STA202 - Séries temporelles

Rapport de projet

Anthony Kalaydjian - Mathieu Occhipinti

2023-03-02

Contents

1 Introduction

Au cours du dernier siècle, l'aggravation de la situation écologique a conduit à une prise de conscience de l'importance de la qualité de l'air. Ainsi, la concentration de certains gaz dans l'air a été étudiée de manière approfondie afin de comprendre les effets de la pollution sur l'environnement et la santé humaine. Les effets néfastes de la pollution sont par exemple visibles auprès des joueurs d'échecs, dont les performances diminuent lorsque la qualité de l'air de leur environnement diminue.

On s'intérèsse ainsi, dans ce document, à l'étude de l'évolution de la concentration de certains gaz dans l'air au cours du temps. Les données que nous allons utiliser proviennent d'une banque de datasets mis à disposition par l'université d'Irvine en Californie. Elles comportent ainsi les mesures de la concentration de certains gaz dans une ville d'Italie, sur une période de 1 an et avec un pas de 1h00.

2 Prétraitement et mise en forme des données

2.1 Prétraitement et gestion des données manquantes

Comme expliqué précédemment, les données que nous allons utiliser et qui sont consignées dans un fichier .csv représentent l'évolution de la concentration de certains gaz au cours du temps. Ces mesures sont en fait des moyennes de ce qu'à mesuré le capteur sur 1h. Le dataset présente également l'évolution de la température (en degrés Farenheit), de l'humidité relative (en %) ainsi que de l'humidité absolue. Toutes ces données sont donc indéxées par la date et l'heure de la mesure.

Un premier problème est que certaines données sont manquantes. On peut voir celà dans le dataset, où certaines valeurs associées à nos variables vallent -200. Plusieurs techniques existent pour pallier ce manque de données, dont le fait de ne pas prendre en compte les valeurs manquantes ou bien de les remplacer par la moyenne des autres valeurs. On effectuera cette dernière technique, qui semble mieux correspondre à l'étude des séries temporelles. On évitera néanmoins les variables pour lesquelles trop de données sont manquantes.

```
# importation des données
air_data <- read.table("AirQualityUCI.csv",header=T,sep=";")

# resize
air_data <- air_data[2:9357,3:15]

i <- c(1:length(air_data))

# Remplacement des -200 par NA.
air_data[, i] <- apply(air_data[, i], 2, function(x) (gsub(-200, NA, x)))

# Comptage du nombre de NA par colonne.
na_count <-sapply(air_data, function(y) sum(length(which(is.na(y)))))

# Conversion des chaînes de caractère en nombres, en respectant la nomenclature française
# des nombres à virgule.
air_data[, i] <- apply(air_data[, i], 2,
                       function(x) as.numeric(as.character(gsub(",", ".", x))))

# Remplacement des NA par la moyenne.
```

```
air_data[, i] <- apply(air_data[, i], 2,
                       function(x) replace(x, is.na(x), mean(x, na.rm = TRUE)))
```

Les données sont bien du type floatant :

```
str(air_data)
```

```
## 'data.frame':    9356 obs. of  13 variables:
##  $ CO.GT.       : num  2 2.2 2.2 1.6 1.2 ...
##  $ PT08.S1.CO.  : num  1292 1402 1376 1272 1197 ...
##  $ NMHC.GT.     : num  112 88 80 51 38 31 31 24 19 14 ...
##  $ C6H6.GT.     : num  9.4 9 9.2 6.5 4.7 3.6 3.3 2.3 1.7 1.3 ...
##  $ PT08.S2.NMHC.: num  955 939 948 836 750 690 672 609 561 527 ...
##  $ NOx.GT.      : num  103 131 172 131 89 ...
##  $ PT08.S3.NOx. : num  1174 1140 1092 1205 1337 ...
##  $ NO2.GT.      : num  92 114 122 116 96 ...
##  $ PT08.S4.NO2. : num  1559 1555 1584 1490 1393 ...
##  $ PT08.S5.O3.  : num  972 1074 1203 1110 949 ...
##  $ T            : num  13.3 11.9 11 11.2 11.2 11.3 10.7 10.7 10.3 10.1 ...
##  $ RH           : num  47.7 54 60 59.6 59.2 56.8 60 59.7 60.2 60.5 ...
##  $ AH           : num  0.726 0.75 0.787 0.789 0.785 ...
```

Les NA ont bien été remplacés :

```
summary(air_data)
```

```
##      CO.GT.        PT08.S1.CO.      NMHC.GT.         C6H6.GT.
##  Min.   : 0.100   Min.   : 647   Min.   :   7.0   Min.   : 0.10
##  1st Qu.: 1.200   1st Qu.: 941   1st Qu.: 218.9   1st Qu.: 4.60
##  Median : 2.153   Median :1075   Median : 218.9   Median : 8.60
##  Mean   : 2.153   Mean   :1100   Mean   : 218.9   Mean   :10.08
##  3rd Qu.: 2.600   3rd Qu.:1221   3rd Qu.: 218.9   3rd Qu.:13.60
##  Max.   :11.900   Max.   :2040   Max.   :1189.0   Max.   :63.70
##  PT08.S2.NMHC.       NOx.GT.        PT08.S3.NOx.       NO2.GT.
##  Min.   : 383.0   Min.   :   2.0   Min.   : 322.0   Min.   :  2.0
##  1st Qu.: 742.8   1st Qu.: 112.0   1st Qu.: 666.0   1st Qu.: 86.0
##  Median : 923.0   Median : 229.0   Median : 818.0   Median :113.1
##  Mean   : 939.1   Mean   : 246.9   Mean   : 835.5   Mean   :113.1
##  3rd Qu.:1105.0   3rd Qu.: 284.0   3rd Qu.: 960.0   3rd Qu.:133.0
##  Max.   :2214.0   Max.   :1479.0   Max.   :2683.0   Max.   :340.0
##   PT08.S4.NO2.   PT08.S5.O3.         T               RH              AH
##  Min.   : 551   Min.   : 221   Min.   :-1.90   Min.   : 9.20   Min.   :0.1847
##  1st Qu.:1242   1st Qu.: 742   1st Qu.:12.00   1st Qu.:36.60   1st Qu.:0.7460
##  Median :1456   Median : 983   Median :18.30   Median :49.23   Median :1.0154
##  Mean   :1456   Mean   :1023   Mean   :18.32   Mean   :49.23   Mean   :1.0256
##  3rd Qu.:1662   3rd Qu.:1255   3rd Qu.:24.10   3rd Qu.:61.90   3rd Qu.:1.2963
##  Max.   :2775   Max.   :2523   Max.   :44.60   Max.   :88.70   Max.   :2.2310
```

Sur l'ensemble des données que l'on a, on peut voir qu'il manque beaucoup de données pour les gaz CO.GT, NMHC.GT, NOx.GT et NO2.GT avec respectivement 1683, 8444, 1640 et 1643 données manquantes sur un total de 9357 valeurs. On évitera ainsi de porter l'étude sur ces données. Pour les autres colonnes, il ne nous

manque que 366 ou 367 valeurs, ce qui représente 4% des valeurs. Ce n'est pas parfait, mais suffisamment raisonnable pour mener l'étude. Ces valeurs manquantes peuvent être dues à des pannes générales du capteur, qui ont provoqué le même nombre de valeurs manquantes pour chaque colonne.

```
print(na_count)
```

```
##        CO.GT.   PT08.S1.CO.      NMHC.GT.      C6H6.GT. PT08.S2.NMHC.
##          1683           366          8443           366           366
##       NOx.GT.  PT08.S3.NOx.       NO2.GT.  PT08.S4.NO2.   PT08.S5.O3.
##          1639           366          1642           366           366
##             T            RH            AH
##           366           366           366
```

Les séries que nous allons étudier sont donc les suivantes :
PT08.CO, C6H6.GT, PT08.NMHC, PT08.NOx, PT08.NO2 et PT08.O3.

```
air_data <- air_data[, -c(1, 3, 5, 7)]
```

2.2 Création des séries temporelles

```
date1 <- strptime("03/10/2004 18:00:00", "%m/%d/%Y %H:%M:%S")
date2 <- strptime("04/04/2005 14:00:00", "%m/%d/%Y %H:%M:%S")
Date_air <- seq.POSIXt(date1,date2, by = "1 hour")

ts_PT08.CO <- xts(air_data$PT08.S1.CO., order.by=Date_air)
ts_C6H6.GT <- xts(air_data$C6H6.GT., order.by=Date_air)
ts_PT08.NMHC <- xts(air_data$PT08.S2.NMHC., order.by=Date_air)
ts_PT08.NOx <- xts(air_data$PT08.S3.NOx., order.by=Date_air)
ts_PT08.NO2 <- xts(air_data$PT08.S4.NO2.,order.by=Date_air)
ts_PT08.O3 <- xts(air_data$PT08.S5.O3.,order.by=Date_air)
```

3 Analyse descriptive des données

3.1 Corrélation des variables observées

La figure 1 montre la matrice de correlation de nos variables. Elle montre ainsi que la température et l'humidité absolue ne sont visiblement corrélées qu'avec les émissions de NO2(GT). Hormis cela, aucun des trois paramètres que sont la température, l'humidité relative et l'humidité absolue ne semble être corrélés avec les concentrations de gaz mesurés dans l'air. Cette matrice nous montre également une forte corrélation positive entre les autres gaz, ce qui peut montrer que leur comportement est similaire.

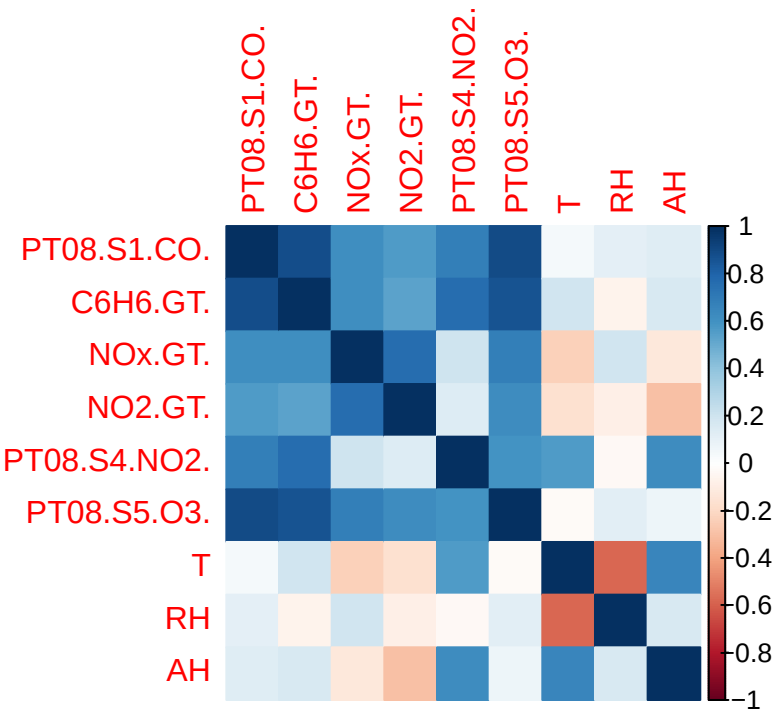


Figure 1: Matrice de corrélation des variables observées

3.2 Analyse en série temporelle

L'observation des différentes concentration moyennes périodiques, ainsi que des séries temporelles elles-mêmes montre un comportement très similaires entre l'ensemble des gas, si ce n'est pour le Nox.GT dont le comportement varie. Les émissions hebdomadaires semblent ainsi inversées, avec un pic d'émissions les lundis et dimanches contre des pics d'émission en milieu de semaine pour les autres gas. Il en est de même pour les émissions horaires, avec plus d'émissions tôt le matin (vers 5h) contre des émissions plus présentes au milieu de la journée avec les autres gas. Ceci pourrait être expliqué par le fait que les émissions de ce gaz soient majoritairement dues aux émissions de transports. Ainsi, les livraisons des magasins par les camions transporteurs, qui se font en début de semaine et tôt le matin, peuvent expliquer ces émissions différentes.

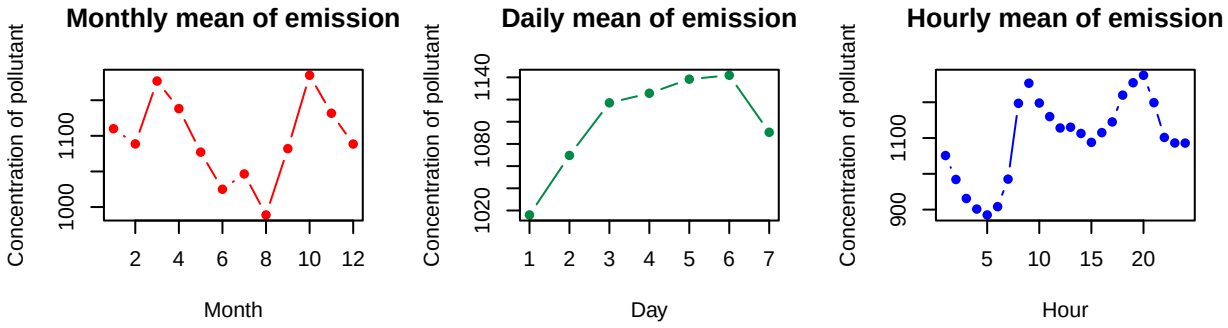


Figure 2: Concentrations moyennes périodiques PT08.CO

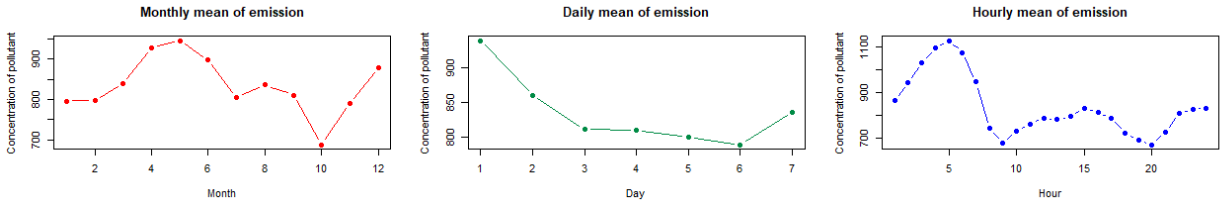


Figure 3: Concentrations moyennes périodiques NOx.GT

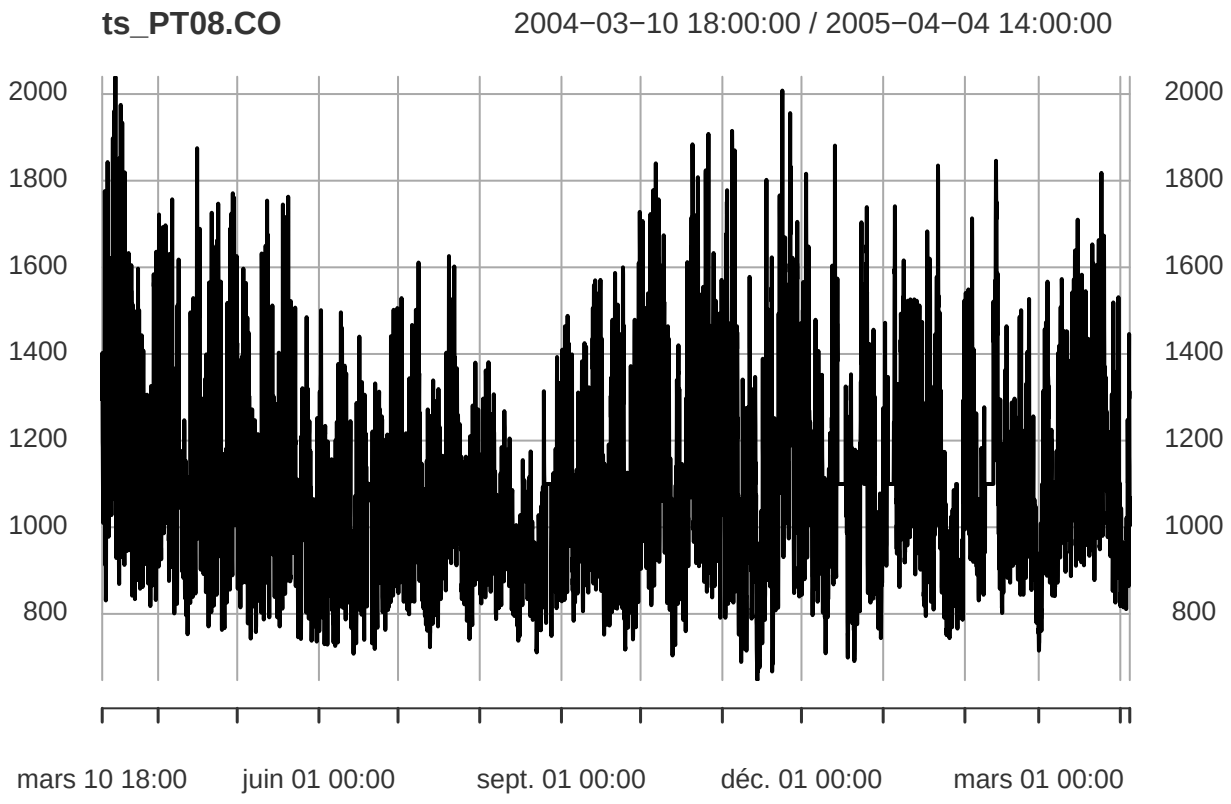
La plupart des gaz ayant un comportement similaire, on choisira par la suite de ne travailler que sur la série temporelle associée à la concentration de PT08.CO.

4 Modélisation des données

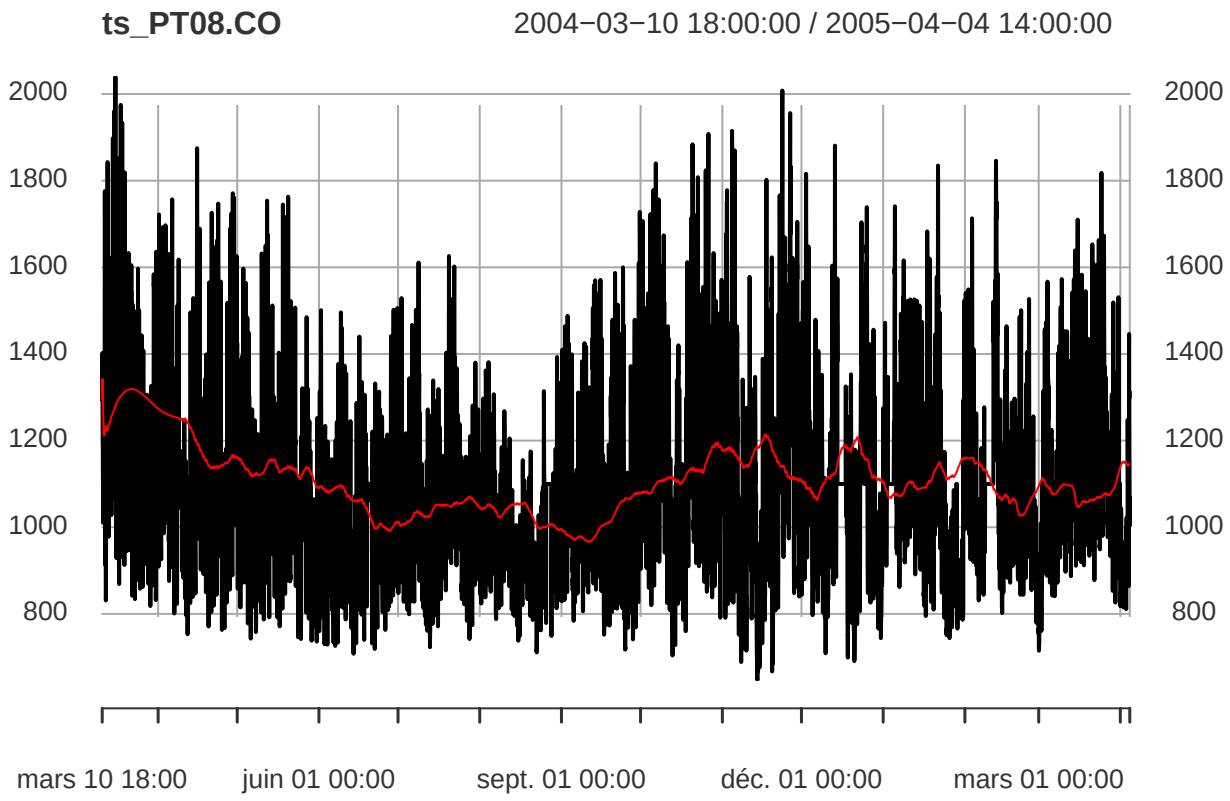
4.1 Tendances

L'analyse de la tendance de la série temporelle peut être faite à l'aide de la moyenne mobile. Un modèle approprié doit pouvoir faire abstraction du cycle journalier des données. Pour ce faire nous allons utiliser la fonction movavg de la librairie pracma. Nous allons passer comme argument n=720 : le nombre de points que l'on regarde avant pour prédire la suite (cela correspond à un moins dans notre cas d'étude). Le second argument est le type de moyenne mobile, nous avons choisi 'w' pour weighted. Cela signifie que plus le point est éloigné de notre point de prédiction, plus son poids dans le calcul de la moyenne mobile est faible. La décroissance des poids est linéaire. Voici, la tendance de notre série chronologique obtenue.

```
MA<-movavg(ts_PT08.CO,n=720,type='w')
MA<-xts(MA,order.by=Date_air)
plot(ts_PT08.CO)
```



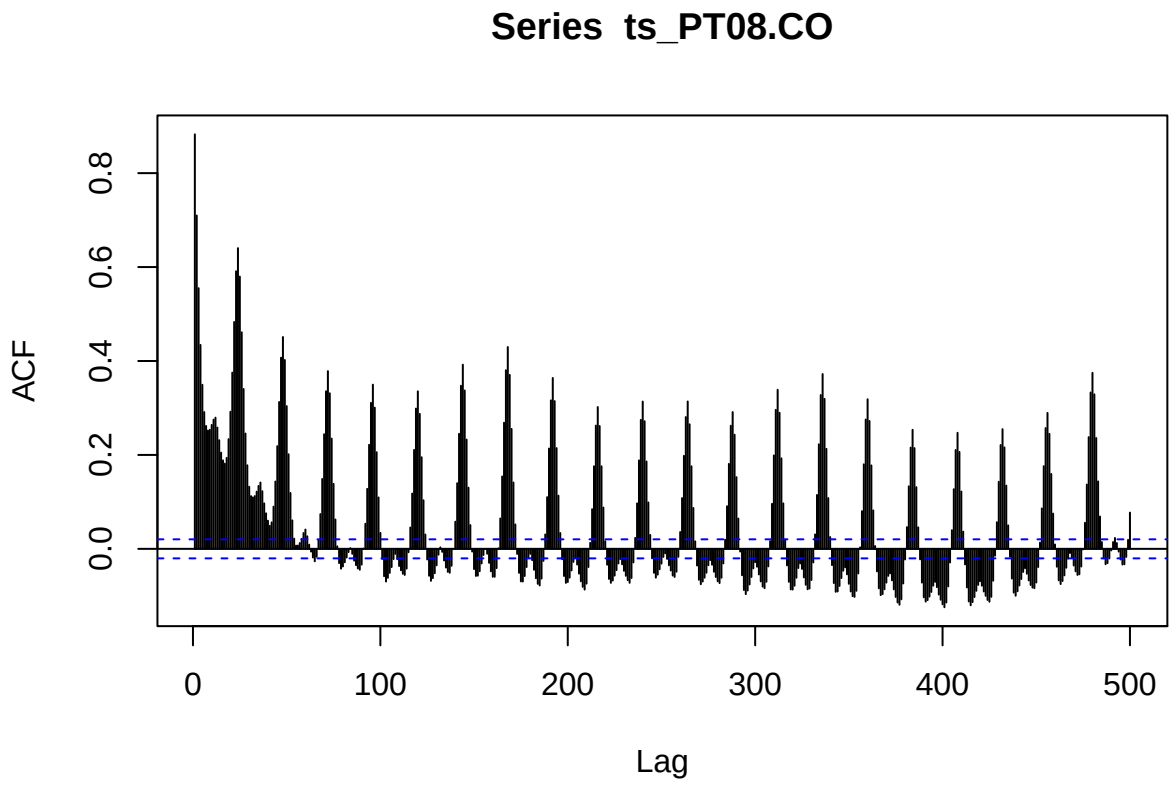

```
lines(MA,col='red',type='l')
```



4.2 Saisonnalité

Pour étudier la saisonnalité de nos données nous allons regarder l'autocorélogramme de notre série.

```
Acf(ts_PT08.CO,lag.max=500)
```



L'oscillation de l'ACF avec des pics réguliers toutes les 24h (et même 12h) montre l'existence d'une saisonnalité journalière dans nos données. L'existance de cette saisonnalité dément la stationnarité de la série temporelle. Ceci peut également être vérifié en utilisant le test KPSS (Kwiatkowski–Phillips–Schmidt–Shin), qui teste l'hypothèse nulle suivante :

$$(H_0) : \text{La série temporelle est stationnaire}$$

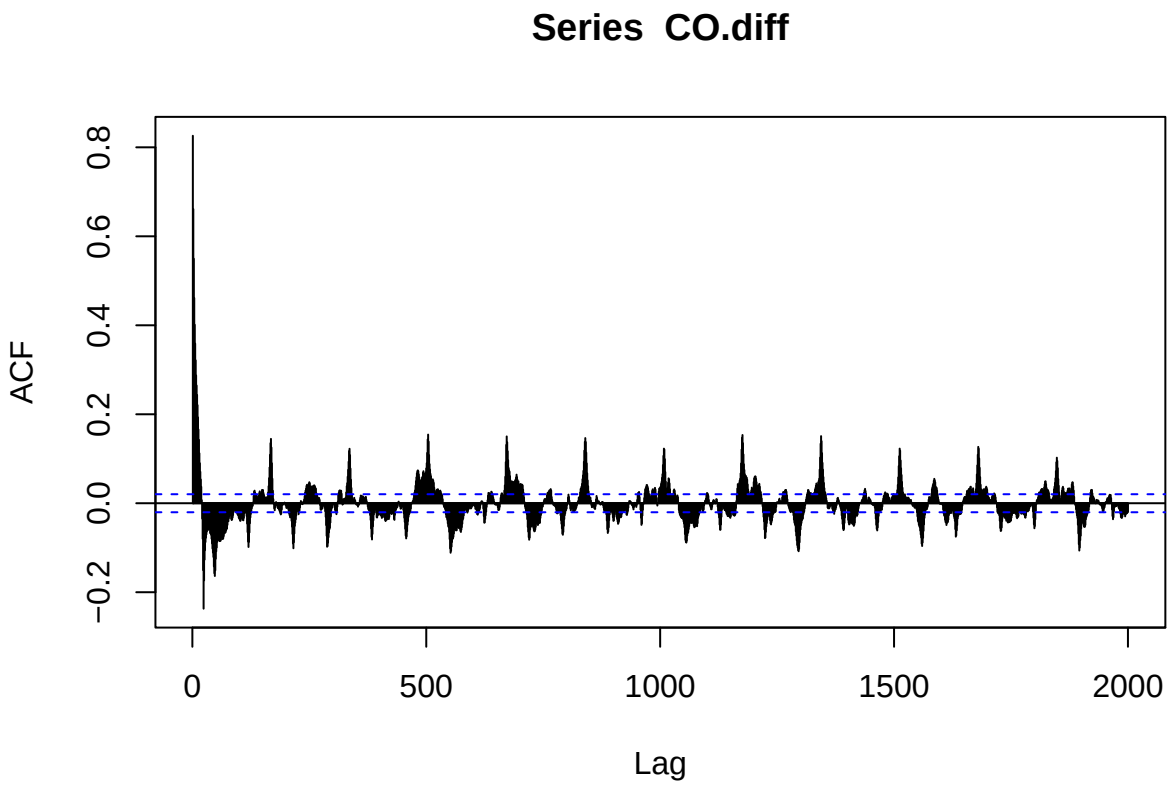
Le test KPSS affiche une valeur de test de 1.1339, ce qui est bien au dessus de toutes les valeurs critiques. On rejette ainsi l'hypothèse nulle. La série temporelle n'est donc pas stationnaire.

```
ur.kpss(ts_PT08.CO) %>%
  summary()
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 12 lags.
##
## Value of test-statistic is: 1.134
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

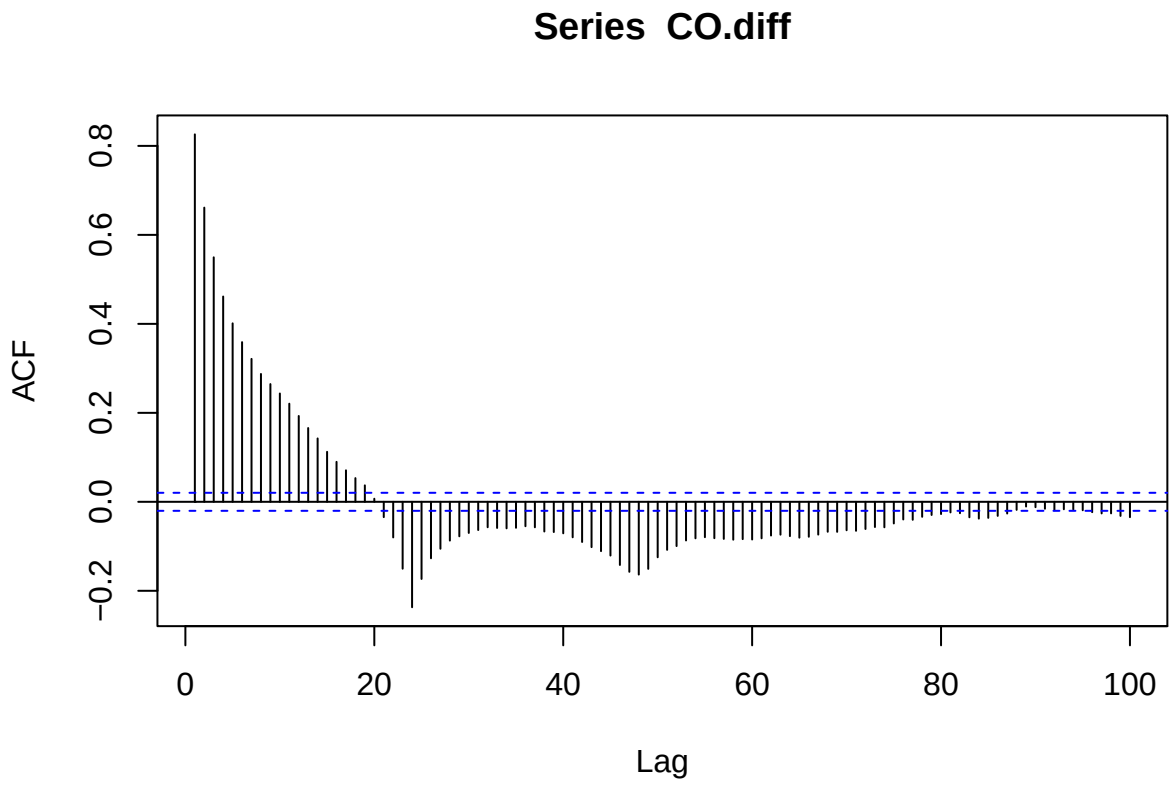
La stationnarité va pouvoir être atteinte en différenciant la série temporelle. La saisonnalité journalière nous pousse ainsi à différencier nos données avec un lag de 24 via l'opérateur $1 - L^{24}$. Nous affichons maintenant l'autocorrélogramme de notre série différenciée :

```
CO.diff<-na_mean(diff(ts_PT08.CO,lag=24,differences=1))

Acf(CO.diff,lag.max=2000)
```

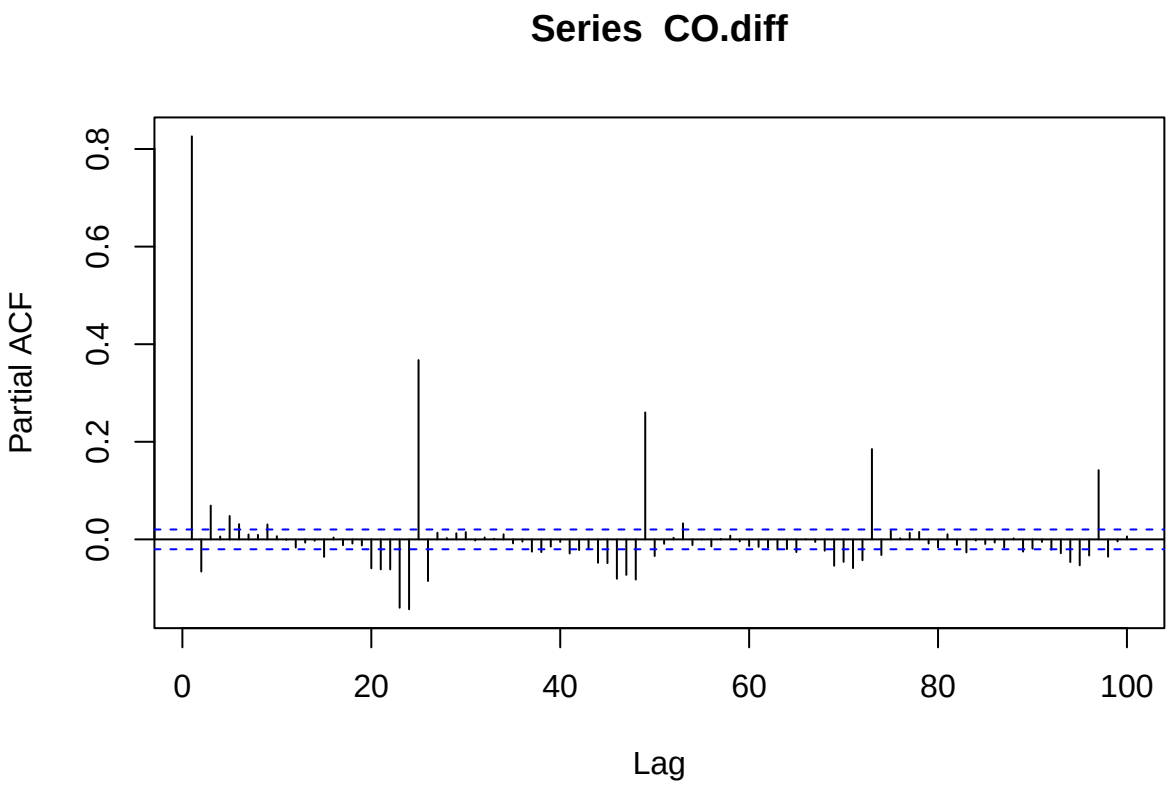


On obtient un graphique satisfaisant avec une décroissance exponentielle vers zéro. On souhaite maintenant déduire les plages de paramètres possibles pour un modèle SARIMA, (p,P,q,Q). Pour ce faire, on étudie l'autocorrélogramme et l'autocorrélogramme partiel de la série différenciée.

```
Acf(CO.diff,lag.max=100)
```




```
Pacf(CO.diff,lag=100)
```



Partie aléatoire ## Stationnarité

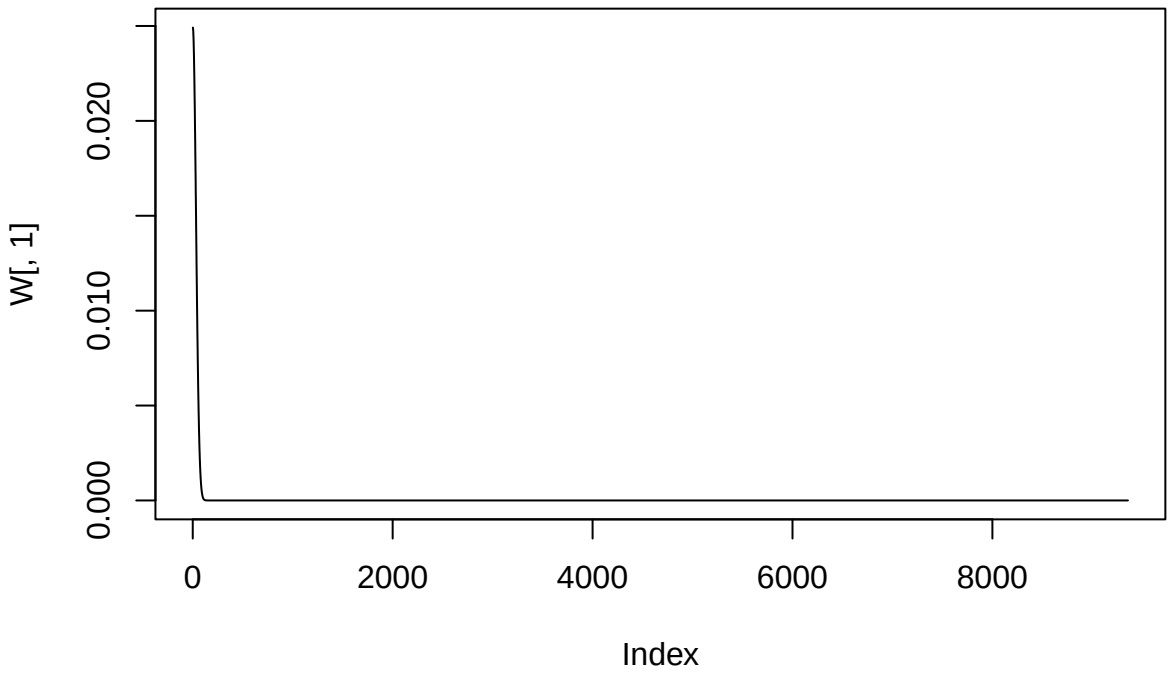
5 Simulation

```
# kernel
X<-(air_data$PT08.S1.CO. - mean(air_data$PT08.S1.CO.))/sd(air_data$PT08.S1.CO.)

head(X)
```

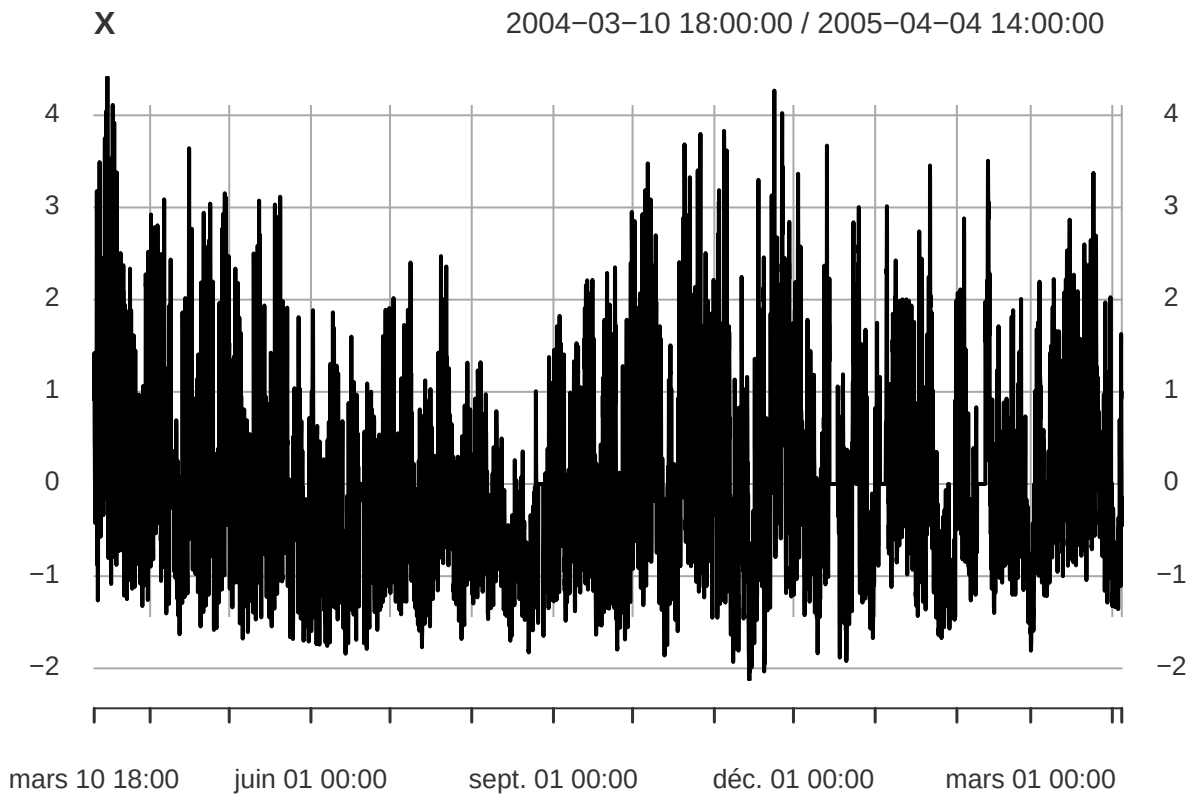
```
## [1] 0.9032349 1.4201861 1.2979976 0.8092437 0.4567770 0.4003823
```

```
X<-xts(X,order.by=Date_air)
n <- length(Date_air)
t <- c(1:n)
h<-2000
#h <- 100
x<-seq(1,max(t),length=n)
# test<-lapply(x,function(x){x^2})
# test[[1]]
# test
noyau <- function(x){dnorm(x-t,0,sd=sqrt(h/2))/sum(dnorm(x-t,0,sd=sqrt(h/2)))}

W<-matrix(unlist(lapply(x,noyau)),ncol=n,nrow=n,byrow=F)
```

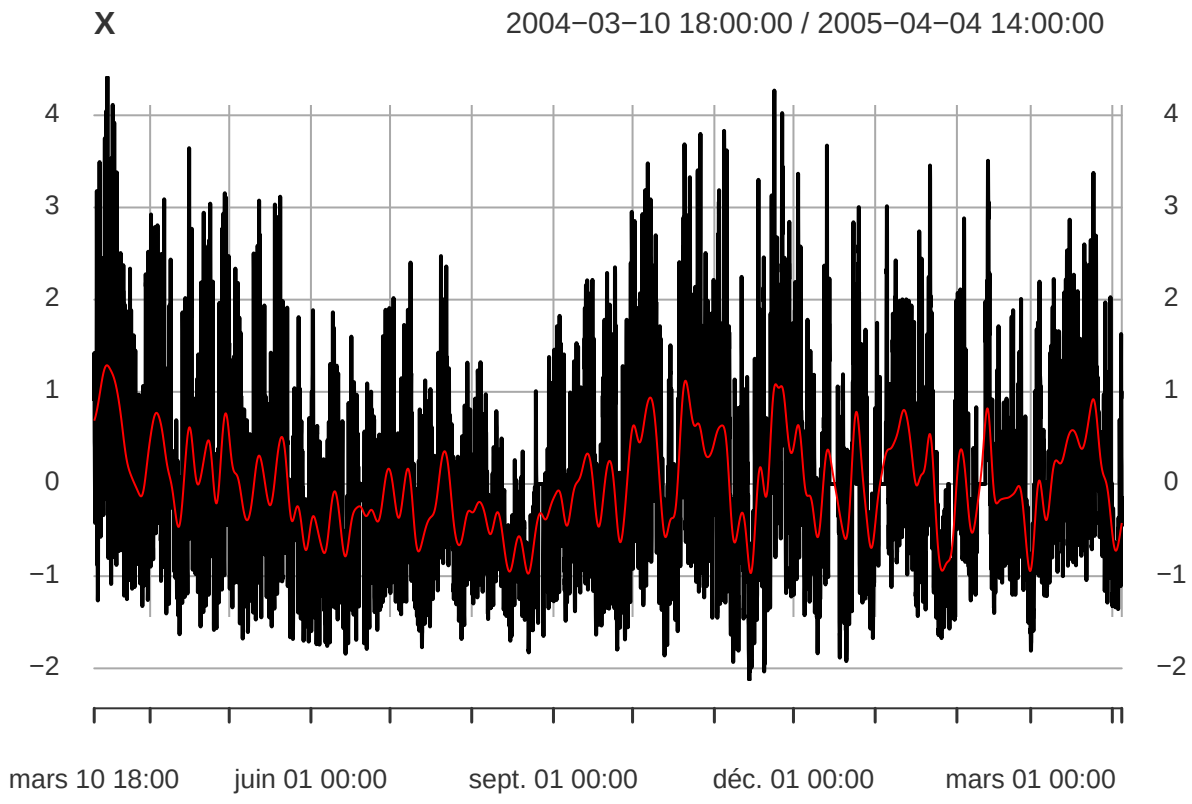
```
plot(W[,1], type='l')
```



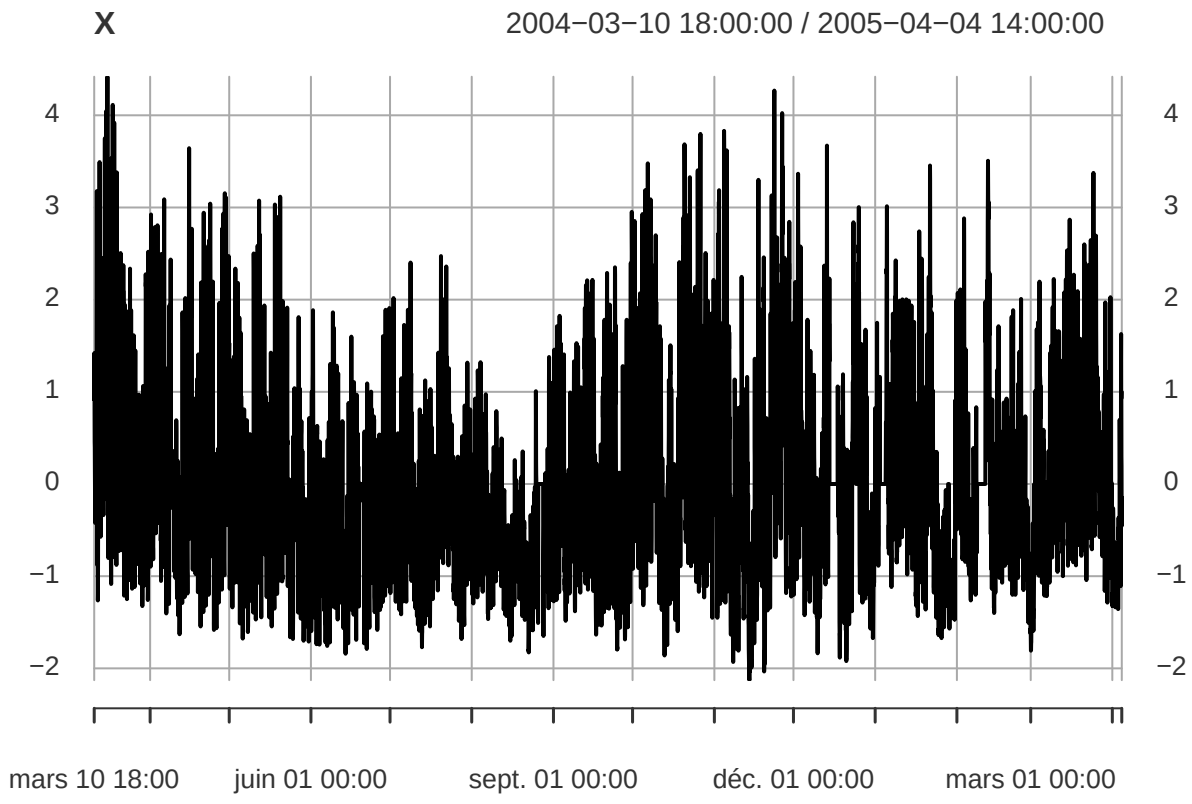

```
ychap.kernel<-colSums(as.numeric(X)*W)
ychap.kernel<-xts(ychap.kernel,order.by=Date_air)
plot(X,type='l')
```



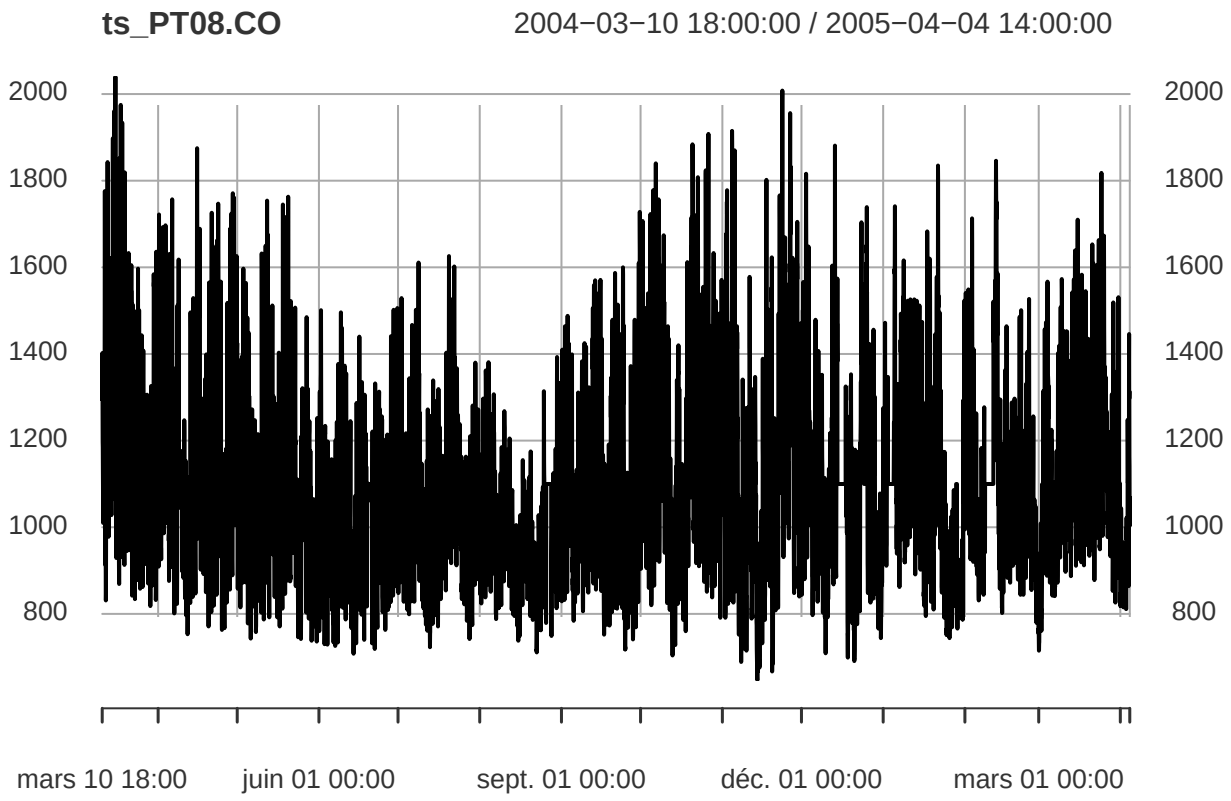

```
lines(ychap.kernel,col='red')
```




```
#Normalization
X= (ts_PT08.CO - mean(air_data$PT08.S1.CO.))/sd(ts_PT08.CO)
plot(X)
```

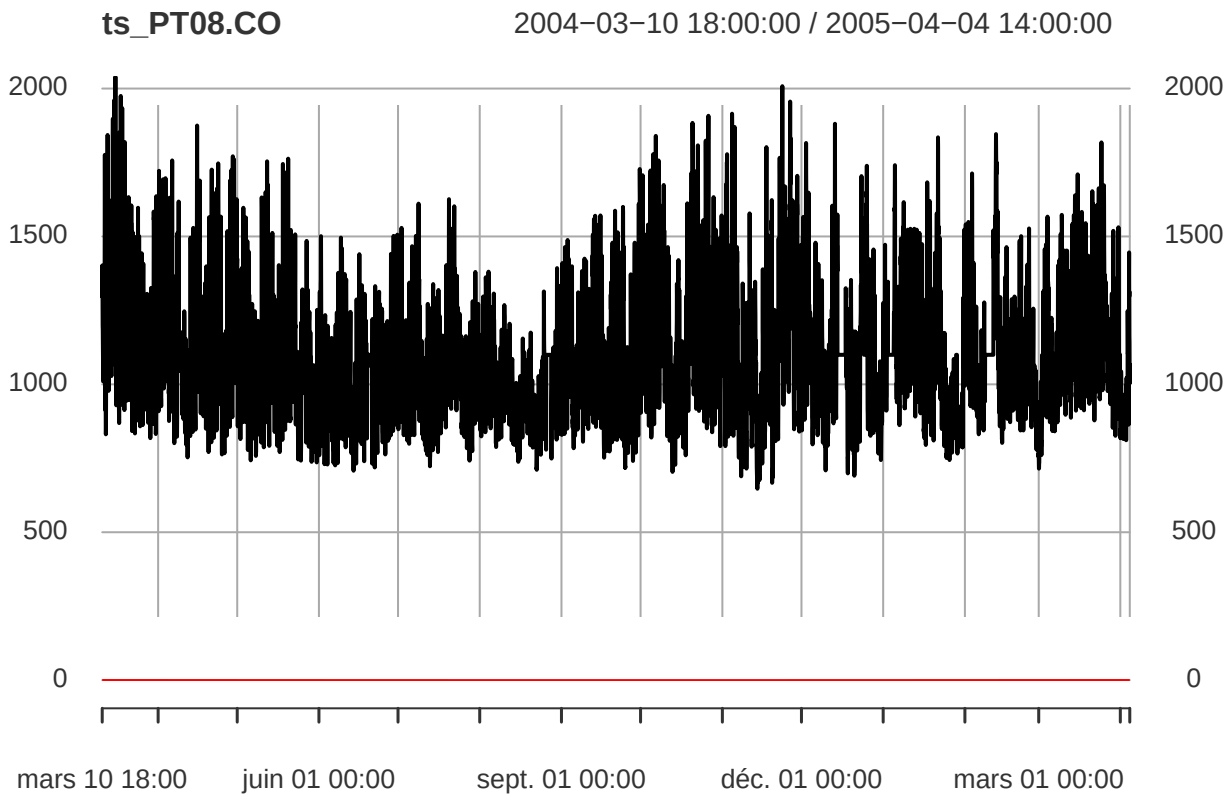



```
plot(ts_PT08.CO)
```

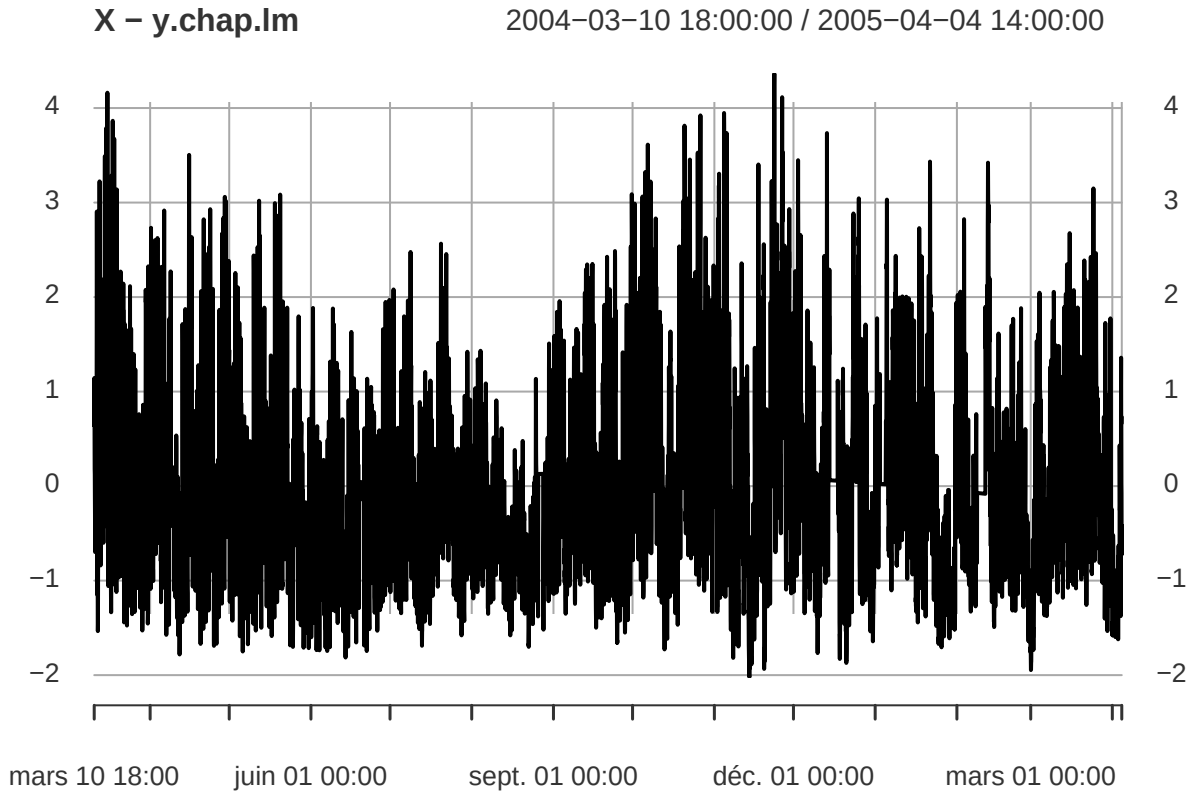



```
dygraph(X) %>% dyRangeSelector()
```

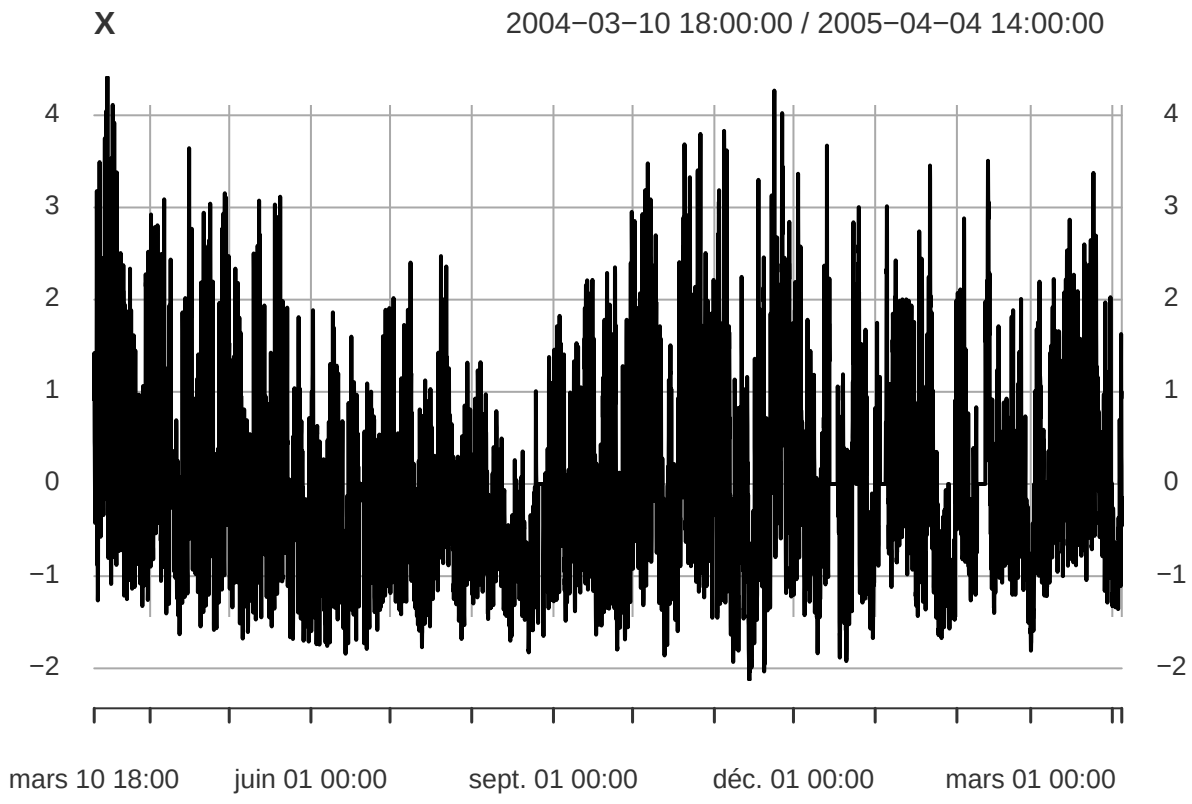
```
# Trend modelisation using a quadratic function
t <- c(1:length(Date_air))
reg<-lm(X~t+I(t^2))
y.chap.lm <- xts(reg$fitted, order.by = Date_air)
lines(y.chap.lm, col='red')
```



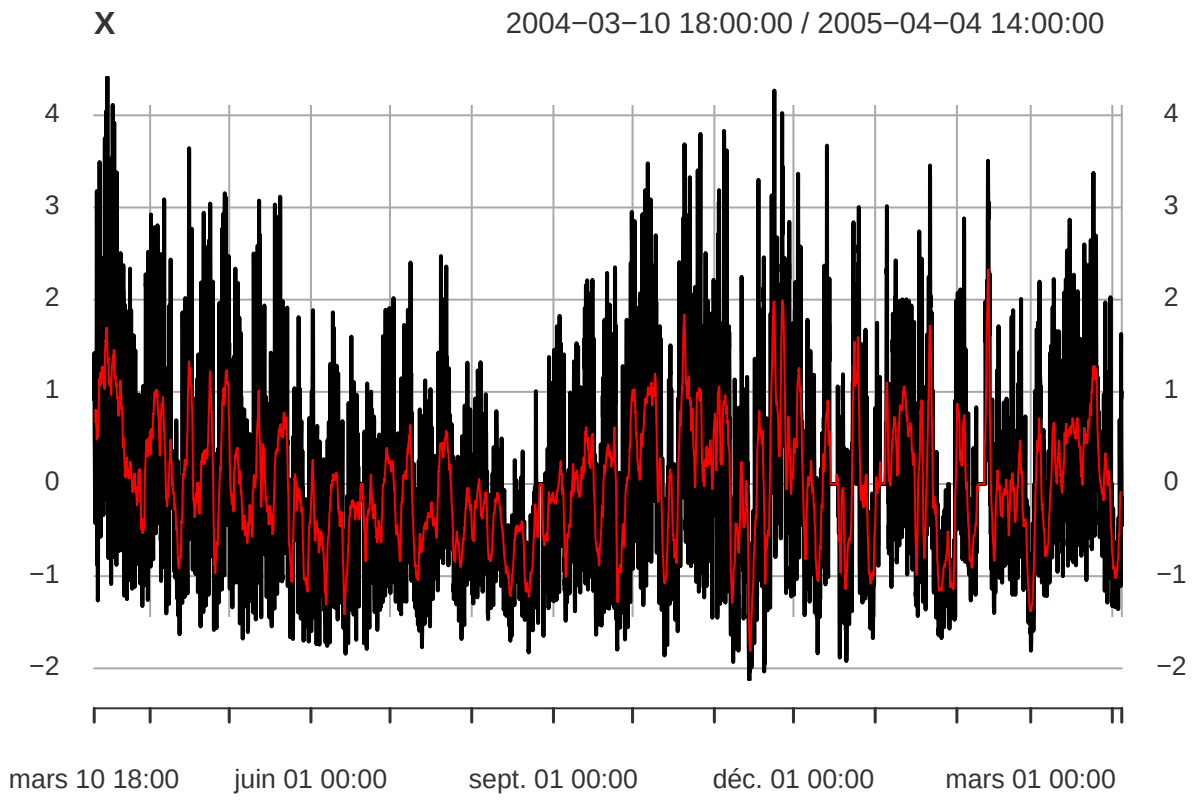

```
plot(X-y.chap.lm)
```



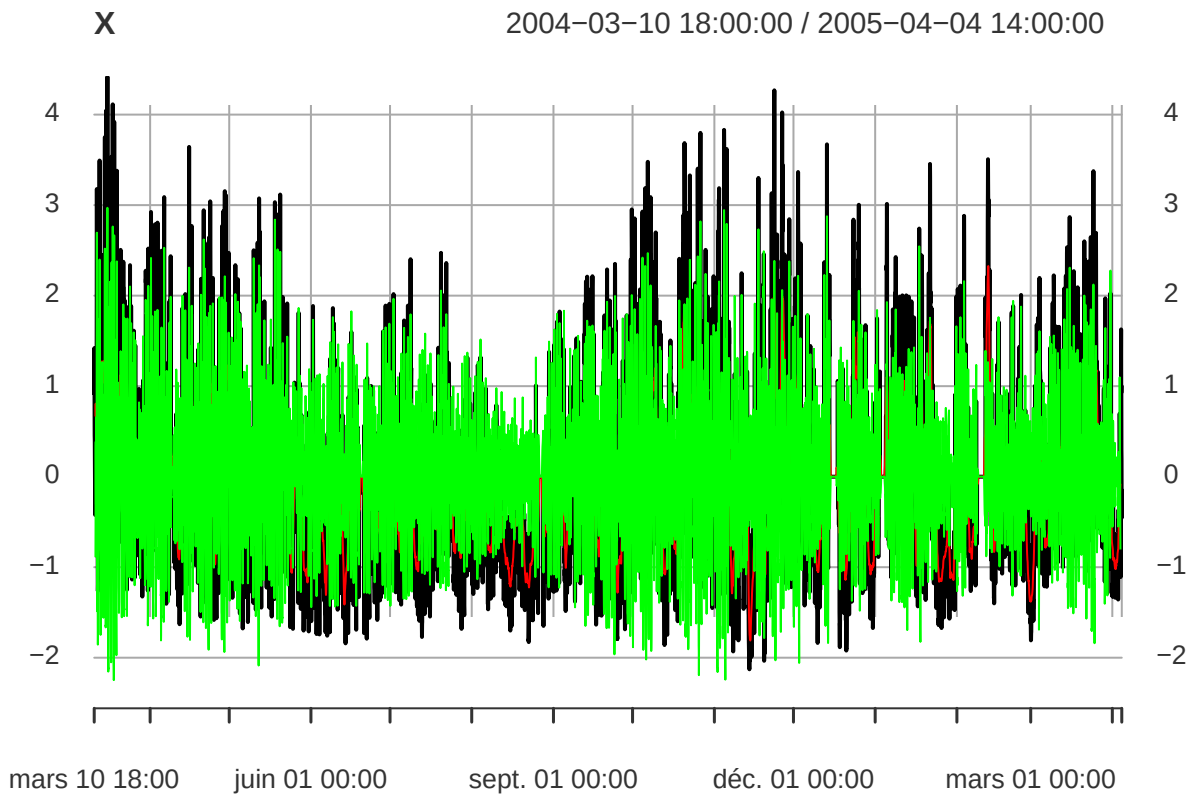

```
# Trend modelisation using Moving average trend
l <- 24
mb <- stats::filter(X, filter=array(1/l,dim=l),
                    method = c("convolution"),
                    sides = 2, circular = F)
mb <- xts(mb,order.by=Date_air)
X.detrend <- X - mb
plot(X)
```




```
lines(mb, col='red')
```

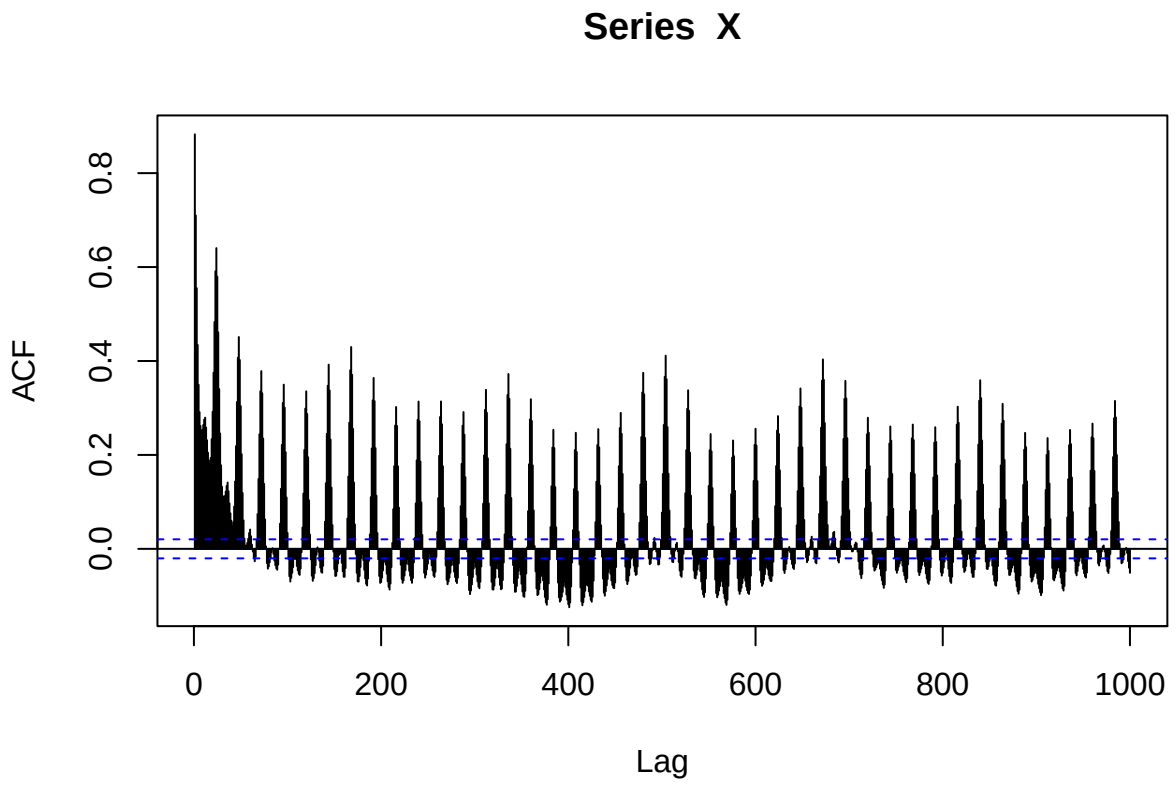



```
lines(X.detrend, col='green')
```

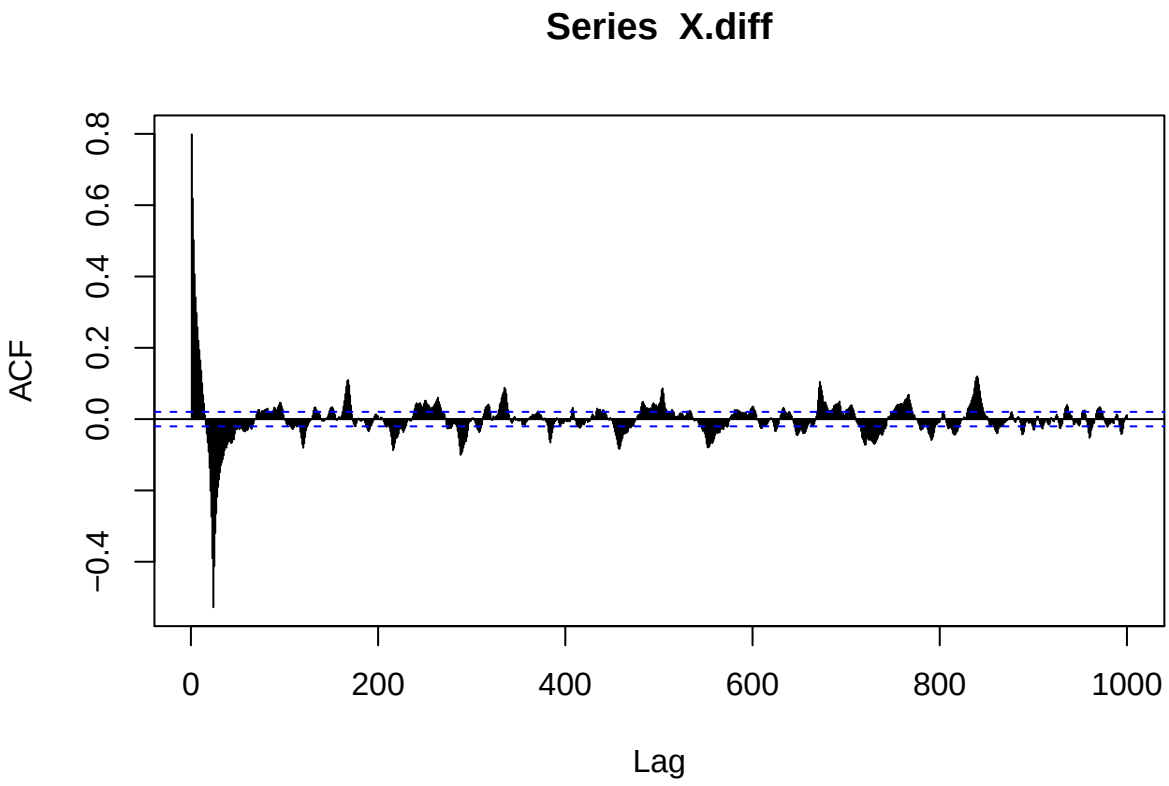



```
dygraph(X.detrend) %>% dyRangeSelector()
```

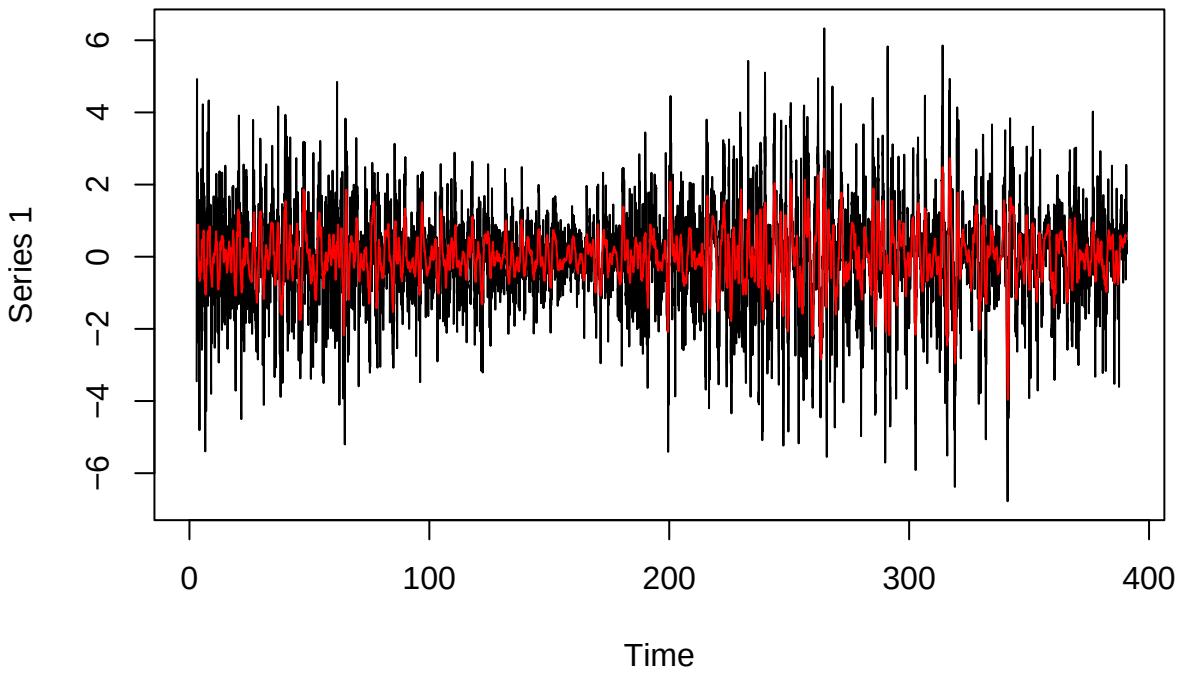
```
#Seasonality
### A FAIRE ###



# TEST Auto decompose
Acf(X, lag.max=1000)
```



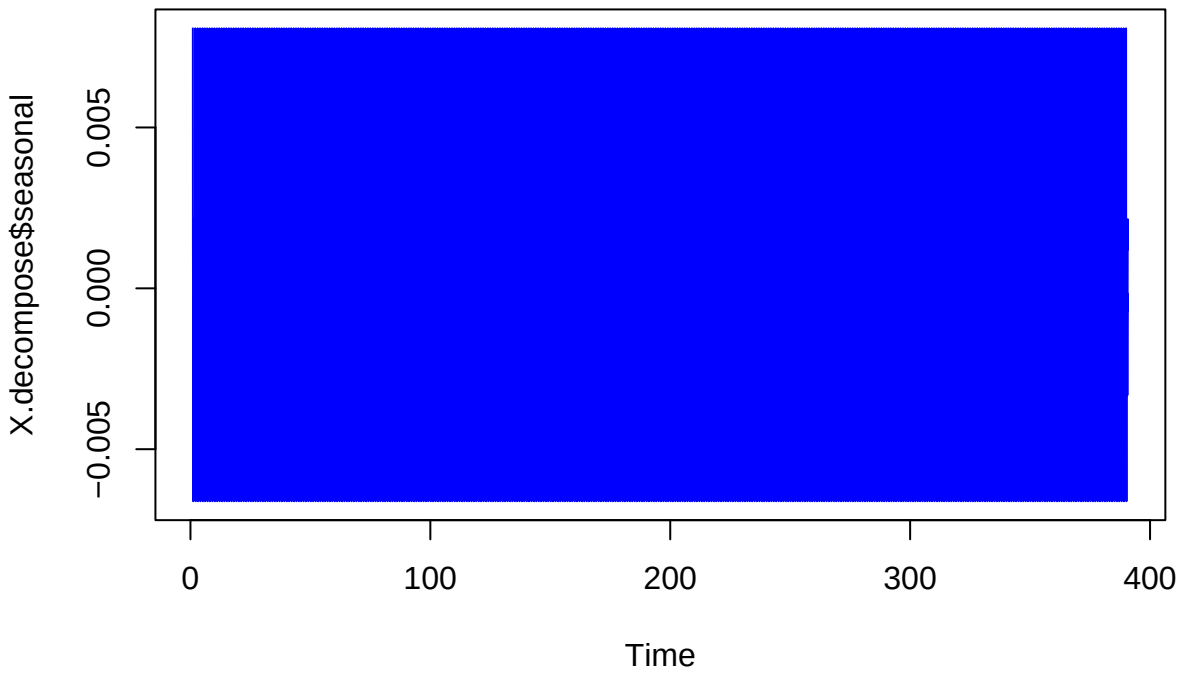

```
X.diff <- diff(X, lag = 24, difference=2)
Acf(X.diff, lag.max=1000)
```



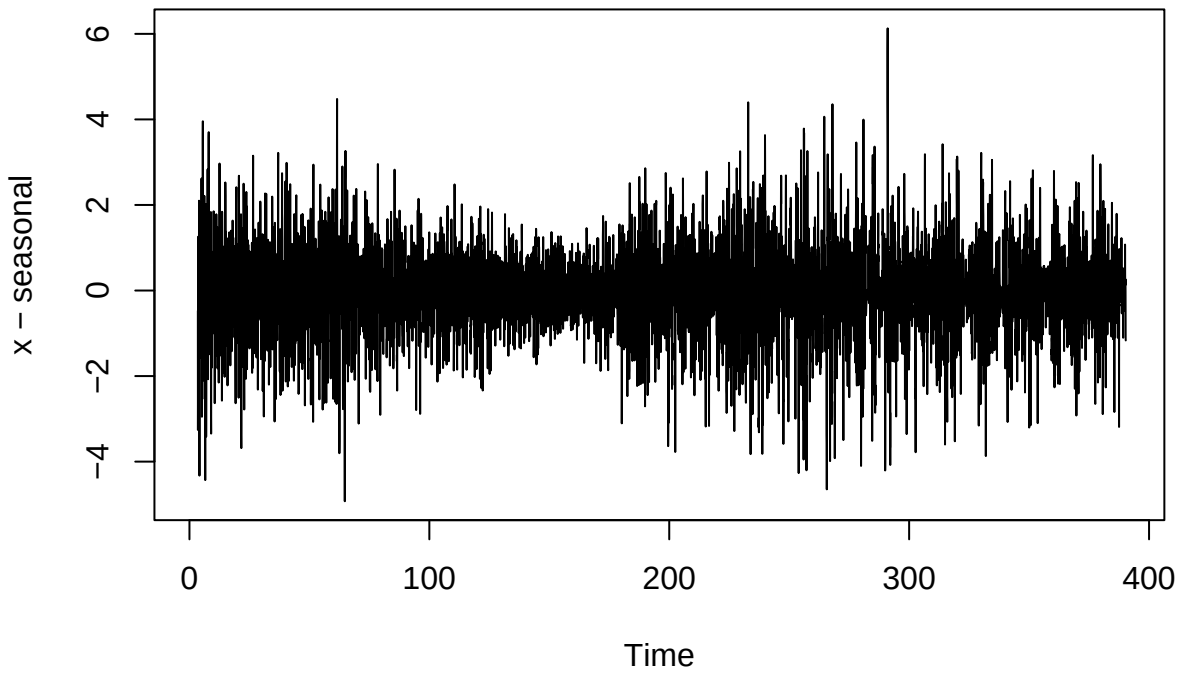

```
X.decompose <- decompose(ts(X.diff, frequency=24))
plot(X.decompose$x)
lines(X.decompose$trend, col='red')
```



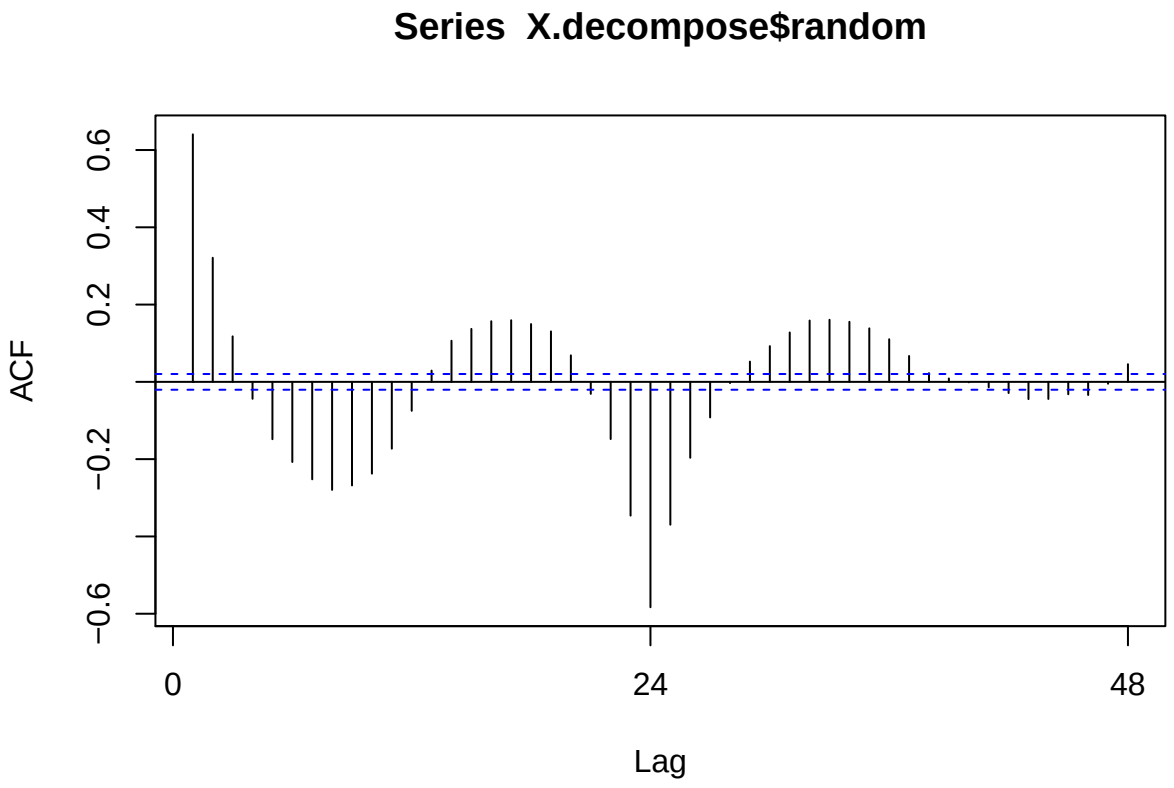

```
plot(X.decompose$seasonal, col='blue')
```



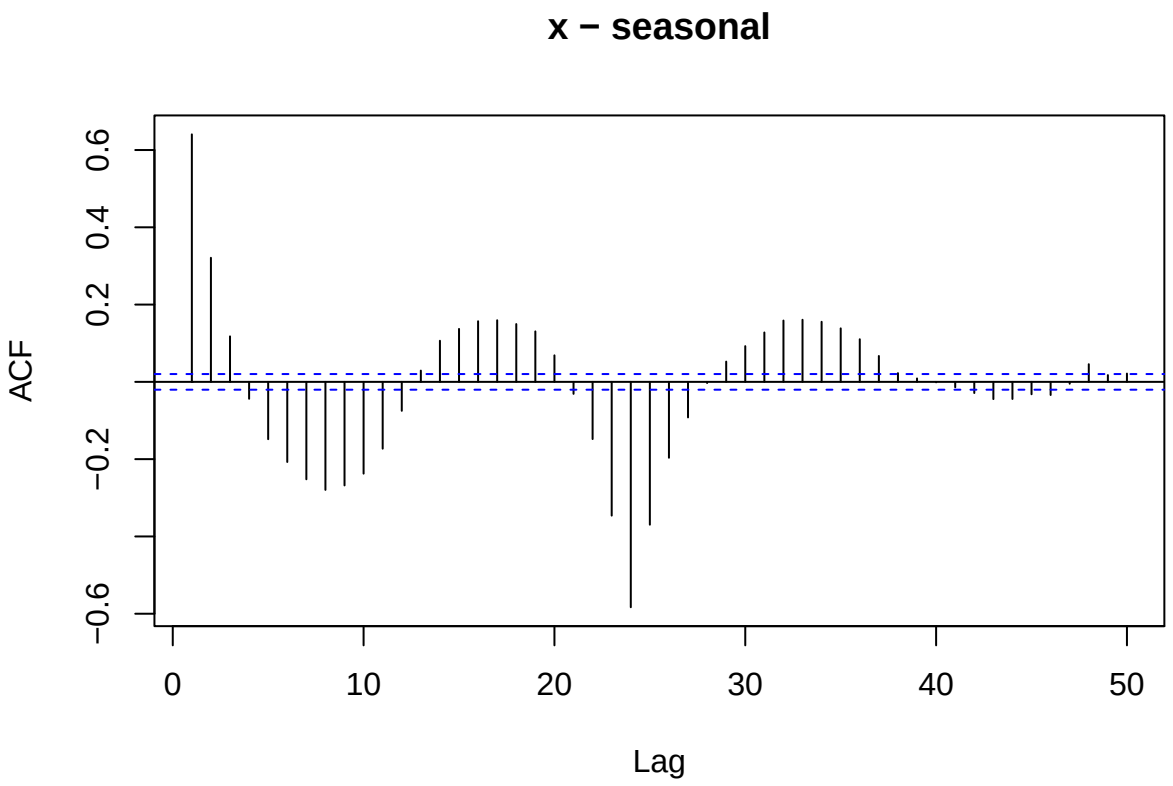

```
plot(X.decompose$random)
```



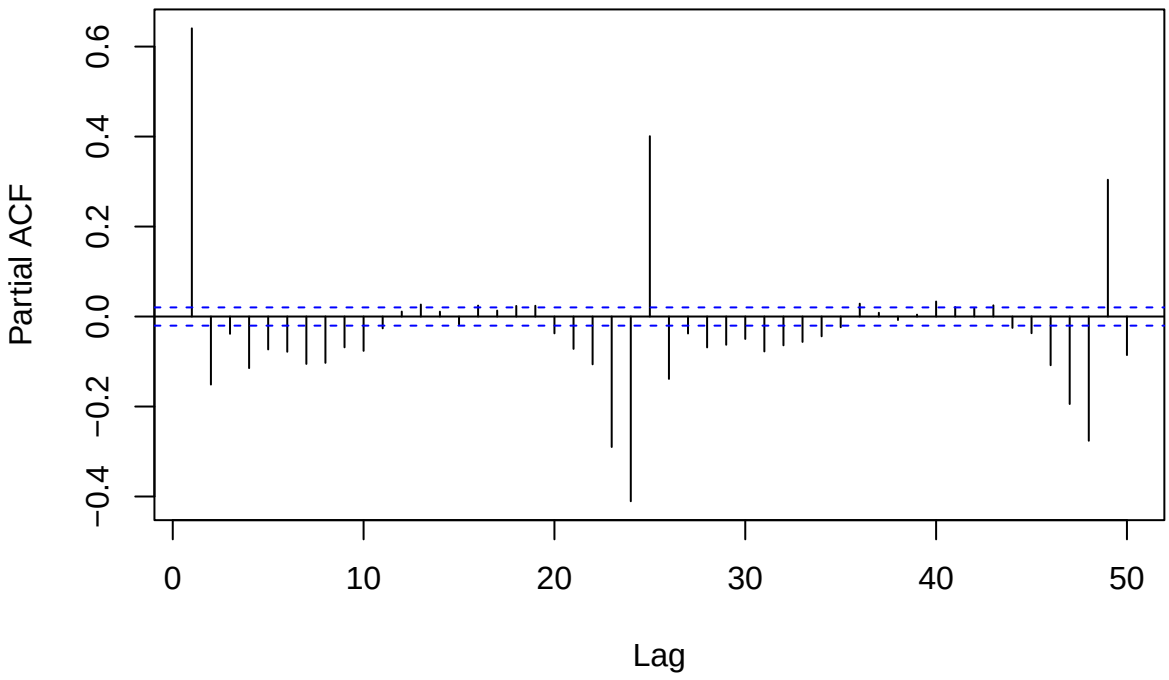

```
Acf(X.decompose$random)
```




```r
# modélisation du résidu
residual = X.decompose$random
residual <- apply(residual, 2,
                  function(x) replace(x, is.na(x), mean(x, na.rm = TRUE)))

Acf(residual, lag.max=50)
```



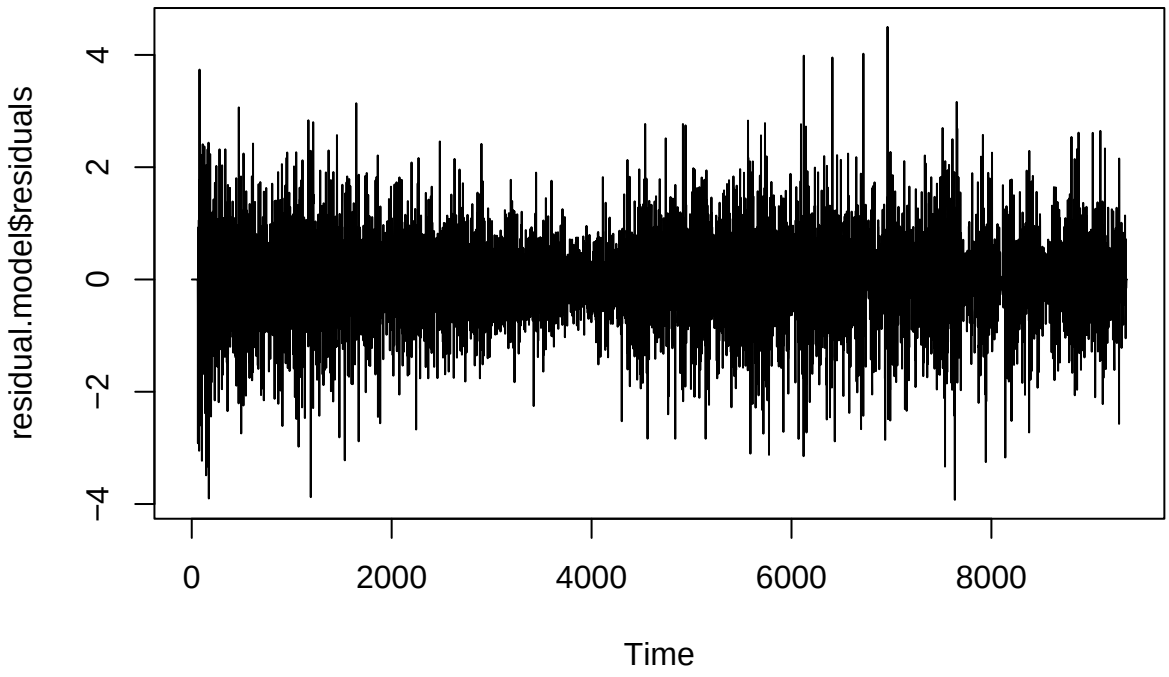

```
Pacf(residual, lag.max=50)
```




```
residual.model <- auto.arima(residual)
summary(residual.model)
```

```
## Series: residual
## ARIMA(5,0,0) with zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4      ar5
##       0.7187  -0.1336  0.0354  -0.0614  -0.0731
## s.e.  0.0103   0.0127  0.0128   0.0127   0.0103
##
## sigma^2 = 0.553:  log likelihood = -10502.37
## AIC=21016.74   AICc=21016.75   BIC=21059.6
##
## Training set error measures:
##                         ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.0004173901 0.7434557 0.5440593 97.47243 354.0662 0.8957985
##                      ACF1
## Training set -0.005648761
```

```
plot(residual.model$residuals)
```




```
adf.test(residual.model$residuals)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag   ADF p.value
##  [1,]   0 -97.3    0.01
##  [2,]   1 -69.7    0.01
##  [3,]   2 -57.8    0.01
##  [4,]   3 -51.2    0.01
##  [5,]   4 -44.5    0.01
##  [6,]   5 -40.5    0.01
##  [7,]   6 -39.0    0.01
##  [8,]   7 -39.4    0.01
##  [9,]   8 -39.1    0.01
## [10,]   9 -40.1    0.01
## [11,]  10 -40.6    0.01
## Type 2: with drift no trend
##       lag   ADF p.value
##  [1,]   0 -97.3    0.01
##  [2,]   1 -69.7    0.01
##  [3,]   2 -57.8    0.01
```

```
##  [4,]   3 -51.2    0.01
##  [5,]   4 -44.5    0.01
##  [6,]   5 -40.5    0.01
##  [7,]   6 -39.0    0.01
##  [8,]   7 -39.4    0.01
##  [9,]   8 -39.1    0.01
## [10,]   9 -40.1    0.01
## [11,]  10 -40.6    0.01
## Type 3: with drift and trend
##       lag   ADF p.value
##  [1,]   0 -97.3    0.01
##  [2,]   1 -69.7    0.01
##  [3,]   2 -57.8    0.01
##  [4,]   3 -51.2    0.01
##  [5,]   4 -44.5    0.01
##  [6,]   5 -40.5    0.01
##  [7,]   6 -39.0    0.01
##  [8,]   7 -39.4    0.01
##  [9,]   8 -39.1    0.01
## [10,]   9 -40.1    0.01
## [11,]  10 -40.6    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
shapiro.test(residual.model$residuals[c(1:5000)])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residual.model$residuals[c(1:5000)]
## W = 0.97417, p-value < 2.2e-16
```