Untitled

2025-02-05

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(readr)
library(hms)
```

```
##
## Attaching package: 'hms'

## The following object is masked from 'package:lubridate':
##
##     hms
```

```r
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(forecast)
```

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

```r
library(tseries)
```

```r
# Importation des données

data <- read.csv("~/github/projet_electricite_temperature/data/powerconsumption.csv", header=T)
```

```r
# Rassemblement des données des 3 centrales en une variable "conso"

data$conso =
  data$PowerConsumption_Zone1+
  data$PowerConsumption_Zone2+
  data$PowerConsumption_Zone3
```

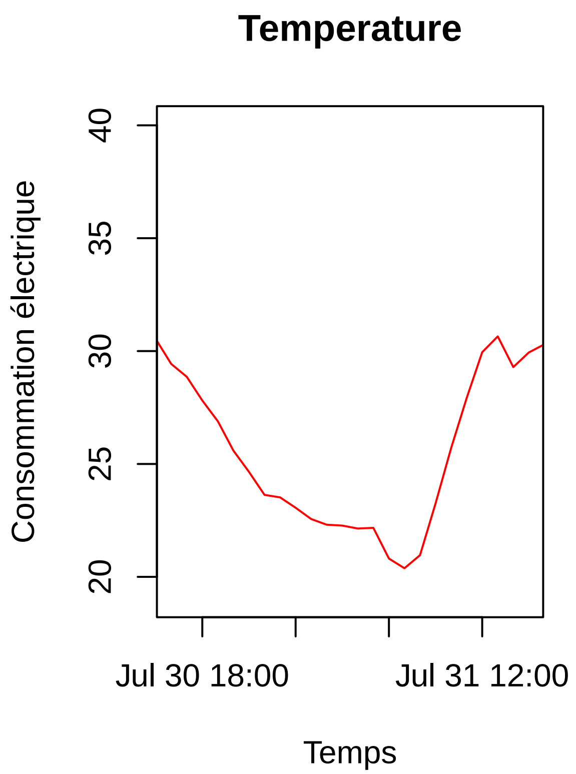
```r
# Suppression des variables inutiles

data = data %>%
  select(-PowerConsumption_Zone1, -PowerConsumption_Zone2, -PowerConsumption_Zone3, -Humidity, -WindSpe
```

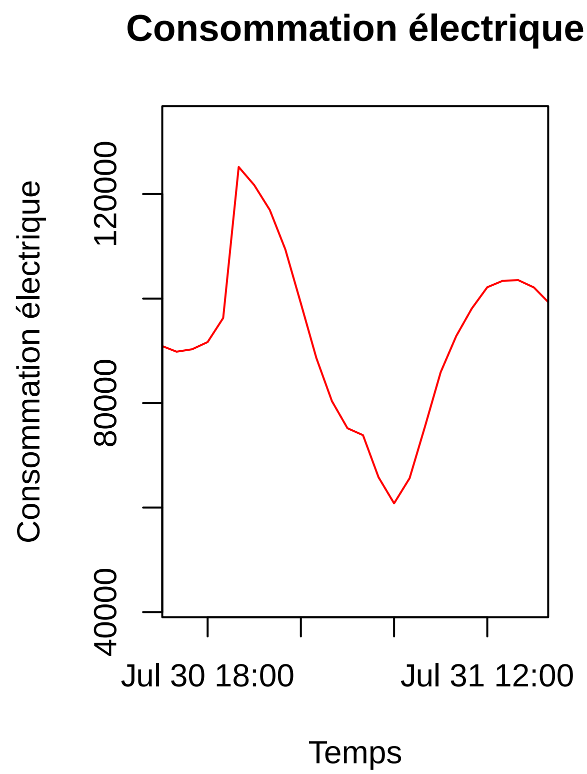
```r
# Modification du format de date pour faciliter la manipulation

data$Datetime <- as.POSIXct(data$Datetime, format="%m/%d/%Y %H:%M")
```

#Pour aout

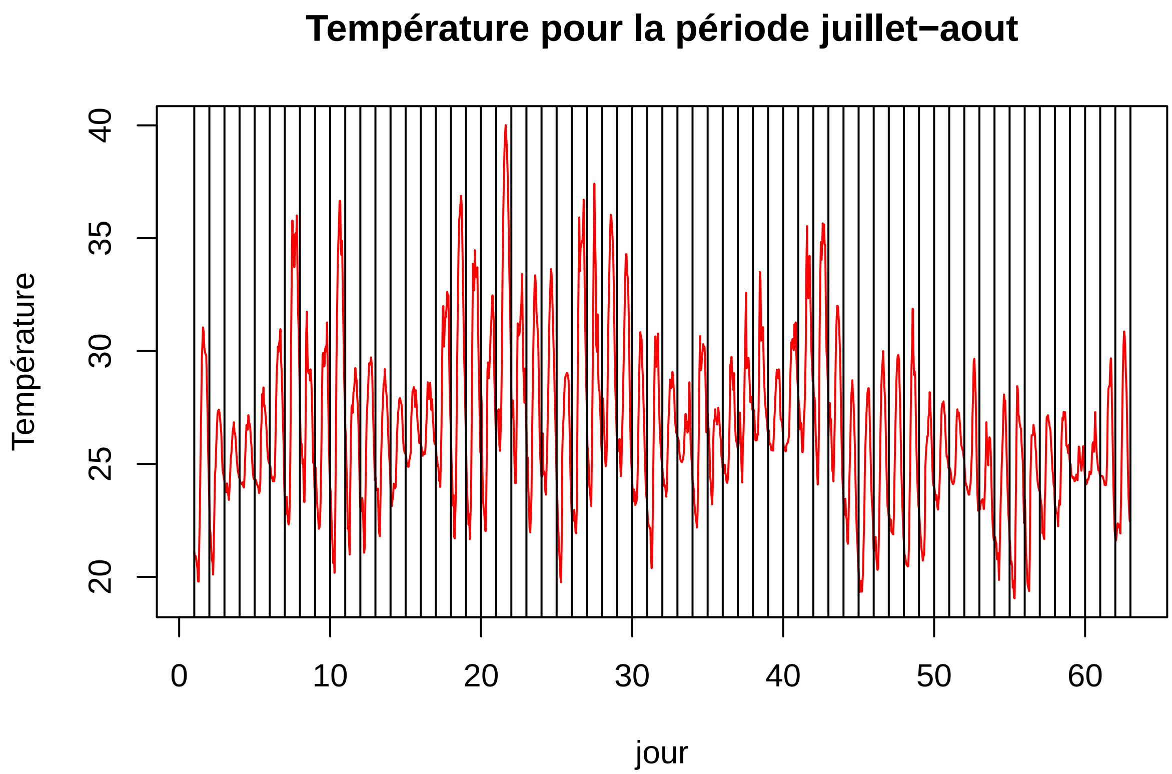
```r
# Creation d'une df pour le mois d'Aout (20 premiers jours)

data_ete <- data %>%
  filter(month(Datetime) == 7 | month(Datetime) == 8) %>%
  filter(minute((Datetime)) == 0) %>%
  filter(day(Datetime) <= 31)
```

```r
# Convertir les limites en objets POSIXct
xlim_start <- as.POSIXct("2017-07-30 16:00:00")
xlim_end <- as.POSIXct("2017-07-31 15:00:00")

par(mfrow = c(1, 2))
# Tracer le graphique avec xlim correctement défini
plot(data_ete$Datetime, data_ete$Temperature, col = "red", type = "l",
     xlim = c(xlim_start, xlim_end),
     xlab = "Temps", ylab = "Consommation électrique",
     main = "Temperature")

plot(data_ete$Datetime, data_ete$conso, col = "red", type = "l",
     xlim = c(xlim_start, xlim_end),
     xlab = "Temps", ylab = "Consommation électrique",
     main = "Consommation électrique")
```





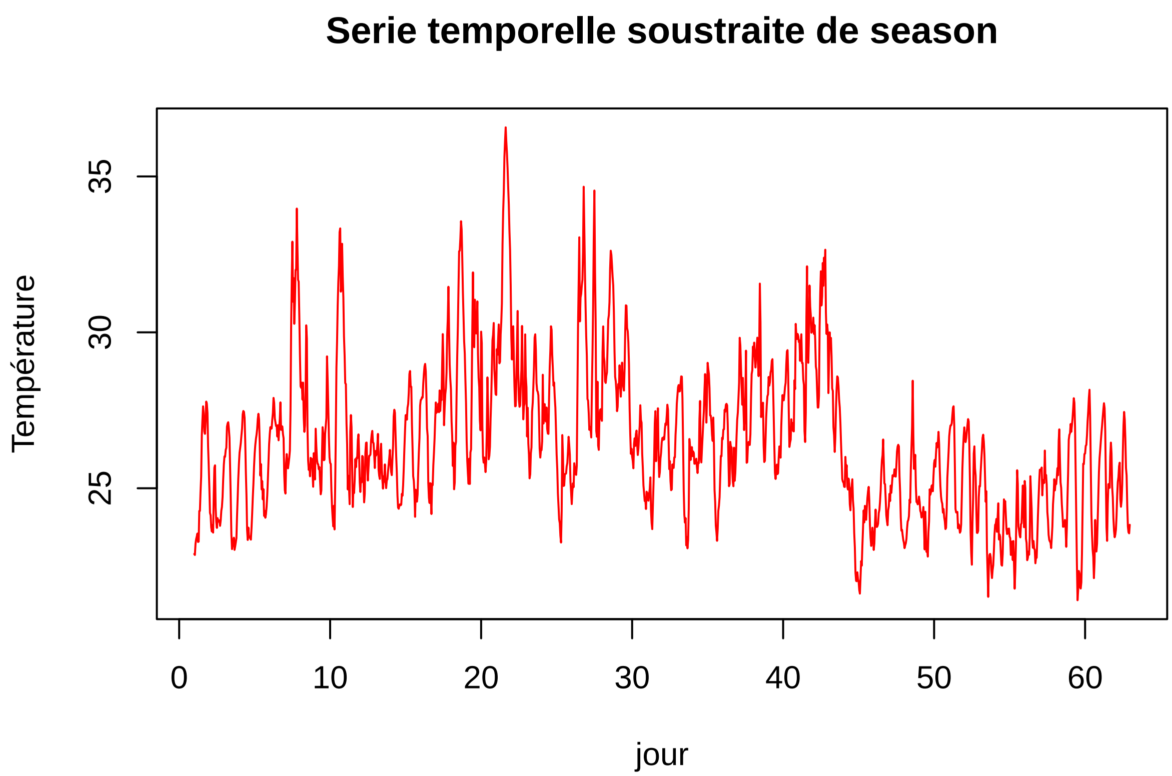

```r
# Creation de la série temporelle pour la température

ts_temp = ts(data_ete$Temperature, frequency = 24, start = c(1,1))

plot.ts(ts_temp,
        main="Température pour la période juillet-aout",
        xlab= "jour",
        ylab="Température",
        col="red")
for (i in 1:63) {
  abline(v=i)
}
```




```r
# Desaissonaliser la série
decompose_temp = decompose(ts_temp)

ts_temp_deseason = ts_temp - decompose_temp$seasonal
mean(ts_temp_deseason) # 26.47263
```

```
## [1] 26.47263
```

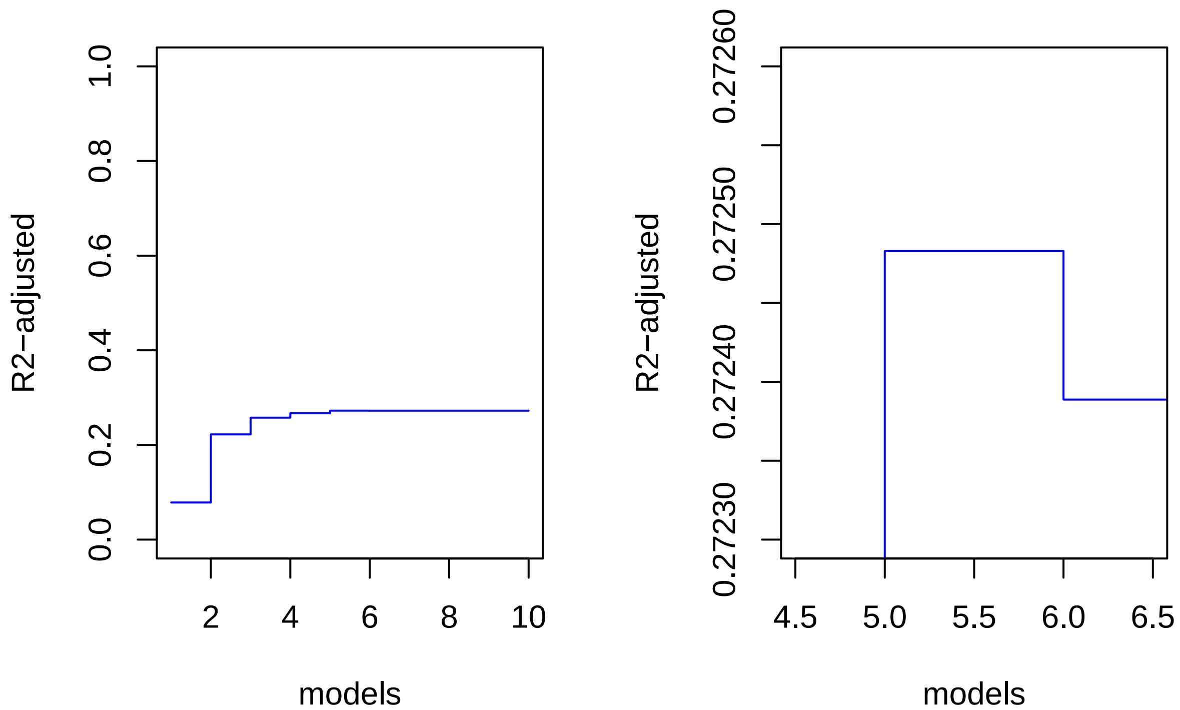
```r
plot.ts(ts_temp_deseason,
        main="Serie temporelle soustraite de season",
        xlab= "jour",
        ylab="Température",
        col="red")
```



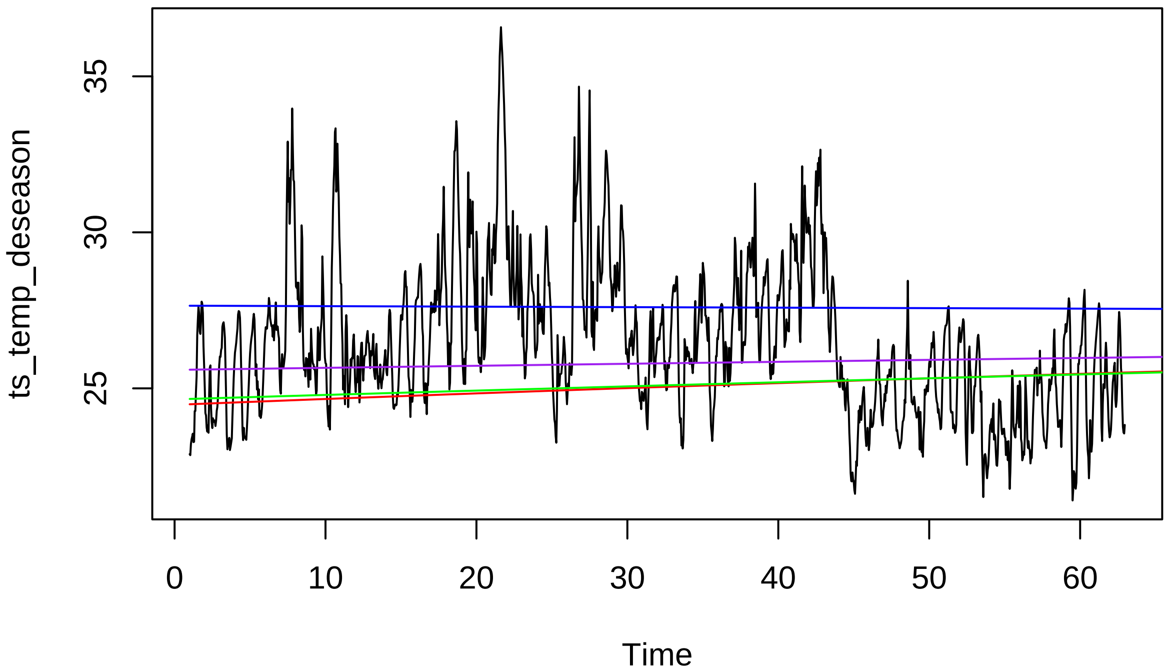

```r
t<-1:1488
t1<-ts(t,start=c(1,1),freq=24)
t2<-ts(t^2,start=c(1,1),freq=24)
t3<-ts(t^3,start=c(1,1),freq=24)
t4<-ts(t^4,start=c(1,1),freq=24)
t5<-ts(t^5,start=c(1,1),freq=24)
t6<-ts(t^6,start=c(1,1),freq=24)
t7<-ts(t^6,start=c(1,1),freq=24)
t8<-ts(t^6,start=c(1,1),freq=24)
t9<-ts(t^6,start=c(1,1),freq=24)
t10<-ts(t^6,start=c(1,1),freq=24)

lm1 = lm(ts_temp_deseason~t1)
lm2 = lm(ts_temp_deseason~t1+t2)
lm3 = lm(ts_temp_deseason~t1+t2+t3)
lm4 = lm(ts_temp_deseason~t1+t2+t3+t4)
lm5 = lm(ts_temp_deseason~t1+t2+t3+t4+t5)
lm6 = lm(ts_temp_deseason~t1+t2+t3+t4+t5+t6)
lm7 = lm(ts_temp_deseason~t1+t2+t3+t4+t5+t6+t7)
lm8 = lm(ts_temp_deseason~t1+t2+t3+t4+t5+t6+t7+t8)
lm9 = lm(ts_temp_deseason~t1+t2+t3+t4+t5+t6+t7+t8+t9)
lm10 = lm(ts_temp_deseason~t1+t2+t3+t4+t5+t6+t7+t8+t9+t10)
```

```r
temp_rsq = as.vector(c(summary(lm1)$adj.r.squared,
        summary(lm2)$adj.r.squared,
        summary(lm3)$adj.r.squared,
        summary(lm4)$adj.r.squared,
        summary(lm5)$adj.r.squared,
        summary(lm6)$adj.r.squared,
        summary(lm7)$adj.r.squared,
```

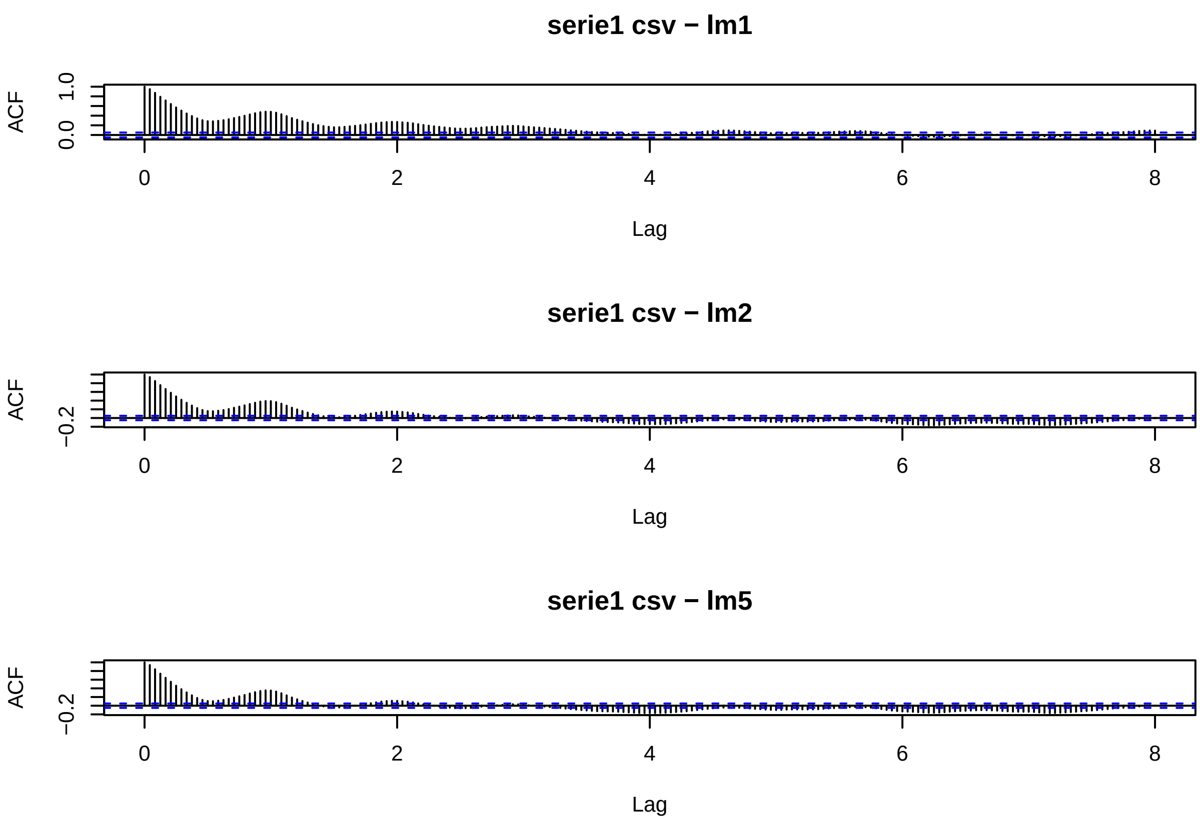
```
        summary(lm8)$adj.r.squared,
        summary(lm9)$adj.r.squared,
        summary(lm10)$adj.r.squared))

par(mfrow = c(1,2))
plot(temp_rsq, type = "s", col = "blue", ylim = c(0, 1), xlim = c(1,10), xlab = "models", ylab = "R2-ad
plot(temp_rsq, type = "s", col = "blue", ylim = c(0.2723, 0.2726), xlim = c(4.5,6.5), xlab = "models", y
```




```
par(mfrow=c(1,1))
plot.ts(ts_temp_deseason)
lines(predict(lm1), col = "blue", )
lines(predict(lm2), col = "purple")
lines(predict(lm5), col = "red")
lines(predict(lm6), col = "green")
```

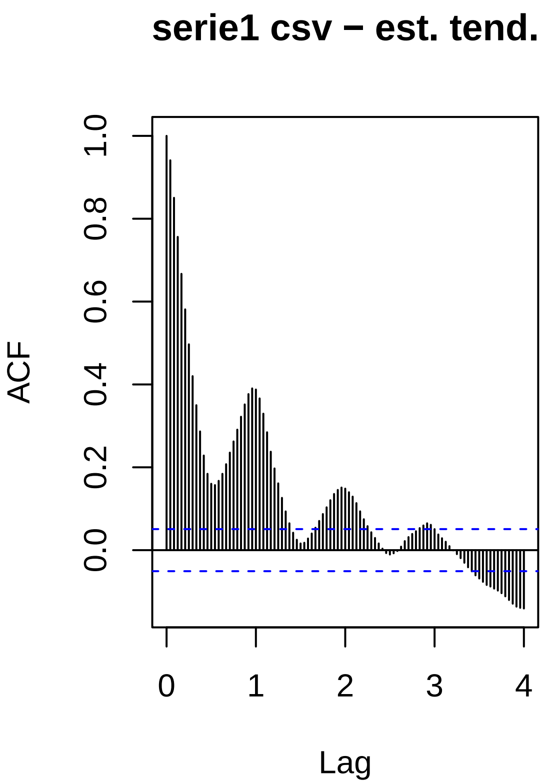


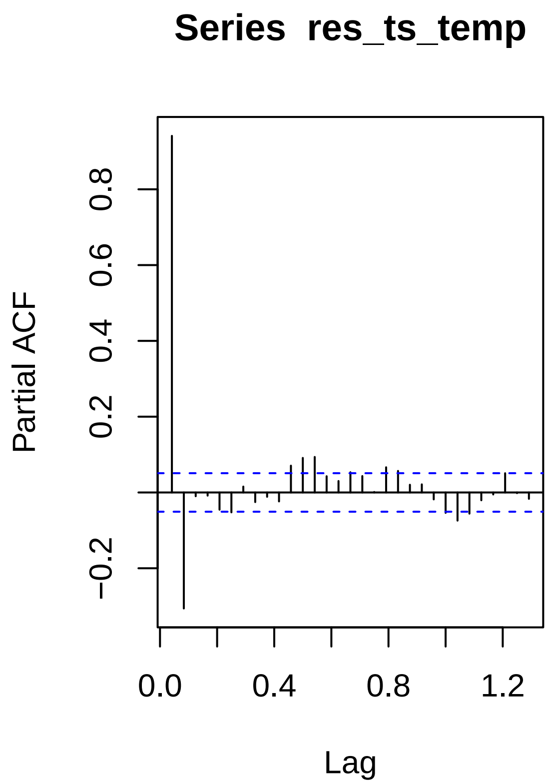

```r
par(mfrow=c(3,1))
acf(ts_temp_deseason - lm1$fitted.values, main="serie1 csv - lm1", lag.max = 192)
acf(ts_temp_deseason - lm2$fitted.values, main="serie1 csv - lm2", lag.max = 192)
acf(ts_temp_deseason - lm5$fitted.values, main="serie1 csv - lm5", lag.max = 192)
```




```r
# Calcul de la série résiduelle :
```

```r
res_ts_temp = ts_temp_deseason - lm2$fitted.values
pvalBox = Box.test(res_ts_temp, type="Ljung")
par(mfrow=c(1,2))
acf(res_ts_temp, main="serie1 csv - est. tend.", lag.max = 96)
pacf(res_ts_temp)
```





A partir de l'ACF, on voit bien que la décroissance est lente, ce qui nous pousse à modéliser par MA(q) sans savoir la valeur exacte du paramètre. De plus, la forme sinusoidale de l'ACF montrerait une saissonalité avec une périodicité de 24h (le même pattern se reitère à chaque lag). Ainsi, nous choisirons aussi des termes MA saissoniers (Q). A partir du PACF, nous voyons que le premier lag est significatif. Ainsi, nous allons modéliser par AR(p=1).

```r
adf.test(res_ts_temp)
```

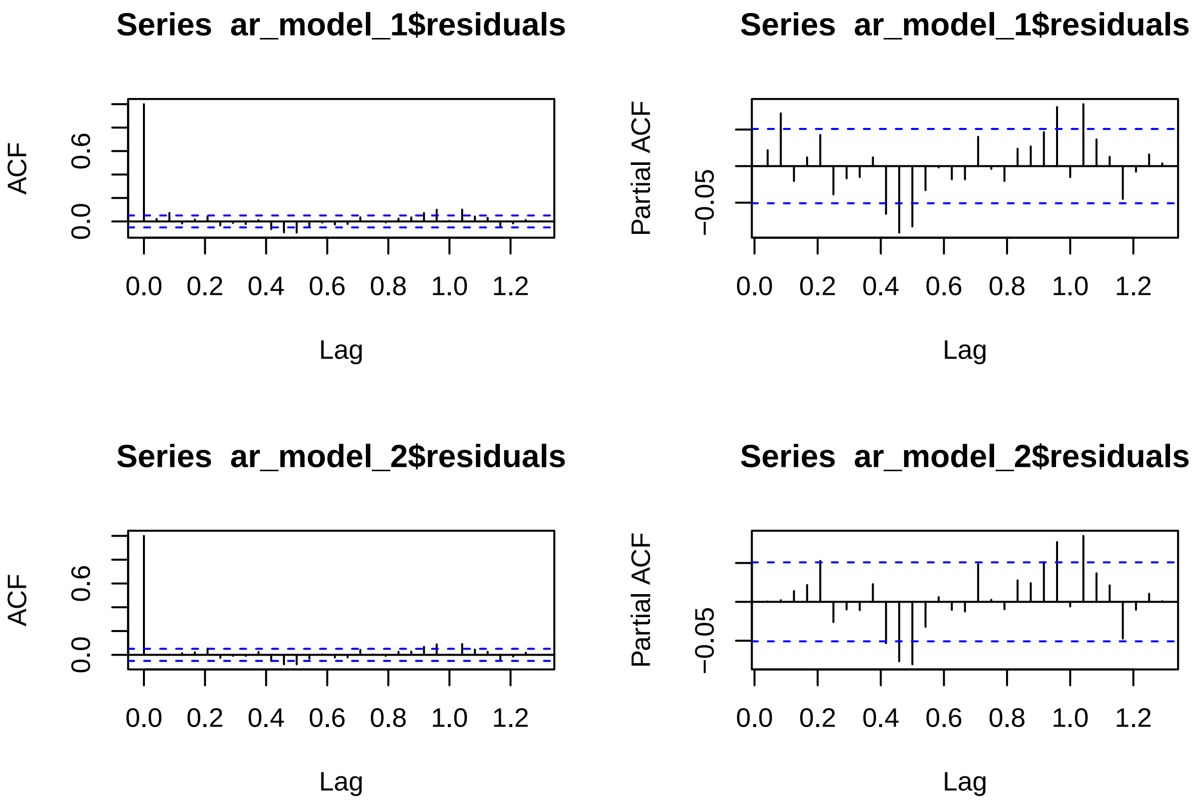
```
## Warning in adf.test(res_ts_temp): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  res_ts_temp
## Dickey-Fuller = -6.9611, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(res_ts_temp)
```

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

##
##  KPSS Test for Level Stationarity
##
## data:  res_ts_temp
## KPSS Level = 0.19815, Truncation lag parameter = 7, p-value = 0.1
```

La série est stationnaire donc on ne fera pas de différenciation dans notre modèle.

```r
ar_model_1 = arima(res_ts_temp, order = c(1, 0, 1), seasonal = list(order = c(0, 0, 1), period = 24))
ar_model_2 = arima(res_ts_temp, order = c(1, 0, 2), seasonal = list(order = c(0, 0, 2), period = 24))

par(mfrow=c(2,2))
acf(ar_model_1$residuals)
pacf(ar_model_1$residuals)
acf(ar_model_2$residuals)
pacf(ar_model_2$residuals)
```




```r
Box.test(ar_model_1$residuals, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  ar_model_1$residuals
## X-squared = 0.70799, df = 1, p-value = 0.4001
```

```r
Box.test(ar_model_2$residuals, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  ar_model_2$residuals
## X-squared = 0.00048985, df = 1, p-value = 0.9823
```