

# Projet Covid

Margaux Bailleul / Oriane Duclos / Marie Guibert

2023-05-22

```
library(tidyverse)
library(forecast)
library(tidyquant)
library(caschrono)
library(stats)
library(tseries)
library(lmtest)
```

## Importations des données

Tout d'abord, on importe les données et on sélectionne les données concernant la France.

```
donnees_fr <- read.csv("covid_france.csv", sep=",")
summary(donnees_fr)
```

```
##      date          new_cases
## Length:1203      Min.       :    0
## Class :character 1st Qu.: 2968
## Mode  :character Median  :12174
##                  Mean   :32315
##                  3rd Qu.:32913
##                  Max.   :500563
##                  NA's   :1
```

Les données ci-dessus comprennent une variable temporelle et une variable caractérisée par un enregistrement journalier des nouveaux cas de Covid-19 en France.

Avec ce résumé, nous pouvons voir une étendue très importante du nombre de nouveaux cas de covid-19 sur notre période. En effet, le maximum est de 500 563 nouveaux cas par jour alors que certains jours n'ont enregistré aucun nouveaux cas. Ce constat nous montre bien une évolution importante de l'épidémie.

De plus, la médiane est de 12 174 alors que la moyenne est de 32 315 nouveaux cas par jour. Ceci nous montre bien l'effet épidémique puisque certaines valeurs enregistrées sont très importantes et fluctuent beaucoup.

Nous allons maintenant identifier la période d'étude :

```
min(donnees_fr$date)
```

```
## [1] "2020-01-03"
```

```
max(donnees_fr$date)
```

```
## [1] "2023-04-19"
```

Grâce à cette étape, nous pouvons observer que notre série temporelle débute le 1er Mars 2020 et se termine le 19 Avril 2023. Notre étude a donc une plage d'environ de 3 ans.

## Transformation des données en série temporelle

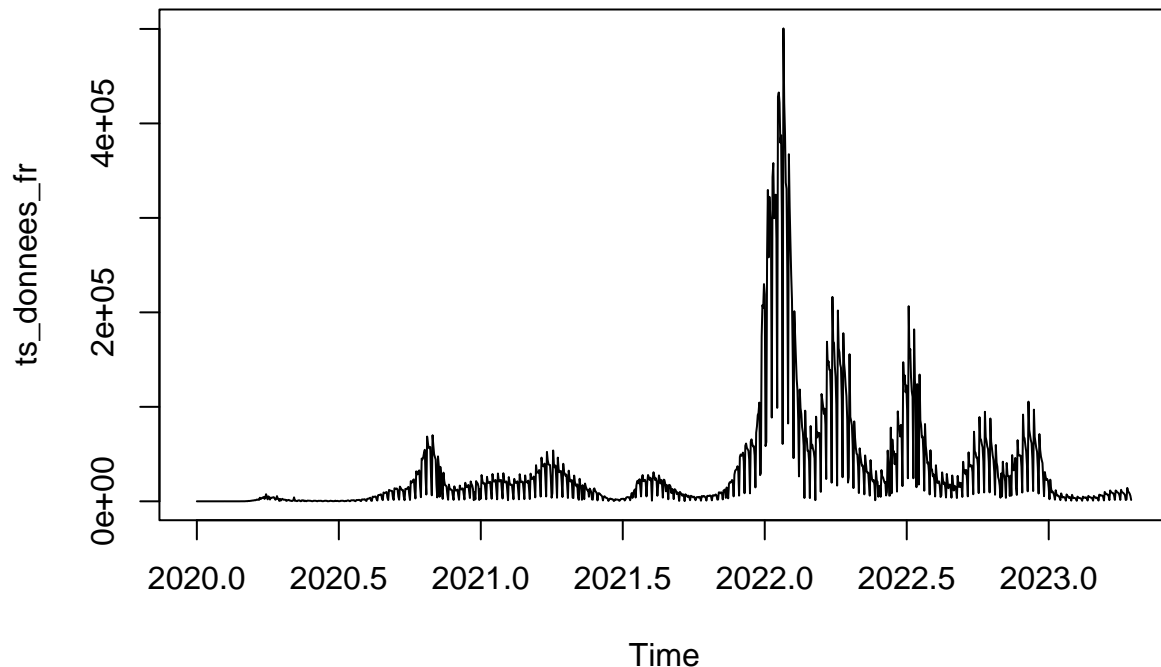
Premièrement, nous allons transformer nos données en séries temporelles pour pouvoir réaliser notre analyse.

```
ts_donnees_fr <- ts(donnees_fr$new_cases, start = c(2020,1,3), frequency = 365)
class(ts_donnees_fr)
```

```
## [1] "ts"
```

## Prise en main du jeu de données

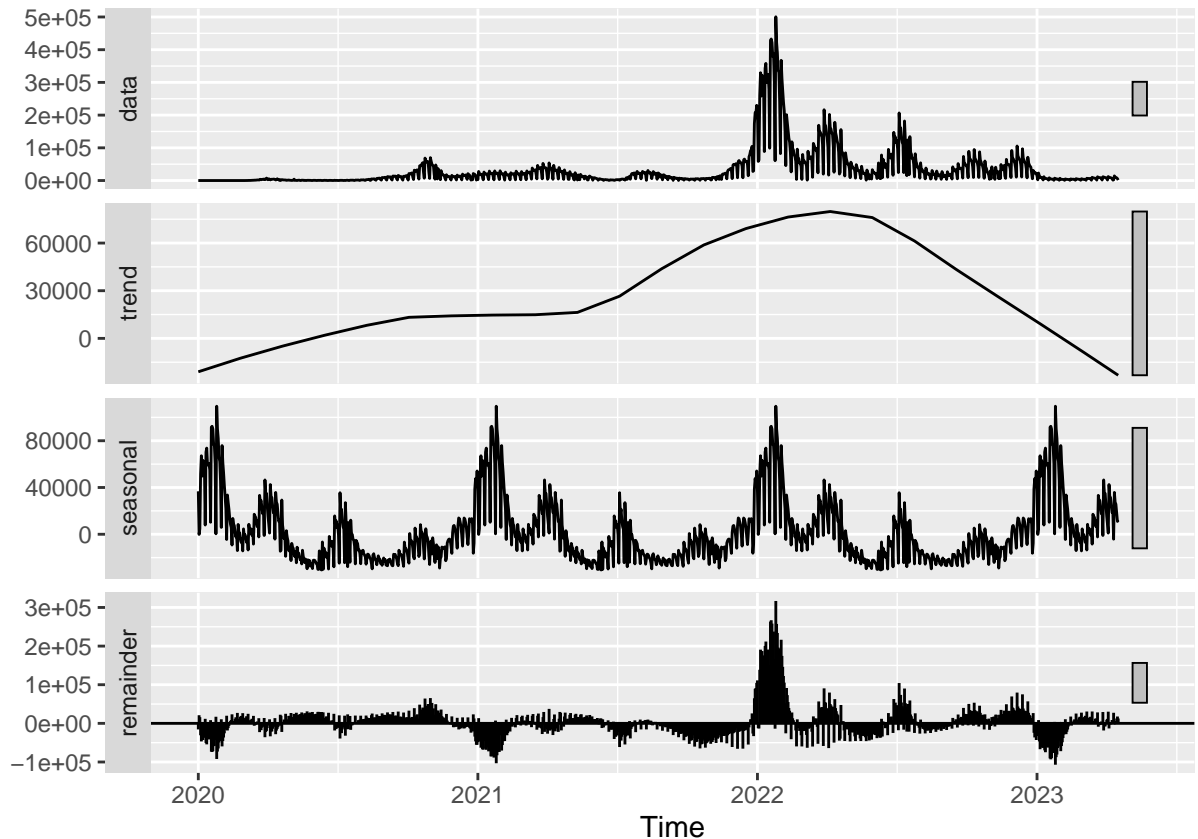
```
plot(ts_donnees_fr)
```



Ce premier graphique nous montre une hausse brutale des nouveaux cas de covids en 2022. Afin de pouvoir continuer notre analyse de façon cohérente, nous allons diviser notre série en 3 parties : avant, pendant et après ce choc en 2022.

Nous allons appliquer la décomposition saisonnière à la série temporelle pour visualiser les tendances et les motifs saisonniers. Nous supprimons les données manquantes afin de ne garder que celles qui sont pertinentes.

```
decomp_ts <- stl(na.omit(ts_donnees_fr), s.window = "periodic")
autoplot(decomp_ts)
```



Cette étape nous permet d'observer une tendance à la hausse entre 2020 et 2022, puis à partir de 2022, une tendance à la baisse. En effet, l'année marquante a été le début de l'année 2022 car il présente un nombre très important de nouveaux cas de covid en France.

De plus, nous pouvons voir une saisonnalité annuelle composée de 2 voire 3 pics, correspondant aux saisons les plus propices à la transmission ou aux mouvements de foule (vacances).

Enfin, la partie concernant les résidus nous présente des valeurs importantes, impliquant des complications pour émettre des prévisions. Nous pouvons avoir plus d'informations sur ce site web : <https://drees.solidarites-sante.gouv.fr/delais-covid19-2023-02-02>

## Division de notre série

Nous décidons de créer trois sous-séries de notre série initiale afin de pouvoir réaliser le traitement des données. Notre objectif est d'isoler le cas particulier de l'année 2022 pour avoir une étude correcte.

```
serie1 <- donnees_fr |>
  filter(date<="2021-12-22")
# serie1
ts_serier1 <- ts(serie1$new_cases,start = c(2020,1,3), frequency = 365)

serie2 <- donnees_fr |>
  filter(date>"2021-12-22",date<="2023-01-05")
# serie2
ts_serier2 <- ts(serie2$new_cases,start = c(2021,23,12),end = c(2023,1,5), frequency = 365)
```

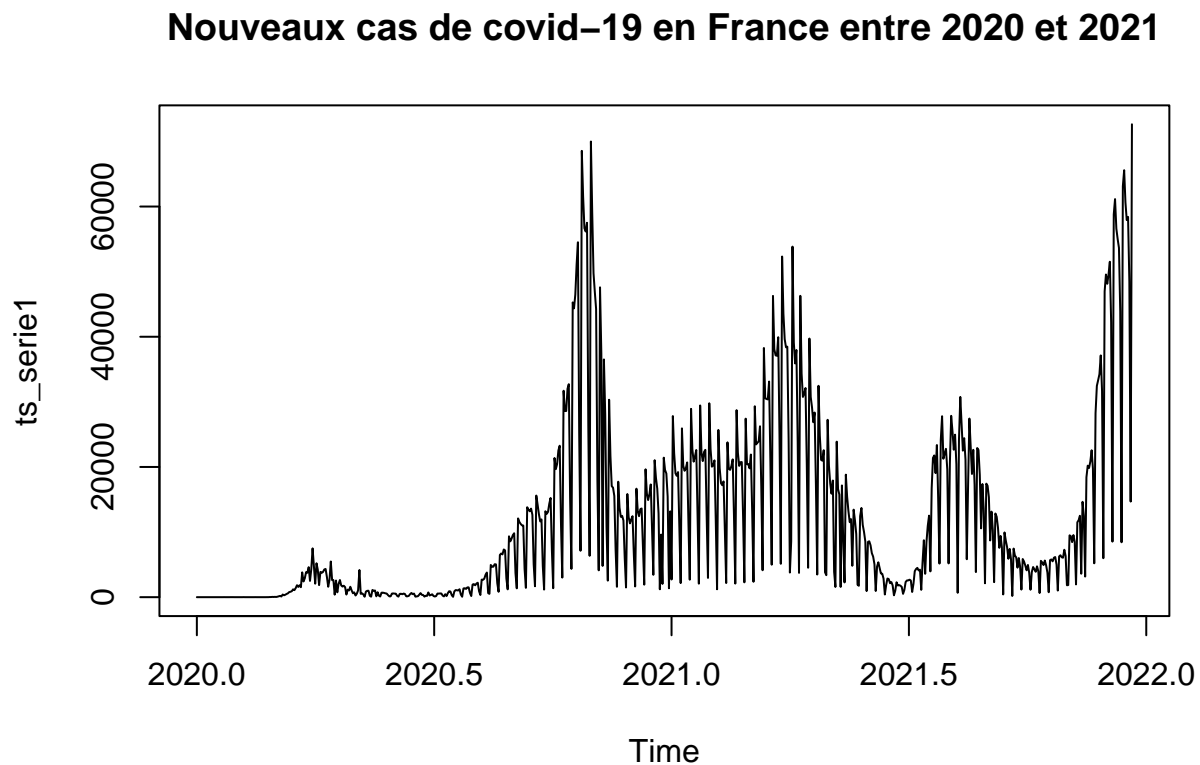
```
serie3 <- donnees_fr |>
  filter(date>"2023-01-05")
# serie3
ts_serier3 <- ts(serie3$new_cases,start = c(2023,2,5), frequency = 365)
```

Nous avons choisi de scinder notre série en trois périodes :

- avant le 22 Décembre 2021
- entre le 23 Décembre 2021 et le 5 Janvier 2023
- après le 6 Janvier 2023

Nous pouvons maintenant les visualiser :

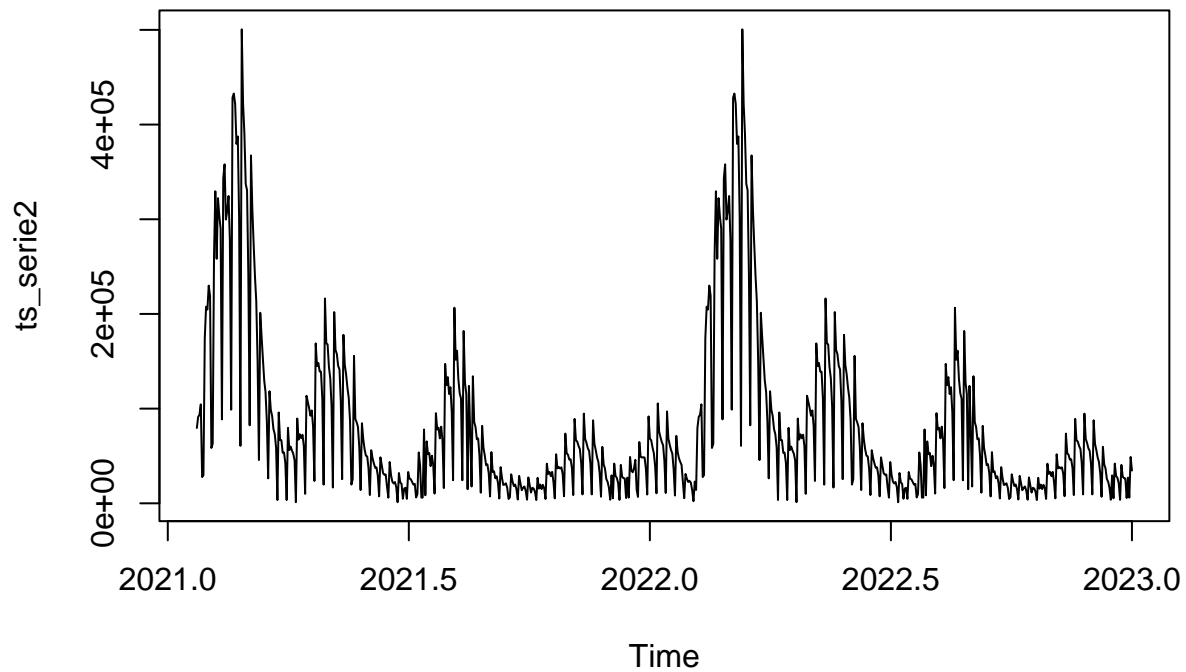
```
plot(ts_serier1,main="Nouveaux cas de covid-19 en France entre 2020 et 2021")
```



Dans cette sous-série, nous pouvons observer une saisonnalité avec des pics lors de la fin de l'année, pouvant correspondre à la période hivernale mais aussi au niveau des vacances scolaires (vers le mois de Mars). Ce constat est expliqué par les déplacements de populations et les concentrations de personnes (réunions de famille, lieux festifs).

```
plot(ts_serier2,main="Nouveaux cas de covid-19 en France entre 2021 et 2023")
```

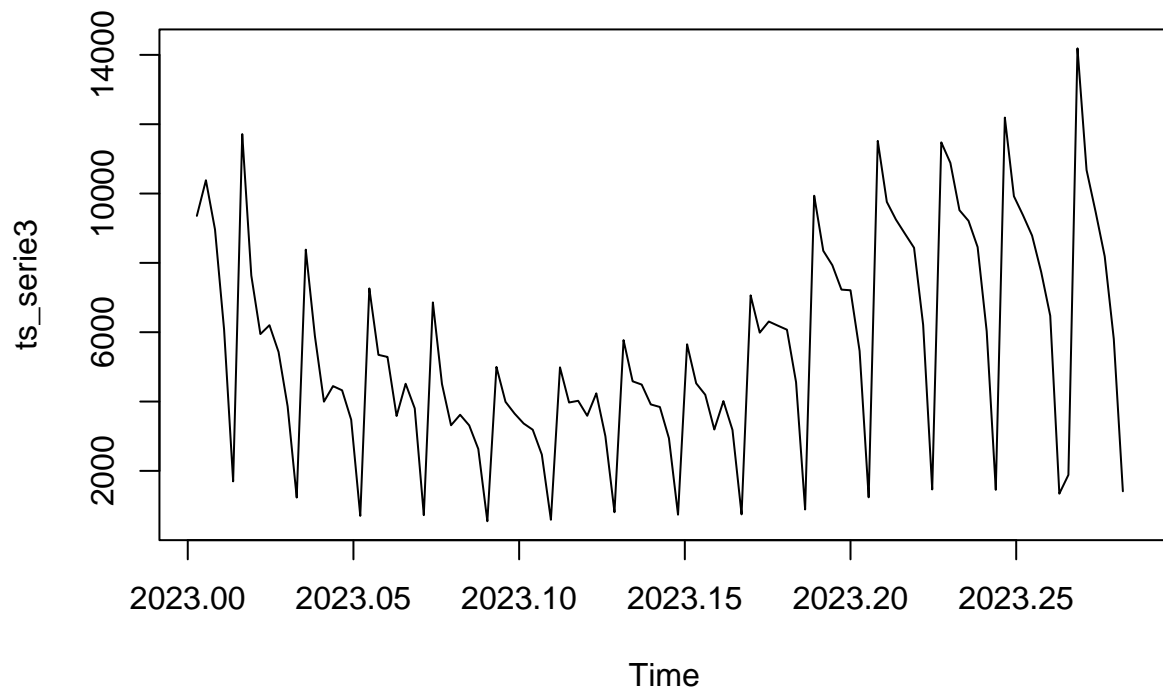
## Nouveaux cas de covid-19 en France entre 2021 et 2023



Cette seconde série chronologique nous montre une tendance à la baisse des nouveaux cas de covid 19. Aussi, nous remarquons une saisonnalité environ tous les deux mois. Au début de l'année 2021 le nombre de nouveaux cas est nettement plus important qu'en 2022. Ce constat peut être expliqué par la diminution des tests pour détecter le covid 19.

```
plot(ts_serie3,main="Nouveaux cas de covid-19 en France en 2023")
```

## Nouveaux cas de covid-19 en France en 2023



Enfin, cette sous-série est caractérisée par une tendance à la baisse dans un premier temps puis à la hausse. Ce constat est peut-être expliqué par la reprise de la vie active de la population française.

Grâce à cette division, nous allons pouvoir sélectionner la sous-série la plus pertinente.

### Analyse de la première sous-série

Nous avons décidé de nous focaliser sur la première sous-série.

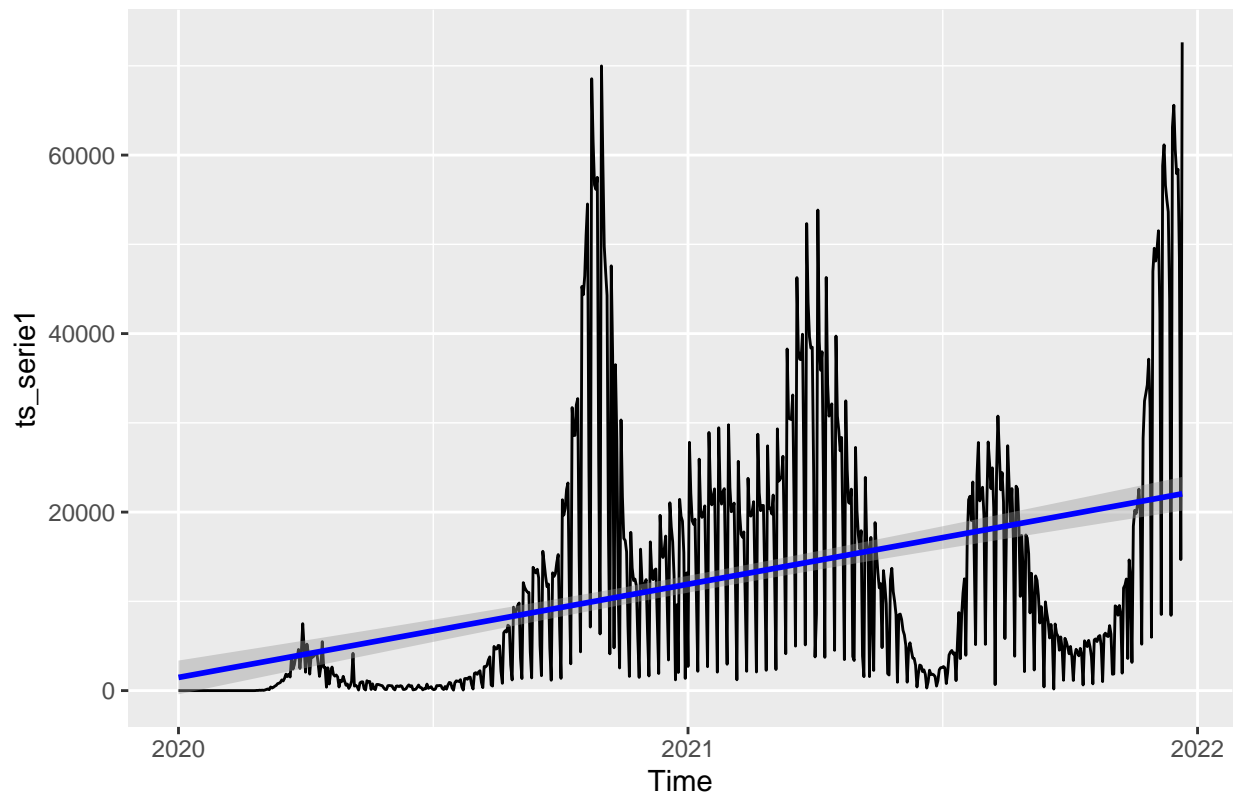
Ce choix est expliqué grâce à notre connaissance des événements durant cette année particulière. En effet, les confinements ont pu avoir des conséquences sur notre série et nos données. Notre étude commence donc le 1er Mars 2020 et s'étend jusqu'au 22 décembre 2021.

Pour rappel, notre série présente une tendance à la hausse comme le montre le graphique ci-dessous. Elle présente aussi une saisonnalité, mais elle n'est pas régulière. En effet, les différentes hausses de nouveaux cas de covid dépendent des confinements et des mesures sanitaires mises en place.

```
autoplot(ts_serie1)+  
  geom_smooth(method = lm,color="blue")+  
  ggtitle("Nouveaux cas de covids en France entre 2020 et 2022")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

## Nouveaux cas de covids en France entre 2020 et 2022



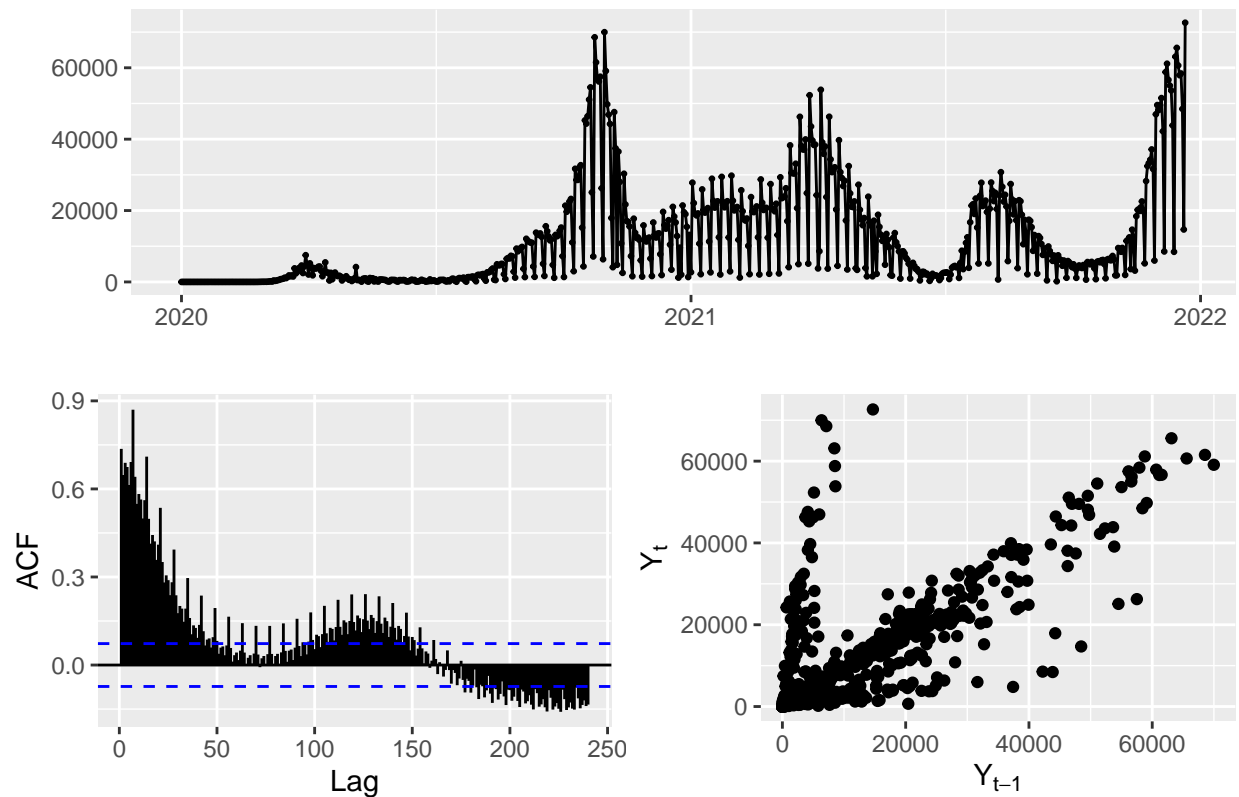
On va d'abord chercher à décrire notre série grâce à des indicateurs descriptifs simples.

```
mean(ts_serie1)
```

```
## [1] 11763.34
```

Entre le 1er Mars 2020 et le 22 Décembre 2022, la moyenne des nouveaux cas de covids par jour était de 11 763 cas en France.

```
ts_serie1 |>  
  ggtsdisplay(plot.type = "scatter",smooth=FALSE)
```



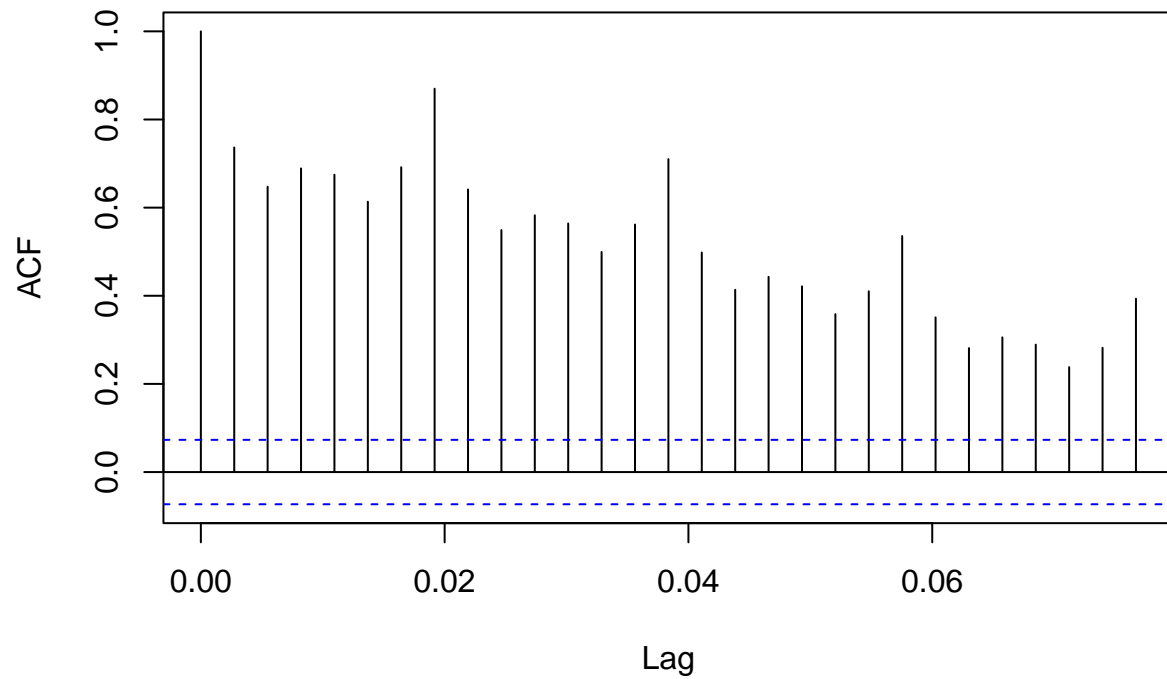
Nous pouvons faire quelques observations sur le graphique de l'ACF. Ce graphique nous permet de détecter une structure de corrélation du réseau. Dans notre cas, plusieurs autocorrélations présentent des valeurs significativement non nulles, ce qui signifie que la série chronologique n'est pas aléatoire. Aussi, nous pouvons observer un nuage de points plutôt aligné, on peut donc se poser la question d'une éventuelle corrélation.

Afin d'avoir une analyse plus exhaustive, nous pouvons analyser l'ACF et la PACF. En effet, l'étude de l'ACF va nous permettre de détecter la périodicité de la série.

```
acf <- acf(ts_serie1)
```



## Series ts\_serie1



```
print(data.frame(acf$lag,acf$acf))
```

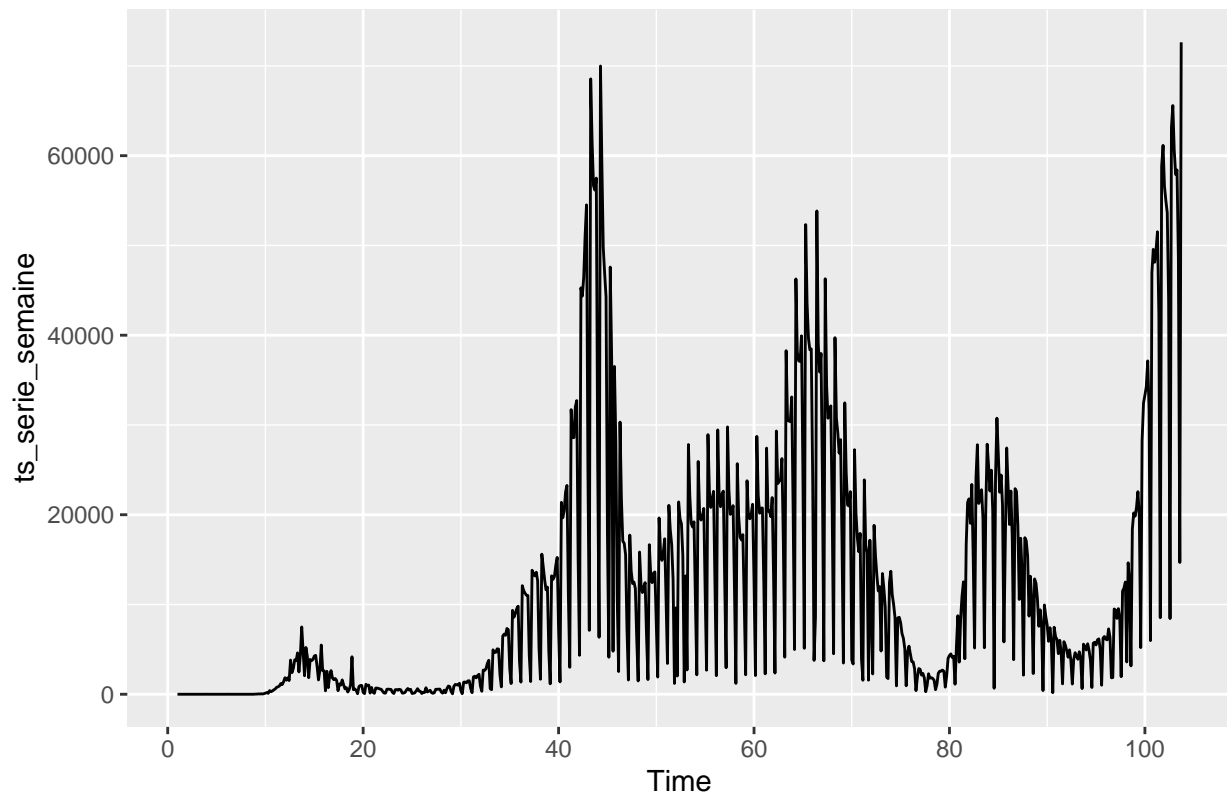
```
##      acf.lag  acf.acf
## 1 0.00000000 1.000000
## 2 0.002739726 0.7365088
## 3 0.005479452 0.6474386
## 4 0.008219178 0.6890097
## 5 0.010958904 0.6747133
## 6 0.013698630 0.6134778
## 7 0.016438356 0.6918098
## 8 0.019178082 0.8699193
## 9 0.021917808 0.6413418
## 10 0.024657534 0.5491586
## 11 0.027397260 0.5827231
## 12 0.030136986 0.5640830
## 13 0.032876712 0.4993613
## 14 0.035616438 0.5617146
## 15 0.038356164 0.7100309
## 16 0.041095890 0.4981678
## 17 0.043835616 0.4136221
## 18 0.046575342 0.4429968
## 19 0.049315068 0.4213644
## 20 0.052054795 0.3582470
## 21 0.054794521 0.4101775
## 22 0.057534247 0.5356155
## 23 0.060273973 0.3509638
```

```
## 24 0.063013699 0.2813732
## 25 0.065753425 0.3056083
## 26 0.068493151 0.2891480
## 27 0.071232877 0.2380992
## 28 0.073972603 0.2820158
## 29 0.076712329 0.3934595
```

Ce graphique nous permet d'observer une corrélation hebdomadaire. En effet, nous pouvons remarquer un pic plus élevé tous les 7 jours.

Puisque notre série montre une corrélation hebdomadaire, nous avons choisi d'étudier une série temporelle avec une fréquence de 7 jours.

```
ts_serie_semaine <- ts(ts_serie1, frequency = 7)
autoplot(ts_serie_semaine) # Visualisation des données
```



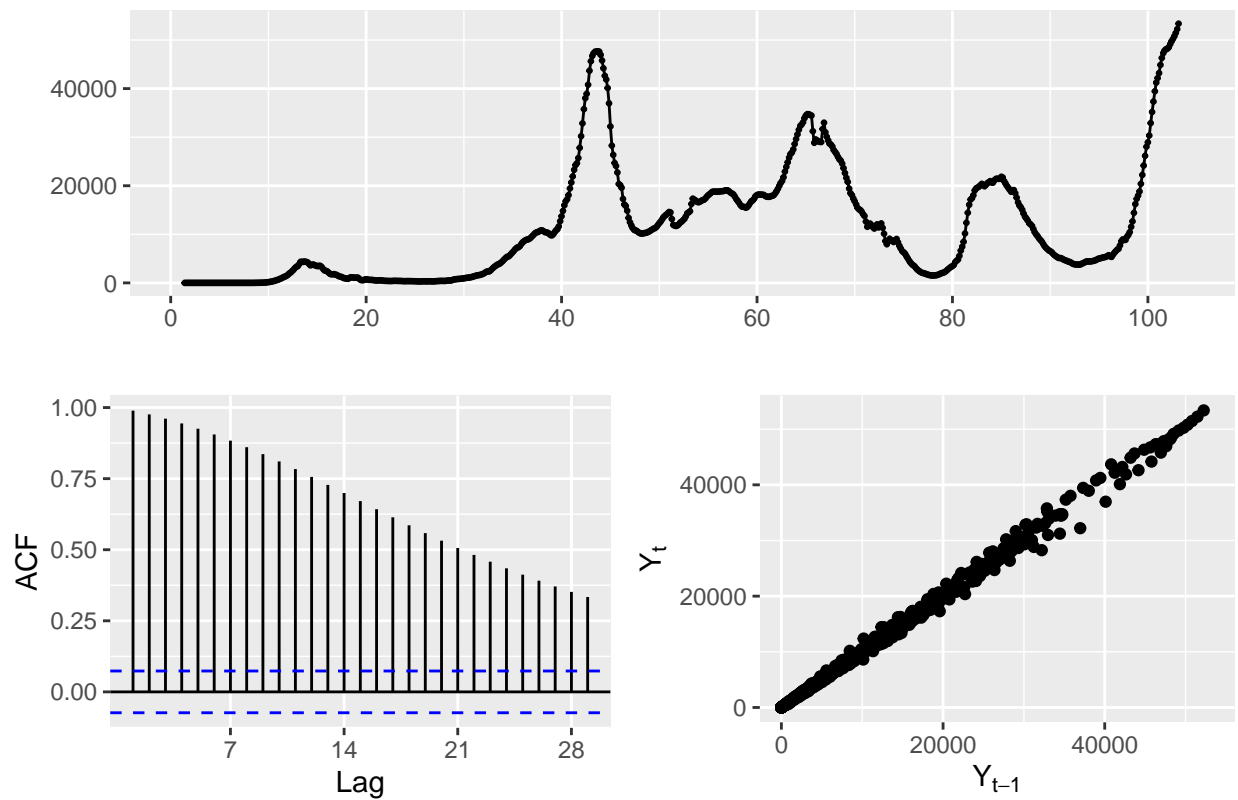
Ayant choisi d'analyser notre série en série hebdomadaire, l'axe des abscisses change puisqu'il représente le nombre de semaines. Ainsi, entre 2020 et 2022, nous avons environ 100 semaines.

## Décomposition saisonnière

Cette première étape consiste à isoler la saisonnalité de la tendance pour comprendre comment est constitué la série. Nous allons d'abord utiliser les moyennes mobiles :

```
lissage <- c(.5,rep(1,6),.5)/7
serie_lissage <- ts_serie_semaine |>
  stats::filter(filter=lissage,sides=2)
ggtsdisplay(serie_lissage,plot.type = "scatter")
```

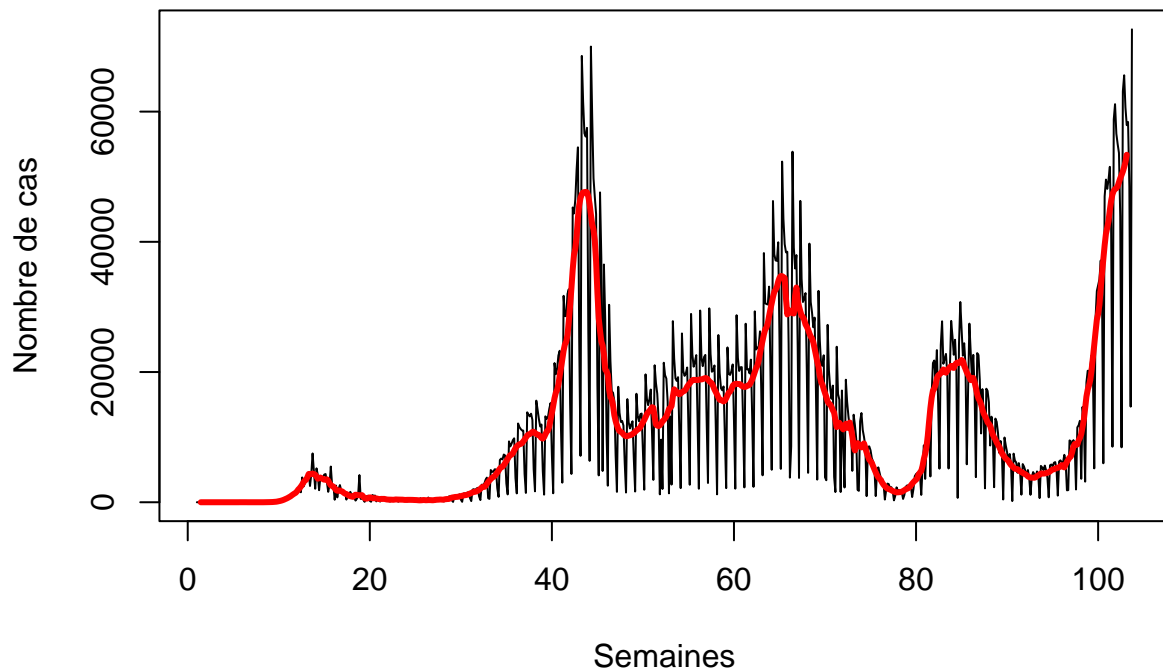
```
## Warning: Removed 7 rows containing missing values (`geom_point()`).
## Removed 7 rows containing missing values (`geom_point()`).
```



Cette étape nous permet d'écarter l'hypothèse de linéarité de notre tendance. Nous savons d'ores et déjà que le modèle linéaire ne sera pas adapté à notre série chronologique. Ces graphiques nous permettent tout de même d'identifier une croissance puisque la pandémie s'accroît avec le temps dans notre série.

```
plot(ts_serie_semaine,main="Série initiale et série lissée",xlab="Semaines",ylab="Nombre de cas")
lines(serie_lissage,col='red',lwd=3)
```

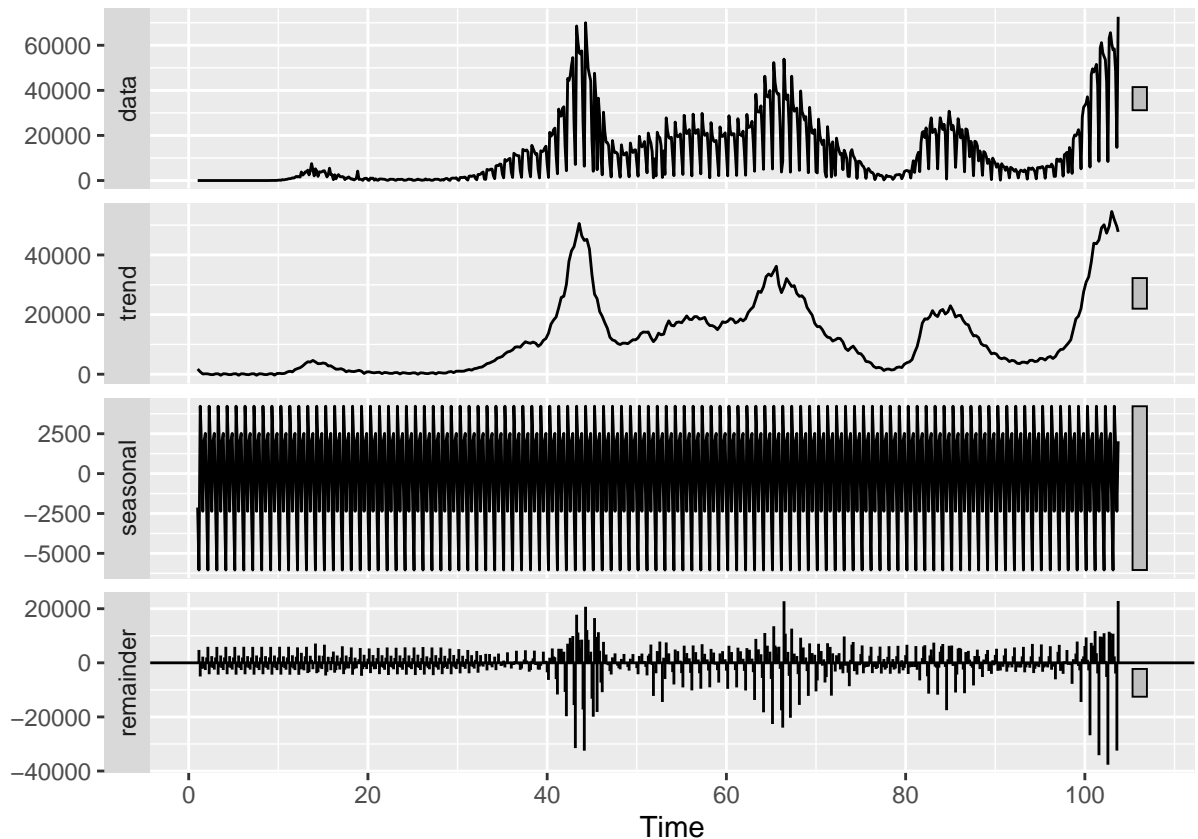
## Série initiale et série lissée



Grâce à ce lissage, nous avons isolé la saisonnalité de la série.

Nous allons à présent décomposer notre série avec la fonction `stl` :

```
decomp_ts <- stl(na.omit(ts_serie_semaine), s.window = "periodic")
autoplot(decomp_ts)
```



Cette décomposition nous présente les différentes composantes de notre série chronologique. Nous pouvons encore observer une importance des résidus, impliquant les mêmes conséquences que précisées ci-dessus.

## Stationnarité

Nous allons d'abord tester la stationnarité de notre série pour savoir si nous avons besoin d'effectuer des modifications sur celle-ci. Le premier test que nous avons choisi est celui de Dickey-Fuller :

```
adf.test(ts_serie_semaine)
```

```
##
## Augmented Dickey-Fuller Test
##
## data: ts_serie_semaine
## Dickey-Fuller = -1.3122, Lag order = 8, p-value = 0.8695
## alternative hypothesis: stationary
```

Dans notre cas, la p-value est supérieure à 5% donc nous acceptons l'hypothèse de non-stationnarité. Notre série est non stationnaire.

```
kpss.test(ts_serie_semaine)
```

```
## Warning in kpss.test(ts_serie_semaine): p-value smaller than printed p-value
##
## KPSS Test for Level Stationarity
##
## data: ts_serie_semaine
```

```
## KPSS Level = 2.7148, Truncation lag parameter = 6, p-value = 0.01
```

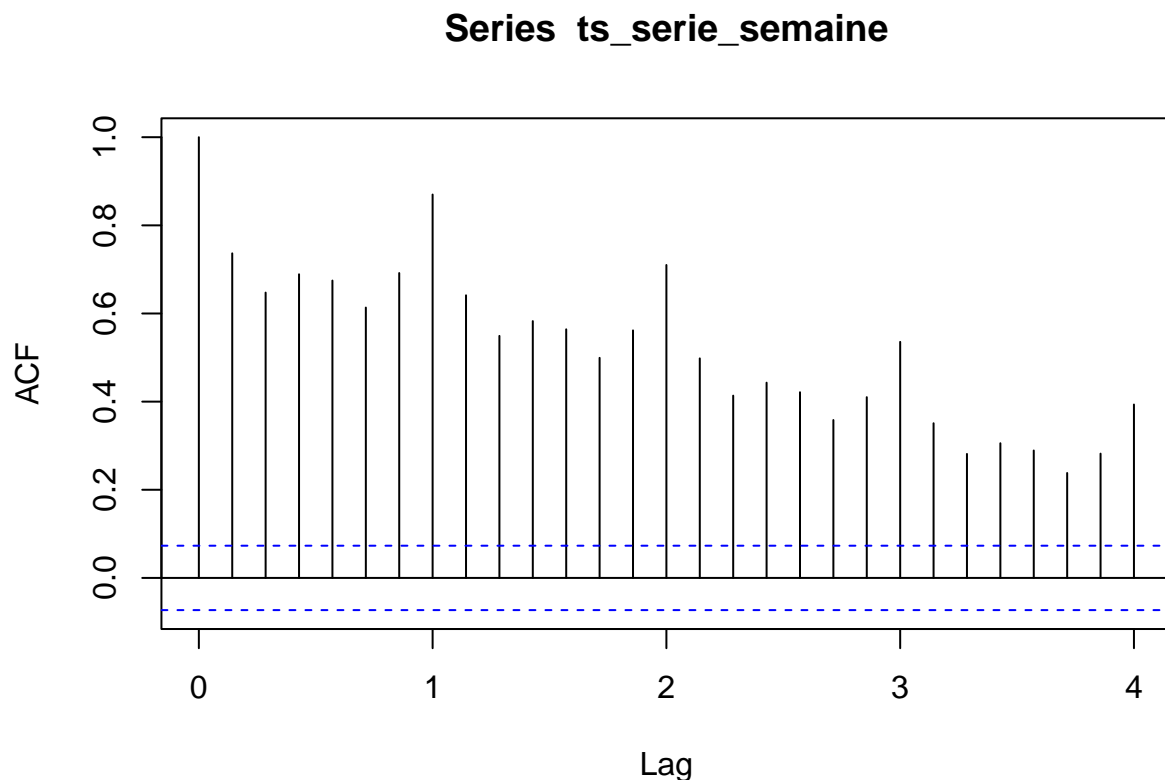
Ce second test nous permet aussi de confirmer notre hypothèse de non stationnarité.

Ainsi, nous avons besoin de différencier la série pour poursuivre l'analyse puisque notre série n'est pas stationnaire. L'hypothèse de stationnarité va nous permettre d'avoir des estimations correctes et plus fiables.

## Bruit blanc

Ensuite, nous allons analyser si notre série est un bruit blanc.

```
acf(ts_serie_semaine)
```



Nous n'avons pas une série de bruit blanc car les autocorrélations ne se situent pas entre les deux lignes en pointillés bleus. Pour une série de bruit blanc, nous nous attendons à avoir 95% des pics entre ces deux lignes mais ici ce n'est pas le cas donc notre série n'est probablement pas un bruit blanc.

Nous allons maintenant tester si la série temporelle peut être différenciée d'un bruit blanc :

```
Box.test(ts_serie_semaine)
```

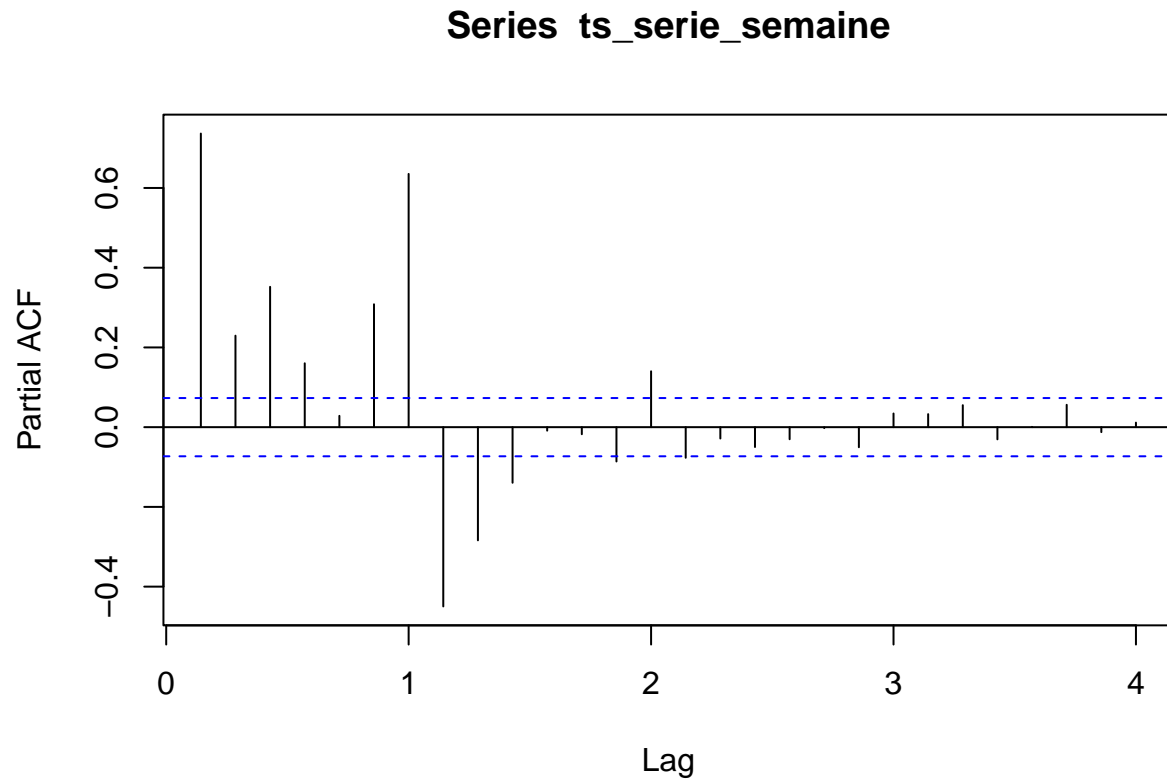
```
##
## Box-Pierce test
##
## data: ts_serie_semaine
## X-squared = 390.56, df = 1, p-value < 2.2e-16
```

Notre p-value est bien inférieure à 5%, la probabilité que la série soit un bruit blanc est presque nulle.

## Différenciation

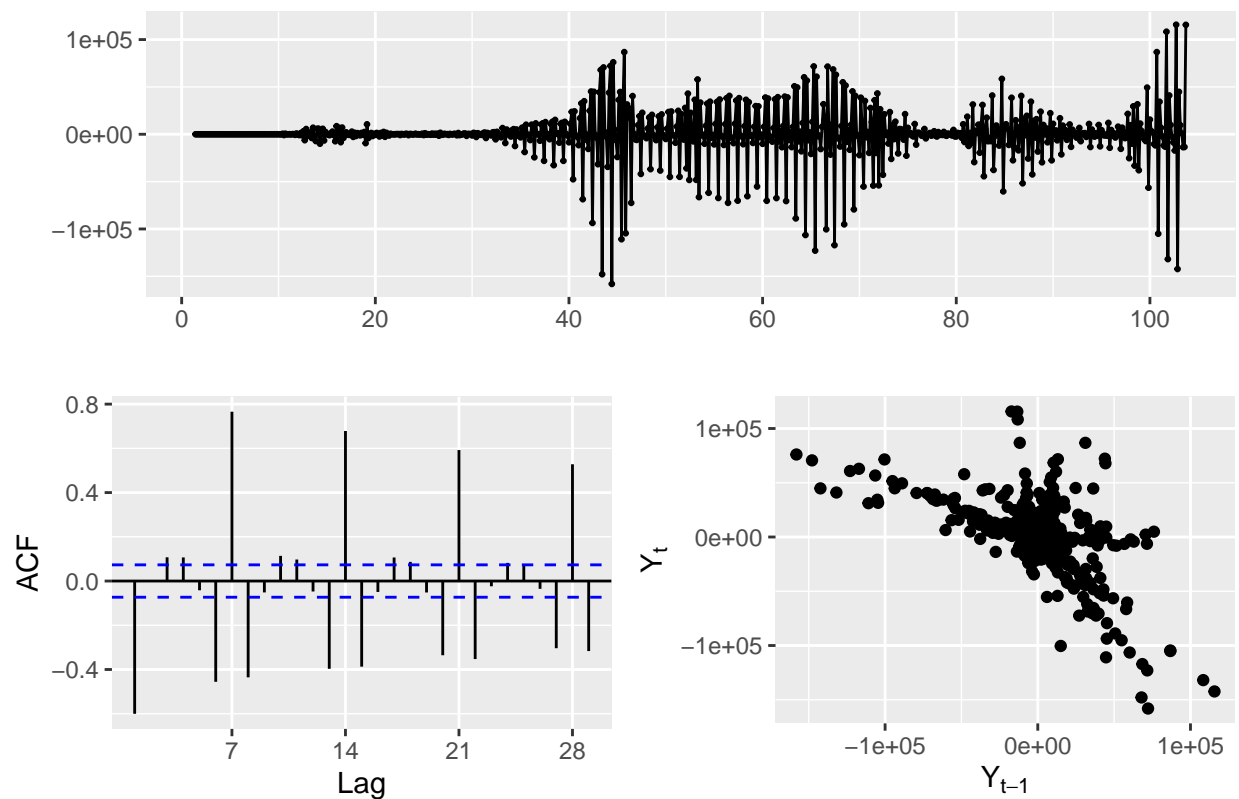
Nous allons observer les autocorrélations partielles de la série pour savoir comment différencier notre série. La différenciation de la série va nous permettre de rendre la série stationnaire puisqu'elle diminue la variance, élimine la tendance et/ou la saisonnalité.

```
pacf(ts_serie_semaine)
```



Ce graphique nous montre 3 pics importants, nous avons donc choisi de différencier 3 fois notre série.

```
serie_diff <- ts_serie_semaine |>  
  diff(differences = 3)  
serie_diff |>  
  ggtsdisplay(plot.type="scatter")
```



Grâce à cette différenciation, nous avons supprimé la tendance et il ne nous reste plus que la saisonnalité. Les auto-corrélations nous confirment cette hypothèse car nous n'avons plus de fortes valeurs successives dans le graphique comme auparavant.

Normalement, notre série est bien devenue stationnaire puisque les valeurs sont centrées et fluctuent autour de 0.

Suite à cette étape, nous allons tester la stationnarité de la série pour pouvoir continuer l'analyse en toute cohérence.

```
adf.test(serie_diff)
```

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

```
##
```

```
## Augmented Dickey-Fuller Test
```

```
##
```

```
## data: serie_diff
```

```
## Dickey-Fuller = -31.973, Lag order = 8, p-value = 0.01
```

```
## alternative hypothesis: stationary
```

```
kpss.test(serie_diff)
```

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

```
##
```

```
## KPSS Test for Level Stationarity
```

```
##
```

```
## data: serie_diff
```

```
## KPSS Level = 0.15682, Truncation lag parameter = 6, p-value = 0.1
```



En conclusion, notre série différenciée est bien stationnaire comme le prouve ces deux tests.

## Modélisation de notre série

Nous allons essayer de choisir le meilleur modèle afin d'estimer notre série.

### Modèle polynomial

Avant de pouvoir faire un modèle polynomial, il faut vérifier la normalité des résidus. Nous pouvons effectuer ceci grâce à un test de Shapiro.

```
shapiro.test(ts_serie_semaine)
```

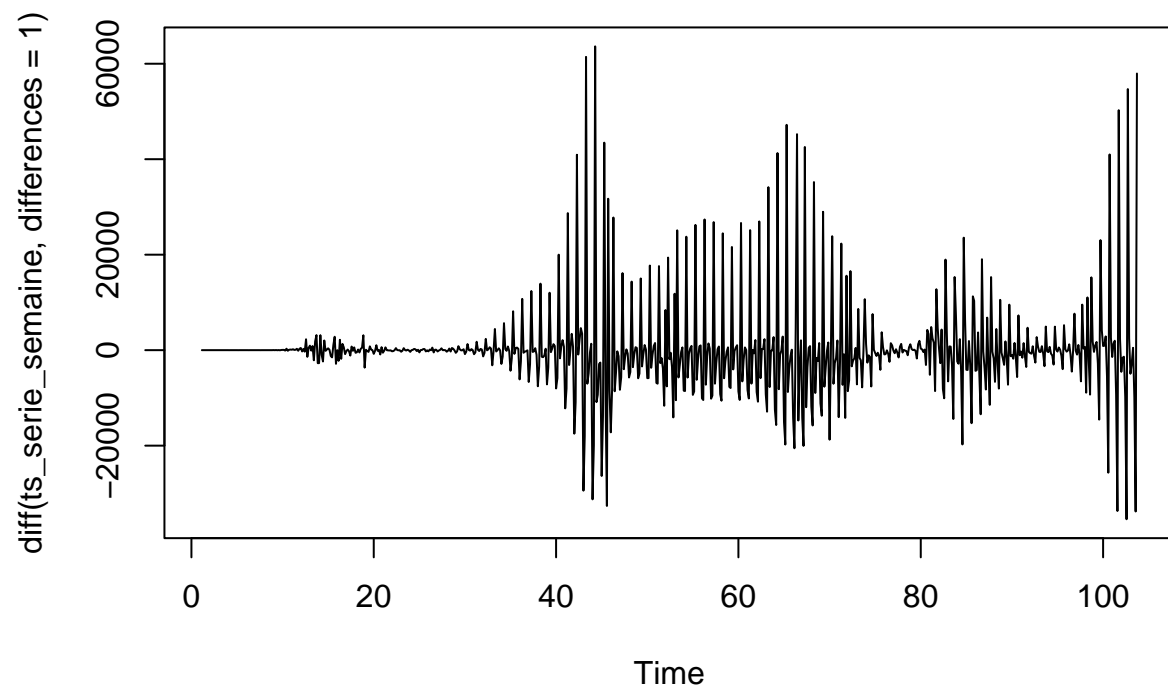
```
##  
## Shapiro-Wilk normality test  
##  
## data:  ts_serie_semaine  
## W = 0.79257, p-value < 2.2e-16
```

La p-value est inférieure à 5%, ce qui nous amène à rejeter l'hypothèse nulle. Nos résidus suivent donc une loi normale.

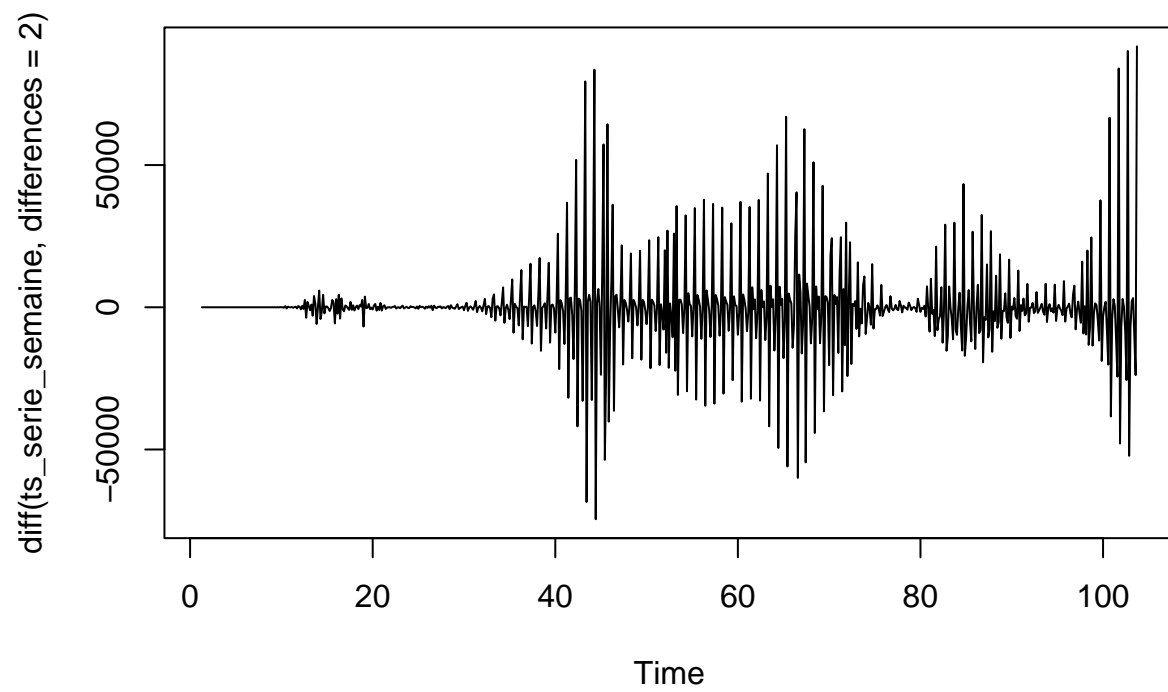
Nous pouvons alors construire notre modèle polynomial.

Il peut être utile de modéliser le nombre de nouveaux cas de COVID-19 en utilisant une méthode de régression polynomiale. Les données montrent une tendance générale à la hausse au fil du temps, une régression polynomiale peut être utilisée pour décrire cette tendance.

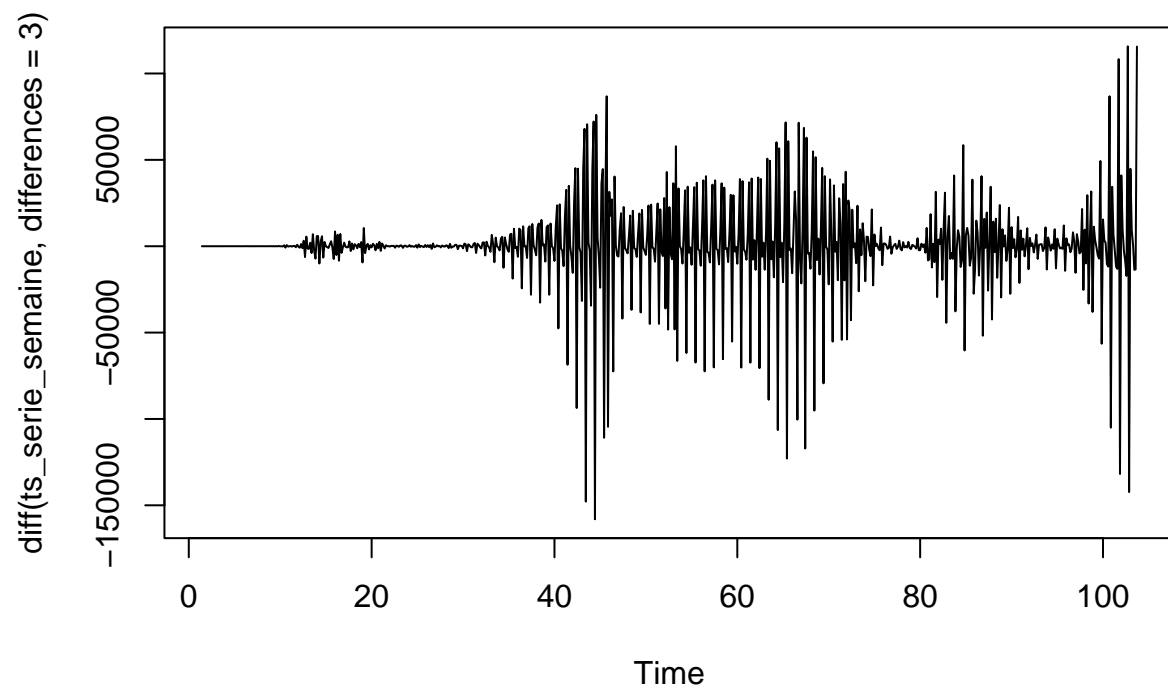
```
plot(diff(ts_serie_semaine, differences = 1), type="l")
```



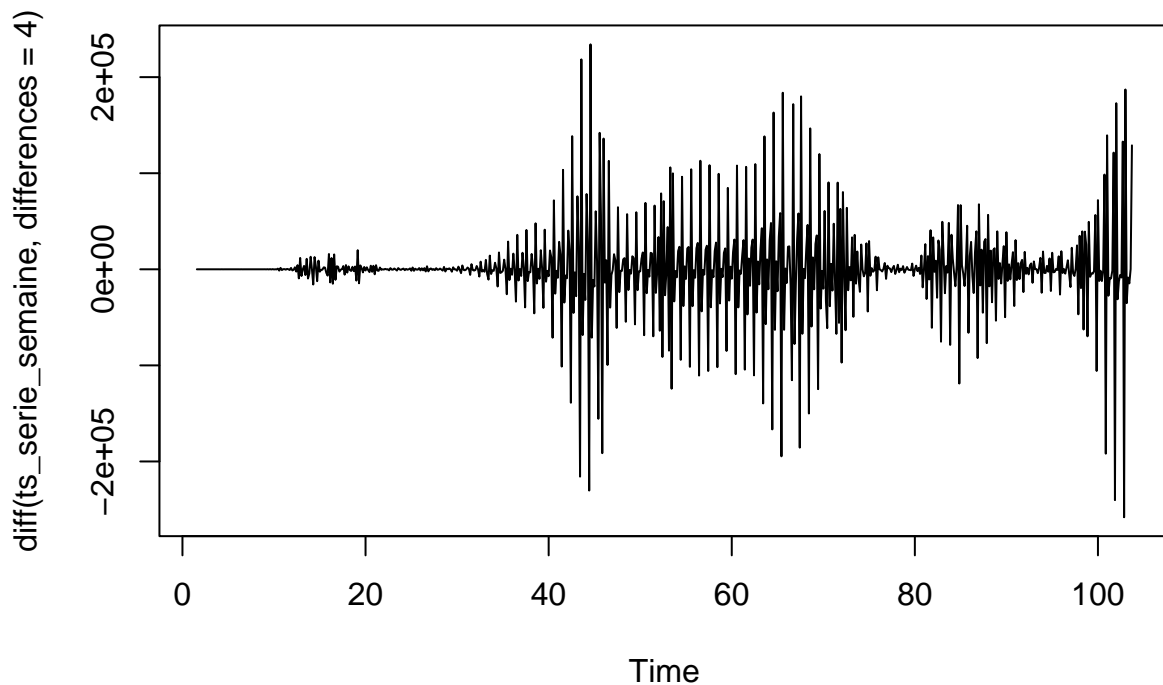
```
plot(diff(ts_serie_semaine, differences = 2),type="l")
```



```
plot(diff(ts_serie_semaine, differences = 3),type="l")
```



```
plot(diff(ts_serie_semaine, differences = 4), type="l")
```



Le degré  $d$  de la tendance polynomiale est 2. On a un graphique avec  $d = 2$  qui est à peu près centré donc on choisit  $d-1 = 1$ .

La période est  $T = 7$  car nous avons des données hebdomadaires.

Nous avons donc créé un modèle polynomial de degré 2 pour modéliser la série. Nous devons donc avoir 2 régresseurs pour la tendance et 6 régresseurs pour la saisonnalité.

Il faut générer les variables explicatives pour ajuster les variables du modèle au sens du MCO.

```
t <- 1:length(ts_serie_semaine)
x <- outer(t,1:6)*(pi/6)
df <- data.frame(ts_serie_semaine,t,cos(x),sin(x[, -6]))

x <- matrix(1,nrow=nrow(serie1),ncol=8) # car 8 régresseurs au total
t <- 1:nrow(serie1)
x[,2] <- t
x[,3] <- t**2
x[,5] <- cos((2*pi*t)/7)
x[,6] <- cos((2*pi*t**2)/7)
x[,7] <- sin((2*pi*t)/7)
x[,8] <- sin((2*pi*t**2)/7)

ts_serie1_lm <- lm(data=df,ts_serie_semaine~.)

summary(ts_serie1_lm)
```

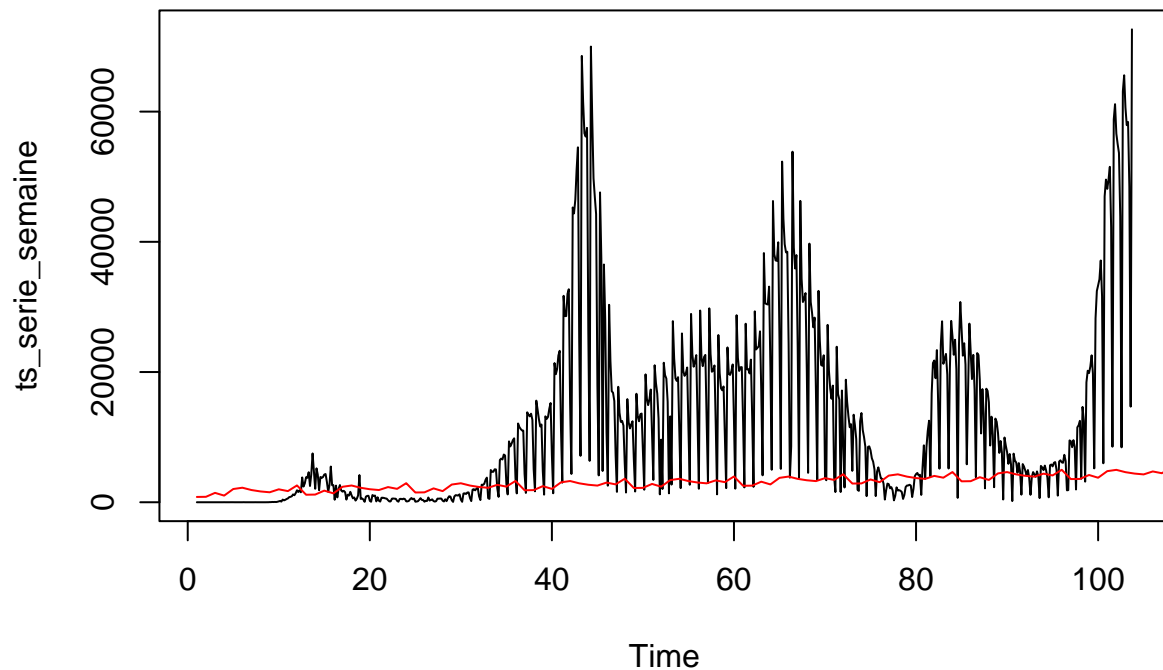
```
##
## Call:
```

```
## lm(formula = ts_serie_semaine ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19426  -7072  -2665   4848  60392
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1456.745    971.698   1.499   0.134
## t           28.590      2.335  12.243 <2e-16 ***
## X1          -157.233    686.311  -0.229   0.819
## X2           355.712    686.311   0.518   0.604
## X3            6.227    686.311   0.009   0.993
## X4           274.870    686.311   0.401   0.689
## X5           241.654    686.311   0.352   0.725
## X6            61.208    485.294   0.126   0.900
## X1.1         -228.513    686.363  -0.333   0.739
## X2.1         -177.238    686.319  -0.258   0.796
## X3.1          -80.877    686.311  -0.118   0.906
## X4.1          -61.130    686.309  -0.089   0.929
## X5.1          188.947    686.308   0.275   0.783
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13020 on 707 degrees of freedom
## Multiple R-squared:  0.1761, Adjusted R-squared:  0.1621
## F-statistic: 12.59 on 12 and 707 DF,  p-value: < 2.2e-16
```

Cette régression polynomiale semble être peu fiable puisque le  $R^2$  est de 0.1761. Cependant, le test de significativité globale du modèle est vérifié pour ce modèle.

Nous allons ensuite le représenter graphiquement pour voir s'il est en adéquation avec les données.

```
plot(ts_serie_semaine)
lines(ts_serie1_lm$fitted.values,col="red")
```



Cette représentation graphique nous permet de voir que le modèle polynomial ne semble pas être le plus adapté pour représenter la série chronologique.

## Modèle auto ARIMA

Afin de pouvoir l'estimer, nous avons utilisé la fonction `auto.arima()` du package `forecast` qui permet d'effectuer une modélisation automatique. En précisant les arguments `trace=T` et `ic=aic`, nous avons donné la main au logiciel R de sélectionner le meilleur modèle sur la base du critère AIC.

Nous avons établi un modèle ARIMA. Cependant, la seule condition pour que la modélisation soit correcte est que la série temporelle modélisée doit être stationnaire. Cette hypothèse explique nos choix précédents de différenciation.

```
model_arima <- auto.arima(serie_diff, ic = "aic", trace=TRUE)
```

```
##
## Fitting models using approximations to speed things up...
##
## ARIMA(2,0,2)(1,1,1)[7] with drift : Inf
## ARIMA(0,0,0)(0,1,0)[7] with drift : 15759.93
## ARIMA(1,0,0)(1,1,0)[7] with drift : 15156.75
## ARIMA(0,0,1)(0,1,1)[7] with drift : Inf
## ARIMA(0,0,0)(0,1,0)[7] : 15757.93
## ARIMA(1,0,0)(0,1,0)[7] with drift : 15208.88
## ARIMA(1,0,0)(2,1,0)[7] with drift : 15165.45
## ARIMA(1,0,0)(1,1,1)[7] with drift : 15158.31
## ARIMA(1,0,0)(0,1,1)[7] with drift : 15151.11
## ARIMA(1,0,0)(0,1,2)[7] with drift : 15151.77
```

```

## ARIMA(1,0,0)(1,1,2)[7] with drift : 15160.28
## ARIMA(0,0,0)(0,1,1)[7] with drift : 15650.08
## ARIMA(2,0,0)(0,1,1)[7] with drift : 14836.19
## ARIMA(2,0,0)(0,1,0)[7] with drift : 14877.04
## ARIMA(2,0,0)(1,1,1)[7] with drift : 14843.59
## ARIMA(2,0,0)(0,1,2)[7] with drift : 14836.89
## ARIMA(2,0,0)(1,1,0)[7] with drift : 14841.63
## ARIMA(2,0,0)(1,1,2)[7] with drift : 14845.58
## ARIMA(3,0,0)(0,1,1)[7] with drift : 14611.42
## ARIMA(3,0,0)(0,1,0)[7] with drift : 14656.16
## ARIMA(3,0,0)(1,1,1)[7] with drift : 14620.43
## ARIMA(3,0,0)(0,1,2)[7] with drift : 14613.39
## ARIMA(3,0,0)(1,1,0)[7] with drift : 14620.93
## ARIMA(3,0,0)(1,1,2)[7] with drift : Inf
## ARIMA(4,0,0)(0,1,1)[7] with drift : 14518.48
## ARIMA(4,0,0)(0,1,0)[7] with drift : 14585.87
## ARIMA(4,0,0)(1,1,1)[7] with drift : 14526.06
## ARIMA(4,0,0)(0,1,2)[7] with drift : 14519.56
## ARIMA(4,0,0)(1,1,0)[7] with drift : 14525.2
## ARIMA(4,0,0)(1,1,2)[7] with drift : 14527.73
## ARIMA(5,0,0)(0,1,1)[7] with drift : 14310.14
## ARIMA(5,0,0)(0,1,0)[7] with drift : 14338.32
## ARIMA(5,0,0)(1,1,1)[7] with drift : 14310.68
## ARIMA(5,0,0)(0,1,2)[7] with drift : 14302.42
## ARIMA(5,0,0)(1,1,2)[7] with drift : 14299.16
## ARIMA(5,0,0)(2,1,2)[7] with drift : 14307.12
## ARIMA(5,0,0)(2,1,1)[7] with drift : 14319.03
## ARIMA(5,0,1)(1,1,2)[7] with drift : Inf
## ARIMA(4,0,1)(1,1,2)[7] with drift : Inf
## ARIMA(5,0,0)(1,1,2)[7] : 14297.16
## ARIMA(5,0,0)(0,1,2)[7] : 14300.42
## ARIMA(5,0,0)(1,1,1)[7] : 14308.68
## ARIMA(5,0,0)(2,1,2)[7] : 14305.12
## ARIMA(5,0,0)(0,1,1)[7] : 14308.14
## ARIMA(5,0,0)(2,1,1)[7] : 14317.03
## ARIMA(4,0,0)(1,1,2)[7] : 14526.05
## ARIMA(5,0,1)(1,1,2)[7] : Inf
## ARIMA(4,0,1)(1,1,2)[7] : Inf

```

```

##
## Now re-fitting the best model(s) without approximations...
##

```

```

## ARIMA(5,0,0)(1,1,2)[7] : 14428.43
##

```

```

## Best model: ARIMA(5,0,0)(1,1,2)[7]

```

```

model_arima

```

```

## Series: serie_diff
## ARIMA(5,0,0)(1,1,2)[7]
##
## Coefficients:
##      ar1      ar2      ar3      ar4      ar5      sar1      sma1      sma2
##    -1.8483  -2.2310  -2.0496  -1.4615  -0.7288  0.7733  -0.4650  -0.3934
## s.e.   0.0257   0.0496   0.0629   0.0590   0.0364   0.0608   0.0785   0.0530
##

```



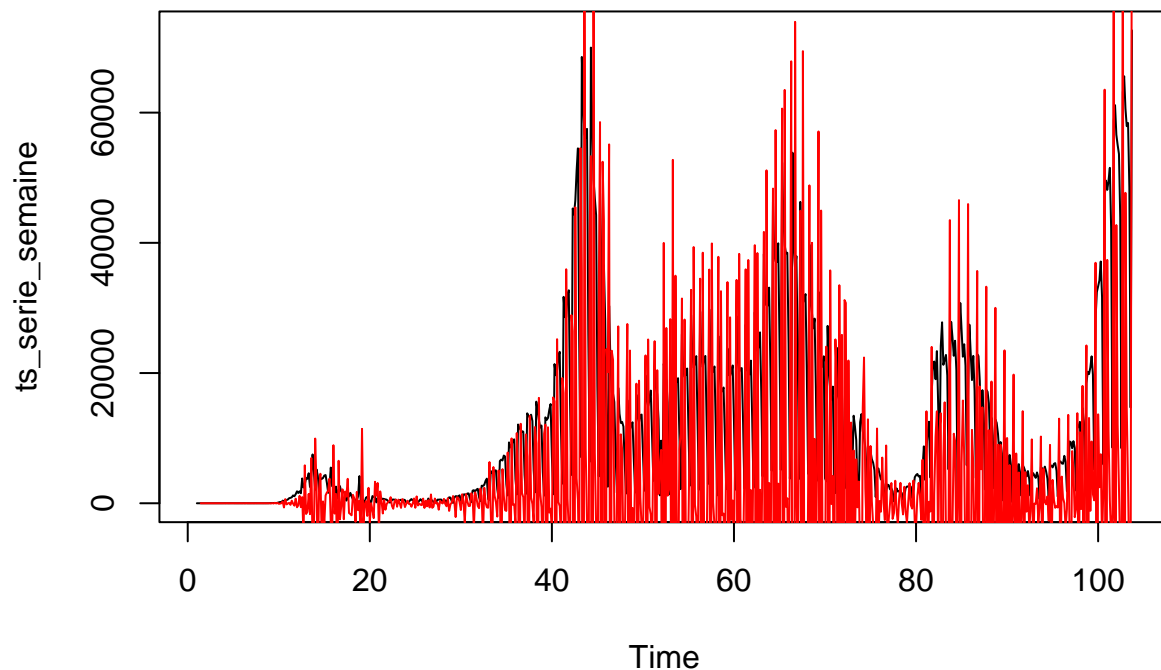
```
## sigma^2 = 38293572: log likelihood = -7205.22
## AIC=14428.43 AICc=14428.69 BIC=14469.52
```

Modèles identifiés : ARIMA(5,0,0)(1,1,2)

```
summary(model_arima)
```

```
## Series: serie_diff
## ARIMA(5,0,0)(1,1,2)[7]
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      sar1      sma1      sma2
##      -1.8483  -2.2310  -2.0496  -1.4615  -0.7288   0.7733  -0.4650  -0.3934
## s.e.   0.0257   0.0496   0.0629   0.0590   0.0364   0.0608   0.0785   0.0530
##
## sigma^2 = 38293572: log likelihood = -7205.22
## AIC=14428.43 AICc=14428.69 BIC=14469.52
##
## Training set error measures:
##              ME      RMSE      MAE MPE MAPE      MASE      ACF1
## Training set 5.417546 6123.108 3022.002 NaN  Inf  0.4098015 -0.1952531
```

```
plot(ts_serie_semaine)
lines(model_arima$fitted,col="red")
```



## Validité du modèle

```
t_stat(model_arima)
```

```
##          ar1          ar2          ar3          ar4          ar5          sar1          sma1
## t.stat -71.98221 -44.94924 -32.58278 -24.77639 -20.00935 12.72634 -5.921194
## p.val   0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
##          sma2
## t.stat -7.427172
## p.val   0.000000
```

Grâce à la fonction `t_stat()` nous pouvons voir que les coefficients de notre modèle diffèrent beaucoup de 0, en particulier pour les coefficients `ar1`, `ar2` et `ar3`. En effet, ceux-ci présentent une différence moyenne d'environ 50 de la valeur 0. Les valeurs `p` correspondantes sont toutes très proches de zéro (0.00000, 0.0000, 0.000001), ce qui suggère que les coefficients sont significativement différents de zéro.

Pour que le modèle soit valide, il faut vérifier la normalité des résidus.

```
shapiro.test(model_arima$residuals)
```

```
##
## Shapiro-Wilk normality test
##
## data:  model_arima$residuals
## W = 0.70669, p-value < 2.2e-16
```

Le test de Shapiro-Wilk nous permet de prouver que cette hypothèse est bien respectée.

## Modèle ARIMA

Pour ajuster le modèle aux données, nous allons essayer d'estimer les coefficients `p`, `d` et `q` du modèle ARIMA.

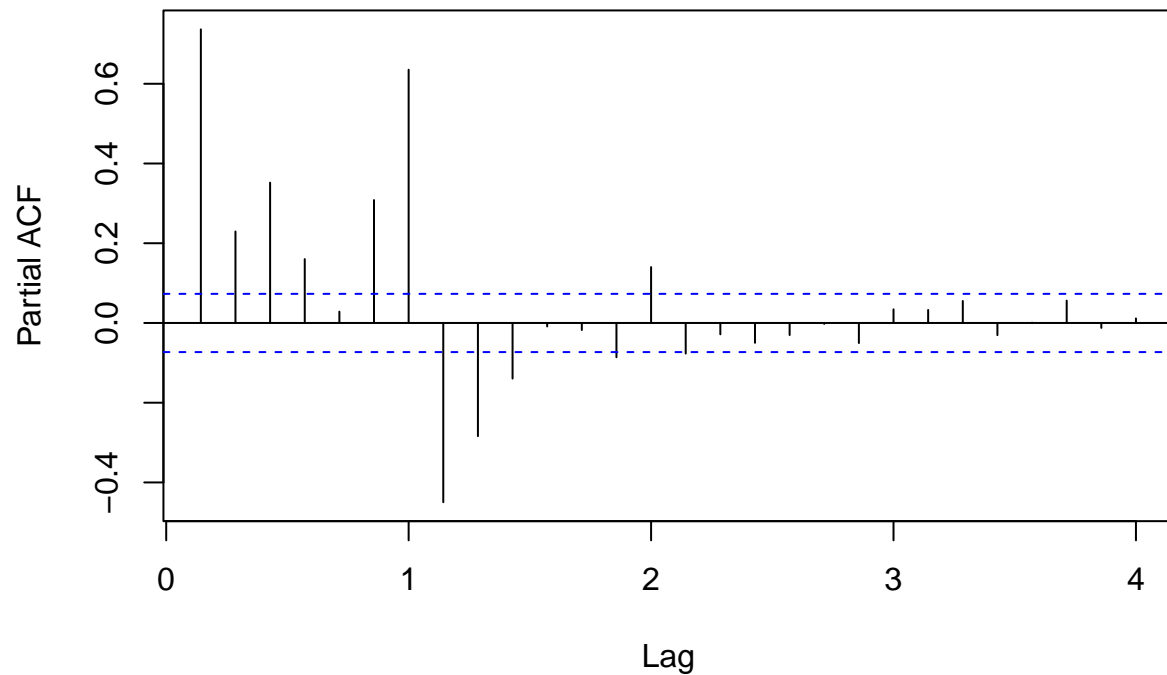
Premièrement, nous allons chercher le paramètre `p`, nombre de termes auto-régressifs. Nous avons choisi de déterminer `p` égal à 0.

Deuxièmement, le paramètre `d` correspond au nombre de différenciations et nous allons tenter de le déterminer. Puisque notre série n'est pas stationnaire, `d` est égal à 1. Si nous avons une série déjà stationnarisée, nous aurions choisi 0.

Dernièrement, nous allons trouver `q`, le nombre de termes de moyenne mobile. Grâce au graphique des auto-corrélations partielles, nous pouvons le choisir. Comme vu précédemment, nous avons déterminé `q = 3`.

```
pacf(ts_serie_semaine)
```

## Series ts\_serie\_semaine



Après ces différents choix, nous obtenons donc un modèle ARIMA tel que :

```
modele_arima1 <- arima(ts_serie_semaine,order=c(0,1,3))
summary(modele_arima1)
```

```
##
## Call:
## arima(x = ts_serie_semaine, order = c(0, 1, 3))
##
## Coefficients:
##      ma1      ma2      ma3
##    -0.8964 -0.5257  0.6902
## s.e.    0.0284   0.0425  0.0283
##
## sigma^2 estimated as 55193608:  log likelihood = -7430.49,  aic = 14868.97
##
## Training set error measures:
##              ME       RMSE       MAE  MPE  MAPE       MASE       ACF1
## Training set 284.4816 7424.079 4430.149 NaN   Inf  0.8951975 0.08225525
```

## Comparaison des modèles

```
c(AIC(model_arima),AIC(modele_arima1),AIC(ts_serie1_lm))
```

```
## [1] 14428.43 14868.97 15701.26
```

```
c(BIC(model_arma),BIC(modele_arma1),BIC(ts_serie1_lm))
```

```
## [1] 14469.52 14887.28 15765.37
```

Nous aurons tendance à privilégier le modèle auto.ARIMA puisque les critères de l'AIC et le BIC sont minimisés. Notre modèle ARIMA manuel a tout de même l'air correct mais il est moins pertinent. Le modèle le moins intéressant est donc le modèle polynomial.

Il aurait été intéressant d'effectuer un test anova pour comparer les modèles. Cependant, nous ne pouvons pas utiliser la méthode anova() sur un objet de classe Arima.

## Etude des résidus

La fonction Box.test examine l'hypothèse nulle de nullité des H premières auto-covariance. Par défaut H est fixé à 1, et seule la nullité de l'auto-covariance d'ordre 1 est testée.

Pour tester si la série peut-être apparentée à un bruit blanc, nous fixerons un H de l'ordre de 7.

```
Box.test(ts_serie_semaine,lag=7)
```

```
##  
## Box-Pierce test  
##  
## data: ts_serie_semaine  
## X-squared = 2522.4, df = 7, p-value < 2.2e-16
```

Puisque la p-value est inférieure à 5%, on rejette l'hypothèse de non-autocorrélation. Cela implique que la série temporelle présente une autocorrélation significative et que les valeurs successives de la série temporelle sont dépendantes les unes des autres.

La fonction ks.test() est une fonction de test de Kolmogorov-Smirnov dans R, qui permet de comparer une distribution empirique à une distribution théorique normale.

```
ks.test(ts_serie_semaine, "pnorm", mean(ts_serie_semaine), sd(ts_serie_semaine))
```

```
## Warning in ks.test.default(ts_serie_semaine, "pnorm", mean(ts_serie_semaine), :  
## aucun ex-aequo ne devrait être présent pour le test de Kolmogorov-Smirnov  
##  
## Asymptotic one-sample Kolmogorov-Smirnov test  
##  
## data: ts_serie_semaine  
## D = 0.20414, p-value < 2.2e-16  
## alternative hypothesis: two-sided
```

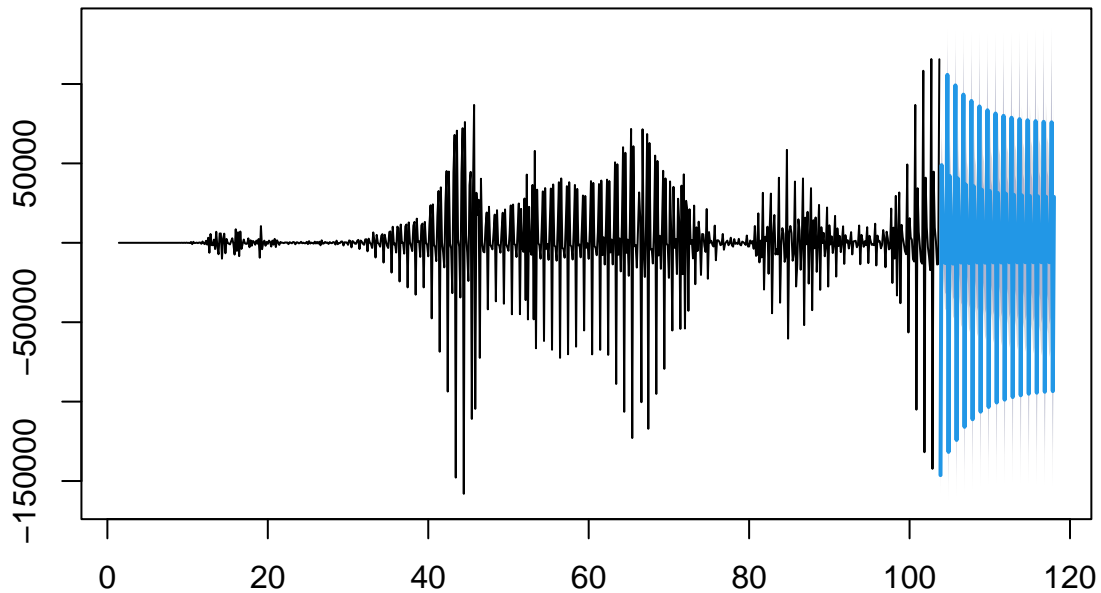
Les données ne suivent pas une distribution théorique car la p-value est inférieure à 5%. On rejette donc l'hypothèse nulle.

## Prévisions

Suite à notre analyse, nous avons utilisé la fonction forecast pour émettre des prévisions sur notre série pour les 100 prochains jours (en fixant h = 100).

```
forecast_cases <- forecast::forecast(model_arma, h = 100)  
plot(forecast_cases)
```

## Forecasts from ARIMA(5,0,0)(1,1,2)[7]



Nous pouvons observer des prévisions assez stables dans le temps. Cependant, nos données correspondant à une pandémie, les prévisions sont d'autant plus compliquées à émettre.

Nous pouvons observer les valeurs de ces prévisions :

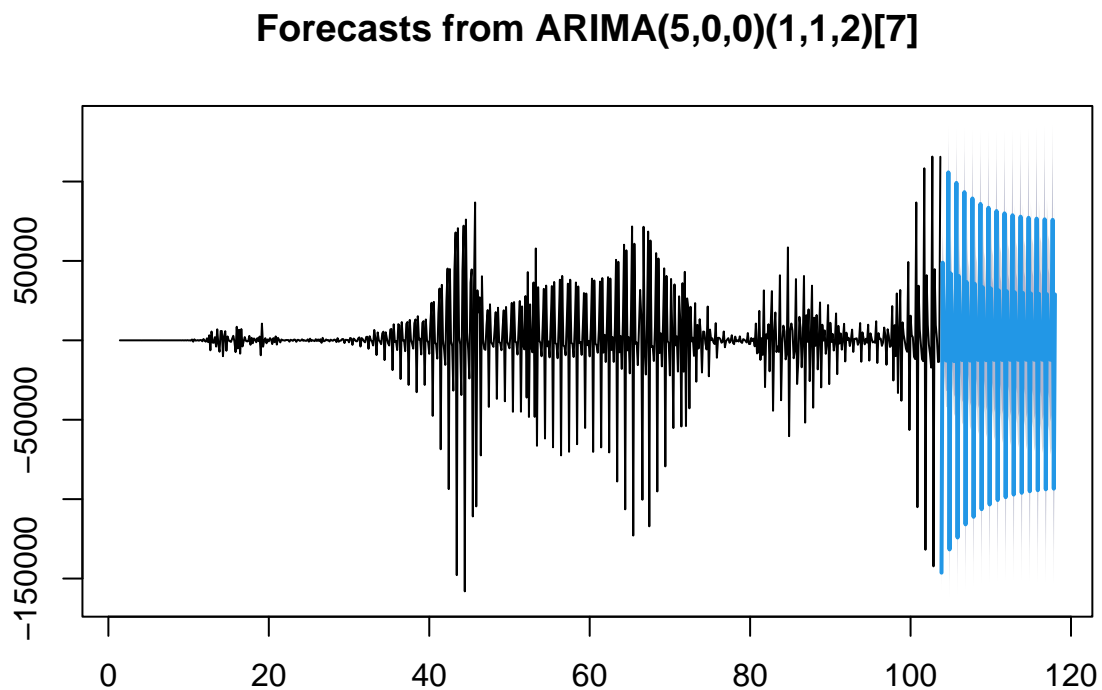
```
predict(model_arima)
```

```
## $pred
## Time Series:
## Start = c(103, 7)
## End = c(103, 7)
## Frequency = 7
## [1] -146297.3
##
## $se
## Time Series:
## Start = c(103, 7)
## End = c(103, 7)
## Frequency = 7
## [1] 6188.18
```

Afin de rendre notre analyse plus pertinente, nous allons pouvoir comparer si nos prévisions sont en adéquation avec la deuxième sous-série.

## ANALYSER AVEC LA DEUXIEME SOUS SERIE -> REUSSIR A FAIRE LE GRAPHIQUE

```
plot(forecast_cases)
```



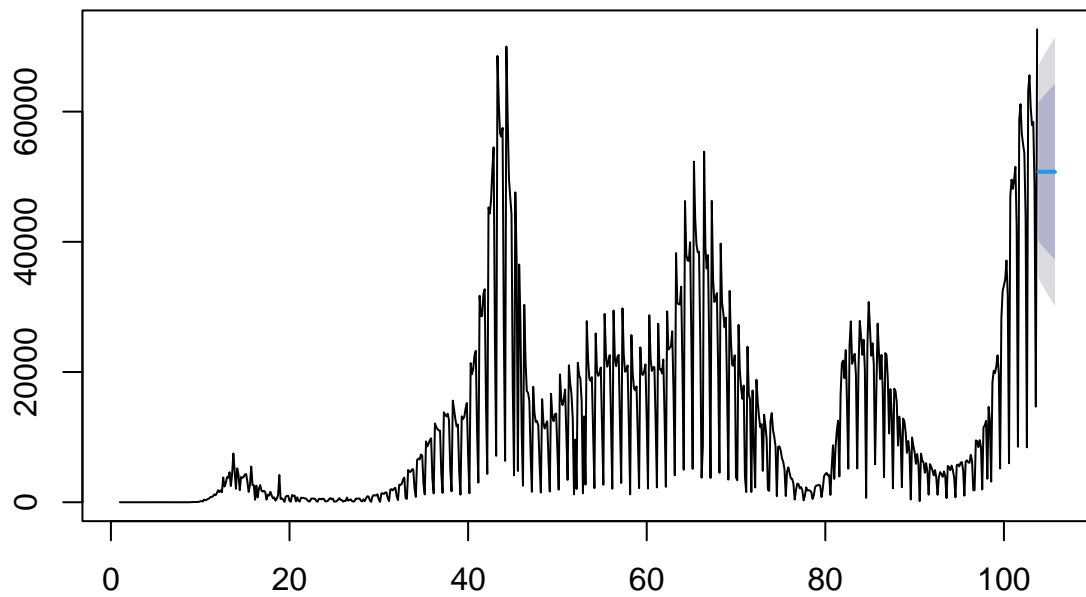
```
# lines(ts_seri2,col="red")
```

### Lissage exponentiel

Nous allons tenter de réaliser des prévisions avec un lissage exponentiel :

```
les <- ets(ts_serie_semaine,model="ANN")  
pred <- predict(les)  
plot(pred)
```

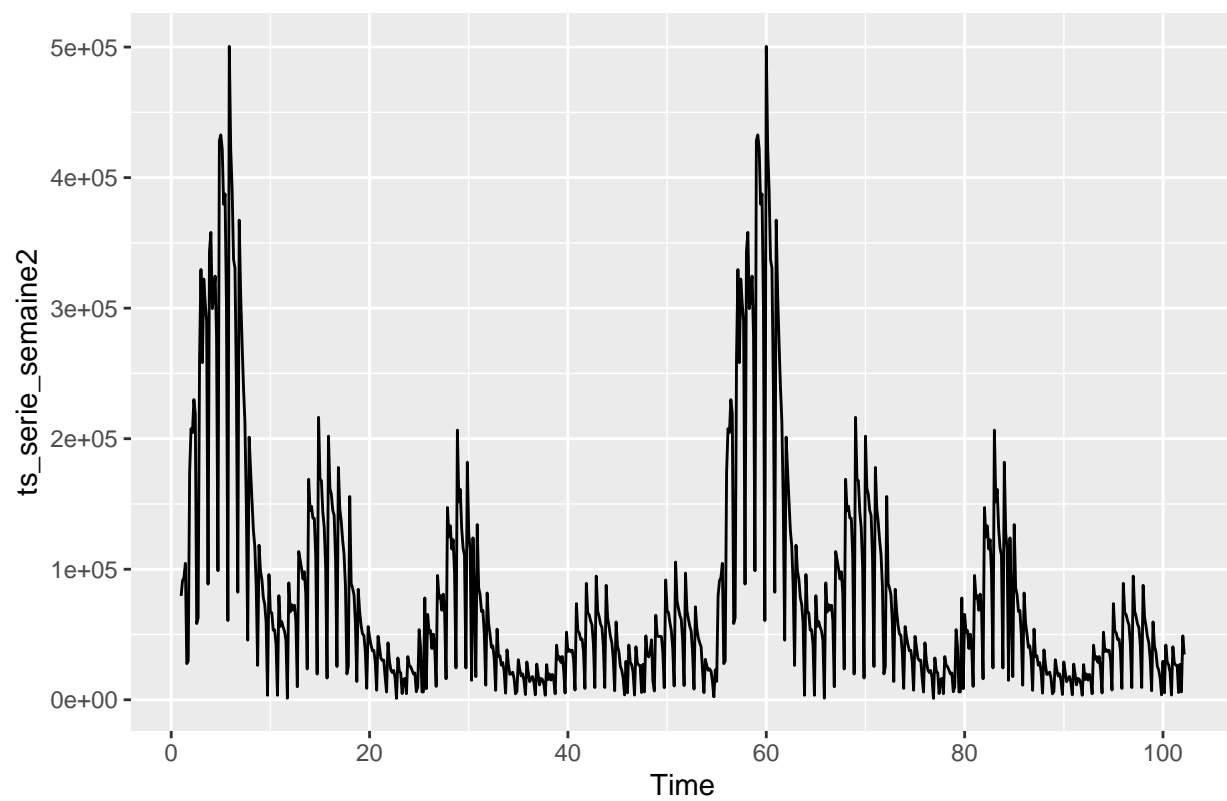
## Forecasts from ETS(A,N,N)



**A INTERPRETER !!**

**MODIF TS\_SERIE\_2**

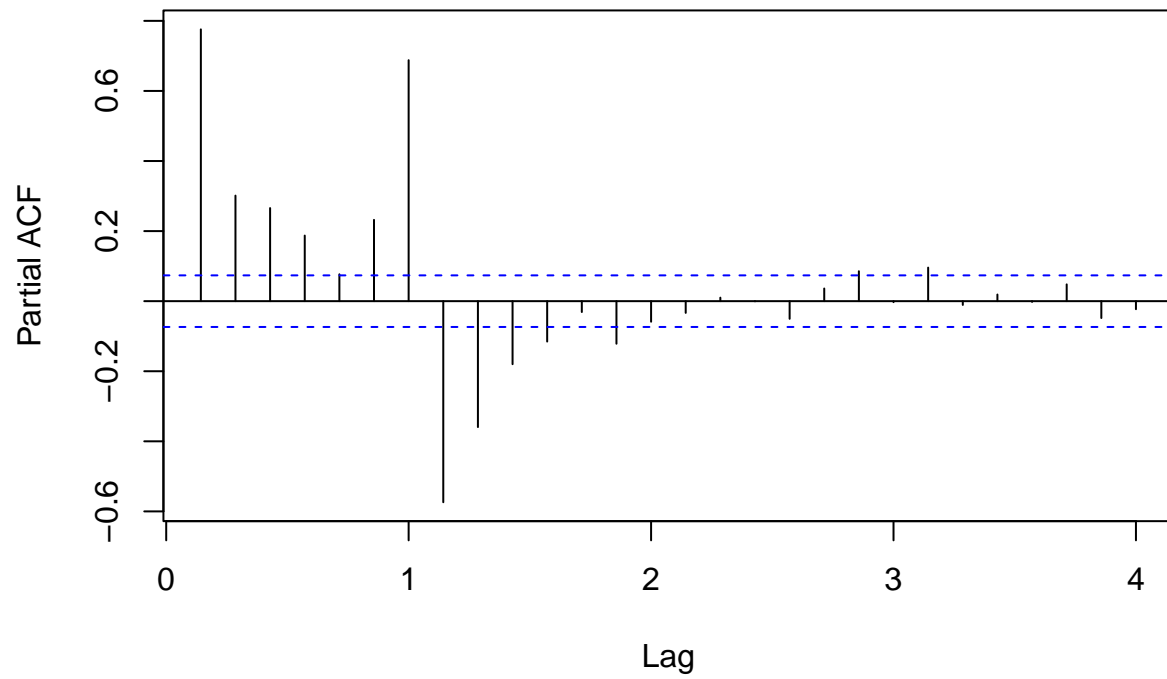
```
ts_serie_semaine2 <- ts(ts_serie2, frequency = 7)
autoplot(ts_serie_semaine2) # Visualisation des données
```



```
pacf(ts_serie_semaine2)
```

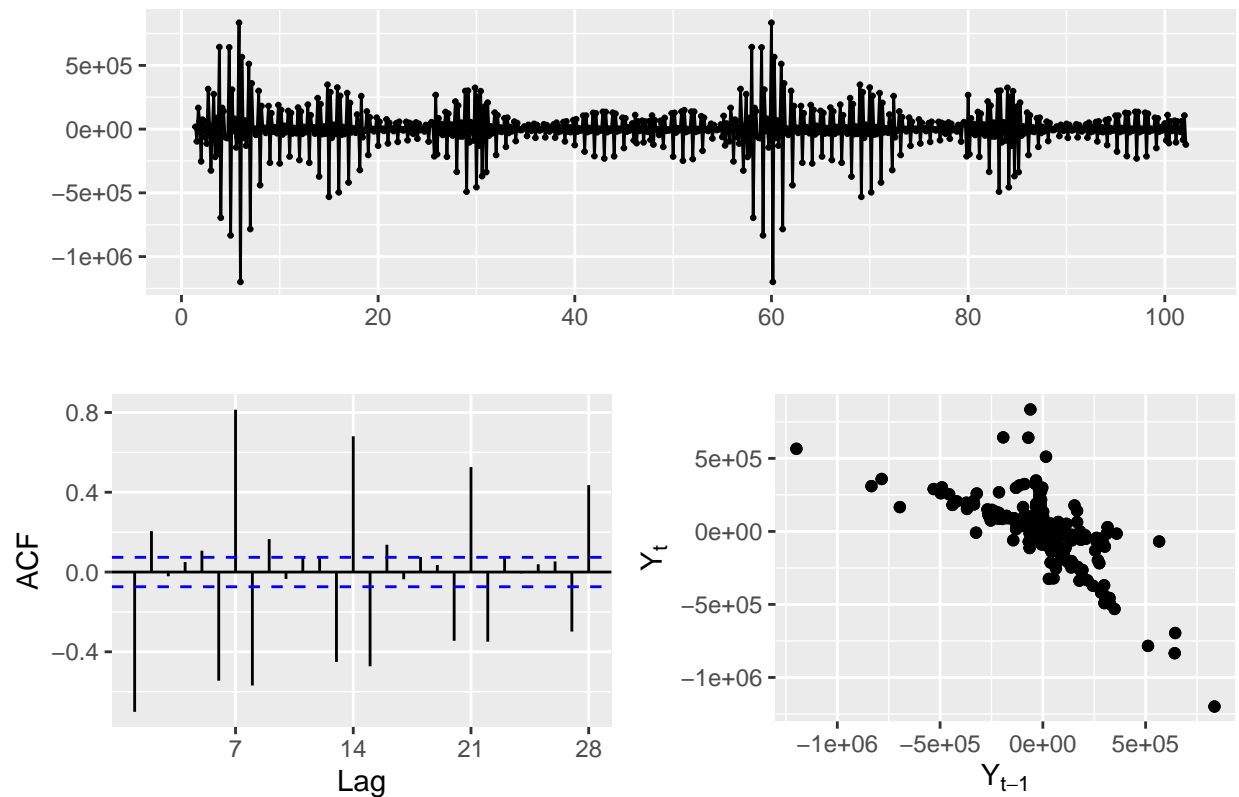


## Series ts\_serie\_semaine2



Il faut différencier la série 3 fois.

```
serie_diff2 <- ts_serie_semaine2 |>  
  diff(differences = 3)  
serie_diff2 |>  
  ggtsdisplay(plot.type="scatter")
```



Nous allons maintenant tester la stationnarité de la série :

```
adf.test(serie_diff2)
```

```
## Warning in adf.test(serie_diff2): p-value smaller than printed p-value
```

```
##
```

```
## Augmented Dickey-Fuller Test
```

```
##
```

```
## data: serie_diff2
```

```
## Dickey-Fuller = -32.58, Lag order = 8, p-value = 0.01
```

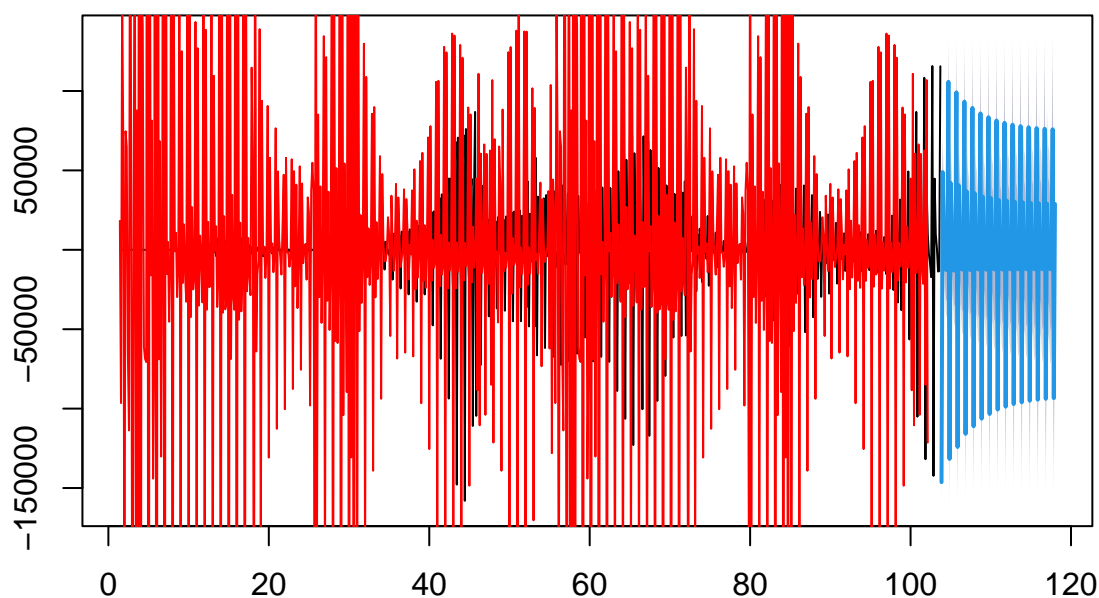
```
## alternative hypothesis: stationary
```

La p-value associée au test de Dickey-Fuller est inférieure à 5%, cela nous permet donc d'affirmer que la série est bien stationnaire à présent.

Nous allons donc pouvoir comparer nos prévisions à cette partie de la série pour vérifier la cohérence de nos prévisions.

```
plot(forecast_cases)
lines(serie_diff2,col="red")
```

## Forecasts from ARIMA(5,0,0)(1,1,2)[7]



serie\_diff2

```
## Time Series:
## Start = c(1, 4)
## End = c(102, 2)
## Frequency = 7
## [1] 18206 -96510 166659 63581 -253592 74681 64820 -64502
## [9] -113738 314938 29299 -325085 -8610 274725 -218135 87786
## [17] -192874 643815 -695654 166384 140953 -61265 -68528 -70663
## [25] 641560 -834011 310015 -16066 81373 -144093 -60503 835023
## [33] -1199045 565753 -68953 68384 -130201 15334 511555 -784170
## [41] 359156 -15289 4999 -45299 -3102 299910 -439787 181469
## [49] -952 10727 -29796 -10395 181417 -263940 125022 -20838
## [57] 16035 -17179 -32705 189295 -269524 150663 -42321 26878
## [65] -24928 -16753 143448 -215129 126961 -35154 8898 -4812
## [73] -33108 169352 -242980 138515 -37696 17603 -23542 -22703
## [81] 192437 -263955 109574 -938 14888 -36055 -18286 243681
## [89] -372192 196509 -39595 21214 -37505 -33011 348766 -531076
## [97] 290064 -68733 34610 -29972 -37776 326723 -496345 261149
## [105] -44026 17072 -38786 -18462 283085 -418793 205933 -26094
## [113] 7540 -16885 -28567 114717 52096 -321358 259597 -64067
## [121] -15452 -6664 138948 -203118 93813 -2420 1985 -17728
## [129] -3360 90571 -130848 57949 -1383 4031 -14142 -4281
## [137] 76536 -112727 49361 6905 -4656 -15600 15310 47260
## [145] -100370 57913 -1582 -1431 -10689 -3183 57015 -87646
## [153] 49536 -22991 40986 -36694 -2370 52573 -75547 41740
```

## [161]	-8391	2022	3688	-18785	33030	22606	-98039	36957
## [169]	54829	37632	-213341	267970	-196982	85227	-29772	39901
## [177]	-46031	1445	134537	-217020	121282	-32100	36541	-50469
## [185]	-8625	216922	-335018	184736	-57436	51405	-58686	-11673
## [193]	299234	-491034	301736	-104075	54366	-5778	-89554	324339
## [201]	-456049	254512	-130780	298019	-369050	153845	177984	-337286
## [209]	208214	-51232	21896	-33427	-892	129797	-203263	108863
## [217]	-17234	13522	-22145	-3746	85685	-139138	89922	-29997
## [225]	17408	-18379	-406	59007	-93810	47710	4613	-7109
## [233]	-7155	-763	20499	9022	-54274	34078	4251	-13290
## [241]	929	41515	-68450	32936	9643	-10854	-5068	995
## [249]	38100	-64747	31940	12585	-18664	3171	-2636	37463
## [257]	-67101	50668	-24837	18322	-17468	785	55903	-96579
## [265]	61080	-19509	15455	-23972	8289	70743	-125464	77887
## [273]	-19563	6115	-11378	-7783	105761	-176556	106426	-23201
## [281]	9441	-15552	-9014	127796	-212001	124340	-24446	4575
## [289]	-13084	-6160	136177	-230475	134910	-24345	4582	-13937
## [297]	-7523	128995	-213376	120931	-19343	4361	-10506	-4859
## [305]	89614	-148529	85232	-20577	14577	-11286	-3152	44869
## [313]	-88426	110834	-110509	60596	-14526	-8290	66359	-103908
## [321]	60381	-9802	-21460	62546	-84975	106859	-121387	69626
## [329]	-6327	-1104	-45717	65070	41203	-139115	88342	-13542
## [337]	-1781	-11197	-3075	124420	-215537	127670	-27535	7377
## [345]	-9294	-6812	138006	-248132	148784	-23630	1498	-15256
## [353]	-4025	137501	-235043	137383	-22034	1471	-14782	1835
## [361]	101702	-170347	89334	-6533	-2587	-15088	15056	42335
## [369]	-73346	31877	3270	-7671	-2776	1481	36966	-62169
## [377]	103782	-128014	44264	18206	-96510	166659	63581	-253592
## [385]	74681	64820	-64502	-113738	314938	29299	-325085	-8610
## [393]	274725	-218135	87786	-192874	643815	-695654	166384	140953
## [401]	-61265	-68528	-70663	641560	-834011	310015	-16066	81373
## [409]	-144093	-60503	835023	-1199045	565753	-68953	68384	-130201
## [417]	15334	511555	-784170	359156	-15289	4999	-45299	-3102
## [425]	299910	-439787	181469	-952	10727	-29796	-10395	181417
## [433]	-263940	125022	-20838	16035	-17179	-32705	189295	-269524
## [441]	150663	-42321	26878	-24928	-16753	143448	-215129	126961
## [449]	-35154	8898	-4812	-33108	169352	-242980	138515	-37696
## [457]	17603	-23542	-22703	192437	-263955	109574	-938	14888
## [465]	-36055	-18286	243681	-372192	196509	-39595	21214	-37505
## [473]	-33011	348766	-531076	290064	-68733	34610	-29972	-37776
## [481]	326723	-496345	261149	-44026	17072	-38786	-18462	283085
## [489]	-418793	205933	-26094	7540	-16885	-28567	114717	52096
## [497]	-321358	259597	-64067	-15452	-6664	138948	-203118	93813
## [505]	-2420	1985	-17728	-3360	90571	-130848	57949	-1383
## [513]	4031	-14142	-4281	76536	-112727	49361	6905	-4656
## [521]	-15600	15310	47260	-100370	57913	-1582	-1431	-10689
## [529]	-3183	57015	-87646	49536	-22991	40986	-36694	-2370
## [537]	52573	-75547	41740	-8391	2022	3688	-18785	33030
## [545]	22606	-98039	36957	54829	37632	-213341	267970	-196982
## [553]	85227	-29772	39901	-46031	1445	134537	-217020	121282
## [561]	-32100	36541	-50469	-8625	216922	-335018	184736	-57436
## [569]	51405	-58686	-11673	299234	-491034	301736	-104075	54366
## [577]	-5778	-89554	324339	-456049	254512	-130780	298019	-369050
## [585]	153845	177984	-337286	208214	-51232	21896	-33427	-892

## [593]	129797	-203263	108863	-17234	13522	-22145	-3746	85685
## [601]	-139138	89922	-29997	17408	-18379	-406	59007	-93810
## [609]	47710	4613	-7109	-7155	-763	20499	9022	-54274
## [617]	34078	4251	-13290	929	41515	-68450	32936	9643
## [625]	-10854	-5068	995	38100	-64747	31940	12585	-18664
## [633]	3171	-2636	37463	-67101	50668	-24837	18322	-17468
## [641]	785	55903	-96579	61080	-19509	15455	-23972	8289
## [649]	70743	-125464	77887	-19563	6115	-11378	-7783	105761
## [657]	-176556	106426	-23201	9441	-15552	-9014	127796	-212001
## [665]	124340	-24446	4575	-13084	-6160	136177	-230475	134910
## [673]	-24345	4582	-13937	-7523	128995	-213376	120931	-19343
## [681]	4361	-10506	-4859	89614	-148529	85232	-20577	14577
## [689]	-11286	-3152	44869	-88426	110834	-110509	60596	-14526
## [697]	-8290	66359	-103908	60381	-9802	-21460	62546	-84975
## [705]	106859	-121387						