Working Paper 2017:2

Department of Statistics

Estimating a dynamic factor model in EViews using the Kalman Filter and smoother

Martin Solberger & Erik Spånberg

Working Paper 2017:2
April 2017
Department of Statistics
Uppsala University
Box 513
SE-751 20 UPPSALA
SWEDEN

Working papers can be downloaded from www.statistics.uu.se

TITLE: Estimating a dynamic factor model in EViews using the Kalman Filter and smoother

AUTHOR: Martin Solberger & Erik Spånberg
E-MAIL: *martin.solberger@statistik.uu.se*

Estimating a Dynamic Factor Model in EViews Using the Kalman Filter and Smoother

Martin Solberger
Uppsala University
Ministry of Finance, Sweden

Erik Spånberg
Ministry of Finance, Sweden

Abstract

In this paper, we set up a dynamic factor model in EViews using only a small amount of programming. The model is particularly useful for nowcasting the economy, that is, forecasting of the very recent past, the present, or the very near future of economic activity. A subroutine that estimates the model is provided. In a simulation study, the precision of the estimated factors are evaluated, and in an empirical example, the usefulness of the model is illustrated.

Keywords: dynamic factor model, state space, kalman filter, EViews.

1. Introduction

Factor models are used in data-rich environments. The basic idea is to separate a possibly large number of observable variables into two independent and unobservable, yet estimable, components: a 'common component' that captures the main bulk of co-movement between the observable variables, and an 'idiosyncratic component' that captures any remaining individual movement. The common component is assumed to be driven by a few common factors, thereby reducing the dimension of the system. Whereas early contributions assume independent and identically distributed (i.i.d.) random variables, subsequent research, mainly within signal processing and macroeconometrics, have developed the notion of dynamic factor models, modeling dependent time series variables.

In Economics, dynamic factor models are motivated by theory, which predicts that macroeconomic shocks should be pervasive and affect most variables within an economic system. They have therefore become popular among macroeconometricians (see, e.g., Breitung and Eickmeier 2006, for an overview). In particular, it has been demonstrated that dynamic factor models are valuable in business cycle analysis (e.g. Forni and Reichlin 1998; Eickmeier 2007; Ritschl, Sarferaz, and Uebele 2016), forecasting (e.g. Stock and Watson 2002a,b) and nowcasting the state of the economy, that is, forecasting of the very recent past, the present, or the very near future of indicators for economic activity, such as the Gross Domestic Product (GDP) (see, e.g., Banbura, Giannone, Modugno, and Reichlin 2012, and references therein). Information of the economic activity is of great importance for decision makers in, for instance, governments, central banks and financial markets. However, first official estimates of GDP are published with a significant delay, usually about 6-8 weeks after the reference quarter, which makes nowcasting very useful. Other areas of economic analysis using dynamic factor models include, for example, yield curve modeling (e.g. Diebold and Li 2006; Diebold, Rudebusch, and Aruoba 2006), financial risk-return analysis (Ludvigson and Ng 2007), and monetary policy analysis (e.g. Bernanke, Boivin, and Eliasz 2005; Boivin, Giannoni, and Mihov 2009). Despite the attractiveness of dynamic factor models for macroeconomists, econometric or statistical softwares do not in general provide these models within standard packages. In this paper, we illustrate how to, by means of simple programming, set up the popular two-step estimator of Doz, Giannone, and Reichlin (2011) in EViews (IHS Global Inc. 2015a,b,c,d), a statistical software specialized in time series analysis that is broadly used by economists, econometricians, and statisticians.

The parameters of dynamic factor models can be estimated by the method of principal components. This method is easy to compute, and is consistent under quite general assumptions as long as both the cross-section and time dimension grow large. It suffers, however, from a large drawback: the data set must be balanced, where the start and end points of the sample are the same across all observable variables. In practice, data are often released at different dates. A popular approach is therefore to cast the dynamic factor model in a state space representation and then estimate it using the Kalman filter, which allows unbalanced data sets and offers the possibility to smooth missing values. The state space representation contains a signal equation, which links observed variables to latent *states*, and a state equation, which describes how the states evolve over time. The Kalman filter and smoother provide mean-square optimal projections for both the signal and state variables. The method we set up in this paper uses the principal components estimates to re-estimate the factors decomposed as the latent states using the Kalman filter and smoother. We provide a subroutine for this method, which is evaluated in a simulation study and applied in an empirical example. All coding was done in EViews 9.5.

The rest of the paper is organized as follows. Section 2 describes the properties of dynamic factor models and conventional estimation procedures. Section 3 derives a state space solution. Section 4 goes through the specific method of estimation used in this paper. Section 5 evaluates the method of estimation by means of simulation. Section 6 provides an empirical example. Section 7 concludes. Complete codes are placed in appendices: A subroutine with the method of estimation is provided in Appendix A, and the code used for the simulation study is provided in Appendix B.

Some remarks on notation: We let $\mathbf{I}_n$ denote the $n \times n$ identity matrix. For any square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\varphi_1(\mathbf{A}) \geq \varphi_2(\mathbf{A}) \geq \cdots \geq \varphi_n(\mathbf{A})$ denote the ordered eigenvalues, and $\mathbf{A} = \operatorname{diag}(a_1, a_2, \ldots, a_n)$ states that $\mathbf{A}$ is a diagonal matrix with entries $a_1, a_2, \ldots, a_n$ on its main diagonal. For any matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, $||\mathbf{A}|| = [\operatorname{tr}(\mathbf{A}\mathbf{A}')]^{1/2}$ denotes the Frobenius norm, and $\mathbf{A} = [a_{i,j}]_{n\times m}$ states that $\mathbf{A}$ has elements $a_{i,j}$ corresponding to the ith row and jth column ($i = 1, 2, ..., n; j = 1, 2, ..., m$). We let $\mathcal{N}$ denote the normal distribution and $\mathcal{U}$ denote the uniform distribution. For a panel, N and T denote the cross-section and time dimension, respectively, $N \rightarrow$ denotes the limit taken over N, and $N, T \rightarrow$ denotes the joint limit taken over T and N simultaneously. We use $\xrightarrow{p}$ to denote convergence in probability.

2. A dynamic factor model

Let $\mathbf{x}_t = (x_{1,t}, x_{2,t}, ..., x_{N,t})'$ be a vector of N time series, and let the empirical information available at time t ($t = 1, 2, ..., T$) be condensed into the information set $\mathcal{F}_t = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$.

A dynamic factor model is usually specified such that each observable $x_{i,t}$ $(i = 1, 2, ..., N)$ is the sum of two independent and unobservable components: a common component $\chi_{i,t}$, which is driven by a small number of factors that are common to all individuals, and a remaining idiosyncratic (individual-specific) component $\epsilon_{i,t}$ (see, e.g., Bai and Ng 2008). In panel notation, the model is

$$\begin{aligned} x_{i,t} &= \chi_{i,t} + \epsilon_{i,t}, \qquad (i = 1, 2, ..., N; t = 1, 2, ..., T), \\ \chi_{i,t} &= \boldsymbol{\upsilon}_i(L)'\mathbf{z}_t, \end{aligned} \tag{1}$$

where $\boldsymbol{\upsilon}_i(L) = \boldsymbol{\upsilon}_{i,0} + \boldsymbol{\upsilon}_{i,1}L + \cdots + \boldsymbol{\upsilon}_{i,\kappa}L^\ell$ is a vector lag-polynomial, possibly of infinite order, loading onto a vector of $\mathcal{K}$ unobservable common factors, $\mathbf{z}_t = (z_{1,t}, z_{2,t}, ..., z_{\mathcal{K},t})'$.[1] Thus, only the left-hand side of Equation (1) is observed; the right-hand side is unobserved. If the dimension of $\mathbf{z}_t$ is finite $(\mathcal{K} < \infty)$, then there exists for every i an $\mathcal{R} \times 1$ vector $(\mathcal{R} \geq \mathcal{K})$ of constants $\boldsymbol{\lambda}_i = (\lambda_{i,1}, \lambda_{i,2}, ..., \lambda_{i,\mathcal{R}})'$, such that $\boldsymbol{\upsilon}_i(L)' = \boldsymbol{\lambda}_i'\mathbf{C}(L)$, where $\mathbf{C}(L)$ is an $\mathcal{R} \times \mathcal{K}$ matrix lag-polynomial, of infinite order in general, $\mathbf{C}(L) = \sum_{m=0}^{\infty} \mathbf{C}_m L^m$, that is absolutely summable, $\sum_{m=0}^{\infty} ||\mathbf{C}_m|| < \infty$ (see Forni, Giannone, Lippi, and Reichlin 2009). Thus, letting $\mathbf{f}_t = (f_{1,t}, f_{2,t}, ..., f_{\mathcal{R},t})' = \mathbf{C}(L)\mathbf{z}_t$, the dynamic factor model can be cast in the static representation

$$\begin{aligned} x_{i,t} &= c_{i,t} + \epsilon_{i,t}, \\ c_{i,t} &= \boldsymbol{\lambda}_i'\mathbf{f}_t, \end{aligned} \tag{2}$$

which, equivalently, can be written in vector notation as

$$\begin{aligned} \mathbf{x}_t &= \mathbf{c}_t + \boldsymbol{\epsilon}_t, \\ \mathbf{c}_t &= \boldsymbol{\Lambda}\mathbf{f}_t, \end{aligned} \tag{3}$$

where $\mathbf{c}_t = (c_{1,t}, c_{2,t}, ..., c_{N,t})'$, $\boldsymbol{\epsilon}_t = (\epsilon_{1,t}, \epsilon_{2,t}, ..., \epsilon_{N,t})'$ and $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1', \boldsymbol{\lambda}_2', ..., \boldsymbol{\lambda}_N')'$. The common factors in $\mathbf{z}_t$ are often referred to as *dynamic factors*, while the common factors in $\mathbf{f}_t$ are referred to as *static factors*. The number of static factors, $\mathcal{R}$, cannot be smaller than the number of dynamic factors, and is typically much smaller than the number of cross-sectional individuals, $\mathcal{K} \leq \mathcal{R} \ll N$. As with $\chi_{i,t}$ in the dynamic representation (1), we refer to the scalar process c_i in (2), or the multivariate process $\mathbf{c}_t$ in (3), as the *common component*.

In general, we suppose that every $x_{i,t}$ is a weakly stationary process with mean zero that has, at least, finite second-order moments, $\mathsf{E}(x_{i,t}) = 0$, $\mathsf{E}(x_{i,t}x_{i,t-s}) < \infty$ $(s = 0, 1, 2, ...)$. To uphold this, something close to the following is usually assumed:

A1 (common factors) The q-variate process $\mathbf{z}_t$ is orthonormal white noise, that is, independent and identically distributed (i.i.d.) over both the cross-section and time dimension with zero mean: $\mathsf{E}(\mathbf{z}_t) = \mathbf{0}$, $\mathsf{E}(\mathbf{z}_t\mathbf{z}_t') = \text{diag}(\omega_1^2, \omega_2^2, ..., \omega_{\mathcal{R}}^2)$, where $\omega_j^2 < \infty$ for all j, and $\mathsf{E}(\mathbf{z}_t\mathbf{z}_s') = \mathbf{0}$ for all $t \neq s$.

A2 (idiosyncratic components) The univariate process $\epsilon_{i,t}$ admits the Wold representation $\epsilon_{i,t} = \theta_i(L)u_{i,t} = \sum_{m=0}^{\infty} \theta_{i,m}u_{i,t-m}$, where $\sum_{m=0}^{\infty} |\theta_{i,m}| < \infty$ and $u_{i,t}$ is i.i.d. white noise with limited cross-sectional dependence: $\mathsf{E}(u_{i,t}) = 0$, $\mathsf{E}(u_{i,t}u_{j,s}) = 0$ for all $t \neq s$, and $\mathsf{E}(u_{i,t}u_{j,t}) = \tau_{i,j}$, with $\sum_{i=0}^{N} |\tau_{i,j}| < \mathcal{J}$, where $\mathcal{J}$ is some positive number that does not depend on N or T.

[1]Here, the lag transpose is defined as $\boldsymbol{\upsilon}_i(L)' = \boldsymbol{\upsilon}_{i,1}'L + \boldsymbol{\upsilon}_{i,2}'L^2 + \cdots + \boldsymbol{\upsilon}_{i,\ell}'L^\ell$, where, for $q = 1, 2, ..., \ell$, $\boldsymbol{\upsilon}_{i,q}$ are $\mathcal{K} \times 1$ vectors, such that $\chi_{i,t} = \boldsymbol{\upsilon}_{i,0}'\mathbf{z}_t + \boldsymbol{\upsilon}_{i,1}'\mathbf{z}_{t-1} + \cdots + \boldsymbol{\upsilon}_{i,\kappa}'\mathbf{z}_{t-\ell}$.

A3 (independence) The common shocks $\mathbf{z}_t$ and idiosyncratic errors $\mathbf{u}_t = (u_{i,t}, u_{2,t}, ..., u_{N,t})'$, are mutually independent groups, $\mathsf{E}(\mathbf{u}_t\mathbf{z}_s') = \mathbf{0}$ for all t, s.

From Assumption A1, the static factors are stationary and variance-ergodic processes admitting a Wold representation $\mathbf{f}_t = \mathbf{C}(L)\mathbf{z}_t = \sum_{m=0}^{\infty} \mathbf{C}_m\mathbf{z}_{t-m}$. Assuming invertibility, the static factors may therefore follow some stationary VAR(p) process,

$$\mathbf{A}(L)\mathbf{f}_t = \mathbf{z}_t, \tag{4}$$

where $\mathbf{A}(L) = \mathbf{I}_{\mathcal{R}} - \mathbf{A}_1L - \mathbf{A}_2L^2 - \cdots - \mathbf{A}_pL^p = \mathbf{C}(L)^{-1}$.

We have assumed here, for the sake of argument, that the dynamic factors $\mathbf{z}_t$ (sometimes referred to as the *primitive shocks*) enters as errors in the static factor VAR process (4). This is unnecessarily strict. To be more precise, by suitably defining $\mathbf{f}_t$, the dynamic factor model (1) can in general be cast in the static representation (2), where $\mathbf{f}_t$ follows a VAR process which exact order depends on the specific dynamics of $\mathbf{z}_t$. The number of static factors is always $\mathcal{R} = \mathcal{K}(\ell+1)$; see Bai and Ng (2007). In practice, the static representation is typically stated without reference to a more general dynamic factor model. Additionally, it is often assumed that the static common factors follow a stationary VAR process. Assumption A1 is innocuous, as we may assume that $\mathbf{z}_t$ is an orthonormal error of the static factor VAR process, and that this, in general, relates to some dynamic factor model (1).

Now, from Assumptions A1-A3, the autocovariance of the panel data is

$$\mathbf{\Gamma}_x(h) = \mathsf{E}(\mathbf{x}_t\mathbf{x}_{t-h}') = \mathbf{\Lambda}\mathbf{\Gamma}_f(h)\mathbf{\Lambda}' + \mathbf{\Gamma}_\epsilon(h), \tag{5}$$

where $\mathbf{\Gamma}_f(h) = \mathsf{E}(\mathbf{f}_t\mathbf{f}_{t-h}')$ and $\mathbf{\Gamma}_\epsilon(h) = \mathsf{E}(\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_{t-h}')$. By denoting $\mathbf{\Sigma} = \mathbf{\Gamma}_x(0)$, $\mathbf{\Upsilon} = \mathbf{\Gamma}_f(0)$ and $\mathbf{\Psi} = \mathbf{\Gamma}_\epsilon(0)$, we can write the contemporary covariance matrix of $\mathbf{x}_t$ as $\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Upsilon}\mathbf{\Lambda}' + \mathbf{\Psi}$. This covariance matrix is of central importance for dynamic factor models. If the largest eigenvalue of $\mathbf{\Psi}$ is bounded as $N \to \infty$, then we have an *approximate* factor model as defined by Chamberlain and Rothschild (1983). Approximate factor models have become very popular within, for instance, panel data econometrics, because they allow for a cross-sectional dependence among $\boldsymbol{\epsilon}_t$, whilst letting the factor structure be identified. If $\mathbf{\Psi}$ is diagonal, then we have the *exact* (or *strict*) factor model. Since the diagonal elements of $\mathbf{\Psi}$ are real and finite, the exact factor model is nested within the approximate factor model as a special case. The largest eigenvalue of $\mathbf{\Psi}$ is, in fact, smaller than $\max_j \sum_{i=1}^{N} |\mathsf{E}(\epsilon_{i,t}\epsilon_{j,t})|$ (see Bai 2003), which is bounded with respect to both N and T by Assumption A2.[2] Hence, by assumption, we have an approximate factor model.

On top of this, some technical requirements are usually placed on the factor loadings, such that the common component is pervasive as the number of cross-sectional individuals increase. In essence, as $N \to \infty$, all eigenvalues of $N^{-1}\mathbf{\Lambda}'\mathbf{\Lambda}$ should be positive and finite, implying that the $\mathcal{R}$ largest eigenvalues of $\mathbf{\Sigma}$ are unbounded asymptotically.[3]

[2]From Assumption A2, $\mathsf{E}(\epsilon_{i,t}\epsilon_{j,t}) = \sum_{m=0}^{\infty} \theta_{i,m}\mathsf{E}(u_{i,t-m}u_{j,t-m}) = \sum_{m=0}^{\infty} \theta_{i,m}\tau_{i,j} = \tau_{i,j}\sum_{m=0}^{\infty} \theta_{i,m}$. Thus, $|\mathsf{E}(\epsilon_{i,t}\epsilon_{j,t})| = |\tau_{i,j}||\sum_{m=0}^{\infty} \theta_{i,m}|$, and so $\max_j \sum_{i=1}^{N} |\mathsf{E}(\epsilon_{i,t}\epsilon_{j,t})| = |\sum_{m=0}^{\infty} \theta_{i,m}| \max_j \sum_{i=1}^{N} |\tau_{i,j}| < \infty$.

[3]By construction, the $\mathcal{R}\times\mathcal{R}$ matrix $\mathbf{\Lambda}'\mathbf{\Lambda}$ is positive definite, whereas the $N \times N$ matrix $\mathbf{\Lambda}\mathbf{\Lambda}'$ is positive semi-definite. It is easily established (see, e.g., Zhou and Solberger 2017, Lemma A1) that the (positive) eigenvalues of $\mathbf{\Lambda}'\mathbf{\Lambda}$ correspond exactly to the non-zero eigenvalues of $\mathbf{\Lambda}\mathbf{\Lambda}'$. That is, for $i = 1, 2, ..., \mathcal{R}$, $\varphi_i(\mathbf{\Lambda}\mathbf{\Lambda}') = \varphi_i(\mathbf{\Lambda}'\mathbf{\Lambda})$, while for $i = \mathcal{R}+1, \mathcal{R}+2, ..., N$, $\varphi_i(\mathbf{\Lambda}\mathbf{\Lambda}') = 0$. Because the eigenvalues of $N^{-1}\mathbf{\Lambda}'\mathbf{\Lambda}$ are non-zero, the eigenvalues of $\mathbf{\Lambda}\mathbf{\Lambda}'$ are unbounded as $N \to \infty$.

Note finally that, unless we impose restrictions, the factors and factor loadings are only identified up to pre-multiplication with an arbitrary $\mathcal{R} \times \mathcal{R}$ full rank matrix $\mathbf{M}$. That is, Equation (3) is observationally equivalent to $\mathbf{x}_t = \mathbf{\Lambda}^0\mathbf{f}_t^0 + \boldsymbol{\epsilon}_t$ where $\mathbf{\Lambda}^0 = \mathbf{\Lambda}\mathbf{M}$ and $\mathbf{f}_t^0 = \mathbf{M}^{-1}\mathbf{f}_t$. Likewise, in terms of variances, we have that $\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Upsilon}\mathbf{\Lambda}' + \mathbf{\Psi} = \mathbf{\Lambda}\mathbf{M}\mathbf{M}^{-1}\mathsf{E}(\mathbf{f}_t\mathbf{f}_t)'\mathbf{M}'^{-1}\mathbf{M}'\mathbf{\Lambda}' = \mathbf{\Lambda}\mathbf{M}\mathsf{E}(\mathbf{M}^{-1}\mathbf{f}_t\mathbf{f}_t'\mathbf{M}'^{-1})\mathbf{M}'\mathbf{\Lambda}' + \mathbf{\Psi} = \mathbf{\Lambda}^0\mathsf{VAR}(\mathbf{f}_t^0)\mathbf{\Lambda}^{0'} + \mathbf{\Psi}$. Due to this rotational indeterminacy, we may without loss of generality impose the normalization $\mathbf{\Upsilon} = \mathbf{I}_{\mathcal{R}}$, restricting $\mathbf{M}$ to be an orthogonal matrix. This implies that only the spaces spanned by, respectively, the columns of $\mathbf{\Lambda}$ and those of $\mathbf{f}_t$, are identifiable from the contemporary covariance $\mathbf{\Sigma}$. In general, this identification requires a large N.

In many cases, estimating the space spanned by the factors is as good as estimating the factor itself. For instance, in forecasting under squared error loss, the object of interest is the conditional mean, which is unaffected by rotational multiplication. If, however, the factors themselves or the coefficients associated with the factors are the parameters of interest, then we need to impose some identifying, yet arbitrary, restrictions such that the factors and factor loadings are exactly identified; see Bai and Ng (2013) and Bai and Wang (2014).

2.1. Estimation

Estimation of dynamic factor models concern foremost the common component. The idiosyncratic component is generally considered residual. The common component of the dynamic factor model (1) may be consistently estimated in the frequency domain by spectral analysis; see Forni, Hallin, Lippi, and Reichlin (2000, 2004). The main benefit of the static factor model (3), is that for approximate factor models, the common component may be consistently estimated in the time domain, which computational methods are generally much easier to accomplish. Because the factor model is a panel (i.e. $x_{i,t}$ is doubly indexed over dimensions N and T), asymptotic theory has been developed as both dimensions N and T tend to infinity, which requires some special attention. Conceptually, estimation theory for panels can be derived in three ways: sequentially, diagonally, and jointly (see, e.g., Phillips and Moon 1999; Bai 2003). The methods presented in this paper all concern the latter limit. This limit is the most general, and its existence implies the existence of the other two.

From Assumption A2, each idiosyncratic component is a stationary and variance-ergodic process that, assuming invertibility, may be stated as a finite-order autoregressive process, $\phi_i(L)\epsilon_{i,t} = u_{i,t}$, where $\phi_i(L) = \theta_i(L)^{-1}$. Let $\mathbf{\Phi}(L) = \text{diag}(\phi_1(L), \phi_2(L), ..., \phi_N(L))$, so that the idiosyncratic process in (3) may be written as $\mathbf{\Phi}(L)\boldsymbol{\epsilon}_t = \mathbf{u}_t$, where $\mathbf{u}_t = (u_{1,t}, u_{2,t}, , , .u_{N,t})$. In terms of parameters, the static factor model (3) may then be characterized by

$$\{\mathbf{\Lambda}, \mathbf{A}(L), \mathbf{\Phi}(L), \mathbf{\Gamma}_f(h), \mathbf{\Gamma}_\epsilon(h); h \in (-\infty, \infty)\}. \tag{6}$$

The autocovariances $\mathbf{\Gamma}_f(h)$ and $\mathbf{\Gamma}_\epsilon(h)$ for $h \neq 0$ are rarely of direct interest, and are not necessary for consistent estimation of the common component. Furthermore, because $\mathbf{\Lambda}$ is, up to a rotation, asymptotically identifiable from $\mathbf{\Sigma}$, for most cases we are merely interested in $\{\mathbf{\Lambda}, \mathbf{\Upsilon}, \mathbf{\Psi}\}$. Given these parameters, the minimum mean square error predictor of the static factors is the projection (see, e.g., Anderson 2003, Section 14.7)

$$\hat{\mathbf{f}}_t = \left(\mathbf{\Upsilon}^{-1} + \mathbf{\Lambda}'\mathbf{\Psi}^{-1}\mathbf{\Lambda}\right)^{-1}\mathbf{\Lambda}'\mathbf{\Psi}^{-1}\mathbf{x}_t, \tag{7}$$

where, under conventional normalization, $\mathbf{\Upsilon} = \mathbf{I}_{\mathcal{R}}$ (see aforementioned).

Naturally, the parameters $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ are unknown, and need to be estimated. On theoretical grounds, maximum likelihood (ML) estimation is attractive. It is generally efficient and provides means for incorporating restrictions based on theory. However, ML estimators for dynamic factor models tend to be very complicated to derive, and full ML estimation is only available for special cases. For instance, suppose that the factors and idiosyncratic components are white noise, $\mathbf{A}(L) = \mathbf{I}_{\mathcal{R}}$, $\mathbf{\Phi}(L) = \mathbf{I}_N$, and that the idiosyncratic contemporary variance is $\mathbf{\Psi} = \psi \mathbf{I}_N$ (the static factor model with spherical noise). In that case, there are explicit ML estimators of $\mathbf{\Lambda}$ and ψ (see Stoica and Jansson 2009; Doz, Giannone, and Reichlin 2012). When the idiosyncratic component exhibits either time series dynamics, cross-sectional heteroscedasticity, or cross-sectional correlations, then full ML estimation is not attainable. However, by imposing misspecifying restrictions, subsets of the parameters may be consistently estimated by quasi-ML in the sense of White (1982). For example, by falsely assuming an exact factor model when the true model is an approximate factor model, the diagonal elements of $\mathbf{\Psi}$ (i.e. the contemporary idiosyncratic variances) and the space spanned by the columns of $\mathbf{\Lambda}$ may be consistently estimated by numerical quasi-ML estimation based on the iterative Expectation-Maximization (EM) algorithm; see Bai and Li (2012, 2016). In a similar fashion, Doz *et al.* (2012) have shown that the space spanned by the factors may be directly and consistently estimated by quasi-ML using the Kalman filter. If the procedure is iterated, then it is equivalent to the EM algorithm.

The workhorse for the static factor model is the method of principal components (PC). Consider the covariance matrix of $\mathbf{x}_t$, $\mathbf{\Sigma}$. Because every covariance matrix is positive semi-definite, it may be decomposed as $\mathbf{\Sigma} = \mathbf{V\Pi V}'$, where $\mathbf{\Pi} = \text{diag}(\varphi_1(\mathbf{\Sigma}), \varphi_2(\mathbf{\Sigma}), ..., \varphi_N(\mathbf{\Sigma}))$ is a diagonal matrix with the ordered positive eigenvalues of $\mathbf{\Sigma}$ (the principle components) on its main diagonal, and $\mathbf{V}$ is a matrix with the associated eigenvectors as columns, such that $\mathbf{V}'\mathbf{V} = \mathbf{I}_N$. Under the normalization $\mathbf{\Upsilon} = \mathbf{I}_{\mathcal{R}}$, the linear transformation $\mathbf{m}_t = \mathbf{V}'\mathbf{x}_t$ is the population PC estimator of the factors $\mathbf{f}_t$. It has contemporary covariance $\mathsf{VAR}(\mathbf{m}_t) = \mathbf{V}'\mathbf{\Sigma V} = \mathbf{\Pi}$. Because $\mathbf{\Pi}$ is a diagonal matrix, the population factors are uncorrelated. Now, let $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N)$. The first PC factor, $\hat{f}_{1,t} = \mathbf{v}_1'\mathbf{x}_t$, is the projection which maximizes the variance among all linear projections from unit vectors. Its variance is the first principal component $\varphi_1(\mathbf{\Sigma})$. The second PC factor, $\hat{f}_{2,t} = \mathbf{v}_2'\mathbf{x}_t$, maximizes the variance under the restriction of being orthogonal to the first PC factor. Its variance is the second principle component $\varphi_2(\mathbf{\Sigma})$, and so on. The PC estimator of the factor loadings is found be setting $\hat{\mathbf{\Lambda}}$ equal to the eigenvectors of $\mathbf{\Sigma}$ associated with its $\mathcal{R}$ largest eigenvalues. Replacing $\mathbf{\Sigma}$ with its sample counterpart $\mathbf{S} = T^{-1}\sum_{t=1}^{T}\mathbf{x}_t\mathbf{x}_t'$, gives the sample PC estimators. Under an approximate factor model, they are consistent as $N, T \to \infty$ for the spaces spanned by the factors and factor loadings, respectively; see, foremost, Bai (2003) and Stock and Watson (2002a).

The method of PC is a dimension reducing technique, and does, as opposed to ML, not require the existence of the static factor model (3). Yet, the PC and ML estimators are closely related. Under a Gaussian static factor model with spherical noise, the ML estimator of the factors is proportional to the PC estimator. The PC estimators of the factors and factor loadings are therefore often used as initial estimators for ML algorithms. For approximate factor models, the largest drawbacks of the PC estimators are that (i) they are inconsistent for fixed N, and (ii) they require a balanced panel. Meanwhile, ML estimation can be consistent for fixed N, and numerical procedures, such as the Kalman filter and the EM algorithm can smooth over missing values, allowing an unbalanced panel with missing values at the end or start of the panel; the so called "ragged edge" or "jagged edge" problem. This feature is very valuable for economic forecasting, because key economic indicators tend to be released at different dates. In particular, and of special interest in this paper, Doz *et al.* (2011) show that, by consistently estimating $\mathbf{A}(L)$, the precision in estimating the factor space may be improved by setting up a state space solution and perform one run with the Kalman smoother to re-estimate the factors $\mathbf{f}_t$ for $t = 1, 2, ..., T$. This method is presented in detail in Section 4, and can be implemented in EViews by using the code we provide in Appendix A.

For any estimation approach, the number of factors $\mathcal{R}$ is generally unknown, and needs to be either estimated or assumed. Popular estimators for the number of factors in approximate factor models can be found in Bai and Ng (2002), Onatski (2010) and Ahn and Horenstein (2013). Throughout the paper, we will treat the number of factors as known.

3. A state space representation of the static factor model

Let, as before, $\mathbf{x}_t$ be an N-dimensional observable time series. Linear time series processes can be cast in the state space form

$$\mathbf{x}_t = \mathbf{H}_t\boldsymbol{\alpha}_t + \boldsymbol{\xi}_t, \tag{8}$$

$$\boldsymbol{\alpha}_{t+1} = \mathbf{T}_t\boldsymbol{\alpha}_t + \mathbf{R}_t\boldsymbol{\eta}_t, \tag{9}$$

where $\boldsymbol{\alpha}_t$ $(k \times 1)$ is a latent state vector, $\mathbf{H}_t$ $(N \times k)$ and $\mathbf{T}_t$ $(k \times k)$ are possibly time-varying parameter matrices and $\mathbf{R}_t$ $(k \times q;\ q \leq k)$ is, in general, either the identity matrix or a selection matrix consisting of a subset of the columns of the identity matrix (see, e.g., Durbin and Koopman 2012). The system is stochastic through the $N \times 1$ vector $\boldsymbol{\xi}_t$ and the $k \times 1$ vector $\boldsymbol{\eta}_t$, which are mutually and serially uncorrelated with zero mean and contemporary covariance matrices $\boldsymbol{\Sigma}_\xi$ and $\boldsymbol{\Sigma}_\eta$, respectively. In the Gaussian state space form, which is what EViews handles, the errors are normally distributed: $\boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\xi)$, $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\eta)$. We refer to Equation (8) as the *signal equation*, and to Equation (9) as the *state equation*. Note that the state equation is a VAR(1) process.

The static factor model (3) can be written as a state space solution defined by Equations (8) and (9), were the number of states relates to the latent components of the model, that is, the common factors and the idiosyncratic components. Moreover, as shown by Doz *et al.* (2011), neglecting the idiosyncratic time series dynamics, and thereby possibly misspecifying the underlying model, can still lead to consistent estimation of the central parameters of the factor model, given by the common component. Specifically, imposing the misspecification that $\boldsymbol{\epsilon}_t$ in Equation (3) is white noise, the static factor model can be written in a state space form where the number of states k is equal to the number of factors $\mathcal{R}$ times the number of VAR lags p, $k = \mathcal{R}p$. To see this, note that the factor VAR(p) process (4) can be written in stacked form as the VAR(1) process (see, e.g., Lütkepohl 2007, p. 15)

$$\tilde{\mathbf{f}}_t = \tilde{\mathbf{A}}\tilde{\mathbf{f}}_{t-1} + \tilde{\mathbf{z}}_t,$$

where (with dimensions underneath)

$$\tilde{\mathbf{f}}_t = \underset{\mathcal{R}p\times 1}{\begin{pmatrix} \mathbf{f}_t \\ \mathbf{f}_{t-1} \\ \vdots \\ \mathbf{f}_{t-p+1} \end{pmatrix}}, \quad \tilde{\mathbf{A}} = \underset{\mathcal{R}p\times\mathcal{R}p}{\begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_{p-1} & \mathbf{A}_p \\ \mathbf{I}_\mathcal{R} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_\mathcal{R} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_\mathcal{R} & \mathbf{0} \end{pmatrix}}, \quad \tilde{\mathbf{z}}_t = \underset{\mathcal{R}p\times 1}{\begin{pmatrix} \mathbf{z}_t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}}. \tag{10}$$

Thus, if the static factors follow the VAR process (4) and the idiosyncratic components are serially uncorrelated, then the static factor model (3) has a state space representation defined by Equations (8) and (9) with

$$\mathbf{H}_t = \begin{pmatrix}\mathbf{\Lambda} & \underset{N\times\mathcal{R}p}{\mathbf{0}} & \cdots & \mathbf{0}\end{pmatrix}, \quad \boldsymbol{\alpha}_t = \tilde{\mathbf{f}}_t, \quad \boldsymbol{\xi}_t = \boldsymbol{\epsilon}_t, \quad \mathbf{R}_t' = \begin{pmatrix}\mathbf{I}_{\mathcal{R}} & \underset{\mathcal{R}\times\mathcal{R}p}{\mathbf{0}} & \cdots & \mathbf{0}\end{pmatrix}, \quad \mathbf{T}_t = \tilde{\mathbf{A}}, \quad \boldsymbol{\eta}_t = \mathbf{z}_t, \tag{11}$$

where $\mathbf{\Lambda}$ and $\boldsymbol{\epsilon}_t$ are the factor loadings and idiosyncratic error in Equation (3), $\tilde{\mathbf{f}}_t$ and $\tilde{\mathbf{A}}$ are the stacked factors and parameters in Equation (10) and $\mathbf{z}_t$ is the error in Equation (4). Here, the subscripts may be dropped from $\mathbf{H}_t$, $\mathbf{T}_t$ and $\mathbf{R}_t$, since in this case, the parameters are not time-varying.

Specification and estimation of state space models in EViews is outlined in IHS Global Inc. (2015c, Chapter 39). A recent demonstration is found in van den Bossche (2011). Estimation concerns two aspects: (i) measuring the unknown states $\boldsymbol{\alpha}_t$ for $t = 1, 2, ..., T$, involving prediction, filtering and smoothing (see Section 3.1), and (ii) estimation of the unknown parameters $\mathbf{H}_t$, $\mathbf{T}_t$, $\mathbf{\Sigma}_\xi$ and $\mathbf{\Sigma}_\eta$. Doz *et al.* (2011) propose to estimate the parameters by PC, and leave only the estimation of the factors (i.e. the states) to the state space form (see Section 4).

Remark 3.1. Equation (9) is specified for the states in period $t+1$, given the errors in period t, which requires some consideration when modeling correlations between the signal and state errors in EViews. By Assumption A3, the factors and idiosyncratic components are mutually independent at all leads and lags, and so, the state and signal errors are modeled as mutually uncorrelated. Therefore, the construction of the temporal indices of the state and signal equations do not affect the methods we use in this paper. For a state space specification with contemporary error indices, see e.g. Hamilton (1994, p. 372).

3.1. The Kalman filter and smoother

The latent state vector $\boldsymbol{\alpha}_t$ can be estimated numerically by the Kalman filter and smoother (see, e.g., Harvey 1989; Durbin and Koopman 2012, for thorough treatments). Consider the conditional distribution of $\boldsymbol{\alpha}_t$, based on the available information at time t, $\mathcal{F}_t$. Under the Gaussian state space model, the distribution is normal $\boldsymbol{\alpha}_t|\mathcal{F}_t \sim \mathcal{N}(\mathbf{a}_{t|t-1}, \mathbf{P}_{t|t-1})$, where

$$\mathbf{a}_{t|t-1} = \mathsf{E}(\boldsymbol{\alpha}_t|\mathcal{F}_{t-1}), \qquad \mathbf{P}_{t|t-1} = \mathsf{VAR}(\boldsymbol{\alpha}_t|\mathcal{F}_{t-1}).$$

By construction, $\mathbf{a}_{t|t-1}$ is the minimum MSE estimator of (the Gaussian) $\boldsymbol{\alpha}_t$, with MSE matrix $\mathbf{P}_{t|t-1}$. Given the conditional mean, we can find the minimum MSE estimator of $\mathbf{x}_t$ from Equation (8),

$$\hat{\mathbf{x}}_t = \mathsf{E}(\mathbf{x}_t|\mathbf{a}_{t|t-1}) = \mathbf{H}_t\mathbf{a}_{t|t-1},$$

with error $\mathbf{v}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ and associated $N \times N$ error covariance matrix

$$\mathbf{F}_t = \mathsf{VAR}(\mathbf{v}_t) = \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t' + \mathbf{\Sigma}_\xi,$$

where $\mathbf{\Sigma}_\xi$ was defined in relation to Equation (8).

The Kalman filter is a recursion over $t = 1, 2, ..., T$ that, based on the error $\mathbf{v}_t$ and the dispersion matrix $\mathbf{F}_t$, sequentially updates the means and variances in the conditional distributions $\boldsymbol{\alpha}_t|\mathcal{F}_t \sim \mathcal{N}(\mathbf{a}_{t|t}, \mathbf{P}_{t|t})$ and $\boldsymbol{\alpha}_{t+1}|\mathcal{F}_t \sim \mathcal{N}(\mathbf{a}_{t+1|t}, \mathbf{P}_{t+1|t})$ by

$$\begin{aligned}\mathbf{a}_{t|t} &= \mathbf{a}_{t|t-1} + \mathbf{P}_{t|t-1}\mathbf{H}_t'\mathbf{F}_t^{-1}\mathbf{v}_t, \\ \mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\mathbf{H}_t'\mathbf{F}_t^{-1}\mathbf{H}_t\mathbf{P}_t, \\ \mathbf{a}_{t+1|t} &= \mathbf{T}_t\mathbf{a}_{t|t}, \\ \mathbf{P}_{t+1|t} &= \mathbf{T}_t\mathbf{P}_{t|t}\mathbf{T}_t' + \mathbf{R}_t\boldsymbol{\Sigma}_\eta\mathbf{R}_t',\end{aligned} \tag{12}$$

where $\mathbf{T}_t$ and $\boldsymbol{\Sigma}_\eta$ were defined in relation to Equation (9). The computational complexity of the recursion depends largely on the inversion of $\mathbf{F}_t$. Note that the second term in Equation (12) may be viewed as a correction term, based on the last observed error $\mathbf{v}_t$. EViews refers to $\mathbf{a}_{t|t}$ as the *filtered estimate* of $\boldsymbol{\alpha}_t$, and to $\mathbf{a}_{t+1|t}$ as the *one-step ahead prediction* of $\boldsymbol{\alpha}_t$.

The recursion requires the initial one-step ahead predicted values $\mathbf{a}_{1|0}$ and its associated covariance matrix $\mathbf{P}_{1|0}$. If $\mathbf{x}_t$ is a stationary process, then they can be found analytically. Otherwise, EViews uses *diffuse priors*, following Koopman, Shephard, and Doornik (1999). The user may also provide the initial conditions; see van den Bossche (2011) and IHS Global Inc. (2015d, p. 683). Additionally, we need estimates of the parameters $\mathbf{H}_t$, $\mathbf{T}_t$, $\boldsymbol{\Sigma}_\xi$ and $\boldsymbol{\Sigma}\eta$. Conveniently, the Kalman filter provides the likelihood function as a by-product from the one-step ahead prediction errors (see Harvey 1989, Section 3.4.), and the recursion can therefore be based on maximum likelihood estimators. EViews offers various optimization routines to find the associated maximum likelihood estimates. In this paper, however, we follow Doz *et al.* (2011) and estimate the parameters by PC. The factors alone are then estimated using the Kalman filter. This procedure is consistent and generally much faster than a recursion that includes parameter estimation.

The Kalman filter is a forward recursion. By applying a backward (smoothing) recursion using the output from the Kalman filter, we can find $\mathbf{a}_{t|T}$, the $\mathcal{F}_T$-conditional (i.e. conditional on *all* available observations) minimum MSE estimator of $\boldsymbol{\alpha}_t$, and its associated MSE matrix $\mathbf{P}_{t|T}$. There are different kinds of smoothing algorithms (see, e.g. Durbin and Koopman 2012, Section 4.4). EViews uses a fixed-interval smoothing, which in its classical form returns the estimated means and variances of the conditional distributions $\boldsymbol{\alpha}_t|\mathcal{F}_t \sim \mathcal{N}(\mathbf{a}_{t|T}, \mathbf{P}_{t|T})$ by

$$\begin{aligned}\mathbf{a}_{t|T} &= \mathbf{a}_{t|t} + \mathbf{P}_{t|t}\mathbf{T}_t'\mathbf{P}_{t+1|t}^{-1}\left(\mathbf{a}_{t+1|T} - \mathbf{a}_{t+1|t}\right), \\ \mathbf{P}_{t|T} &= \mathbf{P}_{t|t} - \mathbf{P}_{t|t}\mathbf{T}_t'\mathbf{P}_{t+1|t}^{-1}\left(\mathbf{P}_{t+1|T} - \mathbf{P}_{t+1|t}\right)\mathbf{P}_{t+1|t}^{-1}\mathbf{T}_t\mathbf{P}_{t|t},\end{aligned}$$

for $t = T, T-1, ..., 1$.

Because the smoothed estimator $\mathbf{a}_{t|T}$ is based on all observations, its MSE cannot be larger than the MSE from the filtered estimator $\mathbf{a}_{t|t}$, in the sense that the MSE matrix of the latter is the MSE matrix of the smoothed estimator plus some positive definite matrix. If the state space model is not Gaussian, then the Kalman filter and smoother do not in general provide the conditional means, and the associated estimators are no longer the minimum MSE estimators. They are, however, the *linear* minimum MSE estimators.

The Kalman filter and smoother offer an exceptionally easy way of handling missing values, whereby the matrix $\mathbf{H}_t$ is simply set to zero (see, e.g., Durbin and Koopman 2012, Section 4.1). This treatment preserves optimality. Similarly, MSE-optimal forecasts are conducted by treating future values of $\mathbf{x}_t$ as missing observations.

4. The two-step estimator of the common factors

Doz *et al.* (2011) show that misspecifying the static factor model (3) with respect to some of the dynamics and cross-sectional properties may still lead to consistent estimation of the space spanned by the common factors. They propose to estimate the common factors in two steps. In the first step, preliminary parameter estimates are computed by PC. In the second step, the factors are re-estimated by linear projection. This projection can be found from one run of the Kalman smoother, allowing for idiosyncratic cross-sectional heteroscedasticity, common factor dynamics, as well as an unbalanced panel.

Because the factors and factor loadings are not uniquely identified, Doz *et al.* (2011) consider a specific rotation, outlined in the following way. Under the normalization $\mathbf{\Upsilon} = \mathbf{I}_{\mathcal{R}}$, we have that $\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Lambda}' + \mathbf{\Psi}$, so that $\mathbf{\Lambda}$ is identified up to an orthogonal multiplication; see Section 2. Let $\mathbf{\Lambda}'\mathbf{\Lambda}$ have spectral decomposition $\mathbf{\Lambda}'\mathbf{\Lambda} = \mathbf{QDQ}'$, where $\mathbf{D} = \text{diag}(\varphi_1(\mathbf{\Lambda}'\mathbf{\Lambda}), ..., \varphi_{\mathcal{R}}(\mathbf{\Lambda}'\mathbf{\Lambda}))'$ and $\mathbf{QQ}' = \mathbf{I}_N$, and consider the following representation of the factor model (3),

$$\mathbf{x}_t = \mathbf{\Lambda}_+\mathbf{g}_t + \boldsymbol{\epsilon}_t, \tag{13}$$

where $\mathbf{\Lambda}_+ = \mathbf{\Lambda Q}$ and $\mathbf{g}_t = \mathbf{Q}'\mathbf{f}_t$. By construction, $\mathbf{\Lambda}_+'\mathbf{\Lambda}_+ = \mathbf{D}$ and $\mathbf{\Lambda}_+\mathbf{\Lambda}_+' = \mathbf{\Lambda\Lambda}'$. For future reference, we define $\mathbf{P} = \mathbf{\Lambda}_+\mathbf{D}^{-1/2}$, with the property $\mathbf{P}'\mathbf{P} = \mathbf{I}_{\mathcal{R}}$. Because $\mathbf{f}_t$ has a VAR representation (4), $\mathbf{g}_t$ also has a VAR representation,

$$\mathbf{A}^+(L)\mathbf{g}_t = \mathbf{w}_t, \tag{14}$$

where $\mathbf{A}^+(L) = \mathbf{Q}'\mathbf{A}(L)\mathbf{Q}$ and $\mathbf{w}_t = \mathbf{Q}'\mathbf{z}_t$. In many cases, estimating $\mathbf{g}_t$, or any other rotation of $\mathbf{f}_t$, is as good as estimating $\mathbf{f}_t$ itself (see Section 2). In particular, the conditional mean, which is used when forecasting under squared error loss, is unaffected by the rotation.

Suppose $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ is the $N \times T$ matrix of standardized and balanced panel data with sample covariance matrix, $\mathbf{S} = T^{-1}\sum_{t=1}^{T}\mathbf{x}_t\mathbf{x}_t' = T^{-1}\mathbf{XX}'$. Let $\hat{\mathbf{D}} = \text{diag}(d_1, d_2, ..., d_{\mathcal{R}})$ be a diagonal matrix with the $\mathcal{R}$ largest eigenvalues of $\mathbf{S}$ on its main diagonal, and let $\hat{\mathbf{P}}$ be the $\mathcal{R} \times \mathcal{R}$ matrix of the associated eigenvectors as columns. Under the specific rotation $\mathbf{Q}'\mathbf{f}_t$, the PC estimators of the factors and factor loadings (see Section 2) are

$$\hat{\mathbf{g}}_t = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{P}}'\mathbf{x}_t, \tag{15}$$

$$\hat{\mathbf{\Lambda}}_+ = \hat{\mathbf{P}}\hat{\mathbf{D}}^{1/2}, \tag{16}$$

where $\hat{\mathbf{g}}_t \xrightarrow{p} \mathbf{g}_t$ and $\hat{\mathbf{\Lambda}}_+ \xrightarrow{p} \mathbf{\Lambda}_+$, as $N, T \to \infty$. Based on the PC estimators and the sample covariance matrix $\mathbf{S}$, the sample idiosyncratic covariance matrix is $\hat{\mathbf{\Psi}} = \mathbf{S} - \hat{\mathbf{\Lambda}}_+\hat{\mathbf{\Lambda}}_+'$, where $\hat{\mathbf{\Psi}} \xrightarrow{p} \mathbf{\Sigma} - \mathbf{\Lambda}_+\mathbf{\Lambda}_+' = \mathbf{\Sigma} - \mathbf{\Lambda\Lambda}' = \mathbf{\Psi}$, as $N, T \to \infty$.

The PC estimators are consistent, and in that sense they suffice.[4] They do not, however, exploit the factor time series dynamics imposed by $\mathbf{A}(L)$, or the fact that the idiosyncratic components are potentially cross-sectionally heteroscedastic (i.e. that the diagonal elements of $\mathbf{\Psi}$ are possibly different). Also, they require a balanced panel. Conveniently, the Kalman filter offers a solution to these issues.

[4]It should be noted that the consistency results presented in this paper hold under Assumptions A1-A3 and some additional general assumptions, such as technical requirements regarding the pervasiveness of the factors; see Doz *et al.* (2011) for the explicit assumptions.

Suppose that the true model specified by (3) fulfills Assumptions A1-A3. Let this model be characterized by $\Omega = \{\mathbf{\Lambda}, \mathbf{A}(L), \mathbf{\Phi}(L), \mathbf{\Psi}\}$. Additionally, let $\mathbf{\Psi}_d = \text{diag}(\psi_{1,1}, ..., \psi_{N,N})$, where $\psi_{1,1}, ..., \psi_{N,N}$ are the diagonal elements of $\mathbf{\Psi}$. Doz *et al.* (2011) consider four misspecifications of Ω, leading to approximating models defined by, respectively, $\Omega^{\mathcal{A}1} = \{\mathbf{\Lambda}, \mathbf{I}_{\mathcal{R}}, \mathbf{I}_N, \psi\mathbf{I}_N\}$, $\Omega^{\mathcal{A}2} = \{\mathbf{\Lambda}, \mathbf{I}_{\mathcal{R}}, \mathbf{I}_N, \mathbf{\Psi}_d\}$, $\Omega^{\mathcal{A}3} = \{\mathbf{\Lambda}, \mathbf{A}(L), \mathbf{I}_N, \psi\mathbf{I}_N\}$, and $\Omega^{\mathcal{A}4} = \{\mathbf{\Lambda}, \mathbf{A}(L), \mathbf{I}_N, \mathbf{\Psi}_d\}$. Under each approximating model the parameters implied by respective set $\Omega^{\mathcal{A}m}$ $(m = 1, 2, 3, 4)$ can be consistently estimated even though the true model is characterized by Ω. Moreover, the static factors can be consistently estimated by linear projection from each of the restricted parameter sets. For $\Omega^{\mathcal{A}1}$ and $\Omega^{\mathcal{A}2}$, the projection formula (7) can be used, replacing $\mathbf{\Psi}$ with $\psi\mathbf{I}_N$ and $\mathbf{\Psi}_d$, respectively. For $\Omega^{\mathcal{A}3}$ and $\Omega^{\mathcal{A}4}$, the projection can be provided by the Kalman filter based on the state space solution in Section 3, allowing us to exploit the time series dynamics composed in $\mathbf{A}(L)$. Here, imposing the rotation $\mathbf{g}_t = \mathbf{Q}'\mathbf{f}_t$, implies that $\mathbf{g}_t$ by assumption follows the VAR representation (14). Hence, for the state space solution in Section 3, $\{\mathbf{g}_t, \mathbf{A}^+, \mathbf{w}_t, \mathbf{\Lambda}^+\}$ replace $\{\mathbf{f}_t, \mathbf{A}, \mathbf{z}_t, \mathbf{\Lambda}\}$ in the representation (11). The signal and state equations (8) and (9) are then, respectively,

$$\begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{N,t} \end{pmatrix} = (\mathbf{\Lambda}_+ \ \mathbf{0} \ \cdots \ \mathbf{0}) \begin{pmatrix} \mathbf{g}_t \\ \mathbf{g}_{t-1} \\ \vdots \\ \mathbf{g}_{t-p+1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \\ \vdots \\ \epsilon_{N,t} \end{pmatrix}, \tag{17}$$

$$\begin{pmatrix} \mathbf{g}_t \\ \mathbf{g}_{t-1} \\ \vdots \\ \mathbf{g}_{t-p+1} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1^+ & \mathbf{A}_2^+ & \cdots & \mathbf{A}_{p-1}^+ & \mathbf{A}_p^+ \\ \mathbf{I}_{\mathcal{R}} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{\mathcal{R}} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{\mathcal{R}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{g}_{t-1} \\ \mathbf{g}_{t-2} \\ \vdots \\ \mathbf{g}_{t-p} \end{pmatrix} + \begin{pmatrix} \mathbf{w}_t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}. \tag{18}$$

Doz *et al.* (2011) prove that the following steps provide a consistent method for estimating the factors under $\Omega^{\mathcal{A}3}$ and $\Omega^{\mathcal{A}4}$:

1. Estimate $\mathbf{\Lambda}_+$ and $\mathbf{g}_t$ for $t = 1, 2, ..., T$ by the rotated PC-estimators (15) and (16).
2. Estimate the VAR polynomial $\mathbf{A}^+(L)$ by the ordinary least squares (OLS) regression

$$\hat{\mathbf{g}}_t = \mathbf{A}_1^+\hat{\mathbf{g}}_{t-1} + \mathbf{A}_2^+\hat{\mathbf{g}}_{t-2} + \cdots + \mathbf{A}_p^+\hat{\mathbf{g}}_{t-p} + \hat{\mathbf{w}}_t. \tag{19}$$

   Under standard aforementioned assumptions it holds as $N, T \to \infty$ that $\hat{\mathbf{A}}_m^+ \xrightarrow{p} \mathbf{A}_m^+$, for $m = 1, 2, ..., p$.
3. Run the Kalman smoother over the state space model defined by (17) and (18) to re-estimate the static factors $\mathbf{g}_t$ for $t = 1, 2, ..., T$, conditional on the estimates $\hat{\mathbf{\Lambda}}_+$ and $\hat{\mathbf{A}}^+(L)$.

Under each approximating model, we need to impose the respective misspecifying parameter restrictions. The model defined by $\Omega^{\mathcal{A}4}$ is the focus of this paper. It is the most general of the approximating models, and is therefore expected to have the highest precision for estimating the factors, unless any of the other approximating models is in fact the true, or very close to the true, model. In a simulation study in Section 5, the precision of $\Omega^{\mathcal{A}4}$ and $\Omega^{\mathcal{A}3}$ are compared.

4.1. Implementing the two-step estimator in EViews

EViews works around *objects*, consisting of information related to a specific choice of analysis. Series and matrices, for example, are different objects. Whereas matrix objects are collections of data ordered simply by their observation numbers, series objects are indexed collections of data, typically mapped against time. For example, the observed time series $x_{i,t}$, for $i = 1, 2, 3, ..., N$, are each placed in a series object, where t refers to some pre-specified frequency (e.g. monthly or quarterly). To implement the two-step estimator for the model $\Omega^{\mathcal{A}4}$ in EViews, we need to use both matrix algebra and time series commands. A convenient feature is that series may be collected and handled in groups. To move data between groups (series) and matrix objects, we use series-to-matrix and matrix-to-series commands. These and other commands and functions that we use are shown in Table 3 in Appendix A.

Two types of objects that are commonly used, are string objects (i.e. sequences of characters) and scalar objects (i.e. numbers). We will, however, make more extensive use of the related concepts *string variables* and *control variables*, which are temporary variables whose values are strings and scalars, respectively, and that only exist while the EViews program executes. String variables have names that begin with a "%" (see IHS Global Inc. 2015a, p. 92). Control variables have names that begin with a "!" (see IHS Global Inc. 2015a, p. 126). For example, the commands

```
%ser = "gdp"
!n = 5
```

create, respectively, a string variable `%ser` containing the characters 'gdp' and a control variable `!n` containing the number 5. By enclosing these variables in curly brackets, "{" and "}", EViews will replace the expressions with the underlying variable value (see section on *replacement variables* in IHS Global Inc. 2015a, p. 130). For example, the commands

```
Series {%ser}
Group G{!n}
```

create, respectively, a series object named `gdp` and a group object named `G5`.

In Appendix A, we provide a subroutine by the name of `DFM` that can be used to estimate the approximating model characterized by $\Omega^{\mathcal{A}4}$. The subroutine is defined by

```
DFM(Group XGrp, Scalar FNum, Scalar VLag, Sample S)
```

where each argument is specified by an EViews object: a group `XGrp` that should contain the observable time series in $\mathbf{x}_t$, a scalar `FNum` with the number of factors, a scalar `VLag` with the number of lags that should be used in the factor VAR representation (14), and a sample `S` over which the Kalman smoother should estimate the states. The subroutine is called by the keyword `Call`. For the scalar arguments, the subroutine may be called with a scalar object or a scalar expression. For the group and sample objects, however, the subroutine must be called with object names referring to objects of the same type. For instance, suppose there is a group object `G` and a sample object `J`. For a factor model with two factors that are VAR(1), say, the subroutine may be called by

```
Call DFM(G,2,1,J)
```

or, equivalently, by

```
!L = 1
!R = 2
Call DFM(G,!R,!L,J)
```

Within the subroutine, XGrp is then assigned the series in the group G, FNum is assigned the number 1, VLag is assigned the number 2, and S is assigned the sample J. By construction, the objects within the subroutine are global, meaning that any changes to the objects inside the subroutine will change the objects or variables that are passed into the subroutine. In the subroutine, the series in XGrp are standardized prior to the PC estimation (see below). This will also standardize the series in the group that is used to call the subroutine. EViews offers also the possibility to use local subroutines, see IHS Global Inc. (2015a, p. 156). The code is commented, and should be straightforward to follow using Table 3 in Appendix A. We will address some key points in the following subsections.

Estimation over the balanced panel: PC and factor VAR

For the PC estimation over the balanced panel, we will use matrix functions. To keep track of the position over time we then need a one-to-one correspondence between such matrices and their positions in time. For this, we use the series-to-matrix command Stomna, which preserves NA-values over the current sample. Suppose we have a group XGrp with standardized data. The following lines will create a matrix named XMat that ranges over the entire workfile range:

```
Smpl @All
Stomna(XGrp,XMat)
```

To correctly map matrix estimated components (such as the estimated factors) against time, we need to locate the balanced part of the panel. The balanced panel, which is a subset of XMat, is restricted either by the chosen sample or by the range of the data, whichever is the smallest. Here, the sample is decided by the argument S when calling the subroutine DFM, where the commands @WLeft(@PageSmpl,1) and @WRight(@PageSmpl,1) return the sample endpoints as separate strings, and the date-to-observation commands @Dtoo(@WLeft(@PageSmpl,1)) and @Dtoo(@WLeft(@PageSmpl,1)) returns the corresponding observation numbers (see Table 3). Concerning the range of the data, the function @Cifirst(XMat) creates a vector containing the time indices of the first non-missing values in the columns of XMat. The command @Max(@Cifirst(XMat)) finds the largest of these indices. Likewise, @Min(@Cilast(XMat)) finds the smallest non-missing observation number. We use these observation numbers to decide the size of the balanced panel, whose start and end points are put in the string variables %BalStart and %BalEnd, respectively.

Of course, the data set could contain missing values within the balanced panel. In that case, for matrix operations EViews will disregard entire rows (relating to an observation number), whereby the one-to-one relation between time and the position in matrices will be lost. To avoid such cases, we check each series for missing values within the balanced panel. If, for a series, missing values are found, then a message is displayed using the dialog display function @UiPrompt, and the corresponding series is removed from the group XGrp. If, in the extreme case, there are no variables left after inspection, then an exit is forced from the subroutine using the command Return.

Before the PC estimation, the data are standardized over the balanced panel, and a matrix `XMatBal` of the balanced panel is created using the command `Stom(XGrp,XMatBal)`, removing all instances of NA. The PC estimates of the factors and factor loadings (under the proposed rotation) are found by Equations (15) and (16) using matrix commands in Table 3. For the sample covariance matrix of the panel data, here named `CovXHat`, we need to use a `Sym` object, stating a symmetric matrix, instead of a regular matrix object, declared by the command `Matrix`. The former is stored in lower triangular form, and is required by EViews for the functions `@Eigenvalues` and `@Eigenvectors`, collecting eigenvalues and eigenvectors, respectively, to work. For future reference, $\hat{\mathbf{g}}_t$ is named `GHat`, $\hat{\mathbf{\Lambda}}_+$ is named `LambdaHat` and the estimated idiosyncratic covariance matrix $\hat{\mathbf{\Psi}}$ is named `CovEpsHat`.

We need to make series from the PC estimated factors. Here, we first create $\mathcal{R}$ empty series, `pc_1`, `pc_2`, ..., `pc_R`, that are placed in a group named `FGrp`. We then place the transpose of `GHat` in this group, thereby filling the empty series with the rows of `GHat` (i.e. the factor series), and produce a string variable `%Glist` containing the factor series names that can be used to estimate the factor VAR model. This VAR model, that we name `GVar`, is estimated by OLS with the commands

```
Var GVar.Ls(noconst) 1 {VLag} {%Glist}
```

where `VLag` is a scalar with the number of VAR lags that is specified as an argument when calling the subroutine `DFM`. The `noconst` option specifies that the VAR is estimated without a constant, and the part `1 {VLag}` states that lags `1` through `VLag` will be used. We place the VAR coefficients and residual covariances in matrices `AHat` and `CovWHat`, respectively. The matrix `AHat` is of size $\mathcal{R}p \times \mathcal{R}$, where each row corresponds to a column in the autoregressive coefficient matrices $\mathbf{A}_j^+ (j = 1, 2, ..., p)$ in the regression (19). The first row is the first column in $\mathbf{A}_1^+$, the second row is the first column in $\mathbf{A}_2^+$, and so on. For more details on VAR model specification in EViews, see IHS Global Inc. (2015b, p. 815) and IHS Global Inc. (2015d, Chapter 38).

Setting up the state space object

Setting up the state space object and running the Kalman smoother in EViews require only a few commands that are shown at the end of Table 3. In the subroutine, the state space object is named `DFMSS` by

```
SSpace DFMSS
```

For the state space object we need to declare signal and state properties (see van den Bossche 2011), where each line in Equations (17) and (18) has to be specified. For each equation line, if an error should exist, then it must be specified. In the state Equation (18), only the first $\mathcal{R}$ lines should have errors (relating to $\mathbf{w}_t$), whereas in the signal Equation (17), all lines should have errors. These errors are named by the keyword `@ename`.

We may also specify the error variances and covariances using the keyword `@evar`; variances and covariances that are not specified are by default zero. Following the estimation procedure for the model defined by $\Omega^{\mathcal{A}4}$, the signal errors are, by misspecification, uncorrelated over both time and the cross-section. However, the contemporary variances (the diagonal elements of $\mathbf{\Psi}$, that also correspond to the diagonal elements of $\mathbf{\Sigma}_\xi$ in Section 3) are assigned the PC estimated variances $\hat{\psi}_{1,1}, \hat{\psi}_{2,2}, ..., \hat{\psi}_{N,N}$ that are collected in the diagonal of `CovEpsHat`. The state errors relate to the factor VAR residuals, and are, under correct specification, also white noise. The variances and covariances of the $\mathcal{R}$ first state errors are assigned the elements in the residual covariance matrix from the estimated regression (19), collected in `CovWHat`.

After declaring the signal and state error properties, the subroutine `DFM` defines and appends the signal and state equations to the state space object `DFMSS` using the keywords `@signal` and `@state`. The output of main interest are the estimated factors. The subroutine lets `SVk_m` refer to factor `k`, lag `m`. Thus, we are primarily interested in `SV1_0`, `SV2_0`, ..., `SVR_0`, i.e. the states referring the contemporary factors. The remaining states, `SV1_1`, `SV2_1`, ..., `SVR_p`, refer to lags of the factors, and are simply created for the sole reason to complete the markovian state space solution for the Kalman filter and smoother (see Section 4). Accordingly, the state space object allows only one-period lags of the states.

As an example, say that we wish to model 50 observable series in $\mathbf{x}_t = (x_{1,t}, x_{2,t}, ..., x_{50,t})'$ by the static factor model (3) with two factors that follow a VAR(2) process. There are then $\mathcal{R}p = 2 \times 2 = 4$ states in the state vector $\boldsymbol{\alpha}_t = (g_{1,t}, g_{2,t}, g_{1,t-1}, g_{2,t-1})'$, which by the subroutine are named `SV1_0`, `SV2_0`, `SV1_1` and `SV2_1`, respectively. The signal equation (17) is now

$$\begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{50,t} \end{pmatrix} = \begin{pmatrix} \lambda^+_{1,1} & \lambda^+_{1,2} & 0 & 0 \\ \lambda^+_{2,1} & \lambda^+_{2,2} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \lambda^+_{50,1} & \lambda^+_{50,2} & 0 & 0 \end{pmatrix} \begin{pmatrix} g_{1,t} \\ g_{2,t} \\ g_{1,t-1} \\ g_{2,t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \\ \vdots \\ \epsilon_{50,t} \end{pmatrix}, \tag{20}$$

where $[\lambda^+_{i,j}] = \mathbf{\Lambda}_+$, and the state equation (18) is

$$\begin{pmatrix} g_{1,t} \\ g_{2,t} \\ g_{1,t-1} \\ g_{2,t-1} \end{pmatrix} = \begin{pmatrix} a^+_{(1)1,1} & a^+_{(1)1,2} & a^+_{(2)1,1} & a^+_{(2)1,2} \\ a^+_{(1)2,1} & a^+_{(1)2,2} & a^+_{(2)2,1} & a^+_{(2)2,2} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} g_{1,t-1} \\ g_{2,t-1} \\ g_{1,t-2} \\ g_{2,t-2} \end{pmatrix} + \begin{pmatrix} w_{1,t} \\ w_{2,t} \\ 0 \\ 0 \end{pmatrix}, \tag{21}$$

where $[a^+_{(1)j,k}] = \mathbf{A}^+_1$ and $[a^+_{(2)j,k}] = \mathbf{A}^+_2$.

Suppose that the elements of $\mathbf{x}_t$ are named `x1`, `x2`, ..., `x50` in the EViews workfile. The subroutine declares the equations (20) and (21) line by line. First, the $N = 50$ signal equations are declared as

```
x1 = LambdaHat(1,1)*SV1_0 + LambdaHat(1,2)*SV2_0 + e1
x2 = LambdaHat(2,1)*SV1_0 + LambdaHat(2,2)*SV2_0 + e2
.
.
x50 = LambdaHat(50,1)*SV1_0 + LambdaHat(50,2)*SV2_0 + e50
```

Then, the $\mathcal{R} = 2$ first states are declared as

```
SV1_0 = AHat(1,1)*SV1_0(-1) + AHat(2,1)*SV1_1(-1) +
        AHat(3,1)*SV2_0(-1) + AHat(4,1)*SV2_1(-1) + w1
SV2_0 = AHat(1,2)*SV2_0(-1) + AHat(2,2)*SV1_1(-1) +
        AHat(3,2)*SV2_0(-1) + AHat(4,2)*SV2_1(-1) + w2
```

and, lastly, the remaining $\mathcal{R}(p-1) = 2$ states are declared as

```
SV1_1 = SV1_0(-1)
SV2_1 = SV2_0(-1)
```

The last two states simply say that the third and fourth states are equal to the lags of states one and two, respectively, i.e. that for $j = 1, 2$, $g_{j,t-1}$ is equal to the lag of $g_{j,t}$.

To run the Kalman smoother in EViews, we need to set up an ML estimation procedure (see Section 3), even though in our case, there are no state space parameters to be estimated, as we only seek the smoothed states. To set up the ML estimation procedure, we type

```
DFMSS.ml
```

Finally, the smoothing algorithm is provided by the command

```
DFMSS.Makestates(t=smooth) *
```

The `*` implies that output will have the same name as the input, meaning that the smoothed states will have names `SV1_0`, `SV2_0`, `SV3_0`, and so on, each relating to a factor and its lags. If we want filtered states instead of smoothed states (see Section 3.1), then we can change the option to `t=filt`, see Table 3.

Remark 4.1. We have assumed that the number of factors $\mathcal{R}$ is known. In practice, this is rarely the case. We could of course estimate the number of factors (see references in Section 2.1). However, for forecasting or nowcasting, the appropriate number of factors is rather a practical concern, and can be found from forecasting evaluations. The same reasoning applies to the number of lags in the factor VAR regression (19). In case we wish to estimate the number of lags, however, we may use the criteria available in EViews; see IHS Global Inc. (2015b, p. 838) and IHS Global Inc. (2015d, Section 38). These criteria are also discussed in Lütkepohl (2007, Section 4.3).

Remark 4.2 We may iterate the approach described in this section to possibly achieve higher precision in the estimated factors. From the smoothed states, we may re-estimate the factor loadings by OLS, treating the estimated factors as being the true factors. This, in turn, can be used to find new estimates of the factors by the Kalman filter, conditional on the iterated parameters $\hat{\mathbf{\Lambda}}_{+}$ and $\hat{\mathbf{A}}_{j}^{+}$. If this procedure is iterated until convergence, then it is equivalent to the EM algorithm; see Doz *et al.* (2012).

Remark 4.3 Forecasts can be made for the signal and state equations using EViews built-in procedures; see IHS Global Inc. (2015d), p. 676. Because future values are treated as missing data by the Kalman smoother, smoothed state forecasts can be achieved by extending the sample to include the forecast period.

5. Monte Carlo simulation

For consistency, we replicate the parts of the simulation study by Doz *et al.* (2011) that compares the precision of the models $\Omega^{\mathcal{A}4}$ and $\Omega^{\mathcal{A}3}$. They consider a one-factor ($\mathcal{R} = 1$)

setup with the following data generating process (DGP):

$$\begin{aligned} x_{i,t} &= \lambda_i f_t + \boldsymbol{\epsilon}_t \ (i = 1,2,...,N; t = 1,2,...,T); \\ \lambda_i &\sim \text{ i.i.d. } \mathcal{N}(0,1); \\ (1-aL)f_t &= z_t, \text{ with } z_t \text{ i.i.d. } \mathcal{N}(0,1-a^2); \\ \boldsymbol{\Phi}(L)\boldsymbol{\epsilon}_t &= \mathbf{u}_t, \text{ with } \mathbf{u}_t \text{ i.i.d. } \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_u) \text{ and } \boldsymbol{\Phi}(L) = (1-\phi L)\mathbf{I}_N; \\ \boldsymbol{\Sigma}_u &= [\tau_{i,j}], \text{ where } \tau_{i,j} = \sqrt{\kappa_i\kappa_j}\delta^{|i-j|}(1-\phi^2), \kappa_i = \tfrac{\beta_i}{1-\beta_i}\lambda_i^2, \\ & \qquad \text{with } \beta_i \text{ i.i.d. } \mathcal{U}(b,1-b), (i,j = 1,2,...,N). \end{aligned}$$

That is, the factor and idiosyncratic components are AR(1) processes, $f_t = af_{t-1} + z_t$, $\epsilon_{i,t} = \phi\epsilon_{i,t-1} + u_{i,t}$ (i.e., the idiosyncratic AR-coefficients are the same over the cross-section), where the idiosyncratic components are possibly cross-sectionally dependent through the covariance matrix $\boldsymbol{\Sigma}_u$. The constant δ controls for the amount of idiosyncratic cross-correlation, where $\delta = 0$ implies the exact factor model with no cross-sectional dependence. The constant β_i controls the signal-to-noise ratio, i.e. the ratio between the variance of $x_{i,t}$ and the variance of the idiosyncratic component $\epsilon_{i,t}$. The autoregressive coefficients are set to $a = 0.9$ and $\phi = 0.5$, and the constants δ and b are set to $\delta = 0.5$ and $b = 0.1$. Note that the variances of the errors z_t and $u_{i,t}$ are scaled, so that the variances of f_t and $\epsilon_{i,t}$ are 1 and κ_i, respectively.[5]
The dimensions are $N = 5, 10, 25, 50, 100$ and $T = 50, 100$. The panel is unbalanced between the time points $T-3$ and T by letting $x_{i,t}$ be available for $t = 1,2,...,T-j$ if $i \leq (j+1)\frac{N}{5}$. That is, at time $T-3$, 80% of the data are available; at time $T-2$, 60% are available; at time $T-1$, 40% are available; and at time T, 20% are available. For each set of dimensions $\{N,T\}$, the parameters β_i and λ_i are drawn 50 times. Then, for each such draw, the error processes z_t and $\mathbf{u}_t$ are generated 50 times. Hence, the total number of replications is $2,500$. At each replication, we execute steps 1-3 from Section 4, providing the Kalman smoothed factors $\hat{\mathbf{g}}_t$. To evaluate the precision of the smoothed factors, the following measure is used:

$$\Delta_t = \text{Trace}\left[(\mathbf{f}_t - \hat{\mathbf{Q}}\hat{\mathbf{g}}_t)(\mathbf{f}_t - \hat{\mathbf{Q}}'\hat{\mathbf{g}}_t)'\right], \tag{22}$$

where $\hat{\mathbf{Q}}$ is the coefficient from an OLS regression of $\mathbf{f}_t$ on $\hat{\mathbf{g}}_t$, using the sample between $t = 1$ and $t = T-4$, $\hat{\mathbf{Q}} = \sum_{t=1}^{T-4}\mathbf{f}_t\hat{\mathbf{g}}_t'\left(\sum_{t=1}^{T-4}\hat{\mathbf{g}}_t\hat{\mathbf{g}}_t'\right)^{-1} = \left(\hat{\mathbf{G}}'\hat{\mathbf{G}}\right)^{-1}\hat{\mathbf{G}}'\mathbf{F}$, where $\hat{\mathbf{G}} = (\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, ..., \hat{\mathbf{g}}_{T-4})'$ and $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_{T-4})'$. A value closer to zero indicates higher precision. Because in this particular study, $\mathbf{f}_t$ is a univariate process, we have that $\Delta_t = (f_t - \hat{Q}\hat{g}_t)^2$. The simulation code is provided in Appendix B. For transparency, we will go through key steps in the code.

Initial code

To initiate the simulations a page has to be defined in the EViews work file. The initial lines therefore (i) create a work file named `Simulation` with a page on yearly frequency (say) containing 101 time points (equal to the largest T, plus 1 observation for starting values), (ii) define some scalars for the DGP, and (iii) put the estimation samples in the sample objects `E0`, `E1`, ..., `E4`, corresponding to the samples $t \in (1,T)$, $t \in (1,T-1)$, ..., $t \in (0,T-4)$, respectively. In the provided code, $T = 50$ and $N = 5$. To complete the simulation study,

[5]The variance of an AR(1) process $v_t = \varphi v_{t-1} + \varepsilon_t$, is $\mathsf{VAR}(v_t) = (1-\varphi^2)^{-1}\mathsf{VAR}(\varepsilon_t)$. Hence, $\mathsf{VAR}(f_t) = (1-a^2)^{-1}\mathsf{VAR}(z_t) = 1$, whereas $\mathsf{VAR}(\epsilon_{i,t}) = (1-\phi^2)^{-1}\mathsf{VAR}(u_{i,t}) = (1-\phi^2)^{-1}\tau_{i,i} = \kappa_i$.

these values could simply be changed for any other dimensions, say in a loop. The status line at the bottom of the EViews main window will be set to display the iteration number using the command `Statusline`. All other messages are turned off by the command `Mode Quiet`.

Generating data

We set an arbitrary randomization seed using the function `rndseed`. The idiosyncratic components are then generated as follows. The covariance matrix $\mathbf{\Sigma}_u$ depends on the parameters β_i and λ_i. For convenience, these values are generated first and placed into vectors `BetaVec` and `LambdaVec`, respectively, from which κ_i:s are generated into a vector `KappaVec`. Here, we have used the well-known result that a uniformly distributed variable in the interval $[b_-, b_+]$ may be generated as $b_- + (b_+ - b_-)\mu$, where $\mu \sim \mathcal{U}(0,1)$.

To generate the correlated idiosyncratic errors, we use the Cholesky decomposition of the covariance matrix $\mathbf{\Sigma}_u$, that is, we let $\mathbf{\Sigma}_u = \mathbf{L}\mathbf{L}'$, where $\mathbf{L}$ is a lower diagonal matrix. The multivariate normal data $\mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_u)$ is then generated as $\mathbf{L}\mathbf{e}_t$, where $\mathbf{e}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N)$.

For a coherent DGP, the starting values for the factors and idiosyncratic components, f_0 and $\epsilon_{i,0}$, should come from the stationary distributions of f_t and $\epsilon_{i,t}$, respectively. By construction of the DGP, we have that $f_t \sim \mathcal{N}(0,1)$ and $\epsilon_{i,t} \sim \mathcal{N}(0,\kappa_i)$, for all t. Hence, f_0 and $\epsilon_{i,0}$ should also be $\mathcal{N}(0,1)$ and $\mathcal{N}(0,\kappa_i)$, respectively. For convenience, we generate $T+1$ observations, and discard the starting values $x_{i,0}$ for $i = 1, 2, ..., N$. We make the panel unbalanced by assigning NA-values when data should not be available (see aforementioned). The panel data are standardized over the balanced part and placed in a group `XGrpSim`.

Remark 5.1 In the simulation study we run a large number of replications. This increases the probability that the OLS estimator of the VAR coefficient in (19), by chance may provide estimates that are non-consistent with a stationary VAR (i.e that not necessarily all values of z that satisfies $|\hat{\mathbf{A}}^{+}(z)| = 0$ lie within the unit circle; see e.g. Hamilton 1994, p. 259). To avoid such cases (they are very rare), we assign non-stationary parameter estimates values close to non-stationary values. Since we have a single factor in the simulations, we use the following conditional statement right after the matrix `AHat` is created in the subroutine `DFM`:

```
If AHat >= 1 then
  AHat = 0.999
EndIf
```

Simulation results

Let $\bar{\Delta}^{A4}_{T-s}$ and $\bar{\Delta}^{A3}_{T-s}$ be averages of the evaluation measure (22) for the models Ω^{A4} and Ω^{A3}, respectively, relating to the evaluation of a smoothed factor in the time point $T-s$ at the end of the unbalanced panel ($s = 0, ..., 4$). As for the single measure (22), a values closer to zero for the averages indicate higher precision. We call the subroutine `DFM` (see Section 4.1) and calculate $\bar{\Delta}^{A4}_{T-s}$ and $\bar{\Delta}^{A3}_{T-s}$ for each associated sample `E0`, ..., `E4` (see aforementioned). For each model (requiring a separate run of the associated scripts, see Remark 5.2), the averages are placed in the vector `DeltaMean`. Table 1 shows the results from the simulation study. The upper part of the table displays Δ^{A4}_{T-s}, and the lower part displays the ratios $\bar{\Delta}^{A4}_{T-s}/\bar{\Delta}^{A3}_{T-s}$. That is, for the ratios, a value below 1 indicates that Ω^{A4} is more accurate. Our simulation results are close to the results in Table 1 in Doz *et al.* (2011). However, we note that we

	$T=50$					$T=100$				
s	$N=5$	$N=10$	$N=25$	$N=50$	$N=100$	$N=5$	$N=10$	$N=25$	$N=50$	$N=100$
Kalman smoother with cross-sectional heteroscedasticity: $\bar{\Delta}^{A4}_{T-s}$										
4	0.33	0.32	0.32	0.34	0.33	0.20	0.19	0.17	0.18	0.18
3	0.33	0.32	0.32	0.34	0.33	0.20	0.18	0.17	0.18	0.18
2	0.35	0.33	0.32	0.34	0.33	0.21	0.19	0.17	0.18	0.18
1	0.34	0.33	0.32	0.33	0.33	0.22	0.19	0.18	0.18	0.18
0	0.37	0.34	0.32	0.34	0.33	0.25	0.20	0.18	0.19	0.18
Relative performance of Kalman smoother with cross-sectional homoscedasticity: $\bar{\Delta}^{A4}_{T-s}/\bar{\Delta}^{A3}_{T-s}$										
4	0.99	0.99	0.99	1.00	1.00	0.99	0.98	0.99	1.00	0.99
3	0.99	0.98	0.98	0.99	0.99	0.98	0.98	0.99	0.99	0.99
2	0.98	0.98	0.98	0.99	0.99	0.96	0.97	0.99	0.99	0.99
1	0.98	0.98	0.98	0.99	0.99	0.97	0.97	0.98	0.99	0.99
0	0.97	0.97	0.97	0.98	0.99	0.97	0.94	0.96	0.97	0.98

Table 1: Monte Carlo simulation.

have higher precision for small N, while slightly less precision for large N. As expected, the precision of $\Omega^{\mathcal{A}4}$ is uniformly better than the precision of $\Omega^{\mathcal{A}3}$.

Remark 5.2 For the model $\Omega^{\mathcal{A}3}$ with cross-sectional homoscedastic idiosyncratic errors, we need to impose the restriction $\mathbf{\Psi} = \psi\mathbf{I}_N$, where, following Doz *et al.* (2011), ψ is estimated by the mean of the observed idiosyncratic variances, $\hat{\psi} = N^{-1}\sum_{i=1}^{N}\hat{\psi}_{i,i} = N^{-1}\text{trace}(\hat{\mathbf{\Psi}})$. For the model $\Omega^{\mathcal{A}3}$ we therefore call a subroutine which is identical to `DFM`, except for the part specifying the variances of the signal errors, which is replaced by

```
!PsiMean = @Trace(CovEpsHat)/XNum
For !i = 1 to XNum
  DFMSS.Append @ename e{!i}
  DFMSS.Append @evar Var(e{!i}) = !PsiMean
Next
```

where `@Trace(CovEpsHat)` returns the trace of the matrix `CovEpsHat`. For the other commands, see Table 3 in Appendix A.

6. An empirical example: Nowcasting GDP

A nowcast is a forecast of the very recent past, the present, or the near future. Nowcasting the state of the economy is important for economic analysts, because key statistics on economic activity, such as GDP, are published with a significant delay. For example, in the euro area, first official estimates of GDP are published 6-8 weeks after the reference quarter. Additionally, GDP is subject to substantial revisions, as more source data become available. Meanwhile, a large number of indicators related to economic activity tend to be released well before official estimates of GDP are available, and typically at higher frequencies. For instance, whereas GDP is typically measured at quarterly frequency, data on industrial production (relating to the production side of GDP), personal consumption (relating to the expenditure side of GDP), unemployment, business survey data, and various financial data are available on monthly, weekly, or daily frequencies (for overviews, see e.g. Banbura, Giannone, and Reichlin 2011; Banbura *et al.* 2012). As an illustration, we nowcast the Swedish GDP growth, following in spirit the popular procedure by Giannone, Reichlin, and Small (2008). To be able to compare the nowcast to an outcome, we nowcast the growth for the last available quarter, which is the third quarter of 2016, denoted 2016Q3. We use the last vintage of data, where most series are from Macrobond Financial.

Let y_t denote the Swedish quarterly GDP growth at time t, measured as percentage change from period $t-1$, and let $x_{i,t}$ $(i = 1, 2, ..., N)$ denote the monthly predictors of GDP growth outlined in Table 2. The number of predictors is very large, $N = 187$. Hence, constructing models based directly on the predictors would quickly suffer from the curse of dimensionality, with limited degrees of freedom and large uncertainty in parameter estimation. Therefore, the dynamic factor model presented in Section 4 has a natural place here as an efficient dimension reducing technique. Furthermore, while collinearity is generally bad for conventional estimation methods, such as OLS, collinearity is rather preferred when extracting factors, since the goal for the extracted factors is to cover the main bulk of variation in the elements of $\mathbf{x}_t$. As can be seen from Table 2, all 187 predictors would be available within 5-6 weeks after the last day of the reference quarter. This is 2-3 weeks before the actual outcome of GDP is released. We could also choose to nowcast GDP well before this date, either by smoothing missing data or by simply leaving the data with later publication dates out of the sample.

A nowcast for y_t may be found by first estimating monthly common factors $\hat{f}_{j,t}$ for $j = 1, 2, ..., \mathcal{R}$ using the two-step estimator outlined in Section 4, and then by regressing the forecast variable on the quarterly (collapsed) factors,

$$\hat{y}_t = \hat{\alpha} + \sum_{j=1}^{\mathcal{R}} \hat{\beta}_j \hat{f}^Q_{j,t}, \tag{23}$$

where $\hat{\alpha}$ and $\hat{\beta}_j$ are estimated regression parameters, and $\hat{f}^Q_{j,t}$ are quarterly averages of the monthly factors $\hat{f}_{j,t}$.[6] We might consider also lagging the quarterly factors. However, Giannone *et al.* (2008) advocate to use only contemporary factors, because they should capture the bulk of dynamic interaction among the indicators, as well as the bulk of dynamics in GDP growth.[7] We make transformations when necessary to achieve stationarity (e.g. percentage change and seasonal adjustments of CPI). Some series are by default stationary (e.g. the Economic Tendency Survey). In other cases we perform augmented Dickey-Fuller tests (see, e.g., Lütkepohl and Krätzig 2004, p. 54). If the null hypothesis of non-stationarity cannot be rejected at the 5 percent level, then we take first-differences. For simplicity, series that are not stationary after taking first-differences are discarded, leaving $N = 184$ series. We estimate two factors ($\mathcal{R} = 2$) over the monthly sample 2001M01-2016M09, which are then used to nowcast GDP growth for 2016Q3.

There are numerous outputs and tools available for analyzing a state space object in EViews; see van den Bossche (2011) and IHS Global Inc. (2015d, Chapter 39). Here, we are mainly

[6] Another approach is to let y_t be a part of the observed vector so that $\mathbf{x}_t = (y_t, x_{1,t}, ..., x_{N,t})'$ (see, e.g., Schumacher and Breitung 2008; Banbura and Runstler 2011). At point t, y_t is missing. By projection from the Kalman filter or the EM algorithm, the minimum mean-square error estimator of y_t can be achieved.

[7] Naturally, the forecast horizon could be changed, so that we aim to make a forecast of y_{t+h}, for some $h > 0$ (see, e.g., Stock and Watson 2002a,b). The main difference between forecasting and nowcasting is that, for nowcasting we aim to exploit contemporary predictors. For forecasting, Stock and Watson (2002b) suggest lagging the factors and the dependent variable.

Producer	No. of series	Description	Publication lag
National Institute of Economic Research	47	Economic Tendency Survey: Confidence indicators, Micro index, Macro index, and choices of subsectors	No lag
Statistics Sweden	17	Industrial new orders	5-6 weeks
Statistics Sweden	30	Industrial production: Industrial production index (IPI) and subcategories	5-6 weeks
Statistics Sweden	3	Energy production data: Total, manufacturing, and residential sector	5-6 weeks
Statistics Sweden	2	Foreign trade of goods: Total export, total import	3-4weeks
Statistics Sweden	1	Household consumption indicator	5-6 weeks
Statistics Sweden & HUI Research	1	Retail-sales index	3-4 weeks
Statistics Sweden & BIL Sweden	1	New registrations of cars	2 days
Statistics Sweden	1	Producer price index (PPI)	2-3 weeks
Statistics Sweden	18	Consumer price indices (CPI, CPIF) and choices of subcategories	2-3 weeks
Statistics Sweden	6	Labor data	2-3 weeks
Public Employment Services	9	Vacancies and employment training programs	
Swedbank & Silf	21	Purchasing Managers' Index (PMI): Total and subcategories	< 1 week
SEB	1	Housing indicator	No lag
Nasdaq OMX Valueguard	3	Housing indices: HOX, HOX flats, HOX houses	2-3 weeks
Nasdaq Nordic	1	OMX Stockholm index	No lag
Riksbanken	11	Exchange rates against large currencies, and effective exchange rates	No lag
Riksbanken	6	Swedish interest rates: repo rate, treasury-bills and government bonds	No lag
ECB	4	Euro area interest rates: refi rate, German treasury bills and German government bonds	No lag
Federal Reserve	4	US interest rates: Fed funds rate, treasury bills and government bonds	No lag

Table 2: Variables for nowcasting GDP growth.

interested in the estimated factors and their associated error bounds. EViews allows the user to create filtered states, one-step ahead predictions of states or smoothed states (see Section 3.1). In the subroutine, we use smoothed states. To create a graph of them, we can go to View\State Views\Graph State Series in the interface menu and choose `Smoothed State Estimates`, or we can use the command `DFMSS.Stategraph(t=smooth)`. By default, EViews provides uncertainty bounds constructed from 2 standard errors. In our case, the estimated uncertainty in terms of standard errors is somewhat larger for the second smoothed state than for the first (see Figure 1).

The estimated regression (23) and its nowcast can easily be obtained in EViews using the interface menus (see IHS Global Inc. 2015c, Chapter 2), or by programming (see IHS Global Inc. 2015a, p. 360-362). Suppose that there is page named `M` on monthly frequency, containing the monthly predictors and smoothed states, and a page named `Q` on quarterly frequency, containing the GDP series, named `gdp`. The following lines copy the estimated factors from the monthly page to the quarterly page, rename them `f1` and `f2`, respectively, and estimate the nowcasting regression:

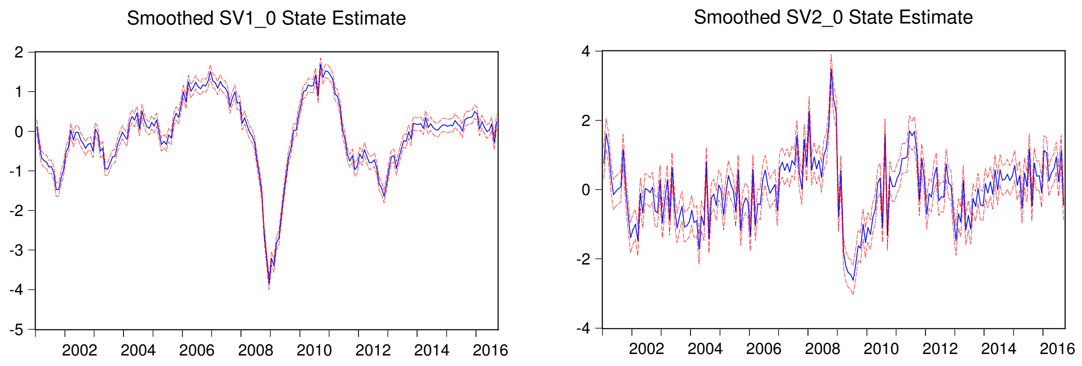

Figure 1: Smoothed monthly state estimates with error bounds.

```
Smpl 2001Q1 2016Q3
Copy(c=a) M\SV1_0 Q\f1
Copy(c=a) M\SV2_0 Q\f2
Smpl @First 2016Q2
Equation Nowcast.Ls gdp c f1 f2
Smpl 2016Q3 2016Q3
Nowcast.Forecast gdpnow
```

The option `c=a` in the `Copy` command specifies that an average should be used as the conversion method. The fourth and fifth lines estimate the regression (23) by OLS over the sample 2001Q1-2016Q2, where the name of the equation object is `nowcast`. The last two lines makes the forecast (nowcast) for 2016Q3, which is saved in the series `gdpnow`. By construction, this series also contains the outcome of the series `gdp` before 2016Q3.

Our nowcast for the GDP growth 2016Q3 is 0.50 percent. The true outcome was 0.49 percent. Having this information available before the actual outcome is released can be of great value to analysts. Naturally, it would be desirable to have the associated error bounds for the nowcast. However, basing the error bounds on theory is a bit more cumbersome than usual, since the regressors in Equation (23) (i.e. the factors) are estimated, and not observed variables (see Bai and Ng 2006). EViews provides conventional error bounds based on the regression (23) (see IHS Global Inc. 2015d, p. 144). In this particular case, they would be too narrow.

It could also be of interest to plot the quarterly collapsed factors together with GDP growth. The following lines creates a graph named `Fig` of the series `gdpnow` (i.e. GDP growth including its nowcast) and the factors `f1` and `f2` (see Figure 2):

```
Smpl 2001Q1 2016Q3
Graph Fig.Line gdpnow f1 f2
Fig.Options Linepat
Fig.SetElem(1) lpat(solid)
Fig.SetElem(2) lpat(dash1)
Fig.SetElem(3) lpat(dash2)
```

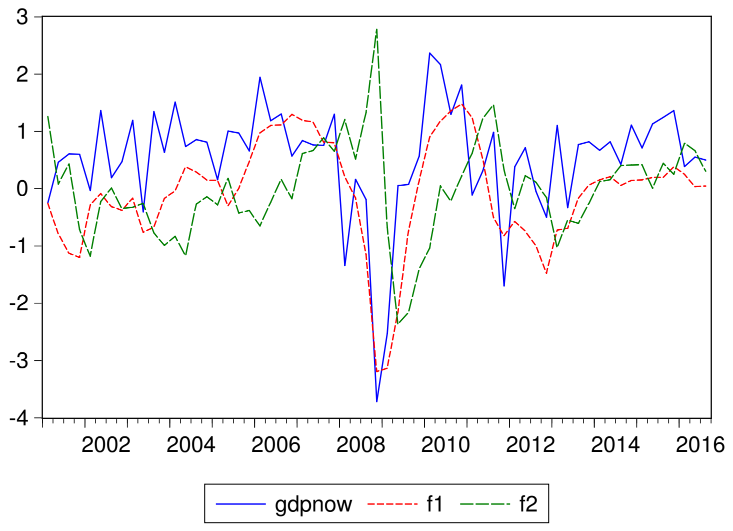

Figure 2: GDP growth and quarterly factors.

The last four lines change the settings so that the series `gdp` is displayed as a solid line, and the factor series are displayed as dashed lines (see about `Graph` procedures in IHS Global Inc. 2015b, p. 228). Judging from Figure 2, it is tempting to conclude that the second factor is lagging. However, its movement is rather a consequence of being (almost) orthogonal to the first factor.

7. Conclusions

In this paper, we have shown how to estimate a dynamic factor model by the Kalman filter and smoother in EViews. Using only a small amount of programming we have provided a global subroutine based on the two-step estimator of Doz *et al.* (2011) that can be useful to a broad range of economists or statisticians using large panel data to fit a dynamic factor model. Because the estimator requires only to compute principal components and then make one run with the Kalman smoother, the procedure is fast. The code could easily be altered to meet specific user needs. For example, a user may find it more convenient to work with a local subroutine. Or a user may wish to add estimation procedures for the number of factors and the number of lags in the factor VAR. These modifications would require only basic knowledge in the EViews programming language.

A. A subroutine with the two-step estimator

In this Appendix, we provide a subroutine by the name of DFM that can be used to estimate a dynamic factor model as outlined in Section 4. The subroutine has four arguments: (i) `XGrp`, a group containing the observable time series, (ii) `FNum`, a scalar with the number of factors, (iii) `VLag`, a scalar with the number of lags that should be used for the factor VAR, and (iv) S, a sample over which the kalman smoother should estimate the states. To call the subroutine, the code should be placed in an EViews program file (.prg) that is either the main script or is included in the main script using the command Include; see Chapter 6 in IHS Global Inc. (2015a). The code is commented, and EViews commands and functions that are used are shown in Table 3, in order of first occurrence. In Section 4.1, we illustrate how to call the subroutine and explain key steps in the code.

```
Subroutine DFM(Group XGrp, Scalar FNum, Scalar VLag, Sample S)

'SCALAR WITH THE NUMBER OF OBSERVED VARIABLES
Scalar XNum = XGrp.@Count

'FINDING SAMPLE FOR BALANCED PANEL
'Creating matrix that preserves NAs with name 'XMat'
Smpl @All
Stomna(XGrp,XMat)
'Finding start and end dates for balanced panel, restricted by sample S
Smpl S
If @Dtoo(@WLeft(@PageSmpl,1)) >= @Max(@Cifirst(XMat)) Then
  %BalStart = @WLeft(@PageSmpl,1)
Else
  %BalStart = @Otod(@Max(@Cifirst(XMat)))
EndIf
If @Dtoo(@WRight(@PageSmpl,1)) <= @Min(@Cilast(XMat)) Then
  %BalEnd = @WRight(@PageSmpl,1)
Else
  %BalEnd = @Otod(@Min(@Cilast(XMat)))
EndIf

'CHECKING THAT THERE ARE NO MISSING VALUES WITHIN BALANCED PANEL
Smpl {%BalStart} {%BalEnd}
!i = 0
While !i < XGrp.@Count
  !i = !i+1
  %SerName = XGrp.@Seriesname(!i)
  Series NAtest = @IsNa({%SerName})
  If @Sum(NAtest) > 0 Then
    %PromptStr = %SerName + " has NAs within the balanced panel."
    %PromptStr = %PromptStr + " The variable is removed."
    @UiPrompt(%PromptStr)
    XGrp.Drop {%SerName}
    If XGrp.@Count > 0 Then
      Matrix LambdaHat = LambdaHat.@Droprow(!i)
      Matrix CovEpsHat = CovEpsHat.@Droprow(!i)
      !i = !i-1
    Else
      'If no variables remain, the subroutine is ended
      @UiPrompt("There are no variables left. The subroutine is ended.")
      Return
    EndIf
  EndIf
WEnd
'Recreating scalar with number of variables
XNum = XGrp.@Count

'ESTIMATING BY PC
'Standardizing data over balanced panel
For !i = 1 to XNum
  %Series = XGrp.@Seriesname(!i)
  Smpl {%BalStart} {%BalEnd}
  !Std = @StDev({%Series})
  !Mean = @Mean({%Series})
  Smpl @All
  {%Series} = ({%Series}-!Mean)/!Std
Next
Smpl {%BalStart} {%BalEnd}
'Creating matrix of balanced panel (T times N)
Stom(XGrp,XMatBal)
'Computing sample covariance matrix of x
Sym CovXHat = (@Transpose(XMatBal))*XMatBal/@Rows(XMatBal)
'Computing ordered eigenvalues and associated eigenvectors
Vector EigVals = @Sort(@Eigenvalues(CovXHat),"d")
Vector EigRanks = @Ranks(EigVals,"a","i")
Matrix EigVecs = @Rapplyranks(@Eigenvectors(CovXHat),EigRanks)
'Estimating factors (GHat: N times T) and loadings (LambdaHat: N times R),
'and residual covariance matrix (CovEpsHat: N times N)
Matrix DHat = @Makediagonal(@Subextract(EigVals,1,1,FNum,1))
Matrix PHat = @Subextract(EigVecs,1,1,XNum,FNum)
Matrix GHat = (@Sqrt(@Inverse(DHat)))*(@Transpose(PHat))*(@Transpose(XMatBal))
Matrix LambdaHat = PHat*(@Sqrt(DHat))
'Estimating residual covariance matrix
Matrix CovEpsHat = CovXHat-(LambdaHat*(@Transpose(LambdaHat)))
'Creating factor series, that will be used for constructing states
Group FGrp
For !i = 1 to FNum
  Series pc_{!i}
  FGrp.Add pc_{!i}
Next
'Placing values in factor series
Matrix TGHat = @Transpose(GHat)
Mtos(TGHat,FGrp)
'Creating list with names of factor series
%Glist = FGrp.@Members
```

```
'ESTIMATING VAR ON FACTORS, WITHOUT CONSTANT
Smpl {%BalStart} {%BalEnd}
Var GVar.Ls(noconst) 1 {VLag} {%Glist}
'Placing VAR coefficients in matrix
Matrix AHat = GVar.@Coefmat
'Creating VAR residual covariance matrix
Matrix CovWHat = GVar.@Residcov

'CREATING STATE SPACE OBJECT WITH NAME 'DFMSS'
SSpace DFMSS

'NAMING SIGNAL RESIDUALS AND ASSIGNING THEM PC-ESTIMATED VARIANCES
For !i = 1 to XNum
  DFMSS.Append @ename e{!i}
  DFMSS.Append @evar Var(e{!i}) = CovEpsHat(!i,!i)
Next

'NAMING STATE RESIDUALS AND ASSIGNING THEM ESTIMATED VAR RESIDUAL
'VARIANCE/COVARIANCES
For !i = 1 to FNum
  DFMSS.Append @ename w{!i}
  DFMSS.Append @evar Var(w{!i}) = CovWHat(!i,!i)
  If FNum > 1 Then
    For !j = !i+1 to FNum
      DFMSS.Append @evar Cov(w{!i},w{!j}) = CovWHat(!i,!j)
    Next
  EndIf
Next
'DEFINING THE SIGNAL EQUATIONS
For !i = 1 to XNum
  'Making string variable that is filled with signal equations
  %Signal = XGrp.@Seriesname(!i)+" ="
  For !j = 1 to FNum
    %Signal = %Signal + " LambdaHat(" + @Str(!i) + "," + @Str(!j) + ")*SV"
    %Signal = %Signal + @Str(!j) + "_0 +"
  Next
  'Adding error and appending signal equations to state space object
  %Signal = %Signal + " e" + @Str(!i)
  DFMSS.Append @Signal {%Signal}
Next

'DEFINING THE R (= NUMBER OF FACTORS) FIRST STATE EQUATIONS
For !i = 1 to FNum
 'Making string variable that is filled with state equation
  %State = "SV" + @Str(!i) + "_0 ="
  For !a = 1 to FNum
    For !j = 1 to VLag
```

```
      %State = %State + " AHat(" + @Str(!j + VLag*(!a-1)) + "," + @Str(!i)
      %State = %State + ")*SV" + @Str(!a) + "_" + @Str(!j-1) + "(-1) +"
    Next
  Next
  'Adding error and appending state equations to state space object
  %State = %State + " w" + @Str(!i)
  DFMSS.Append @State {%State}
Next

'DEFINING THE REMAINING STATE EQUATIONS, WITHOUT ERRORS
For !i = 1 to FNum
  For !j = 1 to VLag-1
    %State = "SV" + @Str(!i) + "_" + @Str(!j) + " = SV" + @Str(!i) + "_"
    %State = %State + @Str(!j-1) + "(-1)"
    DFMSS.Append @State {%State}
  Next
Next

'SETTING UP SMOOTHER
Smpl S
DFMSS.ml
DFMSS.Makestates(t=smooth) *

'DELETING USED OBJECTS
Delete FNum XNum EigVals EigRanks EigVecs GHat TGHat GVar FGrp PHat
Delete DHat CovXHat XMat XMatBal NAtest pc_*

EndSub
```

Command/function	Description
`Group G`	Creates group object G
`Scalar c`	Creates scalar object c
`Sample S`	Creates sample object S
`Smpl` *spec*	Sets workfile sample to *spec*. For keywords, see IHS Global Inc. (2015a), p. 474.
`G.@Count`	Returns number of series in group G
`Stomna(A,B,`*smp*`)`	Series-to-matrix with NAs: Puts data from series A into matrix B, restricted by sample *smp*, while preserving missing values
`@Dtoo(`*str*`)`	Date-to-observation: Returns observation number corresponding to date in string *str*
`@WLeft(`*str*`,`*j*`)`	Returns *j*th word from the left in string *str*
`@PageSmpl`	Returns space delimited list with the start and end points of the current sample
`@Max(a)`	Returns maximum value of elements in vector a
`@Cifirst(A)`	Returns vector with indices of first non-missing values in the columns of matrix A
`@Otod(`*n*`)`	Observation-to-date: Returns date corresponding to observation number *n*
`@WRight(`*str*`,`*j*`)`	Returns *j*th word from the right in string *str*
`@Min(a)`	Returns minimum value of elements in vector a
`@Cilast(A)`	Returns vector with indices of last non-missing values in the columns of matrix A
`G.@Seriesname(`*i*`)`	Returns name of *i*th series in group G
`@IsNa(x)`	Returns series that is 1 when series x is non-missing and 0 when x is missing
`@Sum(x)`	Returns sum of series x within the current sample
`@Uiprompt(`*str*`)`	Displays dialog with message in string *str*
`G.Drop` *ser*	Drops series *ser* from group G
`@Std(x)`	Returns sample standard deviation of series x within the current sample
`@Mean(x)`	Returns sample mean of series x within the current sample
`Stom(A,B,`*smp*`)`	Series-to-matrix: Puts data from series A into matrix B, restricted by sample *smp*
`Sym(`*n*`) A`	Creates symmetric matrix object A of size $n \times n$
`@Transpose(A)`	Forms transpose of matrix A
`@Rows(A)`	Returns number of rows of matrix A
`@Sort(A,`*o*`)`	Returns sorted elements of matrix A by *o*: ”a” (ascending), ”d” (descending)
`@Eigenvalues(A)`	Returns vector with the eigenvalues of matrix A
`@Ranks(A,`*o*`,`*t*`)`	Returns ranks of the elements of matrix A by *o*: ”a” (ascending), ”d” (descending). Ties are broken by *t*; e.g. ”i” (ignore), see (IHS Global Inc. 2015a), p. 683
`@Rapplyranks(A,b)`	Returns matrix where the columns of matrix A are re-ordered by ranks in vector b
`@Eigenvectors(A)`	Returns matrix with the eigenvectors of matrix A as columns
`@Makediagonal(a,`*k*`)`	Creates diagonal matrix with vector a in the *k*th diagonal
`@Subextract(A,`*n*1`,`*n*2`,`*n*3`,`*n*4`)`	Returns submatrix of matrix A from upper left row *n*1 and column *n*2 to lower right row *n*3 and column *n*4
`@Sqrt(A)`	Returns square root of matrix A
`@Inverse(A)`	Returns inverse of matrix A
`G.Add` *ser*	Adds series *ser* to group G
`Mtos(A,G,`*smp*`)`	Matrix-to-series: Puts data from matrix A into group G, within sample *smp*
`G.@Members`	Returns space delimited list of names of the series in group G
`V.Ls(`*o*`)`	Estimates VAR object V by OLS. For options *o*, see IHS Global Inc. (2015b, p. 839)
`V.@Coefmat`	Returns array with estimated VAR coefficients for VAR object V
`V.@Residcov`	Returns estimated VAR residual covariance matrix of VAR object V
`sspace SS`	Creates state space object SS
`SS.Append`	Appends specification line to state space object SS
`SS.ML`	Declares maximum likelihood estimation for state space object SS
`SS.Makestates(`*t*`)`	Generates state series or their standard errors from estimated state space object SS by output type *t*; e.g. ”filt” (filtered states), ”smooth” (smoothed states), ”pred” (one-step ahead predictions), see IHS Global Inc. (2015b, p. 639)
`SS.Stategraph(`*t*`)`	Displays graphs of state series or their standard errors from estimated state space object SS by output type *t*; e.g. ”filt” (filtered states), ”smooth” (smoothed states), ”pred” (one-step ahead predictions), see IHS Global Inc. (2015b, p. 646)
`@Signal`	Keyword that specifies a signal equation
`@State`	Keyword that specifies a measurement equation
`@ename`	Keyword that names an error
`@evar`	Keyword that specifies covariances between errors
`Delete` *obj*	Deletes object *obj* from workfile

Table 3: Some functions and commands in EViews used by the subroutine `DFM`.

B. The simulation code

In this Appendix, we provide the code that was used to run the Monte Carlo simulations in Section 5. The code is commented. Commands and functions in EViews that are used in the code, but are not shown in Table 3, are shown in Table 4 in order of first occurrence. Note that to execute the simulation code, either the simulation code has to be placed in the same script as the subroutine presented in Appendix A or the path to the subroutine has to be included in the simulation script using the command `Include`; see Chapter 6 in IHS Global Inc. (2015a). For example, if the subroutine is placed in a program file with name `dynfacsub.prg` that is placed in the folder `C:\SUB`, then the following command should be placed early in the simulation script:

`Include C:\SUB\dynfacsub.prg`

The following code was used in the simulation study (see Section 5 for details).

```
'CREATING A WORKFILE WITH 101 TIME POINTS
WfCreate(Wf = Simulation, Page = A) A 2000 2100

'CREATING INPUT FOR SIMULATION STUDY
Scalar a = 0.9     'Autoregressive coefficient for factor
Scalar phi = 0.5   'Autoregressive coefficient for idiosyncratic components
Scalar gamma = 0.5 'Parameter that controls idiosyncratic covariance
Scalar b = 0.1     'Parameter for uniform distribution
Scalar T = 50      'Sample size (time dimension)
Scalar N = 5       'Number of variables (cross-section dimension)
Scalar Rep1 = 50   'First number of replications
Scalar Rep2 = 50   'Second number of replications

'CREATING EMPTY MATRIX FOR EVALUATION, AND DEFINING ASSOCIATED SAMPLES
'(FIRST VALUE IS DISCARDED)
Matrix(Rep1*Rep2,5) DeltaMat
Sample E4 2001 2000+T-4
Sample E3 2001 2000+T-3
Sample E2 2001 2000+T-2
Sample E1 2001 2000+T-1
Sample E0 2001 2000+T

'TURNING OFF MESSAGES
Mode Quiet

'SETTING RANDOMIZATION SEED
rndseed 123

'GENERATING DATA AND APPLYING KALMAN FILTER
For !Rep1 = 1 to Rep1
  'GENERATING LAMBDA, BETA AND KAPPA
  Vector LambdaVec = @Mnrnd(N)
  Vector BetaVec = @Ones(N)*b + (1-2*b)*@Mrnd(N)
  Vector(N) KappaVec
  For !s = 1 to N
    KappaVec(!s) = (BetaVec(!s)/(1-BetaVec(!s)))*((LambdaVec(!s))^2)
  Next

  For !Rep2 = 1 to Rep2
    'SHOWING THE ITERATION NUMBER IN STATUSLINE
    !Rep = !Rep2 + (!Rep1-1)*Rep2
    Statusline Rep: !Rep

    'GENERATING CORRELATED IDIOSYNCRATIC COMPONENTS
    '(I) Creating idiosyncratic error covariance matrix, CovU
    Sym(N,N) CovU
    For !s = 1 to N
      For !ss = 1 to N
        CovU(!s,!ss) = (@Sqrt(KappaVec(!s)*KappaVec(!ss)))
        CovU(!s,!ss) = CovU(!s,!ss)*(gamma^(@Abs(!s-!ss)))*(1-(phi^2))
      Next
    Next
    '(II) Creating idiosyncratic error matrix (N times T):
    'Multivariate normal with covariance matrix CovU
    Matrix CholHalf = @Cholesky(CovU)
    Matrix UMat = CholHalf*@Mrnd(N,T)
    '(III) Generating epsilon
    For !s = 1 to N
      Smpl @First @First
      Series eps_{!s} = Nrnd*@Sqrt(KappaVec(!s))
      For !ss = 1 to T
        Smpl @First+!ss @First+!ss
        Series eps_{!s} = phi*eps_{!s}(-1) + UMat(!s,!ss)
      Next
    Next

    'GENERATING FACTOR
    Smpl @First @First
    Series f = Nrnd
    For !ss = 1 to T
      Smpl @First+!ss @First+!ss
      Series f = a*f(-1) + @Sqrt(1-(a^2))*Nrnd
    Next

    'GENERATING UNBALANCED PANEL DATA
    Group XGrpSim
    For !s = 1 to N
      !j = @Ceiling(5*!s/N-1)
      Smpl @First @First+T-!j
```

```
      Series x_{!s} = LambdaVec(!s)*f + Eps_{!s}
      XGrpSim.Add x_{!s}
    Next

    'CALLING SUBROUTINE AND CALCULATING EVALUATION MEASURE FOR SAMPLES E0,...,E4
    Stom(f,Fser,E4)
    For !s = 0 to 4
      Call DFM(XGrpSim,1,1,E{!s})
      Stom(SV1_0,Gser,E4)
      Scalar QHat = (@Inverse((@Transpose(Gser)*Gser)))*(@Transpose(Gser)*Fser)
      Smpl E{!s}
      DeltaMat(!Rep,5-!s) = (@Last(f)-(QHat*@Last(SV1_0)))^2
      Delete SV1_0
    Next

    'DELETING INNER LOOP OBJECTS
    Delete CovU UMat Eps_* f XGrp* Fser Gser QHat CholHalf DFMSS CovEpsHat
    Delete CovWHat AHat LambdaHat
  Next

  'DELETING OUTER LOOP OBJECTS
  Delete LambdaVec BetaVec KappaVec
Next

'AVERAGING EVALUATION MEASURE
Vector DeltaMean = @Cmean(DeltaMat)

'DELETING REMAINING USED OBJECTS
Delete E* x_* *vec DeltaMat gamma phi rep1 rep2 N T a b
```

Command/function	Description
`WFCreate(`o`)` *arg*	Creates new workfile. For options o and arguments *arg*, see IHS Global Inc. (2015a), p. 508
`Matrix(`k`,`m`) A`	Creates matrix object A of size $k \times m$
`Mode`	Sets program mode, see IHS Global Inc. (2015a), p. 135
`rndseed`	Seeds random number generator
`Vector a`	Creates vector object a
`@Mnrnd(`k`,`m`)`	Returns matrix of size $k \times m$ from the standard normal distribution
`@Ones(`k`,`m`)`	Returns matrix of ones of size $k \times m$
`@Mrnd(`k`,`m`)`	Returns matrix of size $k \times m$ from the standard uniform distribution
`Statusline` *message*	Sends *message* to the status line
`@Abs()`	Returns absolute value of n
`@Cholesky(A)`	Returns lower triangular Cholesky decomposition of symmetric matrix A
`Series` *ser*	Creates series object *ser*
`Nrnd`	Returns draws from the standard normal distribution
`@Ceiling(`n`)`	Returns smallest integer not less than n
`Call`*sub*	Calls subroutine *sub*
`@Last(x)`	Returns last non-missing value of series x within current sample
`@Cmean(A)`	Returns vector with column means of matrix A

Table 4: Some functions and commands in EViews used in the simulation code.

References

Ahn SC, Horenstein AR (2013). “Eigenvalue Ratio Test for the Number of Factors.” *Econometrica*, **81**, 1203–1227.

Anderson TW (2003). *An Introduction to Multivariate Statistical Analysis*. Wiley, New York.

Bai J (2003). “Inferential Theory for Factor Models of Large Dimensions.” *Econometrica*, **71**, 135–171.

Bai J, Li K (2012). “Statistical Analysis of Factor Models of High Dimension.” *The Annals of Statistics*, **40**, 436–465.

Bai J, Li K (2016). “Maximum Likelihood Estimation and Inference for Approximate Factor Models of High Dimension.” *The Review of Economics and Statistics*, **98**, 298–309.

Bai J, Ng S (2002). “Determining the Number of Factors in Approximate Factor Models.” *Econometrica*, **70**, 191–221.

Bai J, Ng S (2006). “Confidence Intervals for Diffusion Index Forecasts and Inference for Factor-Augmented Regressions.” *Econometrica*, **74**, 1133–1150.

Bai J, Ng S (2007). “Determining the Number of Primitive Shocks in Factor Models.” *Journal of Business & Economic Statistics*, **25**, 52–60.

Bai J, Ng S (2008). “Large Dimensional Factor Analysis.” *Foundations and Trends in Econometrics*, **3**, 89–163.

Bai J, Ng S (2013). “Principal Components Estimation and Identification of Static Factors.” *Journal of Econometrics*, **176**, 18–29.

Bai J, Wang P (2014). “Identification Theory for High Dimensional Static and Dynamic Factor Models.” *Journal of Econometrics*, **178**, 794–804.

Banbura M, Giannone D, Modugno M, Reichlin L (2012). “Now-Casting and the Real-Time Data Flow.” In G Elliott, A Timmermann (eds.), *Handbook of Economic Forecasting*, volume 2A. Elsevier-North Holland, Amsterdam.

Banbura M, Giannone D, Reichlin L (2011). “Nowcasting.” In M Clements, D Hendry (eds.), *The Oxford Handbook of Economic Forecasting*. Oxford University Press, Oxford.

Banbura M, Runstler G (2011). “A Look Into the Factor Model Black Box: Publication Lags and the Role of Hard and Soft Data in Forecasting GDP.” *International Journal of Forecasting*, **27**, 333–346.

Bernanke BS, Boivin J, Eliasz P (2005). “Measuring the Effects of Monetary Policy: A Factor-Augmented Vector Autoregressive (FAVAR) Approach.” *The Quarterly Journal of Economics*, **120**, 387–422.

Boivin J, Giannoni MP, Mihov I (2009). “Sticky Prices and Monery Policy: Evidence from Disaggregaed US Data.” *American Economic Review*, **99**, 350–384.

Breitung J, Eickmeier S (2006). "Dynamic Factor Models." *Allgemeines Statistisches Archiv*, **90**, 27–42.

Chamberlain G, Rothschild M (1983). "Arbitrage, Factor Structure, and Mean-variance Analysis on Large Asset Markets." *Econometrica*, **51**, 1281–1304.

Diebold FX, Li C (2006). "Forecasting the Term Structure of Government Bond Yields." *Journal of Econometrics*, **130**, 337–364.

Diebold FX, Rudebusch GD, Aruoba SB (2006). "The Macroeconomy and the Yield Curve: A Dynamic Latent Factor Approach." *Journal of Econometrics*, **131**, 309–338.

Doz C, Giannone D, Reichlin L (2011). "A Two-Step Estimator for Large Approximate Dynamic Factor Models Based on Kalman Filtering." *Journal of Econometrics*, **164**, 180–205.

Doz C, Giannone D, Reichlin L (2012). "A Quasi-Maximum Likelihood Approach for Large, Approximate Dynamic Factor Models." *The Review of Economics and Statistics*, **94**, 1014–1024.

Durbin J, Koopman SJ (2012). *Time Series analysis by State Space Methods.* Oxford University Pres, Oxford.

Eickmeier S (2007). "Business Cycle Transmission from the US to Germany - A Structural Factor Approach." *European Economic Review*, **51**, 521–551.

Forni M, Giannone D, Lippi M, Reichlin L (2009). "Opening the Black Box: Structural Factor Models With Large Cross Sections." *Econometric Theory*, **25**, 1319–1347.

Forni M, Hallin M, Lippi M, Reichlin L (2000). "The Generalized Dynamic-Factor Model: Identification and Estimation." *The Review of Economics and Statistics*, **82**, 540–554.

Forni M, Hallin M, Lippi M, Reichlin L (2004). "The Generalized Dynamic Factor Model Consistency and Rates." *Journal of Econometrics*, **119**, 231–255.

Forni M, Reichlin L (1998). "Let's Get Real: A Factor Analytical Approach to Disaggretgated Business Cycle Dynamics." *Review of Economic Studies*, **65**, 453–473.

Giannone D, Reichlin L, Small D (2008). "Nowcasting: The Real-Time Informational Content of Macroeconomic Data." *Journal of Monetary Economics*, **55**, 665–676.

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

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

IHS Global Inc (2015a). "EViews 9 Command and Programming Reference." Irvine CA, USA.

IHS Global Inc (2015b). "EViews 9 Object Reference." Irvine CA, USA.

IHS Global Inc (2015c). "EViews 9 User's Guide I." Irvine CA, USA.

IHS Global Inc (2015d). "EViews 9 User's Guide II." Irvine CA, USA.

Koopman SJ, Shephard N, Doornik JA (1999). "Statistical Algorithms for Models in State Space Using SsfPack 2.2." *The Econometrics Journal*, **2**, 107–160.

Lütkepohl H, Krätzig M (2004). *Applied Time Series Econometrics.* Cambridge University Press, Cambridge.

Ludvigson SC, Ng S (2007). "The Empirical Risk-Return Relation: A Factor Analysis Approach." *Journal of Financial Economics*, **83**, 171–222.

Lütkepohl H (2007). *New Introduction to Multiple Time Series Analysis.* Springer, Berlin.

Onatski A (2010). "Determining the Number of Factors from Empirical Distribution of eigenvalues." *The Review of Economic and Statistics*, **92**, 1004–1016.

Phillips PCB, Moon HR (1999). "Linear Regression Limit Theory for Nonstationary Panel Data." *Econometrica*, **67**, 1057–1111.

Ritschl A, Sarferaz S, Uebele M (2016). "The U.S. Business Cycle, 1867-2006: A Dynamic Factor Approach." *The Review of Economics and Statistics*, **98**, 159–172.

Schumacher C, Breitung J (2008). "Real-Time Forecasting of German GDP Based on a Large Factor Model With Monthly and Quarterly Data." *International Journal of Forecasting*, **24**, 386–398.

Stock JH, Watson MW (2002a). "Forecasting Using Principal Components from a Large Number of Predictors." *Journal of the American Statistical Association*, **97**, 1167–1179.

Stock JH, Watson MW (2002b). "Macroeconomic Forecasting Using Diffusion Indexes." *Journal of Business and Economic Statistics*, **20**, 147–162.

Stoica P, Jansson M (2009). "On Maximum Likelihood Estimation in Factor Analysis - an Algebraic Derivation." *Signal Processing*, **89**, 1260–1262.

van den Bossche FAM (2011). "Fitting State Space Models with EViews." *Journal of Statistical Software*, **41**, 1–16.

White H (1982). "Maximum Likelihood Estimation of Misspecified Models." *Econometrica*, **50**, 1–25.

Zhou X, Solberger M (2017). "A Lagrange Multiplier-Type Test for Idiosyncratic Unit Roots in the Exact Factor Model." *Journal of Time Series Analysis*, **38**, 22–50.

Affiliation:

Martin Solberger
Department of Statistics
Uppsala University
PO Box 513, SE-75120, Uppsala, Sweden
E-mail: `Martin.Solberger@statistics.uu.se`
URL: `http://www.statistics.uu.se/`