

1 Understanding the VARCode

The code folder structure should be as follows:

1) VarCode

- a. Code → this folder contains the code to run the VAR,
- b. Data → this folder contains the data run in the VAR,
- c. Outdata → this folder contains the output of the VAR,
- d. Documentation → this folder contains the documentation to the code (this file you're reading right now!).

1.1 Code

The following folder contains two files, the *main.jl* file which is the only one that you need to run in julia by opening Julia-0.5 and running the command:

```
include("main.jl").
```

The second file, *option.jl* is the only file you need to modify in the Code folder in order to run whatever version of the VAR you wish to run. Here are the options:

- VAR options:
- p: number of lags for the VAR,
 - date_format: format the dates in the *.csv* file are in (e.g. "yyyy-mm-dd"),
 - start_date: the date you wish the VAR to start (remember that if you are using differences then you lose one period!),
 - end_date: the date you wish the VAR to end,
 - Vars: the names you wish the code to use for the variables you are including in the VAR (be consistent with the order across *Vars*, *Mnems*, *Vars_names* and *Transformations*),
 - Mnems: the mnemonics that these variables have in your *.csv* sheet (necessary for identification of the columns),
 - Vars_names: the full names of the series as you wish they appear in plot titles,
 - Transformations: transformations applied to each series (in order),
 - dates_mnems: mnemonic for the date column,
- IRF options:
- ndraws: number of draws to compute the confidence intervals of the IRFs,
 - nperiods: number of periods to show in IRF plot
 - conf_int: size of confidence interval (e.g. 90%, 70%, etc).

Furthermore, the folder contains a subfolder called "functions" which contains functions used by the main file. These need not be touched but can be modified ad hoc. The functions are:

Data_organizer: This function takes in the data as a dataframe, the mnemonics, the start line and the end line (corresponding to the start_date and end_date), the transformations for each series as well as the number of periods and number of variables. It uses these inputs to create a matrix of the data where each row corresponds to the data series specified in the option file.

- `var_reg`: This function runs the VAR in OLS form and spits out the estimated coefficients, the estimated variance of the residuals, the fits, the actual (dependent data, i.e. LHS of regression) and the independent variables (RHS of regression).
- `forecast`: This function does the h -period ahead forecast of the variables with the VAR, where h is indicated by the user ($h = 1$ in *main.jl*).
- `imp_res`: This function, given the estimated VAR outputs, gives a 4-dimensional matrix with the IRF data ($n_{\text{periods}} \times n \times n \times n_{\text{draws}}$), where the second n indicates the response and the third indicates the shock.
- `RMSE`: Computes the RMSE given the Actual and the Fit.
- `dummy_fores`: Computes fake data based on the “initial data” (i.e. the first p periods of the data necessary to forecast). The “dummy data” is a necessary part of the boot-strapping used to get confidence intervals.

1.2 Data

The data folder only contains the data file which, by default, is called *Data.csv*. This file should be a *.csv* file and should contain the dates (with some column title) and each series with their respective (unique) titles.

It is important to give it this structure so that the program is capable of reading the table correctly and structuring the VAR properly.

1.3 Outdata

This folder contains the following sub-folders:

- IRFs: Here you will find all the impulse response functions named using the following convention: *IRF_shockvar_responsevar.pdf*.

2 Some Theory

2.1 Estimating the VAR

The way we estimated the VAR is by constructing first the LHS and the RHS variables given all data available

$$Y_{1:T} = \begin{pmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ \cdots & \cdots & \cdots & \cdots \\ y_{T,1} & y_{T,2} & \cdots & y_{T,n} \end{pmatrix}_{T \times n}$$

The model will then look like:

$$LHS = RHS \cdot \beta,$$

where

$$LHS_{(T-p) \times n} = \begin{pmatrix} y_{p+1,1} & y_{p+1,2} & \cdots & y_{p+1,n} \\ \cdots & \cdots & \cdots & \cdots \\ y_{T,1} & y_{T,2} & \cdots & y_{T,n} \end{pmatrix}$$

and

$$RHS_{(T-p) \times np} = \begin{pmatrix} y_{p,1} & \cdots & y_{p,n} & \cdots & \cdots & y_{1,1} & \cdots & y_{1,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{T-1,1} & \cdots & y_{T-1,n} & \cdots & \cdots & y_{T-p,1} & \cdots & y_{T-p,n} \end{pmatrix}$$

As we include a constant term (which the code does) our RHS will be augmented as follows,

$$RHS_{(T-p) \times (np+1)} = \begin{pmatrix} 1 & y_{p,1} & \cdots & y_{p,n} & \cdots & \cdots & y_{1,1} & \cdots & y_{1,n} \\ 1 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & y_{T-1,1} & \cdots & y_{T-1,n} & \cdots & \cdots & y_{T-p,1} & \cdots & y_{T-p,n} \end{pmatrix}$$

This brings us to the estimation (in simple OLS form):

$$\hat{\beta}_{(np+1) \times n} = (RHS'_{(np+1) \times (T-p)} \cdot RHS_{(T-p) \times (np+1)})^{-1} \cdot RHS'_{(np+1) \times (T-p)} \cdot LHS_{(T-p) \times n} \quad (1)$$

and

$$\hat{\Sigma}_{n \times n} = \frac{(LHS - RHS \times \hat{\beta})' * (LHS - RHS \times \hat{\beta})}{T} \quad (2)$$

Furthermore, assuming that the regression has been run with a constant term, we can re-organize $\hat{\beta}$ as

$$C_{np \times 1} = \begin{pmatrix} \hat{\beta}[1, :]' \\ 0_{(p-1) * n \times 1} \end{pmatrix} \quad (3)$$

containing the constant term, and

$$\Psi_{np \times np} = \begin{pmatrix} \hat{\beta}[2 : end, :]' \\ [I_{(p-1) \cdot n \times (p-1) \cdot n} 0_{(p-1) \cdot n, n}] \end{pmatrix} \quad (4)$$

which allows us to write the VAR in companion form

$$Z_t = C_{np \times 1} + \Psi_{np \times np} \cdot Z_{t-1} + \epsilon_t_{np \times 1}$$

where $Z_t = \begin{pmatrix} z_t \\ z_{t-1} \\ \vdots \\ z_{t-p+1} \end{pmatrix}$, $z_t = \begin{pmatrix} y_{1,t} \\ \vdots \\ y_{n,t} \end{pmatrix}$ and $\vec{\epsilon}_t = \begin{pmatrix} \epsilon_t \\ 0_{n(p-1) \times 1} \end{pmatrix}$, with $\epsilon_t = \begin{pmatrix} \varepsilon_{1,t} \\ \vdots \\ \varepsilon_{4,t} \end{pmatrix}$ in particular
 $\vec{\epsilon}_t \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix}\right)$

2.2 Forecast

We derive the equation used for our forecasting applying recursively the following

$$\begin{aligned} Z_{t+1|t} &= E[C + \Psi Z_t + \epsilon_t] = C + \Psi Z_t \\ Z_{t+h|t} &= C + \Psi Z_{t+h-1|t} = C + \Psi(C + \Psi Z_{t+h-2|t}) \\ &= \dots \\ &= \left(\sum_{j=0}^{h-1} \Psi^j\right) C + \Psi^h Z_t \end{aligned}$$

2.3 Impulse Response Functions

The aim here is, for $h=1:H$, to get $\frac{\partial Y_{t+h}}{\partial \epsilon_{t,j}}$, i.e. looking at how the vector of variables at times $t+h$ respond to a shock of variable j at time t .

Our system can be written as above (using the explicit form of eq. 4):

$$\begin{pmatrix} z_t \\ \vdots \\ z_{t-p+1} \end{pmatrix} = C + \begin{pmatrix} A_1 & \dots & A_p \\ I_{(p-1) \cdot n \times (p-1) \cdot n} & \dots & O_{(p-1) \cdot n \times n} \end{pmatrix} \begin{pmatrix} z_{t-1} \\ \vdots \\ z_{t-p} \end{pmatrix} + \begin{pmatrix} \epsilon_t \\ 0_{n(p-1) \times 1} \end{pmatrix}$$

Renaming appropriately, this system can be written as

$$Z_t = C + \Psi Z_{t-1} + S' \epsilon_t$$

where $S = (I_{n \times n} O_{n \times n(p-1)})$.

Because the shocks are not necessarily orthogonal, we use the Cholesky decomposition $P u_t = \epsilon_t$ where $P = \text{chol}(\hat{\Sigma})$ (lower-triangular). Thus we have

$Z_{t+1} = C + \Psi(C + \Psi Z_{t-1} + S' P u_t) + S' P u_{t+1}$, $Z_{t+2} = C + \Psi(C + \Psi(C + \Psi Z_{t-1} + S' P u_t) + S' P u_{t+1}) + S' P u_{t+2}$. However, once we take the partial with respect to u_t we notice that all terms except for u_t do not matter. Thus we have

$$\frac{\partial Z_{t+h}}{\partial u_t} = \frac{\partial Z_{t+h}}{\partial u_t} = \Psi^h \cdot S' \cdot P$$

If we want to select the response at a certain time t' we then pre-multiply this result by the appropriate selection matrix. For example, to select the response on the latest period $t+h$ we pre-multiply by S .

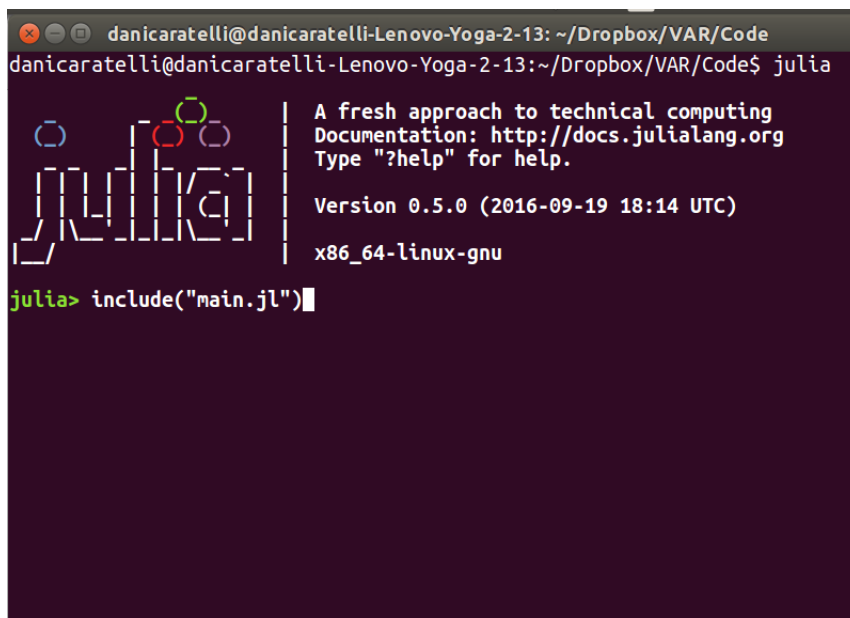
3 An example

Included in this folder is an example used for a weekly briefing.

We run a VAR with three variables: unemployment rate (UNRATE), housing starts (HSN1F) and industrial production (INDPRO). Notice, both the names you want the variables to be called and the mnemonics of reference in the *Data.csv* need to be inputted in the data file. Furthermore, the data contains also a date column whose name needs to be specified (in this case it is *observation_date*).

Since we want the series to be in level for all three series, we include these transformations in the *options.jl* file as 'lvl'.

We run the main file opening julia in the terminal as follows (this will look different from different terminals):



```
danicaratelli@danicaratelli-Lenovo-Yoga-2-13: ~/Dropbox/VAR/Code
danicaratelli@danicaratelli-Lenovo-Yoga-2-13:~/Dropbox/VAR/Code$ julia

      _       _
     / \     / \
    _/  \_   _/  \_
   / _  \ / _  \
  / _  \|/ _  \|
 / _  \|/ _  \|
/_ _  \|/ _  \|

A fresh approach to technical computing
Documentation: http://docs.julialang.org
Type "?help" for help.

Version 0.5.0 (2016-09-19 18:14 UTC)
x86_64-linux-gnu

julia> include("main.jl")
```

This will produce IRF plots contained in the appropriate folder such as the following:

