Séries chronologiques

Université de Bordeaux

Radwan SARMINI DET SATOUF

March 29, 2020

Contents

1 Introduction

Dans ce document nous allons appliquer plusieurs modèles de séries chronologique(comme moyenne mobile, autorégressif et ARIAM) pour aider la police scientifique bordelaise à trouver des températures manquant de 15 Août, soit du 15 au 18 Août inclus.

2 Objectifs de la modélisation chronologique

Une série chronologique ou série temporelle est une suite d'observations d'un phénomène physique faites au cours du temps : température, consommation d'électricité, cours du pétrole, trafic Internet, ventes de voitures, côte de popularité du président, etc. Il s'agit d'une suite finie de valeurs réelles, indicées par un temps continu ou discret régulier, typiquement d'un signal échantillonné à une fréquence fixe.

3 Les étapes suivants sont utilisé pour étudier la series

- Importer les donnés et les regrouper dans des catégories
- Visualisation de la série
- Tester la saisonnalité - tendance
- Tester la stationnarité
- Premier essaye avec Moving average method (on modeilse seulment par raport à la premiére partie de donnés)
  - Modélisation
  - Prédiction
  - (Pour tester) Visualisation de la série avec un intervalles de confiance
  - Visualisation de la série avec un intervalles de confiance
- Deuxième essaye avec autorégressif
  (on modélise par rapport à les deux parties de donnés en ranche la deuxième partie est à l'inverse)
  - Modélisation
  - Prédiction
  - Visualisation de la série la prédiction de: AR & MA
- Dernière essaye en ARIAM
  - Modélisation et Prédictionn et les prémiere 8 valeurs (Test)
  - Modélisation et Prédictionn les derneirs 8 valeurs (Test)
  - Modélisation et Prédiction et Visualisation les valeurs manquants avec un intervalles de confiance
  -

4 Quels donnés

4.1 Présentation

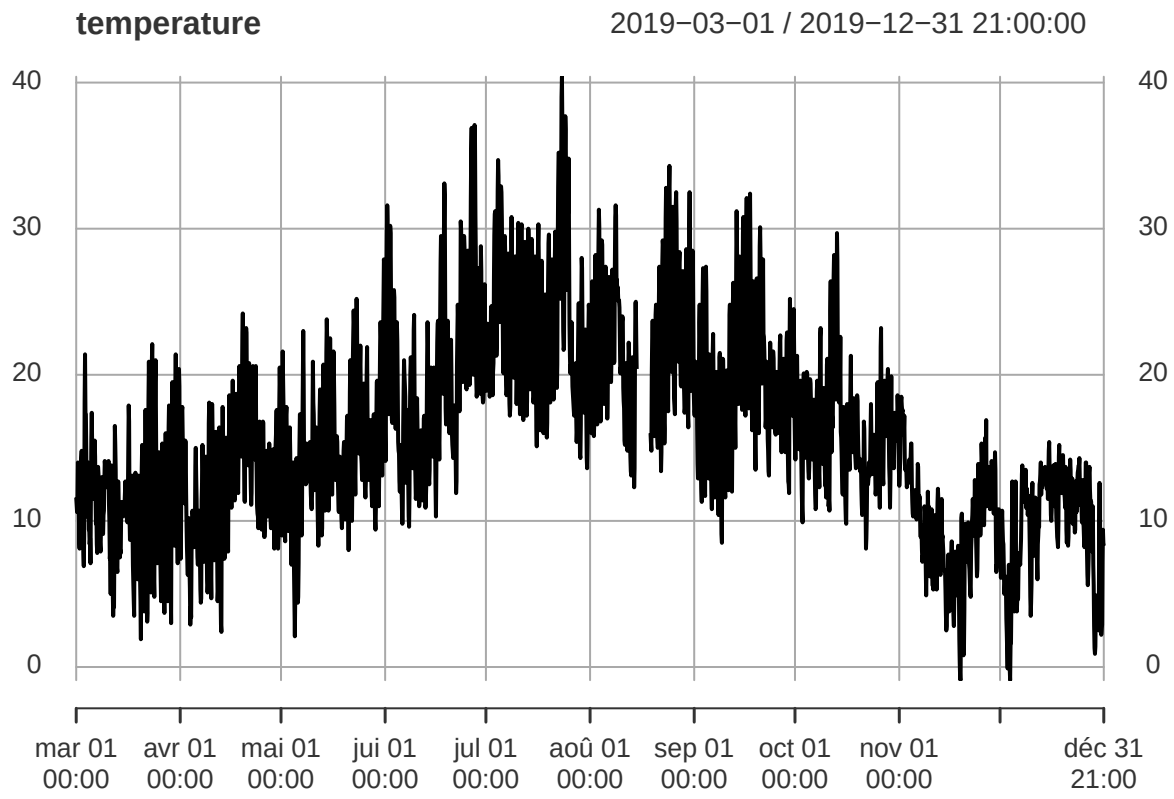
La police scientifique bordelaise enquête sur un meurtre commis cette année. Cependant, les experts manquent d'indices pour avancer et ont besoin de données manquantes telles que les températures du weekend du 15 Août, soit du 15 au 18 Août inclus. Par chance, des capteurs météorologiques ont été installés sur Bordeaux depuis Mars 2019. Cependant, les capteurs ne fonctionnaient pas sur le weekend en question. Coïncidence ? Ils ne pensent pas. . . Afin de prédire les températures manquantes sur cette période, ils requièrent vos compétences. A vous de jouer !

```
setwd("~/School/Miashs/MIASHS 6/seris chro/final project/markdown")
tmp<- read.csv("Data/tmp.csv")
```

4.2 Les données

Les données à disposition sont dans le fichier "tmp.csv" disponible sur Moodle. Les températures ont été relevées toutes les trois heures de Mars 2019 à Décembre 2019, soit 8 mesures par jours aux heures suivantes : 00h, 03h, 06h, 09h, 12h, 15h, 18h et 21h. La Figure ci-dessous illustre la série temporelle relevée avec, en Août, les données manquantes à prédire.

```
temperature <- xts::xts(tmp$temperature,
                        order.by=as.POSIXct(tmp$dateheure))
plot(temperature)
```



4.3 Regrouper et organiser les donnés

Tout D'abord on devise les donnés en quatre catégories :

- **first_part** : de 1ère température jusqu'à avent dernier la premier valeur manquant de la température.
- **last_part** : pour les températures après dernier température manquant jusqu'à la fin
- **tset_first** : méme que **first_part** mais sans les derniers 8 valeurs
- **test_second** : même que **last_part** mais sans les première 8 valeurs

```
last_one <- length(tmp$temperature) #de 1 à 2448
#tmp[is.na(tmp$temperature),c('X','temperature')] #show all the missing values
first_part<-c(1,1336) #premier section de donnés
last_part<- c(1369,last_one) #deuxeme section de donnés
tset_first<- c(1,(first_part[2]-8)) #test how good is our model for the first part
test_second <- c((last_part[1]+8),last_one)#test how good is our model for the second part
```

5 Saisonnalité & Tendance

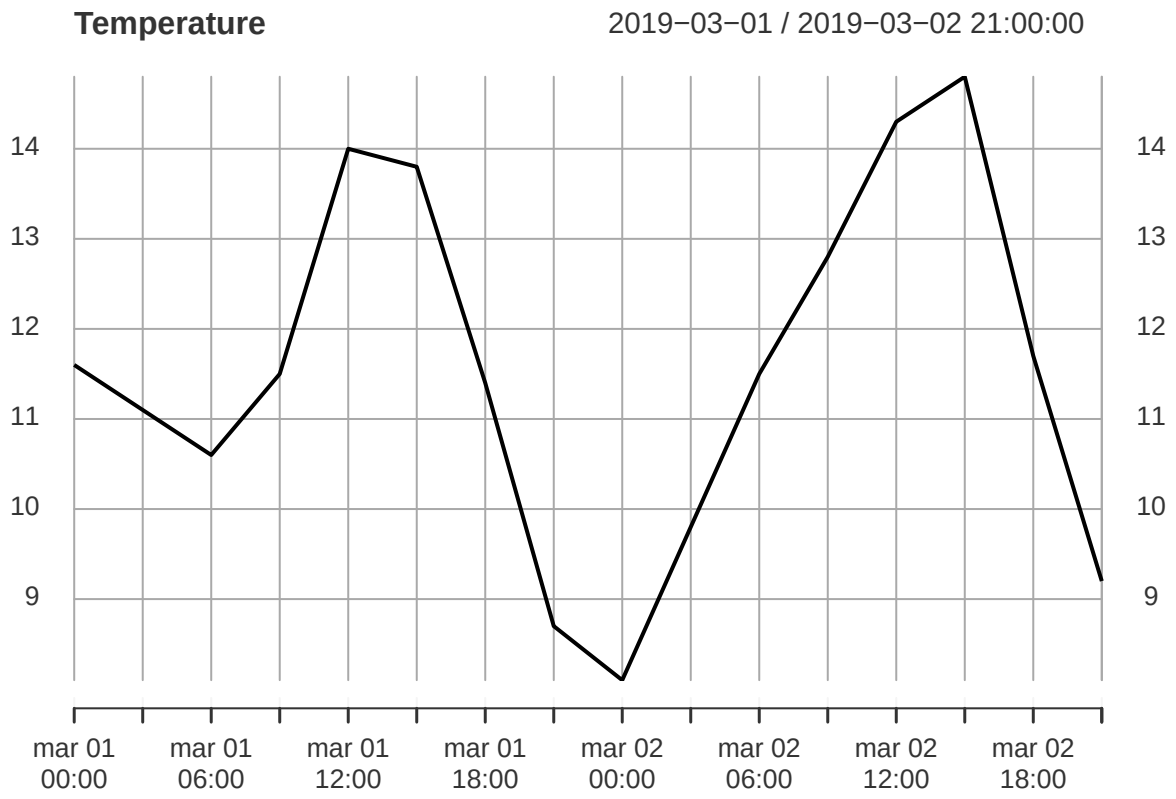
Sur la graphique précédent, on a pris une idée général et on remarque qui il n'y pas un tendance. Mais pour la saisonnalité ce n'est pas si claire Cependant, on peut penser qu'il y peut-être une

saisonnalité comme ces sont des donnés de la température quotidien donc on sait bien que le matin c'est froid et à midi la température devine plus chaud et le soir, la température diminue mais pour être sûr mathématiquement, on peut soit zoomer plus près pour une période plus court , soit regarder le autocorréligramme. Pour cela on peut écrire une fonction qui peut généraliser tous et éviter la répétition de code chaque fois. cette fonction prend en entré "choisir la catégories de donnés", "les donnés", "inverser ou pas", "lag", "différence" et "log" Et la sorti de cette fonction est les donnés demandé après quelques opérations.

```
main_data <- function(inverse=F,log=0,diff=0,lag=0,de_a=c(1,2448),data=tmp){
  P<-data$dateheure[de_a[1]:de_a[2]]
  X<-data$temperature[de_a[1]:de_a[2]]

  if (log){
    X<-log(X)}
  if (inverse){
    X<-rev(X)} #inverser les donnés
  if (lag){
    X<-diff(X,lag = lag)} #mettre un lag au donnés
  if (diff){
    X<- diff(X,differences = diff)}
  P<-data$dateheure[(de_a[1]+lag+diff):de_a[2]]
  Temperature<-  xts::xts(X, order.by=as.POSIXct(P))
  plot_<<- plot(Temperature)
  return(X)}
```

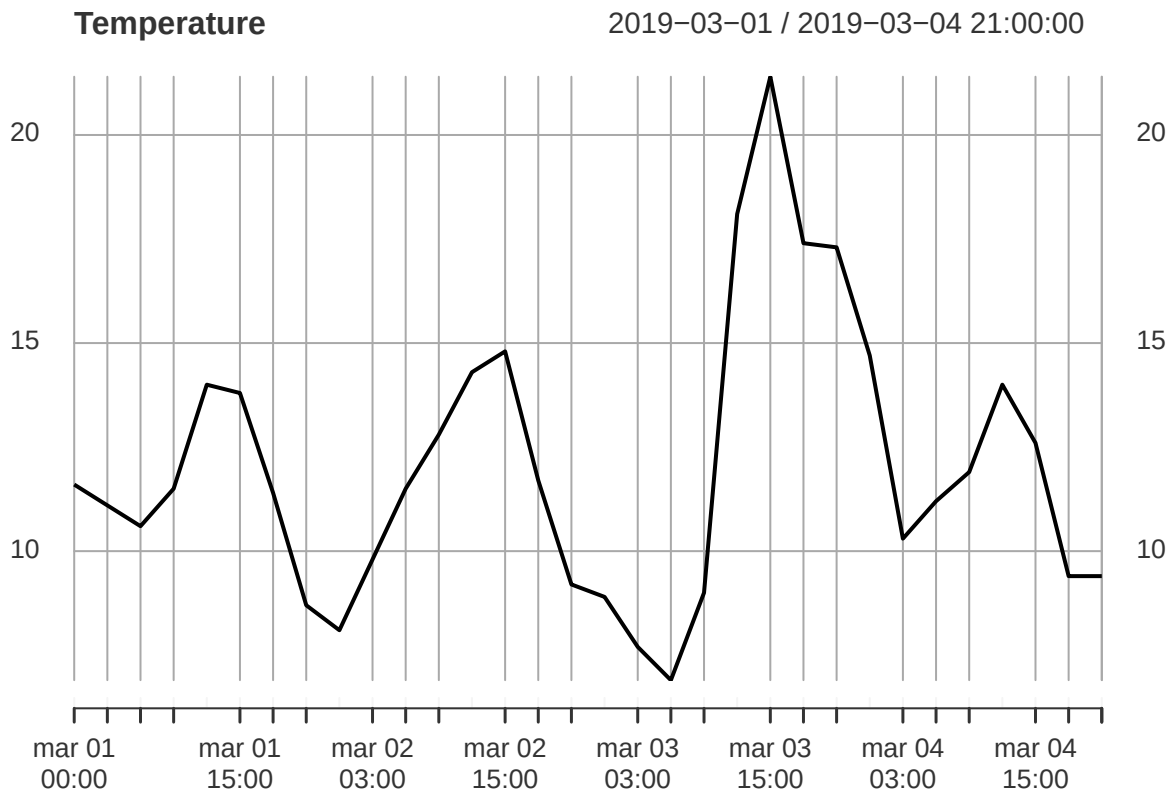
Alors on plot les premiers deux jours et on peut voir deux répétitions :

```
data_2j<- main_data(de_a=c(1,16))
plot_
```



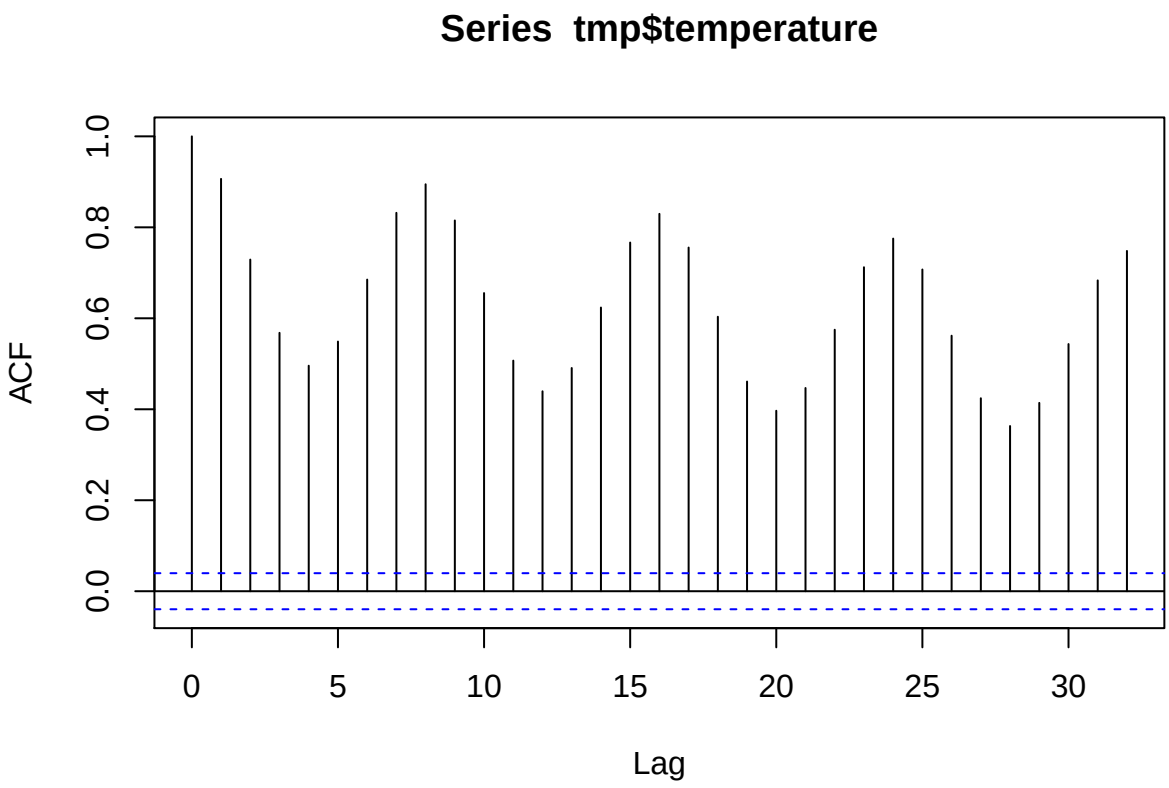
On voit bien qu'il y a 4 répétitions quand on plot les premiers quatre jours :

```
data_4j<- main_data(de_a=c(1,32))
plot_
```



Cependant, on peut aussi prouver par ça par son autocorréligramme.. dans le code suivant on prend tous les donnés et ignore les valeurs manquants.

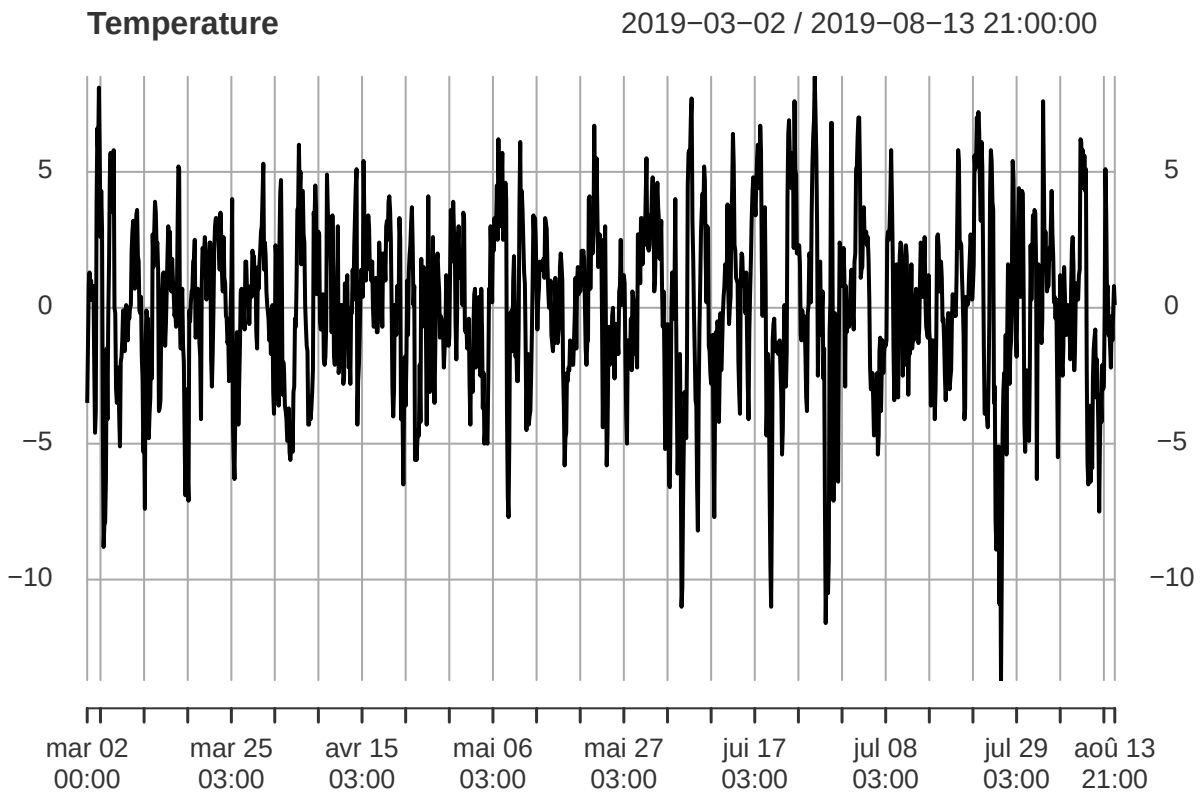
```
acf(tmp$temperature ,plot = TRUE, na.action = na.pass,lag.max = 32)
```



on voit bien la répétition sur tout les 8 valeurs.

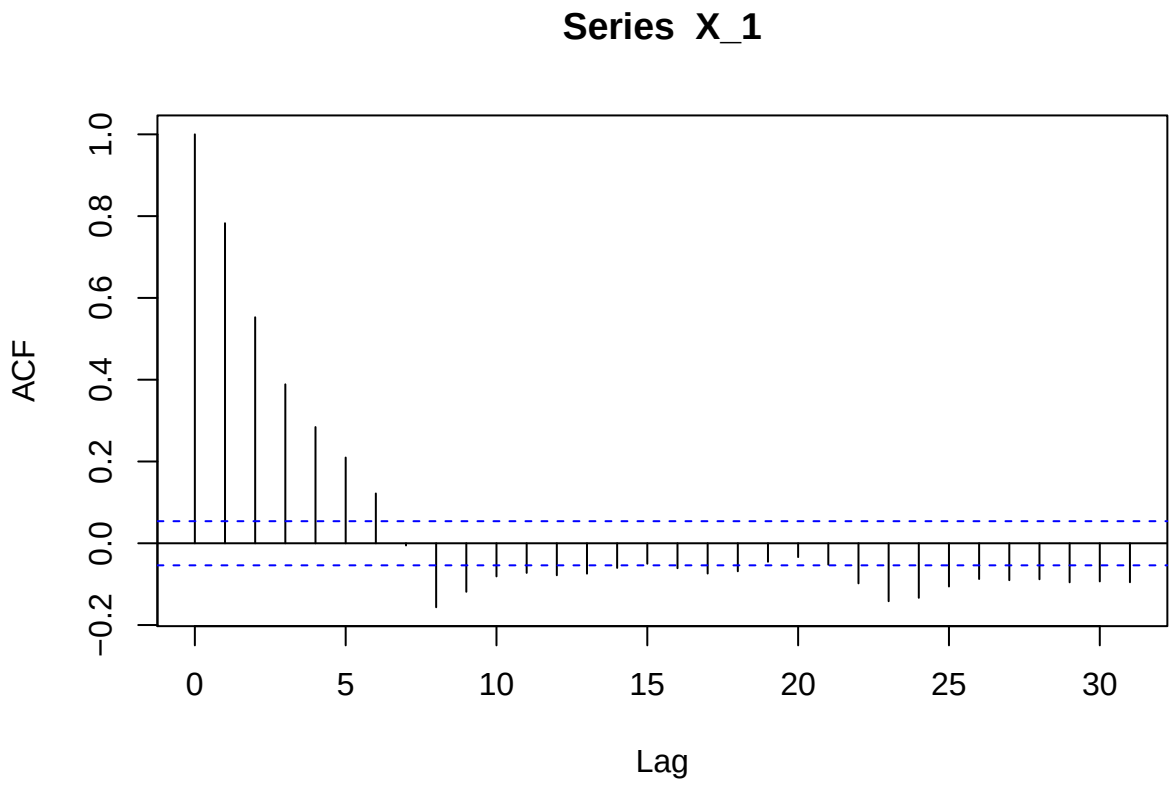
On choisi la série laquelle on va faire la analyse :

```
X_1 <- main_data(de_a=tset_first,lag=8)
plot_
```

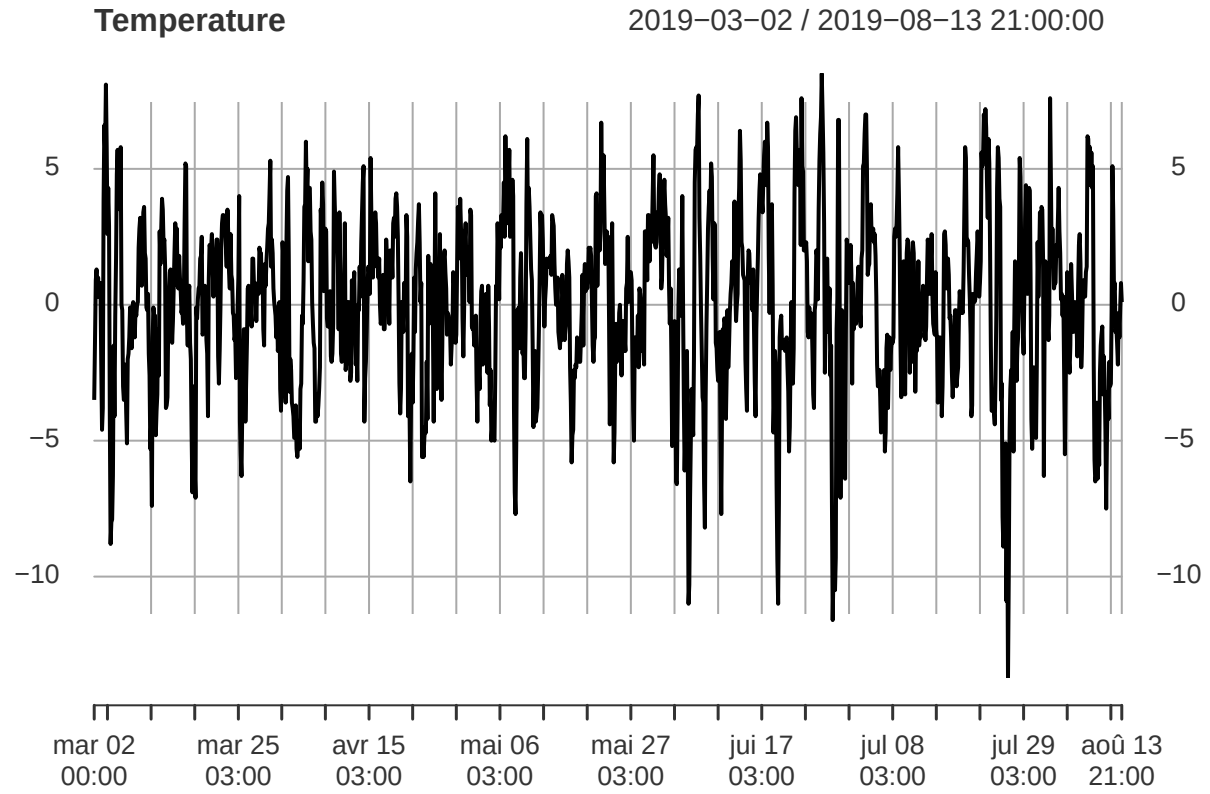


On vois bien l'amélioration par rapport si on fait ACF(X_1)

```
acf(X_1)
```




```
plot_
```



Et aussi le chronographe qui ressemble à un bruit blanc. on peut avoir l'impression qu'il est bien stationnaire.

Remarque : On a pris la première categorié pour analyser et tester

6 Test la stastionatité

Une série chronologique stationnaire est une série où les propriétés statistiques - comme la moyenne et la variance - sont constantes dans le temps.

6.1 Test ADF :

Le test Dickey Fuller augmenté (test ADF) est un test statistique utilisé pour tester si une série chronolgique donnée est stationnaire ou non. Il s'agit de l'un des tests statistiques les plus couramment utilisés lorsqu'il s'agit d'analyser le stastioonarité d'une série.

$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p+1} + \varepsilon_t,$

Où α est une constante, β le coefficient sur une tendance temporelle et p l'opérature retard du processus autorégressif. Imposer les contraintes $\alpha = 0$ et $\beta = 0$ correspond à la modélisation d'une marche aléatoire et l'utilisation de la contrainte $\beta = 0$ correspond à la modélisation d'une marche aléatoire avec une dérive. Par conséquent, il existe trois versions principales du test. si vous voulez savoir plus cliquer ici

6.2 Test KPSS

Le test de Kwiatkowski – Phillips – Schmidt – Shin (KPSS) détermine si une série chronologique est stationnaire autour d'une tendance moyenne ou linéaire, ou non stationnaire en raison d'une racine unitaire.

Le test se développe sur les hypothèses suivantes:

- H0: les données dérivent d'un processus stationnaire ou de processus stationnaires avec tendance en moyenne (tendance)
- H1: les données dérivent d'un processus non stationnaire Afin de définir les statistiques, considérons le modèle autorégressif suivant:

$x_{(t)} = v_{(t)} + \theta v_{(t-1)}\ y_{(t)} = \xi + \beta y_{(t-1)} + x_{(t)}$

La statistique de test est calculée à l'aide des multiplicateurs de Lagrange:

$S = \frac{\left(T^{-2} \sum_{t=1}^{T} \hat{S}_t^2\right)}{\sigma_\varepsilon^2}$

où

- T est la taille de l'échantillon
- σ_ε^2 est la variance à long terme
- $\hat{S}_t = \sum_{i}^{t} e_{(t)}$ est la somme partielle des erreurs de la régression

6.3 On applique les test

Cette fonction prend les donnés et dit si la série est stationnaire ou pas pour chaque test :

```
options(warn=-1) #Ca aide de ne pas afficher "Warning message:"
library("tseries")
```

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

```
test_ <- function(data){
  ADF <- adf.test(data)
  KPSS <- kpss.test(data, null="Level")
  return(sprintf(c("Le test ADF donne %s pour la stastionarité "
                   ,"Et le test KPSS donne %s pour la stastionarité"),
                   ADF$p.value < 0.05 , KPSS$p.value >= 0.05 ))}
```

On choisir la série et on applique et on test :

```
test_(X_1)
```

```
## [1] "Le test ADF donne TRUE pour la stastionarité "
## [2] "Et le test KPSS donne TRUE pour la stastionarité"
```

7 Moving average

La moyenne mobile est un type de moyenne statistique utilisée pour analyser des séries ordonnées de données, le plus souvent des séries chronolgique/temporelles.

Le modèle de série chronologique du formulaire:$X_t = \sum_{j=0}^{q} b_j \varepsilon_{t-j}$
où ε_t - bruit blanc, b_j - paramètres du modèle (b_0 peut être considéré comme égal à 1 sans perte de généralité). Le processus de bruit blanc peut être formellement considéré comme un processus de moyenne mobile d'ordre zéro - MA (0). En pratique, le processus de moyenne mobile de premier ordre MA(1) est plus utilisé : $X_t = \varepsilon_t + b\varepsilon_{t-1}$.

7.1 Modélisation

le processus $MA(q)$ où q est l'ordre chronolgique.
Tout d'abord le clé principale pour bien modéliser est L'AIC , plus l'AIC est petit plus la model est bien. Alors oui, on doit minimiser l'AIC dans un model pour choisir le bon ordre de q et d où d est l'opératour différence.

Bien sur qu'on peut choisir q à partir de son autocorrélogramme alors c'est bien lié à notre observation et notre goût statistique, cependant j'ai lassé cette méthode de modélisation pour la section suivant **autorégressif**.

Et bien, Pour minimiser l'AIC on peut créer deux boucles pour d et q et laisser le code tourne pour tester plusieurs ordres jusqu'elle trouve le plus petit AIC.
La fonction est :

```
min_aic<-function(X,P,D,Q) {
  min_p=P[1]
  min_d=D[1]
  min_q=Q[1]
  cpt=0
  for(d in D[1]:D[2]) {
    for(p in P[1]:P[2]) {
      for(q in Q[1]:Q[2] ) {
        tryCatch( {
          Mod<- arima(X,order = c(p,d,q),method='ML')
          Mod____<<-Mod
          aic=Mod$aic
        },error=function(e){})
        if(cpt==0){
          aic_min=aic
```

```
          print(aic_min)}
        else{
          if(aic<aic_min){
            aic_min=aic
            min_p=p
            min_d=d
            min_q=q
            print(sprintf
                  ('Tour %s et les orders sont : %s , %s , %s et aic est %1f :'
                          ,cpt,min_p,min_d,min_q,aic_min))
        }}
        cpt = cpt+1
      }
    }
    print("yes")
  }
  aic_<<-aic_min
  return(c(min_p,min_d,min_q))
}
```

On exécute la fonction avec un opératoire différence de 0 jusqu'à 9 et l'ordre q de 0 à 30 : pour_model<- min_aic(X_1,c(0,0),c(0,9),c(0,30)) Et on trouve que le plus petite AIC est quand d = 0 et quand q = 30

Alors notre model pour la meilleur AIC est ARIMA(0,0,30) avec un aic = 4882.53

```
MA_model <-arima(X_1,order=c(0,0,30))
```

7.2 Prédiction

Après créer le model, on va à l'étape suivant **Prédiction** En effet on peut utiliser la méthode **predict** pour prédire les 8 valeurs qu'on voudrait comparer avec la série de mère. Dans notre série on doit faire attention puisque nous avons utiliser l'opérateur retarde de 8 valeurs. Par conséquent on doit la transforme au forme normal.

On peut ulitser l'équation suivent :

$Z_t = X_t - X_{(t-12)}$

$Y_t = Z_t + Y_{(t-12)}$

Et pour faire ça on peut utiliser cette fonction (fait mains aussi comme tout code le) popur rédire les valeurs et aussi construire un intervalles de confiance de 95%:

```
prediction_ <- function(model,pr_=8,data_n=tset_first[2],sta_p=100) {
  MA_predict<- predict(model , pr_ )
  data_<-data_n
  MA_X<-MA_sup<-MA_inf<- tmp$temperature
  for(i in (data_+1):(data_+pr_)){
```

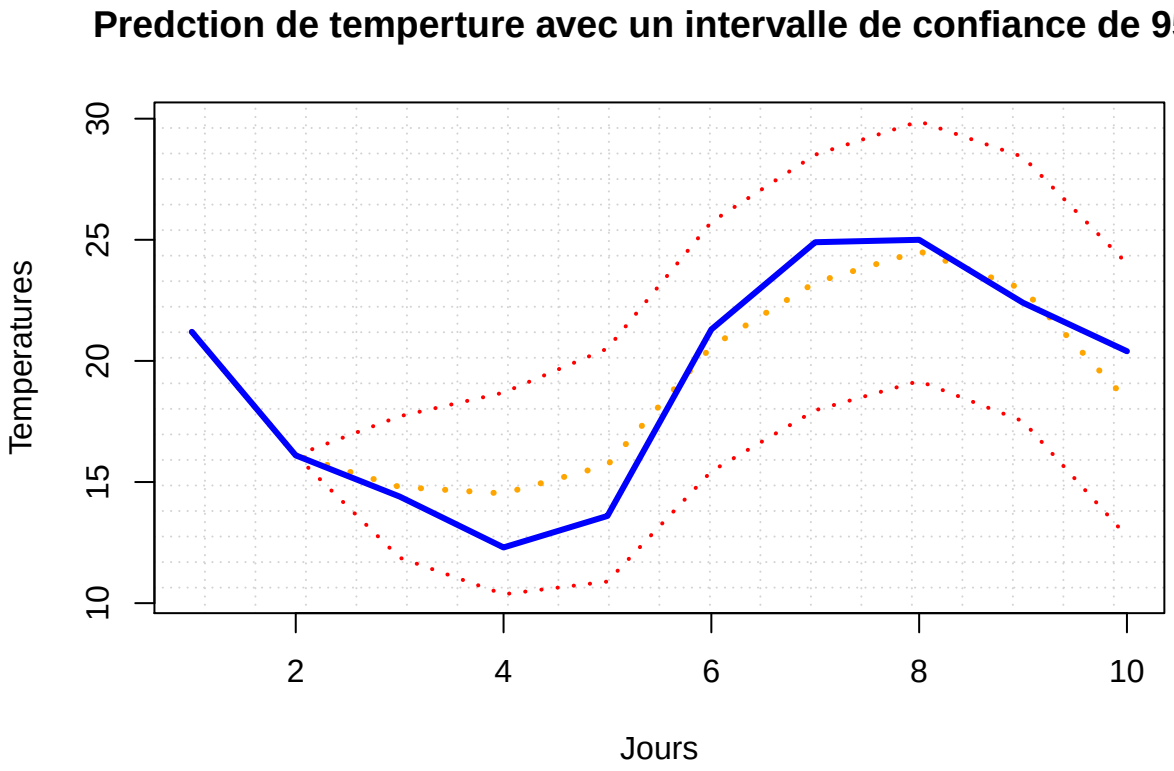
```
    MA_X[i]<-MA_predict$pred[i-data_]+MA_X[i-8]
    MA_inf[i]<-MA_predict$pred[i-data_]-qnorm(0.975)*MA_predict$se[i-data_]+MA_X[i-8]
    MA_sup[i]<-MA_predict$pred[i-data_]+qnorm(0.975)*MA_predict$se[i-data_]+MA_X[i-8]
  }
  end_p<- pr_+data_
  sta_p<- sta_p
  plot(tmp$temperature[sta_p:(end_p)],
      type = 'l',col='blue',
      main="Predction de temperture avec un intervalle de confiance de 95%"
      ,ylab="Temperatures",xlab="Jours",
      ylim = c(min(MA_inf[sta_p:end_p]),
      max(MA_sup[sta_p:end_p])),panel.first = grid(20))
  lines(MA_X[sta_p:end_p],type = 'l',col='orange',lty=3,lwd=3)
  lines(MA_sup[sta_p:end_p],type='l',col='red',lty=3,lwd=2)
  lines(MA_inf[sta_p:end_p],type='l',col='red',lty=3,lwd=2)
  lines(tmp$temperature[sta_p:end_p],lwd=3,
       type = 'l',col='blue')
  MES<- (1/8)*sum((MA_X[(data_+1):end_p]
                   -tmp$temperature[(data_+1):end_p])^2)
  if (!is.na(MES)){
    print(sprintf(
      "le MES de la %s valeurs par rapport à la série mére = %s"
      ,pr_,MES))}
  return(MA_X[(data_n+1):(data_n+pr_)])

}
```

7.3 (Pour tester) Visualisation de la série(8 Valeurs) avec un intervalles de confiance

On utilise la fonction **production_** qui prend en entrée le model , nombre de predetion , le numbre de de donnés et à quelle valeur il aréte.

Alors on excéute le code et :

```
prediction_(MA_model,8,tset_first[2],1327)
```




```
## [1] "le MES de la 8 valeurs par rapport à la série mére = 2.17640387327436"
```

```
## [1] 14.79655 14.52950 15.70121 20.58952 23.23768 24.52995 22.96530 18.38515
```

Dans la graphique ci-dessus on on remarque 4 linges :

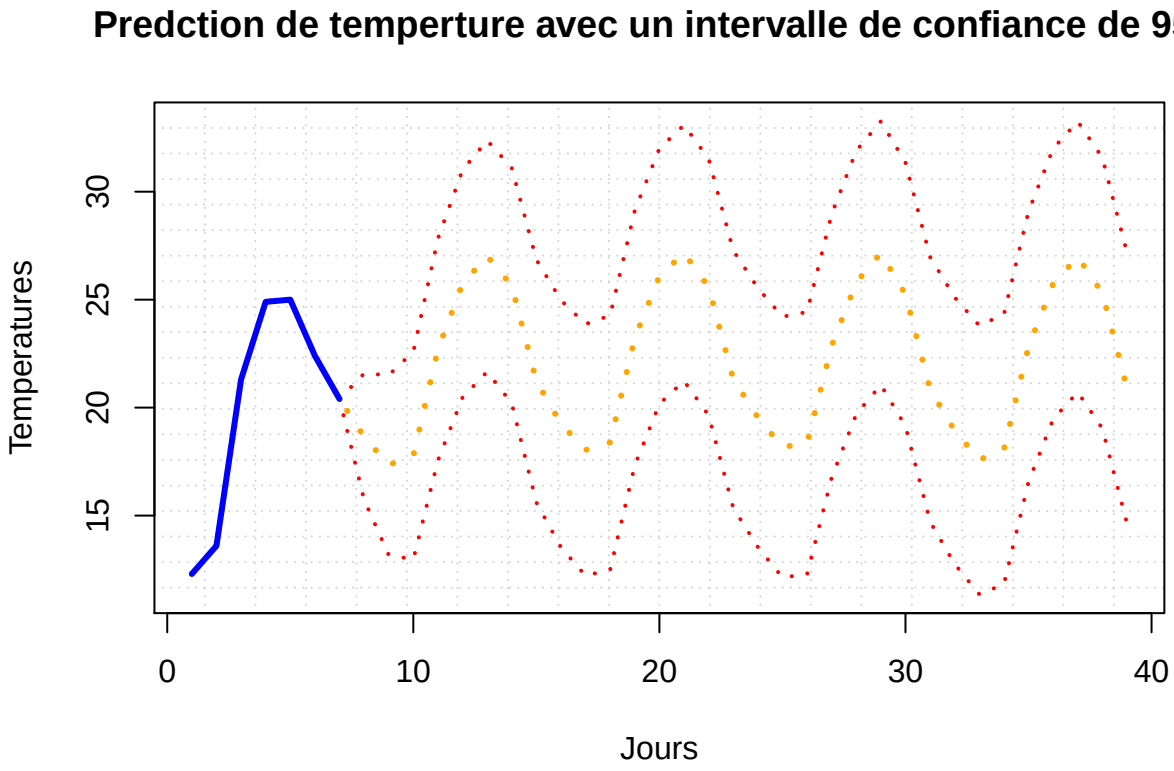
- En bleu c'est la série mère(la vrai température)
- En orange c'est notre prédiction avec notre model MA(30) et AIC = 4882.53
- En fin en rouge ces sont les intervalle de confiance de notre prédiction (C'est magnifique la statistique+R on peut faire confiance à cette intervalle de confiance)

Ici on prédire seulement 8 valeurs pour tester l'efficacité de notre model

7.4 Visualisation de la série(32 Valeurs) avec un intervalles de confiance

En fin, voilà les 32 valeurs manquants :

```
#maintantit on predre la partie 1 complet 32 Valeurs
X_2 <- main_data(de_a=first_part,lag=8)
MA_model_2 <-arima(X_2,order=c(0,0,30))
#avec le modele complet on trouve l'aic =4882.53 qui ne change pas vraiment
MA_tmp <-prediction_(MA_model_2,32,first_part[2],1330)
```



Autorégressif

Le modèle autorégressif (AR-) est un modèle de série chronologique dans lequel les valeurs de la série chronologique au moment dépendent linéairement des valeurs précédentes de la même série. Le processus autorégressif d'ordre p (processus AR (p)) est défini comme suit:

$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \varepsilon_t$

où $a_1, \ldots, a_p$ - paramètres du modèle (coefficients d'autorégression), c est une constante (souvent pour simplifier, elle est supposée être nulle), et ε_t bruit blanc. L'exemple le plus simple est le processus autorégressif du premier ordre du processus AR (1): $X_t = c + rX_{t-1} + \varepsilon_t$ Pour ce processus, le coefficient d'autorégression coïncide avec le coefficient d'autocorrélation du premier ordre.

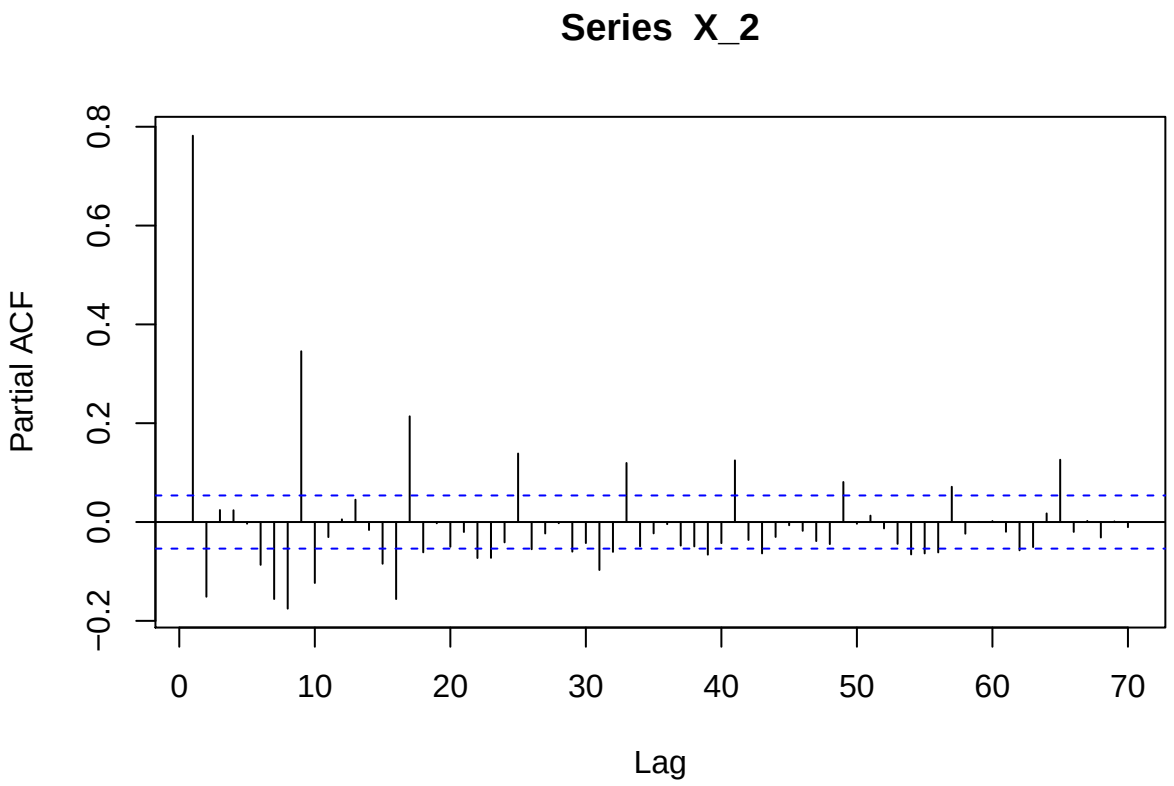
Un autre processus simple est le processus Yule - le processus AR (2): $Tx_T = C + a_1 X_{T-1} + a_2 X_{T-2} + \varepsilon_T$

7.5 Modélisation

Pour determiner l'ordre de AR model on trace le PACF pour 100 valeurs et on trouve qu'il y plusieurs ordres sont intéressant pour bien choisire
on peut testet et chercher la plus petit valeur d'AIC

Alors d'après ce autocorrélogramme on peut essyer plusieurs ordre

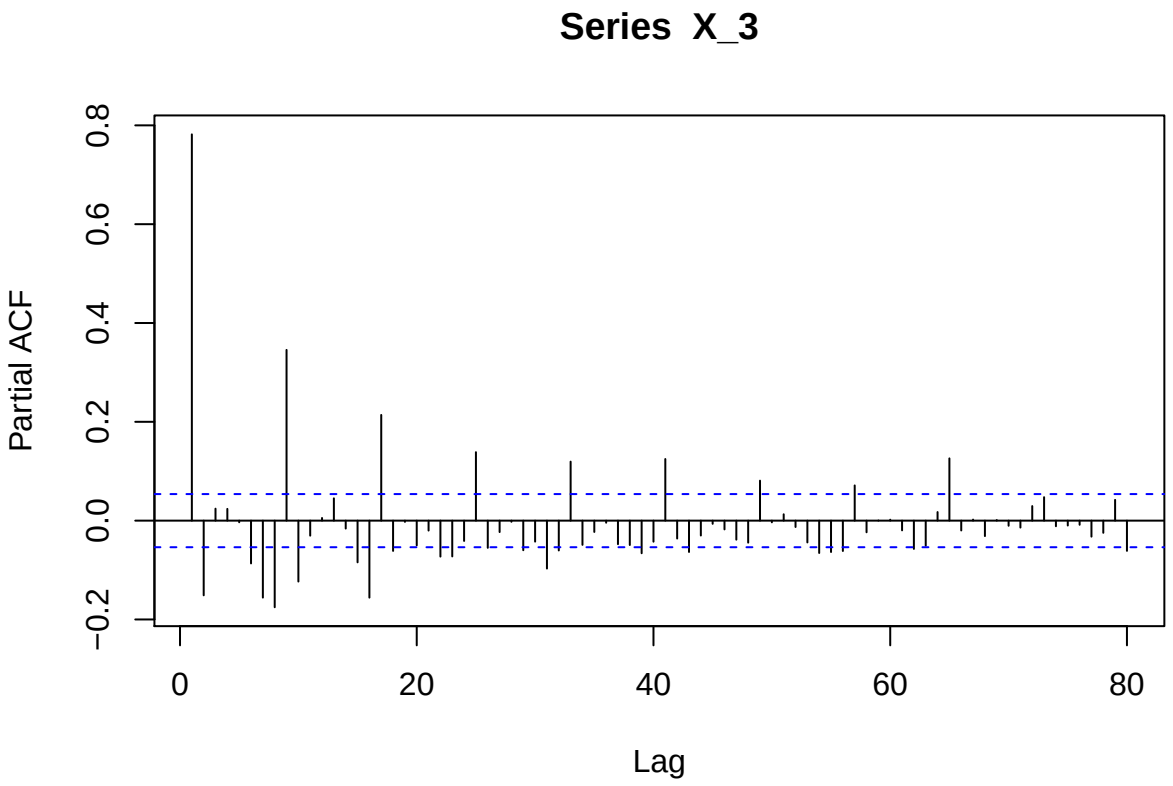
```r
pacf(X_2,lag.max = 70)
```




```r
#model.AR <- arima(X_2, c(49,0,0))
#model.AR$aic #5038.33 order 49

#model.AR <- arima(Y, c(65,0,0))
#model.AR$aic #4999.464 order 65
```

- Pour la premiére partie de donnés **first_part**
  - Pour un ordre de 40 on trouve que l'AIC = 5038.33
  - Pour un ordre de 65 on trouve que l'AIC = 4999.464

```r
X_3 <- main_data(inverse=T,de_a=first_part,lag=8)
pacf(X_3,lag.max = 80)
```




```
#model.AR <- arima(X_3, c(73,0,0))
#model.AR$aic #3822.84 order 73
```

- Pour la deuxieme partie de donnés **last_part** par d'aprés inverser les donnés on trouve :
  - un ordre de 73 on trouve que l'AIC = 3822.84

7.6 prediction

Pour prédire on utilise la fonction **predict** comme l'habitude Et on n'oublie pas de remettre notre prédiction à la normal comme on a pris lag = 8 encore. on s'intéresse à prédire X et pas seulement Z pour cela on peut faire un boucle avec notre fonction **prediction_**

Note : Je n'exécute pas le code de Ar(73) car il va prendre 3h chaque fois je corrige un erreur que je trouve dans le document sur Markdown. Alors en cas de doute vous pouvez voir mon dernier envoi ou essayer le code mais ça va prendre beaucoup trop de tempes pour tourner.

```
#model.AR2 <- arima(Y, c(65,0,0))
#model.AR2$aic# order 65 aic = 4999.464
#acf(model.AR$residuals)
#predictY1.AR2 <- predict(model.AR2 , 32 )
#predictY1.AR2 #5038.33
```

```
#Z.AR2 <- X
#for( i in 1337:1369){ Z.AR2[i] <- predictY1.AR2$pred[i-1336] + Z.AR2[i-8]}
#plot.ts(Z.AR2[1337:1368],ylab="temperatures")
#Z.AR2[1337:1368]
#mini_2=1369 #
#maxi_2=2448 #
#X22<-tmp[mini_2:maxi_2,'temperature']
#tail(X22)
#X33<-rev(X22) #je prende l'invers
#head(X33)
#pacf(X33)
#Y33 <- diff(X33,lag=8) #lag de 8
#pacf(Y33, lag.max=75)
#modelY33<- arima(Y33 ,c(73,0,0))
#modelY33$aic
#predY33  <- predict(modelY33, 16)
#aic = 3822.84, ordre=  73
#Z.ARY33 <- X33
#for( i in (length(X22)+1):(length(X22)+16)){ Z.ARY33[i] <- predY33$pred[i-length(X33)]  + #Z.AR
#Z.ARY33[(length(X22)+1):(length(X22)+16)]
#inverser pour etre plus correct
#Z_final <- rev(Z.ARY33[(length(X22)+1):(length(X22)+16)])
#aic = 3822.84
#Les_deux_valure<- c(Z.AR2[1337:1352],Z_final)
#plot.ts(Les_deux_valure,ylab="temperatures",main="Part 1 + (Part 2 à l'inverse )")
```

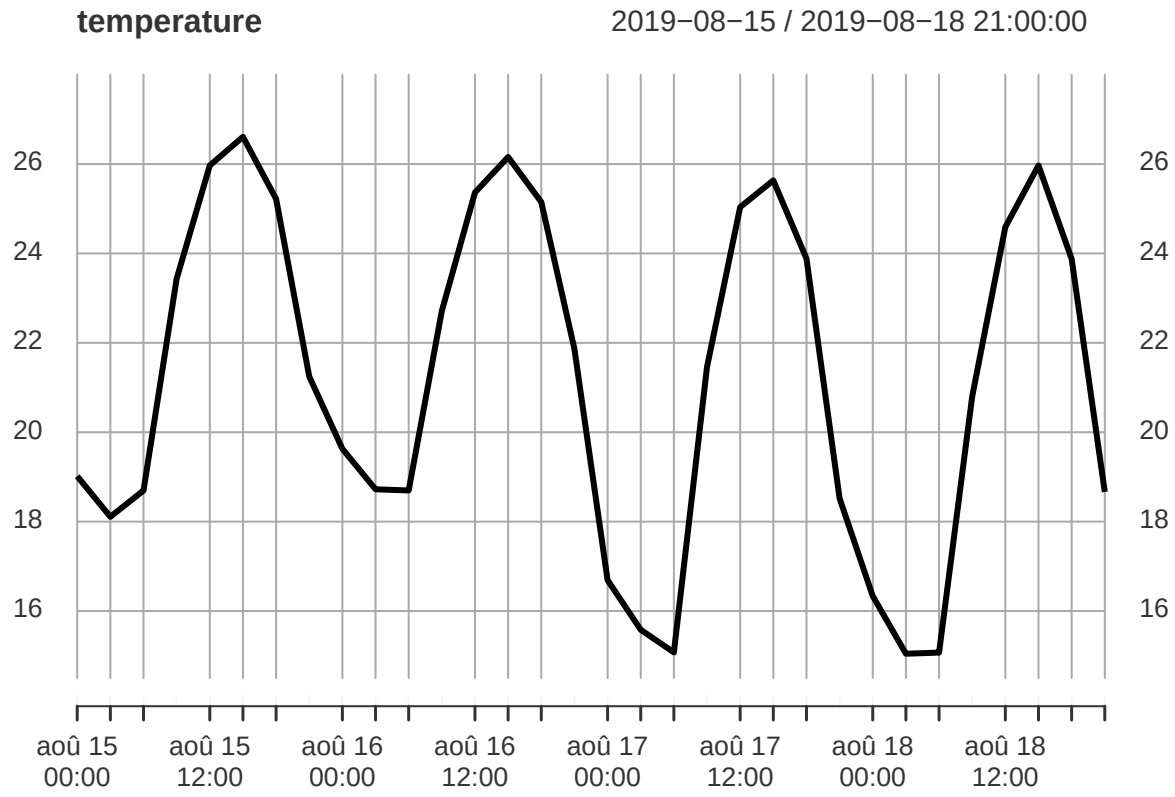
7.7 Visualisation de la série la prédiction de: AR & MA

ET donc on utilise les sortir de code pour la prédiction de la première partie(1:16) et la deuxième partie(17:32) des donnés qui sont à l'inverse. on trouve les valeurs :

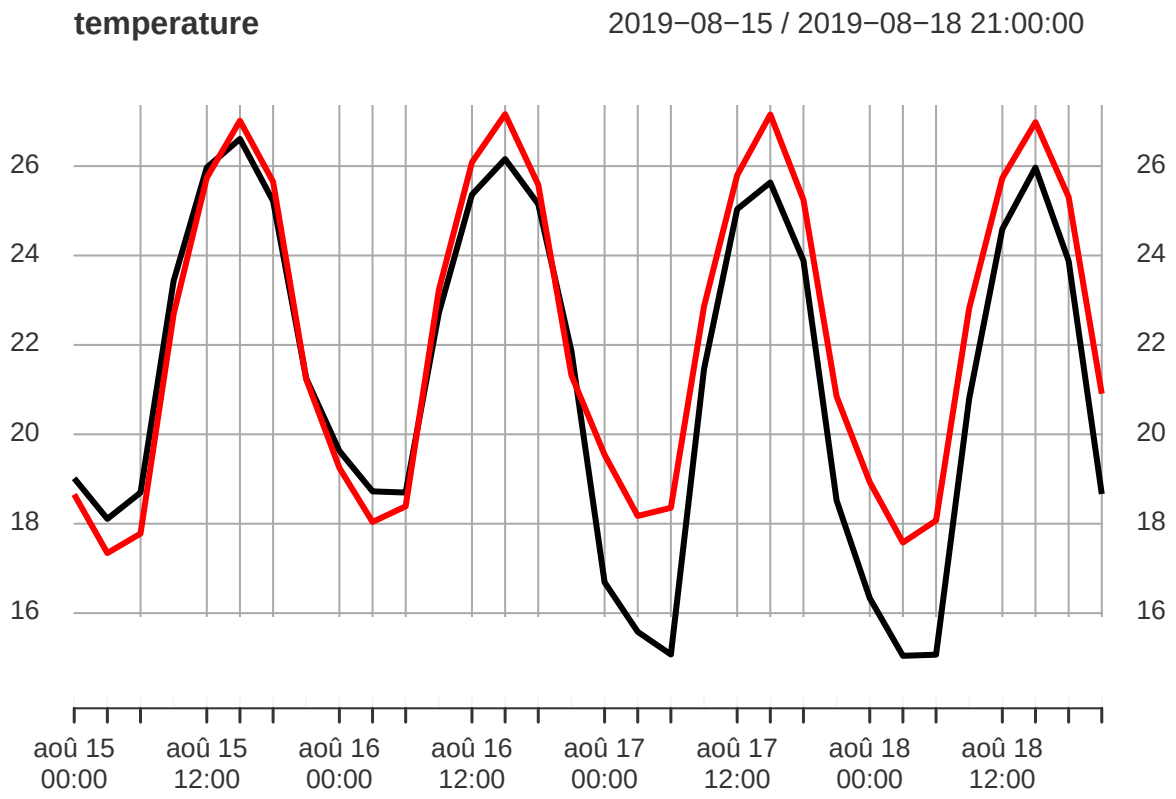
```
Ar_tmp<-c(19.01625 ,18.10806, 18.69576, 23.43390, 25.96839, 26.60851, 25.21964,
  21.24813, 19.62780, 18.72355, 18.69688, 22.71182, 25.36334, 26.15806,
 25.14977, 21.86366, 16.69093, 15.58118, 15.07383, 21.46077, 25.03677,
25.63444, 23.88621, 18.51874, 16.33463, 15.04461, 15.06990, 20.80072,
 24.59275, 25.96713, 23.87793, 18.66418)
# et MA
f_manq <- (first_part[2]+1)
l_manq<- (last_part[1]-1)

#plot(temperature)
```

ET on Visualise les AR & MA, on trouve pour certains températures, il une grande différence entre elles, et pour d'autres températures sont sont trés proche

```
temperature <- xts::xts(Ar_tmp,
                        order.by=as.POSIXct(tmp$dateheure[f_manq:l_manq]))
MA_tmpp <- xts::xts(MA_tmp,
                        order.by=as.POSIXct(tmp$dateheure[f_manq:l_manq]))
plot(temperature,lwd=3,ylim = c(14.5,28))
```




```
lines(MA_tmpp ,type = 'l',col='red',lwd=3)
```



- En rouge les températures ont modelé avec MA(30)
- En noire les températures ont modelé avec AR(65) pour les 16 premières valeurs et AR(73) pour les 16 derniers valeurs

8 ARIAM

ARIMA (Autorégressive intégrée moyenne mobile, parfois modèle Box-Jenkins, méthodologie Box-Jenkins) - un modèle intégré d'autorégression - moyenne mobile - modèle et méthodologie d'analyse des séries chronologiques. Il s'agit d'une extension des modèles ARMA pour les séries chronologiques non stationnaires, qui peuvent être rendues stationnaires en prenant des différences d'un certain ordre de la série temporelle d'origine (les séries chronologiques dites intégrées ou stationnaires par différence).

$ ARIMA (p, d, q) $ signifie que les différences de séries chronologiques d'ordre d :

$$\triangle^d X_t = c + \sum_{i=1}^{p} a_i \triangle^d X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t$$

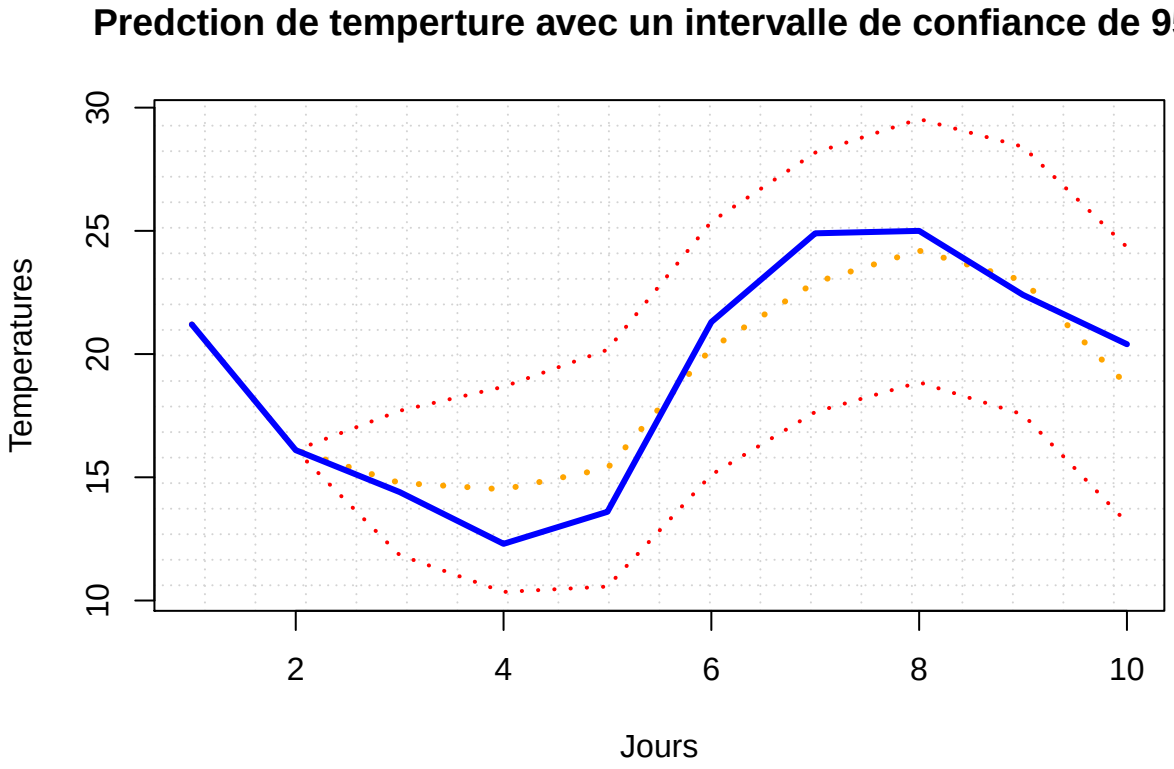
où:

- ε_t - séries chronologiques stationnaires;
- c, a_i, b_j - paramètres du modèle.
- $\triangle^d$ est l'opérateur de la différence dans les séries chronologiques d'ordre d.

8.1 Modélisation et Prédictionn les prémiere 8 valeurs (Test)

Tout d'abord on va tester notre modele avec les vrais valeurs

```
ARIMA_X8 <- main_data(de_a=tset_first,lag=8)
#pour_model<- min_aic(ARIMA_X,c(0,20),c(0,9),c(0,20))
ARIMA_model8 <-arima(ARIMA_X8,order=c(12,0,11))
#"20,2,20 "Tour 30 et les orders sont : 12 , 0 , 11 et aic est 4874.780577 :"

ARIMA_pre8 <-prediction_(ARIMA_model8,8,tset_first[2],1327)
```




```
## [1] "le MES de la 8 valeurs par rapport à la série mére = 2.1174998885771"
```

"20,2,20"Tour 30 et les orders sont : 12 , 0 , 11 et aic est 4874.780577 :"

- En bleu c'est la série mère(la vrai température)
- En orange c'est notre prédiction avec notre model ARIMA(12,0,11) et AIC = 4874.78
- En fin en rouge ces sont les intervalle de confiance de notre prédiction

8.2 Modélisation et Prédictionn les derneirs 8 valeurs (Test)

Ensuit on va prédire la deuxième partie des valeurs mais à l'inverse après utiliser la fonction min_aic :

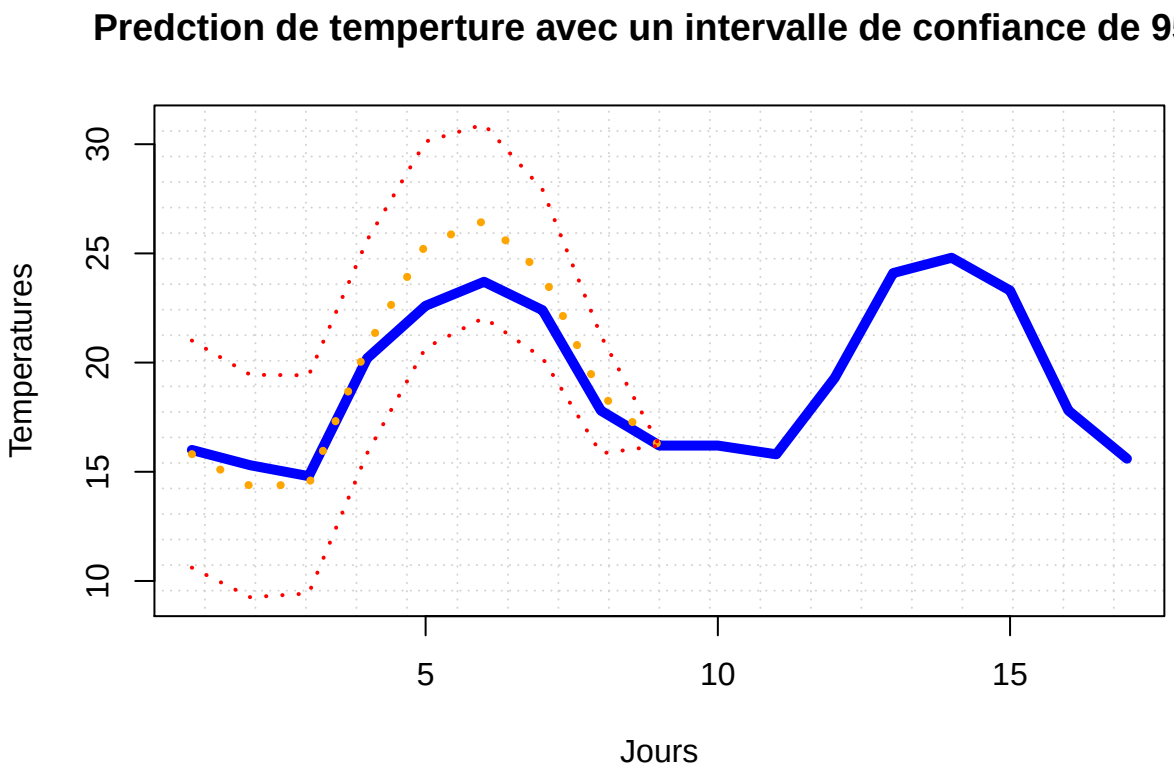
#pour_model_<- min_aic(ARIMA_X,c(0 ,20),c(0,2),c(0,20))

On trouve que ARIMA(11, 0, 10) est le meilleure par rapport à l'AIC = 3789.63 cependant, cette fois on va être un peu plus moderne puisque, on va utiliser la bib forecast pour prédire directement tous les values y compris l'intervalle de confiance de 95 %

```
#ARIMA_X<- main_data(inverse=T,de_a=test_second,lag=8)
#pour_model_<- min_aic(ARIMA_X,c(0 ,20),c(0,2),c(0,20))
library(forecast)
ARIMA_X_8 <- main_data(inverse=T,de_a=test_second)
test_(ARIMA_X_8)
```

```
## [1] "Le test ADF donne TRUE pour la stastionarité "
## [2] "Et le test KPSS donne TRUE pour la stastionarité"
```

```
#pour_model_<- min_aic(ARIMA_X,c(0 ,20),c(0,2),c(0,20))
pdqParam = c(11, 0, 10)
manualFit <- arima(ARIMA_X_8, pdqParam, seasonal = list(order = c(0,0,0), period = 8))
nu_=8

for_=forecast(manualFit,nu_)
value =tmp$temperature[last_part[1]+8]
mean_<-c(rev(for_$mean),value)
inf<-c(rev(for_$lower[,"95%"]),value)
sup<-c(rev(for_$upper[,"95%"]),value)



plot(tmp$temperature[last_part[1]:(last_part[1]+16)],type = "l",col='blue',
     main="Predction de temperture avec un intervalle de confiance de 95%"
     ,ylab="Temperatures",xlab="Jours",
     ylim = c(min(for_$lower[,"95%"]),
              max(for_$upper[,"95%"])),panel.first = grid(20),lwd=5)
lines(mean_,type="l",col='orange',lty=3,lwd=4)
lines(inf,type="l",col='red',lty=3,lwd=2)
lines(sup,type="l",col='red',lty=3,lwd=2)
```



- En bleu c'est la série mère(les vrai températures)
- En orange c'est notre prédiction avec notre model ARIMA(11,0,10) et AIC = 3789.63
- En fin en rouge ces sont les intervalle de confiance de notre prédiction

8.3 Modélisation et Prédictionn les valeurs manquants

Ensuite on va prédire les valeurs manquants directement et tracer l'intervalle de confiance :

Note : les choix d'ordre et des sessions et test de stationnarité été déjà choisir dans les chapitres précédents

```
SARIMA1 <- main_data(inverse=F,de_a=first_part)

p_o_m1 =c(14,0,14)
SARIMA_fit1 <- arima(SARIMA1, p_o_m1, seasonal = list(order = c(0,0,0), period = 8))
SARIMA_fit1$aic
```

```
## [1] 4952.281
```

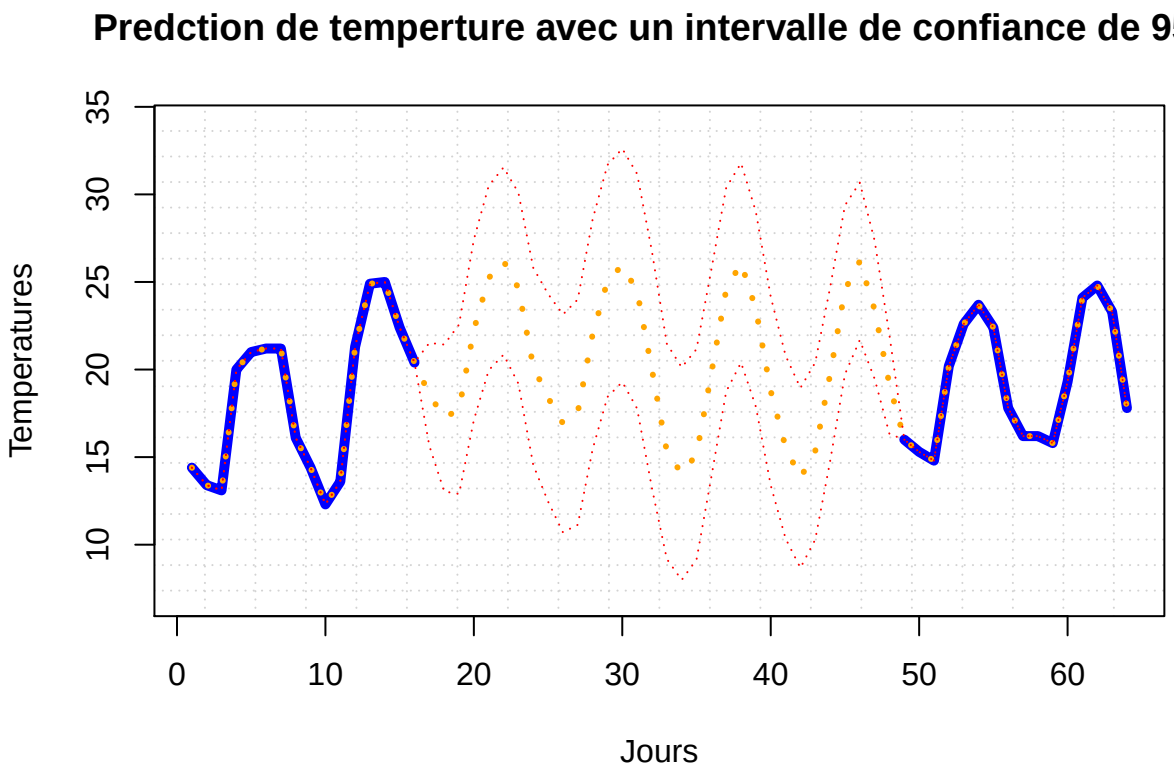
```
nu_=16
SARIMA1_for=forecast(SARIMA_fit1,nu_)
```

```r
value1 =tmp$temperature[(first_part[2]-15):first_part[2]]
mean1<-c(value1,SARIMA1_for$mean)
inf1<-c(value1,SARIMA1_for$lower[,"95%"])
sup1<-c(value1,SARIMA1_for$upper[,"95%"])
##############################################################################
SARIMA2 <- main_data(inverse=T,de_a=last_part)
p_o_m2 = c(7, 0, 10)
SARIMA_fit2 <- arima(SARIMA2, p_o_m2, seasonal = list(order = c(0,0,0), period = 8))
SARIMA_fit2$aic
```

```
## [1] 3900.621
```

```r
SARIMA2_for=forecast(SARIMA_fit2,nu_)
value2=tmp$temperature[last_part[1]:(last_part[1]+15)]
mean2<-c(mean1,rev(SARIMA2_for$mean),value2)
inf2<-c(inf1,rev(SARIMA2_for$lower[,"95%"]),value2)
sup2<-c(sup1,rev(SARIMA2_for$upper[,"95%"]),value2)
```

```r
plot(tmp$temperature[(first_part[2]-15):(last_part[1]+15)],type = "l",
     main="Predction de temperture avec un intervalle de confiance de 95%"
     ,ylab="Temperatures",xlab="Jours",col='blue',
     ylim = c(min(7),
              max(34)),panel.first = grid(20),lwd=5)

lines(mean2,type="l",col='orange',lty=3,lwd=3)
lines(inf2,type="l",col='red',lty=3,lwd=1)
lines(sup2,type="l",col='red',lty=3,lwd=1)
```



- En bleu c'est la série mère(les vrai températures).
- En orange c'est notre prédiction.
- En fin en rouge ces sont les intervalle de confiance de notre prédiction.

9 conclusion

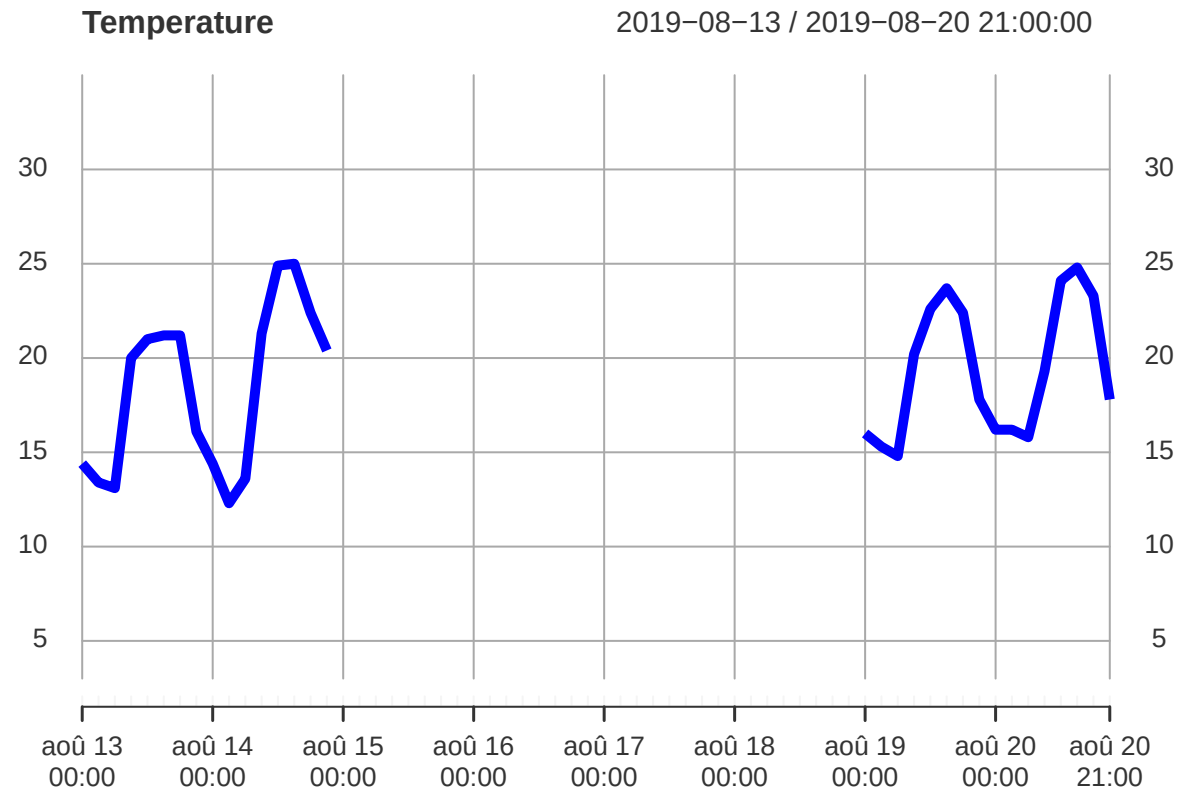
Nous avons utiliser les trois modèles AR,MA et ARIMA pour trouver les meilures prédiction pour les températures manquants qui sont :

(18.58599, 17.23393, 17.72986, 22.27037, 25.22742, 26.19669, 24.64631, 20.18139, 18.39082 ,16.92696, 17.56980, 22.00711, 25.05800, 25.95036, 24.46870, 19.97408, 15.33073, 14.05435 ,15.15566, 19.78602, 24.56405, 26.09198, 23.44119, 18.77144, 15.57616, 13.81157, 15.34598, 19.71572, 24.55016, 26.17968, 23.41399, 19.13377)

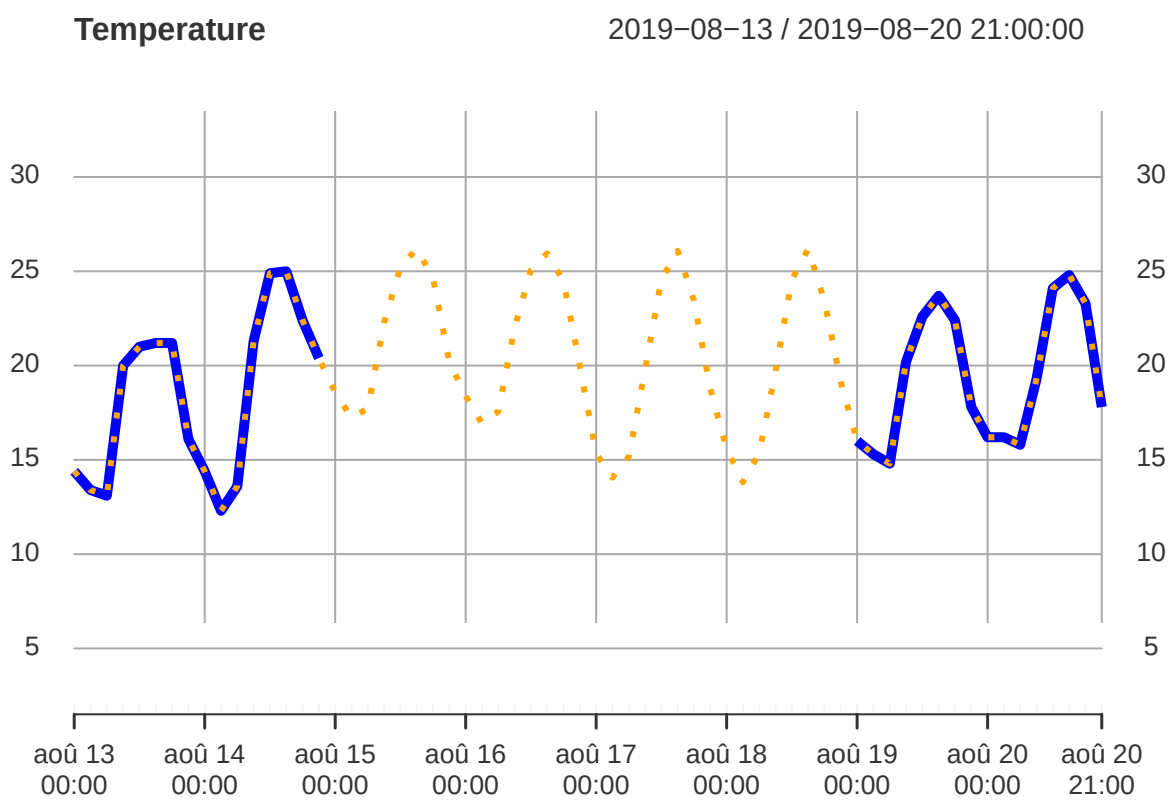
"Tous les modéles sont faux mais certains sont utiles"

```
Temperature <- xts::xts(tmp$temperature[(first_part[2]-15):(last_part[1]+15)],
                        order.by=
                          as.POSIXct(tmp$dateheure[(first_part[2]-15):(last_part[1]+15)]))

meannn <- xts::xts(mean2,
                        order.by=
```

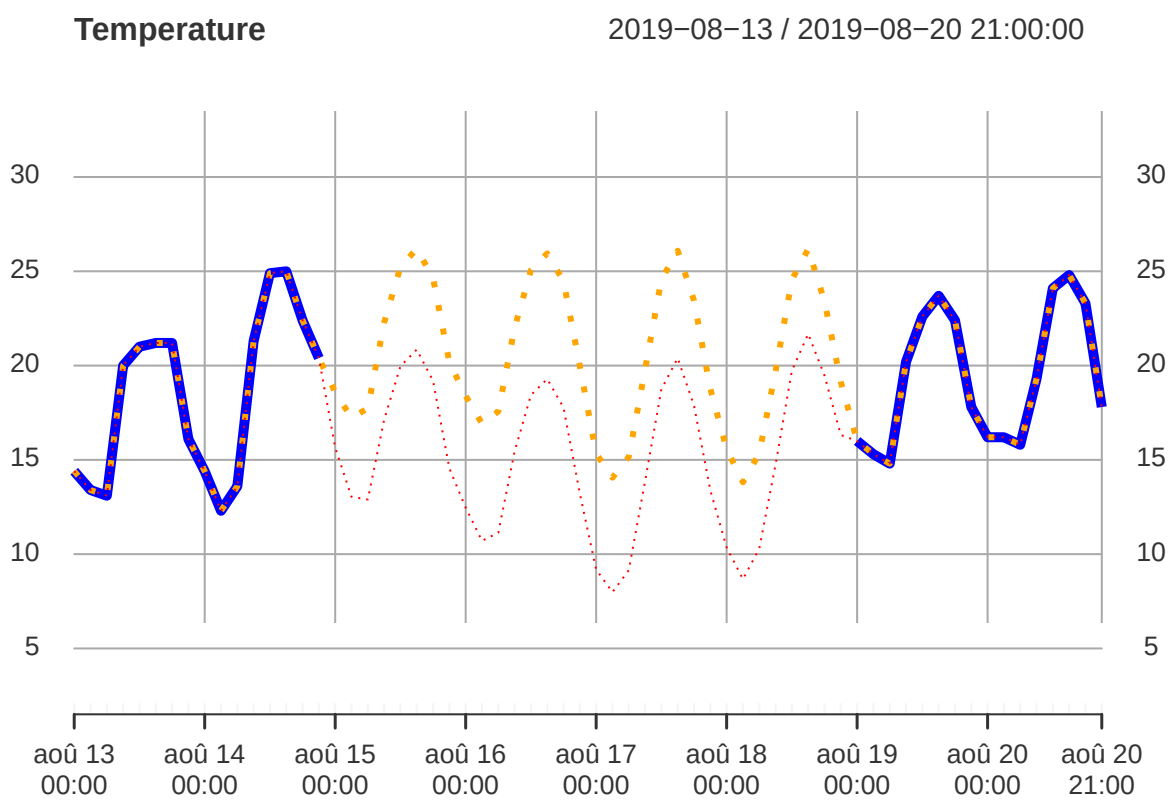
```
                      as.POSIXct(tmp$dateheure[(first_part[2]-15):(last_part[1]+15)]))
infff <- xts::xts(inf2,
                  order.by=
                  as.POSIXct(tmp$dateheure[(first_part[2]-15):(last_part[1]+15)]))
suppp <- xts::xts(sup2,
                  order.by
                 =as.POSIXct(tmp$dateheure[(first_part[2]-15):(last_part[1]+15)]))

plot(Temperature,lwd=5,ylim = c(3,35),col='blue')
```



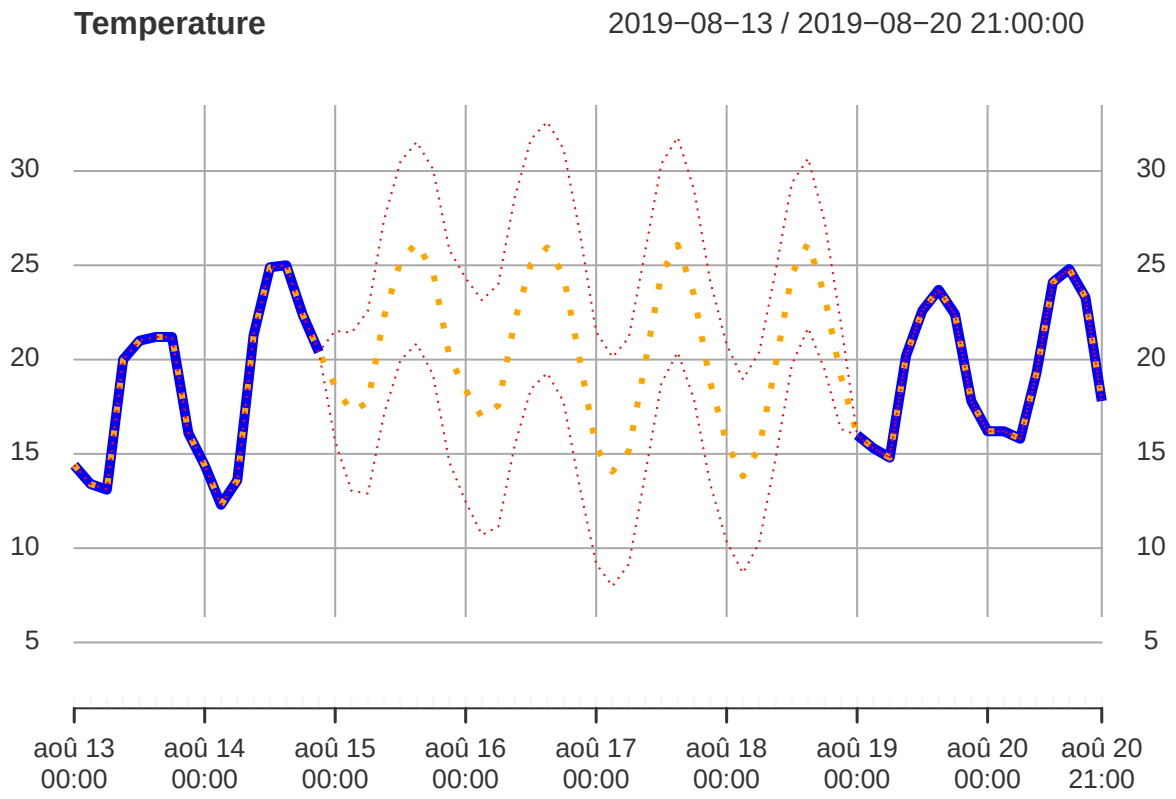

```
lines(meannn ,type = 'l',col='orange',lty=3,lwd=3)
```




```
lines(infff,type="l",col='red',lty=3,lwd=1)
```




```
lines(suppp,type="l",col='red',lty=3,lwd=1)
```


Temperature
2019−08−13 / 2019−08−20 21:00:00
30
25
20
15
10
5
aoû 13 00:00
aoû 14 00:00
aoû 15 00:00
aoû 16 00:00
aoû 17 00:00
aoû 18 00:00
aoû 19 00:00
aoû 20 00:00
aoû 20 21:00