Journal of Statistical Software

May 2011, Volume 41, Issue 8. *http://www.jstatsoft.org/*

Fitting State Space Models with EViews

Filip A. M. Van den Bossche
Hogeschool-Universiteit Brussel

Abstract

This paper demonstrates how state space models can be fitted in EViews. We first briefly introduce EViews as an econometric software package. Next we fit a local level model to the Nile data. We then show how a multivariate "latent risk" model can be developed, making use of the EViews programming environment. We conclude by summarizing the possibilities and limitations of the software package when it comes to state space modeling.

Keywords: EViews, Kalman filter, state space methods, unobserved components.

1. Introduction

EViews (Quantitative Micro Software 2007a,b,c) is a statistical software package for data analysis, regression and forecasting. As a direct successor of MicroTSP, EViews is especially powerful in analysing univariate and multivariate time series, but it also knows how to handle cross-sectional and panel data. EViews offers useful data management and econometric analysis tools and produces high-quality graphical and tabular model output. It comes with a windows-based graphical user interface and allows structuring and analyzing data by means of point-and-click commands and built-in windows, menus and dialogs. Alternatively, users can write their own programs using the command and batch processing language. The most recent release of the software package is EViews 7, but the current state space features were added to the program from version 4 onwards. The analyses in the paper at hand are fitted in version 6.

At the heart of an econometric analysis in EViews is the construction of objects. Data series and models are all stored in a workfile as separate objects which can be viewed in various ways. For an equation object (e.g. a regression model), one can ask for the model specification, the estimation output, the fitted values and residuals, etc. as alternate views on the same object.

For state space models, an object `sspace` should be created. This object offers specification

and estimation options for single or multiple dynamic equations in state space form and navigates the user to the application of the built-in Kalman filtering algorithm. Exogenous variables can be included in the state equations and variances for all equations can be specified in terms of model parameters. Model output is available in tabulated format or in graphs. New series can be created from the model output and subsequently stored as new objects for further analysis. These typically include one-step ahead and smoothed states and signals, filtered states and corresponding residuals and disturbances. More details can be found in the EViews User's Guide (Quantitative Micro Software 2007c, p. 383–406).

This paper illustrates how linear Gaussian state space models can be fitted to time series data in EViews. Section 2 presents the specification of a local level model for data on the annual flow volume from the river Nile in Aswan (1871–1970). Section 3 shows how a "latent risk model" (Bijleveld, Commandeur, Gould, and Koopman 2008) applied to Belgian road accident and exposure data can be developed and estimated in EViews. Section 4 concludes.

2. Case 1: The local level model applied to the Nile data

To analyze the Nile data, we need a workfile which contains the data. As is typical in EViews, a state space model is defined as an object within a workfile which contains, among others, the time series to be analyzed. The relevant object for a state space model specification is `sspace`. Creating a new state space object opens an empty specification window.

2.1. Model specification

Specification can be done in two ways. The first is to use EViews' auto-specification feature, which allows indication of the dependent variable(s), the regressors with fixed and/or recursive coefficients, the stochastic regressors and the variance specification. EViews subsequently derives the text representation of the model, which can be edited and estimated. The second – more general – way of describing a state space model is by using keywords and commands to define the measurement and state equations, the corresponding error terms and, if desired, aspects related to the estimation procedure like initial conditions and starting values for the parameters. This second method is more flexible and will be used here.

Using the appropriate commands, the measurement equation, state equation, errors and variances are defined. To estimate the local level model

$$\begin{aligned} y_t &= \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim \text{NID}(0, \sigma_\varepsilon^2), \\ \mu_{t+1} &= \mu_t + \xi_t, \qquad \xi_t \sim \text{NID}(0, \sigma_\xi^2), \end{aligned} \tag{1}$$

for the Nile data, we create an `sspace` object as shown in Figure 1.

Note that the created `sspace` object is named `LOCAL_LEVEL` and that it belongs to the workfile `NILEDATA` (all workfiles have extension `.wf1`). The created `sspace` object consists of three parts. In a first part (lines 1 and 2), the error terms for the measurement and state equation (`e1` and `e2`) are named using the keyword `@ename`. Contrary to other objects in EViews, error terms are not included in the equations of a state space object unless they have been explicitly specified. In this case, we include an error term for both equations.

The second part (lines 3 and 4) specifies the variances. Error variances are preceded by the `@evar` keyword and may be constants or expressions in terms of unknown parameters. In our

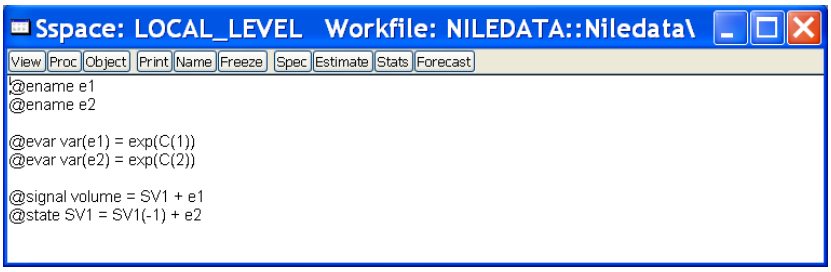


Figure 1: A state space object in EViews.

model, the variances are expressed as exponential functions of the coefficients `C(1)` and `C(2)`, to guarantee nonnegative variance estimates (Quantitative Micro Software 2007c). Using the `@evar` command, one can also specify covariances between error terms, for example `@evar cov(e1,e2)=0.5`. Variances and/or covariances that have not been specified are assumed to be equal to zero.

In the third part (lines 5 and 6), the measurement and state equations are defined. The `@signal` keyword specifies the measurement equation for the dependent variable `volume` and includes an unobserved level `SV1` to represent μ_t and an observation disturbance `e1` which corresponds to ε_t and has been declared in line 1. Signal equations may include expressions of the dependent variable, but no current values or leads of signal variables. Nonlinearities in the states and leads or lags of states are not allowed. Note that the `@signal` keyword may be dropped.

The `@state` equation defines the random walk for the unobserved model component μ_t. State equations should not include expressions of unobserved components (like `log(SV1)`) nor (lags or leads of) signal equation dependent variables, but may contain (possibly nonlinear transformations of) exogenous variables. They should be linear in the one-period lags of the states, where the one-period lag restriction is easily circumvented by including new state variables for higher order lags.

Error terms should not necessarily be named before specifying the state and observation equation. One can simply add the variance structure to define an error term, like in `@state SV1 = SV1(-1) + [var=exp(C(2))]`. This is called the "error variance specification" (Quantitative Micro Software 2007c, p. 389), whereas in our program we used the "named error" approach (Quantitative Micro Software 2007c, p. 391).

The model presented above can easily be constructed using the auto-specification feature in EViews. All we need to do is set `volume` as the dependent variable and include a unit random walk coefficient.

2.2. Estimation

Once the model has been specified as shown above, the unknown parameters for the variances and the unobserved component can be estimated. Estimation is done by maximum likelihood. The loglikelihood function in EViews (Quantitative Micro Software 2007c, p. 387) corresponds to the one given by Durbin and Koopman (2001, p. 138) and is refered to by Harvey as the "prediction error decomposition" (Harvey 1989, p. 126).

Unless stated otherwise, the starting values for the parameters `C(1)` and `C(2)` are those

Sspace: LOCAL_LEVEL
Method: Maximum likelihood (Marquardt)
Sample: 1871 1970
Included observations: 100
Convergence achieved after 155 iterations

	Coefficient	Std. Error	z-Statistic	Prob.
C(1)	9.600350	0.171398	56.01195	0.0000
C(2)	7.348705	0.565220	13.00151	0.0000

	Final State	Root MSE	z-Statistic	Prob.
SV1	795.5689	75.03906	10.60206	0.0000

Log likelihood	-645.1781	Akaike info criterion	12.94356
Parameters	2	Schwarz criterion	12.99566
Diffuse priors	1	Hannan-Quinn criter.	12.96465

Figure 2: Output of the local level model for the Nile data.

specified in the coefficient vector, which is yet another object in the EViews workfile. If necessary, the starting values can be changed by the user with `@param` statements.

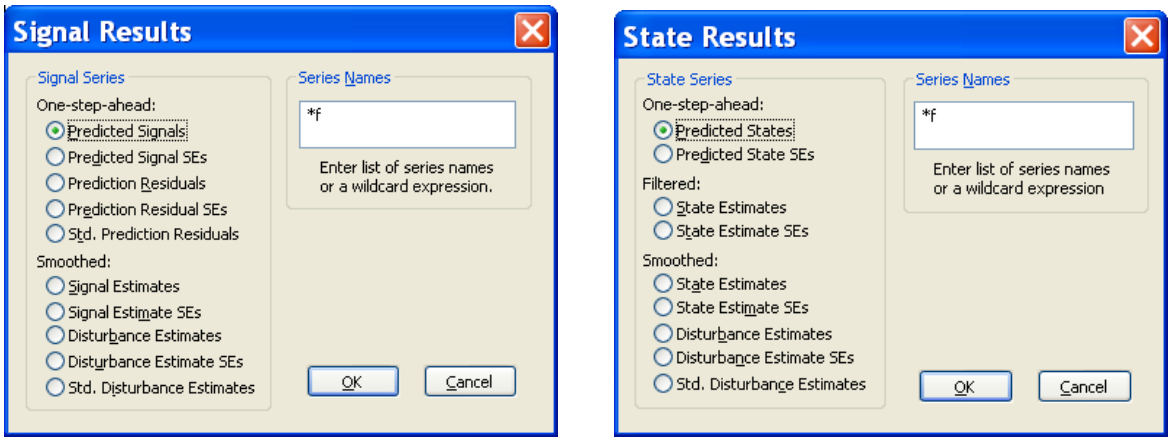
Usually, the end user should not handle the initial conditions. Whenever possible, the steady state conditions are solved for the mean a_1 and variance P_1 of the initial state vector α_1. Otherwise, estimation is started from a diffuse initialization. In case prior information on a_1 and P_1 is available, the user can supply initial values by means of the `@mprior` and `@vprior` keywords, for the mean and variance respectively.

EViews offers two first derivative methods for optimizing the loglikelihood function: Marquardt and Berndt-Hall-Hall-Hausman. The first is a modification of the Gauss-Newton algorithm, while the second builds on Newton-Raphson. EViews further allows setting the estimation sample, the maximum number of iterations and the convergence tolerance. Note the importance of the starting values, whichever optimization method is used. In general, "you may have to experiment to find good starting values" (Quantitative Micro Software 2007c, p. 626).

2.3. Results

Figure 2 shows the estimation output for the local level model fitted to the Nile data. The output shows that the model has been fitted on 100 observations using the Marquardt optimization algorithm. EViews needed 155 iterations to achieve a converging solution. At convergence the maximum of the log likelihood is found to be -645.1781. The coefficients `C(1)` and `C(2)` are the logs of the variances of the error terms for the measurement and state equations. We therefore estimate that $\sigma_\varepsilon^2 = \exp(9.6004) = 14769.9537$ and $\sigma_\xi^2 = \exp(7.3487) = 1554.1828$. The final state of the unobserved component is 816.7538. The value shown in the output, 795.5689, is the one-step ahead predicted value for the first out-of-sample period (Quantitative Micro Software 2007a, p. 428). The initial state value of the level is not reported, but can be found in the model output to be $\hat{\mu}_1 = 1119.9989$.

EViews allows creating new series based on the results of the estimation process. These can be generated using the appropriate keywords, or they can be selected from the menu screens (see for example the state and signal screens shown in Figure 3). Some of the generated series for



(a) "Make signal series" screen (b) "Make state series" screen

Figure 3: Creating signal and state series.

the analysis of the Nile data are summarized in Figure 4. Figure 4a shows the Nile volume data and the smoothed state estimates together with 90% confidence bands. Figure 4b contains the standardized prediction errors, which can be calculated from the output as the ratio of the prediction residuals and the prediction residuals standard errors. They can also readily be obtained from the EViews output by asking for the "standardized prediction residuals". Figures 4c and 4d show the auxiliary residuals for the signal and the state respectively. They are obtained by selecting the smoothed standardized disturbance estimates in the signal and state results (via the `proc/make signal series` menu). Alternatively, they are calculated by dividing the smoothed observation and state disturbance estimates ($\hat{\varepsilon}_t$ and $\hat{\xi}_t$) by their corresponding standard errors. Note that the graphs also include 95% confidence bands to check for outliers and structural breaks.

Once a series is generated, the usual statistical tools available in EViews can be applied. For example, in Figure 5 the correlogram with Box-Ljung statistics and the Jarque-Bera normality test for the standardized prediction errors are shown. Note, however, that these diagnostics are not readily available and should be generated outside the `sspace` object. This implies that the sample for generating the correlogram, as well as the associated degrees of freedom, are usually incorrect. In the standard setting, the correlogram is generated using the complete sample ($n = 100$) and the degrees of freedom in EViews are equal to the number of lags.

To be more precise, we generate the correlogram after a sample adjustment (`smpl 1872 1970`) and calculate the degrees of freedom as $k - w + 1$, where k is the number of lags in the Box-Ljung statistic and w is the number of diffuse priors (Commandeur and Koopman 2007). For example, for $Q(10)$ we find a value of 13.117 for $n = 99$ (instead of 13.310 for $n = 100$). Given two disturbance variances in the model, the corresponding p value should be based on a $\chi^2_{(10-2+1)}$ distribution, resulting in a p value of 0.1574 (instead of 0.2172 in Figure 5a). Correcting for the number of disturbance variances renders the test slightly more conservative, in the sense that the null hypothesis of independence will be rejected more often. For this model, the difference in degrees of freedom is small, however, and in practice the standard output may be used. After sample adjustment, the Jarque-Bera test for normality can easily be asked for (see Figure 5b: JB = 0.0417, p value = 0.9794).

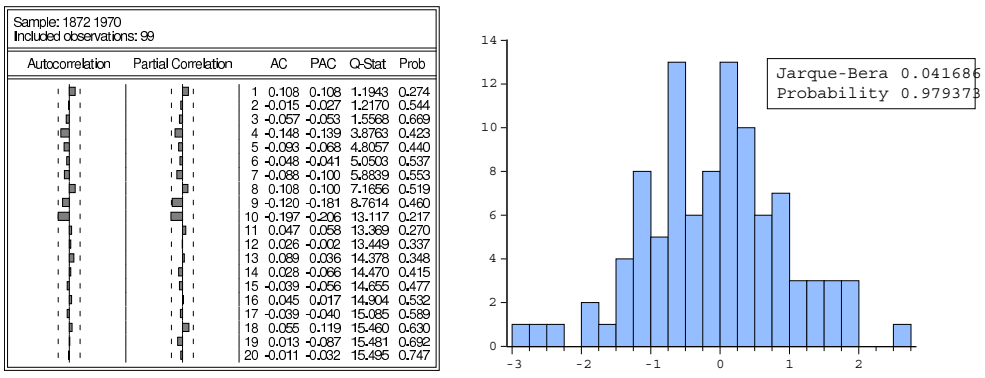
(a) Smoothed State and 90% CI

(b) Standardized Prediction Errors

(c) Auxiliary residuals (signal)

(d) Auxiliary residuals (state)

Figure 4: Nile data and Kalman filter output.



(a) Correlogram

(b) Normality test

Figure 5: Analysis of standardized prediction errors.

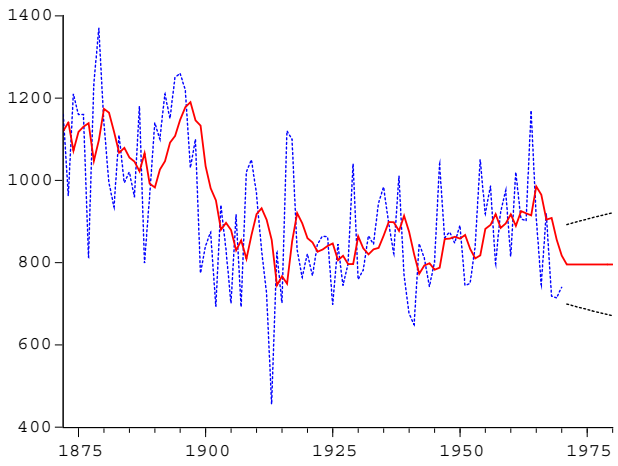


Figure 6: Forecasts and 50% confidence interval.

Finally, one-step-ahead forecasts and corresponding standard errors were generated from the `sspace` object for the years 1971–1980. The value of the level at $t = n + 1$ is forecasted to be 795.5689, and the forecast remains constant for the other years. The forecasts and the 50% confidence interval are shown in Figure 6.

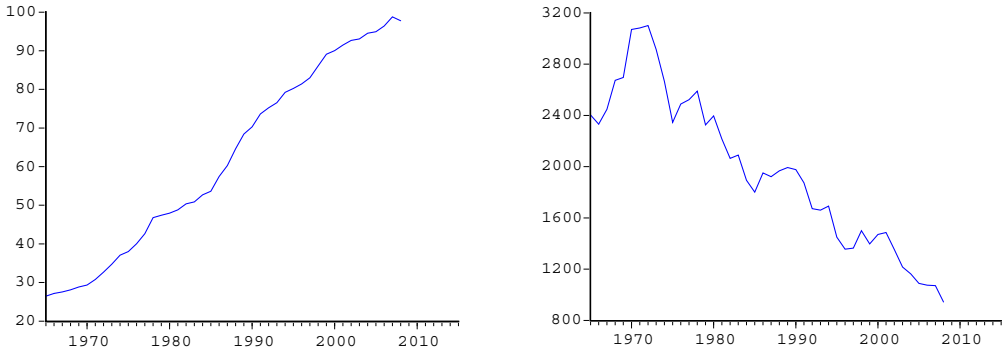
3. Case 2: Fitting a latent risk model in EViews

3.1. Introduction

In this section we show how the latent risk model (Bijleveld *et al.* 2008) can be fitted in EViews. Apart from the model specification, an iteration program is presented that can be used to facilitate convergence of the optimization procedure.

The application presented here belongs to the domain of macroscopic road safety modeling. One objective of road safety modeling is to describe and explain long term trends in the number of road fatalities. The annual number of fatalities is an important indicator of the road safety performance of a country. Policy makers refer to it to investigate past trends in road safety and to set targets for future improvement. In many countries, long-term quantitative objectives are expressed in terms of the number of fatalities (e.g. "half the number of fatalities by 2010"). In Belgium, new road safety targets have been formulated in 2007, aiming at no more than 500 fatalities in 2015.

Consider the time series in Figure 7, representing the yearly number of vehicle kilometres (Figure 7a) and the number of road fatalities (Figure 7b) in Belgium for the period 1965–2008. The number of vehicle kilometres is typically used as a measure of exposure to risk. It is assumed that the level of exposure is one of the major factors influencing the number of fatalities. In particular, if "risk" is defined as the ratio of the number of fatalities to the level of exposure, then the number of fatalities can be expressed as the level of exposure multiplied by the level of risk, thereby disentangling the number of fatalities in its major components.



(a) Number of vehicle kilometres ($\times 10^9$)

(b) Number of fatalities

Figure 7: Belgian road fatalities and vehicle exposure.

However, as both exposure and risk can never be faultlessly observed, we model them by means of unobserved components. A precursor of this application was presented in Van den Bossche (2006).

3.2. Model development

The latent risk model is a special case of the general Gaussian state space model which can be written as:

$$\begin{aligned} y_t &= Z_t\alpha_t + \varepsilon_t, & \varepsilon_t &\sim \text{NID}(0, H_t), \\ \alpha_{t+1} &= T_t\alpha_t + R_t\eta_t, & \eta_t &\sim \text{NID}(0, Q_t), \end{aligned} \tag{2}$$

for $t = 1, \ldots, n$. The first equation is the observation equation, in which y_t represents a $p \times 1$ vector containing the observed values at time t for the p dependent variables. The ε_t vector of dimension $(p \times 1)$ contains the p corresponding observation disturbances. These are assumed to be NID, with zero means and a variance-covariance structure collected in the $p \times p$ matrix H_t. Assuming m state components in this model, Z_t is a $p \times m$ observation matrix, and α_t is the state vector of order $m \times 1$. The transition matrix T_t is a block diagonal matrix of order $m \times m$. The m state disturbances are gathered in the $m \times 1$ vector η_t. They have zero means and an unknown variance-covariance matrix Q_t of order $m \times m$. Finally, R_t is usually an identity matrix of order $m \times m$, but it can also be a selection matrix of order $m \times r$, with $r < m$, containing the first r columns of the identity matrix. For further details on the formulation of this multivariate model, see Harvey (1989), Durbin and Koopman (2001) and the introductory article of this special volume.

To develop the latent risk model, consider a local linear trend model with two dependent variables ($p = 2$), namely the observed annual exposure or mobility M_t (expressed as the number of vehicle kilometres driven per year) and the observed annual number of fatalities F_t. Each observation equation is further described in two state equations, one for the level and one for the slope ($m = 4$). To model the mobility and fatalities series simultaneously in a multivariate model, define the vector y_t as (Bijleveld and Commandeur 2006):

$$y_t = \begin{pmatrix} y_t^{(1)} \\ y_t^{(2)} \end{pmatrix} = \begin{pmatrix} log(M_t) \\ log(F_t) \end{pmatrix}. \tag{3}$$

In addition, define the vectors α_t, ε_t and η_t, and the matrices T_t, R_t, Z_t, H_t and Q_t as follows:

$$\alpha_t = \begin{pmatrix} \mu_t^{(1)} \\ \nu_t^{(1)} \\ \mu_t^{(2)} \\ \nu_t^{(2)} \end{pmatrix}, \eta_t = \begin{pmatrix} \xi_t^{(1)} \\ \zeta_t^{(1)} \\ \xi_t^{(2)} \\ \zeta_t^{(2)} \end{pmatrix}, T_t = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, R_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$Z_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \varepsilon_t = \begin{bmatrix} \varepsilon_t^{(1)} \\ \varepsilon_t^{(2)} \end{bmatrix}, H_t = \begin{bmatrix} \sigma_{\varepsilon^{(1)}} & \operatorname{cov}(\varepsilon_t^{(1)}, \varepsilon_t^{(2)}) \\ \operatorname{cov}(\varepsilon_t^{(1)}, \varepsilon_t^{(2)}) & \sigma_{\varepsilon^{(2)}} \end{bmatrix}, \tag{4}$$

$$Q_t = \begin{bmatrix} \sigma^2_{\xi^{(1)}} & 0 & \operatorname{cov}(\xi^{(1)}, \xi^{(2)}) & 0 \\ 0 & \sigma^2_{\zeta^{(1)}} & 0 & \operatorname{cov}(\zeta^{(1)}, \zeta^{(2)}) \\ \operatorname{cov}(\xi^{(1)}, \xi^{(2)}) & 0 & \sigma^2_{\xi^{(2)}} & 0 \\ 0 & \operatorname{cov}(\zeta^{(1)}, \zeta^{(2)}) & 0 & \sigma^2_{\zeta^{(2)}} \end{bmatrix}.$$

These vectors and matrices completely define the latent risk model. Writing out these components in scalar notation yields the following two observation equations:

$$y_t^{(1)} = \mu_t^{(1)} + \varepsilon_t^{(1)}, \tag{5a}$$
$$y_t^{(2)} = \mu_t^{(1)} + \mu_t^{(2)} + \varepsilon_t^{(2)}, \tag{5b}$$

while the four state equations can be written as:

$$\mu_t^{(1)} = \mu_{t-1}^{(1)} + \nu_{t-1}^{(1)} + \xi_t^{(1)}, \tag{6a}$$
$$\nu_t^{(1)} = \nu_{t-1}^{(1)} + \zeta_t^{(1)}, \tag{6b}$$
$$\mu_t^{(2)} = \mu_{t-1}^{(2)} + \nu_{t-1}^{(2)} + \xi_t^{(2)}, \tag{6c}$$
$$\nu_t^{(2)} = \nu_{t-1}^{(2)} + \zeta_t^{(2)}. \tag{6d}$$

The first observation equation (5a) is for the log of the observed mobility (exposure). Equations (6a) and (6b) are the corresponding state equations for the mobility trend and slope components. The second observation equation (5b) represents the log of the number of fatalities, for which the dynamics (trend and slope) are determined by the state equations (6c) and (6d). Given the vector and matrix definitions in the equation set (4), covariances between trend and slope components are assumed to be zero. Covariances are estimated mutually for the trend errors, the slope errors and the observation errors.

Given the fact that this model is linear in the logarithms, it is essentially a multiplicative model, representing the number of fatalities as the product of exposure and risk. This is in line with the models developed by Oppe (1989, 1991). The difference with these models, however, is that the model at hand is multivariate in nature, and that the exposure and risk components are unobserved, without any assumption about their functional form. Hence the "latent risk" designation.

```
1   ' generate state space models and save loglikelihood

2   sspace solution
3   !loglik=-1000000

4   table results
5   results(1,1)="LogLik"
6   results(1,2)="Result"
7   results(1,3)="AIC"
8   results(1,4)="Schwarz"
9   results(1,5)="Hannan-Quinn"

10  !row = 1
11  for !i = 1 to 1000 step 1
12              smpl @all
13              series start = (-1+2*rnd)/100

14              for !j = 1 to 9  step 1
15                      !startvar = start(!j)
16                      param c(!j) !startvar
17              next

18  smpl 1965 2008

19  sspace ss1

20  'define the error terms
21  ss1.append @ename e1
22  ss1.append @ename e2
23  ss1.append @ename e3
24  ss1.append @ename e4
25  ss1.append @ename e5
26  ss1.append @ename e6

27  'define error variances and covariances
28  ss1.append @evar var(e1) =  c(1)^2+c(7)^2
29  ss1.append @evar var(e2) =  c(2)^2+c(8)^2
30  ss1.append @evar var(e3) =  c(3)^2+c(9)^2
31  ss1.append @evar var(e4) =  c(4)^2
32  ss1.append @evar var(e5) =  c(5)^2
33  ss1.append @evar var(e6) =  c(6)^2
34  ss1.append @evar cov(e1,e4) = c(4)*c(7)
35  ss1.append @evar cov(e2,e5) = c(5)*c(8)
36  ss1.append @evar cov(e3,e6) = c(6)*c(9)

37  'equations for vehkm
38  ss1.append @signal log(vehkm) = sv1 + e1
39  ss1.append @state sv1 = sv1(-1) + sv2(-1) + e2
40  ss1.append @state sv2 = sv2(-1) + e3

41  'equations for killed
42  ss1.append @signal log(killed) = sv1 + sv3 + e4
43  ss1.append @state sv3 = sv3(-1) + sv4(-1) + e5
44  ss1.append @state sv4 = sv4(-1) + e6

45  ss1.ml(m=1000,c=1e-6)
46  freeze(out1) ss1.output

47  %st1 =@left(out1(6,1),11)

48  if %st1="Convergence" then
49              !row = !row+1
50              results(!row,1) = ss1.@logl
51              results(!row,2) = out1(6,1)
52              results(!row,3) = ss1.@aic
53              results(!row,4) = ss1.@sc
54              results(!row,5) = ss1.@hq

55  ' store initial values of the parameters
56              for !k = 1 to 9 step 1
57                      results(!row,5+!k) = start(!k)
58              next

59  ' store final values of the parameters
60              for !k = 1 to 9 step 1
61              results(!row,5+9+!k)=C(!k)
62              next

63              if ss1.@logl > !loglik then
64                      !loglik = ss1.@logl
65                      delete solution
66                      copy ss1 solution
67                      endif
68  endif

69  delete out1
70  delete ss1

71  next
72  stop
```

Figure 8: Iteration program for the latent risk model.

3.3. Iteration program

The errors and variances and the observation and state equations, as described in the previous section, define the latent risk model. Using the `Estimate` button in the `sspace` object, this model can in principle be estimated. However, because of the more complex multivariate nature of the model, the optimization procedure may find a suboptimal or no solution. To decrease the likelihood of estimation problems, a multiple random start procedure is set up, which runs the optimization algorithm repeatedly, each time starting from a different set of initial values for the model parameters. The program that executes these iterations is shown in Figure 8. It is an illustration of how the programming environment in EViews can be used to support the specification and estimation of state space models. Line numbers are added for ease of reference, they do not belong to the program.

Preparatory declarations

The first part of the program contains declarations of variables and objects that are needed in the remainder of the program. In line 2, a state space object `solution` is created. This object will hold the optimal solution when the iteration program finishes (see below). In line 3, a variable `!loglik` is declared, holding the value -1000000 at the start. This is deliberately chosen to be a large negative value that will be overwritten by the program as soon as a model is obtained with a loglikelihood value greater than -1000000. The loglikelihood will be used as an indicator to compare model solutions obtained during the iteration process. Line 4 creates a table object called `results`. This is a table that will hold all solutions for which the optimization procedure converged. It contains one new row for each converging estimation run.

The cells in the table are referred to by means of row and column numbers. Lines 5–9 create the headings *LogLik*, *Result*, *AIC*, *Schwarz* and *Hannan-Quinn* in the first row. The `LogLik` column will hold the loglikelihood value, and the `Result` column will show the number of iterations that were necessary to obtain the solution (for example: *Convergence achieved after 31 iterations*). The other headings indicate model quality statistics.

In line 10, the variable `row` is set equal to 1. This is a counter that will be used to appropriately fill the `results` table.

Selection of starting values

In line 11, a `for`-loop is used to iterate the subsequent code a number of times. In this program the state space model is solved 1000 times starting from different initial parameter values. This `for`-loop ends in line 71. The different steps will be explained below.

Line 12 restores the full sample to create a new series named `start` in line 13. We will use this series to select starting values for the parameters that determine the variances and covariances in the model. The series `start` is based on a uniform $[0, 1]$ random variable that is generated by the built-in function `rnd`. Note that this code generates more starting values than needed (it is a series), so we will have to select as much values from this series as needed to define the variances and covariances in the model.

In lines 14 to 17, another `for`-loop is executed. These lines are used to assign starting values to the parameters. For the latent risk model, 9 parameters are to be estimated, so the counter variable takes values 1 up to 9. The scalar `!startvar` picks the j-th value from the `start` series, and this value is assigned to the j-th parameter in line 16. The keyword `next` initiates the next run in the `for`-loop that started at line 14.

In line 18, the sample for the analysis is set. This may be different from the sample size in the data set when, for example, forecasting is required. In the program, data from 1965 up to 2008 will be used to estimate the model.

Definition of the state space program

Once the starting values for the parameters are determined, the state space object `ss1` is created in line 19. This object will contain, in every iteration, the estimated solution of the model. Lines 20–44 contain the state space program, using similar keywords as in Figure 1. Note the `ss1.append` command at the beginning of each line, that is used for adding lines to the state space model in a program.

The latent risk model contains two observation equations and four state equations, leading to six error terms ($\varepsilon_t^{(1)}$ and $\varepsilon_t^{(2)}$ for the observation equations, $\xi_t^{(1)}$ and $\xi_t^{(2)}$ for the trends, $\zeta_t^{(1)}$ and $\zeta_t^{(2)}$ for the slopes). These are defined in lines 21–26 of the program. An error term is declared by including a line with the keyword `@ename`, followed by the name of the error. Errors are named `e1`, `e2`, and so on. The correspondence in errors with the latent risk model specification is: `e1` $= \varepsilon_t^{(1)}$, `e2` $= \xi_t^{(1)}$, `e3` $= \zeta_t^{(1)}$, `e4` $= \varepsilon_t^{(2)}$, `e5` $= \xi_t^{(2)}$ and `e6` $= \zeta_t^{(2)}$.

The next step in building the error structure of the model is to define the variances and covariances associated with the errors. This is done in lines 28–33 (for the variances) and 34–36 (for the covariances). Each of these lines starts with the keyword `@evar`, followed by an assignment statement. Three covariances will be estimated, for the observation errors, the trend errors and the slope errors respectively. All other covariances are not specified, and are thus assumed to be zero. If an error term is included without a corresponding `@evar` specification, its variance is assumed to be zero.

Note that the variances and covariances are defined in terms of the coefficients `c(1)`, ..., `c(9)` in such a way that the variance matrices for the measurement, level and slope disturbances are positive semi-definite. For example, using the coefficient notation from Figure 8, the matrix H_t is defined as:

$$H_t = \begin{bmatrix} \texttt{c(1)} & \texttt{c(7)} \\ 0 & \texttt{c(4)} \end{bmatrix} \begin{bmatrix} \texttt{c(1)} & 0 \\ \texttt{c(7)} & \texttt{c(4)} \end{bmatrix} = \begin{bmatrix} \texttt{c(1)}^2 + \texttt{c(7)}^2 & \texttt{c(4)c(7)} \\ \texttt{c(4)c(7)} & \texttt{c(4)}^2 \end{bmatrix}, \tag{7}$$

which is a positive semi-definite matrix for any value of `c(1)`, `c(4)` and `c(7)`. Similar expressions are used for the level and slope disturbances.

Lines 38 and 42 show the two observation equations of the latent risk model. `sv1` represents the unobserved trend component $\mu_t^{(1)}$ for the dependent variable `log(vehkm)`, which is the log of the number of vehicle kilometres. In the second observation equation, where `log(killed)` is the log of the number of persons killed, a new component `sv3` is added to represent the risk trend $\mu_t^{(2)}$.

Lines 39–40 and 43–44 contain the state equations for `log(vehkm)` and `log(killed)` respectively. The unobserved component $\mu_t^{(1)}$ (`sv1`) is modelled in the first state equation. `sv2` represents the slope $\nu_t^{(1)}$ for this component, which is specified in the second state equation. The same goes for the level component $\mu_t^{(2)}$ (`sv3`) and its slope $\nu_t^{(2)}$ (`sv4`). Note that the set of state equations represents equations (6a) through (6d).

When this code is executed, the complete state space model is written to the object `ss1`, which can subsequently be estimated. In line 45, the command is given to estimate the model. Maximum likelihood estimation is done with a maximum of 1000 iterations each time the model is fitted, and the convergence criterion is set to `1E-6`. Line 46 saves the `ss1` model output by "freezing" the view in a `table` object called `out1`.

Treatment of the state space results

In lines 47–68, the state space results are handled. In particular, a distinction is made between converging and non-converging solutions. The program essentially checks the solution for convergence, verifies whether it is better than the previous converging solution and saves it in the `results` table.

In line 47, a string (`%st1`) is taken from the contents in the cell on row 6 and column 1 of the `out1` table, which contains the convergence message. In line 48, it is assessed whether the first 11 characters in this cell form the word *Convergence*, which is the case if a valid solution is obtained.

Once a solution is found, two steps follow: (i) the main results are written on a new line in the table `results`, and (ii) the loglikelihood is checked. In addition, if the loglikelihood value is better than the one of the previous best solution, it has to be saved. In line 49, a new row in the table `results` is selected. In the first column, the log likelihood is written (line 50), and the second column contains the convergence message (line 51). The next columns contain the Akaike, Schwarz and Hannan-Quinn criteria (lines 52–54). The `for`-loop in lines 56–58 stores the initial values for the 9 parameters, while lines 60–62 are for the final values. In line 63, it is checked whether the last loglikelihood value is better than the one previously stored. If the most recent loglikelihood is indeed better, it is written to the variable `!loglik` (line 64), and the `ss1` model is copied into an object called `solution` (line 66). If the latter object already exists, it is first deleted (line 65).

In lines 69–70, the temporary objects `out1` and `ss1` are deleted. They will be re-created in every new run of the program. In line 71, the next iteration in the `for`-loop is initiated, and in line 72 the program terminates. The object `solution` now contains the final (best and converging) model, and the table `results` shows details on all converging solutions. Note, however, that this program will not always work. The outcome is sensitive to good starting values. Sometimes no solution and, in rare occasions, a degenerate solution is found.

Results

As an example, the latent risk model has been fitted to the Belgian fatalities and exposure data shown in Figure 7. In a first run, a possible level shift was noticed in 1978 for the number of vehicle kilometres. Also, we explicitly modelled the "top" year 1970 by including a pulse intervention in the slope equation (6d) of the latent risk, and fixing the error of this equation to zero renders the slope deterministic. In combination with a pulse intervention this results in a piecewise linear slope component.

We adjust the counter in line 14 of the program to select starting values for 11 instead of 9 parameters. We drop the error declaration for `e6` (line 26), its variance (line 33) and the covariance between `e3` and `e6` (line 36). Then we change line 38 to `ss1.append @signal log(vehkm) = sv1 + c(10)*level1978 + e1`, where `level1978` is a level shift variable that equals 0 at all time points before 1978 and equals 1 in 1978 and subsequent years. Line 44 now becomes `ss1.append @state sv4 = sv4(-1) + c(11)*pulse1970`. Note that `level1978` and `pulse1970` need to be defined as new series in the work file.

The final model solution has a loglikelihood value of 137.3071 (AIC = −5.7867). The estimated variance-covariance matrices are as follows:

$$H_t = \begin{bmatrix} 0.000019 & -0.000064 \\ -0.000064 & 0.000352 \end{bmatrix},$$
$$Q_t = \begin{bmatrix} 0.000071 & 0 & 0.000272 \\ 0 & 0.000117 & 0 \\ 0.000272 & 0 & 0.002399 \end{bmatrix}. \tag{8}$$

			log(vehkm)		log(killed)	
	statistic	critical value	value	p value	value	p value
Independence	$Q(15)$	24.9958	17.9518	0.2652	8.0704	0.9209
Homoscedasticity	$1/H(13)$	3.1150	2.7297	0.0816	1.0918	0.8765
Normality	JB	5.9915	0.0308	0.9847	0.4174	0.8116

Table 1: Diagnostic tests for the latent risk model.

Note that the dimension of the matrix Q_t is now 3×3 because of the deterministic slope component for the second observation equation. The smoothed final states are $\mu_T^{(1)} = 4.5298$, $\mu_T^{(2)} = 2.2597$, $\nu_T^{(1)} = 0.0020$ and $\nu_T^{(2)} = -0.0626$. This would indicate that, in 2008, we estimate a yearly growth in exposure of $100(e^{0.0020} - 1) = 0.2\%$ (not significant), while the fatality risk is decreasing at a rate of $100(e^{-0.0626} - 1) = -6.07\%$ per year and the number of fatalities is decreasing at a rate of $100(e^{0.0020-0.0626} - 1) = -5.88\%$ per year. The parameter for the 1978 level intervention in the `log(vehkm)` equation is estimated to be 0.0553 (p value = 0.0000). In the `sv4` slope equation we find a parameter estimate for the 1970 pulse intervention of -0.0868 (p value = 0.0003), implying that we expect a smoothed slope component of $-0.0626+0.0868 = 0.0242$ for all periods before 1970, which also happens to be the initial value of the slope component $\nu_t^{(2)}$.

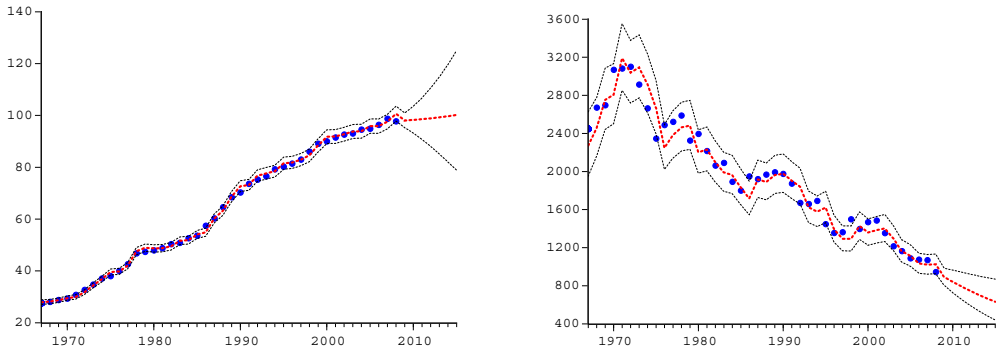
The auxiliary residuals for the two observation equations and three state equations were checked, and although some of the graphs indicated a few possible outliers and structural breaks, none of them were exceptional in comparison with the 95% confidence interval. We further test the model validity by inspecting the standardized prediction errors of both the log of the number of fatalities and the log of the number of vehicle kilometers for independence, homoscedasticity and normality (based on Commandeur and Koopman 2007, p. 90–96). Results are summarized in Table 1. They indicate that all of the model assumptions are satisfied.

Although EViews allows storing all kinds of series generated by the `sspace` object, there is no built-in function for diagnostic testing in state space models. Alternatively, one can save the residuals and perform the standard tests for independence, normality and homoscedasticity after correcting for the number of estimated hyperparameters and/or the number of diffuse initial values, as was done in the analysis of the Nile data in Section 2.3. However, for non-homogeneous multivariate models it is not clear how the degrees of freedom should be corrected (Harvey 1989, p. 443). In practice, the corrections are therefore often omitted.

Figure 9 shows the one-step-ahead predictions for the two series. The point estimates on the original scale were calculated as $\exp\left(\hat{y}_t^{(i)} + \log\left(1 + 0.5\text{var}(\hat{y}_t^{(i)})\right)\right)$. For road safety policy makers, these are interesting outcomes of the latent risk model. The graph in Figure 9b shows that the number of fatalities in 2015 is estimated to be 631, and we are 90% confident that the true value will be between 439 and 869. According to this model, the target of at most 500 fatalities in 2015 (see Section 3.1) is within reach.

4. Conclusion

This paper illustrated how state space models can be fitted in the econometric software package EViews. First we demonstrated how a local level model can be fitted to the Nile data.



(a) Predicted exposure

(b) Predicted number of fatalities

Figure 9: Predictions of fatalities and exposure, with 90% confidence interval.

Next we presented an iteration program that can be used to estimate a latent risk model. This is a multivariate linear state space model in which a second signal equation depends on two unobserved factors: exposure and risk. The latent risk model was applied to Belgian road safety data.

In general EViews is a flexible and user friendly environment when it comes to developing a wide range of econometric models. The same goes for state space models. The interface is accessible and straightforward. Simple models can easily be fitted via the autospecification option or by writing the `sspace` object directly. A nice thing about EViews is that the output can be obtained either by programming or by "point-and-click" action.

Estimation is fast and happens – sometimes regrettably – completely out of sight. Therefore some aspects of the implemented Kalman filtering algorithm and the corresponding optimization of the loglikelihood are not clear to the end user, like for example the diffuse initialization of the unobserved components. This hinders the comparison of state space results in EViews with those obtained in other software packages.

For more complex models, converging solutions are usually not obtained in one estimation run of the `sspace` object. However, the programming environment holds out a hand here. Programming estimation runs increases the odds of finding a suitable solution.

Although statistics for diagnostic checking are readily available in EViews, they are not fully tuned to the output of a state space object. More than a practical knowledge of state space models is then required. Novice state space developers may experience difficulties in disentangling (the quality of) their output. On the other hand, because state space models are treated in almost the same way as any other object, the barrier is presumably quite low for experienced EViews users.

References

Bijleveld FD, Commandeur JJF (2006). "Test Modelling One-Sided Accidents with the Basic Evaluation Model." *Technical Report D-2006-3*, SWOV, Leidschendam.

Bijleveld FD, Commandeur JJF, Gould P, Koopman SJ (2008). "Model-Based Measurement

of Latent Risk in Time Series with Applications." *Journal of the Royal Statistical Society A*, **171**(1), 265–277.

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

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

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

Oppe S (1989). "Macroscopic Models for Traffic and Traffic Safety." *Accident Analysis and Prevention*, **21**, 225–232.

Oppe S (1991). "The Development of Traffic and Traffic Safety in Six Developed Countries." *Accident Analysis and Prevention*, **23**(5), 401–412.

Quantitative Micro Software (2007a). *EViews 6 Command Reference*. Irvine CA, USA. URL http://www.eviews.com.

Quantitative Micro Software (2007b). *EViews 6 User's Guide I*. Irvine CA, USA. URL http://www.eviews.com.

Quantitative Micro Software (2007c). *EViews 6 User's Guide II*. Irvine CA, USA. URL http://www.eviews.com.

Van den Bossche F (2006). *Road Safety, Risk and Exposure in Belgium: An Econometric Approach*. Ph.D. thesis, Hasselt University, Belgium.

Affiliation:

Filip A. M. Van den Bossche
Hogeschool-Universiteit Brussel
Faculty of Economics and Management
Stormstraat 2
BE-1000 Brussels, Belgium
and
Katholieke Universiteit Leuven
Faculty of Business and Economics
Naamsestraat 69
BE-3000 Leuven, Belgium
E-mail: filip.vandenbossche@hubrussel.be

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 41, Issue 8 *Submitted:* 2010-02-07
May 2011 *Accepted:* 2010-12-08