-year floods. The results are shown in Table 5.

Statistics on Series MAXFLOW			
Observations	10000		
Minimum	609.753641	Maximum	2020.295589
01-%ile	795.832123	99-%ile	1675.898684
05-%ile	883.317055	95-%ile	1493.236460
10-%ile	933.600888	90-%ile	1401.955157
25-%ile	1024.391101	75-%ile	1270.733608
Median	1136.462805		

Table 5: Percentiles from maximum simulated flows.

```
dlm(a = 1.0, c = 1.0, sv = 15099.0, sw = 1469.1, presample = diffuse,$
  y = nile, type = filter) / xstates vstates
compute ndraws = 10000
set maxflow 1 ndraws = 0.0
do reps = 1, ndraws
   dlm(a = 1.0, c = 1.0, sv = 15099.0, sw = 1469.1,$
     x0 = xstates(1970:1), sx0 = vstates(1970:1),$
     type = simulate, yhat = yhat) 1971:1 2020:1 xstates
   set simflow 1971:1 2020:1 = %scalar(yhat)
   ext(noprint) simflow
   compute maxflow(reps) = %maximum
end do reps
stats(fractiles, nomoments) maxflow
```

3.6. Conditional simulations

Drawing the states and disturbances *conditional* on the observed data is more complicated, but more useful than the unconditional draws. The Kalman smoother gives us the mean and covariance matrix of the states and disturbances individually, but that is not enough to allow us to do draws, since conditional on the data, the states are highly correlated. Conditional simulation can be done with a pair of Kalman smoothing passes, one on simulated data, as described in Durbin and Koopman (2002), which is the algorithm used in **RATS**.

The main use of conditional simulation is in Bayesian techniques such as Markov Chain Monte Carlo. In fact, it is sometimes called (inaccurately) "Carter-Kohn", after the (alternative) algorithm described in Carter and Kohn (1994) as a step in a Gibbs sampler. For the Gibbs sampler, the states and/or disturbances are added to the parameter set, and conditional simulation gives a draw for those given the data and the underlying model parameters. The next step would then be to produce a draw for those model parameters conditional on the states and disturbances, which often takes quite a simple form once the latent information from the state space model can be treated as known.

We now examine the model estimated in Section 3.2. The two hyperparameters are modeled as the precision h of the measurement error and the relative variance ψ of the state shock to the measurement error. These are given very loose priors, with h being inverse gamma with 1 degree of freedom and ψ being gamma with 1 degree of freedom. We start each at the mean of its prior.

```
compute nuh = 1.0
compute s2h = 100.0^2
compute hdraw = nuh/s2h
*
compute nupsi = 1.0
compute s2psi = 0.1
compute psidraw = s2psi/nupsi
```

In the example, the Gibbs sampler is run with 1000 burn-in draws and 2000 keeper draws. `DLM` with `TYPE = CSIMULATE` does a draw from the joint distribution of the measurement error and state disturbances conditional on the observed values for Y. The `WHAT` and `VHAT` options are used to get the simulated values of the disturbances, while the states parameter gets the simulated values of the states. This does the simulation conditional on the current draws for h and ψ.

```
dlm(a = 1.0, c = 1.0, sv = 1.0/hdraw, sw = psidraw/hdraw, y = nile,$
   type = csimulate, presample = diffuse,$
   what = what, vhat = vhat) / xstates
```

The hyperparameters are then drawn conditional on the just-created draws for the disturbances. For this, we need the sums of squares for each of the two disturbances. This is easily done with the `SSTATS` instruction, which can compute sums, means, and other types of statistics on general expressions.

```
sstats / %scalar(vhat)^2>>sumvsq %scalar(what)^2>>sumwsq
```

Using these, we can draw first ψ, then h:

```
compute psidraw = (hdraw*sumwsq + nupsi*s2psi)/%ranchisqr(%nobs + nupsi)
compute hdraw =$
   %ranchisqr(nuh + %nobs*2.0)/(nuh*s2h + sumvsq + sumwsq/psidraw)
```

Estimates of the local level and its 90% confidence interval are shown in Figure 6. This is similar to Figure 1, but allows for the uncertainty regarding the hyperparameters.

This computes and graphs (Figure 7) estimated density functions for the two parameters:

```
density(smooth = 1.5) hgibbs 1 ndraws xh fh
density(smooth = 1.5) psigibbs 1 ndraws xpsi fpsi
*
spgraph(hfields = 2, footer = "Estimated Density Functions")
scatter(style = lines, header = "Precision")
# xh fh
scatter(style = lines, header = "Relative Variance")
# xpsi fpsi
spgraph(done)
```

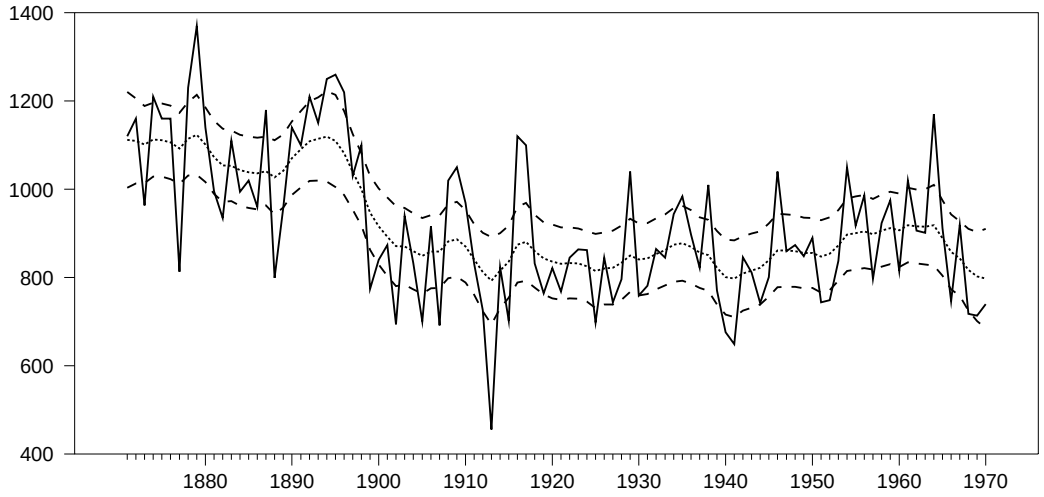

Figure 6: Local level and 90% CI from Gibbs sampling.

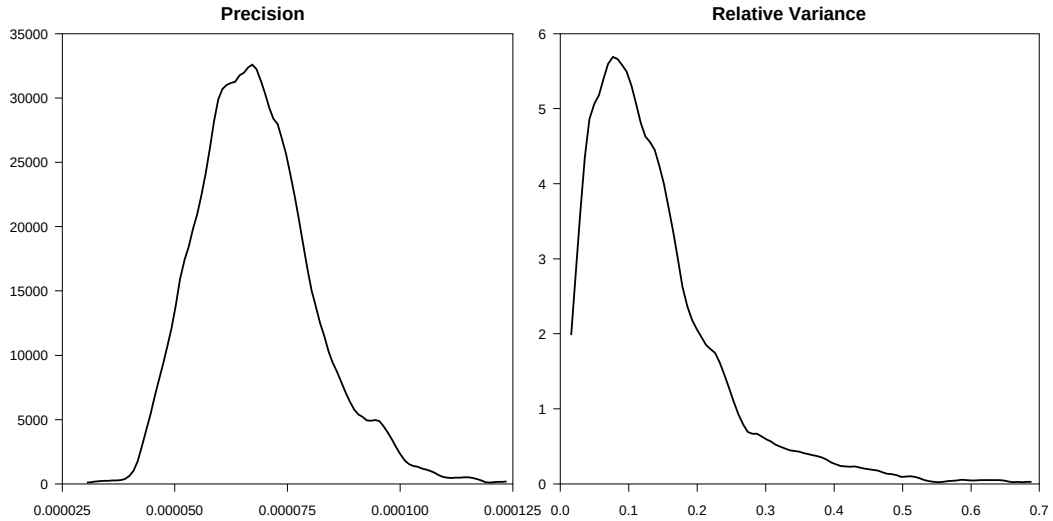

Figure 7: Estimated densities of parameters.

4. A larger model

A recent addition to **RATS** is the instruction `DSGE` (for dynamic stochastic general equilibrium (model)), which takes a model with expectational terms and solves it symbolically for a backwards-looking state space representation. If necessary, it will linearize or log-linearize the model to create an approximate state space form. The combination of `DSGE` and `DLM` can be used to evaluate the likelihood (for Gibbs sampling) or directly estimate by maximum likelihood the deep parameters in a DSGE. As an example, we include with RATS a program for replicating the estimation of the model from Ireland (2004).

The model that we will show here does not have expectational terms, but instead, is a dynamic

model in a non-standard form.[8] DSGE can also help here by solving out for a standard state space representation. The original model is from Sargent (1977). The two observables are money growth (μ_t) and inflation (x_t). With two observables, we need two shock processes: one will be to money demand, the other to money supply. The model (reduced to growth rates) can be written as:

$$\begin{aligned} x_t &= x_{t-1} + a_{1t} - \lambda a_{1t-1} \\ \mu_t &= (1-\lambda)\, x_{t-1} + \lambda \mu_{t-1} + a_{2t} - \lambda a_{2t-1} \\ a_{1t} &= (\lambda + (1-\lambda)\,\alpha)^{-1} (\varepsilon_t - \eta_t) \\ a_{2t} &= (\lambda + (1-\lambda)\,\alpha)^{-1} ((1+\alpha(1-\lambda))\,\varepsilon_t - (1-\lambda)\eta_t) \end{aligned}$$

The underlying shocks are ε_t and η_t, which are assumed to be independent. Since those are both tangled up in the definitions of a_1 and a_2, we will use separate equations to put them into the model. The model set up is[9]

```
frml(identity) f1 = x - (x{1} + a1 - lambda*a1{1})
frml(identity) f2 = mu-$
    ((1-lambda)*x{1} + lambda*mu{1} + a2 - lambda*a2{1})
frml(identity) f3 = a1 - 1.0/(lambda + (1 - lambda)*alpha)*(eps - eta)
frml(identity) f4 = a2 - (1.0/(lambda + (1 - lambda)*alpha)*$
                      ((1 + alpha*(1 - lambda))*eps - (1 - lambda)*eta))
frml           d1 = eps
frml           d2 = eta
group cagan f1 f2 f3 f4 d1 d2
dsge(model = cagan, a = adlm, f = fdlm) x mu a1 a2 eps eta
```

The DSGE instruction solves symbolically for a state space representation (returned by the A and F options) given a particular set of deep parameters (here λ and α). It requires the user to list the endogenous variables—the order of listing determines the order of placement within the state space model.

Note that x_t and μ_t are cointegrated with $x_t - \mu_t$ being stationary—the process has mixed stationary and non-stationary dynamics. As a result, a fully diffuse prior is not technically correct. However, with the option PRESAMPLE = ERGODIC, the DLM instruction can handle this automatically, analyzing the transition matrix to isolate the unit roots and create a partially diffuse prior.

If we want to estimate the free parameters of this model (λ, α and the two variances), we need to use the DSGE instruction at the start of every function evaluation to get the revised system matrices. The most convenient way to do this is to define a FUNCTION, which does the calculation and creates the required matrices (called here ADLM and FDLM).

```
function EvalModel
dsge(model = cagan, a = adlm, f = fdlm) x mu a1 a2 eps eta
end EvalModel
```

This technique is applied with many models, since the transition matrix is often very sparse when there are more than a few states. In this case, we compute the full matrices, but you

[8] Current values of the states are on both sides of the equation.

[9] In **RATS**, the notation series{n} is used for *lag* n. For use with DSGE, expectational terms are entered as leads (negative lag numbers).

can also use a function to "poke" values which depend upon the parameters into the proper locations in a larger matrix.

The shocks are both in the state equation, so there is no direct measurement error, thus no `SV` option. The model can be estimated with:

```
nonlin alpha lambda sig_eta sig_eps
dlm(start = %(EvalModel(), sw = %diag(||sig_eps^2, sig_eta^2||)),$
  a = adlm, f = fdlm, y = ||x, mu||, c = cdlm, sw = sw,$
  presample = ergodic, pmethod = simplex, piters = 5, method = bfgs)
```

At the start of each function evaluation, the `EvalModel` function is called to solve for state space matrices, and the (diagonal) matrix `SW` is created from the standard deviations. The option `y = ||x, mu||` describes the 2-vector of observables. This uses an option available on `DLM` (and other estimation instructions in **RATS**) to use two separate optimization methods. `PMETHOD = SIMPLEX` (`PMETHOD` for preliminary method) uses a small number of derivative-free simplex iterations at the start of the estimation, to refine guess values before switching to the derivative-based BFGS. This can be helpful in many types of models where the log likelihood function may be ill-behaved.

5. Conclusion

This paper has used several examples to give a taste of how **RATS** can be used with state space models. The `DLM` instruction has many options, allowing it to handle a wide range of tasks. Its internal calculations for filtering, smoothing and simulation have been highly optimized. When combined with the programming flexibility of the **RATS** package, many models which are quite cumbersome when done with matrix languages or less flexible packages can be done simply and quickly. We invite you to check our web site or e-mail us for more information.

References

Carter CK, Kohn R (1994). "On Gibbs Sampling for State Space Models." *Biometrika*, **81**(3), 541–553.

Commandeur JJF, Koopman SJ (2007). *An Introduction to State Space Time Series Analysis*. Oxford University Press, Oxford.

Doan TA (2010). "Practical Issues with State-Space Models with Mixed Stationary and Non-Stationary Dynamics." *Estima Technical Paper 1*. URL http://www.estima.com/resources_articles.shtml.

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

Durbin J, Koopman SJ (2002). "A Simple and Efficient Simulation Smoother for Time Series Models." *Biometrika*, **89**(3), 603–616.

Estima (2010). ***RATS**: Regression Analysis of Time Series*. Evanston, IL. Version 8, URL http://www.estima.com/.

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

Ireland P (2004). "A Method for Taking Models to the Data." *Journal of Economic Dynamics and Control*, **28**(6), 1205–1226.

Koopman SJ (1997). "Exact Initial Kalman Filtering and Smoothing for Nonstationary Time Series Models." *Journal of the American Statistical Association*, **92**(6), 1630–1638.

Koopman SJ, Harvey AC, Doornik JA, Shephard N (2010). ***STAMP*** *8.3: Structural Time Series Analyser, Modeler and Predictor*. Timberlake Consultants, London.

Sargent TJ (1977). "The Demand for Money During Hyperinflations under Rational Expectations: I." *International Economic Review*, **18**(1), 59–82.

West M, Harrison J (1997). *Bayesian Forecasting and Dynamic Models*. Springer-Verlag, New York.

Affiliation:

Thomas Doan
Estima
1560 Sherman Ave #510
Evanston, IL 60201, United States of America
E-mail: tomd@estima.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 41, Issue 9 *Submitted:* 2010-02-01
May 2011 *Accepted:* 2010-12-06