A Hitchhiker Guide to Empirical Macro Models Toolbox[1]

Filippo Ferroni (Federal Reserve Bank of Chicago)
Fabio Canova (Norwegian Business School, Norges Bank, and CEPR)

Bank of Italy
June 9, 2021

[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 (EMT) is a package which uses MATLAB (Octave) functions to:

- Estimate reduced form and structural macro models with Classical or Bayesian methods.

Introduction

The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions to:

- Estimate reduced form and structural macro models with Classical or Bayesian methods.

- Allows inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks; nowcasts and mixed-frequency estimation, heterogeneous dynamic panels, regularization methods.

Introduction

The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions to:

- Estimate reduced form and structural macro models with Classical or Bayesian methods.
- Allows inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks; nowcasts and mixed-frequency estimation, heterogeneous dynamic panels, regularization methods.
- It can also be used for decomposing a series in trend and cycle, for dating turning points, and for computing statistics of cyclical phases.

Introduction

The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions to:

- Estimate reduced form and structural macro models with Classical or Bayesian methods.
- Allows inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks; nowcasts and mixed-frequency estimation, heterogeneous dynamic panels, regularization methods.
- It can also be used for decomposing a series in trend and cycle, for dating turning points, and for computing statistics of cyclical phases.
- And to study interconnectedness and spillovers in networks.

Introduction

The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions to:

- Estimate reduced form and structural macro models with Classical or Bayesian methods.
- Allows inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks; nowcasts and mixed-frequency estimation, heterogeneous dynamic panels, regularization methods.
- It can also be used for decomposing a series in trend and cycle, for dating turning points, and for computing statistics of cyclical phases.
- And to study interconnectedness and spillovers in networks.
- Source codes with examples can be forked/downloaded at https://github.com/naffe15/BVAR_ or https://sites.google.com/view/fabio-canova-homepage/home/empirical-macro-toolbox

Introduction

The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions to:

- Estimate reduced form and structural macro models with Classical or Bayesian methods.
- Allows inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks; nowcasts and mixed-frequency estimation, heterogeneous dynamic panels, regularization methods.
- It can also be used for decomposing a series in trend and cycle, for dating turning points, and for computing statistics of cyclical phases.
- And to study interconnectedness and spillovers in networks.
- Source codes with examples can be forked/downloaded at https://github.com/naffe15/BVAR_ or https://sites.google.com/view/fabio-canova-homepage/home/empirical-macro-toolbox
- The hitchhiker guide can be downloaded at https://www.filippoferroni.com/empiricalmacrotoolbox or https://sites.google.com/view/fabio-canova-homepage/home/empirical-macro-toolbox

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation; Point/density forecasts; several identification procedures, computation of IRFs, historical and variance decompositions; missing values/mixed frequency estimation; VARXs; panels of VARs; time varying coefficient and heteroschedastic VARs.

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation; Point/density forecasts; several identification procedures, computation of IRFs, historical and variance decompositions; missing values/mixed frequency estimation; VARXs; panels of VARs; time varying coefficient and heteroschedastic VARs.

- Compression methods:
  Static (PC) FAVARs.

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation; Point/density forecasts; several identification procedures, computation of IRFs, historical and variance decompositions; missing values/mixed frequency estimation; VARXs; panels of VARs; time varying coefficient and heteroschedastic VARs.

- Compression methods:
  Static (PC) FAVARs.

- Local projection methods:
  Classic and Bayesian estimation, point and density forecasts, structural IRFs.

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation; Point/density forecasts; several identification procedures, computation of IRFs, historical and variance decompositions; missing values/mixed frequency estimation; VARXs; panels of VARs; time varying coefficient and heteroschedastic VARs.

- Compression methods:
  Static (PC) FAVARs.

- Local projection methods:
  Classic and Bayesian estimation, point and density forecasts, structural IRFs.

- Filtering methods:
  Extracting trend/cycles (permanent/trasnitory) components, dating BC, computing statistics of cyclical phases.

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation; Point/density forecasts; several identification procedures, computation of IRFs, historical and variance decompositions; missing values/mixed frequency estimation; VARXs; panels of VARs; time varying coefficient and heteroschedastic VARs.

- Compression methods:
  Static (PC) FAVARs.

- Local projection methods:
  Classic and Bayesian estimation, point and density forecasts, structural IRFs.

- Filtering methods:
  Extracting trend/cycles (permanent/trasnitory) components, dating BC, computing statistics of cyclical phases.

- (NEW) Regularization Methods. Network analysis
  Lasso, Ridge, Elastic-Net, Bayesian shrinkage estimation. Connectedness and spillovers measures.

Road map I

Topics (with examples in the tutorials as blueprints):

I Dating turning points. Cyclical statistics.
`USERLOCATION/BVAR/v4.2/'Trend-Cycle-Dating tutorial'/example_2_dating.m`

II Trend and cycle decompositions. Comparison of the financial and the real cycle.
`USERLOCATION/BVAR/v4.2/'Trend-Cycle-Dating tutorial'/example_1_cycles.m`

III Classical VARs (Bayesian estimation with flat prior).
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_1_classical.m`

IV VARX: estimation and inference.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_6_VARX.m`

V Bayesian VARs: Minnesota and conjugate priors. Estimating Minnesota hyper-parameters.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_2_minn.m`

VI IRFs, Historical and Variance Decomposition with various identification schemes.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_3_irf.m`

Road map II

Topics (examples in the tutorial as blueprints):

VII PCVAR (FAVAR) estimation and inference.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_5_favar.m`

VIII Panels of VARs.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_8_panels.m`

IX Nowcasts and Mixed-Frequency VAR estimation.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_4_mfvar.m`

X Point and Density Forecasts. Conditional Forecasts.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_9_prediction.m`

XI Local projections and direct forecasts.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_7_LP.m`

XII (NEW) Regularization methods and network analysis.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_11_connectedness.m`

Getting started

I Open MATLAB, and add the toolbox to the MATLAB path.
Assuming that EMT is stored in `C:\USERLOCATION`, and you are using version 4.2, you can type in the command window (or add to your script):

```
addpath C:/USERLOCATION/BVAR/v4.2
```

You need to this every time you start MATLAB. Alternatively, go to the set-path icon in the MATLAB HOME window, click on it, add with BVAR/v4.2 with subfolders, and save.

II Use the same procedure to install Chris Sims' optimization routines (used to optimally select the Minnesota prior hyper-parameters). In this case, type in the command window (add to your script)

```
addpath C:/USERLOCATION/BVAR/cmintools
```

III Load the data

- If the data is in matlab format, type in the command window:
  ```
  load data
  ```
  and check the content in the workspace window.

- If the data is in a spreadsheet use (standard MATLAB command):
  ```
  y=xlsread('filename','sheetname')
  ```

where `filename` is the name of the file, and `sheetname` is the name of the sheet where the data is located (in case you have many sheets).

- For information about loading data in other formats, type in the command window:
  ```
  help load
  ```

Today: Fixed coefficients (B)VAR models

I Classical and Bayesian VARs; priors and posteriors. Uninformative, Minnesota and conjugate priors. How to chose the Minnesota hyper parameters.

`tutorial: USERLOCATION/BVAR_/v4.2/'BVAR tutorial'/example_2_minn.m`

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

`tutorial: USERLOCATION/BVAR_/v4.2/'BVAR tutorial'/example_3_irf.m`

III Nowcasts and Mixed-Frequency estimation.

`tutorial USERLOCATION/BVAR_/v4.2/'BVAR tutorial'/example_4_mfvar.m`

IV Point and density forecasts. Conditional forecasts.

`tutorial  USERLOCATION/BVAR_/v4.2/'BVAR tutorial'/example_9_prediction.m`

IV Shrinkage estimation and network analysis

`tutorial  USERLOCATION/BVAR_/v4.2/'BVAR tutorial'/example_11_connectedness.m`

Notation

- *VAR(p)*: A vector auto regression with p lags is:

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

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

- For inference: A VAR(p) can be expressed as a SURE system:

$$Y = X\Phi + E$$

- For forecasts and IRFs: VAR(p) can be expressed as a (companion form) VAR(1):

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

- For decompositions: A VAR(p) 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

VAR Inference

- Classical estimation equivalent to estimation with flat or informative priors. By default, a Jeffrey (uninformative) prior is assumed; other prior options are discussed below.
- Given a prior, draws are generated from the posteriors using a Gibbs sampler algorithm; see the guide for more details.
- The baseline estimation function is

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

- Inputs:
  - `y`, a $(\mathtt{T} \times \mathtt{n})$ array of data (i.e. rows=time, columns=variables)
  - $\mathtt{lags} > 0$.
  - `options` a field structure specifying customized options (i.e. priors, horizon, identification scheme, etc.)

Outputs

- BVAR a field with many sub-fields. Here's the most important ones.
  - BVAR.Phi_draws is a $(n \times \text{lags} + 1) \times n \times K$ array with draws for Φ. K= number of draws. 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 $n \times n \times K$ matrix containing K draws fof Σ.
    BVAR.e_draws is a $(T - \text{lags}) \times n \times K$ 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. $n \times \text{hor} \times n \times K$ matrix) collecting the impulse response function obtained with recursive identification.
  - BVAR.forecasts.no_shocks (with_shocks) is a three dimensional object (i.e. $\text{fhor} \times n \times K$ 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 contains the Akaike information criterion, AIC, Hannan-Quinn information criterion HQIC and the Bayes information criterion,BIC.
- Default value: K=5000 (can change with options.K = numb;).
- By default, a constant is used (change with options.noconstant = 1;).

Priors

The toolbox allows for three types of priors

- Uninformative or Jeffrey prior (default). The estimation command is:

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

- Minnesota prior with default hyper-parameter values. The estimation commands are:

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

- Conjugate prior: Multivariate-Normal Inverse-Wishart with default (loose) values. The estimation commands are:

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

Minnesota prior

- Minnesota prior is controlled by 5 scalar hyper-parameters regulating different layers of shrinkage
  - τ `options.minn_prior_tau`: overall tightness (default 3). The larger it is, the tighter is prior.
  - d `options.minn_prior_decay`: tightness on the lags greater than one (default 0.5). The larger it is, 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 `options`
  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 the different options are selected, estimation simply requires:

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

Minnesota prior with optimal hyper-parameters

- Various options can be set for the maximization step:
  - `options.index_est` is a row vector selecting 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`
- If the maximum is found, the posterior distribution is computed using the optimal hyper-parameter values. If maximization unsuccessful, the posterior distribution is computed with default values (and a warning is issued).
- Tip: better maximize one parameter at time starting the optimal values obtained in the previous step. To speed up computation can use the function (no posterior draws are computed)

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

Conjugate MN-IW

- Multivariate Normal-Inverse Wishart conjugate prior.

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

  Default setup: $\Phi \sim N(0, 10\ I_{np+1})$ 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. Used to adjust for variables with different units.

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

- After options are selected, estimation requires:

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

VARX

- EMT allows a VAR to have exogenous variables. Need the option 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 assuming either Jeffrey or a conjugate prior; a Minnesota prior is not currently supported.

- After the option selection, estimation requires:

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

Outline

IRFs

- For each posterior draw, EMT computes the IRF for a given identification scheme
- Results stored in a four dimensional ($\mathtt{n} \times \mathtt{hor} \times \mathtt{n} \times \mathtt{K}$) (variable, horizon, shock, draw) array.
- Priors allowed: flat, Minnesota, Conjugate MN-IW.
- Default horizon 24; change with `options.hor = hor;`
- By default a Cholesky scheme is used and responses are stored in `BVAR.ir_draws`
- To use other identification schemes the user must specify what she wants in `options` (see next slides) before launching `bvar`.

Long run restrictions

- Use option:

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

- First variable in `y` lmust be specified in log difference
- Assumes that first disturbance has long run effects on the first variable.
- Reference: Gali (1999).
- If more than one long run shock is needed, as in Fisher(2006), introduce more variables in log difference.
- IRFs identified with long run restrictions are stored in `BVAR.irlr_draws`

Sign and Magnitude

- Use option:

```
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 identified with sign or mangitude restrictions are stored in `BVAR.irsign_draws`

Narrative restrictions

- Use option:

  ```
  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. Can be used, for example:

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

  i.e., 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 (2019)

- IRFs identified with narrative restrictions are stored in `BVAR.irnarrsign_draws`

IV identification

- Use the option:

```
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 with IV are stored in `BVAR.irproxy_draws`
- By convention, the structural shock of interest is ordered **first**; i.e. `BVAR.irproxy_draws(:,:,1,:);`

Mixed identification schemes

- Mix of zero (short and long run) and sign restrictions allowed.
- `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 identified with mixed restrictions are stored in `BVAR.irzerosign_draws`

Plotting IRFs

- Plotting function:

```
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 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.

Additional identifications: Maximization of FEVD

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

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

  to find 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)
```

- Can loop over draws (`Phi,Sigma`)

- General (identification) option:

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

Other useful summary functions

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

- 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 (Cholesky).

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

Outline

Mixed Frequency VAR

- A Mixed Frequency VAR (MF-VAR) allows nowcasts, backcast, or retrieving variables that have an arbitrary pattern of missing observations.
- Gibbs sampler is used:
  1. Generate a draw from the posterior distribution of the reduced form VAR parameters, conditional on observables and states.
  2. With the draw run the Kalman smoother and get estimate the latent states (e.g. monthly GDP).
  3. Repeat 1. and 2.
- Can be used with flat, Minnesota (except max), Conjugate MN-IW priors. Allows IRFs with all identification schemes and forecasts.
- MF-VAR estimation is automatically triggered in bvar.m whenever there are `NaN` in the data, `y`. (a warning is printed).

Monthly-Quarterly Frequency

- Example with quarterly and monthly variables. Time unit is month.
- $y = [y^m, y^q]$; the q variables are observed in the last month of the q, elsewhere `NaN`. Define the unobserved states as x_t.
- State space setup: the transition equation is the VAR for x_t and measurement equation maps x_t into y_t.
  - For monthly variables: $y_t^m = x_{m,t}$
  - For **stock** quarterly variables: $y_t^q = x_{q,t}$. [no instruction is needed]
  - For **flow** quarterly variables (e.g. GDP): $y_t^q = \frac{1}{3}(x_{q,t} + x_{q,t-1} + x_{q,t-2})$ use:
    ```
    options.mf_varindex     = num;
    ```
    where `num` is a scalar indicating the position of the q flow variable (column # in `y`)
- Additional outputs produced by MF-VAR: `BVAR.yfill` (smoothed) and `BVAR.yfilt` (filtered) latent variables. Both are $\texttt{T} \times \texttt{n} \times \texttt{K}$ arrays.

Outline

Forecasts

- BVAR.forecasts.no_shocks is a (nfor $\times$ n $\times$ K) matrix (time, variable, draw). Future shocks are zeros.
- BVAR.forecasts.with_shocks is a (nfor $\times$ n $\times$ K) matrix (time, variable, draw). Future shocks are drawn randomly.
- BVAR.forecasts.conditional is a (nfor $\times$ n $\times$ 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 is:

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

  `optnsplt` can be omitted.

- `frcst_to_plot` is a three dimensional array (time,variable,draw) containing the forecasts; 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 period 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

- Are VAR models useful in times like the current ones?
- 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 for estimation. Sensible idea. Setup:

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

- `options.heterosked_weights` is the `(T - lags` $\times$ `1)` array of weights; e.g. for sample 2020m3 to 2020m6

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

Regularization methods

- When `bvar` is used for a large system, it allows the use of regularization methods.
- Can be initialized with options: e.g. `options.Ridge.est=1`,`options.Lasso.est=1`, `options.ElasticNet.est=1`.
- Parameters can be set in option command, e.g. `options.Ridge.lambda=0.02`, `options.ElasticNet.alpha=0.05`.
- Estimation uses the standard command `bvar1= bvar_(y, lags, options)`.
- Statistics computed using the variance decomposition at horizon `option.nethor=H`. Default is Pesaran GIR decomposition. Possible to use other identification methods.
- If any of the standard priors is used, it produces these three measures for K draws, e.g. `options.K = 1000`.

Network analyis I

bvar1 contains:

- a measure of overall connectedness, e.g. `bvar1.Ridge.Connectedness.Index`;
- a measure of spillovers `bvar1.Lasso.Connectedness.FromUnitToAll`;
- a measure of vulnerability `bvar1.ElasticNet.Connectedness.FromAllToUnit`.

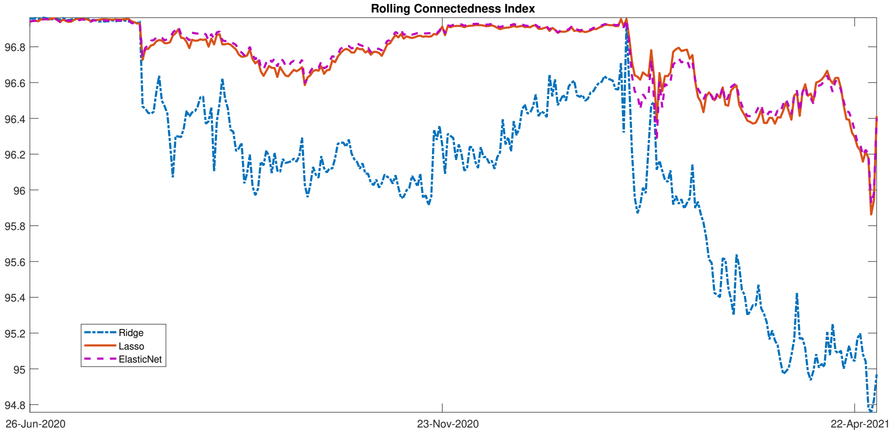
```
Overall Connectedness Index: Crypocurrencies
     'Mean Minnesota'    'Ridge'     'Lasso'     'ElasticNet'
       [95.5434]        [95.0069]    [95.7602]    [95.8177]

Spillover: From Binance Coin to All
     'Mean Minnesota'    'Ridge'     'Lasso'    'ElasticNet'
          [3.7216]        [3.6051]    [3.2901]     [3.2444]

Vulnerability: From All to Binance Coin
    'Mean Minnesota'    'Ridge'      'Lasso'    'ElasticNet'
       [2.9918]           [2.9891]   [3.0058]    [3.0082]
```

Network analysis II

- Can perform recursive analysis



Outline

Forthcoming

- Next version (out at the end of the month) has Lasso, Ridge and other shrinkage methods. Network analysis and statistics.

Forthcoming

- Next version (out at the end of the month) has Lasso, Ridge and other shrinkage methods. Network analysis and statistics.
- Later: Dynamic Factor Models.

Forthcoming

- Next version (out at the end of the month) has Lasso, Ridge and other shrinkage methods. Network analysis and statistics.
- Later: Dynamic Factor Models.
- Much later (if ever): non-linear models, TVP and SV; translate to Python.

Forthcoming

- Next version (out at the end of the month) has Lasso, Ridge and other shrinkage methods. Network analysis and statistics.
- Later: Dynamic Factor Models.
- Much later (if ever): non-linear models, TVP and SV; translate to Python.
- For bugs, open an issue in github. Or send an email.

Forthcoming

- Next version (out at the end of the month) has Lasso, Ridge and other shrinkage methods. Network analysis and statistics.
- Later: Dynamic Factor Models.
- Much 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.