MODELING US REAL GDP AND UNEMPLOYMENT

On the course web site there are data on US real gdp and unemployment, quarterly from 1948:I through 2016:II. You are to fit two kinds of bivariate models to these data.

(1) A second-order VAR, conditioning on the data in the first two time periods and using a proper conjugate prior. This will be done nearly automatically for you if you use the `mgnldnsty()` R function posted on the web site. Its default prior is a reasonable choice, though you will have to specify the `vprior$sig` parameter. For log real gdp and unemployment rate divided by 100, `vprior=list(sig=c(.005,.005), w=1)` is a reasonable choice.

(2) A dynamic factor model of the form

$$\begin{aligned} y_t &= \rho y_{t-1} + \alpha z_t + \delta + e_t \\ z_t &= \gamma z_{t-1} + u_t \,, \end{aligned}$$

where y_t is the 2-dimensional vector of time-t data on log real gdp and unemployment/100, z_t is an unobservable driving process (the business cycle?), δ is a vector of constants, and e_t, u_t form a three-dimensional vector of uncorrelated normal random variables with possibly different variances. ρ is a diagonal matrix and α is a 2×1 column vector. You should again condition on the first two values of y. For this model, only y_2, not y_1 will matter for the distribution of y_t, $t > 2$, but we want to keep the sample being explained by the model matching across the two specifications. You will assume (for convenience, and contrary to the implications of the model) that z_2 is independent of y_1 and y_2, and give z_2 a prior $N(0, 400)$ distribution. You will need to get this model into Kalman filter form and produce a function that will return a likelihood value as a function of the parameters, of which there are nine.

When you have estimates (posterior maxima) for each model, use each model to:

(a) Forecast two years past the end of the sample and compare their implications. For the VAR you can use `{fcast}`.

(b) produce an estimated covariance matrix of innovations. (For the Kalman filtered estimate, one-step-ahead covariance matrices for the observations should converge by the end of the sample to a very good approximation of the innovation covariance matrix. Explain why.)

(c) Calculate the time series of forecast errors (based on the full sample estimate for the VAR and on the recursive filtered errors for the factor model) divided by their model-implied standard errors. Use the results to comment on whether there is strong evidence of non-normality or time-varying heteroskedasticity.

(d) Calculate and plot impulse responses. The R function `impulsdtrf()` calculates impulse resonses for the VAR. For the factor model it can be done by feeding the Kalman

Date: October 8, 2016.

filter artificial data in which only initial values of y_t differ from the model's forecast. For either kind of impulse response, `plotir` plots them.

Remarks

For a two-variable system, the class of all single-index models is dense in the set of linearly regular processes, just as is the class of all finite-order VAR's. In other words, if we allowed lag lengths to be arbitrarily long, the differences between the two models explored in the exercise would become negligible.

The web site contains a directory called `VARex5132016`. If you use R, you can download this entire directory and use it as an R package. With the directory downloaded, you "install" it via `install.packages{''VARex5132016'', repos=NULL}`, assuming R's working directory includes the `VARex5132016` directory as subdirectory. The `repos=NULL` tells R to look for the package on disk, not on a remote repository. Once the package is installed, all the functions mentioned above will be available after you give the `library(``VARex5132016'')` command, and R help files for them can be accessed in the usual way.

To give the factor model a proper prior, you would need a prior on its parameters. The function `yuindex` on the web site just calculates the likelihood, not prior times likelihood. For a later exercise formally comparing these two models, we will need to implement a proper prior on the parameters of the factor model.

The R function `kf2()` does a step of the Kalman filter, and the R function `yuindex()` goes through the whole sample and returns minus the log likelihood (for use with a minimizer function). The R function `csminwelNew()` will work as a minimizer, but there are many others. When you've done this, construct and plot on the same graph the filtered and smoothed estimates of the time path of `z`.

R can be downloaded free from the internet, and comes with tutorials and help files. If you know Matlab and find the prospect of learning R basics daunting, there are on the web site Matlab versions of or predecessors of the R functions mentioned above, except for `yuindex` and `plotir`. You can probably translate `yuindex` to Matlab fairly easily. I have not tested the Matlab functions on this problem, and it's possible they have bugs that will emerge when you try to use them here. Let me know if this occurs and I'll try to make repairs.

And of course anything is possible in Python, but starting from scratch on this in Python may be a lot of work.