Hétéroscédasticité conditionnelle (compléments du Chapitre 12)

Yves Aragon*
Université Toulouse Capitole

12 mai 2020

Exercice 12.1
Faire apparaître A partir du rendement composé quotidien $r_t = \log(x_t) - \log(x_{t-1})$ calculer le rendement composé sur une semaine de 5 jours.

Réponse. Le rendement composé sur 5 jours est

$$\begin{aligned} & r_{5,t} = \log(x_t) - \log(x_{t-5}) \\ & = (\log(x_t) - \log(x_{t-1})) + (\log(x_{t-1}) - \log(x_{t-2}) + \cdots + (\log(x_{t-4}) - \log(x_{t-5})) \end{aligned} \quad (12.1)$$

il est donc la somme des rendements quotidiens des jours de la période.

Exercice 12.2
— Simuler 300 observations de y_t obéissant à :

$$y_t = 5 + \varepsilon_t \quad (12.2)$$
$$z^2\varepsilon_t = \sigma_t z_t,\ z_t \sim \text{BBN}(0, \sigma_z^2) \quad (12.3)$$
$$\sigma_t^2 = 0.1 + 0.9\varepsilon_{t-1}^2 \quad (12.4)$$

(On simulera 10 valeurs avant de commencer à stocker la série et on initialisera à 0 la première variance conditionnelle.) On reproduira explicitement les équations. Visualiser la série et recommencer les simulations avec différentes graines.
— Tester la blancheur du carré centré de la série. Commenter.

Réponse. Il nous faut essentiellement : (1) tirer les valeurs de z_t i.i.d. $\mathcal{N}(0,1)$, (2) calculer les variances conditionnelles par récurrence. Après quoi on peut calculer y_t.

```
> require(caschrono)
> set.seed(1618) ; nobs = 300
> zt = rnorm(nobs+10)
> # initialisation
> sig2 = rep(0, nobs+10)
> epsil = sig2
```

*yves.aragon@gmail.com

```
> y = sig2
> for(i in 2:(nobs+10))
+  {sig2[i] <- 0.1 + 0.9*epsil[i-1]^2
+   epsil[i] <- sig2[i]^.5 *zt[i]
+   y[i] <- 5+epsil[i]
+  }
> y <- y[-(1:10)]
```

Pour obtenir les graphiques on lancera le code

```
plot.ts(y)
acf(y)
```

Exercice 12.3 (Test d'hétéroscédasticité conditionnelle basé sur un test de blancheur)
Tester l'hétéroscédasticité conditionnelle de ces deux séries par un test de blancheur de Box-Pierce.

Réponse. Les deux séries en question sont le processus ARCH(1) simulé et le bruit blanc à la base, colonnes 1 et 3 dans `archsim.1` simulé section 12.3.1 suivant le modèle 12.2. On en refait la simulation ici :

```
> require(fGarch)
> spec.1 <- garchSpec(model = list(mu = 5, omega = 0.1,
+                                  alpha = 0.9, beta = 0),
+                     rseed = 397)
> archsim.1 <- garchSim(extended = TRUE, spec.1, n = 300, n.start=10)
> head(archsim.1,2)
```

```
GMT
              garch     sigma        eps
2019-07-17 5.172777 0.3175684  0.5440636
2019-07-18 4.668813 0.3561837 -0.9298203
```

On a noté que l'hétéroscédasticité conditionnelle d'un bruit blanc x_t, se traduit par la non blancheur de son carré, un test de blancheur du carré de la série est donc un test d'homoscédasticité de la série. La blancheur de l'ARCH(1) simulé dans le livre et celle de la série z_t sous-jacente s'obtiennent par :

```
> moyenne <- mean(archsim.1[,1])
> ret <- c(6, 12, 18)
> a1 <- Box.test.2(archsim.1[,1], ret,
+                  type = "Box-Pierce", decim = 8)
> a2 <- Box.test.2((archsim.1[,1]-moyenne)^2, ret,
+                  type = "Box-Pierce", decim = 8)
> a3 <- Box.test.2(archsim.1[,3]^2, ret,
+                  type = "Box-Pierce", decim = 8)
> a123 <- cbind(a1, a2[,2], a3[,2])
> colnames(a123) <-
+    c("Retard", "p.val y1", "p.val y1 centre carre", "p.val z_t")
```

On centré la série simulée car elle n'est pas de moyenne nulle. On a testé que les coefficients d'autocorrélation jusqu'aux ordres 6, 12, 18 sont nuls, pour la série

	Retard	p.val y1	p.val y1 centre carre	p.val z_t
1	6.0000	0.0987	0.0000	0.5416
2	12.0000	0.0724	0.0000	0.8543
3	18.0000	0.0656	0.0000	0.8044

Tableau 1 – Test de blancheur, p-values

simulée `y1`, son carré centré, et pour la série du bruit blanc gaussien servant à engendrer l'ARCH. On voit (table 1) qu'on accepte l'hypothèse de blancheur pour la série simulée et pour le bruit blanc gaussien sous-jacent, mais qu'on la rejette évidemment pour le carré de la série centrée.

Exercice 12.1 (Tests sur `mod1`)
L'estimation de (12.5) a été obtenue dans `mod1` :

$$\begin{aligned} y_t &= 2+\sigma_t z_t \\ \sigma_t^2 &= 0.09+0.15\varepsilon_{t-1}^2+0.3\varepsilon_{t-2}^2+0.4\sigma_{t-1}^2. \end{aligned} \tag{12.5}$$

Faisons l'hypothèse que les estimateurs des paramètres sont approximativement normalement distribués.

- Extraire de la sortie de l'estimation, la matrice des variances-covariances estimées des paramètres.
- Tester l'hypothèse que $\alpha_2 = 0.3$.
- Tester l'hypothèse que $(\alpha_2, \beta_1) = (0.3, 0.4)$
- Conclure.

Réponse. D'abord nous simulons la même série que dans le chapitre :

```
> spec <- garchSpec(model = list(mu = 2, omega = 0.09,
+                                alpha = c(0.15, 0.3),
+                                beta = 0.4),
+                   rseed = 9647)
> var.margi = 0.09/(1 - 0.15 - 0.3 - 0.4)
> y = garchSim(spec, n = 420, extended = TRUE)
> y1 = y[21:420, 1]
```

La série à étudier est formée des 400 dernières observations contenues dans la première colonne de `y`. Avant d'ajuster un modèle à cette série nous avons examiné son chronogramme :

```
plot.ts(y1, xlab='temps')
```

Il y a plusieurs valeurs extrêmes que nous décidons de les remplacer par des valeurs plus raisonnable.

```
> m1 = mean(y1)
> (q5 = quantile(abs(y1-m1), probs = c(.975, .98, .985, .99, .995)))

   97.5%      98%    98.5%      99%    99.5%
2.965956 3.095066 3.696268 5.500659 6.443857
```

```
> extrem985 = which(abs(y1-m1) > q5[3])
> cat("nombre de points : ", length(extrem985), "\n")

nombre de points :  6

> y1b = y1
> y1b[extrem985] <- q5[3]*sign(y1[extrem985]-m1)
```

Nous travaillons sur la série corrigée `y1b`.

— Extraction de la matrice des variances-covariances estimées des paramètres. L'estimation est conduite par :

```
> mod1 <- garchFit(~garch(2,1), data = y1b, trace = FALSE,
+                  include.mean = TRUE)
```

On situe les statistiques qui nous intéressent par `str(mod1)`. On y récupère aussi les noms précis des paramètres. L'estimation des paramètres alpha2 et beta1 est extraite de `mod1` par :

```
> param = c("alpha2", "beta1")
> estim = as.matrix(coef(mod1)[param])
```

et celle de leur matrice des covariances estimée par :

```
> # matrice des covariances des estimateurs
> vcov.par = mod1@fit$cvar[param,param]
```

On peut noter également que `mod1@fit$se.coef` contient les écarts-types d'estimation, c'est-à-dire la racine carrée de la diagonale de la matrice des covariances.

— Tester l'hypothèse $H_0 : \alpha_2 = 0.3$.
Nous devons former la statistique : $T = \frac{\widehat{\alpha}_2 - 0.3}{s_{\widehat{\alpha}_2}}$ (cf. chap.3). Sous H_0, elle est approximativement normalement distribuée $\mathcal{N}(0,1)$.

```
> t.alpha = (coef(mod1)["alpha2"]-0.3)/mod1@fit$se.coef["alpha2"]
> p.val = 2* pnorm(-abs(t.alpha))
> cat("p-value : ", p.val, "\n")

p-value :  0.543093
```

La p-value est élevée, et donc l'estimation est cohérente avec le modèle simulé.

— Tester l'hypothèse que $(\alpha_2, \beta_1) = (0.3, 0.4)$.
Sous l'hypothèse nulle :

$$H_0 : \begin{bmatrix}\alpha_2 \\ \beta_1\end{bmatrix} = \begin{bmatrix}0.3 \\ 0.4\end{bmatrix}, \quad \begin{bmatrix}\widehat{\alpha}_2 \\ \widehat{\beta}_1\end{bmatrix} \sim \text{AN}(\begin{bmatrix}0.3 \\ 0.4\end{bmatrix}, \widehat{V}),$$

où $\widehat{V}$ est la matrice des covariances estimée pour ce couple d'estimateurs et

$$[\widehat{\alpha}_2 - 0.3, \widehat{\beta}_1 - 0.4]\widehat{V}^{-1}\begin{bmatrix}\widehat{\alpha}_2 - 0.3 \\ \widehat{\beta}_1 - 0.4\end{bmatrix} \sim \text{A}\chi^2_2$$

On rejette l'hypothèse nulle pour de grandes valeurs de cette statistique. Pour effectuer ce test, il nous faut : (1) récupérer les estimations de ces paramètres ainsi que la matrice des covariances estimée correspondante, ce que nous avons fait au début de l'exercice, (2) calculer la distance du χ^2 de la moyenne estimée à la moyenne théorique, et enfin (3) la p-value.

```
> # moyenne th\'eorique
> theo = as.matrix(c(0.3, 0.4))
> # difference
> differ = estim-theo
> # stat de test
> khi = t(differ)%*%solve(vcov.par)%*%differ;
> # statistique de test et p-value
> cat("stat de test et p-value : ",
+     c(khi, 1-pchisq(khi, df=2)), "\n")
```

```
stat de test et p-value :  0.9424129 0.6242487
```

item En résumé, la simplification d'estimation envisagée pour ces deux paramètres est cohérente avec le modèle simulé. Il faut cependant garder à l'esprit que ces estimations de covariances sont basées sur des approximations grossières et donc les conclusions des tests sont d'abord à comprendre comme des indications.

Exercice 12.4 (L'Oréal)
Dans le traitement de L'Oréal, on a enchaîné la modélisation ARMA de la série et la modélisation GARCH du résidu de cet ajustement.

- Effectuer une modélisation GARCH du rendement sans passer par l'étape de modélisation par un ARMA.
- Superposer sur un même graphique : la série à prédire et les intervalles de prévision (à 90%) par un ARMA et par un GARCH. Commenter.

Réponse.

- Effectuer une modélisation GARCH du rendement sans passer par l'étape de modélisation par un ARMA.
  Dans le chapitre, nous avons obtenu le modèle 12.13 en enchaînant une modélisation ARMA du rendement puis une modélisation GARCH du résidu du premier ajustement. Tout d'abord, nous réestimons le modèle ARMA identifié pour L'Oréal :

```
> # require(fBasics)
> data(csdl)
> aa = returns(csdl, percentage = TRUE)
> aab = aa[complete.cases(aa) == TRUE,]
> # ERRATUM: its package not maintained anymore
> # r.csdl = its(aab, as.POSIXct(row.names(aab)))
> # r.lor <- rangeIts(r.csdl[,"L_Oreal"], start = "2007-12-28")
> # its replaced by zoo package
> r.csdl = zoo(aab, as.POSIXct(row.names(aab)))
> r.lor <- window(r.csdl[,"L_Oreal"], start = "2007-12-28")
> r.lor.0 = r.lor[1:(length(r.lor)-51)]
```

```
> r.lor.1 = r.lor[(length(r.lor)-50):length(r.lor)]
> require("forecast")
> mod.r.lor = Arima(as.numeric(r.lor.0), order = c(0, 0, 4),
+                   include.mean = FALSE,
+                   fixed = c(NA, NA, 0, NA))
```

D'autre part, nous estimons sur le rendement, le modèle GARCH identifié sur les résidus de l'ajustement à un ARMA :

```
> var.marg.est <- function(mod)
+  {param.estim <- mod@fit$par
+   std.estim <- mod@fit$se.coef
+   k <- which(names(param.estim)=="omega")
+   value <- param.estim[k]/(1 -
+         sum(param.estim[(k + 1):length(param.estim)]))
+   cat("variance marginale : ", value, "\n")
+  }
> mod.garch.lor <- garchFit(~garch(1,1), data = as.numeric(r.lor.0),
+                           trace = FALSE, include.mean = TRUE,
+                           na.action = na.pass)
> summary(mod.garch.lor)

Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = as.numeric(r.lor.0),
    include.mean = TRUE, trace = FALSE, na.action = na.pass)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x5620ae959de0>
 [data = as.numeric(r.lor.0)]

Conditional Distribution:
 norm

Coefficient(s):
      mu     omega    alpha1     beta1
-0.11752   0.41166   0.16990   0.77714

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu      -0.11752     0.11406   -1.030  0.30285
omega    0.41166     0.21080    1.953  0.05084 .
alpha1   0.16990     0.06137    2.768  0.00564 **
```

```
beta1    0.77714     0.06733   11.542  < 2e-16 ***
---
Signif. codes:
0
```

```
> var.marg.est(mod.garch.lor)
```

```
variance marginale :  7.772938
```

Nous pouvons voir que le modèle capte bien l'hétéroscédasticité du rendement et que les paramètres sont significatifs. Mais le modèle fournit un résidu qui n'est pas blanc, au contraire de ce qu'avait fourni l'ajustement d'un ARMA/GARCH section 12.8.1.

— Superposer sur un même graphique : la série à prédire et les intervalles de prévision (à 90%) par un ARMA et par un GARCH. Commenter.
Il faut donc prédire le rendement de L'Oréal par les deux méthodes.

```
> npred = 50
> pred.arima.lor = predict(mod.r.lor,n.ahead=npred)
> pred.garch.lor = predict(mod.garch.lor,n.ahead=npred)
> str(pred.arima.lor)
```

```
List of 2
 $ pred: Time-Series [1:50] from 327 to 376: 0.32 0.779 0.62 -1.004
 $ se  : Time-Series [1:50] from 327 to 376: 2.36 2.41 2.44 2.44 2.5
```

```
> demi = qnorm(0.95)*pred.arima.lor$se
> binf.arima.lor =  pred.arima.lor$pred-demi
> bsup.arima.lor =  pred.arima.lor$pred+demi
> demi = qnorm(0.95)*pred.garch.lor$standardDeviation
> binf.garch.lor = pred.garch.lor$meanForecast-demi
> bsup.garch.lor = pred.garch.lor$meanForecast+demi
```

`mat.lor` rassemble les différentes séries dont on veut superposer les chronogrammes : l'estimation du rendement moyen, le rendement observé, les bornes inférieures et supérieures obtenues par la modélisation GARCH puis par la modélisation ARMA.

```
> par(oma=rep(0.5,4))
> mat.lor = cbind(binf.arima.lor,bsup.arima.lor,
+  binf.garch.lor, bsup.garch.lor, r.lor.1[1:npred])
> matplot(1:npred,mat.lor, type = "l", col = "black",
+         lty = c(1, 1, 2, 2, 3), lwd = 2,
+         xlab = "horizon 50", ylab = "rendement")
> leg.txt = c("GARCH", "AR", "realisation")
> legend(14, 3, leg.txt, lty = c(1, 2, 3))
```

On note sur le graphique superposant les intervalles de prévision par les deux méthodes (fig. 1) que la seule modélisation par un ARMA fournit des intervalles de prédiction trop amples.

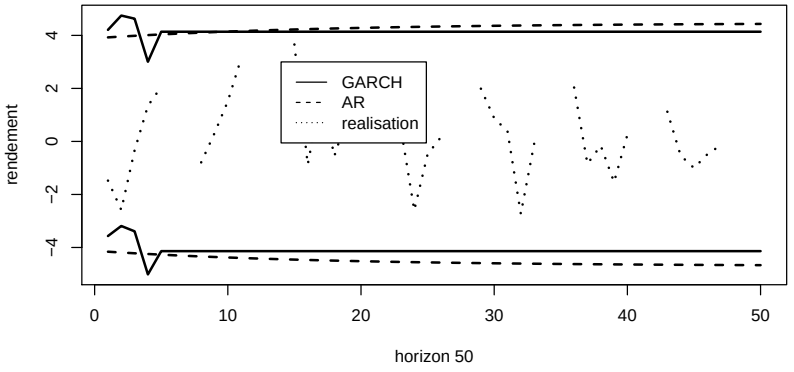

Fig. 1 – L'Oréal : intervalles de prévision (ARMA et GARCH) à 80% et réalisation du rendement.

Exercice 12.5 (Danone)
Dans le traitement de Danone, on a enchaîné la modélisation ARMA de la série et la modélisation GARCH du résidu de cet ajustement.

- Effectuer une modélisation GARCH du rendement sans passer par l'étape de modélisation par un ARMA.
- Superposer sur un même graphique : la série à prédire et les intervalles de prévision (à 90%) par un ARMA, un ARMA-GARCH et un GARCH. Commenter.

Réponse.

- Effectuer une modélisation GARCH du rendement sans passer par l'étape de modélisation par un ARMA. Dans le chapitre, nous avons obtenu le modèle 12.16 en enchaînant une modélisation ARMA du rendement puis une modélisation GARCH du résidu du premier ajustement.

```
> # ERRATUM
> # r.dan = rangeIts(r.csdl[,"Danone"], start= "2007-12-28")
> # replaced by :
> r.dan <- window(r.csdl[, "Danone"], start = "2007-12-28")
> r.dan.0 = r.dan[1:(length(r.dan)-50)]
> r.dan.1 = r.dan[(length(r.dan)-49):length(r.dan)]
```

- Superposer sur un même graphique : la série à prédire et les intervalles de prévision (à 90%) par un ARMA et par un GARCH. Commenter.
  Nous calculons les prédictions par les deux modèles.

```
> npred = 50
> pred.arima.dan = predict(mod.arma.dan, n.ahead = npred)
> pred.garch.dan = predict(mod.garch.dan, n.ahead = npred)
> str(pred.arima.dan)
```

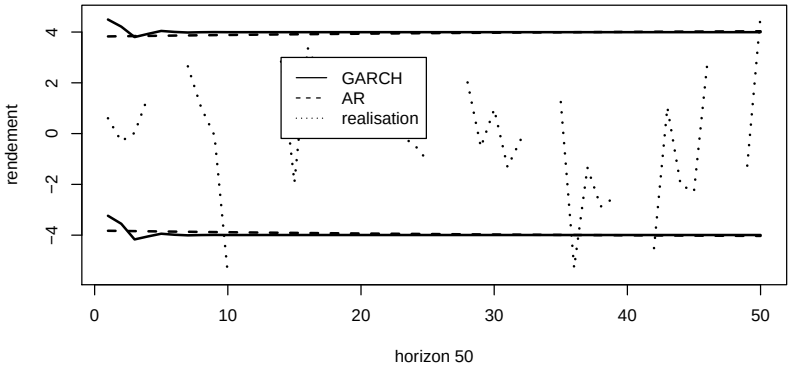

Fig. 2 – Danone : prévision ARMA et GARCH (90%) et réalisation.

```
List of 2
 $ pred: Time-Series [1:50] from 328 to 377: 0.6285 0.3304 -0.1832 -
 $ se  : Time-Series [1:50] from 328 to 377: 2.35 2.36 2.42 2.43 2.4
> demi = qnorm(0.95)*pred.arima.dan$se
> binf.arima.dan = pred.arima.dan$pred - demi
> bsup.arima.dan = pred.arima.dan$pred + demi
> #
> demi = qnorm(0.95)*pred.garch.dan$standardDeviation
> binf.garch.dan = pred.garch.dan$meanForecast - demi
> bsup.garch.dan = pred.garch.dan$meanForecast + demi
```

`mat.dan` rassemble les différentes séries dont on veut superposer les chronogrammes : le rendement observé et les bornes inférieures et supérieures obtenues par la modélisation GARCH puis par la modélisation ARMA.

```
> par(oma=rep(0.5,4))
> mat.dan = cbind(binf.arima.dan, bsup.arima.dan,
+  binf.garch.dan, bsup.garch.dan, r.dan.1[1:npred])
> matplot(1:npred, mat.dan, type = "l", col = "black",
+         lty = c(1, 1, 2, 2, 3), lwd = 2,
+         xlab = "horizon 50", ylab = "rendement")
> leg.txt = c("GARCH", "AR", "realisation")
> legend(14, 3, leg.txt, lty = c(1, 2, 3))
```

Nous observons que pour Danone il y a peu de différence dans les prévisions GARCH et ARMA (fig. 2).