TP: Locations de vélos avec les séries temporelles

Khaldi Jalal Amegah Godwin

2022-11-19

```
if(!require("ggplot2")) install.packages("ggplot2")
```

```
## Loading required package: ggplot2
```

```
library("ggplot2")
if(!require("tidyverse"))install.packages("tidyverse",dependencies = TRUE)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("tidyverse")
if(!require("forecast")) install.packages("forecast", dependencies = TRUE)
```

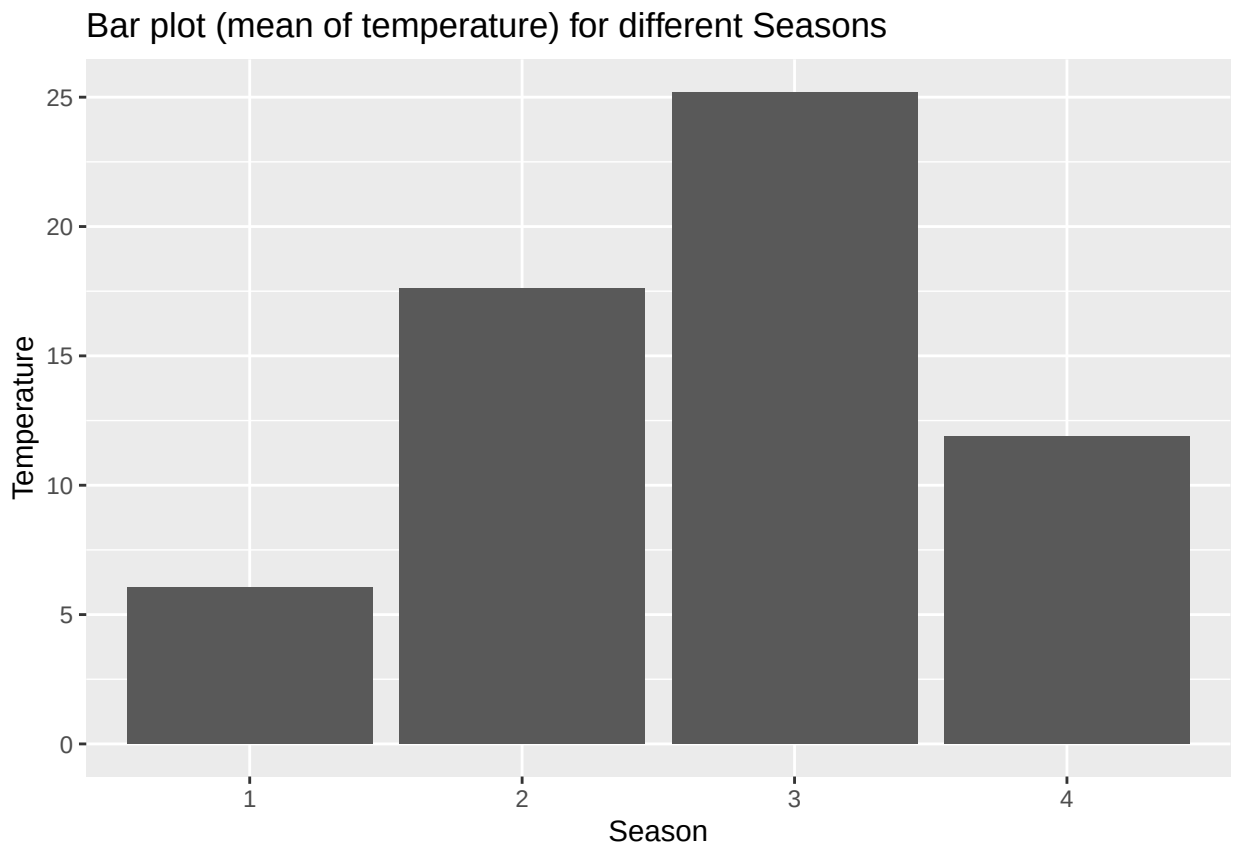
```
## Loading required package: forecast
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library("forecast")
```

```
 t_min<--8;t_max<-39 # t_min, t_max pour la première normalisation
dataset_hour <-read.csv("./datasets/hour.csv",stringsAsFactors = TRUE)
dataset_hour$season <-as.factor(dataset_hour$season)
dataset_hour$temp_wo_n <- (t_max-t_min)*dataset_hour$temp+t_min #température non normalisée
```

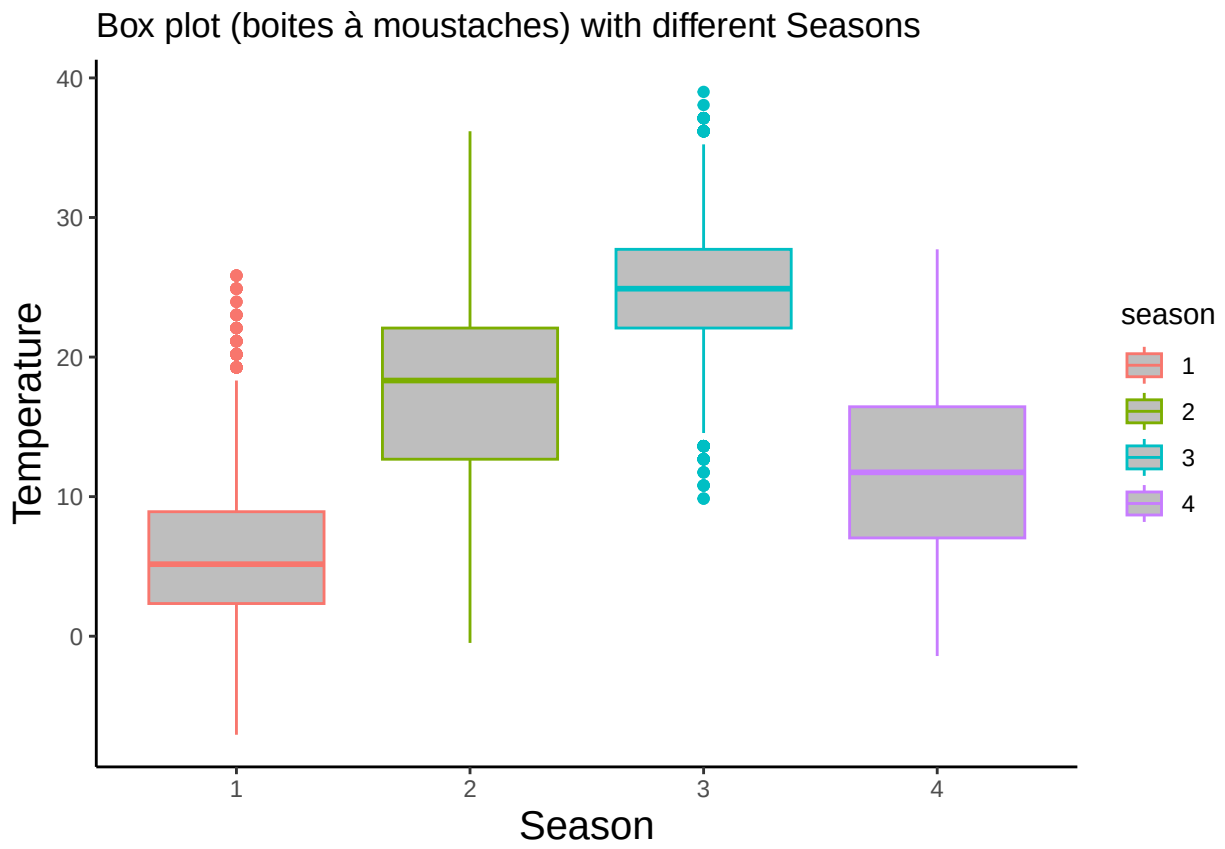
```
graph_mean <- ggplot(dataset_hour,aes(x=season,y=temp_wo_n))+
  geom_bar(stat = "summary") +
  labs(
  title    = "Bar plot (mean of temperature) for different Seasons"
 )+
  xlab("Season") + ylab("Temperature")
graph_mean
```

```
## No summary function supplied, defaulting to `mean_se()`
```




```
graph_box <- ggplot(dataset_hour,aes(x=season,y=temp_wo_n,color=season))+ geom_boxplot(fill='gray') + la
theme_classic()+
theme(
  axis.title = element_text(size=15)
)+
labs(
  title    = "Box plot (boites à moustaches) with different Seasons"
)+
xlab("Season") + ylab("Temperature")

graph_box
```



1. Examine your data On peut voir que boite_a_moustache(temp(printemps)) < boite_a_moustache(temp(été)) < boite_a_moustache(temp(automne)) et boite_a_moustache(temp(printemps)) < boite_a_moustache(temp(hiver)) < boite_a_moustache(temp(été)):
   En regardant le diagramme à barre qui représente les moyennes des températures des saisons, on voit que moy(normalisée)_printemps < moy(normalisée)_été < moy_autonme(normalisée) et moy(normalisée)_printemps < moy(normalisée)_hiver < moy(normalisée)_été De plus en traçant les boites à moustache, les distributions suivent cette ordre. Les boites à moustaches sont des indicateurs statistiques très fiables (plus que la moyenne).

```
mean <- mean(dataset_hour$temp_wo_n)
med <- median(dataset_hour$temp_wo_n)
sprintf("Moyenne: %f et Médiane: %f",mean,med)
```

```
## [1] "Moyenne: 15.358397 et Médiane: 15.500000"
```

Remarque: Moyenne ~ Médiane donc distribution empirique de la température quasi symétrique.

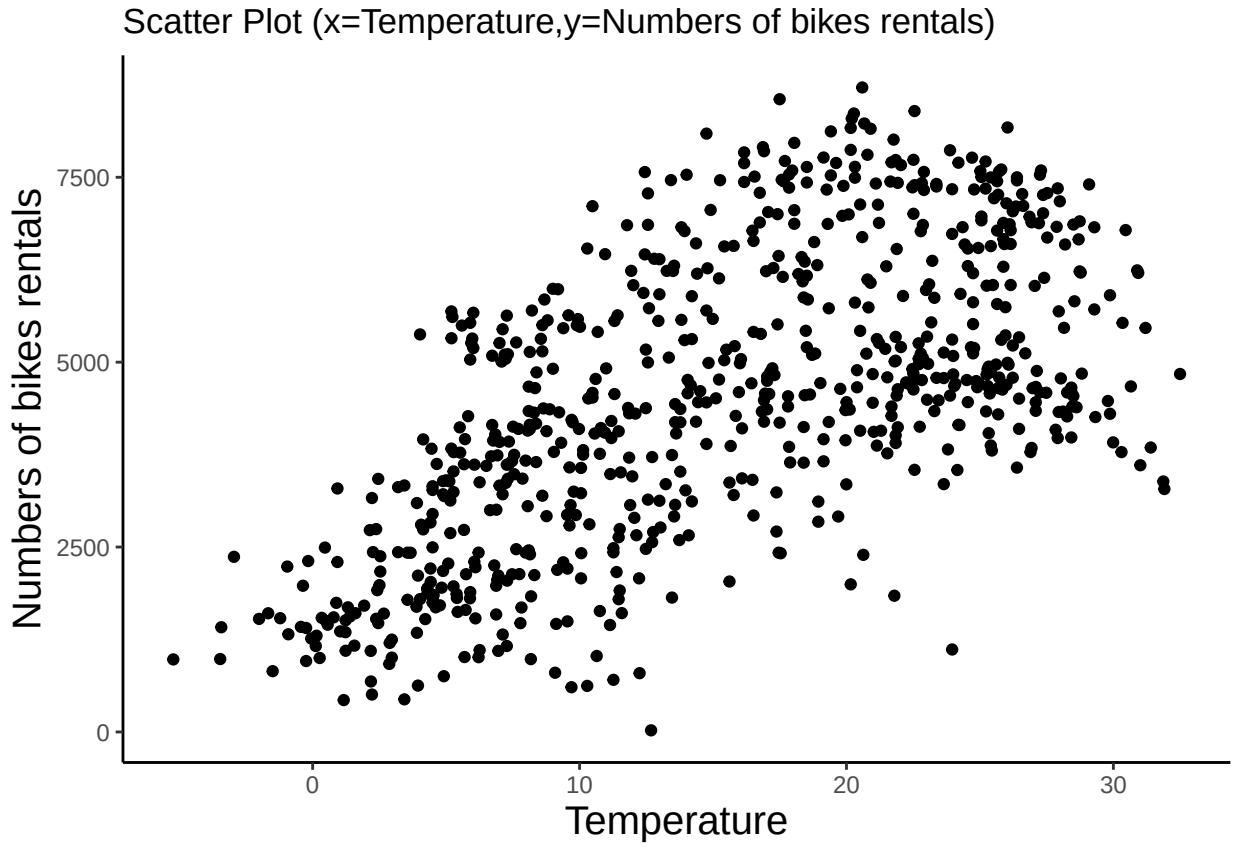
```
t_min<--8;t_max<-39 # t_min, t_max pour la première normalisation
dataset_day <-read.csv("./datasets/day.csv",stringsAsFactors = TRUE)
dataset_day$season <-as.factor(dataset_day$season)
dataset_day$temp_wo_n <- (t_max-t_min)*dataset_day$temp+t_min #température non normalisée
head(dataset_day)
```

```
##   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 temp_wo_n
## 1 0.344167 0.363625 0.805833 0.1604460    331        654  985  8.175849
## 2 0.363478 0.353739 0.696087 0.2485390    131        670  801  9.083466
## 3 0.196364 0.189405 0.437273 0.2483090    120       1229 1349  1.229108
## 4 0.200000 0.212122 0.590435 0.1602960    108       1454 1562  1.400000
## 5 0.226957 0.229270 0.436957 0.1869000     82       1518 1600  2.666979
## 6 0.204348 0.233209 0.518261 0.0895652     88       1518 1606  1.604356
```

```
graph_scatter <- ggplot(dataset_day,aes(x=temp_wo_n,y=cnt))+ geom_point() + labs(title='Title',size=222)
theme_classic()+
theme(
  axis.title = element_text(size=15)
)+
labs(
  title    = "Scatter Plot (x=Temperature,y=Numbers of bikes rentals)"
)+
xlab("Temperature") + ylab("Numbers of bikes rentals")

graph_scatter
```

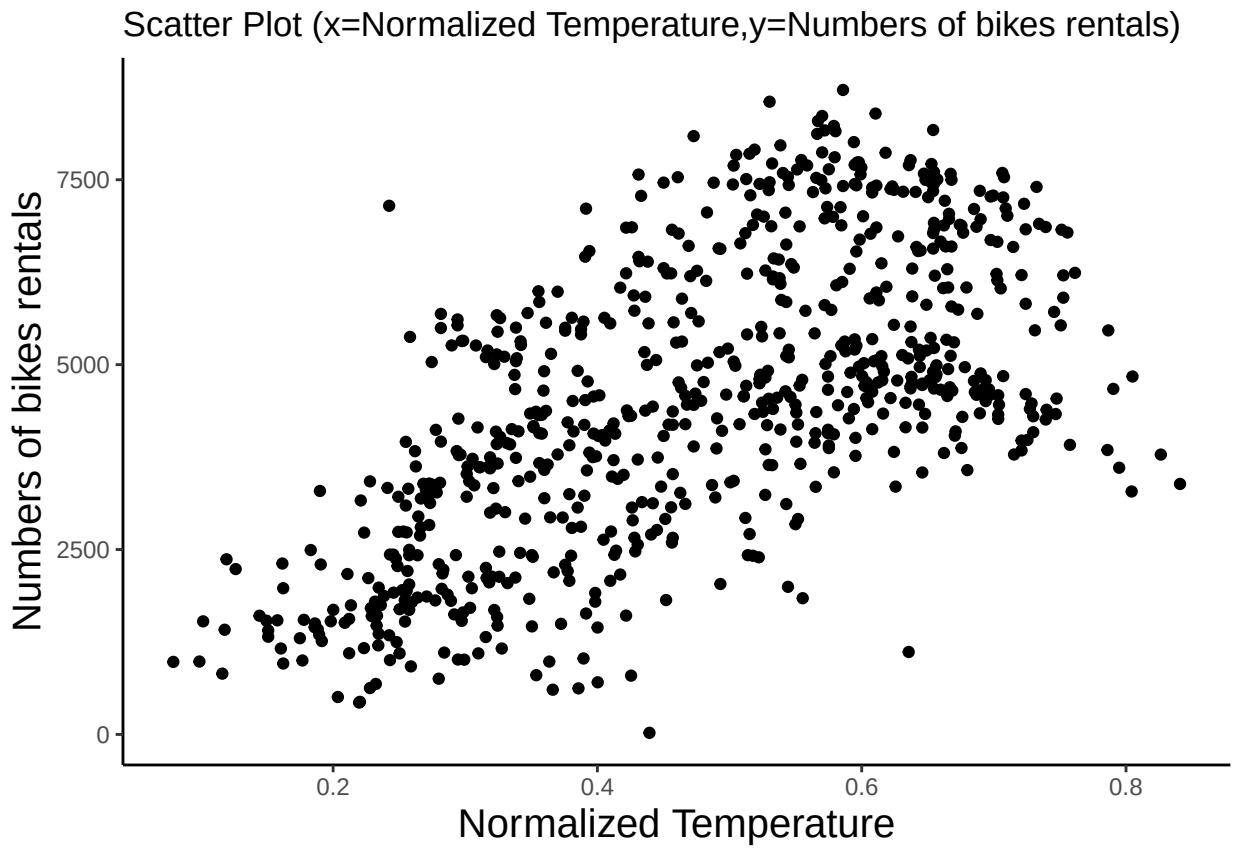


On peut voir qu'il y a une corrélation de type polynomiale du second degré (tendance polynomiale du second degré).

Interprétation: Quand il fait plus de 30°,le nombre de vélos loués diminuent. Pour des températures inférieure à 2°, il y a peu de vélos loués. On observe la distribution symétrique, autour de la moyenne (ou de le médiane) le nombre de vélos loués atteint son maximum. Tandis qu'à gauche ou à droite de la moyenne, on peut voir une décroissance linéaire (droite) du nombre de vélos loués.

```
graph_scatter_n1 <- ggplot(dataset_day,aes(x=atemp,y=cnt))+ geom_point() + labs(title='Title',size=222)+
theme_classic()+
theme(
  axis.title = element_text(size=15)
)+
labs(
  title    = "Scatter Plot (x=Normalized Temperature,y=Numbers of bikes rentals)"
)+
xlab("Normalized Temperature") + ylab("Numbers of bikes rentals")

graph_scatter_n1
```



Il n'est pas nécessaire de regarder les deux températures normalisées, car la tendance sera la meme pour la température non normalisée et les deux températures normalisées.

```
df_months <- as.tibble(dataset_hour)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
```

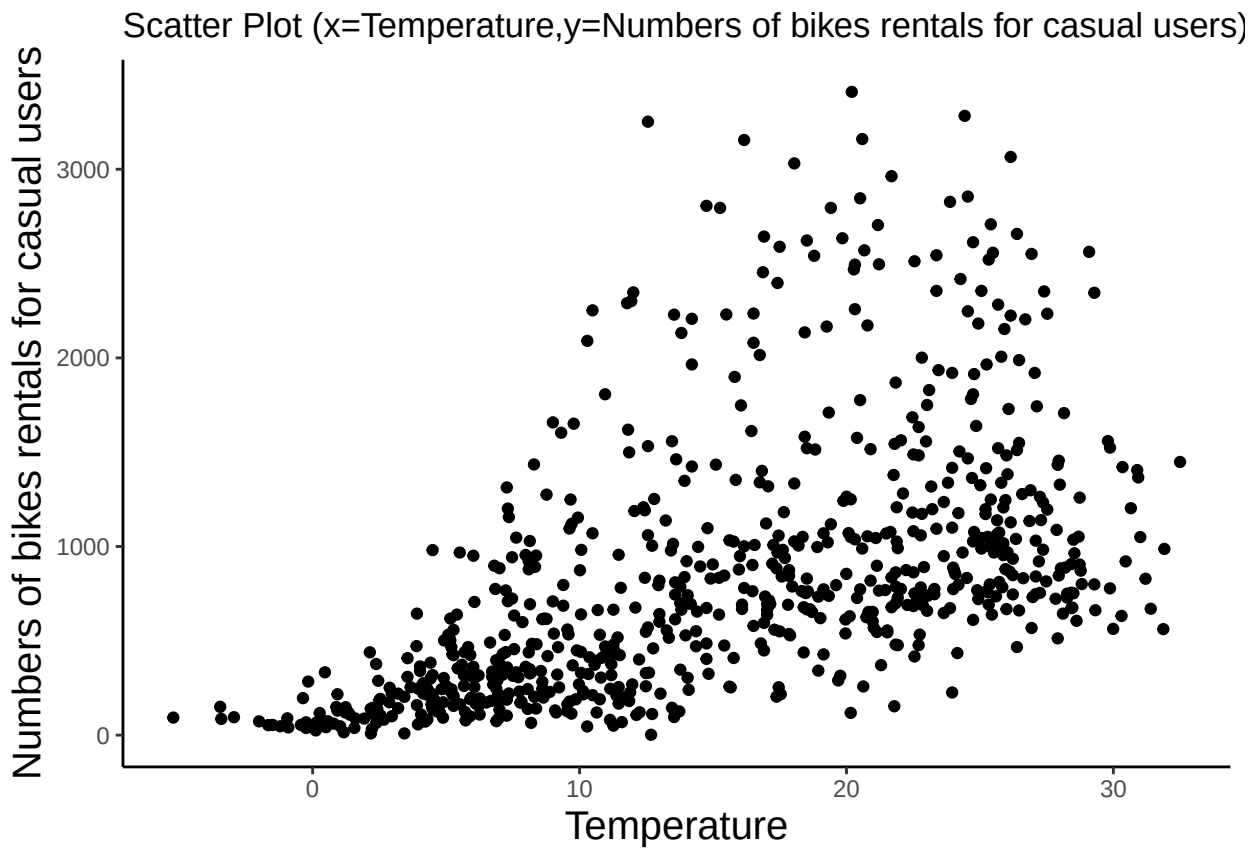
```
## i Please use ‘as_tibble()‘ instead.
## i The signature and semantics have changed, see ‘?as_tibble‘.
```

```
df_months %>%
  group_by(mnth) %>%# agregation by month
  summarise(mean_temp_wo_n=mean(temp_wo_n), #compute mean temp,hum, windspeed and sum of rentals
            mean_hum=mean(hum),
            mean_windspeed=mean(windspeed),
            sum_rentals=sum(cnt))
```

```
## # A tibble: 12 x 5
##     mnth mean_temp_wo_n mean_hum mean_windspeed sum_rentals
##    <int>          <dbl>    <dbl>          <dbl>       <int>
##  1     1           3.17    0.581          0.208      134933
##  2     2           6.11    0.567          0.216      151352
##  3     3          10.4     0.589          0.223      228920
##  4     4          14.1     0.588          0.234      269094
##  5     5          20.0     0.689          0.183      331686
##  6     6          24.2     0.576          0.185      346342
##  7     7          27.5     0.598          0.166      344948
##  8     8          25.3     0.637          0.171      351194
##  9     9          21.0     0.714          0.166      345991
## 10    10          14.9     0.689          0.172      322352
## 11    11           9.35    0.625          0.184      254831
## 12    12           7.24    0.666          0.177      211036
```

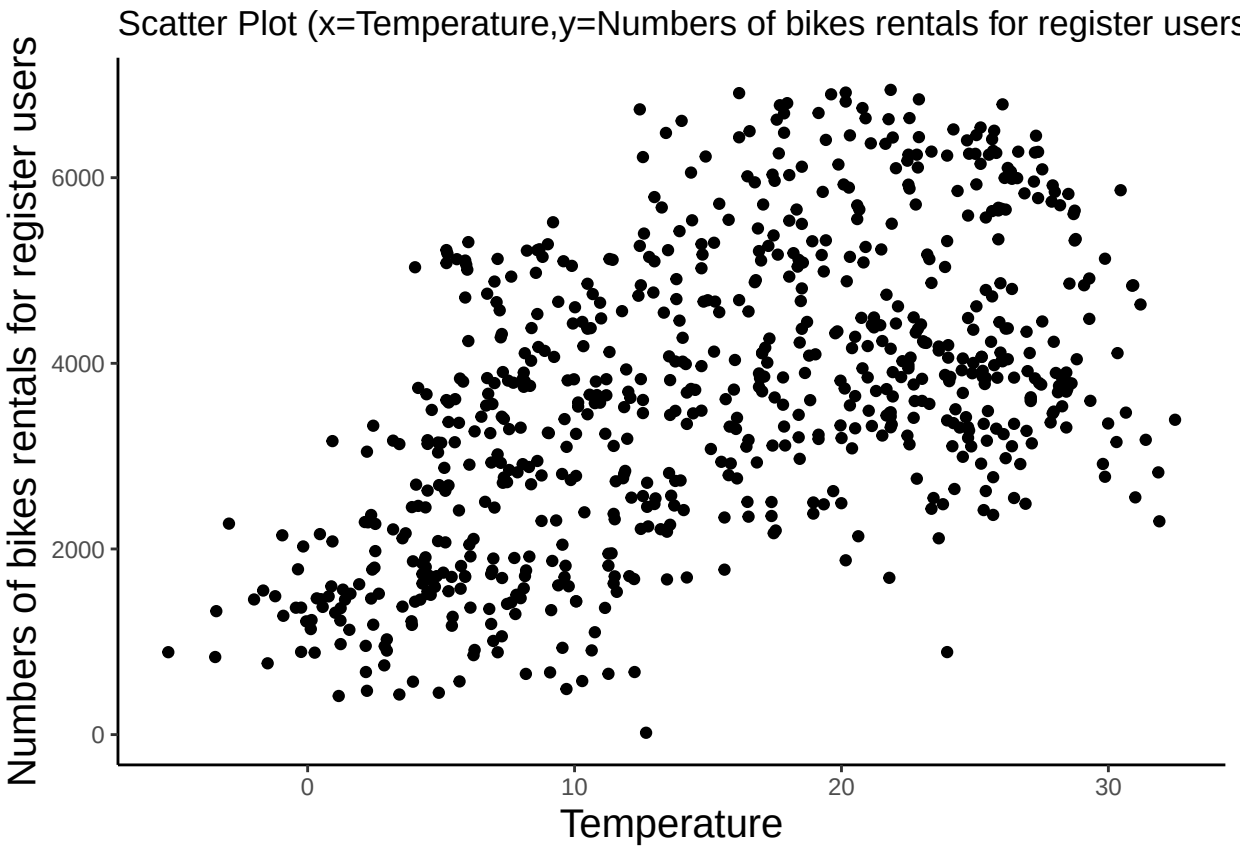
```
graph_scatter_casu <- ggplot(dataset_day,aes(x=temp_wo_n,y=casual))+ geom_point() + labs(title='Title',s
theme_classic()+
theme(
  axis.title = element_text(size=15)
)+
labs(
  title    = "Scatter Plot (x=Temperature,y=Numbers of bikes rentals for casual users)"
)+
xlab("Temperature") + ylab("Numbers of bikes rentals for casual users")

graph_scatter_casu
```



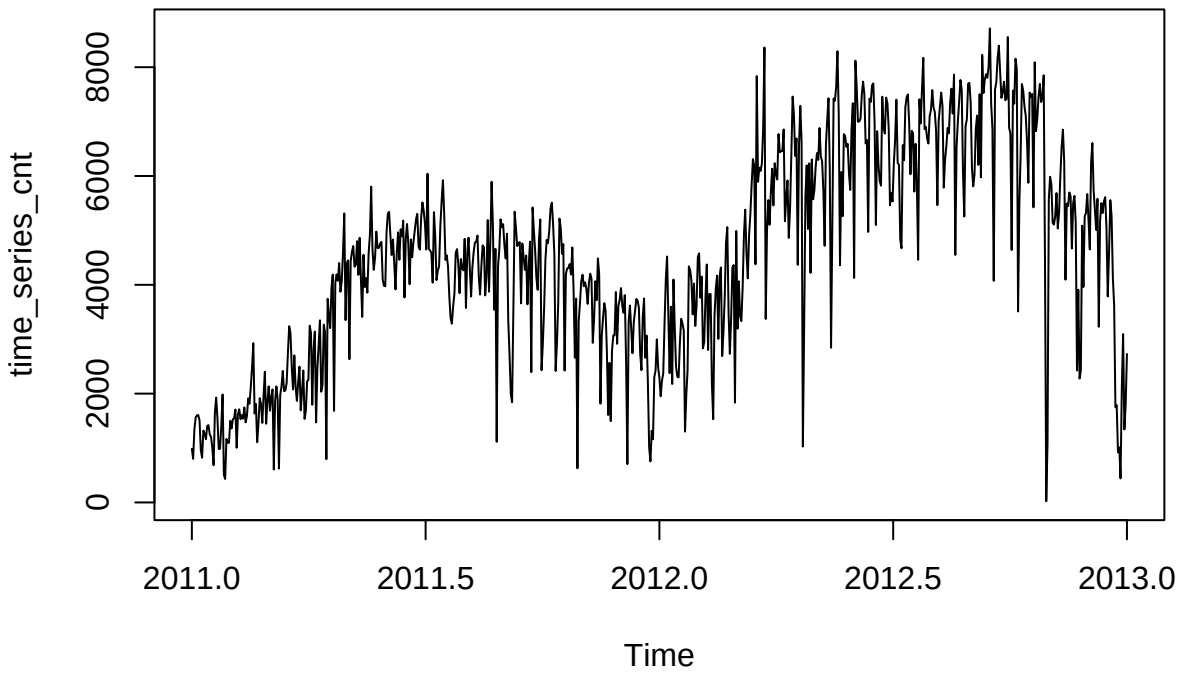

```
graph_scatter_regi <- ggplot(dataset_day,aes(x=temp_wo_n,y=registered))+ geom_point() + labs(title='Tit
theme_classic()+
theme(
  axis.title = element_text(size=15)
)+
labs(
  title    = "Scatter Plot (x=Temperature,y=Numbers of bikes rentals for register users)"
)+
xlab("Temperature") + ylab("Numbers of bikes rentals for register users")

graph_scatter_regi
```

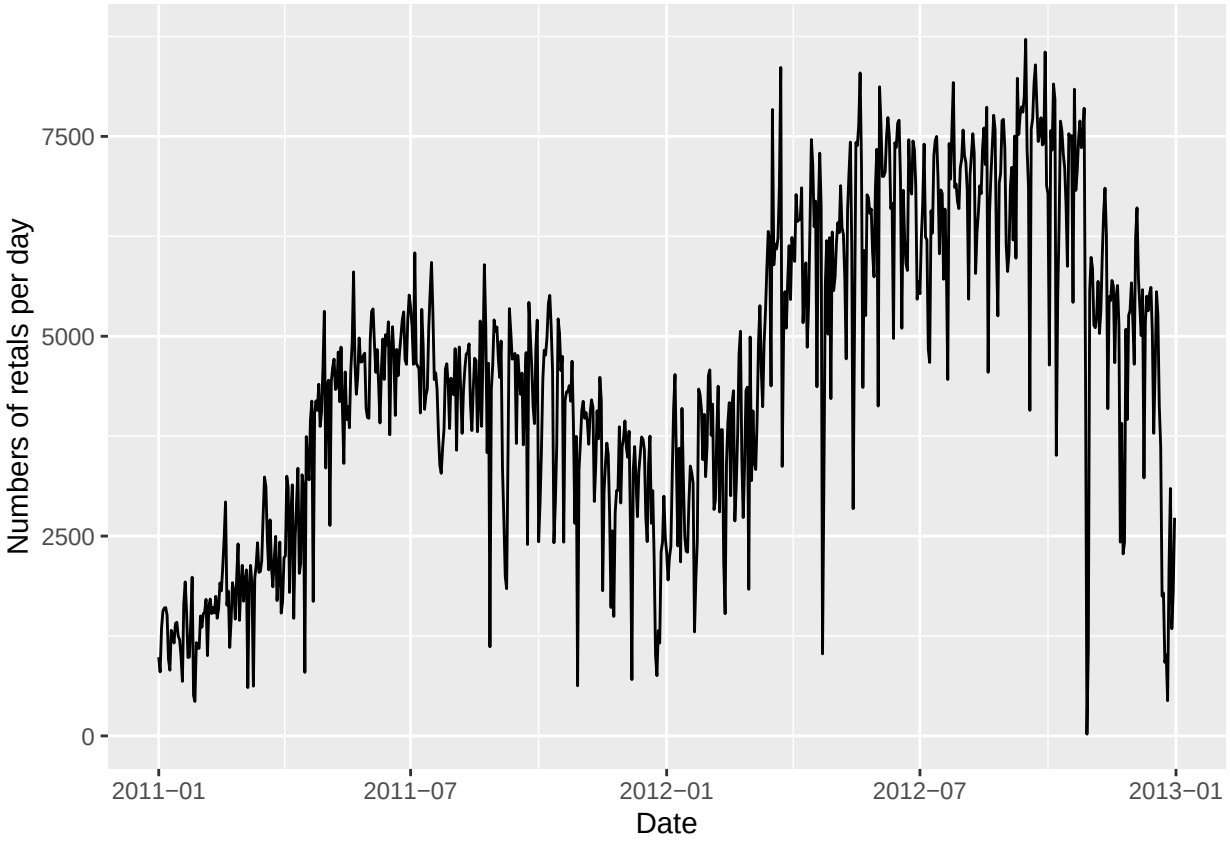


On observe deux choses: -Les personnes qui sont habitués à prendre un vélo ont leur habitude de consommation: Et donc une tendance globale de location de vélos se dégage et est observé dans le nuage de points(x=Température, y=Nombres de vélos loués pour les utilisateurs réguliers) -De plus, on observe la même tendance pour le nuage de points représentant le nombre total de vélos loués en fonction de la température. Sauf que les utilisateurs de vélos occasionnels engendrent du "bruit" qui ne permet pas de pouvoir distinguer la tendance (corrélation) aussi clairement que pour le nuage de points (x=Température, y=Nombres de vélos loués pour les utilisateurs réguliers). Donc on peut considérer, que les utilisateurs occasionnels louent un nombre aléatoires de vélos qui ne dépend pas forcément de la température ou d'une habitude définie des utilisateurs.

```
time_series_cnt <- ts(dataset_day$cnt,frequency=365,start=c(2011,1))
plot(time_series_cnt)
```




```
graph_line <- ggplot(dataset_day,aes(x=as.Date(dteday),y=cnt))+geom_line()+
  xlab("Date")+ylab("Numbers of retals per day")
graph_line
```



On voit très clairement une tendance haussière sur le long-terme et une saisonalité: -On voit qu'en période d'hiver, le nombres de vélos est à son minimum puis une augmentation en période de printemps et d'été (maximum atteint en été) puis un baisse en automne. puis chaque année le phénomène se reproduit (saisonalité). On voit quand même des irrégularités, beaucoup de fluctuations sur une période donnée (exemple: fin 2012) –> une quantité importante de bruit qui cause ces fluctuations.

```
# function qui compte le nombre de valeur manquante
count_na_func = function(x) sum(is.na(x))

# le nombre de valeur manquante au niveau des colonnes
sum_na.cols = dataset_day %>%
summarise_all(~sum(is.na(.)))

dataset_day_wo_na <-dataset_day %>%mutate(count_na = apply(., 1, count_na_func))
sum_na.cols
```

```
##   instant dteday season yr mnth holiday weekday workingday weathersit temp
## 1       0      0      0  0    0       0       0          0          0    0
##   atemp hum windspeed casual registered cnt temp_wo_n
## 1     0   0         0      0          0   0         0
```

```
# le nombre de NA en ligne: ici on crée un nouveau champ qui va contenir  le nombre de valeur maquante
```

On remarque qu'il n'y a pas de valeurs manquantes ni dans les colonnes, ni dans les lignes

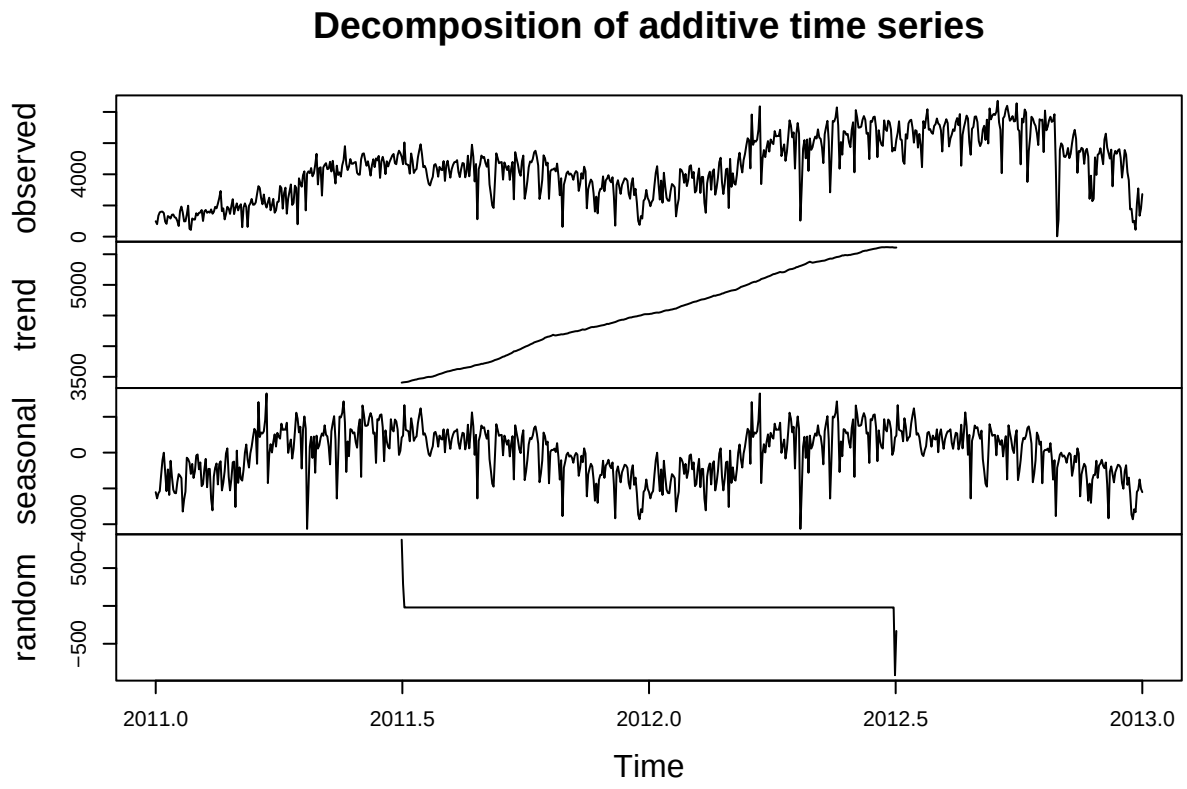
```r
  #return boolean df of non-outliers values
non_outliers <- function(df_colname){
  quantiles <-quantile(df_colname, probs = seq(0, 1, 1/4))
  condition_outliers <- !(quantiles[1]-1.5*(quantiles[3]-quantiles[1]) > df_colname | df_colname >    qu
  return(condition_outliers)
}
df_cnt_wo_outliers <- dataset_day[non_outliers(dataset_day$cnt),c("cnt")]
df_cnt_wo_outliers
```

```
##   [1]  985  801 1349 1562 1600 1606 1510  959  822 1321 1263 1162 1406 1421 1248
##  [16] 1204 1000  683 1650 1927 1543  981  986 1416 1985  506  431 1167 1098 1096
##  [31] 1501 1360 1526 1550 1708 1005 1623 1712 1530 1605 1538 1746 1472 1589 1913
##  [46] 1815 2115 2475 2927 1635 1812 1107 1450 1917 1807 1461 1969 2402 1446 1851
##  [61] 2134 1685 1944 2077  605 1872 2133 1891  623 1977 2132 2417 2046 2056 2192
##  [76] 2744 3239 3117 2471 2077 2703 2121 1865 2210 2496 1693 2028 2425 1536 1685
##  [91] 2227 2252 3249 3115 1795 2808 3141 1471 2455 2895 3348 2034 2162 3267 3126
## [106]  795 3744 3429 3204 3944 4189 1683 4036 4191 4073 4400 3872 4058 4595 5312
## [121] 3351 4401 4451 2633 4433 4608 4714 4333 4362 4803 4182 4864 4105 3409 4553
## [136] 3958 4123 3855 4575 4917 5805 4660 4274 4492 4978 4677 4679 4758 4788 4098
## [151] 3982 3974 4968 5312 5342 4906 4548 4833 4401 3915 4586 4966 4460 5020 4891
## [166] 5180 3767 4844 5119 4744 4010 4835 4507 4790 4991 5202 5305 4708 4648 5225
## [181] 5515 5362 5119 4649 6043 4665 4629 4592 4040 5336 4881 4086 4258 4342 5084
## [196] 5538 5923 5302 4458 4541 4332 3784 3387 3285 3606 3840 4590 4656 4390 3846
## [211] 4475 4302 4266 4845 3574 4576 4866 4294 3785 4326 4602 4780 4792 4905 4150
## [226] 3820 4338 4725 4694 3805 4153 5191 3873 4758 5895 5130 3542 4661 1115 4334
## [241] 4634 5204 5058 5115 4727 4484 4940 3351 2710 1996 1842 3544 5345 5046 4713
## [256] 4763 4785 3659 4760 4511 4274 4539 3641 4352 4795 2395 5423 5010 4630 4120
## [271] 3907 4839 5202 2429 2918 3570 4456 4826 4765 4985 5409 5511 5117 4563 2416
## [286] 2913 3644 5217 5041 4570 4748 2424 4195 4304 4308 4381 4187 4687 3894 2659
## [301] 3747  627 3331 3669 4068 4186 3974 4046 3926 3649 4035 4205 4109 2933 3368
## [316] 4067 3717 4486 4195 1817 3053 3392 3663 3520 2765 1607 2566 1495 2792 3068
## [331] 3071 3867 2914 3613 3727 3940 3614 3485 3811 2594  705 3322 3620 3190 2743
## [346] 3310 3523 3740 3709 3577 2739 2431 3403 3750 2660 3068 2209 1011  754 1317
## [361] 1162 2302 2423 2999 2485 2294 1951 2236 2368 3272 4098 4521 3425 2376 3598
## [376] 2177 4097 3214 2493 2311 2298 2935 3376 3292 3163 1301 1977 2432 4339 4270
## [391] 4075 3456 4023 3243 3624 4509 4579 3761 4151 2832 2947 3784 4375 2802 3830
## [406] 3831 2169 1529 3422 3922 4169 3005 4154 4318 2689 3129 3777 4773 5062 3487
## [421] 2732 3389 4322 4363 1834 4990 3194 4066 3423 3333 3956 4916 5382 4569 4118
## [436] 4911 5298 5847 6312 6192 4378 7836 5892 6153 6093 6230 6871 8362 3372 4996
## [451] 5558 5102 5698 6133 5459 6235 6041 5936 6772 6436 6457 6460 6857 5169 5585
## [466] 5918 4862 5409 6398 7460 7132 6370 6691 4367 6565 7290 6624 1027 3214 5633
## [481] 6196 5026 6233 4220 6304 5572 5740 6169 6421 6296 6883 6359 6273 5728 4717
## [496] 6572 7030 7429 6118 2843 5115 7424 7384 7639 8294 7129 4359 6073 5260 6770
## [511] 6734 6536 6591 6043 5743 6855 7338 4127 8120 7641 6998 7001 7055 7494 7736
## [526] 7498 6598 6664 4972 7421 7363 7665 7702 6978 5099 6825 6211 5905 5823 7458
## [541] 6891 6779 7442 7335 6879 5463 5687 5531 6227 6660 7403 6241 6207 4840 4672
## [556] 6569 6290 7264 7446 7499 6969 6031 6830 6786 5713 6591 5870 4459 7410 6966
## [571] 7592 8173 6861 6904 6685 6597 7105 7216 7580 7261 7175 6824 5464 7013 7273
## [586] 7534 7286 5786 6299 6544 6883 6784 7347 7605 7148 7865 4549 6530 7006 7375
## [601] 7765 7582 6053 5255 6917 7040 7697 7713 7350 6140 5810 6034 6864 7112 6203
## [616] 7504 5976 8227 7525 7767 7870 7804 8009 8714 7333 6869 4073 7591 7720 8167
## [631] 8395 7907 7436 7538 7733 7393 7415 8555 6889 6778 4639 7572 7328 8156 7965
## [646] 3510 5478 6392 7691 7570 7282 7109 6639 5875 7534 7461 7509 5424 8090 6824
```

```
## [661] 7058 7466 7693 7359 7444 7852 4459   22 1096 5566 5986 5847 5138 5107 5259
## [676] 5686 5035 5315 5992 6536 6852 6269 4094 5495 5445 5698 5629 4669 5499 5634
## [691] 5146 2425 3910 2277 2424 5087 3959 5260 5323 5668 5191 4649 6234 6606 5729
## [706] 5375 5008 5582 3228 5170 5501 5319 5532 5611 5047 3786 4585 5557 5267 4128
## [721] 3623 1749 1787  920 1013  441 2114 3095 1341 1796 2729
```

On remarque qu'il n'y a pas de outliers pour la colonne "cnt".

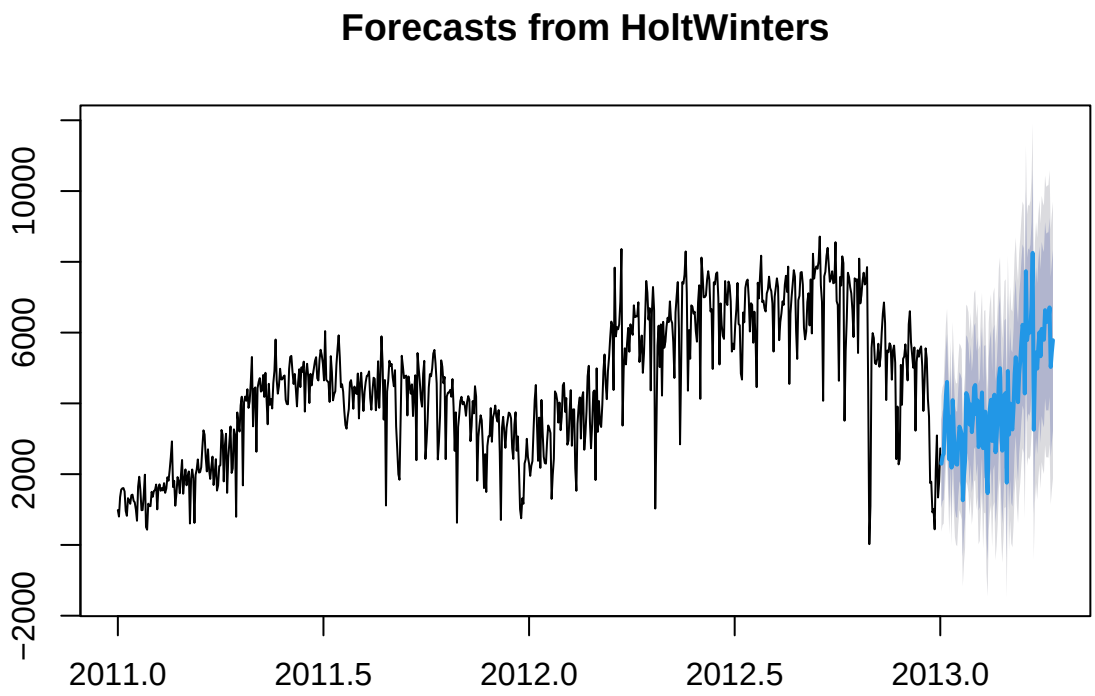
```
decomposition <-decompose(time_series_cnt)
plot(decomposition)
```



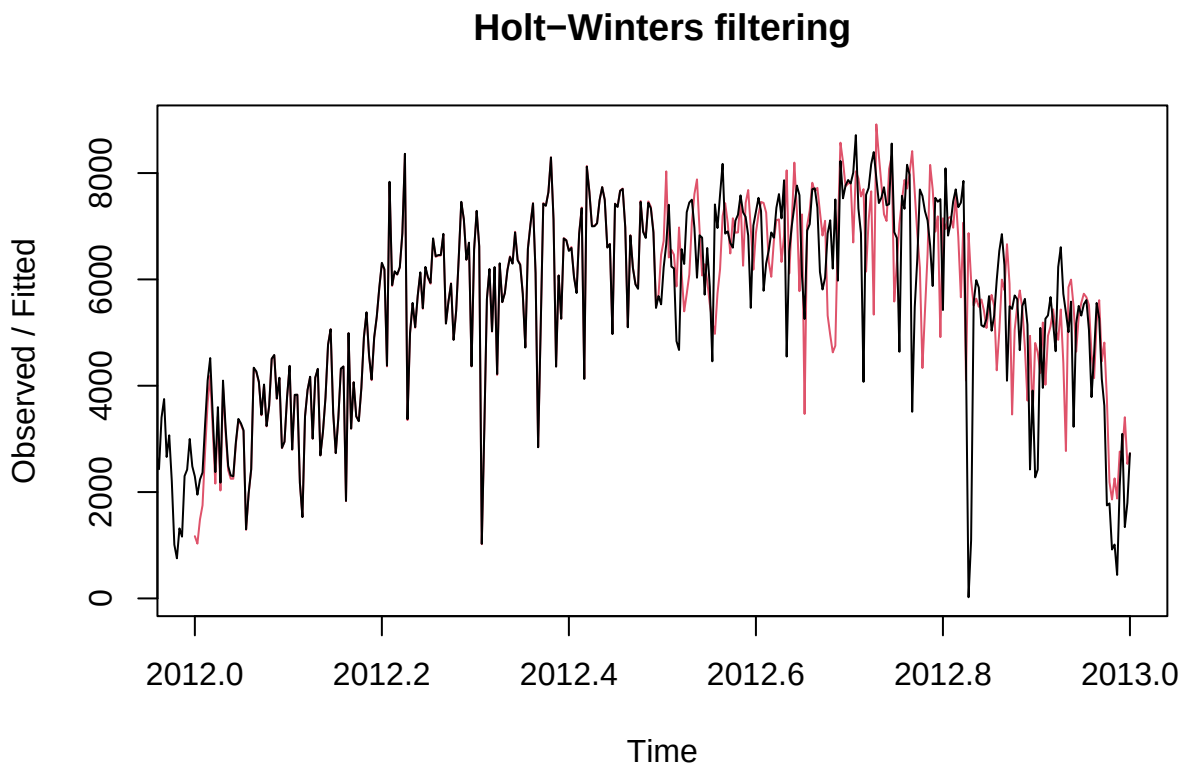
Decomposition of cnt time series

```
ht_cnt <- HoltWinters(time_series_cnt)
```

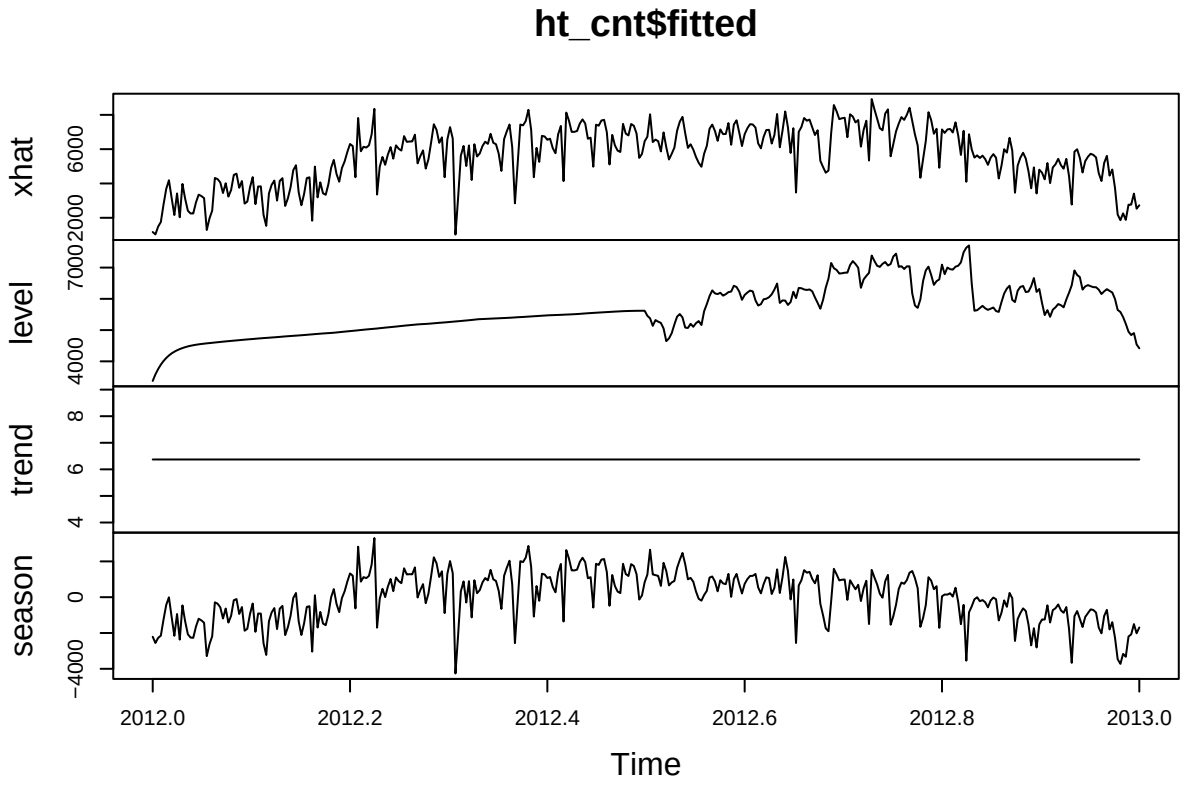
```
plot(forecast(ht_cnt,h=100))
```


Forecasts from HoltWinters
−2000
2000
6000
10000
2011.0
2011.5
2012.0
2012.5
2013.0

```
plot(ht_cnt)
```




```
plot(ht_cnt$fitted)
```

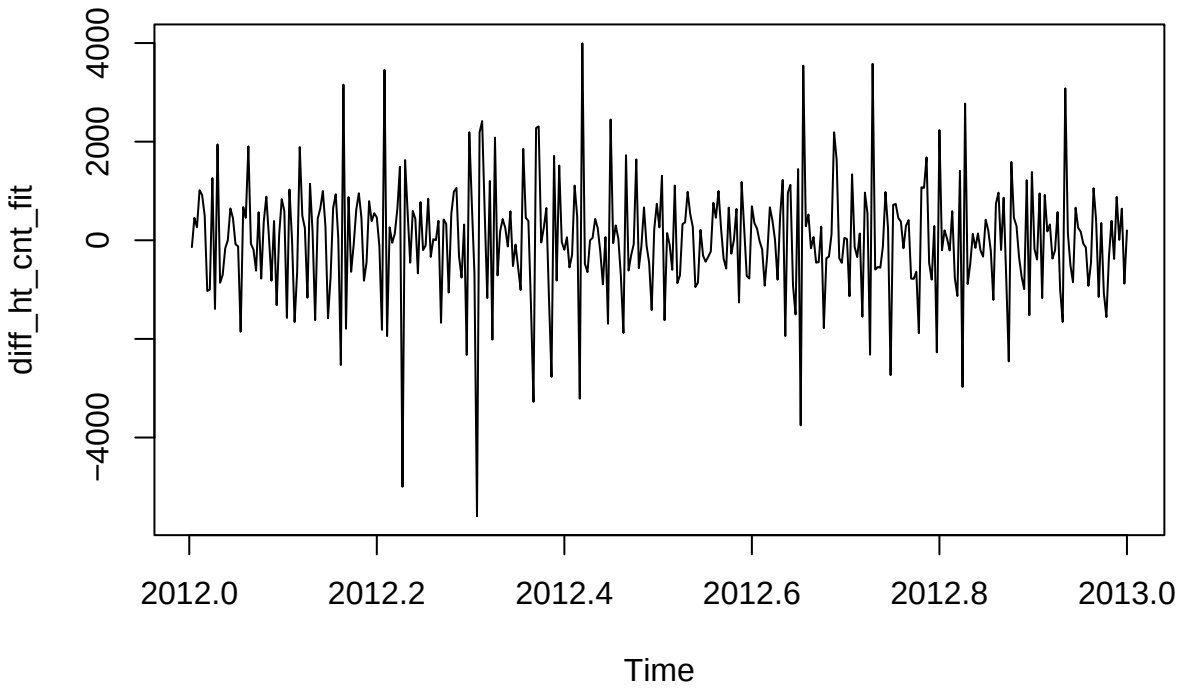


2. On observe que le modèle Holt-Winter's method ht_cnt proposé prévoit bien (avec un intervalle de confiance assez petit) la timeseries originale et arrive à capter les effets de la saisonalité. Dans le modèle proposé ht_cnt on voit que beta=0 est le meilleur paramètre ce qui signifie qu'on ne réagit pas vite au changement. Car HoltWinter ignore complètement la tendance haussière générale sur deux ans. Car il calcule les valeurs prédites sur une seule période (2012-2013) et la tendance haussière est présente sur les deux périodes (2011-2012 et 2012-2013) c'est à dire qu'en se restreignant à une seule période, on obtient une série (la série originale) sans tendance. Pour la valeur de alpha, on choisit une valeur de alpha petite (alpha ~0.18) ce qui signifie que les anciennes observations sont plus pris en compte que les récentes. En effet la série originale a beaucoup de fluctuations ce qui engendre du bruit important, si on met une valeur de alpha trop grande, on réagit vite aux changements et donc les fluctuations occasionnent un résultat moins fiable que pour une valeur de alpha petite. Pour gamma (gamma ~0.5599661), il prend bien en compte la période précédente (2011-2012).

Pour la time series du modèle, Il n'y a pas de saisonalité car par assez de période (time series originale sur 2 période 2011-2012 et 2012-2013). La time series du modèle représente les valeurs sur une seule période (2012-2013). En ce qui concerne la stationarité, on obtient une série pas trop stationnaire (pas une variance constante).

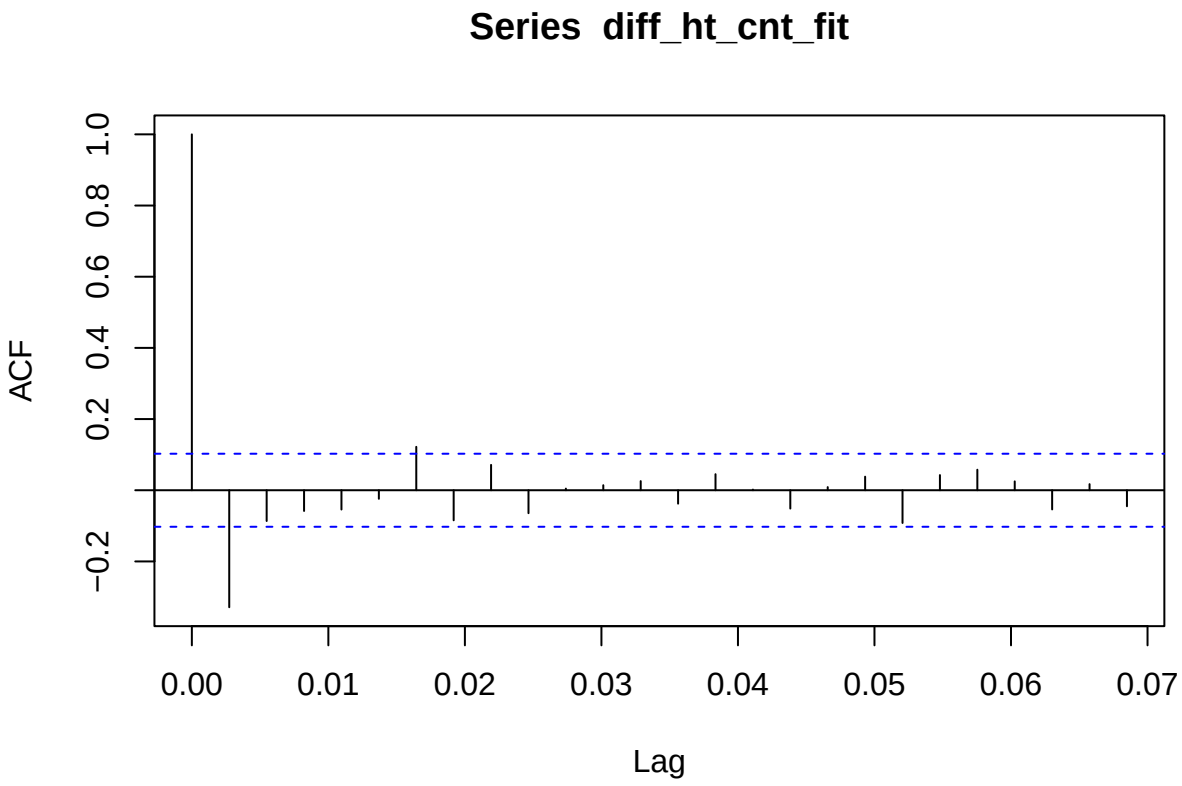
3. La série du modèle n'étant pas stationnaire, on la différencie pour obtenir une série temporelle stationnaire pour pouvoir appliquer le modèle ARIMA.

```
ht_cnt_fit <-ht_cnt$fitted[,'xhat']
diff_ht_cnt_fit<-diff(ht_cnt_fit,differences = 1)
plot(diff_ht_cnt_fit)
```

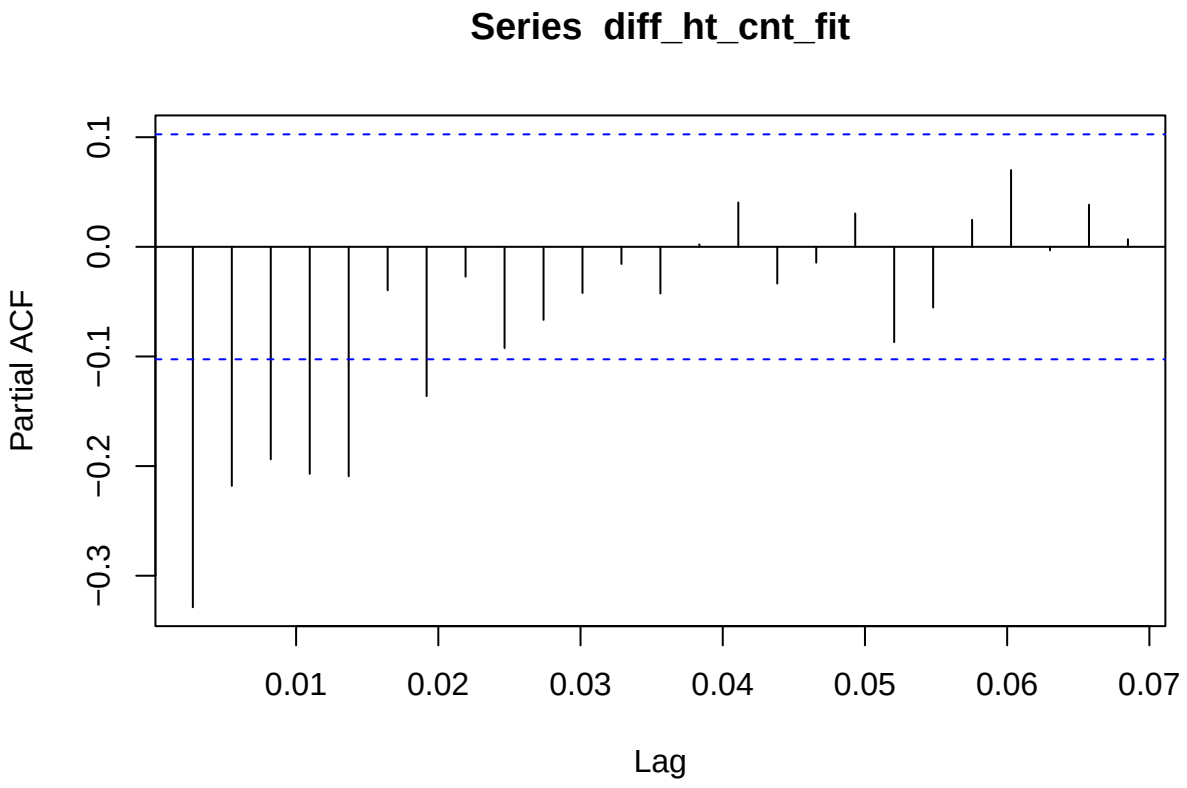


On peut voir qu'à l'ordre 1 (d=1), on obtient une série temporelle stationnaire.

```
acf(diff_ht_cnt_fit)
```




```
pacf(diff_ht_cnt_fit)
```



3. Modèles possibles: ACF tend vers 0 et PACF tend vers 0 donc 3 modèles possibles:

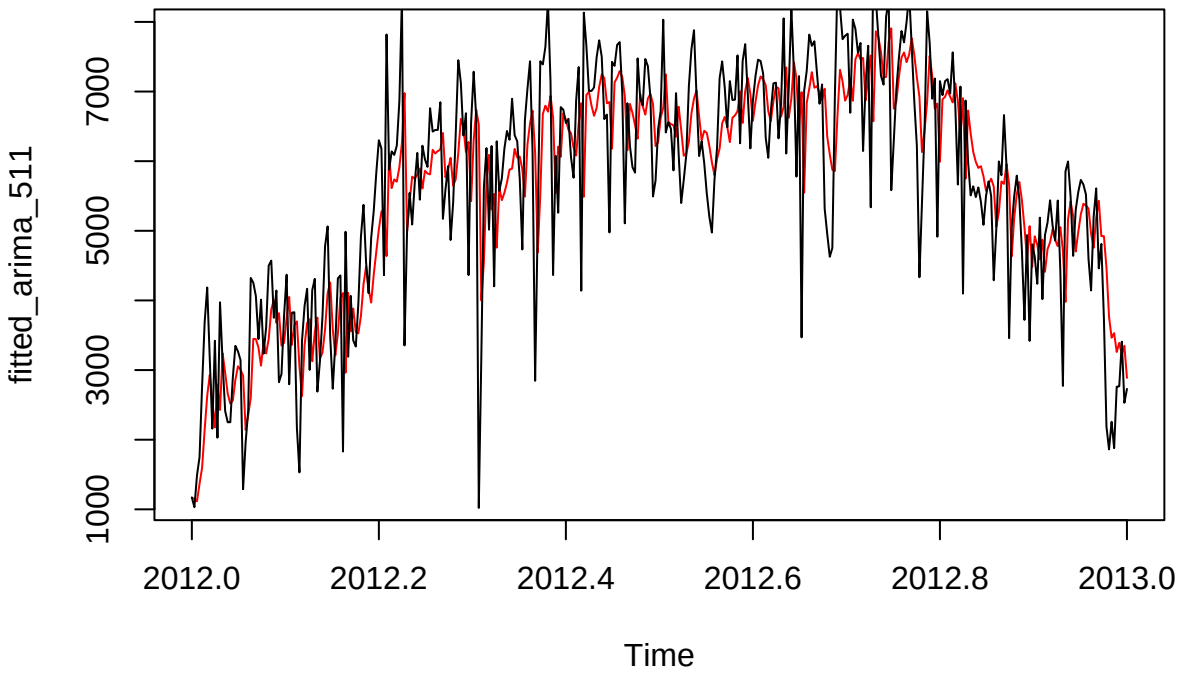
-MA(1) d'après l'ACF (après une différenciation de la série originale). (a)

-AR(5) d'après le PACF (après une différenciation de la série originale). (b)

-ARIMA(p=5,d=1,q=1).

On considère ARIMA(5,1,1) car (a) puis (b).

```
arima_511<-arima(ht_cnt$fitted[,'xhat'],order = c(5, 1, 1))
fitted_arima_511 <-ht_cnt$fitted[,'xhat']-arima_511$residuals
plot(fitted_arima_511,col="red")
lines(ht_cnt$fitted[,'xhat'])
```




```
arima_511
```

```
##
## Call:
## arima(x = ht_cnt$fitted[, "xhat"], order = c(5, 1, 1))
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1
##       0.2174  -0.0522  -0.0854  -0.0821  -0.0199  -0.7914
## s.e.  0.0976   0.0692   0.0646   0.0633   0.0662   0.0832
##
## sigma^2 estimated as 991436:  log likelihood = -3038.17,  aic = 6090.34
```

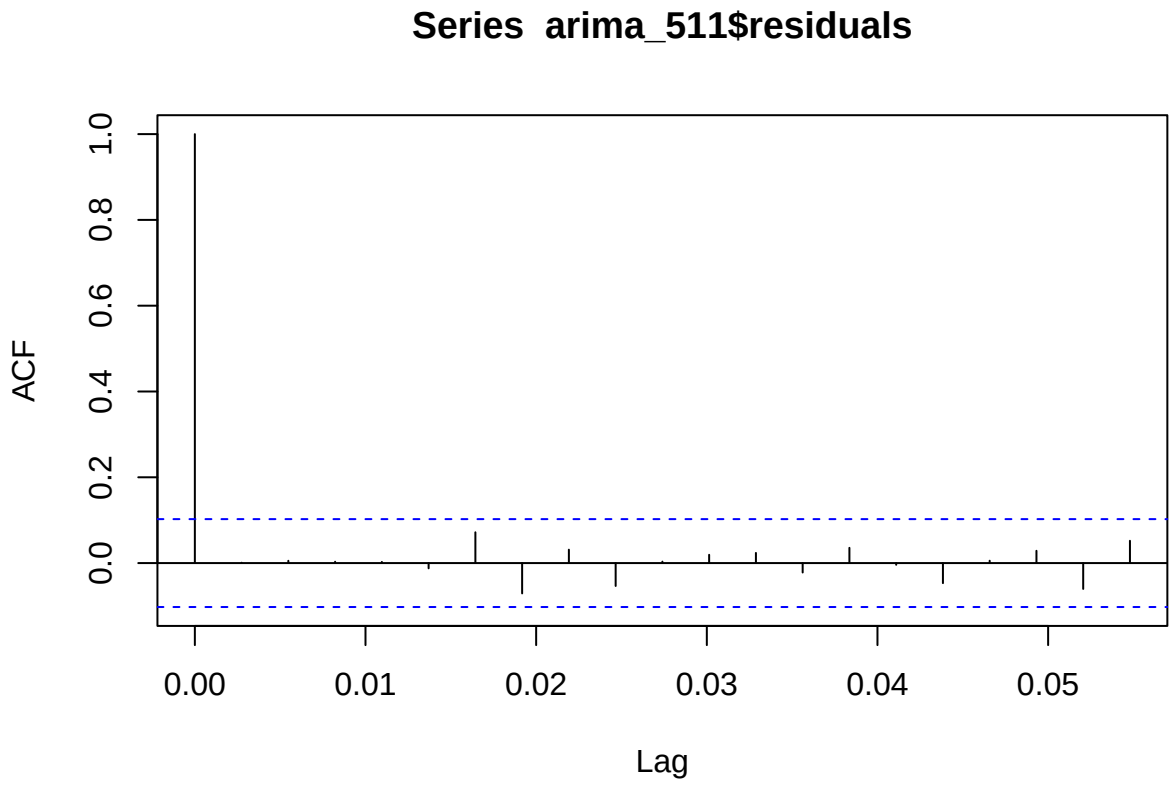
en rouge: modèle ARIMA(5,1,1) pour la série temporelle smoothed. en noir: smoothed time series.

```
Box.test(arima_511$residuals,lag=20,type="Ljung")
```

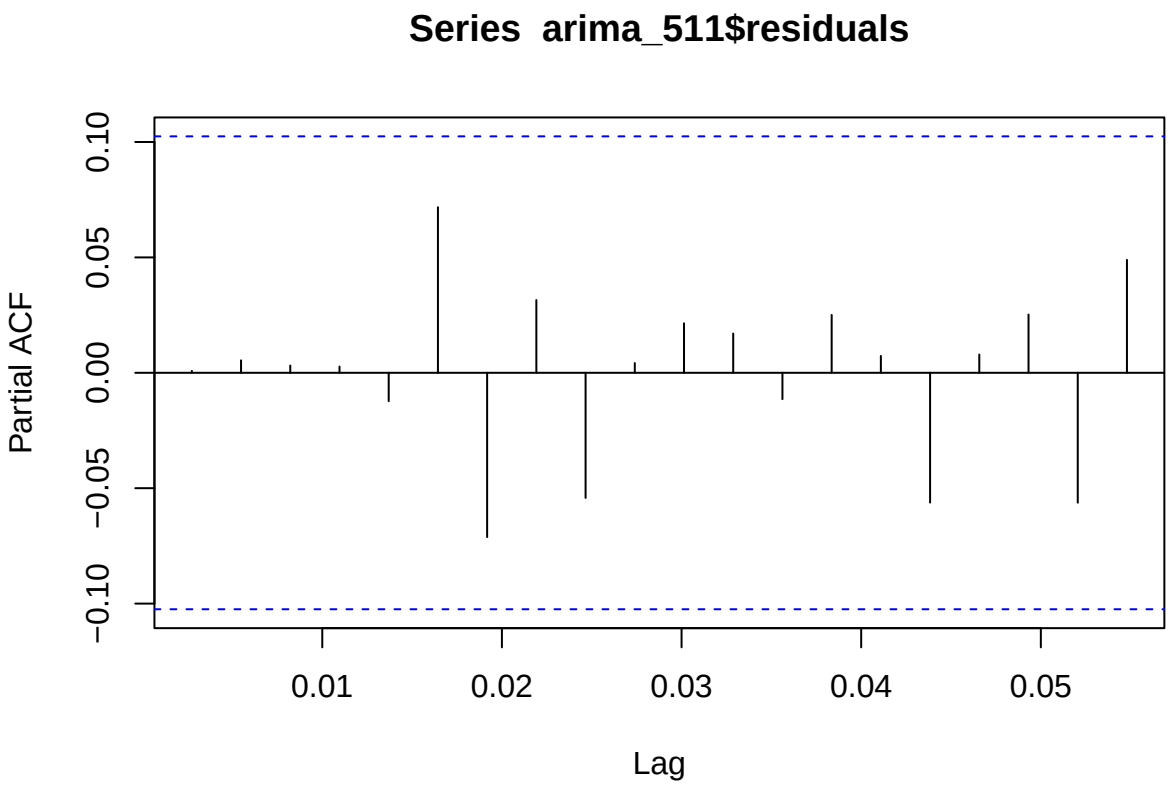
```
##
##  Box-Ljung test
##
## data:  arima_511$residuals
## X-squared = 9.9923, df = 20, p-value = 0.9683
```

On voit que la p-value ~0.9683 pour Box-Ljung test est très proche de 1 (>0.05) donc les 20 premiers résidus sont indépendants.(auto corrélation =0 pour les 20 premiers résidus –>acceptation de l'hypothèse nulle du test). –>résidus valident pour le modèle (bruit blanc).

```
acf(arima_511$residuals,lag.max=20)
```

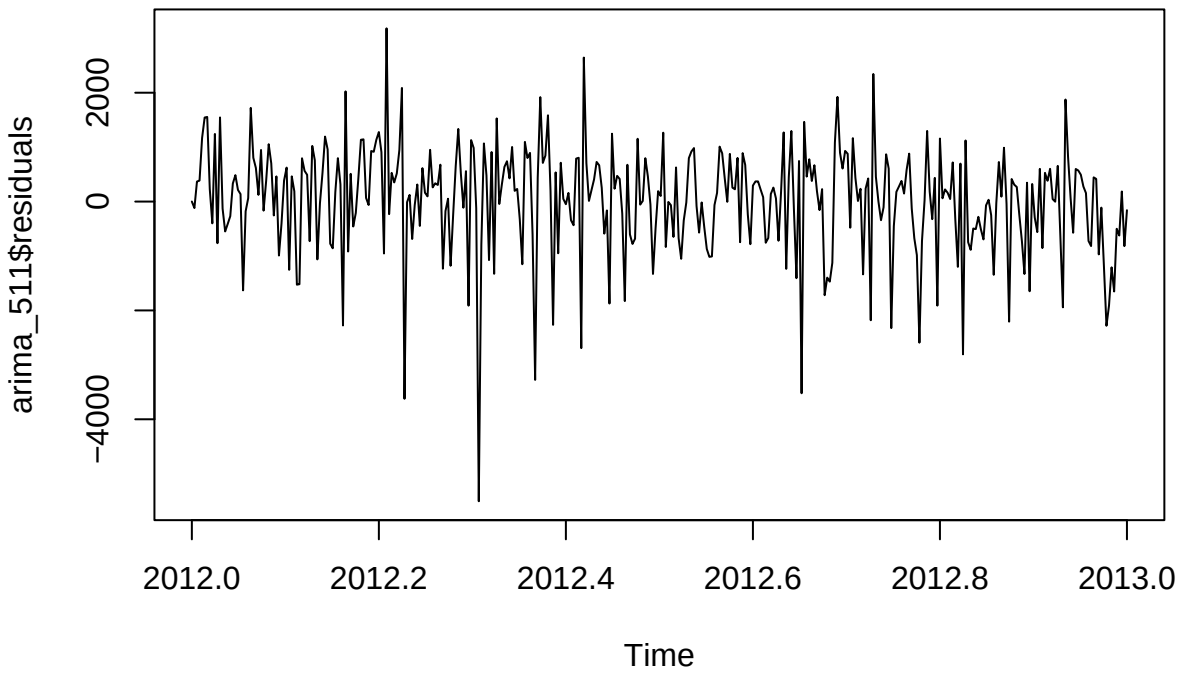



```
pacf(arima_511$residuals,lag.max=20)
```



Pas d'auto-corrélation ni d'auto corrélation partielle pour les résidus (20 premiers) –> résidu est bien un bruit blanc.

```
plot(arima_511$residuals)
```

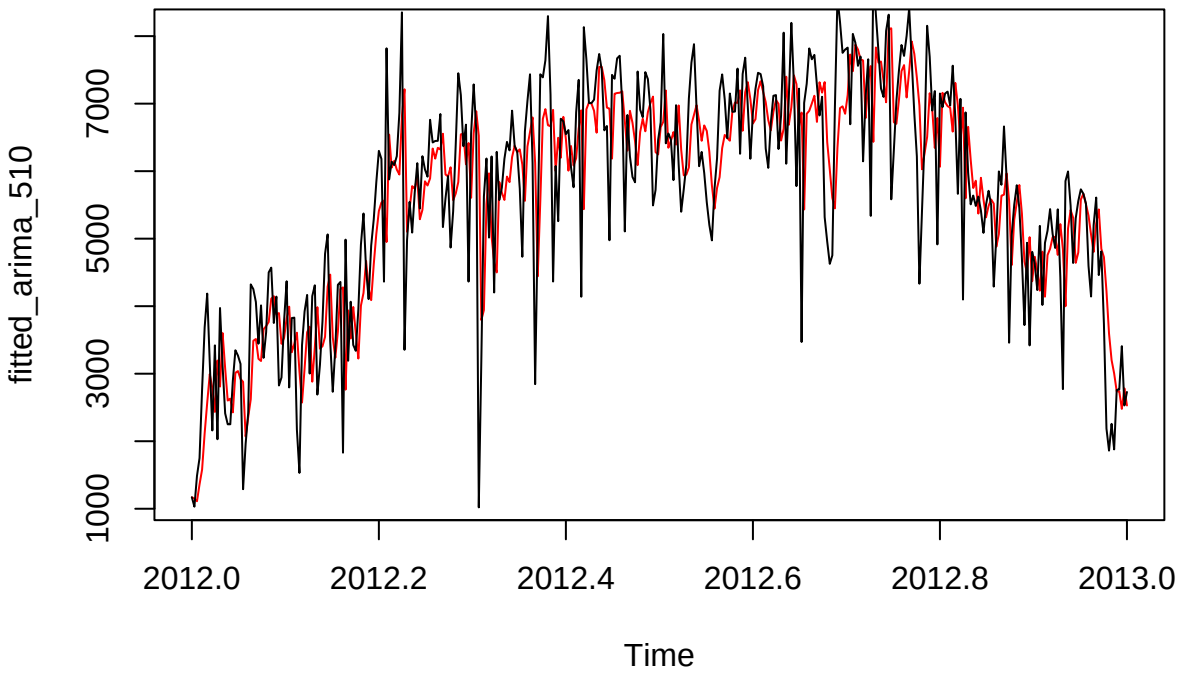



```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_511$residuals),digits=2),
```

```
## [1] "Moyenne des résidus:18.040000 et Variance des résidus: 991109.890000"
```

résidus un peu près centrés + variance un peu près constante (d'après le graphique) –> résidus bruits blancs.

```
arima_510<-arima(ht_cnt$fitted[,'xhat'],order = c(5, 1,0))
fitted_arima_510 <-ht_cnt$fitted[,'xhat']-arima_510$residuals
plot(fitted_arima_510,col="red")
lines(ht_cnt$fitted[,'xhat'])
```




```
arima_510
```

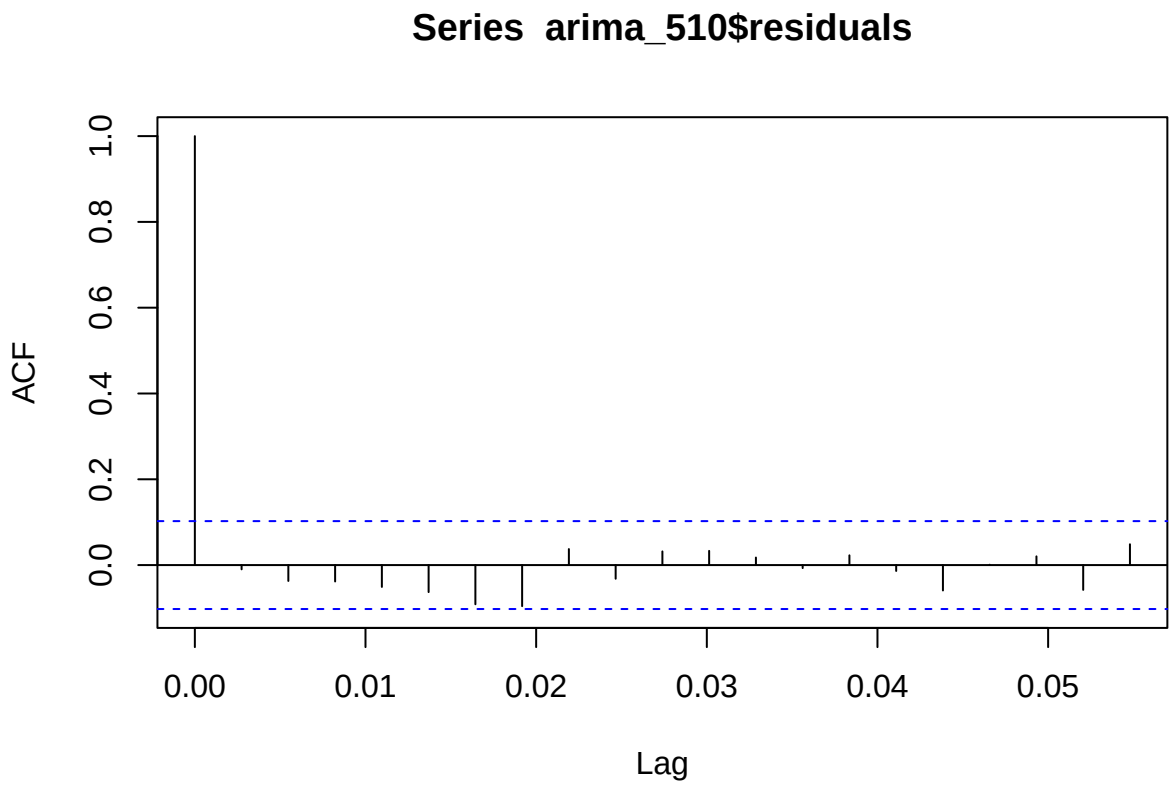
```
##
## Call:
## arima(x = ht_cnt$fitted[, "xhat"], order = c(5, 1, 0))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.5264  -0.4172  -0.3613  -0.3099  -0.2114
## s.e.   0.0511   0.0558   0.0569   0.0558   0.0512
##
## sigma^2 estimated as 1018405:  log likelihood = -3042.94,  aic = 6097.87
```

```
Box.test(arima_510$residuals,lag=20,type="Ljung")
```

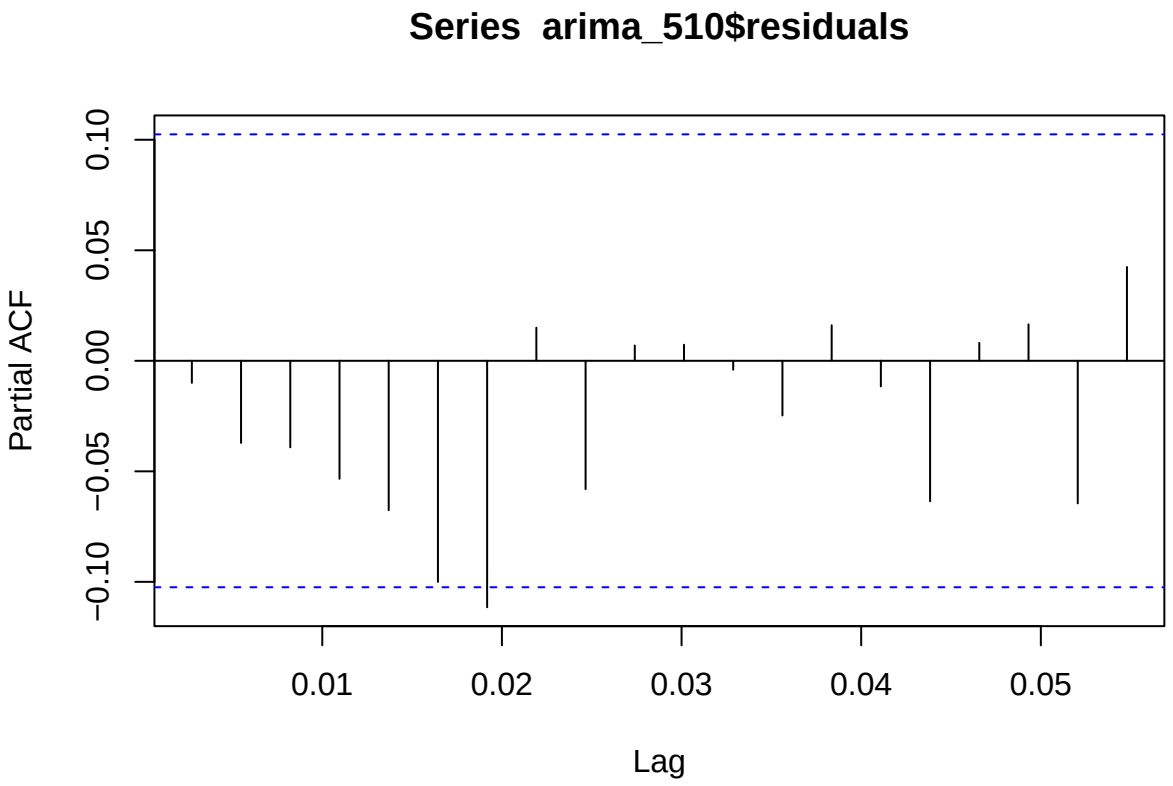
```
##
##  Box-Ljung test
##
## data:  arima_510$residuals
## X-squared = 15.952, df = 20, p-value = 0.7196
```

On voit que la p-value ~0.7196 pour Box-Ljung test est proche de 1 (>0.05) donc les 20 premiers résidus sont indépendants. (auto corrélation =0 pour les 20 premiers résidus –>acceptation de l'hypothèse nulle du test). –>résidus valident pour le modèle (bruit blanc).

```
acf(arima_510$residuals,lag.max=20)
```

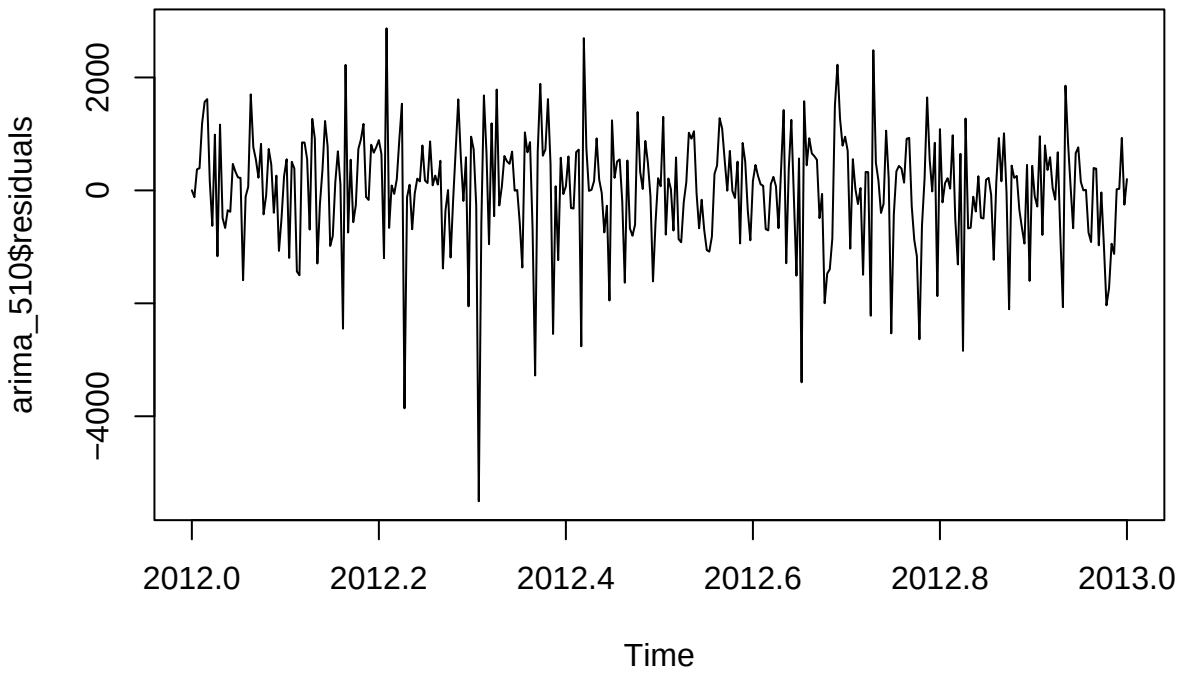



```
pacf(arima_510$residuals,lag.max=20)
```



Pas d'auto-corrélation ni d'auto corrélation partielle pour les résidus (20 premiers) –> résidu est bien un bruit blanc.

```
plot(arima_510$residuals)
```



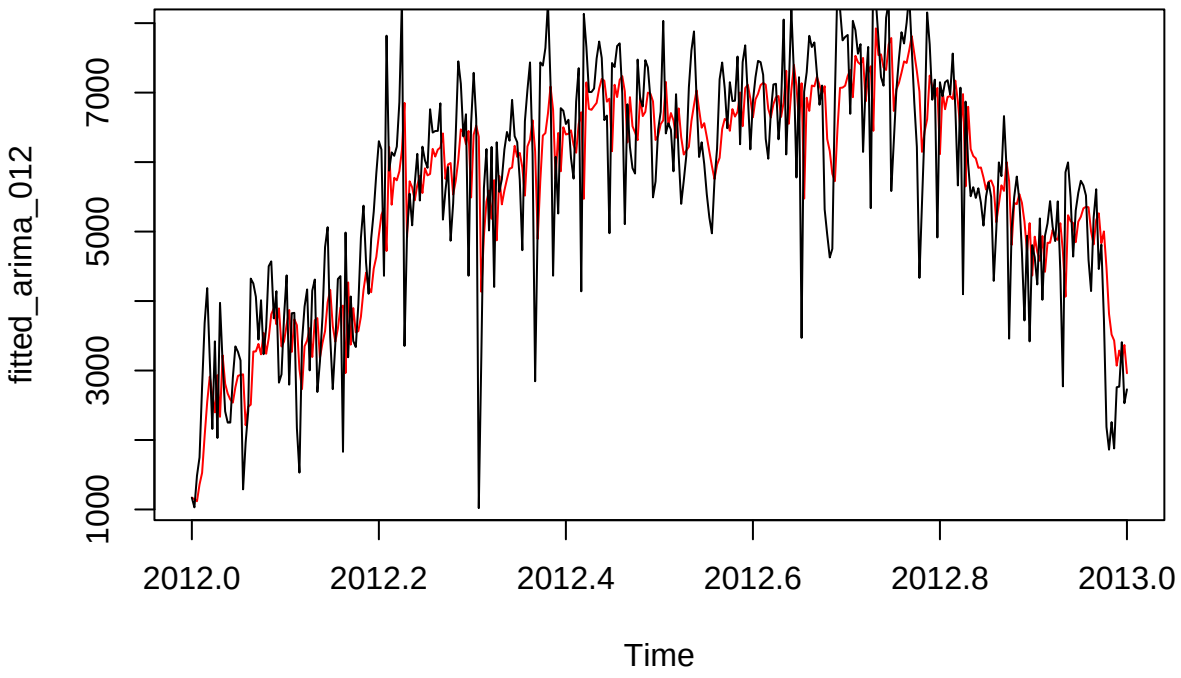

```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_510$residuals),digits=2),
```

```
## [1] "Moyenne des résidus:11.360000 et Variance des résidus: 1018275.670000"
```

résidus un peu près centrés + variance un peu près constante (d'après le graphique) –> résidus bruits blancs.

À la place du ARIMA(0,1,1), nous choisissons de prendre ARIMA(0,1,2) car les résultats sont bien meilleurs pour ce modèle ci-dessous.

```
arima_012<-arima(ht_cnt$fitted[,'xhat'],order = c(0, 1,2))
fitted_arima_012<-ht_cnt$fitted[,'xhat']-arima_012$residuals
plot(fitted_arima_012,col="red")
lines(ht_cnt$fitted[,'xhat'])
```




```
arima_012
```

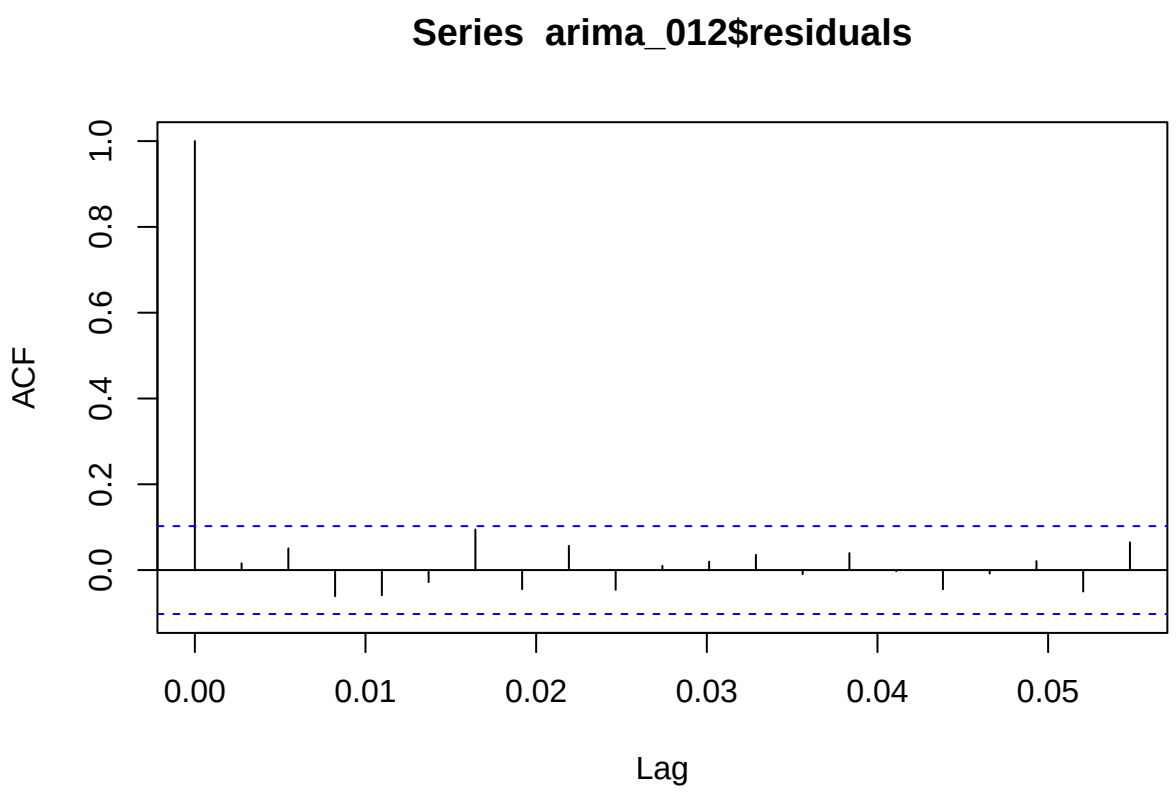
```
##
## Call:
## arima(x = ht_cnt$fitted[, "xhat"], order = c(0, 1, 2))
##
## Coefficients:
##           ma1      ma2
##       -0.5862  -0.2227
## s.e.   0.0484   0.0472
##
## sigma^2 estimated as 1005374:  log likelihood = -3040.69,  aic = 6087.39
```

```
Box.test(arima_012$residuals,lag=20,type="Ljung")
```

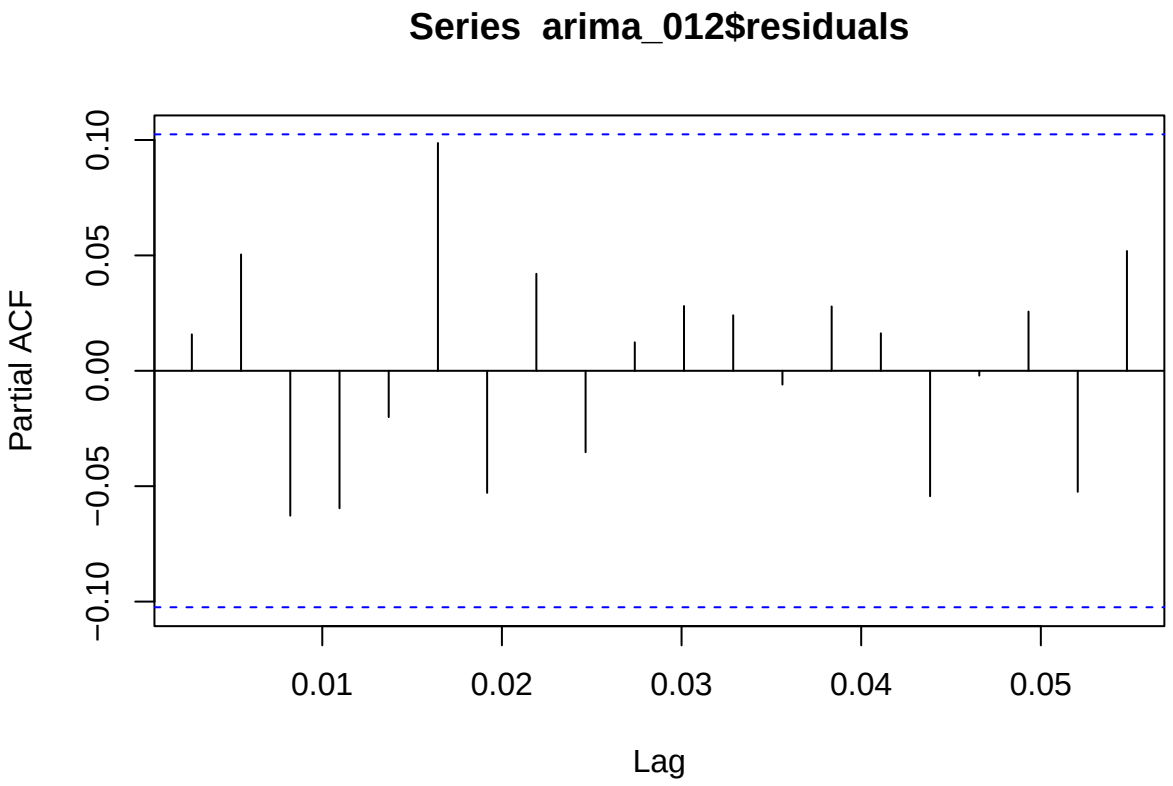
```
##
##  Box-Ljung test
##
## data:  arima_012$residuals
## X-squared = 14.932, df = 20, p-value = 0.7803
```

On voit que la p-value ~0.7803 pour Box-Ljung test est proche de 1 (>0.05) donc les 20 premiers résidus sont indépendants. (auto corrélation =0 pour les 20 premiers résidus –>acceptation de l'hypothèse nulle du test). –>résidus valident pour le modèle (bruit blanc).

```
acf(arima_012$residuals,lag.max=20)
```

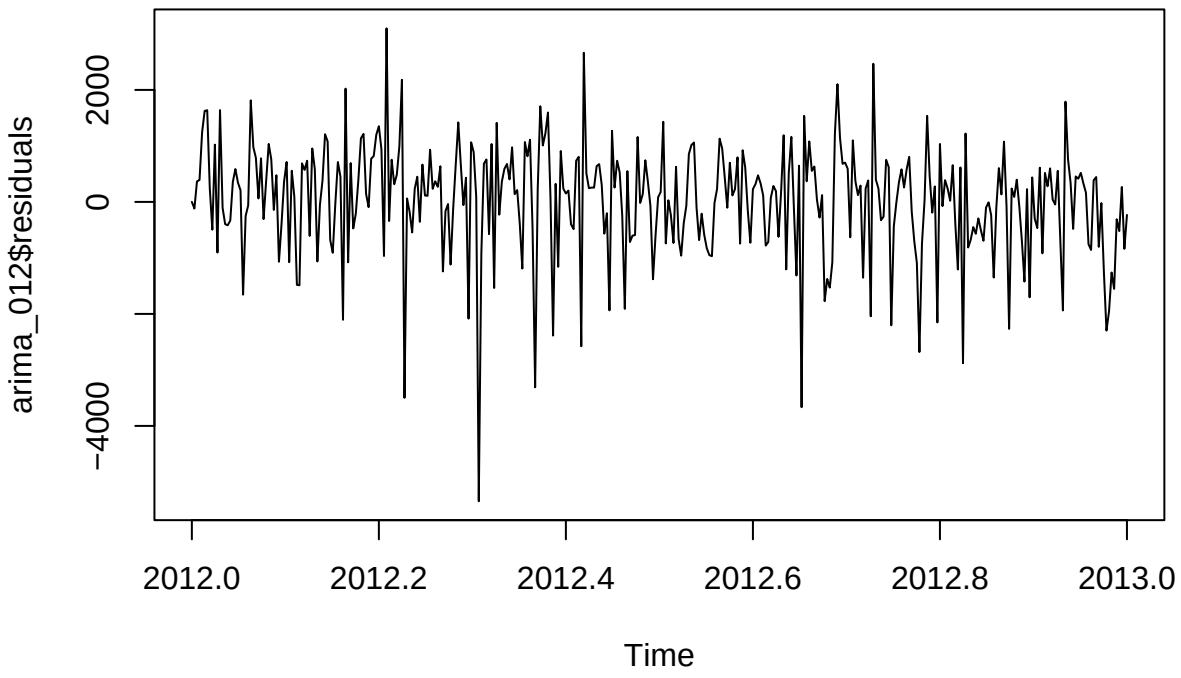



```
pacf(arima_012$residuals,lag.max=20)
```



Pas d'auto-corrélation ni d'auto corrélation partielle pour les résidus (20 premiers) –> résidu est bien un bruit blanc.

```
plot(arima_012$residuals)
```




```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_012$residuals),digits=2),
```

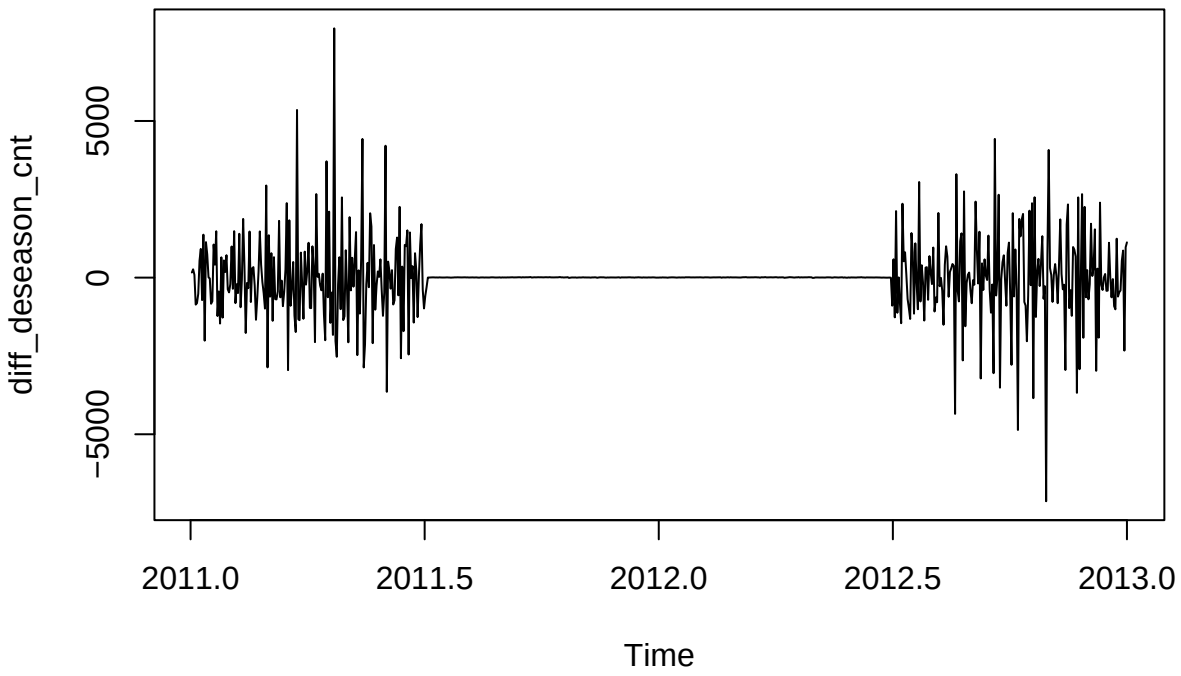
```
## [1] "Moyenne des résidus:21.770000 et Variance des résidus: 1004898.430000"
```

résidus un peu près centrés + variance un peu près constante (d'après le graphique) –> résidus bruits blancs.

Conclusion: Les trois modèles sont validents (résidus bruits blancs entre autres), nous prenons le modèle avec le plus petit AIC: AIC=6087.39 pour ARIMA(0,1,2) / AIC=6090.34 pour ARIMA(5,1,1) / AIC=6097.87 pour ARIMA(5,1,0). ARIMA(0,1,2) est le meilleur modèle car c'est le plus simple (moins de paramètres –>prévenir l'overfitting) et le plus petit AIC.

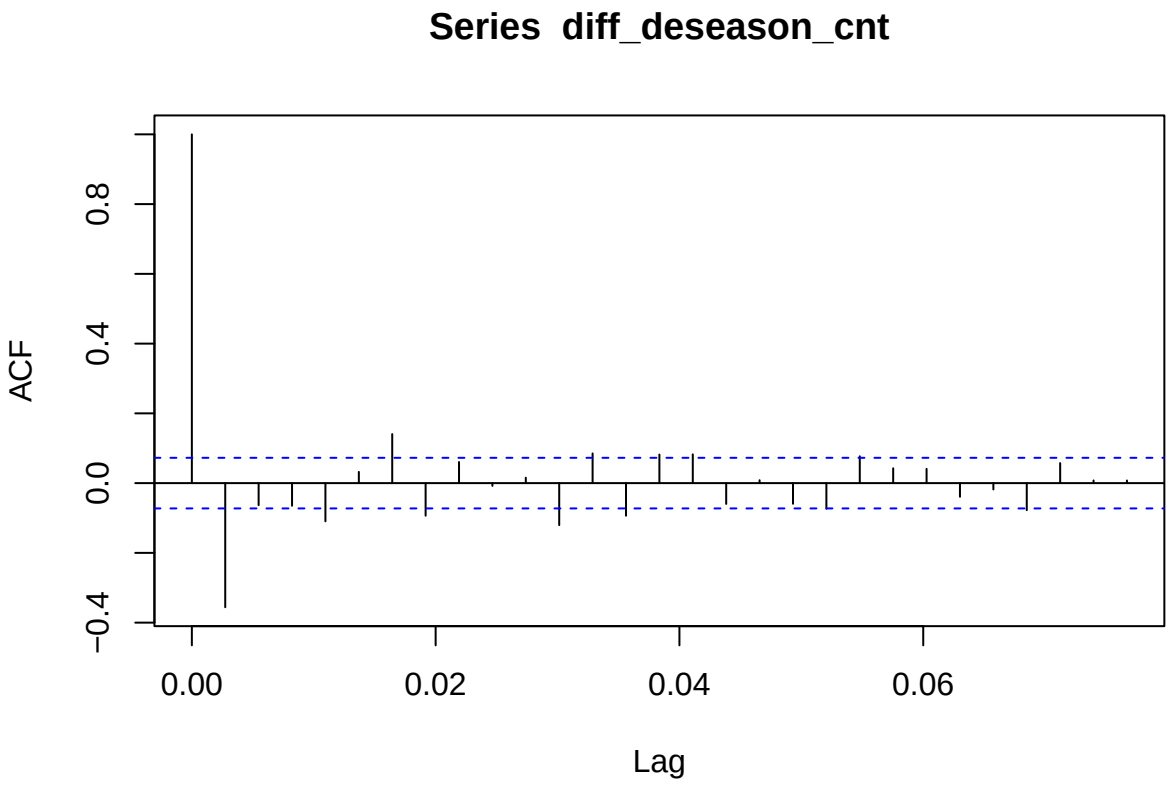
4.I

```
deseason_cnt=seasadj(decomposition) #retire la saisonalité
diff_deseason_cnt=diff(deseason_cnt,differences=1) #différenciation
plot(diff_deseason_cnt)
```

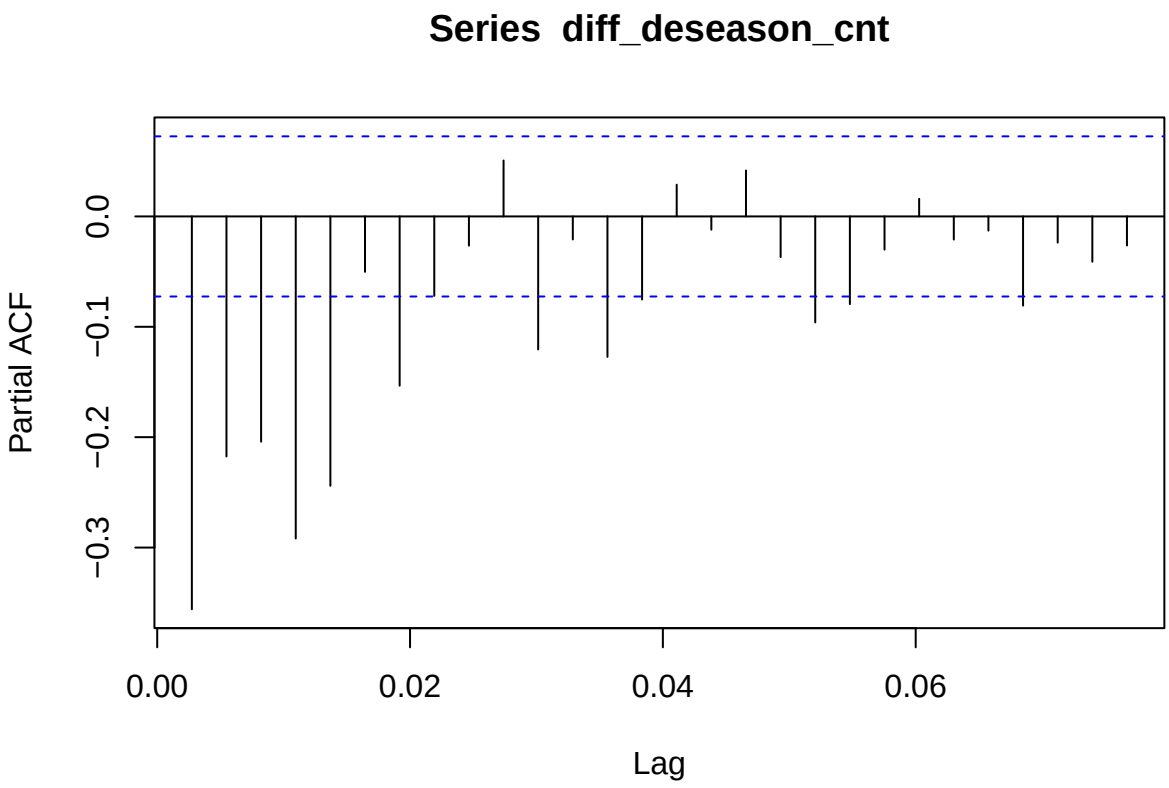


Avec d=1 –> série temporelle stationnaire (moyenne autour de 0).

```
acf(diff_deseason_cnt)
```




```
pacf(diff_deseason_cnt)
```



ACF et PACF tend vers 0 de manière exponentielle:

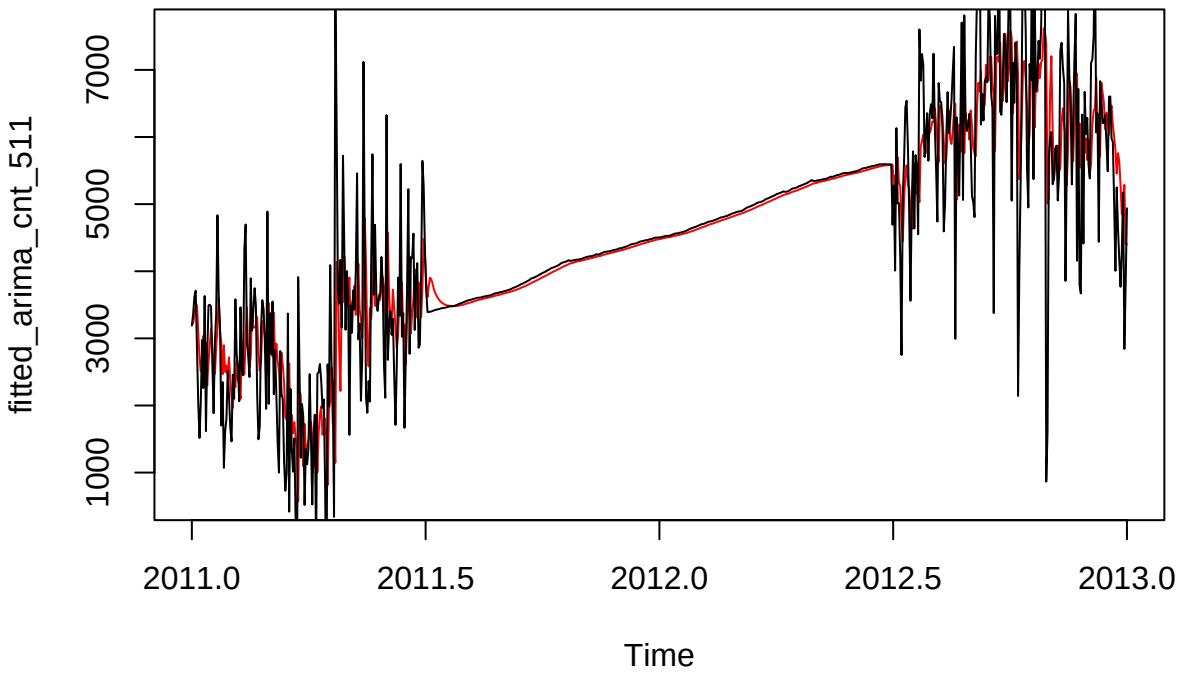
-MA(1) d'après l'ACF (après une différenciation de la série originale). (a)

-AR(5) d'après le PACF (après une différenciation de la série originale). (b)

-ARIMA(p=5,d=1,q=1).

On considère dans un premiers temps ARIMA(5,1,1) car (a) puis (b).

```
arima_cnt_511<-arima(deseason_cnt,order = c(5, 1,1))
fitted_arima_cnt_511 <-deseason_cnt-arima_cnt_511$residuals
plot(fitted_arima_cnt_511,col="red")
lines(deseason_cnt)
```




```
arima_cnt_511
```

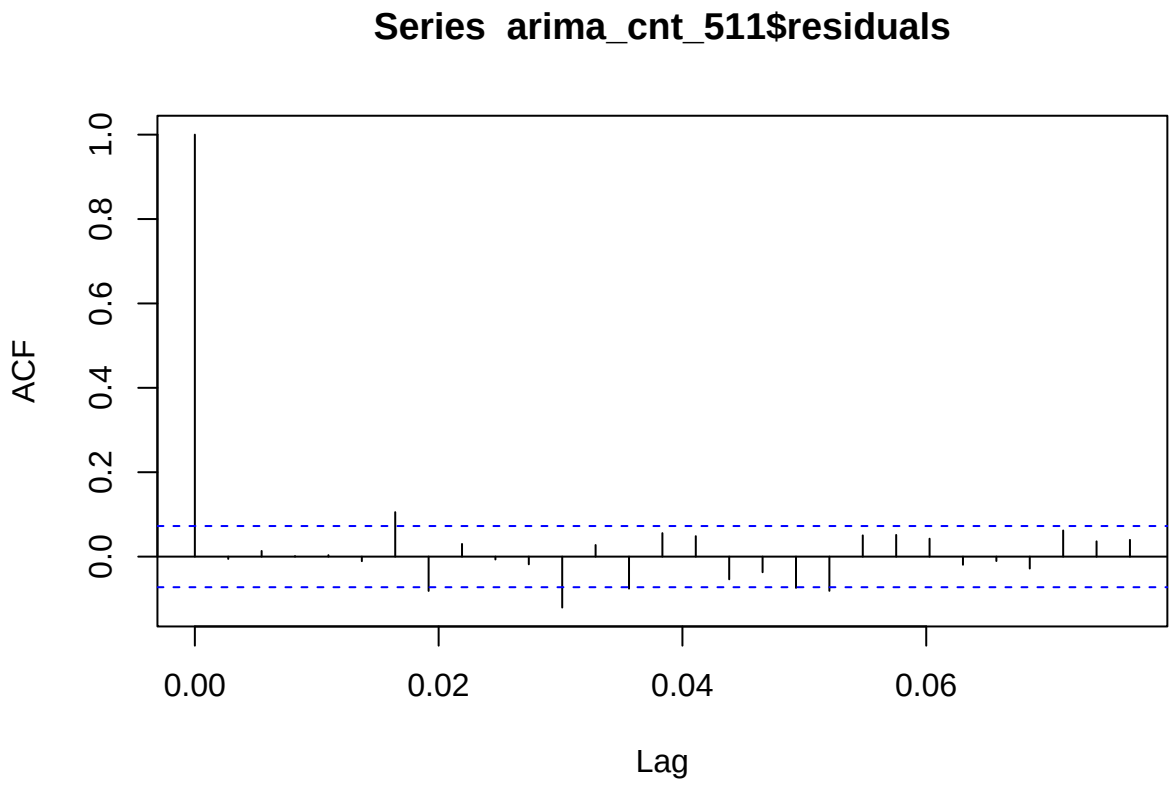
```
##
## Call:
## arima(x = deseason_cnt, order = c(5, 1, 1))
##
## Coefficients:
##          ar1      ar2      ar3      ar4     ar5      ma1
##       0.1794  -0.0424  -0.1088  -0.1224  0.0504  -0.8388
## s.e.  0.0852   0.0595   0.0528   0.0536  0.0602   0.0749
##
## sigma^2 estimated as 782310:  log likelihood = -5989.55,  aic = 11993.1
```

```
Box.test(arima_cnt_511$residuals,lag=20,type="Ljung")
```

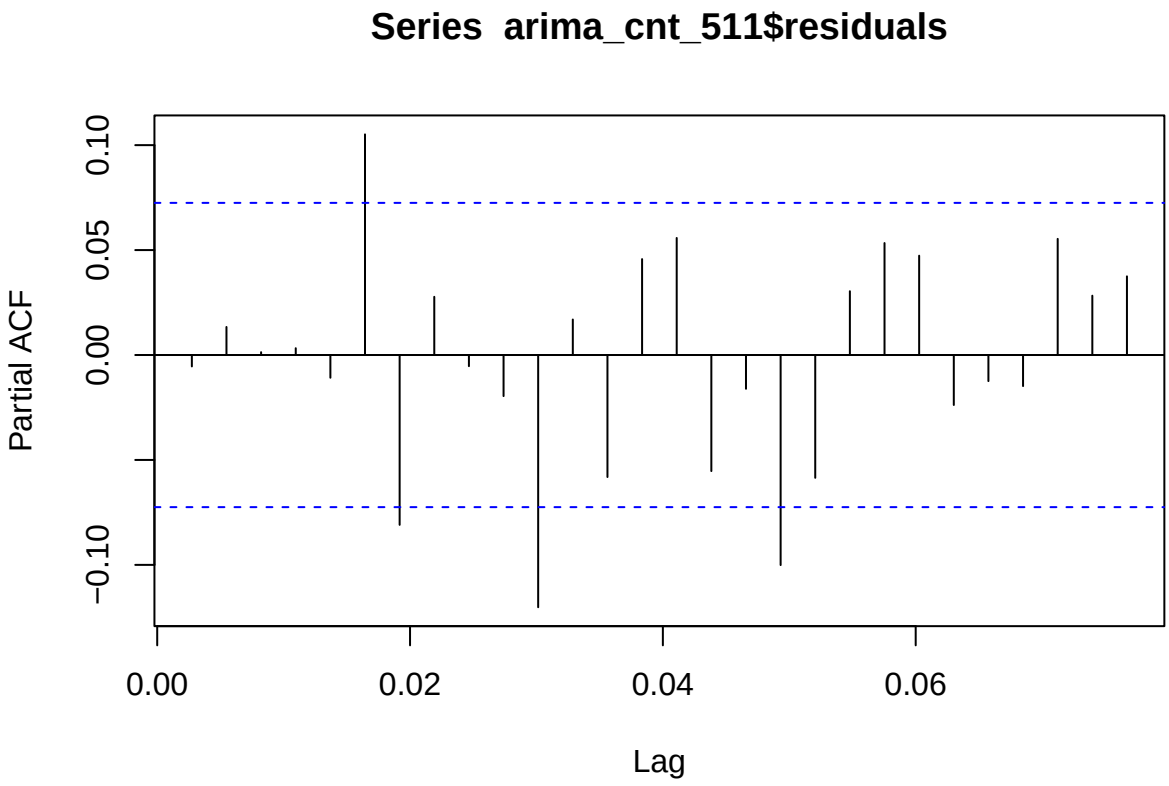
```
##
##  Box-Ljung test
##
## data:  arima_cnt_511$residuals
## X-squared = 48.23, df = 20, p-value = 0.0003949
```

À un seuil de risque de 5%, p-value ~0.0003949 (<0.05) donc on rejete l'hypothèse d'indépendance des 20 premiers résidus. –> mauvais modèle: la probabilité qu'un résidu (les 20 premiers) dépend du temps car les résidus (les 20 premiers) ne sont pas indépendants.

```
acf(arima_cnt_511$residuals)
```

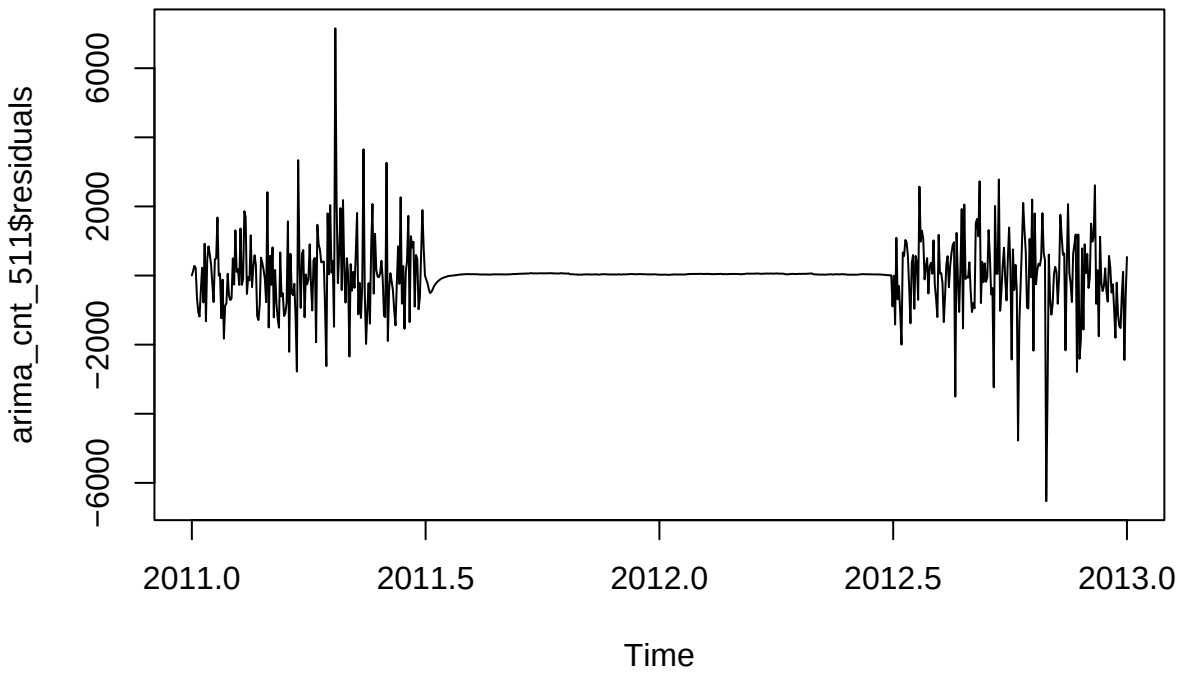



```
pacf(arima_cnt_511$residuals)
```



On voit que certains lags dépassent (ACF) l'intervalle de confiance de 95% –> possiblement auto corrélation différente de 0 pour certains résidus (ça se confirme avec le Test Q de Ljung-Box).

```
plot(arima_cnt_511$residuals)
```



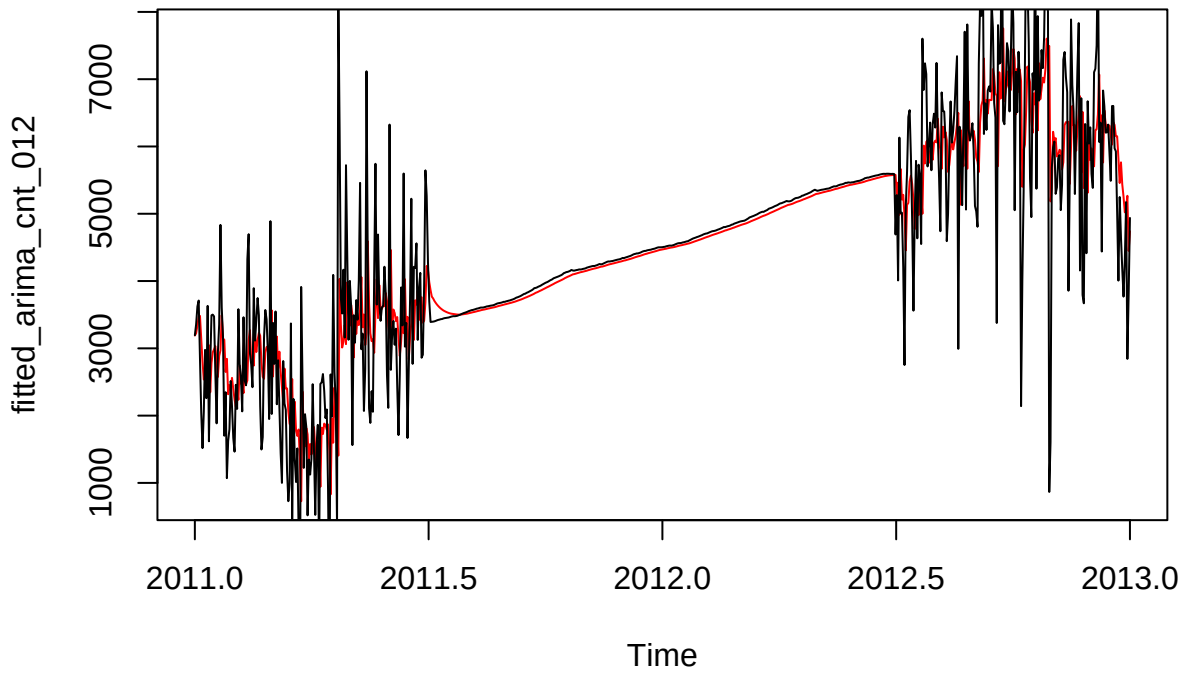

```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_cnt_511$residuals),digits=
```

```
## [1] "Moyenne des résidus:13.240000 et Variance des résidus: 782134.780000"
```

Résidus nulles pour la période mi 2011-mi 2012 –> logique car la série temporelle deseason_cnt ressemble à une droite linéaire sans bruit ni aléas (facile à prédire). C'est la tendance. Moyenne centrée en 0 et variance constante (avec quelques variation (pics hauts) à certaines périodes).

Nous choisissons ARIMA(0,1,2) au lieu de ARIMA(0,1,1) car il est plus performant.

```
arima_cnt_012<-arima(deseason_cnt,order = c(0, 1,2))
fitted_arima_cnt_012 <-deseason_cnt-arima_cnt_012$residuals
plot(fitted_arima_cnt_012,col="red")
lines(deseason_cnt)
```




```
arima_cnt_012
```

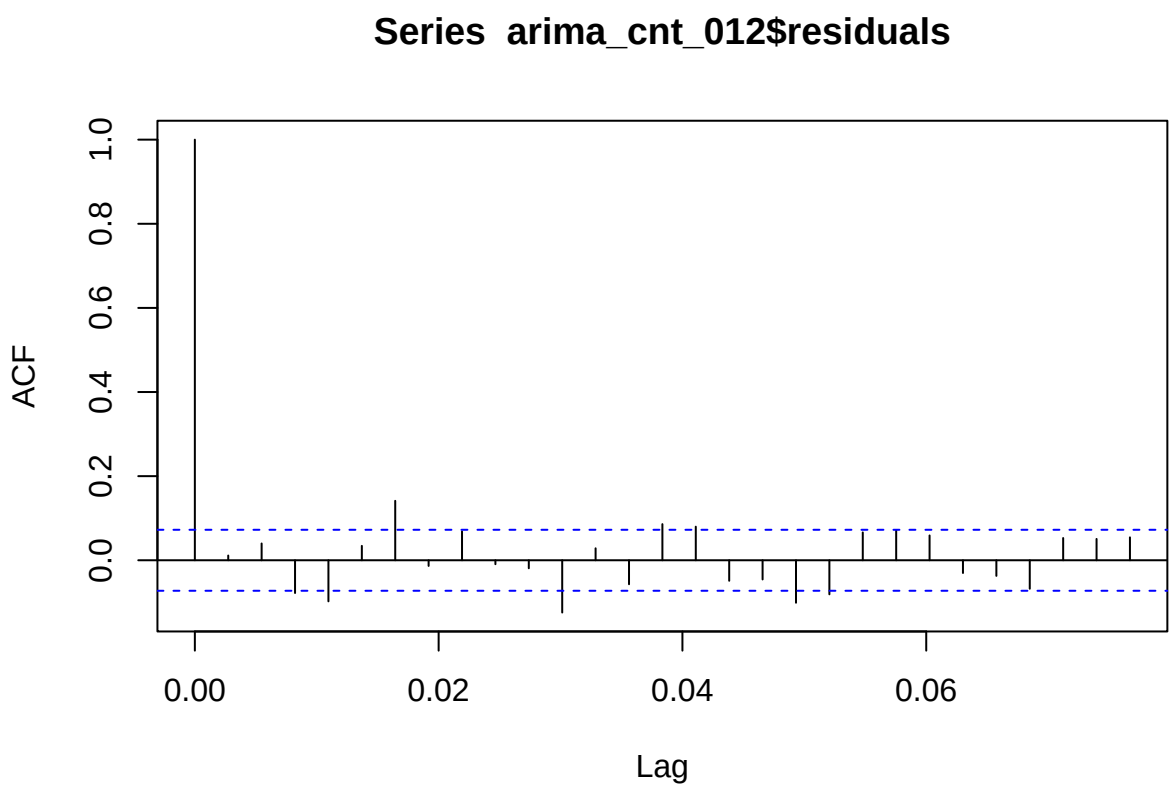
```
##
## Call:
## arima(x = deseason_cnt, order = c(0, 1, 2))
##
## Coefficients:
##           ma1      ma2
##       -0.6686  -0.2075
## s.e.   0.0345   0.0346
##
## sigma^2 estimated as 802906:  log likelihood = -5999.03,  aic = 12004.06
```

```
Box.test(arima_cnt_012$residuals,lag=20,type="Ljung")
```

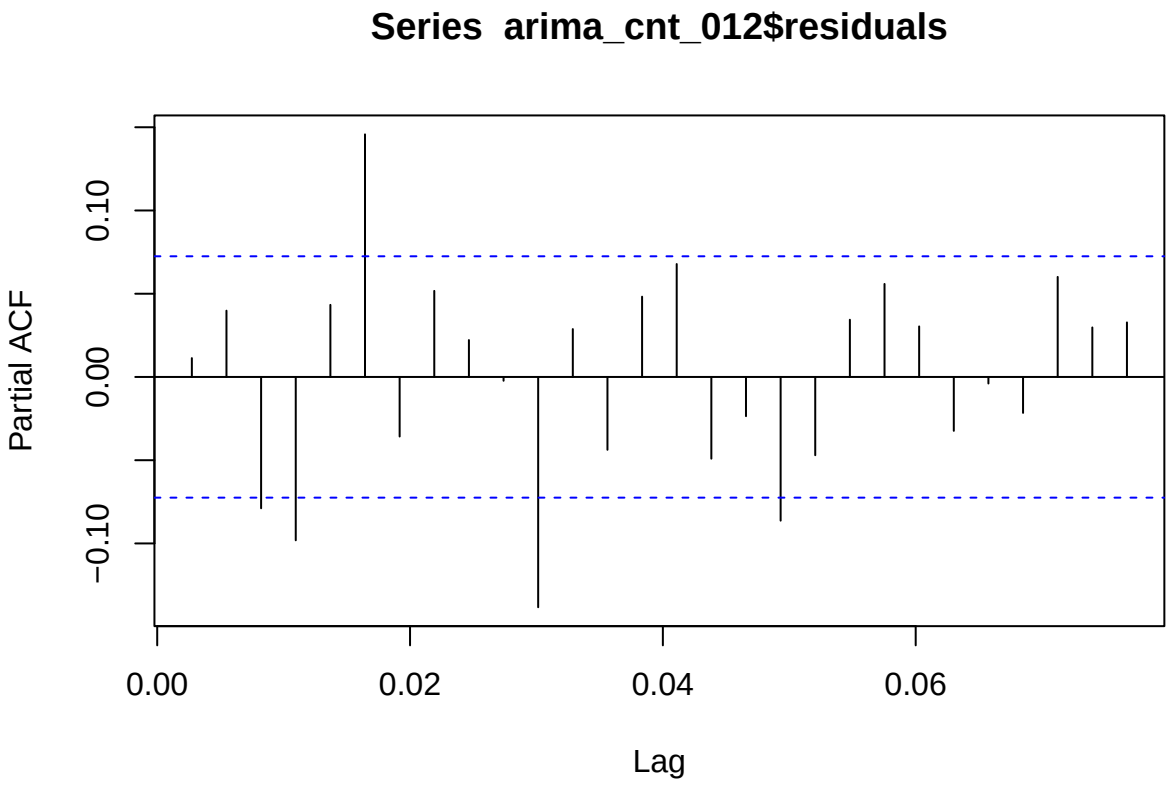
```
##
##  Box-Ljung test
##
## data:  arima_cnt_012$residuals
## X-squared = 76.434, df = 20, p-value = 1.568e-08
```

Les 20 premiers résidus ne sont pas indépendants: p-value~1.568e-08 très petite p-value!!!

```
acf(arima_cnt_012$residuals)
```

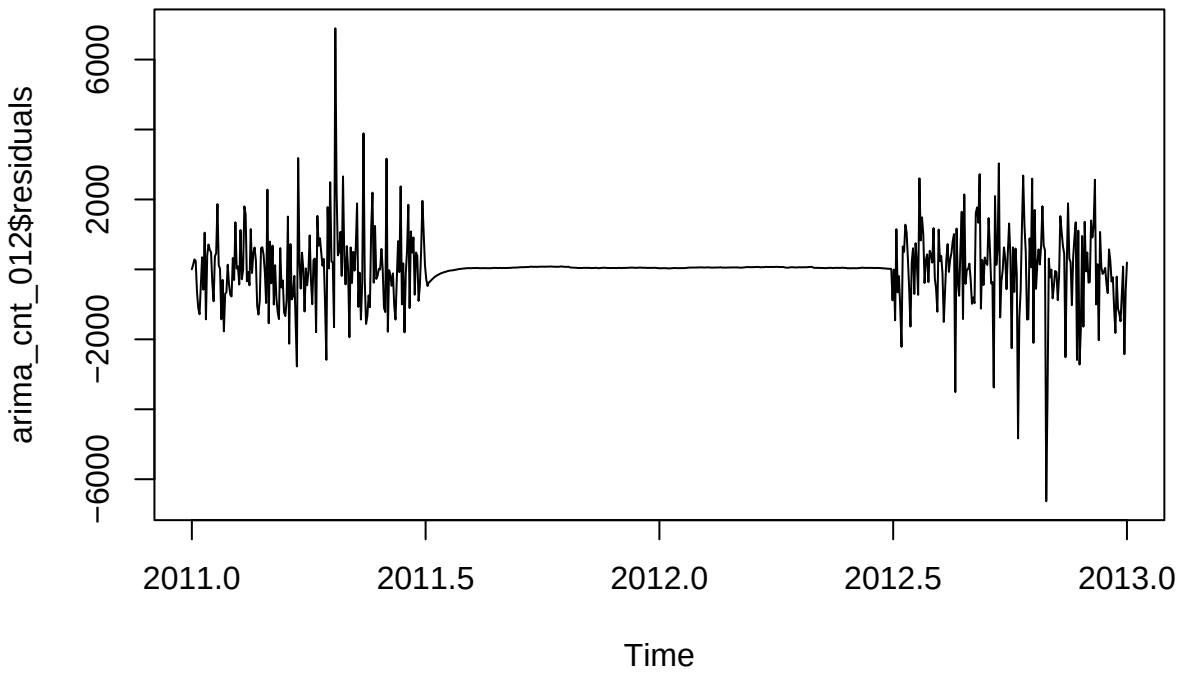



```
pacf(arima_cnt_012$residuals)
```



On voit que certains lags (ACF) dépassent l'intervalle de confiance de 95% –> possiblement auto corrélation différente de 0 pour certains résidus (ça se confirme avec le Test Q de Ljung-Box).

```
plot(arima_cnt_012$residuals)
```

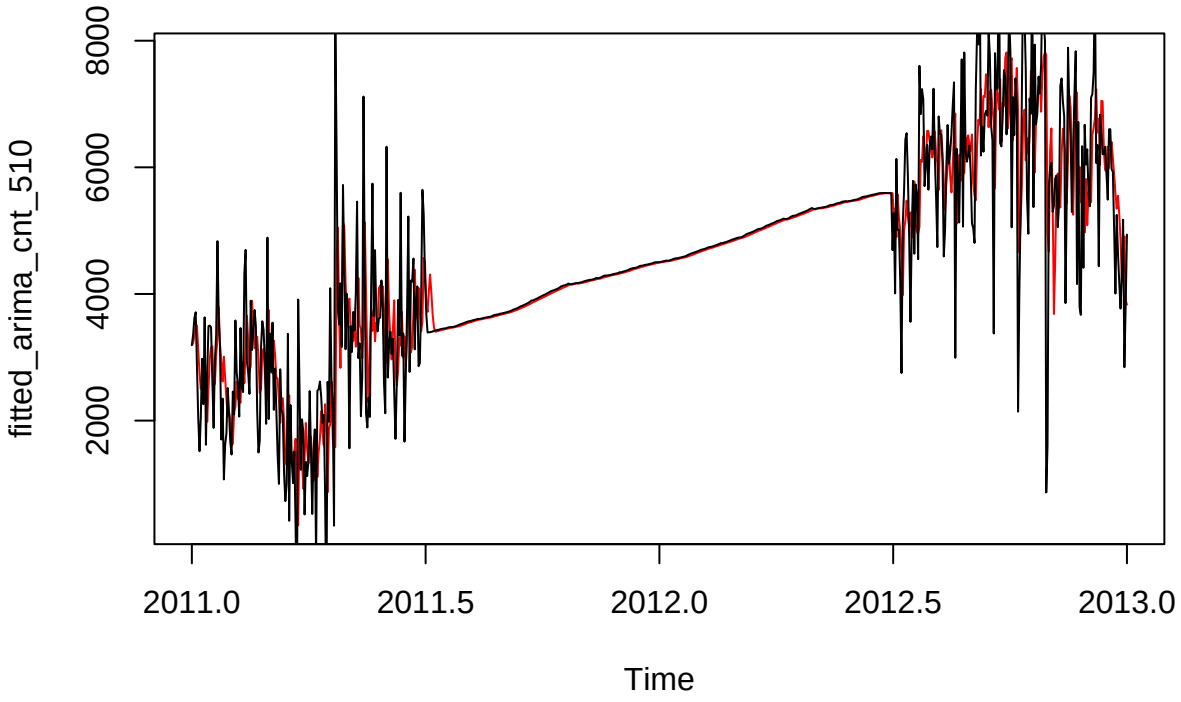



```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_cnt_012$residuals),digits=
```

```
## [1] "Moyenne des résidus:19.990000 et Variance des résidus: 802506.430000"
```

Résidus nulles pour la période mi 2011-mi 2012 –> logique car la série temporelle deseason_cnt ressemble à une droite linéaire sans bruit ni aléas (facile à prédire). C'est la tendance. Moyenne centrée en 0 et variance constante (avec quelques variations (pics hauts) à certaines périodes).

```
arima_cnt_510<-arima(deseason_cnt,order = c(5, 1,0))
fitted_arima_cnt_510 <-deseason_cnt-arima_cnt_510$residuals
plot(fitted_arima_cnt_510,col="red")
lines(deseason_cnt)
```




```
arima_cnt_510
```

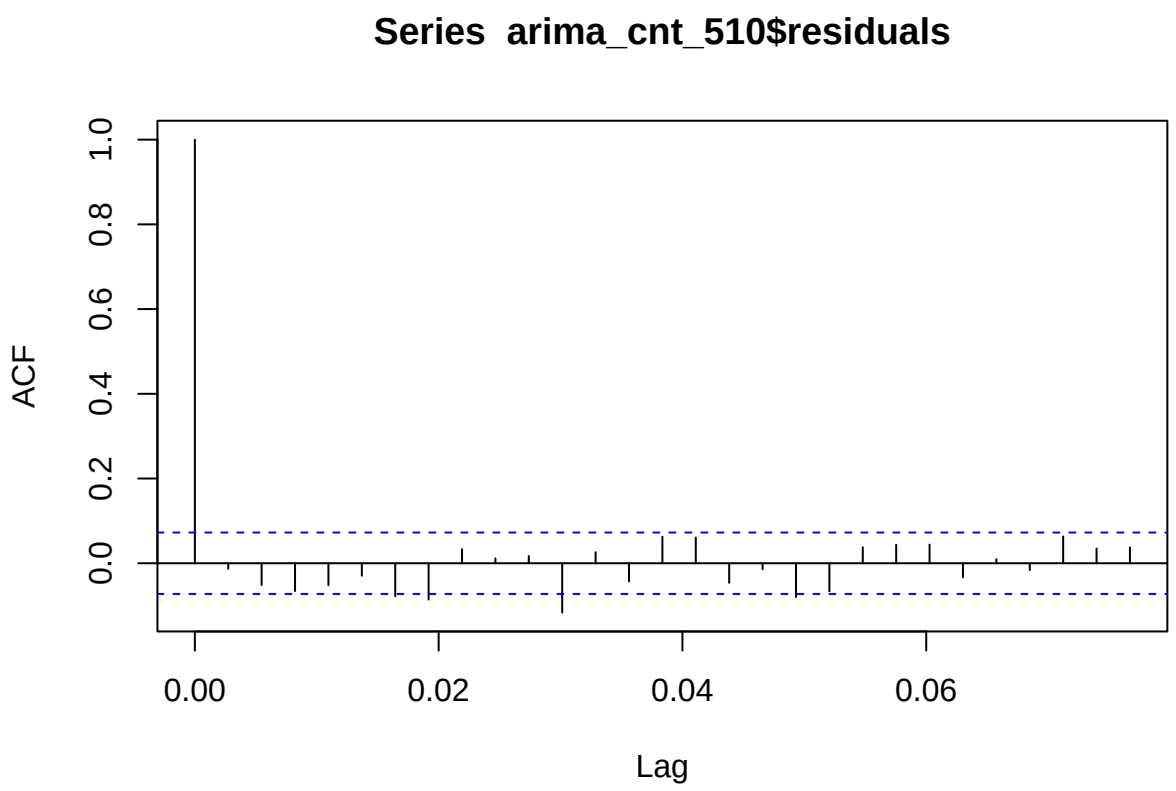
```
##
## Call:
## arima(x = deseason_cnt, order = c(5, 1, 0))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.6101  -0.4805  -0.4398  -0.4245  -0.2455
## s.e.   0.0359   0.0394   0.0400   0.0394   0.0360
##
## sigma^2 estimated as 803101:  log likelihood = -5998.97,  aic = 12009.93
```

```
Box.test(arima_cnt_510$residuals,lag=20,type="Ljung")
```

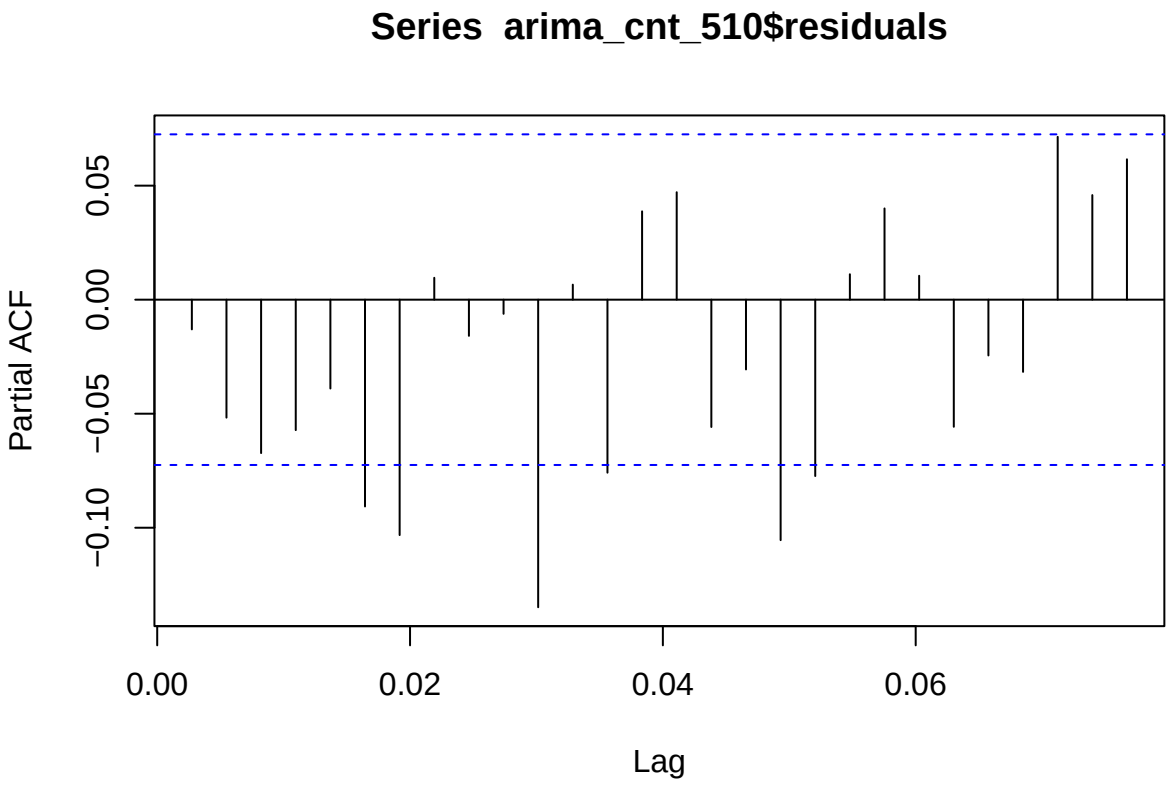
```
##
##  Box-Ljung test
##
## data:  arima_cnt_510$residuals
## X-squared = 47.527, df = 20, p-value = 0.0004955
```

–>Dépendance des 20 premiers résidus.

```
acf(arima_cnt_510$residuals)
```

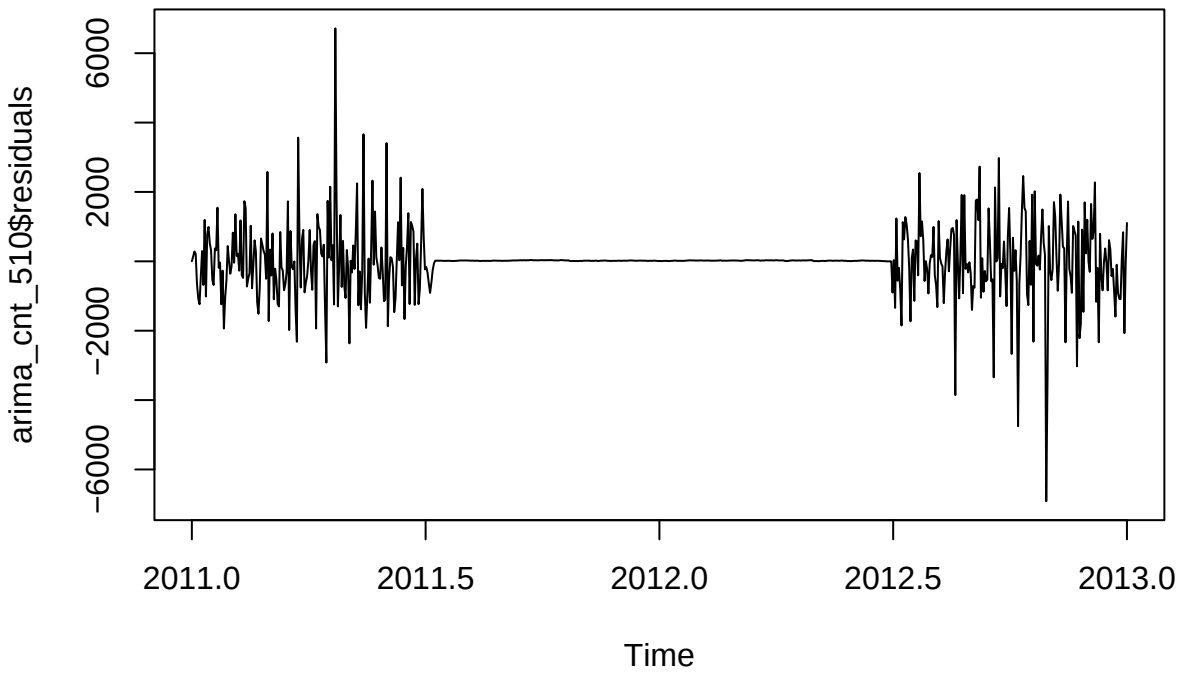



```
pacf(arima_cnt_510$residuals)
```



On voit que certains lags (ACF) dépassent l'intervalle de confiance de 95% –> possiblement auto corrélation différente de 0 pour certains résidus (ça se confirme avec le Test Q de Ljung-Box) mais comme la p-value plus grande que les deux autres modèles, c'est moins flagrant en regardant juste l'ACF.

```
plot(arima_cnt_510$residuals)
```




```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_cnt_510$residuals),digits=
```

```
## [1] "Moyenne des résidus:4.400000 et Variance des résidus: 803081.870000"
```

Résidus nulles pour la période mi 2011-mi 2012 –> logique car la série temporelle deseason_cnt ressemble à une droite linéaire sans bruit ni aléas (facile à prédire). C'est la tendance. Moyenne centrée en 0 et variance constante (avec quelques variations (pics hauts) à certaines périodes).

Conclusion: On remarque tout d'abord que dans les 3 modèles, les résidus ne sont pas indépendants, ce qui s'explique: La time series se sépare entre trois parties: avant milieu 2011 et après milieu 2012 –> time series stationnaire sans tendance. Entre les deux période: une tendance linéaire (droite) sans bruit (sans phénomènes aléatoires). Les modèles ARIMA prédisent très bien la droite (deterministe) ce qui conduit à des résidus très proche de 0. Cependant avant milieu 2011 et après milieu 2012 les résidus fluctuent beaucoup autour de 0 (bruit aléatoire). Ce qui rend les probabilités des valeurs des résidus dépendant du temps et donc ils ne sont pas indépendants. Les résidus ne sont donc pas des bruit blancs car pas indépendants dans les 3 modèles. Donc ces modèles ARIMA ne sont pas bons pour cette série mais si on devrait en choisir un on choisit ARIMA(0,1,2):

ARIMA 511: Log(vraisemblance)=-5989.55 AIC= 11993.1

ARIMA 510: Log(vraisemblance)=-5998.97 AIC=12009.93

ARIMA 012: Log(vraisemblance)=-5999.03 AIC=12004.06

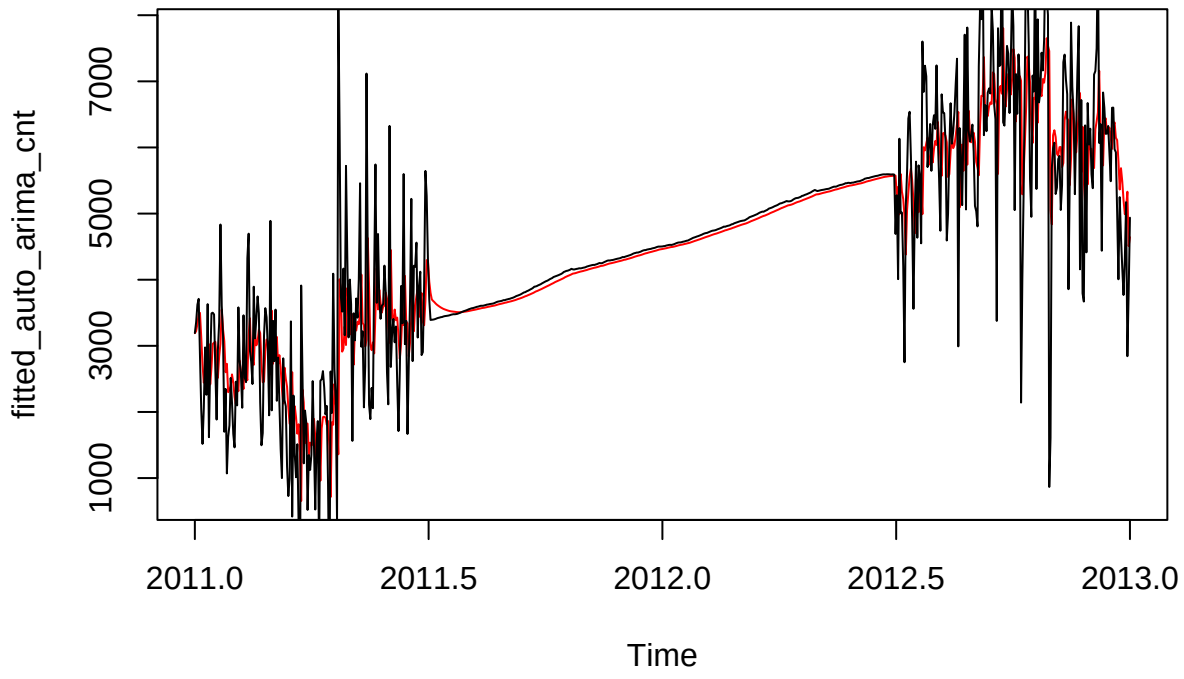
En effet la différence maximale d'AIC entre ARIMA(0,1,2) et les deux autres modèles est de l'ordre de 0.091%:

(12004.06-11993.1)/11993.1 ~ 0.00091.

Nous choisissons donc le modèle le moins complexe pour prévenir le risque d'overfitting ARIMA(0,1,2) car la différence d'AIC est vraiment minime entre les modèles.

4.II

```
auto_arima_cnt<-auto.arima(deseason_cnt,d=1)
fitted_auto_arima_cnt <-deseason_cnt-auto_arima_cnt$residuals
plot(fitted_auto_arima_cnt,col="red")
lines(deseason_cnt)
```



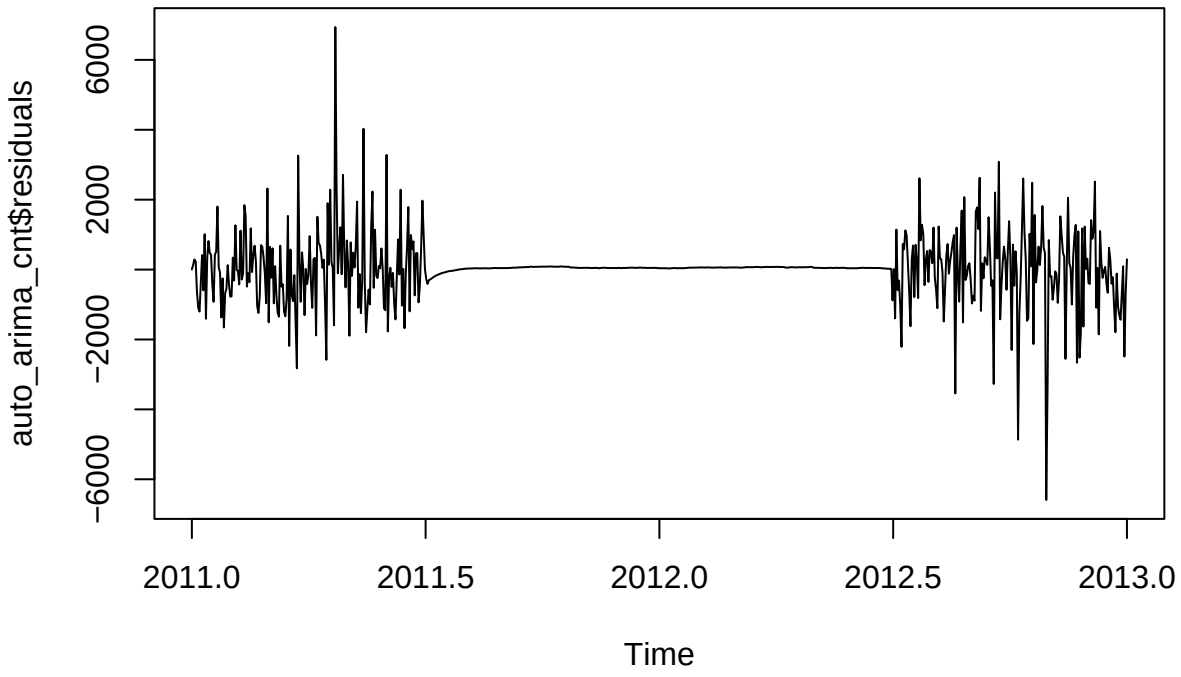

```
auto_arima_cnt
```

```
## Series: deseason_cnt
## ARIMA(0,1,3)
##
## Coefficients:
##           ma1      ma2      ma3
##       -0.6535  -0.1541  -0.0795
## s.e.   0.0366   0.0472   0.0428
##
## sigma^2 = 802357:  log likelihood = -5997.29
## AIC=12002.58   AICc=12002.64   BIC=12020.95
```

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

```
##
##  Box-Ljung test
##
## data:  auto_arima_cnt$residuals
## X-squared = 72.09, df = 20, p-value = 8.272e-08
```

```
plot(auto_arima_cnt$residuals)
```



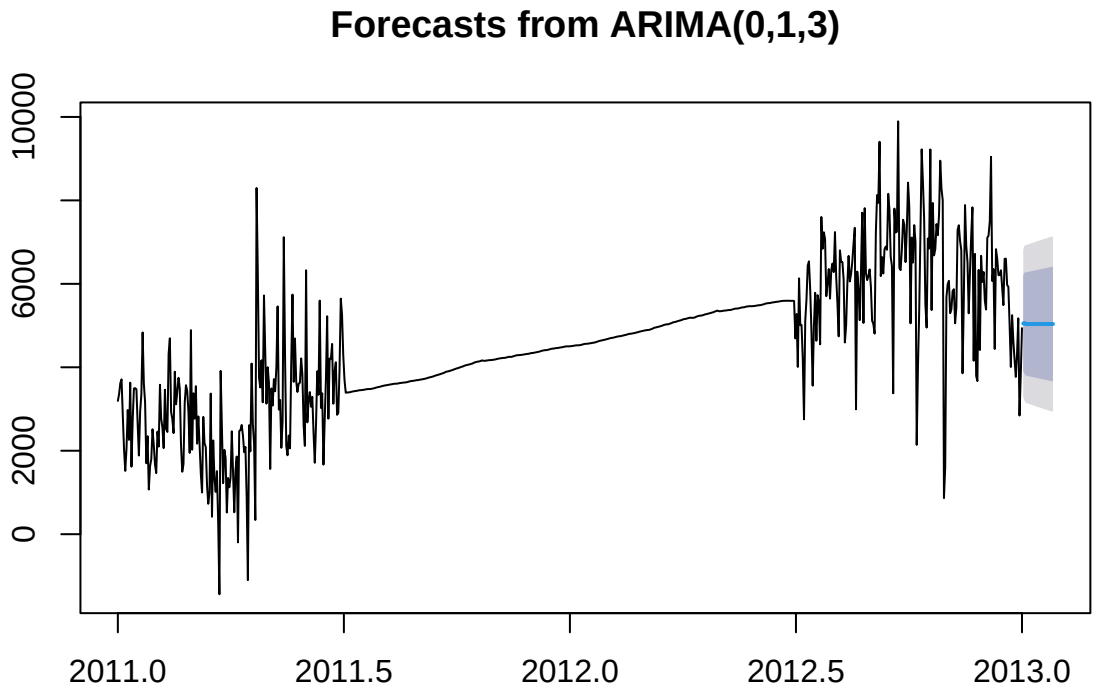

```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(auto_arima_cnt$residuals),digit
```

```
## [1] "Moyenne des résidus:24.050000 et Variance des résidus: 798480.410000"
```

On remarque, les résidus ne sont pas indépendants, ce qui s'explique: La time series se sépare entre trois parties: avant milieu 2011 et après milieu 2012 –> time series stationnaire sans tendance. Entre les deux période: une tendance linéaire (droite). Le modèle auto.ARIMA prédit très bien la droite (deterministe) ce qui conduit à des résidus très proche de 0. Cependant avant milieu 2011 et après milieu 2012 les résidus fluctuent beaucoup autour de 0 (bruit aléatoire). Ce qui rend les probabilités des valeurs des résidus dépendant du temps et donc ils ne sont pas indépendants.

4.III La démarche d'évaluations et d'itérations a été faite précédemment.

```
forecast_cnt_best_model <-forecast(auto_arima_cnt,h=25)
plot(forecast_cnt_best_model)
```



4.IV

```
deseason_cnt
```

```
## Time Series:
## Start = c(2011, 1)
## End = c(2013, 1)
## Frequency = 365
##   [1]  3194.0812  3357.4044  3622.0949  3710.5962  2849.0127  2033.4373
##   [7]  1516.6291  2059.8099  2976.4346  2259.4729  3630.7086  1618.2127
##  [13]  2751.8291  3491.7496  3501.0455  3474.2318  2639.6099  1884.8209
##  [19]  2942.0099  3353.7250  4834.6620  3607.9633  3167.1688  1700.4483
##  [25]  2348.2647  1070.3058  1621.1935  1797.9716  2514.7853  2139.4647
##  [31]  1667.5469  1464.0401  2458.1414  2099.2620  3581.6264  2766.8318
##  [37]  2556.6757  2062.7496  3461.7825  2515.6483  2450.3716  4324.1907
##  [43]  4696.7496  2929.1414  2759.8428  2422.0264  3894.0017  3114.1606
##  [49]  3412.3305  3747.5716  3491.8510  2145.0099  1496.0647  1681.2839
##  [55]  3157.3524  3570.1661  3432.5086  2939.5853  1949.1770  4890.0072
##  [61]  2024.2812  3377.4044  2768.2757  3547.9086  2168.9058  2822.5305
##  [67]  2135.5907  1439.1168   999.6291  2811.2921  2181.1880  2085.9798
##  [73]  1174.3469   727.8592   992.1305  3370.0483   417.8812  2247.6127
##  [79]  1347.7223  1012.4455  1512.2675   298.4949 -1439.2668  3912.1716
##  [85]  2580.9770  1222.6236  2021.5907  1832.4894   518.0401  1348.0976
##  [91]  1123.2839  1354.5031  2467.0784  1500.0072   524.5442  1523.3990
##  [97]  1862.6894  -196.1462  2470.6510  2494.5606  2618.0538  2368.6236
## [103]  1963.7442  2091.7140   898.2072 -1100.8969  2612.3880  1984.5086
```

```
## [109]  4090.9414  2646.8729  2170.2401   340.6127  8297.5058  6272.8401
## [115]  3744.8236  3517.0592  4168.5524  3160.6620  5721.9086  4365.4072
## [121]  3127.3414  4002.2921  3627.3962  1562.3277  3492.4592  3083.4510
## [127]  3716.6866  3426.0976  4004.6209  5458.8949  2986.1990  3218.5798
## [133]  2069.2592  2691.8894  7117.8812  4249.8072  2109.3688  1891.3086
## [139]  2363.5551  2056.6839  4112.4401  5742.8620  3650.7223  4691.4181
## [145]  3667.0318  3408.6483  3607.2373  3629.4729  4212.9962  3823.2483
## [151]  2601.6757  2115.3606  6325.6784  2680.1058  3191.9414  3406.4729
## [157]  3053.1305  3292.7195  2434.5140  1711.1332  2625.5086  3905.6127
## [163]  3340.2620  5598.2647  3025.1853  3377.0949  1667.3058  2711.3332
## [169]  3713.2017  5223.1031  2769.0044  4212.1606  4194.1825  4560.7031
## [175]  3125.4428  3905.5688  4121.0236  2860.1907  2905.2154  3937.7003
## [181]  5645.5414  5263.9990  4280.0921  3699.4812  3391.8099  3394.2401
## [187]  3396.4483  3401.0291  3407.8565  3416.1058  3422.8620  3427.1195
## [193]  3433.3579  3435.8620  3443.9031  3448.8565  3451.7935  3454.7058
## [199]  3457.7031  3463.0044  3470.3825  3474.8812  3478.2675  3477.6044
## [205]  3480.3332  3484.2949  3492.3031  3498.5633  3508.3414  3516.6291
## [211]  3524.4538  3530.3305  3537.2565  3545.4976  3554.3168  3560.4401
## [217]  3567.5661  3570.6455  3575.9661  3581.8866  3589.1825  3592.6675
## [223]  3598.7633  3605.0455  3606.2044  3606.3606  3611.3825  3616.8866
## [229]  3623.3360  3625.7743  3630.3743  3634.1853  3637.0729  3640.6812
## [235]  3647.9962  3657.1003  3665.7168  3670.3195  3673.8017  3677.6921
## [241]  3682.9524  3690.9442  3690.8976  3698.7223  3702.8565  3708.6702
## [247]  3712.3579  3719.8318  3725.5414  3733.1661  3742.7305  3753.5414
## [253]  3759.4072  3767.0209  3774.9140  3785.3277  3796.9880  3807.9469
## [259]  3812.4236  3825.0181  3832.6209  3842.7086  3853.7113  3863.3743
## [265]  3876.3880  3894.1880  3897.3716  3904.2209  3914.8099  3923.2318
## [271]  3932.1990  3944.7935  3955.1332  3966.1140  3976.4949  3983.8565
## [277]  3993.8757  4006.5907  4016.5880  4025.6812  4040.4373  4047.8729
## [283]  4055.2428  4062.2839  4070.0318  4078.9277  4087.5058  4099.3798
## [289]  4116.7414  4123.9360  4132.8729  4136.0592  4143.2401  4151.7360
## [295]  4165.2729  4157.0291  4154.3524  4158.6264  4163.5469  4166.7086
## [301]  4172.6675  4171.6401  4174.3579  4180.4428  4184.1113  4188.8181
## [307]  4199.1962  4204.3003  4210.5332  4215.0401  4220.3551  4224.0976
## [313]  4223.8620  4230.4099  4236.3442  4245.4510  4252.8729  4248.1880
## [319]  4251.3579  4260.4017  4270.0702  4278.4647  4287.7168  4291.3442
## [325]  4290.5195  4295.4483  4297.5524  4302.4620  4308.0976  4313.1853
## [331]  4318.2072  4321.6455  4326.1524  4334.0236  4343.2401  4340.9360
## [337]  4348.6291  4354.9277  4360.6592  4367.3798  4373.4675  4381.9414
## [343]  4392.4099  4400.3880  4404.8592  4410.8976  4410.7661  4417.6976
## [349]  4423.6784  4434.3579  4442.1880  4447.2812  4448.2538  4455.9661
## [355]  4459.7360  4463.5661  4466.3962  4473.1551  4477.7825  4481.8209
## [361]  4489.3113  4496.6729  4501.2044  4501.0620  4501.9524  4503.0812
## [367]  4507.4044  4509.0949  4516.5962  4521.0127  4525.4373  4527.6291
## [373]  4525.8099  4530.4346  4536.4729  4544.7086  4553.2127  4559.8291
## [379]  4563.7496  4564.0455  4568.2318  4574.6099  4577.8209  4584.0099
## [385]  4589.7250  4592.6620  4603.9633  4613.1688  4623.4483  4633.2647
## [391]  4639.3058  4646.1935  4653.9716  4659.7853  4667.4647  4675.5469
## [397]  4683.0401  4693.1414  4700.2620  4705.6264  4708.8318  4717.6757
## [403]  4725.7496  4733.7825  4740.6483  4743.3716  4747.1907  4753.7496
## [409]  4762.1414  4768.8428  4776.0264  4784.0017  4793.1606  4803.3305
## [415]  4801.5716  4808.8510  4815.0099  4819.0647  4826.2839  4837.3524
## [421]  4841.1661  4852.5086  4859.5853  4866.1770  4873.0072  4880.2812
## [427]  4886.4044  4890.2757  4893.9086  4896.9058  4906.5305  4918.5907
```

```
## [433]  4930.1168  4945.6291  4952.2921  4960.1880  4966.9798  4975.3469
## [439]  4983.8592  4992.1305  5004.0483  5014.8812  5022.6127  5029.7223
## [445]  5028.4455  5039.2675  5048.4949  5057.7332  5074.1716  5080.9770
## [451]  5087.6236  5095.5907  5105.4894  5115.0401  5122.0976  5131.2839
## [457]  5143.5031  5154.0784  5157.0072  5165.5442  5172.3990  5181.6894
## [463]  5189.8538  5184.6510  5184.5606  5188.0538  5196.6236  5210.7442
## [469]  5222.7140  5232.2072  5236.1031  5238.3880  5246.5086  5253.9414
## [475]  5267.8729  5271.2401  5281.6127  5288.5058  5295.8401  5304.8236
## [481]  5313.0592  5322.5524  5335.6620  5346.9086  5357.4072  5348.3414
## [487]  5341.2921  5345.3962  5350.3277  5355.4592  5358.4510  5361.6866
## [493]  5366.0976  5370.6209  5372.8949  5376.1990  5384.5798  5393.2592
## [499]  5400.8894  5407.8812  5406.8072  5410.3688  5420.3086  5427.5551
## [505]  5433.6839  5436.4401  5441.8620  5449.7223  5459.4181  5459.0318
## [511]  5465.6483  5464.2373  5462.4729  5467.9962  5468.2483  5474.6757
## [517]  5479.3606  5484.6784  5488.1058  5490.9414  5498.4729  5506.1305
## [523]  5514.7195  5527.5140  5532.1332  5537.5086  5537.6127  5544.2620
## [529]  5550.2647  5555.1853  5560.0949  5565.3058  5569.3332  5572.2017
## [535]  5578.1031  5584.0044  5588.1606  5592.1825  5593.7031  5592.4428
## [541]  5594.5688  5595.0236  5594.1907  5592.2154  5591.7003  5593.5414
## [547]  5588.9990  4692.0921  5277.4812  4008.8099  6132.2401  5008.4483
## [553]  5016.0291  4207.8565  2752.1058  5110.8620  5631.1195  6439.3579
## [559]  6539.8620  5858.9031  4879.8565  3559.7935  4982.7058  5785.7031
## [565]  4635.0044  5729.3825  5560.8812  4550.2675  7602.6044  6840.3332
## [571]  7236.2949  7075.3031  5703.5633  6022.3414  6355.6291  5646.4538
## [577]  6333.3305  6487.2565  6280.4976  7241.3168  6159.4401  5525.5661
## [583]  4740.6455  6803.9661  6528.8866  6521.1825  6098.6675  4592.7633
## [589]  4999.0455  6000.2044  6669.3606  6057.3825  6238.8866  6534.3360
## [595]  6968.7743  7342.3743  2992.1853  6294.0729  5888.6812  5127.9962
## [601]  6292.1003  7705.7168  5062.3195  7813.8017  6260.6921  6088.9524
## [607]  6183.9442  6345.8976  5933.7223  5115.8565  5034.6702  4806.3579
## [613]  7232.8318  8127.5414  7940.1661  9404.7305  6185.5414  6641.4072
## [619]  6246.0209  6828.9140  6892.3277  6815.9880  8157.9469  7766.4236
## [625]  6647.0181  6427.6209  3376.7086  7803.7113  7231.3743  7248.3880
## [631]  9894.1880  6381.3716  6330.2209  6822.8099  7536.2318  7418.1990
## [637]  6520.7935  7308.1332  8426.1140  7836.4949  5052.8565  7109.8757
## [643]  6508.5907  7407.5880  7005.6812  2141.4373  4014.8729  5330.2428
## [649]  7190.2839  9224.0318  8447.9277  7552.5058  5521.3798  4950.7414
## [655]  7087.9360  6845.8729  9221.0592  5372.2401  7937.7360  6681.2729
## [661]  6834.0291  7433.3524  7164.6264  7628.5469  8951.7086  8277.6675
## [667]  8003.6401   865.3579  1607.4428  5682.1113  5988.8181  6072.1962
## [673]  5296.3003  5391.5332  5825.0401  5871.3551  5054.0976  5429.8620
## [679]  7289.4099  7404.3442  7030.4510  6804.8729  3856.1880  5551.3579
## [685]  7888.4017  6915.0702  6515.4647  5293.7168  6270.3442  7159.5195
## [691]  7834.4483  4156.5524  6717.4620  3793.0976  3669.1853  6334.2072
## [697]  4413.6455  6672.1524  6044.0236  6284.2401  5591.9360  5383.6291
## [703]  7103.9277  7155.6592  7502.3798  9043.4675  6067.9414  6354.4099
## [709]  4438.3880  6831.8592  6601.8976  6206.7661  6209.6976  6325.6784
## [715]  5904.3579  5489.1880  6601.2812  6602.2538  5972.9661  5927.7360
## [721]  5018.5661  4006.3962  5249.1551  4643.7825  4177.8209  3768.3113
## [727]  4308.6729  5173.2044  2843.0620  3812.9524  4938.0812
```

```
training_set <- window(deseason_cnt,2011,end=2012.915)
test_set <-window(deseason_cnt,start=2012.915)
training_set
```

```
## Time Series:
## Start = c(2011, 1)
## End = c(2012, 334)
## Frequency = 365
##   [1]  3194.0812  3357.4044  3622.0949  3710.5962  2849.0127  2033.4373
##   [7]  1516.6291  2059.8099  2976.4346  2259.4729  3630.7086  1618.2127
##  [13]  2751.8291  3491.7496  3501.0455  3474.2318  2639.6099  1884.8209
##  [19]  2942.0099  3353.7250  4834.6620  3607.9633  3167.1688  1700.4483
##  [25]  2348.2647  1070.3058  1621.1935  1797.9716  2514.7853  2139.4647
##  [31]  1667.5469  1464.0401  2458.1414  2099.2620  3581.6264  2766.8318
##  [37]  2556.6757  2062.7496  3461.7825  2515.6483  2450.3716  4324.1907
##  [43]  4696.7496  2929.1414  2759.8428  2422.0264  3894.0017  3114.1606
##  [49]  3412.3305  3747.5716  3491.8510  2145.0099  1496.0647  1681.2839
##  [55]  3157.3524  3570.1661  3432.5086  2939.5853  1949.1770  4890.0072
##  [61]  2024.2812  3377.4044  2768.2757  3547.9086  2168.9058  2822.5305
##  [67]  2135.5907  1439.1168   999.6291  2811.2921  2181.1880  2085.9798
##  [73]  1174.3469   727.8592   992.1305  3370.0483   417.8812  2247.6127
##  [79]  1347.7223  1012.4455  1512.2675   298.4949 -1439.2668  3912.1716
##  [85]  2580.9770  1222.6236  2021.5907  1832.4894   518.0401  1348.0976
##  [91]  1123.2839  1354.5031  2467.0784  1500.0072   524.5442  1523.3990
##  [97]  1862.6894  -196.1462  2470.6510  2494.5606  2618.0538  2368.6236
## [103]  1963.7442  2091.7140   898.2072 -1100.8969  2612.3880  1984.5086
## [109]  4090.9414  2646.8729  2170.2401   340.6127  8297.5058  6272.8401
## [115]  3744.8236  3517.0592  4168.5524  3160.6620  5721.9086  4365.4072
## [121]  3127.3414  4002.2921  3627.3962  1562.3277  3492.4592  3083.4510
## [127]  3716.6866  3426.0976  4004.6209  5458.8949  2986.1990  3218.5798
## [133]  2069.2592  2691.8894  7117.8812  4249.8072  2109.3688  1891.3086
## [139]  2363.5551  2056.6839  4112.4401  5742.8620  3650.7223  4691.4181
## [145]  3667.0318  3408.6483  3607.2373  3629.4729  4212.9962  3823.2483
## [151]  2601.6757  2115.3606  6325.6784  2680.1058  3191.9414  3406.4729
## [157]  3053.1305  3292.7195  2434.5140  1711.1332  2625.5086  3905.6127
## [163]  3340.2620  5598.2647  3025.1853  3377.0949  1667.3058  2711.3332
## [169]  3713.2017  5223.1031  2769.0044  4212.1606  4194.1825  4560.7031
## [175]  3125.4428  3905.5688  4121.0236  2860.1907  2905.2154  3937.7003
## [181]  5645.5414  5263.9990  4280.0921  3699.4812  3391.8099  3394.2401
## [187]  3396.4483  3401.0291  3407.8565  3416.1058  3422.8620  3427.1195
## [193]  3433.3579  3435.8620  3443.9031  3448.8565  3451.7935  3454.7058
## [199]  3457.7031  3463.0044  3470.3825  3474.8812  3478.2675  3477.6044
## [205]  3480.3332  3484.2949  3492.3031  3498.5633  3508.3414  3516.6291
## [211]  3524.4538  3530.3305  3537.2565  3545.4976  3554.3168  3560.4401
## [217]  3567.5661  3570.6455  3575.9661  3581.8866  3589.1825  3592.6675
## [223]  3598.7633  3605.0455  3606.2044  3606.3606  3611.3825  3616.8866
## [229]  3623.3360  3625.7743  3630.3743  3634.1853  3637.0729  3640.6812
## [235]  3647.9962  3657.1003  3665.7168  3670.3195  3673.8017  3677.6921
## [241]  3682.9524  3690.9442  3690.8976  3698.7223  3702.8565  3708.6702
## [247]  3712.3579  3719.8318  3725.5414  3733.1661  3742.7305  3753.5414
## [253]  3759.4072  3767.0209  3774.9140  3785.3277  3796.9880  3807.9469
## [259]  3812.4236  3825.0181  3832.6209  3842.7086  3853.7113  3863.3743
## [265]  3876.3880  3894.1880  3897.3716  3904.2209  3914.8099  3923.2318
## [271]  3932.1990  3944.7935  3955.1332  3966.1140  3976.4949  3983.8565
## [277]  3993.8757  4006.5907  4016.5880  4025.6812  4040.4373  4047.8729
## [283]  4055.2428  4062.2839  4070.0318  4078.9277  4087.5058  4099.3798
## [289]  4116.7414  4123.9360  4132.8729  4136.0592  4143.2401  4151.7360
## [295]  4165.2729  4157.0291  4154.3524  4158.6264  4163.5469  4166.7086
```

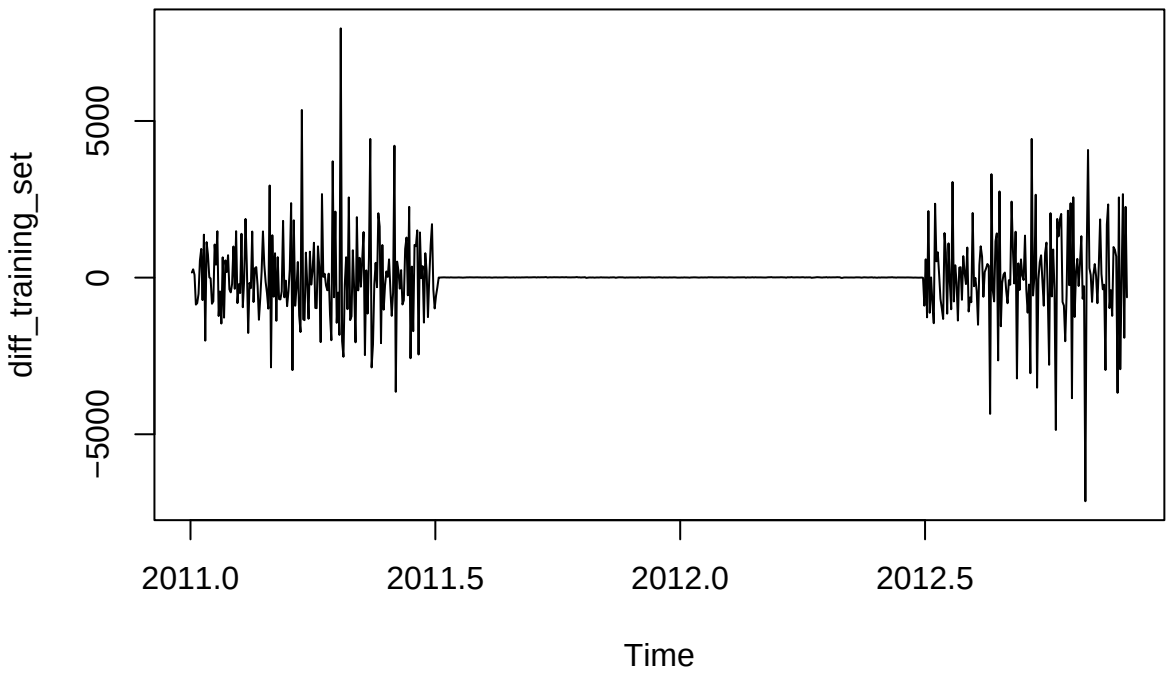
```
## [301]  4172.6675  4171.6401  4174.3579  4180.4428  4184.1113  4188.8181
## [307]  4199.1962  4204.3003  4210.5332  4215.0401  4220.3551  4224.0976
## [313]  4223.8620  4230.4099  4236.3442  4245.4510  4252.8729  4248.1880
## [319]  4251.3579  4260.4017  4270.0702  4278.4647  4287.7168  4291.3442
## [325]  4290.5195  4295.4483  4297.5524  4302.4620  4308.0976  4313.1853
## [331]  4318.2072  4321.6455  4326.1524  4334.0236  4343.2401  4340.9360
## [337]  4348.6291  4354.9277  4360.6592  4367.3798  4373.4675  4381.9414
## [343]  4392.4099  4400.3880  4404.8592  4410.8976  4410.7661  4417.6976
## [349]  4423.6784  4434.3579  4442.1880  4447.2812  4448.2538  4455.9661
## [355]  4459.7360  4463.5661  4466.3962  4473.1551  4477.7825  4481.8209
## [361]  4489.3113  4496.6729  4501.2044  4501.0620  4501.9524  4503.0812
## [367]  4507.4044  4509.0949  4516.5962  4521.0127  4525.4373  4527.6291
## [373]  4525.8099  4530.4346  4536.4729  4544.7086  4553.2127  4559.8291
## [379]  4563.7496  4564.0455  4568.2318  4574.6099  4577.8209  4584.0099
## [385]  4589.7250  4592.6620  4603.9633  4613.1688  4623.4483  4633.2647
## [391]  4639.3058  4646.1935  4653.9716  4659.7853  4667.4647  4675.5469
## [397]  4683.0401  4693.1414  4700.2620  4705.6264  4708.8318  4717.6757
## [403]  4725.7496  4733.7825  4740.6483  4743.3716  4747.1907  4753.7496
## [409]  4762.1414  4768.8428  4776.0264  4784.0017  4793.1606  4803.3305
## [415]  4801.5716  4808.8510  4815.0099  4819.0647  4826.2839  4837.3524
## [421]  4841.1661  4852.5086  4859.5853  4866.1770  4873.0072  4880.2812
## [427]  4886.4044  4890.2757  4893.9086  4896.9058  4906.5305  4918.5907
## [433]  4930.1168  4945.6291  4952.2921  4960.1880  4966.9798  4975.3469
## [439]  4983.8592  4992.1305  5004.0483  5014.8812  5022.6127  5029.7223
## [445]  5028.4455  5039.2675  5048.4949  5057.7332  5074.1716  5080.9770
## [451]  5087.6236  5095.5907  5105.4894  5115.0401  5122.0976  5131.2839
## [457]  5143.5031  5154.0784  5157.0072  5165.5442  5172.3990  5181.6894
## [463]  5189.8538  5184.6510  5184.5606  5188.0538  5196.6236  5210.7442
## [469]  5222.7140  5232.2072  5236.1031  5238.3880  5246.5086  5253.9414
## [475]  5267.8729  5271.2401  5281.6127  5288.5058  5295.8401  5304.8236
## [481]  5313.0592  5322.5524  5335.6620  5346.9086  5357.4072  5348.3414
## [487]  5341.2921  5345.3962  5350.3277  5355.4592  5358.4510  5361.6866
## [493]  5366.0976  5370.6209  5372.8949  5376.1990  5384.5798  5393.2592
## [499]  5400.8894  5407.8812  5406.8072  5410.3688  5420.3086  5427.5551
## [505]  5433.6839  5436.4401  5441.8620  5449.7223  5459.4181  5459.0318
## [511]  5465.6483  5464.2373  5462.4729  5467.9962  5468.2483  5474.6757
## [517]  5479.3606  5484.6784  5488.1058  5490.9414  5498.4729  5506.1305
## [523]  5514.7195  5527.5140  5532.1332  5537.5086  5537.6127  5544.2620
## [529]  5550.2647  5555.1853  5560.0949  5565.3058  5569.3332  5572.2017
## [535]  5578.1031  5584.0044  5588.1606  5592.1825  5593.7031  5592.4428
## [541]  5594.5688  5595.0236  5594.1907  5592.2154  5591.7003  5593.5414
## [547]  5588.9990  4692.0921  5277.4812  4008.8099  6132.2401  5008.4483
## [553]  5016.0291  4207.8565  2752.1058  5110.8620  5631.1195  6439.3579
## [559]  6539.8620  5858.9031  4879.8565  3559.7935  4982.7058  5785.7031
## [565]  4635.0044  5729.3825  5560.8812  4550.2675  7602.6044  6840.3332
## [571]  7236.2949  7075.3031  5703.5633  6022.3414  6355.6291  5646.4538
## [577]  6333.3305  6487.2565  6280.4976  7241.3168  6159.4401  5525.5661
## [583]  4740.6455  6803.9661  6528.8866  6521.1825  6098.6675  4592.7633
## [589]  4999.0455  6000.2044  6669.3606  6057.3825  6238.8866  6534.3360
## [595]  6968.7743  7342.3743  2992.1853  6294.0729  5888.6812  5127.9962
## [601]  6292.1003  7705.7168  5062.3195  7813.8017  6260.6921  6088.9524
## [607]  6183.9442  6345.8976  5933.7223  5115.8565  5034.6702  4806.3579
## [613]  7232.8318  8127.5414  7940.1661  9404.7305  6185.5414  6641.4072
## [619]  6246.0209  6828.9140  6892.3277  6815.9880  8157.9469  7766.4236
```

```
## [625]  6647.0181  6427.6209  3376.7086  7803.7113  7231.3743  7248.3880
## [631]  9894.1880  6381.3716  6330.2209  6822.8099  7536.2318  7418.1990
## [637]  6520.7935  7308.1332  8426.1140  7836.4949  5052.8565  7109.8757
## [643]  6508.5907  7407.5880  7005.6812  2141.4373  4014.8729  5330.2428
## [649]  7190.2839  9224.0318  8447.9277  7552.5058  5521.3798  4950.7414
## [655]  7087.9360  6845.8729  9221.0592  5372.2401  7937.7360  6681.2729
## [661]  6834.0291  7433.3524  7164.6264  7628.5469  8951.7086  8277.6675
## [667]  8003.6401   865.3579  1607.4428  5682.1113  5988.8181  6072.1962
## [673]  5296.3003  5391.5332  5825.0401  5871.3551  5054.0976  5429.8620
## [679]  7289.4099  7404.3442  7030.4510  6804.8729  3856.1880  5551.3579
## [685]  7888.4017  6915.0702  6515.4647  5293.7168  6270.3442  7159.5195
## [691]  7834.4483  4156.5524  6717.4620  3793.0976  3669.1853  6334.2072
## [697]  4413.6455  6672.1524  6044.0236
```

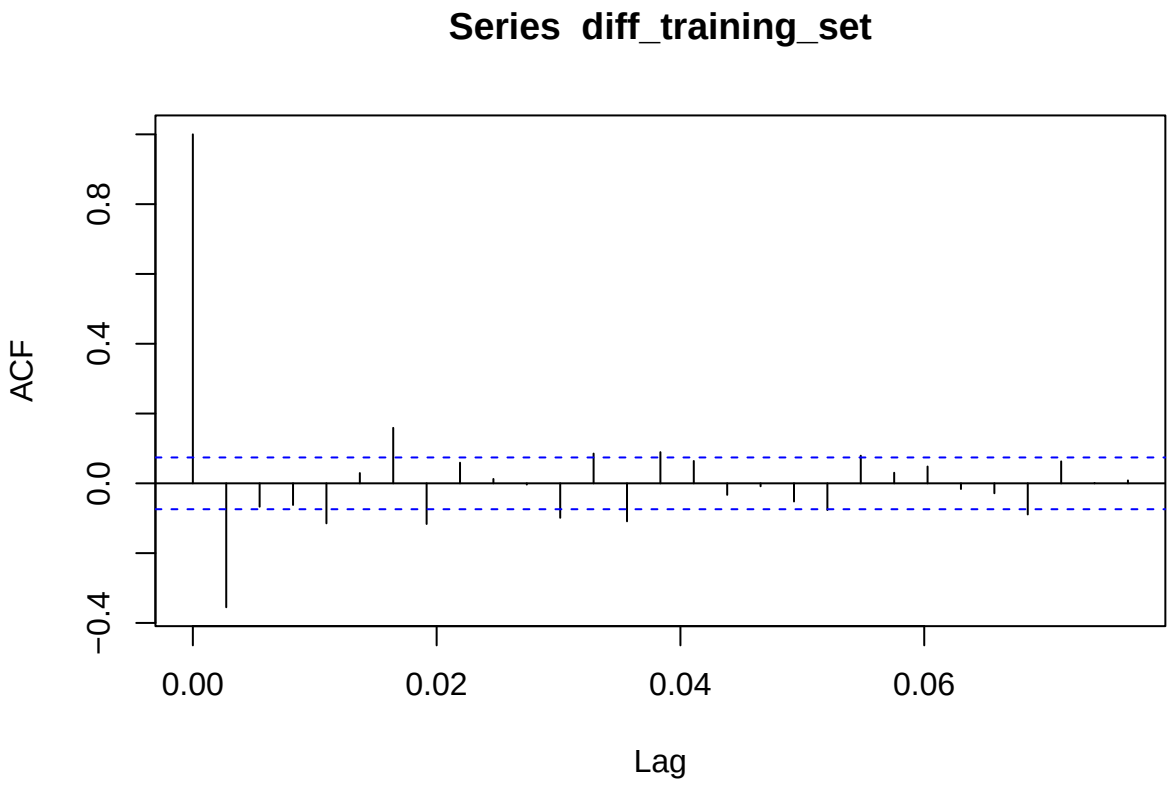
```
test_set
```

```
## Time Series:
## Start = c(2012, 335)
## End = c(2013, 1)
## Frequency = 365
##   [1] 6284.240 5591.936 5383.629 7103.928 7155.659 7502.380 9043.467 6067.941
##   [9] 6354.410 4438.388 6831.859 6601.898 6206.766 6209.698 6325.678 5904.358
## [17] 5489.188 6601.281 6602.254 5972.966 5927.736 5018.566 4006.396 5249.155
## [25] 4643.783 4177.821 3768.311 4308.673 5173.204 2843.062 3812.952 4938.081
```

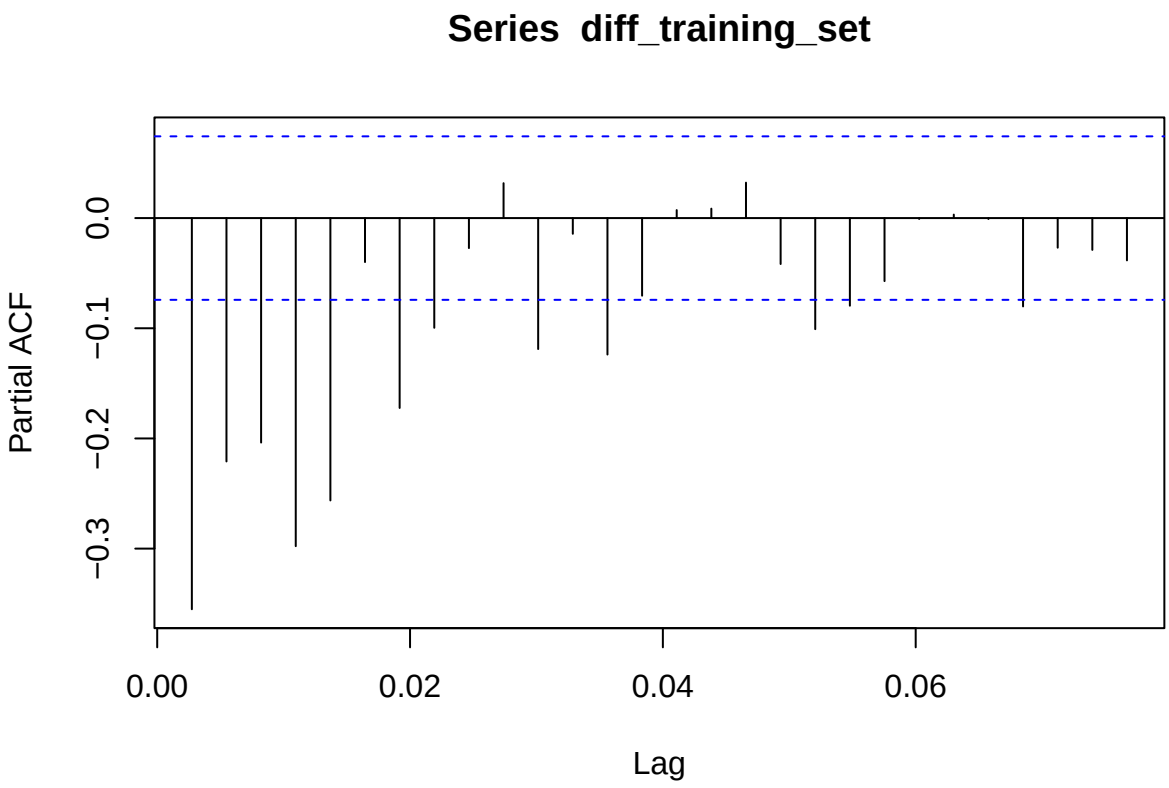
```
diff_training_set <- diff(training_set,differences=1) #on obtient une time series stationnaire
plot(diff_training_set)
```




```
acf(diff_training_set)
```




```
pacf(diff_training_set)
```



ACF et PACF tend vers 0 de manière exponentielle:

ARIMA(5,1,0) d'après le PACF

ARIMA(0,1,1) d'après l'ACF

ARIMA(5,1,1)

```
arima_training_set_510 <- arima(diff_training_set,order=c(5,1,0))
arima_training_set_510
```

```
##
## Call:
## arima(x = diff_training_set, order = c(5, 1, 0))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -1.1789  -1.0810  -0.9147  -0.7397  -0.4123
## s.e.   0.0346   0.0492   0.0537   0.0492   0.0348
##
## sigma^2 estimated as 1159512:  log likelihood = -5856.48,  aic = 11724.96
```

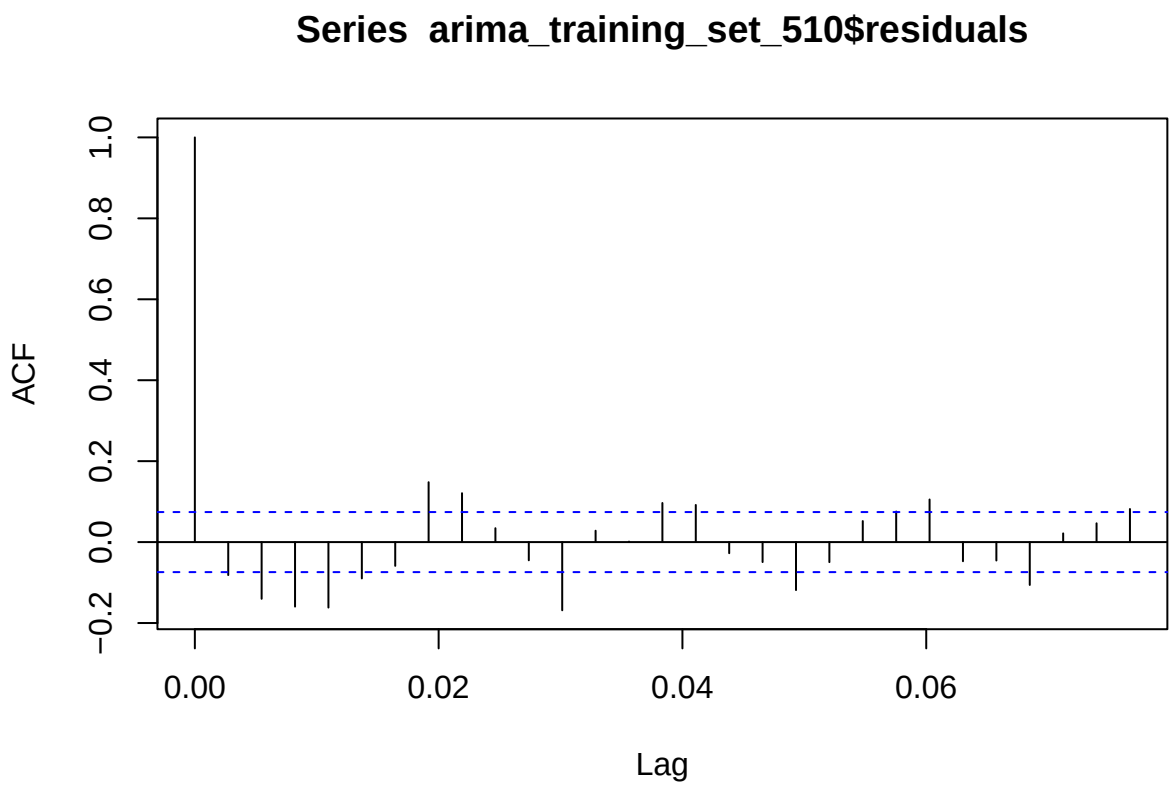
```
Box.test(arima_training_set_510$residuals,lag=20,type="Ljung")
```

```
##
##  Box-Ljung test
```

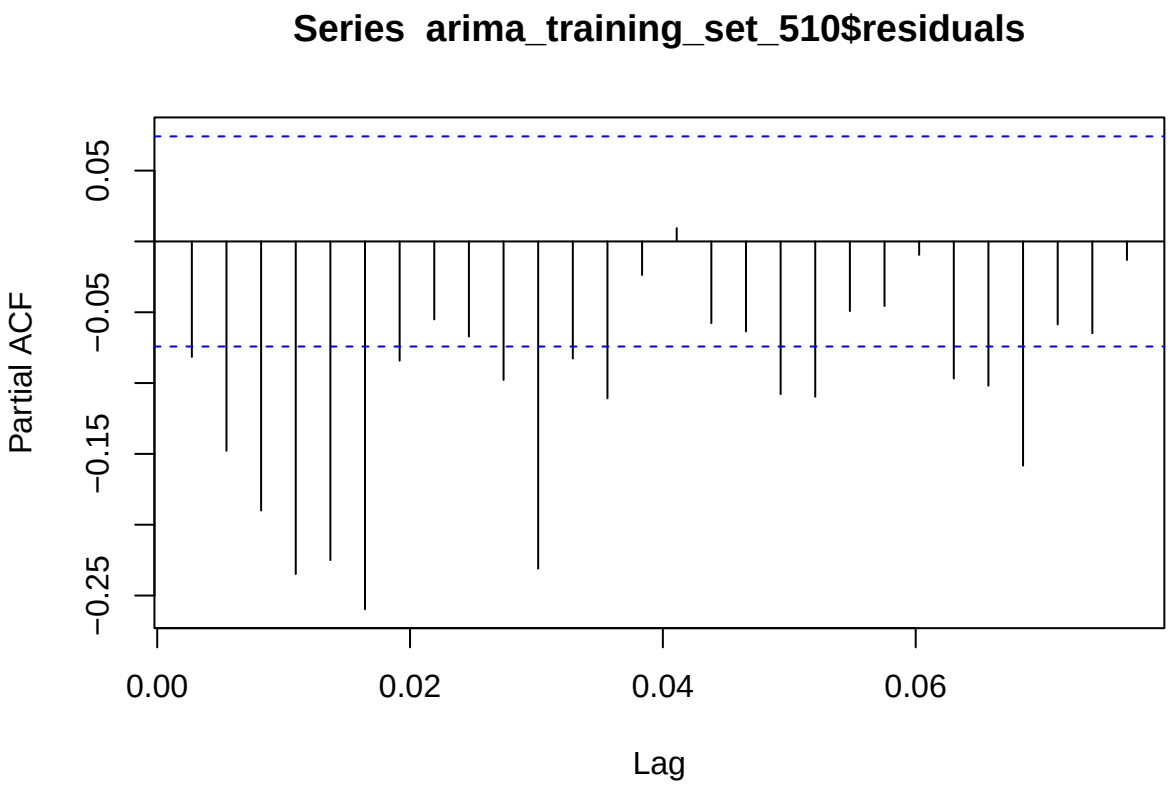
```
##
## data:  arima_training_set_510$residuals
## X-squared = 140.57, df = 20, p-value < 2.2e-16
```

Rejet de l'hypothèse: Résidus (les 20 premiers) autocorrélés (2.2e-16<0.05)

```
acf(arima_training_set_510$residuals)
```

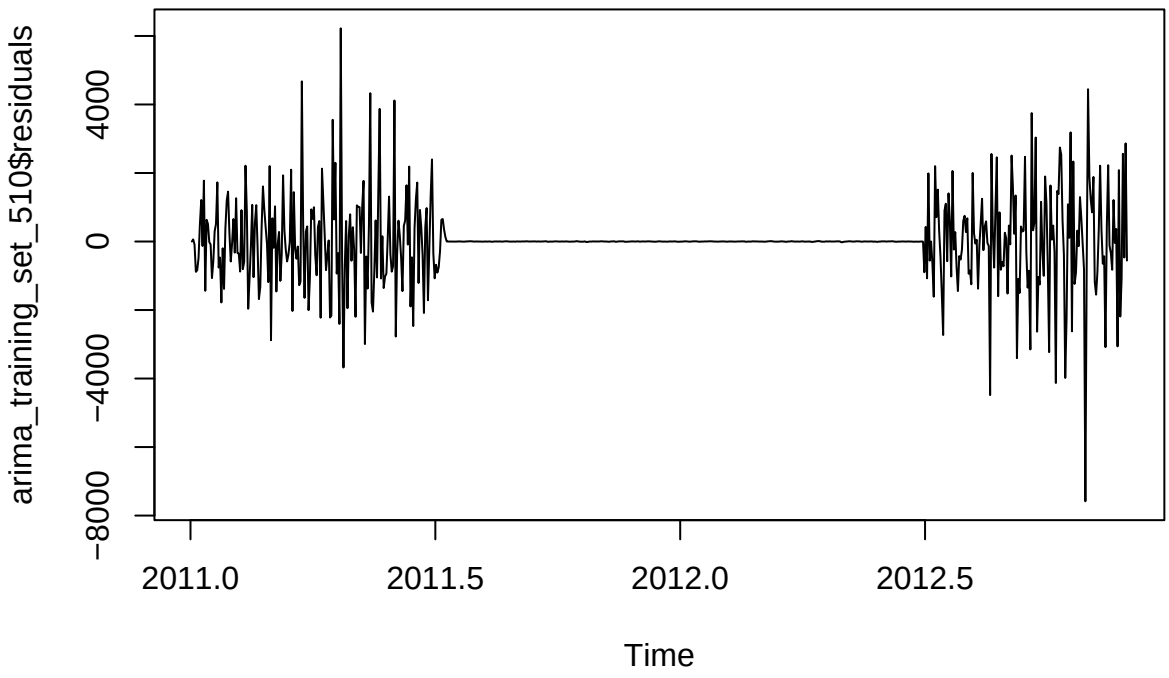



```
pacf(arima_training_set_510$residuals)
```



Logique d'après le test précédent.

```
plot(arima_training_set_510$residuals)
```




```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_training_set_510$residuals
```

```
## [1] "Moyenne des résidus:1.340000 et Variance des résidus: 1159510.200000"
```

résidus un peu près centrés + variance un peu près constante (d'après le graphique).

On obtient des meilleures résultats pour ARIMA(0,1,2) que pour ARIMA(0,1,1): On utilise ARIMA(0,1,2).

```
arima_training_set_012 <- arima(diff_training_set,order=c(0,1,2))
arima_training_set_012
```

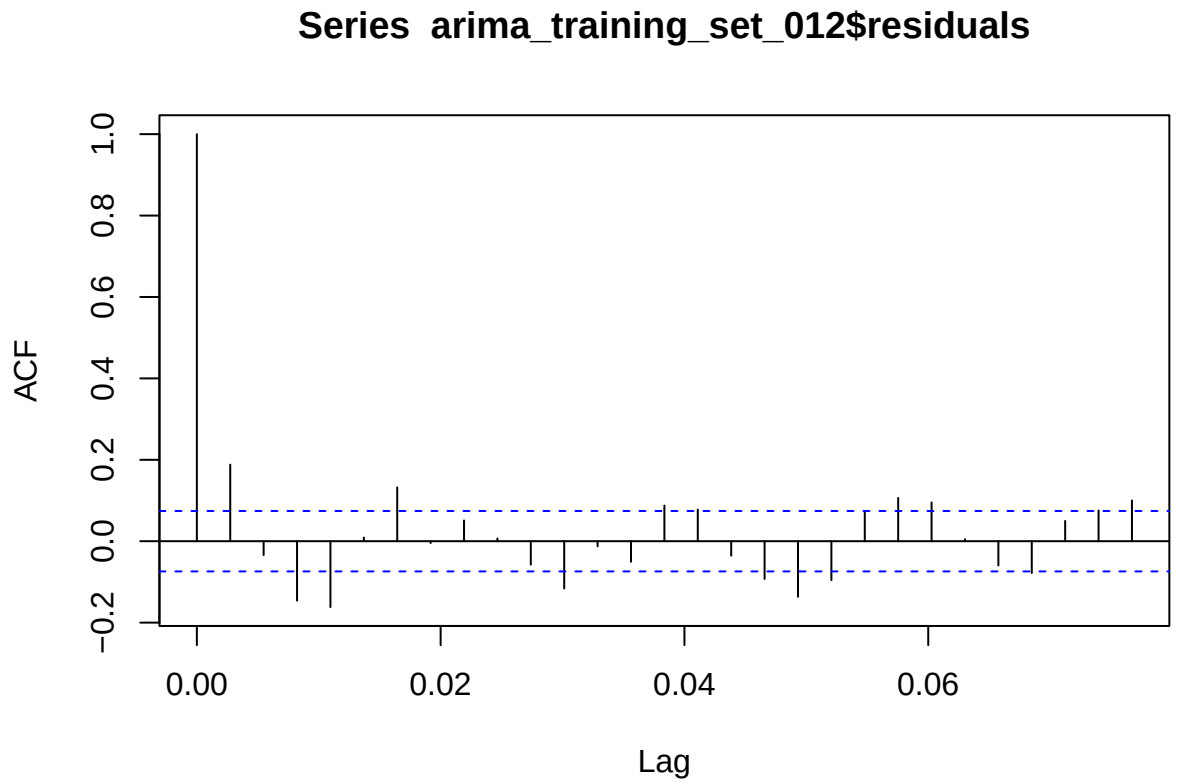
```
##
## Call:
## arima(x = diff_training_set, order = c(0, 1, 2))
##
## Coefficients:
##           ma1     ma2
##       -1.8600  0.8600
## s.e.   0.0254  0.0252
##
## sigma^2 estimated as 827219:  log likelihood = -5743.5,  aic = 11493.01
```

```
Box.test(arima_training_set_012$residuals,lag=20,type="Ljung")
```

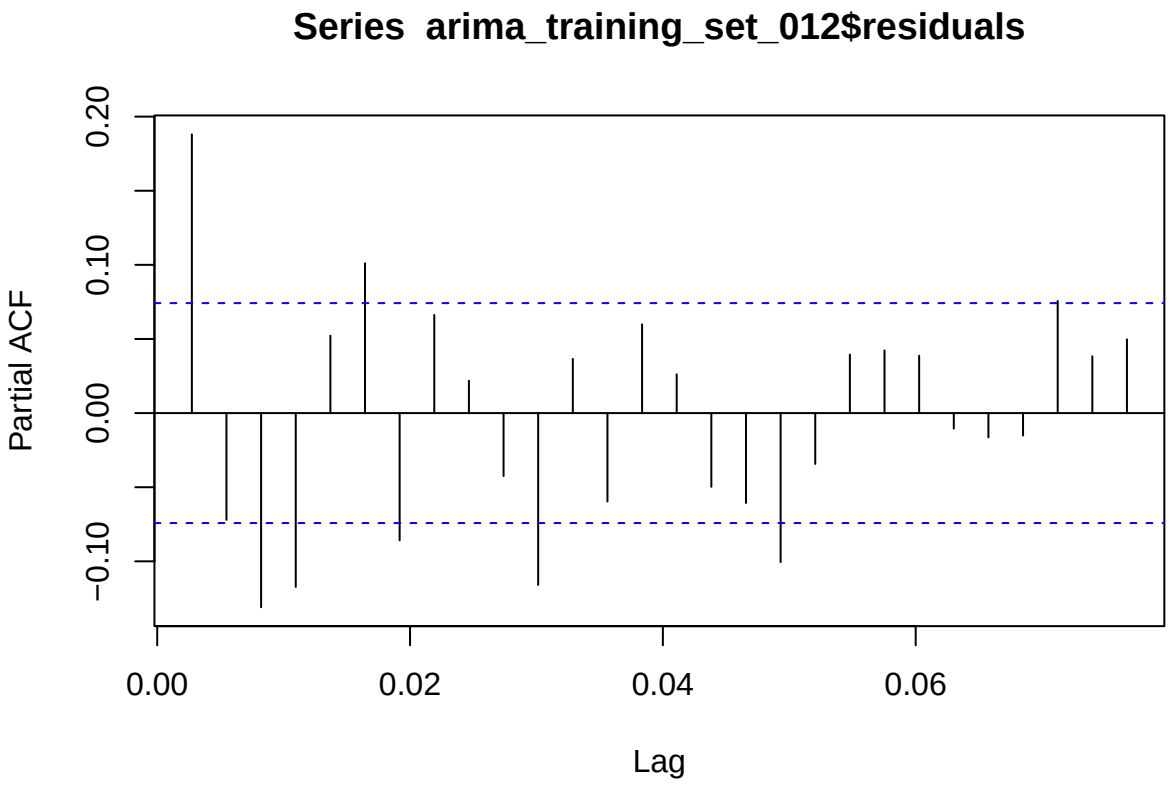
```
##
##  Box-Ljung test
##
## data:  arima_training_set_012$residuals
## X-squared = 127.67, df = 20, p-value < 2.2e-16
```

Rejet de l'hypothèse Résidus: (les 20 premiers) autocorrélés (2.2e-16<0.05).

```
acf(arima_training_set_012$residuals)
```

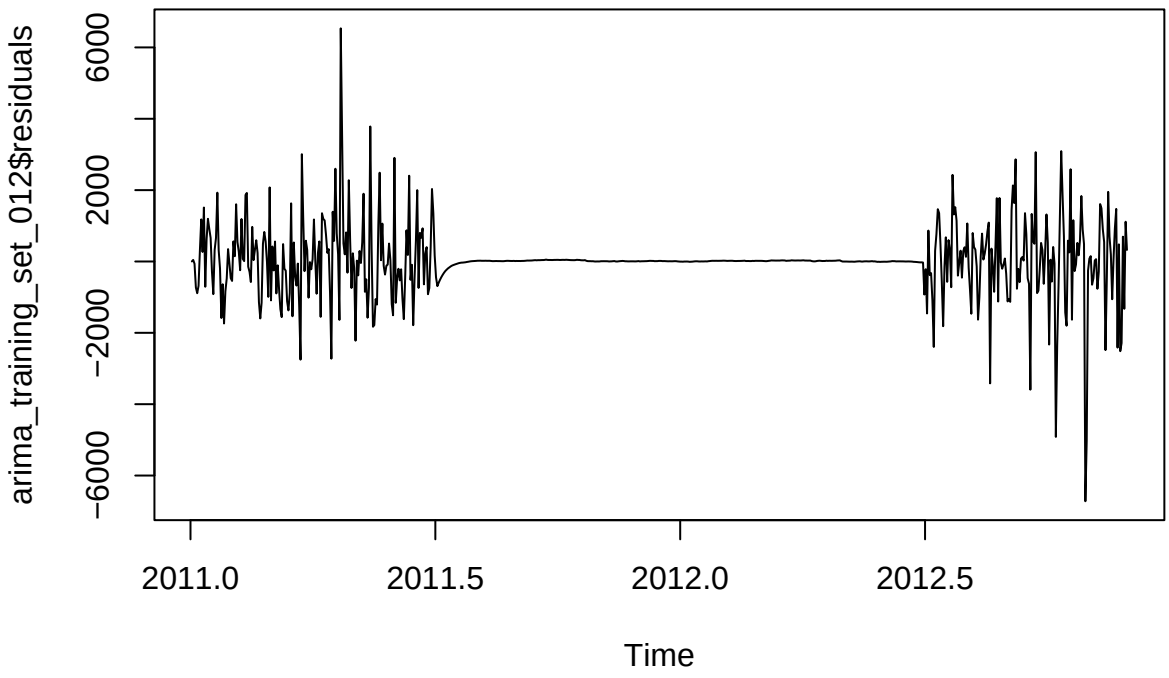



```
pacf(arima_training_set_012$residuals)
```



Logique d'après le test précédent.

```
plot(arima_training_set_012$residuals)
```




```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_training_set_012$residuals
```

```
## [1] "Moyenne des résidus:17.850000 et Variance des résidus: 826900.180000"
```

résidus un peu près centrés + variance un peu près constante (d'après le graphique).

```
arima_training_set_511 <- arima(diff_training_set,order=c(5,1,1))
arima_training_set_511
```

```
##
## Call:
## arima(x = diff_training_set, order = c(5, 1, 1))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ma1
##       -0.6158  -0.4924  -0.4495  -0.4386  -0.2590  -1.0000
## s.e.   0.0366   0.0403   0.0409   0.0403   0.0369   0.0038
##
## sigma^2 estimated as 792949:  log likelihood = -5727.85,  aic = 11469.69
```

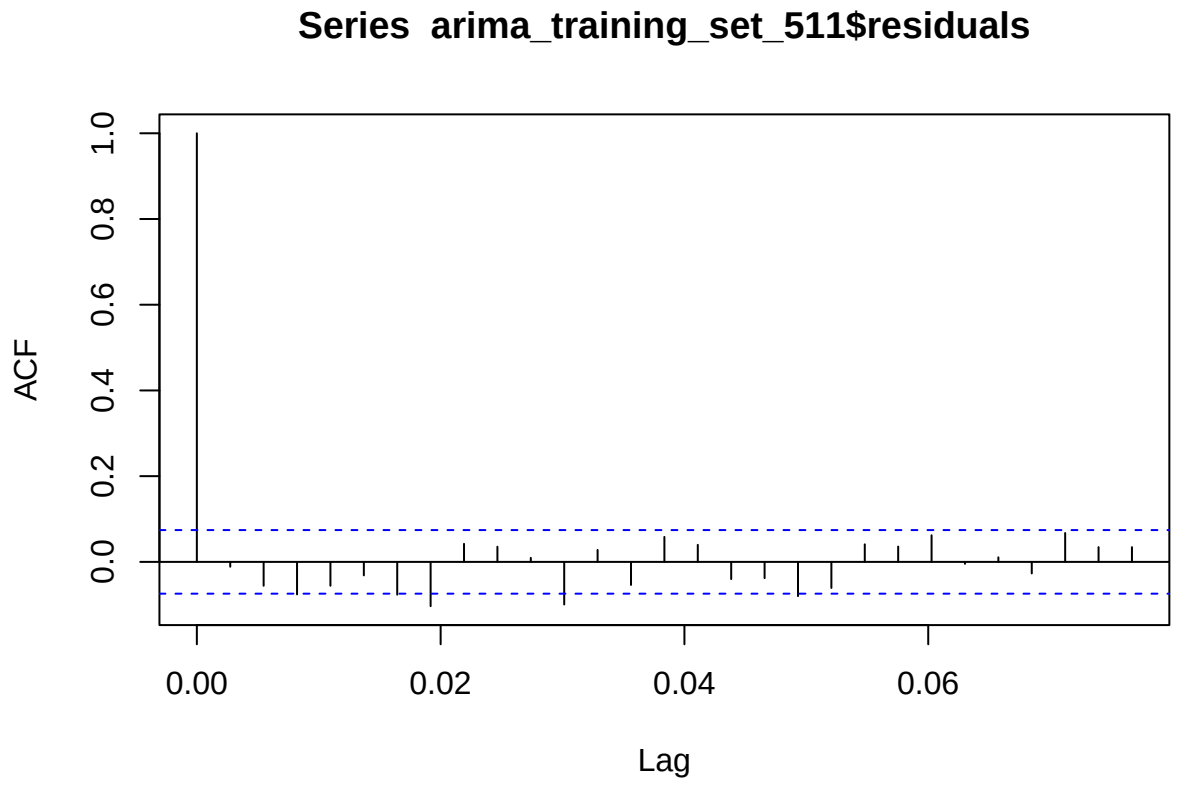
```
Box.test(arima_training_set_511$residuals,lag=20,type="Ljung")
```

```
##
```

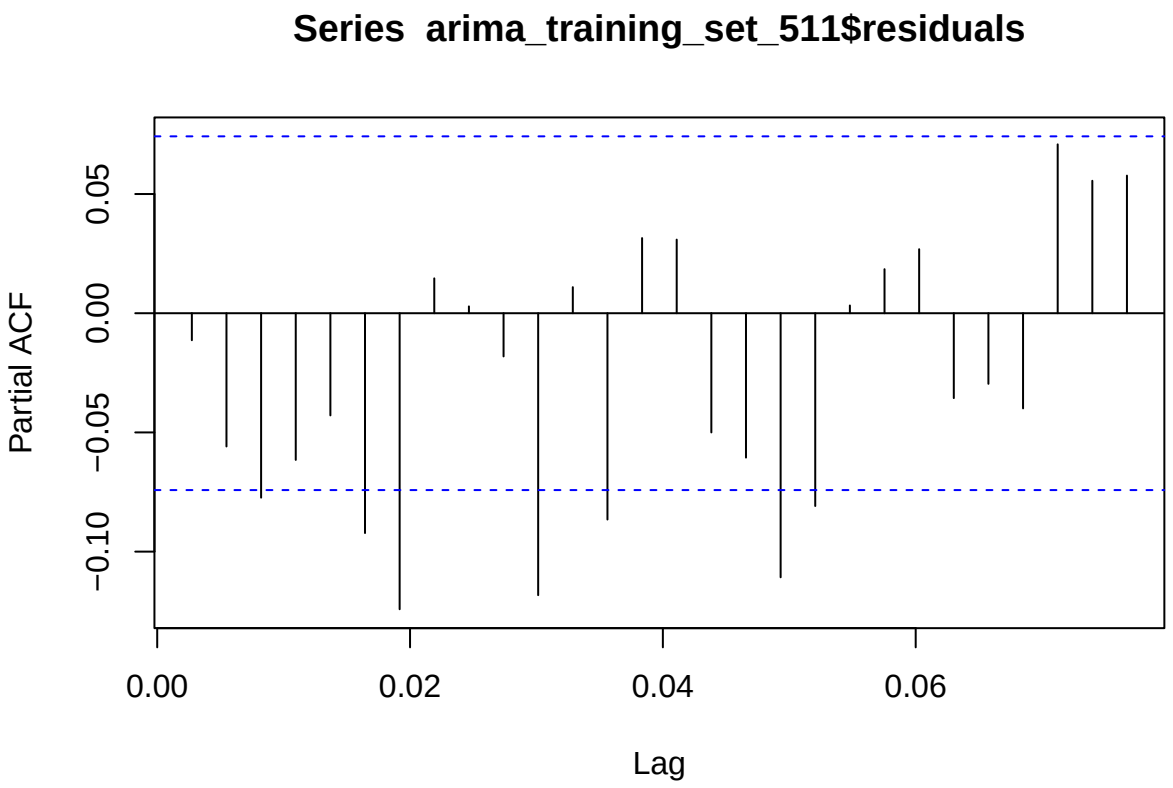
```
##  Box-Ljung test
##
## data:  arima_training_set_511$residuals
## X-squared = 46.945, df = 20, p-value = 0.000597
```

Rejet de l'hypothèse: Résidus (les 20 premiers) autocorrélés (0.000597<0.05).

```
acf(arima_training_set_511$residuals)
```

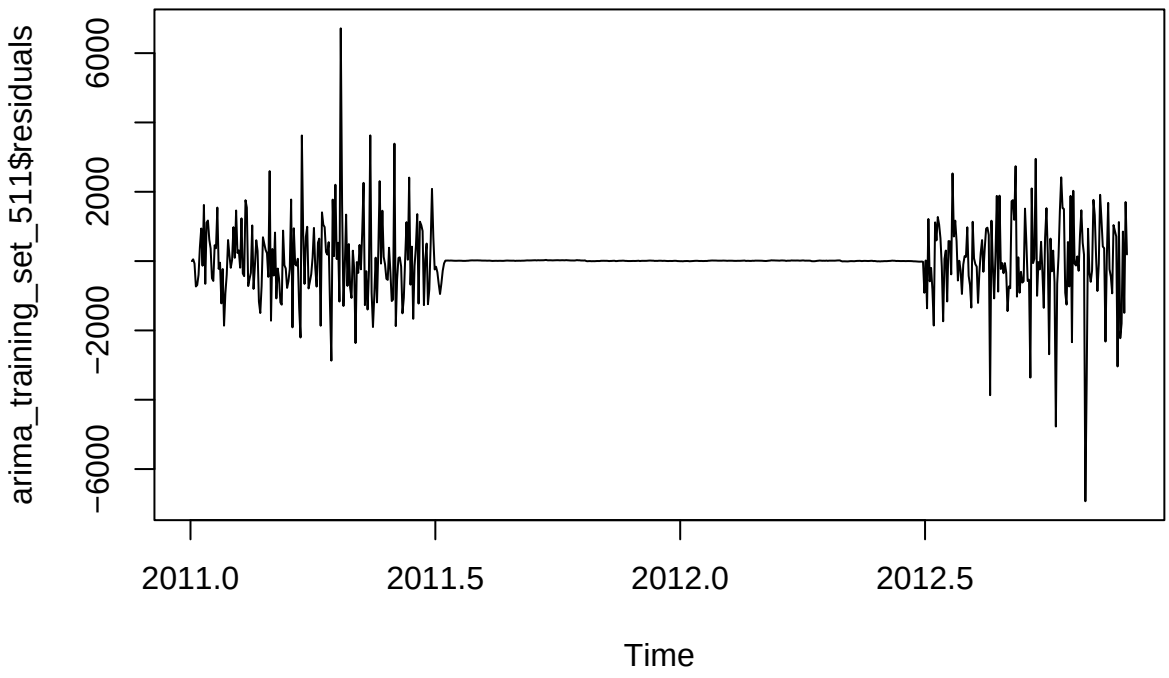



```
pacf(arima_training_set_511$residuals)
```



On observe qu'il y a moins d'auto corrélations ou d'auto corrélations partielles que les deux autres modèles –> logique car la p-value du test ci-dessus est plus grande que pour les deux autres modèles précédents.

```
plot(arima_training_set_511$residuals)
```




```
sprintf("Moyenne des résidus:%f et Variance des résidus: %f",round(mean(arima_training_set_511$residuals
```

```
## [1] "Moyenne des résidus:13.310000 et Variance des résidus: 792771.040000"
```

résidus un peu près centrés + variance un peu près constante (d'après le graphique).

AIC=11469.69 –>ARIMA(5,1,1)

AIC=11493.01 –>ARIMA(0,1,2)

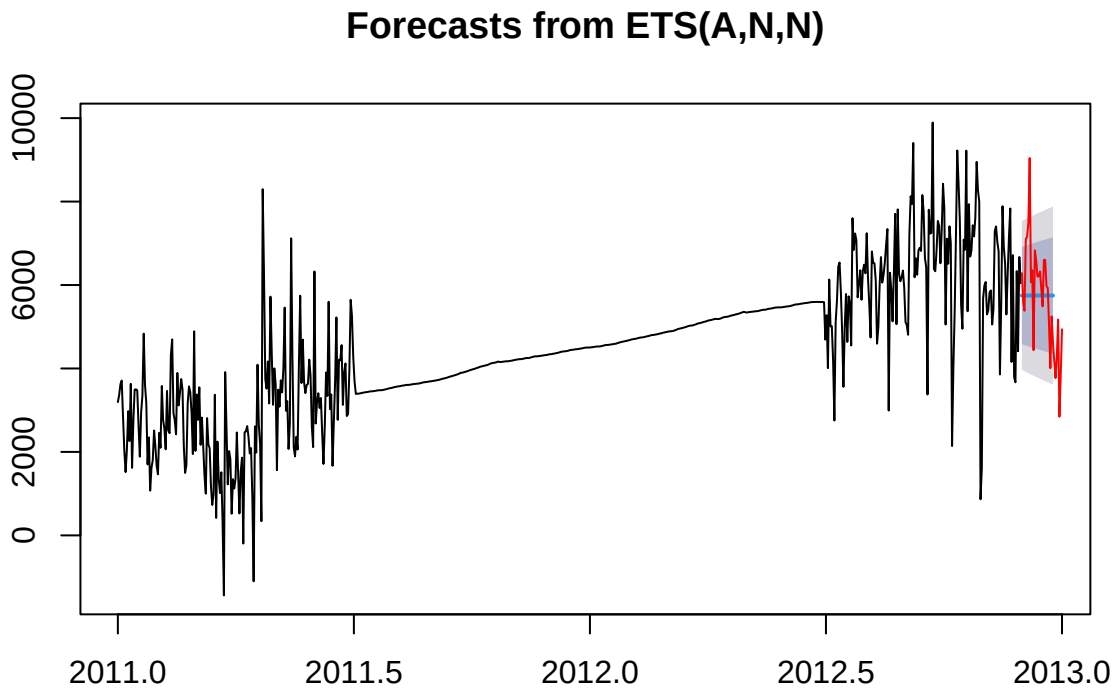
AIC=11724.96 –>ARIMA(5,1,0)

(*) Modèle le plus petit AIC mais 6 paramètre on chosit ARIMA(0,1,2)–>modèle moins complexe avec une différence d'AIC négligeable.

```
auto_arima_cnt<-auto.arima(training_set,d=1)
auto_arima_cnt #le meilleure modèle
```

```
## Series: training_set
## ARIMA(0,1,3)
##
## Coefficients:
##           ma1      ma2      ma3
##       -0.6675  -0.1625  -0.0666
## s.e.   0.0376   0.0484   0.0441
##
## sigma^2 = 789128:  log likelihood = -5728.6
## AIC=11465.2   AICc=11465.26   BIC=11483.39
```

```
plot(forecast(training_set,h=25))
lines(test_set,col="red")
```



On observe que les données réelles (test set) sont en majorité dans l'intervalle de confiance du modèle ce qui est bien. Mais le modèle a du mal à cerner l'évolution de la time series (augmentation/diminution), il se contente à prédire une donnée stable.