Modélisation GARCH de la volatilité

Chapitre 20 : Étude de cas sur données réelles — R

Jaouad Madkour

Faculté d'Économie et de Gestion de Tanger · 27 août 2026

Plan du chapitre

Plan du chapitre

Avant de commencer

Ce que ce chapitre ajoute

Des données que personne n'a fabriquées

Les chapitres 18 et 19 travaillaient sur une série **simulée**, dont on connaissait la réponse. C'était voulu : on pouvait vérifier que la méthode retrouve ce qu'on y avait mis.

Ici, personne ne connaît la réponse. C'est le seul exercice qui ressemble à ce qu'on fait en salle de marché — et le seul qui puisse échouer.

La chaîne complète, en un script

Télécharger $\longrightarrow$ prix en rentabilités $\longrightarrow$ stationnarité $\longrightarrow$ ARMA $\longrightarrow$ test ARCH $\longrightarrow$ GARCH $\longrightarrow$ validation $\longrightarrow$ VaR $\longrightarrow$ Kupiec et Christoffersen.

Obtenir les données

Google Finance : ce qui existe et ce qui n'existe pas

Un point à clarifier d'emblée

Google Finance n'expose **aucune interface de programmation publique**. L'API historique a été fermée en 2012 et n'a jamais été remplacée. Aucun paquet R ne peut donc interroger Google Finance directement.

Ce qui existe, et qui fonctionne parfaitement : la fonction `GOOGLEFINANCE()` de **Google Sheets**.

La voie Google Finance, par le tableur

Dans une feuille Google Sheets

```
=GOOGLEFINANCE("INDEXSP:.INX"; "close"; DATE(2015;1;1); TODAY(); "DAILY")
```

Puis **Fichier** → **Télécharger** → **CSV**, et l'on obtient `donnees.csv`.

Quelques codes utiles

Actif	Code Google Finance
S&P 500	`INDEXSP:.INX`
CAC 40	`INDEXEURO:PX1`
Apple	`NASDAQ:AAPL`
Attijariwafa Bank	`CAS:ATW`

Avantage et limite

C'est la source la plus simple pour un étudiant, et elle couvre la Bourse de Casablanca. Mais elle exige une **manipulation manuelle** : le script n'est pas reproductible d'un seul appel.

Pour un script reproductible : quantmod

```
install.packages(c("quantmod", "rugarch", "forecast", "tseries", "FinTS"))
library(quantmod); library(rugarch)
library(forecast); library(tseries); library(FinTS)

getSymbols("^GSPC", src = "yahoo",
           from = "2015-01-01", to = "2025-01-01")
P <- Cl(GSPC)                       # colonne de clôture
```

Pourquoi Yahoo plutôt que Google

Ce n'est pas un choix de qualité : les deux fournissent les mêmes cotations de clôture. C'est un choix de **disponibilité** — Yahoo est interrogeable par programme, Google ne l'est pas.

Le reste du chapitre est rigoureusement identique quelle que soit la source : une fois le CSV chargé, plus rien ne dépend du fournisseur.

Charger un fichier téléchargé de Google Sheets

```r
don <- read.csv("donnees.csv", stringsAsFactors = FALSE)
names(don) <- c("date", "prix")
don$date <- as.Date(don$date, format = "%d/%m/%Y")
don <- don[!is.na(don$prix), ]
P <- don$prix
```

Trois pièges du fichier Google Sheets

Le **format de date** dépend de la langue de la feuille — vérifier avant de convertir.

La première ligne contient des **en-têtes en anglais** (`Date`, `Close`) qu'on renomme.

Les jours fériés produisent des **lignes vides** qu'il faut retirer, sans quoi les rentabilités seront fausses.

Étape 1 — Des prix aux rentabilités

```r
r <- 100 * diff(log(P))            # rentabilités logarithmiques, en %
r <- r[!is.na(r)]
T <- length(r)
```

Trois décisions, toutes justifiées par le cours

Logarithme : les rentabilités logarithmiques s'additionnent dans le temps (cours ARMA, chapitre 4).

Différence : le log-prix est $I(1)$, sa différence est $I(0)$ (cours ARMA, chapitre 9).

Facteur 100 : sans lui, ω serait de l'ordre de 10^{-6} et l'optimiseur du chapitre 14 convergerait mal.

Décrire

```
library(moments)
c(n = T, moyenne = mean(r), ecart_type = sd(r),
  annualise = sd(r) * sqrt(252),
  asymetrie = skewness(r), kurtosis = kurtosis(r))
jarque.bera.test(r)
```

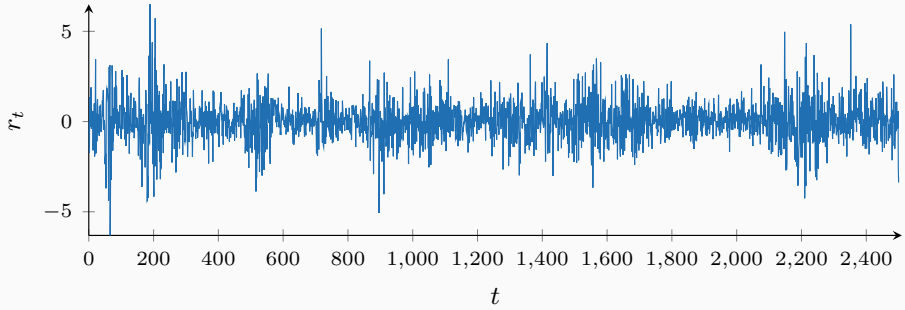
Ce qu'on attend de voir

Sur un indice large et dix ans de données : écart-type quotidien entre $0{,}8$ et $1{,}3\,\%$, soit 13 à 20 % annualisés ; asymétrie **négative** ; kurtosis entre **5 et 12**.

Un kurtosis proche de 3 doit alerter : la série a probablement été lissée, ou la fréquence n'est pas quotidienne.

Regarder avant de calculer

```
par(mfrow = c(2, 1), mar = c(3, 4, 2, 1))
plot(don$date[-1], r, type = "l", col = "steelblue",
     ylab = "rentabilité (%)", xlab = "")
plot(don$date, P, type = "l", col = "grey30", ylab = "prix", xlab = "")
```



Ce que l'œil doit chercher

Des **blocs** d'agitation. Sur une période couvrant 2020, ils sautent aux yeux : la crise sanitaire produit le

Étape 2 — Stationnarité

```r
adf.test(r)                          # H0 : racine unitaire
kpss.test(r, null = "Level")         # H0 : stationnarité
adf.test(log(P))                     # pour comparaison : le log-prix
```

Le résultat attendu, et il est très robuste

Série	ADF	KPSS	Conclusion
$\ln P_t$	ne rejette pas	rejette	$I(1)$
r_t	rejette nettement	ne rejette pas	$I(0)$

La première ligne de la table de décision

C'est le cas sans ambiguïté du chapitre 9 du cours ARMA. Et c'est ce qui justifie **après coup** d'avoir différencié : on ne l'a pas fait par habitude, on l'a fait parce que le test le demande.

Étape 3 — L’équation de moyenne

```r
par(mfrow = c(1, 2))
Acf(r,  lag.max = 20, main = "ACF")
Pacf(r, lag.max = 20, main = "PACF")

Box.test(r, lag = 20, type = "Ljung-Box")

m <- auto.arima(r, max.p = 3, max.q = 3, d = 0,
                seasonal = FALSE, ic = "bic", stepwise = FALSE)
m
```

Deux issues possibles, toutes deux normales

Le cas le plus fréquent : ARMA(0,0), une simple constante. L'efficience faible tient.

Le cas fréquent aussi : un AR(1) avec $\hat{\alpha}_1$ entre $-0{,}10$ et $-0{,}05$, significatif. C'est l'effet de **rebond à court terme** (*bid-ask bounce*), bien documenté sur données quotidiennes. Le coefficient est réel mais trop petit pour être exploité après coûts.

La règle à suivre

Le test ARCH sur les résidus

```r
e <- residuals(m)
Box.test(e,   lag = 20, type = "Ljung-Box")   # doit être non rejeté
Box.test(e^2, lag = 20, type = "Ljung-Box")   # sera rejeté massivement
ArchTest(e, lags = 12)
```

Ce qu'on obtient invariablement

Sur e_t : non rejeté, l'équation de moyenne suffit.

Sur e_t^2 : statistique de plusieurs centaines, p-valeur inférieure à 10^{-16}.

Le contraste entre ces deux lignes est tout le cours.

Étape 4 — Le modèle de variance

Trois modèles emboîtés

```r
ordre_moy <- arimaorder(m)[c(1, 3)]        # (p, q) de l'ARMA retenu

faire <- function(modele, loi) {
  ugarchfit(ugarchspec(
    variance.model = list(model = modele, garchOrder = c(1, 1)),
    mean.model     = list(armaOrder = ordre_moy, include.mean = TRUE),
    distribution.model = loi), data = r)
}

g1 <- faire("sGARCH",   "norm")
g2 <- faire("sGARCH",   "std")
g3 <- faire("gjrGARCH", "std")

sapply(list(g1, g2, g3), function(g) infocriteria(g)[2])   # BIC
```

Trois modèles emboîtés (suite)

Le classement, à peu près certain

$$\text{BIC}(\text{GJR-t}) < \text{BIC}(\text{GARCH-t}) < \text{BIC}(\text{GARCH-normal})$$

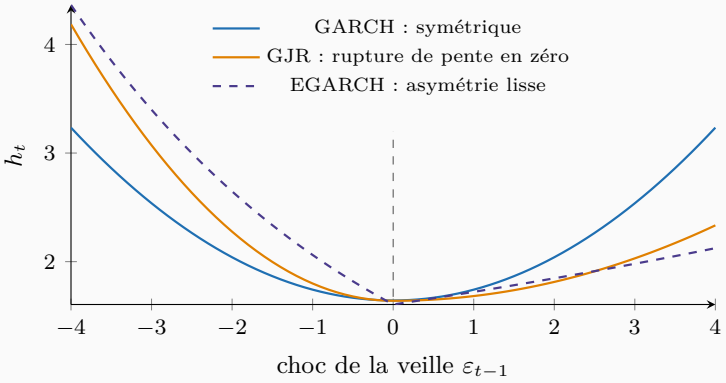
Les ordres de grandeur à attendre sur un indice

Paramètre	Valeur usuelle
$\hat{\alpha}_1$	$0{,}02$ à $0{,}08$ (GJR)
$\hat{\gamma}$	$0{,}10$ à $0{,}20$
$\hat{\beta}_1$	$0{,}88$ à $0{,}93$
persistance	$0{,}97$ à $0{,}995$
$\hat{\nu}$	5 à 10

Lire l'effet de levier

```
coef(g3)["gamma1"]
persistance <- sum(coef(g3)[c("alpha1", "beta1")]) + coef(g3)["gamma1"]/2
c(persistance = persistance,
  demi_vie    = log(0.5) / log(persistance))
```

Lire l'effet de levier (suite)



Lire l'effet de levier (suite 2)

Sur actions, γ est presque toujours significatif

Et souvent **plus grand que** α_1 : une baisse pèse deux à quatre fois une hausse de même ampleur.

Sur un taux de change, en revanche, $\hat{\gamma}$ est proche de zéro — ce qui se comprend, puisqu'une devise qui baisse est une autre qui monte.

Étape 5 — Valider

Lire l'effet de levier (suite 3)

```r
z <- residuals(g3, standardize = TRUE)
Box.test(z,   lag = 20, type = "Ljung-Box")
Box.test(z^2, lag = 20, type = "Ljung-Box")     # le test décisif
signbias(g3)
plot(g3, which = 9)                             # QQ-plot
```

Lire l'effet de levier (suite 4)

Le critère d'arrêt

$Q_{\mathrm{LB}}(20)$ sur z^2 **non rejeté**. Sur données réelles, on passe typiquement de plusieurs centaines à une valeur inférieure à 30.

Ce qui peut résister, et que faire

Si z^2 reste significatif : essayer un GARCH(2,1), ou un EGARCH.

Si `signbias` reste rejeté après le GJR : l'asymétrie est plus lisse que ne le permet l'indicatrice — passer à l'EGARCH du chapitre 11.

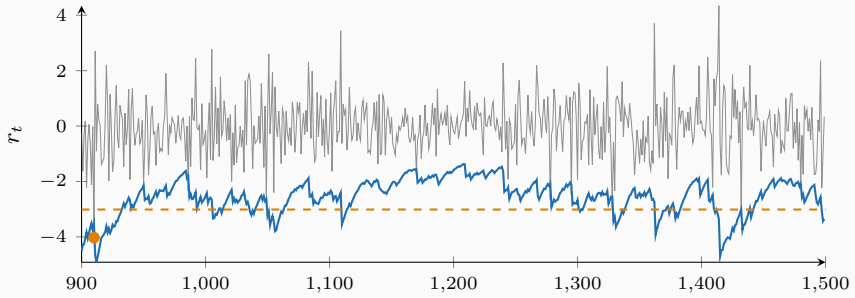
Si le QQ-plot décroche **seulement à gauche** : ce n'est pas h_t qu'il faut changer, c'est la loi. Passer à une Student asymétrique (`distribution.model = "sstd"`).

Étape 6 — VaR et backtesting

Calculer la VaR filtrée

```r
q1    <- quantile(z, 0.01)                    # simulation historique filtrée
var_c <- fitted(g3) + sigma(g3) * q1          # VaR conditionnelle
var_u <- rep(quantile(r, 0.01), T)            # seuil fixe, pour comparaison

dep_c <- as.integer(r < var_c)
dep_u <- as.integer(r < var_u)
c(conditionnelle = sum(dep_c), inconditionnelle = sum(dep_u), attendu = 0.01 * T)
```



Kupiec et Christoffersen

```
VaRTest(alpha = 0.01, actual = as.numeric(r),
        VaR = as.numeric(var_c), conf.level = 0.95)
VaRTest(alpha = 0.01, actual = as.numeric(r),
        VaR = as.numeric(var_u), conf.level = 0.95)
```

Kupiec et Christoffersen (suite)

Ce que renvoie VaRTest

Champ	Test
`uc.LRstat`, `uc.Decision`	Kupiec, couverture
`cc.LRstat`, `cc.Decision`	Christoffersen, couverture conditionnelle

Le résultat sur données réelles

La VaR **inconditionnelle** échoue souvent aux deux tests : elle groupe ses dépassements en 2020, et en compte trop sur l'ensemble.

La VaR **conditionnelle** passe le test d'indépendance dans la grande majorité des cas. Quand elle échoue, c'est presque toujours sur des échantillons contenant un krach isolé — ce qu'aucun modèle à paramètres constants ne prévoit.

Ce que les données réelles apprennent en plus

Quatre différences avec la série simulée

Les ruptures de régime. Mars 2020 déplace le niveau de volatilité pour des mois. Le modèle l'interprète comme une persistance très élevée — c'est le piège du chapitre 13.

La persistance touche la borne. $\widehat{\pi}$ dépasse souvent $0{,}99$: on ne peut pas rejeter l'IGARCH.

Les paramètres bougent selon la fenêtre. Estimer sur 2015–2019 puis sur 2015–2024 donne des résultats sensiblement différents. C'est le **risque de modèle**, et c'est le chapitre 6 du cours de Value at Risk.

La validation n'est jamais parfaite. Un ou deux retards sortent des bandes. Il faut savoir s'arrêter : un modèle utile n'est pas un modèle sans reproche.

Le script complet

`etude-de-cas.R`, livré avec le cours, enchaîne les six étapes en une soixantaine de lignes. Il suffit d'y changer le symbole et les dates.

Synthèse

Ce qu'il faut retenir

L'essentiel du chapitre 20

- Google Finance n'a **pas d'API** : passer par `GOOGLEFINANCE()` dans Sheets, ou par **`quantmod`** sur Yahoo.
- $r_t = 100\,\Delta \ln P_t$: logarithme, différence, facteur 100 — trois décisions justifiées par le cours.
- ADF rejette sur r_t, KPSS ne rejette pas : série $I(0)$, la différenciation est validée.
- Équation de moyenne d'abord : ARMA(0,0) ou AR(1) faible ; test ARCH sur ses résidus.
- Trois modèles emboîtés, classés par le BIC : GJR-t l'emporte presque toujours sur actions.
- La VaR filtrée passe Christoffersen là où le seuil fixe échoue.

Vers le chapitre 21

Le même travail, sur la même série, en Python — et l'occasion de vérifier que deux chaînes logicielles indépendantes concluent la même chose.