Chapter 3

Estimating Variances

3.1 The Likelihood Function

In the local level, local trend and seasonal models, the only free parameters are the variances. To this point, we've pegged values for those. How can we *estimate* them?

The Kalman filter gives us the predictive density for $\mathbf{Y}_t$ given $t-1$ as:

$$\mathbf{Y}_t \sim N\left(\mu_t + \mathbf{C}_t'\mathbf{X}_{t|t-1}, \mathbf{C}_t'\Sigma_{t|t-1}\mathbf{C}_t + \mathbf{N}_t\right)$$

We can use this with the standard time-series trick of factoring by sequential conditionals to get the full sample likelihood:

$$p(\mathbf{Y}_1,\ldots,\mathbf{Y}_T) = p(\mathbf{Y}_1)p(\mathbf{Y}_2|\mathbf{Y}_1)p(\mathbf{Y}_3|\mathbf{Y}_1,\mathbf{Y}_2)\cdots p(\mathbf{Y}_T|\mathbf{Y}_1,\mathbf{Y}_2,\ldots,\mathbf{Y}_{T-1}) \quad (3.1)$$

The variances governing the state-space model are often called *hyperparameters*, since the states are also parameters, and may, in many cases, be the actual objects of interest. Our task is to maximize the likelihood over these hyperparameters, in order to better estimate the states.

It looks as if this should be quite straightforward. A pass through the Kalman filter produces the predictive errors and variances, and we can evaluate the log likelihood by summing the log densities of the Normals. Evaluating the log likelihood given the parameters *is* just that simple. What complicates this is that there is nothing in the likelihood itself that constrains the individual variances to be non-negative. The only requirement for evaluating the log likelihood element at t is that the predictive covariance matrix:

$$\mathbf{C}_t'\Sigma_{t|t-1}\mathbf{C}_t + \mathbf{N}_t$$

be positive definite. That means that $\mathbf{N}_t$ could be negative if the first term is positive enough and vice versa. And, unfortunately, this problem comes up all too often.

There are two main approaches for dealing with this. The first is to estimate the model unconstrained: if one variance comes in negative, zero it out and re-estimate. If that works, it should be the global maximum. If *two* variances come in negative, it's not as clear how to proceed. In that case, it's probably best to use RATS' ability to put inequality constraints on parameters.

The other method is to parameterize the variances in logs. Although that doesn't allow a true zero value, a variance which really should be zero will show up as something like -30 in logs.

It's also possible to use Bayesian methods rather than straight maximum likelihood. With a prior on the variance parameters that's constrained to be non-negative (an inverse chi-squared is the standard choice), you'll never even look at the likelihood at negative values.

There's one other numerical problem that can come up when you estimate the variances directly. Even non-zero variances can sometimes be *very* small. For instance, the variance on the trend rate shock is often a value like 10^{-8}. It can be very hard for numerical optimizers to deal with parameters that have such naturally small scales, since it's not easy to distinguish between a parameter which is naturally small and one that is small because it's really supposed to be zero. You can sometimes fix this fairly easily by just multiplying the data (*after* such things as log transformation) by 100. That increases all the component variances by a factor of 10^4, which is usually enough to fix this problem.

One other way to deal with the problem of scale is to concentrate out one variance. As has been mentioned several times, the estimates of the means for the states are the same if all variances are multiplied by the same constant. For simplicity, let's assume that we have just one observable. Suppose that we write one of the variances (typically the measurement error variance) as λ and write all other variances as something like $\lambda \times \theta$. The sequence of predictive densities can be written as:

$$e_t \sim N\left(0, \lambda\sigma_t^2\right)$$

where e_t is the prediction error and σ_t^2 is the predictive variance computed for $\lambda = 1$. We can do this because the prediction (and thus the prediction error) is independent of λ, and the prediction error variance just scales by λ.[1] The log likelihood is thus (ignoring constants):

$$\sum_t \left(-\frac{1}{2}\log\left(\lambda\sigma_t^2\right) - \frac{1}{2}\frac{e_t^2}{\lambda\sigma_t^2}\right) = -\frac{T}{2}\log\lambda - \frac{1}{2\lambda}\sum_t \frac{e_t^2}{\sigma_t^2} - \frac{1}{2}\sum_t \log\left(\sigma_t^2\right)$$

The maximizing value for λ is

$$\hat{\lambda} = \frac{1}{T}\sum_t \frac{e_t^2}{\sigma_t^2}$$

and the concentrated log likelihood reduces to

$$-\frac{T}{2}\log\hat{\lambda} - \frac{T}{2} - \frac{1}{2}\sum_t \log\left(\sigma_t^2\right) \tag{3.2}$$

which can be optimized over the reduced parameter set θ.

[1]Both e_t and σ_t^2 are functions of θ. We're suppressing that to keep the focus on λ.

What does this accomplish? The main advantage is that it takes one important part of the variance scale and allows it to be calculated exactly and directly—no numerical derivatives, no convergence checks. It also makes it more likely that you can come up with reasonable initial guess values for the other parameters, since they will be independent of scale of the data. Generally you just need to be within an order of magnitude or two on the guess values in order to get a well-behaved optimization.

If you do decide to concentrate out a variance, make sure you pick one that you think unlikely (impossible?) to be zero. If you standardized on a zero, the ratios for the other parameters won't be defined.

3.2 Estimating the Local Level Model

Example **3.1** shows four different parameter setups for estimating the two parameters in the local level model for the Nile river data. These are:

1. Both parameters included; both in direct (not logged) form
2. Both parameters included; both in log form
3. Measurement variance concentrated out; ratio in direct form
4. Measurement variance concentrated out; ratio in log form

In order to estimate parameters, you need to include a **NONLIN** instruction to indicate what parameters are to be estimated, give them initial guess values (at least in direct form, the default zero values won't work) and include an option on **DLM** to pick the optimization method to be used. With most simple models `METHOD=BFGS`, which is the main "hill-climbing" algorithm, will work fine.

The estimation can be quite sensitive to guess values, at least when you're not concentrating out one variance. In this example, we get a guess value for the measurement error variance with:

```
filter(type=centered,width=11,extend=repeat) nile / fnile
sstats(mean) / (nile-fnile)^2>>initsigsq
```

This takes a simple 11 term centered moving average as a crude estimate of the local level, and uses the mean squared difference between the data and that as the guess value. For either the local level or local trend model, this should be fairly close (within maybe 30%) for σ_ε^2, which is all you need. The ratio of the two variances is guessed to be around .01. That turns out to be low by a factor of 10, but it's close enough to work.

The first estimates are done with:

```
compute sigsqeps=initsigsq,sigsqeta=sigsqeps*.01
nonlin sigsqeps sigsqeta
dlm(a=1.0,c=1.0,y=nile,sv=sigsqeps,sw=sigsqeta,$
  presample=diffuse,method=bfgs)
```

The fourth method (which is what is used in DK) is done with:

```
nonlin psi
compute psi=log(.01)
dlm(a=1.0,c=1.0,y=nile,sv=1.0,sw=exp(psi),var=concentrate,$
  exact,method=bfgs)
disp "Sigsqeps" %variance
disp "Sigsqeta" %variance*exp(psi)
```

As this is parameterized, PSI is the log of the ratio $\sigma^2_\eta/\sigma^2_\varepsilon$. The differences here are:

1. We include the option VARIANCE=CONCENTRATE
2. We use SV=1.0 to peg **SV** as the standardized variance
3. %VARIANCE is the maximum likelihood estimate of the variance scale factor. We need to multiply the ratio by that in order to get the estimate of the second variance.

3.3 Estimating the Local Trend Model

Before you do this, you should first ask whether you really want to. Attempts to estimate the variances in local trend models often produce results which look “wrong”. In particular, the common problem is that maximum likelihood produces a trend which isn't stiff enough to fit our understanding of what a trend should be. In short, there's a reason that the HP filter has pegged rather than estimated ratios.

There are potentially three variance parameters in the local trend model. We'll start with all three, though most of the time, you'll find (as we will) that the level variance is zero. We set up the system matrices for the model using the **@LocalDLM** procedure allowing for both shocks.

```
@localdlm(type=trend,shocks=both,a=at,f=ft,c=ct)
```

For convenience, we'll use the **@LocalDLMInit** procedure to get the guess values for the two most important variances: the irregular and the trend rate. The σ^2_ξ is initialized at a fraction of the irregular variance.

```
@localdlminit(irreg=sigsqeps,trend=sigsqzeta) logy
compute sigsqxi=sigsqeps*.01
```

With two shocks in the state equation, the SW option needs a 2×2 matrix. If we (as typical) assume that the two state shocks are uncorrelated, this will be a diagonal matrix. The simplest way to input this is with the %DIAG function, which makes a square diagonal matrix out of a vector. Our first go at this is:

```
dlm(a=at,c=ct,f=ft,sv=sigsqeps,sw=%diag(||sigsqxi,sigsqzeta||),$
  presample=diffuse,y=logy,method=bfgs)
```

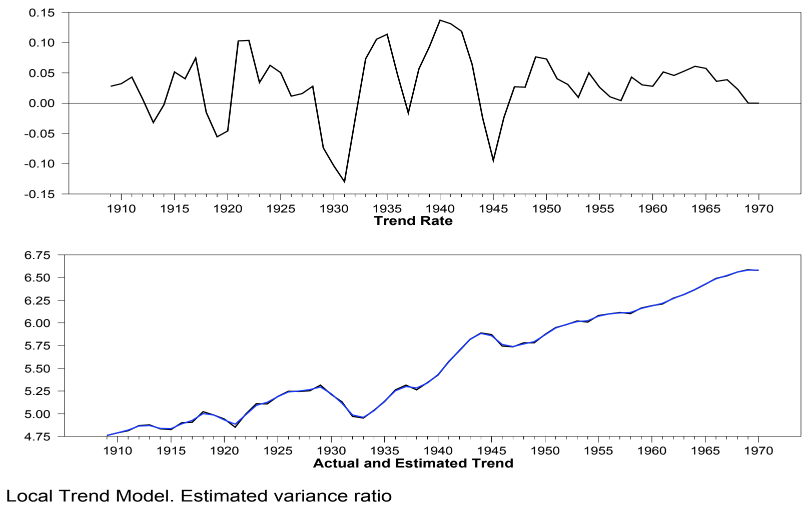

Figure 3.1: Local Trend Overfit

which gives us only one positive parameter. Interestingly, that turns out to be σ_ξ^2. However, when two parameters go negative, you really don't have any clear guidance. The "obvious" fix here is to zero out the variance that usually causes problems. Thus, we do:

```
nonlin sigsqeps sigsqzeta sigsqxi=0.00
@localdlminit(irreg=sigsqeps,trend=sigsqzeta) logy
dlm(a=at,c=ct,f=ft,sv=sigsqeps,sw=%diag(||sigsqxi,sigsqzeta||),$
  presample=diffuse,y=logy,method=bfgs)
```

Setting up the parameter set that way drops SIGSQXI out of the estimation output.[2] The likelihood is down a bit, but that's to be expected, since we put a constraint on.

The "problem" comes when we re-do this with TYPE=SMOOTH to get the full-sample estimates of the components:

```
dlm(a=at,c=ct,f=ft,sv=sigsqeps,sw=%diag(||sigsqxi,sigsqzeta||),$
  presample=diffuse,y=logy,type=smooth) / xstates
set trend = xstates(t)(1)
set rate  = xstates(t)(2)
```

This produces the estimates for the trend rate and for the trend shown in Figure 3.1. Notice that the trend rate is quite variable and the smoothed estimate of the "trend" almost exactly matches the data.

As a final cautionary tale, let's go back and peg to zero the two variances which were negative in the first estimation:

[2]We need to redo the initial values since the previous estimation made the two remaining parameters negative.

```
nonlin sigsqeps=0.00 sigsqzeta=0.00 sigsqxi
```

Redoing the estimates with this produces a higher likelihood. In fact, this seems to be the global maximum across the permitted parameter set. And yet it's pointless as a method of estimating a trend in GNP. Without a measurement error, the smoothed value of the trend is not just similar to the data, it is *exactly* the data.

Why does this model create so many problems? Given the length of economic time series, there just isn't strong enough information to sort out information by level of persistence, particularly into *three* categories (stationary, one unit root, double unit root). This is not surprising considering that we still haven't even settled the question of whether there's even one unit root in a series like this. In practice, some sort of variance ratio pegging or at least a prior on the variance ratios would seem to be the best bet.

3.4 Diagnostics

The main diagnostics are based upon the predictive errors. Under the assumption that the model is correct, these are an independent sequence. When scaled through by the standard error of prediction, they are i.i.d standard Normal.[3] Basically any test which can be applied to such a sequence can be used.

The procedure **@STAMPDiags** will do a standard set of diagnostics used by the STAMP software (co-written by Siem Jan Koopman). This includes a Ljung-Box Q test for serial correlation, a test for normality, and a test for heteroscedasticity. You can also use the procedure **@CUSUMTests**, which does the CUSUM and CUSUMQ tests for structural change as described in Brown, Durbin, and Evans (1975). In example **3.3**, we add the VHAT and SVHAT options to fetch the predictive residuals and their variance, convert to the recursive (or standardized predictive) residuals and run the tests:

```
dlm(a=1.0,c=1.0,y=nile,sv=exp(logeps),sw=exp(logeta),$
  presample=diffuse,method=bfgs,vhat=vhat,svhat=svhat)
set resids = %scalar(vhat)/sqrt(%scalar(svhat))
@STAMPDiags resids
@CUSUMTests resids
```

The only one of these that is even slightly indicative of a problem is the CUSUMQ test, which goes slightly out of bounds at 1917. However, this seems to be due to the three largest residuals being within a small span near that point. There certainly doesn't seem to be any systematic change to the variance.

[3]This is describing what happens with one observable. A generalization to a vector of observables requires pre-multiplying the prediction errors by the inverse of a factor of the covariance matrix.

Example 3.1 Estimating the Local Level Model

This demonstrates various methods for estimating the component variances in the local level model. It's based upon Illustration 2.10.3 in Durbin and Koopman (2012) and described in detail in Section 3.2.

```
open data nile.dat
calendar 1871
data(format=free,org=columns,skips=1) 1871:1 1970:1 nile
*
* Guess values
*
filter(type=centered,width=11,extend=repeat) nile / fnile
sstats(mean) / (nile-fnile)^2>>initsigsq
*
*************************************************************
*
* Method one: all parameters included, in direct form
*
compute sigsqeps=initsigsq,sigsqeta=sigsqeps*.01
nonlin sigsqeps sigsqeta
*
dlm(a=1.0,c=1.0,y=nile,sv=sigsqeps,sw=sigsqeta,$
  presample=diffuse,method=bfgs)
*
*************************************************************
*
* Method two: all parameters included, log form
*
nonlin logeps logeta
compute logeps=log(initsigsq),logeta=logeps-4.0
*
dlm(a=1.0,c=1.0,y=nile,sv=exp(logeps),sw=exp(logeta),$
  presample=diffuse,method=bfgs)
disp "Sigsqeps" exp(logeps)
disp "Sigsqeta" exp(logeta)
*
*************************************************************
*
* Method three: concentrating out measurement error variance. Direct
* form. SV=1.0, SW is the ratio parameter. Include the option
* VARIANCE=CONCENTRATE.
*
compute theta=.01
nonlin theta
*
dlm(a=1.0,c=1.0,y=nile,sv=1.0,sw=theta,var=concentrate,$
  presample=diffuse,method=bfgs)
disp "Sigsqeps" %variance
disp "Sigsqeta" %variance*theta
*
*************************************************************
*
```

```
* Method four: concentrating out measurement error variance. Log
* form on other ratios. SV=1.0, SW=exp(log ratio). Include the
* option VARIANCE=CONCENTRATE.
*
nonlin psi
compute psi=log(.01)
*
dlm(a=1.0,c=1.0,y=nile,sv=1.0,sw=exp(psi),var=concentrate,$
  presample=diffuse,method=bfgs)
disp "Sigsqeps" %variance
disp "Sigsqeta" %variance*exp(psi)
```

Example 3.2 Estimating the Local Trend Model

This demonstrates estimation of the component variances in the local trend model. See Section 3.3 for details.

```
open data nelsonplosser.rat
calendar(a) 1909
data(format=rats) 1909:1 1970:1 realgnp
set logy  = log(realgnp)
*
nonlin sigsqeps sigsqzeta sigsqxi
*
* Set up the DLM equations allowing for both shocks
*
@localdlm(type=trend,shocks=both,a=at,f=ft,c=ct)
*
* Get guess values for the variances for the trend rate and the
* irregular. Start with the level variance as a small fraction of
* the irregular.
*
@localdlminit(irreg=sigsqeps,trend=sigsqzeta) logy
compute sigsqxi=sigsqeps*.01
*
dlm(a=at,c=ct,f=ft,sv=sigsqeps,sw=%diag(||sigsqxi,sigsqzeta||),$
  presample=diffuse,y=logy,method=bfgs)
*
* Redo with sigsqxi pegged to 0
*
nonlin sigsqeps sigsqzeta sigsqxi=0.00
@localdlminit(irreg=sigsqeps,trend=sigsqzeta) logy
dlm(a=at,c=ct,f=ft,sv=sigsqeps,sw=%diag(||sigsqxi,sigsqzeta||),$
  presample=diffuse,y=logy,method=bfgs)
*
* Use type=smooth to get component estimates
*
dlm(a=at,c=ct,f=ft,sv=sigsqeps,sw=%diag(||sigsqxi,sigsqzeta||),$
  presample=diffuse,y=logy,type=smooth) / xstates
*
* Extract the local trend (first state) and the trend rate (second
* state)
```

```
*
set trend = xstates(t)(1)
set rate  = xstates(t)(2)
*
spgraph(vfields=2,footer="Local Trend Model. Estimated variance ratio")
 graph(hlabel="Trend Rate")
 # rate
 graph(hlabel="Actual and Estimated Trend") 2
 # logy
 # trend
spgraph(done)
*
* Redo estimation, pegging the two variances that were negative in
* the first run.
*
nonlin sigsqeps=0.00 sigsqzeta=0.00 sigsqxi
@localdlminit(irreg=sigsqeps,trend=sigsqzeta) logy
compute sigsqxi=sigsqeps*.01
dlm(a=at,c=ct,f=ft,sv=sigsqeps,sw=%diag(||sigsqxi,sigsqzeta||),$
  presample=diffuse,y=logy,method=bfgs,type=smooth) / xstates
set trend = xstates(t)(1)
set rate  = xstates(t)(2)
graph(hlabel="Actual and Estimated Trend") 2
# logy
# trend
```

Example 3.3 Diagnostics

Demonstrates various diagnostics for state-space models. This is adapted from Illustration 2.12.3 in Durbin and Koopman (2012) and is discussed in Section 3.4.

```
open data nile.dat
calendar 1871
data(format=free,org=columns,skips=1) 1871:1 1970:1 nile
*
* Guess values
*
filter(type=centered,width=11,extend=repeat) nile / fnile
sstats(mean) / (nile-fnile)^2>>initsigsq
*
* Estimate
*
nonlin logeps logeta
compute logeps=log(initsigsq),logeta=logeps-4.0
*
dlm(a=1.0,c=1.0,y=nile,sv=exp(logeps),sw=exp(logeta),$
  presample=diffuse,method=bfgs,vhat=vhat,svhat=svhat)
*
* Get recursive residuals
*
set resids = %scalar(vhat)/sqrt(%scalar(svhat))
```

```
*
* Compute standard diagnostics
*
@STAMPDiags resids
@CUSUMTests resids
```