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.

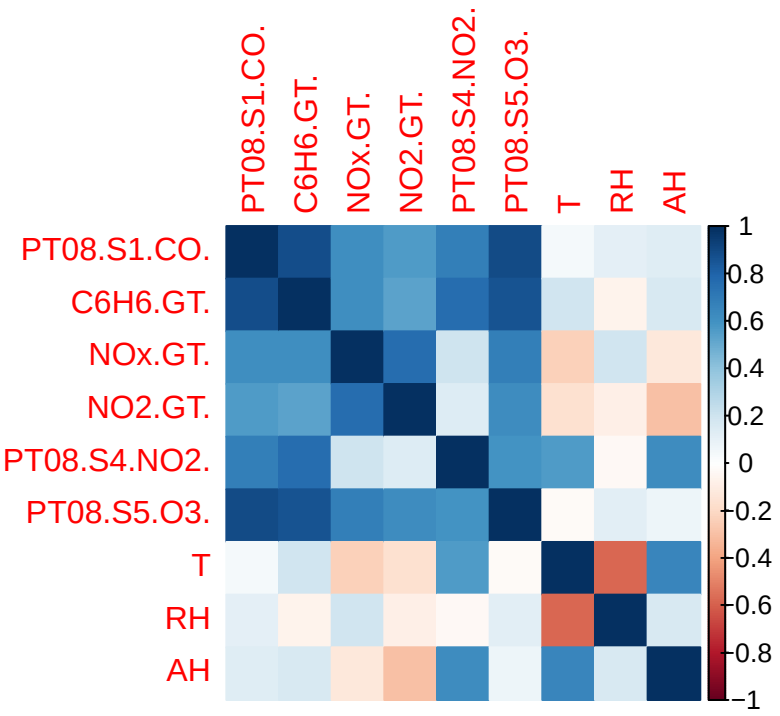


Figure 1: Matrice de corrélation des variables observées

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.

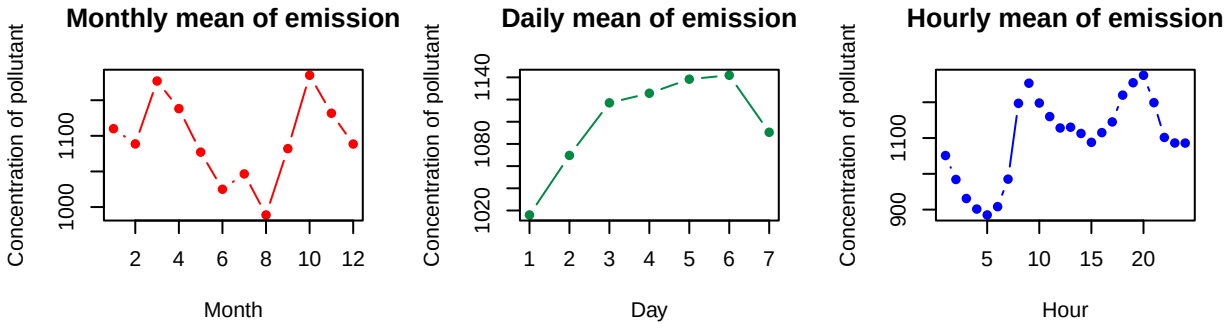


Figure 2: Concentrations moyennes périodiques PT08.CO

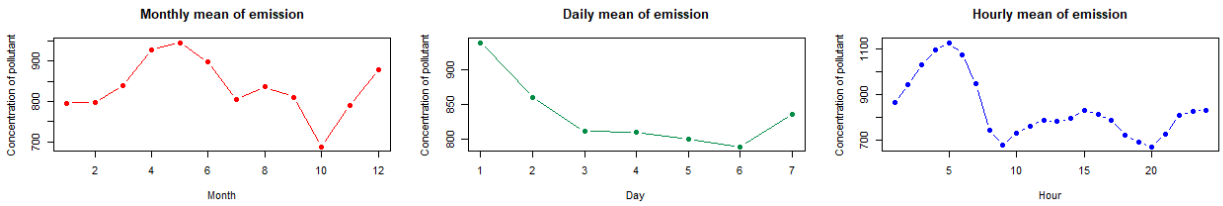


Figure 3: Concentrations moyennes périodiques NOx.GT

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

Pour l'analyse de la série temporelle, nous allons effectuer une décomposition de la série en tendance, saisonnalité.

Avant de nous attaquer à cette décomposition, une bonne pratique dans des travaux liés au Machine Learning et aux statistiques est de normaliser les données. Ceci peut être utile surtout lorsque l'on souhaite comparer cette série temporelle avec d'autres séries temporelles qui ne seraient pas de la même échelle.

```
X<-(air_data$PT08.S1.CO. - mean(air_data$PT08.S1.CO.))/sd(air_data$PT08.S1.CO.)
X<-xts(X,order.by=Date_air)
```

4.1 Tendances

Notre série temporelle ne semble à première vue pas comporter de composante tendancielle. Néanmoins, en zoomant sur sa figure, on observe des intervalles sur lesquelles elle admet des tendances locales. L'objet de cette partie sera donc d'extraire ces tendances locales. Nous allons faire appel à plusieurs méthodes pour annalyser la tendance et la saisonnalité. Parmi ces méthodes, nous nous intéresserons à la regression linéaire, la moyenne mobile, la convolution avec un noyau gaussien, la regression sur une base de splines, la regression par polynômes locaux.

Pour prédire nos données à l'aides de modèles, il faut d'abord que l'on sépare notre série temporelle en deux parties. Une partie utilisée pour l'entraînement des modèles, et une utilisée pour déterminer leur efficacité.

```
n <- length(Date_air)
n.train = as.integer(70*n/100)
X.train = X[1:n.train]
X.test = X[(n.train+1):n]
X <- X.train
Date_air.train = Date_air[1:n.train]
```

4.1.1 Régression linéaire

```
t <- c(1:n.train)

reg.1 <- lm(X~t)
```

```
reg.2 <- lm(X~t+I(t^2))
reg.3 <- lm(X~t+I(t^2)+I(t^3))
reg.4 <- lm(X~t+I(t^2)+I(t^3)+I(t^4))
reg.5 <- lm(X~t+I(t^2)+I(t^3)+I(t^4)+I(t^5))

y.chap.lm.1 <- xts(reg.1$fitted, order.by = Date_air.train)
y.chap.lm.2 <- xts(reg.2$fitted, order.by = Date_air.train)
y.chap.lm.3 <- xts(reg.3$fitted, order.by = Date_air.train)
y.chap.lm.4 <- xts(reg.4$fitted, order.by = Date_air.train)
y.chap.lm.5 <- xts(reg.5$fitted, order.by = Date_air.train)
```

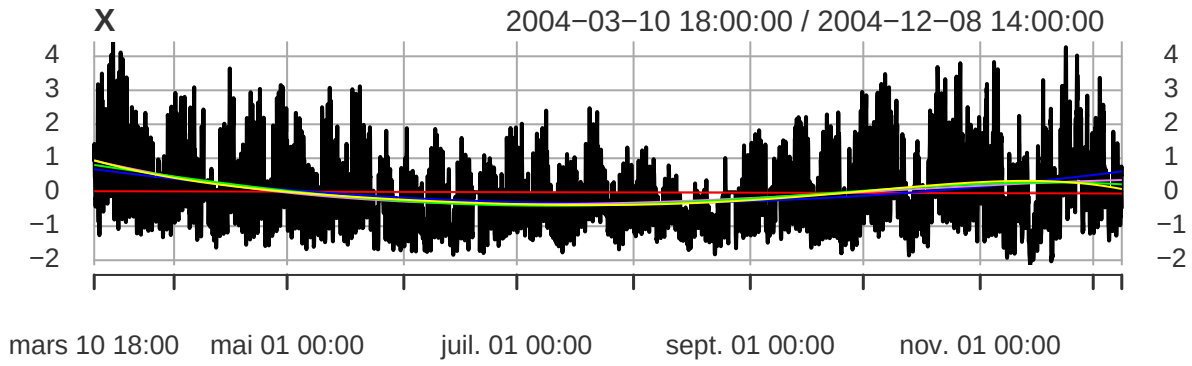


Figure 4: Régression linéaire ; d°1:rouge, d°2:bleu, d°3:violet, d°4:vert, d°5:jaune

On observe sur la figure 4 que la régression linéaire sur les droites nous affichent une pente quasi nulle devant l'amplitude des données. Pour l'ordre 2, on remarque qu'il y a un léger comportement convexe sur les données.

La régression à l'ordre 4 semble bien capturer le comportement basse-fréquence de la série, sans montrer d'abérration comme le fait le modèle d'ordre 5 vers la fin du graphe, en remontant. Pour éviter le surapprentissage, un modèle à l'ordre 4 semble raisonnable.

En soustrayant à X la tendance estimée, on obtient un signal qui est bien centré en 0.

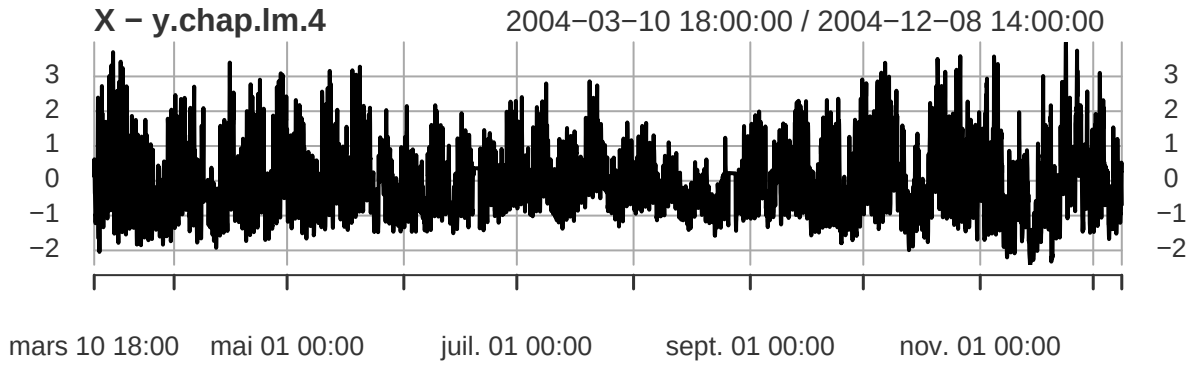


Figure 5: X.detrend lm

4.1.2 Moyenne mobile

La moyenne mobile peut être utilisée pour capter la tendance de notre série. Notre série étant périodique de période 24h, il sera important d'ajuster la fenêtre correctement pour filtrer la saisonnalité.

```
l <- 24
MA.trend <- stats::filter(X, filter=array(1/l,dim=l),
                  method = c("convolution"),
                  sides = 2, circular = F)
MA.trend <- xts(MA.trend, order.by=Date_air.train)
X.detrend <- X - MA.trend
```

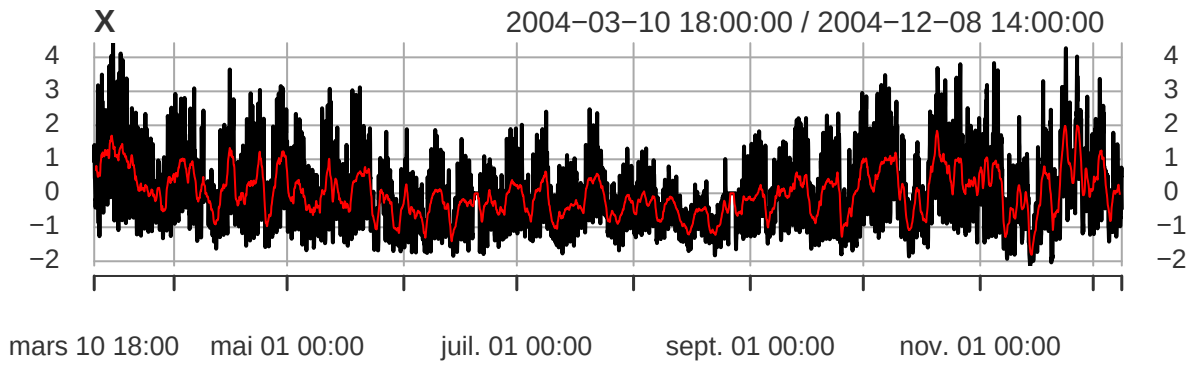


Figure 6: Moyenne mobile de fenêtre l=24

L'utilisation de la moyenne mobile calcule une tendance avec des fréquences beaucoup plus hautes et semble tout de même. Néamoins, la série corrigée de la tendance reste bien centrée.

4.1.3 Convolution sur un noyau gaussien

```
h<-10000

x<-seq(1,max(t),length=n.train)

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.train,nrow=n.train,byrow=F)

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

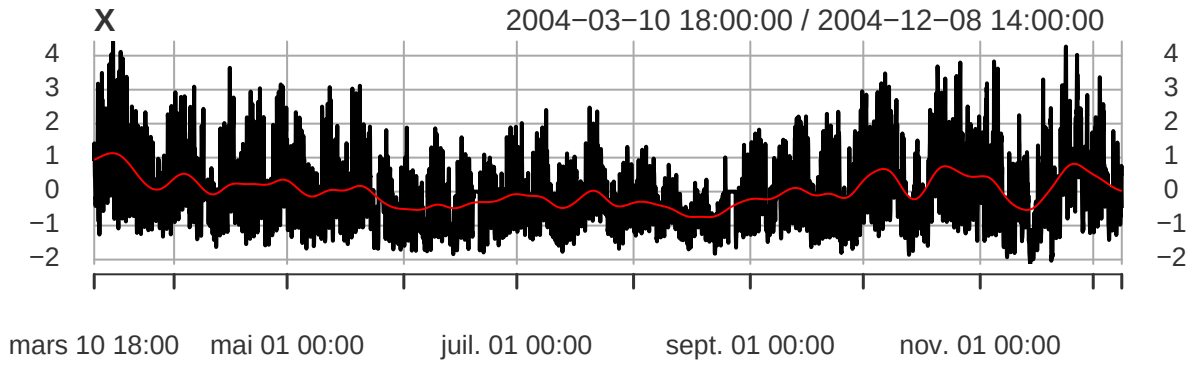


Figure 7: Noyau gaussien, fenêtre h=10000

L'utilisation du noyau gaussien avec un paramètre h raisonnable semble donner un bon compromis entre la moyenne mobile de fenêtre 24, et la régression linéaire.

4.1.4 Régression sur base de splines

```
g <- gam(X~s(t, k=5))
ychap.gam<-xts(g$fitted,order.by=Date_air.train)
```

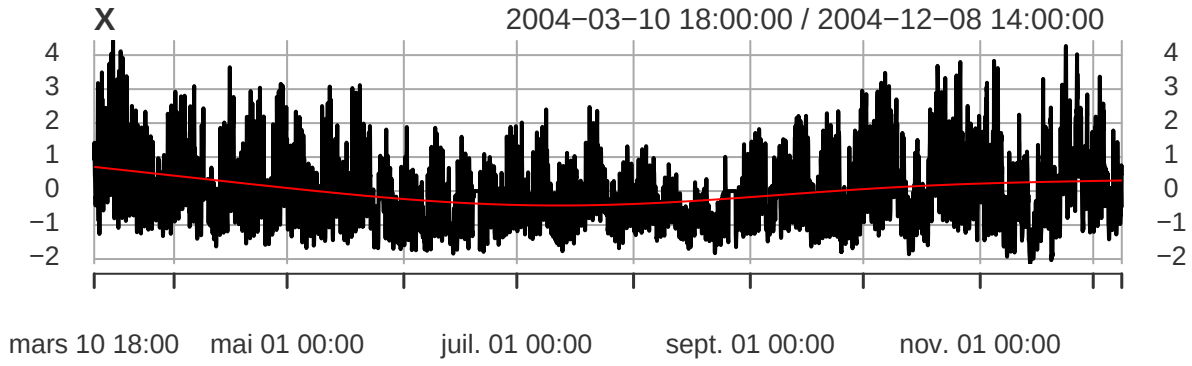


Figure 8: Base de splines, k=5

La régression sur la base de spline montre un très bon résultat, les basses fréquences ont bien été extraites du signal.

4.1.5 Régression par polynômes locaux

```
X <- xts(X,order.by=Date_air.train)
lo <- loess(X~t, degree=2, span=0.7)
ychap.lo <- xts(lo$fitted,order.by=Date_air.train)
```

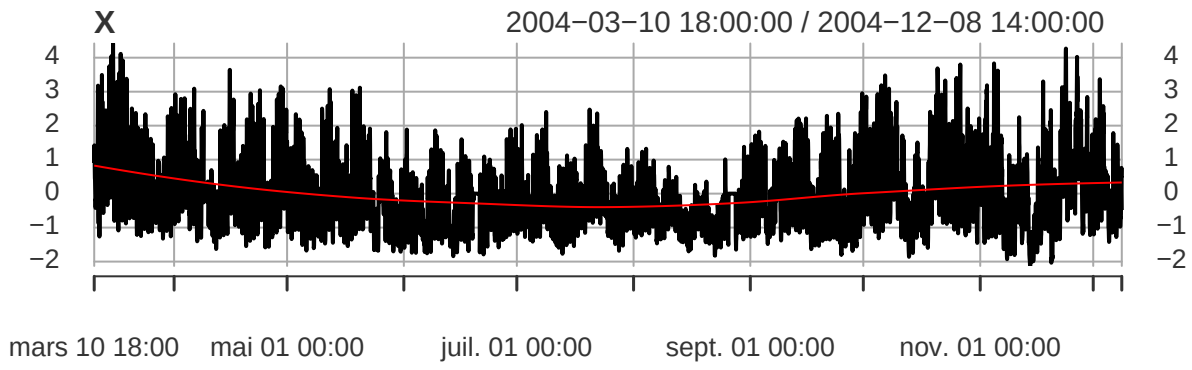


Figure 9: Polynômes locaux

Un degré adapté pour les polynômes locaux nous permet aussi d'extraire les basses fréquences du signal.

On choisira finalement arbitrairement de modéliser la tendance de la série à l'aide de sa décomposition sur la base de splines.

Le signal ainsi corrigé est le suivant :

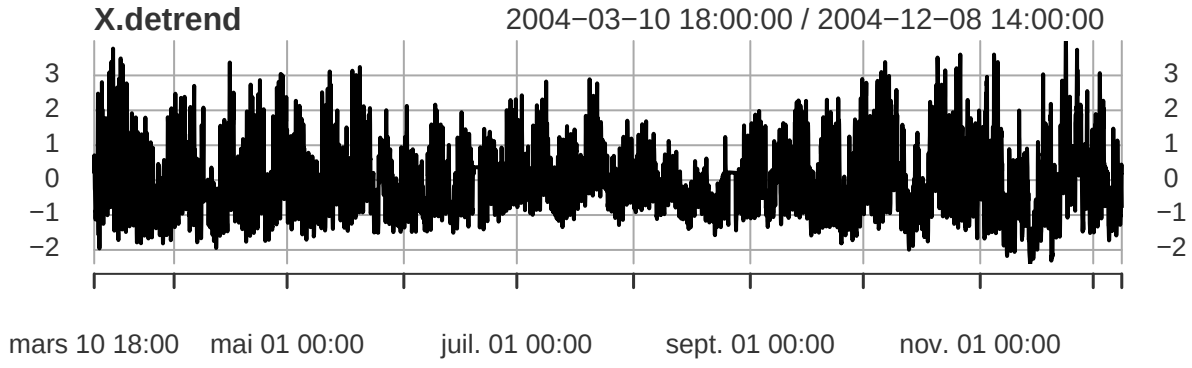
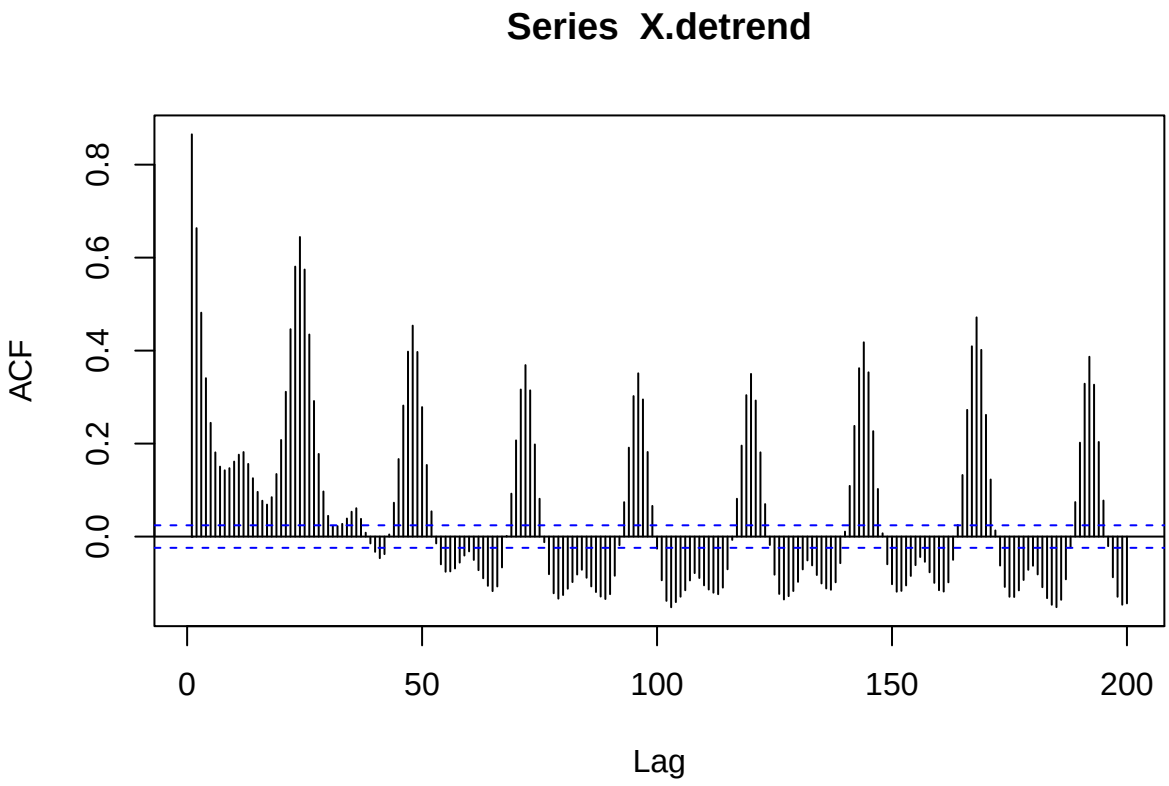


Figure 10: Signal centré

4.2 Saisonnalité

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

```
Acf(X.detrend,lag.max=200)
```



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 est d'autant plus justifié qu'il n'y a pas de décroissance exponentielle. Ceci peut également être vérifié en utilisant le test ADF (Augmented Dickey-Fuller), qui teste l'hypothèse nulle suivante :

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

Le test ADF affiche une p-value de 0.99, ce qui est bien au dessus de 0.05. On ne peut donc pas rejeter l'hypothèse nulle. La série temporelle est donc non stationnaire.

```
adf.test(X.detrend)
```

4.2.1 Extraction de la saisonnalité par décomposition en série de fourier

On effectue une régression linéaire sur une base de fourier associée à la pulsation de fréquence 1/24. L'affichage de la figure **??** montre bien un signal périodique, qui semble représenter la saisonnalité de la série temporelle. Néanmoins, l'amplitude de ce signal est relativement faible, ce qui contredit l'analyse que l'on a fait de l'ACF...

```
w=2*pi/24
fourier<-cbind(cos(w*t), sin(w*t))
K<-30
for(i in c(2:K))
{
  fourier<-cbind(fourier,cos(i*w*t), sin(i*w*t))
}
```

```
reg<-lm(X.detrend~fourier[,1:K]-1)
ychap.lm.season<-xts(as.numeric(reg$fitted),order.by=Date_air.train)
```

Le résidu devient finalement le suivant.

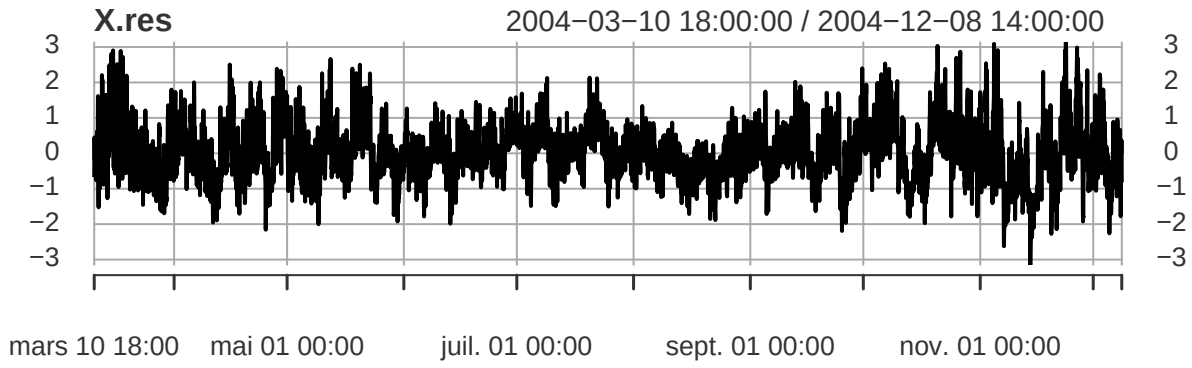


Figure 11: Résidu

4.2.2 Différentiation

Plutôt que d'utiliser la méthode précédente pour éliminer la saisonnalité, nous allons procéder à une différentiation.

On va pouvoir se rapprocher d'un comportement stationnaire 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 :

```
X.detrend.diff <- na_mean(diff(X.detrend, lag=24, differences=1))
```

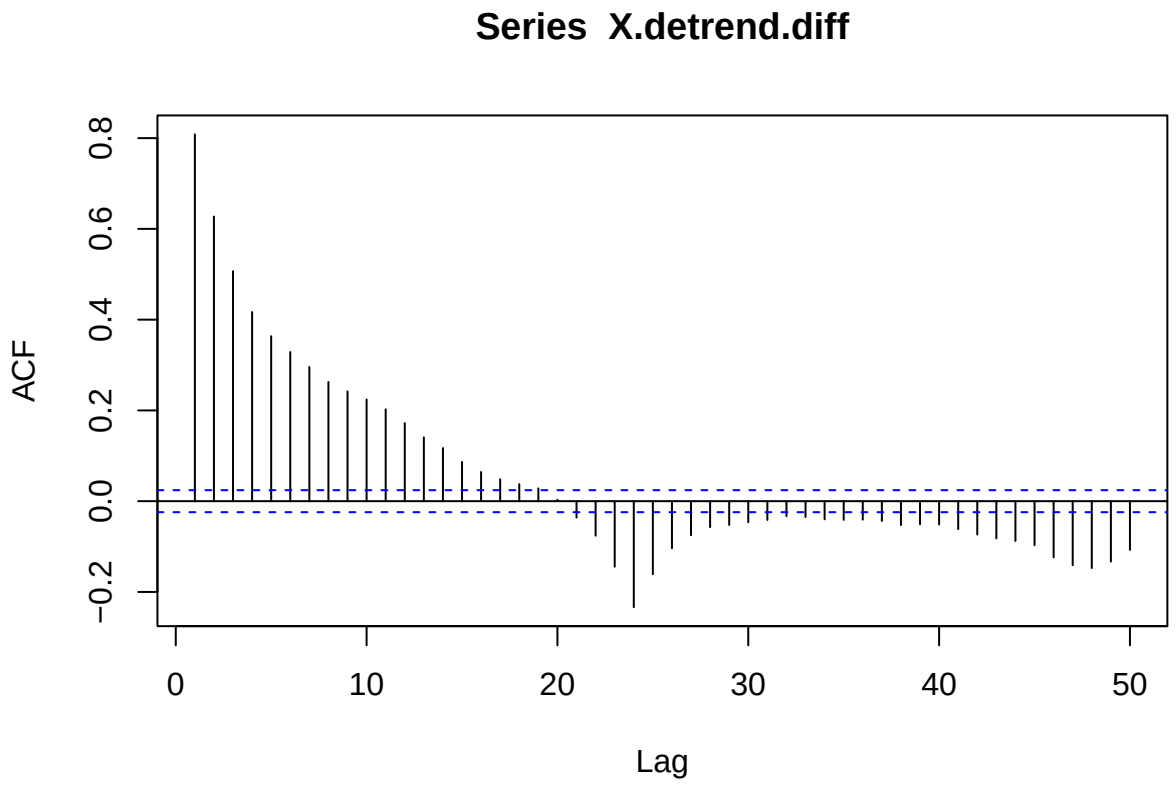
On obtient un graphique satisfaisant avec une décroissance vers zéro, mais cette décroissance est très lente... On souhaite maintenant déduire les plages de paramètres possibles pour un modèle $ARMA(p, q)$ du résidu. Pour ce faire, on étudie l'autocorrélogramme et l'autocorrélogramme partiel de la série différenciée.

```
par(c(1, 2))
```

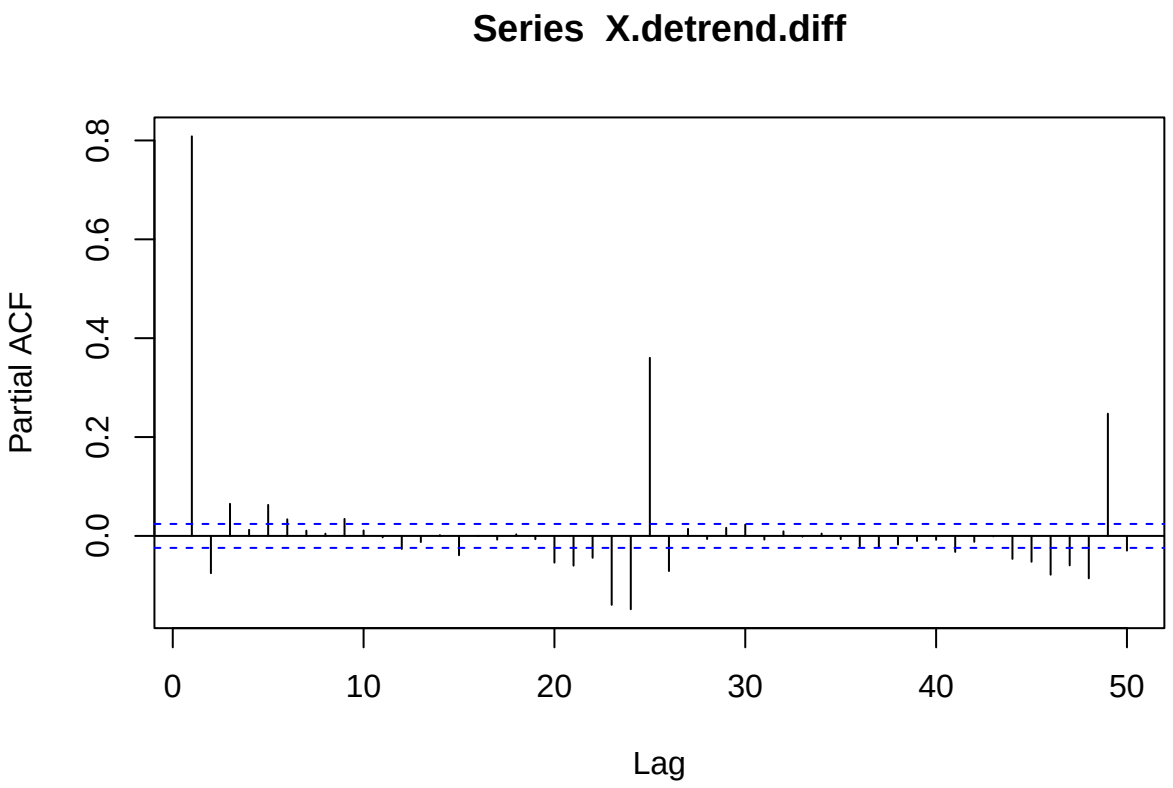
```
## Warning in par(c(1, 2)): argument 1 does not name a graphical parameter
```

```
## NULL
```

```
Acf(X.detrend.diff,lag.max=50)
```




```
Pacf(X.detrend.diff, lag.max=50)
```



On va donc modéliser notre résidu par un modèle ARMA. Il nous reste à choisir les paramètres p et q. Pour le paramètre, on note que l'autocorrélogramme partiel possède un pic au lag=1,2,3 qui sont significatifs. Il y a également des pics significatifs entre 20 et 25 mais le nombre de pics non significatifs avant ces lags nous fait penser que l'on peut se restreindre aux 3 premiers, c'est à dire pmax=3. L'étude de l'autocorrélogramme nous apporte moins d'informations car les pics sont significatifs jusqu'aux lags 40 au moins. Cependant, les pics décroissent exponentiellement jusq'au lag 20, on peut donc penser prendre qmax=20. Il nous faut ensuite tester tous les modèles et choisir le meilleur selon le critère AIC. En suivant cette méthodologie, c'est le modèle ARMA(3, 0, 20) qui possède le plus faible AIC parmis l'ensemble des modèles calculés, avec un AIC de 8115.5.

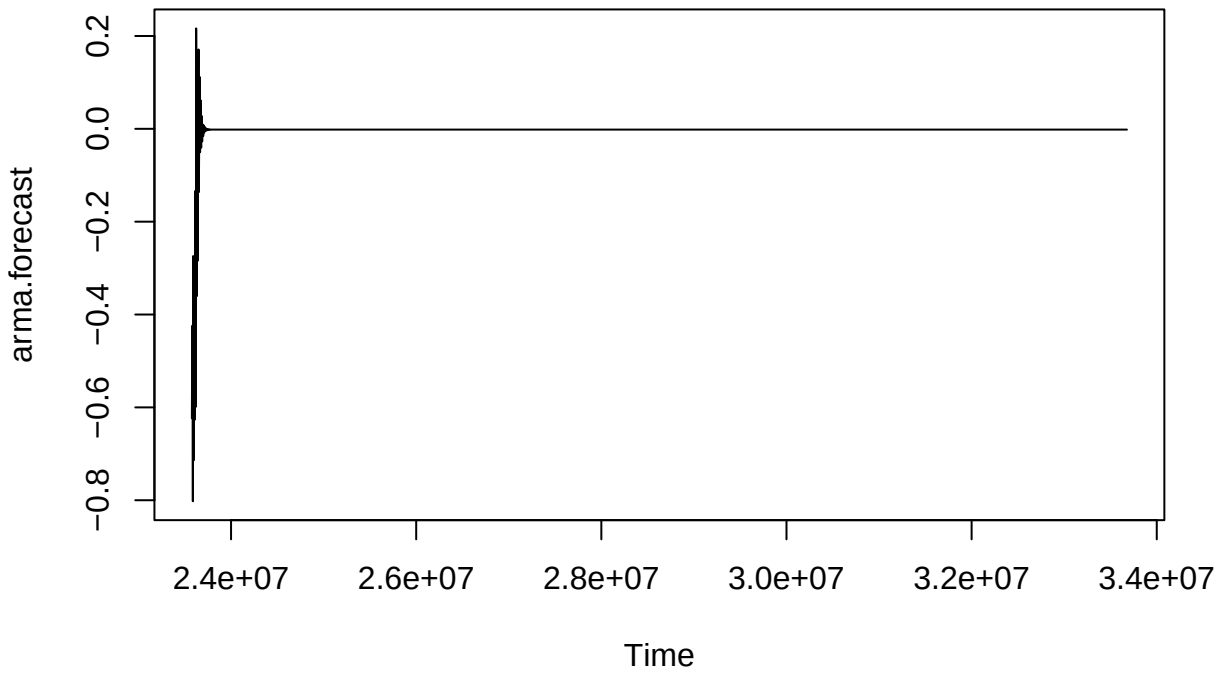
Bien que le résidu semble être gaussien, le test de Ljung-Box nous indique que ce n'est pas le cas. Ceci est sûrement du à la persistance de la saisonnalité, qui malgré une différentiation (et même plus lors de nos tests), semble ne pas vouloir disparaître.

5 Prediction

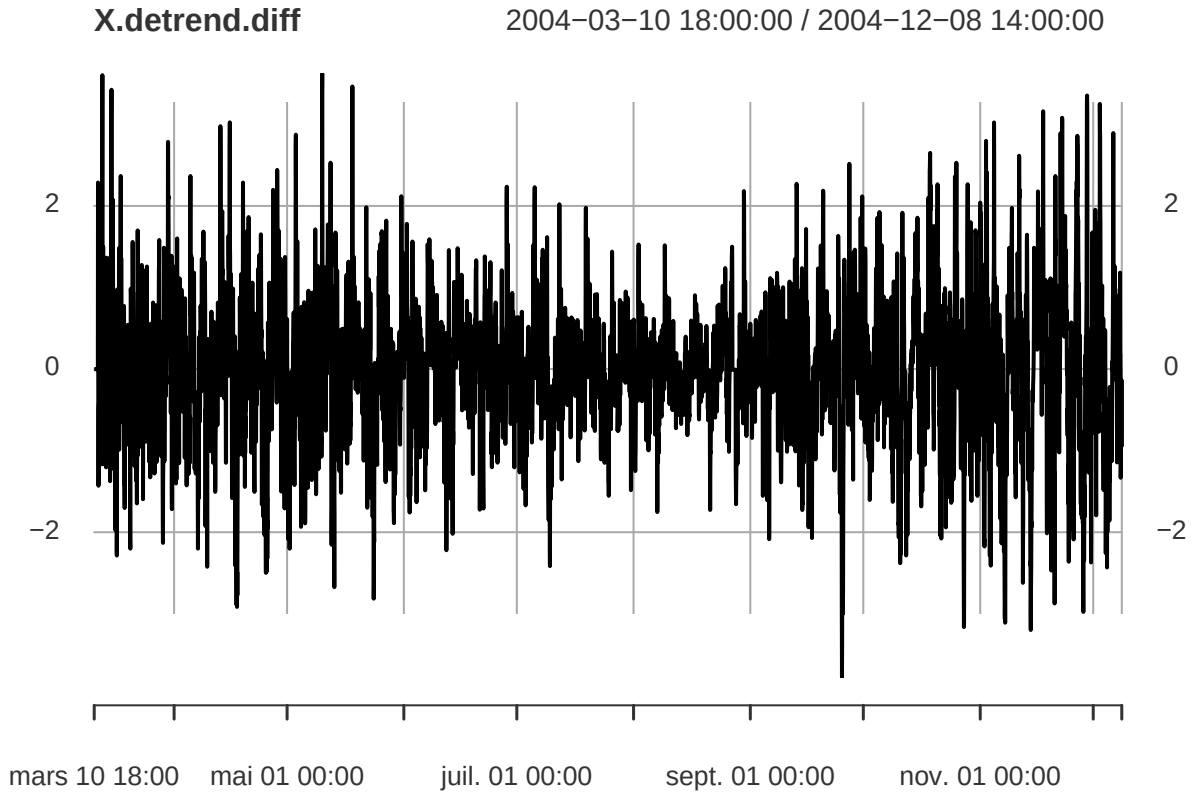
La prédiction sur un jeu de données tests va permettre d'évaluer les modèles. On procèdera ici à la prédiction par le modèle ARMA calculé précédemment, mais aussi à un modèle de prédiction par lissage exponentiel.

5.1 Modèle ARMA

```
n.test = n - n.train
arma.forecast <- predict(model, n.ahead = n.test, se.fit =F)
plot(arma.forecast)
```



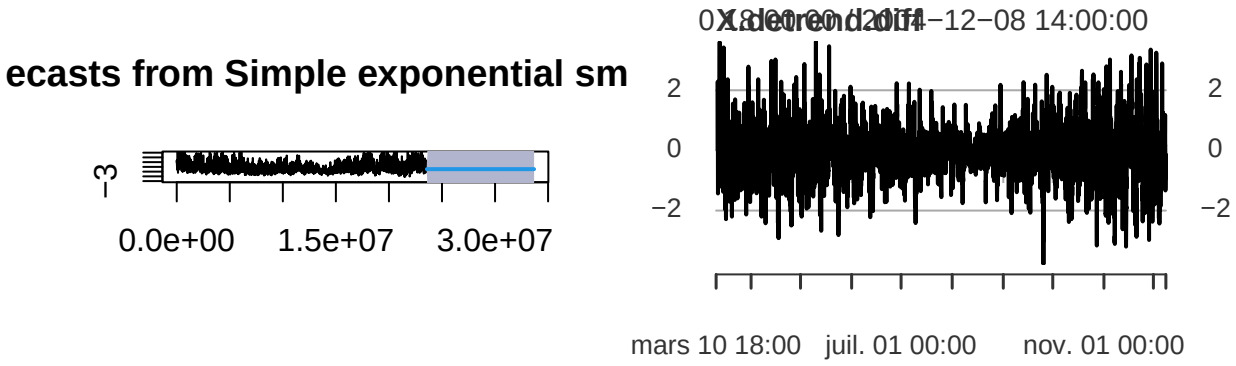

```
plot(X.detrend.diff,xlim=c(1,nrow(data)+n.test))
```



Les problèmes que l'on a eu avec le modèle ARMA du résidu se répercutent ici...

5.2 Lissage exponentiel

```
ses.model <- forecast::ses(X.train, alpha=0.99, h=n.test)
forecast.ses <- forecast::forecast(ses.model, n.test)
par(mfrow=c(1, 2))
plot(forecast.ses, ylim=c(-3, 3))
plot(X.detrend.diff)
```



Le lissage exponentiel donne une estimation à l'ordre 0 des données.

5.3 Inversion

Une fois que le résidu a été prédit avec un bon modèle, il ne resterait plus qu'à inverser les étapes de stationnarisation. En appliquant l'inverse de la différentiation de période 24, à l'aide de la fonction $invdiff$, puis en ajoutant la composante saisonnière prédite sur lintervalle choisi. Il ne resterait alors plus qu'à dénormaliser en mumtipliant par la déviation standard et en ajoutant la moyenne.