Chapter 5

Multivariate GARCH: Basics

It didn't take long for GARCH models to make the jump from univariate to multivariate settings. In financial econometrics, it's rare to have only one asset of interest—if you're analyzing bonds, there are different maturities, if exchange rates, multiple currencies, and, of course, there are thousands of equities. Not only is the volatility for each likely to be described fairly well by a GARCH process, but within a class of assets, the movements are likely to be highly correlated. As a result, there would be expected to be substantial gains in (statistical) efficiency in modeling them jointly. In addition, having a time-varying estimate of the covariances would permit calculation of rolling optimal portfolio allocations.

There are *many* more variants of multivariate GARCH models than univariate. In some cases, this is driven by the need to answer particular questions. Many of the differences, though, are due to the difficulties in estimating the more general forms of multivariate GARCH models. You might notice that we really didn't say much about the technical options, such as initial guess values and algorithms, for estimating univariate models in Chapters 3 and 4. This is because the univariate log likelihood is quite well-behaved—even if you estimate a model with a data set which isn't well-described by a GARCH, you will generally get converged estimates without much fuss. That is far from true for multivariate GARCH models, so we will have to be much more careful, both in choice of a specific form for the model, and also in choosing the algorithm to use.

5.1 Preliminaries

Before you do anything else, do yourself a favor and graph the data. Make sure that it at least *looks* like it could be generated by a GARCH process. The data set we will work with here is the daily exchange rate data from section 3.11 of Enders (2010). This has nine years of daily observations on five exchange rates vs the US dollar, though we will work only with three of them: the Euro, pound and Swiss franc.[1] We read the data and convert the three of interest to returns with:

[1]The two we are omitting are the Australian and Canadian dollars.

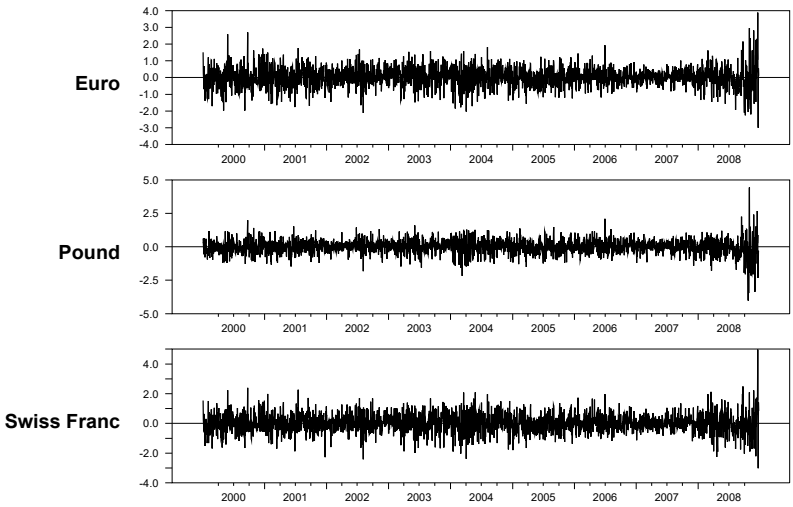

Figure 5.1: Exchange Rate Returns

```
open data "exrates(daily).xls"
calendar(d) 2000:1:3
data(format=xls,org=columns) 2000:01:03 2008:12:23 $
  aust euro pound sw ca
*
set reuro  = 100.0*log(euro/euro{1})
set rpound = 100.0*log(pound/pound{1})
set rsw    = 100.0*log(sw/sw{1})
```

This scales by 100 as described on page 5, which, in practice, is even more important for multivariate models because of the greater numerical difficulties they create. The following graphs the three return series, producing Figure 5.1.[2]

```
spgraph(vfields=3,ylabels=||"Euro","Pound","Swiss Franc"||)
 dofor r = reuro rpound rsw
   graph(picture="*.#")
   # r
 end dofor
spgraph(done)
```

What could you see at this point which might raise a flag that the GARCH model might be a mistake (at least across the sample in question)? A common error is including a part of the sample where a price was regulated, so the returns are zero for long stretches with occasional one-period spikes. This may be interesting from an economic standpoint, but you won't be able to include that (at least as an endogenous variable) in any sensible GARCH model. Data

[2] In this chapter's *Tips and Tricks* (Section 5.8.1), we'll see how to convert this into a better-looking graph.

Table 5.1: Lag Length Selection for Exchange Rates

VAR Lag Selection	
Lags	SBC/BIC
0	7243.00043*
1	7272.86486
2	7319.51443
3	7373.53640
4	7422.21572
5	7465.46548

which have visible variation only at one or two episodes will only fit a non-stationary GARCH model, which may be quite hard to interpret.

As with the univariate case, we need to decide upon a model for the mean. If Ω_{t-1} denotes the information available at time $t-1$, then a general structure for (the variance model in) a multivariate GARCH model is:

$$E\left(\mathbf{u}_t|\Omega_{t-1}\right) = 0$$
$$E\left(\mathbf{u}_t\mathbf{u}_t'|\Omega_{t-1}\right) \equiv \mathbf{H}_t = f(\mathbf{H}_{t-1}, \mathbf{H}_{t-2}, \ldots, \mathbf{u}_{t-1}, \mathbf{u}_{t-2}, \ldots)$$

Before we even concern ourselves with the form of f, note that the first condition means that the series is (at a minimum) a vector white noise—the residuals not only have to be serially uncorrelated individually, but they need to have zero correlation with the lags of the *other* components. An obvious model to entertain for eliminating joint autocorrelation is a low-order VAR. The vector analogue of the **@ARAutoLags** procedure from Section 3.3 is **@VARLagSelect**:

```
@varlagselect(crit=bic,lags=5)
# reuro rpound rsw
```

The minimum BIC lag length (Table 5.1) is 0, which is what is used in the Enders book. Hannan-Quinn (option CRIT=HQ) slightly favors 1 over 0, and Akaike (option CRIT=AIC) favors five and prefers any positive number of lags over 0.

As with **@ARAutoLags**, **@VARLagSelect** assumes homoscedastic residuals and so can't offer more than a rough guide to choice of lag length. And as before, we're better off starting small and testing the results for residual correlation. Thus, we will use the intercepts only as the mean model, which is the default for the **GARCH** instruction. However, for demonstration purposes, we will use the MODEL option on **GARCH**, which is what you would use for any more general type of mean model. The simplest way to define the MODEL when we have the same right-side variables in each equation is to use the **SYSTEM** definition instructions:

```
system(model=mvmean)
variables reuro rpound rsw
lags
det constant
end(system)
```

We left an empty LAGS instruction to show where you would put in VAR lags if you wanted or needed them.

We can do a preliminary test for multivariate ARCH effects using the procedure **@MVARCHTest**. Since this is mainly designed for diagnostics, it assumes that the input series are already (roughly) mean zero, that is, it doesn't subtract means itself. We can do the least squares estimates and test for ARCH with

```
estimate(resids=resids)
@mvarchtest
# resids
```

which produces the (unsurprising) result that lack of ARCH is overwhelming rejected:

```
Test for Multivariate ARCH
Statistic Degrees Signif
   370.11      36 0.00000
```

@MVARCHTest (from Hacker (2005)) works by running a multivariate regression of all unique combinations of $u_{it}u_{jt}$ on constant and all unique combinations of $u_{i,t-k}u_{j,t-k}$ for each k up to the number of lags tested (which is 1 by default, and controlled by the LAGS option) and testing the significance of the lag coefficients. Thus, the null is that the u series has a fixed covariance matrix against the alternative of *some* 2nd order dependence. $\mathbf{u}_t\mathbf{u}_t'$ is symmetric, so there are $n(n+1)/2$ unique elements in it, thus 6 for the case in question with $n = 3$. The degrees of freedom is the square of that times the number of lags, which can get quite large quite quickly. For diagnostic purposes, we will need to be a bit careful about relying too much on it alone, because some of the tested coefficients will be more likely to be informative than others—only 6 out of 36 will be "own" effects, testing second order correlation of $u_{it}u_{jt}$ with its own lag rather than a different i, j combination.

Again, note that (as with the univariate test) rejecting the null does *not* mean that a GARCH model is correct, only that a fixed covariance *isn't*. Even a change in the correlation structure without any overall changes in the variances can trigger a significant test.

5.2 GARCH Instruction

You use the same **GARCH** instruction for multivariate models as you do for univariate; however, there are a different set of options to deal with choice

of model, and the output information for variances and residuals is different. For the univariate model, we used the %RESIDS series for the residuals and the HSERIES option to get the estimated variances. Neither of those will be sufficient (and neither even works) with a multivariate model—the residuals will be an n vector at each point in time, and the variances will be a complete symmetric $n \times n$ matrix in order to capture the correlations as well as the variances. The options for getting the multivariate information are now RVECTORS for the residuals and HMATRICES for the covariance matrices. RVECTORS returns a SERIES of VECTORS and HMATRICES a SERIES of SYMMETRIC matrices. For instance, if we use the option HMATRICES=HH, then in a **SET** instruction or other formula, HH(T) is an $n \times n$ symmetric matrix: if we need the 1,1 element of that at time T, we use the double-subscripted expression HH(T)(1,1).

As we will see, there are many ways to set up a multivariate GARCH model. The main option for choosing the type is MV=*model type* . For illustration, we'll use MV=DIAG, which isn't really a useful model in practice. This ignores the covariances and models the variances separately using univariate methods. The point estimates should be almost identical to what you would get by estimating separate univariate models—the parameters are estimated jointly so none are considered converged until all of them are, which leads to very slight differences from one-at-a-time estimation. Assuming Normal residuals, the diagonal model can be written:

$$\begin{aligned} h_{ii,t} &= c_i + a_i u_{i,t}^2 + b_i h_{ii,t-1} \\ h_{ij,t} &= 0 \text{ if } i \neq j \\ \log L_t &= \text{const.} - \frac{1}{2}\log|\mathbf{H}_t| - \frac{1}{2}\mathbf{u}_t'\mathbf{H}_t^{-1}\mathbf{u}_t \end{aligned}$$

Because of the diagonality of $\mathbf{H}$, the log likelihood is just the sum of the log likelihoods across i. The instruction for estimating this (saving the residuals and $\mathbf{H}$ matrices) is:

```
garch(model=mvmean,mv=diag,p=1,q=1,rvectors=rd,hmatrices=hh)
```

You use the MODEL option to input the mean model that we defined earlier. Note that the process of estimating the model also sets the coefficients of this model to the GARCH estimates of the mean coefficients, so if we followed this by a **FORECAST**, it would use the **GARCH** estimates rather than the OLS estimates that we calculated first. Note that, because means-only is the default, we could also have estimated this with:

```
garch(mv=diag,p=1,q=1,rvectors=rd,hmatrices=hh) / reuro rpound rsw
```

This is the reason the range parameters come first on **GARCH**: to allow for the open-ended list of dependent variables in this form. The results are in Table 5.2.

Table 5.2: MV-GARCH Diagonal Model

MV-GARCH, Diagonal - Estimation by BFGS
Convergence in 46 Iterations. Final criterion was 0.0000071 <= 0.0000100
Daily(5) Data From 2000:01:04 To 2008:12:23
Usable Observations 2341
Log Likelihood -6015.5736

	Variable	Coeff	Std Error	T-Stat	Signif
1.	Constant	0.0240	0.0114	2.0974	0.0360
2.	Constant	0.0114	0.0107	1.0680	0.2855
3.	Constant	0.0195	0.0130	1.5033	0.1327
4.	C(1)	0.0006	0.0006	1.0818	0.2793
5.	C(2)	0.0035	0.0014	2.5913	0.0096
6.	C(3)	0.0010	0.0008	1.2795	0.2007
7.	A(1)	0.0325	0.0049	6.6572	0.0000
8.	A(2)	0.0483	0.0077	6.2504	0.0000
9.	A(3)	0.0276	0.0045	6.0641	0.0000
10.	B(1)	0.9672	0.0052	187.0732	0.0000
11.	B(2)	0.9406	0.0106	88.8573	0.0000
12.	B(3)	0.9713	0.0050	195.3141	0.0000

First, note that the model being estimated is in the first line of the output—here "MV-GARCH Diagonal". Rather than just `C`, `A` and `B`, the GARCH coefficients are `C(1)`, `C(2)`, `C(3)`, and similarly for `A` and `B`. The `C`, `A` and `B` prefixes will be used for the constant, lagged squared residual and lagged variance terms in all multivariate GARCH models, but the subscripting will change depending upon the form of the multivariate model. The mean model coefficients will always be at the top, grouped by equation.

5.3 Diagnostics

The diagnostics for the univariate GARCH models were based upon the standardized residuals, which should be (if the model is correct) serially uncorrelated and homoscedastic. The residuals from the multivariate model should also be serially uncorrelated, and show no remaining ARCH effects. However, there are two possible approaches to testing these: using univariate tests or multivariate tests. The latter is what is actually implied by the assumptions of the model, but the multivariate tests are both more complicated and not as precisely defined.

The univariate tests would just apply the same pair of tests (for serial correlation on standardized residuals and their squares) to each of the variables. A quick, but not very pretty, set of tests can be done with:

```
do i=1,%nvar
   set ustd  = rd(t)(i)/sqrt(hh(t)(i,i))
   set ustdsq = ustd^2
   @regcorrs(number=10,nocrits,nograph,qstat) ustd
   compute q1=%cdstat,q1signif=%signif
   @regcorrs(number=10,nocrits,nograph,qstat,dfc=2) ustdsq
   compute q2=%cdstat,q2signif=%signif
   disp q1 q1signif q2 q2signif
end do i
```

%NVAR is defined by **GARCH** as the number of dependent variables, so this will work for any size GARCH model as long as you use RD and HH for the RVECTORS and HMATRICES options. The first **SET** instruction shows how to divide component I of the residuals by its corresponding (time-varying) standard deviation.

In *Tips and Tricks* (Section 5.8.2), we'll show how to convert this to a nicer table. What this simple version produces is:

```
     8.73373       0.55755      16.30334       0.03824
    13.55638       0.19420       9.68835       0.28758
     4.44765       0.92490       9.14915       0.32987
```

The only one of the six numbers that is significant at conventional levels is the one on the squares for the first variable (the Euro, row 1, third number is the statistic and fourth is the significance level) but with over 2000 data points, that's unlikely to be anything fixable with any simple change to the model (Appendix C).

These, however, are just univariate diagnostics, and we have a model which is supposed to produce *vector* white noise. The problem in moving to multivariate diagnostics is dealing with the time-varying covariance matrices. The univariate standardized residuals give three series which are (approximately) each variance one, but they ignore the fact that the correlations vary from time period to time period. Instead, we need a matrix-standardized set of residuals, which transform to (approximately) the identity matrix. However, that standardization isn't unique, as if we take *any* sequence of matrices $\mathbf{G}_t$ such that $\mathbf{G}_t'\mathbf{G}_t = \mathbf{H}_t^{-1}$, then $E\left(\mathbf{G}_t\mathbf{u}_t\mathbf{u}_t'\mathbf{G}_t'\right) = \mathbf{I}$. Once we have two or more variables, there are an infinite number of such "square root" matrices (at each t) and each sequence of $\mathbf{G}_t\mathbf{u}_t$ will produce a different test statistic for correlation—whether these are likely to be qualitatively different is unclear.

The simplest such G to compute is the inverse of the Cholesky factor of H. However, a better choice under the circumstances is to use the one derived from an eigen decomposition of H, which will be independent of the order in which you list the dependent variables. Unlike the univariate tests, where we could generate the standardized residuals one-at-a-time, the multivariate tests require that we have the full set of generated residuals simultaneously. They also need to be organized differently: as a VECTOR of SERIES rather than a SERIES like RD. You generate these with the STDRESIDS option on **GARCH**. The choice of

G is controlled by the FACTORBY option, which is either FACTORBY=CHOLESKY (the default) or FACTORBY=EIGEN.

Adding the options to do the eigen standardization to the **GARCH** instruction gives us:

```
garch(model=mvmean,mv=diag,p=1,q=1,$
  rvectors=rd,hmatrices=hh,stdresids=rstd,factorby=eigen)
```

The procedures designed for the diagnostics test on this are **@MVQSTAT** and **@MVARCHTest** that we've already used. We compute a 10-lag Q-statistic with:

```
@mvqstat(lags=10)
# rstd
```

which gives us

```
Multivariate Q(10)=     195.71117
Significance Level as Chi-Squared(90)=  8.24580e-010
```

Note that if we were using a VAR with lags, we would need to include the DFC (Degrees for Freedom Correction) option on **@MVQSTAT** with correction $n^2 \times L$ where L is the number of VAR lags.

The degrees of freedom on the test will be n^2 times the number of tested lags (here 10). The Q test looks for correlation between $u_{i,t}$ and $u_{j,t-k}$ for all i, j combinations and for each tested lag k.

Despite the fact that the individual Q statistics were insignificant, this is significant beyond any doubt. So what happened? The problem isn't that the residuals are actually serially correlated, it's that the variance model is so completely wrong that the standardization process produces nonsense. MV=DIAG assumes the off-diagonal elements of $\mathbf{H}_t$ are zero, and that shows up in the HH matrices. The sample residuals, however, are quite strongly contemporaneously correlated, so the standardization is simply wrong.

The test for ARCH (here with two lags) confirms the fact that the model estimated is inadequate:

```
@mvarchtest(lags=2)
# rstd
```

gives us

```
Test for Multivariate ARCH
Statistic Degrees Signif
   465.60      72 0.00000
```

It's important to recognize the difference in reading the diagnostics: in the univariate case, an incorrect GARCH model will rarely produce a highly significant finding of serial correlation in the mean when the mean model is, in fact, adequate, so if we get the result that the Q is significant, the fix will generally be to

change the mean model. The multivariate statistics are much more dependent upon a good variance model, even for testing the adequacy of the mean, because the standardization depends not just on scale (which the diagonal model gets right) but on "shape".

5.4 VECH, DVECH and BEKK Models

The first important paper to use a multivariate GARCH model was Bollerslev, Engle, and Wooldridge (1988). This actually did a multivariate GARCH-M, but for now we'll just discuss the variance model in isolation. The first form that they introduce for that is what is now known as the VECH or Full-VECH, after the "vech" operator which takes an $n \times n$ symmetric matrix and converts it to an $n(n+1)/2$ vector by eliminating the duplicated entries. A VECH 1,1 model is written:

$$vech(\mathbf{H}_t) = \mathbf{C} + \mathbf{A}\, vech(\mathbf{u}_{t-1}\mathbf{u}_{t-1}') + \mathbf{B}\, vech(\mathbf{H}_{t-1}) \tag{5.1}$$

In this form, C is a vector which should be the $vech$ of a positive semi-definite matrix. A and B are full $n(n+1)/2 \times n(n+1)/2$ matrices. Bollerslev, Engle and Wooldridge didn't estimate this model, and it is rarely used. One reason is fairly obvious—the number of free parameters goes up very fast with n. Even for our example with $n = 3$, we have 78 in the variance model (6 in C, $36 = 6^2$ for each of A and B). That's a very large number of free parameters for non-linear estimation. Also, this has an unrestricted coefficient on each of the elements of $\mathbf{H}_{t-1}$ for explaining each element of $\mathbf{H}_t$. If the shocks are highly correlated (which is what we are expecting), then the variances and covariances will tend to move together. Thus (with our $n = 3$ data set), each element of $\mathbf{H}_t$ has free coefficients on six different, and relatively similar, pieces of information. So not only is it a high-dimension set of non-linear parameters, but many of them are poorly determined from the data.

The VECH model is estimated with RATS by using the `MV=VECH` option. However, you have very little chance of getting a model with anything more than two variables to estimate properly without a great deal of experimentation with guess values.

However, even if the unrestricted (5.1) isn't very useful for estimation, it is (with restrictions) useful for forecasting and similar calculations. If we can cast a GARCH recursion in this form, then, just as with the univariate model, the forecasts can be generated as

$$\begin{aligned} vech(\hat{\mathbf{H}}_{t+1}) &= \mathbf{C} + \mathbf{A}\, vech(\mathbf{u}_t\mathbf{u}_t') + \mathbf{B}\, vech(\mathbf{H}_t) \\ vech(\hat{\mathbf{H}}_{t+k}) &= \mathbf{C} + (\mathbf{A}+\mathbf{B})\, vech(\hat{\mathbf{H}}_{t+k-1}) \end{aligned} \tag{5.2}$$

that is, the first period out-of-sample forecast uses the computed residuals and variance, while all steps after that are a matrix difference equation in the forecast variance/covariance matrix.

Two important variance models are restrictions on the VECH: the diagonal or DVECH model and the BEKK. Several others (CC and DCC) are not. In this section, we will deal with the restricted VECH forms.

5.4.1 Diagonal VECH (Standard) Model

The model estimated by Bollerslev, Engle and Wooldridge was the diagonal VECH or DVECH model, which, if you don't state otherwise, is the standard for a multivariate GARCH, and is the default for the RATS **GARCH** instruction. This assumes that $\mathbf{A}$ and $\mathbf{B}$ in (5.1) are diagonal. Another (more compact) way to write this is

$$\mathbf{H}_t = \mathbf{C} + \mathbf{A} \circ \left(\mathbf{u}_{t-1}\mathbf{u}_{t-1}'\right) + \mathbf{B} \circ \mathbf{H}_{t-1} \tag{5.3}$$

where $\circ$ represents the *Hadamard* or elementwise product,[3] and $\mathbf{C}$, $\mathbf{A}$ and $\mathbf{B}$ are now $n \times n$ symmetric matrices. Written out, this means

$$h_{ij,t} = c_{ij} + a_{ij}u_{i,t-1}u_{j,t-1} + b_{ij}h_{ij,t-1} \tag{5.4}$$

This greatly decreases the number of free parameters: we still have six in $\mathbf{C}$, but only six each in $\mathbf{A}$ and $\mathbf{B}$.

Despite being a much smaller model, there are still some numerical problems that must be overcome in estimating this. (5.4) describes "independent" recursions on each of the components of the covariance matrix. For this to produce a computable log likelihood, each $\mathbf{H}_t$ matrix must be positive-definite, but there's nothing in the structure of (5.4) that will enforce that. If an off-diagonal recursion has a higher persistence than its corresponding diagonal elements, it's possible for the recursion to stray into non-positive-definite territory if the correlations among the residuals are high. The parameters will never actually go into an area where the log likelihood isn't computable, but if the estimation process ever gets close to a singular $\mathbf{H}$ at *any* data point, then it may be hard to move away from that test set of parameters, because the predictions for the log likelihood from the gradient can be quite inaccurate due to the function being nearly non-differentiable.

This is a problem that's specific to the two derivative-based optimization algorithms: BFGS and BHHH. BHHH is almost *never* a good choice as a basic estimation algorithm for GARCH models—its prediction of the curvature of the log likelihood is likely to be good only when you're fairly close to the optimum, and getting *to* the optimum is the challenge. BFGS is generally more successful, but it does have the problem that it estimates the curvature (and thus the covariance matrix of the parameter estimates) using an update method which will give a different answer for different initial guess values. In the GARCH literature, it's not uncommon for standard errors to be reported as generated by either BHHH, or corrected for possible misspecification (in RATS done with the

[3] $\mathbf{C} = \mathbf{A} \circ \mathbf{B} \Rightarrow c_{ij} = a_{ij}b_{ij}$
If you need a Hadamard product in RATS as part of a calculation, you can do it with `A.*B`

ROBUSTERRORS option). In practice, the latter is more conservative and should be preferred.

The recommendation that we make for estimating multivariate GARCH models is to use a combination of preliminary "simplex" iterations, followed by BFGS for doing the final estimates. The problem with starting off with BFGS from the initial guesses is that BFGS also has some difficulty with the function curvature at the guess values since it builds up its estimate of the inverse Hessian indirectly by seeing how the gradient changes from iteration to iteration. A poor estimate of the curvature can lead to some wildly inaccurate moves in the early iterations. Because the log likelihood can have a very odd shape, it's possible to make a big move in the wrong direction and then get stuck on the wrong "hill"—one that doesn't include the global optimum.

We generally recommend somewhere between 5 and 20 simplex "iterations".[4] We frequently see people who think that if 20 simplex iterations is good, then 200 must be better. That's rarely the case. The simplex method accomplishes quite a bit less in an "iteration" worth of work than one of the climbing algorithms, which is intentional—it relies upon much weaker assumptions about the behavior of the function being optimized and so doesn't (and *can't*) make very quick moves up the "hill". Simplex should *eventually* get to the same optimum, but it can take thousands of iterations to do so with a parameter set of this size—20 vs 200 probably will make little difference. What the first 20 iterations do is to move the parameter set (cautiously) off the guess values in what is likely to be the correct direction. From there, the more standard algorithm can take over.

The estimation code is in Example **5.2**. This uses the same data and mean model as Example **5.1**. The instruction for estimating the DVECH model using the recommended options is

```
garch(model=mvmean,rvectors=rd,hmatrices=hh,robusterrors,$
  pmethod=simplex,piters=20,method=bfgs,iters=500)
```

which produces Table 5.3. The MV option isn't necessary, since this model is the default, but it is called MV=STANDARD. This model nests the MV=DIAG model from Table 5.2 (the diagonal model has all the off-diagonal coefficients zeroed), so the log likelihood must be higher here, but being higher by over 3000 is quite a difference—the diagonal model is obviously thoroughly misspecified.

The A, B and C coefficients are labeled as shown in (5.4) and they are in the rowwise order of the lower triangle that is used for SYMMETRIC matrices in

[4] Iterations is in quotes because, in RATS, a simplex iteration is counted as a set of calculations with roughly the same number of function evaluations as an iteration in one of the derivative-based algorithms. Simplex is very different from "climbing" algorithms, as, rather than explicitly looking for directions of increase, it spends most of its effort eliminating directions of decrease.

Table 5.3: Multivariate GARCH-Diagonal VECH

MV-GARCH - Estimation by BFGS
Convergence in 194 Iterations. Final criterion was 0.0000000 <= 0.0000100
With Heteroscedasticity/Misspecification Adjusted Standard Errors
Daily(5) Data From 2000:01:04 To 2008:12:23
Usable Observations 2341
Log Likelihood -2758.8148

	Variable	Coeff	Std Error	T-Stat	Signif
1.	Constant	0.0247	0.0103	2.4017	0.0163
2.	Constant	0.0149	0.0100	1.4936	0.1353
3.	Constant	0.0199	0.0123	1.6154	0.1062
4.	C(1,1)	0.0033	0.0017	1.9282	0.0538
5.	C(2,1)	0.0026	0.0017	1.5065	0.1319
6.	C(2,2)	0.0033	0.0022	1.4992	0.1338
7.	C(3,1)	0.0035	0.0018	1.9824	0.0474
8.	C(3,2)	0.0022	0.0012	1.8255	0.0679
9.	C(3,3)	0.0047	0.0023	2.0432	0.0410
10.	A(1,1)	0.0413	0.0104	3.9534	0.0001
11.	A(2,1)	0.0332	0.0092	3.5961	0.0003
12.	A(2,2)	0.0389	0.0091	4.2776	0.0000
13.	A(3,1)	0.0387	0.0099	3.9140	0.0001
14.	A(3,2)	0.0308	0.0064	4.8414	0.0000
15.	A(3,3)	0.0390	0.0102	3.8068	0.0001
16.	B(1,1)	0.9508	0.0147	64.7672	0.0000
17.	B(2,1)	0.9565	0.0165	57.9902	0.0000
18.	B(2,2)	0.9513	0.0156	60.9383	0.0000
19.	B(3,1)	0.9530	0.0141	67.6920	0.0000
20.	B(3,2)	0.9609	0.0108	88.7553	0.0000
21.	B(3,3)	0.9514	0.0145	65.6489	0.0000

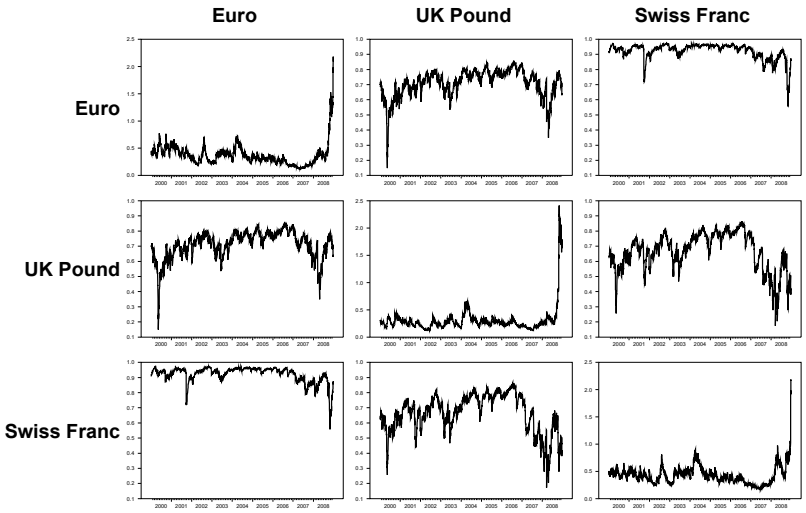

Figure 5.2: Variances (Diagonal) and Correlations (Off-Diagonal)

RATS. If we want to extract the A, B and C matrices from (5.3), we can do it with

```
dec vect meanparms(%nregmean)
dec symm cdvech(%nvar,%nvar) advech(%nvar,%nvar) bdvech(%nvar,%nvar)
nonlin(parmset=dvech) meanparms cdvech advech bdvech
compute %parmspoke(dvech,%beta)
```

which is an extension to a (multivariate) DVECH model of the use of the PARMSET and %PARMSPOKE function. If we extract these and add them, we'll get the persistence on an element by element basis:

```
    0.99210
    0.98966     0.99014
    0.99169     0.99172     0.99043
```

Note that these are all very close to 1, and the 3,2 element is, in fact, larger than the 2,2 and 3,3 elements, which is the type of behavior that *could* cause problems with the positive-definite boundary.

We can extract the time-varying correlations and the time-varying variances using the following, which organizes them into a SYMMETRIC of SERIES, with variances on the diagonals and correlations on the off-diagonals. The simplest way to get the correlations is to use %CVTOCORR(C), which returns the correlation matrix formed from the covariance matrix C, as we can just pull elements out of that. Note that this works for *any* type of multivariate GARCH model.

```
dec symm[series] hhx(%nvar,%nvar)
do i=1,%nvar
   set hhx(i,i) = hh(t)(i,i)
   do j=1,i-1
      set hhx(i,j) = %cvtocorr(hh(t))(i,j)
   end do j
end do i
```

We can use some of the ideas from Section 5.8.1 to present this information in an interesting way (Figure 5.2) with:

```
table / hhx(2,1) hhx(3,1) hhx(3,2)
compute corrmin=%minimum
table / hhx(1,1) hhx(2,2) hhx(3,3)
compute varmax=%maximum
*

spgraph(vfields=%nvar,hfields=%nvar,$
    xlabels=longlabel,ylabels=longlabel)
do i=1,%nvar
   do j=1,%nvar
   if i==j {
      graph(row=i,col=i,maximum=varmax)
      # hhx(i,i)
   }
   else {
      graph(row=i,col=j,maximum=1.0,min=corrmin)
      # hhx(i,j)
   }
   end do j
end do i
spgraph(done)
```

This puts the variances in the diagonal fields and the correlations in the off-diagonals. To avoid a misleading impression, this forces all the correlations to use the same range, For instance, without that, it wouldn't be clear without a careful look at the scale that the Swiss Franc-Euro correlation is so extremely high across almost the entire range if its values (roughly .6 to 1) are spread across the full box, and the UK correlations (roughly .1 to .9) are graphed in the same sized box.

5.4.2 BEKK Model

There are a few drawbacks to the DVECH model. First, as we described above, it can have some numerical issues because the parameterization doesn't enforce positive-definiteness.[5] Second, it doesn't allow for certain types of more complicated interactions among the variables; for instance, one interesting possibility is that shocks in one market could have a "spillover" effect on another (Section 5.5). That's precluded by the structure of the DVECH where the only thing that determines the variance of one series is its own shocks.

To allow a greater range of interactions (though not as general as the VECH), while also enforcing positive-definiteness by construction, Engle and Kroner (1995) proposed what is now known as the BEKK model, after the four authors of an earlier working paper. Note that this does *not* nest the DVECH, so even though it has a few additional parameters, it doesn't necessarily fit better.

For a 1,1 model, the BEKK recursion is

$$\mathbf{H}_t = \mathbf{C}\mathbf{C}' + \mathbf{A}'\mathbf{u}_{t-1}\mathbf{u}_{t-1}'\mathbf{A} + \mathbf{B}'\mathbf{H}_{t-1}\mathbf{B} \tag{5.5}$$

where C is now a lower triangular matrix and A and B are general $n \times n$ matrices. The last two terms could also have the transpose on the post-multiplying matrix rather than the pre-multiplying one—we're following the original Engle and Kroner formula, which is the one most commonly used. By construction, this is positive semi-definite regardless of the values of the parameters, and, in practice, will maintain positive definiteness as long as the B or C is full rank.[6] C has the same number of parameters as it does in the DVECH, just in a different form. A and B now have n^2 rather than $n(n+1)/2$ and so will always have more free parameters than the corresponding matrices in the DVECH: with $n = 3$, there are 9 for each rather than 6.

That this is a restricted version of the VECH form (5.1) is clear if you look at the expansions of the matrix multiplications. In fact, if we write (5.1) in a "vec" form (which includes the full $n \times n$ matrix including duplicates, stacked by columns), you get

$$vec(\mathbf{H}_t) = vec(\mathbf{C}\mathbf{C}') + \left(\mathbf{A}' \otimes \mathbf{A}'\right) vec(\mathbf{u}_{t-1}\mathbf{u}_{t-1}') + \left(\mathbf{B}' \otimes \mathbf{B}'\right) vec(\mathbf{H}_{t-1}) \tag{5.6}$$

which is easier to analyze for most purposes than the equivalent VECH form. That the BEKK *can't* be a generalization of DVECH is clear from the fact that $\mathbf{A}'\mathbf{u}_{t-1}\mathbf{u}_{t-1}'\mathbf{A}$ is rank one and the analogous term in the DVECH will be full rank for almost any set of parameters.

[5] Again, sample estimates will *always* give positive-definite matrices for every entry because the likelihood isn't computable otherwise. However, not all sets of parameters will do that, and the boundary between parameters that do and parameters that don't can be very complicated.

[6] The lagged $\mathbf{u}$ term is rank (at most) one, since it's the outer product of the n vector $\mathbf{A}'\mathbf{u}_{t-1}$ and so isn't as critical for maintaining the full rank.

Table 5.4: Multivariate GARCH-BEKK Estimates

MV-GARCH, BEKK - Estimation by BFGS
Convergence in 172 Iterations. Final criterion was 0.0000000 <= 0.0000100
With Heteroscedasticity/Misspecification Adjusted Standard Errors
Daily(5) Data From 2000:01:04 To 2008:12:23
Usable Observations 2341
Log Likelihood -2727.5812

	Variable	Coeff	Std Error	T-Stat	Signif
1.	Constant	0.0286	0.0115	2.5000	0.0124
2.	Constant	0.0174	0.0109	1.5937	0.1110
3.	Constant	0.0217	0.0129	1.6845	0.0921
4.	C(1,1)	0.0391	0.0094	4.1639	0.0000
5.	C(2,1)	0.0505	0.0105	4.8315	0.0000
6.	C(2,2)	-0.0055	0.0089	-0.6103	0.5417
7.	C(3,1)	0.0230	0.0114	2.0098	0.0445
8.	C(3,2)	-0.0277	0.0086	-3.2393	0.0012
9.	C(3,3)	0.0007	0.0064	0.1144	0.9089
10.	A(1,1)	0.2369	0.0621	3.8180	0.0001
11.	A(1,2)	0.1308	0.0660	1.9837	0.0473
12.	A(1,3)	-0.0862	0.0880	-0.9795	0.3273
13.	A(2,1)	0.0491	0.0289	1.7015	0.0888
14.	A(2,2)	0.1920	0.0318	6.0459	0.0000
15.	A(2,3)	0.0285	0.0287	0.9941	0.3202
16.	A(3,1)	-0.1201	0.0555	-2.1641	0.0305
17.	A(3,2)	-0.1323	0.0554	-2.3898	0.0169
18.	A(3,3)	0.1824	0.0890	2.0499	0.0404
19.	B(1,1)	0.9582	0.0161	59.4924	0.0000
20.	B(1,2)	-0.0254	0.0187	-1.3555	0.1753
21.	B(1,3)	0.0137	0.0268	0.5116	0.6089
22.	B(2,1)	-0.0208	0.0051	-4.0500	0.0001
23.	B(2,2)	0.9707	0.0063	154.5004	0.0000
24.	B(2,3)	-0.0091	0.0078	-1.1661	0.2436
25.	B(3,1)	0.0402	0.0161	2.4943	0.0126
26.	B(3,2)	0.0320	0.0191	1.6765	0.0936
27.	B(3,3)	0.9839	0.0288	34.1251	0.0000

To estimate, use the **GARCH** instruction with the option MV=BEKK. This is from Example **5.3**, producing Table 5.4.

```
garch(model=mvmean,mv=bekk,robusterrors,pmethod=simplex,piters=20,$
   method=bfgs,iters=500,rvectors=rd,hmatrices=hh,$
   stdresids=rstd,factorby=eigen)
```

The log likelihood is quite a bit higher than in Table 5.3. The models don't nest so we can't do a formal likelihood ratio test, but the difference is large enough that the BEKK model would be preferred even for a stringent information criterion like BIC.

We often get questions about the number of negative parameters in the typical BEKK output. One thing to note is that the BEKK model isn't global identified—

you get exactly the same fit if you change the sign of the entire A or B matrix, or even any column of C. However, the guess values used by **GARCH** will steer it in the direction of positive "own" contributions, so it would be very rare that you get the parameters with the opposite from the expected set of signs.

If you first look at the C coefficients, you'll note that `C(2,2)` is (slightly) negative and `C(3,3)` is barely positive. Because this is a factor of the variance intercept (rather than the variance intercept itself) the coefficients other than the 1,1 don't have simple interpretations. However, taken as a whole, these mean that the variance intercept matrix is close to being rank two.

It's easier to interpret the values in the A matrix. $\mathbf{A}'\mathbf{u}_{t-1}$ is an n vector, call it $\mathbf{v}_{t-1}$. The contribution to the variance at t is $\mathbf{v}_{t-1}\mathbf{v}'_{t-1}$, which means that the squares of the elements of v will be the contributions to the variances. To have "spillover" effects, so that shocks in $\mathbf{u}_j$ directly affect the variance of i, v will have to be a linear combination of the different components of u. Negative coefficients in the off-diagonals of $\mathbf{A}'$ mean that the variance is affected more when the shocks move in opposite directions than when they move in the same direction, which probably isn't unreasonable in many situations. Here, we see that the most statistically significant spillovers are from 3 (Swiss Franc) to the other two (`A(3,1)` and `A(3,2)` coefficients) and these *do* have negative signs.

Another common question (which doesn't apply here) is how it's possible for the off-diagonals in the A and B matrices to be larger than the diagonals, since one would expect that the "own" effect would be dominant. However, the values of the coefficients are sensitive to the scales of the variables, since nothing in the recursion is standardized to a common variance. If you multiply component i by .01 relative to j, its residuals also go down by a factor of .01, so the coefficient a_{ij} which applies u_i to the variance of j has to go *up* by a factor of 100. Rescaling a variable keeps the *diagonals* of A and B the same, but forces a change in scale of the off-diagonals. Even without asymmetrical scalings, the tendency will be for (relatively) higher variance series to have lower off-diagonal coefficients than lower variance series.

You can convert the coded up BEKK coefficients to the equivalent VECH representation with the procedure **@MVGARCHtoVECH**, here with

```
@MVGARCHToVECH(mv=bekk)
```

This creates the matrices `%%VECH_C`, `%%VECH_A` and `%%VECH_B`. With $n = 3$, the first will be a 6-vector, the others 6×6 matrices. We can take a closer look at the implied A in the VECH representation with:

```
disp ##.### %%vech_a
```

which gives us

```
 0.056  0.023  0.002 -0.057 -0.012  0.014
 0.031  0.052  0.009 -0.047 -0.030  0.016
 0.017  0.050  0.037 -0.035 -0.051  0.018
-0.020  0.003  0.001  0.054  0.006 -0.022
-0.011 -0.013  0.005  0.035  0.031 -0.024
 0.007 -0.005  0.001 -0.031  0.010  0.033
```

The "vech" operator is by the rows of the lower triangle, so the elements (in each direction) are in the order (1,1),(2,1),(2,2), etc. The variances (and squared residuals) are thus in positions 1, 3 and 6.

The BEKK estimates give rise to a stable recursion if the sum of the VECH A and B matrices has eigenvalues less than one. Since the eigenvalues could be complex, we can compute and display those with:

```
eigen(cvalues=cv) %%vech_a+%%vech_b
disp ##.### cv
```

and, as we can see, these are just barely on the stable side:

```
( 0.998,-0.000) ( 0.994, 0.003) ( 0.994,-0.003)
( 0.980,-0.015) ( 0.980, 0.015) ( 0.974, 0.000)
```

The multivariate diagnostics from Section 5.3 give us

```
Multivariate Q(10)=     109.59742
Significance Level as Chi-Squared(90)=       0.07849

Test for Multivariate ARCH
Statistic Degrees Signif
   107.91      72 0.00394
```

We would like the test for residual ARCH to be a *bit* better than this, but, in practice, with a data set this size (2300 observations), this isn't unreasonable.

Some people recommend reporting BEKK estimates in the *vech* form with standard errors (which can be computed using the delta method from Appendix D). This seems to be a step backwards, particularly with regards to the coefficients for A where the *vech* form is a reduced form to the more structural BEKK. If you are required to do this by a referee, you can use **SUMMARIZE** using instructions like:

```
dec vect means(%nregmean)
dec packed cx(%nvar,%nvar)
dec rect ax(%nvar,%nvar) bx(%nvar,%nvar)
nonlin(parmset=garchparms) means cx ax bx

summarize(parmset=garchparms) bx(1,1)^2
summarize(parmset=garchparms) bx(1,1)*bx(1,2)*2.0
summarize(parmset=garchparms) bx(1,2)^2
```

The first four lines decompose the estimated parameters into the separate matrices used by a BEKK model. The last three compute the standard errors for the first three elements of the first row of the VECH B matrix. Calculating

standard errors for the full matrix requires the following, which has quite a bit of "bookkeeping" to deal with what's basically four level subscripting, since each direction in the matrix represents a flattening of a symmetric matrix:

```
compute ncomp=%nvar*(%nvar+1)/2
dec rect %%vech_bse(ncomp,ncomp)
do m=1,ncomp
   do n=1,ncomp
      compute i=%symmrow(m),j=%symmcol(m),$
              k=%symmrow(n),l=%symmcol(n)
      if k==l {
         summarize(noprint,parmset=garchparms) bx(i,k)*bx(j,l)
         compute %%vech_bse(m,n)=sqrt(%varlc)
      }
      else {
         summarize(noprint,parmset=garchparms) $
             bx(i,k)*bx(j,l)+bx(i,l)*bx(j,k)
         compute %%vech_bse(m,n)=sqrt(%varlc)
      }
   end do n
end do m
```

The values and standard errors can be displayed in a matrix form (Table 5.5) with[7]

```
report(action=define,title="VECH-B from BEKK")
do j=1,ncomp
   report(atrow=1,atcol=j+1,align=center) $
       %string(%symmrow(j))+","+%symmcol(j)
end do i
do i=1,ncomp
   report(atrow=2*i,atcol=1) %string(%symmrow(i))+","+%symmcol(i)
   do j=1,ncomp
      report(atrow=2*i,atcol=j+1) %%vech_b(i,j)
      report(atrow=2*i+1,atcol=j+1,special=parens) %%vech_bse(i,j)
   end do j
end do i
report(action=format,atrow=2,picture="##.####",align=decimal)
report(action=show)
```

5.5 Spillover

"Spillover" in the context of multivariate GARCH models is a rather vaguely defined term. In general, the term spillover in economics refers to an event producing an effect elsewhere despite there being no obvious connection. However,

[7] It's not clear what this really adds.

Table 5.5: VECH-B from BEKK

	1,1	2,1	2,2	3,1	3,2	3,3
1,1	0.9180	−0.0398	0.0004	0.0773	−0.0017	0.0016
	(0.0248)	(0.0112)	(0.0002)	(0.0242)	(0.0008)	(0.0011)
2,1	−0.0245	0.9306	−0.0202	0.0298	0.0385	0.0013
	(0.0141)	(0.0142)	(0.0056)	(0.0129)	(0.0124)	(0.0010)
2,2	0.0007	−0.0496	0.9423	−0.0016	0.0624	0.0010
	(0.0008)	(0.0292)	(0.0130)	(0.0017)	(0.0285)	(0.0009)
3,1	0.0131	−0.0091	0.0002	0.9432	−0.0208	0.0397
	(0.0212)	(0.0070)	(0.0002)	(0.0148)	(0.0061)	(0.0137)
3,2	−0.0003	0.0135	−0.0089	−0.0247	0.9548	0.0316
	(0.0005)	(0.0211)	(0.0073)	(0.0154)	(0.0218)	(0.0150)
3,3	0.0002	−0.0003	0.0001	0.0269	−0.0180	0.9681
	(0.0006)	(0.0003)	(0.0001)	(0.0426)	(0.0152)	(0.0458)

it's hard to argue that there is no obvious connection between, for instance, two exchange rates (which presumably have the same numeraire currency, and thus are both directly affected by any event which touches the numeraire). Spillover has come to mean a model-based "Granger-style" causality, either in the mean or (more commonly) in the variance, that is a question of whether a set of coefficients on "other" variables are zero.

As we described earlier, the DVECH model doesn't allow for spillover/causality in the variance because the variance for each series depends only upon its own lagged variance and its own lagged (squared) residuals. However, BEKK does, so we'll look more carefully at the results from it.

One important thing to note is that, as with standard Granger causality tests, when you have three or more variables (we have three in this example), the exclusion of a single variable from a single equation doesn't really tell you much—a shock to variable Z can affect variable X indirectly if a shock to Z affects Y's variance and Y's variance affects X's. Thus, tests should really be of *block* exclusions, either Z affects neither Y nor X, or X is not affected by either Z nor Y. Example **5.4** does tests for causality (spillover), both in the mean and in the variance. These are all done as "Wald tests", which are most easily set up using the Regression Tests wizard (the "Exclusion Tests" choice within that) after running the GARCH model.

This time the GARCH model will be run on a one-lag VAR:

```
system(model=mvmean)
variables reuro rpound rsw
lags 1
det constant
end(system)
garch(model=mvmean,mv=bekk,robusterrors,pmethod=simplex,piters=20,$
   method=bfgs,iters=500,rvectors=rd,hmatrices=hh)
```

The first test is for exogeneity in the mean for all the variables (basically, that the mean model could have been separate autoregressions on each variable):

```
test(zeros,title="Test of Exogeneity in Mean of All Variables")
# 2 3 5 7 9 10
```

which rather strongly rejects the null:

```
Test of Exogeneity of All Variables
Chi-Squared(6)=     29.165801 or F(6,*)=      4.86097 with Significance Level 0.00005660
```

Note, however, that because these are all exchange rates in terms of the US dollar, it's not all that surprising to find "causality" among them, as the use of multiple series will make it simpler to distinguish between news that affects mainly a single currency, and news that affects all of them (through the effect on the relative value of the dollar).

Exogeneity tests in the mean for each of the three currencies require excluding lags of the other two variables:

```
test(zeros,title="Test of Exogeneity in Mean of Euro")
# 2 3
*
test(zeros,title="Test of Exogeneity in Mean of Pound")
# 5 7
*
test(zeros,title="Test of Exogeneity in Mean of Swiss Franc")
# 9 10
```

all of which are strongly significant at conventional levels.

Moving on to the tests of the variance parameters, a test for whether there is any spillover/causality in the variance for any of the series requires excluding all the "non-diagonal" elements of both **A** and **B**. Diagonality of **A** alone is not enough, as that would only be testing for first-period effects—if **B** is not diagonal, there can be effects at multiple steps.

```
test(zeros,title="Wald Test of Diagonal BEKK")
# 20 21 22 24 25 26 29 30 31 33 34 35
```

```
Wald Test of Diagonal BEKK
Chi-Squared(12)=     94.838308 or F(12,*)=      7.90319 with Significance Level 0.0000
```

The block exclusions for the variance are:

```
test(zeros,title="Block Exclusion Test, Euro Variance")
# 22 25 31 34
*
test(zeros,title="Block Exclusion Test, Pound Variance")
# 20 26 29 35
*
test(zeros,title="Block Exclusion Test, Swiss Franc Variance")
# 21 24 30 33
```

Note that these are excluding both the **A**'s and the **B**'s for both the other variables—if you don't, you could end up missing multiple-step effects—if those four coefficients are zero, shocks to the other variables can't (according to the model) affect the variance of interest. Note also that, because of the way the BEKK matrices are set up (with the transpose on the pre-multiplication term), the coefficients that hit variance i have column (not row) i, so the test for the Euro (variable 1) is a joint test for $A(2,1)$, $A(3,1)$, $B(2,1)$ and $B(3,1)$. The only one of these that isn't *strongly* significant is for the Swiss Franc (.019 significance level).

5.6 CC Models: Constant Correlation

An alternative approach for creating a model which is easier to fit than the DVECH model is the *Constant Correlation* (or CC) model of Bollerslev (1990). The DVECH model uses a simple GARCH model for the variances—the numerical problems arise from the lack of connection between the variance recursions and the covariance recursions. The CC model assumes that the covariances are generated with a constant (but unknown) correlation. Whether this is *too* restrictive will depend upon the application: from Figure 5.2, a constant correlation between the Swiss Franc and Euro doesn't seem unreasonable, but it appears to be less likely to do well explaining the relationships of those with the Pound.

The number of free parameters is reduced quite a bit. C, A and B are now just n vectors, and the sub-diagonal of the correlation matrix has only $n(n-1)/2$ free parameters—in our model, that's 3 for a total of 12. The estimation typically behaves well enough that it isn't even necessary to choose a parameterization of the correlation matrix which enforces positive-definiteness—in practice, it doesn't stray close enough to the region where the correlation matrix becomes singular to need that extra protection.

This is estimated with **RATS** using the MV=CC option. You typically don't need any help from simplex iterations, so:

```
garch(model=mvmean,mv=cc,rvectors=rd,hmatrices=hh,robusterrors)
```

which gives us Table 5.6. The likelihood is quite a bit worse than for the DVECH and BEKK models, so it seems that the graphical evidence that constant correlation is inappropriate is confirmed.

Table 5.6: Multivariate GARCH, CC Estimates

MV-GARCH, CC - Estimation by BFGS
Convergence in 50 Iterations. Final criterion was 0.0000017 <= 0.0000100
With Heteroscedasticity/Misspecification Adjusted Standard Errors
Daily(5) Data From 2000:01:04 To 2008:12:23
Usable Observations 2341
Log Likelihood -3053.8808

	Variable	Coeff	Std Error	T-Stat	Signif
1.	Constant	0.0313	0.0136	2.2933	0.0218
2.	Constant	0.0168	0.0124	1.3598	0.1739
3.	Constant	0.0289	0.0157	1.8327	0.0668
4.	C(1)	0.0045	0.0012	3.6068	0.0003
5.	C(2)	0.0048	0.0019	2.5703	0.0102
6.	C(3)	0.0059	0.0016	3.5804	0.0003
7.	A(1)	0.0547	0.0083	6.5670	0.0000
8.	A(2)	0.0496	0.0093	5.3639	0.0000
9.	A(3)	0.0460	0.0064	7.1888	0.0000
10.	B(1)	0.9344	0.0093	100.3212	0.0000
11.	B(2)	0.9343	0.0129	72.2737	0.0000
12.	B(3)	0.9416	0.0071	133.3927	0.0000
13.	R(2,1)	0.7082	0.0110	64.5446	0.0000
14.	R(3,1)	0.9180	0.0048	193.1722	0.0000
15.	R(3,2)	0.6503	0.0142	45.9335	0.0000

Of course, in practice, you would start with the simpler CC. The multivariate diagnostics point to the inadequacy of the model, with a highly significant test for lack of residual ARCH:

```
Multivariate Q(10)=     108.10427
Significance Level as Chi-Squared(90)=       0.09389

Test for Multivariate ARCH
Statistic Degrees Signif
   353.00      72 0.00000
```

So if we did these in the more standard order of simplest first, it would point us towards the need for a more complicated model. Tse (2000) provides an LM test for CC against an alternative that the correlation that allows greater adaptation to the observed (lagged) outer product of the residuals. This is implemented with the procedure **@TseCCTest**. It requires that you first estimate the CC model, saving the VECTOR[SERIES] of derivatives of the log likelihood with respect to the free parameters. This requires the DERIVES option, used before in Section 4.3 in doing fluctuations tests. It also needs the sequences of residuals and variances that we're already saving. For this model, we do:

```
garch(model=mvmean,mv=cc,rvectors=rd,hmatrices=hh,derives=dd)
@tsecctest(rvector=rd,hmatrices=hh,derives=dd)
```

which produces the somewhat disappointing (given what we know) result:

```
Tse Test for CC
Chi-Squared(3)=      7.712254 with Significance Level 0.05234836
```

Table 5.7: Multivariate GARCH, CC with Spillover

MV-GARCH, CC with Spillover Variances - Estimation by BFGS
Convergence in 64 Iterations. Final criterion was 0.0000087 <= 0.0000100
With Heteroscedasticity/Misspecification Adjusted Standard Errors
Daily(5) Data From 2000:01:04 To 2008:12:23
Usable Observations 2341
Log Likelihood -3032.9529

	Variable	Coeff	Std Error	T-Stat	Signif
1.	Constant	0.0272	0.0085	3.1882	0.0014
2.	Constant	0.0144	0.0091	1.5938	0.1110
3.	Constant	0.0240	0.0098	2.4594	0.0139
4.	C(1)	0.0049	0.0011	4.5878	0.0000
5.	C(2)	0.0047	0.0016	2.9046	0.0037
6.	C(3)	0.0064	0.0015	4.3300	0.0000
7.	A(1,1)	0.0898	0.0135	6.6510	0.0000
8.	A(1,2)	-0.0244	0.0096	-2.5541	0.0106
9.	A(1,3)	-0.0265	0.0094	-2.8074	0.0050
10.	A(2,1)	0.0451	0.0114	3.9589	0.0001
11.	A(2,2)	0.0386	0.0092	4.1735	0.0000
12.	A(2,3)	-0.0457	0.0095	-4.8358	0.0000
13.	A(3,1)	0.0334	0.0156	2.1460	0.0319
14.	A(3,2)	-0.0180	0.0110	-1.6299	0.1031
15.	A(3,3)	0.0240	0.0101	2.3768	0.0175
16.	B(1)	0.9399	0.0078	121.0810	0.0000
17.	B(2)	0.9462	0.0095	99.0982	0.0000
18.	B(3)	0.9435	0.0067	139.8309	0.0000
19.	R(2,1)	0.7120	0.0113	63.0283	0.0000
20.	R(3,1)	0.9181	0.0045	202.9850	0.0000
21.	R(3,2)	0.6556	0.0144	45.5062	0.0000

Since we've estimated a more general model and found that it fit *much* better, the fact that the Tse test doesn't pick up on that is unfortunate. It's an LM test against what would appear in this case to be an alternative which doesn't define a sharp enough contrast. The lesson is not to rely solely upon the Tse test as a diagnostic for the CC model.

One nice feature of the CC framework is that the GARCH models for the variances can take almost any form. For instance, a multivariate EGARCH model is created by using EGARCH models for the variances with the CC to handle the covariances. To choose this, add the option VARIANCES=EXP to the **GARCH** instruction:

```
garch(model=mvmean,mv=cc,variances=exp,$
   rvectors=rd,hmatrices=hh,robusterrors)
```

which, here, fits even worse (log likelihood -3080.4548).

An alternative variance model which allows for a greater interaction is VARIANCES=SPILLOVER, which adds "spillover" terms to the variance calcu-

lation:

$$h_{ii,t} = c_{ii} + \sum_j a_{ij} u_{j,t-1}^2 + b_i h_{ii,t-1}$$

```
garch(model=mvmean,mv=cc,variances=spillover,$
   rvectors=rd,hmatrices=hh,robusterrors,$
   pmethod=simplex,piters=20,iters=500)
```

This is no longer a "simpler" model than the DVECH—in this case, it has exactly the same number of free parameters (21). It provides greater flexibility in the variances, but less in the covariances. This produces some interesting results (Table 5.7).

First, despite also having 21 parameters, the fit is much worse than the DVECH. And note that most of the off-diagonal parameters in **A** are *negative*. Because these are used directly (not squared, as in the BEKK), this would seem to imply that spillover is negative rather than positive. However, this may be due to a misspecification of the spillover effect. If it is the case that volatility goes up more when the shocks have opposite signs, as we seemed to see in the BEKK model, that can't be captured by this model which only uses the magnitudes. And, in fact, if it's a linear combination with differing signs whose square matters, then it turns out that the closest (though not very close) fit to that given only the squares has a negative coefficient on one of them.

5.7 DCC Models: Dynamic Conditional Correlation

The CC model has several advantages over the "vech" forms:

1. It doesn't have the problem of possible non-positive definite covariance matrices of the VECH and DVECH.
2. It allows more flexibility in handling the variances than BEKK, where the positive-definiteness comes at a cost of a very specific variance recursion.
3. It can handle larger sets of series, as the number of parameters doesn't increase as fast with n.

But that comes at the major drawback of assuming that the conditional correlation is constant. It didn't take long for models to be proposed which relaxed that assumption. If we have a model which, at time t, produces an n vector of variances h_t, and an $n \times n$ correlation matrix $\mathbf{R}_t$ (positive-definite symmetric matrix with 1's on the diagonal), then $\mathbf{R}_t$ and h_t can be converted into a covariance matrix **H** by

$$H_{t,ij} = \sqrt{h_{t,i}}\sqrt{h_{t,j}} R_{t,ij}$$

If $n = 2$, there are many ways to produce such an $\mathbf{R}_t$, as there is only one free value in it (since it's symmetric with 1's on the diagonal), and it's positive definite if and only if the $(1, 2)$ element has absolute value less than 1. For instance, if it is expected that the correlation would be monotonic in some exogenous variable Z, then a logistic mapping such as

$$\log\left(\frac{1+\rho_t}{1-\rho_t}\right) = \gamma + \delta Z_t$$

can be used, at least as an approximation.[8] And it's possible to take this idea and make the logistic index a function of the past residuals, so the correlation adapts to the recent past of the data.

However, the logistic mapping doesn't generalize well to more than $n = 2$. Engle (2002) proposed a more general method which he labeled *Dynamic Conditional Correlations* which can be applied to an arbitrary number of series.[9] This uses a GARCH-like secondary recursion that generates a symmetric matrix which is converted into the correlation matrix:

$$\begin{aligned} \mathbf{Q}_t &= (1-a-b)\bar{\mathbf{Q}} + b\mathbf{Q}_{t-1} + a\varepsilon_{t-1}\varepsilon_{t-1}' \\ \mathbf{R}_t &= diag(\mathbf{Q}_t)^{-1/2}\mathbf{Q}_t diag(\mathbf{Q}_t)^{-1/2} \end{aligned} \tag{5.7}$$

where $\bar{\mathbf{Q}}$ is a fixed matrix towards which the Q matrices "shrink". ε_{t-1} is the lagged univariate standardized residual, that is, each component is divided by its own estimated standard deviation. If the GARCH models are correct, the *expected value* of $\varepsilon_{t-1}\varepsilon_{t-1}'$ is a "correlation matrix":

1. it *has* to be positive semi-definite (it's rank one, so it is only semi-definite)
2. the expected values of the diagonal elements are 1

Since $\bar{\mathbf{Q}}$ and the pre-sample Q (which is made equal to $\bar{\mathbf{Q}}$) will be positive definite by construction, as long as b is non-zero, each $\mathbf{Q_t}$ will be positive definite as well. A weighted average like this of actual correlation matrices would also be a correlation matrix, but because the final term is only a correlation matrix in expected value, $\mathbf{Q}_t$ itself isn't a correlation matrix, but as a positive definite symmetric matrix can be converted to one as in (5.7).

In Engle's procedure,

$$\bar{\mathbf{Q}} = \frac{1}{T}\sum_t \varepsilon_t\varepsilon_t' \tag{5.8}$$

that is, it's the average outer product of the standardized residuals. One problem with this in practice is that ε_t, as the standardized residual, depends upon the estimated GARCH variances. While each individual ε_{t-1} in (5.7) will be

[8] This takes a logistic and remaps it from (0,1) to (-1,1).

[9] Note that the phrase "Dynamic Conditional Correlations" has also been applied in some papers to *other* methods of generating time-varying correlations.

known when they're needed, the $\bar{\mathbf{Q}}$ depends upon the entire data range (and would change with changes to the univariate GARCH parameters). That specific calculation is really only feasible with the "two-step" procedure that Engle describes:

1. fit univariate GARCH processes to all the series
2. take the residuals and variances from those as given, compute $\bar{\mathbf{Q}}$ (which will now be a fixed matrix once the residuals and variances are given) and estimate the a and b values and generate the time-varying correlations with them.

Engle shows that the two-step procedure is consistent, though inefficient (since the univariate GARCH estimates aren't taking into account the correlations with the other processes). However, it can feasibly be applied to rather large sets of series because there are only two free parameters in the multivariate maximization—the univariate GARCH models are estimated separately.

One major drawback to the two-step procedure (inefficiency aside) is that it can only be used if the univariate GARCH models are self-contained. The models designed for use with CC or DCC that provide for some type of "spillover" in computing the variances can't be used since they need joint estimation anyway to generate all the information needed.

To allow for feasible use of DCC for a broader set of models, in **GARCH** with the option MV=DCC, the $\bar{\mathbf{Q}}$ matrix is computed using the sample correlation matrix of (non-standardized) residuals from a preliminary estimate of the mean model. In practice, the difference is likely to be minor.[10]

An alternative for generating the correlation matrices which is, in fact, what the **GARCH** instruction does by default, is to do the recursion (5.7) using covariance matrices rather than correlations—instead of the standardized residuals ε_{t-1}, it uses the equation residuals u_t and instead of $\bar{\mathbf{Q}}$ being the sample correlation matrix, it's the sample *covariance* matrix. In practice, the difference ends up being minor, though when it isn't, the recursion in the covariances is generally the one that does better. The recursion in correlations gives greater weight to entries which are outliers according to the GARCH model (large *standardized* residuals), while the recursion in covariances puts greater weight on entries which have large non-standardized residuals—in most cases, the two are the same. The choice between the two is controlled by the option DCC=COVARIANCES (default) or DCC=CORRELATIONS.[11]

A third option is the "corrected DCC" of Aielli (2013). Aielli shows that the original Engle recursion isn't internally consistent (in the non-statistical sense)—the expectation of $\varepsilon_{t-1}\varepsilon_{t-1}'$ isn't Q_{t-1} as it would be if (5.7) were a GARCH model,

[10] The effect of the $\bar{\mathbf{Q}}$ matrix is short-lived unless a and b are fairly small, which generally means that a CC model would have been a better choice.

[11] These were added with Version 10.

and Engle's $\bar{\mathbf{Q}}$ isn't the stationary solution of (5.7) (unless the true DGP is CC rather than DCC). The corrected DCC scales up ε_{t-1} by the (square roots of) the diagonal elements of Q_{t-1}. You get this by using the option DCC=CDCC.

While DCC is a common choice, particularly with larger numbers of series, there is no GARCH model which works well under all conditions, and this is no exception:

1. If CC is correct, then the parameters of DCC aren't identified: any combination with $a = 0$ and any value of b will produce the same correlation matrix all the way through the data set, since the presample value of $\mathbf{Q}$ is the same as $\bar{\mathbf{Q}}$.
2. If the correlations are small (that is, near zero), even if they aren't fixed, it will be hard to estimate the DCC parameters because the log likelihood is very flat near zero.
3. Because DCC is using only one pair of free parameters for an n variable correlation subsystem, if you have many series and they have very different dynamics, the DCC may end up overdoing the "dynamics" in some pairs to fit the others reasonably.

You should always do a CC model as well (first!) as it may give you some information as to whether you are running into problems with 1 and 2. Note that CC and DCC don't really nest—DCC uses just the two parameters to generate the correlation matrix, as the $\bar{\mathbf{Q}}$ isn't freely estimated.

Example **5.6** estimates a standard DCC-GARCH model (that is, with standard GARCH models for the univariate variance estimates) to the Enders data set used in the other examples in this chapter.

```
garch(model=mvmean,mv=dcc,robusterrors,$
  rvectors=rd,hmatrices=hh,stdresids=rstd,factorby=eigen,$
  iters=500,pmethod=simplex,piters=10)
```

The output is in Table 5.8. DCC(A) and DCC(B) are the a and b parameters from the recursion to generate the correlation matrix. When DCC "works", this is the type of pattern you will generally see—a is fairly small (typically under .1) and b is large, with the two generally summing to somewhere above .9, often, as here, nearly to 1. If we compare the log likelihood to the other models, we see that this is *much* better than CC, is somewhat worse than DVECH and BEKK, but is better than both of those if we apply the SBC as it has many fewer parameters than those other models.

A common task is to graph the conditional correlations generated by the model. Note that all other forms of multivariate GARCH models (other than CC) will also produce time-varying correlations, which can be computed and displayed in exactly the same way—we already did this in Example **5.2**.

Table 5.8: DCC GARCH Model

MV-DCC GARCH - Estimation by BFGS
Convergence in 74 Iterations. Final criterion was 0.0000071 <= 0.0000100

With Heteroscedasticity/Misspecification Adjusted Standard Errors
Daily(5) Data From 2000:01:04 To 2008:12:23
Usable Observations 2341
Log Likelihood -2774.2692

	Variable	Coeff	Std Error	T-Stat	Signif
Mean Model(REURO)					
1.	Constant	0.0256	0.0087	2.9312	0.0034
Mean Model(RPOUND)					
2.	Constant	0.0154	0.0096	1.6041	0.1087
Mean Model(RSW)					
3.	Constant	0.0210	0.0098	2.1345	0.0328
4.	C(1)	0.0024	0.0010	2.4998	0.0124
5.	C(2)	0.0031	0.0014	2.2454	0.0247
6.	C(3)	0.0031	0.0012	2.6588	0.0078
7.	A(1)	0.0408	0.0067	6.1000	0.0000
8.	A(2)	0.0481	0.0059	8.0903	0.0000
9.	A(3)	0.0373	0.0049	7.5720	0.0000
10.	B(1)	0.9539	0.0083	115.2548	0.0000
11.	B(2)	0.9438	0.0087	108.2947	0.0000
12.	B(3)	0.9568	0.0065	146.6784	0.0000
13.	DCC(A)	0.0226	0.0059	3.8156	0.0001
14.	DCC(B)	0.9752	0.0071	136.4902	0.0000

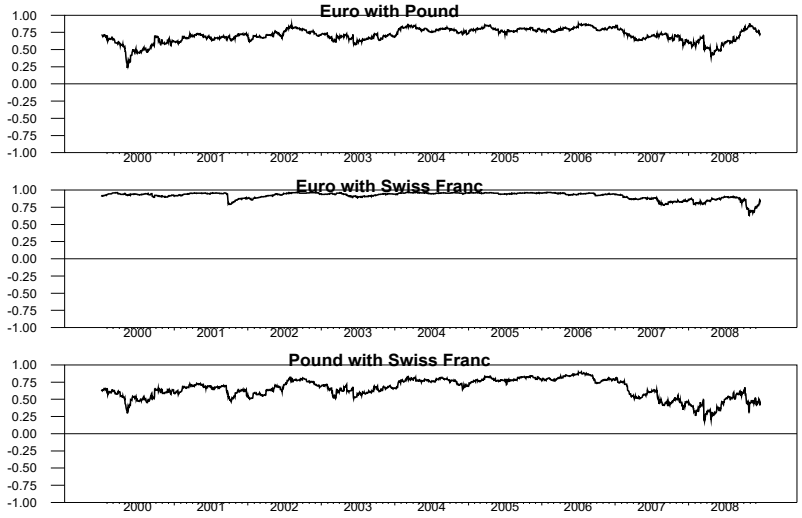

Figure 5.3: DCC Correlations

With two or three series, it's generally easiest to just set up the correlation graphs directly:

```
set r_euro_pound %regstart() %regend() = %cvtocorr(hh(t))(1,2)
set r_euro_sw    %regstart() %regend() = %cvtocorr(hh(t))(1,3)
set r_pound_sw   %regstart() %regend() = %cvtocorr(hh(t))(2,3)
spgraph(vfields=3,footer="Conditional Correlations from DCC")
 graph(header="Euro with Pound",min=-1.0,max=1.0)
 # r_euro_pound
 graph(header="Euro with Swiss Franc",min=-1.0,max=1.0)
 # r_euro_sw
 graph(header="Pound with Swiss Franc",min=-1.0,max=1.0)
 # r_pound_sw
spgraph(done)
```

This (Figure 5.3) uses -1 to 1 ranges on all the series. If you don't include those, you could easily have graphs which overemphasize what may be very small changes to correlations. For instance, the Euro with the Swiss Franc, over almost the entire range (other than the last year) shows almost no change at all, but without pinning the range, the (actually fairly minor) dip in late 2001 would be spread across almost the entire vertical range of the graph. It's important not to overanalyze the movements in these. We'll see later Section 10.2.3) how it is possible to put some type of error band on these, but remember that these are just point estimates.

An alternative to manually putting together a correlation graph is the following, which generates the same graph programmatically using "long labels" input for the series. This is useful if you have more than 3 series (though the graphs get rather crowded above 5), or if you're using 2 or 3 series from a longer list.

```
compute totalfields=%nvar*(%nvar-1)/2
compute hfields=fix(sqrt(totalfields))
compute vfields=(totalfields-1)/hfields+1
spgraph(vfields=vfields,hfields=hfields,$
   footer="Conditional Correlations from DCC")
 do i=1,%nvar
   do j=i+1,%nvar
      set ccorr = %cvtocorr(hh(t))(i,j)
      graph(header=longlabel(i)+" with "+longlabel(j),$
                           min=-1.0,max=1.0)
      # ccorr
   end do j
 end do i
spgraph(done)
```

5.8 RATS Tips and Tricks

5.8.1 Graphics with Multiple Series

It's very common in working with multivariate GARCH models to need something similar to Figure 5.1: a graph with multiple fields of relatively similar types of data. There are a number of decisions you need to make in trying to make these presentable for publication. One is how to label the fields. Here, we did it by using the YLABELS option on the enclosing **SPGRAPH**, which labels them in the left margin, as shown in the figure. The most common alternative is to include a label on each **GRAPH** instruction. In many cases, the graphs are simply lettered with (a), (b) and (c), with descriptions in the captions. For three series, it's probably easiest to do three separate **GRAPH** instructions, rather than using the **DOFOR** loop:

```
spgraph(vfields=3)
 graph(picture="*.#",hlabel="(a)")
 # reuro
 graph(picture="*.#",hlabel="(b)")
 # rpound
 graph(picture="*.#",hlabel="(c)")
 # rsw
spgraph(end)
```

(or the same with more descriptive titles). It usually looks best to use an HLABEL (which puts the information below each graph) rather than a HEADER (which puts it above). However, labeling on the left margin in some form typically gives a better appearance. There are two problems with inserting the labels between graphs

1. It's not clear (to the reader) at a quick glance to which graph the label applies.
2. The individual graphs are already disproportionately wide. Inserting labels between graphs makes them even smaller in the vertical direction, while marginal labels make them narrower horizontally.

Aside from the layout, as you can see in Figure 5.1, the labeling of the values on the vertical axis varies from graph to graph, which can look a bit odd, particularly when the scales aren't actually that different. We used the PICTURE option on the **GRAPH** instructions in the example to force them to at least align the same, even if the values were different.[12] However, where the series are as similar in scale as these are, a better solution both visually, and for interpretation, is to force them onto a common scale. The easiest way to do that is to

[12]The pound has labels at 2.5 and -2.5 and thus needs a digit right of the decimal, while the other two can (and by default will) just use integers.

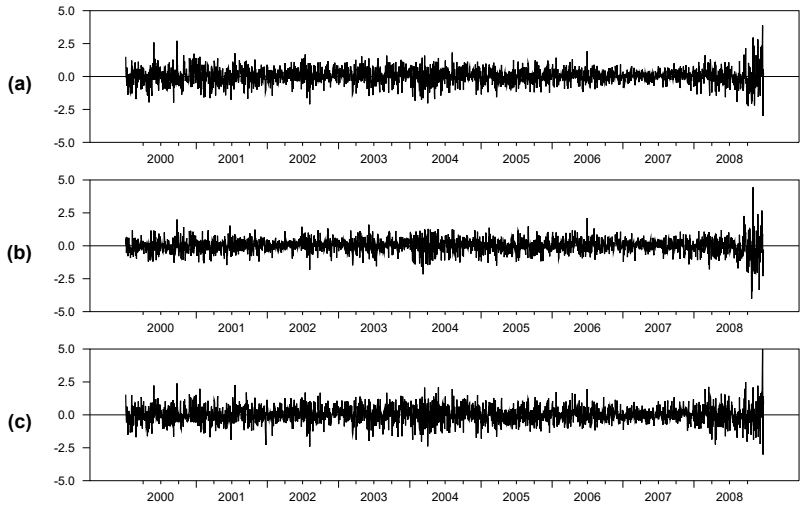

Figure 5.4: Exchange Rate Returns, Prettied Up

use the **TABLE** instruction, which sets the variables %MAXIMUM and %MINIMUM as the overall maximum and minimum values of all the series covered. If we switch now to marginal labels with (a), (b), and (c):

```
table / reuro rpound rsw
spgraph(vfields=3,ylabels=||"(a)","(b)","(c)"||)
dofor r = reuro rpound rsw
   graph(min=%minimum,max=%maximum)
   # r
end dofor
spgraph(done)
```

we get Figure 5.4, which has a much cleaner appearance.

5.8.2 Fancy Table of Diagnostics

To make an interesting report out of this, we will need more descriptive labels of the currencies, which we can get with:

```
dec vect[string] longlabel(3)
compute longlabel=||"Euro","UK Pound","Swiss Franc"||
```

A first go at this would insert a few **REPORT** instructions (and delete the **DISPLAY**).

Table 5.9: Sample Univariate Diagnostics

Currency	Q	Signif	ARCH	Signif
Euro	8.734	0.558	16.303	0.038
UK Pound	13.556	0.194	9.688	0.288
Swiss Franc	4.448	0.925	9.149	0.330

```
report(action=define)
report(atrow=1,atcol=1) "Currency" "Q" "Signif" "ARCH" "Signif"
do i=1,%nvar
   set ustd  = rd(t)(i)/sqrt(hh(t)(i,i))
   set ustdsq = ustd^2
   @regcorrs(number=10,nocrits,nograph,qstat) ustd
   compute q1=%cdstat,q1signif=%signif
   @regcorrs(number=10,nocrits,nograph,qstat,dfc=2) ustdsq
   compute q2=%cdstat,q2signif=%signif
   report(atrow=i+1,atcol=1) longlabel(i) q1 q1signif q2 q2signif
end do i
report(action=show)
```

Without additional formatting, that produces:

Currency	Q	Signif	ARCH	Signif
Euro	8.733736	0.557546	16.303304	0.038239
UK Pound	13.556423	0.194203	9.688304	0.287588
Swiss Franc	4.447671	0.924901	9.149159	0.329870

We need to show fewer digits (3 is probably good for everything), and it would look better to center the column headers over the numerical columns. However, the "Currency" header should still be left-justified. Two **REPORT** instructions with ACTION=FORMAT will take care of this. These are inserted just before the **REPORT(ACTION=SHOW)**:

```
report(action=format,align=center,atrow=1,torow=1,atcol=2)
report(action=format,picture="*.###")
```

On the first of these, we need to make sure the ALIGN=CENTER only applies where we want it: ATROW=1 and TOROW=1 limit it to row 1, and ATCOL=2 (without TOCOL) limits it to columns from 2 until the end. We don't have to worry about that with the second, since PICTURE only applies to numerical cells. The end result is Table 5.9.

Example 5.1 Multivariate GARCH: Preliminaries/Diagnostics

This includes the programs from Sections 5.1, 5.2 and 5.3.

```
open data "exrates(daily).xls"
calendar(d) 2000:1:3
data(format=xls,org=columns) 2000:01:03 2008:12:23 $
   aust euro pound sw ca
*
set reuro  = 100.0*log(euro/euro{1})
set rpound = 100.0*log(pound/pound{1})
set rsw    = 100.0*log(sw/sw{1})
*
spgraph(vfields=3,ylabels=||"Euro","Pound","Swiss Franc"||)
 dofor r = reuro rpound rsw
   graph(picture="*.#")
   # r
 end dofor
spgraph(done)
*
* Look at possible lag lengths
*
@varlagselect(crit=bic,lags=5)
# reuro rpound rsw
*
system(model=mvmean)
variables reuro rpound rsw
lags
det constant
end(system)
*
* Test for ARCH effects
*
estimate(resids=resids)
@mvarchtest
# resids
*
garch(model=mvmean,mv=diag,p=1,q=1,$
  rvectors=rd,hmatrices=hh,stdresids=rstd,factorby=eigen)
*
* Univariate diagnostics
*
do i=1,%nvar
   set ustd  = rd(t)(i)/sqrt(hh(t)(i,i))
   set ustdsq = ustd^2
   @regcorrs(number=10,nocrits,nograph,qstat) ustd
   compute q1=%cdstat,q1signif=%signif
   @regcorrs(number=10,nocrits,nograph,qstat,dfc=2) ustdsq
   compute q2=%cdstat,q2signif=%signif
   disp q1 q1signif q2 q2signif
end do i
*
* Multivariate diagnostics
```

```
*
@mvqstat(lags=10)
# rstd
@mvarchtest(lags=2)
# rstd
```

Example 5.2 Multivariate GARCH: DVECH Estimates

This is the program from Section 5.4.1.

```
open data "exrates(daily).xls"
calendar(d) 2000:1:3
data(format=xls,org=columns) 2000:01:03 2008:12:23 $
   aust euro pound sw ca
*
set reuro  = 100.0*log(euro/euro{1})
set rpound = 100.0*log(pound/pound{1})
set rsw    = 100.0*log(sw/sw{1})
*
dec vect[string] longlabel(3)
compute longlabel=||"Euro","UK Pound","Swiss Franc"||
*
system(model=mvmean)
variables reuro rpound rsw
lags
det constant
end(system)
*
garch(model=mvmean,rvectors=rd,hmatrices=hh,robusterrors,$
  pmethod=simplex,piters=20,method=bfgs,iters=500)
*
dec vect meanparms(%nregmean)
dec symm cdvech(%nvar,%nvar) advech(%nvar,%nvar) bdvech(%nvar,%nvar)
nonlin(parmset=dvech) meanparms cdvech advech bdvech
compute %parmspoke(dvech,%beta)
*
* Display the sum of the A and B matrices
*
disp advech+bdvech
*
* Extract the time-varying correlations and the variances
*
dec symm[series] hhx(%nvar,%nvar)
do i=1,%nvar
   set hhx(i,i) = hh(t)(i,i)
   do j=1,i-1
      set hhx(i,j) = %cvtocorr(hh(t))(i,j)
   end do j
end do i
*
* This is specific to a 3 variable system
```

```
*
table / hhx(2,1) hhx(3,1) hhx(3,2)
compute corrmin=%minimum
*
table / hhx(1,1) hhx(2,2) hhx(3,3)
compute varmax=%maximum
*
spgraph(vfields=%nvar,hfields=%nvar,$
    xlabels=longlabel,ylabels=longlabel)
do i=1,%nvar
   do j=1,%nvar
   if i==j {
      graph(row=i,col=i,maximum=varmax)
      # hhx(i,i)
   }
   else {
      graph(row=i,col=j,maximum=1.0,min=corrmin)
      # hhx(i,j)
   }
   end do j
end do i
spgraph(done)
```

Example 5.3 Multivariate GARCH: BEKK Estimates

This is the program from Section 5.4.2.

```
open data "exrates(daily).xls"
calendar(d) 2000:1:3
data(format=xls,org=columns) 2000:01:03 2008:12:23 $
   aust euro pound sw ca
*
set reuro  = 100.0*log(euro/euro{1})
set rpound = 100.0*log(pound/pound{1})
set rsw    = 100.0*log(sw/sw{1})
*
dec vect[string] longlabel(3)
compute longlabel=||"Euro","UK Pound","Swiss Franc"||
*
system(model=mvmean)
variables reuro rpound rsw
lags
det constant
end(system)
*
garch(model=mvmean,mv=bekk,robusterrors,pmethod=simplex,piters=20,$
   method=bfgs,iters=500,rvectors=rd,hmatrices=hh,$
   stdresids=rstd,factorby=eigen)
*
* Convert to equivalent VECH representation
*
```

```
@MVGARCHToVECH(mv=bekk)
disp ##.### %%vech_a
*
* Display eigenvalues of persistence matrix
*
eigen(cvalues=cv) %%vech_a+%%vech_b
disp ##.### cv
*
@mvqstat(lags=10)
# rstd
@mvarchtest(lags=2)
# rstd
*******************************************************
*
dec vect means(%nregmean)
dec packed cx(%nvar,%nvar)
dec rect ax(%nvar,%nvar) bx(%nvar,%nvar)
nonlin(parmset=garchparms) means cx ax bx
summarize(parmset=garchparms) bx(1,1)^2
summarize(parmset=garchparms) bx(1,1)*bx(1,2)*2.0
summarize(parmset=garchparms) bx(1,2)^2
*
compute ncomp=%nvar*(%nvar+1)/2
dec rect %%vech_bse(ncomp,ncomp)
do m=1,ncomp
   do n=1,ncomp
      compute i=%symmrow(m),j=%symmcol(m),$
              k=%symmrow(n),l=%symmcol(n)
      if k==l {
         summarize(noprint,parmset=garchparms) bx(i,k)*bx(j,l)
         compute %%vech_bse(m,n)=sqrt(%varlc)
      }
      else {
         summarize(noprint,parmset=garchparms) $
             bx(i,k)*bx(j,l)+bx(i,l)*bx(j,k)
         compute %%vech_bse(m,n)=sqrt(%varlc)
      }
   end do n
end do m
*
report(action=define,title="VECH-B from BEKK")
do j=1,ncomp
   report(atrow=1,atcol=j+1,align=center) $
       %string(%symmrow(j))+","+%symmcol(j)
end do i
do i=1,ncomp
   report(atrow=2*i,atcol=1) %string(%symmrow(i))+","+%symmcol(i)
   do j=1,ncomp
      report(atrow=2*i,atcol=j+1) %%vech_b(i,j)
      report(atrow=2*i+1,atcol=j+1,special=parens) %%vech_bse(i,j)
   end do j
end do i
report(action=format,atrow=2,picture="##.####",align=decimal)
report(action=show)
```

Example 5.4 Multivariate GARCH: Spillover Tests

This is the example from Section 5.5.

```
open data "exrates(daily).xls"
calendar(d) 2000:1:3
data(format=xls,org=columns) 2000:01:03 2008:12:23 $
   aust euro pound sw ca
*
set reuro  = 100.0*log(euro/euro{1})
set rpound = 100.0*log(pound/pound{1})
set rsw    = 100.0*log(sw/sw{1})
*
dec vect[string] longlabel(3)
compute longlabel=||"Euro","UK Pound","Swiss Franc"||
*
system(model=mvmean)
variables reuro rpound rsw
lags 1
det constant
end(system)
*
garch(model=mvmean,mv=bekk,robusterrors,pmethod=simplex,piters=20,$
   method=bfgs,iters=500,rvectors=rd,hmatrices=hh)
*
test(zeros,title="Test of Exogeneity in Mean of All Variables")
# 2 3 5 7 9 10
*
test(zeros,title="Test of Exogeneity in Mean of Euro")
# 2 3
*
test(zeros,title="Test of Exogeneity in Mean of Pound")
# 5 7
*
test(zeros,title="Test of Exogeneity in Mean of Swiss Franc")
# 9 10
*
test(zeros,title="Wald Test of Diagonal BEKK")
# 20 21 22 24 25 26 29 30 31 33 34 35
*
test(zeros,title="Block Exclusion Test, Euro Variance")
# 22 25 31 34
*
test(zeros,title="Block Exclusion Test, Pound Variance")
# 20 26 29 35
*
test(zeros,title="Block Exclusion Test, Swiss Franc Variance")
# 21 24 30 33
```

Example 5.5 Multivariate GARCH: CC Estimates

This is the example from Section 5.6.

```
open data "exrates(daily).xls"
calendar(d) 2000:1:3
data(format=xls,org=columns) 2000:01:03 2008:12:23 $
   aust euro pound sw ca
*
set reuro  = 100.0*log(euro/euro{1})
set rpound = 100.0*log(pound/pound{1})
set rsw    = 100.0*log(sw/sw{1})
*
dec vect[string] longlabel(3)
compute longlabel=||"Euro","UK Pound","Swiss Franc"||
*
system(model=mvmean)
variables reuro rpound rsw
lags
det constant
end(system)
*
garch(model=mvmean,mv=cc,robusterrors,$
  rvectors=rd,hmatrices=hh,stdresids=rstd,factorby=eigen)
*
* Multivariate diagnostics
*
@mvqstat(lags=10)
# rstd
@mvarchtest(lags=2)
# rstd
*
* Tse test for CC
* We need the derivatives of the log likelihood, which we save into
* the VECTOR[SERIES] called DD.
*
garch(model=mvmean,mv=cc,rvectors=rd,hmatrices=hh,derives=dd)
@tsecctest(rvector=rd,hmatrices=hh,derives=dd)
*
* EGARCH with CC
*
garch(model=mvmean,mv=cc,variances=exp,$
   rvectors=rd,hmatrices=hh,robusterrors)
*
* CC GARCH with spillover
*
garch(model=mvmean,mv=cc,variances=spillover,$
   rvectors=rd,hmatrices=hh,robusterrors,$
   pmethod=simplex,piters=20,iters=500)
```

Example 5.6 Multivariate GARCH: DCC Estimates

This is the example from Section 5.7.

```
open data "exrates(daily).xls"
calendar(d) 2000:1:3
data(format=xls,org=columns) 2000:01:03 2008:12:23 $
   aust euro pound sw ca
*
set reuro  = 100.0*log(euro/euro{1})
set rpound = 100.0*log(pound/pound{1})
set rsw    = 100.0*log(sw/sw{1})
*
dec vect[string] longlabel(3)
compute longlabel=||"Euro","UK Pound","Swiss Franc"||
*
system(model=mvmean)
variables reuro rpound rsw
lags
det constant
end(system)
*
garch(model=mvmean,mv=dcc,robusterrors,$
  rvectors=rd,hmatrices=hh,stdresids=rstd,factorby=eigen,$
  iters=500,pmethod=simplex,piters=10)
*
* Multivariate diagnostics
*
@mvqstat(lags=10)
# rstd
@mvarchtest(lags=2)
# rstd
*
* This does the conditional correlation graphs with hard coding
* (which probably is simplest with 2 or 3 series).
*
set r_euro_pound %regstart() %regend() = %cvtocorr(hh(t))(1,2)
set r_euro_sw    %regstart() %regend() = %cvtocorr(hh(t))(1,3)
set r_pound_sw   %regstart() %regend() = %cvtocorr(hh(t))(2,3)
spgraph(vfields=3,footer="Conditional Correlations from DCC")
 graph(header="Euro with Pound",min=-1.0,max=1.0)
 # r_euro_pound
 graph(header="Euro with Swiss Franc",min=-1.0,max=1.0)
 # r_euro_sw
 graph(header="Pound with Swiss Franc",min=-1.0,max=1.0)
 # r_pound_sw
spgraph(done)
*
* This does the conditional correlations for an arbitrary number of
* series using the <<longlabel>> VECTOR of STRINGS to manage the
* labels. This probably gets too cluttered when N is bigger than 5.
*
compute totalfields=%nvar*(%nvar-1)/2
```

```
compute hfields=fix(sqrt(totalfields))
compute vfields=(totalfields-1)/hfields+1
spgraph(vfields=vfields,hfields=hfields,$
   footer="Conditional Correlations from DCC")
 do i=1,%nvar
   do j=i+1,%nvar
      set ccorr = %cvtocorr(hh(t))(i,j)
      graph(header=longlabel(i)+" with "+longlabel(j),$
   min=-1.0,max=1.0)
      # ccorr
   end do j
 end do i
spgraph(done)
```