A Hitchhiker Guide to Empirical Macro Models[1]

Filippo Ferroni (Chicago Fed)
Fabio Canova (Norwegian Business School, CAMP and CEPR)

WGEM
September 18, 2020

[1] The views in this paper are solely the responsibility of the authors and should not be interpreted as reflecting the views of the Federal Reserve Bank of Chicago or any other person associated with the Federal Reserve System.

Outline

Introduction

- The Empirical Macro Toolbox is a package which uses MATLAB/Octave functions and routines to estimate a number of reduced form macro models with classical or Bayesian methods.

Introduction

- The Empirical Macro Toolbox is a package which uses MATLAB/Octave functions and routines to estimate a number of reduced form macro models with classical or Bayesian methods.
- Source codes with examples can be forked/downloaded at https://github.com/naffe15/BVAR_

Introduction

- The Empirical Macro Toolbox is a package which uses MATLAB/Octave functions and routines to estimate a number of reduced form macro models with classical or Bayesian methods.
- Source codes with examples can be forked/downloaded at https://github.com/naffe15/BVAR_
- The hitchhiker guide can be downloaded here https://www.filippoferroni.com/empiricalmacrotoolbox or https://sites.google.com/view/fabio-canova-homepage/home/empirical-macro-toolbox

Bird's eye view of the toolbox

- Recursive methods (VAR):
  Classic and Bayesian Inference, Point/density Forecasts, IRF (many identifications), historical and variance decomposition, missing values/mixed frequency, panels of VAR.

Bird's eye view of the toolbox

- Recursive methods (VAR):
  Classic and Bayesian Inference, Point/density Forecasts, IRF (many identifications), historical and variance decomposition, missing values/mixed frequency, panels of VAR.

- Direct methods (LP):
  Classic and Bayesian Inference, Forecasts, IRF (cholesky, IV)

Bird's eye view of the toolbox

- Recursive methods (VAR):
  Classic and Bayesian Inference, Point/density Forecasts, IRF (many identifications), historical and variance decomposition, missing values/mixed frequency, panels of VAR.

- Direct methods (LP):
  Classic and Bayesian Inference, Forecasts, IRF (cholesky, IV)

- Compression methods (FAVAR):
  Static and Dynamic (TBA) FAVAR

Bird's eye view of the toolbox

- Recursive methods (VAR):
  Classic and Bayesian Inference, Point/density Forecasts, IRF (many identifications), historical and variance decomposition, missing values/mixed frequency, panels of VAR.

- Direct methods (LP):
  Classic and Bayesian Inference, Forecasts, IRF (cholesky, IV)

- Compression methods (FAVAR):
  Static and Dynamic (TBA) FAVAR

- Filtering methods (TBA):
  Extracting trend/cycles and gaps, Dating BC.

Road map for today: VAR

Topics (use examples in the tutorial as blueprints):

I Bayesian VAR inference: priors and posteriors. Uninformative and Minnesota priors. How to chose the Minnesota hyper parameters.

open `YOUR_LOCATION/BVAR_/v4.1/'BVAR tutorial'/example_0_minn.m`

II IRFs with various identification schemes. Historical and Variance Decomposition.

open `YOUR_LOCATION/BVAR_/v4.1/'BVAR tutorial'/example_1_irf.m`

III Nowcasts and Mixed-Frequency estimation.

open `YOUR_LOCATION/BVAR_/v4.1/'BVAR tutorial'/example_2_mfvar.m`

IV Point and Density Forecasts. Conditional Forecasts.

open `YOUR_LOCATION/BVAR_/v4.1/'BVAR tutorial'/example_4_prediction.m`

Some notation

- *VAR(p)*: A vector of autoregression with p lags can be expressed as

$$y_t = \Phi_1 y_{t-1} + ... + \Phi_p y_{t-p} + \Phi_0 + u_t \quad u_t \sim N(0, \Sigma) \tag{1}$$

where y_t is $n \times 1$ vector, Φ_j suitable matrices

- Inference: A VAR(p) can be expressed as a SURE

$$Y = X\Phi + E$$

- Forecasts and IRF: Any VAR(p) can be expressed as a VAR(1) (companion form)

$$x_t = Fx_{t-1} + F_0 + Gu_t$$

- Decompositions: Any VAR model can be expressed as a Vector MA (VMA) model:

$$y_t = u_t + \Psi_1 u_{t-1} + ... + \Psi_t u_1 + \overline{\Psi}_t$$

where Ψ_j for $j = 1, ..., t$ are functions of $(\Phi_1, ..., \Phi_p)$ and $\overline{\Psi}_t$ is deterministic.

Outline

Inference

- Estimation can be performed using flat and informative priors. By default, Jeffrey (uninformative) prior is assumed; other options for the prior are available and discussed below.

- Given the prior, draws are generated from the posterior distribution of the parameters using a Gibbs sampler algorithm; see the guide for more details.

- The baseline estimation function is

```
[BVAR] = bvar(y, lags, options)
```

- Inputs: data, lags and options (which can be omitted)
  - `y`, a ($\mathtt{T} \times \mathtt{n}$) array of data (i.e. rows=time columns=var)
  - `lags` > 0.
  - `options` a field structure that specifies various customized options (i.e. priors, horizons, identification schemes)

Outputs

- BVAR a field with many sub-fields. Here's the most important ones.
  - BVAR.Phi_draws is a $(\mathtt{n} \times \mathtt{lags} + 1) \times \mathtt{n} \times \mathtt{K}$ array. Last dimension draws.
    BVAR.Phi_draws(:,:,k) stacks vertically the AR matrix; last row of BVAR.Phi_draws(:,:,k) contains the constant, Φ_0, if it assumed to be present.
    BVAR.Sigma_draws is a $\mathtt{n} \times \mathtt{n} \times \mathtt{K}$ matrix containing K draws from Σ.
    BVAR.e_draws is a $(\mathtt{T} - \mathtt{lags}) \times \mathtt{n} \times \mathtt{K}$ matrix containing K draws of the innovations.
    BVAR.Phi_ols, BVAR.Sigma_ols and BVAR.e_ols are the OLS point estimate analogs.
  - BVAR.ir_draws is a four dimensional object (i.e. $\mathtt{n} \times \mathtt{hor} \times \mathtt{n} \times \mathtt{K}$ matrix) that collects the impulse response functions with recursive identification.
  - BVAR.forecasts.no_shocks (with_shocks) is a three dimensional object (i.e. $\mathtt{fhor} \times \mathtt{n} \times \mathtt{K}$ matrix) that collects the forecasts assuming zero (non-zero) shocks in the future.
  - BVAR.logmlike contains the log marginal likelihood – a measure of fit.
    BVAR.InfoCrit: the Akaike information criterion, AIC, Hannan-Quinn information criterion HQIC and the Bayes information criterion,BIC.
- By default, K=5000 (change with options.K = numb;).
- By default, a constant is assumed (change with options.noconstant = 1;).

Priors

The toolbox allows for three types of priors

- Uninformative or Jeffrey prior (default, no instruction needed).

```
[BVAR] = bvar(y, lags)
```

- Minnesota prior with default hyper-parameter values.

```
options.priors.name = 'Minnesota';
[BVAR] = bvar(y, lags, options)
```

- Multivariate-Normal Inverse-Wishart Conjugate with default (loose) values.

```
options.priors.name = 'Conjugate';
[BVAR] = bvar(y, lags, options)
```

Minnesota prior

- Minnesota prior is controlled by 5 hyper-parameters allowing for different layers of shrinkage (all scalar)
  - τ `options.minn_prior_tau`: overall tightness (default 3). The larger the tighter is prior.
  - d `options.minn_prior_decay`: tightness on the lags greater than one (default 0.5). The larger the faster is the lag decay.
  - λ `options.minn_prior_lambda`: the Sum-of-Coefficient prior (default 5)
  - μ `options.minn_prior_mu`: co-persistence prior (default 2)
  - ω `options.minn_prior_omega`: the covariance matrix (default 2)

- Three ways to activate the Minnesota prior by defining the appropriate `option`
  1. default values:
     ```
     options.priors.name = 'Minnesota'
     ```
  2. customized values:
     ```
     options.minn_prior_tau=5;
     options.minn_prior_lambda=0.01;
     ```
  3. maximized values:
     ```
     options.max_minn_hyper = 1; % default maximization
     ```

- After typing the desired option in the command window, launch the estimation

```
BVAR  = bvar(y,lags,options);
```

Minnesota prior with optimal hyper-param

- Various options can be set for the maximization step:
  - `options.index_est` is a row vector that selects the parameters to be optimized (default, `options.index_est=1:5`)
  - `options.lb (.ub)` set the lower (upper) bound for the optimization. same size of `options.index_est`.
  - `options.max_compute` is a scalar selecting the maximization routine to be employed:
    - `options.max_compute = 1` uses the MATLAB `fminunc.m`
    - `options.max_compute = 2` uses the MATLAB `fmincon.m`
    - `options.max_compute = 3` uses the Chris Sims's `cminwel.m` (default)
    - `options.max_compute = 7` uses the MATLAB `fminsearch.m`
- Once the maximum is found, the posterior distribution is computed using the optimal hyper-parameter values. If unsuccessful, the posterior distribution is computed with default values (warning is issued).
- Tip: max one param at time starting the optimization from the values obtained in the previous step

```
[hyperp,~,~] = bvar_max_hyper(x0,y,lags,options);
```

Conjugate MN-IW

- Multivariate Normal-Inverse Wishart conjugate prior.

$$\texttt{options.priors.name} = \texttt{'Conjugate'}$$

  Default: $\Phi \sim N(0, 10\ I_{np+1})$ and $\Sigma \sim IW(I_n, n+1)$

- `options.priors.Phi.mean` is a $(\texttt{n} \times \texttt{lags} + 1) \times \texttt{n}$ matrix containing the prior means for the autoregressive parameters.

- `options.priors.Phi.cov` is a $(\texttt{n} \times \texttt{lags} + 1) \times (\texttt{n} \times \texttt{lags} + 1)$ matrix containing the prior covariance for the autoregressive parameters.

- `options.priors.Sigma.scale` is a $(\texttt{n} \times \texttt{n})$ matrix containing the prior scale of the covariance of the residuals.
  $\rightarrow$ adjust for variables with different units.

- `options.priors.Sigma.df` is a scalar defining the prior degrees of freedom.

- After typing the desired option in the command window, launch the estimation

```
BVAR  = bvar(y,lags,options);
```

VARX

- VAR with exogenous variables (X) are allowed in the toolbox. Need to specify it as an option

```
options.controls = z;
```

  where `z` is a ($\mathtt{T} \times \mathtt{q}$) matrix containing the exogenous variables

- The first dimension of `z` is time and must coincide with the time dimension of `y` or with the sum of the time dimension of `y` and the out-of-sample forecast horizon

- Lags of exogenous controls can be used and specified as additional columns in `z`.

- A *VARX* model can be estimated in the toolkit assuming either Jeffrey priors or conjugate priors; Minnesota priors in VARs with exogenous are not currently supported.

- After typing the desired option in the command window, launch the estimation

```
BVAR  = bvar(y,lags,options);
```

Outline

IRF

- For each draw of the posterior distribution, the toolbox computes the IRF for a given identification scheme
- IRFs are stored as ($\texttt{n} \times \texttt{hor} \times \texttt{n} \times \texttt{K}$) arrays (variable, horizon, shock, draw)
- Priors: flat, Minnesota, Conjugate MN-IW
- Default horizon 24; change with `options.hor = hor;` before launching `bvar`
- By default Recursive or Cholesky: `BVAR.ir_draws`
  default (no command needed);
- Other identification schemes need to be specified by the user in `options` (see next slides) before launching `bvar`.

Long run

- Long Run:

```
options.long_run_irf = 1;
```

- First variable in the `y` lists is specified in log difference
- First disturbance has long run effects on the first variable.
- Reference: Gali (1999).
- More than one long run shock, as in Fisher(2006), can be estimated by introducing more variables in first difference.
- IRFs are stored in `BVAR.irlr_draws`

Sign and Magnitude

- Sign and Magnitude:

  `options.signs{1} ='y(a,b,c)>0'`

  where the array `'y'` refers to the IRF (variable, horizon, shock).

- The syntax means that the shock `c` has a positive impact on variable `a` at horizon `b`

- Flexible syntax (e.g. lower bound on elasticity or threshold on the cumulative impact)

  `options.signs{2} = 'y(a_1,1,c)/y(a_2,1,c)  >m'`

  `options.signs{3} = 'max(cumsum(y(a,:,c),2))<M'`

- Reference: Rubio-Ramirez, Waggoner and Zha (2010)

- IRFs are stored in `BVAR.irsign_draws`

Narrative

- Narrative restrictions:

```
options.narrative{1} ='v(tau,n)>0'
```

  where the array `'v'` refers to the structural innovation (time, shock).

- The syntax means that shock `n` is positive on the time periods `tau`

- Flexible syntax

```
options.narrative{1} = 'sum(v(tau_0:tau_1),n)>0'
```

  The syntax means that the sum of the shock `n` between periods `tau_0` and `tau_1` is positive.

- Reference: Ben Zeev (2018) and Antolin-Diaz and Rubio-Ramirez (2018)

- IRFs are stored in `BVAR.irnarrsign_draws`

IV

- External, instrumental or proxy variable:

```
options.proxy = instrument;
```

- Instrument cannot be longer than `(T - lags)`
- Instrument ends when the VAR ends (default). When this is not the case,

```
options.proxy_end = periods;
```

  `periods` is the number that separates the last observation of the instrument and the last observation of the VAR innovations.

- Multiple proxy variables are allowed to identify one structural shocks.
- Reference: Miranda-Agrippino and Ricco (2020)
- IRFs are stored in `BVAR.irproxy_draws`
- By convention, the structural shock of interest is ordered **first**; i.e. `BVAR.irproxy_draws(:,:,1,:);`

Mixed identification

- Mix of zero (short and long run) and sign restrictions
- `options.zero_signs{1} ='y(j,k)=+1'`
  shock `k` has a positive effect on the `j`-th variable **on impact**
- `options.zero_signs{2} ='ys(j,k_1)=0'`
  shock `k_1` has a zero impact effect on the `j`-th variable.
- `options.zero_signs{3} ='yl(j,k_2)=0'`
  shock `k_2` has a zero long run effect on the `j`-th variable.
- Reference: Arias, Rubio-Ramirez, Waggoner and Zha (2018) and Binning (2013)
- IRFs are stored in `BVAR.irzerosign_draws`

Plotting IRFs

- Plotting IRF:

```
plot_irfs_(irfs_to_plot, optnsplt)
```

- `irfs_to_plot` is a fourth dimensional array; e.g. `irfs_to_plot = BVAR.ir_draws` or `irfs_to_plot = BVAR.irproxy_draws(:,:,1,:);`

- `optnsplt` is optional and defines various plotting options
    - `optnsplt.varnames (.shocksnames)` is a cell string with variable (shock) names
    - `optnsplt.conf_sig (.conf_sig_2)` is a number between 0 and 1 indicating the size of (second) HPD set to be plotted; the default is 0.68.
    - `optnsplt.saveas_strng (.saveas_dir)` a string array with name of (directory where to save) the plot.
    - `optnsplt.add_irfs` allows to add additional IRF to be plotted when one wants to compare responses.
    - `optnsplt.nplots` is a 1×2 array indicating the structure of the subplots.

Other useful functions

- Forecast error variance decomposition: `FEVD = fevd(hor,Phi,Sigma,Omega);`
  `FEVD` is a `n` $\times$ `n` array; the `(i,j)` element the share of variance of variable `i` explained by shock `j` at horizon `h`. `Omega` is the rotation (omit if Choleski)

- Historical decomposition: `[yDecomp,v] = histdecomp(BVAR,opts);`
  `yDecomp` is a `T` $\times$ `n` $\times$ `n+1` array (time, variable, shock and initial condition $\overline{\Psi}_t$)
  `v` is the `T` $\times$ `n` array of structural innovations.
  `BVAR` is the output of the `bvar.m` function.
  `opts.Omega` declares a different rotation/identification from the default (Choleski)

- Plotting yDecomp: `plot_shcks_dcmp_(yDecomp, BVAR, optnsplt)`
  `optnsplt` is optional and controls various plotting options (see guide)

Additional identifications: Max FEVD

- Given a draw (`Phi, Sigma`) of the reduced form VAR parameters, one can use

```
Qbar  = max_fevd(i, hor, j, Phi, Sigma)
```

  which finds the orthonormal rotation maximizing the forecast error variance decomposition (FEVD) of variable `i` explained by shock `j` at horizon `hor`.

- Given `Q_bar`, IRF can be computed

```
y_fevd = iresponse(Phi, Sigma, hor, Q_bar)
```

- Loop over draws (`Phi,Sigma`)

- Useful functions

```
Q = generateQ(n)
y = iresponse(Phi, Sigma, hor, Q)
```

Outline

Mixed Frequency VAR

- Mixed Frequency VAR (MF-VAR): now-casts, back-cast or retrieving variables that have an arbitrary pattern of missing observations.
- Gibbs sampler:
  1. generate a draw from the posterior distribution of the reduced form VAR parameters, conditional on observables and states.
  2. with this draw run the Kalman smoother and get estimate the unobserved states (monthly GDP).
  3. Repeat 1. and 2.
- Prior: flat, Minnesota (except max), Conjugate MN-IW. IRF: all identification. Forecasts.
- MF-VAR estimation is triggered automatically in bvar.m whenever there are `NaN` in the the data, `y`. (warning is printed).

Monthly-Quarterly Frequency

- Example with quarterly and monthly variables. Time unit is month.

- Pool vars together in $y = [y^m, y^q]$; the q variable is observed in the last month of the q, elsewhere `NaN`. Define the unobserved states as x_t.

- State space where the transition equation is the VAR (x_t) and measurement equation maps x_t into y_t.

  - Monthly variables: $y_t^m = x_{m,t}$

  - **Stock** quarterly variables: $y_t^q = x_{q,t}$. [no instruction is needed]

  - **Flow** quarterly variables (e.g. GDP): $y_t^q = \frac{1}{3}(x_{q,t} + x_{q,t-1} + x_{q,t-2})$

    ```
    options.mf_varindex     = num;
    ```

    where `num` is a scalar indicating the position of the q flow variable (column # in `y`)

- Additional outputs: `BVAR.yfill` (smoothed) and `BVAR.yfilt` (filtered) `T` $\times$ `n` $\times$ `K` arrays

Outline

Forecasts

- `BVAR.forecasts.no_shocks` is a $(\mathtt{nfor} \times \mathtt{n} \times \mathtt{K})$ matrix (time, variable, draw). Future shocks are zeros.

- `BVAR.forecasts.with_shocks` is a $(\mathtt{nfor} \times \mathtt{n} \times \mathtt{K})$ matrix (time, variable, draw). Future shocks are drawn randomly.

- `BVAR.forecasts.conditional` is a $(\mathtt{nfor} \times \mathtt{n} \times \mathtt{K})$ matrix containing the conditional forecasts.
  - `options.endo_index` is a row array containing the index of the variable constrained to a specified path.
  - `options.endo_path` is a matrix containing the path for each variable (rows horizon, column variables). `size(options.endo_path,1)` $=$ `options.fhor`.
  - `options.exo_index` [optional] specifies the shocks of the VAR used to generate the assumed paths of the endogenous variables. `exo_index` could be one or more shocks. If no structural identification is performed, the program uses a Cholesky factorization by default.

- `BVAR.forecasts.EPS` contains the shocks used to generate the conditional forecasts.

- Use `options.fhor` to change the forecasting horizon

Plotting the Forecasts

- The plot command for forecast is:

  ```
  plot_frcst_(frcst_to_plot,y,T,optnsplt)
  ```

  `options` can be omitted

- `frcst_to_plot` is a three dimensional array (time,variable,draw) containing the forecast. E.g. `frcst_to_plot = BVAR.forecasts.no_shocks`.

- `y` is the ($\mathtt{T} \times \mathtt{n}$) array of data

- `T` is the ($\mathtt{T} \times 1$) array with the in-sample time span.
  E.g. for quarterly data, `T= 1990 : 0.25 : 2010.25` coincides with the sample span 1990Q1 to 2010Q2.

- `optnsplt` defines a number of options. Similar to the ones of `plot_irf_`

- `options.order_transform` is a $1 \times \mathtt{n}$ array with values
  - =0 $\rightarrow$ no transformation
  - =1 $\rightarrow$ period-by-period change
  - =12 $\rightarrow$ 12 period change multiplied by 100, i.e. $100(y_{t+12} - y_t)$.
  - =4 $\rightarrow$ 4 period change multiplied by 100, i.e. $100(y_{t+3} - y_t)$.
  - =100 $\rightarrow$ period over period change multiplied by 100.

Forecasting in times of pandemics (preliminary)

- The elephant in the room: are VAR models useful now?
- Schorfheide and Song (2020): MF-VAR to generate real-time macroeconomic forecasts for the U.S. during the COVID-19 pandemic. MF-VAR outlook is quite pessimistic. Long-lasting recession.
- Primiceri and Lenza (2020): Heteroskedasticity adjusted VAR. Scale down the observables during the peak of the COVID-19 pandemic. Seems a sensible idea.

$$y_t = \mathbf{x}_t' \Phi + w_t u_t$$

- `options.heterosked_weights` is the (`T - lags` $\times$ 1) array of weights; e.g. sample 2000m1 to 2020m7

```
options.heterosked_weights =  [1 .... 1 30 20 10 1 1];
```

- Optimize the weights (along with Minnesota hyper-parameters) by defining

```
options.objective_function = 'bvar_opt_heterosked';
options.tstar              = find(time==2020) + 2;  %march 2020
[postmode,logmlike,HH]     = bvar_max_hyper(hyperpara,y,lags,options);
```

Outline

Loose ends

- TBA soon: Dynamic FAVAR, trend/cycles and gaps extraction, dating BC.

Loose ends

- TBA soon: Dynamic FAVAR, trend/cycles and gaps extraction, dating BC.
- TBA later (if ever): non-linear models, TVP and SV; translate to Python

Loose ends

- TBA soon: Dynamic FAVAR, trend/cycles and gaps extraction, dating BC.
- TBA later (if ever): non-linear models, TVP and SV; translate to Python
- For bugs, open an issue in github. Or send an email.

Loose ends

- TBA soon: Dynamic FAVAR, trend/cycles and gaps extraction, dating BC.
- TBA later (if ever): non-linear models, TVP and SV; translate to Python
- For bugs, open an issue in github. Or send an email.
- Feedback and suggestions for improvements are welcome

Outline

Empirical and structural innovations

- Reduced-form VAR innovations u_t and structural disturbances ν_t

$$u_t = \Omega \ \nu_t = \Omega_0 \ Q \ \nu_t$$

- $E(\nu_t \nu_t') = I$ and $\Omega\Omega' = \Sigma$,
- Ω_0 is the Cholesky decompostion of Σ and Q is an orthonormal rotation such that $Q'Q = QQ' = I_n$.
- To recover ν_t, we need to impose restrictions on Ω.
- This is because Σ only contains $n(n+1)/2$ estimated elements, while Ω has n^2 elements.