MARKOV-SWITCHING GARCH MODELS IN R: THE MSGARCH PACKAGE

Keven Bluteau

joint work with:

David Ardia
Kris Boudt
Leopoldo Catania
Brian Peterson
Denis-Alexandre Trottier

R/Finance 2017, May 19-20
https://CRAN.R-project.org/package=MSGARCH
IN BRIEF

- **MSGARCH** implements Haas et al. (2004a) specification:

  2. Possibly *K* separate conditional distributions.
  3. A Markov chain dictates the switches between regimes.
  4. Assumes a zero mean process.

- Core of the package is in C++ (thanks to **Rcpp**) to allow for fast and efficient computations.
- Easy estimation and specification creation (similar to **rugarch**).
- Functionality for visualization, simulation, model selection, and risk measure forecasting.
### VOLATILITY MODELS

<table>
<thead>
<tr>
<th>Conditional volatility models</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GARCH model (model = &quot;sGARCH&quot;)</strong></td>
<td></td>
</tr>
<tr>
<td>$h_t \equiv \alpha_0 + \alpha_1 y_{t-1}^2 + \beta h_{t-1}$</td>
<td></td>
</tr>
<tr>
<td><strong>EGARCH model (model = &quot;eGARCH&quot;)</strong></td>
<td></td>
</tr>
<tr>
<td>$\ln(h_t) \equiv \alpha_0 + \alpha_1(</td>
<td>y_{t-1} - E[</td>
</tr>
<tr>
<td><strong>GJR model (model = &quot;gjrGARCH&quot;)</strong></td>
<td></td>
</tr>
<tr>
<td>$h_t \equiv \alpha_0 + \alpha_1 y_{t-1}^2 + \alpha_2 y_{t-1}^2 1_{{y_{t-1}&lt;0}} + \beta h_{t-1}$</td>
<td></td>
</tr>
<tr>
<td><strong>TGARCH model (model = &quot;tGARCH&quot;)</strong></td>
<td></td>
</tr>
<tr>
<td>$h_t^{1/2} \equiv \alpha_0 + \alpha_1 y_{t-1} 1_{{y_{t-1} \geq 0}} + \alpha_2 y_{t-1} 1_{{y_{t-1}&lt;0}} + \beta h_{t-1}^{1/2}$</td>
<td></td>
</tr>
<tr>
<td><strong>GAS model (model = &quot;GAS&quot;)</strong></td>
<td></td>
</tr>
</tbody>
</table>
| $h_t \equiv \alpha_0 + \alpha_1 s_{t-1} + \beta h_{t-1}$,  
$s_{t-1} \equiv S_{t-1} \nabla_{t-1}$,  
$\nabla_{t-1} \equiv \frac{\partial \ln f(y_{t-1}|h_{t-1}, \lambda)}{\partial h_{t-1}}$,  
$S_{t-1} \equiv \mathbb{E}[\nabla_{t-1} \nabla_{t-1}']^{-1}$ |  |

Bollerslev (1986)  
Nelson (1991)  
Glosten et al. (1993)  
Zakoian (1994)  
Creal et al. (2013)
CONDITIONAL DISTRIBUTIONS

Conditional distributions

Normal distribution (distribution = "norm")
\[ f_N(z) \equiv \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}, \quad z \in \mathbb{R} \]

Student–t distribution (distribution = "std")
\[ f_S(z; \nu) \equiv \sqrt{\frac{\nu}{\nu-2}} \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu \pi} \Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{z^2}{\nu}\right)^{-\frac{\nu+1}{2}}, \quad z \in \mathbb{R} \]

GED distribution (distribution = "ged")
\[ f_{GED}(z; \nu) \equiv \frac{\nu e^{-\frac{1}{2}|z/\lambda|^\nu}}{\lambda^{1+1/\nu} \Gamma(1/\nu)}, \quad \lambda \equiv \left(\frac{\Gamma(1/\nu)}{4^{1/\nu} \Gamma(3/\nu)}\right)^{1/2}, \quad z \in \mathbb{R} \]

- Skewed versions also available using the Fernández and Steel (1998) transformation.
CREATING A SPECIFICATION

– First step is to create a specification

> create.spec(model, distribution, do.skew, do.mix, do.shape.ind)

– Inputs:
  – model: “sGARCH”, “eGARCH”, “gjrGARCH”, “tGARCH”, “GAS”
  – distribution: “norm”, “std”, “ged”
  – do.skew: Skewed distribution Boolean.
  – do.shape.ind: Make it so that only the conditional volatility models switches (distribution and shape parameter stays the same across regime).
EXAMPLES

– Simple GARCH(1,1) normal:

```r
> spec = create.spec(model = "sGARCH", distribution = "norm")
```

– Two-state MSGARCH model with GARCH(1,1) normal in both regimes:

```r
> spec = create.spec(model = c("sGARCH", "sGARCH"),
+ distribution = c("norm", "norm"))
```

– Complex MSGARCH model:

```r
> spec = create.spec(model = c("sGARCH", "tGARCH", "eGARCH"),
+ distribution = c("norm", "std", "ged"),
+ do.skew = c(TRUE, FALSE, TRUE))
```
WHAT IS INSIDE?

A specification is an S3 R class that gives you access to all the **MSGARCH** functionalities.

- Embedded C++ templated class inside. Why?
  - C++: Fast calculations.
  - Templated: Easy future extensions.
    - This means adding conditional volatility models and conditional distributions with minimal work (and debugging).
ILLUSTRATION – DATA

- SMI log-returns from 1990-11-12 to 2000-10-20.

```r
> require(DEoptim)
> data(SMI)
> plot(SMI)
```
ILLUSTRATION – MLE ESTIMATION

- Make use of `DEoptim` (global) & `nmkb` from `dfoptim` (local)

```r
> out.mle = fit.mle(spec = spec, y = SMI, ctrl = list(do.init = TRUE))
> summary(out.mle)
```

```
[1] "DEoptim initialization: TRUE"
[1] "Fitted Parameters:"
   alpha0.1  alpha1.1 alpha2.1  beta1  nu1  xi1  alpha0.2
[1,] 0.2226311 0.001360713 0.212886 0.5401085 5.943159 0.8521142 0.08298225
   alpha1.2  alpha2.2  beta2  nu2  xi2
[1,] 0.006268897 0.1393344 0.8773668 20.0118 0.8581828 0.9980548 0.003125602
[1] "Transition matrix:"
   t = 1   t = 2
t + 1 = 1 0.998054778 0.003125602
  t + 1 = 2 0.001945222 0.996874398
[1] "Stable probabilities:"
  Stable probabilities
State 1   0.5463842
State 2   0.4536158
[1] "Unconditional volatility:"
  State 1  State 2
[1,] 0.8101812  1.422818
Log-kernel: -3364.587
AIC:  6687.677
BIC:  6769.213
```
ILLUSTRATION – BAYESIAN ESTIMATION

– Make use of adaptMCMC

```r
> ctr.mcmc = list(N.burn = 5000, N.mcmc = 10000, N.thin = 10, theta0 = out.mle$theta)
> out.mcmc = fit.bayes(spec = spec, y = SMI, ctr = ctr.mcmc)
generate 15000 samples
|=================================================================| 100%
> summary(out.mcmc)
[1] "Bayesian posterior mean:"
        alpha0_1 alpha1_1 alpha2_1   beta_1   nu_1  xi_1
  0.213301726 0.018197992 0.22144331 0.535624781 5.962895732 0.862990160
        alpha0_2 alpha1_2 alpha2_2   beta_2   nu_2  xi_2
  0.073552525 0.015786970 0.133891540 0.878384572 19.989053308 0.852081140

[1] "Posterior mean transition matrix:"
  t   t

  t + 1 = 1 0.996794482 0.004340049
  t + 1 = 2 0.0003205518 0.995659951

[1] "Posterior mean stable probabilities:"

    State 1    State 2

    Stable probabilities

    State 1    0.5399292
    State 2    0.4600708

[1] "Posterior mean unconditional volatility:"

    State 1    State 2

    [1,] 0.8124884 1.489994

    Acceptance rate: 0.988

    AIC:  6690.017
    BIC:  6771.553
    DIC:  6674.852
```
AND SO WHAT?

- Available functionalities:
  - Filtered volatilities.
  - Filtered probabilities.
  - 1-step ahead simulation.
  - Predictive density.
  - Risk measures (VaR and ES).
  - Information criteria.
  - And more!

- All functionalities are compatible for both MLE and Bayesian estimation.
ILLUSTRATION – VOLATILITIES & STATE

> vol = ht(out.mle)
> plot(vol, date = index(SMI))

> state = Pstate(out.mle)
> plot(state, date = index(SMI))
1. **Object** can take a specification:
   1. In case of a specification, `theta` and `y` must be provided.
   2. Useful when using the same fitted model on new data `y`.

2. **Object** can take a fitted model:
   1. No need to input `theta` and `y`.
   2. Useful shortcut.

3. The variable `x` are what we want to evaluate.

4. If `do.its = TRUE`, `x` is not needed as we evaluated the function with the in-sample observation (in-sample).

5. If `do.its = FALSE`, `x` is evaluated as a 1-step ahead draws.

---

**Log-likelihood function:**

```r
> log_like = pred(object = out.mle,
+                  log = TRUE, do.its = TRUE)
> sum(log_like$pred, na.rm = TRUE) # first observation = NA
[1] -3329.838
```

**Use kernel() to include the priors:**

```r
> kernel(out.mle)
[1] -3364.587
```
ILLUSTRATION – PREDICTIVE DENSITY

```r
> grid = seq(-4,4,length.out = 1000)
> pred_dist = pred(out.mle, x = grid,
+     log = FALSE, do.its = FALSE)
> plot(pred_dist)
```

```r
> grid = seq(-4,4,length.out = 1000)
> pred_dist = pred(out.mcmc, x = grid,
+     log = FALSE, do.its = FALSE)
> plot(pred_dist)
```
ILLUSTRATION – RISK MEASURES

> risk(object, theta, y, level, ES, do.its, ctr)

- The risk function works similarly to the pred function.
- It also leverages the pred function to calculate risk measures.
- do.its = TRUE will calculate the in-sample risk measures for all dates.
- do.its = FALSE will calculate the one-step ahead risk measures.

> VAR95 = risk(out.mle.2,level = c(0.95), ES = TRUE, do.its = TRUE)
> plot(VAR95,date = index(SMI))
WHAT NEXT?

– Google Summer of Code 2017 (Leopoldo Catania).
– Wish list:
  – Improved starting value strategy for faster optimization.
  – Multi-step ahead forecasts (by simulation).
  – Parameters constraints.
  – Standard errors of the estimates (MLE).
  – Custom MLE and MCMC optimizers (including custom priors).
  – Multivariate model with regime-switching copulas.
  – And more!

Some are currently implemented in MSGARCH 0.18.4 (available on GitHub)!
Thanks for your attention and hope you’ll enjoy our package!!

https://CRAN.R-project.org/package=MSGARCH
https://github.com/keblu/MSGARCH