Indice brut de la production industrielle : Construction aéronautique et spatiale

HADDOUCHE Théo, EL BAZ Avner

2025-04-08

Introduction

Librairies

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(readr)
library(forecast)
library(ggplot2)
```

Import et délimitation du dataset

```
valeurs_mensuelles <- read_delim("sorbe.csv", delim = ";", col_types = cols())
vm <- valeurs_mensuelles[-c(1, 2, 3), ]

colnames(vm) <- c("date","value","code")
vm <- vm[nrow(vm):1, ]
vm$value <- as.numeric(vm$value)

vm$log <- log(vm$value)
vm$diff <- c(NA,diff(vm$value, lag = 1))
vm$season <-  c(rep(NA, 12), diff(vm$value, lag = 12))
vm$seasonlog <- c(rep(NA, 12), diff(vm$log, lag = 12))

head(vm)
```

```
## # A tibble: 6 x 7
##   date    value code    log  diff season seasonlog
##   <chr>   <dbl> <chr> <dbl> <dbl>  <dbl>     <dbl>
## 1 1990-01  54.7 A      4.00 NA         NA        NA
## 2 1990-02  59.4 A      4.08  4.71      NA        NA
## 3 1990-03  90.2 A      4.50 30.8       NA        NA
## 4 1990-04 115.  A      4.75 24.9       NA        NA
## 5 1990-05 122.  A      4.81  7.11      NA        NA
## 6 1990-06 130.  A      4.87  7.44      NA        NA
```

Part I : The Data

1. What does the chosen series represent ? (sector, potential data processing, logarithmic transformation, etc.)

La série représente la production

2. Transform the series to make it stationary if necessary (differentiate it, correct the deterministic trend, etc.). Thoroughly justify your choices.

```r
serie_ts <- ts(vm$value, start = c(1990, 01), frequency = 12)

diff_series <- ts(vm$diff, start = c(1990, 02), frequency = 12)
diff_series <- na.omit(diff_series)

season_series <- ts(vm$season, start = c(1991, 02), frequency = 12)
season_series <- na.omit(season_series)

seasonlog_series <- ts(vm$seasonlog, start = c(1991, 02), frequency = 12)
seasonlog_series <- na.omit(seasonlog_series)

# Dickey-Fuller Test
adf.test(serie_ts, alternative="stationary")
```

```
## Warning in adf.test(serie_ts, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  serie_ts
## Dickey-Fuller = -13.53, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(season_series, alternative="stationary")
```

```
## Warning in adf.test(season_series, alternative = "stationary"): p-value smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  season_series
## Dickey-Fuller = -6.899, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(diff_series, alternative="stationary")
```

```
## Warning in adf.test(diff_series, alternative = "stationary"): p-value smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_series
## Dickey-Fuller = -19.584, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(seasonlog_series, alternative="stationary")
```

```
## Warning in adf.test(seasonlog_series, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  seasonlog_series
## Dickey-Fuller = -7.011, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(diff_series, alternative="stationary")
```

```
## Warning in pp.test(diff_series, alternative = "stationary"): p-value smaller
## than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff_series
## Dickey-Fuller Z(alpha) = -295.34, Truncation lag parameter = 5, p-value
## = 0.01
## alternative hypothesis: stationary
```

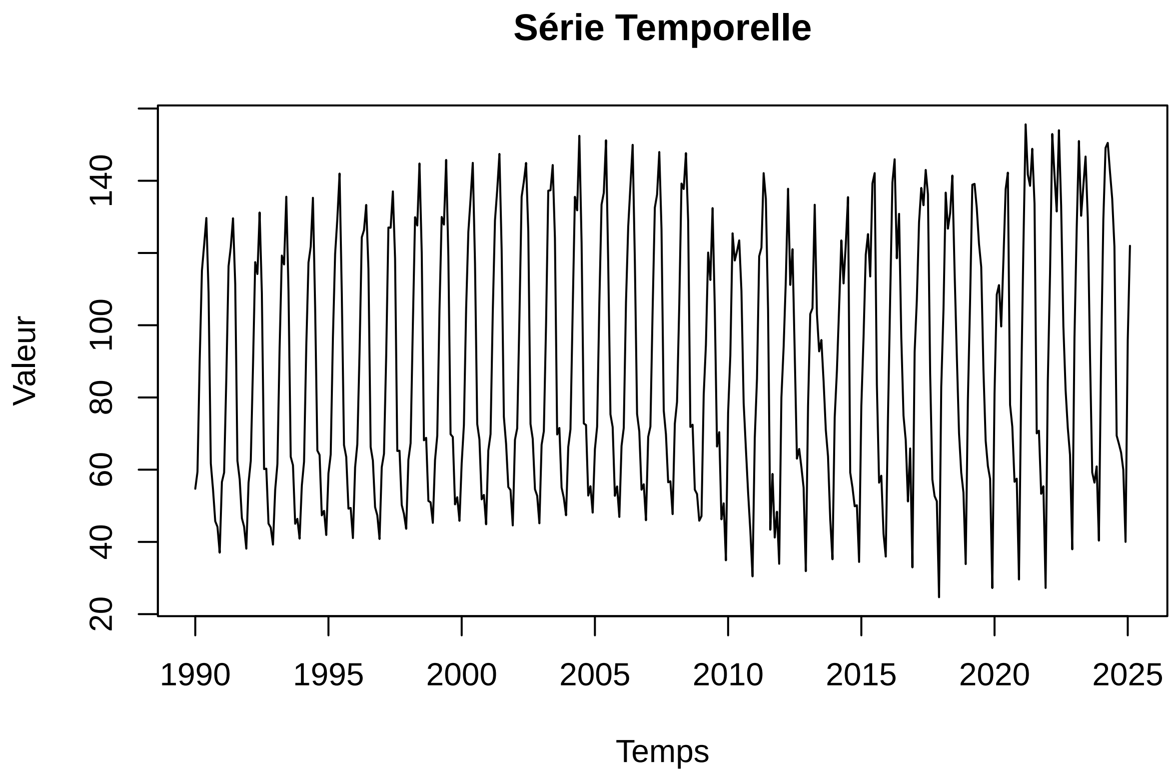
```r
kpss.test((diff_series))
```

```
## Warning in kpss.test((diff_series)): p-value greater than printed p-value
```

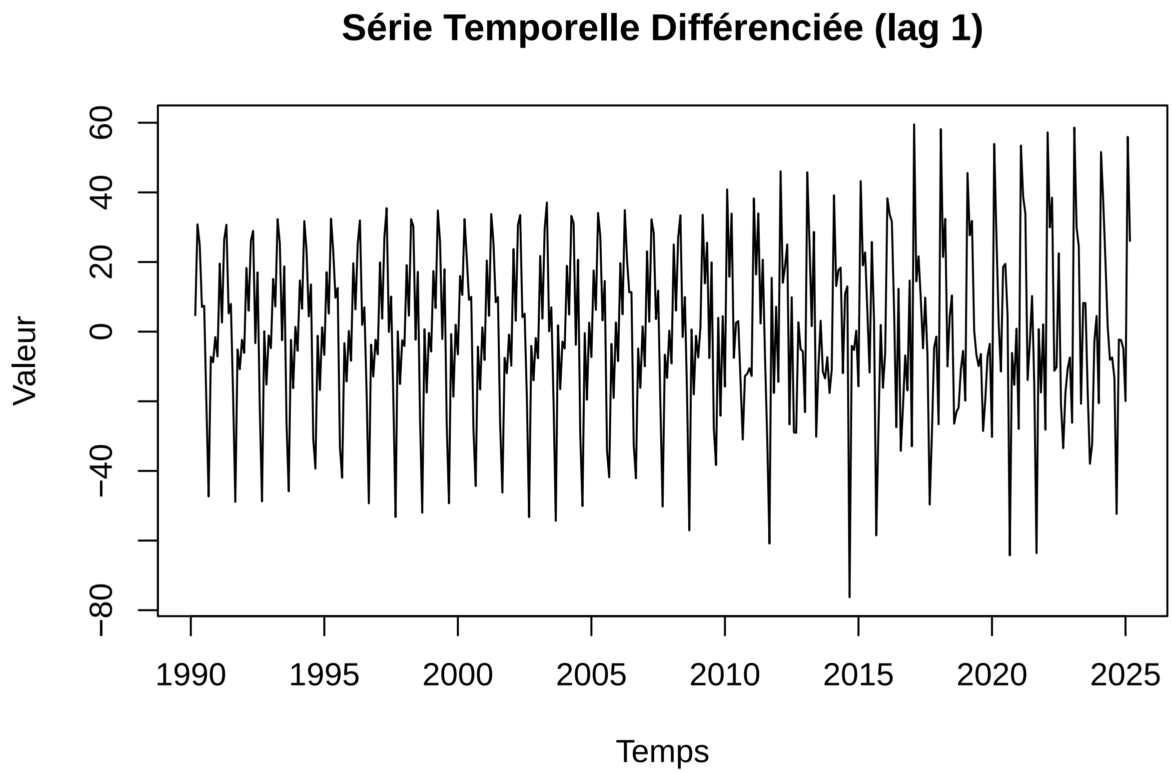
```
##
##  KPSS Test for Level Stationarity
##
## data:  (diff_series)
## KPSS Level = 0.0044531, Truncation lag parameter = 5, p-value = 0.1
```

3. Graphically represent the chosen series before and after transforming it.

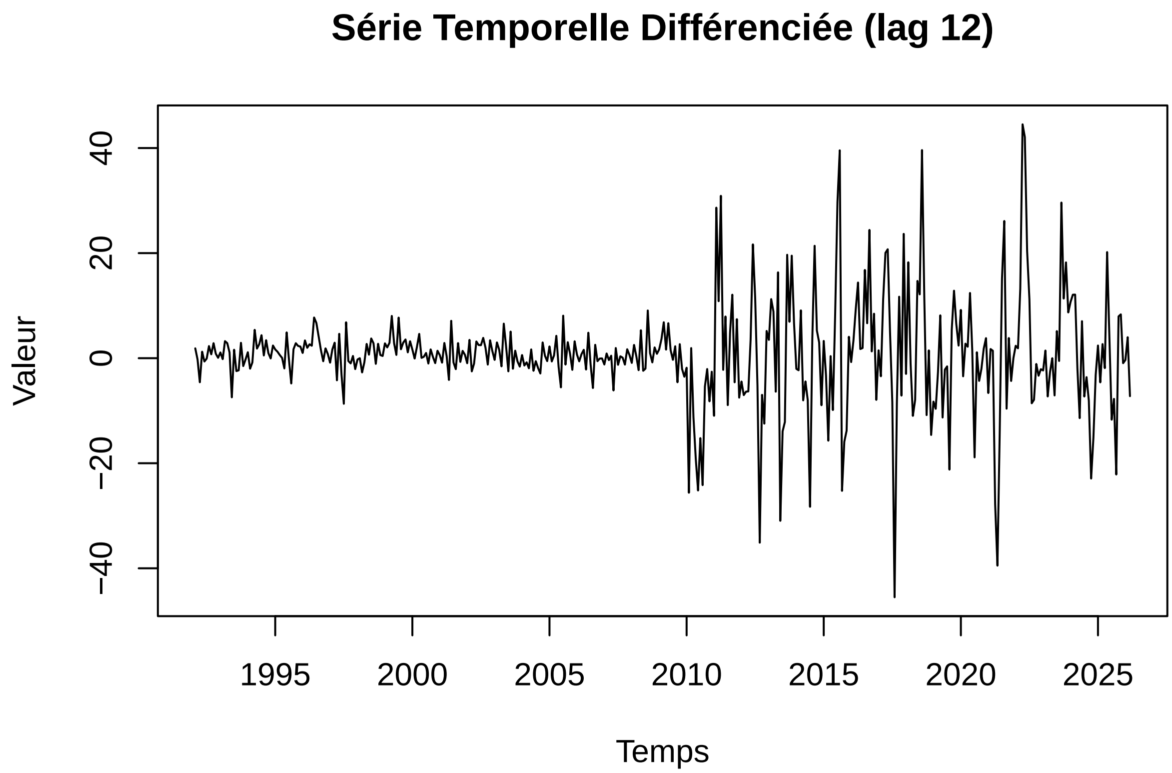
```r
plot(serie_ts, main="Série Temporelle", xlab="Temps", ylab="Valeur")
```



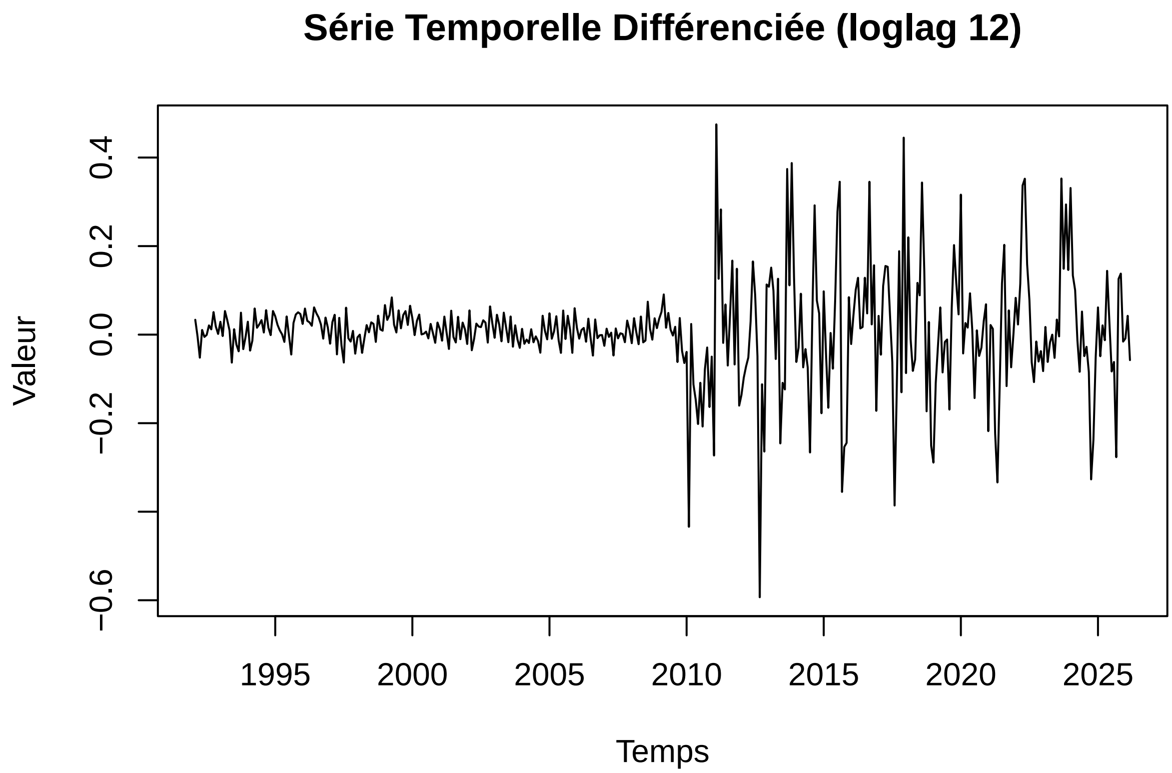

```
plot(diff_series, main="Série Temporelle Différenciée (lag 1)", xlab="Temps", ylab="Valeur")
```




```r
plot(season_series, main="Série Temporelle Différenciée (lag 12)", xlab="Temps", ylab="Valeur")
```




```
plot(seasonlog_series, main="Série Temporelle Différenciée (loglag 12)", xlab="Temps", ylab="Valeur")
```



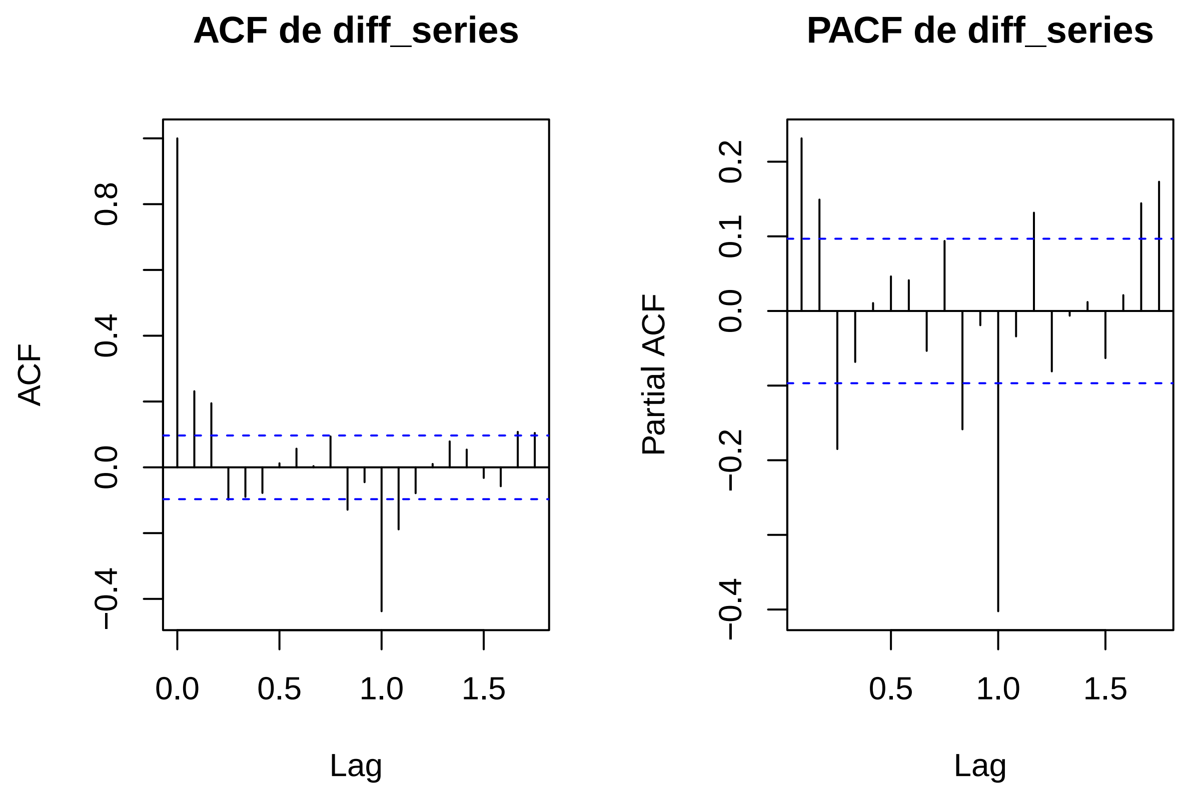
Part II : ARMA models

4. Pick (and justify your choice) an ARMA(p,q) model for your corrected time series Xt. Estimate the model parameters and check its validity.

```r
# Définir la disposition des graphiques
par(mfrow=c(1,2), mar=c(5,4,4,2) + 0.1)

# Tracer l'ACF avec un ajustement de l'axe des abscisses
acf(seasonlog_series, main="ACF de diff_series", lag.max = 21, xlab="Lag")


# Tracer le PACF avec un ajustement de l'axe des abscisses
pacf(seasonlog_series, main="PACF de diff_series", lag.max = 21)
```




```
# Réinitialiser la disposition des graphiques
par(mfrow=c(1,1))
```

On peut voir que

```
# Charger les bibliothèques nécessaires
library(forecast)

# Définir les valeurs de p et q
p_values <- 0:10
q_values <- 0:10

# Créer une matrice pour stocker les résultats
results <- expand.grid(p = p_values, q = q_values)
results$AIC <- NA
results$BIC <- NA

# Boucle pour ajuster les modèles et calculer AIC & BIC
for (i in 1:nrow(results)) {
  p <- results$p[i]
  q <- results$q[i]  # Utiliser "i" au lieu de "j"

  # Ajuster le modèle ARMA(p, q)
  model <- tryCatch(
    arima(diff_series, order = c(p, 0, q)),  # Modèle ARMA sur la série différenciée
    error = function(e) return(NULL)
  )
```

```r
  # Si le modèle est valide, stocker AIC et BIC
  if (!is.null(model)) {
    results$AIC[i] <- model$aic
    results$BIC[i] <- BIC(model)
  }
}
```

```
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
```

```
## Warning in log(s2): NaNs produced
```

```
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
```

```
## Warning in log(s2): NaNs produced
```

```
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(diff_series, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
```

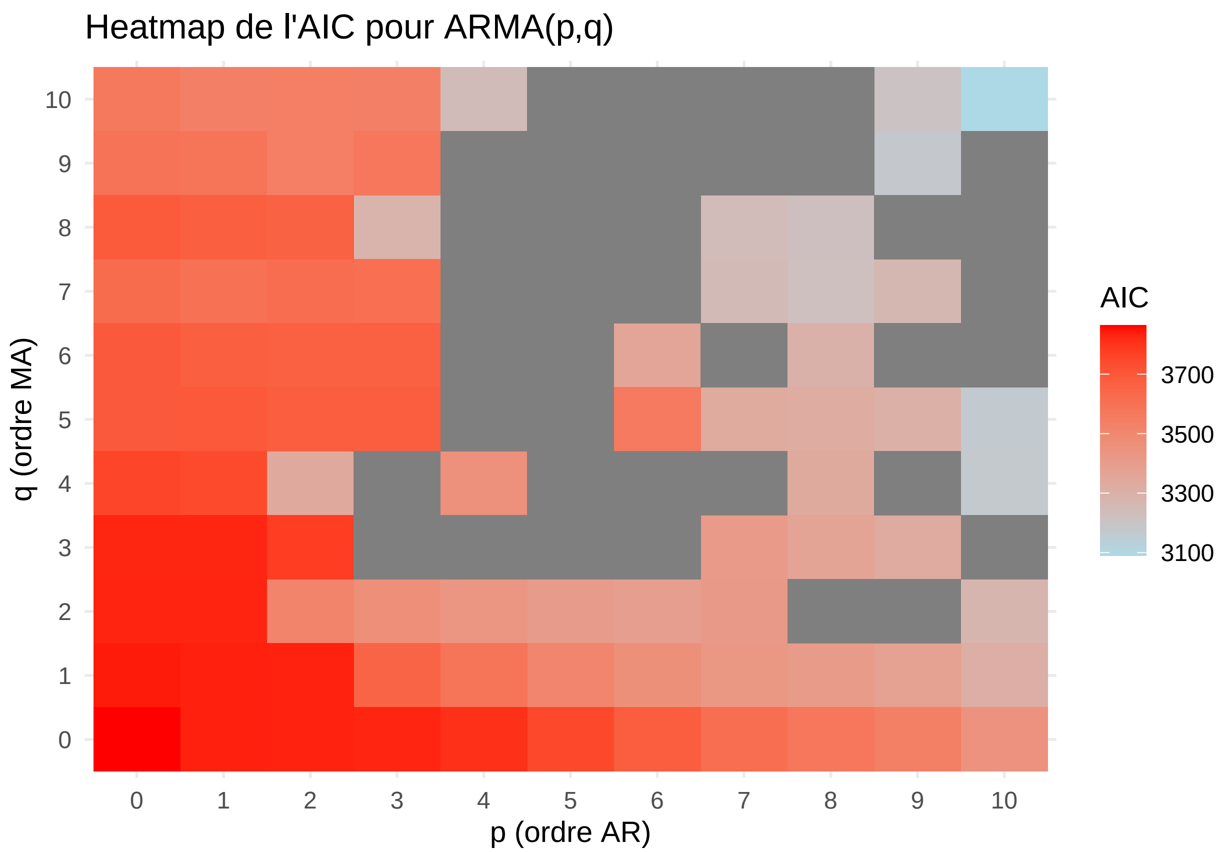
```r
# Afficher le tableau trié par AIC
results <- results[order(results$AIC), ]
print(results)
```

```
##      p  q      AIC      BIC
## 121 10 10 3091.263 3180.201
## 66  10  5 3169.348 3238.073
## 55  10  4 3174.446 3239.128
## 109  9  9 3181.469 3262.322
## 120  9 10 3209.066 3293.961
## 86   8  7 3222.598 3291.323
## 97   8  8 3223.863 3296.630
## 96   7  8 3242.811 3311.536
## 115  4 10 3246.940 3311.622
## 85   7  7 3249.642 3314.324
## 87   9  7 3270.008 3342.776
## 33  10  2 3276.585 3333.182
## 92   3  8 3284.077 3336.631
## 75   8  6 3299.633 3364.315
## 65   9  5 3304.869 3369.551
```

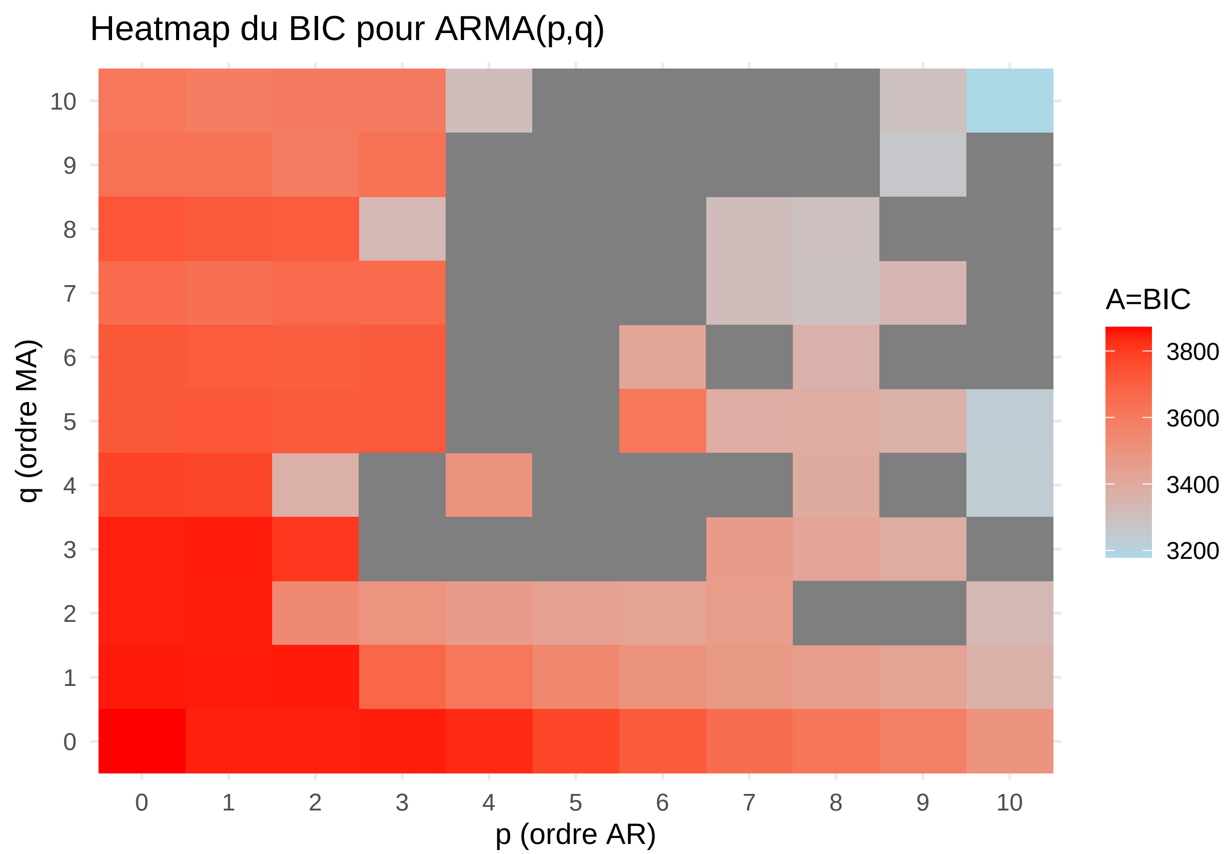
```
## 22   10  1 3313.766 3366.320
## 64    8  5 3326.477 3387.116
## 63    7  5 3332.409 3389.006
## 43    9  3 3333.389 3389.986
## 53    8  4 3337.583 3394.180
## 47    2  4 3339.572 3371.913
## 73    6  6 3360.546 3417.143
## 42    8  3 3369.506 3422.060
## 21    9  1 3379.901 3428.412
## 29    6  2 3390.579 3431.005
## 28    5  2 3403.681 3440.064
## 20    8  1 3412.027 3456.496
## 41    7  3 3416.890 3465.402
## 30    7  2 3419.038 3463.507
## 19    7  1 3431.688 3472.114
## 27    4  2 3434.679 3467.020
## 11   10  0 3452.109 3500.621
## 49    4  4 3458.465 3498.891
## 18    6  1 3464.587 3500.971
## 26    3  2 3467.917 3496.215
## 17    5  1 3514.096 3546.437
## 25    2  2 3518.068 3542.324
## 112   1 10 3538.920 3591.474
## 114   3 10 3538.994 3599.633
## 10    9  0 3540.724 3585.193
## 113   2 10 3544.116 3600.713
## 102   2  9 3544.745 3597.299
## 62    6  5 3562.894 3615.448
## 111   0 10 3568.997 3617.509
## 9     8  0 3577.926 3618.352
## 103   3  9 3579.215 3635.812
## 101   1  9 3587.745 3636.256
## 16    4  1 3589.207 3617.505
## 100   0  9 3593.380 3637.849
## 79    1  7 3601.899 3642.325
## 81    3  7 3612.946 3661.457
## 8     7  0 3616.880 3653.264
## 80    2  7 3619.466 3663.935
## 78    0  7 3623.839 3660.223
## 15    3  1 3655.279 3679.535
## 91    2  8 3662.957 3711.468
## 69    2  6 3667.227 3707.653
## 70    3  6 3669.089 3713.557
## 90    1  8 3675.317 3719.786
## 68    1  6 3675.410 3711.793
## 58    2  5 3678.113 3714.497
## 59    3  5 3679.059 3719.485
## 7     6  0 3680.701 3713.042
## 89    0  8 3693.262 3733.688
## 67    0  6 3694.868 3727.209
## 56    0  5 3695.746 3724.044
## 57    1  5 3696.518 3728.859
## 46    1  4 3744.627 3772.925
## 6     5  0 3749.270 3777.569
```

```
## 45    0  4 3756.656 3780.912
## 36    2  3 3779.809 3808.107
## 5     4  0 3809.001 3833.257
## 35    1  3 3827.758 3852.014
## 34    0  3 3827.945 3848.158
## 4     3  0 3829.305 3849.518
## 24    1  2 3830.067 3850.280
## 23    0  2 3830.438 3846.608
## 3     2  0 3832.387 3848.557
## 14    2  1 3833.003 3853.216
## 13    1  1 3835.305 3851.476
## 2     1  0 3835.613 3847.741
## 12    0  1 3842.448 3854.576
## 1     0  0 3863.303 3871.389
## 31    8  2       NA       NA
## 32    9  2       NA       NA
## 37    3  3       NA       NA
## 38    4  3       NA       NA
## 39    5  3       NA       NA
## 40    6  3       NA       NA
## 44   10  3       NA       NA
## 48    3  4       NA       NA
## 50    5  4       NA       NA
## 51    6  4       NA       NA
## 52    7  4       NA       NA
## 54    9  4       NA       NA
## 60    4  5       NA       NA
## 61    5  5       NA       NA
## 71    4  6       NA       NA
## 72    5  6       NA       NA
## 74    7  6       NA       NA
## 76    9  6       NA       NA
## 77   10  6       NA       NA
## 82    4  7       NA       NA
## 83    5  7       NA       NA
## 84    6  7       NA       NA
## 88   10  7       NA       NA
## 93    4  8       NA       NA
## 94    5  8       NA       NA
## 95    6  8       NA       NA
## 98    9  8       NA       NA
## 99   10  8       NA       NA
## 104   4  9       NA       NA
## 105   5  9       NA       NA
## 106   6  9       NA       NA
## 107   7  9       NA       NA
## 108   8  9       NA       NA
## 110  10  9       NA       NA
## 116   5 10       NA       NA
## 117   6 10       NA       NA
## 118   7 10       NA       NA
## 119   8 10       NA       NA
```

```r
ggplot(results, aes(x = factor(p), y = factor(q), fill = AIC)) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "red") +
  labs(title = "Heatmap de l'AIC pour ARMA(p,q)",
       x = "p (ordre AR)",
       y = "q (ordre MA)",
       fill = "AIC") +
  theme_minimal()
```




```r
ggplot(results, aes(x = factor(p), y = factor(q), fill = BIC)) +
  geom_tile() +
  scale_fill_gradient(low = "lightblue", high = "red") +
  labs(title = "Heatmap du BIC pour ARMA(p,q)",
       x = "p (ordre AR)",
       y = "q (ordre MA)",
       fill = "A=BIC") +
  theme_minimal()
```




```
best_model <- auto.arima(seasonlog_series, ic="aic", seasonal = FALSE, approximation = FALSE, trace=TRUE
```

```
##
##  ARIMA(2,0,2)            with non-zero mean : -646.0801
##  ARIMA(0,0,0)            with non-zero mean : -601.6794
##  ARIMA(1,0,0)            with non-zero mean : -622.1878
##  ARIMA(0,0,1)            with non-zero mean : -615.8829
##  ARIMA(0,0,0)            with zero mean     : -601.5612
##  ARIMA(1,0,2)            with non-zero mean : -639.083
##  ARIMA(2,0,1)            with non-zero mean : -633.8079
##  ARIMA(3,0,2)            with non-zero mean : -640.1184
##  ARIMA(2,0,3)            with non-zero mean : -645.962
##  ARIMA(1,0,1)            with non-zero mean : -623.5451
##  ARIMA(1,0,3)            with non-zero mean : -643.442
##  ARIMA(3,0,1)            with non-zero mean : -641.0123
##  ARIMA(3,0,3)            with non-zero mean : -638.2236
##  ARIMA(2,0,2)            with zero mean     : -646.5384
##  ARIMA(1,0,2)            with zero mean     : -639.9843
##  ARIMA(2,0,1)            with zero mean     : -634.7828
##  ARIMA(3,0,2)            with zero mean     : -640.5781
##  ARIMA(2,0,3)            with zero mean     : -646.571
##  ARIMA(1,0,3)            with zero mean     : -642.208
##  ARIMA(3,0,3)            with zero mean     : -638.6837
##  ARIMA(2,0,4)            with zero mean     : -638.7961
##  ARIMA(1,0,4)            with zero mean     : -640.7921
##  ARIMA(3,0,4)            with zero mean     : -650.2246
```

```
##  ARIMA(4,0,4)            with zero mean     : Inf
##  ARIMA(3,0,5)            with zero mean     : -664.5033
##  ARIMA(2,0,5)            with zero mean     : -661.6694
##  ARIMA(4,0,5)            with zero mean     : -649.433
##  ARIMA(3,0,5)            with non-zero mean : -665.7669
##  ARIMA(2,0,5)            with non-zero mean : -660.778
##  ARIMA(3,0,4)            with non-zero mean : -636.2116
##  ARIMA(4,0,5)            with non-zero mean : -650.6496
##  ARIMA(2,0,4)            with non-zero mean : -640.0299
##  ARIMA(4,0,4)            with non-zero mean : Inf
##
##  Best model: ARIMA(3,0,5)            with non-zero mean
```

```
summary(best_model)
```

```
## Series: seasonlog_series
## ARIMA(3,0,5) with non-zero mean
##
## Coefficients:
##           ar1     ar2     ar3     ma1      ma2      ma3      ma4      ma5
##       -0.6946  0.5323  0.6687  0.9520  -0.1267  -0.7457  -0.3697  -0.3096
## s.e.   0.0788  0.1098  0.0741  0.0834   0.1252   0.1045   0.0630   0.0588
##         mean
##       0.0086
## s.e.  0.0043
##
## sigma^2 = 0.01118:  log likelihood = 342.88
## AIC=-665.77   AICc=-665.22   BIC=-625.61
##
## Training set error measures:
##                         ME      RMSE        MAE  MPE MAPE     MASE       ACF1
## Training set -2.461424e-05 0.1045639 0.06655398 -Inf  Inf 0.556558 0.01039516
```

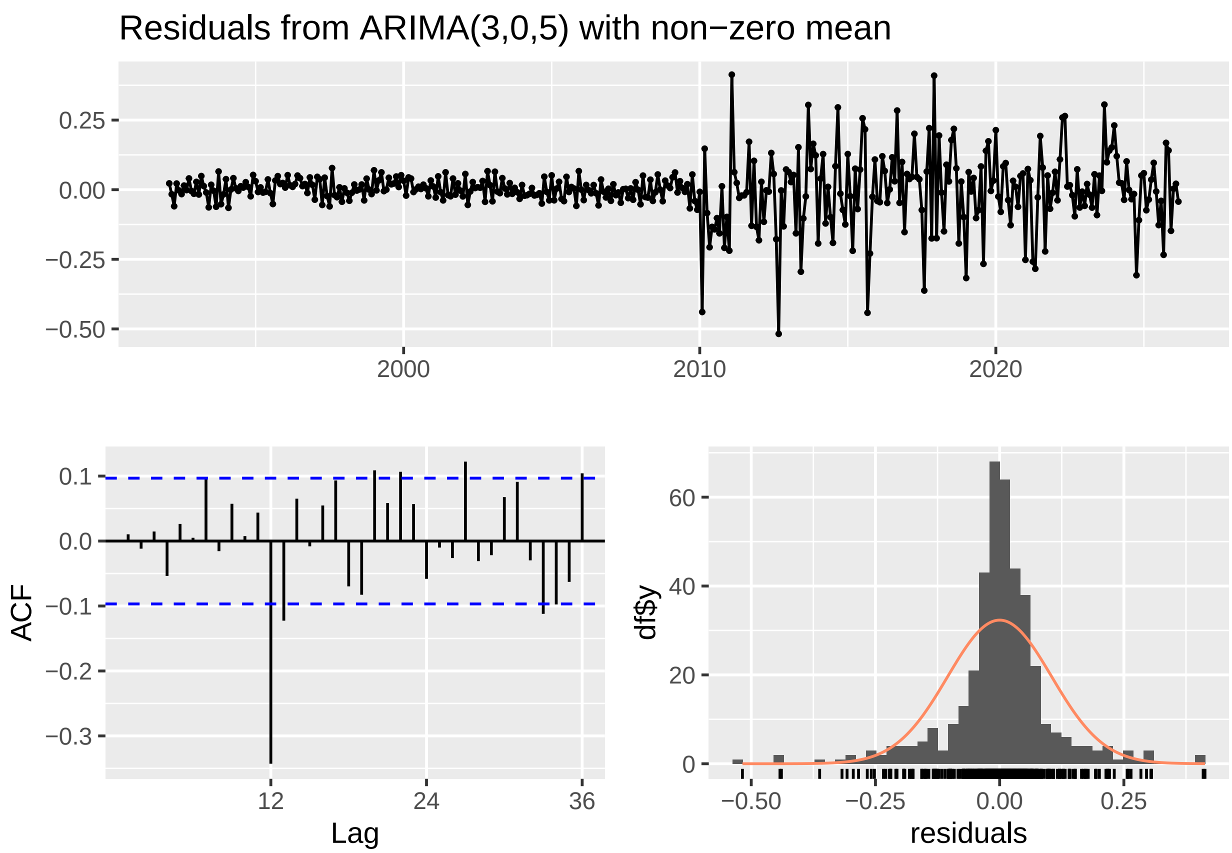
5. Write the ARIMA(p,d,q) model for the chosen series.

```
modele_arima <- Arima(seasonlog_series, order = c(3, 0, 5))
summary(modele_arima)
```

```
## Series: seasonlog_series
## ARIMA(3,0,5) with non-zero mean
##
## Coefficients:
##           ar1     ar2     ar3     ma1      ma2      ma3      ma4      ma5
##       -0.6946  0.5323  0.6687  0.9520  -0.1267  -0.7457  -0.3697  -0.3096
## s.e.   0.0788  0.1098  0.0741  0.0834   0.1252   0.1045   0.0630   0.0588
##         mean
##       0.0086
## s.e.  0.0043
##
## sigma^2 = 0.01118:  log likelihood = 342.88
## AIC=-665.77   AICc=-665.22   BIC=-625.61
##
## Training set error measures:
##                         ME      RMSE        MAE  MPE MAPE     MASE       ACF1
## Training set -2.461424e-05 0.1045639 0.06655398 -Inf  Inf 0.556558 0.01039516
```

Tracé des résidus

```r
checkresiduals(modele_arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,5) with non-zero mean
## Q* = 90.639, df = 16, p-value = 1.907e-12
##
## Model df: 8.   Total lags used: 24
```