Production industrielle mensuelle de glaces et sorbets (1990-2023)

ANDRU Kilian - LACOUR Xavier

Mardi 16 mai 2023

Téléchargement et formatage du fichier

Définition de l'espace de travail puis téléchargement de la série.

```
rm(list = objects())

# Xavier - Ordinateur ENSAE
# path <- "W:/Bureau/S4_ENSAE/STL"

# Xavier - Ordinateur perso
#path <- "D:/Etudes/ENSAE/S4/Séries temporelles linéaires/Projet"

# Kilian - Ordinateur perso
path <- "C:/Users/Kilian/Desktop/ENSAE/S4/LTS/Projet---S-rie-temporelle-lin-aire"

setwd(path)

library(readr)
donnees_choco <- read.csv("valeurs_mensuelles_chocolat.csv",
                          sep = ";", col.names = c("Periode", "Indice", "Code"))
```

On supprime les trois premieres lignes inutiles.

```
donnees_choco = donnees_choco[-c(1,2,3), ]
```

On vérifie que la colonne Code ne prend que la valeur "A" et que donc l'ensemble des valeurs sont normales

```
if(any(donnees_choco["Code"] != "A") == F){
  donnees_choco = subset(donnees_choco, select = c("Periode", "Indice"))
}
```

On transforme la colonne Indice en valeurs numériques

```
donnees_choco["Indice"] = as.numeric(unlist(donnees_choco["Indice"]))
```

Possibilité d'enlever les données d'après mars 2020 pour ne pas subir les perturbation du Covid-19 (On enlève si covid_garde = FALSE)

```
covid_garde = FALSE
if(covid_garde == FALSE){donnees_choco = donnees_choco[-c(363:length(donnees_choco$Indice)), ]}
```

Convertion en format zoo (Si besoin : “install.packages(‘zoo’)” et “install.packages(‘tseries’)”)

```
require(zoo)
require(tseries)

choco <- zoo(donnees_choco["Indice"])
T <- length(choco)
```

Partie I

Question 1

Voir rapport de projet.

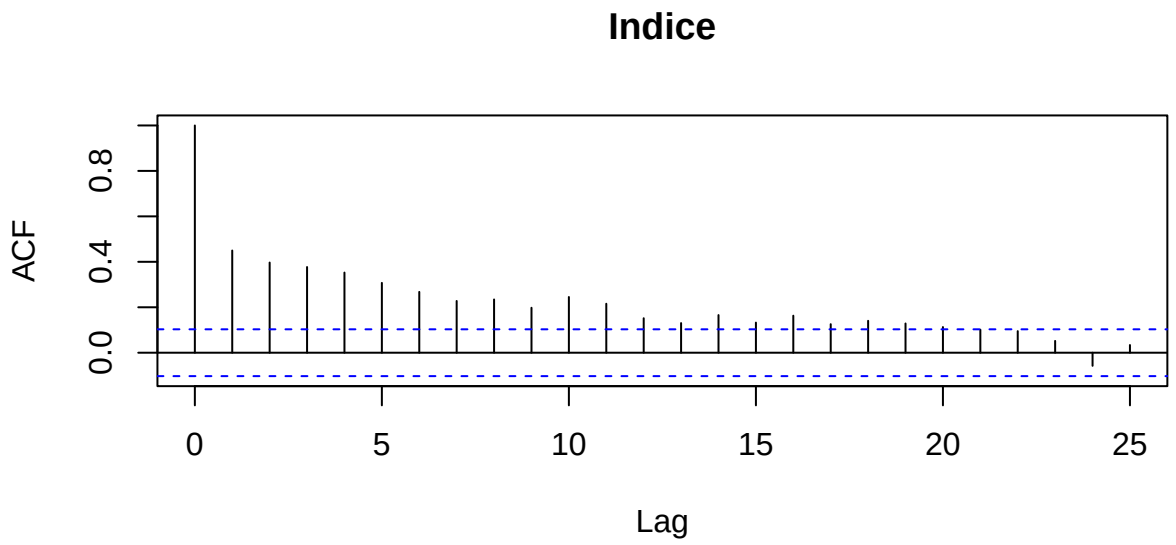
Question 2

Vérifions qu’il n’y a pas de tendance linéaire.

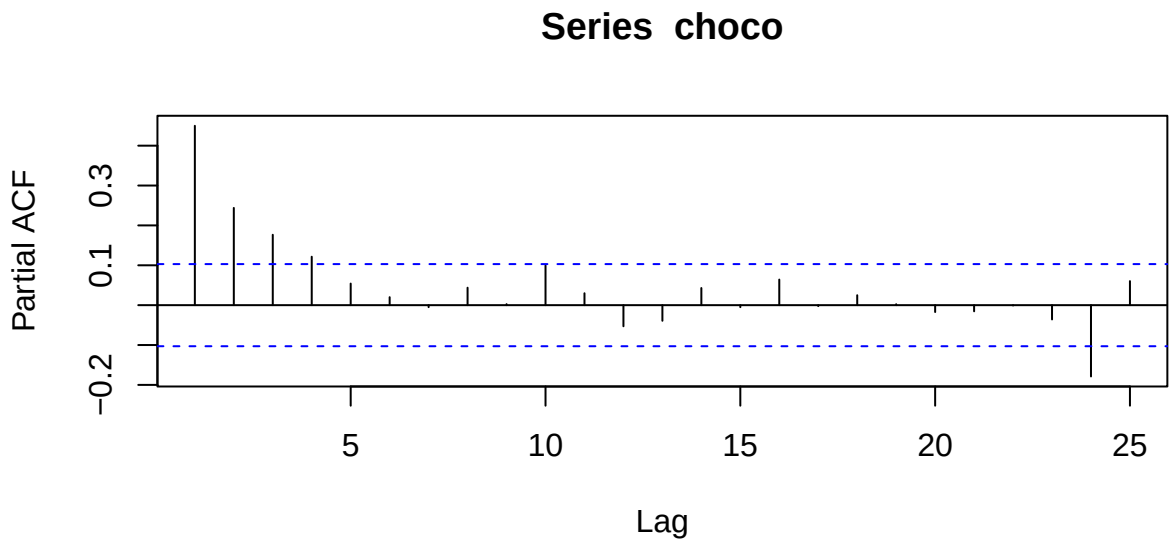
```
library(dplyr)
donnees_choco_reg <-  select(donnees_choco, "Indice")
donnees_choco_reg["temps"] <-  0:(T-1)
summary(lm(Indice~., data = donnees_choco_reg))
```

```
##
## Call:
## lm(formula = Indice ~ ., data = donnees_choco_reg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.7440  -3.3443   0.0781   3.6556  25.7193
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 97.617343   0.569773 171.327   <2e-16 ***
## temps        0.003363   0.002732   1.231    0.219
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 5.432 on 360 degrees of freedom
## Multiple R-squared:  0.004192,   Adjusted R-squared:  0.001425
## F-statistic: 1.515 on 1 and 360 DF,  p-value: 0.2191
```

```
acf(choco)
```

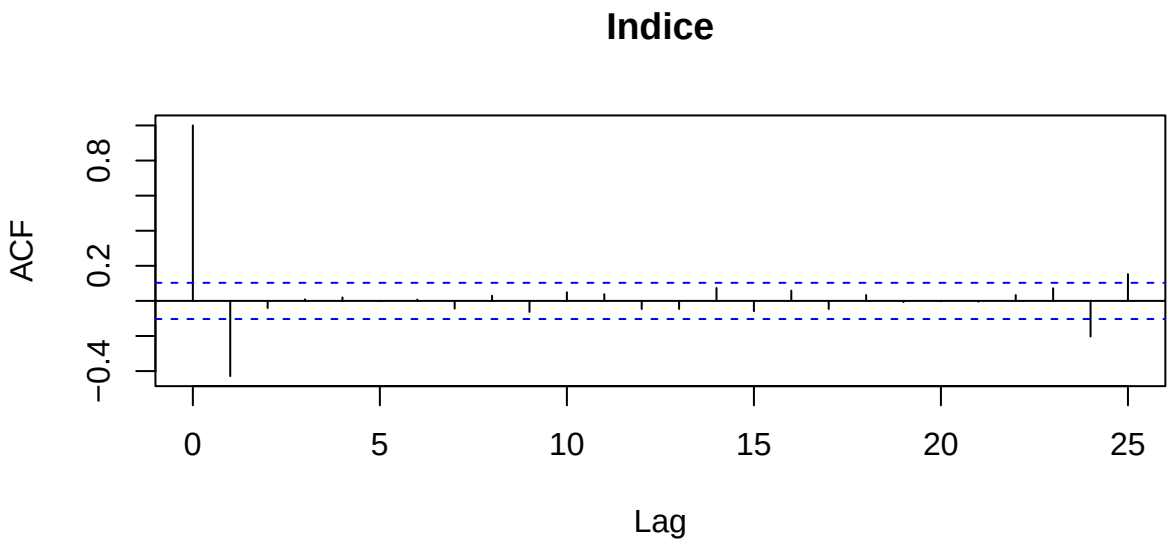



```
pacf(choco)
```

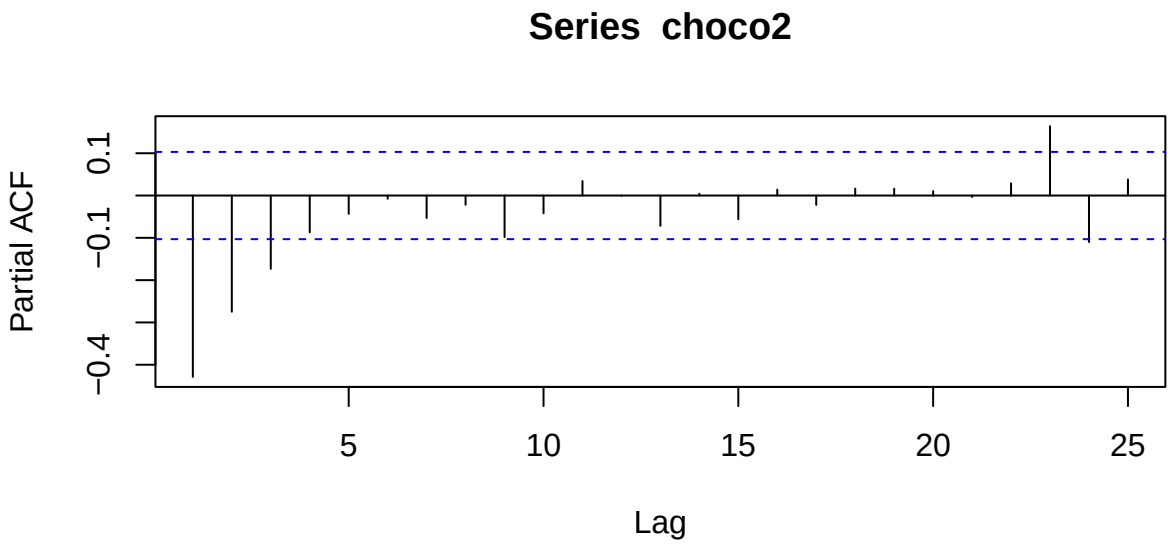


On remarque que la différence la plus présente est de 1. Donc on différentie à l'ordre 1.

```
choco2 = diff(choco, 1)
acf(choco2)
```




```
pacf(choco2)
```



Nous allons effectuer trois tests de stationnarité :

```
# Test de Phillips-Perron
pp.test(choco2)
```

```
## Warning in pp.test(choco2): p-value smaller than printed p-value

##
##  Phillips-Perron Unit Root Test
##
```

```
## data:  choco2
## Dickey-Fuller Z(alpha) = -434.69, Truncation lag parameter = 5, p-value
## = 0.01
## alternative hypothesis: stationary
```

```
# Test de Dickey-Fuller augmenté
adf.test(choco2)
```

```
## Warning in adf.test(choco2): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  choco2
## Dickey-Fuller = -8.7816, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Test KPSS
kpss.test(choco2, null = "Level")
```

```
## Warning in kpss.test(choco2, null = "Level"): p-value greater than printed
## p-value
```

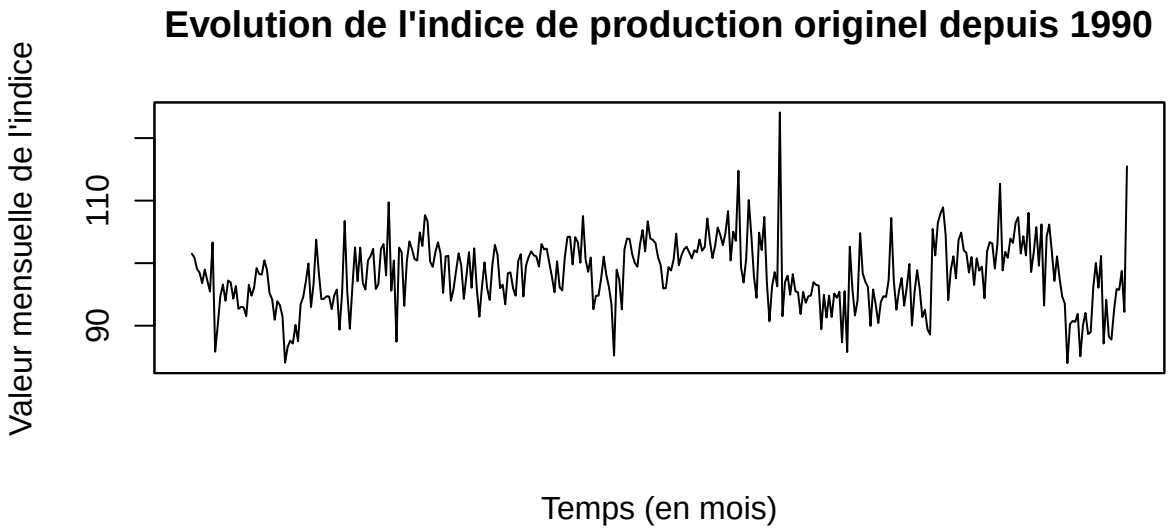
```
##
##  KPSS Test for Level Stationarity
##
## data:  choco2
## KPSS Level = 0.051388, Truncation lag parameter = 5, p-value = 0.1
```

Les trois tests donnent la série stationnaire.

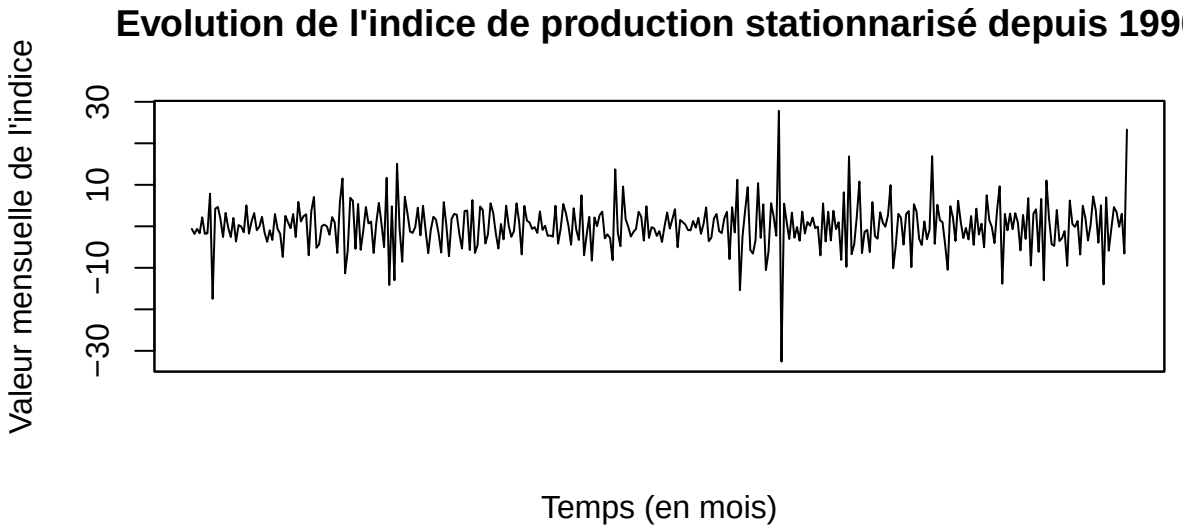
```
Qtests <- function(series, k, fitdf=0) {
  pvals <- apply(matrix(1:k), 1, FUN=function(l) {
    pval <- if (l<=fitdf) NA else Box.test(series, lag=l, type="Ljung-Box", fitdf=fitdf)$p.value
    return(c("lag"=l,"pval"=pval))
  })
  return(t(pvals))
}
```

Question 3

```
# Indice originel
plot(choco, xaxt = "n", xlab ="Temps (en mois)", ylab = "Valeur mensuelle de l'indice",
     main = "Evolution de l'indice de production originel depuis 1990")
```

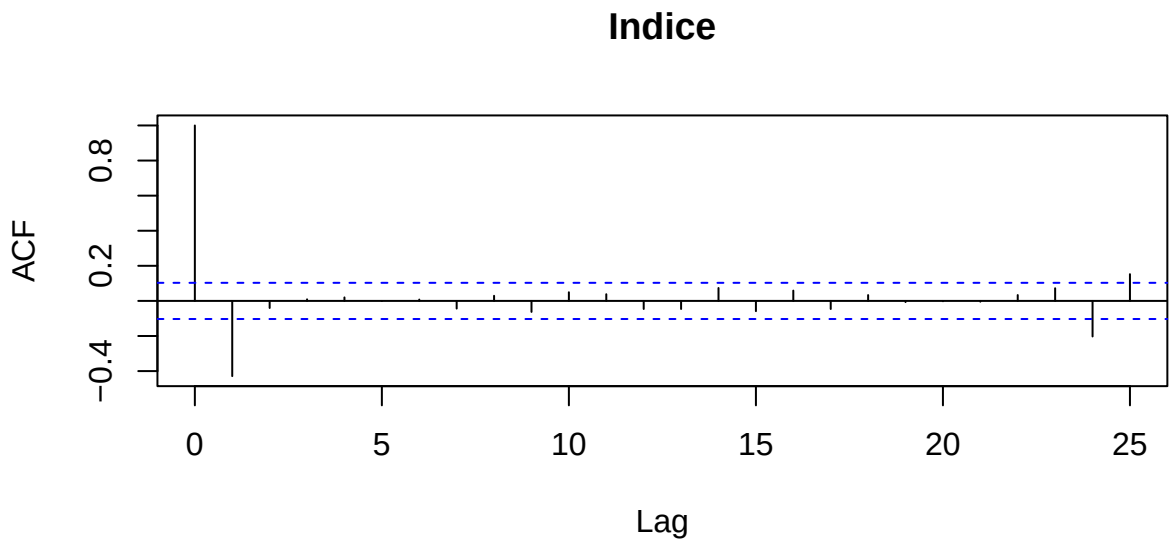



```
# Indice stationnarisé

plot(choco2, xaxt = "n", xlab ="Temps (en mois)", ylab = "Valeur mensuelle de l'indice",
     main = "Evolution de l'indice de production stationnarisé depuis 1990")
```

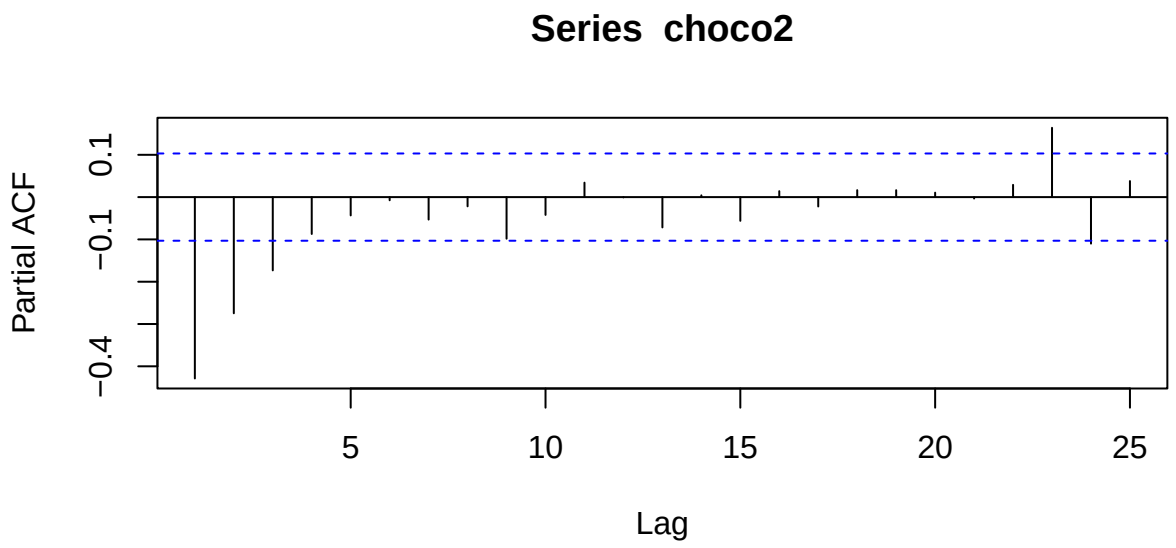


Question 4

```
acf(choco2)
```




```
pacf(choco2)
```



On garde les ordres maximuaux $p^* = 3$ et $q^* = 2$. Testons les combinaisons (p, q) où $0 \leq p \leq p^*$ et $0 \leq q \leq q^*$.

```
p_etoile = 3
q_etoile = 2

pqs <- expand.grid(0:p_etoile,0:q_etoile) # Combinaisons possibles de p<=p* et q<=q*
mat <- matrix(NA, nrow=p_etoile+1, ncol=q_etoile+1)
rownames(mat) <- paste0("p=",0:p_etoile) # Renomme les lignes
colnames(mat) <- paste0("q=",0:q_etoile) # Renomme les colonnes
AICs <- mat # Matrice ou assigner les AIC
```

```
BICs <- mat # Matrice ou assigner les BIC
for (row in 1:dim(pqs)[1]){
  p <- pqs[row,1]
  q <- pqs[row,2]
  estim <- try(arima(choco2,c(p, 0, q), include.mean=F)) # Tente d'estimer l'ARMA
  AICs[p+1,q+1] <- if (class(estim)=="try-error") NA else estim$aic
  BICs[p+1,q+1] <- if (class(estim)=="try-error") NA else BIC(estim)
}
```

On regarde quelles valeurs de (p, q) minimisent l'AIC et le BIC.

```
AICs == min(AICs)
```

```
##       q=0   q=1   q=2
## p=0 FALSE FALSE FALSE
## p=1 FALSE FALSE  TRUE
## p=2 FALSE FALSE FALSE
## p=3 FALSE FALSE FALSE
```

```
BICs == min(BICs)
```

```
##       q=0   q=1   q=2
## p=0 FALSE  TRUE FALSE
## p=1 FALSE FALSE FALSE
## p=2 FALSE FALSE FALSE
## p=3 FALSE FALSE FALSE
```

On garde donc les modèles ARMA(1, 2) et ARMA(0, 1).

```
arima102 <- arima(choco2,c(1,0,2),include.mean=F)
arima001 <- arima(choco2,c(0,0,1),include.mean=F)
```

On vérifie que les deux modèles sont bien ajustés.

```
arima102
```

```
##
## Call:
## arima(x = choco2, order = c(1, 0, 2), include.mean = F)
##
## Coefficients:
##          ar1      ma1     ma2
##       0.9139  -1.6453  0.6453
## s.e.  0.0342   0.0653  0.0649
##
## sigma^2 estimated as 21.04:  log likelihood = -1063.99,  aic = 2135.98
```

```
arima001
```

```
##
## Call:
## arima(x = choco2, order = c(0, 0, 1), include.mean = F)
##
## Coefficients:
##           ma1
##       -0.7322
## s.e.   0.0447
##
## sigma^2 estimated as 21.8:  log likelihood = -1068.93,  aic = 2141.86
```

Pour vérifier cela, on regarde si AR et le MA le plus grand de chaque modèle est significatif. Pour cela, on regarde si la valeur absolue du coefficient divisée par l'écart-type est plus grande que 1.96 (ce qui correspond au seuil de 5%) : $\frac{|Coeff|}{se(Coeff)} \geq 1.96$.

Concernant l'ARMA(1, 2), AR(1) et MA(2) sont tous les deux significatifs car $\frac{0.9139}{0.0342} \simeq 26.72 \geq 1.96$ et $\frac{0.6453}{0.0649} \simeq 9.94 \geq 1.96$.Donc le modèle est bien ajusté.

L'ARMA(0, 1) ne prend en compte que que MA(1) donc on va regarder ce coefficient seulement. Or, $\frac{|-0.7322|}{0.0447} \simeq 16.38 \geq 1.96$. Donc l'ARMA(0, 1) est bien ajusté.

Nous allons maintenant vérifier si ces modèles sont susceptibles de présenter de l'autocorrélation (On prend $fitdf = p + q$).

```
#install.packages("fUnitRoots")
require("fUnitRoots")
```

Pour l'ARMA(1, 2) :

```
Qtests(arima102$residuals,20,fitdf=3)
```

```
##       lag      pval
##  [1,]   1        NA
##  [2,]   2        NA
##  [3,]   3        NA
##  [4,]   4 0.4824044
##  [5,]   5 0.7811763
##  [6,]   6 0.8910427
##  [7,]   7 0.8157325
##  [8,]   8 0.9055844
##  [9,]   9 0.9121785
## [10,]  10 0.7901616
## [11,]  11 0.7941658
## [12,]  12 0.8014378
## [13,]  13 0.7708642
## [14,]  14 0.8120455
## [15,]  15 0.8583173
## [16,]  16 0.8461977
## [17,]  17 0.8907869
## [18,]  18 0.8990984
## [19,]  19 0.9145627
## [20,]  20 0.9324906
```

On ne rejette pas H_0 et l'absence de corrélation.Donc l'ARMA(1, 2) paraît valide.

Pour l'ARMA(0, 1) :

```
Qtests(arima001$residuals,20,fitdf=1)
```

```
##       lag      pval
##  [1,]   1        NA
##  [2,]   2 0.4700417
##  [3,]   3 0.7667304
##  [4,]   4 0.9025284
##  [5,]   5 0.9439162
##  [6,]   6 0.9056300
##  [7,]   7 0.7006264
##  [8,]   8 0.7577959
##  [9,]   9 0.6916572
## [10,]  10 0.7035659
## [11,]  11 0.7710588
## [12,]  12 0.7041430
## [13,]  13 0.5913742
## [14,]  14 0.6689876
## [15,]  15 0.7061732
## [16,]  16 0.7318438
## [17,]  17 0.7877226
## [18,]  18 0.8131196
## [19,]  19 0.8399142
## [20,]  20 0.8689194
```

On ne peut pas rejeter l'absence de corrélation ici non plus donc l'ARMA(0, 1) semble valide.

Départageons les deux modèles en regardant lequel des deux est le plus efficace.

```
adj_r2 <- function(model){
  ss_res <- sum(model$residuals^2)
  ss_tot <- sum(choco2[-c(1:max(p,q))]^2)
  p <- model$arma[1]
  q <- model$arma[2]
  n <- model$nobs-max(p,q)
  adj_r2 <- 1-(ss_res/(n-p-q-1))/(ss_tot/(n-1))
  return(adj_r2)
}
adj_r2(arima102)
```

```
## [1] 0.3292794
```

```
adj_r2(arima001)
```

```
## [1] 0.3090064
```

L'ARMA(1, 2) a le meilleur R^2 ajusté, c'est donc celui que nous sélectionnons.

Moyenne temporelle de la série différenciée (choco2):

```
mean(choco2)
```

```
## [1] 0.03872576
```

Question 5

On peut vérifier pour être sur que l'ARIMA(1, 1, 2) est meilleur que l'ARIMA(0, 1, 1).

```
arima112 <- arima(choco,c(1,1,2),include.mean=F)
arima011 <- arima(choco,c(0,1,1),include.mean=F)

adj_r2(arima112)
```

```
## [1] 0.3292785
```

```
adj_r2(arima011)
```

```
## [1] 0.3090055
```

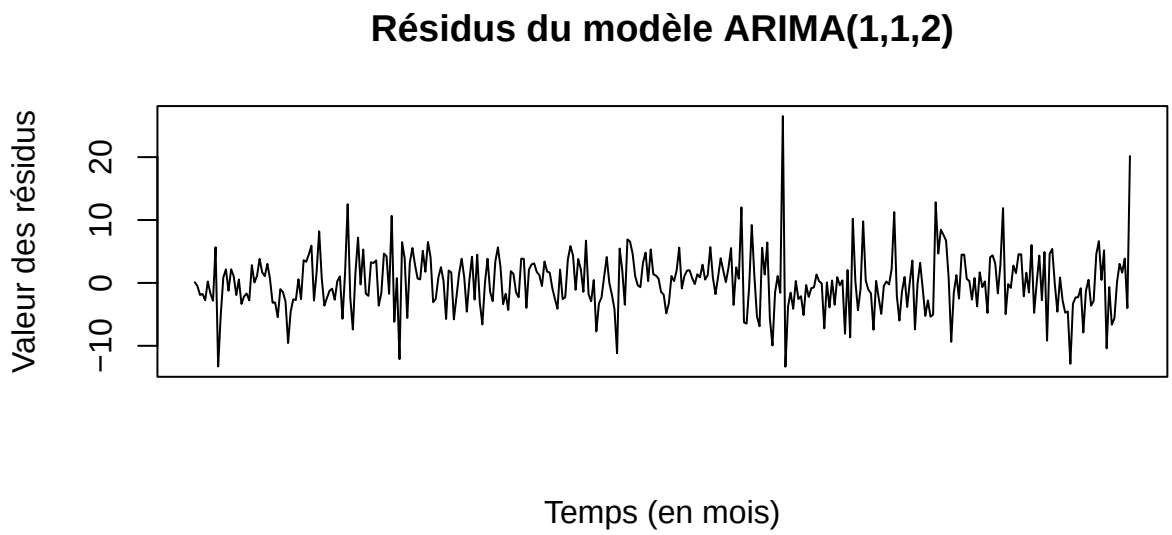
Question 6

Voir Rapport

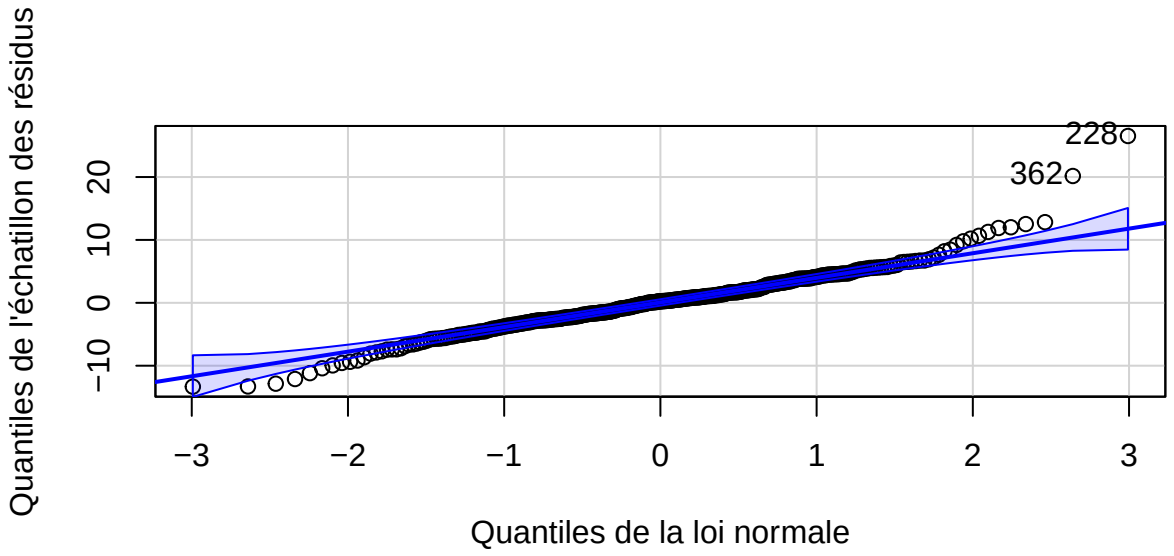
Question 7

On teste la normalité des résidus.

```
plot(arima112$residuals, xaxt = "n", xlab ="Temps (en mois)",
     ylab = "Valeur des résidus", main = "Résidus du modèle ARIMA(1,1,2)")
```



```
library(car)
qqPlot(arima112$residuals, xlab ="Quantiles de la loi normale",
       ylab = "Quantiles de l'échatillon des résidus")
```




```
## [1] 228 362
```

Question 8

On utilise la librairie “forecast” pour les prédictions, “dplyr”, “ggplot2” et “reshape2” pour les représenter.

```
#install.packages("forecast")
require(forecast)
library(dplyr)
library(ggplot2)
library(reshape2)
```

On effectue la prédiction pour deux périodes supplémentaires au seuil de 95%. On récupère la prédiction et les intervalles de confiance.

```
pred_arima112=forecast(arima112,h=2,level=95)

prediction = zoo(pred_arima112$mean)
prediction_inf= zoo(pred_arima112$lower)
prediction_sup= zoo(pred_arima112$upper)
```

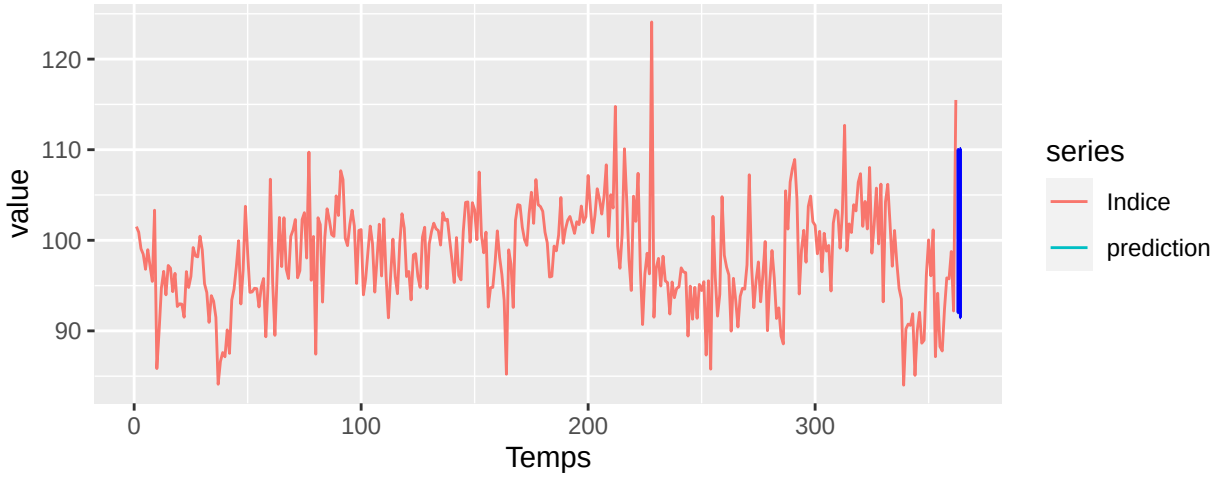
Avant de tracer la prédiction avec ggplot, on prépare un tableau contenant une colonne avec les indices de la série et ses prédictions et d’autres colonnes avec les bornes des intervalles de confiance.

```
#Tableau de la série
SERIE <-  data.frame(choco)
```

```
SERIE["Temps"] <-  as.numeric(rownames(SERIE))

#Tableau des prédictions
PREDICTIONS <-data.frame(prediction)
PREDICTIONS["Temps"] <- as.numeric(rownames(PREDICTIONS))

#Tableau avec série+prédictions
PRED_SERIE = merge(x=SERIE, y=PREDICTIONS, by="Temps", all=TRUE)
PRED_SERIE_MELTED <- melt(PRED_SERIE ,  id.vars = 'Temps', variable.name = 'series')

#Tableau avec les intervalles de confiance
CONFIANCE <- data.frame(prediction_inf, prediction_sup)
CONFIANCE["Temps"] <- as.numeric(rownames(CONFIANCE))

#Tableau final
ALL <-  merge(x=PRED_SERIE_MELTED, y=CONFIANCE, by="Temps", all=TRUE)
names(ALL) <- c("Temps", "series", "value", "inf", "sup")
```

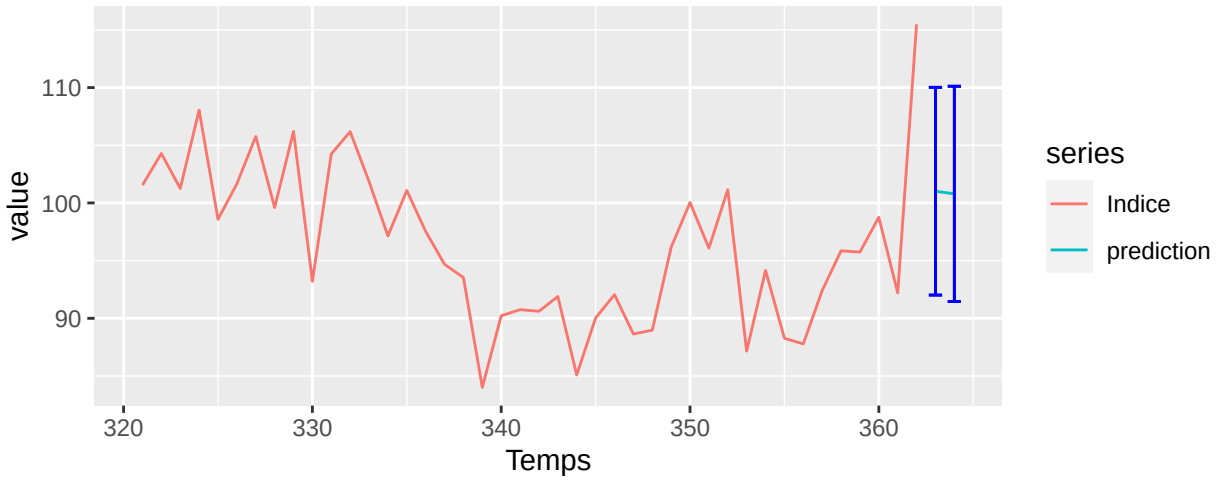
On trace la série et ses prédictions avec ggplot, on utilise des barres d'erreur pour représenter l'intervalle de confiance.

```
ggplot(ALL, aes(Temps, value)) +
  geom_line(aes(colour =series)) +
  geom_errorbar(aes(ymin = inf, ymax = sup),
                width = 0.5, color = "blue")
```



Les prédictions sont peu visibles car il y a trop de valeurs, on se concentre uniquement sur la fin de la série.

```
ALL_red <- ALL[ALL$Temps > 320, ]
ggplot(ALL_red, aes(Temps, value)) +
  geom_line(aes(colour =series)) +
  geom_errorbar(aes(ymin = inf, ymax = sup),
                width = 0.7, color = "blue")
```



On trace l'ellipse correpsondant à la region de confiance trouvée en question 7. On utilise les equations paramétriques d'une ellipse.

```
xc <- as.numeric(prediction[1]) # center x
yc <- as.numeric(prediction[2]) # y
a <- as.numeric(prediction_sup[1])-as.numeric(prediction[1]) # axe x
b <- as.numeric(prediction_sup[2])-as.numeric(prediction[2]) # axe y

t <- seq(0, 2*pi, 0.01)
x <- xc + a*cos(t) - b*sin(t)
y <- yc + a*cos(t) + b*sin(t)
x <- c(x,xc)
y <- c(y,yc)
plot(x,y,pch=19, col='blue')
```