Housing1.R

alatour

2019-10-31

```
rm(list = ls())
options(OutDec = ",")
require(astsa)
```

```
## Loading required package: astsa
```

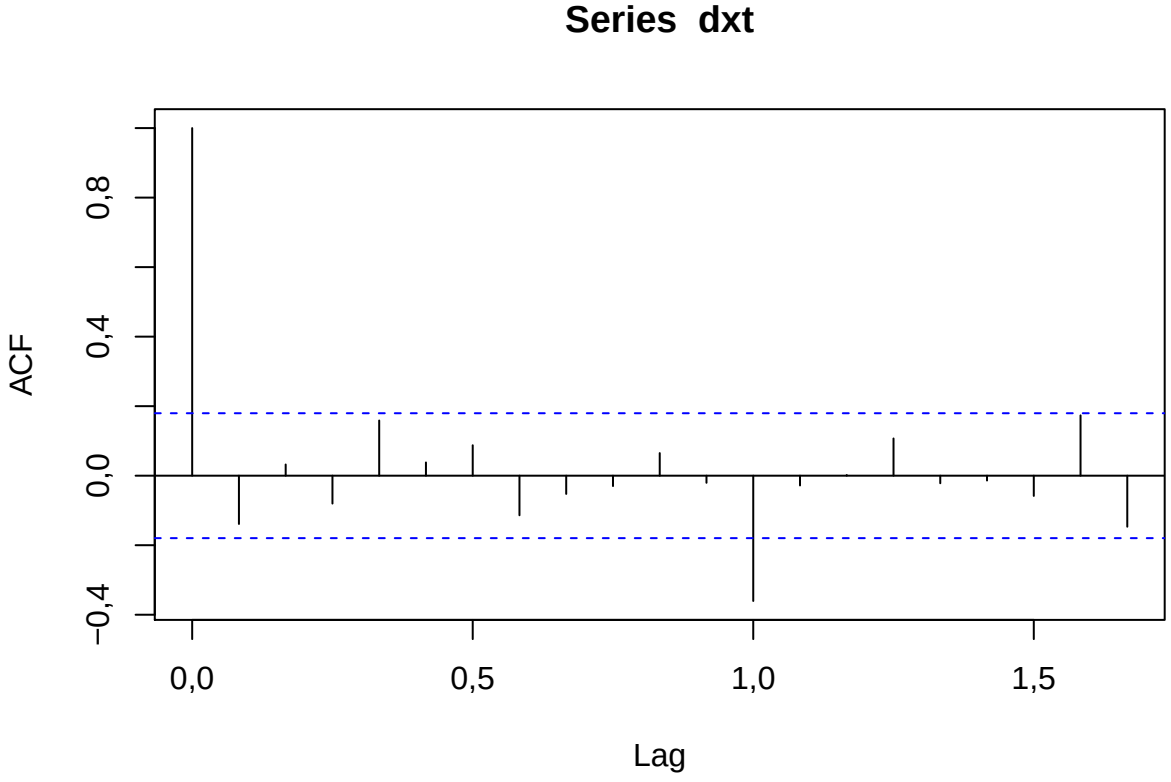
```
# Monthly Sales of U.S. Houses (in thousands of units),
# January 1965 to December 1975
sales <- c(38, 44, 53, 49, 54, 57, 51, 58, 48, 44, 42, 37,
           42, 43, 53, 49, 49, 40, 40, 36, 29, 31, 26, 23,
           29, 32, 41, 44, 49, 47, 46, 47, 43, 45, 34, 31,
           35, 43, 46, 46, 43, 41, 44, 47, 41, 40, 32, 32,
           34, 40, 43, 42, 43, 44, 39, 40, 33, 32, 31, 28,
           34, 29, 36, 42, 43, 44, 44, 48, 45, 44, 40, 37,
           45, 49, 62, 62, 58, 59, 64, 62, 50, 52, 50, 44,
           51, 56, 60, 65, 64, 63, 63, 72, 61, 65, 51, 47,
           54, 58, 66, 63, 64, 60, 53, 52, 44, 40, 36, 28,
           36, 42, 53, 53, 55, 48, 47, 43, 39, 33, 30, 23,
           29, 33, 44, 54, 56, 51, 51, 53, 45, 45, 44, 38)
# Monthly U.S. Housing Starts of Privately Owned Single-Family Structures (in thousands of units),
#January 1965 to Dcccmber 1975, ii
starts <- c(52.149, 47.205, 82.150, 100.931, 98.408, 97.351,
            96.489, 88.830, 80.876, 85.750, 72.351, 61.198,
            46.561, 50.361, 83.236, 94.343, 84.748, 79.828,
            69.068, 69.362, 59.404, 53.530, 50.212, 37.972,
            40.157, 40.274, 66.592, 79.839, 87.341, 87.594,
            82.344, 83.712, 78.194, 81.704, 69.088, 47.026,
            45.234, 55.431, 79.325, 97.983, 86.806, 81.424,
            86.398, 82.522, 80.078, 85.560, 64.819, 53.847,
            51.300, 47.909, 71.941, 84.982, 91.301, 82.741,
            73.523, 69.465, 71.504, 68.039, 55.069, 42.827,
            33.363, 41.367, 61.879, 73.835, 74.848, 83.007,
            75.461, 77.291, 75.961, 79.393, 67.443, 69.041,
            54.856, 58.287, 91.584, 116.013, 115.627, 116.946,
            107.747, 111.663, 102.149, 102.882, 92.904, 80.362,
            76.185, 76.306, 111.358, 119.840, 135.167, 131.870,
            119.078, 131.324, 120.491, 116.990, 97.428, 73.195,
            77.105, 73.560, 105.136, 120.453, 131.643, 114.822,
            114.746, 106.806, 85.504, 86.004, 70.488, 46.767,
            43.292, 57.593, 76.946, 102.237, 96.340, 99.318,
            90.715, 79.782, 73.443, 69.460, 57.898, 41.041,
            39.791, 39.959, 62.498, 77.777, 92.782, 90.284,
            92.782, 90.655, 84.517, 93.826, 71.646, 55.650)

sales <- ts(sales,start=c(1965,1),frequency=12)
starts <- ts(starts,start=c(1965,1),frequency=12)
# Voir les diapos 4 à 7 du document Housing.pdf
# Estimation des paramètres
```

```r
model1 <- arima(sales,order=c(0,1,1),seasonal=list(order=c(0,1,1),period=12),method="CSS")
model1
```

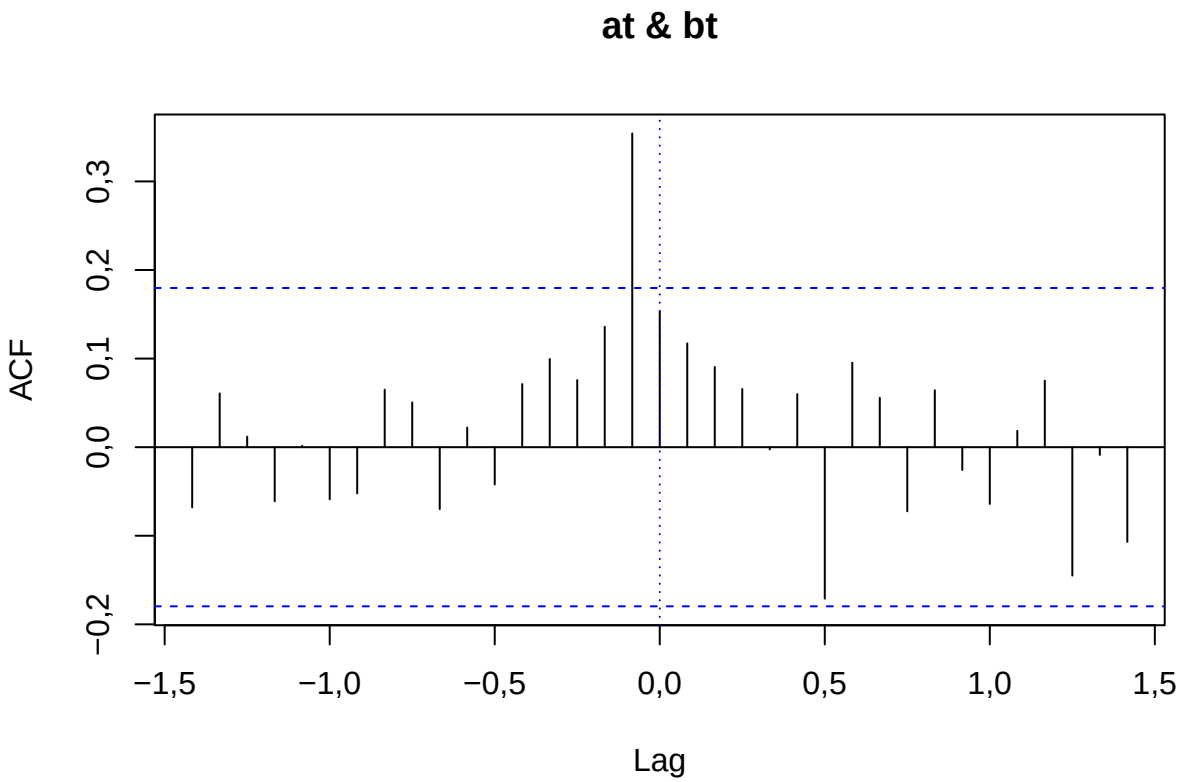
```
##
## Call:
## arima(x = sales, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "CSS")
##
## Coefficients:
##           ma1     sma1
##       -0,2123  -0,7274
## s.e.   0,0854   0,0670
##
## sigma^2 estimated as 15,75:  part log likelihood = -332,87
```

```r
# Récupération des valeurs numériques des paramètres estimés
theta <- model1$coef[1]
Theta <- model1$coef[2]
# On travaille avec les séries différenciées
dxt <- diff(diff(sales,lag=12))
dyt <- diff(diff(starts,lag=12))
acf(dxt)
```




```r
# Filtrage de la série en entrée
at <- filter(dxt, filter=c(rep(0,11),-Theta),method="recursive",init=rep(0,12))
at <- filter(at, filter=c(-theta),method="recursive")
# Filtrage de la série en sortie
bt <- filter(dyt, filter=c(rep(0,11),-Theta),method="recursive",init=rep(0,12))
bt <- filter(bt, filter=c(-theta),method="recursive")
# Corrélation entre les deux séries filtrées
```

```r
crossCorr <- ccf(at,bt)
abline(v=0,lty=3,col="blue")
```

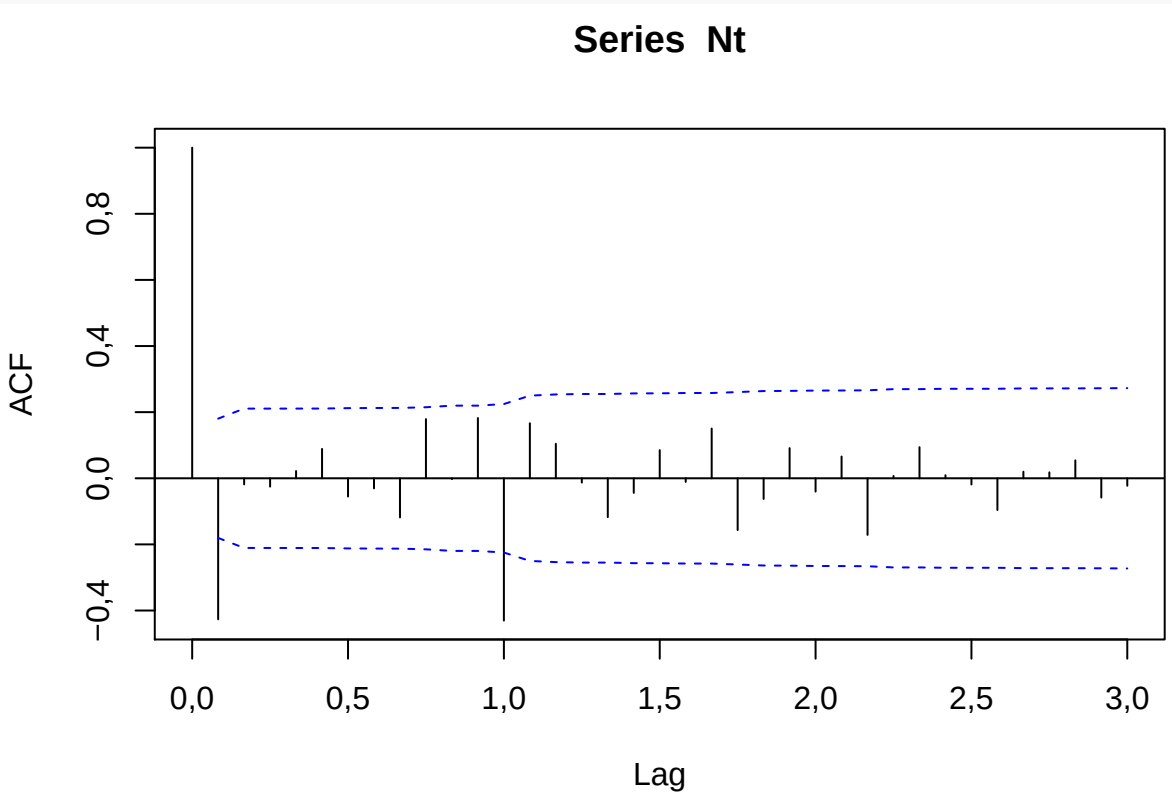



```r
# Esstimation des poids v[k] de la fonction de transfert.
temp <- crossCorr$acf * sd(bt) / sd(at)
cbind(crossCorr$acf,temp)
```

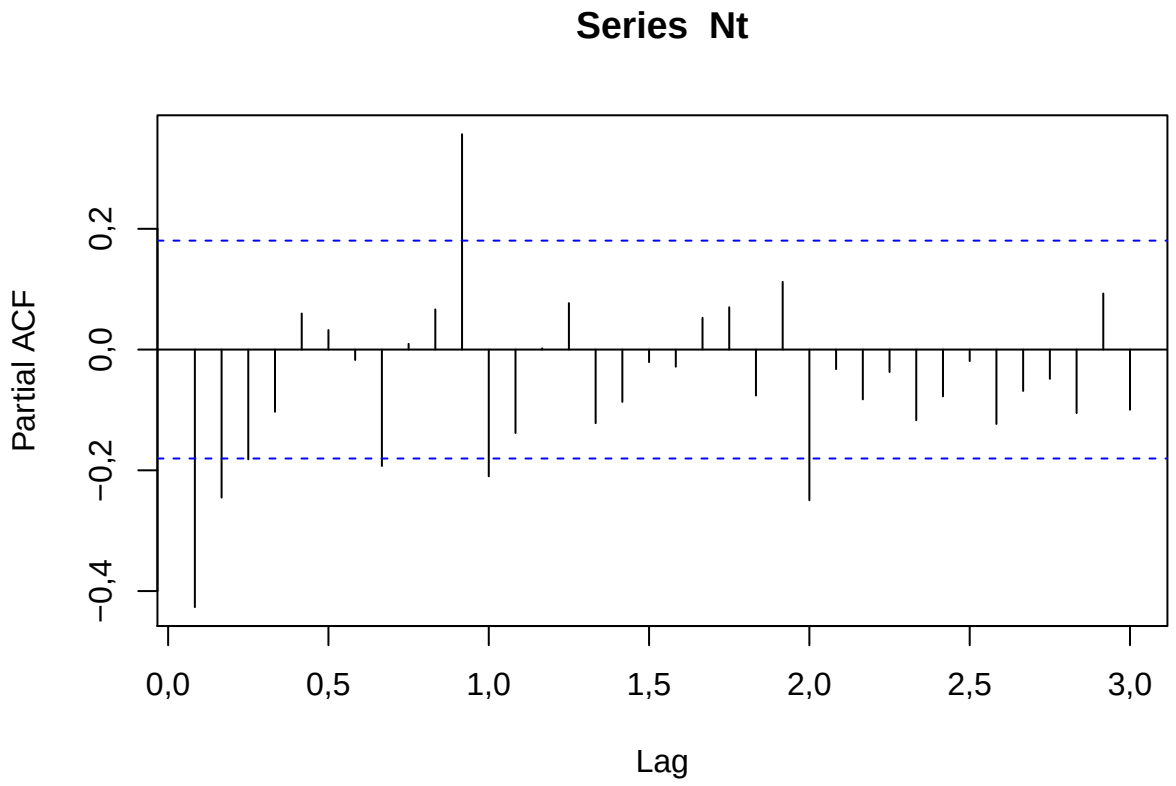
```
##                             temp
##  [1,] -0,068024740 -0,121263105
##  [2,]  0,060710866  0,108225158
##  [3,]  0,011888791  0,021193345
##  [4,] -0,061087658 -0,108896838
##  [5,]  0,001507076  0,002686563
##  [6,] -0,058876548 -0,104955243
##  [7,] -0,052202333 -0,093057571
##  [8,]  0,064924140  0,115735875
##  [9,]  0,050583452  0,090171700
## [10,] -0,069912675 -0,124628600
## [11,]  0,022111614  0,039416880
## [12,] -0,042197873 -0,075223295
## [13,]  0,071248787  0,127010395
## [14,]  0,099515915  0,177400295
## [15,]  0,075730051  0,134998843
## [16,]  0,135878836  0,242222015
## [17,]  0,354190355  0,631391203
## [18,]  0,153208385  0,273114232
## [19,]  0,117226264  0,208971336
## [20,]  0,090556256  0,161428517
## [21,]  0,065659317  0,117046425
```

```
## [22,] -0,002453296 -0,004373324
## [23,]  0,059853494  0,106696777
## [24,] -0,170977792 -0,304790551
## [25,]  0,095377469  0,170022967
## [26,]  0,055789769  0,099452649
## [27,] -0,072381406 -0,129029439
## [28,]  0,064266450  0,114563456
## [29,] -0,025757719 -0,045916545
## [30,] -0,064065633 -0,114205474
## [31,]  0,018435933  0,032864491
## [32,]  0,075094864  0,133866538
## [33,] -0,144997269 -0,258476830
## [34,] -0,008754635 -0,015606295
## [35,] -0,106927940 -0,190613211
```

```r
# Estimation des paramètres omega_0 et delta de la fonction de transfert.
# Voir diapo 35 du fichier transfert.pdf
omega_0 <- temp[17]
delta <- temp[16] / temp[17]
# Calcul du bruit qui sera modélisé par un processus ARMA
temp_t <-
  filter(omega_0 * lag(dxt, k = -1),
         filter = c(delta),
         method = "recursive")
Nt <- dyt - temp_t
# Identification du processus
acf(Nt, ci.type = "ma",lag.max=36)
```



```r
pacf(Nt,lag.max=36)
```




```r
# Estimation des paramètres
modelNt <- arima(Nt, order = c(0, 0, 1),seasonal=list(order=c(0,0,1),period=12))
# Récupération des valeurs numériques des paramètres estimés
theta.N <- as.numeric(modelNt$coef[1])
Theta.N <- as.numeric(modelNt$coef[2])
# Définition de la fonction f qui calcule les résidus du modèle et
# qui retourne la somme des carrés qui elle, est à minimiser
f <- function(v) {
  omega_0 <- v[1]
  delta <- v[2]
  theta.N <- v[3]
  Theta.N <- v[4]
  at <- lag(dxt, -1) * omega_0
  bt <- filter(at, filter = c(delta), method = "recursive")
  ct <- dyt - bt
  dt <- filter(ct, filter = c(-theta.N), method = "recursive")
  dt <- filter(dt, filter=c(rep(0,11),-Theta.N),method="recursive",init=rep(0,12))
  SS <- sum(dt ^ 2, na.rm = TRUE)
  return(SS)}
v <- c(omega_0 = omega_0, delta = delta, theta.N = theta.N, Theta.N=Theta.N)
(par.estim <- optim(v, f, method = "BFGS", hessian = TRUE))
```

```
## $par
##    omega_0      delta    theta.N    Theta.N
##  0,8015312  0,4005373 -0,6303550 -0,7849394
##
## $value
```

```
## [1] 4315,362
##
## $counts
## function gradient
##       55       11
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##           omega_0      delta   theta.N    Theta.N
## omega_0 9039,9605  7834,5880  4478,916  -722,2411
## delta   7834,5880 12712,2477  3122,477  -114,0215
## theta.N 4478,9156  3122,4773 10589,432 -1065,9669
## Theta.N -722,2411  -114,0215 -1065,967 19279,8101
```

```
# Pour connaître la structure de l'objet par.estim
str(par.estim)
```

```
## List of 6
##  $ par        : Named num [1:4] 0,802 0,401 -0,63 -0,785
##   ..- attr(*, "names")= chr [1:4] "omega_0" "delta" "theta.N" "Theta.N"
##  $ value      : num 4315
##  $ counts     : Named int [1:2] 55 11
##   ..- attr(*, "names")= chr [1:2] "function" "gradient"
##  $ convergence: int 0
##  $ message    : NULL
##  $ hessian    : num [1:4, 1:4] 9040 7835 4479 -722 7835 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "omega_0" "delta" "theta.N" "Theta.N"
##   .. ..$ : chr [1:4] "omega_0" "delta" "theta.N" "Theta.N"
```

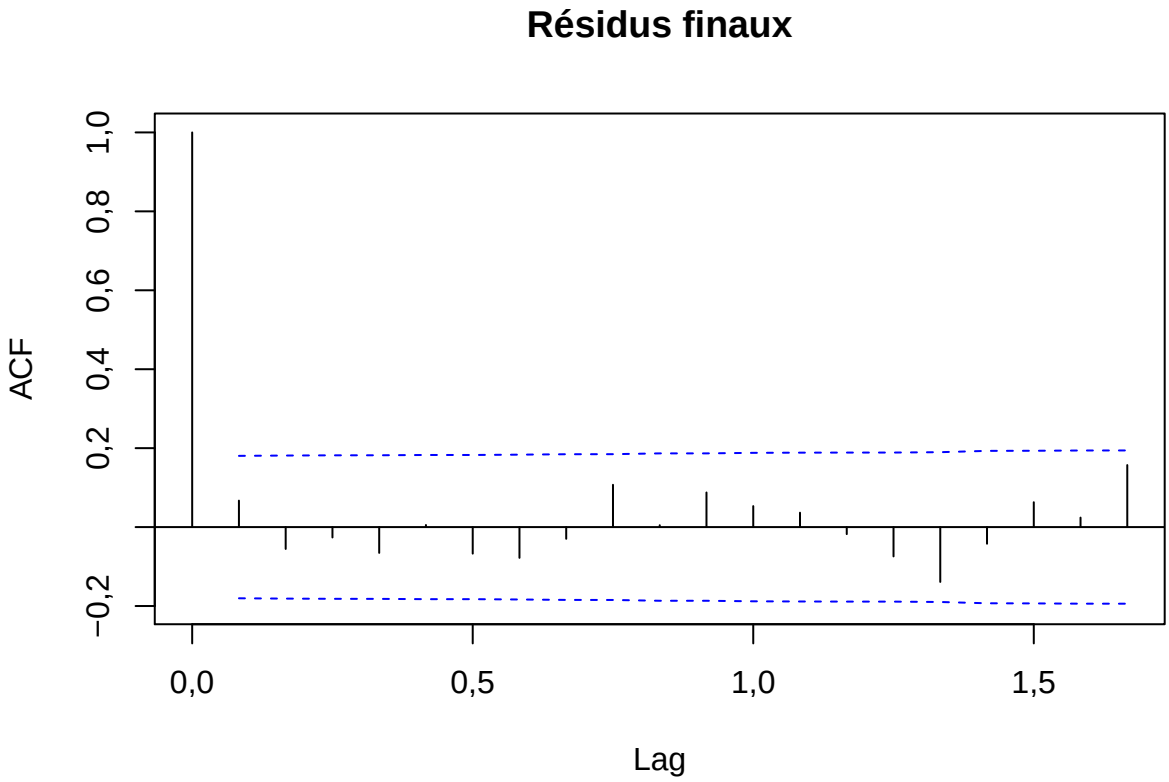
```
# Pour obtenir la matrice des covariances des estimateurs
solve(par.estim$hessian)
```

```
##               omega_0         delta       theta.N       Theta.N
## omega_0  2,825244e-04 -1,561684e-04 -7,288104e-05  5,630523e-06
## delta   -1,561684e-04  1,711454e-04  1,518543e-05 -3,998472e-06
## theta.N -7,288104e-05  1,518543e-05  1,211906e-04  4,060153e-06
## Theta.N  5,630523e-06 -3,998472e-06  4,060153e-06  5,227949e-05
```

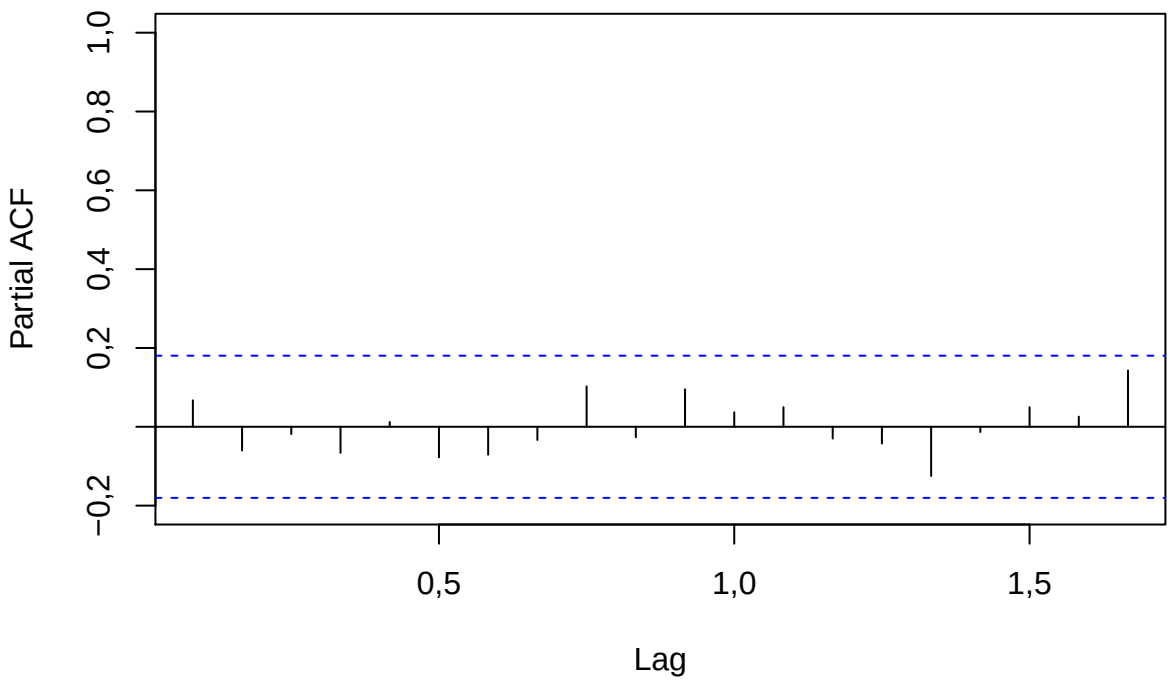
```
# Calcul des résidus du modèle final
omega_0 <- par.estim$par[1]
delta <- par.estim$par[2]
theta.N <- par.estim$par[3]
Theta.N <- par.estim$par[4]
at <- lag(dxt, -1) * omega_0
bt <- filter(at, filter = c(delta), method = "recursive")
ct <- dyt - bt
dt <- filter(ct, filter = c(-theta.N), method = "recursive")
dt <- filter(dt, filter=c(rep(0,11),-Theta.N),method="recursive",init=rep(0,12))
# Estimation de la variance des epsilon_t = dt
(S2 <- sum(dt ^ 2, na.rm = TRUE) / length(dt))
```

```
## [1] 36,57086
```

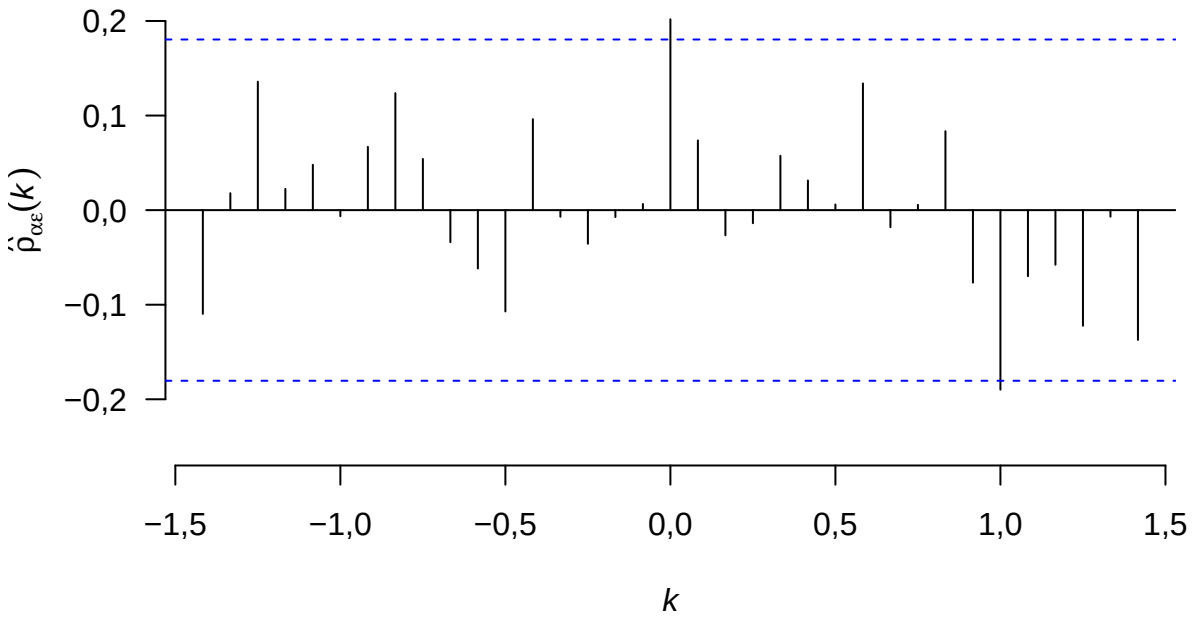
```r
# Vérification que epsilon_t = dt est un bruit blanc
acf(dt, na.action = na.pass, ci.type = "ma",main="Résidus finaux")
```




```r
pacf(dt, na.action = na.pass, ylim=c(-0.2,1),main="Résidus finaux")
```




```
# Vérifiation que l'entrée et les résidus finaux ne sont pas corrélés.
crossCorr <-
  ccf(
    dxt,
    dt,
    ylim = c(-0.25, 0.25),
    main = "Ventes -> Résidus finaux",
    ylab = expression(italic(hat(rho)[alpha * epsilon](k))),
    xlab = expression(italic(k)),
    las = 1,
    frame = FALSE
  )
```


Ventes −> Résidus finaux
0,2
0,1
0,0
−0,1
−0,2
$\hat{\rho}_{\alpha\varepsilon}(k)$
−1,5
−1,0
−0,5
0,0
0,5
1,0
1,5
k