DCC-GARCH

CG

2024-02-26

```
# need the data in matrix form
data <- as.matrix(mydata)

## Specifying the Univariate GARCH models. Needed for the volatilities of each of the time series
# Define a univariate GARCH(1,1) model specification with normal distribution
# try distribution.model = norm, snorm (skew normal), std (student t), sstd (skew student t)
# and nig (normal inverse gaussian)

uspec <- ugarchspec(variance.model=list(model="sGARCH", garchOrder=c(1,1)),
                    mean.model=list(armaOrder=c(0,0), include.mean=TRUE),
                    distribution.model="sstd")
```

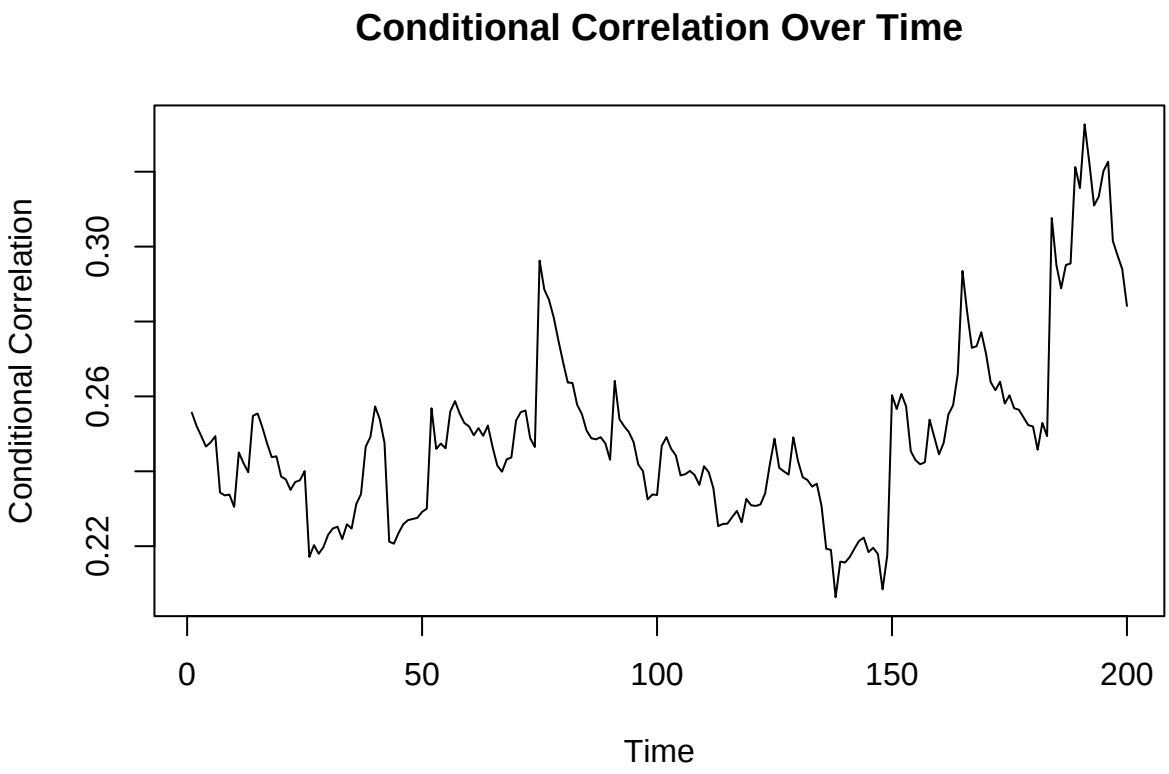
```
## Specifying  DCC(1,1) model
# can try c("mvnorm", "mvt", "mvlaplace")
dcc_spec <- dccspec(uspec = multispec(replicate(ncol(data), uspec)),
                    dccOrder = c(1,1),
                    distribution = "mvt")
```

```
# Fit the DCC model to  data
dcc_fit <- dccfit(dcc_spec, data = data)

# Summary of the fitted model
dcc_fit
```

```
##
## *---------------------------------*
## *          DCC GARCH Fit          *
## *---------------------------------*
##
## Distribution         :  mvt
## Model                :  DCC(1,1)
## No. Parameters       :  43
## [VAR GARCH DCC UncQ] : [0+30+3+10]
## No. Series           :  5
## No. Obs.             :  200
## Log-Likelihood       :  -1954.729
## Av.Log-Likelihood    :  -9.77
##
## Optimal Parameters
## ------------------------------------
##                   Estimate  Std. Error  t value Pr(>|t|)
```

```
## [Asset 1].mu     -0.115115    0.212927 -0.54063 0.588761
## [Asset 1].omega   0.505573    1.633705  0.30946 0.756968
## [Asset 1].alpha1  0.053873    0.049822  1.08130 0.279564
## [Asset 1].beta1   0.888300    0.245420  3.61951 0.000295
## [Asset 1].skew    0.980449    0.079215 12.37710 0.000000
## [Asset 1].shape   4.887769    1.973898  2.47620 0.013279
## [Asset 2].mu     -0.271895    0.139566 -1.94815 0.051398
## [Asset 2].omega   0.291065    0.274926  1.05871 0.289734
## [Asset 2].alpha1  0.113261    0.055923  2.02532 0.042835
## [Asset 2].beta1   0.841138    0.099586  8.44631 0.000000
## [Asset 2].skew    0.960598    0.138255  6.94802 0.000000
## [Asset 2].shape   6.114270    2.417940  2.52871 0.011448
## [Asset 3].mu      0.177517    0.125374  1.41590 0.156806
## [Asset 3].omega   0.065925    0.145648  0.45263 0.650814
## [Asset 3].alpha1  0.058267    0.039606  1.47115 0.141250
## [Asset 3].beta1   0.933802    0.068719 13.58866 0.000000
## [Asset 3].skew    1.101812    0.116094  9.49069 0.000000
## [Asset 3].shape   5.614324    2.109457  2.66150 0.007779
## [Asset 4].mu     -0.052908    0.110681 -0.47802 0.632635
## [Asset 4].omega   0.019787    0.164594  0.12022 0.904309
## [Asset 4].alpha1  0.025640    0.025487  1.00601 0.314409
## [Asset 4].beta1   0.973360    0.121546  8.00816 0.000000
## [Asset 4].skew    1.002601    0.205352  4.88235 0.000001
## [Asset 4].shape   3.177197    3.044910  1.04344 0.296742
## [Asset 5].mu      0.027882    0.091395  0.30507 0.760312
## [Asset 5].omega   0.066235    0.043049  1.53861 0.123899
## [Asset 5].alpha1  0.017417    0.020626  0.84443 0.398430
## [Asset 5].beta1   0.941473    0.019462 48.37402 0.000000
## [Asset 5].skew    1.096845    0.105288 10.41758 0.000000
## [Asset 5].shape  14.580886   11.313510  1.28880 0.197467
## [Joint]dcca1      0.008793    0.009674  0.90888 0.363415
## [Joint]dccb1      0.877182    0.052373 16.74876 0.000000
## [Joint]mshape     7.508430    1.389627  5.40320 0.000000
##
## Information Criteria
## ---------------------
##
## Akaike       19.977
## Bayes        20.686
## Shibata      19.905
## Hannan-Quinn 20.264
##
##
## Elapsed time : 5.268288
```

```r
# extracting correlations
correlations <- rcor(dcc_fit, type = "cor")
corr_df = data.frame(correlations)

# Example for plotting the conditional correlation between company 1 and company 2 over time
plot(correlations[1,2,], type = 'l', xlab = "Time", ylab = "Conditional Correlation", main = "Conditiona
```




```r
# information criteria
info_crit <- infocriteria(dcc_fit)
```

```r
# Initialize a list to store the information criteria for different DCC GARCH models
DCC_InfoCrit <- list()

# Looping through different orders for DCC and GARCH models
for (dcc_order in 1:2) {
  for (garch_order in 1:2) {
    # Define the univariate GARCH model specification for each series
    uspec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(garch_order, garch_order
                        mean.model = list(armaOrder = c(0,0), include.mean = TRUE),
                        distribution.model = "norm")

    # Specify the DCC GARCH model
    dcc_spec <- dccspec(uspec = multispec(replicate(ncol(data), uspec)),
                        dccOrder = c(dcc_order, dcc_order),
                        distribution = "mvnorm")

    # Fit the DCC model to the data
    dcc_fit <- dccfit(dcc_spec, data = data, solver = "solnp", fit.control = list(eval.se = TRUE))

    # Extract the information criteria for the fitted model
    info_crit <- infocriteria(dcc_fit)

    # Store the information criteria in the list
```

```
    model_name <- paste("DCC", dcc_order, dcc_order, "GARCH", garch_order, garch_order, sep = "-")
    DCC_InfoCrit[[model_name]] <- info_crit
  }
}

# Print the information criteria for all DCC GARCH models
infoCrit = data.frame(DCC_InfoCrit)
print(infoCrit)
```

```
##                    V1     V1.1     V1.2     V1.3
## Akaike       19.96096 20.03599 19.96683 20.04779
## Bayes        20.48870 20.72864 20.52754 20.77342
## Shibata      19.91860 19.96665 19.91950 19.97244
## Hannan-Quinn 20.17453 20.31630 20.19374 20.34145
```