The BEKK package for gretl

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

version 0.7

Abstract

This package is used for estimation of a particular flavour of multivariate GARCH models known as BEKK (Engle and Kroner, 1995) via quasi maximum likelihood, employing analytical derivatives as per Lucchetti (2002).

1 Econometrics

The class of models this package handles can be written as

$$\begin{aligned} \Phi(L)\mathbf{y}_t &= \Pi\mathbf{x}_t + \boldsymbol{\varepsilon}_t && (1)\\ \boldsymbol{\varepsilon}_t|\mathscr{F}_t &\sim N(0,\Sigma_t) && (2)\\ \Sigma_t &= CC' + A\boldsymbol{\varepsilon}_{t-1}\boldsymbol{\varepsilon}'_{t-1}A' + B\Sigma_{t-1}B' && (3) \end{aligned}$$

where

- $\mathbf{y}_t$ is an n-element vector of time series (typically asset returns)
- $\mathbf{x}_t$ is a k-element vector of time series (in most cases, just a constant)
- $\Phi(L)$ is a matrix polynomial of order p
- $\boldsymbol{\varepsilon}_t$ is an n-variate Gaussian process whose conditional variance at time t is Σ_t
- C is a lower-triangular $n \times n$ matrix
- A and B are $n \times n$ matrices.

Estimation is carried out by QML on the full set of parameters $\Phi(\cdot)$, Π, C, A and B, using analytical derivatives. At present, there's no provision for other densities than the multivariate normal. In some applications, the A and B matrices are sometimes constrained to be diagonal or scalar; this is not possible at present in this package, but it wouldn't be difficult to add upon request.

The covariance matrix for estimated parameters can be estimated from the Outer Product of Gradients (OPG), the inverse Hessian or, by default, via the robust "sandwich" matrix. The other two choices are provided in the interest of faster estimation.

The workflow is similar to that of other gretl packages:

1. You set up a model by the function `BEKK_Setup`: it produces a bundle with the basic info for your model, that will be used later as a container for the results.

2. You estimate the model parameters via the function `BEKK_Estimate`, that stores the estimates into the model bundle

3. You do things with the returned bundle via ad-hoc functions (eg. `BEKK_Printout` prints out the estimates)

2 Examples

2.1 A minimal example

Here we provide a minimal usage example:

```
bundle Mod = BEKK_Setup(R, 0, const)
BEKK_Estimate(&Mod)
BEKK_Printout(&Mod)
```

Here we're assuming that the vector $\mathbf{y}_t$ is contained in the gretl list object `R`. The first line sets up the model with $p = 0$ lags in $\Phi(L)$ and just a constant term in $\mathbf{x}_t$. The model itself is stored into the gretl bundle `Mod`. The second line carries out estimation (this is typically the CPU-intensive part) and line 3 prints out the estimated parameters. Note that in lines 2 and 3 the bundle argument must be given in pointer form (ie, prefixed with an `'&'` character).

2.2 A fuller example

Here we reproduce a slightly edited version of the sample script provided with the package:

```
set verbose off
include BEKK.gfn

# --- DATA PREPARATION ------------------------------------------------

open IT_companies.gdt --frompkg=BEKK --quiet
smpl 2010-01-01 ;
list Prices = IBM AAPL MSFT
list Returns = ldiff(Prices)

# --- MODEL SETUP, ESTIMATION AND PRINTOUT ----------------------------

scalar VAR_lags = 1
list VAR_exog = const
bundle Model = BEKK_Setup(Returns, VAR_lags, VAR_exog)
scalar verbosity = 1
BEKK_Estimate(&Model, verbosity)
BEKK_Printout(&Model)

# --- POSTESTIMATION --------------------------------------------------

series v_IBM = BEKK_GetVariance(&Model, 1)
series v_AAPL = BEKK_GetVariance(&Model, 2)
series c12 = BEKK_GetCovariance(&Model, 1, 2)
series cc = c12 / sqrt(v_IBM*v_AAPL)
```

The code is divided into 4 blocks:

- Block 1 is just two lines to set things up and import the package;

- In Block 2 we open one of the sample datasets (daily tech stocks), limit the sample to 2010 onwards, and compute the log returns
- Block 3 estimates the model, which is a VAR(1) with a constant for the conditional mean equation (1), and prints out the estimates
- In Block 4, we retrieve three series from the estimated model: the conditional variances for IBM and Apple and their conditional covariances. These are the used to compute the conditional correlation series. The four series are shown in Figure 1.

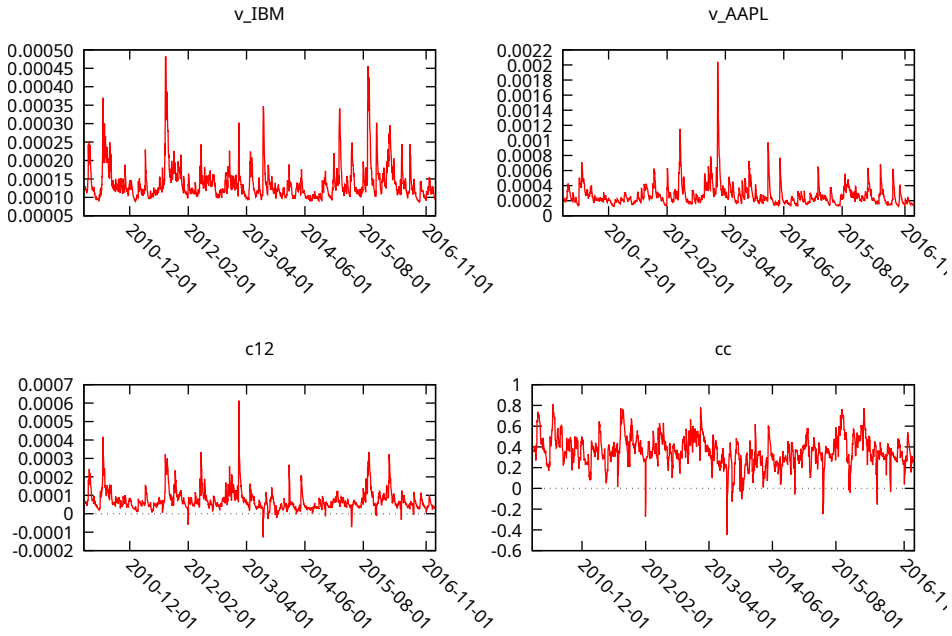

Figure 1: BEKK postestimation example

3 The Graphical interface

When you install the package, you will get a prompt from gretl asking you if you want to create a menu attachment for the package. If you agree, an new item named `BEKK` will appear under the `Model > Multivariate time series` menu. The screenshot is displayed in Figure 2.

The contents should be pretty much self-explanatory: the only notable difference with the ordinary way you use the `BEKK_Setup()` function is that by ticking the "Constant in mean" box, you don't have to put `const` among the list of mean regressors, since this is done automatically.

Upon successful completion, for models with up to 6 dependent variables, the `BEKK_ComboPlot()` function is called automatically and the corresponding plot is displayed.

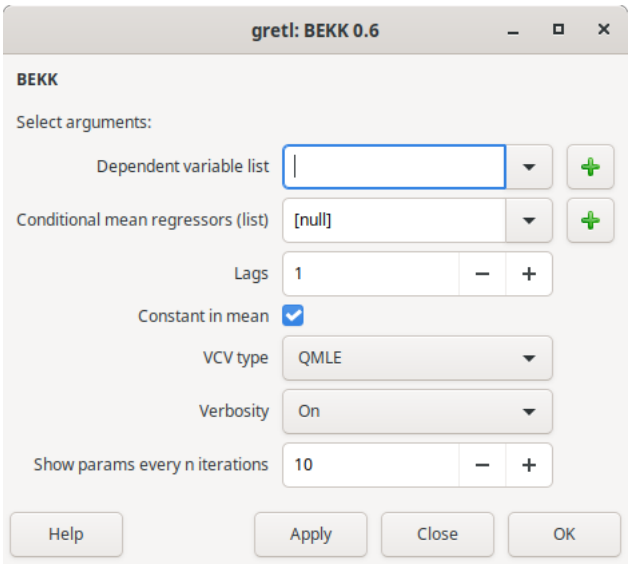

Figure 2: Graphical interface

4 List of functions

4.1 Main functions

`BEKK_Setup(list Y, scalar p, list X)`

Return type : `bundle`

`Y` : list with the dependent variables $\mathbf{y}_t$;

`p` : VAR order (default: 0);

`X` : list with the explanatory variables $\mathbf{x}_t$ (Default: `null`).

Sets up a BEKK model.

`BEKK_Estimate(bundle *mod, int verbose)`

Return type : `scalar`

`*mod` : a bundle created by `BEKK_Setup`, in pointer form;

`verbose` : verbosity level, from 0 to 2 (default: 0).

Estimates the model as per equations (1)– (3). Standard errors can be computed via OPG, inverse Hessian or QMLE (default). In order to choose between available options, set a scalar bundle member named "vcvtype" equal to 1, 2, or 3 before calling the function.

The function returns a scalar holding 0 after successful completion of estimation, and a non-zero value otherwise. If all went well, notable bundle elements that you might want to extract for further computations are:

T : sample size

n : number of series

coeff : coefficient vector

vcv : covariance matrix for coeff

mats : array of system matrices

ht : matrix with the covariance matrix sequence (T rows)

The `ht` matrix has T rows and $n(n+1)/2$ columns; each row contains the vech of Σ_t; therefore, to recover the matrix Σ_t for observation 100, you can use

```
matrix foo = unvech(MyMod.ht[100]')
```

assuming your model is in the bundle `MyMod`. The `mats` array contains the estimates for the system matrices:

1. : vertical concatenation of Π and the Φ matrices
2. : reserved for future use
3. : C
4. : A
5. : B

To extract residuals and predicted covariances, use the specialised functions listed in the next subsection.

`BEKK_Printout(bundle *mod)`

Return type : none

`*mod` : a model bundle, in pointer form;

Prints the estimates. Takes the model bundle (in pointer form) as its only parameter.

4.2 Auxiliary functions for item retrieval

Here are the post-estimation functions (in alphabetical order)

`BEKK_GetCorrel(bundle mod)`

Return type : list

`mod` : a vector with the ordinal numbers of the series to plot (optional, default=all)

This function takes one argument, the model bundle. It returns a list with the estimated conditional correlations; the variable names in the list follow the pattern `Cond_Corr_xx_yy`, where `xx` and `yy` are the variable numbers.

`BEKK_GetCovariance(bundle *mod, scalar x, scalar y)`

Return type : series

`*mod` : model bundle, in pointer form;

`i` : a scalar, ordinal number for first series (starting at 1);

`j` : a scalar, ordinal number for second series (starting at 1);

Returns a series with the estimated conditional covariance between series i and j.

`BEKK_GetLoglik(bundle *b)`

Return type : series

`*mod` : model bundle, in pointer form.

Returns a series with the estimated log-likelihood.

`BEKK_GetResids(bundle *b, scalar i, bool std)`

Return type : series

`*b` : model bundle, in pointer form.

`i` : a scalar, the ordinal number of the wanted variable in the dependent list

`std` : a Boolean optional flag to yield the standardised version (default = no)

Returns a series with the estimated residuals for one of the series in the model.

`BEKK_GetVariance(bundle *mod, scalar i)`

Return type : series

`*mod` : model bundle, in pointer form;

`i` : a scalar, ordinal number for the desired series (starting at 1);

Returns a series with the estimated conditional variance for one of the series in the model.

4.3 Plotting functions

Just one for now.

```
BEKK_ComboPlot(bundle mod, matrix in_sel, string in_dest)
```

Return type : none

`mod` : a vector with the ordinal numbers of the series to plot (optional, default=all);

`in_dest` : a string with the destination for the plot (optional, default=display);

Produces a grid plot with the estimated variance on the diagonal, estimated covariances below the diagonal and estimated correlations above. Returns nothing.

5 CHANGELOG:

0.1 -> 0.11 : get rid of = as a Boolean operator as per new policy

0.11 -> 0.12 : fix the `X == null` case

0.12 -> 0.2 : Add `GetResids`, update the JEL codes, use QML as default vcv type

0.2 -> 0.3 : Add `GetLoglik`, add GUI hook, remove some dead code

0.3 -> 0.31 : prettify output of conditional mean parameters and expand help a little

0.31 -> 0.32 : fix menu attachment (and bump required gretl version); handle non-convergences more gracefully.

0.32 -> 0.33 : (internal) get rid of eigengen() in favour of eigen(); no user-visible changes.

0.33 -> 0.5 : rename the package to BEKK (with consequent renaming of functions); also, rename the `Get*()` functions with a "BEKK" prefix to make possible clashes with other packages less likely. Use the commute() function internally if gretl version allows. Be a little more verbose in the help text. Bump version requirement to 2020c.

0.5 -> 0.6 : add the `BEKK_GetCorrel()` and `BEKK_ComboPlot()` functions; fix the sample script, and make it a little more interesting while at it (3 series instead of 2); switch this file to markdown format; ditch an unused element in GUI hook.

0.6 -> 0.7 : Raise gretl version requirement to 2023c; add pdf help; set $p = 0$ as the default in `BEKK_Setup`. Avoid using `BEKK_ComboPlot()` in the GUI interface for systems with more than 6 variables.

References

ENGLE, R. F. AND K. F. KRONER (1995): "Multivariate simultaneous generalized ARCH," *Econometric theory*, 11, 122–150.

LUCCHETTI, R. (2002): "Analytical score for multivariate GARCH models," *Computational Economics*, 19, 133–143.