Projet Covid

Margaux Bailleul / Oriane Duclos / Marie Guibert

2023-05-11

```
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.

```
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

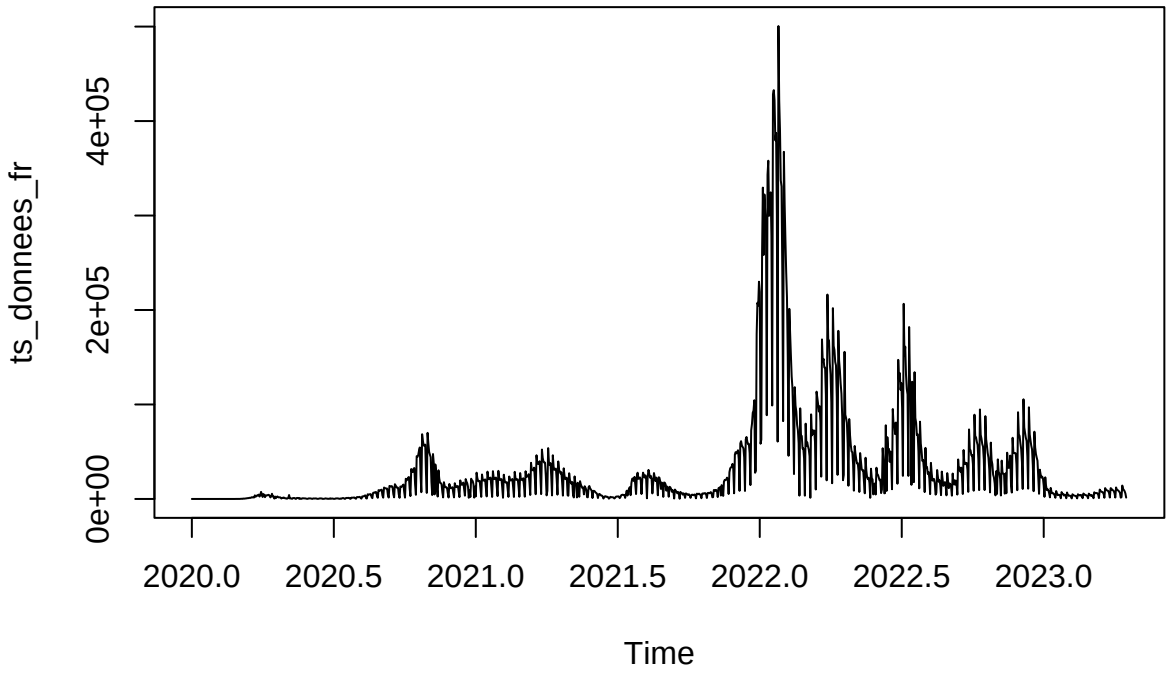
Premièrement, nous allons transformer nos données en séries temporelles pour pouvoir réaliser notre analyse.

```
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

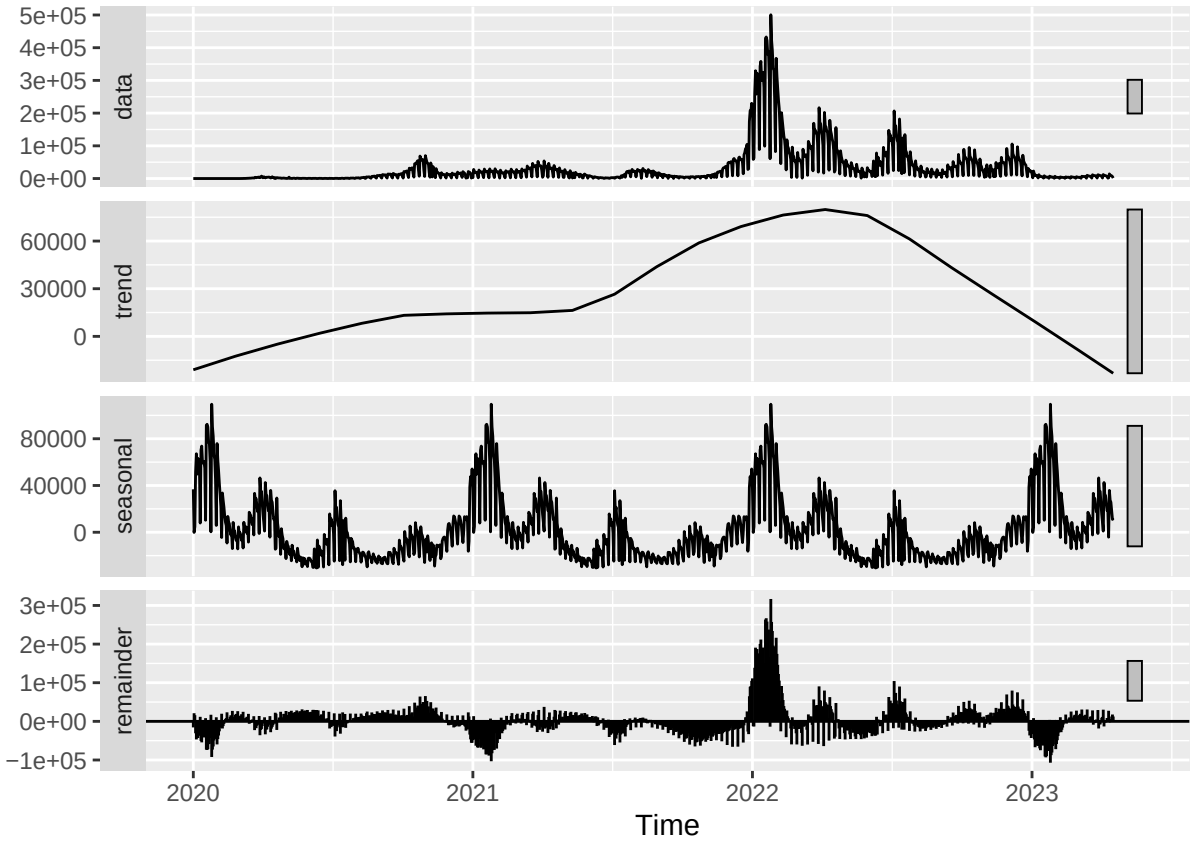
```
plot(ts_donnees_fr)
```



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.
De plus, nous pouvons voir une saisonnalité annuelle.

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.

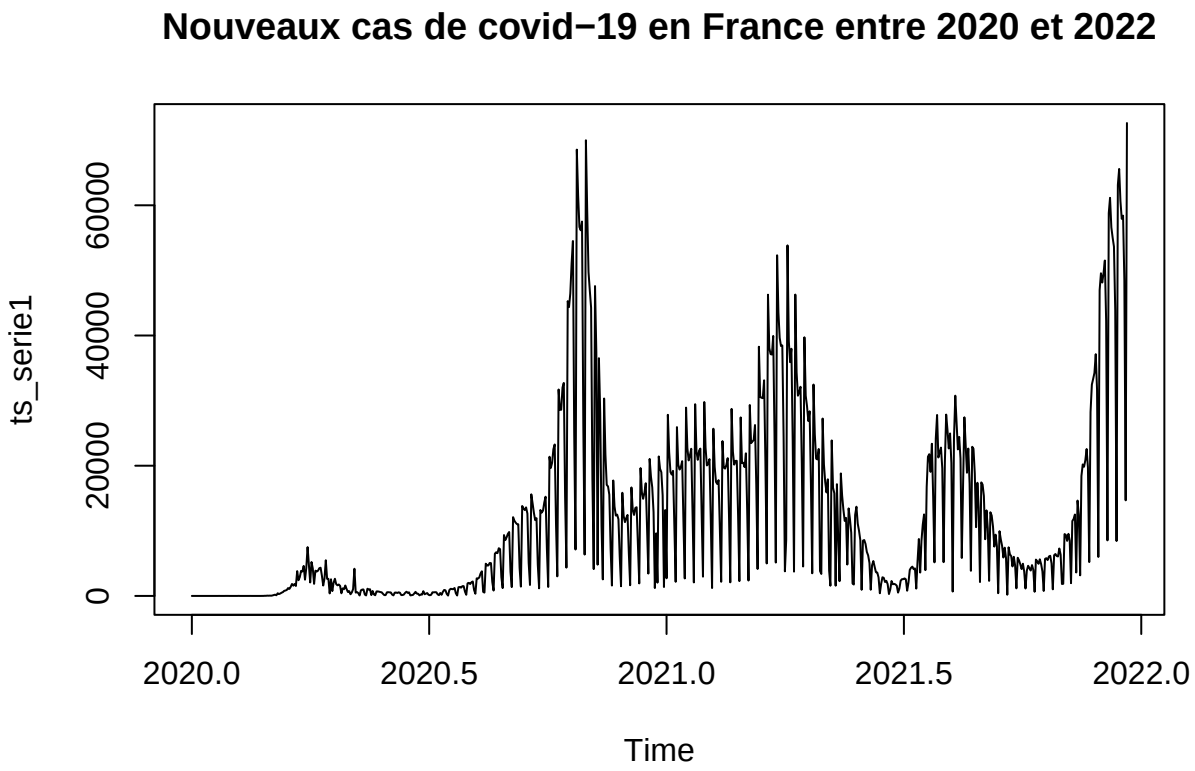
```
serie1 <- donnees_fr |>
  filter(date<="2021-12-22")
# serie1
ts_serie1 <- ts(serie1$new_cases,start = c(2020,1,3), frequency = 365)



serie2 <- donnees_fr |>
  filter(date>"2021-12-22", date<="2023-01-05")
# serie1
ts_serie2 <- ts(serie2$new_cases,start = c(2021,31,12), frequency = 365)



serie3 <- donnees_fr |>
  filter(date>"2023-01-05")
# serie3
ts_serie3 <- 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

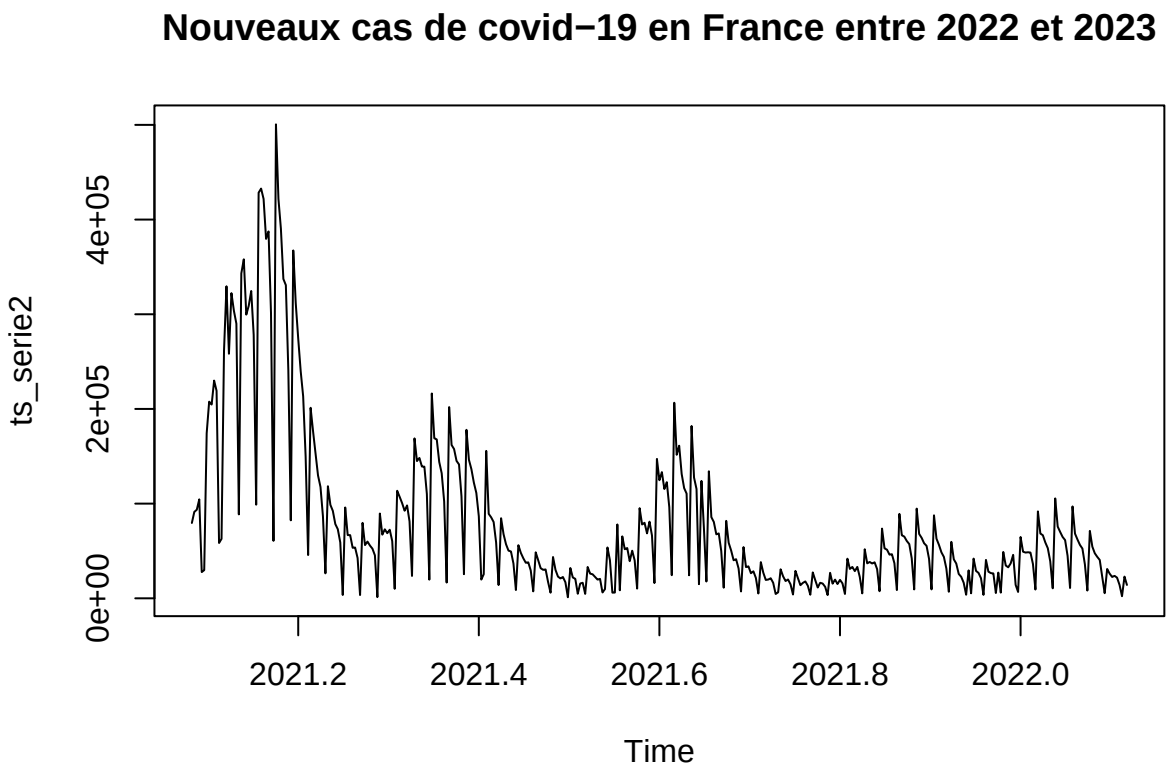
Nous pouvons maintenant les visualiser :

```
plot(ts_serie1,main="Nouveaux cas de covid-19 en France entre 2020 et 2022")
```



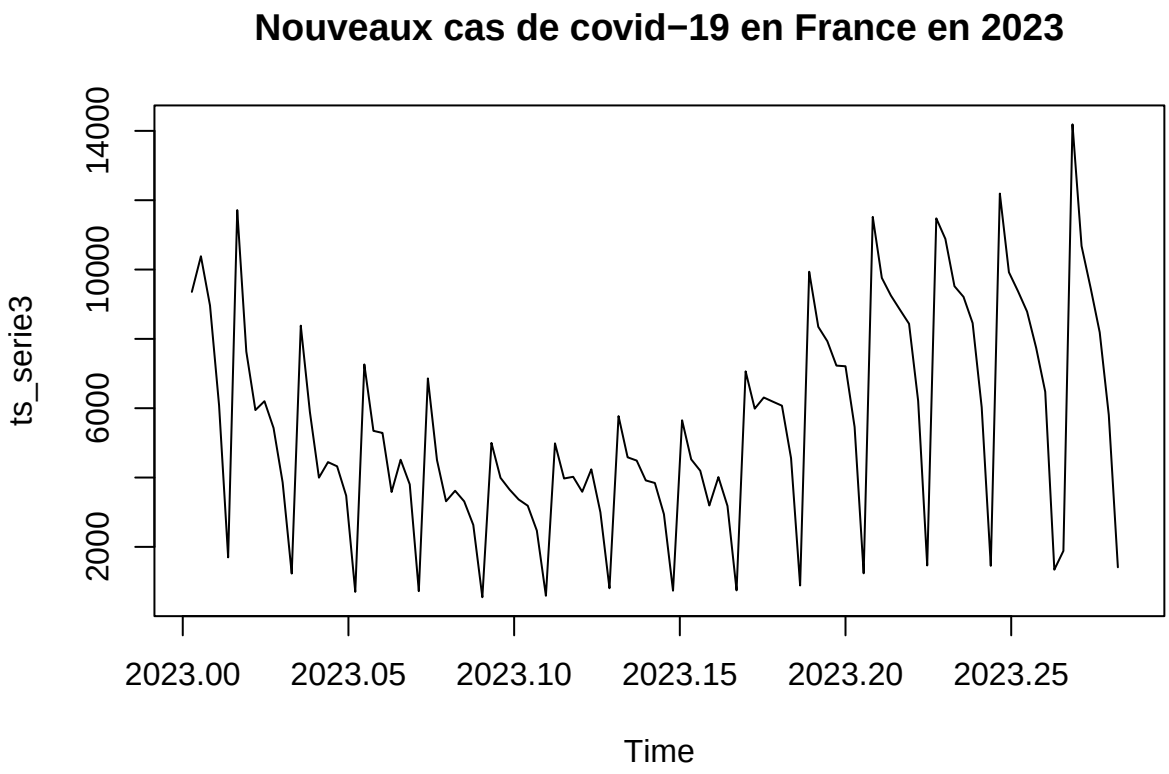
Dans cette sous-série, nous pouvons observer une saisonnalité avec des pics lors de la fin de l'année, pouvant correspondre à la période hivernale mais aussi au niveau des vacances scolaires (vers le mois de Mars). Ce constat est expliqué par les mouvements de foule et le déplacement des populations.

```
plot(ts_serie2,main="Nouveaux cas de covid-19 en France entre 2022 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.

```
plot(ts_serie3,main="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.

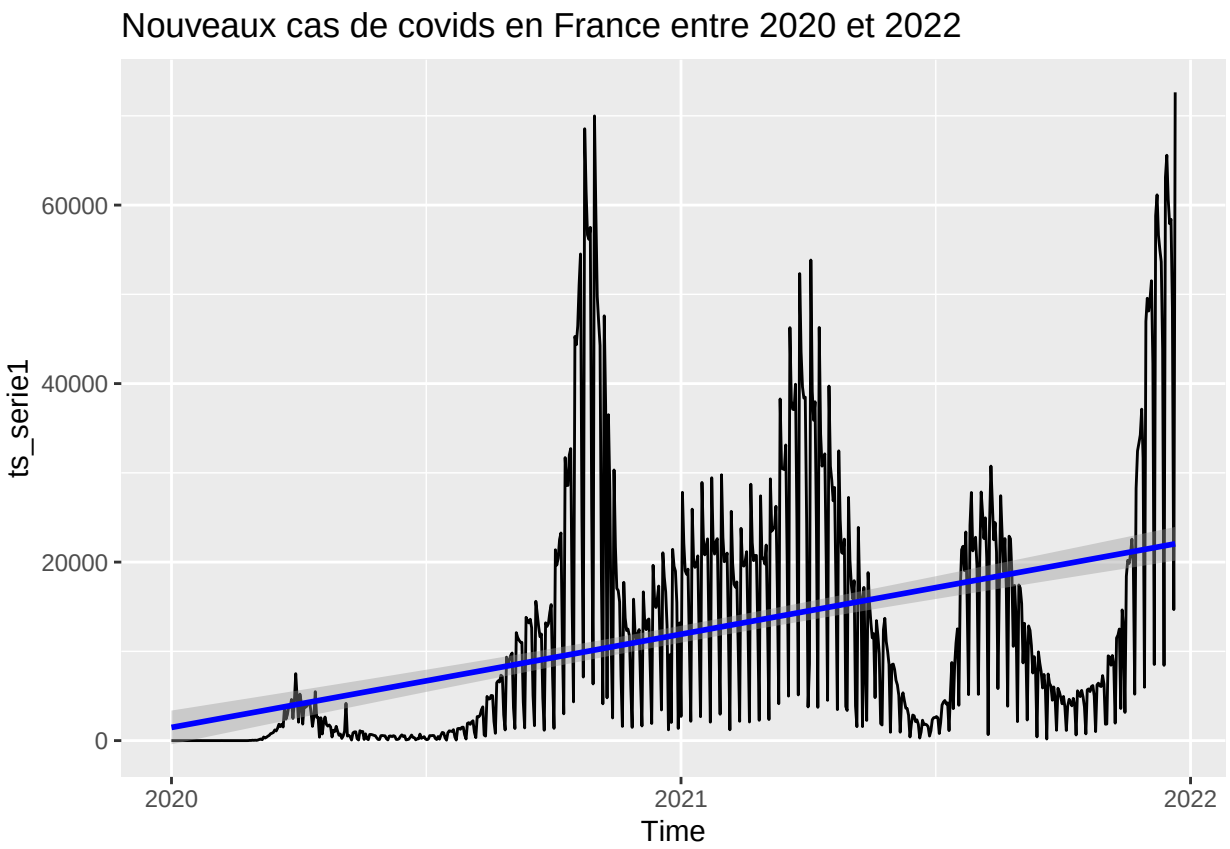
Analyse de la première sous-série

Nous avons décidé de nous focaliser sur la première sous-série.
Ce choix est expliqué grâce à notre connaissance des évènements 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'
```



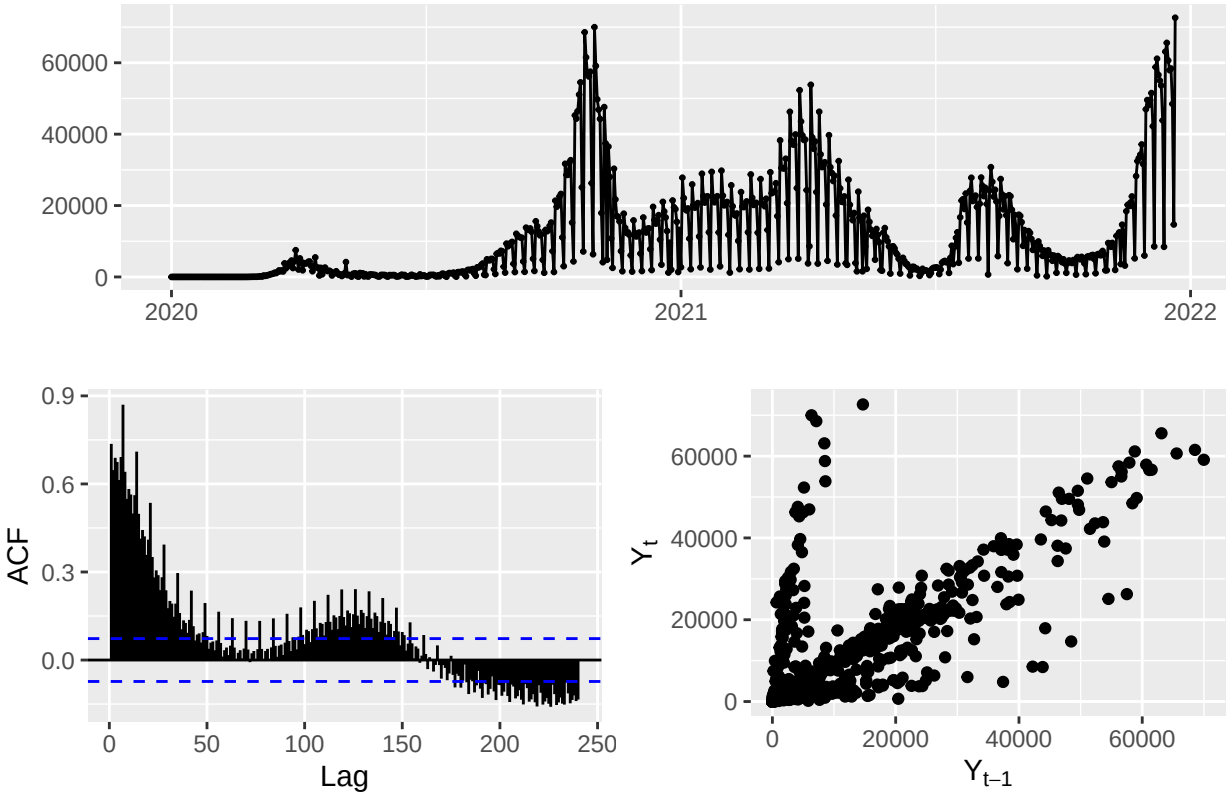
On va d'abord chercher à décrire notre série grâce à des indicateurs descriptifs simples.

```
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.

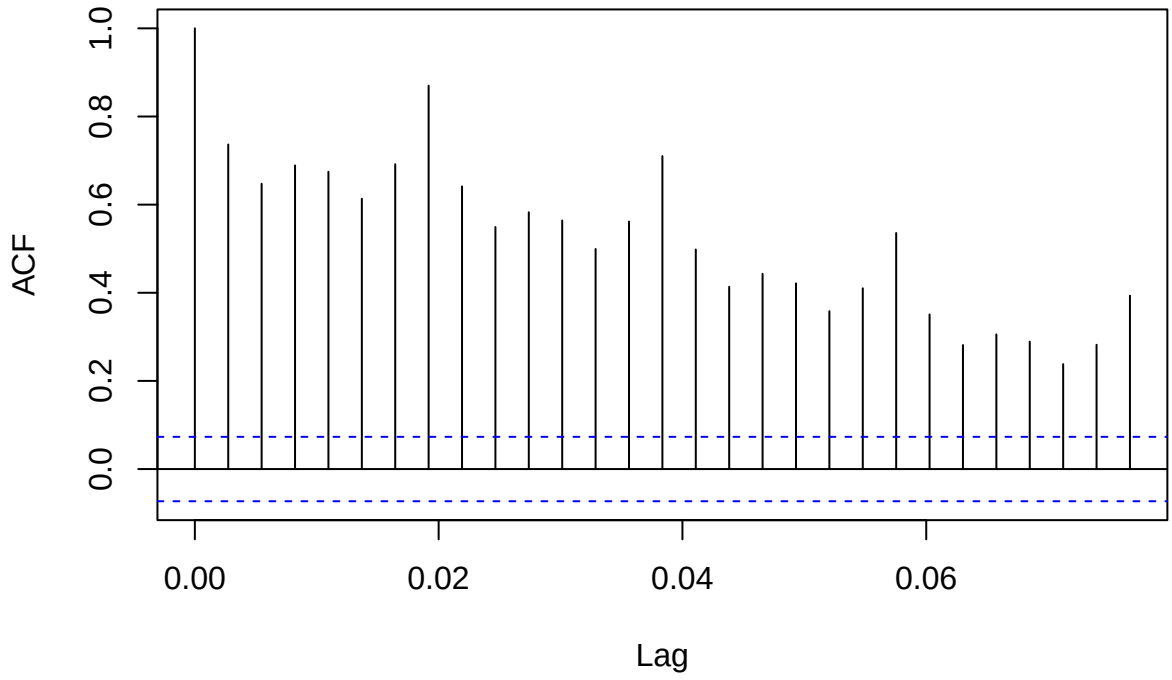
```
ts_serie1 |>
  ggtsdisplay(plot.type = "scatter",smooth=FALSE)
```



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)
```




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

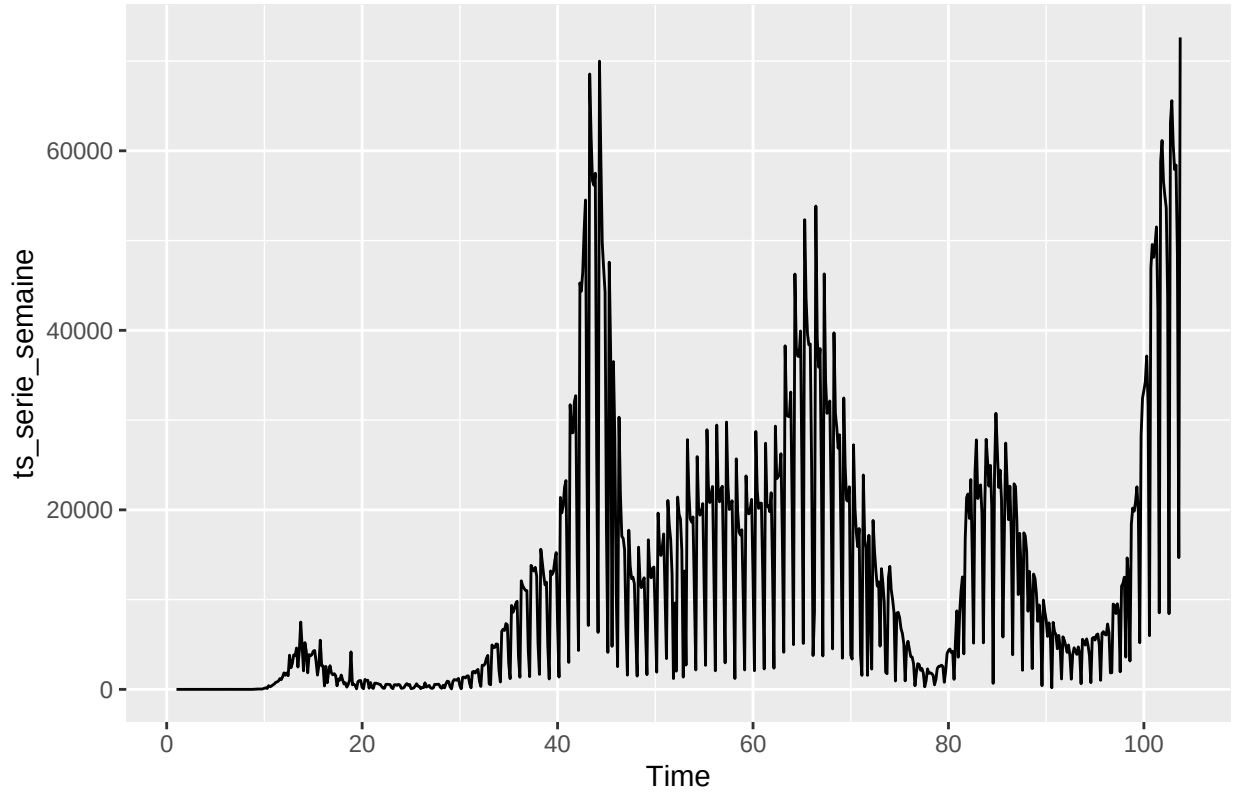
```
##        acf.lag   acf.acf
## 1  0.000000000 1.0000000
## 2  0.002739726 0.7365088
## 3  0.005479452 0.6474386
## 4  0.008219178 0.6890097
## 5  0.010958904 0.6747133
## 6  0.013698630 0.6134778
## 7  0.016438356 0.6918098
## 8  0.019178082 0.8699193
## 9  0.021917808 0.6413418
## 10 0.024657534 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
```

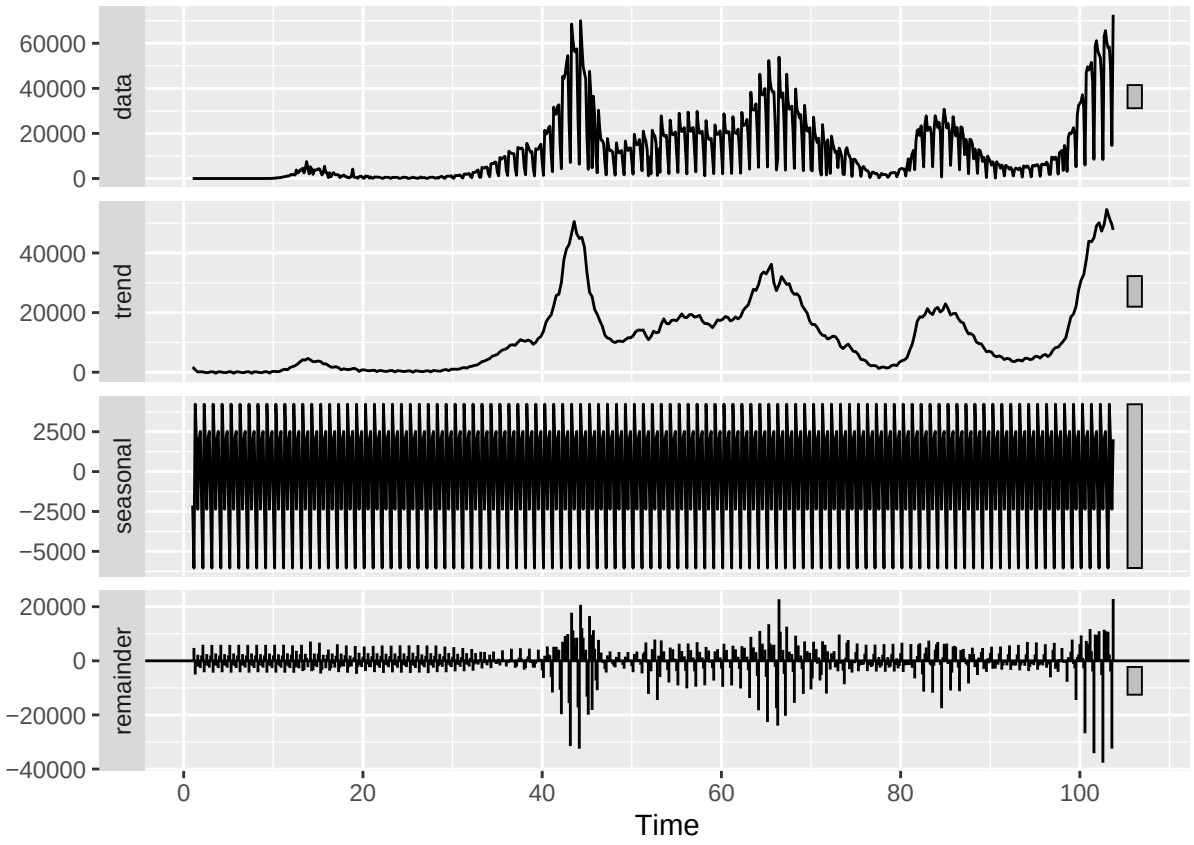
Ce graphique nous permet d'observer une corrélation hebdomadaire. En effet, nous pouvons remarquer un pic plus élevé tous les 7 jours.

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
```




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



Retrait de la tendance / saisonnalité

Nous cherchons à nous ramener à une série sans tendance. Afin de la retirer de notre série, nous allons utiliser l'opérateur diff. Nous allons donc construire un filtre permettant de construire notre analyse.

Tout d'abord, nous avons décider de ne pas transformer notre série en logarithme. Celle-ci nous permettrait de réduire sa variance mais puisque la série présente des valeurs nulles, cette transformation n'est pas pertinente.

Nous allons donc différencier la série.

```
ndiffs(ts_serie_semaine) # On nous conseille de différencier la série 1 fois
```

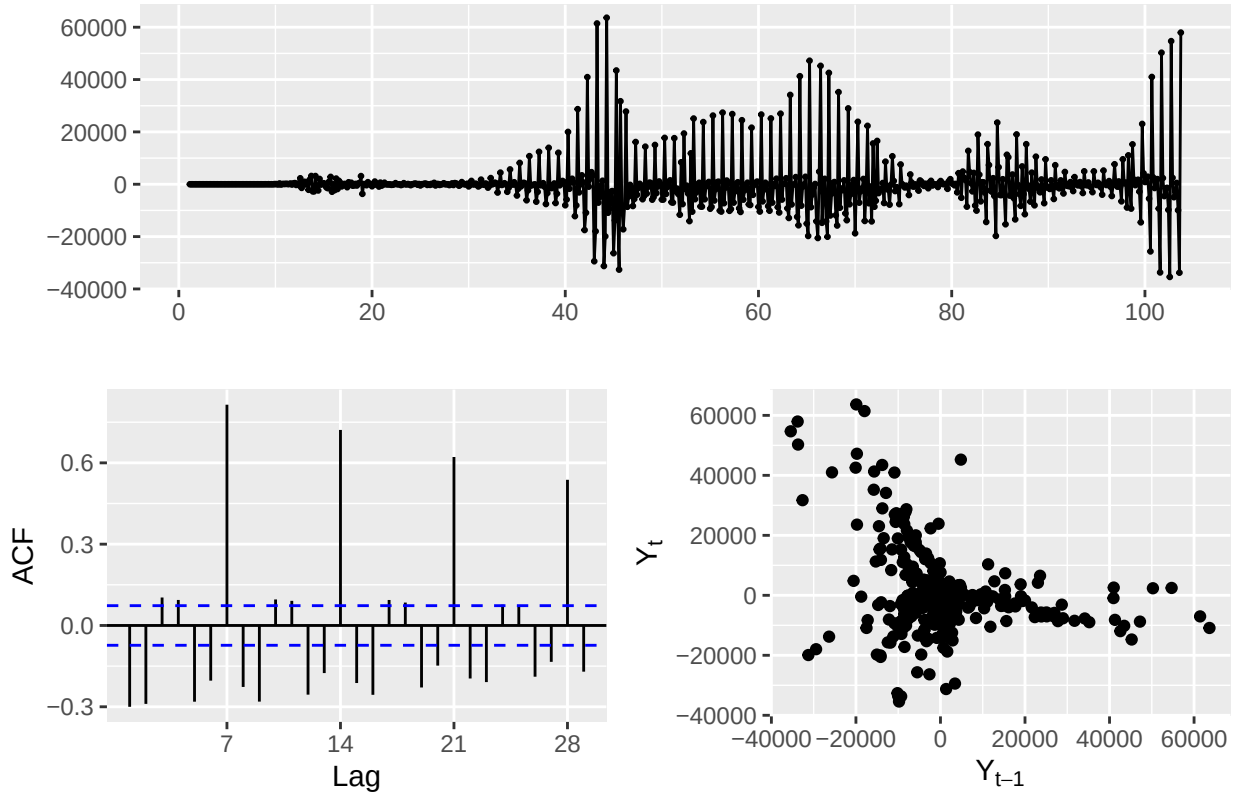
```
## [1] 1
```

```
serie_lissee <- ts_serie_semaine |>
  diff(lag=1)

ndiffs(serie_lissee) # la série a été différencié comme il faut
```

```
## [1] 0
```

Puisque l'on différencie une seule fois, la variance augmente mais reste plus faible que si l'on avait différencié 3 ou 4 fois par exemple.

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



Grâce à cette méthode, nous pouvons faire plusieurs constats :

- **Le chronogramme** : Notre différenciation nous permet de stationariser notre série et supprimer la tendance. Nous pouvons donc émettre l'hypothèse d'un modèle polynomial. Cela nous permet aussi d'écarter l'hypothèse d'un modèle linéaire.
- **L'ACF** : L'auto-corrélation décroit car on fait des moyennes sur de moins en moins de valeurs. De plus, nous pouvons observer une saisonnalité hebdomadaire avec un pic tous les 7 jours (lag).
- **Corrélation** : Le nuage de points ne présente plus de direction, il n'existe donc plus de réelle corrélation.

Stationnarité

Ensuite, nous allons tester la stationnarité de notre série :

```
kpss.test(serie_lissee)
```

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  serie_lissee
## KPSS Level = 0.26302, Truncation lag parameter = 6, p-value = 0.1
```

Nous avons une p-value associée au test supérieure à 5%, nous rejettons donc l'hypothèse nulle de stationnarité. Notre série n'est donc pas stationnaire.

```
adf.test(serie_lissee)
```

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

##
##  Augmented Dickey-Fuller Test
##
## data:  serie_lissee
## Dickey-Fuller = -6.9851, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

Lors de ce test, nous obtenons une p-value inférieure à 5%. Nous acceptons l'hypothèse de non-stationnarité. Ce test nous permet de confirmer notre hypothèse précédente.

En conclusion, notre série n'est pas **stationnaire**. Cette conclusion nous permet de confirmer que la mise en place d'un modèle linéaire n'est pas une solution pour modéliser notre série. Notre série n'étant pas stationnaire, nos différents seront moins fiables et nos prévisions moins précises.

Modélisation de notre série

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

Afin de pouvoir l'estimer, nous avons utiliser 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 selectionner le meilleur modèle sur la base du critère AIC.

Modèle ARIMA

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

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2)(1,1,1)[7] with drift         : 13873.67
##  ARIMA(0,0,0)(0,1,0)[7] with drift         : 14203.17
##  ARIMA(1,0,0)(1,1,0)[7] with drift         : 14005.55
##  ARIMA(0,0,1)(0,1,1)[7] with drift         : 14083.84
##  ARIMA(0,0,0)(0,1,0)[7]                    : 14207.64
##  ARIMA(2,0,2)(0,1,1)[7] with drift         : Inf
##  ARIMA(2,0,2)(1,1,0)[7] with drift         : Inf
##  ARIMA(2,0,2)(2,1,1)[7] with drift         : Inf
##  ARIMA(2,0,2)(1,1,2)[7] with drift         : 13875.34
##  ARIMA(2,0,2)(0,1,0)[7] with drift         : 13881.88
##  ARIMA(2,0,2)(0,1,2)[7] with drift         : 13866.78
##  ARIMA(1,0,2)(0,1,2)[7] with drift         : 13863.23
##  ARIMA(1,0,2)(0,1,1)[7] with drift         : 13861.31
##  ARIMA(1,0,2)(0,1,0)[7] with drift         : 13881.46
##  ARIMA(1,0,2)(1,1,1)[7] with drift         : 13870.03
##  ARIMA(1,0,2)(1,1,0)[7] with drift         : 13868.03
##  ARIMA(1,0,2)(1,1,2)[7] with drift         : 13871.76
##  ARIMA(0,0,2)(0,1,1)[7] with drift         : 14037.8
##  ARIMA(1,0,1)(0,1,1)[7] with drift         : 13859.87
##  ARIMA(1,0,1)(0,1,0)[7] with drift         : 13881.94
##  ARIMA(1,0,1)(1,1,1)[7] with drift         : 13868.72
##  ARIMA(1,0,1)(0,1,2)[7] with drift         : 13861.8
##  ARIMA(1,0,1)(1,1,0)[7] with drift         : 13866.71
##  ARIMA(1,0,1)(1,1,2)[7] with drift         : 13870.36
```

```
##  ARIMA(1,0,0)(0,1,1)[7] with drift         : 14000.4
##  ARIMA(2,0,1)(0,1,1)[7] with drift         : 13862.43
##  ARIMA(0,0,0)(0,1,1)[7] with drift         : 14166.62
##  ARIMA(2,0,0)(0,1,1)[7] with drift         : 13930.59
##  ARIMA(1,0,1)(0,1,1)[7]                    : 13858.35
##  ARIMA(1,0,1)(0,1,0)[7]                    : 13880.46
##  ARIMA(1,0,1)(1,1,1)[7]                    : 13867.17
##  ARIMA(1,0,1)(0,1,2)[7]                    : 13860.3
##  ARIMA(1,0,1)(1,1,0)[7]                    : 13865.18
##  ARIMA(1,0,1)(1,1,2)[7]                    : 13868.86
##  ARIMA(0,0,1)(0,1,1)[7]                    : 14085.88
##  ARIMA(1,0,0)(0,1,1)[7]                    : 14000.64
##  ARIMA(2,0,1)(0,1,1)[7]                    : 13860.94
##  ARIMA(1,0,2)(0,1,1)[7]                    : 13859.81
##  ARIMA(0,0,0)(0,1,1)[7]                    : 14169.47
##  ARIMA(0,0,2)(0,1,1)[7]                    : 14039.22
##  ARIMA(2,0,0)(0,1,1)[7]                    : 13929.86
##  ARIMA(2,0,2)(0,1,1)[7]                    : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(1,0,1)(0,1,1)[7]                    : 13979.11
##
##  Best model: ARIMA(1,0,1)(0,1,1)[7]
```

```
model_arima
```

```
## Series: ts_serie_semaine
## ARIMA(1,0,1)(0,1,1)[7]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.9539  -0.6672  -0.1902
## s.e.  0.0136   0.0298   0.0384
##
## sigma^2 = 1.9e+07:  log likelihood = -6985.55
## AIC=13979.11   AICc=13979.16   BIC=13997.38
```

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

```
summary(model_arima)
```

```
## Series: ts_serie_semaine
## ARIMA(1,0,1)(0,1,1)[7]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.9539  -0.6672  -0.1902
## s.e.  0.0136   0.0298   0.0384
##
## sigma^2 = 1.9e+07:  log likelihood = -6985.55
## AIC=13979.11   AICc=13979.16   BIC=13997.38
##
## Training set error measures:
##                    ME     RMSE      MAE MPE MAPE      MASE        ACF1
## Training set 115 4328.228 2169.786 NaN  Inf 0.7303148 -0.01758392
```

```
t_stat(model_arima)
```

```
##             ar1      ma1      sma1
## t.stat 70.32788 -22.4075 -4.947629
## p.val   0.00000   0.0000  0.000001
```

Le modèle n'est pas simplifiable.

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)
```

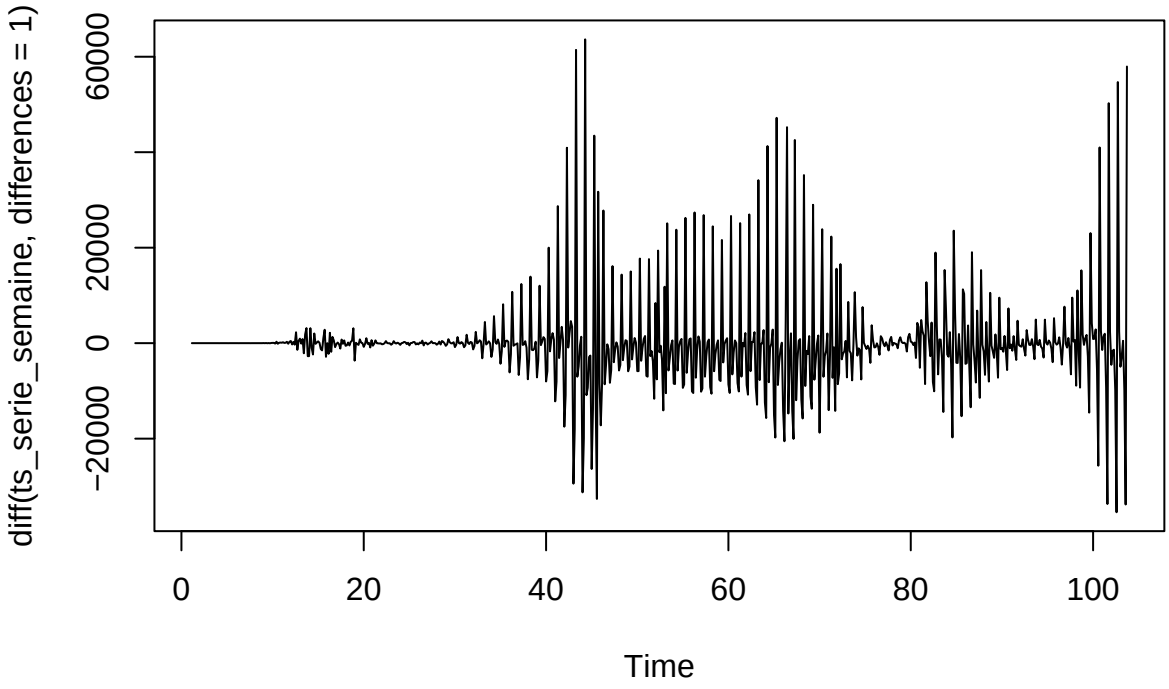
```
##
##  Shapiro-Wilk normality test
##
## data:  ts_serie_semaine
## W = 0.79257, p-value < 2.2e-16
```

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

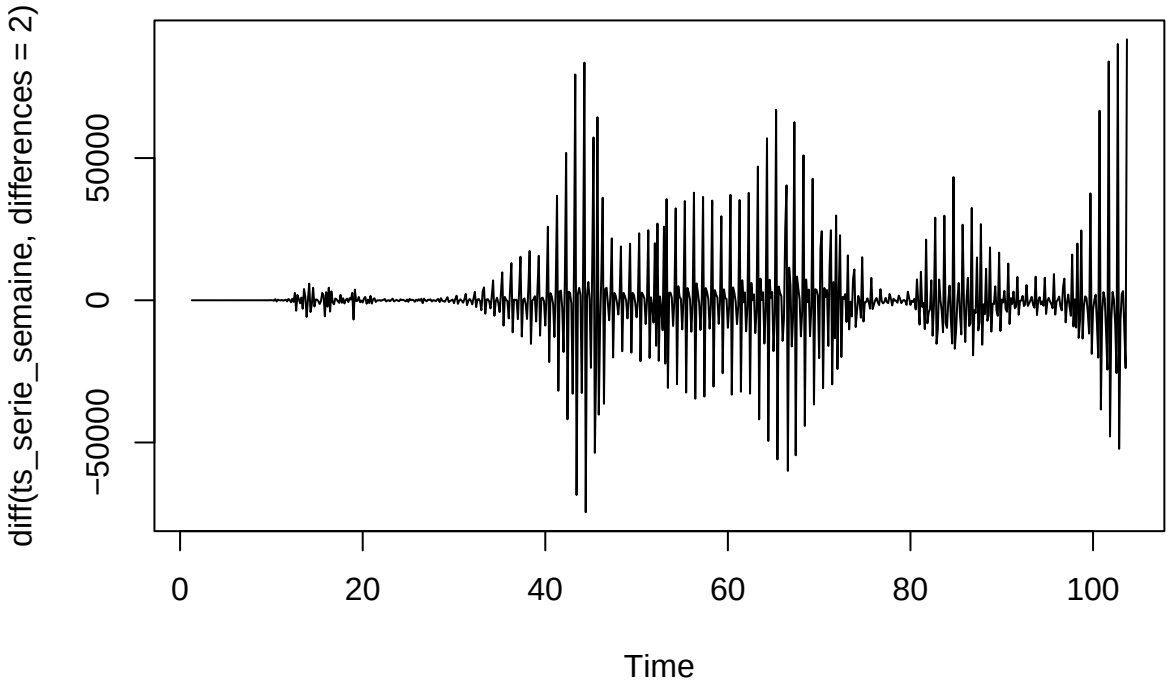
Nous pouvons alors construire notre modèle polynomial.

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.

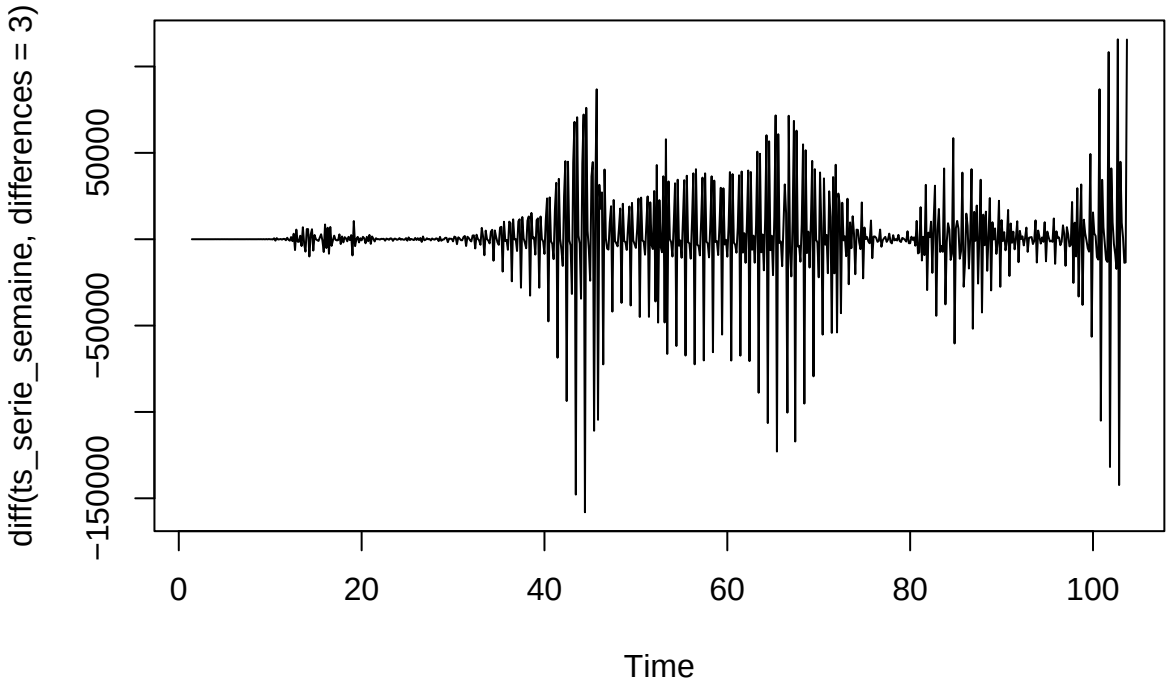
```
plot(diff(ts_serie_semaine, differences = 1),type="l")
```



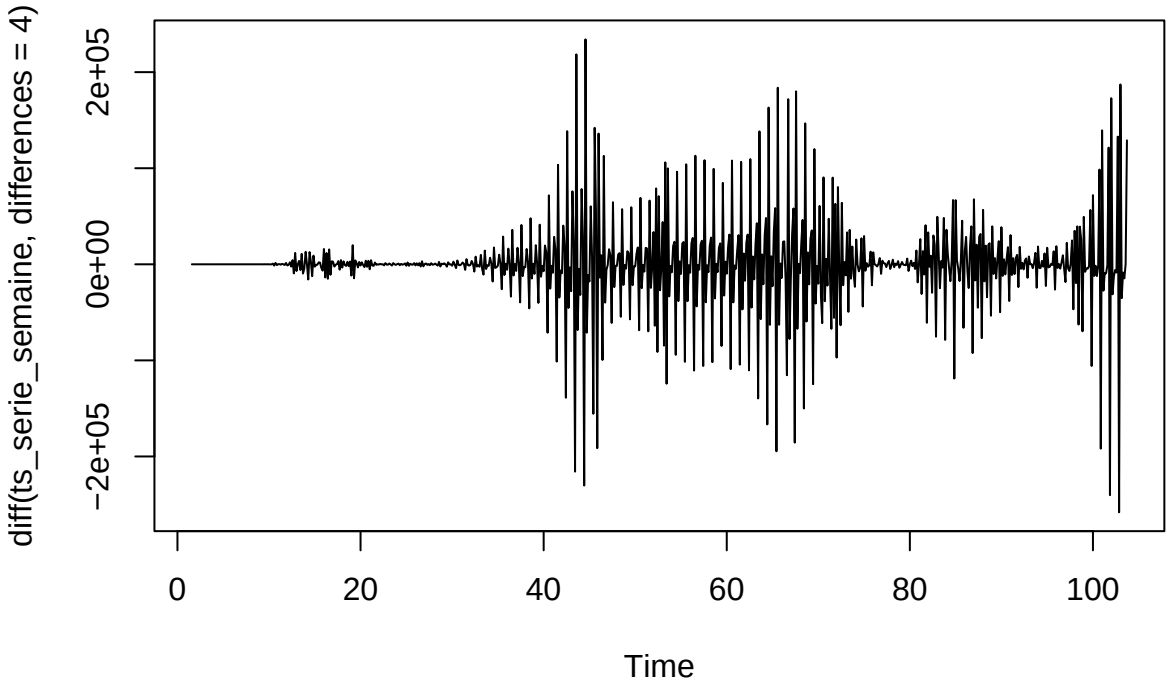

```
plot(diff(ts_serie_semaine, differences = 2),type="l")
```



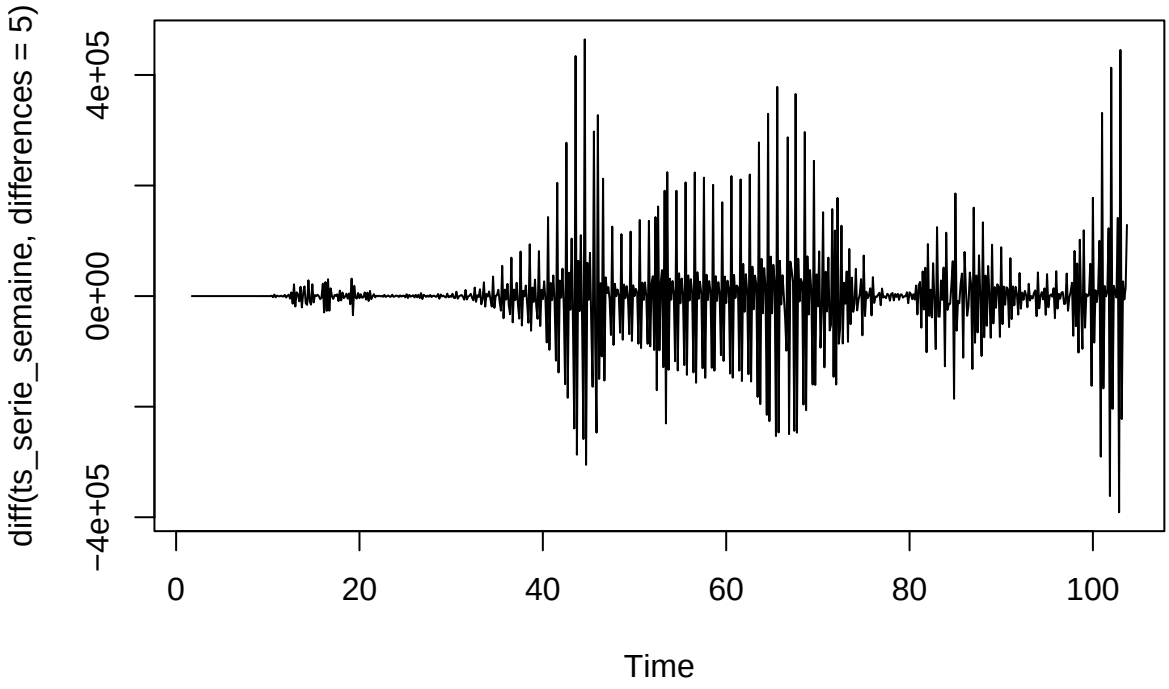

```
plot(diff(ts_serie_semaine, differences = 3),type="l")
```




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




```
plot(diff(ts_serie_semaine, differences = 5),type="l")
```



Le degré d de la tendance polynomiale est 3. On a un graphique avec d = 3qui est à peu près centré donc on choisit d-1 = 2.
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.

```
model <- lm(ts_serie_semaine ~ poly(ts_serie_semaine, 2, raw=TRUE))
summary(model)
```

```
## Warning in summary.lm(model): ajustement pratiquement parfait : le résumé n'est
## peut-être pas fiable
```

```
##
## Call:
## lm(formula = ts_serie_semaine ~ poly(ts_serie_semaine, 2, raw = TRUE))
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.162e-10  5.000e-14  5.100e-13  1.440e-12  1.502e-11
##
## Coefficients:
##                                       Estimate Std. Error    t value
## (Intercept)                         -1.302e-11  8.803e-13 -1.479e+01
## poly(ts_serie_semaine, 2, raw = TRUE)1  1.000e+00  1.072e-16  9.329e+15
## poly(ts_serie_semaine, 2, raw = TRUE)2 -1.792e-21  2.093e-21 -8.560e-01
##                                       Pr(>|t|)
## (Intercept)                             <2e-16 ***
```

```
## poly(ts_serie_semaine, 2, raw = TRUE)1   <2e-16 ***
## poly(ts_serie_semaine, 2, raw = TRUE)2    0.392
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.56e-11 on 717 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.988e+32 on 2 and 717 DF,  p-value: < 2.2e-16
```

Nous devons donc avoir 3 régresseurs pour la tendance et 6 régresseurs pour la saisonnalité. Il faut génerer 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(ts_serie_semaine),ncol=9) # car 9 régresseurs au total
# t <- 1:nrow(ts_serie_semaine)
# x[,2] <- t
# x[,3] <- t**2
# x[,5] <- cos((2*pi*t)/7)
# x[,6] <- cos((4*pi*t**2)/7)
# x[,7] <- sin((2*pi*t)/7)
# x[,8] <- sin((4*pi*t**2)/7)

ts_serie1_lm <- lm(data=df,ts_serie1~.)
```

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.

Comparaison des modèles

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

```
## [1]  13979.11 -33774.34
```

```
c(BIC(model_arima),BIC(ts_serie1_lm))
```

```
## [1]  13997.38 -33705.65
```

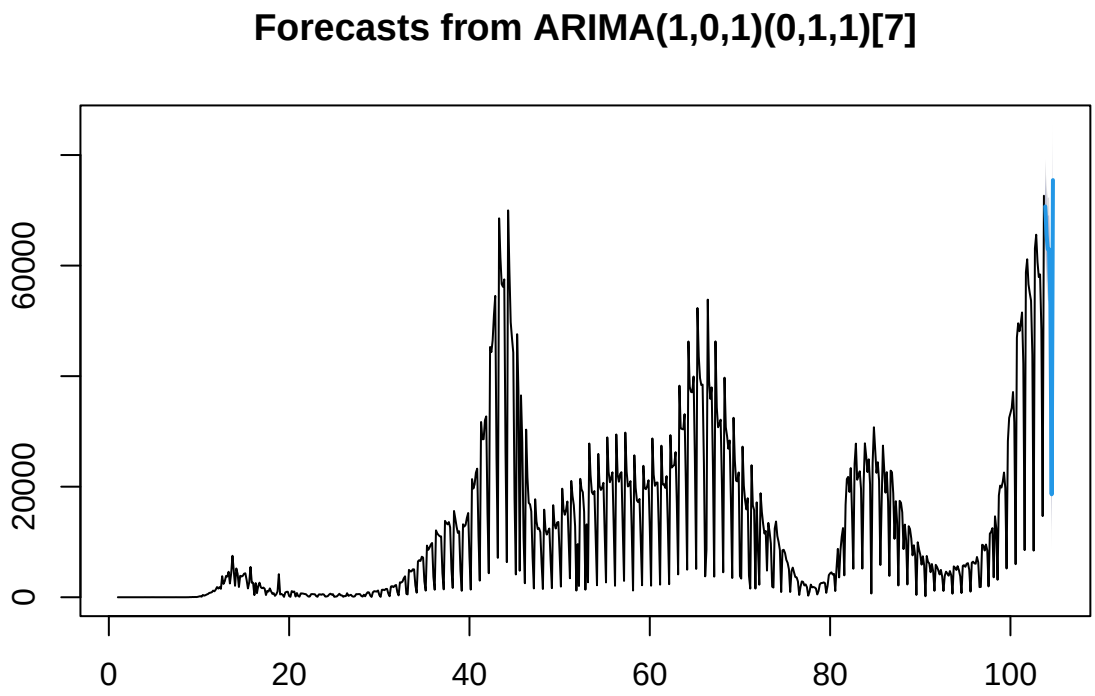
Nous aurons tendance à privilégier le modèle ARIMA puisque les critères de l'AIC et le BIC sont minimisés.

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.

Prévisions

Suite à notre analyse, nous avons utilisé la fonction forecast pour émettre des prévisions sur notre série pour les 7 prochains jours.

```
forecast_cases <- forecast::forecast(model_arima, h = 7)
plot(forecast_cases)
```



Les prédictions :

```
predict(model_arima)
```

```
## $pred
## Time Series:
## Start = c(103, 7)
## End = c(103, 7)
## Frequency = 7
## [1] 70698.91
##
## $se
## Time Series:
## Start = c(103, 7)
## End = c(103, 7)
## Frequency = 7
## [1] 4358.602
```

Nos prévisions ne sont pas très précises puisque notre série n'est pas stationnaire.

Conclusion