Package Development in R: Implementing GO-GARCH models

Dr. Bernhard Pfaff
bernhard_pfaff@fra.invesco.com

Invesco Asset Management Deutschland GmbH, Frankfurt am Main

Rmetrics: Meielisalp Workshop 2009
06/28/ – 07/02/2009
Meielisalp, Lake Thune, Switzerland

Contents

Introduction

Overview

- Task: From academic paper to implementation.
- Putting econometric methods/models into action.
- Design of a program structure that meets the needs of the user: form follows function.
- Helpful tools to fulfill this task.
- Example: Multivariate GARCH model class.

Literature

- Bauwens, L., Laurent, S. and Rombouts, J. (2006), Multivariate GARCH models: a survey, *Journal of Applied Econometrics*, **21(1)**, 79 - 109
- Boswijk, H. P. and van der Weide, R. (2006), Wake me up before you GO-GARCH, *Tinbergen Institute Discussion Paper*, **TI 2006-079/4**, University of Amsterdam and Tinbergen Institute.
- Boswijk, H. P. and van der Weide, R. (2009), Method of Moments Estimation of GO-GARCH Models, *Working Paper*, University of Amsterdam, Tinbergen Institute and World Bank.
- Broda, S.A. and Paolella, M.S. (2008), CHICAGO: A Fast and Accurate Method for Portfolio Risk Calculation, *Swiss Finance Institute*, Research Paper Series **08-08**, Zürich.
- Pfaff, B. (2008), VAR, SVAR and SVEC Models: Implementation Within R Package vars, *Journal of Statistical Software*, **27(4)**, http://www.jstatsoft.org/v27/i04/.
- Pfaff, B. (2008), *Analysis of Integrated and Cointegrated Time Series with R*, Second Edition, Springer, New York.
- Van der Weide, R. (2002), GO-GARCH: A Multivariate Generalized Orthogonal GARCH Model, *Journal of Applied Econometrics*, **17(5)**, 549 - 564.

Introduction

Multivariate GARCH

Generally, a *m*-dimensional GARCH for a second-order stationary process $\{x_t\}$ is given as:

$$x_t|\mathfrak{I}_{t-1} \sim \mathfrak{D}(\phi, V_t) \; , \tag{1}$$

whereby the information set up to time $t-1$ is denoted by $\mathfrak{I}_{t-1}$ and the variance-covariance matrix V_t is time dependent. The information set contains lagged values of the squares and cross-products of x_t and elements of the conditional covariance matrices. The vector ϕ contains distribution dependent parameters. Ordinarily for $\mathfrak{D}$ a Normal distribution is assumed.

Introduction

Typology

According to the survey article of Bauwens *et al* (2006) multivariate GARCH models can be classified into three areas:

- Direct generalizations of the univariate GARCH model, *e.g.* VEC, BEKK and factor models.
- Linear combinations of univariate GARCH models, *e.g.* (generalized) orthogonal and latent factor GARCH models.
- Nonlinear combinations of univariate GARCH models, *e.g.* dynamic conditional correlation and general dynamic covariance models.

Introduction

Resources in R

- Univariate:[1]
  1. Package bayesGARCH (CRAN).
  2. Rmetrics package fGarch (CRAN & R-Forge).
  3. Package rgarch (R-Forge).
  4. Function `garch()` in package tseries (CRAN).
- Multivariate:
  1. Package ccgarch (CRAN).
  2. Package gogarch (CRAN & R-Forge).
  3. Package rgarch (R-Forge).

[1] For the sake of completeness a project RRegArch has been registered on R-Forge, but only an empty project skeleton exists as of the time this presentation is drafted.

GO-GARCH

Model

The observed m-dimensional economic process $\{x_t\}$ is governed by a linear combination of uncorrelated economic components $\{y_t\}$:

$$x_t = Zy_t \tag{2}$$

The linear map Z that links the unobserved components with the observed variables is assumed to be constant over time, and invertible. The unobserved components are normalized to have unit variance, such that:

$$V = \mathbb{E}[x_t x_t'] = ZZ' \tag{3}$$

GO-GARCH

Example GO-GARCH(1, 1)

An explicit example, whereby the Normal distribution is assumed would then be:

$$x_t = Zy_t \text{ with } y_t \sim \mathcal{N}(0, H_t) \tag{4}$$

and each component is described by a GARCH(1, 1) process:

$$H_t = diag(h_{1,t}, \dots, h_{m,t}) \tag{5}$$

$$h_{i,t} = \omega_i + \alpha_i y_{i,t-1}^2 + \beta_i h_{i,t-1} \text{ , for } i = 1, \dots, m \tag{6}$$

The unconditional covariance matrix of the components is $H_0 = I$ and the conditional covariances of $\{x_t\}$ are given by $V_t = ZH_tZ'$.

GO-GARCH

The quest for Z, part I

Let Z be the map that links the uncorrelated components $\{y_t\}$ with the observed process $\{x_t\}$. Then there exists an orthogonal matrix U_0 such that:

$$P\Delta^{\frac{1}{2}}U_0 = Z \tag{7}$$

The matrices P and $\Delta^{\frac{1}{2}}$ can be retrieved by SVD from sample information, *i.e*, the unconditional variance/covariance matrix of $\{x_t\}$.
And what about U_0?

GO-GARCH

The quest for Z, part II

One parametrization of U is given as follows:
Every m-dimensional orthogonal matrix U with $\det(U) = 1$ can be represented as a product of $\binom{m}{2} = [m(m-1)]/2$ rotation matrices:

$$U = \prod_{i<j} R_{ij}(\theta_{ij}) \; with \; -\pi \leq \theta_{ij} \leq \pi \;, \tag{8}$$

where $R_{ij}(\theta_{ij})$ performs a rotation in the plane spanned by e_i and e_j over an angle θ_{ij}; the so-called Euler angles.

GO-GARCH

Estimation by ...

- Maximum-Likelihood, see van der Weide (2002).
- Non-linear Least-Squares, see Boswijk and van der Weide (2006).
- Methods-of-Moments, see Boswijk and van der Weide (2009).
- Fast ICA, see Broda and Paolella (2008).

Design

Guidelines

- Think of a model class in terms of an object, *i.e.*, GO-GARCH.
- Write methods for estimating your model, retrieving and/or displaying items from that object.
- Provide the user with as many options for a tailor-made usage.
- Adhere to a naming convention for classes and functions.
- Error checking and handling as early as possible.
- Use coercing where appropriate and possible.
- Employ checks, tests and/or validation.

Design

Classes

- Class definition for orthogonal matrices: *Orthom*.
- Class for initial model object: *Goinit*.
- Class for GO-GARCH models: *GoGARCH* (inherits from *Goinit*)
- Dependent upon the choosen estimation method (inherits from *GoGARCH*):
  1. Fast ICA: *Goestica*
  2. Methods of Moments: *Goestmm*
  3. Maximum-Likelihood: *Goestml*
  4. Non-linear Least-Squares: *Goestnls*
- Class for `summary` objects: *Gosum*
- Class for `predict` objects: *Gopredict*

Design

Methods

Think of methods as the answers to the question: "What does the user wants or is interested in?".

- Estimation (`goest`) and displaying the result, *i.e.*, `show`, `print`, `summary`.
- Displaying the result graphically, *i.e.*, `plot`.
- Update the model's structure and/or estimation method, *i.e.*, `update`.
- Retrieval the conditional variances, *i.e.*, `cvar`, `ccor`, `ccov`.
- Calculate forecasts, *i.e.*, `predict`, and the predicted conditional variances should be made available (`cvar`, `ccor`, `ccov`).

Design

Functions

- Main functions for creating objects of a certain formal class: `gogarch` and `goinit`.
- Auxiliary functions for clearer code appearance, *i.e.* objective function to optimize (`gollh`, `gonls`), manipulation of objects (`unvech`, `Umatch`, `UprodR`, `Rd2`, `gotheta`).
- Validation: `validGoinitObject` for objects of class *Goinit* and `validOrthomObject` for objects of class *Orthom*.

Hint: Prefix function's name with a dot if it should not be documented in a package.

Implementation

Key Points

- Package gogarch purely written in R.
- Current version 0.6-9 as of 2009-04-29.
- S4 classes/methods and `NAMESPACE` employed.
- Dependencies: R ($>=$ 2.7.0), methods, stats, graphics, fGarch, fastICA
- License: GPL ($>=$ 2)
- Available at:
  1. http://cran.r-project.org
  2. http://r-forge.r-project.org/projects/gogarch/
  3. http://www.pfaffikus.de

Implementation

Guidelines

- Release your code early and often.
- Package your code as early as possible.
- Employ a source control software, *e.g.* Subversion.
- Document your code (within the functions and in form of a manual).
- Provide examples and/or a "How to" and/or a vignette.
- Publicize your package, *e.g.* JSS http://www.jstatsoft.org/.
- If possible, replicate trustworthy results from published articles.

Implementation

Package's Structure

Methods	Classes								
	Goinit	GoGARCH	Goestica	Goestmm	Goestml	Goestnls	Gosum	Gopredict	Orthom
angles					×				
ccor		×	×	×	×	×		×	
ccov		×	×	×	×	×		×	
converged		×	×	×	×	×			
coef		×	×	×	×	×			
cvar		×	×	×	×	×		×	
formula		×	×	×	×	×			
logLik					×				
M									×
plot		×	×	×	×	×			
predict		×	×	×	×	×			
print									×
resid		×	×	×	×	×			
residuals		×	×	×	×	×			
show	×	×	×	×	×	×	×	×	×
summary		×	×	×	×	×			
t									×
update		×	×	×	×	×			

Implementation

Economic coding, Ia

```
> getMethod("coef", "GoGARCH")

Method Definition:

function (object, ...)
{
    .local <- function (object)
    {
        garchc <- matrix(unlist(lapply(object@models, coef)),
            nrow = ncol(object@X), byrow = TRUE)
        colnames(garchc) <- names(object@models[[1]]@fit$par)
        rownames(garchc) <- paste("y", 1:nrow(garchc), sep = "")
        return(garchc)
    }
    .local(object, ...)
}
<environment: namespace:gogarch>

Signatures:
        object
target  "GoGARCH"
defined "GoGARCH"
```

Implementation

Economic coding, Ib

Due to inheritance one only needs:

```
> getMethod("coef", "Goestmm")

Method Definition:

function (object, ...)
{
    .local <- function (object)
    {
        callNextMethod()
    }
    .local(object, ...)
}
<environment: namespace:gogarch>

Signatures:
        object
target  "Goestmm"
defined "Goestmm"
```

which results from the sources:

```
setMethod(f = "coef", signature(object = "Goestmm"), definition = function(object){
  callNextMethod()
})
```

Implementation

Economic coding, IIa

```
> getMethod("cvar", "GoGARCH")

Method Definition:

function (object, ...)
{
    .local <- function (object)
    {
        m <- ncol(object@X)
        n <- nrow(object@X)
        cvar <- matrix(c(unlist(lapply(object@H, function(x) diag(x)))),
            ncol = m, nrow = n, byrow = TRUE)
        colnames(cvar) <- paste("V.", colnames(object@X), sep = "")
        rownames(cvar) <- rownames(object@X)
        cvar <- as.ts(cvar)
        return(cvar)
    }
    .local(object, ...)
}
<environment: namespace:gogarch>

Signatures:
        object
target  "GoGARCH"
defined "GoGARCH"
```

Implementation

Economic coding, IIb

Due to inheritance one only needs:

```
> getMethod("cvar", "Goestmm")

Method Definition:

function (object, ...)
{
    .local <- function (object)
    {
        cvar(as(object, "GoGARCH"))
    }
    .local(object, ...)
}
<environment: namespace:gogarch>

Signatures:
        object
target  "Goestmm"
defined "Goestmm"
```

which results from the sources:

```
setMethod(f = "cvar", signature(object = "Goestmm"), definition = function(object){
  cvar(as(object, "GoGARCH"))
})
```

Implementation

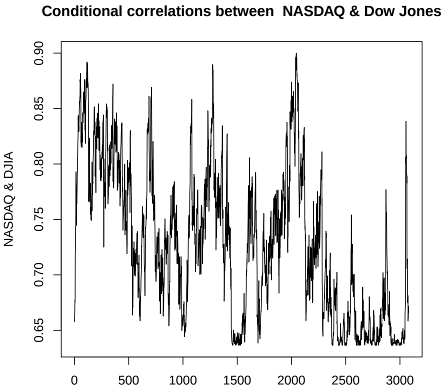
Example Ia, van der Weide (2002)

```
> library(gogarch)
> library(vars)
> data(VDW)
> var1 <- VAR(scale(VDW), p = 1,
+         type = "const")
> resid <- residuals(var1)
> gogml <- gogarch(resid, ~garch(1, 1),
+          scale = TRUE, estby = "ml",
+          control = list(iter.max = 1000))
> gogml
> solve(gogml@Z)
> plot(gogml)
```

```
****************
*** GO-GARCH ***
****************

Components estimated by: maximum likelihood
Dimension of data matrix: (3081 x 2).
Formula for component GARCH models: ~ garch(1, 1)

Orthogonal Matrix U:
          [,1]       [,2]
[1,] 0.6554851 -0.7552081
[2,] 0.7552081  0.6554851

Linear Map Z:
           [,1]       [,2]
[1,] -0.2766750 -0.9607947
[2,] -0.9123176 -0.4090867

Estimated GARCH coefficients:
        omega     alpha1     beta1
y1 0.00303708 0.09098502 0.9071784
y2 0.01001433 0.05231197 0.9401893

Convergence codes of component GARCH models:
y1 y2
 1  1
```

Implementation

Example Ib, van der Weide (2002)



Implementation

Example II, Boswijk and van der Weide (2006)

```
> data(BVDW)
> BVDW <- zoo(x = BVDW[, -1],
+             order.by = BVDW[, 1])
> BVDW <- diff(log(BVDW)) * 100
> gognls <- gogarch(BVDW, formula =
+           ~garch(1,1), scale = TRUE,
+           estby = "nls")
> gognls
```

```
****************
*** GO-GARCH ***
****************

Components estimated by: non-linear Least-Squares
Dimension of data matrix: (2609 x 2).
Formula for component GARCH models: ~ garch(1, 1)

Orthogonal Matrix U:
           [,1]       [,2]
[1,] -0.5241201 -0.8516443
[2,]  0.8516443 -0.5241201

Linear Map Z:
          [,1]       [,2]
[1,] 0.8141711 -0.5802948
[2,] 0.1511830 -0.9883119

Estimated GARCH coefficients:
         omega     alpha1     beta1
y1 0.009343711 0.08749236 0.9049185
y2 0.005866507 0.04367926 0.9520289

Convergence codes of component GARCH models:
y1 y2
 0  1
```

Implementation

Example III, Boswijk and van der Weide (2009)

```
> data(BVDWSTOXX)
> BVDWSTOXX <- zoo(x = BVDWSTOXX[, -1],
+             order.by = BVDWSTOXX[, 1])
> BVDWSTOXX <- window(BVDWSTOXX,
+             end = as.POSIXct("2007-12-31"))
> BVDWSTOXX <- diff(log(BVDWSTOXX))
> sectors <- BVDWSTOXX[, c("AutoParts",
+           "Banks", "OilGas")]
> sectors <- apply(sectors, 2, scale,
+           scale = FALSE)
> gogmm <- gogarch(sectors, formula =
+         ~garch(1,1), estby = "mm",
+         lag.max = 100)
> gogmm
```

```
****************
*** GO-GARCH ***
****************

Components estimated by: Methods of Moments
Dimension of data matrix: (5420 x 3).
Formula for component GARCH models: ~ garch(1, 1)

Orthogonal Matrix U:
            [,1]       [,2]      [,3]
[1,]  0.97243228 -0.1580023 0.1714956
[2,]  0.04054402  0.8388089 0.5429142
[3,] -0.22963370 -0.5209942 0.8220909

Linear Map Z:
            [,1]        [,2]         [,3]
[1,] 0.012150308 0.006416573 -0.003053992
[2,] 0.003914599 0.010216543 -0.003626263
[3,] 0.004364691 0.008865709  0.006737270

Estimated GARCH coefficients:
         omega     alpha1     beta1
y1 0.013982128 0.06081931 0.9256579
y2 0.003267436 0.04320838 0.9542282
y3 0.021353037 0.07085827 0.9065798

Convergence codes of component GARCH models:
y1 y2 y3
 1  1  1
```