Dynamic Factor Models in gretl. The DFM package

Riccardo (Jack) Lucchetti
Univertità Politecnica delle Marche
r.lucchetti@univpm.it

Ioannis A. Venetis
Department of Economics, University of Patras,
University Campus, Rio 26504, Greece
ivenetis@upatras.gr

version 0.43

Abstract

This package deals with the estimation of dynamic factor models (DFM); for the moment, three factor extraction techniques are available, but we plan to add more in future versions. Further additions will include parameter restrictions.

Contents

1 The model 2

2 The estimators 3
   2.1 Principal components (PC) ........................................... 3
   2.2 Two-step estimator (TS) ........................................... 3
   2.3 Quasi ML - EM estimator (ML) .................................... 3
   2.4 Parameter restrictions ............................................ 5

3 The DFM function package 5
   3.1 Installation ........................................................... 5
   3.2 By scripting ......................................................... 5
   3.3 Using the GUI ..................................................... 7
      3.3.1 GUI output .................................................. 9

4 A real-life example. Choosing the number of factors 10

5 Another real-life example 12

6 List of public functions (in alphabetical order) 15

7 Changelog 17
1 The model

The models that the DFM package can handle can be written in state-space representation as

\[
\begin{align*}
    x_t & = \Lambda_0 f_t + \Lambda_1 f_{t-1} + \cdots + \Lambda_s f_{t-s} + e_t \\
    f_t & = A_1 f_{t-1} + A_2 f_{t-2} + \cdots + A_p f_{t-p} + u_t
\end{align*}
\]

where \( x_t \) is a vector of \( N \) standardised observable variables and \( f_t \) is the \( q \)-element vector of (unobserved) common dynamic factor; the shocks to the observation equation [1], \( e_t \), are known as the idiosyncratic component, and are assumed to be uncorrelated with \( f_t \) at all leads and lags. It is assumed that the elements of \( e_t \) are weakly correlated either cross-sectionally and serially, so that the factors \( f_t \) summarize the important cross-covariance properties of the variables. Both processes \( f_t \) and \( e_t \) are assumed to be second-order stationary.

The key characteristic of this setup is that typically \( N \) can be rather large (up to several hundreds) and \( q \) is much smaller. We use \( R \) to indicate the \( N \times N \) covariance matrix of \( e_t \) and \( Q \) for the \( q \times q \) covariance matrix of \( u_t \), the vector of dynamic factor shocks; the two error vectors \( e_t \), \( u_t \) are assumed independent.

A finite order VAR(\( p \)) model is used to approximate the dynamics of the latent factors \( f_t \), with \( A_1, \ldots, A_p \) the \( q \times q \) matrices of autoregressive coefficients. Matrices \( \Lambda_j \) for \( j = 0, \ldots, s \) contain the dynamic factor loadings. The term \( \chi_t = \Lambda_0 \cdot f_t + \cdots + \Lambda_s \cdot f_{t-s} \) is usually referred to as the "common component", which reduces to \( \Lambda_0 \cdot f_t \) in the static case \( s = 0 \).

The dynamic factor model of [1] and [2] can be recast into a static state space representation, that facilitates empirical estimation, by redefining the state vector. Let \( k = \max \{ s + 1, p \} \) and define the state vector as \( F_t = (f_t', \cdots, f_t'_{t-k+1})' \). When \( k > p \), set \( A_{p+1} = \cdots = A_k = 0 \) in the companion matrix

\[
A = \begin{bmatrix}
    A_1 & \cdots & A_p & \cdots & A_k \\
    I_q & 0_q & \cdots & 0_q \\
    0_q & \ddots & \ddots & \ddots \\
    \vdots & \ddots & \ddots & \ddots & 0_q \\
    0_q & 0_q & I_q & 0_q
\end{bmatrix}
\]

When \( s + 1 < k \), set \( \Lambda_{s+1} = \cdots = \Lambda_k = 0 \) in the observation (loadings) matrix \( \Lambda = (\Lambda_0 \Lambda_1 \cdots \Lambda_k) \). Then the static factor form of the state-space model is given by the following pair of equations

\[
\begin{align*}
    x_t & = A F_t + e_t \\
    F_t & = A F_{t-1} + u_t^* \tag{4}
\end{align*}
\]

with \( u_t^* = (u_t', 0_{q \times 1}', \cdots, 0_{q \times 1}')' \).
2 The estimators

2.1 Principal components (PC)

This technique is well known and needs no detailed description. This estimator makes most sense if the model (1) – (2) is in fact static, that is \( s = p = 0 \); the \( qk \) factors are simply obtained by storing eigenvectors of the correlation matrix of \( x_t \) corresponding to the \( qk \) largest eigenvalues into a matrix \( \hat{\Lambda}_{PC} \) and then computing the factors as \( \hat{F}_{PC,t} = \hat{\Lambda}_{PC}' x_t \). A full account can be found, for example, in Bai and Ng (2008b).

PC estimation assumes \( p = 0 \) so that \( k = s + 1 \) and the dimension of \( \hat{F}_{PC,t} \) (the number of “static factors”) is \( q(s + 1) \). If we impose \( s = 0 \), practically there are no dynamic factors or the number of static factors equals the number of dynamic factors. If we allow \( s > 0 \) then the PC method delivers the \( q(s + 1) \) static factors in \( \hat{F}_{PC,t} \) as linear combinations of current and lagged values of the dynamic factors \( f_t \).

Note that gretl natively provides the principal components technique, via the \texttt{pca} command (along with the corresponding menu interface) and the \texttt{princomp} function. See the Gretl Command Reference for further details. A more in-depth static factor analysis can be performed using the \texttt{staticfactor} package.\(^1\) Note, however, that \texttt{staticfactor} uses a different normalisation convention for the extracted factors; thus, if you compute principal components by the two packages, you’ll obtain identical series up to a proportionality constant.

2.2 Two-step estimator (TS)

This estimator applies to dynamic models and was put forward in Doz, Giannone and Reichlin (2011) who show consistency (for the factors) of the two-step estimator in dynamic approximate\(^2\) factor models as the number of cross sections \( N \) and time periods \( T \) go to infinity.

In the first step, \( \Lambda \) and \( F_t \) are estimated using principal components to obtain \( \hat{\Lambda}_{PC} \) and \( \hat{F}_{PC,t} \). When \( s > 0 \), \( \hat{F}_{PC,t} \) is composed by estimated linear transformations of \( f_t \). Therefore, we proceed with an additional PC estimation to determine an initial estimate of the \( q \) linearly independent factors \( f_t \). Let \( \hat{V} \) denote the matrix of eigenvectors corresponding to the \( q \) largest eigenvalues of the residual covariance matrix obtained by regressing \( \hat{F}_{PC,t} \) on its lag. Then, the initial estimate of the dynamic factors is obtained by \( \hat{V}' \hat{F}_{PC,t} \). The remaining model parameters (\( R, A, Q \)) are estimated using multivariate least squares formulas.

In the second step, let \( \hat{\theta} = \{ \hat{\Lambda}_{PC}, \hat{A}, \hat{R}, \hat{Q} \} \cup \{ \hat{F}_{1|0}, \hat{P}_{1|0} \} \) where the initial state vector value \( F_{1|0} \) is set equal to \( \hat{F}_{PC,1} \) and \( P_{1|0} \) is handled automatically by gretl using standard state space initialisation formulas. Factor estimates \( \hat{F}_{PC,t} \) are updated via the smoothing algorithm of the Kalman filter implemented in the DFM in (6) – (7) to produce \( \hat{F}_{TS,t} \) given \( \hat{\theta} \). The first \( q \)-element sub-vector \( \hat{f}_{TS,t} \) of \( \hat{F}_{TS,t} \) is the TS estimate of the dynamic factors.

\(^1\)To access all the packages available from the gretl server via the gretl menu follow “File, Function packages, On server.”

\(^2\)The term “approximate” here is standard in this literature and refers to the possibility that the model (1) – (2) may be just an approximation to a more general model in which \( f_t \) and \( e_t \) are generic stationary processes. See for example Doz et al. (2012), page 1015.
2.3 Quasi ML - EM estimator (ML)

The Quasi ML estimator was developed by Doz, Giannone and Reichlin (2012) by iterating the two-step Doz et al. (2011) estimator described above; it can be proven that iteration is equivalent to the application of the EM algorithm, along the lines described in Ghahramani and Hinton (1996). Doz et al. (2012) show that the QML-EM estimator of \( f_t \) in (1)–(2) produces consistent factor estimates converging to their true value at a rate equal to \( \min \{ pT, N \ln N \} \). The computational complexity of the Kalman smoother depends essentially on the number of common factors, which is typically small.

A summary description of the algorithm follows: first, an “extended” state space model is defined, with one extra lag than the representation

\[
x_t = \Lambda_0 f_t + \Lambda_1 f_{t-1} + \cdots + \Lambda_s f_{t-s} + [0] f_{t-s-1} + e_t
\]

(5)

where \([0]\) indicates a matrix full of zeros. The static representation of this system then becomes

\[
x_t = \Lambda G_t + e_t
\]

(6)

\[
G_t = A G_{t-1} + \tilde{u}_t
\]

(7)

where \(G_t = (F_t', f_{t-k}', \cdots, f_{t-k+1}', f_{t-k}')'\) and the system matrices are redefined accordingly. This modification of the state space representation is necessary to make it possible to compute rapidly and efficiently all the quantities that are required in the M-step.

Therefore, give a preliminary estimate \(\hat{\theta}\), the “extended” system matrices are computed and the Kalman smoother is run, so as to get an unbiased predictor of the factors \(\hat{G}_t\) with the associated covariance matrices. This is the E-step. Subsequently, the estimated factors are used to compute the sufficient statistics

\[
\hat{F}_t = E(F_t | X)
\]

\[
\hat{P}_{0t} = E(F_t F_t' | X)
\]

\[
\hat{P}_{1t} = E(F_t F_{t-1} | X)
\]

where \(X = x_1, \cdots, x_T\), which are used in the M-step to compute (conditional) ML estimates of \(\theta\). Note that, given the covariance matrix of \(G_t\), the calculation of \(\hat{P}_{0t}\) and \(\hat{P}_{1t}\) simply amounts to selecting appropriate submatrices. With the new estimate of \(\theta\) in hand, the process is repeated until convergence.

We control convergence of the EM algorithm by using a stopping rule based on either a likelihood distance criterion \(c^L\) and how small is the increase in log-likelihood between two consecutive steps (Doz et al., 2012, p.1018) or a parameter distance criterion, \(c^P\). In the first case,

\[
c^L_j = \frac{|\mathcal{L} \{ X; \hat{\theta}^{(j)} \} - \mathcal{L} \{ X; \hat{\theta}^{(j-1)} \}|}{(|\mathcal{L} \{ X; \hat{\theta}^{(j)} \}| + |\mathcal{L} \{ X; \hat{\theta}^{(j-1)} \}| + \epsilon)/2}
\]

(8)

with \(\epsilon = 2.2204460e−016\) the machine epsilon for rounding errors while, in the second case, the parameter distance criterion sums the absolute deviation across all estimated parameters \(\hat{\theta}\) between step \(j\) and \(j-1\):

\[
c^P_j = \frac{\sum_{i=1}^{h} |\hat{\theta}_i^{(j)} - \hat{\theta}_i^{(j-1)}|}{h}
\]

(9)
where \( h \) is the number of elements in \( \hat{\theta} \).

The EM iterations \( j = 1, \ldots, M \) continue until the chosen criterion is smaller than a preset tolerance. Typically, we stop after \( M \) iterations if \( c_M^L < 10^{-4} \) or \( c_M^P < 10^{-2} \). The first convergence criterion is fast and useful for forecasting purposes while the second is safest to compute LR statistics on parameter restrictions.

### 2.4 Parameter restrictions

Future versions of the package will include parameter restrictions necessary for identification and structural interpretation of the dynamic factor model.

### 3 The \texttt{DFM} function package

#### 3.1 Installation

The \texttt{DFM} function package can be downloaded and installed like any other \texttt{gretl} package.

If you use the GUI, in the main window, go to File > Function packages > On server... heading. Select and install \texttt{DFM}. Alternatively, in command-line mode, installation is performed by invoking the command

```
pkg install DFM
```

You will get the option of inserting an item into the \texttt{gretl} menu, under Model > Time series > Multivariate.

#### 3.2 By scripting

The standard way to use \texttt{DFM} via scripting involves first setting up the model, then proceeding to estimation; after that, the estimated factors can be retrieved. In order to perform these three steps, \texttt{DFM} provides three functions (see section 6 for a full description):

- \texttt{DFM_Setup()} : this function takes as parameters the basic information on the model and returns a bundle;
- \texttt{DFM_Estimate()} : this function performs estimation with the chosen method and fills the bundle with the results;
- \texttt{DFM_GetFactors()} : this function extracts the factors from the bundle to a list. The list arguments (factor estimates) are named after the estimation method: \texttt{PC_*} (for PC factor estimates), \texttt{TS_*} (for TS factor estimates), \texttt{ML_*} (for PC factor estimates)

Upon successful completion of the \texttt{DFM_Estimate()} function, the bundle created by \texttt{DFM_Setup()} will contain the following items:

- \( \mathbf{X} \): a \( T \times N \) matrix with the data;
- \( \text{valid} \): a series with 0 at observations for which at least one series in the data processed list has a missing value, and 1 otherwise;
- \( \text{cthres} \): convergence threshold (a scalar);
• **dims**: a bundle (see below);
• **Xnames**: variable names, as an array of strings;
• **results**: a bundle (see below);
• **cConvCrit** 1 or 2, choice for the convergence criterion;
• **itermax**: maximum number of iterations before aborting;
• **cVerbose** verbosity, scalar

The bundle **dims** contains several scalars holding the dimension of items in the system:

• **nDynFact**: number of dynamic factors \( q \);
• **StSpDim**: \( k = \max(s + 1, p) \);
• **FactLags**: \( s \), number of lags in the observation equation [1];
• **FactVAROrder**: \( p \), number of lags in the state transition equation [2];
• **nObs**: number of observations;
• **nStaFact**: number of static factors \( q(s + 1) \);
• **nSeries**: \( N \), number of series;

The bundle **results** holds the results from estimation:

• **method**: estimation method;
• **PC_Fac, TS_Fac, ML_Fac**: matrices holding the estimated factors (if available, given the **method** key). The convenience function **DFM_GetFactors()** can be used for converting these into a list of series;
• **Converged**: Boolean;
• **Lambda, A, M, P, Q, R**: estimated system matrices (see equations [1] – [2]);
• **Resid**: matrix holding the idiosyncratic residuals;
• **SSIC**: scalar, total residual sum of squares divided by \( NT \);
• **conviter**: scalars, iterations to convergence;
• **loglik**: scalar

The following is a minimal example:

```plaintext
set verbose off
include DFM.gfn
open AWM17.gdt --quiet

# create a list DXlist with a few macro
# series (transformed for stationarity)
list Xlist = PCR GCR ITR XTR MTR LFN LNN
list DXlist = ldiff(Xlist) diff(ldiff(YED)) diff(STN) diff(LTN)
```

6
# The important part

```plaintext
Mod = DFM_Setup(DXlist, 1, 2, 3)  # set up the model, q=1 , s=2 , p=3
DFM_Estimate(&Mod, 3)            # perform (QML-EM) estimation
list F = DFM_GetFactors(&Mod)    # save the estimated factors

# now display a few results

DFM_Printout(&Mod, 2)
gnuplot PC_01 TS_01 ML_01 --time-series --with-lines --output=display
```

In this example, we specify a model with \( N = 10 \) series (transformed so as to achieve stationarity) and \( q = 1 \) dynamic factor. The factor enters the observation equation with \( s = 2 \) lags and is modelled as a VAR(3) (in fact, an AR(3), being scalar) process. The model is then estimated via QML (method = 3) and the factors are extracted to the list F. The naming convention we adopt is prepend the string PC to factors extracted by principal components and TS and ML to factors extracted by the two-step and the EM procedures, respectively (if any).

![Figure 1: Plot from example script](image)

The function `DFM_Printout` is used to display the results (reproduced in Table 1). Its second parameter controls the level of detail of the output.

Finally, a combined time series plot of the factors produced by all methods is created. Note that PC returns three static factors since \( q(s+1) = 1(2+1) = 3 \) and we plot the first or dominant PC factor along with the single dynamic factor produced by the two-step and QML-EM methods (see Figure 1).

For further experimentation, replace

```plaintext
Mod = DFM_Setup(DXlist, 1, 2, 3)  # set up the model, q=1 , s=2 , p=3
```

with

```plaintext
Mod = DFM_Setup(DXlist, 1, 2, 3,,0.00001,1)
```

to decrease the convergence threshold of the ML distance criterion (1e-005 instead of the default value of 1e-004 or with

```plaintext
Mod = DFM_Setup(DXlist, 1, 2, 3, ,0.01,2)
```

to change the ML distance criterion with the parameter distance criterion and increase the convergence threshold to 0.01 (it will take 187 iterations to converge).
Number of series = 10, number of observations = 186
Number of factors = 1; factors are a VAR(3)
Number of lags in the observation equation = 2

method: ML via EM (26 iterations)
Log-likelihood = -2261.92
Total SSR/WT: 0.62660

Loadings

<table>
<thead>
<tr>
<th></th>
<th>ld_PCR</th>
<th>ld_GCR</th>
<th>ld_ITR</th>
<th>ld_XTR</th>
<th>ld_MTR</th>
<th>ld_LFN</th>
<th>ld_LNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.668</td>
<td>-0.916</td>
<td>-0.183</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.070</td>
<td>-0.746</td>
<td>0.935</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.440</td>
<td>-0.337</td>
<td>-0.903</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3.455</td>
<td>-0.160</td>
<td>-2.113</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4.106</td>
<td>-0.268</td>
<td>-2.047</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.824</td>
<td>0.590</td>
<td>0.926</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.610</td>
<td>1.281</td>
<td>0.538</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.053</td>
<td>0.573</td>
<td>0.216</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.933</td>
<td>1.234</td>
<td>0.430</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.437</td>
<td>0.474</td>
<td>0.031</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Idiosyncratic s.e.

<table>
<thead>
<tr>
<th></th>
<th>ld_PCR</th>
<th>ld_GCR</th>
<th>ld_ITR</th>
<th>ld_XTR</th>
<th>ld_MTR</th>
<th>ld_LFN</th>
<th>ld_LNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.81403</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.97982</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.62448</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.70613</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.82493</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.83815</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.39770</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.98783</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.86330</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.96409</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VAR parameters

| 0.587| 0.471| -0.273|

Eigenvalues of the companion matrix

-0.6941
0.7096
0.5618

Factor innovations covariance matrix

0.031

Table 1: Output of example script
3.3 Using the GUI

By invoking DFM through the graphical interface (go to Model, Time series, Multivariate, Dynamic Factor Models), a window similar to the one shown in Figure 2 will open.

![Figure 2: GUI interface to DFM](Image)

The arguments to select involve:

**Series to process (list)**: a list with the series to be processed. Pre-processing for stationarity is up to the user. Conversely, standardisation is performed internally by the function;

**No. of dynamic factors**: an integer larger or equal to 1 (default = 1). Set the number of dynamic factors to $q$ in equations (1)–(2);

**No. of lags of the dynamic factors**: an integer larger or equal to 0 (default = 0). Set the number of lags $s$ for the dynamic factors in the observation equation (1);

**length of VAR filter on common factors**: an integer larger or equal to 1 (default = 1). Set the number of lags $p$ in the dynamic factors VAR filter; equation (2);

**method**: an integer 1:3, (default = 3) to set the estimation method, 1: "Principal components", 2: "two-step" method, 3: "ML via EM". Choosing the "two-step" method also produces PC estimates of the factors. Choosing "ML via EM" also produces PC and two-step estimates.

**Verbose?**: controls output verbosity

3.3.1 GUI output

Upon successful convergence, an output window should appear, showing the same statistics as in Table 1 unless the "Verbose" flag is unticked, in which case output just includes minimal information, like in Figure 3 a model bundle is also created and can be saved to the session as an icon (for example under the name Mod). This
can be done by clicking on the leftmost icon on top of the output window. The next icon will give you a drop-down list of items that you can save, including the estimated factors. 

![Minimal output from the GUI interface](image)

In addition, three graphical tools are available:

1. a time-series plot of the estimated factors;
2. a scree plot of the eigenvalues from principal components extraction (marginal contribution of each factor to total variance);
3. a correlation heat map, displaying the contemporaneous correlations between all series in $x_t$ and estimated dynamic factors, $\hat{f}_t$.

These three plots are available by clicking the “Plot” icon on top of the output window.

## 4 A real-life example. Choosing the number of factors

The determination of the number of factors is crucial both for structural analysis and for forecasting purposes. Bai and Ng (2002) provide information criteria methods that can consistently estimate the number of static factors, $K = q(s + 1)$, in approximate DFMs (assuming a large $N$). Bai and Ng (2007) propose four criteria to consistently estimate the number of dynamic factors $q$ given a pre-selected or estimated number of static factors $K$.

Gretl’s static factor package `staticfactor` can be employed to compute an estimate $\hat{K}$ based on the minimization of a penalized sum of squares criterion

$$\hat{K} = \arg\min_{K=1, \ldots, K_{\max}} IC_{p2}(K)$$

while the DFM package contains the function `DFM_BNCrit()`, which provides the Bai and Ng (2007) covariance and correlation based criteria to select the number of dynamic factors.

As an illustration, we employ the Stock and Watson (2005) dataset which contains 132 monthly time series (mostly) available from 1959:1 to 2003:12. The objective is to determine the number of primitive, or dynamic, factors in this panel of data.

---

3Note that the graphical aspect of the icon depends on a variety of factors (which operating system you’re using, plus others), so that what you see in Figure 3 may not coincide exactly with what you get on your computer.

4The original (raw) dataset `sim.xls` is contained in a replication material file of the working papers section at [http://www.princeton.edu/~mwatson](http://www.princeton.edu/~mwatson)
set verbose off
include staticfactor.gfn
include DFM.inp

# Inputs
open sw2005datat.gdtb --frompkg=DFM --quiet
scalar thr = 6 # Threshold multiple for IQR
scalar len = 5 # replace outliers with one-sided median (5 preceding obs)
list xlistC = DFM_PrepareData(dataset, 1, {thr, len}, 1)

### ----------------------------------------------------------------------
### Choose number of static factors
### ----------------------------------------------------------------------
# use the "staticfactor" package
btmpINI = staticfactor(xlistC,,1,0,0)
K = btmpINI.numfac[3]
string critname = cnameget(btmpINI.numfac,3)
printf "\n%s (2002, ICp2) criterion selects %d static factors\n", critname, K
delete btmpINI

### ----------------------------------------------------------------------
### Choose number of dynamic factors
### ----------------------------------------------------------------------
VAR_order = 2
q = DFM_BNCrit(xlistC, K, VAR_order, 1)
scalar optimal = maxr(q) # We choose the maximal order
printf "\nBaiNg (2007) dynamic factors (maximal order): %d\n", optimal

### ----------------------------------------------------------------------
### Estimate the model
### ----------------------------------------------------------------------
Mod = DFM_Setup(xlistC, optimal, 1, VAR_order) # Set up the model
DFM_Estimate(&Mod, 3) # perform two-step estimation
list F = DFM_GetFactors(&Mod) # save the estimated factors

# Now display a few results
DFM_Printout(&Mod, 1)

Table 2: Stock-Watson dataset: selecting the number of dynamic factors
Following the Stock and Watson (2005) proposed transformations: (i): levels or logs, (ii): to achieve stationarity either no transformation, first or second differences and (iii): an outlier “correction” on each series, we end up with a balanced panel of monthly time series available from 1960:1 to 2003:12 for a total of 528 observations. This is the dataset sometimes referred to as the Stock-Watson dataset, which has been widely employed in the literature: see inter alia Bai and Ng (2008a, 2013) and McCracken and Ng (2016).

The script BaiNg-example, included in the examples directory for the package, was employed to estimate the number of dynamic factors using the Stock-Watson dataset.

Note that exact replication of results entails a procedure for outlier removal, for which we use the dedicated function DFM_PrepareData() (see section 6 for details on its syntax): datafile sw2005datat.gdtb is loaded that contains the stationarity transformed series of the Stock-Watson raw data, spans the period 1960:1 to 2003:12 and preserves the original series names. As suggested by Stock and Watson (2005), outliers are replaced by the one-sided median (5 preceding observations).

After this preliminary step, the number of static factors and the number of dynamic factors is estimated (see Table 2). The script ends by performing two-step estimation using the DFM package. Part of the output (relevant to the estimation of the number of factors) is shown below:

BaiNg (2002, I(q2)) criterion selects: 7 static factors

<table>
<thead>
<tr>
<th>Number of static factors: 7, Number of dynamic factors: q</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(1,k)</td>
</tr>
<tr>
<td>k=0 :</td>
</tr>
<tr>
<td>k=1 :</td>
</tr>
<tr>
<td>k=2 :</td>
</tr>
<tr>
<td>k=3 :</td>
</tr>
<tr>
<td>k=4 :</td>
</tr>
<tr>
<td>k=5 :</td>
</tr>
<tr>
<td>k=6 :</td>
</tr>
</tbody>
</table>

Suggested q value:

<table>
<thead>
<tr>
<th>D(1,k)</th>
<th>D(2,k)</th>
<th>Dc(1,k)</th>
<th>Dc(2,k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Bai & Ng (2007). Determining the Number of Primitive Shocks in Factor Models. JBES, Vol. 25, No. 1

Another real-life example

We follow the empirical application of Fiorentini, Galesi and Sentana (2018) to construct a common component index that captures the aggregate dynamics of monthly U.S sectoral employment growth rates and explains the bulk of the time variation of the different sectors. The DFMdataset.gdt contains 82 series: total nonfarm employment growth and the 81 sectoral growth rates employed by Fiorentini et al. (2018) for the period 1990M2-2014M4.

In view of their empirical model specification (we do not allow for moving average terms in the state equation), we estimated model (1)–(2) with $q = 1$ (one dy-
namic factor), we set $s = 2$ so that $f_t, f_{t-1}$ and $f_{t-2}$ heterogeneously affect each of the sectoral growth rates while $f_t$ follows an AR(4).

Given our objective to capture sectoral variation, we focus on the common variation which accounts for the largest share of the variance of the sectoral growth rates. So the smoothed component that appears in Fig. 4 is constructed as the first principal component of the projection of the sectoral growth rates onto the common (contemporaneous and lagged) factors.

The script in Table 3 (also included in the examples directory for the package) was used to estimate the dynamic factor model and to produce Figure 4. The output is as follows:

```
Number of series = 81, number of observations = 291
Number of factors = 1; factors are a VAR(4)
Number of lags in the observation equation = 2
-----------------------------------------------------------------------
method: ML via EM (16 iterations)
Log-likelihood = -29778.7
Total SSR/NT: 0.75505
```
set verbose off
include DFM.gfn

open Labor.gdt --frompkg=DFM --quiet
list xlist = dataset - CES00000000001

Mod = DFM_Setup(xlist, 1, 2, 4) # q=1 , s=2 , p=4

set stopwatch
DFM_Estimate(&Mod, 3)
printf "\n\nElapsed time: %g seconds\n", $stopwatch
DFM_Printout(&Mod, 1)

# Save estimation results in a bundle held in Mod
results = Mod.results
Lambda = results.Lambda # matrix: store factor loadings
ML_Fac = results.ML_Fac # matrix: store factors f(t),...,f(t+s)

# Create the smoothed common component CCt
matrix mVec = {}
matrix mVac = eigensym( qform( Lambda , mcov(ML_Fac) ) , &mVec )
matrix mFhat = ML_Fac*(Lambda')*mVec[,cols(mVec)]
m = mean(CES00000000001)
s = sd(CES00000000001)
series CCt = m + (cdemean(mFhat)/sdc(mFhat,rows(mFhat)-1)) * s

list plotList = CCt CES00000000001
setinfo CCt --graph-name="Smoothed employment component"
setinfo CES00000000001 --graph-name="Total nonfarm employment"

plot plotList
options time-series with-lines
literal unset xzeroaxis
literal set grid ytics back lt 1 dt 2 lw 0.5 lc rgb "#808080"
literal set key left bottom
literal set ylabel "Percent change"

end plot --output=display

| Table 3: Example for the US labour market |   |
6 List of public functions (in alphabetical order)

DFM_BNCrit(list X, scalar K, scalar p, scalar bprnt)

**Return type**: matrix

xlist : a list with the observable variables;
K : number of assumed static factors;
p : length of VAR filter on common factors p;
Bprnt : Boolean, print details (optional; default: no)

This function calculates the four information criteria for determining the number of dynamic factors put forward in Bai and Ng (2007) (see section 4). It returns a 1 × 4 matrix with the suggested number of dynamic factors according to the four criteria.

The argument K contains the (assumed) number of static factors, that has to be determined beforehand.

DFM_GetFactors(bundle *mod)

**Return type**: list

*mod : pointer to the model bundle

Returns a list with the estimated factors. The list series are named PC_* (principal components), TS_* (two-step) and ML_* (QML-EM estimation)

DFM_Estimate(bundle *mod, int method)

**Return type**: scalar

*mod : pointer to the model bundle created by the DFM_Setup function;

Returns 0 upon successful completion, non-zero otherwise. It adds a bundle named results in the model bundle *Mod. See section 3.3.1 for details

DFM_PrepareData(list xlist, int transform, matrix outlier_corr, bool verbose)
**Return type**: list

xlist : input list of series;

transform : integer, type of transformation: 0 = none, 1 = center, 2 = standardize (default = 1);

outlier_corr : matrix for outlier correction (see below);

verbose : Boolean: print a report of the outlier correction procedure; default = 1)

Returns a list with a suitable transformation of the variables in xlist. The parameter transform controls for the type of linear transformation of the original series.

The argument outlier_corr can be used to implement an outlier correction procedure of the kind suggested in [Stock and Watson, 2005], which is based on two scalars: a “threshold” \( \tau \) used to mark an observation as an outlier and a “length” \( \ell \) to calculate the correction. An observation \( x_t \) is considered an outlier if

\[
|x_t - m| > R \cdot \tau
\]

where \( m \) is the median of \( x_t \) and \( R \) is its interquartile range. If \( x_t \) is an outlier, it is replaced by the median of the \( \ell \) preceding observations, that is the median of \( x_{t-1}, \ldots, x_{t-\ell} \).

If the argument outlier_corr is skipped or null, the correction is not performed. Otherwise, outlier_corr is understood to be a vector with 2 elements, the first one being \( \tau \) and the second one being \( \ell \). If an empty matrix is passed, the default choices \( \tau = 6, \ell = 5 \) are applied.

DFM_Printout (bundle *mod, int verbose)

**Return type**: void

*mod : pointer to the model bundle;

verbose : an integer type. 0: just print the model dimensions; 1: also print the log-likelihood, EM iterations to convergence and a total residual variance estimate; 2: in addition, print some estimated system matrices (optional; default = 1)

Prints estimation related results.

DFM_Setup (list xlist, int cq, int cs, int cP, int itermax, scalar cthres, int cConvCrit, bool cVerbose)

**Return type**: bundle

xlist : a list with the observable variables; the vector of observables \( x_t \) in equation is a standardized version of the original data contained in xlist;
cq : number of dynamic factors \( q \) (optional; default: 1);

cs : number of lags of the dynamic factors, \( s \) in equation 1 (optional; default: 0);

cP : length of VAR filter on common factors \( p \) (optional; default: 1);

\( \text{itermax} \) : maximum number of EM iterations for QML estimation (optional; default: 5000);

cConvCrit : choice of convergence criterion, 1 for ML distance or 2 for parameter distance as in equations 8 and 9 (optional; default: 1);

cthres : convergence threshold for QML estimation (optional; default: 0.0001 for the ML distance criterion); Depending on the application (and identification of the model that is not handled at the moment), if the parameter distance criterion is employed then the threshold should be set at (or less than) 0.01;

cVerbose : a Boolean type controlling the degree of output verbosity. 0 = no output is printed, 1 = the log-likelihood and parameter distance as in equation 9 are printed at each EM iteration (optional; default: yes)

This function returns the initialised model bundle, for which the preliminary steps are taken: sub-sampling the data as needed, building and standardising the data matrix, handling the default settings.

Note that the function can handle missing values in the series listed in \( xlist \) only at the beginning and end of the sample.

### 7 Changelog

**v 0.1** Initial release

**v 0.2** Bump version requirement; fix a few statements in the example scripts; add some info to the printout of companion eigenvalues; make it possible to save the factors from the GUI.

**v 0.3** Fix a few typos in the pdf help; add scree plot and fix the heatmap; simplify sample script.

**v 0.4** Improve efficiency of EM algorithm (and improve documentation); introduce the \texttt{DFM_PrepareData} function; changed convergence criterion from sum to average.

**v 0.41** Bugfix release; TS and ML estimation with one factor only were not handled properly; handle the cases when \( n < cq \) and when the preliminary estimate of \( A \) in equation 7 contain explosive roots.

**v 0.42** Bugfix release; handle convergence failure in the \texttt{DFM_Estimate} function more gracefully (ie return a proper error message).

**v 0.43** No user-visible changes. The internal identifier \texttt{stdize} was changed into \texttt{transform} because the former has become a reserved word.
References


