Project Timeseries

2022-11-20

Bike sharing Data set Analysis:

Les systèmes de vélos en libre-service sont une nouvelle génération de locations traditionnelles de vélos où l'ensemble du processus, de l'adhésion à la location et au retour, est devenu automatique. Grâce à ces systèmes, l'utilisateur peut facilement louer un vélo à un endroit précis et le rendre à un autre endroit. Actuellement, il existe plus de 500 programmes de vélos en libre-service dans le monde, qui comptent plus de 500 000 vélos. Aujourd'hui, ces systèmes suscitent un grand intérêt en raison de leur rôle important dans les problèmes de circulation, d'environnement et de santé.

Outre les applications intéressantes des systèmes de partage de vélos dans le monde réel, les caractéristiques des données générées par ces systèmes les rendent attrayantes pour la recherche. Contrairement à d'autres services de transport comme le bus ou le métro, la durée du trajet, la position de départ et d'arrivée sont explicitement enregistrées dans ces systèmes. Cette caractéristique transforme le système de vélo en libre-service en un réseau de capteurs virtuel qui peut être utilisé pour détecter la mobilité dans la ville. On s'attend donc à ce que la plupart des événements importants de la ville puissent être détectés par le biais de ces données.

Ce jeu de données contient le nombre de vélos de location par heure et par jour entre 2011 et 2012 dans le système de partage de vélos Capital à Washington, DC, avec les informations météorologiques et saisonnières correspondantes.

```
if(!require(tidyverse)){
    install.packages(tidyverse)
}
```

```
## Le chargement a nécessité le package : tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyverse)
library(forecast)
```

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

```r
library(forecast)
library("TTR")
library(tseries)
library(fpp)
```

```
## Le chargement a nécessité le package : fma

## Le chargement a nécessité le package : expsmooth

## Le chargement a nécessité le package : lmtest

## Le chargement a nécessité le package : zoo

##
## Attachement du package : 'zoo'

## Les objets suivants sont masqués depuis 'package:base':
##
##     as.Date, as.Date.numeric
```

Lecture des datas:

```r
day <- read.csv("day.csv")
hour <- read.csv("hour.csv")
```

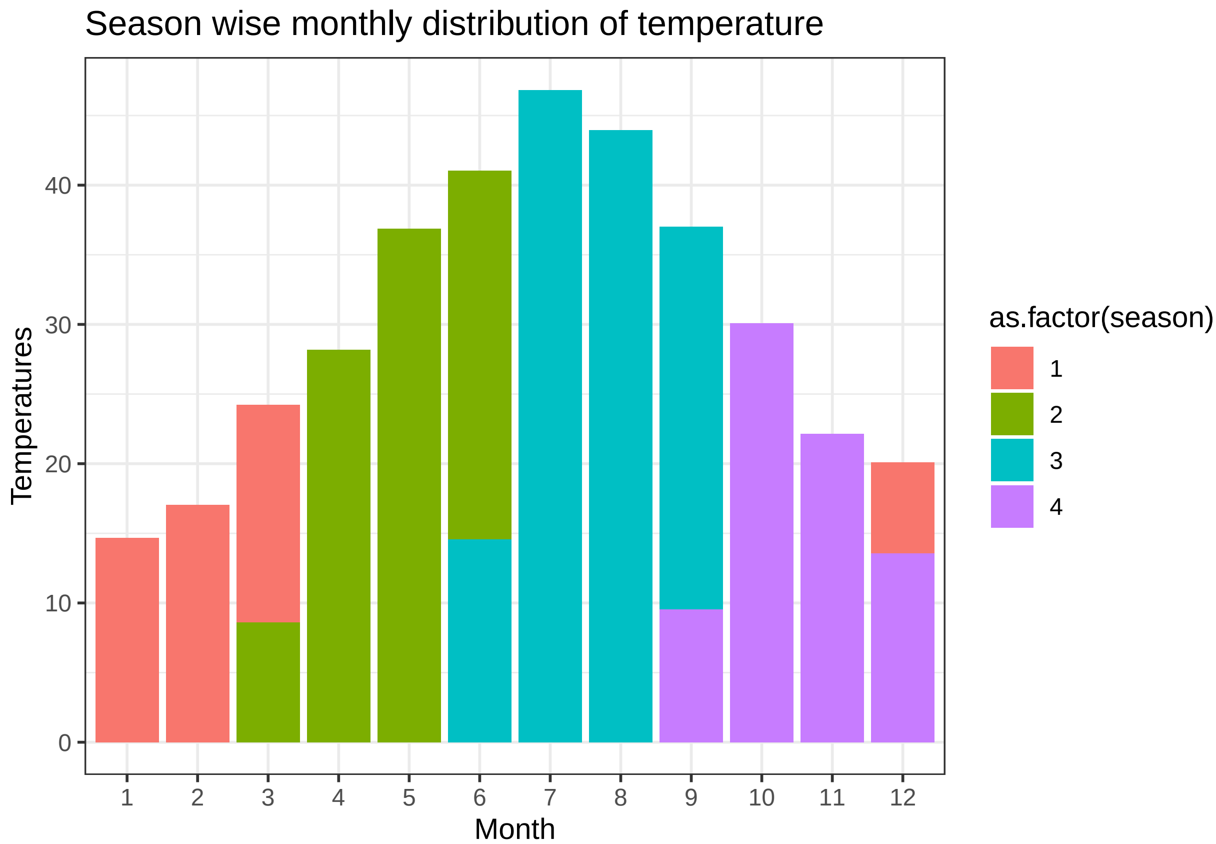
```r
head(day)
```

```
##   instant     dteday season yr mnth holiday weekday workingday weathersit
## 1       1 2011-01-01      1  0    1       0       6          0          2
## 2       2 2011-01-02      1  0    1       0       0          0          2
## 3       3 2011-01-03      1  0    1       0       1          1          1
## 4       4 2011-01-04      1  0    1       0       2          1          1
## 5       5 2011-01-05      1  0    1       0       3          1          1
## 6       6 2011-01-06      1  0    1       0       4          1          1
##       temp    atemp      hum windspeed casual registered  cnt
## 1 0.344167 0.363625 0.805833 0.1604460    331        654  985
## 2 0.363478 0.353739 0.696087 0.2485390    131        670  801
## 3 0.196364 0.189405 0.437273 0.2483090    120       1229 1349
## 4 0.200000 0.212122 0.590435 0.1602960    108       1454 1562
## 5 0.226957 0.229270 0.436957 0.1869000     82       1518 1600
## 6 0.204348 0.233209 0.518261 0.0895652     88       1518 1606
```

1)a)

Distribution de temperatures en fonction des saisons:

```r
ggplot(day,aes(x=as.factor(mnth),y=temp,fill=as.factor(season)))+theme_bw()+geom_col()+
labs(x='Month',y='Temperatures',title='Season wise monthly distribution of temperature')
```

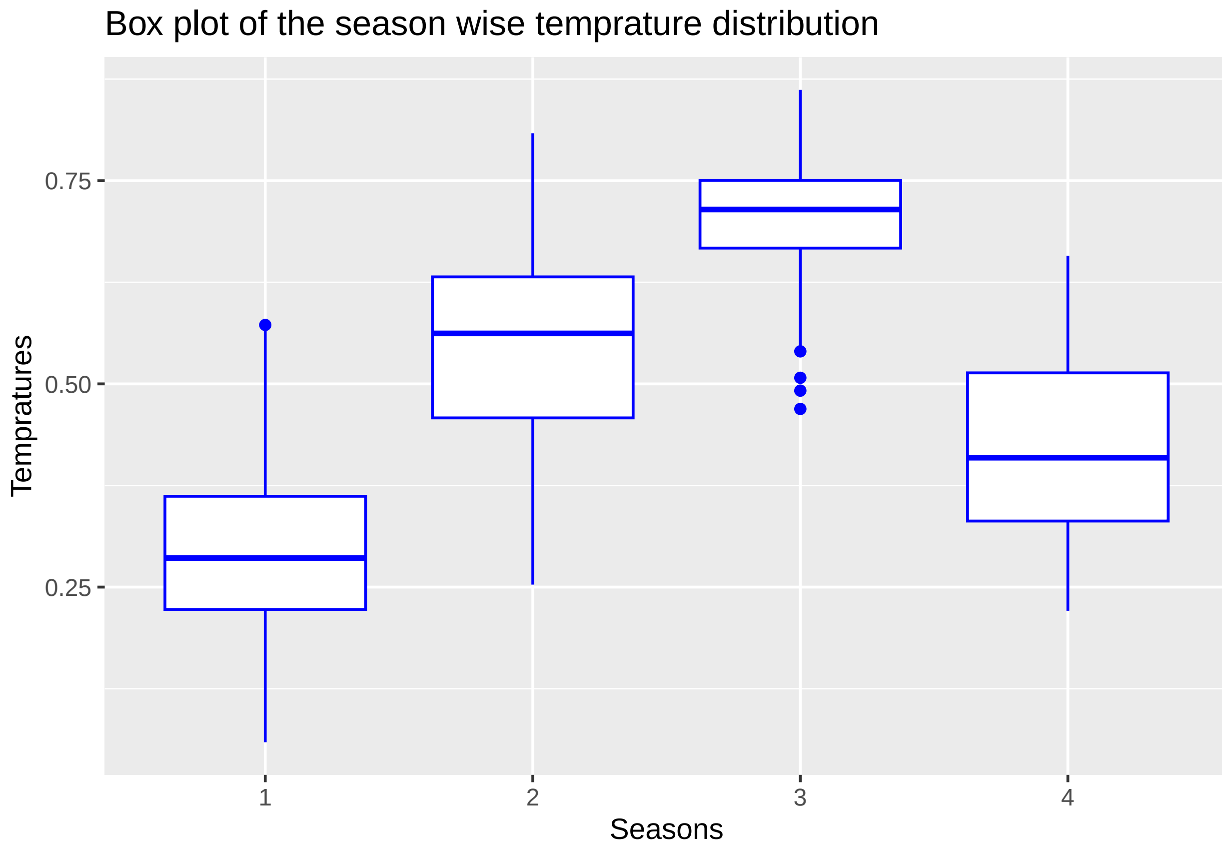



```
#TODO: Change labels (especially as factor of seasons)
```

On voit que les temperatures augmentent jusqu'a la moitie de l'annee puis redescendent ensuite. Par ailleurs, on remarque que les temperatures sont les plus eleves pendant la saison 3(c'est a dire l'été), et baissent pendant la saison 4 (automne) jusqu'a devenir minimales en saison 1 (hivers).

On peut aussi visualiser avec des boxplots:

```
ggplot(day, aes(x=as.factor(season),y=temp)) +
  geom_boxplot(color = "blue") +
  labs(x='Seasons',y='Tempratures',title='Box plot of the season wise temprature distribution')
```



On remarque d'il y a des outliers en saison 1(hiver) et 3(ete),avec des valeurs qui sont respectivement plus elevees que la normale et plus basses.

Moyenne des temperatures:

```
mean(day$temp)
```

```
## [1] 0.4953848
```

Mediane des temperatures:

```
median(day$temp)
```

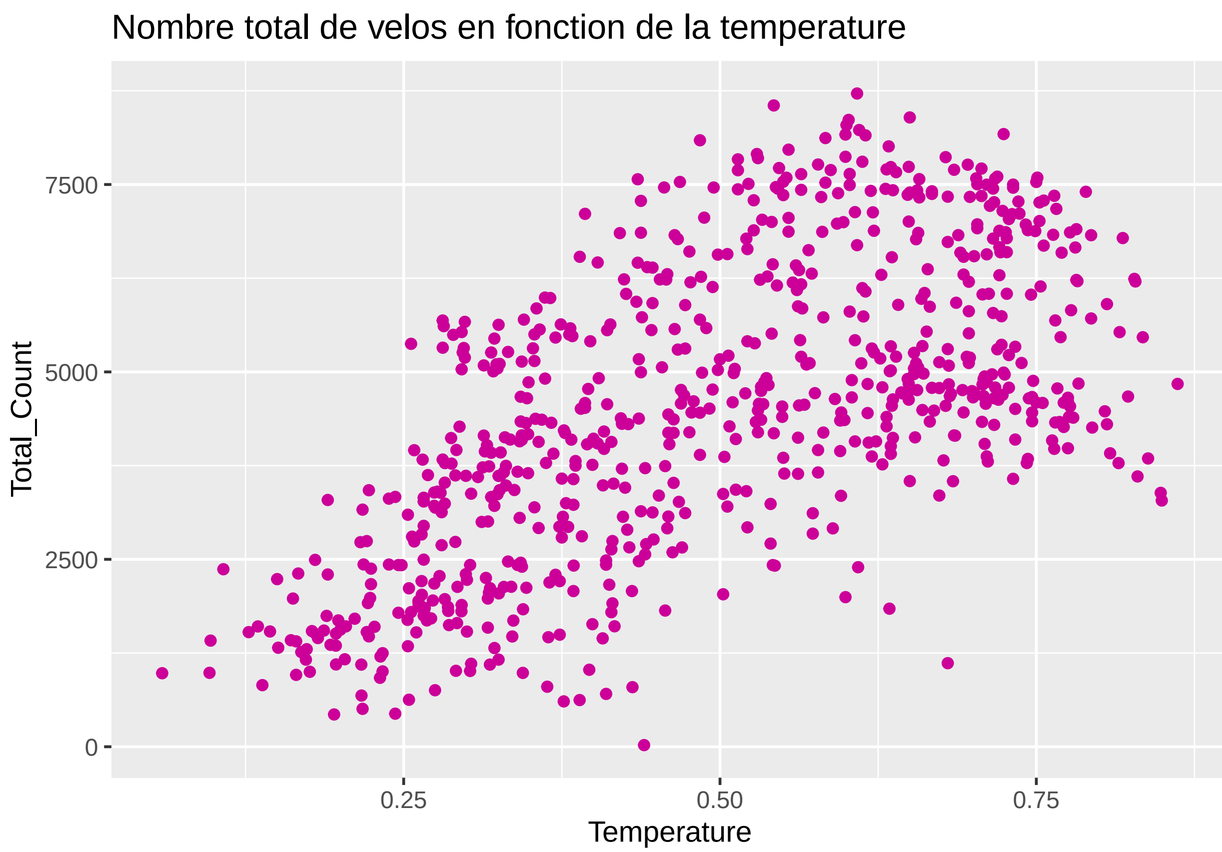
```
## [1] 0.498333
```

On a une temperature moyenne de 0.49538 et une mediane de 0.49833.

1)b)

Correlation entre la temperature temp et le nombre total de location de vélos cnt:

```
ggplot(data = day) +
  geom_point(mapping = aes(x = temp, y = cnt), color = "#CC0099") +
  labs(x='Temperature',y='Total_Count',title='Nombre total de velos en fonction de la temperature')
```



Les points forment une figure etiree. On peut envisager une correlation lineaire entre les deux variables. Pour verifier cela, on calcule la correlation:

```
cor.test(day$temp, day$cnt, method=c("pearson", "kendall", "spearman"))
```

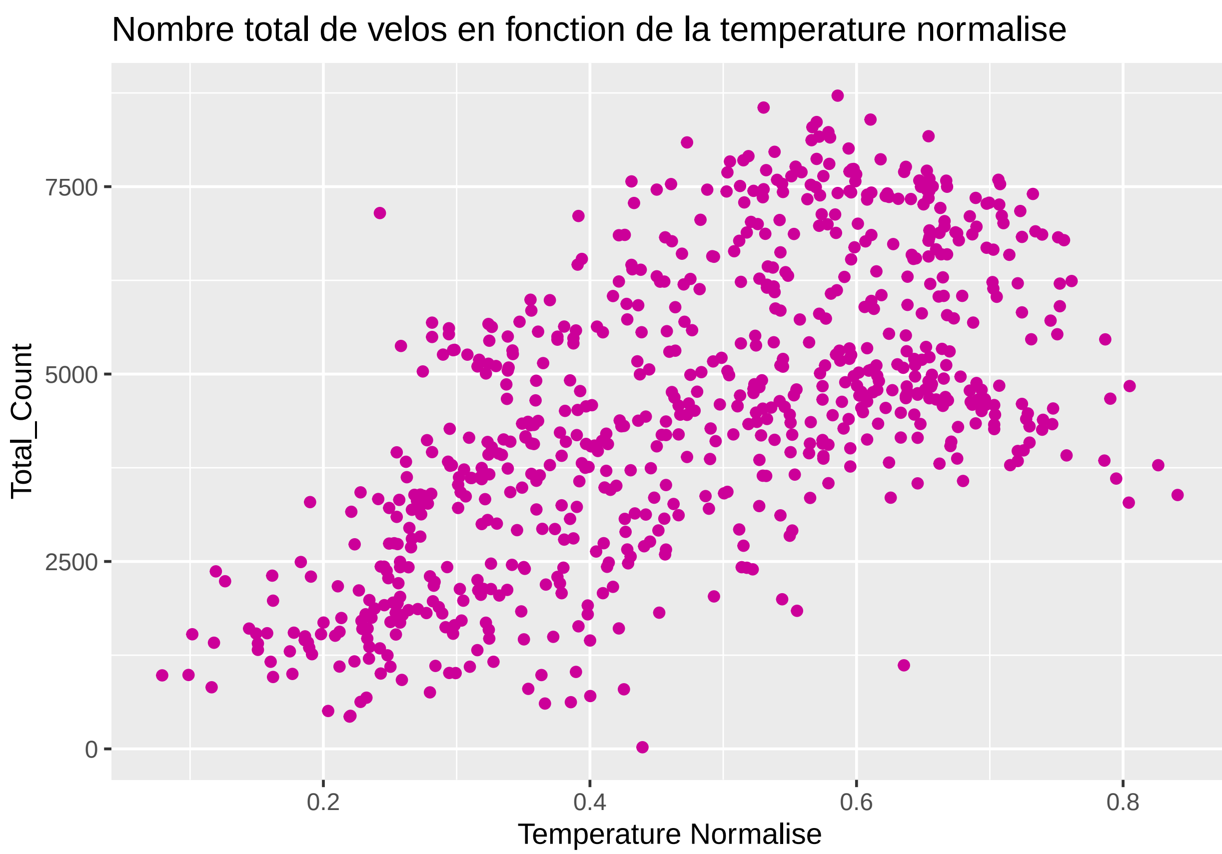
```
##
##  Pearson's product-moment correlation
##
## data:  day$temp and day$cnt
## t = 21.759, df = 729, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5814369 0.6695422
## sample estimates:
##       cor
## 0.627494
```

Le coefficient de correlation est de 0.627, ce qui n'est pas extrêmement élevé, de plus, la p-value est inférieure a 5%, ce qui veut dire que la correlation est statistiquement significative.

On fait de meme pour les variables atemp et cnt:

Correlation entre la temperature ressentie atemp et le nombre total de location de vélos cnt:

```
ggplot(data = day) +
  geom_point(mapping = aes(x = atemp, y = cnt), color = "#CC0099") +
  labs(x='Temperature Normalise',y='Total_Count',title='Nombre total de velos en fonction de la tempera
```




```
cor.test(day$atemp, day$cnt, method=c("pearson", "kendall", "spearman"))
```

```
##
##  Pearson's product-moment correlation
##
## data:  day$atemp and day$cnt
## t = 21.965, df = 729, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5853376 0.6727918
## sample estimates:
##       cor
## 0.6310657
```

On constate la meme chose pour la variable atemp, il n'y a pas une grande differance entre la relation cnt-temp et cnt-atemp.

Correlation entre la moyenne des temperatures réelles et ressenties (temp et atemp) et le nombre total de location de vélos cnt:

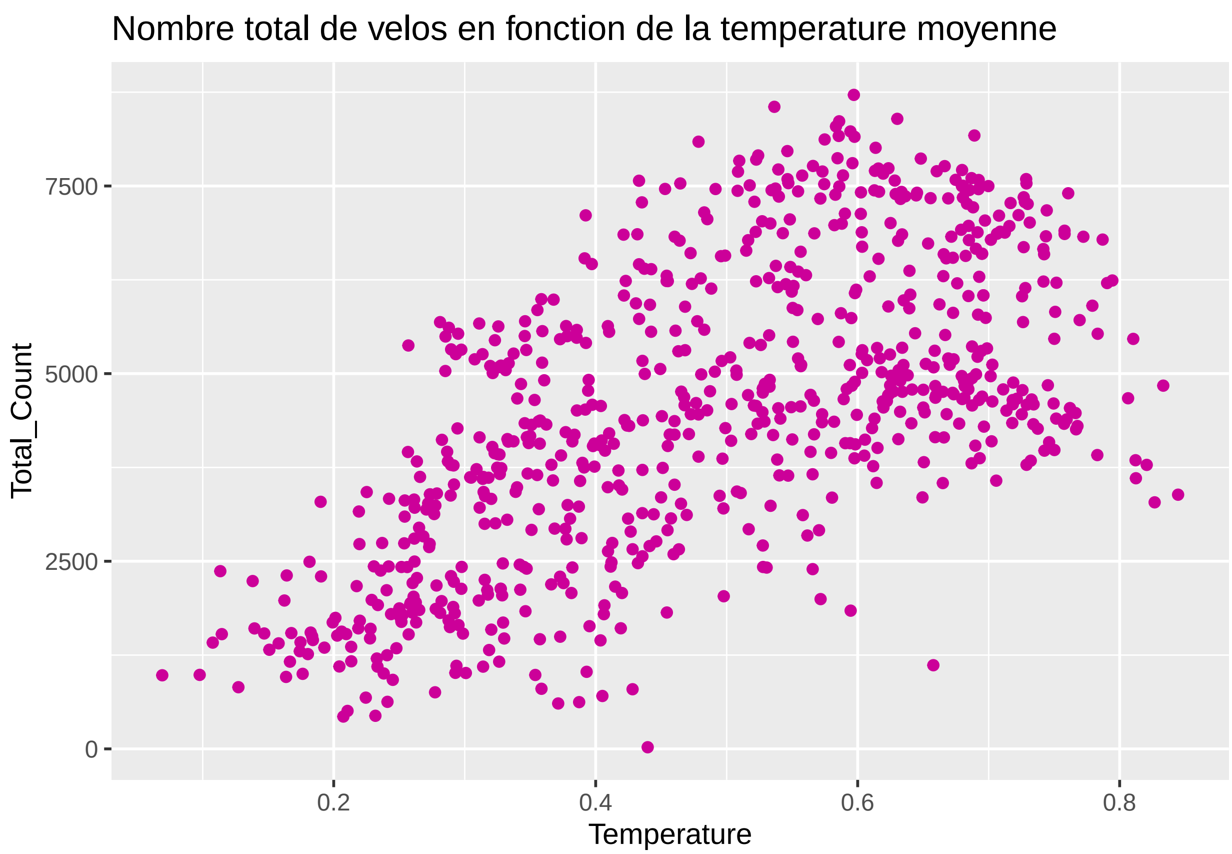
Pour la correlation entre cnt et mean(temp, atemp), on rajoute une colonne qui represente la moyenne entre temp et atemp, puis on repete les memes operations:

```
day$"mean_temp_atemp" <- apply(day[, 10:11], 1, mean)
head(day)
```

```
##   instant     dteday season yr mnth holiday weekday workingday weathersit
## 1       1 2011-01-01      1  0    1       0       6          0          2
## 2       2 2011-01-02      1  0    1       0       0          0          2
## 3       3 2011-01-03      1  0    1       0       1          1          1
## 4       4 2011-01-04      1  0    1       0       2          1          1
## 5       5 2011-01-05      1  0    1       0       3          1          1
## 6       6 2011-01-06      1  0    1       0       4          1          1
##       temp    atemp      hum windspeed casual registered  cnt mean_temp_atemp
## 1 0.344167 0.363625 0.805833 0.1604460    331        654  985       0.3538960
## 2 0.363478 0.353739 0.696087 0.2485390    131        670  801       0.3586085
## 3 0.196364 0.189405 0.437273 0.2483090    120       1229 1349       0.1928845
## 4 0.200000 0.212122 0.590435 0.1602960    108       1454 1562       0.2060610
## 5 0.226957 0.229270 0.436957 0.1869000     82       1518 1600       0.2281135
## 6 0.204348 0.233209 0.518261 0.0895652     88       1518 1606       0.2187785
```

On affiche les données:

```
ggplot(data = day) +
  geom_point(mapping = aes(x = mean_temp_atemp, y = cnt), color = "#CC0099") +
  labs(x='Temperature',y='Total_Count',title='Nombre total de velos en fonction de la temperature moyen
```




```r
cor.test(day$mean_temp_atemp, day$cnt, method=c("pearson", "kendall", "spearman"))
```

```
##
##  Pearson's product-moment correlation
##
## data:  day$mean_temp_atemp and day$cnt
## t = 21.931, df = 729, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.584699 0.672260
## sample estimates:
##       cor
## 0.6304811
```

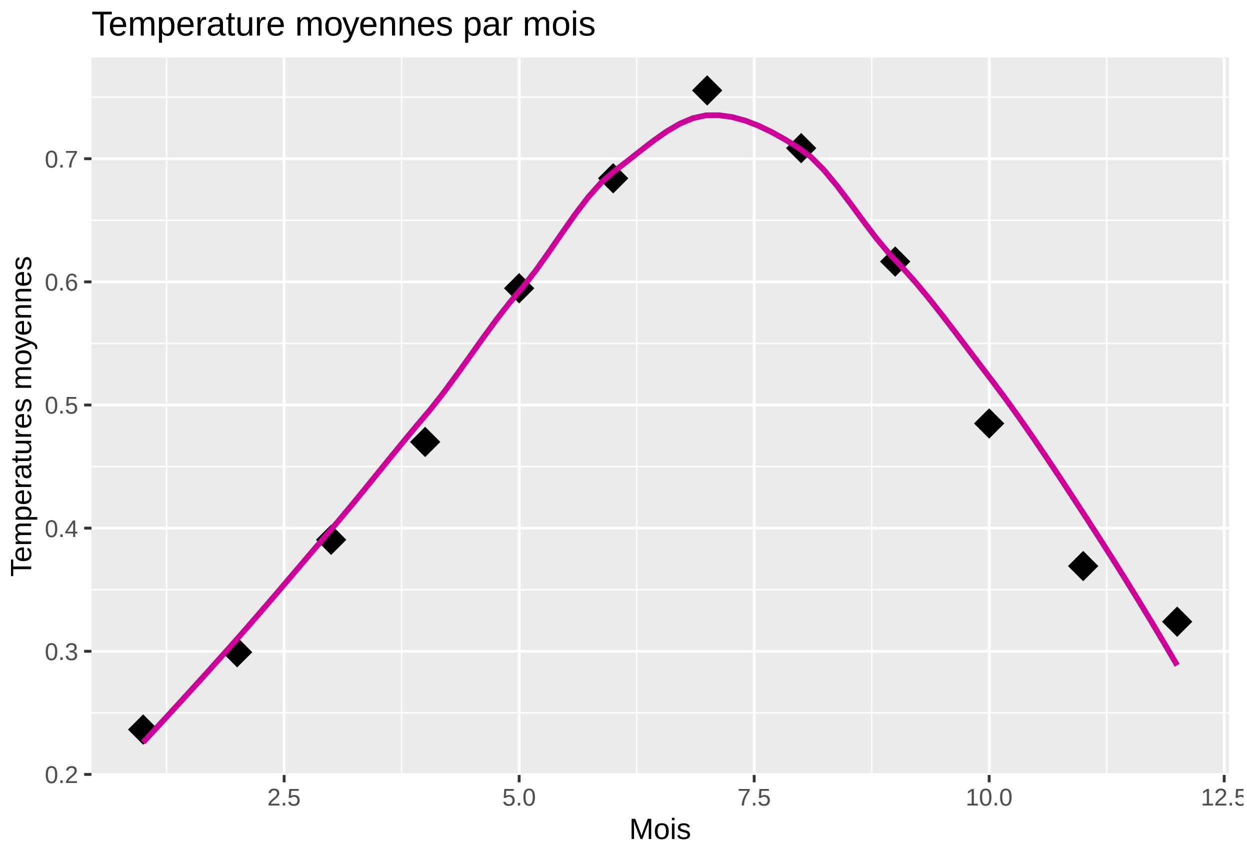
On obtient un resulat similaire avec un coefficient de correlation est de 0.6305, et une p-value inférieure a 5%, ce qui veut dire que la correlation est statistiquement significative.

1)c)

Temperatures moyennes par mois:

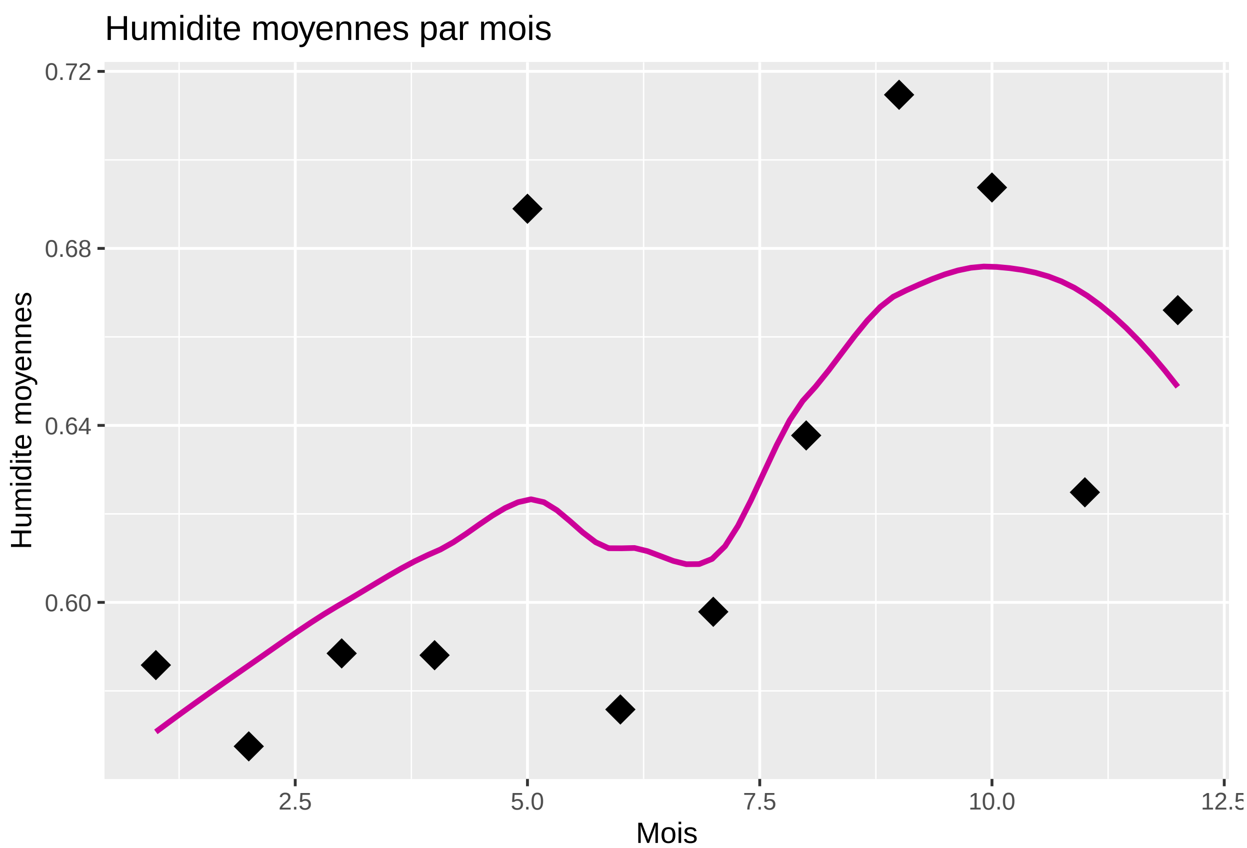
```r
#Temperatures moyennes
Temperatures_moyennes <- aggregate(day$temp, list(day$mnth), FUN=mean, na.rm=TRUE)
```

```r
ggplot(data = Temperatures_moyennes, mapping = aes(x = Group.1, y = x)) +
  geom_point(  shape = 18, size = 5) +
  geom_smooth(formula = y ~ x, method = "loess",se = FALSE,  color = "#CC0099")  +
  labs(x= "Mois", y =  "Temperatures moyennes", title="Temperature moyennes par mois" ) +
  xlim(1,12)
```



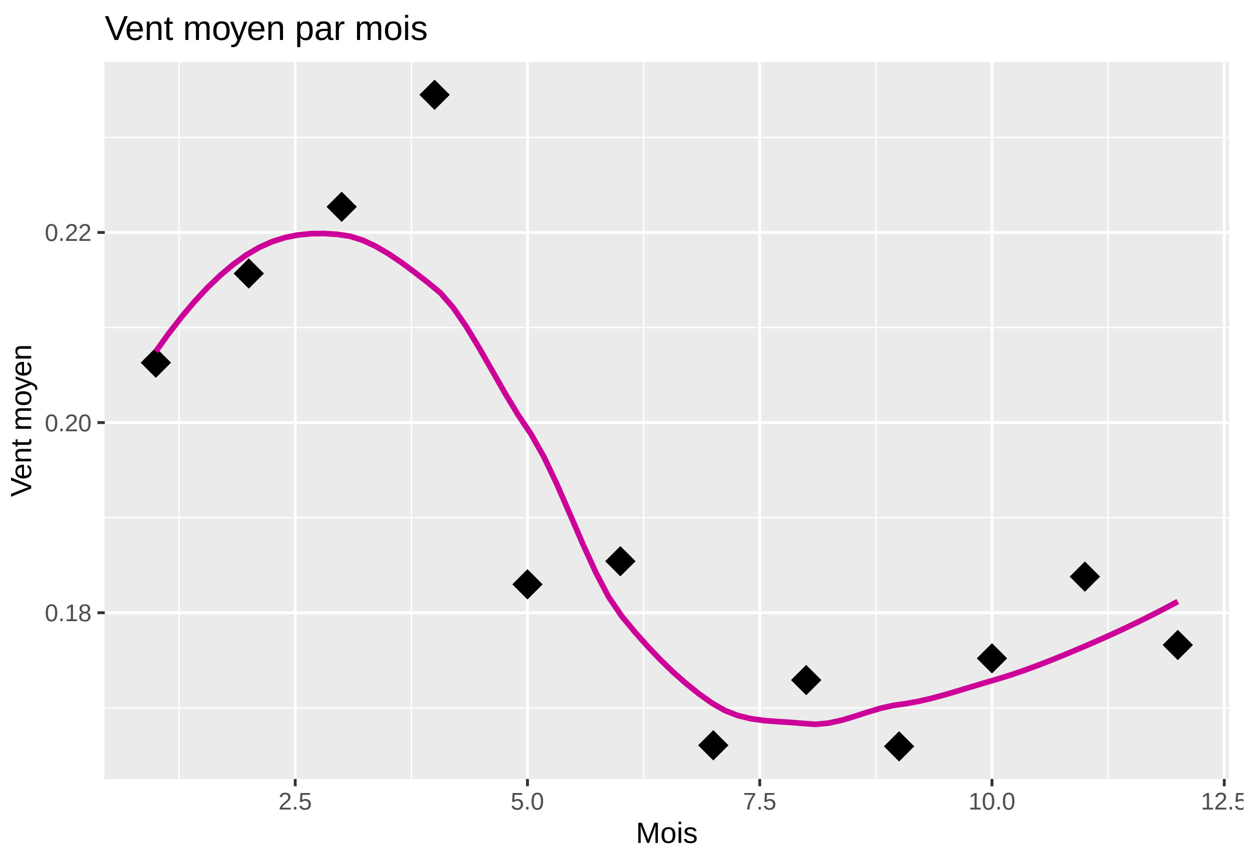
Humidite moyennes par mois:

```r
#Humidite moyenne
Hum_moyennes <- aggregate(day$hum, list(day$mnth), FUN=mean, na.rm=TRUE)
ggplot(data = Hum_moyennes, mapping = aes(x = Group.1, y = x)) +
  geom_point(  shape = 18, size = 5) +
  geom_smooth(formula = y ~ x, method = "loess",se = FALSE,  color = "#CC0099")  +
  labs(x= "Mois", y =  "Humidite moyennes", title="Humidite moyennes par mois" ) +
  xlim(1,12)
```



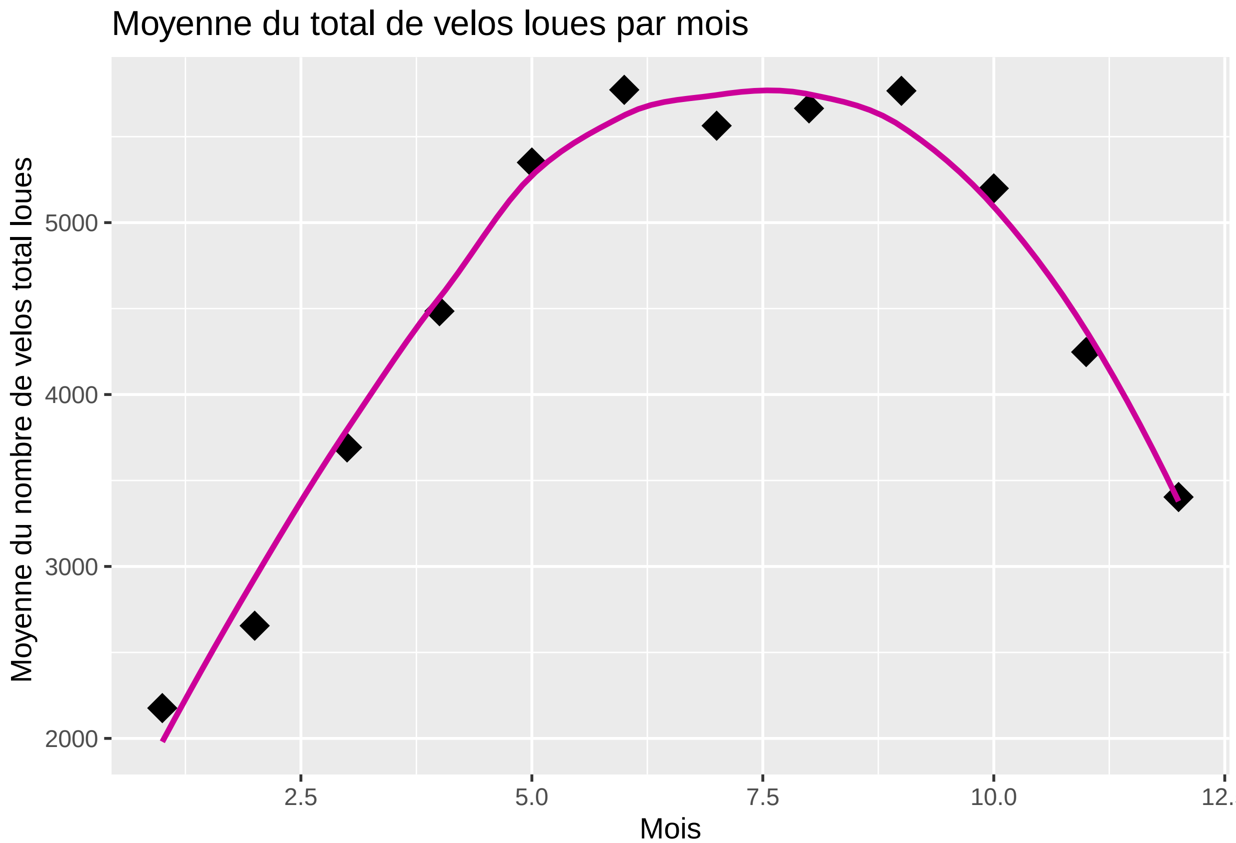
Vent moyen par mois:

```
#Vent moyen
Vent_moyen <- aggregate(day$windspeed, list(day$mnth), FUN=mean, na.rm=TRUE)
ggplot(data = Vent_moyen, mapping = aes(x = Group.1, y = x)) +
  geom_point(  shape = 18, size = 5) +
  geom_smooth(formula = y ~ x, method = "loess",se = FALSE,  color = "#CC0099")  +
  labs(x= "Mois", y =  "Vent moyen", title="Vent moyen par mois" ) +
  xlim(1,12)
```



Moyenne du total de velos loues par mois:

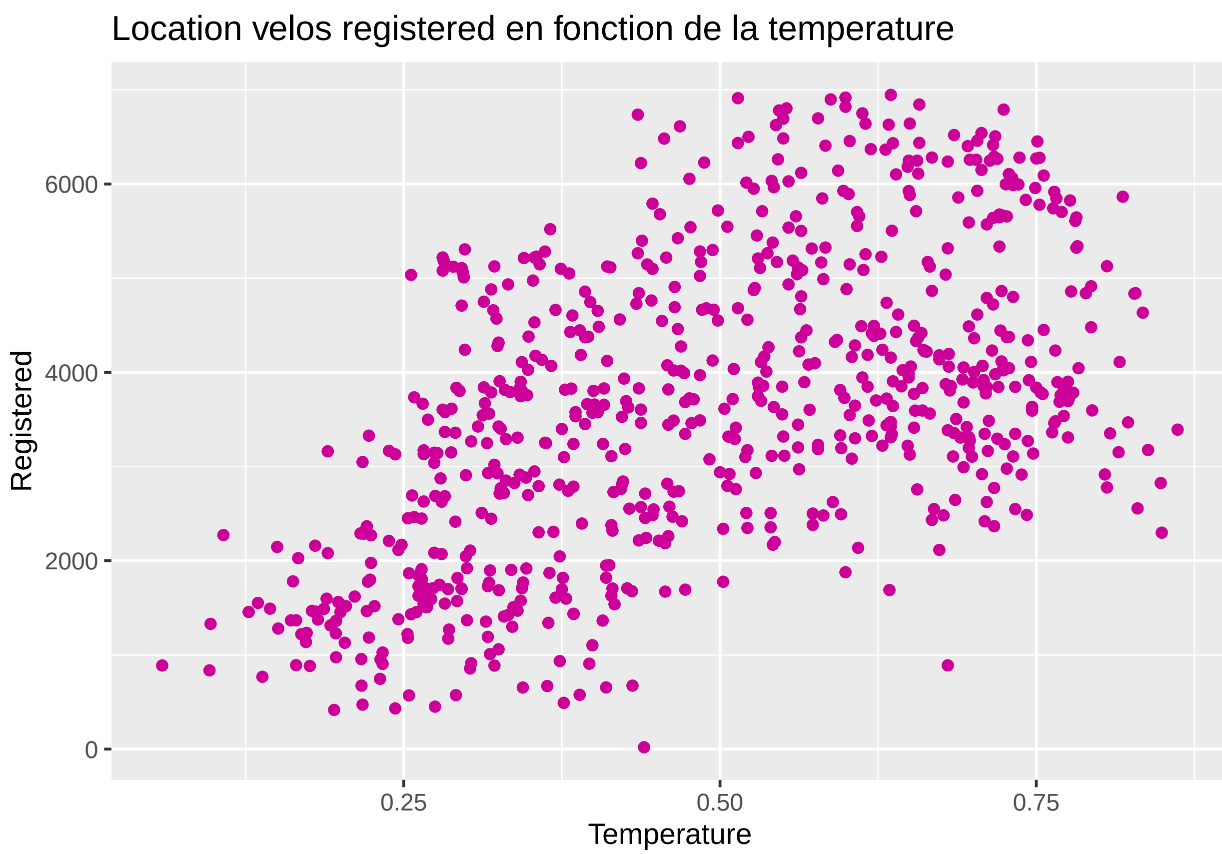
```
#Total velos loues moyennes
Total_moyen <- aggregate(day$cnt, list(day$mnth), FUN=mean, na.rm=TRUE)
ggplot(data = Total_moyen, mapping = aes(x = Group.1, y = x)) +
  geom_point(  shape = 18, size = 5) +
  geom_smooth(formula = y ~ x, method = "loess",se = FALSE,  color = "#CC0099")  +
  labs(x= "Mois", y =  "Moyenne du nombre de velos total loues ", title="Moyenne du total de velos loues
  xlim(1,12)
```



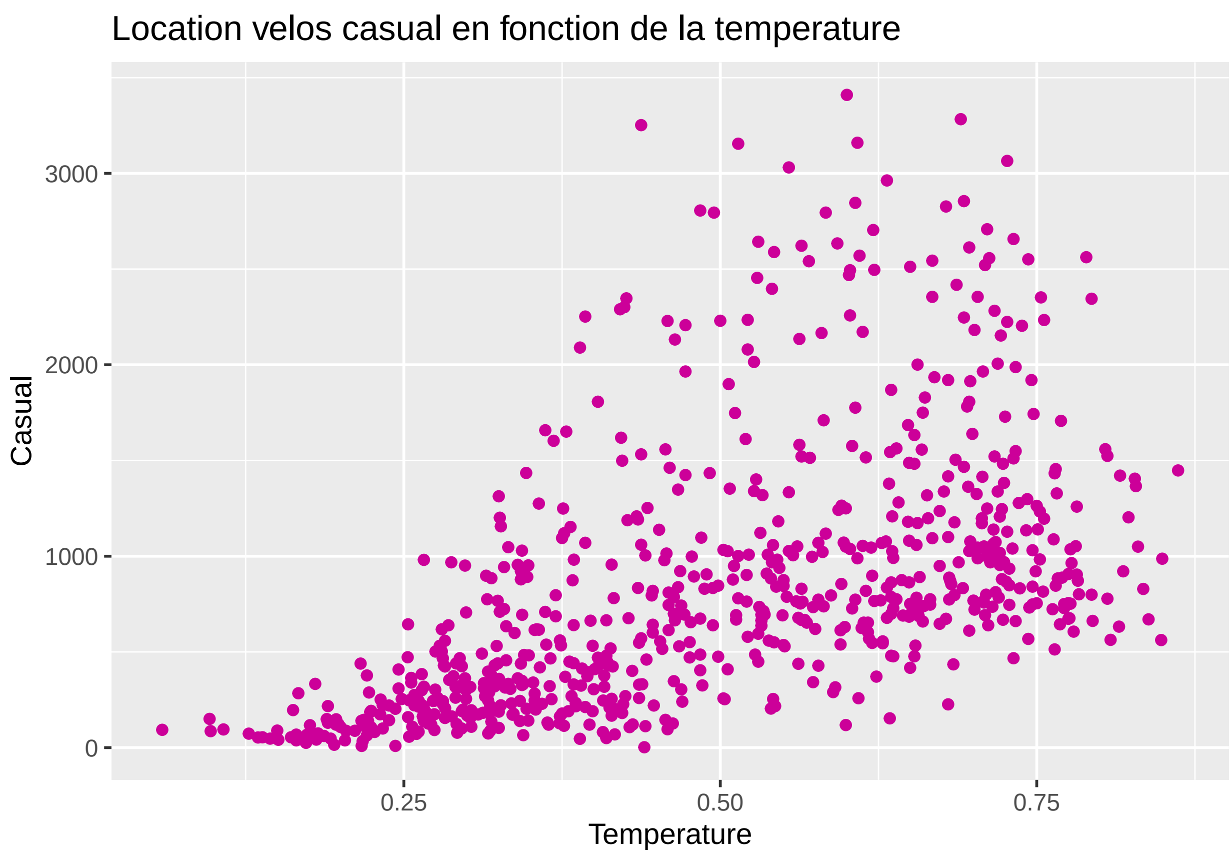
1)d)

correlation temp et les locations de vélos:

```
ggplot(data = day) +
  geom_point(mapping = aes(x =temp, y = registered), color = "#CC0099") +
  labs(x='Temperature',y='Registered',title='Location velos registered en fonction de la temperature')
```




```r
ggplot(data = day) +
  geom_point(mapping = aes(x =temp, y = casual), color = "#CC0099") +
  labs(x='Temperature',y='Casual',title='Location velos casual en fonction de la temperature')
```



Il semble y avoir une relation entre temp et registered/casual, en effet on remarque que les deux variables augmentent dans le meme sense.

On peut calculer le coefficient de corelation:

```
cor.test(day$temp, day$registered, method=c("pearson", "kendall", "spearman"))
```

```
##
##  Pearson's product-moment correlation
##
## data:  day$temp and day$registered
## t = 17.323, df = 729, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4865508 0.5894440
## sample estimates:
##      cor
## 0.540012
```

```
cor.test(day$temp, day$casual, method=c("pearson", "kendall", "spearman"))
```

```
##
##  Pearson's product-moment correlation
##
## data:  day$temp and day$casual
```

```
## t = 17.472, df = 729, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4900779 0.5924581
## sample estimates:
##       cor
## 0.5432847
```

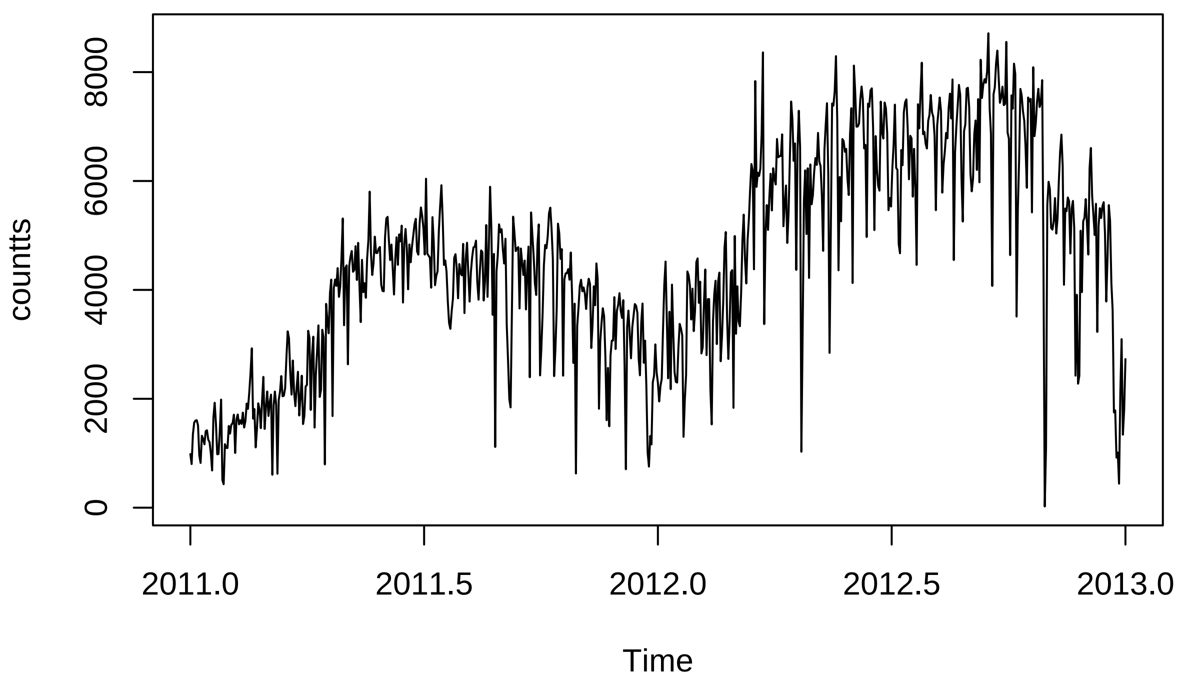
On constate que l'on a un coefficient de corrélation de 0.54 pour registered (et 0.543 pour registered) , ce qui n'est pas tres élevé.On a une p-value inférieure a 5%, donc la correlation est statistiquement significative.

Dans ce qui suit, nous allons construire un modele prédictif du nombre de locations de vélos par jour:

1)e)

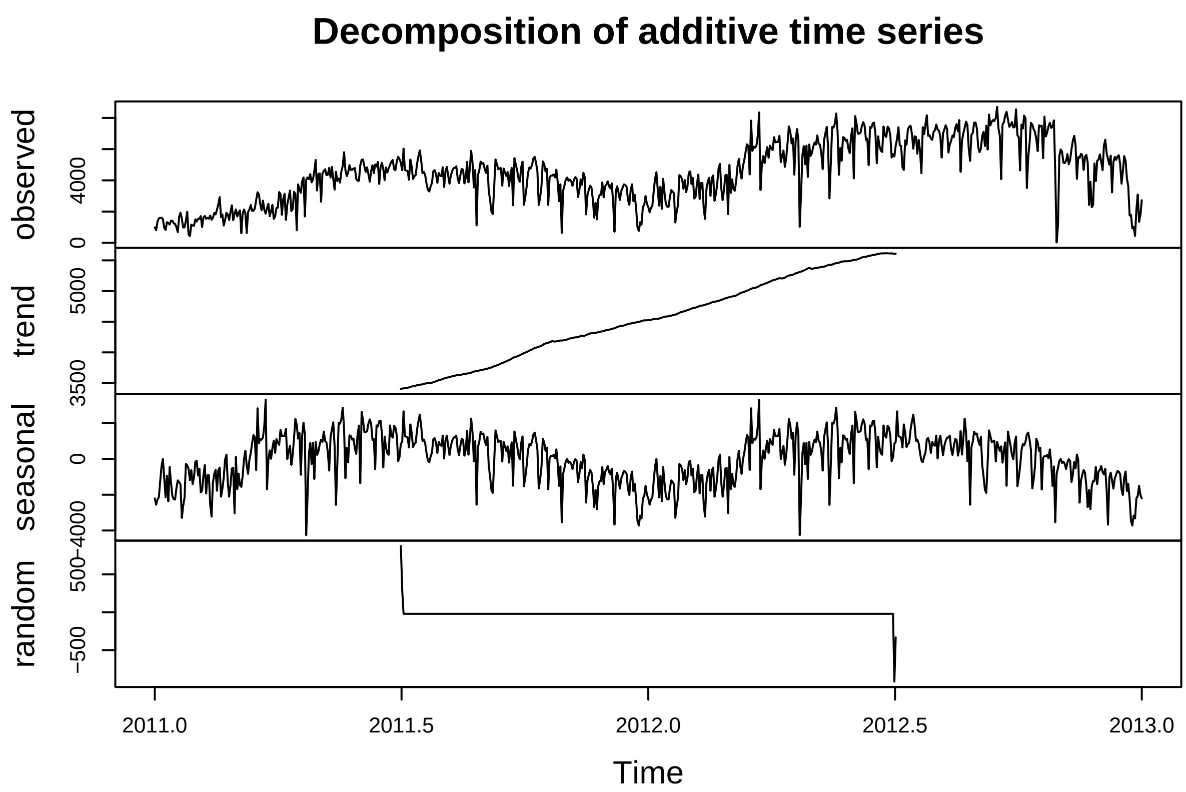
On cree une timeseries de la valiable cnt,qui commence le 1er janvier 2011 et qui a pour frequence 365 (car on a des informations journalieres)

```
countts <- ts(day$cnt, frequency=365, start=c(2011,1))
plot.ts(countts)
```



On decompose la time series pour bien voir la tendence et la saisonalite:

```r
counttscomponents <- decompose(countts)
plot(counttscomponents)
```



On remarque une tendance croissante, et on peut apercevoir deux saisons. La composante random, elle, n'est pas du tout stationnaire autours de 0 et ne forme pas un bruit blanc.

1)f) Enlever les outliers:

Pour enlever les outliers et les valuers manquantes de la time serie, on fait appel a la fonction `tsclean`:

```r
countts.clean <- tsclean(countts)
outliers.missing.val <- countts[countts.clean != countts]
outliers.missing.val
```

```
##  [1] 1543 1851 4036 4191 4595 4553 4660 4968 5515 1842 2395 2416 2424 2659  705
## [16] 8362 6857 7132 6624 4672 4549 4073 3510 5478   22 1096 4094 2277 2424 1341
```

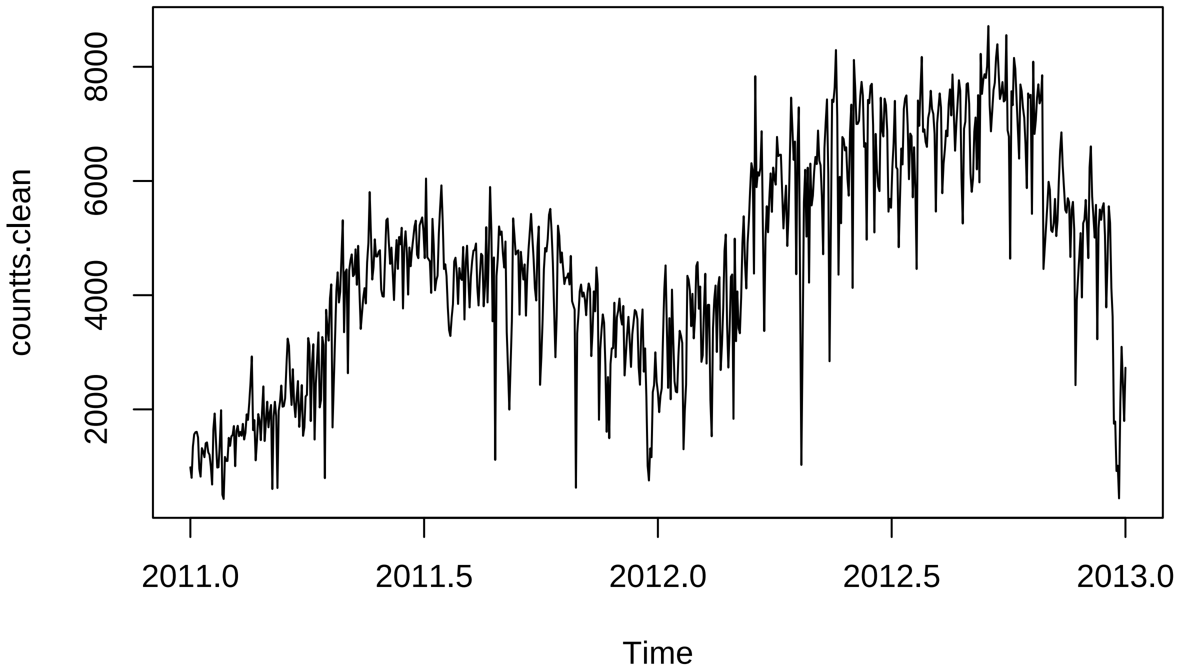
```r
length(outliers.missing.val)
```

```
## [1] 30
```

```
#first.cnt.ts.clean <-  tsclean(first_cnt_ts)
#outliers <- first_cnt_ts[first.cnt.ts.clean != first_cnt_ts ]
#outliers
```

On peut remarquer que la time serie de base countts contenait 30 outliers et/ou valeurs manquantes.

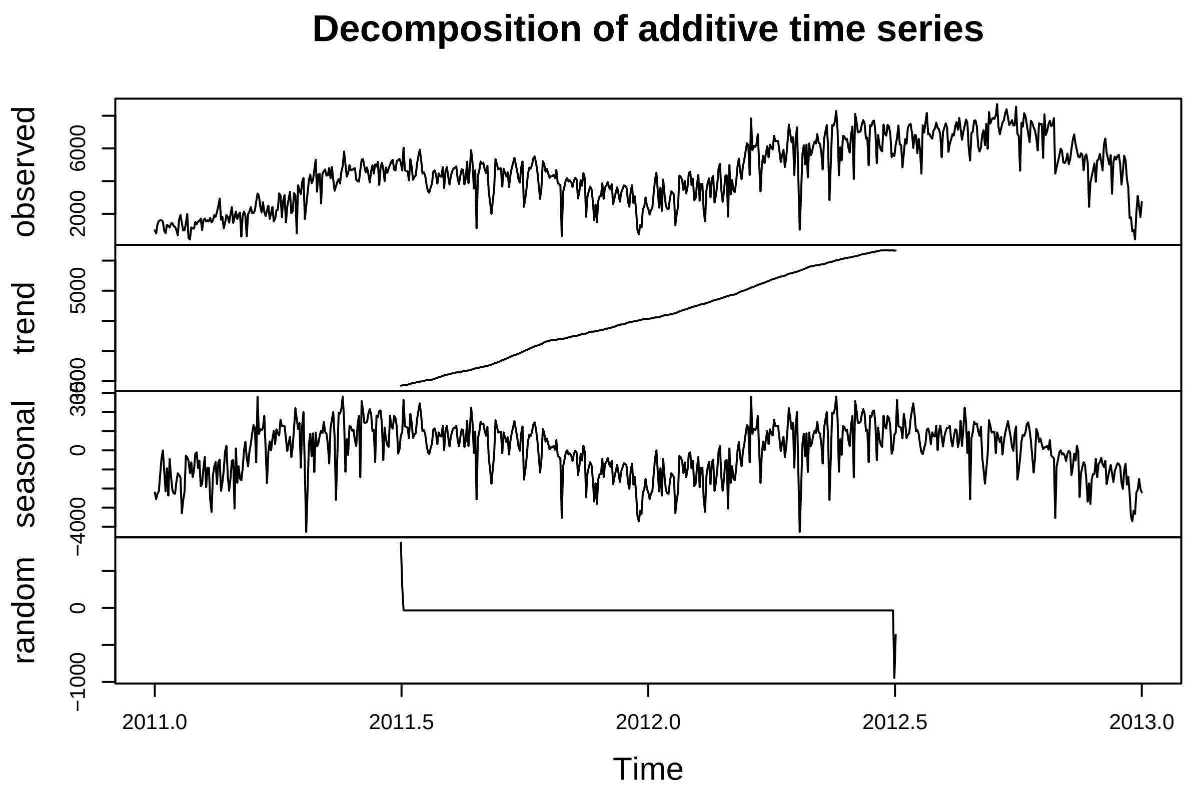
```
plot.ts(countts.clean)
```



Lisser la time series:

Afin de lisser la ts, nous devons choisir la bonne méthode, pour cela, nous allons vérifier la tendance et la saisonnalité de la série, grace a la décomposition de la serie sans outliers:

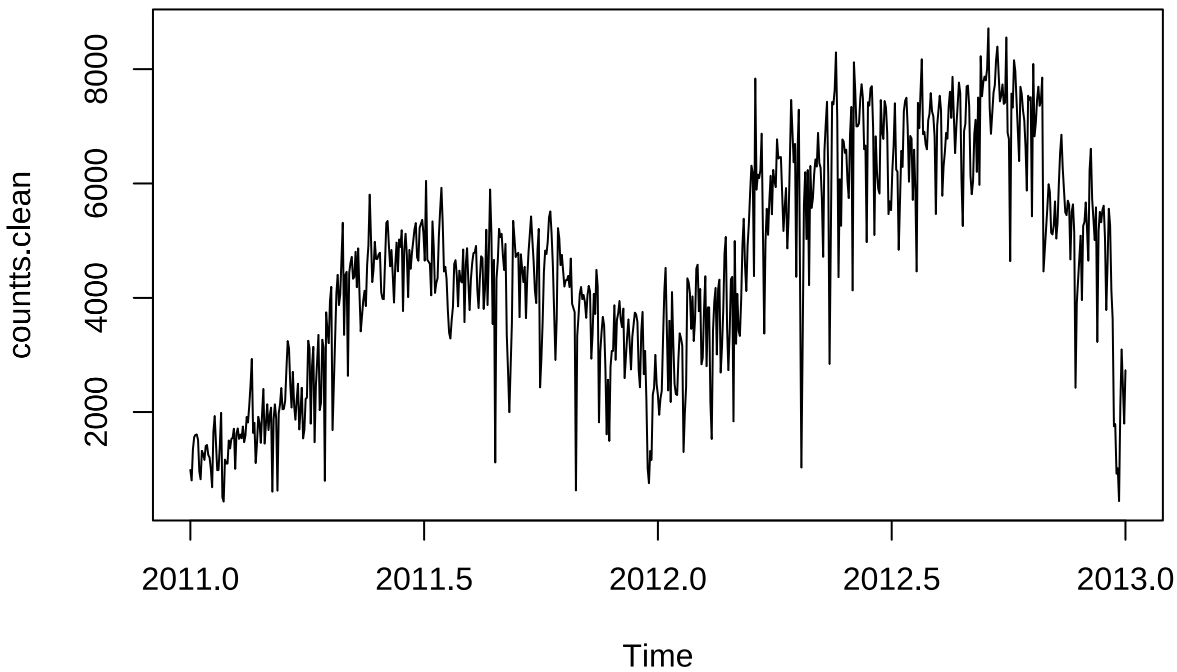
```
cout.clean.components <- decompose(countts.clean)
plot(cout.clean.components)
```



On voit bien que la série a une tendance ascendante, mais on voit aussi que la composante random n'est pas stationnaire du tout et ne ressemble pas à un bruit blanc, ce qui nous fait nous interroger sur la composante saisonnalité de la série chronologique.

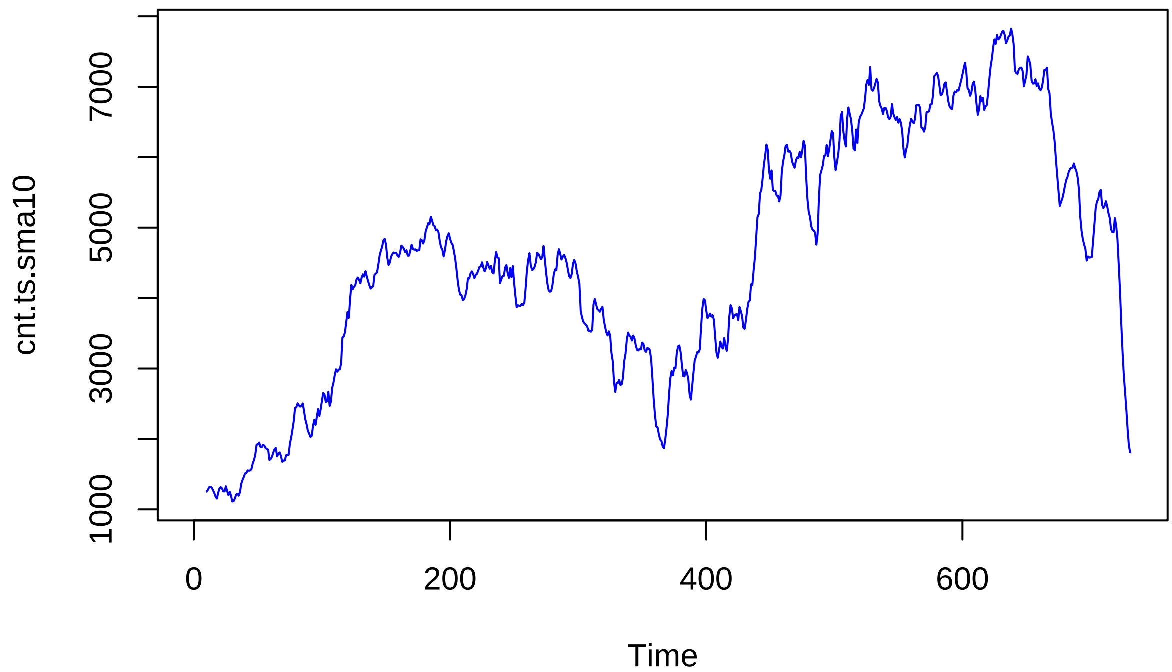
Cependant, nous allons en premier temps essayer de la lisser naïvement avec SMA:

```
cnt.ts.sma10 <- SMA(countts.clean, n=10)
plot.ts(countts.clean)
```



Lissage SMA:

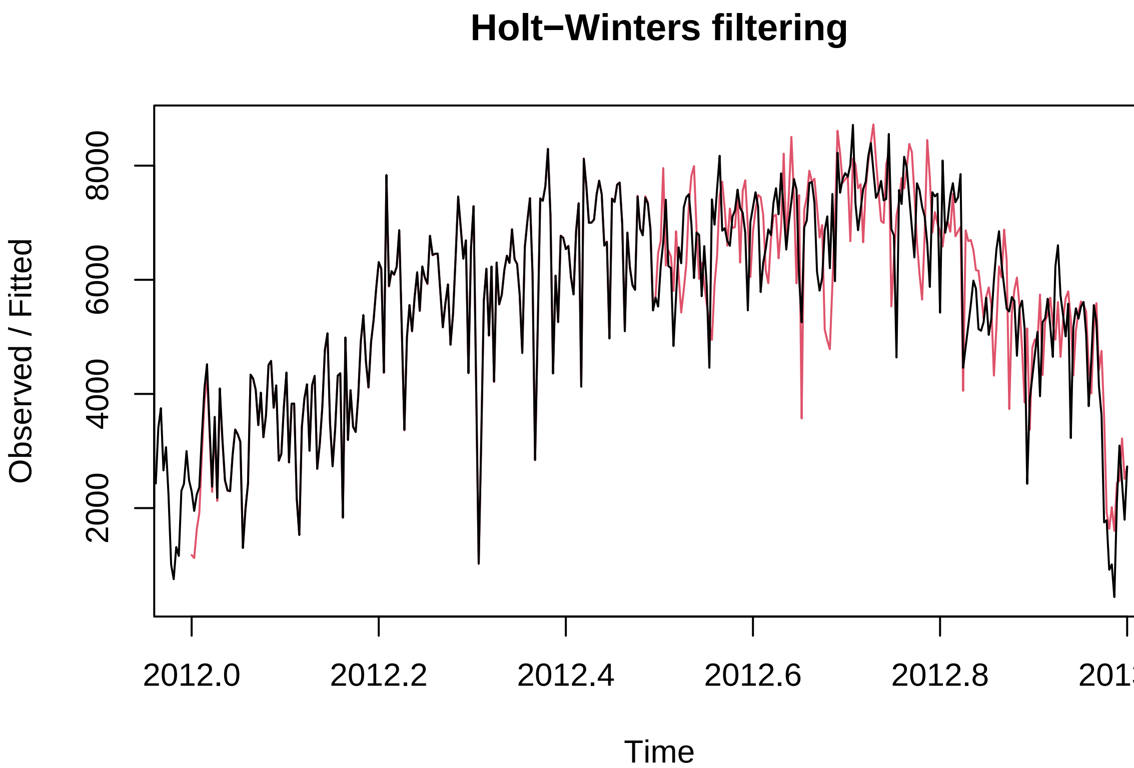
```
plot.ts(cnt.ts.sma10, col = "blue")
```



La serie chronologique parait bien lissée avec SMA, car elle garde sa forme d'origine en enlevant les piques (les valeurs les plus extremes).

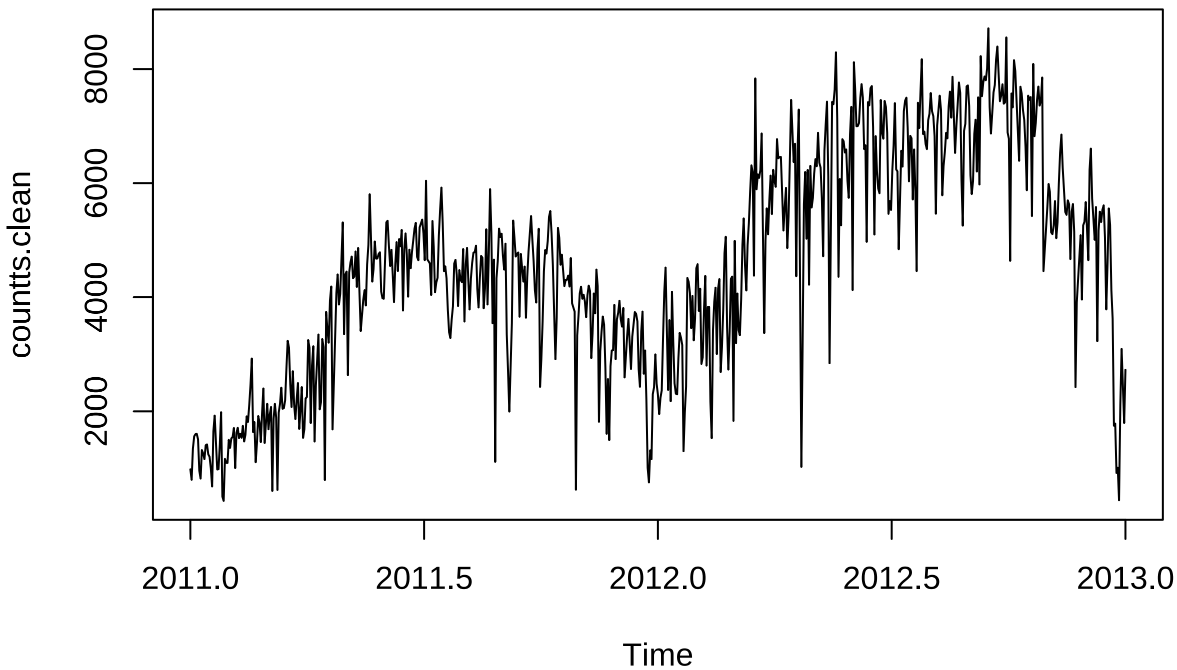
Nous pouvons aussi considerer que la serie chronologique a une saisonnalité, dans ce cas, on peut utiliser la méthode HoltsWinters pour la lisser :

```
cnt.ts.hw <- HoltWinters(countts.clean)
plot(cnt.ts.hw)
```

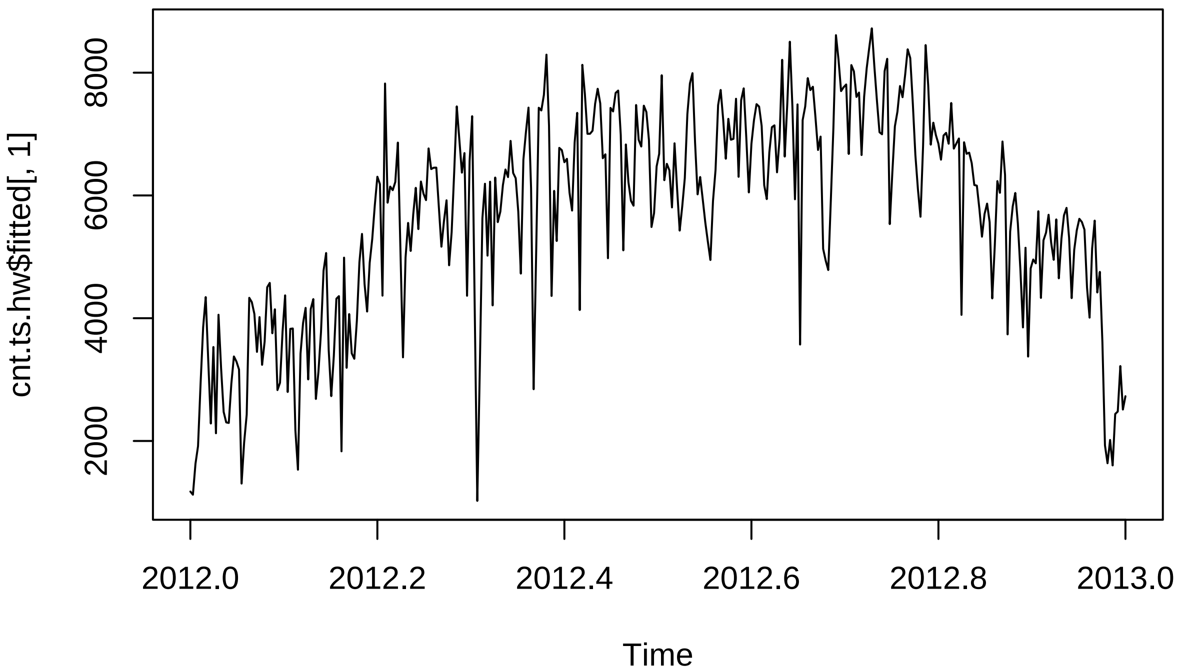


Lissage HoltWinters:

```
plot.ts(countts.clean)
```




```
plot.ts(cnt.ts.hw$fitted[,1])
```




```
#cnt.ts.hw
#?HoltWinters
```

On remarque que la courbe qui resulte du lissage HoltWinters ne lisse pas assez bien l'entiereté de la série chronologique.

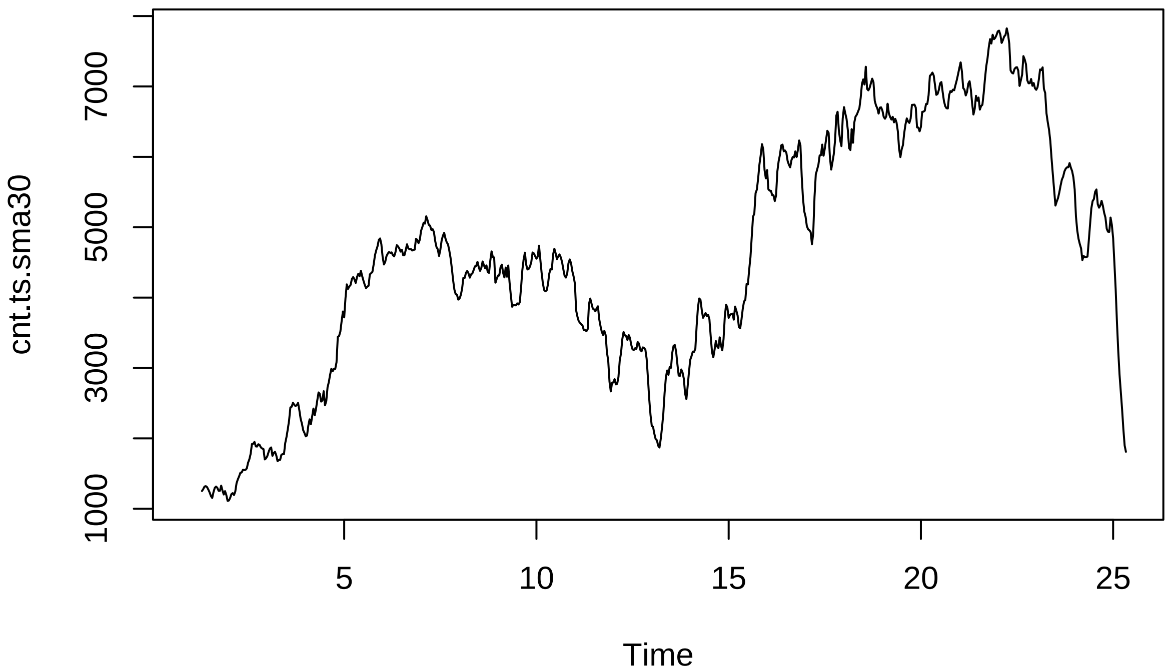
2 Choisir la meilleure methode de lissage:

Nous allons ajouter une fréquence aux deux séries lissées SMA et HoltWinters, puis nous analyserons les différences:

Ajouter une fréquence a la série chronologique SMA:

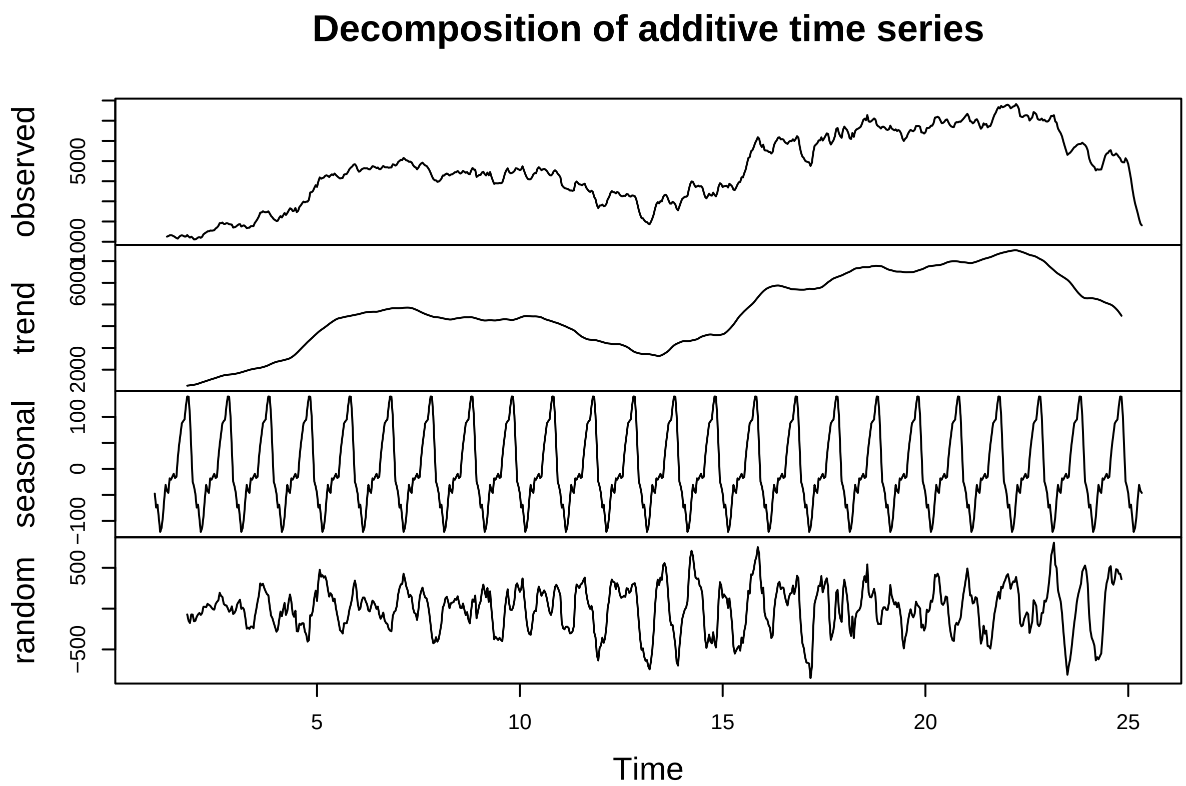
On rajoute une frequence de 30 pour avoir une saisonalité par mois.

```
cnt.ts.sma30 <- ts(cnt.ts.sma10, frequency = 30)
plot.ts(cnt.ts.sma30)
```



Pour avoir plus d'informations sur cette nouvelle série chronologique, nous allons la décomposer:

```
dec.sma30 <- decompose(cnt.ts.sma30)
plot(dec.sma30)
```



On remarque assez facilement qu'il y a un pattern dans la saisonalité, et on peut apercevoir une tendance ascendante, (même si elle est moins claire que dans la décomposition précédante de 365)

Voyons si la composante random représente un bruit blanc, pour cela, nous allons utiliser le test Ljung

```
Box.test(dec.sma30$random, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  dec.sma30$random
## X-squared = 576.26, df = 1, p-value < 2.2e-16
```

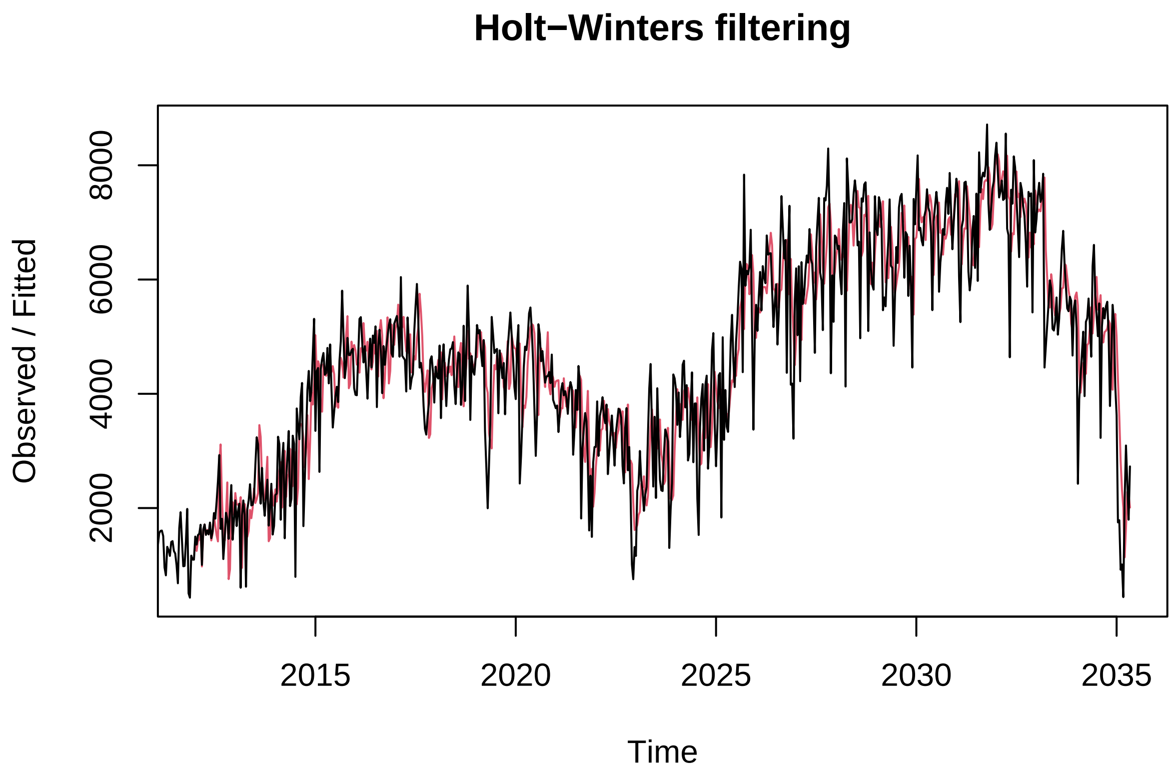
La p-value est bien inferieure de 5%, ce qui nous mène à rejeter l'hypothèse nulle, i.e il reste une autocorrelation dans la série et elle ne représente pas un bruit blanc.

Ajouter une fréquence a la série chronologique HoltWinters:

```
#create a new time series with the right frequence
new.ts.30 <- ts(countts.clean, frequency = 30, start  = c(2011, 1))
#clean the ts
new.ts.clean30 <- tsclean(new.ts.30)

#smooth it using HoltWinters method
```

```r
cnt.ts.hw30 <- HoltWinters(new.ts.clean30)
plot(cnt.ts.hw30)
```




```r
cnt.ts.hw30
```

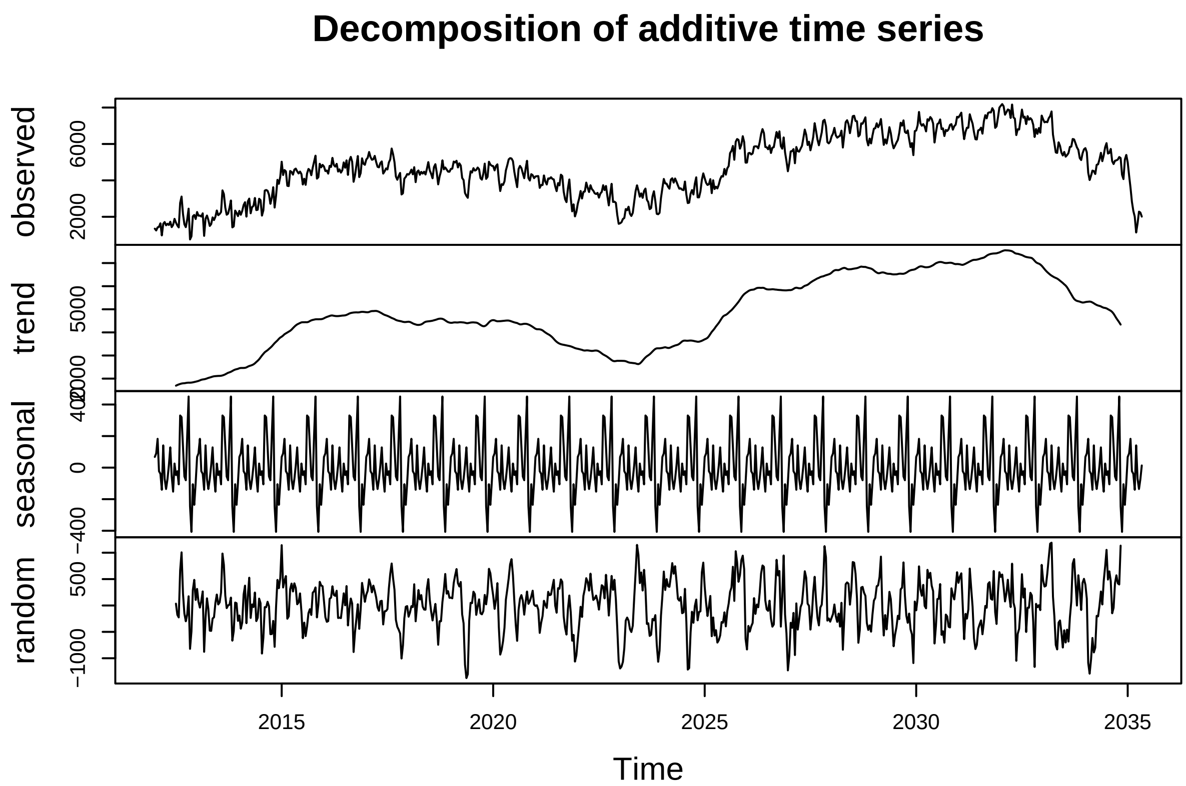
```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = new.ts.clean30)
##
## Smoothing parameters:
##  alpha: 0.3445698
##  beta : 0.001784322
##  gamma: 0.149012
##
## Coefficients:
##            [,1]
## a   2400.53042
## b      1.93023
## s1   107.90503
## s2   138.28150
## s3    52.32732
## s4   -65.10501
## s5   126.40466
## s6  -203.31726
```

```
## s7    38.84877
## s8   -28.06696
## s9   410.89781
## s10  255.36043
## s11  188.27574
## s12  138.90948
## s13   30.57800
## s14  -67.22405
## s15 -166.51876
## s16 -427.11165
## s17 -648.24882
## s18 -222.47586
## s19 -157.91921
## s20 -138.84877
## s21 -515.39701
## s22 -245.25510
## s23 -409.05631
## s24 -197.30943
## s25  -88.52710
## s26 -171.40467
## s27   40.93787
## s28   55.93617
## s29 -111.23303
## s30  -73.26473
```

On remarque que le lissage HoltWinters apparait beaucoup mieux avec une frequence 30 plutot que la frequence 365 testée précédemment

Voyons maintenant sa décomposition:

```
decompose.hw30 <- decompose(cnt.ts.hw30$fitted[,1])
plot(decompose.hw30)
```



Il y a bien une saisonnalité assez claire, la tendance apparait proche que celle qu'on a observé dans la décomposition de SMA. Intéressons nous maintenant a la composante random et voyons si c'est un bruit blanc:

```
Box.test(decompose.hw30$random, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  decompose.hw30$random
## X-squared = 287.34, df = 1, p-value < 2.2e-16
```

Ici aussi, on a une p-value inférieure a 5%, ce qui montre que la composante random n'est pas un bruit blanc. Ce qui veut dire qu'il reste encore de l'information dans la composante random qui n'a pas été pris en compte, dans la saisonalité ou dans la tendance de la série.

Cependant, comme on a bien remarqué une saisonalité dans la série chronologique en mettant la fréquence a 30, il serait mieux de choisir le lissage HoltWinters, qui est approprié pour une timeséries avec saisonalité.

Modeliser la série lissée avec ARIMA:

Avant toute chose, nous devons nous assurer que la série chronologique est bien stationnaire, pour cela, on effectue un test `adf`:

```r
#récuperer la ts:
cnt.ts.hw <- cnt.ts.hw30$fitted[,1]

#tester la stationnarité:
adf.test(cnt.ts.hw)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cnt.ts.hw
## Dickey-Fuller = -0.80104, Lag order = 8, p-value = 0.9615
## alternative hypothesis: stationary
```

La p-value est supérieure a 5%, ce qui veut dire que la serie n'est pas stationnaire, il va falloir la différencier.

Pour estimer le nombre de différentiations qu'on doit faire a la série, on peut utiliser la fonction `ndiffs`

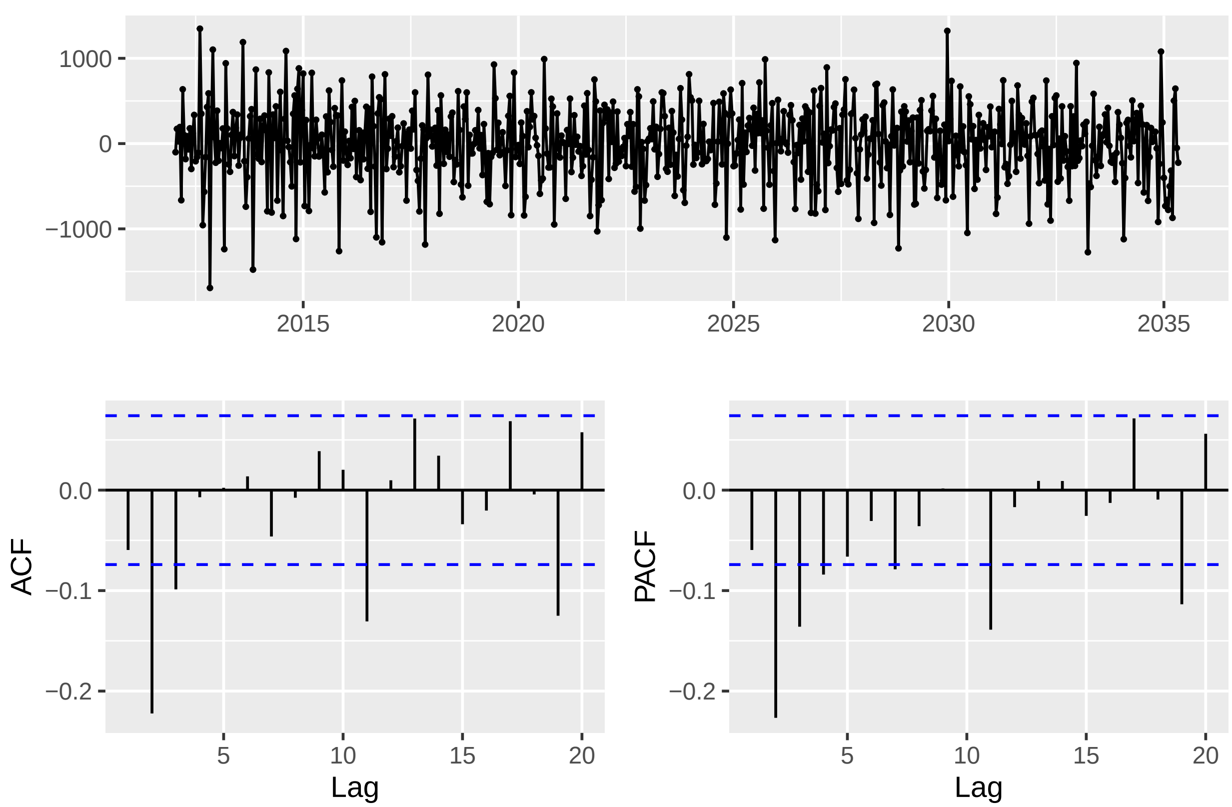
```r
ndiffs(cnt.ts.hw)
```

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

On doit différencier une seule fois.

Intéressons nous maintenant aux ACF et PACF:

```r
cnt.ts.hw.diff <- diff(cnt.ts.hw, differences = 1 )
ggtsdisplay(cnt.ts.hw.diff, lag = 20)
```



Les modeles candidats sont MA(2), AR(4) (apres différenciation) ou ARIMA(4,1,2)

D'apres le principe de parsimonie, on devrait choisir le premier candidat MA(2), car il ne contient que deux parametres.

```
cnt.ts.hw.ma2 <- arima(cnt.ts.hw, order = c(0,1,2))
cnt.ts.hw.ma2
```

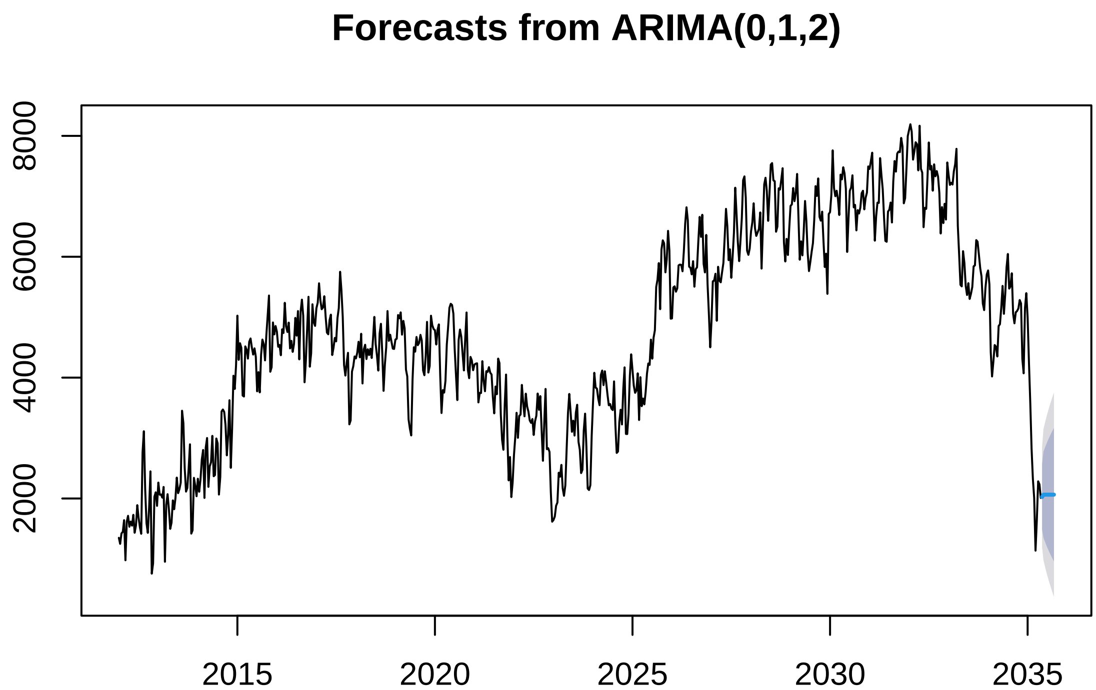
MA(2)

```
##
## Call:
## arima(x = cnt.ts.hw, order = c(0, 1, 2))
##
## Coefficients:
##           ma1      ma2
##       -0.1624  -0.2831
## s.e.   0.0368   0.0374
##
## sigma^2 estimated as 178718:  log likelihood = -5226.11,  aic = 10458.23
```

On a un log likelihood négatif, et un AIC = 10437.32

Predictions du modele MA(2):

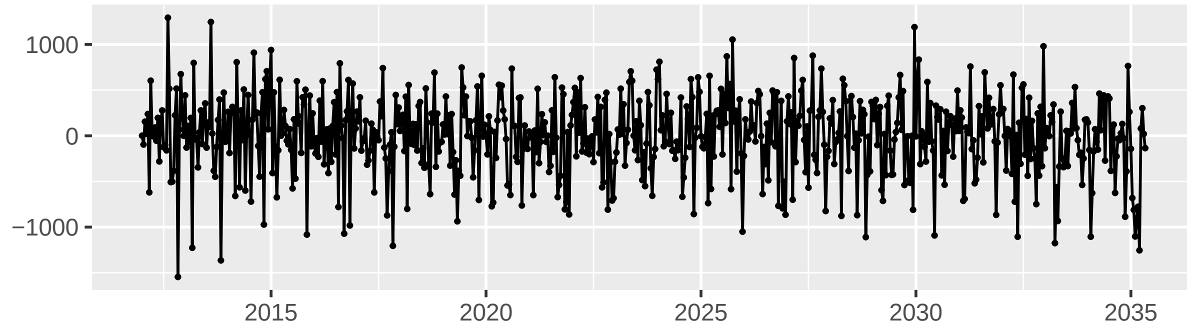
```
cnt.ts.hw.ma2.forecast <- forecast(cnt.ts.hw.ma2, h = 10)
plot(cnt.ts.hw.ma2.forecast)
```

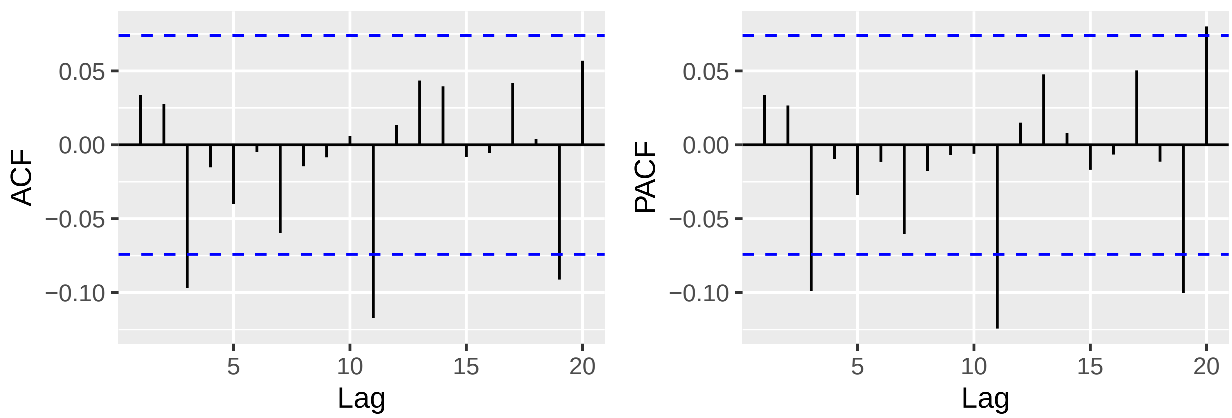


Le forecast n'a pas l'air tres satisfaisant, car la prediction est une ligne droite.

Voyons les résiduels de ce forecast:

```
ggtsdisplay(cnt.ts.hw.ma2.forecast$residuals, lag = 20)
```





On remarque qu'il reste quand meme des lags (3) non nuls dans les ACF des résiduels.

```
Box.test(cnt.ts.hw.ma2.forecast$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  cnt.ts.hw.ma2.forecast$residuals
## X-squared = 0.79688, df = 1, p-value = 0.372
```

La p-value est supérieure a 5%, ce qui veut dire que les résiduels peuvent etre interpretés comme un bruit blanc.

On s'interesse aux deux autres modeles pour comparer leurs resultats:

```
cnt.ts.hw.ar4 <- arima(cnt.ts.hw, order = c(4,1,0))
cnt.ts.hw.ar4
```

AR(4)

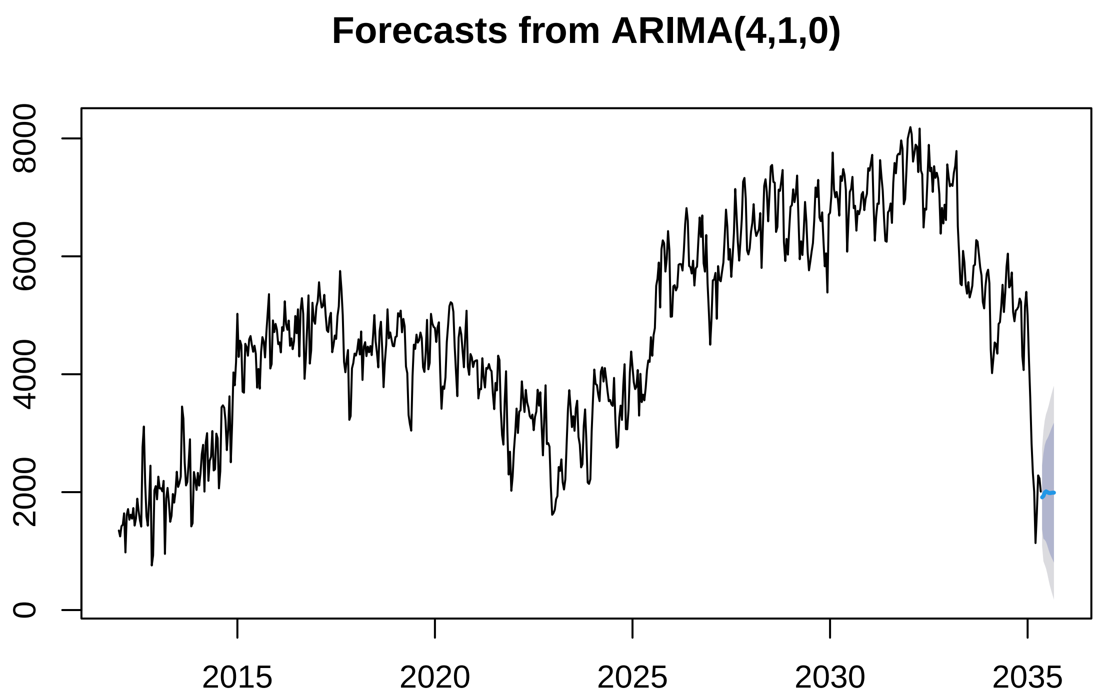
```
##
## Call:
## arima(x = cnt.ts.hw, order = c(4, 1, 0))
```

```
##
## Coefficients:
##           ar1      ar2      ar3      ar4
##       -0.1152  -0.2558  -0.1446  -0.0840
## s.e.   0.0377   0.0375   0.0375   0.0377
##
## sigma^2 estimated as 178294:  log likelihood = -5225.27,  aic = 10460.54
```

Le log likelihood est aussi negatif, et l'AIC est de 10443, ce qui est supérieur a l'AIC du modele MA(2), donc si on cherche a minimiser l'AIC, MA(2) est un meilleur modele.

Predictions du modele AR(4):

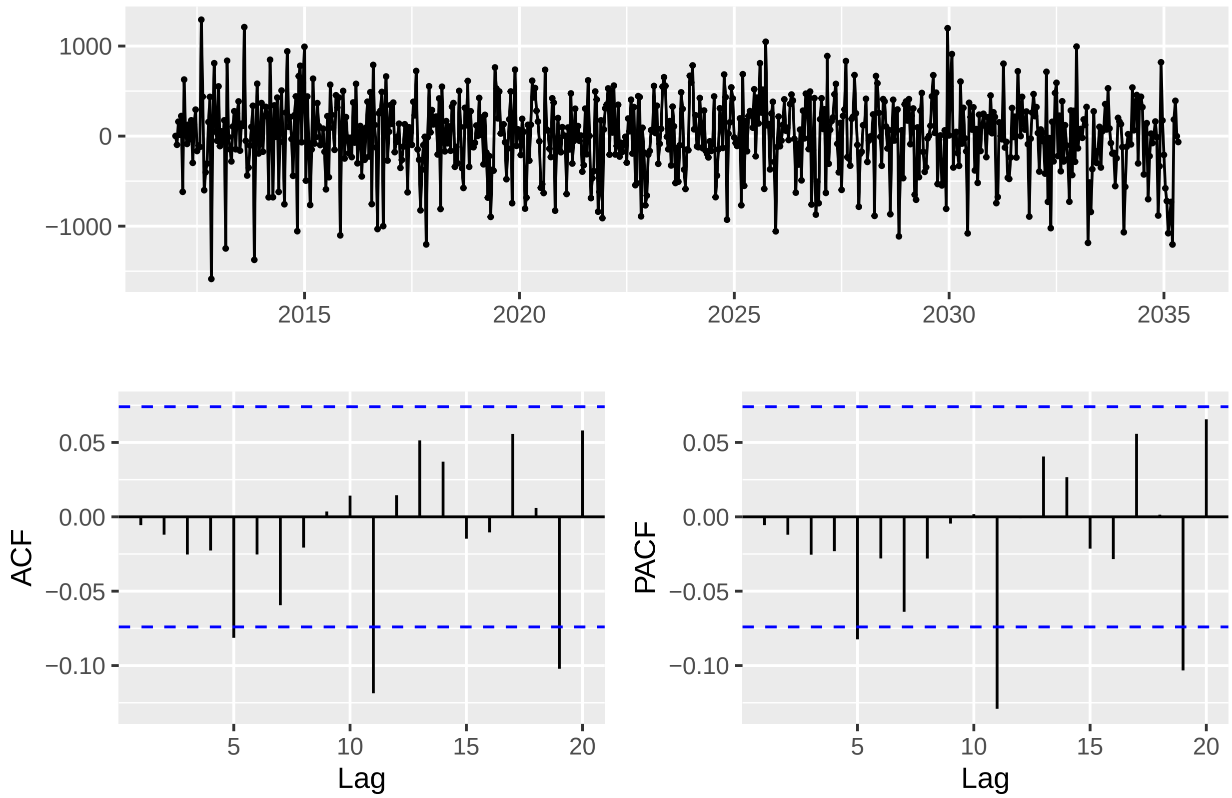
```
cnt.ts.hw.ar4.forecast <- forecast(cnt.ts.hw.ar4, h = 10)
plot(cnt.ts.hw.ar4.forecast)
```



Le forecast semble bien prendre en compte les variations de la timeseries.

Voyons les résiduels de ce forecast:

```
ggtsdisplay(cnt.ts.hw.ar4.forecast$residuals, lag = 20)
```



Ici aussi, on remarque des lags (4) non nuls au niveau de l'ACF des résidus.

```
Box.test(cnt.ts.hw.ar4.forecast$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  cnt.ts.hw.ar4.forecast$residuals
## X-squared = 0.02193, df = 1, p-value = 0.8823
```

On a une p-value supérieure a 5%, les residus forment donc un bruit blanc.

Interessons nous maintenant au dernier modele ARIMA(4,1,2):

```
cnt.ts.hw.arima7 <- arima(cnt.ts.hw, order = c(4,1,2))
cnt.ts.hw.arima7
```

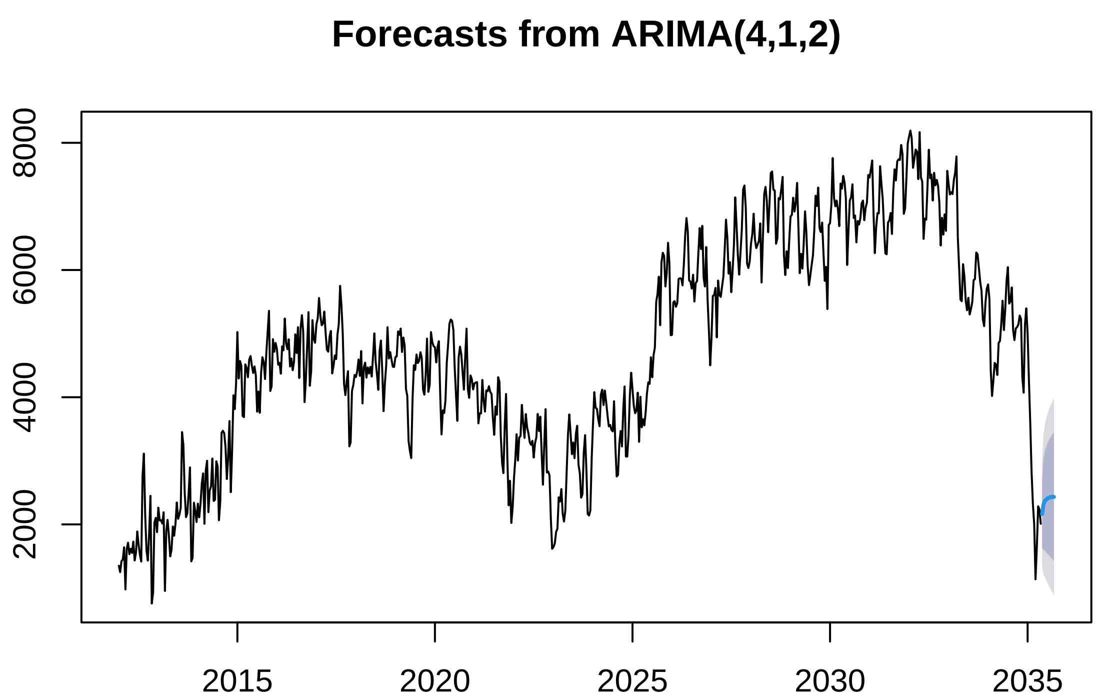
ARIMA(4,1,2)

```
##
## Call:
## arima(x = cnt.ts.hw, order = c(4, 1, 2))
##
```

```
## Coefficients:
##           ar1     ar2      ar3     ar4     ma1      ma2
##       -0.2748  0.5079  -0.1242  0.0904  0.1619  -0.8222
## s.e.   0.0742  0.0777   0.0517  0.0485  0.0625   0.0617
##
## sigma^2 estimated as 172464:  log likelihood = -5214.07,  aic = 10442.15
```

On remarque que ARIMA(4,1,2) minimise l'AIC.

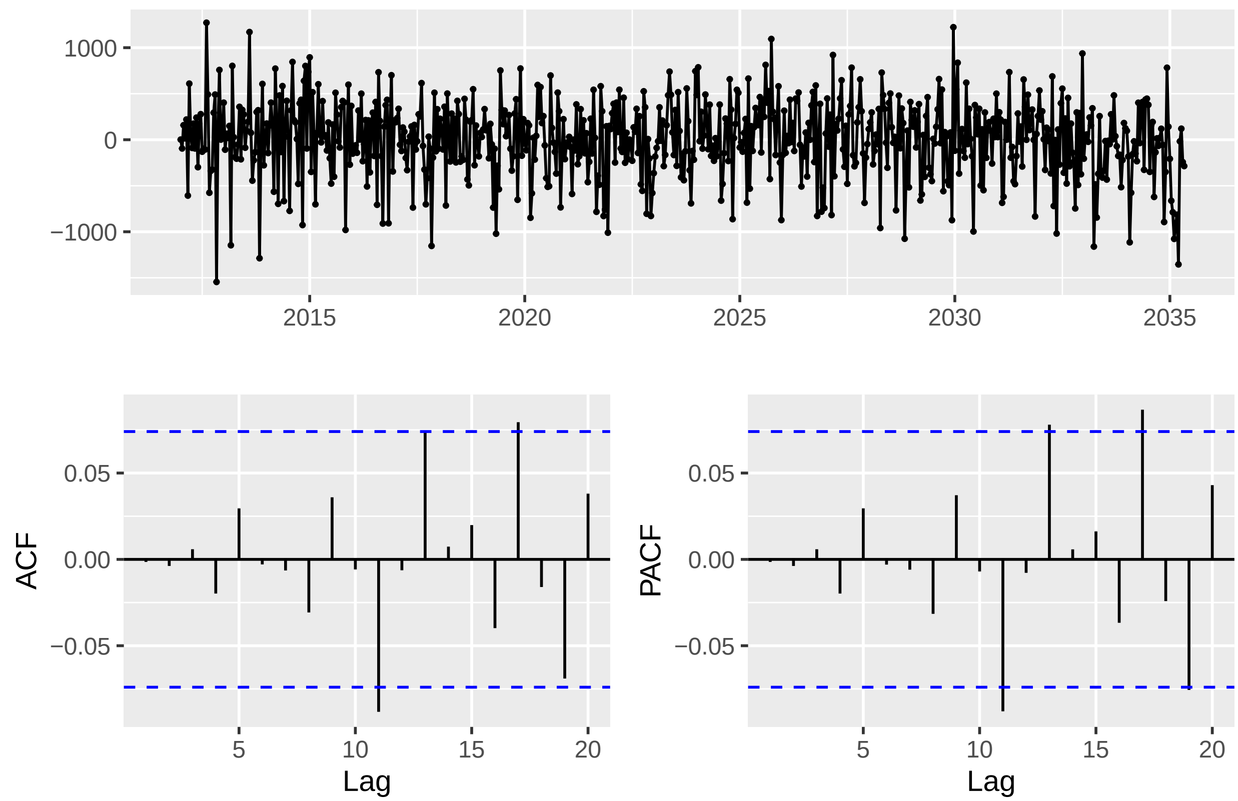
```
cnt.ts.hw.arima7.forecast <- forecast(cnt.ts.hw.arima7, h = 10)
plot(cnt.ts.hw.arima7.forecast)
```



On remarque une hausse dans les predictions est non pas une ligne droite.

Voyons les résiduels de ce forecast:

```
ggtsdisplay(cnt.ts.hw.arima7.forecast$residuals, lag = 20)
```



Ici aussi, on a deux lags non nuls dans les ACF des résiduels.

```
Box.test(cnt.ts.hw.ar4.forecast$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  cnt.ts.hw.ar4.forecast$residuals
## X-squared = 0.02193, df = 1, p-value = 0.8823
```

p-value > 5% donc résiduels sont susceptibles d'etre un bruit blanc.

Pour résumer, en se basant sur le principe de parsimonie, le modele AR(2) parait le plus interessant, mais si on s'interesse a minimiser l'AIC et a avoir des résidus qui se rapprochent le plus d'un bruit blanc, alors le modele ARIMA(4,1,2) est a privilégier. "Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful." -George E. P. Box, Norman R. Draper Empirical Model

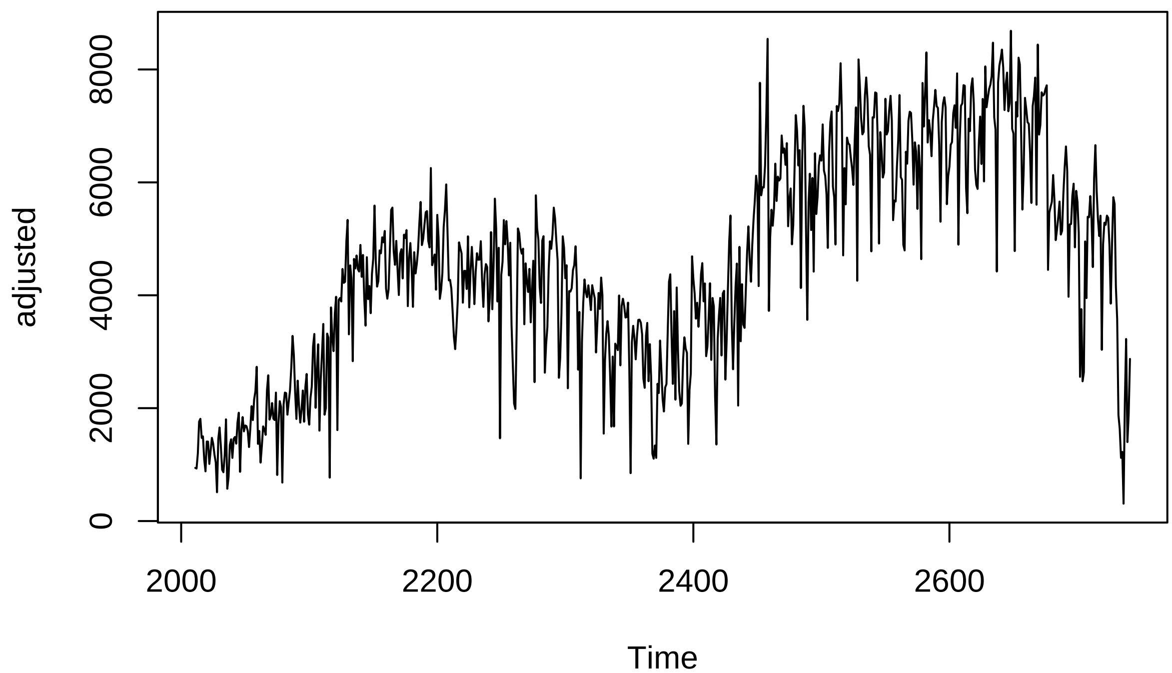
Q4: Prédictions avec le modele ARIMA

4)I) On enleve la saisonalité:

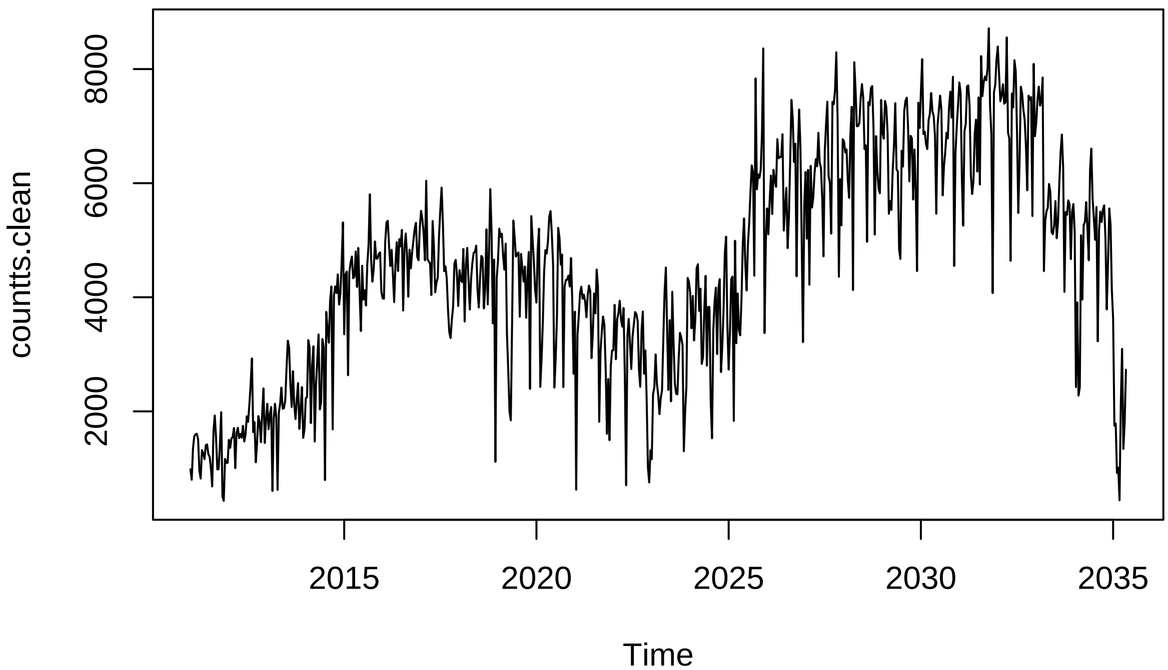
```r
#lire la ts avec frequence 30
countts <- ts(day$cnt, frequency=30, start=c(2011, 1))
#clean la ts
countts.clean <- tsclean(countts)
#decomposer la ts
decomp <- decompose(countts.clean)

#ajuster la ts en enlevant la saisonnalite
adjusted <- countts.clean - decomp$seasonal

adjusted <- ts(adjusted, start=c(2011, 1))
plot.ts(adjusted)
```




```r
plot.ts(countts.clean)
```



Voyons si la timeseries est stationnaire:

```
adf.test(adjusted)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  adjusted
## Dickey-Fuller = -1.2335, Lag order = 9, p-value = 0.9011
## alternative hypothesis: stationary
```

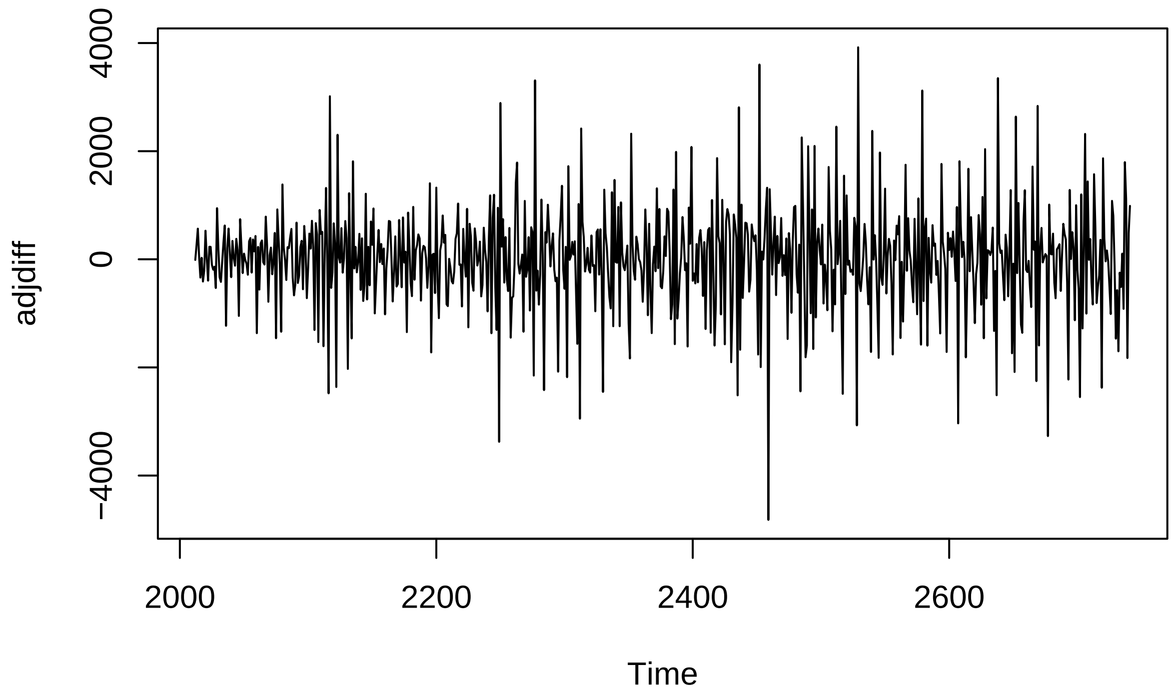
On a une timeseries non stationnaire, donc on doit differencier, on utilise ndiffs pour savoir combien de fois il faut le faire.

```
ndiffs(adjusted)
```

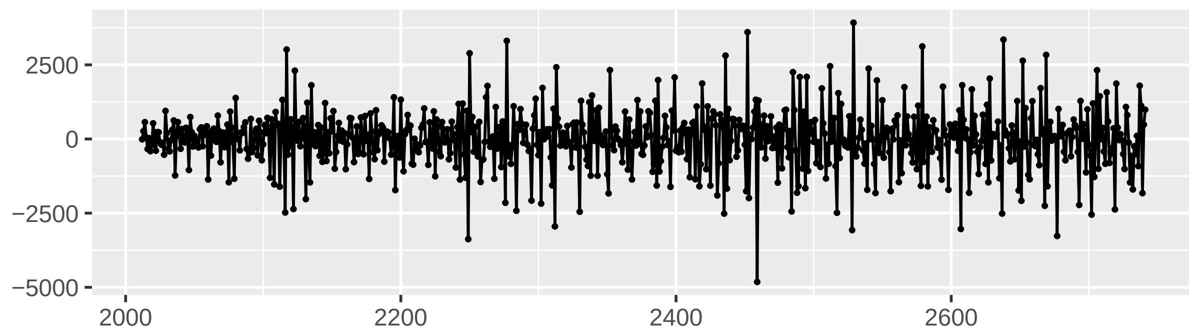
```
## [1] 1
```

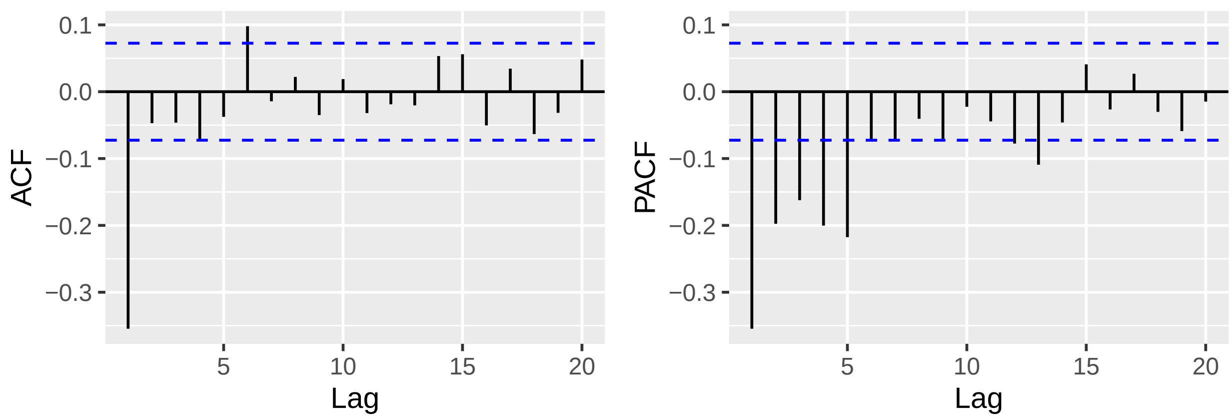
On differencie une fois

```
adjdiff <- diff(adjusted, differences=1)
plot.ts(adjdiff)
```




```
ggtsdisplay(adjdiff, lag=20)
```





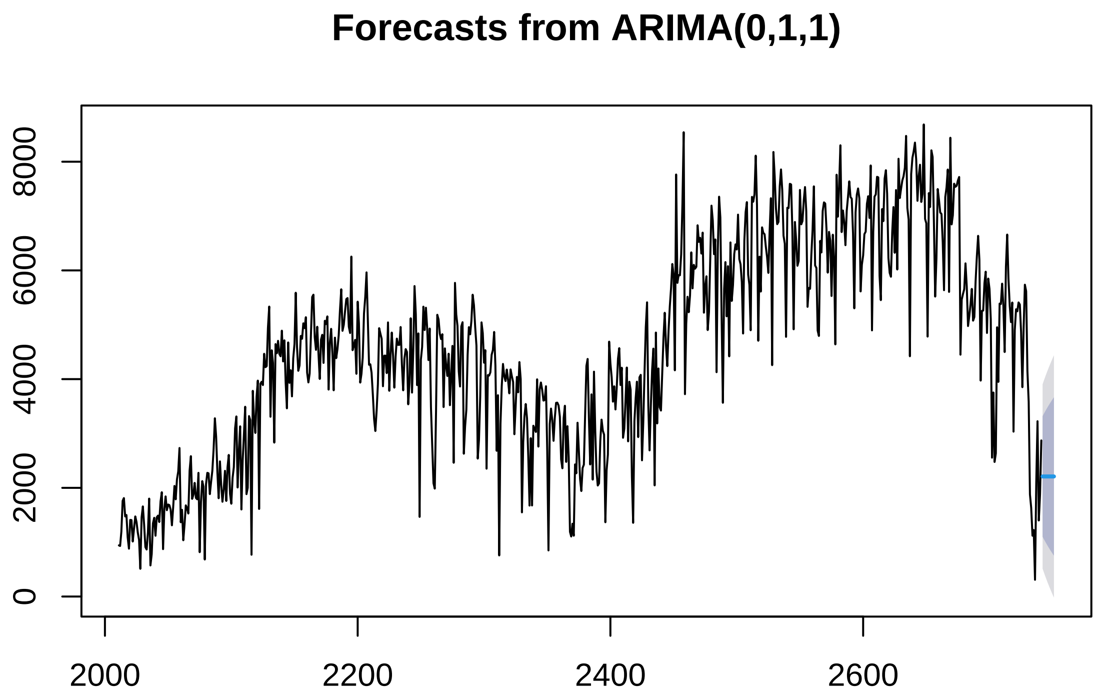
On trouve 3 models possible en regardant les PACF et les ACF:

MA(1):

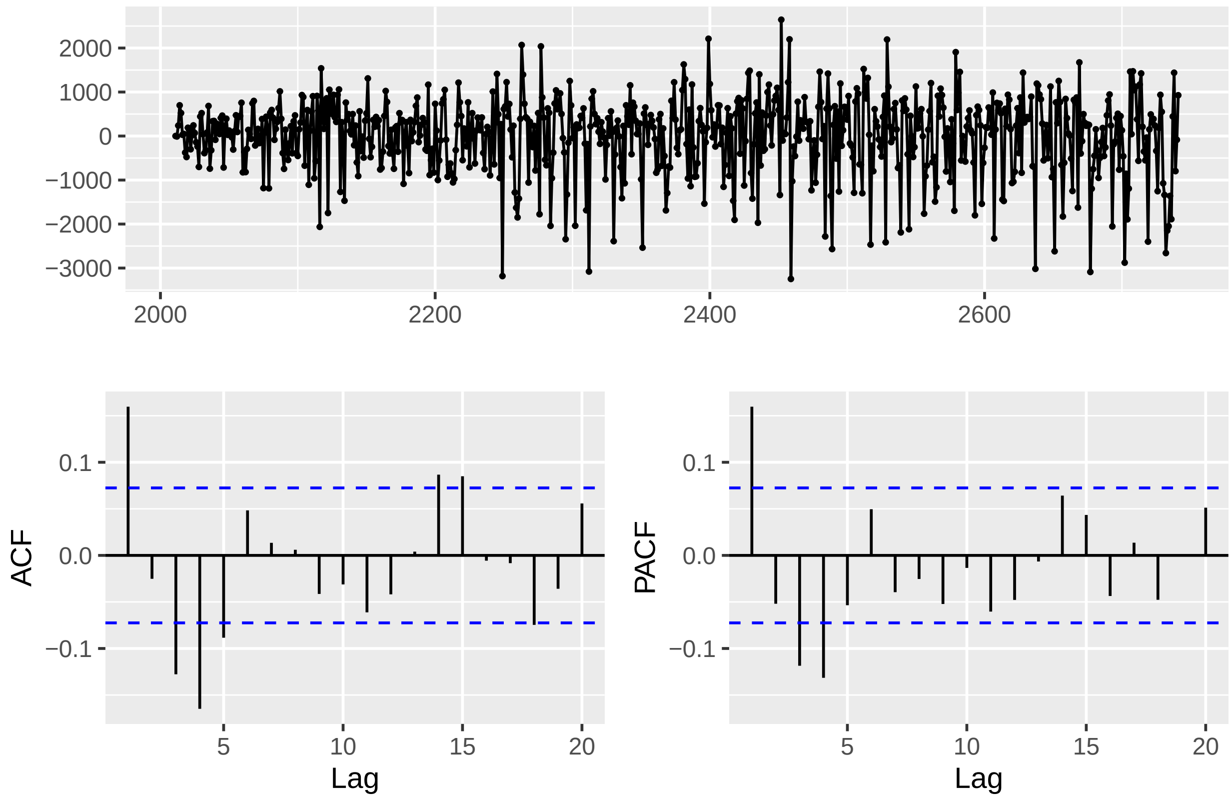
```
#PACF decroit vers 0 et 1 pic significatifs dans ACF => MA(1),
#ce qui donne ARIMA(0,1,1) car on a differencie
adj.ma1 <- arima(adjusted, order=c(0,1,1))
adj.ma1
```

```
##
## Call:
## arima(x = adjusted, order = c(0, 1, 1))
##
## Coefficients:
##           ma1
##       -0.7153
## s.e.   0.0389
##
## sigma^2 estimated as 748434:  log likelihood = -5973.08,  aic = 11950.16
```

```
adjforecasts.ma1 <- forecast(adj.ma1, h=10)
plot(adjforecasts.ma1)
```




```
ggtsdisplay(adjforecasts.ma1$residuals, lag =20)#les residuals sont bien un bruit blanc
```



Il n’y a que deux lags qui ne sont pas nuls dans les ACF.

```
Box.test(adjforecasts.ma1$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  adjforecasts.ma1$residuals
## X-squared = 83.558, df = 20, p-value = 9.692e-10
```

La p-value est superieure a 5%, on a bien un bruit blanc.

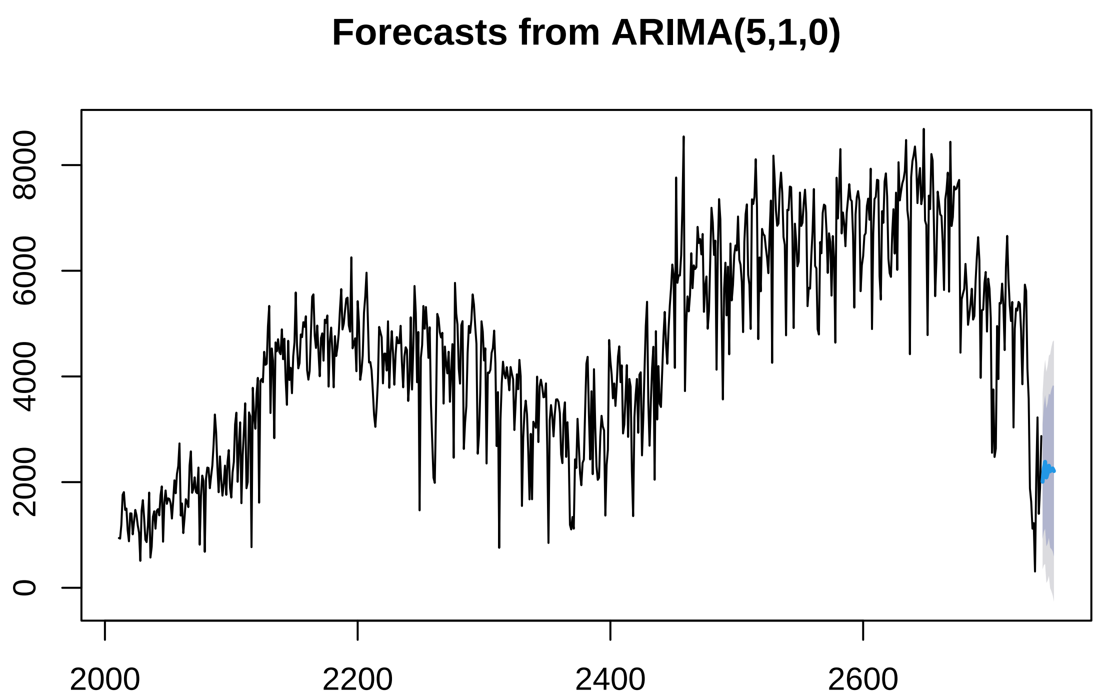
AR(5):

```
#ACF decroit vers 0 et 5 pics significatifs dans PACF => AR(5),
#ce qui donne ARIMA(5,1,0) car on a differencie
adj.ar5 <- arima(adjusted, order=c(5,1,0))
adj.ar5
```

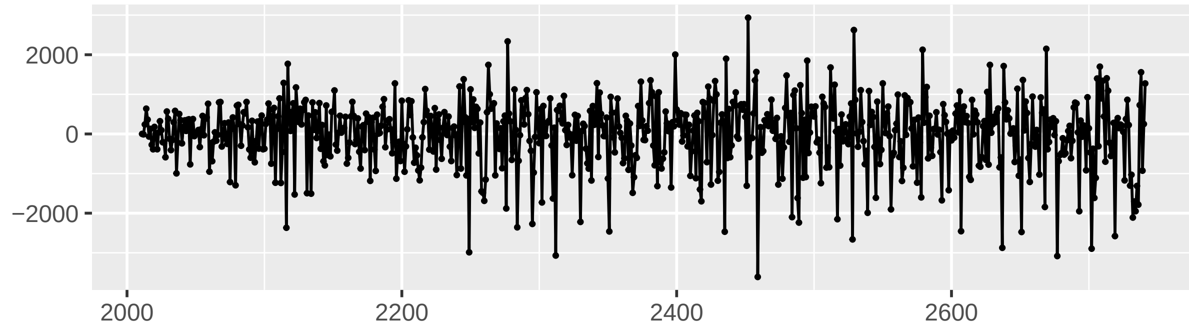
```
##
## Call:
## arima(x = adjusted, order = c(5, 1, 0))
##
```

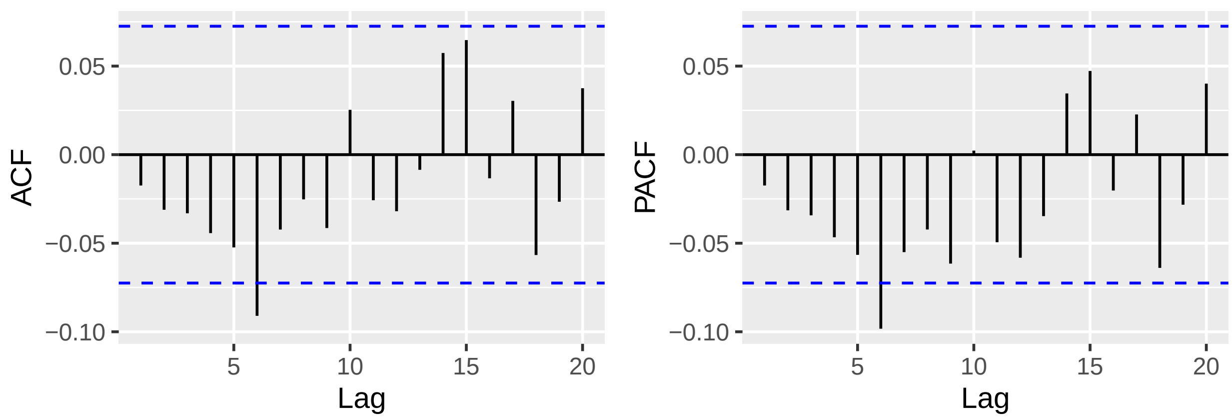
```
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.5348  -0.3767  -0.3230  -0.3092  -0.2211
## s.e.   0.0361   0.0396   0.0401   0.0396   0.0363
##
## sigma^2 estimated as 716420:  log likelihood = -5957.12,  aic = 11926.24
```

```
adjforecasts.ar5 <- forecast(adj.ar5, h=10)
plot(adjforecasts.ar5)
```




```
ggtsdisplay(adjforecasts.ar5$residuals, lag =20)
```






```
Box.test(adjforecasts.ar5$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  adjforecasts.ar5$residuals
## X-squared = 26.611, df = 20, p-value = 0.1466
```

Les residuals sont bien un bruit blanc

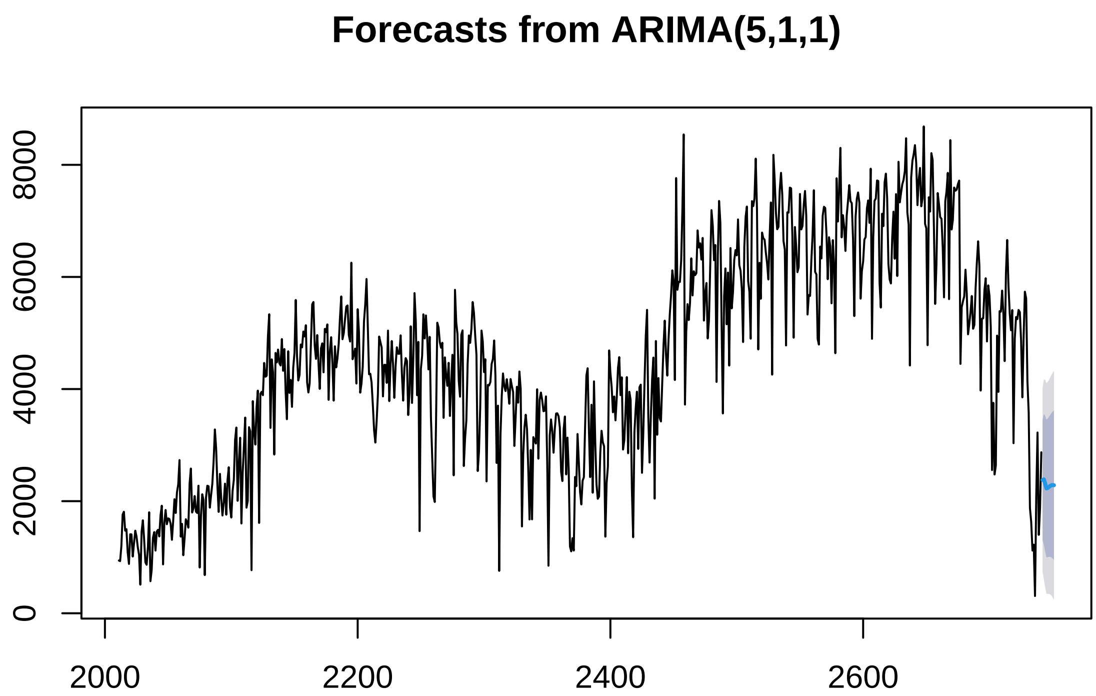
ARIMA(5,1,1):

```
adj.arima <- arima(adjusted, order=c(5,1,1))
adj.arima
```

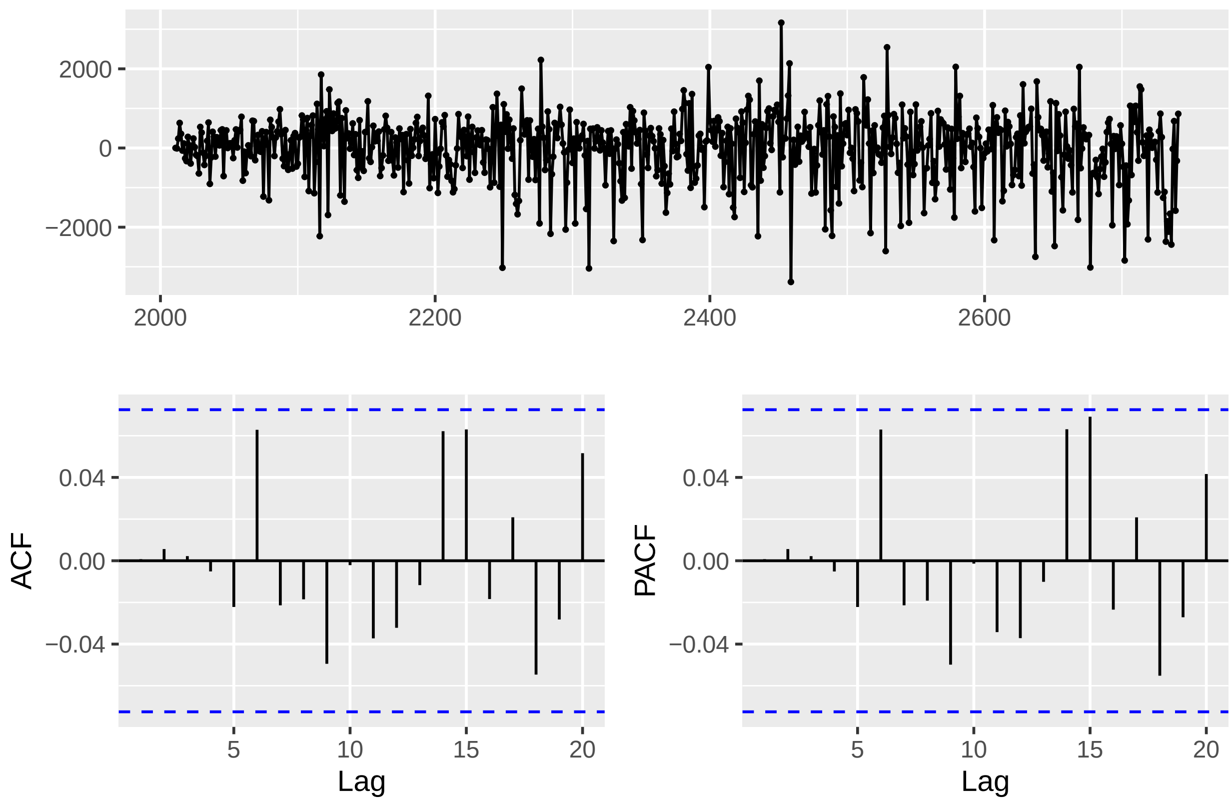
```
##
## Call:
## arima(x = adjusted, order = c(5, 1, 1))
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1
##       0.2042  -0.0071  -0.0745  -0.0995  -0.0242  -0.7809
## s.e.  0.0836   0.0550   0.0474   0.0474   0.0517   0.0754
```

```
##
## sigma^2 estimated as 702069:  log likelihood = -5949.83,  aic = 11913.66
```

```
adjforecasts3 <- forecast(adj.arima, h=10)
plot(adjforecasts3)
```




```
ggtsdisplay(adj.arima$residuals, lag =20)#les residuals sont bien un bruit blanc
```




```
Box.test(adj.arima$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  adj.arima$residuals
## X-squared = 18.935, df = 20, p-value = 0.5261
```

Si on cherche le loglikelihood le plus eleve et minimiser le AIC, le 3eme ARIMA(5,1,1) est le plus approprié, ses résidus se rapprochent le plus d'un bruit blanc. Si on veut se fier au principe de parsimonie, on prendra plutot le MA(1).

4)II)

On regarde le modele donné par auto.arima():

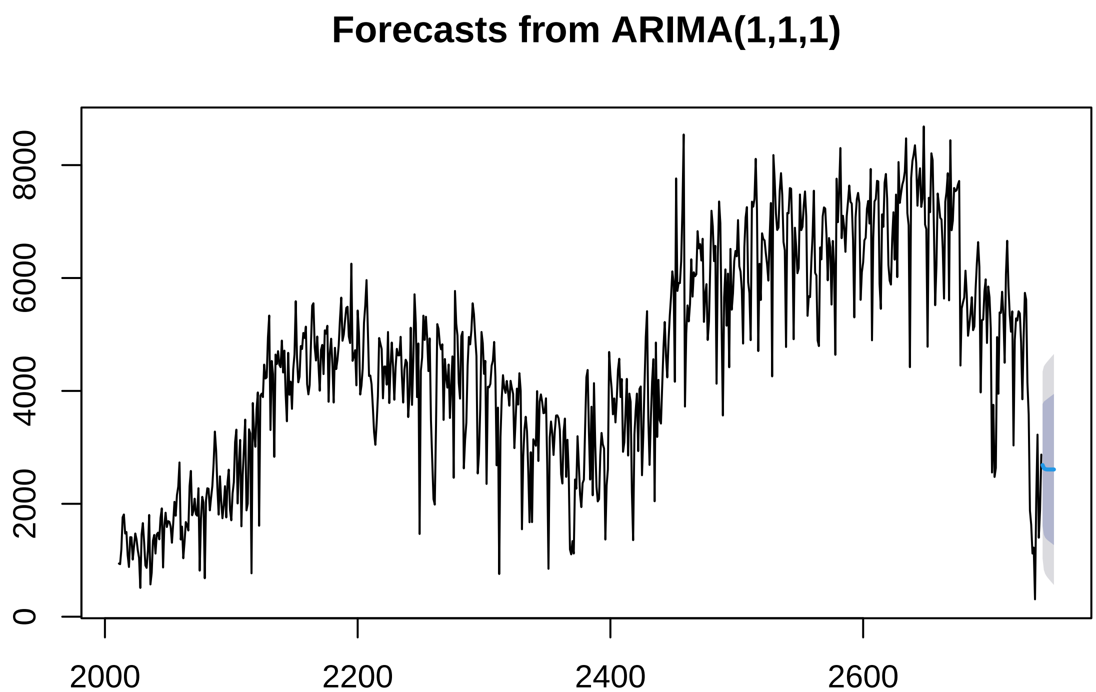
```
autoadj <- auto.arima(adjusted, seasonal=FALSE)
autoadj
```

```
## Series: adjusted
## ARIMA(1,1,1)
##
## Coefficients:
```

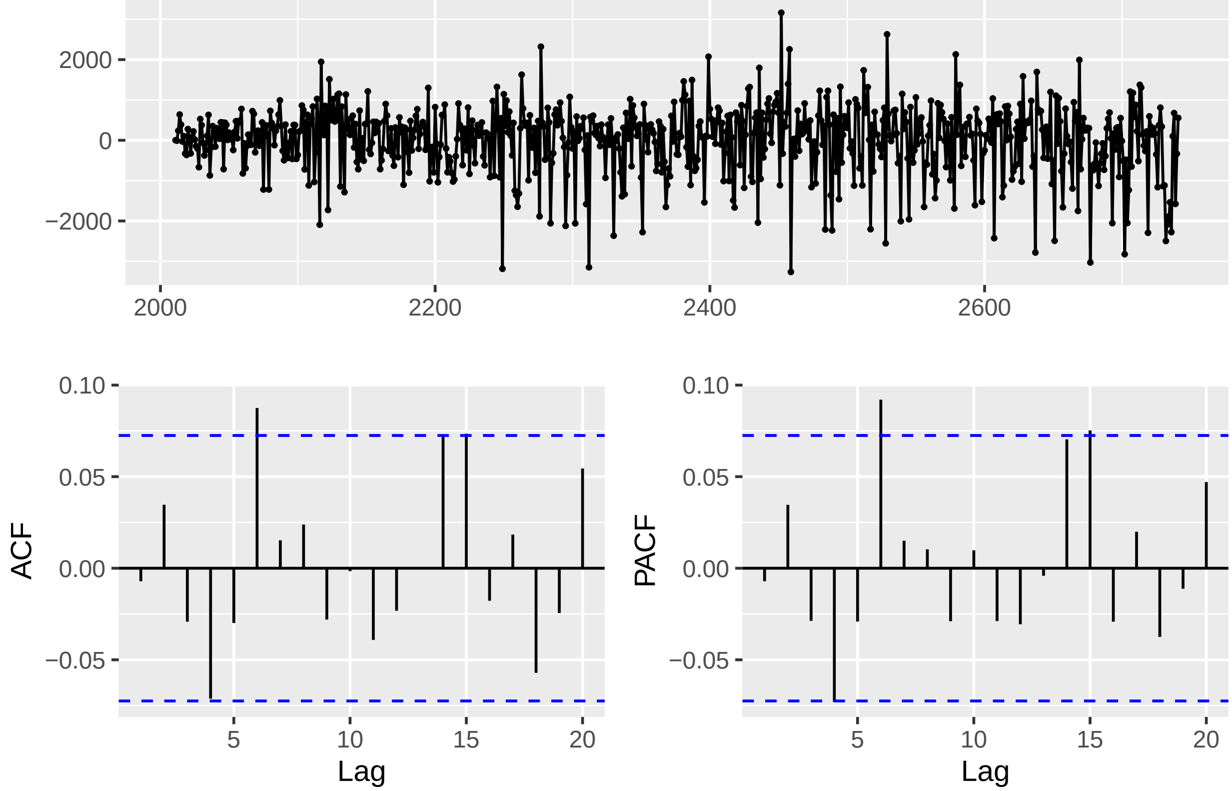
```
##          ar1      ma1
##       0.2976  -0.8640
## s.e.  0.0448   0.0225
##
## sigma^2 = 711338:  log likelihood = -5953.6
## AIC=11913.2   AICc=11913.23   BIC=11926.98
```

L'Auto Arima nous recommande un ARIMA(1,1,1)

```
adjforecasts4 <- forecast(autoadj, h=10)
plot(adjforecasts4)
```




```
ggtsdisplay(adjforecasts4$residuals, lag =20)#les residuals sont bien un bruit blanc
```




```r
Box.test(adjforecasts4$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  adjforecasts4$residuals
## X-squared = 27.938, df = 20, p-value = 0.1109
```

Les résiduels semblent bien etre un bruit blanc.

On compare le modele de auto.arima(), ARIMA(1,1,1), au modele chosit precedemment:

ARIMA(1,1,1) : log likelihood = -5953.6, AIC = 11913.2 ARIMA(5,1,1) : log likelihood = -5949.83, AIC = 11913.66

En termes de log likelihood et AIC ils sont tres similaires, donc on prend le modele avec le moins de parametres,c'est a dire ARIMA(1,1,1).

4)III) Evaluer et Itérer:

Comme vu précédemment, on avait trouvéun auto Arima peu complexe mais ses résidus ne formaient pas spécialement un bruit blanc, On peut donc essayer d'améliorer notre modele: On va commencer par entrainer 100 modeles ARIMA en iterant sur les valeurs de p et de q:

```r
aic.valuesq <- c()
for (p in c(0:9)){
  for  (q in (0:9)){
    deseasonal_cnt.arima <- arima(adjusted, order = c(p,1,q))
    aic.valuesq <- c(aic.valuesq, deseasonal_cnt.arima$aic)
}}

which.min(aic.valuesq)
```

```
## [1] 78
```

```r
print(aic.valuesq[78])
```

```
## [1] 11903.36
```

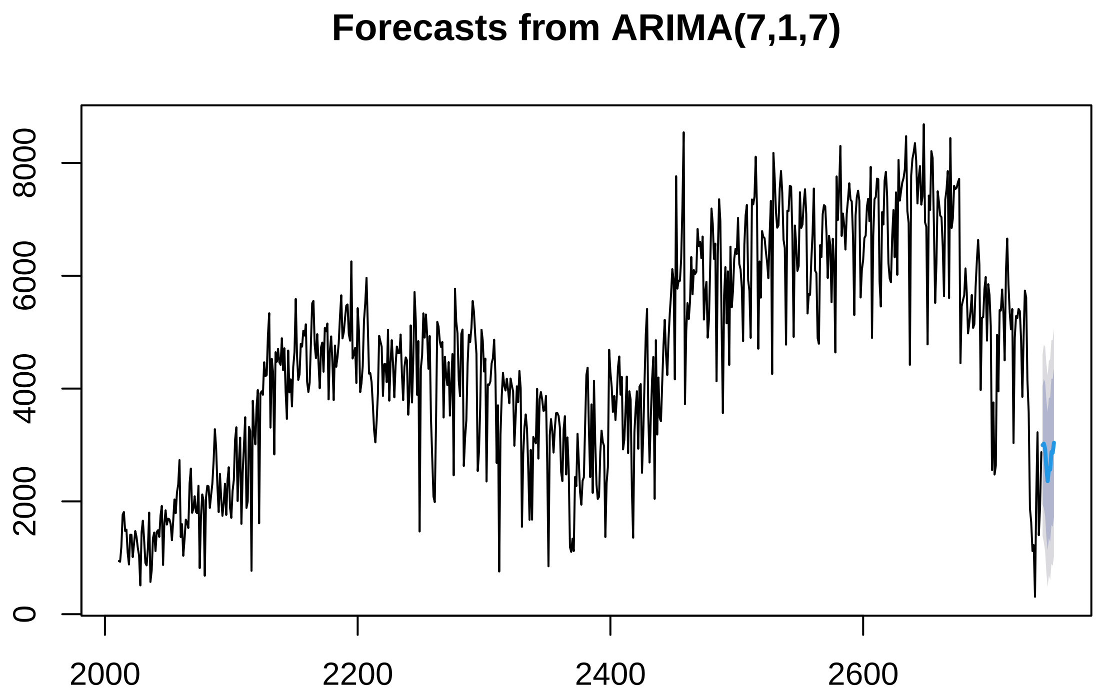
```r
#q en premier aic min = 11903.39
```

On trouve que les parametres qui minimisent l'AIC sont p=7 et q=7. Notre modele est donc un ARIMA(7,1,7)

```r
cnt.arima15 <- arima(adjusted, order = c(7,1,7))
cnt.arima15
```

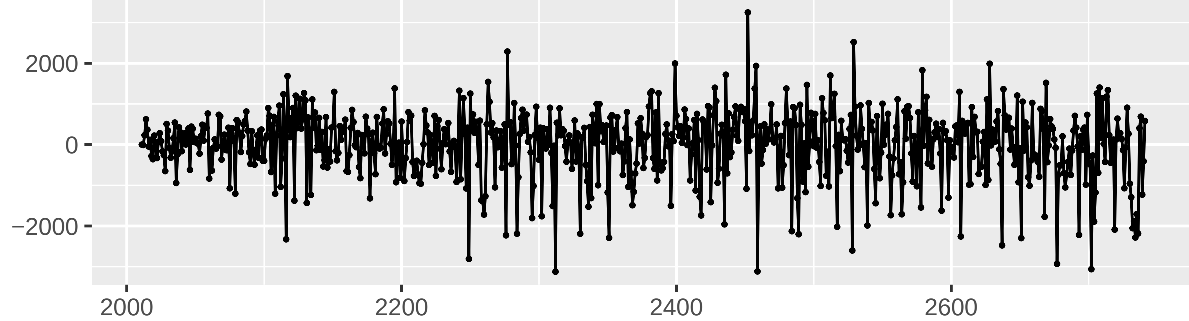
```
##
## Call:
## arima(x = adjusted, order = c(7, 1, 7))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6     ar7      ma1
##       -0.4307  -0.1673  -0.7504  -0.3065  -0.4638  -0.5681  0.2829  -0.1367
## s.e.   0.0762   0.1369   0.0774   0.1030   0.1089   0.1169  0.0529   0.0702
##           ma2     ma3      ma4     ma5     ma6      ma7
##       -0.2202  0.5450  -0.2235  0.1614  0.3136  -0.7723
## s.e.   0.1084  0.0734   0.1134  0.1225  0.0724   0.0934
##
## sigma^2 estimated as 671920:  log likelihood = -5936.68,  aic = 11903.36
```

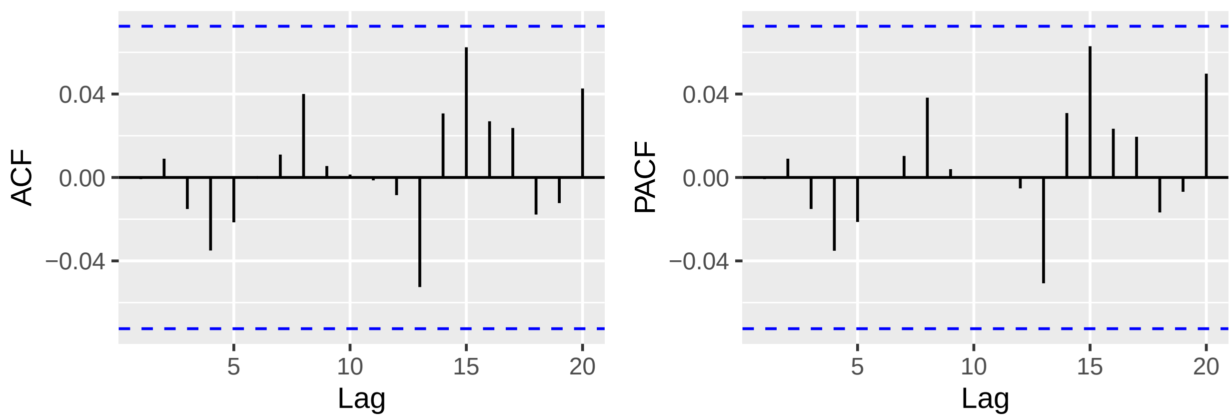
```r
adjforecasts15 <- forecast(cnt.arima15, h=10)
plot(adjforecasts15)
```



On peut apercevoir des variations dans la courbe du forecast et pas juste une ligne droite, ce qui fait apparaitre le forecast plus naturel.

```
ggtsdisplay(adjforecasts15$residuals, lag =20)#les residuals sont bien un bruit blanc
```






```
Box.test(adjforecasts15$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  adjforecasts15$residuals
## X-squared = 11.195, df = 20, p-value = 0.941
```

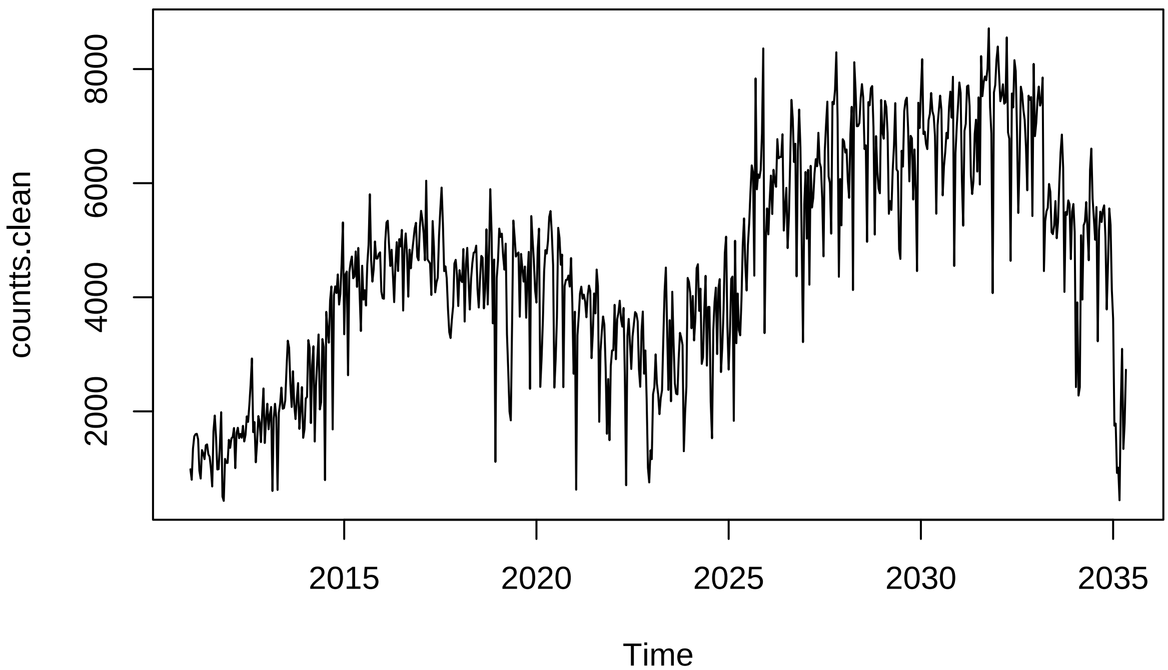
Ce modele a des residus qui se rapprochent le plus d'un bruit blanc.

4)VI)

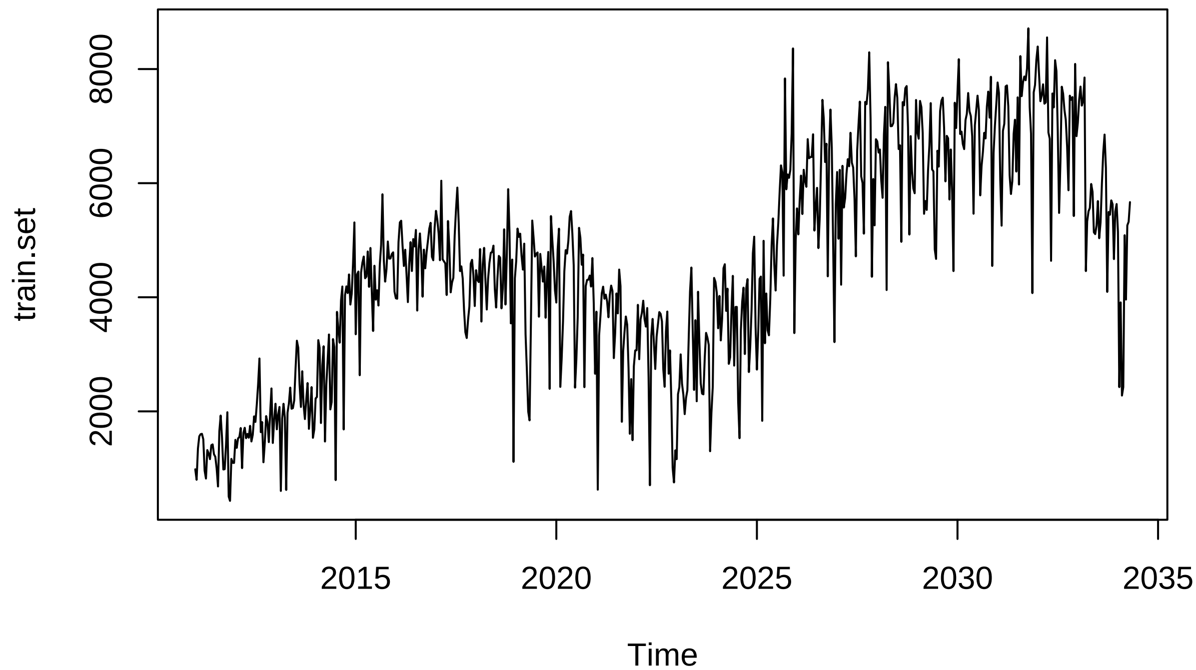
On divise les datas en deux ensembles :

```
end.time = time(countts.clean)[700]
train.set <- window(countts.clean, end=end.time)
test.set <- window(countts.clean, start=end.time)

plot.ts(countts.clean)
```

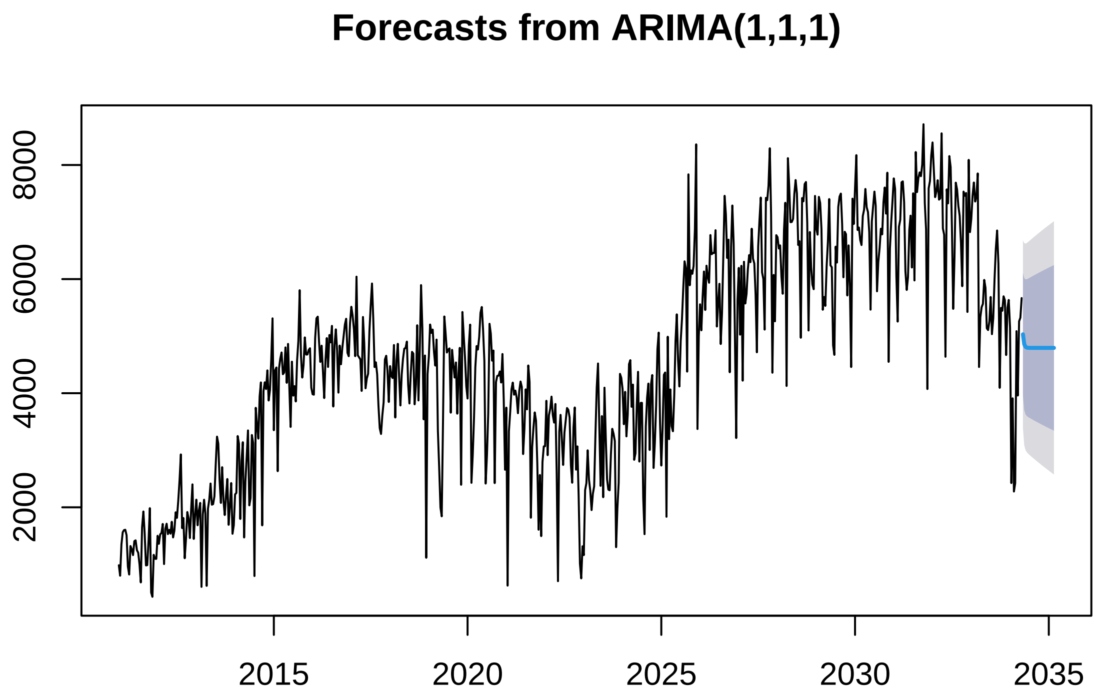



```
plot.ts(train.set)
```



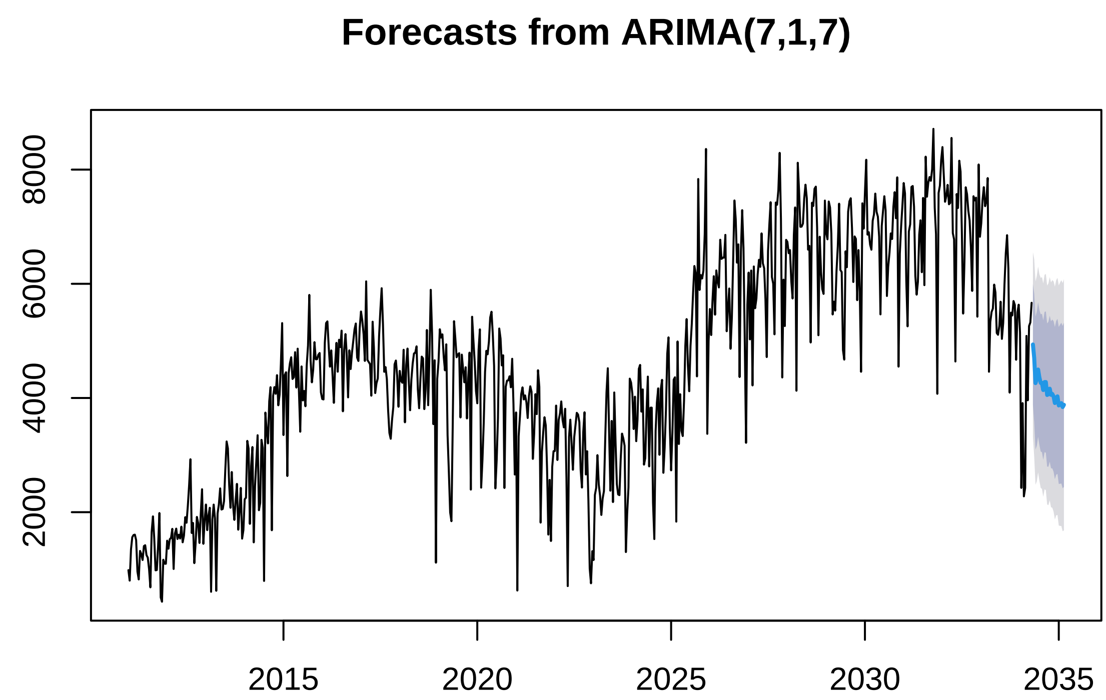
On fit le modele de auto.arima() sur l'ensemble de test et on fait un forecast sur les 25 prochaines valeurs:

```
auto <- auto.arima(train.set)
autoforecast <- forecast(auto, h=25)
plot(autoforecast)
```



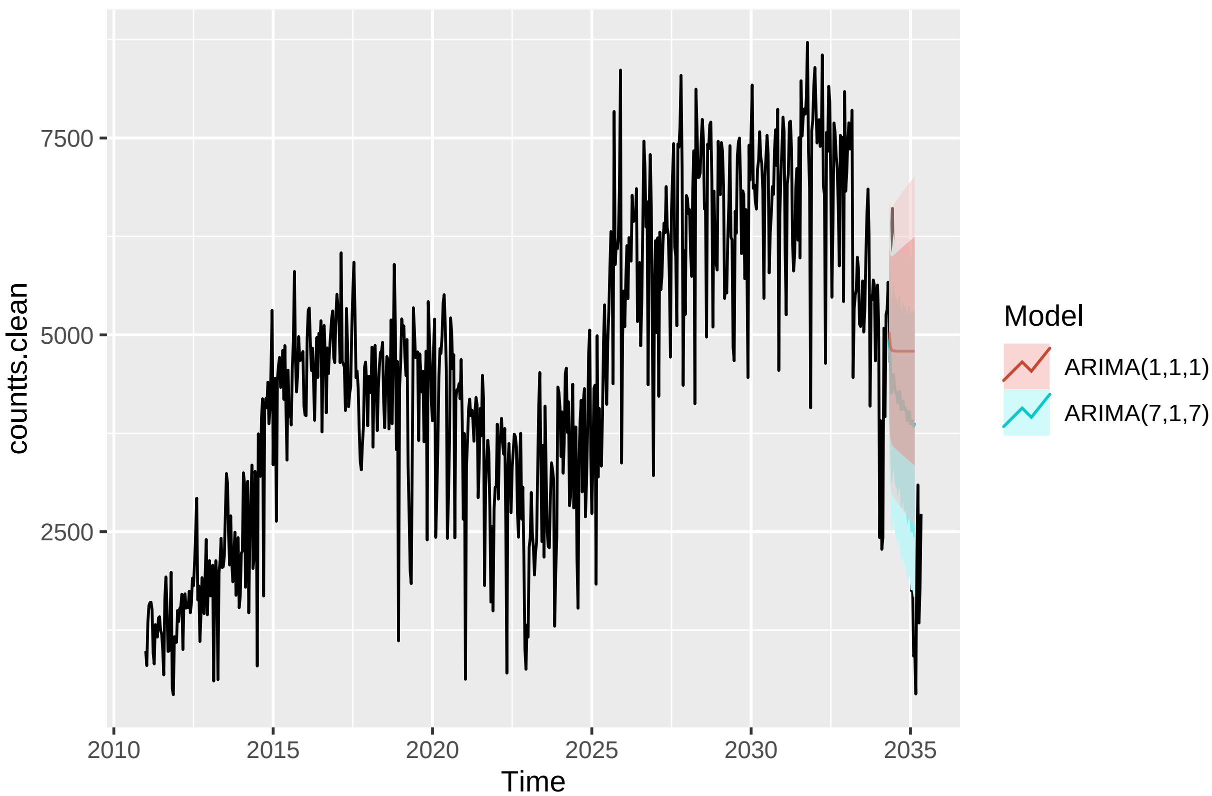
On fit le modele ARIMA(7,1,7) sur l'ensemble de test et on fait un forecast sur les 25 prochaines valeurs:

```
manual <- arima(train.set, order=c(7,1,7))
manualforecast <- forecast(manual, h=25)
plot(manualforecast)
```



La time series originale:

```
autoplot(countts.clean) +
  autolayer(manualforecast, series = "ARIMA(7,1,7)", alpha = 1) +
  autolayer(autoforecast, series = "ARIMA(1,1,1)", alpha = 0.5) +
  guides(colour = guide_legend("Model"))
```



Les deux modeles donnent des courbes de prediction qui sont plus elevées que la time series originale.

Cependant, On peut voir que Arima(7,1,7) se rapproche plus de la time series originale que Arima(1,1,1)

Interessons nous maintenant a l'accuracy des deux modeles:

```
accuracy(auto)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 31.70399 839.1331 620.0776 -5.391477 19.36547 0.5692123
##                      ACF1
## Training set -0.004311416
```

```
accuracy(manual)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 9.905327 824.8849 614.2262 -5.751478 19.32332 0.8833095
##                     ACF1
## Training set 0.002037951
```

On a un RMSE minimal dans le cas du modele manuel, ce qui veut dire qu'on a moins d'erreurs entre les forecast et les valeurs originales.

On remarque que le modele manuel donne une courbe qui a une allure plus naturelle alors que auto.arima() donne une courbe plus lisse. Par ailleurs les modeles on une difficulte a predire les valeurs eloignes dans le temps. En effect, ils devienent moins precis et l'intervalle de confiance s'elargit de plus en plus.

On prefera quand meme prendre le modele ARIMA(1,1,1) par principe de parsimonie, et pour éviter l'overfitting.