

Master 2 Mention Ingénierie Mathématique
Parcours ISDS (Ingénierie Statistique et Data Science)
Parcours IFMA (Ingénierie Financière et Modèles Aléatoires)

Projet : Analyse des séries chronologiques et modèles ARIMA,
ARCH et GARCH



Par : Adnane EL KASMI | Sébastien IBRAHIM
Hmida SEBAA | Albert VARDANYAN

Encadrant principal : Jean-Patrick Baudry

Numéro de groupe : 9

Date : Janvier 2022

Confidentialité : Non Oui (Durée : 1 an 2 ans)

Sorbonne Université – M2 ISDS – 4 Pl. Jussieu, 75005 Paris

Remerciements

Nous tenons tout d'abord à remercier notre professeur Jean-Patrick Baudry pour le temps qu'il nous a consacré pour répondre à nos diverses questions ainsi que pour l'aide qu'il nous a apporté lorsque nous avons rencontré des problèmes dans ce projet.

Nous remercions aussi le corps enseignant de Sorbonne Université, puisque la plupart des notions abordées dans leur cours ont été mises en pratique dans ce projet. En effet, sans un enseignement de qualité de leur part, la réalisation de ce projet n'aurait pas été possible.

Pour finir, nous remercions l'Université de Sorbonne et ISUP car elle nous a permis d'avoir accès à des documents de recherches disponibles à la bibliothèque universitaire et ainsi, a pu faciliter au maximum nos recherches.

Table des matières

Table des matières	5
Introduction	7
1 Analyse de données	9
1.1 Définition : Série temporelle (ou série chronologique)	9
1.2 Definition des données	12
1.3 Charger les bibliothèques et définir les paramètres globaux	12
1.4 Lecture des données	12
1.5 Vue d'ensemble des données	12
1.6 Nettoyage des données	13
1.7 Distributions univariées	14
2 Analyse des séries chronologiques	15
2.1 Créer et tracer des séries temporelles - High	16
2.2 Stationnarité	18
2.3 Décomposition des séries temporelles	18
2.4 Différencier une série temporelle	19
3 ARIMA : Modèle de prédiction des séries temporelles	23
3.1 Le modèle ARMA	23
3.1.1 Le processus AR	23
3.1.2 Le processus MA	23
3.1.3 Le processus ARMA	23
3.2 Le modèle ARIMA	24
3.3 Sélection d'un modèle ARIMA candidat	24
3.4 Fitting d'un modèle ARIMA	27
3.5 Validation du modèle	32
3.6 Choix d'un modèle parmi plusieurs	33
3.7 Prévision à l'aide d'un modèle ARIMA	34
3.8 Tester la fonction arima_modeling sur les autres entreprises	51
4 Modèles ARCH et GARCH	65
4.1 Définitions : ARCH et GARCH	65
4.2 Principe du modèle ARCH	65
4.3 Principe des modèles GARCH	66
4.4 Application	66
4.4.1 Le package quantmod	66
4.4.2 Lecture des informations	66
4.4.3 Visualisation	67

4.4.4 Modélisations	70
4.4.5 GARCH Univarié	70
4.4.6 Variance conditionnelle estimée	72
4.4.7 Prévisions sur le passé	73
4.4.8 Les rendements	74
4.4.9 Modélisation du rendement	75
4.4.10 Résultats	75
4.4.11 Variance conditionnelle	76
4.4.12 Prévision sur le futur	76
Conclusion	79
Bibliographie	81
Annexe : Le test de Durbin-Watson	83
.1 Autocorrelation des résidus	83
.2 Intuition du test de Durbin-Watson	83
.3 Exemple	84
.3.1 Bruits blancs	84
.3.2 Des résidus autocorrelés et non bruits blancs	84
Annexe : Méthodologie de Box-Jenkins	85
Annexe : Algorithme Hyndman-Khandakar	87

Introduction

Les données boursières (actions, obligations ...) peuvent être intéressantes à analyser et, comme incitation supplémentaire, des modèles prédictifs solides peuvent avoir des retombées financières importantes.

La quantité de données financières sur le Web est apparemment infinie. Un ensemble de données volumineux et bien structuré sur un large éventail d'entreprises peut être difficile à trouver. Nous avons fourni ici un ensemble de données avec les cours des actions historiques (12 dernières années) pour 29 des 30 sociétés DJIA.

Ce projet fournit un guide étape par étape pour l'ajustement d'un modèle ARIMA sur les données de stock, à l'aide de packages dans R.

Les données sont présentées dans quelques formats pour répondre aux besoins ou aux limitations de calcul de différents individus.

Cet ensemble de données se prête à des visualisations très intéressantes. On peut examiner des choses simples comme l'évolution des prix au fil du temps, représenter graphiquement et comparer plusieurs actions à la fois, ou générer et représenter graphiquement de nouvelles mesures à partir des données fournies.

À partir de ces données, des statistiques boursières informatives telles que la volatilité et les moyennes mobiles peuvent être facilement calculées.

La question à un million de dollars est : pouvez-vous développer un modèle qui peut battre le marché et nous permettre de faire des transactions statistiquement informées ?

Chapitre 1

Analyse de données

1.1 Définition : Série temporelle (ou série chronologique)

Une série temporelle, ou série chronologique, est une suite de valeurs numériques représentant l'évolution d'une quantité spécifique au cours du temps.

De telles suites de variables aléatoires peuvent être exprimées mathématiquement afin d'en analyser le comportement, généralement pour comprendre son évolution passée et pour en prévoir le comportement futur. Une telle transposition mathématique utilise le plus souvent des concepts de probabilités et de statistique.

Les séries temporelles sont considérées à tort comme étant une branche exclusive de l'économétrie. Cette dernière est une discipline qui est relativement jeune alors que les séries temporelles ont été utilisées bien avant, par exemple en astronomie (1906) et en météorologie (1968).

L'objet des séries temporelles est l'étude des variables au cours du temps. Même s'ils n'ont pas été à l'origine de cette discipline, ce sont les économètres qui ont assuré les grandes avancées qu'a connues cette discipline (beaucoup de « Prix Nobel » d'économie sont des économètres).

Parmi ses principaux objectifs figurent la détermination de tendances au sein de ces séries ainsi que la stabilité des valeurs (et de leur variation) au cours du temps.

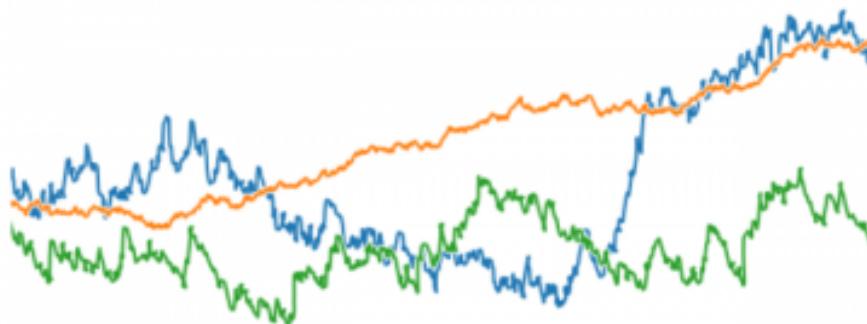


FIGURE 1.1 : Exemple de série temporelles.

C'est de la déception des prévisions issues des modèles structurels d'inspiration keynésienne qu'est née la théorie des séries temporelles telle qu'on la connaît aujourd'hui. Et sur ce point, c'est la publication de l'ouvrage de Box et Jenkins en 1970 qui a été décisive. En effet, dans l'ouvrage les deux auteurs développent le très populaire modèle ARMA (Auto Regressive Moving Average). Par exemple,

pour prévoir le PIB français en 2020, il ne s'agit plus d'utiliser un modèle structurel qui explique le PIB (par l'intermédiaire de la consommation, de l'investissement, des dépenses publiques, du solde commercial, etc.) puis de projeter les tendances passées, mais, avec le modèle ARMA, de prévoir le PIB de 2020 en exploitant les propriétés statistiques du PIB (moyenne, variance etc.). Ce modèle utilise souvent des valeurs retardées du PIB (d'où le terme Auto Regressive) et de chocs aléatoires qui sont en général de moyenne nulle, de variance constante et non autocorrélés (bruit blanc) ; quand la variable qui représente ces chocs est retardée, on parle de moyenne mobile.

Le modèle ARMA est un cas particulier d'un modèle beaucoup plus général nommé ARIMA (en) où le I désigne Integrated (« Intégrée » en français). En effet, le modèle ARMA ne permet de traiter que les séries dites stationnaires (des moments du premier ordre qui sont invariants au cours du temps). Les modèles ARIMA permettent de traiter les séries non stationnaires après avoir déterminé le niveau d'intégration (le nombre de fois qu'il faut différencier la série avant de la rendre stationnaire).

Bien que possédant d'excellentes qualités prévisionnelles, le modèle ARIMA ou ARMA souffre d'une lacune majeure : **il est incapable de traiter simultanément plus d'une variable (série)**. Par exemple, si les modèles structurels sont capables de répondre à une question telle que : « quel est l'effet de la hausse des taux d'intérêt sur le PIB ? », un modèle ARIMA est incapable d'y répondre. Pour contourner ce problème, il faut pouvoir généraliser le modèle ARIMA dans le cas à plusieurs variables. C'est ce qu'a fait en partie Christopher Sims en proposant en 1980 le modèle Vector Auto Regressive (VAR) qui permet de traiter concomitamment plusieurs variables. Mais, contrairement au modèle structurel à plusieurs variables, dans les modèles VAR, toutes les variables sont endogènes.

Ces modèles (ARIMA et VAR) ne permettent de traiter que des phénomènes qui sont linéaires ou approximativement (par exemple le PIB) mais ne permettent pas de « capturer » les propriétés des phénomènes qui sont non linéaires (les variables financières par exemple, inflation, cours d'action etc.). Pour prendre en compte à la fois la non-linéarité et la forte variabilité de ces variables, l'économètre américain Robert F. Engle a le premier développé le modèle dit ARCH (Auto Regressive Conditional Heteroscedasticity) en 1982.

Décomposition d'une série temporelle

Une série temporelle est décomposée selon trois éléments :

- Une tendance (T_t)
- Une saisonnalité (S_t)
- Un résidu ou erreur (ϵ_t)

Mathématiquement on peut donc traduire une série temporelle par :

$$X_t = T_t + S_t + \epsilon_t$$

Avec T la tendance, S la saisonnalité, ϵ le résidu et t l'indexation temporelle.

La tendance

La tendance correspond à un comportement croissant ou décroissant d'une série au cours du temps. Elle reflète souvent un phénomène de croissance ou décroissance sur le long terme.

La tendance d'une série temporelle peut prendre différentes formes :

- Linéaire $T_t = \alpha + \beta t$
- Quadratique $T_t = \alpha + \beta t + \gamma t^2$
- Exponentielle $T_t = \alpha + \beta e^t$

La saisonnalité

La saisonnalité reflète la présence d'un phénomène périodique qui se répète au long de la série temporelle.

Ainsi, si la composante saisonnière se répète selon période k . On aura $S_t + k = S_t$.

De nombreuses données présentent des saisonnalités, en particulier les données météorologiques (évolution de la température au cours du temps).

Certaines séries temporelles présentent à la fois une tendance et une saisonnalité, c'est le cas du trafic aérien mondial. On observe à la fois une croissance du trafic aérien tout en remarquant une différence forte entre le trafic hivernal et estival.

Le résidu

Le résidu du modèle correspond à la partie de la série temporelle que la décomposition ne permet pas d'expliquer. On ne peut pas décomposer intégralement une série temporelle uniquement selon une tendance et une saisonnalité.

Idéalement le résidu du modèle est stationnaire, c'est-à-dire que le processus restant n'évolue pas avec le temps (moyenne et variance constante). Si le résidu de notre série temporelle n'est pas stationnaire cela signifie que certaines composantes temporelles ne sont pas expliquées dans le modèle.

Une fois la tendance et la saisonnalité de la série temporelle expliquées, on peut donc chercher à expliquer le résidu de la décomposition avec des processus d'auto-régression ou de moyennes mobiles qui ont donné naissance au fameux modèle ARMA.

Le graal de la modélisation des séries temporelles est d'obtenir un résidu de type bruit blanc, c'est-à-dire un résidu qui ne contient plus aucune information temporelle. En pratique c'est donc un signal stationnaire aléatoire et décorrélé.

En plus de la saisonnalité, on définit quelquefois un cycle qui peut être considéré comme une saisonnalité de période plus longue, on peut alors définir plusieurs cycles différents.

La décomposition $X_t = T_t + S_t + \epsilon_t$ est dite additive, on peut aussi modéliser une série temporelle selon une décomposition multiplicative $X_t = T_t(1 + S_t)(1 + \epsilon_t)$

1.2 Definition des données

On a inclus des fichiers contenant 13 ans de données de stock (actions) dans le fichier TS_DATA.csv. Se sont des actions d'entreprises comme : IBM, Google,

Tous les fichiers ont les colonnes suivantes :

Date - au format : aaaa-mm-jj.

Open - prix de l'action à l'ouverture du marché (il s'agit de données NYSE donc toutes en USD).

High - Prix le plus élevé atteint dans la journée.

Low - Prix le plus bas atteint dans la journée.

Close - prix de l'action à la fermeture du marché.

Volume - Nombre d'actions échangées.

Name - le nom du télécopieur de l'action.

1.3 Charger les bibliothèques et définir les paramètres globaux

D'abord nous avons commencé par l'installation les packages de R :

```
packages = c("ggplot2", "dplyr", "tidyverse", "data.table", 'corrplot', 'gridExtra',
'forecast', 'tseries', 'TSA', 'tibble', 'TTR', 'xts', 'dygraphs', 'assertthat')

my.install <- function(pkg, ...){
  if (!(pkg %in% installed.packages()[,1])) {
    install.packages(pkg)
  }
  return (library(pkg, ...))
}

purrr::walk(packages, my.install, character.only = TRUE, warn.conflicts = FALSE)
```

1.4 Lecture des données

Pour lire le fichier csv qui représente les données à étudier, on utiliser la commande :

```
s_data <- read.csv(file = "H:/Desktop/TS_DATA.csv")
```

1.5 Vue d'ensemble des données

Nous utiliserons les fonctions « summary » et « str » de R pour examiner les données.

```
summary(s_data)
```

On obtient la sortie suivante :

```

#      Date          Open          High          Low
# Min. :2006-01-03  Min. : 0.00  Min. : 0.00  Min. : 0.00
# 1st Qu.:2008-12-31 1st Qu.: 33.93  1st Qu.: 34.28  1st Qu.: 33.59
# Median :2011-12-30 Median : 60.01  Median : 60.62  Median : 59.47
# Mean   :2012-01-01 Mean  : 85.60  Mean  : 86.38  Mean  : 84.82
# 3rd Qu.:2015-01-02 3rd Qu.: 93.99  3rd Qu.: 94.74  3rd Qu.: 93.24
# Max.  :2017-12-29 Max.  :1204.88  Max.  :1213.41  Max.  :1191.15

#      Close         Volume        Name
# Min. : 6.66    Min. :       0 Length:93612
# 1st Qu.: 33.96  1st Qu.: 5040180 Class :character
# Median : 60.05  Median : 9701142 Mode  :character
# Mean   : 85.64  Mean  : 20156670
# 3rd Qu.: 94.01  3rd Qu.: 20752222
# Max.  :1195.83  Max.  :843264044

str(s_data)

# 'data.frame': 93612 obs. of 7 variables:
# $ Date  : Date, format: "2006-01-03" "2006-01-04" ...
# $ Open   : num  77.8 79.5 78.4 78.6 78.5 ...
# $ High   : num  79.3 79.5 78.7 78.9 79.8 ...
# $ Low    : num  77.2 78.2 77.6 77.6 78.5 ...
# $ Close  : num  79.1 78.7 78 78.6 79 ...
# $ Volume: int  3117200 2558000 2529500 2479500 1845600 1919900 1911900 ...
# $ Name   : chr  "MMM" "MMM" "MMM" "MMM" ...

```

1.6 Nettoyage des données

Les données ont des valeurs manquantes, que nous remplacerons par zéro. De plus, la fonction « Date » est répertoriée comme facteur, nous la convertirons en structure « Date ».

```
s_data[is.na(s_data)] <- 0
s_data$Date <- as.Date(s_data$Date, format = "%Y-%m-%d")
summary(s_data)
```

On obtient la sortie suivante :

```

#      Date          Open          High          Low
# Min. :2006-01-03  Min. : 0.00  Min. : 0.00  Min. : 0.00
# 1st Qu.:2008-12-31 1st Qu.: 33.93  1st Qu.: 34.28  1st Qu.: 33.59
# Median :2011-12-30 Median : 60.01  Median : 60.62  Median : 59.47
# Mean   :2012-01-01 Mean  : 85.60  Mean  : 86.38  Mean  : 84.82
# 3rd Qu.:2015-01-02 3rd Qu.: 93.99  3rd Qu.: 94.74  3rd Qu.: 93.24
# Max.  :2017-12-29 Max.  :1204.88  Max.  :1213.41  Max.  :1191.15

#      Close         Volume        Name
# Min. : 6.66    Min. :       0 Length:93612
# 1st Qu.: 33.96  1st Qu.: 5040180 Class :character
# Median : 60.05  Median : 9701142 Mode  :character
# Mean   : 85.64  Mean  : 20156670
# 3rd Qu.: 94.01  3rd Qu.: 20752222
# Max.  :1195.83  Max.  :843264044

```

```
str(s_data)

# 'data.frame': 93612 obs. of 7 variables:
# $ Date   : Date, format: "2006-01-03" "2006-01-04" "2006-01-05" "2006-01-06" ...
# $ Open    : num  77.8 79.5 78.4 78.6 78.5 ...
# $ High    : num  79.3 79.5 78.7 78.9 79.8 ...
# $ Low     : num  77.2 78.2 77.6 77.6 78.5 ...
# $ Close   : num  79.1 78.7 78.78.6 79 ...
# $ Volume : int  3117200 2558000 2529500 2479500 1845600 1919900 1911900 ...
# $ Name    : chr  "MMM" "MMM" "MMM" "MMM" ...
```

1.7 Distributions univariées

Regardons quelques distributions univariées :

```
options(repr.plot.width=12, repr.plot.height=12)

p1 = ggplot(s_data, aes(Open)) + geom_histogram(bins = 50, aes(y = ..density..),
col = "red", fill = "red", alpha = 0.3) + geom_density()
p2 = ggplot(s_data, aes(High)) + geom_histogram(bins = 50, aes(y = ..density..),
col = "red", fill = "red", alpha = 0.3) + geom_density()
p3 = ggplot(s_data, aes(Low)) + geom_histogram(bins = 50, aes(y = ..density..),
col = "red", fill = "red", alpha = 0.3) + geom_density()
p4 = ggplot(s_data, aes(Close)) + geom_histogram(bins = 50, aes(y = ..density..),
col = "red", fill = "red", alpha = 0.3) + geom_density()

grid.arrange(p1,p2,p3,p4, nrow=2,ncol=2)
```

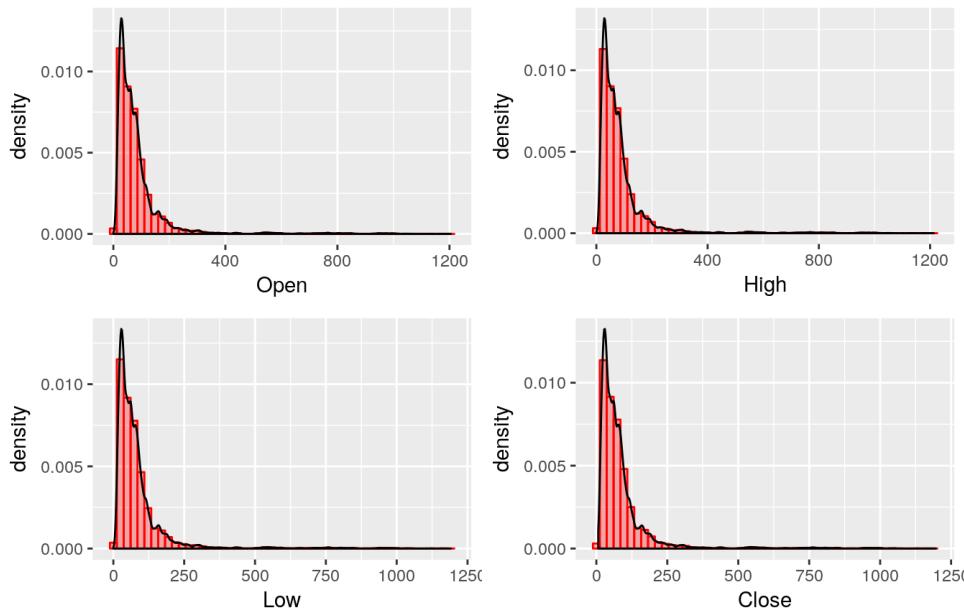


FIGURE 1.2 : Distributions univariées de **Open**, **High**, **Low** et **Close**.

On constate que la distribution des **Open** est proche de la distribution des **High**, et la distribution des **Low** est proche de la distribution des **close**. La plupart des prix sont compris entre **0** et **350** dollars pour **Open** et **High**, et entre **0** et **300** pour **Low** et **Close**.

Chapitre 2

Analyse des séries chronologiques

Avant de commencer par l'analyse des séries chronologiques, passons brièvement en revue la théorie.

Qu'est-ce que le modèle AutoRegressif ou AR ?

Les modèles autorégressifs (AR) sont des modèles où la valeur de la variable dans une période est liée aux valeurs de la période précédente. AR(p) est un modèle autorégressif avec p retard (lags).

Qu'est-ce que la moyenne mobile ou le modèle MA ?

Le modèle de moyenne mobile (MA) tient compte de la possibilité d'une relation entre une variable et le résidu de la période précédente. MA(q) est un modèle de moyenne mobile avec q décalages.

Qu'est-ce que le modèle ARMA ?

Le modèle de moyenne mobile autorégressive combine à la fois p termes autorégressifs et q termes de moyenne mobile, également appelés ARMA(p,q).

Regardons maintenant quelques actions individuelles et séries chronologiques individuelles (Open, Close, High, Low, Volume).

```
sample_ticker <- as.character(sample(tmp>Name, sample_num))
sample_ticker <- c(sample_ticker, 'GOOGL')
candidate_ticker <- unique(sample_ticker)
candidate_ticker <- c("IBM", "BA", "AAPL", "GS", "GOOGL")
candidate_num <- length(candidate_ticker)
stock_list <- vector(mode="list", length=candidate_num)
names(stock_list) <- candidate_ticker
i = 1
for (ticker in candidate_ticker){
  stock_list[[i]] <- filter(s_data, Name == ticker)
  i <- i+1
}
str(stock_list)
```

```
#List of 5
# $ IBM : 'data.frame': 3020 obs. of 7 variables:
#   ..$ Date : Date[1:3020], format: "2006-01-03" "2006-01-04" "2006-01-05" ...
#   ..$ Open : num [1:3020] 82.5 82.2 81.4 84 84.1 ...
#   ..$ High : num [1:3020] 82.5 82.5 82.9 85 84.2 ...
#   ..$ Low : num [1:3020] 80.8 81.3 81 83.4 83.4 ...
#   ..$ Close : num [1:3020] 82.1 82 82.5 85 83.7 ...
#   ..$ Volume: int [1:3020] 11715200 9840600 7213500 8197400 6858200 5701000 ...
#   ..$ Name : chr [1:3020] "IBM" "IBM" "IBM" "IBM" ...
# $ BA : 'data.frame': 3020 obs. of 7 variables:
#   ..$ Date : Date[1:3020], format: "2006-01-03" "2006-01-04" "2006-01-05" ...
#   ..$ Open : num [1:3020] 70.4 70.1 70.5 70.3 69.3 ...
#   ..$ High : num [1:3020] 70.6 71.3 70.5 70.5 69.4 ...
#   ..$ Low : num [1:3020] 69.3 69.9 69.6 69 68.7 ...
#   ..$ Close : num [1:3020] 70.4 71.2 70.3 69.3 68.8 ...
#   ..$ Volume: int [1:3020] 4943100 3165000 4598300 4820200 4567700 3869000 ...
#   ..$ Name : chr [1:3020] "BA" "BA" "BA" "BA" ...
# $ AAPL : 'data.frame': 3019 obs. of 7 variables:
#   ..$ Date : Date[1:3019], format: "2006-01-03" "2006-01-04" "2006-01-05" ...
#   ..$ Open : num [1:3019] 10.3 10.7 10.7 10.8 11 ...
#   ..$ High : num [1:3019] 10.7 10.8 10.7 11 11 ...
#   ..$ Low : num [1:3019] 10.3 10.6 10.5 10.6 10.8 ...
#   ..$ Close : num [1:3019] 10.7 10.7 10.6 10.9 10.9 ...
#   ..$ Volume: int [1:3019] 201853036 155225609 112396081 176139334 168861224 ...
#   ..$ Name : chr [1:3019] "AAPL" "AAPL" "AAPL" "AAPL" ...
# $ GS : 'data.frame': 3020 obs. of 7 variables:
#   ..$ Date : Date[1:3020], format: "2006-01-03" "2006-01-04" "2006-01-05" ...
#   ..$ Open : num [1:3020] 127 127 126 127 128 ...
#   ..$ High : num [1:3020] 129 129 127 129 131 ...
#   ..$ Low : num [1:3020] 124 126 126 127 128 ...
#   ..$ Close : num [1:3020] 129 127 127 129 130 ...
#   ..$ Volume: int [1:3020] 6188700 4862000 3717600 4319600 4723500 5539800 ...
#   ..$ Name : chr [1:3020] "GS" "GS" "GS" "GS" ...
# $ GOOGL: 'data.frame': 3019 obs. of 7 variables:
#   ..$ Date : Date[1:3019], format: "2006-01-03" "2006-01-04" "2006-01-05" ...
#   ..$ Open : num [1:3019] 211 222 223 229 233 ...
#   ..$ High : num [1:3019] 218 225 226 235 237 ...
#   ..$ Low : num [1:3019] 209 220 221 227 231 ...
#   ..$ Close : num [1:3019] 218 223 226 233 234 ...
#   ..$ Volume: int [1:3019] 13137450 15292353 10815661 17759521 12795837 ...
#   ..$ Name : chr [1:3019] "GOOGL" "GOOGL" "GOOGL" "GOOGL" ...
```

2.1 Créeer et tracer des séries temporelles - High

Il y a 5 séries temporelles dans les données fournies - (High, Low, Open, Close, Volume). Nous examinerons d'abord les valeurs High.

```

xts_list <- vector(mode="list", length=candidate_num)
ts_list <- vector(mode="list", length=candidate_num)

names(xts_list) = candidate_ticker
names(ts_list) = candidate_ticker

for (ticker in candidate_ticker){
  stock = stock_list[[ticker]]
  xts = xts(stock$Close, order.by=stock>Date)
  attr(xts, 'frequency') <- length(xts)/12
  ts = as.ts(xts, start = c(2006))
  xts_list[[ticker]] <- xts
  ts_list[[ticker]] <- ts
}
xts_table= do.call(cbind, xts_list)
dygraph(xts_table, xlab = "Time", ylab = "High value",
main = "S ries temporelles") %>%
  dyRangeSelector()

```

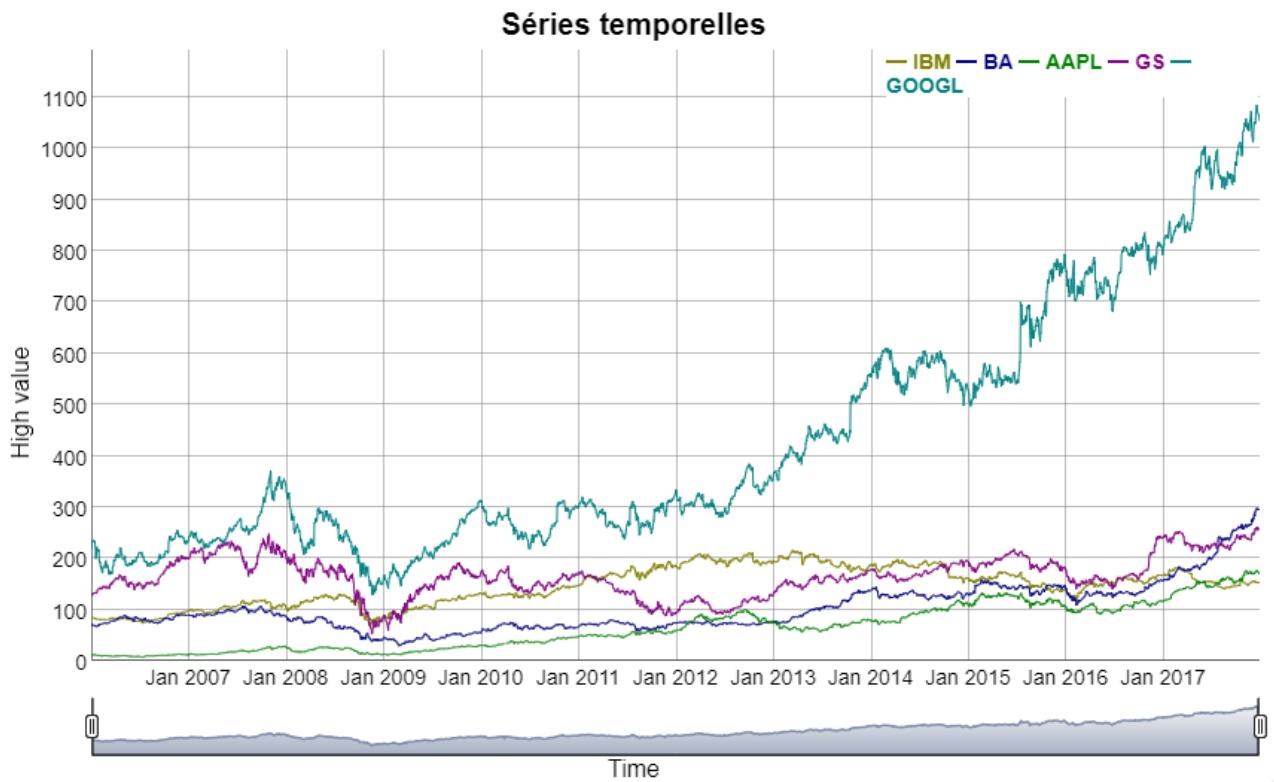


FIGURE 2.1 : Graphe des série temporelles.

Ensuite, nous allons d'abord démontrer le processus de modélisation des séries chronologiques sur « GOOGL ».

2.2 Stationnarité

Qu'est-ce qu'une série temporelle stationnaire ?

Un processus stationnaire a une moyenne et une variance qui ne changent pas dans le temps et le processus n'a pas de tendance.

La série chronologique ci-dessus ne semble pas stationnaire.

Pour confirmer cela, nous utiliserons le «Dickey-Fuller test» pour déterminer la stationnarité.

Le test de Dickey-Fuller ou test de racine unitaire de Dickey-Fuller est un test statistique qui vise à savoir si une série temporelle est stationnaire c'est-à-dire si ses propriétés statistiques varient ou pas dans le temps et si leur valeur est bien finie.

Test de Dickey-Fuller pour la variable :

```
xts = xts_list[['GOOGL']]
ts = ts_list[['GOOGL']]
adf.test(xts, alternative = "stationary", k = 0)
```

Le résultat du Dickey-Fuller Test nous montre que :

```
Augmented Dickey-Fuller Test

data: xts
Dickey-Fuller = -1.3188, Lag order = 0, p-value = 0.8667
alternative hypothesis: stationary
```

Interprétation du test :

H_0 : La série temporelle comporte une racine unitaire (non stationnaire).

H_1 : La série temporelle ne comporte pas une racine unitaire (stationnaire).

Etant donné que la p-value calculée est supérieure au niveau de signification $\alpha = 5\%$, on doit accepter l'hypothèse nulle H_0 .

Le risque de rejeter l'hypothèse nulle H_0 alors qu'elle est vraie est inférieur à 5%.

2.3 Décomposition des séries temporelles

La décomposition d'une série chronologique consiste à séparer la série chronologique en composantes tendancielles et irrégulières.

Nous testons à la fois le modèle additif et le modèle multiplicatif.

```
tscomponents_add <- decompose(ts, type = "additive")
tscomponents_mul <- decompose(ts, type = "multiplicative")

plot(tscoponents_add, col = "red")
plot(tscoponents_mul, col = "blue")
```

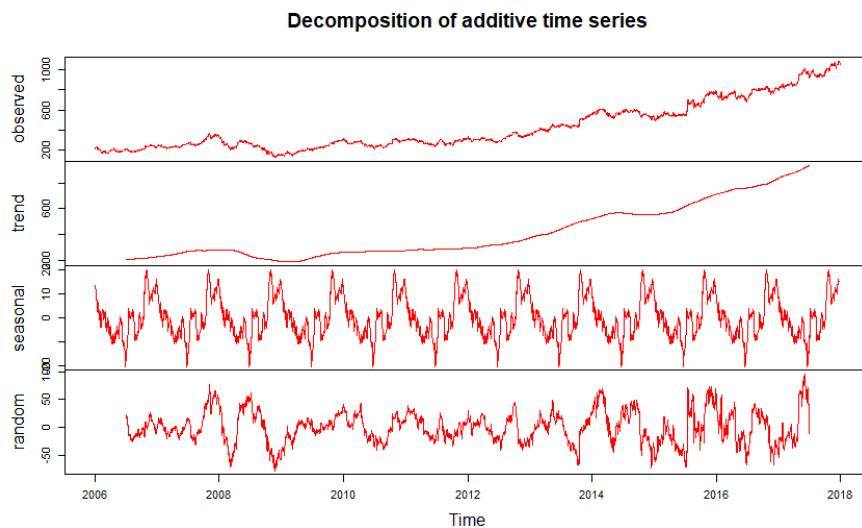


FIGURE 2.2 : La décomposition additive de la série chronologique.

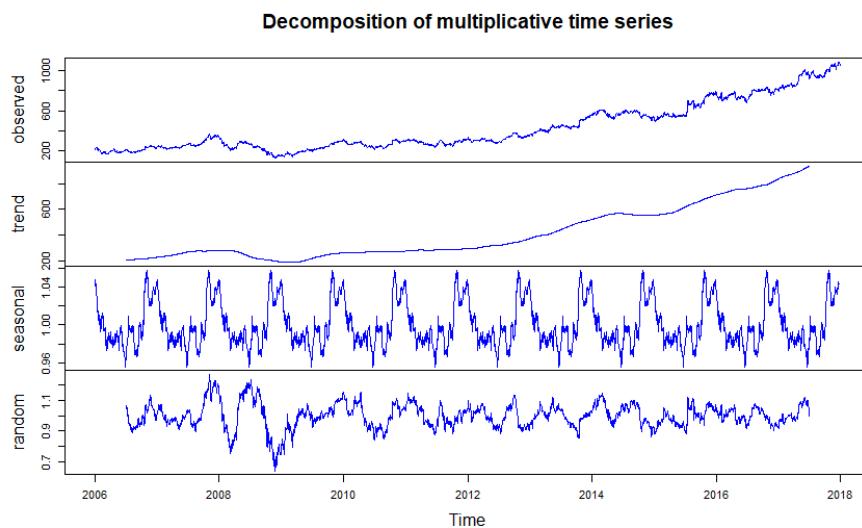


FIGURE 2.3 : La décomposition multiplicative de la série chronologique.

On observe que dans la décomposition additive et multiplicative, il y a une tendance haussière et aussi une saisonnalité. Le bruit de la série temporelle semble corrélé à la saisonnalité.

2.4 Différencier une série temporelle

La différenciation peut aider à stabiliser la moyenne de la série chronologique en supprimant les changements de niveau d'une série chronologique, et ainsi en éliminant (ou en réduisant) la tendance et la saisonnalité. La différenciation est une solution courante utilisée pour stationner la variable. Nous allons effectuer une différenciation à l'aide de la fonction R diff.

```
xtsdiff1 <- diff(xts, differences=1)
tsdiff1 <- diff(ts, differences=1)
plot.xts(xtsdiff1, col = "blue")
```

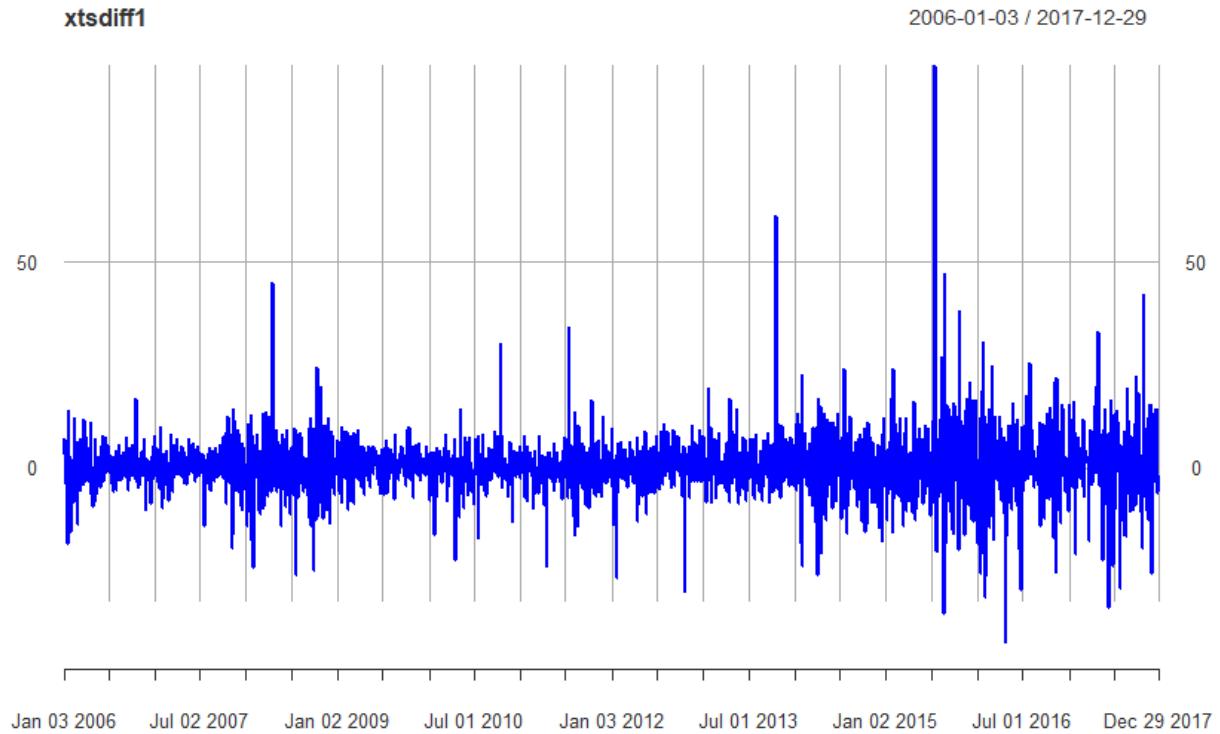


FIGURE 2.4 : Différencier la série chronologique.

On fait un Dickey-Fuller test :

```
adf.test(tsdiff1, alternative = "stationary", k = 0)
```

On obtient le résultat suivant :

```
#      Augmented Dickey-Fuller Test

# data: tsdiff1
# Dickey-Fuller = -53.448, Lag order = 0, p-value = 0.01
# alternative hypothesis: stationary
```

Interprétation du test :

H_0 : La série temporelle comporte une racine unitaire (non stationnaire).

H_1 : La série temporelle ne comporte pas une racine unitaire (stationnaire).

Etant donné que la p-value calculée est inférieur au niveau de signification $\alpha = 5\%$, on doit rejeter l'hypothèse nulle H_0 . Le risque de rejeter l'hypothèse nulle H_0 alors qu'elle est vraie est inférieur à 5%.

Pour trouver la fréquence dominante de la série chronologique d'origine, on utilise la commande suivante :

```
findfrequency(xts)
```

On trouve :

```
# [1] 1
```

Pour trouver la fréquence dominante des séries chronologiques différencierées, on utilise la commande suivante :

```
findfrequency(xts)
```

On trouve :

```
# [1] 18
```

La série temporelle (ci-dessus) semble être stationnaire (p-value < 0.05 ce qui est affirmé par le test de Dickey-Fuller).

Chapitre 3

ARIMA : Modèle de prédiction des séries temporelles

3.1 Le modèle ARMA

En statistique, les modèles ARMA, ou aussi modèle de Box-Jenkins, sont les principaux modèles de séries temporelles. Étant donné une série temporelle X_t , le modèle ARMA est un outil pour comprendre et prédire, éventuellement, les valeurs futures de cette série.

3.1.1 Le processus AR

Le processus AR signifie autorégressif. Concrètement si l'on considère un processus stationnaire X_t , on considère qu'il est autorégressif d'ordre p si l'on peut expliquer sa valeur à l'instant t en utilisant ses p termes précédents.

Mathématiquement cela signifie que :

$$\forall t : X_t = \sum_{i=1}^p \alpha_i X_{t-i} + \epsilon_t \text{ avec } \epsilon_t \text{ l'erreur et } \alpha_i \text{ des réels.}$$

3.1.2 Le processus MA

MA signifie ‘moving average’ ou en français moyenne mobile. Soit X_t une série temporelle, on considère que c'est un processus MA d'ordre p si on peut exprimer sa valeur à l'instant t comme une combinaison linéaire d'erreurs aléatoires (bruit blanc).

Mathématiquement cela signifie que :

$$\forall t : X_t = \sum_{i=1}^p \alpha_i \epsilon_{t-i} + \epsilon_t \text{ avec } \epsilon_t \text{ l'erreur et } \alpha_i \text{ des réels.}$$

3.1.3 Le processus ARMA

Le modèle ARMA est tout simplement une combinaison d'un processus AR et d'un processus MA. Cela permet de modéliser des séries temporelles plus complexes.

Un modèle ARMA d'ordre (p,q) s'écrit donc sous la forme :

$$\forall t : X_t = \sum_{i=1}^p \alpha_i X_{t-i} + \epsilon_t + \sum_{i=1}^q \beta_i \epsilon_{t-i} \text{ avec } \beta_i \text{ des réels.}$$

Cependant l'une des limitations de ce modèle est qu'il ne peut modéliser que des séries temporelles stationnaires. Ainsi il ne peut modéliser une série temporelle avec une tendance linéaire croissante. C'est pour pallier ce problème que le modèle **ARIMA** a été développé.

3.2 Le modèle ARIMA

Le I du modèle ARIMA signifie ‘integrated’ pour intégration. En différenciant les séries temporelles, il est possible de retirer les tendances qu'elles présentent pour les stationnariser.

En différenciant la série une fois , la dépendance temporelle linéaire est éliminée et la différence est stationnaire. De la même façon, une tendance quadratique peut être éliminée en différenciant la série deux fois. Une fois la série stationnarisée, il est alors possible d'appliquer le modèle ARMA.

Le modèle ARIMA est donc une combinaison de ce processus de différenciation et du processus ARMA classique.

3.3 Sélection d'un modèle ARIMA candidat

L'étape suivante consiste à sélectionner le modèle ARIMA approprié, ce qui signifie trouver les valeurs les plus appropriées de **p** et **q** pour un modèle **ARIMA(p,d,q)**. Pour cela, On doit généralement examiner le corrélogramme et le corrélogramme partiel de la série temporelle stationnaire.

Autocorrélation

L'autocorrélation (ou l'autocovariance) d'une série fait référence au fait que dans une série temporelle ou spatiale, la mesure d'un phénomène à un instant t peut être corrélée aux mesures précédentes (au temps $t - 1, t - 2, t - 3$, etc.) ou aux mesures suivantes (à $t + 1, t + 2, t + 3, \dots$).

Une série autocorrélée est ainsi corrélée à elle-même, avec un décalage (lag) donné.

Voici la définition mathématiques de l'autocovariance et de l'autocorrélation pour une variable X_t de moyenne μ et d'écart-type σ .

Autocovariance de X pour un décalage de k :

$$\gamma_k = E[(X_t - \mu)(X_{t+k} - \mu)]$$

Autocorrélation de X pour un décalage de k :

$$\rho_k = \frac{\gamma_k}{\sigma^2}$$

Pour tracer un corrélogramme et un corrélogramme partiel, On peut utiliser les fonctions **acf()** et **pacf()** dans R, respectivement.

On peut tracer l'autocorrélation de la série (pour différents décalages k) de la manière suivante :

```
Acf(xtsdiff1, lag.max=60)
```

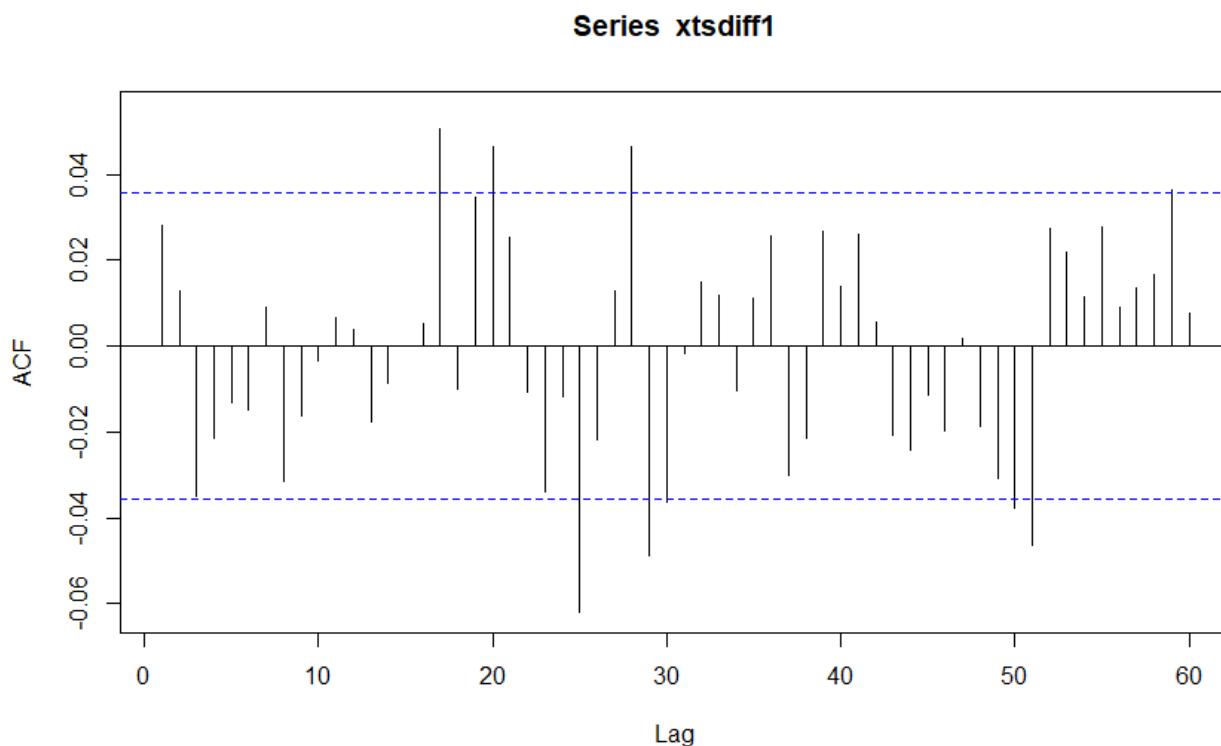


FIGURE 3.1 : Corrélogramme de la série chronologique.

Le graphique ainsi obtenu est un corrélogramme.

L'autocorrélation est forte pour les décalages qui dépassent la ligne bleue.

L'autocorrélation est particulièrement forte pour les décalages 16, 19, 24, 27, 28, 50 et 51.

Significativité

La droite horizontale pointillée sur le graphique issu de la fonction "acf" nous indique le seuil critique au-delà duquel l'autocorrélation est considérée significative.

En effet, sous hypothèse d'indépendance, la corrélation croisée de deux séries X et Y (de même taille n , et de même moyenne et écart-type) sera dans 95% des cas comprise dans l'intervalle.

Pour obtenir les valeurs d'autocorrélation, on utilise la commande suivante :

```
Acf(xtsdiff1, lag.max=60)
```

On obtient les valeurs suivantes :

```
# Autocorrelations of series 'xtsdiff1', by lag

#    0     1     2     3     4     5     6     7     8     9
# 1.000  0.028  0.013 -0.035 -0.021 -0.013 -0.015  0.009 -0.031 -0.016
# 10     11     12     13     14     15     16     17     18     19
# -0.004  0.007  0.004 -0.018 -0.008  0.000  0.005  0.051 -0.010  0.035
# 20     21     22     23     24     25     26     27     28     29
# 0.047  0.025 -0.011 -0.034 -0.012 -0.062 -0.022  0.013  0.046 -0.049
# 30     31     32     33     34     35     36     37     38     39
# -0.036 -0.002  0.015  0.012 -0.010  0.011  0.026 -0.030 -0.022  0.027
# 40     41     42     43     44     45     46     47     48     49
# 0.014  0.026  0.006 -0.021 -0.024 -0.011 -0.020  0.002 -0.019 -0.031
# 50     51     52     53     54     55     56     57     58     59
# -0.038 -0.046  0.027  0.022  0.012  0.028  0.009  0.014  0.017  0.036
# 60
# 0.008
```

Autocorrélation partielle

Observons à nouveau le corrélogramme. On a une forte autocorrélation pour un décalage de 16, de 19, de 24... Or l'autocorrélation pour le décalage de 19 et 24 pourrait très bien être une conséquence de celle existant pour le décalage de 16.

L'autocorrélation partielle permet de mesurer l'autocorrélation d'un signal pour un décalage k "indépendamment" des autocorrélations pour les décalages inférieurs.

```
Pacf(xtsdiff1, lag.max=60)
```

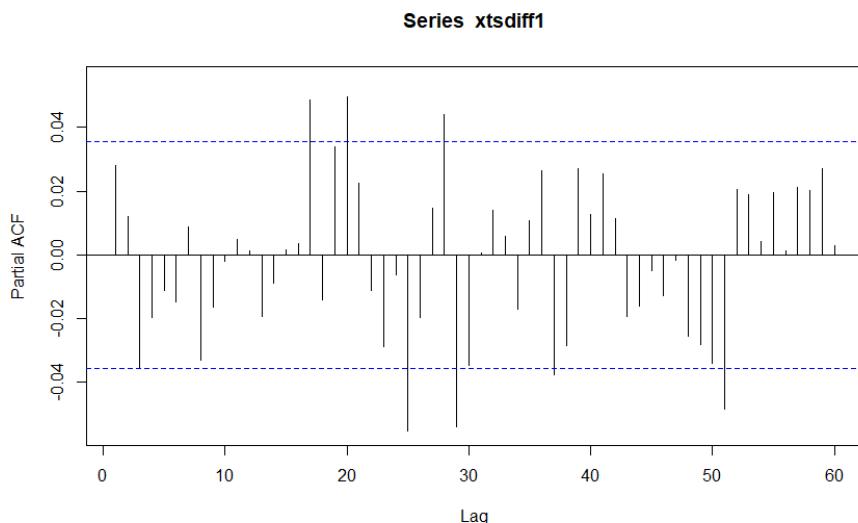


FIGURE 3.2 : Corrélogramme partiel de la série chronologique.

On vérifie ici que les autocorrélations observées aux décalages 19 et 24 n'étaient pas un effet résiduel de l'autocorrélation pour un décalage de 16.

Pour obtenir les valeurs d'autocorrélation partiel, On utilise la commande suivante :

```
Pacf(xtsdiff1, lag.max=60, plot=FALSE)
```

On obtient les valeurs suivantes :

```
# Partial autocorrelations of series 'xtsdiff1', by lag
#
#   1     2     3     4     5     6     7     8     9     10
# 0.028  0.012 -0.036 -0.020 -0.011 -0.015  0.009 -0.033 -0.016 -0.002
# 11    12    13    14    15    16    17    18    19    20
# 0.005  0.001 -0.019 -0.009  0.002  0.004  0.049 -0.014  0.034  0.050
# 21    22    23    24    25    26    27    28    29    30
# 0.023 -0.011 -0.029 -0.006 -0.055 -0.020  0.015  0.044 -0.054 -0.035
# 31    32    33    34    35    36    37    38    39    40
# 0.001  0.014  0.006 -0.017  0.011  0.026 -0.038 -0.028  0.027  0.013
# 41    42    43    44    45    46    47    48    49    50
# 0.026  0.011 -0.019 -0.016 -0.005 -0.013 -0.002 -0.026 -0.028 -0.034
# 51    52    53    54    55    56    57    58    59    60
# -0.048  0.021  0.019  0.004  0.020  0.001  0.021  0.020  0.027  0.003
```

Maintenant, nous pourrions comparer l'échantillon ACF et PACF à ceux de divers modèles théoriques ARMA. En utilisant les propriétés de l'ACF et du PACF comme guide pour estimer les modèles plausibles et sélectionnez les p, q et d appropriés. L'alternative à cela est discutée ensuite.

3.4 Fitting d'un modèle ARIMA

R fournit une fonction **auto.arima**, qui renvoie le meilleur modèle ARIMA selon la valeur AIC, AICc ou BIC. La fonction effectue une recherche sur un modèle possible dans les contraintes d'ordre fournies. **auto.arima** utilise une variante de l'algorithme Hyndman-Khandakar [[voir annexe](#)] (Hyndman & Khandakar, 2008) , qui combine des tests de racine unitaire, la minimisation de l'AICc et le MLE pour obtenir un modèle ARIMA.

Nous entraînons 6 modèles avec différentes données d'entraînement. Par exemple, le modèle « tsarima240 » est entraîné avec l'ensemble de la série chronologique à l'exclusion des 240 dernières données quotidiennes.

```
tsarima240 <- auto.arima(head(xts, -240), max.p = 3, max.q = 3, max.d = 3)
print(tsarima240)
```

On obtient la sortie suivante :

```
# Series: head(xts, -240)
# ARIMA(0,1,0) with drift
#
# Coefficients:
#       drift
#       0.2200
# s.e. 0.1264
#
# sigma^2 estimated as 44.43: log likelihood=-9210.91
# AIC=18425.82    AICc=18425.82    BIC=18437.67

autoplot(tsarima240)
```

Les conditions de stationnarité du modèle sont que les p racines complexes se trouvent à l'extérieur du cercle unité, et les conditions d'inversibilité sont que les q racines complexes se trouvent en dehors du cercle unité. On peut donc voir si le modèle est proche de l'inversibilité ou de la stationnarité par un tracé des racines par rapport au cercle unité complexe. Il est plus facile de tracer les racines inverses à la place, car elles devraient toutes se trouver dans le cercle unité. La stationnarité et l'inversibilité reposent sur l'étude des racines du polynôme autorégressif (stationnarité) et du polynôme moyenne mobile (inversibilité).

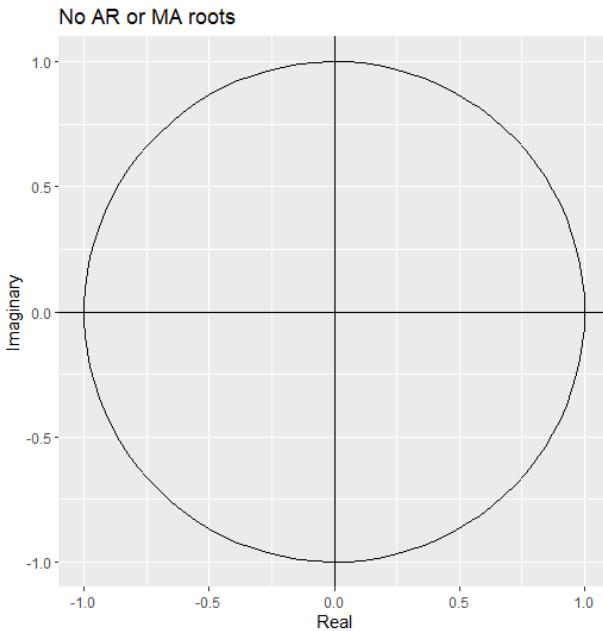


FIGURE 3.3 : Graphe autoplot de la série chronologique à l'exclusion des 240 dernières données.

On constate qu'il n'existe pas de racine pour le modèle **tsarima240**, donc le processus est ni stationnaire, ni inversible (**ARIMA (0,1,0)**).

le modèle « tsarima120 » est entraîné avec l'ensemble de la série chronologique à l'exclusion des 120 dernières données quotidiennes.

```
tsarima120 <- auto.arima(head(xts, -120), max.p = 3, max.q = 3, max.d = 3)
print(tsarima120)
```

On obtient la sortie suivante :

```
# Series: head(xts, -120)
# ARIMA(0,1,1) with drift
#
# Coefficients:
#          ma1     drift
#        0.0260   0.2539
# s.e.  0.0183   0.1291
#
# sigma^2 estimated as 45.91:  log likelihood=-9656
# AIC=19317.99    AICc=19318    BIC=19335.91

autoplot(tsarima120)
```

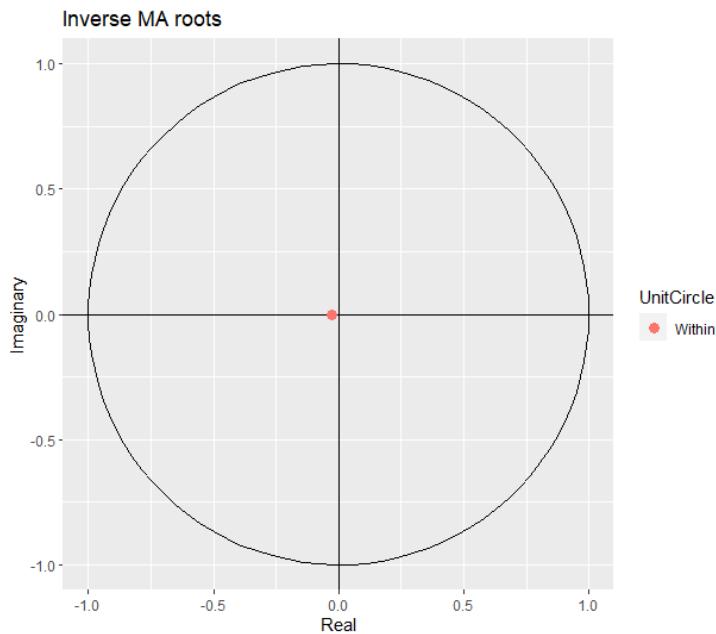


FIGURE 3.4 : Graphe autoplot de la série chronologique à l'exclusion des 120 dernières données.

On constate qu'il existe une racine pour le polynôme moyenne mobile, donc le processus n'est pas stationnaire car il n'y a pas de racine à l'extérieur du cercle, mais inversible (**ARIMA (0,1,1)**).

le modèle « tsarima60 » est entraîné avec l'ensemble de la série chronologique à l'exclusion des 60 dernières données quotidiennes.

```
tsarima60 <- auto.arima(head(xts, -60), max.p = 3, max.q = 3, max.d = 3)
print(tsarima60)
```

On obtient la sortie suivante :

```
# Series: head(xts, -60)
# ARIMA(0,1,1) with drift
#
# Coefficients:
#          ma1     drift
#        0.0258   0.2532
# s.e.  0.0181   0.1286
#
# sigma^2 estimated as 46.55:  log likelihood=-9876.46
# AIC=19758.93    AICc=19758.94    BIC=19776.9

autoplot(tsarima60)
```

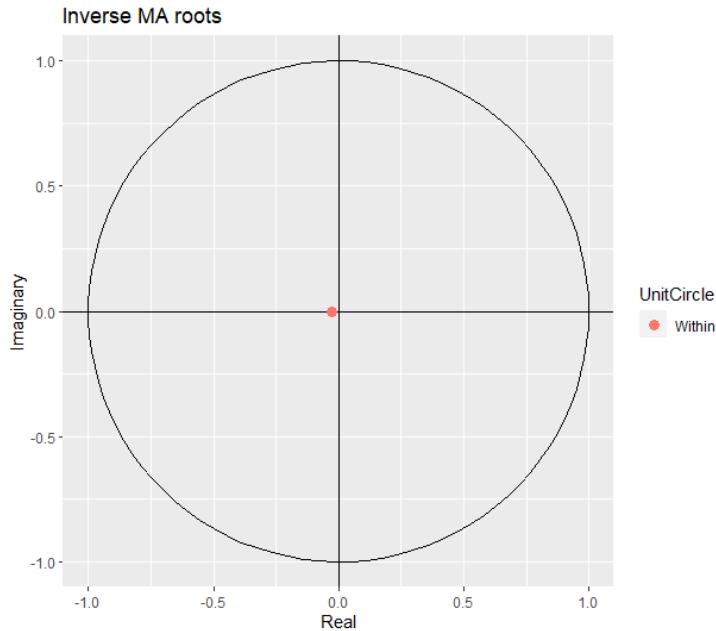


FIGURE 3.5 : Graphe autoplot de la série chronologique à l'exclusion des 60 dernières données.

On constate qu'il existe une racine pour le polynôme moyenne mobile, donc le processus n'est pas stationnaire car il n'y a pas de racine à l'extérieur du cercle, mais inversible (**ARIMA (0,1,1)**).

le modèle « tsarima30 » est entraîné avec l'ensemble de la série chronologique à l'exclusion des 30 dernières données quotidiennes.

```
tsarima30 <- auto.arima(head(xts, -30), max.p = 3, max.q = 3, max.d = 3)
print(tsarima30)
```

On obtient la sortie suivante :

```
# Series: head(xts, -30)
# ARIMA(0,1,0) with drift
#
# Coefficients:
#       drift
#       0.2740
# s.e.  0.1258
#
# sigma^2 estimated as 47.28:  log likelihood=-10000.36
# AIC=20004.72    AICc=20004.72    BIC=20016.73

autoplot(tsarima30)
```

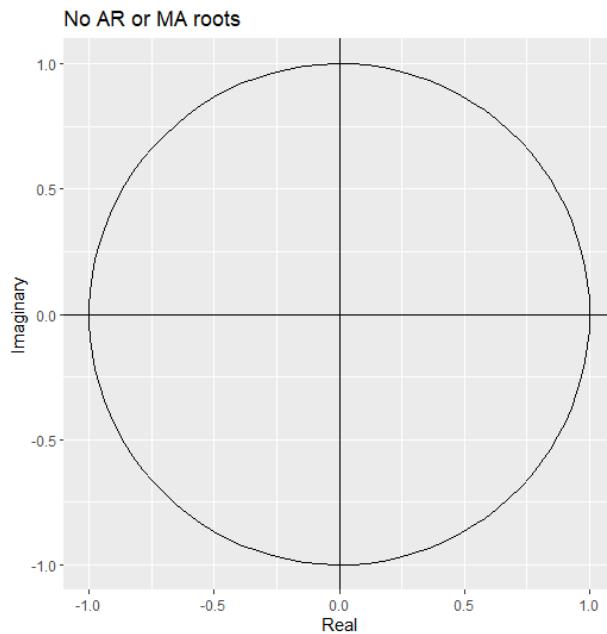


FIGURE 3.6 : Graphe autoplot de la série chronologique à l'exclusion des 30 dernières données.

On constate qu'il n'existe pas de racine pour le modèle **tsarima240**, donc le processus est ni stationnaire, ni inversible (**ARIMA (0,1,0)**).

le modèle « tsarima7 » est entraîné avec l'ensemble de la série chronologique à l'exclusion des 7 dernières données quotidiennes.

```
tsarima7 <- auto.arima(head(xts, -7), max.p = 3, max.q = 3, max.d = 3)
print(tsarima7)
```

On obtient la sortie suivante :

```
# Series: head(xts, -7)
# ARIMA(0,1,1) with drift
#
# Coefficients:
#          ma1     drift
#        0.0264   0.2863
# s.e.  0.0179   0.1293
#
# sigma^2 estimated as 47.8:  log likelihood=-10093.26
# AIC=20192.53    AICc=20192.54    BIC=20210.56

autoplot(tsarima7)
```

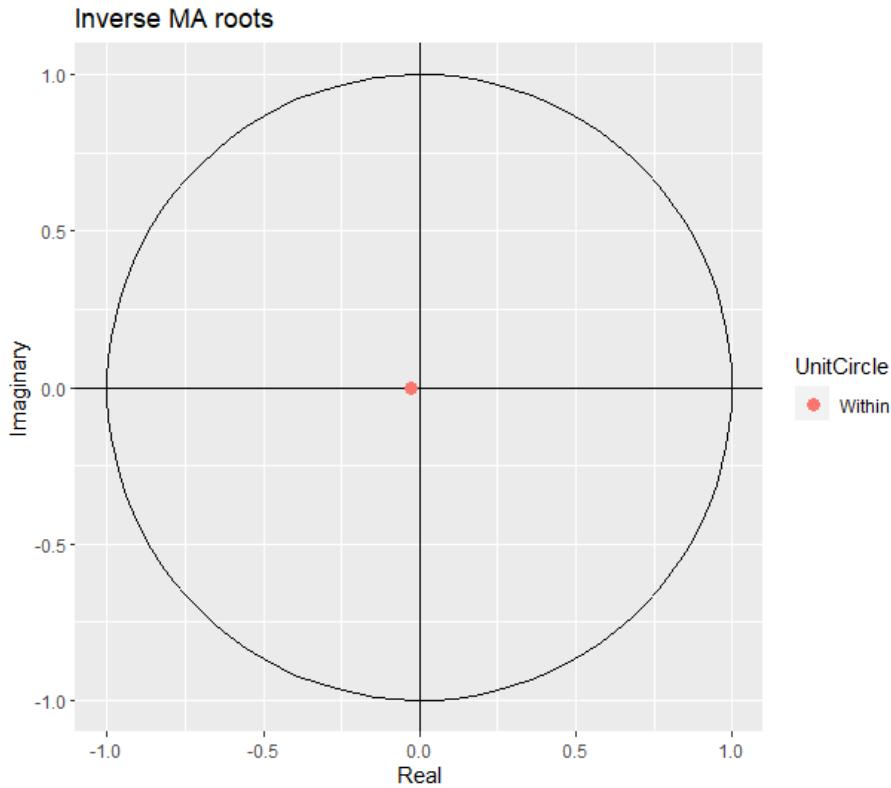


FIGURE 3.7 : Graphe autoplot de la série chronologique à l'exclusion des 7 dernières données.

On constate qu'il existe une racine pour le polynôme moyenne mobile, donc le processus n'est pas stationnaire car il n'y a pas de racine à l'extérieur du cercle, mais inversible (**ARIMA (0,1,1)**).

3.5 Validation du modèle

Il s'agit de vérifier notamment que les résidus du modèle ARMA estimé, résidus notés $\hat{\epsilon}_t$, vérifient les propriétés requises pour que l'estimation soit valide, à savoir qu'ils suivent un processus BB, non autocorrélé et de même variance, et qu'ils suivent une loi normale.

Si ces hypothèses ne sont pas rejetées, on peut alors mener des tests sur les paramètres.

- **Tests sur les résidus (Test de bruité) :**

- Par l'étude de la fonction d'auto-corrélation empirique.
- Test de normalité
- Test de stationnarité (test ADF)

- **Tests sur les paramètres :** On vérifie tout d'abord que les racines des polynômes AR et MA ne sont pas égales à 1. On teste la significativité des retards du modèle ARMA par des tests de Student.

3.6 Choix d'un modèle parmi plusieurs

Comme on l'a vu précédemment l'étude des auto-covariances, auto-corrélations et auto-corrélations partielles peut conduire à certaines hypothèses sur la nature du modèle. Une fois quelques modèles choisis, et leur paramètres estimés, on utilise les critères suivants pour choisir le meilleur modèle :

Critères d'information Pour choisir le modèle qui effectue le meilleur compromis entre :

- ajustement à la série de données.
- complexité du modèle.

Il est en effet très important de prendre en compte ce compromis, car si on ne s'intéressait qu'à coller au mieux aux données, on serait tenter de choisir un modèle ARMA avec un très grand nombre de paramètres. Or, plus il y a de paramètres, plus il faut de données pour les estimer. Et donc pour un nombre d'observations fixé de la série, plus le modèle sera complexe, moins bien seront estimés les paramètres. Les critères de choix de modèles les plus courants sont :

- **Le critère AIC (Akaike Information Criterion)** : , qui sera généralement préféré si l'objectif de l'étude est de faire de la prévision.
- **Le critère BIC (Bayesian Information Criterion)** : sera quant à lui généralement préféré si l'objectif de l'étude est de s'ajuster à la série observée.

Les modèles ayant la plus petite valeur du critère devront être choisis.

Ces deux critères conduisent donc à sélectionner des modèles dont la vraisemblance est grande, en la pénalisant par la complexité du modèle.

Principe de parcimonie :

Lorsqu'on veut modéliser une série chronologique, par un processus Stochastique et dans le cas où les critères d'information AIC et BIC de deux ou plusieurs modèles retenus seraient très proches ou contradictoires, nous faisons intervenir ce principe qui cherche à minimiser le nombre de paramètre requis ; il est préférable de conserver un modèle qui est moins bon, mais qui contient moins de paramètres.

3.7 Prévision à l'aide d'un modèle ARIMA

Prévoir les 240 prochaines séries temporelles :

```
tsforecasts240 <- forecast(tsarima240, h = 240)
autoplot(tsforecasts240)
```

Les prévisions du modèle **tsforecasts240** choisi sont présentées sur la figure suivante :

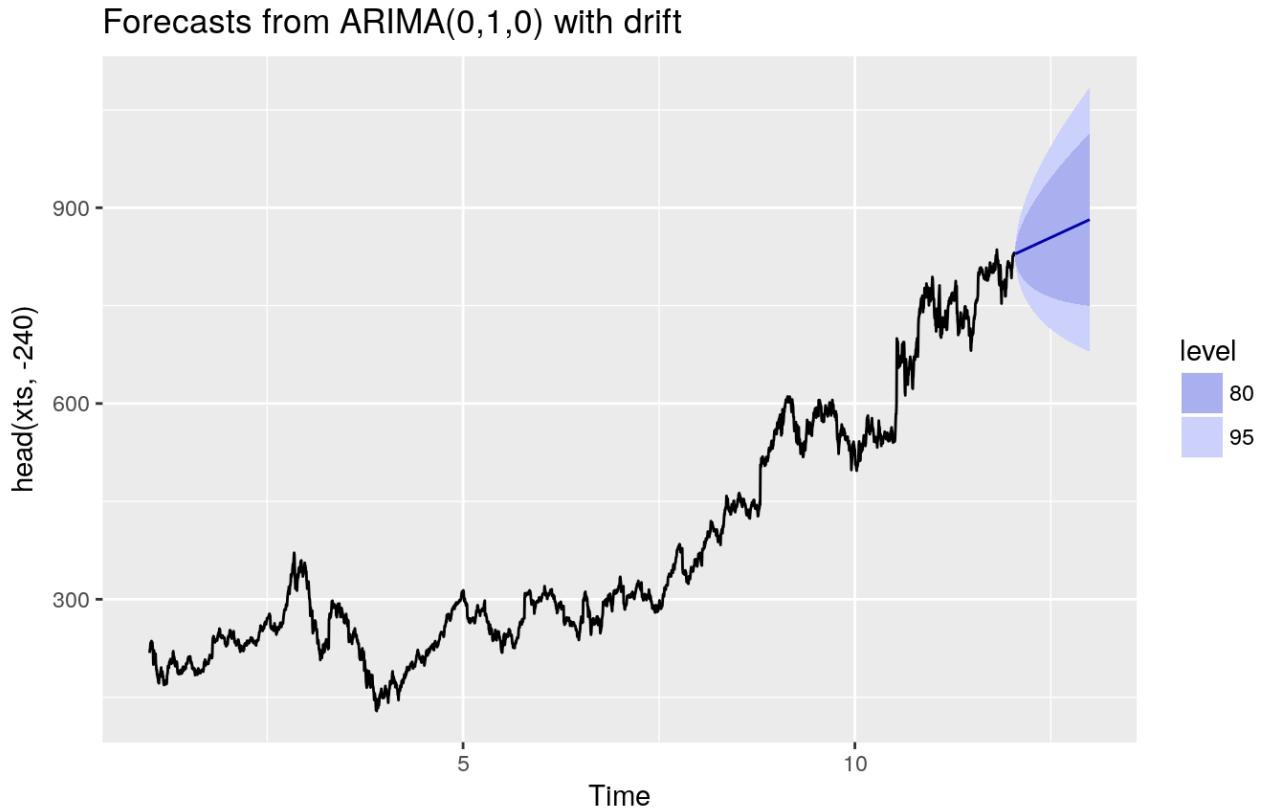


FIGURE 3.8 : Graphe de prévision des 240 prochaines séries temporelles.

```
accuracy(tsforecasts240, head(tail(xts, 240), 240))

#               ME        RMSE       MAE       MPE      MAPE
# Training set 7.830510e-05  6.662823  4.358564 -0.04023279 1.255910
# Test set     8.958999e+01 107.462582 90.356608  9.04790109 9.141123

#           MASE      ACF1
# Training set 0.05516544 0.02251657
# Test set     1.14362484          NA
```

```

accuracy(tsforecasts240, head(tail(xts, 240), 120))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 7.83051e-05 6.662823 4.358564 -0.04023279 1.255910
# Test set     5.47361e+01 76.420359 56.269332  5.74033804 5.926781

#               MASE      ACF1
# Training set 0.05516544 0.02251657
# Test set     0.71218926        NA

accuracy(tsforecasts240, head(tail(xts, 240), 60))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 0.0000783051 6.662823 4.358564 -0.04023279 1.255910
# Test set     9.1885039597 15.253580 12.254974  1.06607535 1.438961

#               MASE      ACF1
# Training set 0.05516544 0.02251657
# Test set     0.15510867        NA

accuracy(tsforecasts240, head(tail(xts, 240), 30))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 0.0000783051 6.662823 4.358564 -0.04023279 1.255910
# Test set     6.1004992801 13.506026 11.611246  0.70604785 1.377561

#               MASE      ACF1
# Training set 0.05516544 0.02251657
# Test set     0.14696113        NA

accuracy(tsforecasts240, head(tail(xts, 240), 7))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 7.830510e-05 6.662823 4.358564 -0.04023279 1.25591
# Test set     1.395139e+01 18.322229 15.711395  1.63313782 1.84643

#               MASE      ACF1
# Training set 0.05516544 0.02251657
# Test set     0.19885586        NA

```

La fonction **accuracy** renvoie une plage de mesures récapitulatives de la précision des prévisions. On remarque que la prédiction est de moins en moins bonne avec le temps, ce qui est logique.

Prévoir les 120 prochaines séries temporelles :

```
tsforecasts120 <- forecast(tsarima120, h = 120)
autoplot(tsforecasts120)
```

Les prévisions du modèle **tsforecasts120** choisi sont présentées sur la figure suivante :

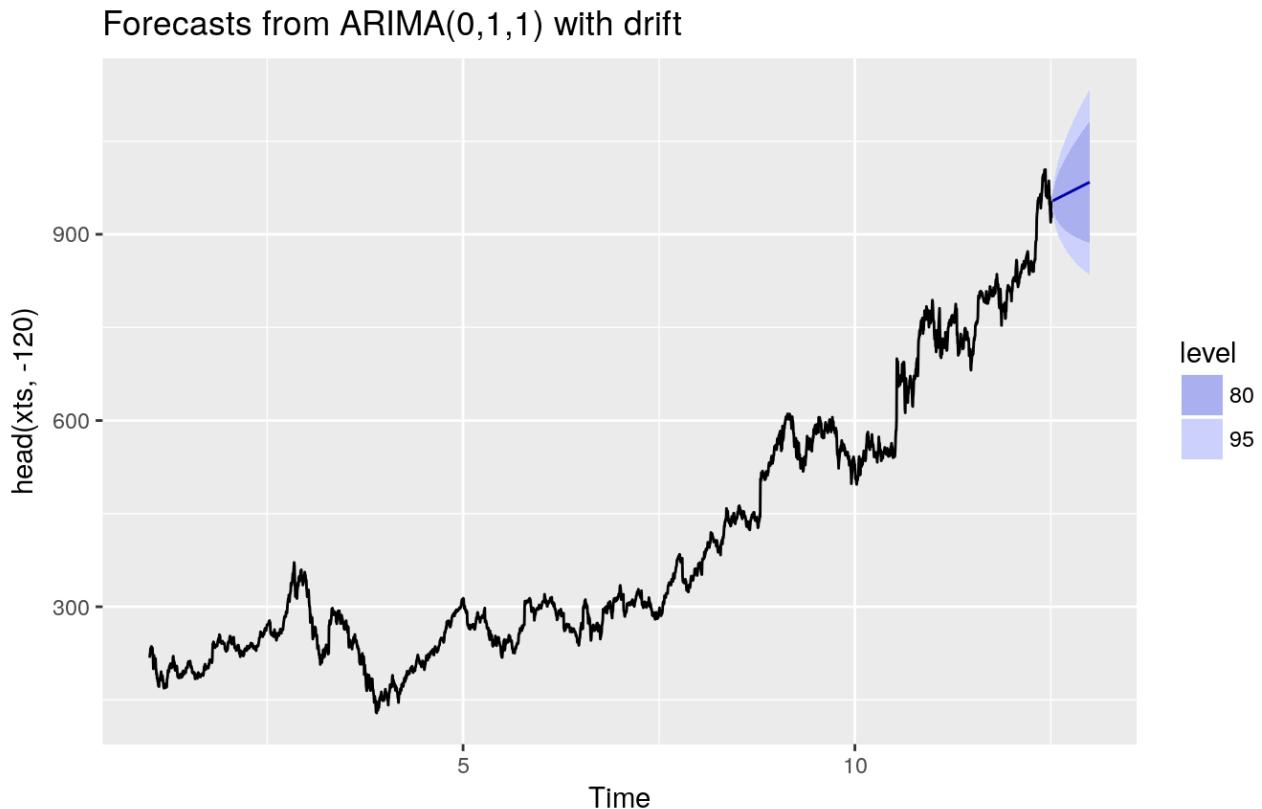


FIGURE 3.9 : Graphe de prévision des 120 prochaines séries temporelles.

```
accuracy(tsforecasts120, head(tail(xts, 120), 120))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 8.622115e-05 6.77230 4.444226 -0.04438106 1.233619
# Test set     2.423396e+01 47.76573 39.743165  2.24668716 3.900841

#             MASE      ACF1
# Training set 0.0536555 0.0003654603
# Test set     0.4798225          NA

accuracy(tsforecasts120, head(tail(xts, 120), 60))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 8.622115e-05 6.77230 4.444226 -0.04438106 1.233619
# Test set     -1.063842e+01 23.06933 20.380001 -1.15983472 2.148473
```

```
#           MASE      ACF1
# Training set 0.0536555 0.0003654603
# Test set     0.2460494      NA
```

Pour :

```
accuracy(tsforecasts120, head(tail(xts, 120), 30))
```

et

```
accuracy(tsforecasts120, head(tail(xts, 120), 7))
```

On obtient :

```
#           ME      RMSE      MAE      MPE      MAPE
# Training set 8.622115e-05 6.77230 4.444226 -0.04438106 1.233619
# Test set     -2.701019e+00 24.38488 21.571816 -0.34178642 2.256133
```

```
#           MASE      ACF1
# Training set 0.0536555 0.0003654603
# Test set     0.2604383      NA
```

et

```
#           ME      RMSE      MAE      MPE      MAPE
# Training set 8.622115e-05 6.77230 4.444226 -0.04438106 1.233619
# Test set     2.558612e+01 27.20241 25.586116 2.60125011 2.601250
```

```
#           MASE      ACF1
# Training set 0.0536555 0.0003654603
# Test set     0.3089033      NA
```

La fonction **accuracy** renvoie une plage de mesures récapitulatives de la précision des prévisions. On remarque que la prédiction est de moins en moins bonne avec le temps, ce qui est logique.

Prévoir les 60 prochaines séries temporelles :

```
tsforecasts60 <- forecast(tesarima60, h = 60)
autoplot(tsforecasts60)
```

Les prévisions du modèle **tsforecasts60** choisi sont présentées sur la figure suivante :

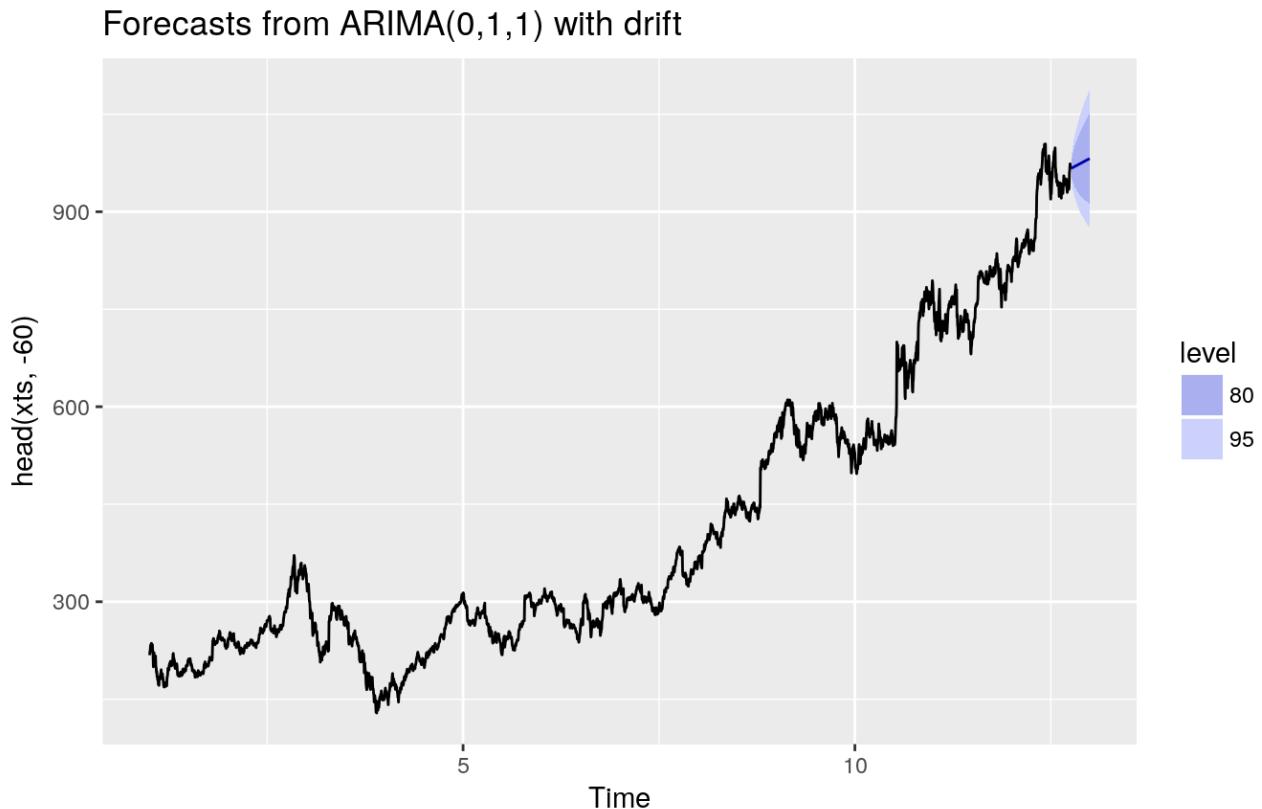


FIGURE 3.10 : Graphe de prévision des 60 prochaines séries temporelles.

```
accuracy(tsforecasts60, head(tail(xts, 60), 60))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 2.558875e-05 6.819537 4.490829 -0.04345203 1.222988
# Test set     6.130805e+01 65.547323 61.308054  5.86591739 5.865917

#             MASE      ACF1
# Training set 0.0531086 0.0004171777
# Test set     0.7250299          NA

accuracy(tsforecasts60, head(tail(xts, 60), 30))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 2.558875e-05 6.819537 4.490829 -0.04345203 1.222988
# Test set     4.860496e+01 53.330998 48.604964  4.72159500 4.721595
```

```
#           MASE      ACF1
# Training set 0.0531086 0.0004171777
# Test set     0.5748030          NA

accuracy(tsforecasts60, head(tail(xts, 60), 7))

#           ME      RMSE      MAE      MPE      MAPE
# Training set 2.558875e-05 6.819537 4.490829 -0.04345203 1.222988
# Test set     2.922622e+01 30.361430 29.226218 2.92484821 2.924848
#           MASE      ACF1
# Training set 0.0531086 0.0004171777
# Test set     0.3456297          NA
```

La fonction **accuracy** renvoie une plage de mesures récapitulatives de la précision des prévisions. On remarque que la prédiction est de moins en moins bonne avec le temps, ce qui est logique.

Prévoir les 30 prochaines séries temporelles :

```
tsforecasts30 <- forecast(tsarima30, h = 30)
autoplot(tsforecasts30)
```

Les prévisions du modèle **tsforecasts30** choisi sont présentées sur la figure suivante :

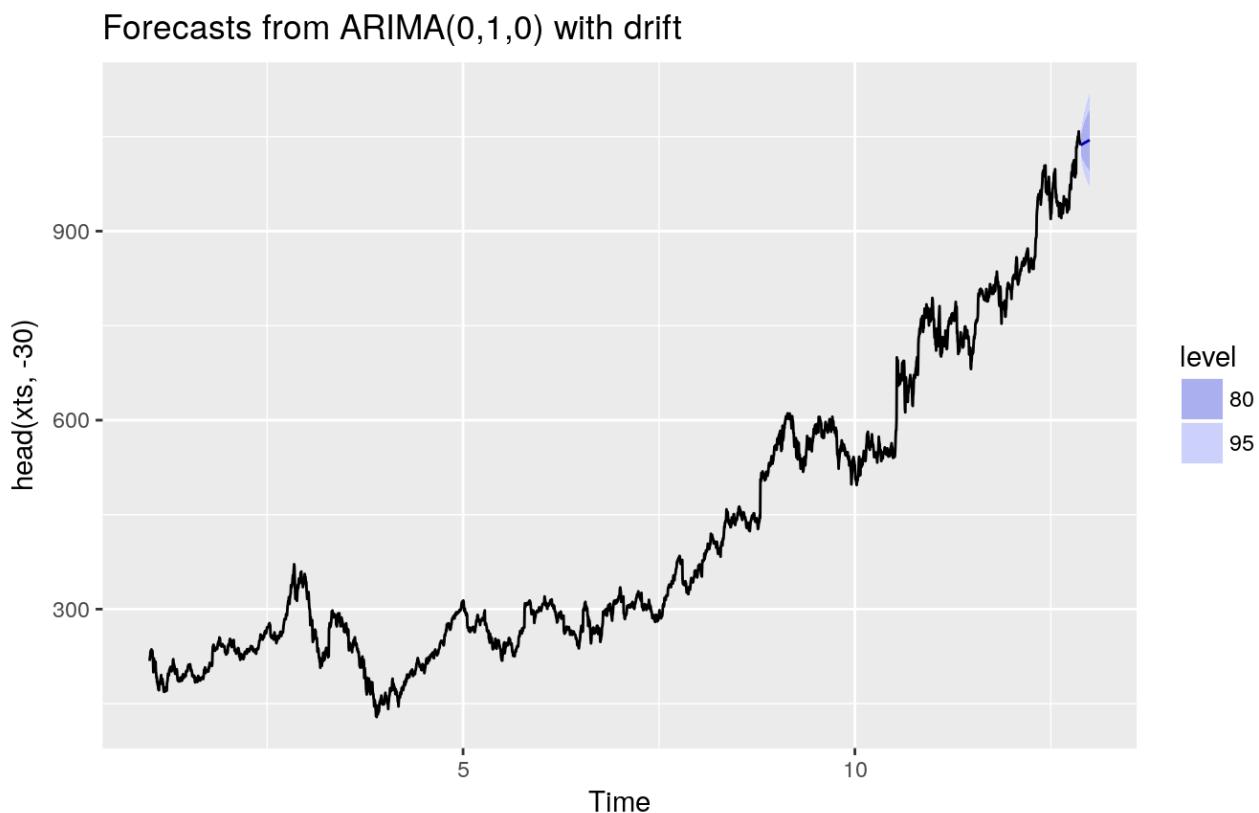


FIGURE 3.11 : Graphe de prévision des 30 prochaines séries temporelles.

```
accuracy(tsforecasts30, head(tail(xts, 30), 30))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 7.278553e-05 6.873924 4.512587 -0.04844246 1.217538
# Test set     1.150902e+01 20.032703 16.721867 1.06840749 1.579102

#             MASE      ACF1
# Training set 0.05248212 0.02360246
# Test set     0.19447805      NA

accuracy(tsforecasts30, head(tail(xts, 30), 7))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 7.278553e-05 6.873924 4.512587 -0.04844246 1.217538
# Test set     1.246132e+01 16.842987 13.501256 1.17494948 1.275424
```

```
#           MASE      ACF1
# Training set 0.05248212 0.02360246
# Test set     0.15702183      NA
```

La fonction **accuracy** renvoie une plage de mesures récapitulatives de la précision des prévisions. On remarque que la prédiction est de moins en moins bonne avec le temps, ce qui est logique.

Prévoir les 7 prochaines séries temporelles :

```
tsforecasts7 <- forecast(tesarima7, h = 7)
autoplot(tsforecasts7)
```

Les prévisions du modèle **tsforecasts7** choisi sont présentées sur la figure suivante :

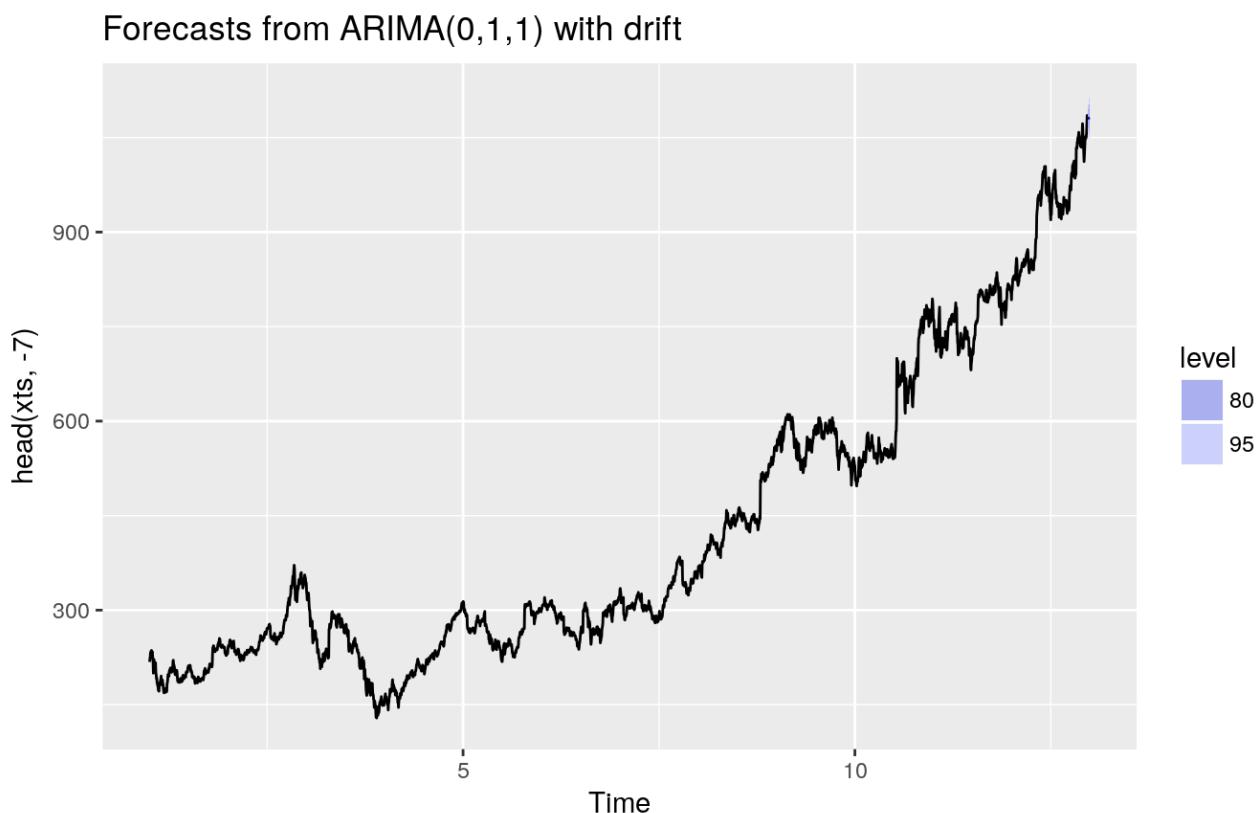


FIGURE 3.12 : Graphe de prévision des 7 prochaines séries temporelles.

```
accuracy(tsforecasts7, head(tail(xts, 7), 7))

#               ME      RMSE      MAE      MPE      MAPE
# Training set 2.198881e-05 6.910389 4.548387 -0.04940408 1.215092
# Test set     -1.667274e+01 18.368058 16.672738 -1.57178382 1.571784

#           MASE      ACF1
# Training set 0.0520307 0.0004004823
# Test set     0.1907257          NA
```

La fonction **accuracy** renvoie une plage de mesures récapitulatives de la précision des prévisions. On remarque que la prédition est de moins en moins bonne avec le temps, ce qui est logique.

Analyse de la prédition des 240 prochaines séries temporelles :

```
ggplot(data.frame(residuals = tsforecasts240$residuals), aes(residuals)) +
  geom_histogram(bins = 50, aes(y = ..density..), col = "red", fill = "red",
  alpha = 0.3) + geom_density()
```

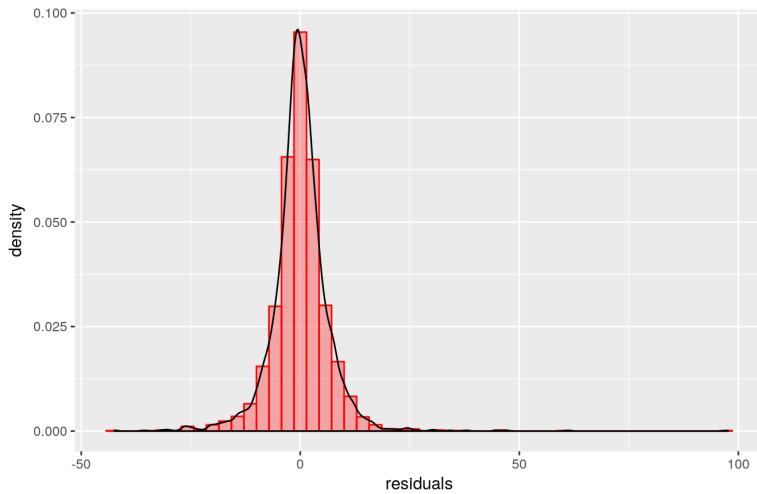


FIGURE 3.13 : Histogramme et densité des résiduels.

On remarque que la densité des résiduels semble à une densité Gaussienne centré en 0.

```
checkresiduals(tsforecasts240)
```

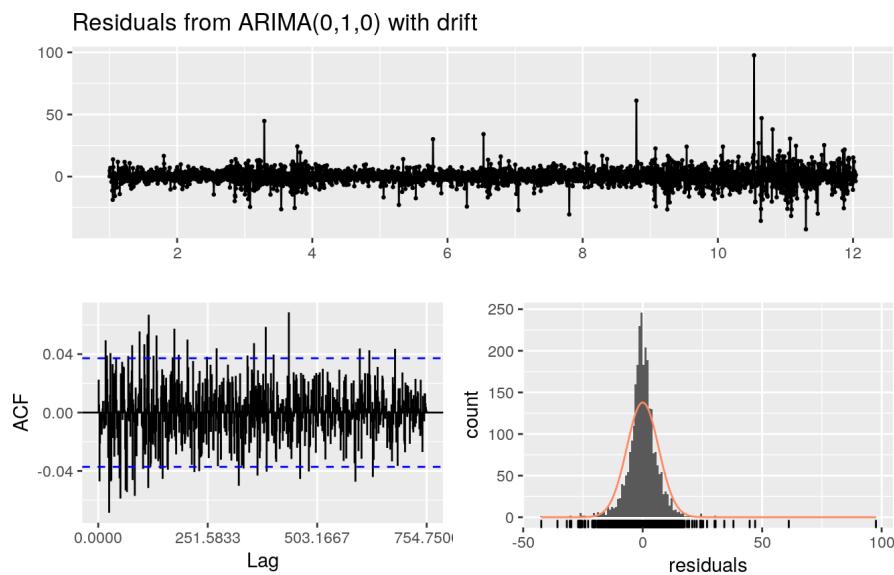


FIGURE 3.14 : Graphe des résiduels d'ARIMA(0,1,0)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance

résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

Le Test Q de Ljung-Box ou Test de Ljung-Box est un test statistique qui teste l'auto-corrélation d'ordre supérieur à 1. Il s'agit d'un test asymptotique qui n'a donc qu'une puissance très faible dans le cadre de petits échantillons.

```
Ljung-Box test

data: Residuals from ARIMA(0,1,1) with drift
Q* = 748.08, df = 501.17, p-value = 4.732e-12

Model df: 2. Total lags used: 503.166666666667
```

FIGURE 3.15 : Test de Ljung-Box

Le test de Ljung-Box tente de rejeter l'indépendance de certaines valeurs. **p-value < 0,05** : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

En général, ce qui est important ici est de garder à l'esprit qu'une valeur de p-value <0,05 vous permet de rejeter l'hypothèse nulle, mais une valeur de p-value>0,05 ne vous permet pas de confirmer l'hypothèse nulle.

En particulier, nous ne pouvons pas prouver l'indépendance des valeurs des séries chronologiques à l'aide du test Ljung-Box. Nous ne pouvons que prouver la dépendance.

On remarque que **p-value < 0,05** : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

Analyse de la prédition des 120 prochaines séries temporelles :

```
ggplot(data.frame(residuals = tsforecasts120$residuals), aes(residuals)) +
  geom_histogram(bins = 50, aes(y = ..density..), col = "red", fill = "red",
  alpha = 0.3) + geom_density()
```

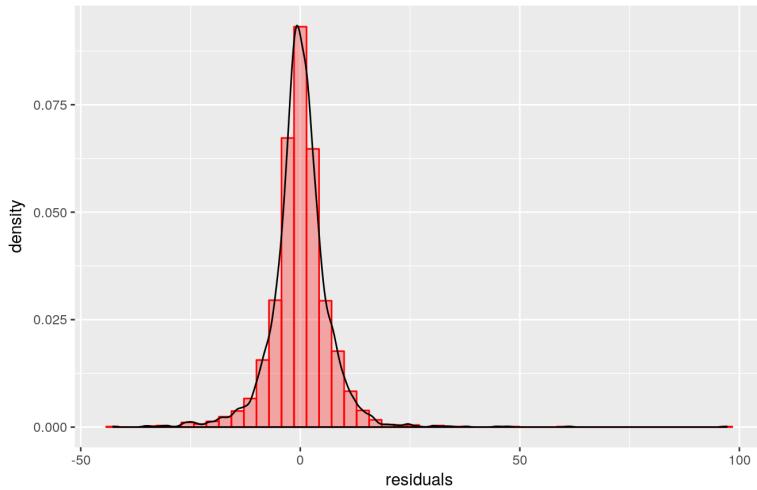


FIGURE 3.16 : Histogramme et densité des résiduels.

On remarque que la densité des résiduels semble à une densité Gaussienne centé en 0.

```
checkresiduals(tsforecasts120)
```

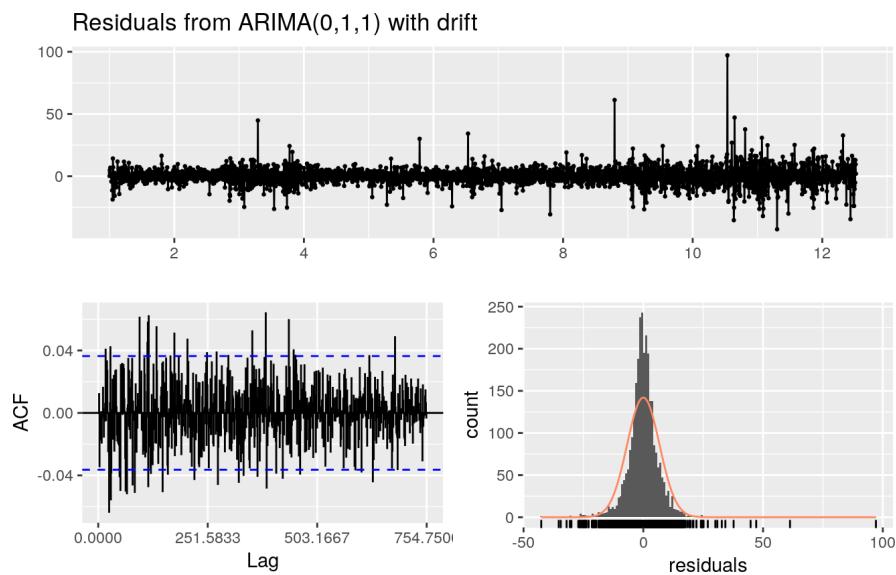


FIGURE 3.17 : Graphe des résiduels d'ARIMA(0,1,1)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance

résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

```
Ljung-Box test

data: Residuals from ARIMA(0,1,1) with drift
Q* = 748.08, df = 501.17, p-value = 4.732e-12

Model df: 2. Total lags used: 503.1666666666667
```

FIGURE 3.18 : Test de Ljung-Box

p-value < 0,05 : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

Analyse de la prédition des 60 prochaines séries temporelles :

```
ggplot(data.frame(residuals = tsforecasts60$residuals), aes(residuals)) +
  geom_histogram(bins = 50, aes(y = ..density..), col = "red", fill = "red",
  alpha = 0.3) + geom_density()
```

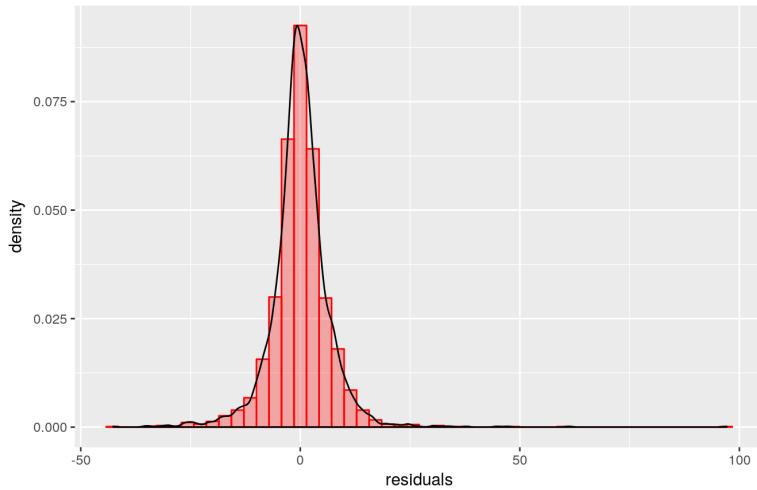


FIGURE 3.19 : Histogramme et densité des résiduels.

On remarque que la densité des résiduels semble à une densité Gaussienne centré en 0.

```
checkresiduals(tsforecasts60)
```

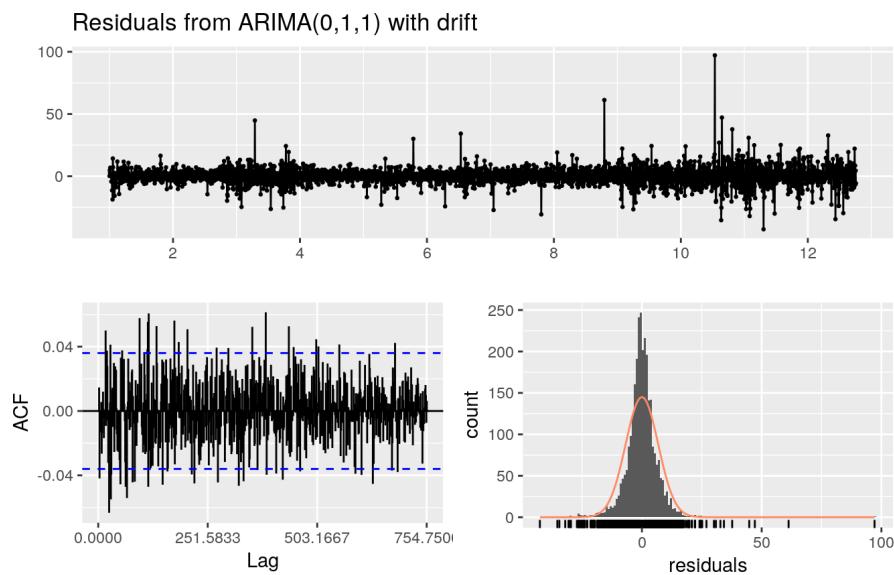


FIGURE 3.20 : Graphe des résiduels d'ARIMA(0,1,1)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance

résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

```
Ljung-Box test

data: Residuals from ARIMA(0,1,1) with drift
Q* = 741.62, df = 501.17, p-value = 1.395e-11

Model df: 2. Total lags used: 503.166666666667
```

FIGURE 3.21 : Test de Ljung-Box

p-value < 0,05 : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

Analyse de la prédiction des 30 prochaines séries temporelles :

```
ggplot(data.frame(residuals = tsforecasts30$residuals), aes(residuals)) +
  geom_histogram(bins = 50, aes(y = ..density..), col = "red", fill = "red",
  alpha = 0.3) + geom_density()
```

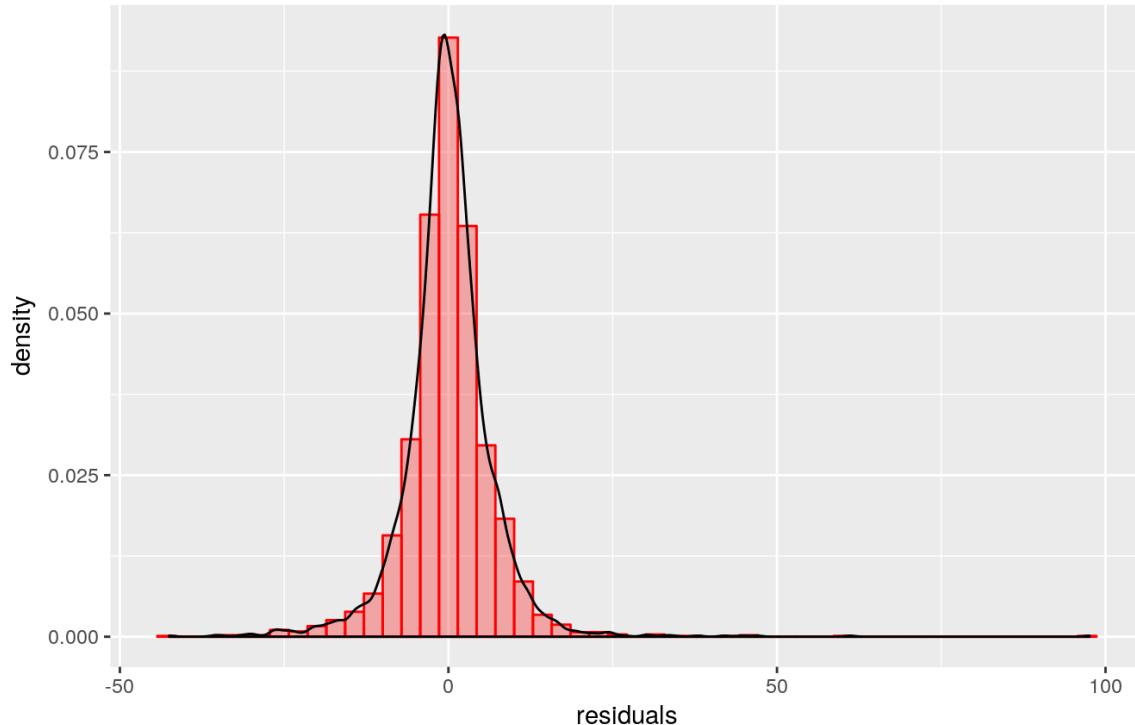


FIGURE 3.22 : Histogramme et densité des résiduels.

On remarque que la densité des résiduels semble à une densité Gaussienne centré en 0.

```
checkresiduals(tsforecasts30)
```

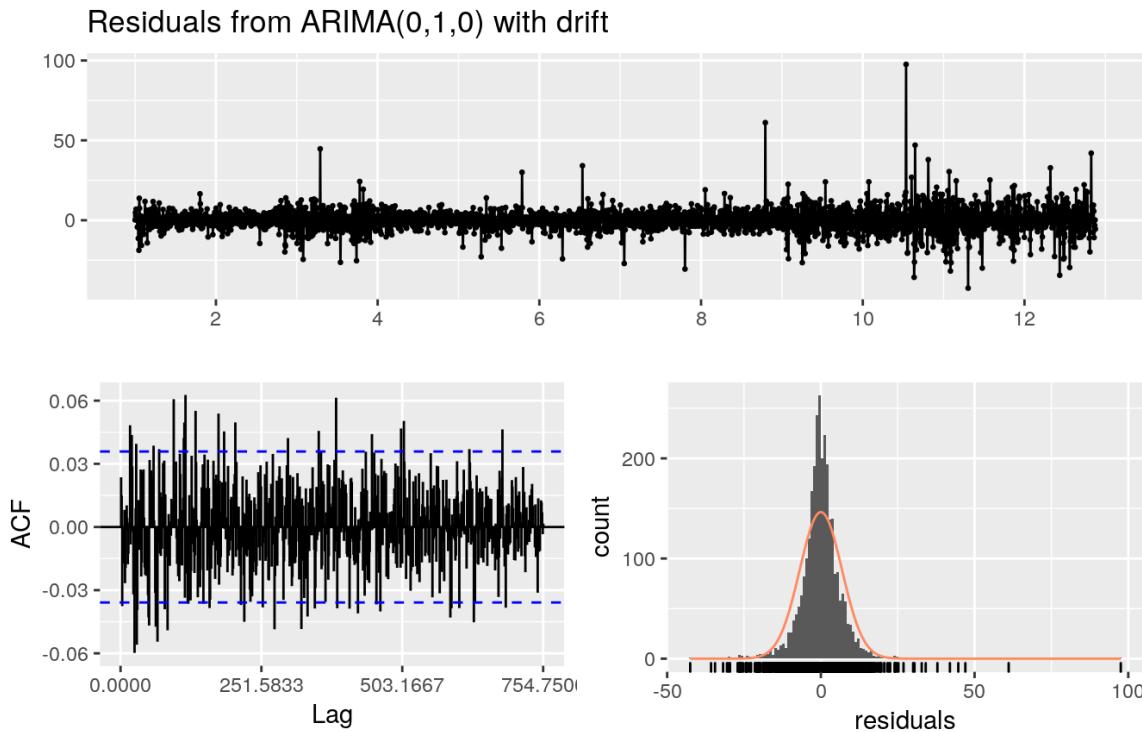


FIGURE 3.23 : Graphe des résiduels d'ARIMA(0,1,0)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

```
Ljung-Box test
data: Residuals from ARIMA(0,1,0) with drift
Q* = 744.82, df = 502.17, p-value = 1.002e-11
Model df: 1. Total lags used: 503.166666666667
```

FIGURE 3.24 : Test de Ljung-Box

p-value < 0,05 : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

Analyse de la prédition des 7 prochaines séries temporelles :

```
ggplot(data.frame(residuals = tsforecasts7$residuals), aes(residuals)) +
  geom_histogram(bins = 50, aes(y = ..density..), col = "red", fill = "red",
  alpha = 0.3) + geom_density()
```

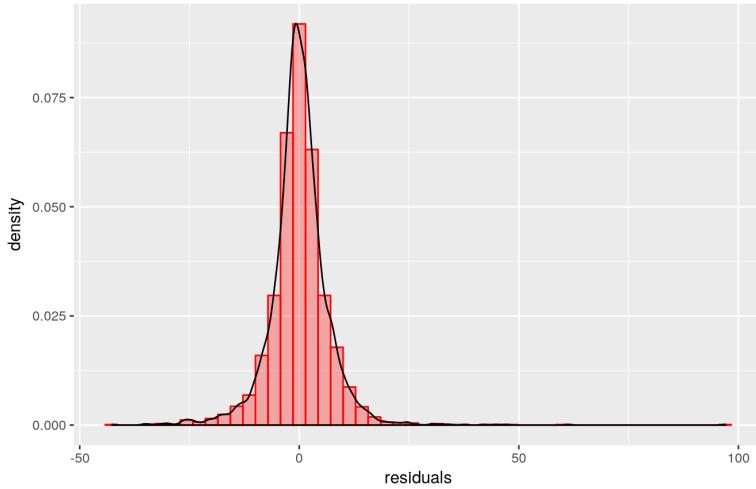


FIGURE 3.25 : Histogramme et densité des résiduels.

On remarque que la densité des résiduels semble à une densité Gaussienne centé en 0.

```
checkresiduals(tsfcasts7)
```

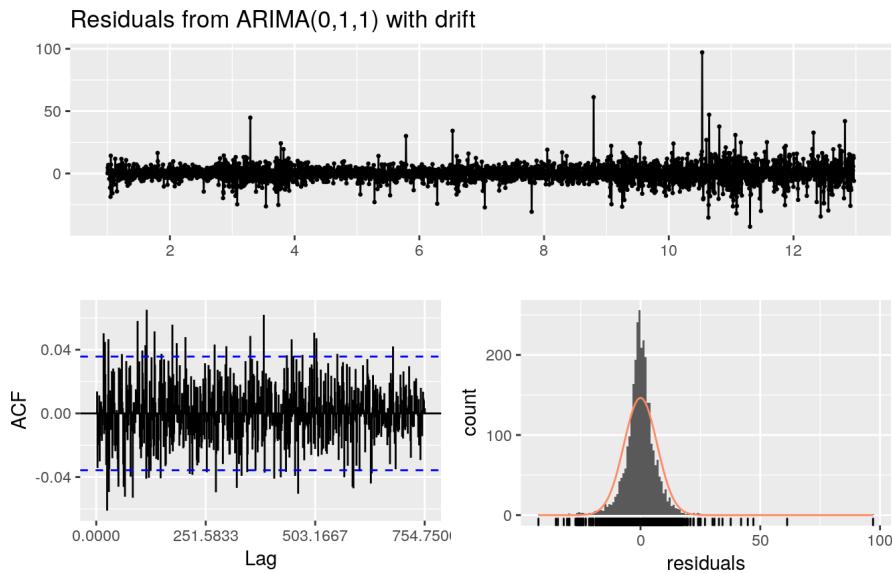


FIGURE 3.26 : Graphe des résiduels d'ARIMA(0,1,0)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

```
Ljung-Box test

data: Residuals from ARIMA(0,1,1) with drift
Q* = 747.93, df = 501.17, p-value = 4.856e-12

Model df: 2. Total lags used: 503.16666666667
```

FIGURE 3.27 : Test de Ljung-Box

p-value < 0,05 : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

3.8 Tester la fonction arima_modeling sur les autres entreprises

```
arima_modeling <- function(xts, ts, ticker){
  ## Stationnarite
  print(ticker)
  adf.test(xts, alternative = "stationary", k = 0)

  ## Decomposition des series temporelles
  tscomponents <- decompose(ts)
  plot(tscomponents, col = "red")

  ## Differencier une serie temporelle
  xtsdiff1 <- diff(xts, differences=1)
  tsdiff1 <- diff(ts, differences=1)
  plot.xts(xtsdiff1, col = "blue")
  # Trouver la frequence dominante de la serie chronologique d'origine
  findfrequency(xts)
  # Trouver la frequence dominante des series chronologiques differenciees
  findfrequency(xtsdiff1)

  ## Selection d'un modele ARIMA candidat
  print(ticker)
  print("Selecting a candidate ARIMA Model")
  # Tracer le correlogramme
  Acf(xtsdiff1, lag.max=60)
  # Obtenir les valeurs d'autocorrelation
  Acf(xtsdiff1, lag.max=60, plot=FALSE)

  # Tracer un correlogramme partiel
  Pacf(xtsdiff1, lag.max=60)
  # Obtenir les valeurs d'autocorrelation partielle
  Pacf(xtsdiff1, lag.max=60, plot=FALSE)

  ## Fitting le modele ARIMA
  tsarima <- auto.arima(head(xts, -30), max.p = 3, max.q = 3, max.d = 3)

  # En excluant les 120 dernieres series temporelles (donnees de test)
  print(tsarima)
  autoplot(tsarima)
```

```

print(ticker)

## Prevision a l'aide d'un modele ARIMA
print(ticker)
# Prevoir les 120 prochaines series temporelles
tsforecasts <- forecast(tsarima, h = 30)
acc <- accuracy(tsforecasts, head(tail(xts, 30), 7))
print(acc)
autoplot(tsforecasts)

print(ticker)

# Creer l'histogramme
ggplot(data.frame(residuals = tsforecasts$residuals), aes(residuals)) +
  geom_histogram(bins = 50, aes(y = ..density..), col = "red", fill = "red",
  alpha = 0.3) + geom_density()
checkresiduals(tsforecasts)
}

for (ticker in candidate_ticker){
  if (ticker != 'GOOGL'){
    arima_modeling(xts_list[[ticker]], ts_list[[ticker]], as.character(ticker))
  }
}
# [1] "IBM"

```

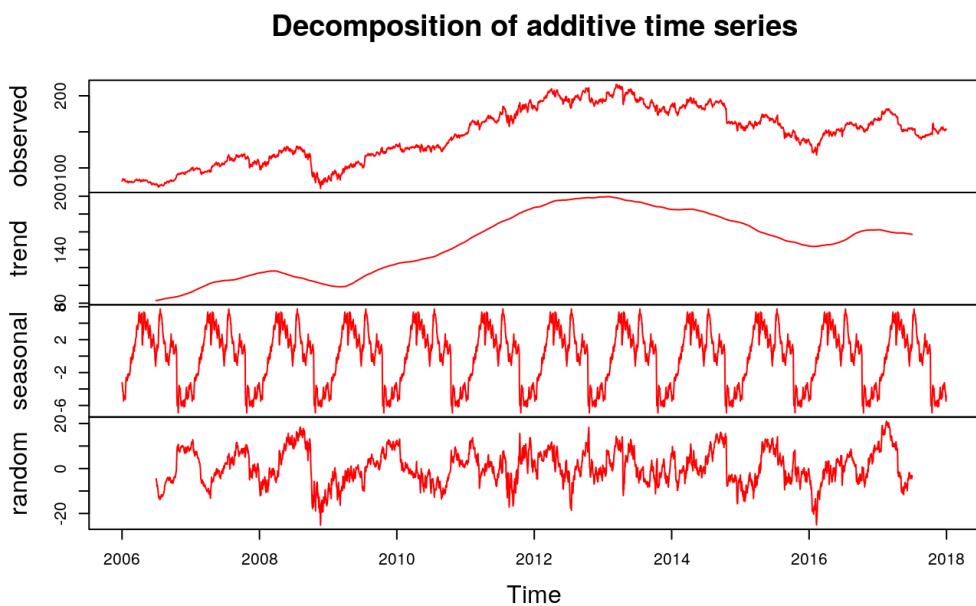


FIGURE 3.28 : Décomposition additive de la série temporelle.

```

# [1] "IBM"
# [1] "Selection d'un modele ARIMA candidat"

```

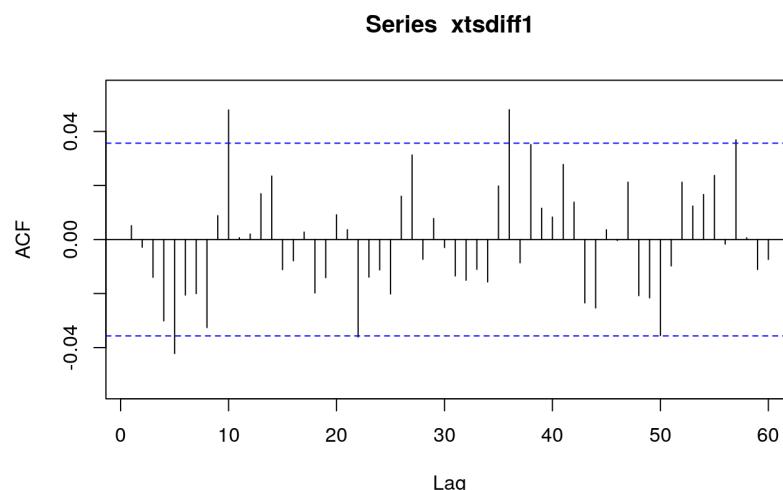


FIGURE 3.29 : Corrélogramme de la série temporelle.

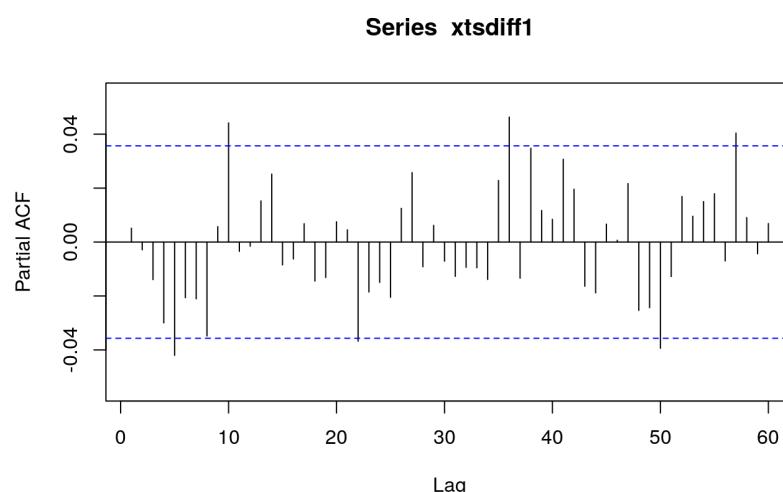


FIGURE 3.30 : Correlogramme partiel de la série temporelle.

```

# Series: head(xts, -30)
# ARIMA(0,1,0)
#
# sigma^2 estimated as 3.488: log likelihood=-6108.41
# AIC=12218.82    AICc=12218.82    BIC=12224.82
# [1] "IBM"
# [1] "IBM"
#               ME      RMSE      MAE      MPE      MAPE      MASE
# Training set 0.02177995 1.867356 1.307867 0.01033524 0.9398062 0.06090065
# Test set     3.77714286 3.978851 3.777143 2.49672142 2.4967214 0.17588217
#             ACF1
# Training set 0.005630018
# Test set     NA

```

```
# [1] "IBM"
```

```
Ljung-Box test

data: Residuals from ARIMA(0,1,0)
Q* = 550.98, df = 503.33, p-value = 0.06975

Model df: 0. Total lags used: 503.333333333333
```

FIGURE 3.31 : Ljung-Box test

p-value > 0,05 : On n'a pas suffisamment de preuves statistiques pour rejeter l'hypothèse nulle. On ne peut donc pas supposer que vos valeurs sont dépendantes. Cela peut signifier que nos valeurs dépendent de toute façon ou cela peut signifier que nos valeurs sont indépendantes. Mais on ne peut aucune possibilité spécifique, ce que notre test a réellement dit, c'est qu'on ne peut pas affirmer la dépendance des valeurs, ni affirmer l'indépendance des valeurs.

```
# [1] "BA"
```

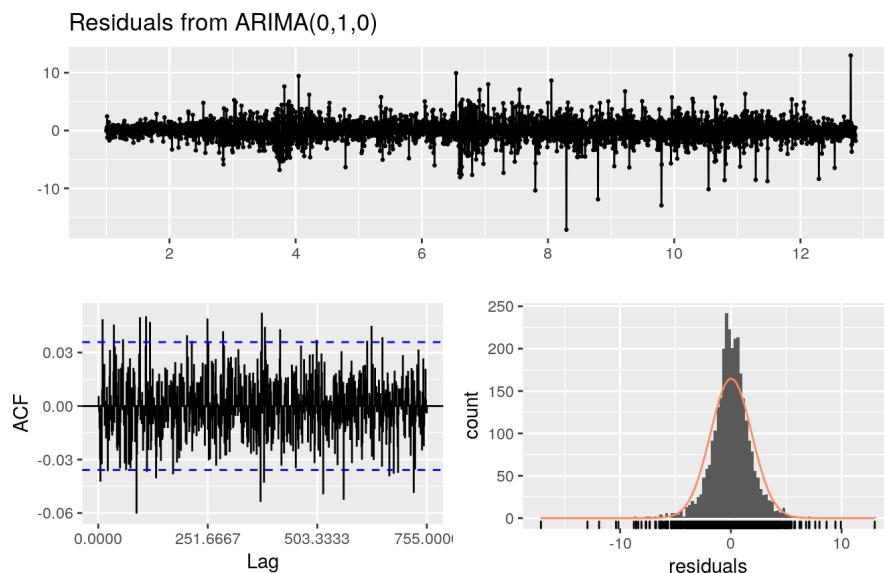


FIGURE 3.32 : Graphe des résiduels d'ARIMA(0,1,0)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

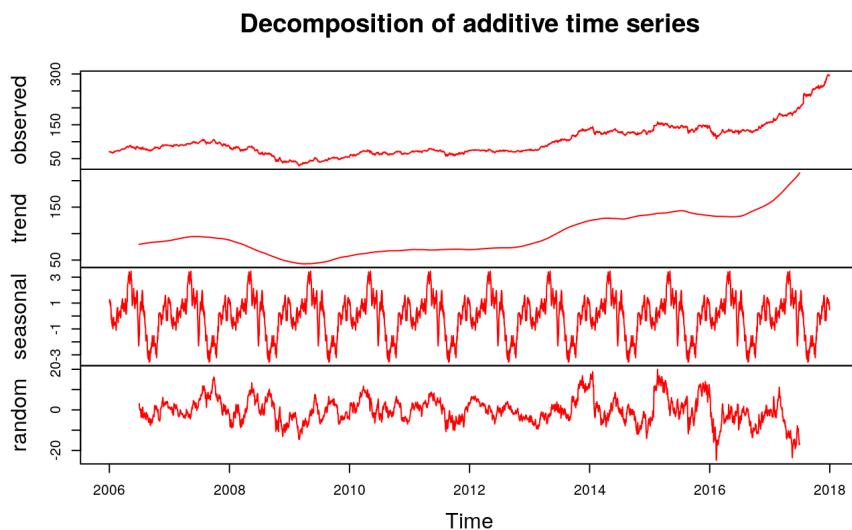


FIGURE 3.33 : Décomposition additive de la série temporelle.

```
# [1] "BA"
# [1] "Selection d'un modèle ARIMA candidat"
```

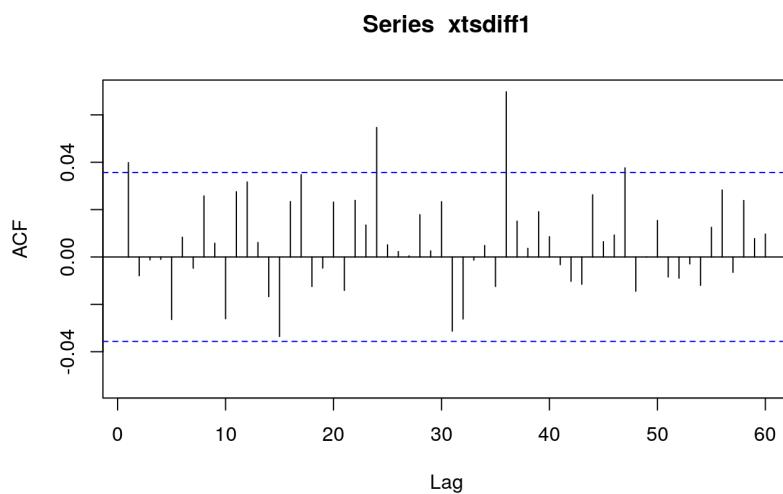


FIGURE 3.34 : Corrélogramme de la série temporelle.

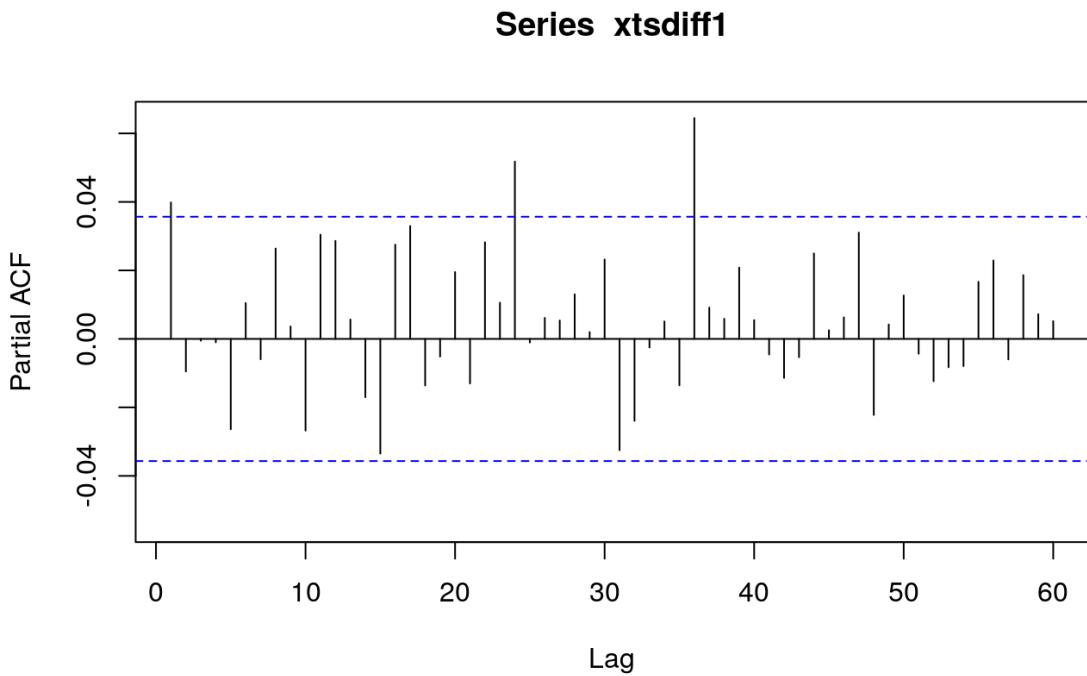


FIGURE 3.35 : Correlogramme partiel de la série temporelle.

```

# Series: head(xts, -30)
# ARIMA(1,2,0)
#
# Coefficients:
#       ar1
#     -0.4590
# s.e.   0.0163
#
# sigma^2 estimated as 3.502:  log likelihood=-6111.99
# AIC=12227.97    AICc=12227.97    BIC=12239.98
# [1] "BA"
# [1] "BA"
#               ME      RMSE      MAE      MPE      MAPE
# Training set -5.161552e-05 1.870486 1.3570057 -0.0002147785 1.5270628
# Test set      2.120379e-02 1.083634 0.8306223  0.0065797532 0.3135072
#           MASE      ACF1
# Training set 0.05804228 -0.1535589
# Test set      0.035552764        NA
# [1] "BA"

```

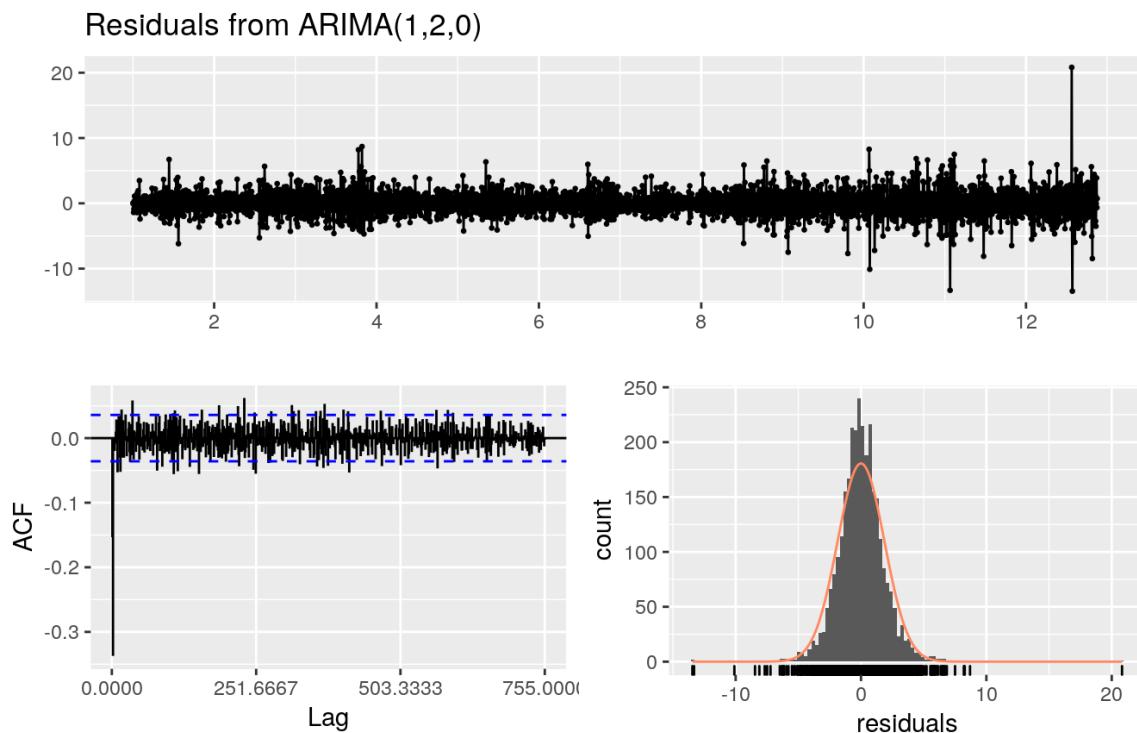


FIGURE 3.36 : Graphe des residuels d'ARIMA(1,2,0)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

```
Ljung-Box test

data: Residuals from ARIMA(1,2,0)
Q* = 1167.8, df = 502.33, p-value < 2.2e-16

Model df: 1. Total lags used: 503.333333333333
```

FIGURE 3.37 : Ljung-Box test

p-value < 0,05 : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

```
# [1] "AAPL"
```

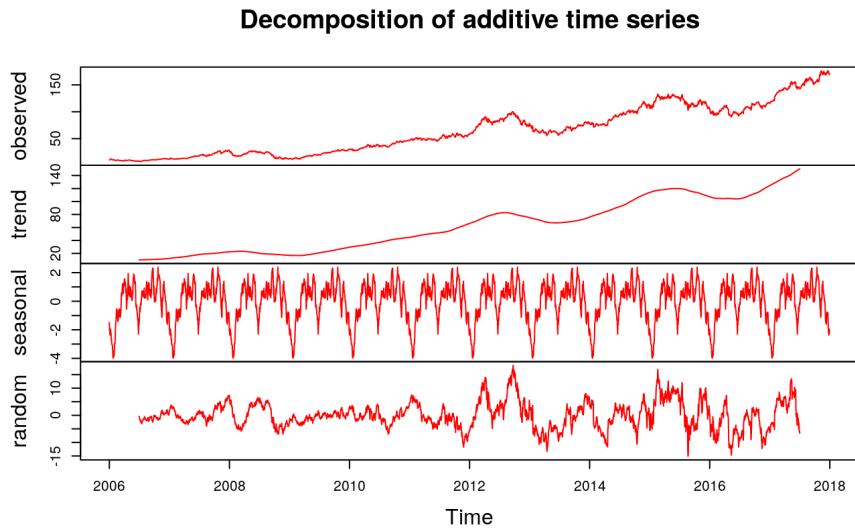


FIGURE 3.38 : Décomposition additive de la série temporelle.

```
# [1] "AAPL"
# [1] "Selection d'un modèle ARIMA candidat"
```

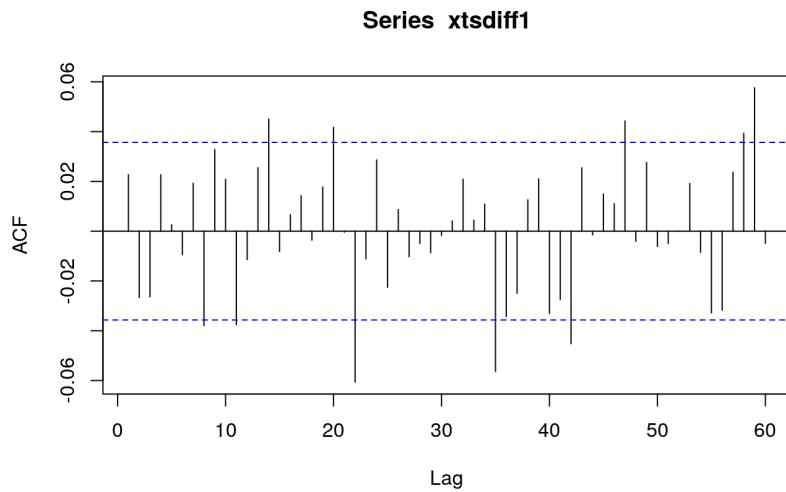


FIGURE 3.39 : Corrélogramme de la série temporelle.

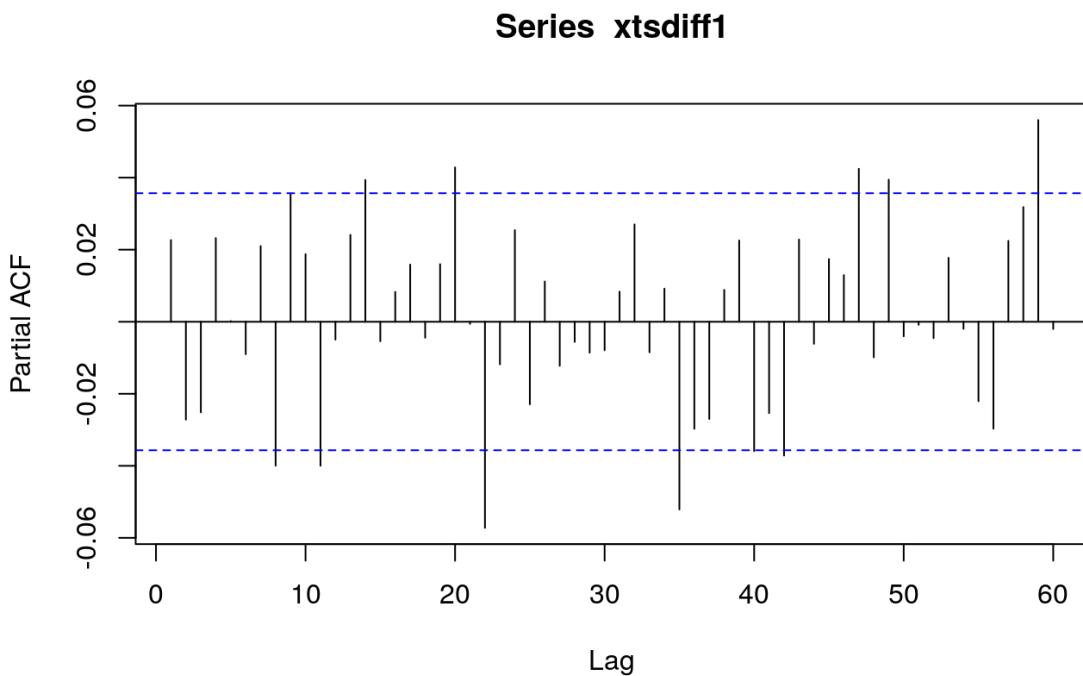


FIGURE 3.40 : Correlogramme partiel de la série temporelle.

```
# Series: head(xts, -30)
# ARIMA(0,1,1) with drift
#
# Coefficients:
#          ma1     drift
#        0.0271   0.0530
# s.e.  0.0187   0.0218
#
# sigma^2 estimated as 1.342:  log likelihood=-4678.4
# AIC=9362.8    AICc=9362.81    BIC=9380.81
# [1] "AAPL"
```

```
# [1] "AAPL"
#               ME      RMSE      MAE      MPE      MAPE
# Training set -1.426077e-05 1.157919 0.7466738 -0.0923504 1.43854
# Test set      3.395825e+00 3.911686 3.3958252  1.9541796 1.95418
#             MASE      ACF1
# Training set 0.03844376 -0.0005956725
# Test set     0.17483980           NA
# [1] "AAPL"
```

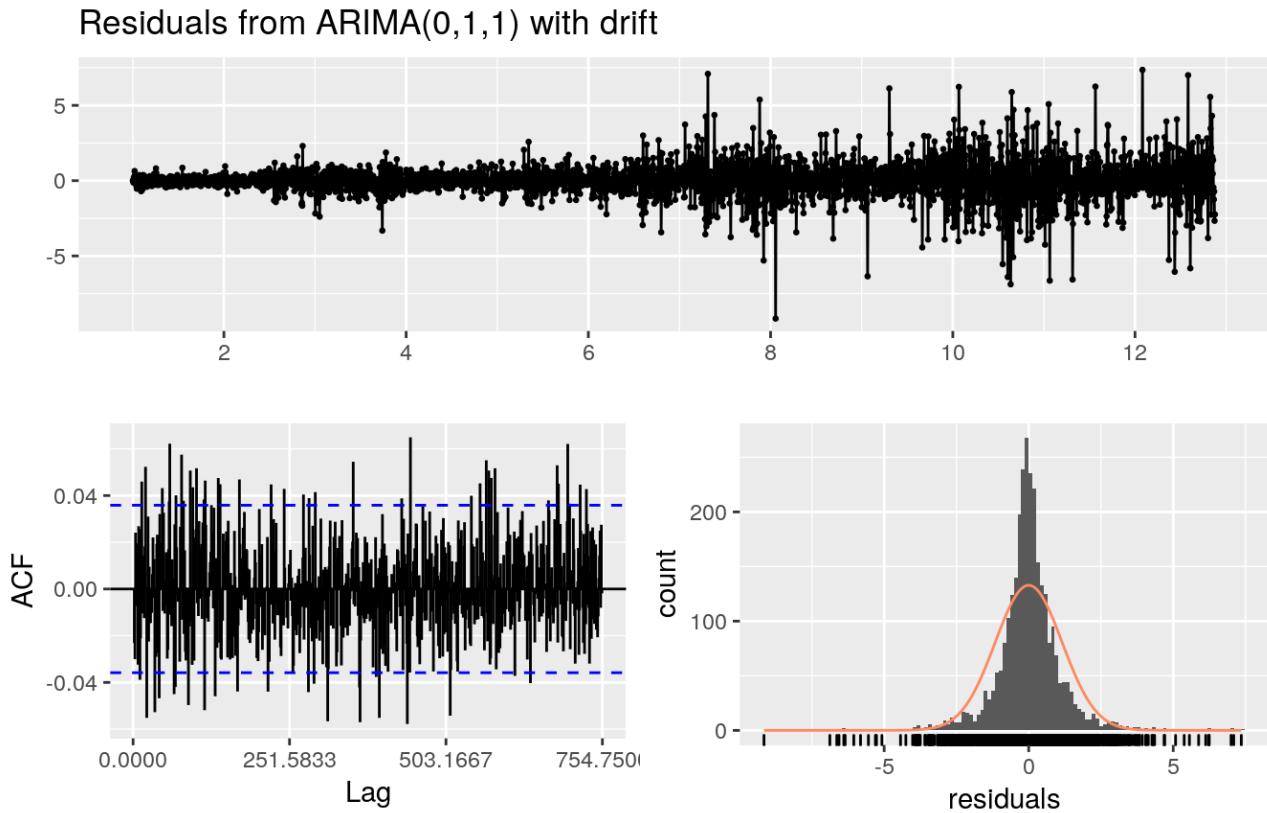


FIGURE 3.41 : Graphe des residuels d'ARIMA(0,1,1)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

```
Ljung-Box test

data: Residuals from ARIMA(0,1,1) with drift
Q* = 760.19, df = 501.17, p-value = 5.924e-13

Model df: 2. Total lags used: 503.166666666667
```

FIGURE 3.42 : Ljung-Box test

p-value < 0,05 : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

```
# [1] "GS"
```

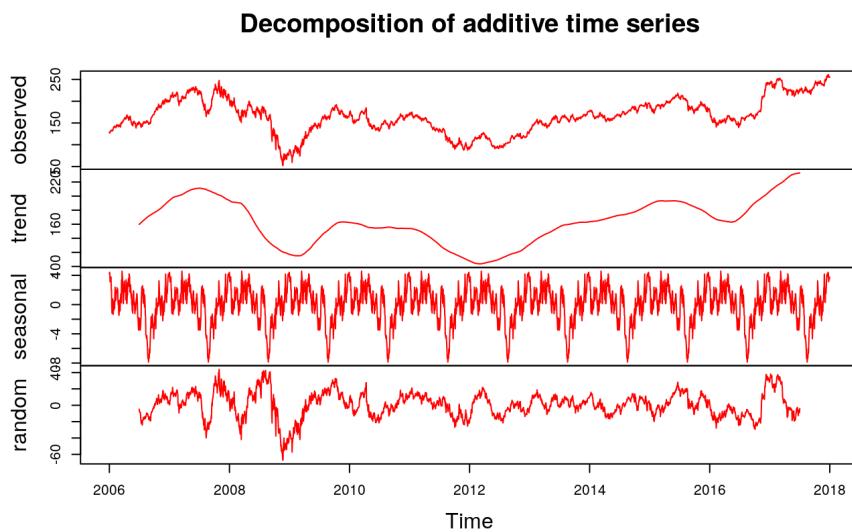


FIGURE 3.43 : Décomposition additive de la série temporelle.

```
# [1] "AAPL"
# [1] "Selection d'un modèle ARIMA candidat"
```

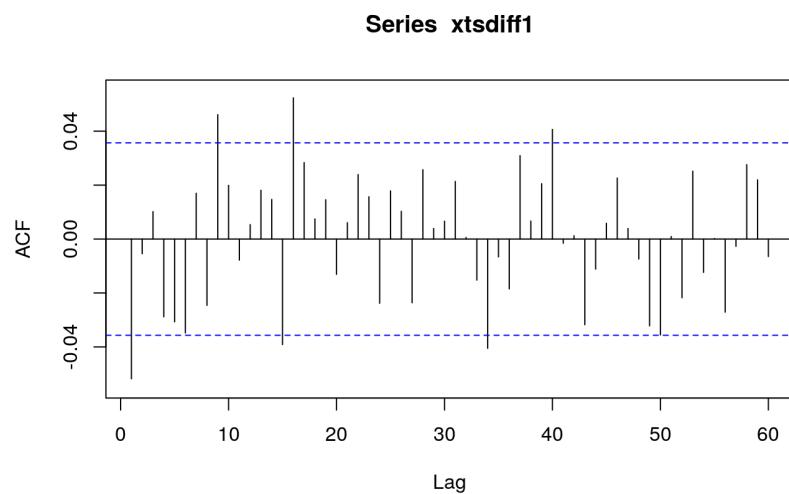


FIGURE 3.44 : Corrélogramme de la série temporelle.

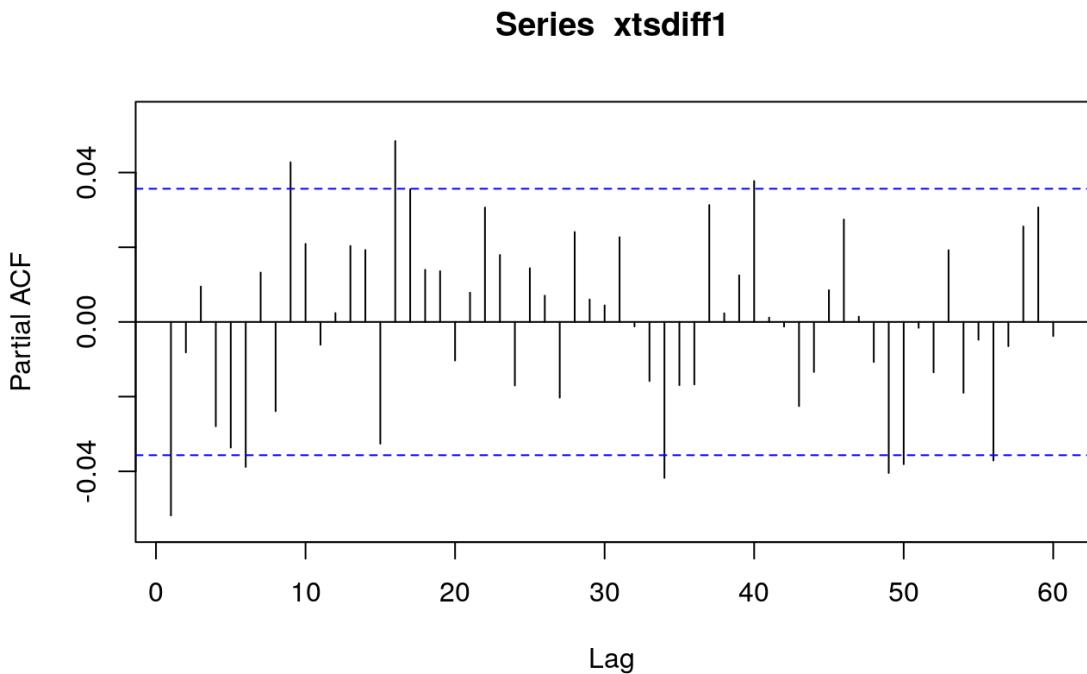


FIGURE 3.45 : Correlogramme partiel de la série temporelle.

```
# Series: head(xts, -30)
# ARIMA(1,1,0)
#
# Coefficients:
#       ar1
#     -0.0517
# s.e.  0.0183
#
# sigma^2 estimated as 10.82:  log likelihood=-7799.02
# AIC=15602.05  AICc=15602.05  BIC=15614.05
# [1] "GS"

# [1] "GS"
#               ME      RMSE      MAE      MPE      MAPE
# Training set  0.03828716 3.287527 2.295956 -0.008158638 1.5156666
# Test set      -0.30166919 1.415540 1.208443 -0.130535329 0.5102662
#             MASE      ACF1
# Training set 0.05571161 -0.0005618904
# Test set      0.02932299          NA
# [1] "GS"
```

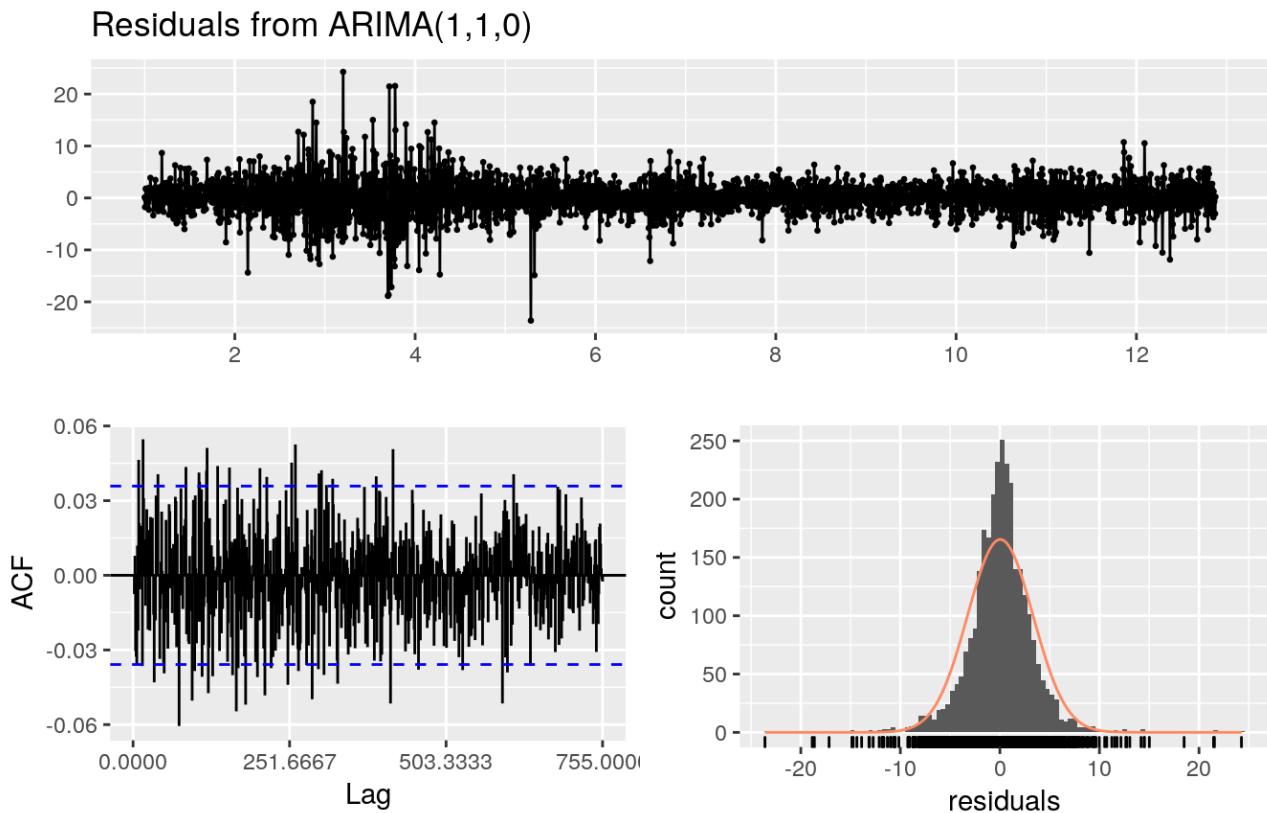


FIGURE 3.46 : Graphe des residuels d'ARIMA(1,1,0)

La moyenne des résidus est proche de zéro et il n'y a pas de corrélation significative dans la série des résidus. Le graphique temporel des résidus montre que la variation des résidus reste sensiblement la même à travers les données historiques, à l'exception d'une valeur aberrante, et donc la variance résiduelle peut être traitée comme constante. Cela peut également être vu sur l'histogramme des résidus. L'histogramme suggère que les résidus peuvent être normaux. Par conséquent, les prévisions de cette méthode seront probablement assez bonnes.

```
Ljung-Box test

data: Residuals from ARIMA(1,1,0)
Q* = 710.04, df = 502.33, p-value = 2.605e-09

Model df: 1.   Total lags used: 503.333333333333
```

FIGURE 3.47 : Ljung-Box test

p-value < 0,05 : On peut rejeter l'hypothèse nulle en supposant 5% de chances de faire une erreur. On peut donc supposer que nos valeurs montrent une dépendance les unes envers les autres.

Chapitre 4

Modèles ARCH et GARCH

4.1 Définitions : ARCH et GARCH

En économétrie, les modèles ARCH (AutoRegressive Conditional Heteroskedasticity) sont utilisés pour caractériser et modéliser des séries chronologiques. Ces modèles sont souvent appelés les modèles ARCH (Robert F. Engle, 1982), bien qu'une variété d'autres acronymes sont appliqués à des structures particulières du modèle qui ont une base similaire. Les modèles ARCH sont employés couramment dans la modélisation de séries temporelles financières, qui comportent des volatilités variables c'est-à-dire des périodes agitées suivies par des périodes de calme relatif. Dans ces modèles, la variance conditionnelle au temps t est variable. Elle dépend par exemple du carré des réalisations précédentes du processus ou du carré des innovations.

Ces acronymes un peu spéciaux sont en anglais et signifient pour l'ARCH : AutoRegressive Conditional Heteroskedacity et le GARCH pour Generalised ARCH. Il s'agit d'un type de modèle qui permet d'estimer et prévoir la volatilité du prix d'un action ou d'un rendement à court terme en se basant sur les valeurs que prennent ces prix ou rendements quelques périodes de temps auparavant. Le terme AutoRegressive signifie qu'on regresse le modèle à partir de lui-même, c'est-à-dire que les variables exogènes sont les retards de la variable endogène à l'ordre q . Le terme Heteroskedacity accentue le fait que la variance n'est constante dans le temps. Nous disons qu'il y a hétéroscédasticité si dans la série temporelle il y a une ou des sous-périodes dont la variance est différente de la variance des autres périodes. Dans les séries financières, souvent la variation de la variance est souvent causée par un événement particulier qui peut surgir sur le marché, donc si notre variance est hétéroscléastique, elle l'est conditionnellement aux interactions du marché d'où le terme conditional heteroskedacity.

La grande nouveauté qu'apporte les modèles de types ARCH est que non seulement il prend en compte la valeur de la variable à $N-p$ périodes mais aussi le changement dans la valeur de la variance à $N-p$, ce faisant il améliore grandement la prévision de la volatilité.

4.2 Principe du modèle ARCH

Le cas le plus simple des modèle ARCH est le modèle $ARCH_{(1)}$ qui ne prend en compte que le changement d'ordre 1 de la variance de la série temporelle. Formellement elle se présente sous cette forme :

$$\epsilon_t = w_t * \sqrt{\alpha_0 + \alpha_1 * \epsilon_{t-1}^2}$$

L'expression sous la racine représente la variance conditionnelle. Pour simplifier cette écriture on peut dire que le processus ARCH(1) s'écrit de la manière suivante :

$$\epsilon = \sigma \cdot w_t$$

où σ est l'expression obtenue plus haut.

4.3 Principe des modèles GARCH

Les modèles GARCH suivent le même principe que le modèle ARCH mais ajoute un second membre à l'équation qui est la moyenne mobile d'ordre q. Les modèles GARCH s'écrivent de manière globale sous cette forme :

$$\begin{aligned}\epsilon_t^2 &= \sum w_t * \sqrt{\alpha_0 + \alpha_1 * \epsilon_{t-1}^2} \\ \epsilon_t^2 &= \alpha_0 + \sum_{i=1}^q \alpha_i \sigma_{t_i}^2 + \sum_{i=1}^p \beta_i \sigma_{t_i}^2\end{aligned}$$

4.4 Application

Dans cette partie, on va analyser la volatilité des actifs financier (le prix de l'action) de l'entreprise américaine IBM. on va lire les données sur une période de 15 ans et on va appliquer différentes familles de modèles GARCH sur les données dans le but de voir si on peut réellement prévoir la volatilité du cours de l'action avec le temps.

4.4.1 Le package quantmod

Pour cette partie, on va utiliser l'excellent package quantmod pour lire directement les informations financières concernant IBM dont on a besoin.

NB : Il faut installer le package "tidyverse" via la commande :

```
install.packages("tidyverse")

# Chargement les librairies
library(tidyverse)
library(quantmod)
```

4.4.2 Lecture des informations

```
getSymbols(Symbols = "IBM", src = "yahoo",
           from = as.Date("2004-01-01"), to = as.Date("2019-03-01"))
```

On vient de lire les données d'IBM directement à partir de Yahoo Finance pour la période du 1er janvier 2004 au 1er mars 2019. En lisant ces données, R crée automatiquement un objet IBM dans notre environnement de travail. On peut voir quel type d'objet :

```
class(IBM)

#[1] "xts" "zoo"
```

L'objet hérite de deux classes **xts** et **zoo** qui sont des classes pour séries temporelles. Le package **quantmod** vient avec une série de méthodes (fonctions) qui pour analyser automatiquement les objets de ces classes. Nous allons les explorer, mais avant, voyons comment la base de donnée se présente.

```
head(IBM)

#           IBM.Open IBM.High IBM.Low IBM.Close IBM.Volume IBM.Adjusted
# 2004-01-02     92.86    93.05   91.20     91.55    5327800    64.01743
# 2004-01-05     92.00    93.09   92.00     93.05    5276300    65.06632
# # 2004-01-06     92.20    93.19   92.14     93.06    4380000    65.07331
# 2004-01-07     93.14    93.38   92.47     92.78    4927600    64.87749
# 2004-01-08     93.21    93.21   92.03     93.04    6179800    65.05934
# 2004-01-09     91.75    92.35   91.00     91.21    7930900    63.77967
```

La base de donné commence à partir du 2 janvier 2004 alors qu'on avait demandé à partir du 1er. Des raisons particulières ? Oui c'est le nouvel an, et ce n'était pas un business day. On a les mêmes observations entre les 2 et 5 de cette même année.

```
dim(IBM)
```

```
# [1] 3815      6
```

On a 3815 observations et 6 variables.

4.4.3 Visualisation

La meilleure manière de comprendre la tendance d'une série financière c'est de l'observer graphiquement, nous allons dans cette partie visualiser la série sur toute la période considérée et dégager une première impression.

```
# parametres graphiques, res pour resolution
options(repr.plot.res = 300, repr.plot.height = 3)

plot(IBM[, "IBM.Open"], main = "Prix d'ouverture de l'action IBM",
col = "lightblue")
```

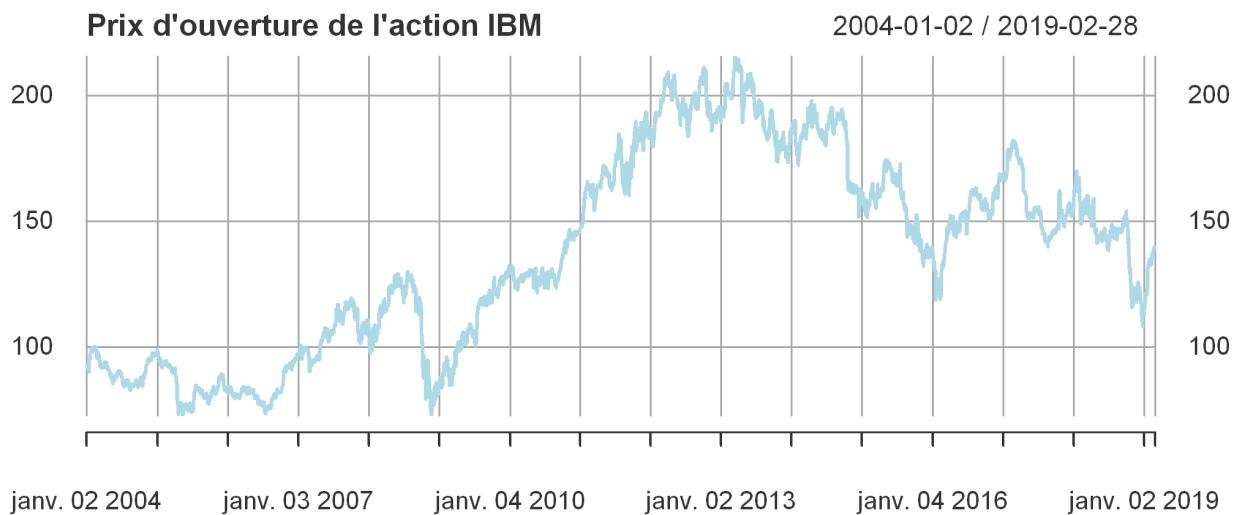


FIGURE 4.1 : Graphe du prix d'ouverture de l'action IBM

On a pris l'avantage des classes `xts` et `zoo` pour facilement visualiser cette série temporelle avec la fonction générique `plot()`. Sur la période considérée on observe une nette augmentation du prix d'ouverture de l'action IBM.

Ce prix est à son plus bas niveau entre 2008 et 2009, on peut tout de suite voir que la crise des subrpimes a quelque chose à y voir. Les prix des actions sont très volatiles car ils dépendent TOTALEMENT du comportement des acteurs qui interviennent sur le marché financier.

On peut également représenter toutes les prix sur un même graphique pour voir comment ils évoluent. Y a-t-il des différences entre les prix d'ouverture, de fermeture...

```
# parametres graphiques, res pour resolution
options(repr.plot.res = 300, repr.plot.height = 4.4)

plot.xts(IBM[,1:4],legend.loc = "left", main = "Prix de l'action IBM",
col = rainbow(4))
```



FIGURE 4.2 : Graphe du prix de l'action IBM

La plupart du temps ces valeurs se superposent, et la tendance générale est la même pour tous les différents prix. Il y a toutefois une nette différence entre les prix haut et bas. Regardons les dans les détails :

```
# parametres graphiques, res pour resolution
options(repr.plot.res = 300, repr.plot.height = 4.4)
plot.xts(IBM[,2:3],legend.loc = "left", main = "Prix plus haut et plus bas",
col = rainbow(n= 2))
```

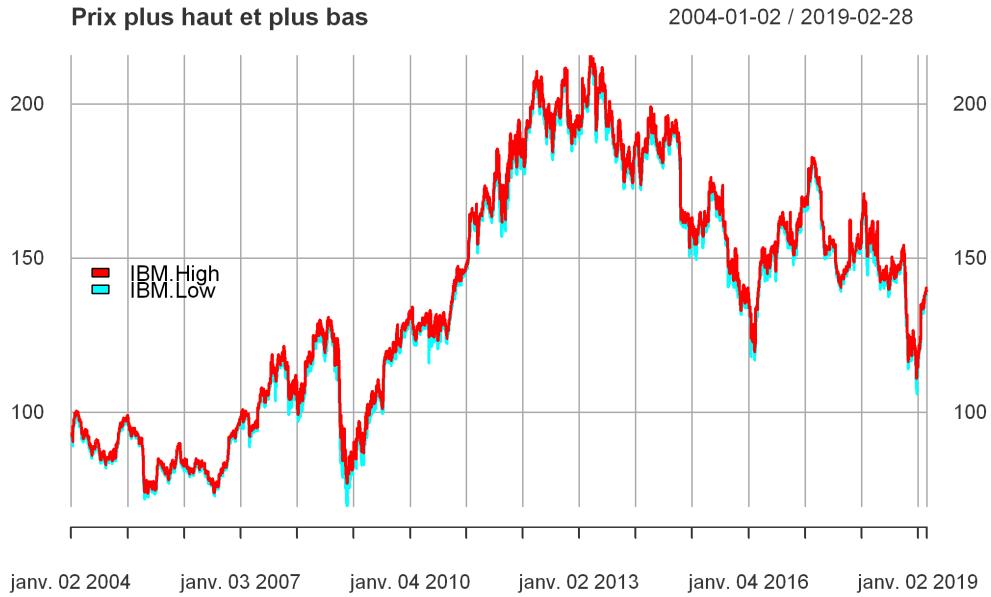


FIGURE 4.3 : Graphe du prix plus haut et plus bas

Intéressons-nous maintenant au volume :

```
plot(IBM[, "IBM.Volume"]/1000000, col = "lightblue",
main = "Volume d'actions échangées en millions")
```

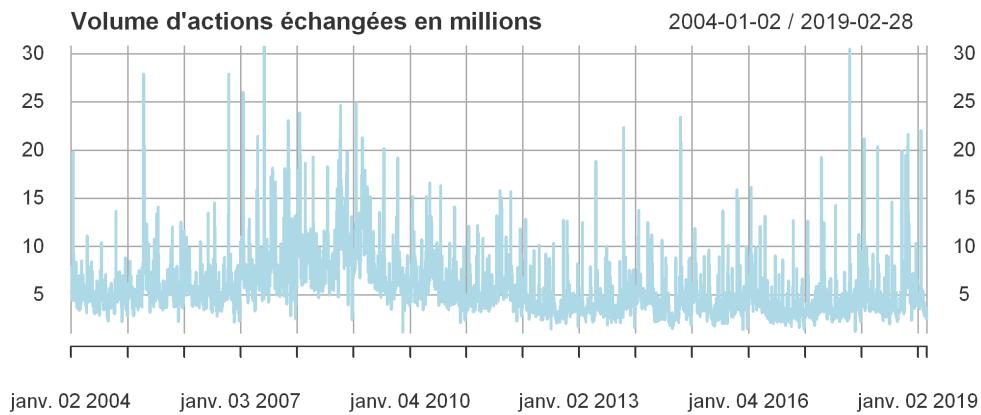


FIGURE 4.4 : Graphe de Volume d'actions échangées en millions

Les volumes échangés fluctuent les plus. Il n'y a pas une tendance à la hausse comme on a pu l'observer pour les prix. Le volume échangé est plus aléatoire. La librairie quantmod a une série de fonctions pour représenter les visuellement les graphiques dont candleChart.

```
# paramètres graphiques, res pour résolution
options(repr.plot.res = 300, repr.plot.height = 5)
candleChart(IBM, type = "line", theme = "white", show.grid = T)
```

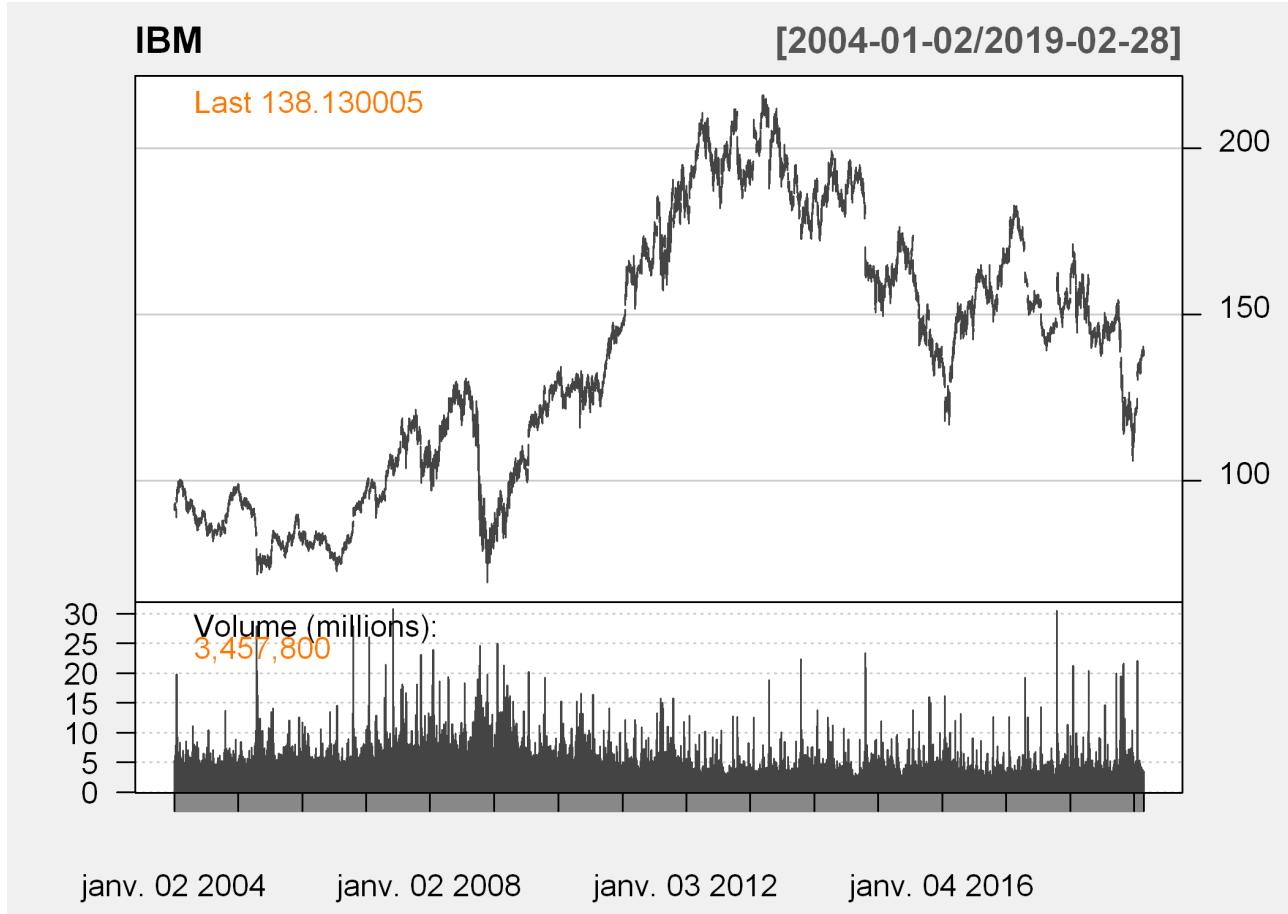


FIGURE 4.5 : Les graphiques de candleChart

4.4.4 Modélisations

Il existe de nombreux packages pour faire la modélisation GARCH avec R.

On fera la modélisation avec les packages **rugarch** et **rmgarch** respectivement pour Univariate GARCH et Multivariate GARCH.

```
# Installons les packages
install.packages(c("rugarch", "rmgarch"))

library(rugarch)
library(rmgarch)
```

4.4.5 GARCH Univarié

Nous n'allons utiliser que le GARCH univarié car nous n'avons pas d'exogènes à proprement parler à inclure dans le modèle. Pour un GARCH multivarié il faudra lire les données de plusieurs entreprises en même temps.

```
# On commence par instancier l'objet ugarch sans y ajouter de paramètres
univ_garch = ugarchspec()

univ_garch
```

```

# -----
#      GARCH Model Spec
# -----
#
# Conditional Variance Dynamics
# -----
# GARCH Model           : sGARCH(1,1)
# Variance Targeting   : FALSE
#
# Conditional Mean Dynamics
# -----
# Mean Model            : ARFIMA(1,0,1)
# Include Mean          : TRUE
# GARCH-in-Mean         : FALSE
#
# Conditional Distribution
# -----
# Distribution : norm
# Includes Skew : FALSE
# Includes Shape   : FALSE
# Includes Lambda  : FALSE

```

Ce sont les configurations par défaut. sGARCH signifie standard GARCH c'est le GARCH simple car il existe une grande variété de modèles GARCH. Faisons une première estimation avec ces spécifications.

```
mod <- ugarchfit(spec = univ_garch, data = IBM[, "IBM.Open"])
```

On peut afficher les résultats directement en exécutant l'objet mod mais les résultats sont très nombreux et risquent de prendre de la place sur l'écran.

Donc nous n'allons sélectionner que ce qui nous intéresse dans un modèle GARCH, c'est-à-dire la variance conditionnelle σ^2 .

L'objet mod qu'on a créé contient de familles de résultats : les résultats du modèle et les résultats d'estimation. Pour accéder à l'un ou l'autre de ces résultats on utilise @ après mod.

```

# affichons les noms des informations contenues dans ces deux objets
print("Objets contenus dans model@fit")
names(mod@fit)
print("Objets contenus dans model@model")
names(mod@model)

> print("objets contenus dans model@fit")
[1] "objets contenus dans model@fit"
> names(mod@fit)
[1] "hessian"        "cvar"           "var"             "sigma"          "condH"          "z"
[7] "LLH"            "log.likelihoods" "residuals"       "coef"           "robust.cvar"   "A"
[13] "B"              "scores"          "se.coef"         "tval"           "matcoef"       "robust.se.coef"
[19] "robust.tval"    "robust.matcoef"  "fitted.values"  "convergence"    "kappa"          "persistence"
[25] "timer"          "ipars"           "solver"          ""               ""               ""
> names(mod@model)
[1] "modelinc"      "modeldesc"     "modeldata"      "pars"           "start.pars"    "fixed.pars"
[6] "maxOrder"      "pos.matrix"    "fmodel"         "n.start"       "pidx"          ""
>

# Regardons les coefficients d'estimation
mod@fit$matcoef

```

	Estimate	Std. Error	t value	Pr(> t)
mu	92.871630280	1.375150510	67.535611	0.000000e+00
ar1	0.999745478	0.000533952	1872.351017	0.000000e+00
ma1	-0.024373350	0.016509271	-1.476343	1.398518e-01
omega	0.007360053	0.001237242	5.948758	2.701844e-09
alpha1	0.024024398	0.001488674	16.138116	0.000000e+00
beta1	0.974604561	0.002044855	476.612995	0.000000e+00

Nous avons dans ce tableau se trouvent les informations de l'estimation. Dans la spécification du modèle nous avons choisi un modèle GARCH simple garch(p=1,q=1) c'est ce que représentent les α_1 et β_1 dont on a les valeurs. On peut présenter ce modèle comme cela :

$$\mu + AR1 + MA1 + w + \alpha_1 + \beta_1$$

4.4.6 Variance conditionnelle estimée

Nous allons créer un data frame dans lesquel nous sauvons ces résultats pour les visualiser :

```
# variance
# 2004-01-02 2.725968
# 2004-01-05 2.663449
# 2004-01-06 2.618766
# 2004-01-07 2.559674
# 2004-01-08 2.520958
# 2004-01-09 2.463838

plot(variance, main = "variance conditionnelle estimée", col = "lightblue")
```

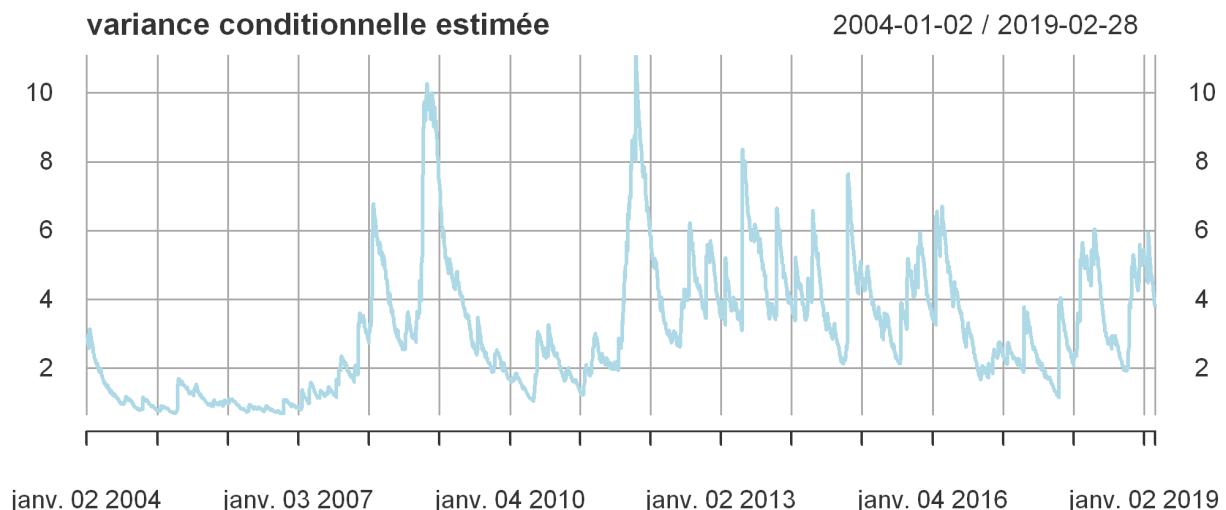


FIGURE 4.6 : Graphe de variance conditionnelle estimée

4.4.7 Prévisions sur le passé

Nous allons comparer les performances de notre modèle à prédire les valeurs du passé. Nous construirons un graphique sur lequel les deux séries se superposent pour observer les différences :

```
prev <- cbind(c(IBM[, "IBM.Open"]), mod@fit$fitted.values)
names(prev) <- c("valeur.observee", "valeur.predite")
head(prev)

#           valeur.observee valeur.predite
# 2004-01-02      88.77629     88.78035
# 2004-01-05      87.95411     88.77639
# 2004-01-06      88.14532     87.97430
# 2004-01-07      89.04398     88.14132
# 2004-01-08      89.11090     89.02198
# 2004-01-09      87.71510     89.10866

plot.xts(prev, col = c("lightblue", "red"), legend.loc= "left", lwd = 1)
```



FIGURE 4.7 : Graphe de la valeur observée et la valeur prédictée

La performance globale du modèle est très bonne pour la prévision dans le passé. Les modèles GARCH sont généralement très bons lorsque la période considérée est relativement courte.

Pour avoir une idée plus claire de cette différence, nous pouvons représenter graphiquement les résidus et le carré de ces résidus.

```
# paramètres graphiques, res pour resolution
options(repr.plot.res = 300, repr.plot.height = 3)
resid <- xts(mod@fit$residuals, order.by = as.Date(index(IBM)))
plot.xts(resid, main = "Résidus du modèle", col = "lightblue")
resid2 <- xts(mod@fit$residuals^2, order.by = as.Date(index(IBM)))
plot.xts(resid2, main = "Carré des résidus du modèle", col = "lightblue")
```



FIGURE 4.8 : Graphe des résidus du modèle

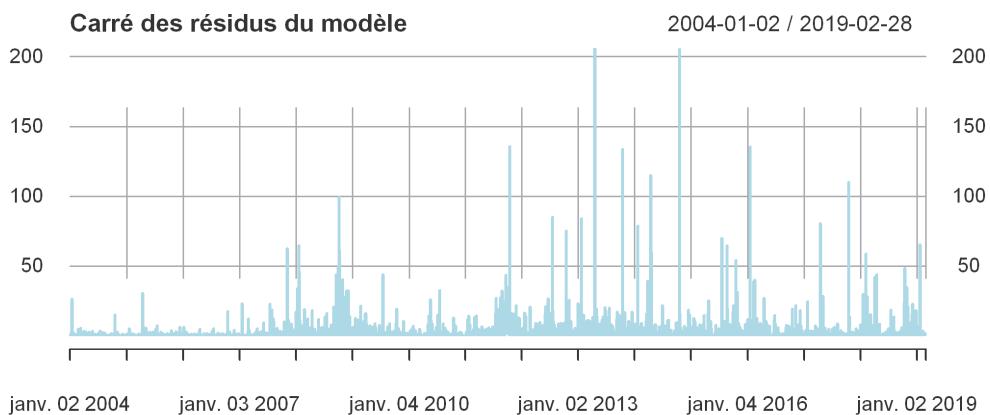


FIGURE 4.9 : Graphe des carré des résidus du modèle

4.4.8 Les rendements

Le calcul du rendement est simple, c'est la variation du prix de fermeture de l'action sur la période considérée. la formule est la suivante :

$$R_t = \frac{\text{prix}_t - \text{prix}_{t-1}}{\text{prix}_{t-1}}$$

Généralement on calcule le rendement journalier. quantmod intègre une série de fonctions qui nous permettent de calculer les rendements dont dailyReturn qui calcule le rendement journalier.

```
rIBM <- dailyReturn(IBM)
head(rIBM)

#           daily.returns
# 2004-01-02 -0.0141072350
# 2004-01-05  0.0163844270
# 2004-01-06  0.0001075452
# 2004-01-07 -0.0030088853
# 2004-01-08  0.0028023286
# 2004-01-09 -0.0196689266

# parametres graphiques, res pour resolution
options(repr.plot.res = 300, repr.plot.height = 3)
plot(rIBM, main = "Rendement journalier de l'action IBM",
col = "lightblue")
```

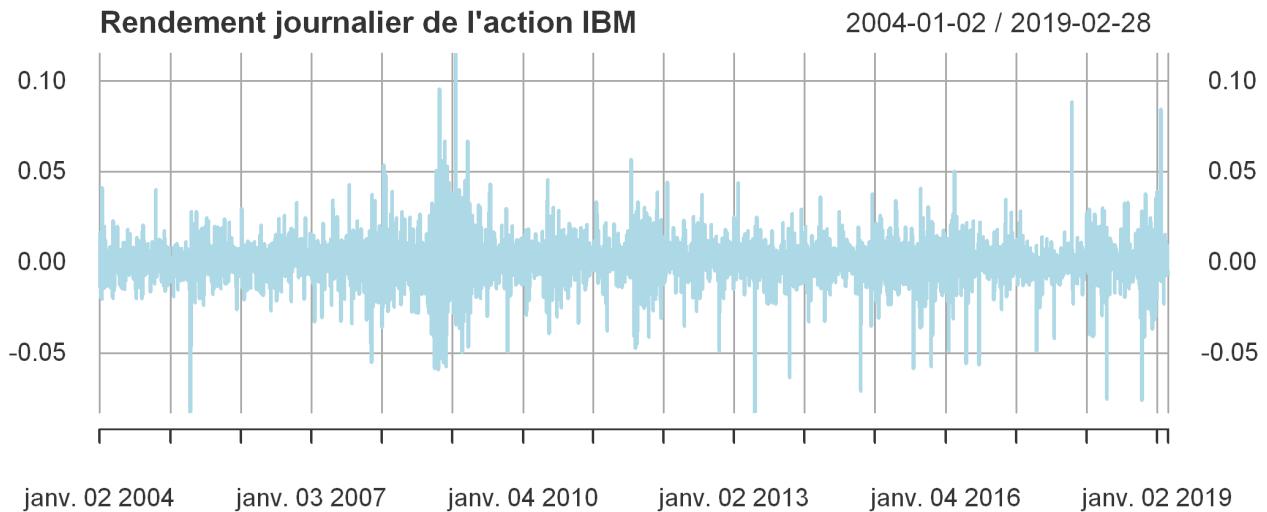


FIGURE 4.10 : Graphe du rendement journalier de l'action IBM

4.4.9 Modélisation du rendement

Pour le rendement, on choisit les spécifications suivantes : on utilise un "eGARCH", ce choix est juste aléatoire, couplé avec un retard d'ordre 3 et d'une moyenne mobile 3 aussi. La moyenne est une moyenne ARCH-in-mean.

```
return_garch <- ugarchspec(variance.model = list(model = "eGARCH",
garchOrder = c(3,3)), mean.model = list(archm = 3))

fit_return <- ugarchfit(spec = return_garch, data = rIBM)
```

4.4.10 Résultats

```
fit_return@fit$matcoef
```

	Estimate	Std. Error	t value	Pr(> t)
mu	0.0003373178	0.0002535284	1.3304929	1.833559e-01
ar1	-0.2801495873	0.0209793094	-13.3536134	0.000000e+00
ma1	0.2575985543	0.0213733469	12.0523265	0.000000e+00
archm	-0.0056631658	0.0277292577	-0.2042307	8.381732e-01
omega	-0.9573623475	0.1969173808	-4.8617463	1.163547e-06
alpha1	-0.0599571073	0.0196478443	-3.0515870	2.276350e-03
alpha2	-0.0876103632	0.0200375304	-4.3723134	1.229369e-05
alpha3	0.0109273856	0.0209578236	0.5213989	6.020889e-01
beta1	0.1315229696	0.0920725229	1.4284714	1.531562e-01
beta2	0.2711635420	0.0662792739	4.0912268	4.290972e-05
beta3	0.4858771558	0.0371937507	13.0634084	0.000000e+00
gamma1	0.2183713118	0.0307523642	7.1009601	1.239009e-12
gamma2	0.1653053408	0.0385267970	4.2906588	1.781438e-05
gamma3	-0.0259919728	0.0309438681	-0.8399717	4.009243e-01

4.4.11 Variance conditionnelle

```
return_var <- xts(fit_return@fit$var, order.by = as.Date(index(rIBM)))
plot(return_var, main = "Variance conditionnelle du rendement",
col = "lightblue")
```

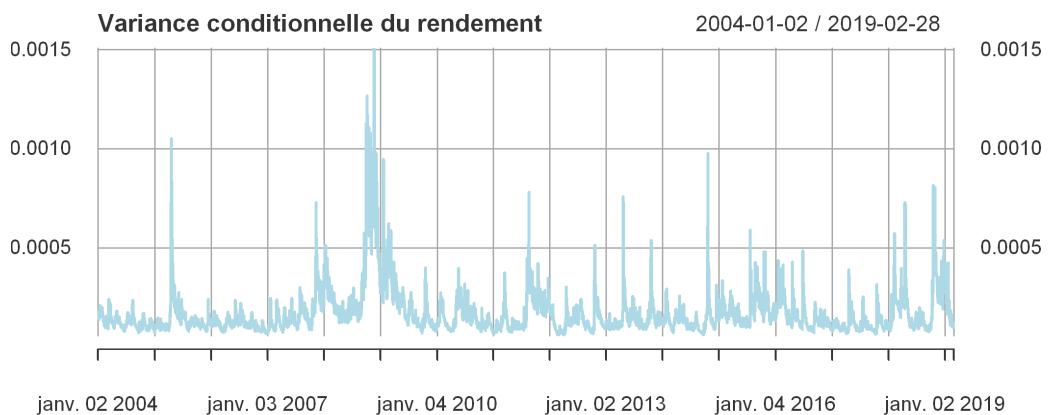


FIGURE 4.11 : Graphe de la variance conditionnelle du rendement

4.4.12 Prévision sur le futur

Nous pouvons faire une prévision des prix futurs avec la fonction **ugarchforecast** :

```
ugarchforecast(fitOrspec = mod, n.ahead = 10)
```

```
# -----
#          GARCH Model Forecast
# -----
# Model: sGARCH
# Horizon: 10
# Roll Steps: 0
# Out of Sample: 0
#
# 0-roll forecast [T0=2019-02-28]:
#      Series Sigma
# T+1    132.7 1.835
# T+2    132.7 1.835
# T+3    132.6 1.836
# T+4    132.6 1.836
# T+5    132.6 1.837
# T+6    132.6 1.838
# T+7    132.6 1.838
# T+8    132.6 1.839
# T+9    132.6 1.839
# T+10   132.6 1.840

mean(IBM[, "IBM.Open"])

# [1] 131.5081
```

Nous venons de faire une prévision sur 10 jours futurs.

Conclusion

L'analyse de certaines séries chronologiques émanant du monde économique et financier (taux de change, taux de rendement d'action, indices,...) montre des caractéristiques spécifiques qui ne sont pas théoriquement prises en compte dans les modélisations ARIMA notamment celle de Box Jenkins.

La première caractéristique concernée est la dépendance de la variance conditionnelle du temps, autrement dit le comportement hétéroscléastique de la variance (en plus, les périodes de grande variabilité sont groupées, tandis que les petites variations sont plutôt suivies par de petits changements).

La deuxième caractéristique concerne la distribution probabiliste de la série (L'apparition de chocs n'est pas compatible avec la loi normale). Alors pour une modélisation plus réaliste de ces séries, les modèles ARCH/GARCH sont les plus utiles.

Bibliographie

- [1] Billingsley, P. (2012). Probability and measure. John Wiley & Sons.
- [2] Borel, É. (1967). Les probabilités et la vie. PUF, 6th edition.
- [3] Brockwell, P. and Davis, R. (1991). Time series : theory and methods ("Jaune"). Springer, 2nd edition edition.
- [4] Brockwell, P. and Davis, R. (2002). Introduction to time series and forecasting ("Rouge"). Springer, 2nd edition edition.
- [5] Cowpertwait (2009). Introductory Time Series with R.
- [6] Hogg, R. V., Tanis, E. A., and Zimmerman, D. L. (2015). Probability and statistical inference. Pearson, 9 edition.
- [7] Le Gall, J.-F. (2006). Intégration, probabilités et processus aléatoires. Notes de cours.
- [8] Durrett, R. (2019). Probability : theory and examples, volume 49. Cambridge university press.
- [9] Shumway, R. H. and Stoffer, D. S. (2010). Time series analysis and its applications : with R examples. Springer, 3rd edition edition. (Si vous avez des difficultés à récupérer la troisième édition, vous aurez peut-être plus de chance avec la 2e édition).
- [10] https://fr.wikipedia.org/wiki/Série_temporelle
- [11] <https://datascientest.com/series-temporelles>
- [12] <http://perso.ens-lyon.fr/lise.vaudor/autocorrelation-de-series-temporelles-ou-spatiales>
- [13] <https://otexts.com/fpp2/arima-r.html>
- [13] https://fr.wikipedia.org/wiki/Modèles_ARCH

Annexe : Le test de Durbin-Watson

.1 Autocorrelation des résidus

Pour garantir la fiabilité d'un modèle de régression linéaire il y a un certain nombre d'hypothèses que le modèle doit vérifier. Une de ces hypothèses concerne la statistique des termes d'erreur qu'on note souvent epsilon ϵ et stipule que chaque erreur doit être identique et indépendamment distribuée (IID) et doit suivre une loi normale :

$$\epsilon \rightarrow N(0, \sigma)$$

Lorsque nous effectuons une régression linéaire sur une donnée à coupe transversale, les résidus sont bien indépendamment distribués car les individus présents dans la base de donnée sont eux aussi indépendants des autres. Par exemple si dans la base nous avons 100 pays et qu'on veut estimer un indicateur économique alors les erreurs dues à ce terme d'estimation seront indépendantes car les résultats d'un pays n'influenceront pas ceux d'un autre. Mais lorsque nous sommes dans le cas d'une série temporelle, dans le cas où on se focalise sur un seul pays par exemple pour observer l'évolution de ses variables, et qu'on veut faire une prévision sur le futur à partir des données du passé il y a un risque d'autocorrelation des erreurs car notre équation, pour prédire le futur, se servira des résultats des années précédentes.

.2 Intuition du test de Durbin-Watson

Supposons que le modèle qu'on veuille estimer est le suivant :

$$Y_t = \beta_0 + \beta_1 X_{t-1} + \epsilon_t$$

L'intuition derrière le test de Durbin-Watson est que les termes d'erreur sont eux aussi le résultat d'une regression. Ainsi une erreur au temps t se présente comme ceci :

$$\epsilon_t = \rho \cdot \epsilon_{t-1} + w_t$$

L'erreur au temps t est égal à un coefficient ρ multiplié par l'erreur de la période précédente plus un bruit blanc.

$$\begin{aligned} & \text{avec} \\ & |\rho| \leq 1 \\ & w_t \rightarrow N(0, 1) \end{aligned}$$

ρ est le coefficient dont on va tester la significativité avec les hypothèses suivantes :

$$\begin{aligned} H_0 : \rho &= 0 \\ H_1 : \rho &\neq 0 \end{aligned}$$

Si donc ρ est égal à 0 alors le coefficient est nul et il n'y a donc pas d'autocorrelation des résidus. Si l'hypothèse nulle est rejettée alors il y a autocorrelation, c'est-à-dire que l'erreur t est correlée avec l'erreur $t - 1$.

.3 Exemple

.3.1 Bruits blancs

Des résidus non correlés sont dits bruits blancs et se présentent sous une forme sinusoïdale. Elle gravitent autour de l'espérance 0 :

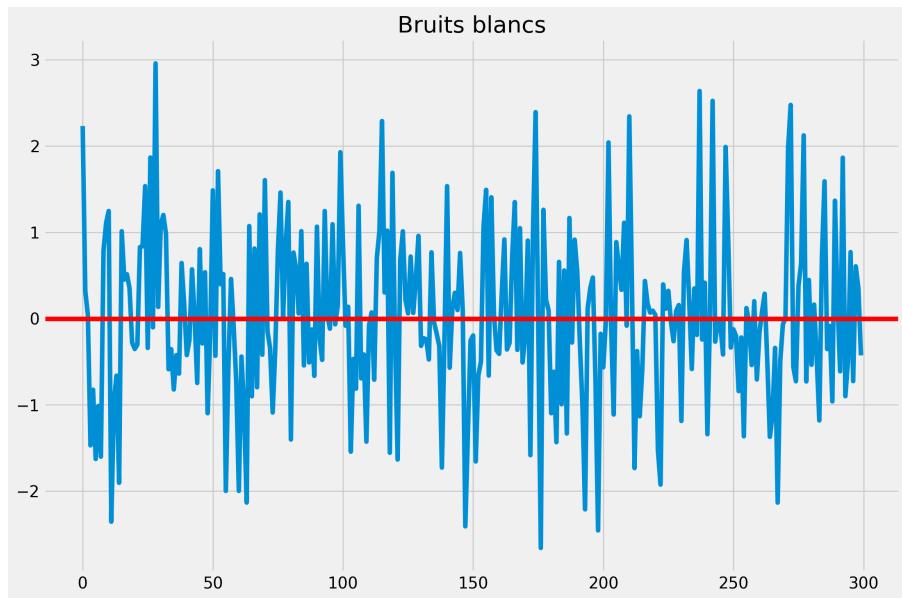


FIGURE 12 : Graphe du bruits blancs.

.3.2 Des résidus autocorrelés et non bruits blancs

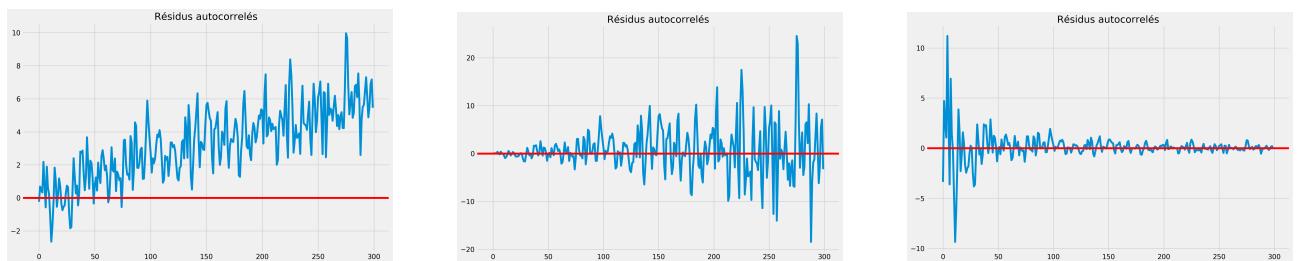


FIGURE 13 : Graphe des résidus autocorrelés et non bruits blancs.

Annexe : Méthodologie de Box-Jenkins

La méthode de Box et Jenkins est un outil systématique qui permet de :

- Déterminer le meilleur modèle de type ARMA décrivant le processus stochastique d'une série observée ou d'une transformation stationnaire de celle-ci ;
- Estimer ce modèle ;
- L'utiliser pour extrapoler les valeurs de la série. La méthodologie de Box et Jenkins comporte essentiellement cinq étapes :

Étape 1 : Transformation des données afin de stabiliser la variance (log, sqrt,...) et différenciation des données pour les stationariser.

Étape 2 : Visualiser les ACF et les PACF empiriques pour identifier les paramètres p et q appropriés.

Étape 3 : Estimation des paramètres du(des) modèle(s) sélectionné(s).

Étape 4 : Diagnostique et tests adéquation du modèle.

Étape 5 : Prévision : La dernière étape consiste la prévision des valeurs futures à travers le modèle retenu.

Annexe : Algorithme Hyndman-Khandakar

Les arguments de **auto.arima()** fournissent de nombreuses variantes de l'algorithme Hyndman-Khandakar. Ce qui est décrit ici est le comportement par défaut :

Hyndman-Khandakar algorithm for automatic ARIMA modelling
1. The number of differences $0 \leq d \leq 2$ is determined using repeated KPSS tests.
2. The values of p and q are then chosen by minimising the AICc after differencing the data d times. Rather than considering every possible combination of p and q , the algorithm uses a stepwise search to traverse the model space.
a. Four initial models are fitted: <ul style="list-style-type: none">◦ ARIMA(0, d, 0),◦ ARIMA(2, d, 2),◦ ARIMA(1, d, 0),◦ ARIMA(0, d, 1). A constant is included unless $d = 2$. If $d \leq 1$, an additional model is also fitted: <ul style="list-style-type: none">◦ ARIMA(0, d, 0) without a constant.
b. The best model (with the smallest AICc value) fitted in step (a) is set to be the “current model.”
c. Variations on the current model are considered: <ul style="list-style-type: none">◦ vary p and/or q from the current model by ± 1;◦ include/exclude c from the current model. The best model considered so far (either the current model or one of these variations) becomes the new current model.
d. Repeat Step 2(c) until no lower AICc can be found.

FIGURE 14 : Algorithme Hyndman-Khandakar

La procédure par défaut utilise quelques approximations pour accélérer la recherche. Ces approximations peuvent être évitées avec l'argument **approximation=FALSE**.

Il est possible que le modèle AICc minimum ne soit pas trouvé en raison de ces approximations, ou en raison de l'utilisation d'une procédure pas à pas. Un ensemble beaucoup plus grand de modèles sera recherché si l'argument **stepwise=FALSE** est utilisé.