DCC-Range-GARCH

Package name: `DCC_RGARCH`

Date: January 2021

Description: This add-in estimates the multivariate volatility models, namely the DCC-Range-GARCH model and the DCC-GARCH model presented in [3]. The constrained estimation of the parameters of the DCC model is implemented. It is possible to assume either the normal or Student's t distributed errors. Additionally, the conditional variances, the conditional covariances and the conditional correlations are obtained. The one-step-ahead forecasts for the conditional variances, covariances and correlations are estimated. It can be accessed from both the dialogs and the command line.

Add-in: DCC-RGARCH model

Version: 1.0

Requirements: *EViews 8* or higher

Author: Marcin Fałdziński

Contents

1 Introduction

This add-in enables to estimate two multivariate volatility models, specifically the Dynamic Conditional Correlation Range Generalized AutoRegressive Conditional Heteroskedasticity model (in short DCC-Range-GARCH or DCC-RGARCH) proposed by Fiszeder et al. [3] and the DCC-GARCH model of Engle [1]. There is a number of additional features available, i.e:

- application of a different error distributions for the RGARCH and the GARCH models, i.e. Student's t and normal distribution
- application of a different multivariate conditional distributions for the DCC model, i.e. Student's t and normal distributions
- obtain the one-step-ahead forecasts for the conditional variances, covariances and correlations
- possibility to set the constraints on the DCC parameters which is incorporated into the estimation by the logit transformation (the parameters standard errors are computed utilizing the delta method)
- run the estimation of the DCC model with a grid of the starting values to avoid some estimation problems,
- for each time series (returns) possibility to set the individual GARCH model specification which are available within *EViews* prodcedures.

The DCC-GARCH model has gained popularity due to the fact that its parameters can be estimated by the quasi-maximum likelihood method using a two-stage approach (see [2]). This means that in the first stage the parameters of the univariate GARCH models can be estimated separately for each of the assets and the estimates of the conditional variances can be obtained. In the second stage residuals transformed by their estimated standard deviations are used to estimate the parameters of the correlation part conditioning on the parameters estimated in the first stage. The same approach is applied for the DCC-Range-GARCH model.
One of the main advantages of this package is that the dimension of the aforementioned models is only restricted by a user's computer capabilities and estimation restrictions. The curse of

dimensionality can be a real issue when it comes to a multivariate models. Fortunately, applying the two stage estimation approach it is possible to estimate the DCC-GARCH model for relatively a large number of assets.

The RGARCH, the GARCH and the GARCH-type models are specified as `logl` objects in *EViews*. Therefore, it is possible to utilize the tools which are available after the estimation. For instance, one may obtain the gradients at the estimated parameters or compute the confidence bands etc. It is also possible to re-estimate the model for the given asset directly from the `logl` object, but it will not re-estimate the DCC model (the correlation part). In order to re-estimate the DCC model it is required to estimate the model starting from the beginning, i.e. either using the package GUI or the command line. The estimation of parameters for the DCC model is specified via `optimize` function. It means that the estimation outputs are given in matrices and vectors (see 3.2 Outputs section). However, the package provides the estimation results of the DCC model in the table along with similar information as for the `logl` object (for instance the average of the logarithm of the likelihood function, the AIC, the BIC or HQC criteria among others).

The choice of starting values for the multivariate models is important. For that reason it is possible to supply the DCC model with starting values of your own. Nevertheless, the optimization algorithm may have a convergence problems due a number of reasons. Therefore, one may choose to estimate the parameters for the DCC model based on a grid search of the starting values. If the given starting values render the failure of an optimization algorithm then the package changes the staring values and run the estimation once again. The details of this iterative procedure are given in Section 2.

2 GUI

This section will illustrate the use of the dialogs. After installing the add-in go to the add-in menu and select *DCC-(R)GARCH model*. Hereafter, (R)GARCH means both the RGARCH and/or the GARCH models. You will see a dialog that looks like in Figure 1. In order to display the main dialog it is required to have an active workfile.

In the first dialog we input the list with names of returns (delimited by a single blank space). The next option is whether returns are multiplied by 100 or not. This option is needed to correctly compute Parkinson estimator. The following two dialogs are for the list of names with the high and low prices, respectively. It is required that the order of returns, high prices and low prices is exactly the same for given assets.

Next is the option to choose one of the three models to estimate, namely: DCC-RGARCH, DCC-GARCH or DCC-GARCH-type model. Notice that in order to estimate the DCC-RGARCH model you need to specify not only the returns, but also the high prices and the low prices. The DCC-RGARCH model takes advantage of the high prices and the low prices by utilizing the Parkinson estimator (see [3] for details). The initial conditional variance for the DCC-RGARCH and DCC-GARCH model is computed assuming the unconditional expectation and it is estimated as $\mathbf{H_0} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{r}_t\mathbf{r}_t'$, where $\mathbf{r}_t$ denotes $n \times 1$ vector of returns of n assets at time t.

The next dialog is to input the specifications of the GARCH-type models. This one is required if the DCC-GARCH-type model is chosen for the previous radio button. In this dialog you need to specify GARCH models (which are already built in *EViews*) for each of the returns given in the first dialog at the top of the main dialog. This gives flexibility because each of the returns processes may have different dynamics when it comes to the conditional mean and the conditional variance. For instance, if you wish to estimate the GARCH(1,1) model with the constant for the mean equation for the first series of returns, the GARCH(1,2) model with Student t-distribution errors for the second series of returns, and the TARCH(1,1) model for the third series of returns then the expression should be of the form:

```
(arch(1,1) returns_1 c); (arch(1,2, tdist) returns_2); (arch(1,1, thrsh=1) returns_3)
```

where `returns_1, returns_2, returns_3` are the series objects with the returns. It is important to ensure that the specifications of each model is in parentheses and delimited with semicolon and a single blank space. Generally speaking, specifications should be expressed in the following manner:

(*the first specification*); (*the second specification*); (*the third specification*)

where at the end there is not semicolon.

All the following dialogs or options are optional and do not have to be specified because the default values are set.

The next dialog is to input a sample period for the given page in a workfile. Only workfiles with times series are supported (this includes undated workfiles). If a sample is not specified then it is the current sample for the active page.

The following option is to choose the error distribution for the RGARCH and the GARCH model. Having chosen the DCC-GARCH-type model this option has no effects because each model specification may have different distribution.

Next option is to choose the conditional multivariate distribution for the error term in the DCC part of the model. This option applies to all three models.

You may wish to choose type of errors when it comes to the parameters estimates (standard and robust (Huber-White)). This option applies for both estimation stages, so for the (R)GARCH model and the DCC model.

The next two check boxes give possibility to choose whether constants should be included into the mean and the variance equations.

Next three dialogs are to set the starting values for the DCC model parameters, namely α, β and υ.

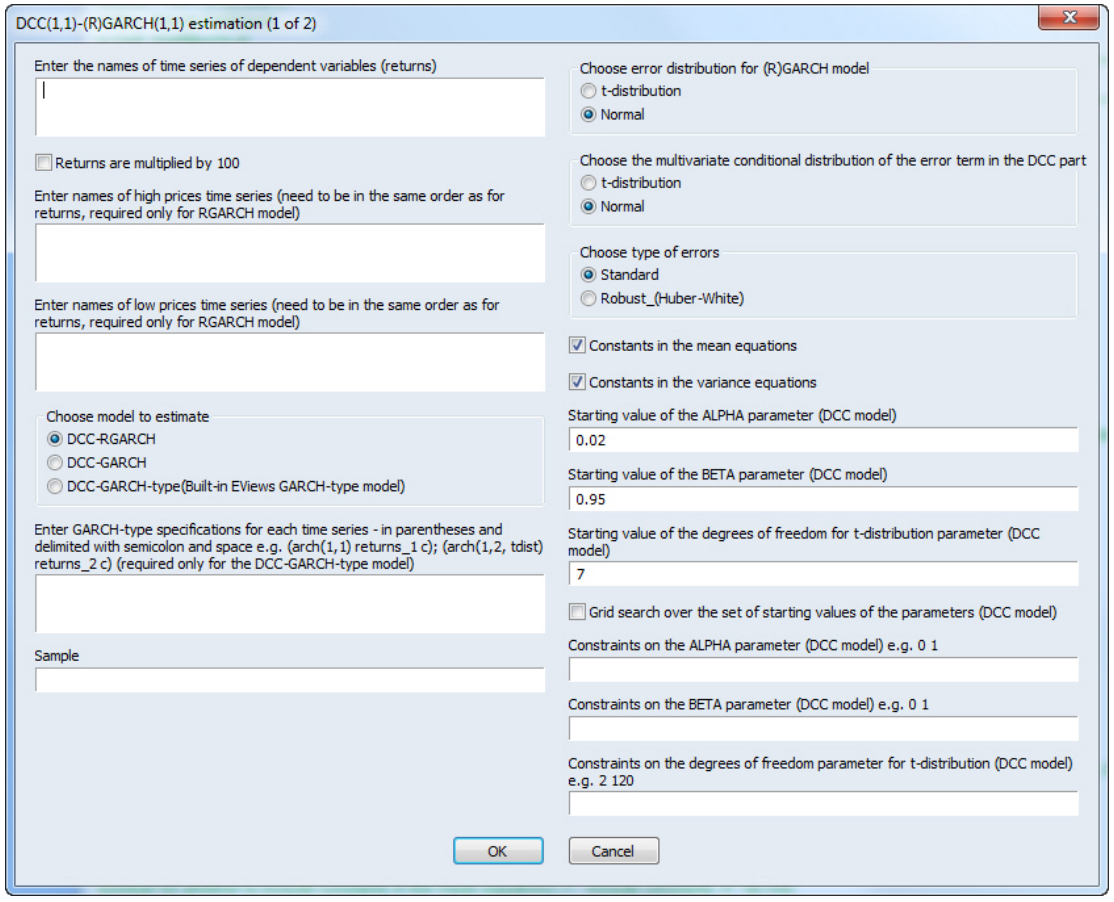

Figure 1: Main dialog with default settings (part 1)

The checkbox after setting the starting values is basically an option to run the estimation of the DCC model with the grid search over the set of the starting values. The choice of the starting values have great importance when it comes to convergence of an optimization algorithm. The basic idea is to generate a grid of the staring values based on the parameter space. In the case of the unconstrained estimation the space for both parameters is set as $(\alpha, \beta) \in [0, 1]^2$. The grid is created in such a way that the parameter space is partitioned into K evenly spaced points. Each point corresponds to a given starting value of one of the parameters. Firstly, the set of starting values for β parameter is obtained. In the next step, for each staring value of β the set of starting values for α parameter is obtained with regard to the condition that $\alpha + \beta < 1$. Therefore, the grid comprises different combinations of both parameters i.e. α and β. If the multivariate Student t-distribution is assumed for the DCC model then the grid is extended by adding the staring values for the degrees of freedom. In the unconstrained case the set of staring values for the degrees of freedom is generated based on the interval $\upsilon \in [2, 100]$. Once again the assumed parameter space is partitioned into a number of points referring to the staring values.

Hence, for each combination of the starting values for α and β, the staring values of the degrees of freedom are generated. For instance, assuming there is K points for each of the parameters $(\alpha, \beta, \upsilon)$ then we are going to get K^3 number of different starting values. In the constrained case, the grid is created in the similar fashion, but taking into account the constraints for the parameters. After generating the grid of the staring values for the parameters, the program initiates the estimation process based on the starting values from the grid. If there is an error resulting in failure of convergence or impossibility to evaluate the likelihood function or any other estimation problems then the add-in will change the starting values to the next possible set from the grid and will try to re-run the estimation. This iteration procedure continues to work until the convergence is achieved or until all generated staring values from the grid are exhausted and the program stops with an error message. After estimation, the list of tested starting values with corresponding error messages are enumerated in a table named `dcc_grid_errs` (see 3.2 Outputs section). The starting values given by a user, in the previous three dialogs, are supplied as the first set of the starting values for the grid search procedure.

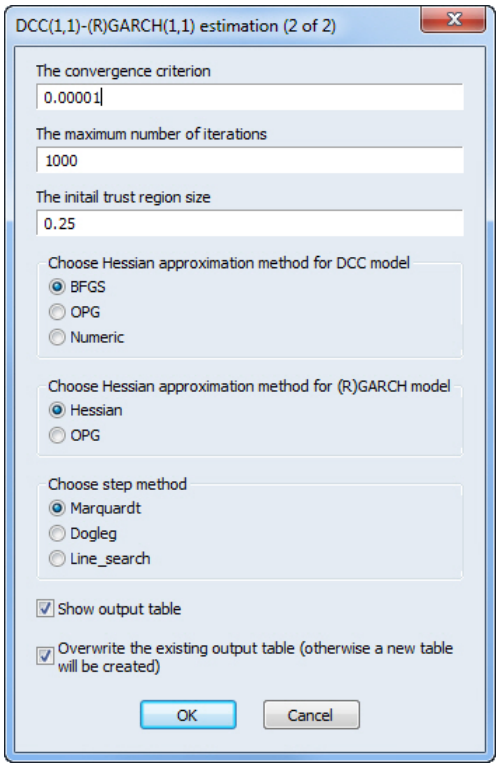

Figure 2: Main dialog with default settings (part 2)

The following three dialogs give possibility to set constraints on the DCC model parameters. It is possible to constraint only some of the parameters of the DCC model (for instance, only for β parameter). To specify bounds, the lower and upper bounds respectively are entered and delimited with blank space. For instance, if you specify: `0 1` for the β parameter then it means β parameter is to be bounded between 0 and 1. The constraints are incorporated into the es-

timation by the utilization of the logit transformation of the parameters. Specifically, in order to constraint the parameter, let say γ to lie between the lower bound LB and the upper bound HB, $LB \leq \gamma \leq UB$, the logit transformation is used. It means that the maximum likelihood method will estimate a new parameter $\tilde{\gamma} = LB + (HB - LB)\frac{exp(\gamma)}{1+exp(\gamma)}$. In outputs the reported coefficients are translated back to their original scale. The standard errors of the coefficients are estimated using the delta method due to the fact that the logit transformation is nonlinear. It is worth noting that the reported Hessian matrix is obtained for the parameter(s) after transformations and pertains to $\tilde{\gamma}$ and not to γ. Lastly, if one chooses the robust errors (Huber-White) and specifies constraints on the parameters of the DCC model then errors are obtained using sandwich method (Huber-White) and the delta method. Hence, the reported errors are robust errors.

That ends the first part of the main dialog. After clicking OK, the next dialog will be shown as presented in Figure 2.

First six dialogs are the settings for the maximum likelihood method. You can set the convergence criterion, the maximum number of iterations and the initial trust region size. It is possible to choose the Hessian approximation method for the DCC and the (R)GARCH model, separately. Additionally, you may choose the step method for the optimization algorithm.

The last two options are whether to show the table with the estimation output, i.e. with both the (R)GARCH models and the DCC model and whether to overwrite the existing (if there is) output table in case the add-in have been already run. Otherwise a new table will be created with the next available name. It should be noted that all other objects will be overwritten (i.e. matrices, series, vectors and so on).

3 Command line

dcc_rgarch(*dep=, multiply=, dep_high=, dep_low=, model=, ev_garch=, tdist_garch=, tdist_dcc=, std_err=, mean=, variance=, smpl=, dcc_a=, dcc_b=, dcc_v=, constraints_a_dcc=, constraints_b_dcc=, constraints_v_dcc=, grid_search=, convergence=, max_ite=, trust=, hess_approx_dcc=, hess_approx_rgarch=, step=, show_out=, overwrite=*)

3.1 Parameters

The parameters in the command line do not have to be in any specific order. There are only two required parameters in the command line, namely: **dep** and **model** for the DCC-GARCH model. For the DCC-GARCH-type models there are three required parameters in the command line, namely: **dep**, **model** and **ev_garch**. For the DCC-RGARCH model there are four required parameters in the command line, namely: **dep, dep_high, dep_low** and **model**.

It is worth noting that the arguments for the parameters in the command line are case sensitive. For instance, setting the parameter `model="GARCH"` will not estimate the GARCH model because the argument which corresponds to the GARCH model is `model="garch"`.

List of the parameters:

dep=: (*required*) list of the names of the time series with returns. Delimited by a single blank space, e.g. *dep="rt1 rt2 rt3"* means three time series of returns. The object type: `series`.

multiply=: (*optional*) whether the returns are multiplied by 100 or not. Possible options are: `"yes", "no"`. Default is `"no"`.

model=: (*required*) specify which model you would like to estimate. Possible options are: `rgarch, garch, ev_garch` which stand for the DCC-RGARCH model, DCC-GARCH model and DCC-GARCH-type (Built-in *EViews* GARCH-type) model, respectively. In order to estimate DCC-RGARCH model the parameters *dep_high* and *dep_low* are required to compute the Parkinson estimator, e.g. *model="ev_garch"*. Default if `rgarch`.

dep_high=: (*optional*) list of the names of the time series with high prices. Delimited by a single blank space, e.g. *dep_high="ht1 ht2 ht3"*. The names of the high prices need to be in the same order as for the returns. It means that *ht1* refers to *rt1*, *ht2* refers to *rt2* and so on. This parameter is required if the DCC-RGARCH model is chosen i.e. **model="rgarch"**. The object type: `series`.

dep_low=: (*optional*) list of the names of the time series with low prices. Delimited by a single blank space, e.g. *dep_low="lt1 lt2 lt3"*. The names of the low prices need to be in the same order as for the returns. It means that *lt1* refers to *rt1*, *lt2* refers to *rt2* and so on This parameter is required if the DCC-RGARCH model is chosen i.e. *model="rgarch"*. The object type: `series`.

ev_garch=: (*optional*) enter GARCH-type specifications for each time series in parentheses and delimited with semicolon and single blank space. By design, the DCC-GARCH model assumes that conditional variances for all return processes follow the univariate GARCH(1,1)

model which might not be appropriate. Here we assume that each return process can be specified individually. This enables flexibility and more appropriate specifications, especially for large number of the times series. Possible GARCH-type specifications are given and accessed by *EViews* equation `arch` (see *EViews* documentation for details). This dialog is required if *model="ev_garch"* is chosen. The GARH-type model specifications need to be in parentheses and delimited with semicolon and a single blank space, e.g. *ev_garch="(arch(1,1) returns_1 c); (arch(1,2, tdist) returns_2); (arch(1,1, egarch) returns_3 c ar(1))"* Notice that in the given example: the first return process follows the GARCH(1,1) model with a constant in the conditional mean, the second return process follows the GARCH(1,2) model with Student t distribution errors, and the third return process follows the EGARCH(1,1) model with the first order autoregressive AR(1) in the conditional mean. This parameter is required if DCC-GARCH-type model is chosen i.e. *model="ev_garch"*.

smpl=: (*optional*) specify sample, e.g. `smpl="1 2000"` for undated time series, `smpl="3/10/2008 7/10/2010"` for dated time series (compliant with a given data representation). If it is not specified then it is the current sample for the active page.

tdist_garch=: (*optional*) specify whether to apply Student t-distributed errors for the DCC-RGARCH and DCC-GARCH model. Possible option are: `yes, no`. The option `yes` means Student t-distributed errors and the option `no` means normally distributed errors, e.g `tdist_garch="yes"`. Default is `no`. Having chosen *model="ev_garch"* this option is overridden and has no effect because the error distribution can be specified for each return process individually.

tdist_dcc=: (*optional*) specify whether to apply the multivariate Student t-distribution errors for the DCC part of model. Possible option are: `yes, no`. The option `yes` means Student t-distributed errors and the option `no` means normally distributed errors, e.g `tdist_dcc="yes"`. Default is `no`.

std_err=: (*optional*) specify type of errors for both DCC and (R)GARCH models. Possible options are: `robust, standard`. e.g. `std_err="robust"`. Robust errors are obtained using sandwich method (Huber-White). Default is `standard`.

mean=: (*optional*) select whether to include constants in the mean equations. Possible options are: `yes, no`, e.g. `mean="yes"`. Default is `yes`.

variance=: (*optional*) whether to include constants in the variance equations. Possible options are: `yes, no`, e.g. `variance="yes"`. Default is `yes`.

dcc_a=: (*optional*) set starting value of the α parameter (DCC model), e.g. `dcc_a="0.03"`. Default is `0.02`.

dcc_b=: (*optional*) set starting value of the β parameter (DCC model), e.g. `dcc_b="0.97"`. Default is `0.95`.

dcc_v=: (*optional*) set starting value of the degrees of freedom for Student t-distribution (DCC model), e.g. `dcc_v="9"`. Default is `7`.

constraints_a_dcc=: (*optional*) set constraints on the α parameter (DCC model), e.g. `constraints_a_dcc="0 1"`. Default is empty meaning no restrictions. If constraints are set on the α parameter then the errors are computed using the delta method.

constraints_b_dcc=: (*optional*) set constraints on the β parameter (DCC model), e.g. `constraints_b_dcc="0 1"`. Default is empty meaning no restrictions. If constraints are set on the β parameter then the errors are computed using the delta method.

constraints_v_dcc=: (*optional*) set constraints on the degrees of freedom for the Student t-distribution parameter (DCC model), e.g. `constraints_v_dcc="2 120"`. Default is empty meaning no restrictions. If constraints are set on the degrees of freedom then the errors are computed using the delta method.

grid_search=: (*optional*) whether to use a grid search over the set of the starting values of the parameters for the DCC model. Possible options are: `yes, no`, e.g. `grid_search="yes"`. Default is `no`. This option allows program to control for optimization errors. Poor choice of starting values can have paramount importance when it comes to achieving convergence. Having chosen this option active, the program try to estimate the parameters of the DCC model based on the grid of the starting values of the parameters. If for some reason optimization fails then the program changes the starting values of the parameters and tries to estimate the parameters of the DCC model once again but with a new set of the starting parameters. It does that consecutively until the optimization ends successfully or until it exhausts all possible starting values. The number of the tested starting values is reported in the final output table. The starting values specified by a user in `dcc_a, dcc_b, dcc_v` is set as the first set of the starting values. Details of the grid search In order to get the starting values which led to convergence refer to the auxiliary table `dcc_grid_errs`.

convergence=: (*optional*) set the convergence criterion, e.g. `convergence="0.00001"`. Default is `0.00001`.

max_ite=: (*optional*) set the maximum number of iterations, e.g. `max_ite="500"`. Default is `1000`.

trust=: (*optional*) set the initial trust region size, e.g. `trust="0.2"`. Default is `0.25`.

hess_approx_rgarch=: (*optional*) choose Hessian approximation method for (R)GARCH model. Possible options are: `Hessian, OPG`. The `OPG` stands for the Outer Products of Gradients. Default is `Hessian`, e.g. `hess_approx_rgarch="OPG"`.

hess_approx_dcc=: (*optional*) choose Hessian approximation method for DCC model. Possible options are: `BFGS, OPG, Numeric`, e.g. `hess_approx_dcc="OPG"`. Default is `BFGS`. It is worth noting that choosing `std_err="robust"` will override the parameter given in `hess_approx_dcc`.

step=: (*optional*) choose step method. Possible options are: `Marquardt, Dogleg, Line_search`, e.g. `step="Dogleg"`. Default is `Marquardt`.

show_out=: (*optional*) whether to show output table after successful estimation. Possible options are: `yes, no`, e.g. `show_out="yes"`. Default is `yes`.

overwrite=: (*optional*) whether to overwrite the existing output table with the new estimation results. Possible options are: `yes, no`, e.g. `overwrite="yes"`. Default is `yes`.

3.2 Outputs

dcc_coefs: the vector comprising parameters estimates for the DCC model. Depending on the assumed error distribution it is $\mathbf{\Theta_2} = [\alpha, \beta]'$ or $\mathbf{\Theta_2} = [\alpha, \beta, \upsilon]'$ where υ is the degrees of freedom for the Student t-distribution. If there are constraints on the parameters then the **dcc_coefs** comprises already re-transformed coefficients.

dcc_covar: the matrix comprising covariances of the parameter estimates for the DCC model. If there are constraints on the parameters then the **dcc_covar** comprises covariances pertaining to the parameters estimates already re-transformed. If errors are chosen to be robust (Huber-White, meaning `std_err="robust"`) then the **dcc_covar** comprises robust covariances.

dcc_rgarch, dcc_garch, dcc_ev_garch: the table with estimation outputs for the DCC-RGARCH, DCC-GARCH and DCC-GARCH-type (Built-in *EViews* GARCH-type) model, respectively. The estimation results for the univariate RGARCH, GARCH or GARCH-type models are firstly placed in the table. The assets are ordered as they were inputted in the first dialog. After these there is the estimation result for the DCC model. The options chosen by a user are displayed in the table. In Figure 3 there is an example of results obtained for the DCC model with Student t-distribution errors. Notice that results are formatted similarly like outputs of the `logl` objects. Additionally, there are restrictions imposed on the β parameter, but results report the translated back coefficients. The errors, z-statistics and p-values refer to translated back coefficients so the inference is valid. The standard errors are computed using the delta method.

dcc_hessian: the estimated Hessian matrix for the DCC model. If there are constraints on the parameters then the Hessian matrix pertains to the transformed parameters, not to the parameters after translating back. If `std_err="robust"` then the Hessian matrix is based on numeric derivatives.

dcc_ht_fcast: the matrix comprising one-period-ahead forecasts of the conditional variances and covariances from the DCC-(R)GARCH and DCC-GARCH-type models. It should be noted that one-period-ahead forecasts are obtained even if the sample equals to the full range of data. In such cases, one observation is added to the full range and as soon as the forecasts are obtained then one observation is dropped. Hence, the range is not changed.

dcc_logl: the series comprising the logarithm of likelihood function for the DCC model es-

144

145 ==========

146 DCC(1,1) (t-distribution)

147 Method: Maximum Likelihood (BFGS / Dogleg steps)

148 Date: 03.12.2020 Time: 12:44

149 Sample: 11/06/2013 11/06/2015

150 Included observations: 505

151 Evaluation order: By observation

152 Estimation settings: tol= 1e-05 derivs=numeric

153 Initial Values: ALPHA=0.02, BETA=0.95, T-DIST. DOF=9 with restrictions: 0<=BETA<=1

154 Convergence achieved after 13 iterations

155 Coefficient covariance computed using observed Hessian and the delta method

156

157		Coefficient	Std. Error	z-Statistic	Prob.
158					
159	ALPHA	0.021652	0.010930	1.980860	0.047607
160	BETA	0.786209	0.116393	6.754754	1.43E-11
161	T-DIST. DOF	11.39560	1.772429	6.429371	1.28E-10
162					
163	Log likelihood	-3271.982	Akaike info criterion		12.96626
164	Avg. log likelihood	-6.479172	Schwarz criterion		12.98299
165	Number of Coefs	3	Hannan-Quinn criter.		12.97283

166

167 Elapsed time: 4.674 seconds (or 0.0779 minutes)

168 Completed successfully

169

Figure 3: The DCC model results

timated at the given parameters.

dcc_mlse: the vector comprising coefficients errors for the DCC model. If there are constraints on the parameters then the **dcc_mlse** comprises coefficients errors pertaining to the parameters estimates already translated back. If errors are chosen to be robust (Huber-White, meaning `std_err="robust"`) then the **dcc_mlse** comprises robust errors.

dcc_pval: the vector comprising coefficients p-values for the DCC model. If there are constraints on the parameters then the **dcc_paval** comprises p-values already translated back.

dcc_r_fcast: the matrix comprising the one-period-ahead forecasts of conditional correlations from the DCC-(R)GARCH model.

dcc_zstat: the vector comprising z-statistics values for the DCC model. If there are constraints on the parameters then the **dcc_zstat** comprises z-statistics pertaining to the parameters estimates already translated back.

dcc_astat: the column vector comprising additional post-estimation statistics, measures and

other relevant values for the DCC model. It is given in the following order: the sum of the logarithm of the likelihood function evaluated at the estimated parameters, the average of the logarithm of the likelihood function evaluated at the estimated parameters, the number of coefficients, the Akaike information criterion, the Schwartz information criterion, the Hannan-Quinn information criterion, the included observations for the estimation (after required adjustments).

dcc_grid_errs: the table comprising all tested starting values of the parameters for the DCC model with their respective error messages. This table is created only when grid search is activated.

ht_init: the matrix comprising the initials values of the conditional variances on the main diagonal. The initial values are computed assuming the unconditional expectation and it is estimated as $\mathbf{H_0} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{r}_t\mathbf{r}_t'$, where $\mathbf{r}_t$ denotes $n \times 1$ vector of returns of n assets at time t.

parkinson_*: the series with the estimates of the Parkinson estimator (see [4]). The Parkinson estimator can be given as: $\sigma_P^2 = [\ln H_t/L_t]^2/(4\ln 2)$, where H_t and L_t denote the high and low prices over a fixed period of time. In these names the wildcard * stands for the assigned number of the time series. The assigned number is set based on the position in the `dep` parameter. The assigned numbers for the time series can be found in the table with estimation outputs (see **dcc_rgarch, dcc_garch, dcc_ev_garch**).

rgarch_*, garch_*, ev_garch_*: the *logl* object with the estimated model RGARCH, GARCH and GARCH-type (Built-in *EViews* GARCH-type) model, respectively. In these names the first wildcard * stands for the assigned number of the time series.

rgarch_ht_*, garch_ht_*, ev_garch_ht_*: the series with the conditional variances for the RGARCH, GARCH and GARCH-type (Built-in *EViews* GARCH-type) model, respectively. In these names the wildcard * stands for the assigned number of the time series.

rgarch_cov_*_*, garch_cov_*_*, ev_garch_cov_*_*: the series with the conditional covariances for the RGARCH, GARCH and GARCH-type (Built-in *EViews* GARCH-type) model, respectively. In these names the wildcard * stands for the assigned number of the time series e.g. `rgarch_cov_12` denotes the conditional covariances between the first and the second time series for the RGARCH model. The initial values are estimated as $\mathbf{COV_0} = E[\mathbf{e}_t\mathbf{e}_t']$ is the unconditional covariance matrix of the standardized errors $\mathbf{e}_t$.

rgarch_cor_*_*, garch_cor_*_*, ev_garch_cor_*_*: the series with the conditional correlations for the RGARCH, GARCH and GARCH-type (Built-in *EViews* GARCH-type) model, respectively. In these names the wildcard * stands for the assigned number of the time series. The initial values are estimated as $\mathbf{COR_0} = Cor[\mathbf{e}_t]$ is the unconditional correlation matrix of the standardized errors $\mathbf{e}_t$.

rgarch_res_*, garch_res_*, ev_garch_res_*: the series with the residuals for the RGARCH, GARCH and GARCH-type (Built-in *EViews* GARCH-type) model. In these names the wildcard * stands for the assigned number of the time series.

garch_spec: the string vector containing the GARCH-type model specifications which where given in the `ev_garch` parameter. This object is only created if the `model="EV_GARCH"` is selected.

d1: the auxiliary series to indicate the first observation in-the-sample. It is required for the estimation of the (R)GARCH models.

temp_out: the auxiliary table comprising estimation outputs done until the occurrence of an error of some kind. All the estimation outputs obtained so far are place here. The table can help trace back where the program terminated. It is only created if error occurs.

ymu_* the auxiliary series with de-meaned values of returns (for instance $ymu\_1 = r_{t,1} - \mu_{t,1}$). If there is no constant in the mean equation then **ymu_*** is the same as the returns. In these names the wildcard * stands for the assigned number of the time series.

zsqt_*: the auxiliary series to estimate the RGARCH, the GARCH and the GARCH-type model with the Student t-distributed errors (i.e. $zsqt\_^* = (ymu\_^*/\sqrt{h_t^*})^2$, where h_t^* is the conditional variance. In these names the wildcard * stands for the assigned number of the time series.

3.3 Examples

Let us consider the following examples:

Example 1.

```
dcc_rgarch(model="rgarch", dep="aapl_rt amzn_rt googl_rt gs_rt ibm_rt",
dep_high="aapl_high amzn_high googl_high gs_high ibm_high",
dep_low="aapl_low amzn_low googl_low gs_low ibm_low", tdist_garch="yes",
smpl="2/01/2008 31/12/2009", constraints_b_dcc="0 1")
```

In the first example above, the command will prompt the estimation of the DCC-RGARCH model with Student t-distributed errors for the RGARCH model and with constraints $[0, 1]$ on the β parameter i.e. $\beta \in [0, 1]$ (DCC model) for five assets namely: aapl_rt, amzn_rt, googl_rt, gs_rt, ibm_rt where the sample is 2/01/2008 31/12/2009.

Example 2.

```
dcc_rgarch(model="garch", smpl="1 500", dep="aapl_rt amzn_rt googl_rt gs_rt ibm_rt",
std_err="robust"
```

In the second example the add-in will estimate the DCC-GARCH model with Huber-White standard errors for the same five assets as before, but the sample is starting at 1 to 500 (undated data). Notice, that the order of the parameters is now different than in the first example, but still this is correct because the parameters do not have to be in a specific order.

Example 3.

```
dcc_rgarch(model="ev_garch", dep="aapl_rt amzn_rt googl_rt gs_rt ibm_rt",
```

```
ev_garch="(arch(1,1) aapl_rt c); (arch(1,1, egarch) amzn_rt c);
(arch(1,2) googl_rt c); (arch(1,1, tdist) gs_rt c); (arch(1,1) ibm_rt c ar(1))",
smpl="1 500"
```

The third example here is the one where the DCC-GARCH-type (Built-in *EViews* GARCH-type) model is estimated. The first time series i.e. aapl_rt follows the GARCH(1,1) model with constant in the mean, the second i.e. amzn_rt follows the EGARCH(1,1) model with constant in the mean, the third i.e. googl_rt follows the GARCH(1,2) model with constant in the mean, the fourth i.e. gs_rt follows the GARCH(1,1) model with Student t-distributed errors and the constant in the mean, the fifth i.e. ibm_rt follows the GARCH(1,1) model with the constant and the first order of autoregression AR(1) in the mean. The sample is starting from 1 observation to 500 (undated data). It means that the DCC model parameters are estimated based on the residuals coming from these five different GARCH-type models. Due to the AR(1) in the mean the DCC model parameters will be estimated based on 499 observations, not 500.

Example 4.

```
dcc_rgarch(model="rgarch", dep="aapl_rt amzn_rt googl_rt gs_rt ibm_rt",
dep_high="aapl_high amzn_high googl_high gs_high ibm_high",
dep_low="aapl_low amzn_low googl_low gs_low ibm_low", dcc_a="0.05", dcc_b="0.99",
max_ite="500", grid_search="yes", constraints_b_dcc="0 1")
```

The fourth example gives a glimpse of how the grid search for the DCC model works. The DCC-RGARCH model is estimated with the constraints $[0, 1]$ on the β parameter i.e. $\beta \in [0, 1]$ (DCC model) for five assets namely: aapl_rt, amzn_rt, googl_rt, gs_rt, ibm_rt where the sample is set to default (the current sample for the active page). The maximum number of iterations for the optimization algorithm is set to 500. The starting values for the DCC model are set to: $\alpha = 0.05$ and $\beta = 0.99$ which will result in an estimation error. Interestingly, the estimation is completed successfully thanks to the grid search option being activated. Figure 4 presents the table `dcc_grid_errs` with the grid search details. In the third row, we can see the starting values specified in the command line which resulted in the algorithm being unable to evaluate the likelihood function at starting parameters. In the next step, the add-in prompted the estimation with new starting values of the parameters (i.e. $\alpha = 0.05$ and $\beta = 0.9$). The optimization algorithm ended with convergence being achieved.

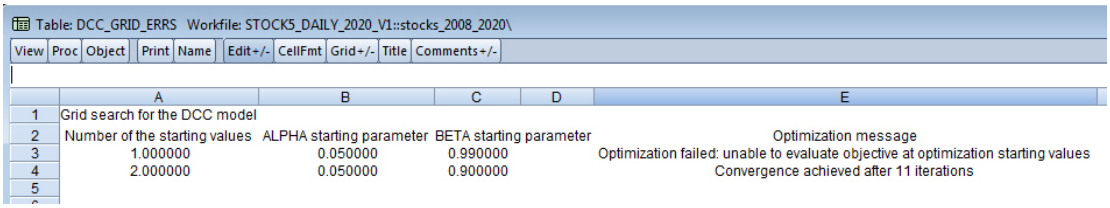
Table: DCC_GRID_ERRS Workfile: STOCK5_DAILY_2020_V1::stocks_2008_2020\

View | Proc | Object | Print | Name | Edit+/- | CellFmt | Grid+/- | Title | Comments+/-

	A	B	C	D	E
1	Grid search for the DCC model				
2	Number of the starting values	ALPHA starting parameter	BETA starting parameter		Optimization message
3	1.000000	0.050000	0.990000		Optimization failed: unable to evaluate objective at optimization starting values
4	2.000000	0.050000	0.900000		Convergence achieved after 11 iterations
5					

Figure 4: An example of the table `dcc_grid_errs` with the grid search output

4 Troubleshooting

The add-in has implemented different measures checking for errors. For instance, the following example shows a simple problem of different number of returns, high and low prices for the DCC-RGARCH model.

```
dcc_rgarch(model="rgarch", dep="aapl_rt amzn_rt googl_rt gs_rt ibm_rt",
dep_high="aapl_high amzn_high googl_high gs_high",
dep_low="aapl_low amzn_low googl_low gs_low ibm_low", tdist_garch="yes",
smpl="2/01/2008 31/12/2009", constraints_b_dcc="0 1")
```

The example above will obviously generate an error due to the fact that there is five series with returns (i.e. `dep="aapl_rt amzn_rt googl_rt gs_rt ibm_rt"`) and only four series with high prices (i.e. `dep_high="aapl_high amzn_high googl_high gs_high"`) deeming the RGARCH model's estimation impossible. The add-in will report error presented in Figure 5. The error message gives some explanation of the encountered error. On other note, the above command will not generate error if one chooses the DCC-GARCH model because the DCC-GARCH model does not require series with high and low prices.

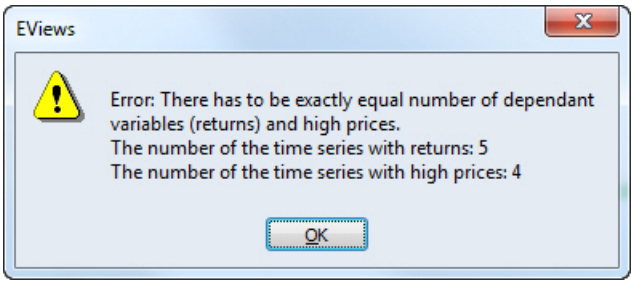

Figure 5: Error message

Even though there are measures to check for some possible errors, a user may encounter some other issues.

It is assumed that the order of series comprising the returns is the same as the order of series comprising high and low prices. If the order is not the same then you should expect doubtful coefficients or an estimation error.

In most cases the estimation of the DCC model encounters problem due to a poor choice of the starting values. You may deal with this issue by selecting the grid search procedure.

Another problem with the DCC model estimation is that there are some conditions on the parameters α and β to guarantee that $\mathbf{H}_t$, the covariance matrix, to be positive definite. Besides to the conditions for the univariate (R)GARCH model to ensure positive unconditional variances, the parameters α and β must satisfy the following: $\alpha \geq 0, \beta \geq 0, \alpha + \beta < 1$. If the parameters are not constrained then it is possible that the estimation could be terminated with an error. In such cases, we may check the logarithm of the likelihood function in the auxiliary vector `dcc_logl_vec`. We may see the coefficients in the vector `dcc_coefs` for which the estimation terminated. Note, if there are constraints on the parameters then the vector

`dcc_coefs` comprises the logit transformation of the parameters. The logit transformation of the coefficients is translated back until the estimation ends successfully.

The outliers in the data may also render the estimation impossible. In such cases it may be useful to check the logarithm of the likelihood function as well.

References

[1] Engle R. F., (2002), Dynamic Conditional Correlation – A Simple Class of Multivariate GARCH Models, Journal of Business and Economic Statistics, 20, 339–350

[2] Engle R. F., Shepprard K., (2001), Theoretical and Empirical Properties of Dynamic Conditional Correlation Multivariate GARCH, Working Paper 8554, National Bureau of Economic Research, 1050 Massachusetts Avenue, Cambridge, MA 02138, http://www.nber.org/papers/w8554

[3] Fiszeder, P., Fałdziński, M., Molnár, P., (2019), Range-based DCC models for covariance and value-at-risk forecasting, Journal of Empirical Finance, 54, 58-76, https://doi.org/10.1016/j.jempfin.2019.08.004

[4] Parkinson M., (1980), The Extreme Value Method for Estimating the Variance of the Rate of Return, The Journal of Business, 53, 1, 61-65.