Projet d'analyse et Prédiction du CAC40 et du Bitcoin en R par Mohamed Falilou Fall

Mohamed Falilou Fall

2024-07-24

Sommaire

1. PRE-TRAITEMENT DE SERIES CHRONOLOGIQUES

1.1 Transformations

1. Importation et stockage des donnees

1.1 CAC40

```
CAC40 <- read.csv("CAC40.csv")
head(CAC40,3)
```

```
##         Date  Indice
## 1 2020-02-10 6015.67
## 2 2020-02-11 6054.76
## 3 2020-02-12 6104.73
```

1.2 CAC40_sans_weekend

```
CAC40_sans_weekend <- ("CAC40_sans_weekend.csv")
head(CAC40_sans_weekend,3)
```

```
## [1] "CAC40_sans_weekend.csv"
```

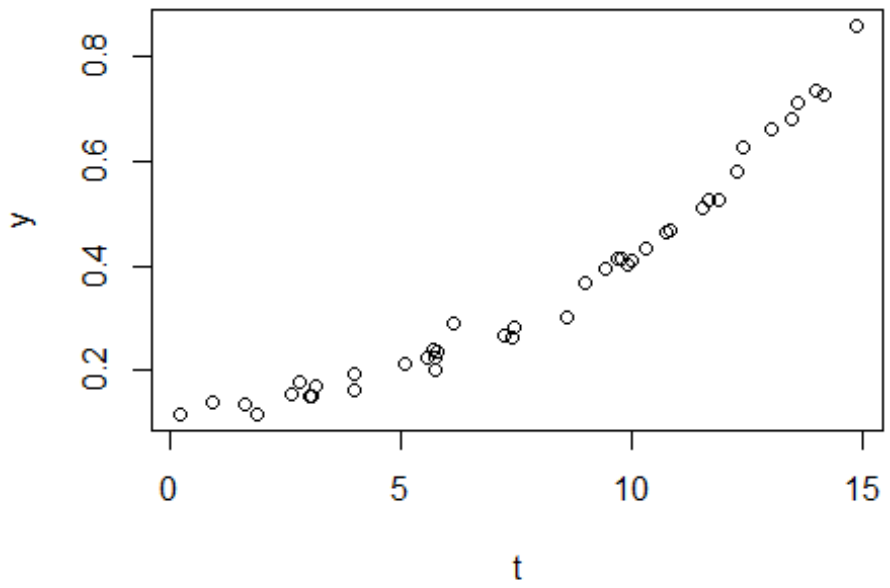
1.3 serie chronologique

```
serie_chronologique <- read.csv("serie_chronologique.csv")
head(serie_chronologique,3)
```

```
##          t         y
## 1 3.982630 0.1922834
## 2 5.581858 0.2234694
## 3 8.592800 0.3014963
```

2. Isolation du temps et des mesures y(i) pour les visualiser

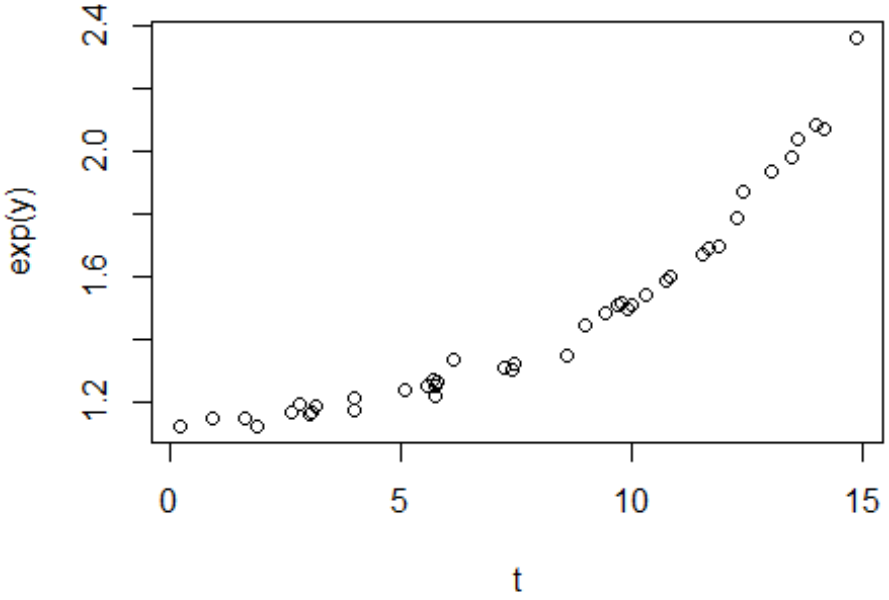
```
t <- serie_chronologique$t
y <- serie_chronologique$y

plot(t,y)
```



3. Visualisation des transformations suivantes: exponentielle, logarithmique ou racine

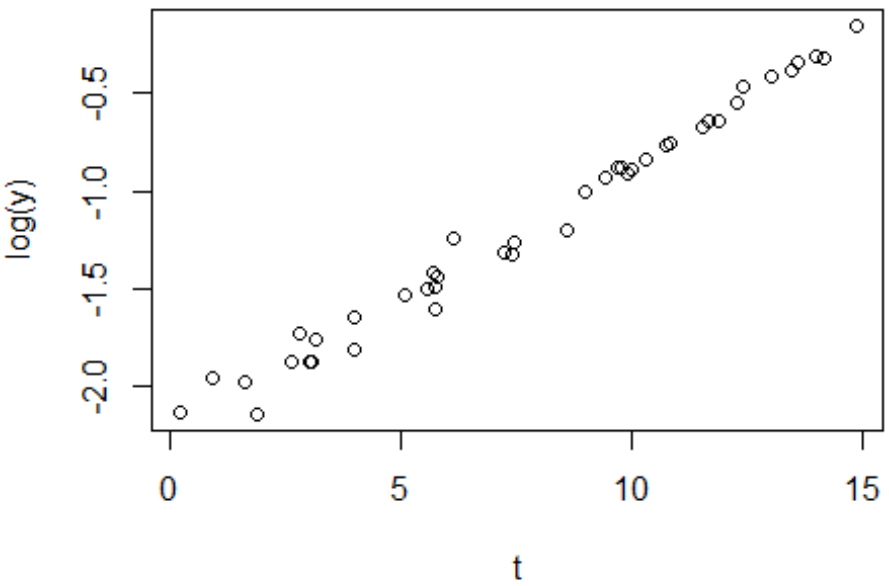
1. Exponentielle

```
plot(t,exp(y))
```



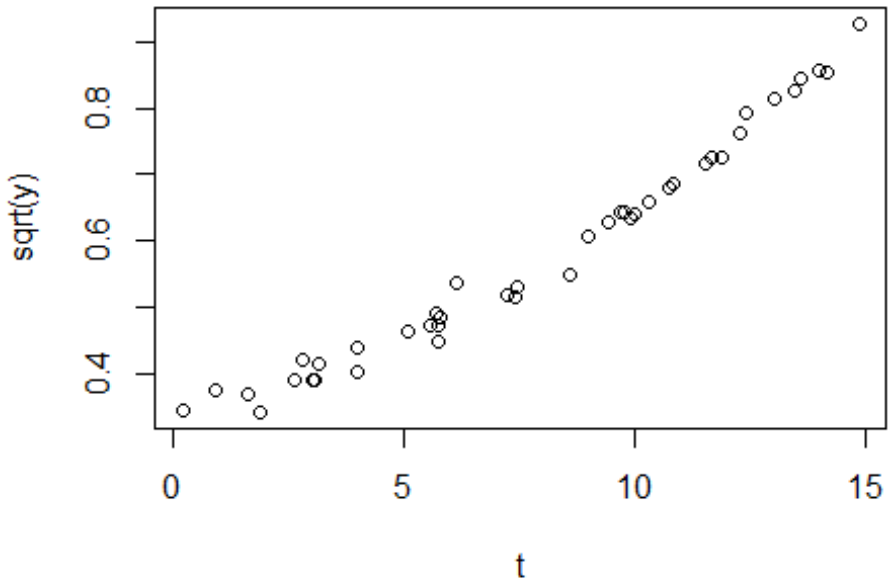
2. logarithmique

```
plot(t,log(y))
```



3. Racine

```
plot(t,sqrt(y))
```



4. Calcul du MSE d'un modele de regression lineaire pour evaluer si la serie brute a une forme lineaire.

```
res_lm  <- lm(y~t)
residus <- y - res_lm[["fitted.values"]]
MSE <- sum(residus^2)
print(MSE)

## [1] 0.1403793
```

5.Transformation logarithmique

```
res_lm_log <- lm(log(y)~t)
residus_log <- y - exp(res_lm_log[["fitted.values"]])
MSE_log <- sum(residus_log^2)
print(MSE_log)

## [1] 0.01485173
```

6.Le MSE relatif a la transformaion racine carre

```
y_sqrt <- sqrt(y)
res_lm_sqrt <- lm(y_sqrt ~ t)
residus_sqrt <- y - (res_lm_sqrt[["fitted.values"]])^2
MSE_sqrt <- sum(residus_sqrt^2)
print(MSE_sqrt)
```

```
## [1] 0.05834192
```

7. Le MSE relatif a la transformation exponentielle

```
res_lm_log <- lm(log(y) ~ t)
fitted_exp <- exp(res_lm_log[["fitted.values"]])
residus_exp <- y - fitted_exp
MSE_exp <- sum(residus_exp^2)
print(MSE_exp)
```

```
## [1] 0.01485173
```

8. Determination de la transformation qui permet le plus de se rapprocher d'une serie chronologique avec une forme lineaire au regard des differentes valeures de MSE

8.1 Comparaison des MSE

```
# Comparaison des MSE
MSE_values <- c(MSE, MSE_log, MSE_sqrt, MSE_exp)
names(MSE_values) <- c("Linéaire", "Logarithmique", "Racine carrée",
"Exponentielle")

# Affichage des MSE
print("Comparaison des MSE:")
```

```
## [1] "Comparaison des MSE:"
```

```
print(MSE_values)
```

```
##      Linéaire Logarithmique Racine carrée Exponentielle
##    0.14037933    0.01485173    0.05834192    0.01485173
```

```
# Conclusion
best_transformation <- names(MSE_values)[which.min(MSE_values)]
print(paste("La transformation qui permet le plus de se rapprocher d'une
serie chronologique avec une forme lineaire est, la transformation:",
best_transformation))
```

```
## [1] "La transformation qui permet le plus de se rapprocher d'une serie
chronologique avec une forme lineaire est, la transformation: Logarithmique"
```

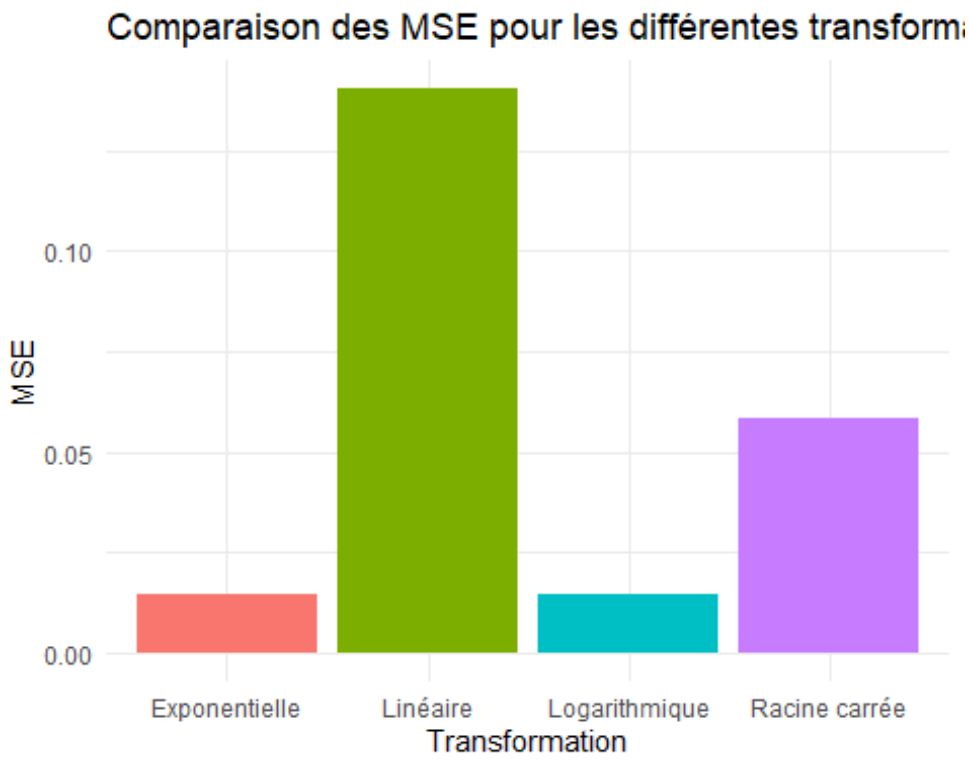
8.2 A l'aide des traces des MSE

```
library(ggplot2)
```

```
## Warning: le package 'ggplot2' a été compilé avec la version R 4.3.3
```

```
mse_data <- data.frame(
  Transformation = c("Linéaire", "Logarithmique", "Racine carrée",
"Exponentielle"),
  MSE = c(MSE, MSE_log, MSE_sqrt, MSE_exp)
)

ggplot(mse_data, aes(x = Transformation, y = MSE, fill = Transformation)) +
  geom_bar(stat = "identity") +
```

```
  theme_minimal() +
  labs(title = "Comparaison des MSE pour les différentes transformations",
       x = "Transformation",
       y = "MSE") +
  theme(legend.position = "none")
```



1.2 Series des differences

1. Import et stockage du fichier CAC40_sans_weekend.csv

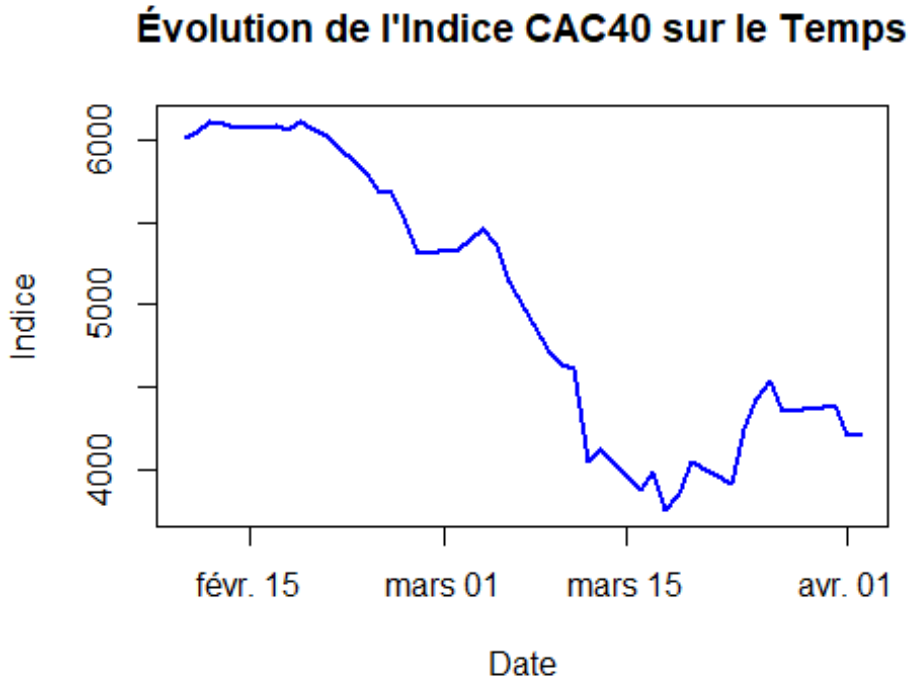
```
serie_brute <- read.csv("CAC40_sans_weekend.csv")
summary(serie_brute)
```

```
##      Date               Indice
##  Length:39          Min.   :3755
##  Class :character   1st Qu.:4232
##  Mode  :character   Median :5139
##                     Mean   :5017
##                     3rd Qu.:5904
##                     Max.   :6111
```

2. Tracage de la serie_brute

```
Date <- serie_brute$Date
Indice <- serie_brute$Indice

serie_brute$Date <- as.Date(serie_brute$Date, format="%Y-%m-%d")
serie_brute <- na.omit(serie_brute)
```

```r
plot(serie_brute$Date, serie_brute$Indice, type="l", col="blue", lwd=2,
     xlab="Date", ylab="Indice",
     main="Évolution de l'Indice CAC40 sur le Temps")
```




```r
print(serie_brute)
```

```
##          Date  Indice
## 1  2020-02-10 6015.67
## 2  2020-02-11 6054.76
## 3  2020-02-12 6104.73
## 4  2020-02-13 6093.14
## 5  2020-02-14 6069.35
## 6  2020-02-17 6085.95
## 7  2020-02-18 6056.82
## 8  2020-02-19 6111.24
## 9  2020-02-20 6062.30
## 10 2020-02-21 6029.72
## 11 2020-02-24 5791.87
## 12 2020-02-25 5679.68
## 13 2020-02-26 5684.55
## 14 2020-02-27 5495.60
## 15 2020-02-28 5309.90
## 16 2020-03-02 5333.52
## 17 2020-03-03 5393.17
## 18 2020-03-04 5464.89
## 19 2020-03-05 5361.10
## 20 2020-03-06 5139.11
```

```
## 21 2020-03-09 4707.91
## 22 2020-03-10 4636.61
## 23 2020-03-11 4610.25
## 24 2020-03-12 4044.26
## 25 2020-03-13 4118.36
## 26 2020-03-16 3881.46
## 27 2020-03-17 3991.78
## 28 2020-03-18 3754.84
## 29 2020-03-19 3855.50
## 30 2020-03-20 4048.80
## 31 2020-03-23 3914.31
## 32 2020-03-24 4242.70
## 33 2020-03-25 4432.30
## 34 2020-03-26 4543.58
## 35 2020-03-27 4351.49
## 36 2020-03-30 4378.51
## 37 2020-03-31 4396.12
## 38 2020-04-01 4207.24
## 39 2020-04-02 4220.96
```

3. Calcul de la serie des differences

```
diff(serie_brute$Indice)
```

```
##  [1]   39.09   49.97  -11.59  -23.79   16.60  -29.13   54.42  -48.94  -
32.58
## [10] -237.85 -112.19    4.87 -188.95 -185.70   23.62   59.65   71.72 -
103.79
## [19] -221.99 -431.20  -71.30  -26.36 -565.99   74.10 -236.90  110.32 -
236.94
## [28]  100.66  193.30 -134.49  328.39  189.60  111.28 -192.09   27.02
17.61
## [37] -188.88   13.72
```

```
serie_diff <- data.frame(
Date = serie_brute$Date[-1],  Indice_diff = diff(serie_brute$Indice))
serie_diff
```

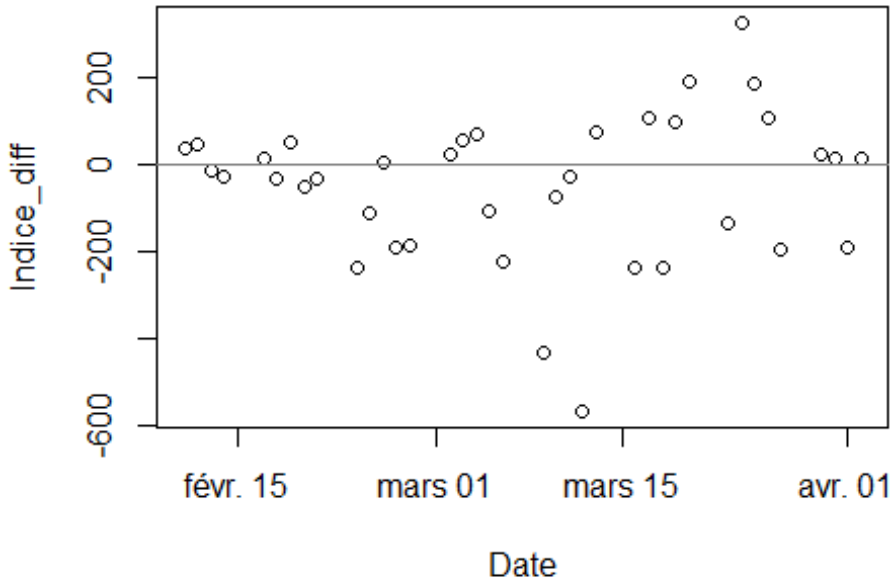
```
##          Date Indice_diff
## 1  2020-02-11       39.09
## 2  2020-02-12       49.97
## 3  2020-02-13      -11.59
## 4  2020-02-14      -23.79
## 5  2020-02-17       16.60
## 6  2020-02-18      -29.13
## 7  2020-02-19       54.42
## 8  2020-02-20      -48.94
## 9  2020-02-21      -32.58
## 10 2020-02-24     -237.85
## 11 2020-02-25     -112.19
## 12 2020-02-26        4.87
## 13 2020-02-27     -188.95
```

```
## 14 2020-02-28     -185.70
## 15 2020-03-02       23.62
## 16 2020-03-03       59.65
## 17 2020-03-04       71.72
## 18 2020-03-05     -103.79
## 19 2020-03-06     -221.99
## 20 2020-03-09     -431.20
## 21 2020-03-10      -71.30
## 22 2020-03-11      -26.36
## 23 2020-03-12     -565.99
## 24 2020-03-13       74.10
## 25 2020-03-16     -236.90
## 26 2020-03-17      110.32
## 27 2020-03-18     -236.94
## 28 2020-03-19      100.66
## 29 2020-03-20      193.30
## 30 2020-03-23     -134.49
## 31 2020-03-24      328.39
## 32 2020-03-25      189.60
## 33 2020-03-26      111.28
## 34 2020-03-27     -192.09
## 35 2020-03-30       27.02
## 36 2020-03-31       17.61
## 37 2020-04-01     -188.88
## 38 2020-04-02       13.72
```

Explications: Le champ date recoit les dates de la serie brute privee du premier element parceque la serie des differences est la serie des mesures `zi = yi - yi-1` pour `i>1` donc la `Date` est egale a `Date[-1]` c'est a dire la date du jour precedent. La serie de date de la `serie_diff`commence a `Date[+1]` de la serie brute.

4. Visualisation de la serie des differences

```
plot(serie_diff)
abline(h=0, col='gray50')
```



5. Calcul de la serie des differences relatives denomme `serie_diff_relative`

```
# Calcul des différences relatives
# Créer un vecteur pour les différences relatives

serie_diff_relative <- numeric(length(serie_diff$Indice_diff) - 1)

# Calculer les différences relatives

for (i in 2:length(serie_diff$Indice_diff)) {
  serie_diff_relative[i - 1] <- (serie_diff$Indice_diff[i] -
serie_diff$Indice_diff[i - 1]) / serie_diff$Indice_diff[i - 1]
}

# Ajouter des NA pour le premier élément

serie_diff_relative <- c(NA, serie_diff_relative)

# Afficher les premières valeurs de la série des différences relatives

print(serie_diff_relative)
```

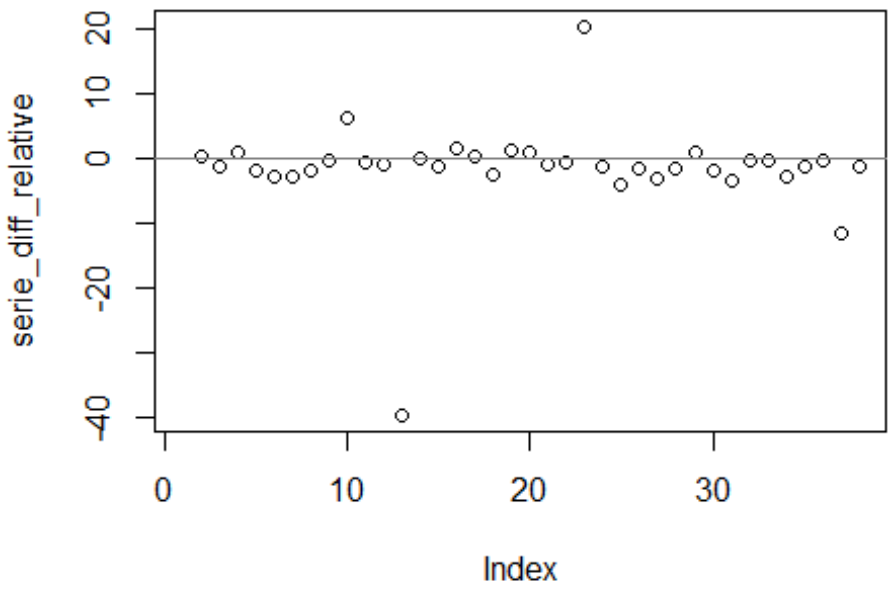
```
##  [1]           NA   0.27833205  -1.23193916   1.05263158  -1.69777217
##  [6]  -2.75481928  -2.86817714  -1.89930173  -0.33428688   6.30049110
## [11]  -0.52831617  -1.04340850 -39.79876797  -0.01720032  -1.12719440
## [16]   1.52540220   0.20234702  -2.44715561   1.13883804   0.94242984
## [21]  -0.83464750  -0.63029453  20.47154780  -1.13092104  -4.19703104
```

```
## [26]  -1.46568172  -3.14775199  -1.42483329   0.92032585  -1.69575789
## [31]  -3.44174288  -0.42263772  -0.41308017  -2.72618620  -1.14066323
## [36]  -0.34826055 -11.72572402  -1.07263871
```

6. Visualisation de la serie_diff_relative avec la fonction `plot

`

```
plot(serie_diff_relative)
abline(h=0, col='gray50')
```



1.3 Evolution du Prix du Bitcoin

1. Installation du package coindeskr

```
#install.packages("devtools")
#devtools::install_github("amrrs/coindeskr")
library(coindeskr)
```

2. Telechargement des donnees du Bitcoin depuis internet

```
bitcoin_data <- get_historic_price(start = "2021-01-01", end = "2021-03-31")
head(bitcoin_data, 3)
```

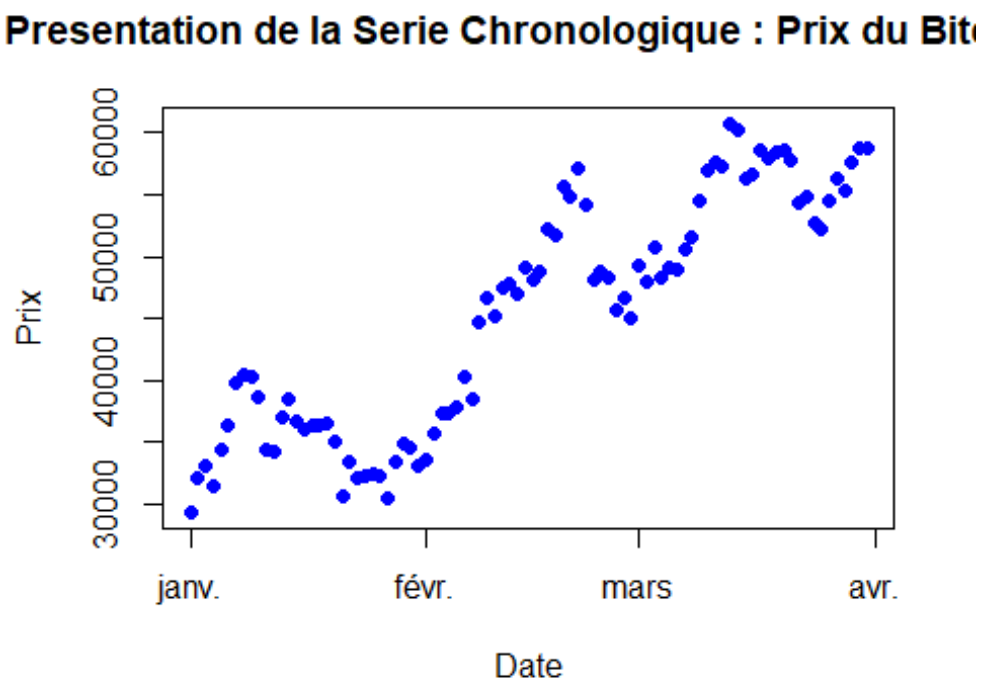
```
##                Price
## 2021-01-01 29333.61
## 2021-01-02 32154.17
## 2021-01-03 33002.54
```

```r
bitcoin_data$Date <- as.Date(rownames(bitcoin_data))
head(bitcoin_data,3)
```

```
##                Price       Date
## 2021-01-01 29333.61 2021-01-01
## 2021-01-02 32154.17 2021-01-02
## 2021-01-03 33002.54 2021-01-03
```

3. Presentation de la serie chronologique

```r
plot(bitcoin_data$Date, bitcoin_data$Price, col = "blue", pch = 19, main =
"Presentation de la Serie Chronologique : Prix du Bitcoin", xlab = "Date",
ylab = "Prix")
```



4. Calcul et Plot de la serie des differences pour le prix du Bitcoin

4.1 Calcul de la serie des differences pour le prix du Bitcoin

```r
diff(bitcoin_data$Price)
```

```
##  [1]  2820.5623   848.3690 -1570.9241  3001.9942  1842.1498  3437.7516
##  [7]   805.9407  -260.5246 -1549.1586 -4300.1230  -195.0321  2802.3972
## [13]  1418.8560 -1684.2785  -734.8054   359.0318   -29.2019   230.9101
## [19] -1572.9870 -4398.3499  2762.1832 -1298.2685   215.6287   214.5299
## [25]  -175.7003 -1789.5563  2873.2189  1434.3391  -220.1842 -1535.0033
## [31]   525.9509  2019.5812  1764.5244  -141.1743   595.3445  2451.2032
## [37] -1841.1184  6255.0041  1958.1662 -1437.3760  2263.4218   383.2854
## [43]  -878.9923  2145.9770 -1025.1754   714.4223  3324.8881  -436.7938
## [49]  3990.6956  -917.5558  2326.9940 -2946.7280 -6009.0371   572.5555
```

```
## [55]  -454.0209 -2539.2972   890.4912 -1549.7995  4156.1074 -1348.1371
## [61]  2911.0783 -2552.3681   890.2437  -270.5789  1715.5467   908.5595
## [67]  2954.7797  2457.1361   721.5841  -330.5917  3436.8755  -545.1398
## [73] -3897.5679   339.4498  1927.4999  -584.1891   468.6368   141.8710
## [79]  -797.1351 -3467.1088   464.9391 -2006.5522  -613.8775  2309.1777
## [85]  1751.3104  -890.4303  2283.7534  1106.7962    -9.8109
```

```
serie_diff_bitcoin <- data.frame(
Date = bitcoin_data$Date[-1],  Indice_diff = diff(bitcoin_data$Price))
serie_diff_bitcoin
```

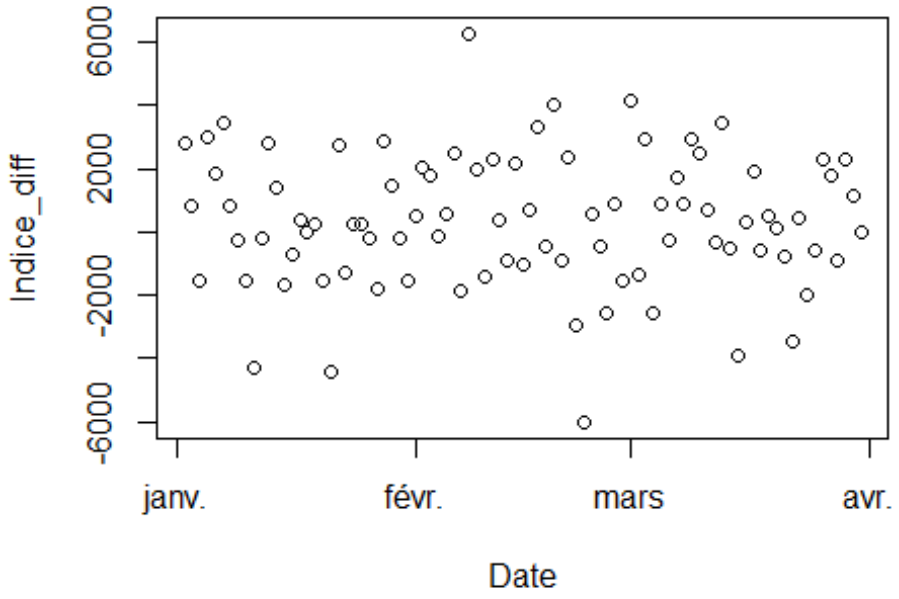
```
##          Date Indice_diff
## 1  2021-01-02   2820.5623
## 2  2021-01-03    848.3690
## 3  2021-01-04  -1570.9241
## 4  2021-01-05   3001.9942
## 5  2021-01-06   1842.1498
## 6  2021-01-07   3437.7516
## 7  2021-01-08    805.9407
## 8  2021-01-09   -260.5246
## 9  2021-01-10  -1549.1586
## 10 2021-01-11  -4300.1230
## 11 2021-01-12   -195.0321
## 12 2021-01-13   2802.3972
## 13 2021-01-14   1418.8560
## 14 2021-01-15  -1684.2785
## 15 2021-01-16   -734.8054
## 16 2021-01-17    359.0318
## 17 2021-01-18    -29.2019
## 18 2021-01-19    230.9101
## 19 2021-01-20  -1572.9870
## 20 2021-01-21  -4398.3499
## 21 2021-01-22   2762.1832
## 22 2021-01-23  -1298.2685
## 23 2021-01-24    215.6287
## 24 2021-01-25    214.5299
## 25 2021-01-26   -175.7003
## 26 2021-01-27  -1789.5563
## 27 2021-01-28   2873.2189
## 28 2021-01-29   1434.3391
## 29 2021-01-30   -220.1842
## 30 2021-01-31  -1535.0033
## 31 2021-02-01    525.9509
## 32 2021-02-02   2019.5812
## 33 2021-02-03   1764.5244
## 34 2021-02-04   -141.1743
## 35 2021-02-05    595.3445
## 36 2021-02-06   2451.2032
## 37 2021-02-07  -1841.1184
## 38 2021-02-08   6255.0041
```

```
## 39 2021-02-09   1958.1662
## 40 2021-02-10  -1437.3760
## 41 2021-02-11   2263.4218
## 42 2021-02-12    383.2854
## 43 2021-02-13   -878.9923
## 44 2021-02-14   2145.9770
## 45 2021-02-15  -1025.1754
## 46 2021-02-16    714.4223
## 47 2021-02-17   3324.8881
## 48 2021-02-18   -436.7938
## 49 2021-02-19   3990.6956
## 50 2021-02-20   -917.5558
## 51 2021-02-21   2326.9940
## 52 2021-02-22  -2946.7280
## 53 2021-02-23  -6009.0371
## 54 2021-02-24    572.5555
## 55 2021-02-25   -454.0209
## 56 2021-02-26  -2539.2972
## 57 2021-02-27    890.4912
## 58 2021-02-28  -1549.7995
## 59 2021-03-01   4156.1074
## 60 2021-03-02  -1348.1371
## 61 2021-03-03   2911.0783
## 62 2021-03-04  -2552.3681
## 63 2021-03-05    890.2437
## 64 2021-03-06   -270.5789
## 65 2021-03-07   1715.5467
## 66 2021-03-08    908.5595
## 67 2021-03-09   2954.7797
## 68 2021-03-10   2457.1361
## 69 2021-03-11    721.5841
## 70 2021-03-12   -330.5917
## 71 2021-03-13   3436.8755
## 72 2021-03-14   -545.1398
## 73 2021-03-15  -3897.5679
## 74 2021-03-16    339.4498
## 75 2021-03-17   1927.4999
## 76 2021-03-18   -584.1891
## 77 2021-03-19    468.6368
## 78 2021-03-20    141.8710
## 79 2021-03-21   -797.1351
## 80 2021-03-22  -3467.1088
## 81 2021-03-23    464.9391
## 82 2021-03-24  -2006.5522
## 83 2021-03-25   -613.8775
## 84 2021-03-26   2309.1777
## 85 2021-03-27   1751.3104
## 86 2021-03-28   -890.4303
## 87 2021-03-29   2283.7534
```

```
## 88 2021-03-30   1106.7962
## 89 2021-03-31     -9.8109
```

4.2 Plot de la serie des differences pour le prix du Bitcoin

```
plot(serie_diff_bitcoin)
```



2. LISSAGE DE SERIES CHRONOLOGIQUES

2.1 Lissage par la moyenne

1. La fonction

```
moyenne_mobile <- function(serie_k){
# Determination de la taille de la serie
n <- length(serie)
# Definition du vecteur qui va recevoir la moyenne mobile d'ordre k
mm_k <- rep(NA, n)


# Boucle pour chaque indice i pour lequel on peut calculer une moyenne mobile
for(i in (k+1):(n-k)) {
  #La moyenne de la serie pour les 2k indices autour de l'indice i est
stockee
  # dans la ieme place du vecteur mm_k
  mm_k[i] <- mean(serie[(i-k) : (i+k)], na.rm=True)
```

```
  #L'option "na.rm=TRUE" de la fonction mean permet d'ignorer les valeures
manquantes
  # (notees NA sous R) potentiellement presente dans la serie
  }



# Renvoie du resultat

return(mm_k)
}
```

2. Importation du fichier CAC40 et stockage dans un un objet CAC40_NA

```
CAC40_NA <- read.csv("CAC40.csv")
CAC40_NA
```

```
##          Date  Indice
## 1  2020-02-10 6015.67
## 2  2020-02-11 6054.76
## 3  2020-02-12 6104.73
## 4  2020-02-13 6093.14
## 5  2020-02-14 6069.35
## 6  2020-02-15      NA
## 7  2020-02-16      NA
## 8  2020-02-17 6085.95
## 9  2020-02-18 6056.82
## 10 2020-02-19 6111.24
## 11 2020-02-20 6062.30
## 12 2020-02-21 6029.72
## 13 2020-02-22      NA
## 14 2020-02-23      NA
## 15 2020-02-24 5791.87
## 16 2020-02-25 5679.68
## 17 2020-02-26 5684.55
## 18 2020-02-27 5495.60
## 19 2020-02-28 5309.90
## 20 2020-02-29      NA
## 21 2020-03-01      NA
## 22 2020-03-02 5333.52
## 23 2020-03-03 5393.17
## 24 2020-03-04 5464.89
## 25 2020-03-05 5361.10
## 26 2020-03-06 5139.11
## 27 2020-03-07      NA
## 28 2020-03-08      NA
## 29 2020-03-09 4707.91
## 30 2020-03-10 4636.61
## 31 2020-03-11 4610.25
## 32 2020-03-12 4044.26
## 33 2020-03-13 4118.36
```

```
## 34 2020-03-14      NA
## 35 2020-03-15      NA
## 36 2020-03-16 3881.46
## 37 2020-03-17 3991.78
## 38 2020-03-18 3754.84
## 39 2020-03-19 3855.50
## 40 2020-03-20 4048.80
## 41 2020-03-21      NA
## 42 2020-03-22      NA
## 43 2020-03-23 3914.31
## 44 2020-03-24 4242.70
## 45 2020-03-25 4432.30
## 46 2020-03-26 4543.58
## 47 2020-03-27 4351.49
## 48 2020-03-28      NA
## 49 2020-03-29      NA
## 50 2020-03-30 4378.51
## 51 2020-03-31 4396.12
## 52 2020-04-01 4207.24
## 53 2020-04-02 4220.96
```

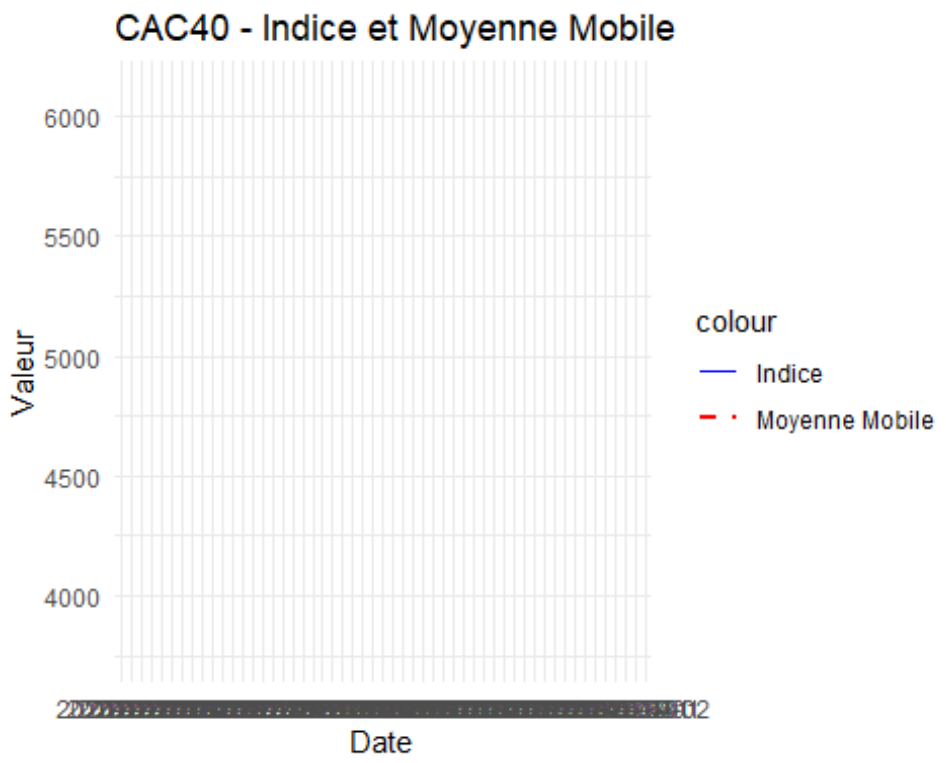
3. Calcul de la moyenne mobile hebdomadaire (d'ordre 3)

```
# Installation du Package `zoo`

#Pour calculer la moyenne mobile hebdomadaire d'ordre 3 en R, il est
nécessaire de s'assurer que la fonction moyenne_mobile est définie ou
d'utiliser une fonction intégrée telle que `rollmean` du package zoo.

#install.packages("zoo")
library(zoo)

## Warning: le package 'zoo' a été compilé avec la version R 4.3.3

##
## Attachement du package : 'zoo'

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

# Le message d'erreur
#Erreur dans `$<-.data.frame`(`*tmp*`, Indice_mm, value = c(6058.38666666667,
:
  #le tableau de remplacement a 51 lignes, le tableau remplacé en a 53

CAC40_NA$Indice_mm <- rollmean(CAC40_NA$Indice, 3, fill = NA, align =
"right")
```

4. Graphe de la serie chronologique de l'indice du CAC40 et la serie de la moyenne mobile(en rouge)

```
ggplot(data = CAC40_NA, aes(x = Date)) +
  geom_line(aes(y = Indice, color = "Indice")) +
  geom_line(aes(y = Indice_mm, color = "Moyenne Mobile"), linetype =
"dashed", size = 1) +
  labs(title = "CAC40 - Indice et Moyenne Mobile",
       x = "Date",
       y = "Valeur") +
  scale_color_manual(values = c("Indice" = "blue", "Moyenne Mobile" = "red"))
+
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Removed 14 rows containing missing values or values outside the
scale range
## (`geom_line()`).

## `geom_line()`: Each group consists of only one observation.
## ℹ Do you need to adjust the group aesthetic?

## Warning: Removed 30 rows containing missing values or values outside the
scale range
## (`geom_line()`).

## `geom_line()`: Each group consists of only one observation.
## ℹ Do you need to adjust the group aesthetic?
```



5. Calcul et graphe des residus de la moyenne mobile

5.1 Calcul des residus de la moyenne mobile

```
CAC40_NA$Residus <- CAC40_NA$Indice - CAC40_NA$Indice_mm
```

5.2 Graphe des residus de la moyenne mobile

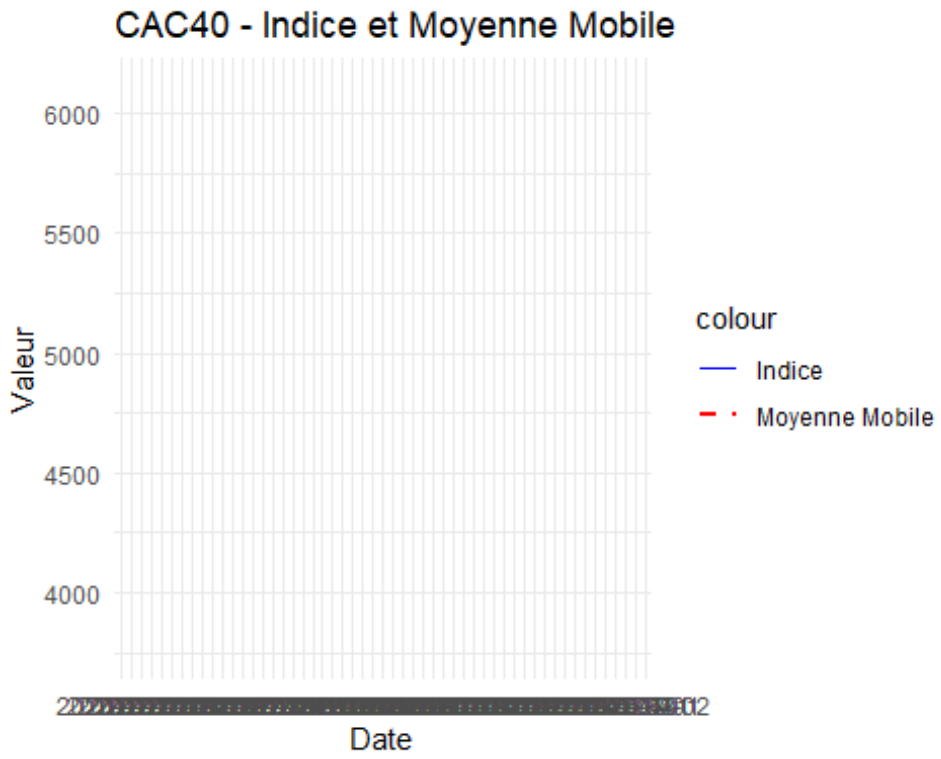
```
ggplot(data = CAC40_NA, aes(x = Date)) +
  geom_line(aes(y = Indice, color = "Indice")) +
  geom_line(aes(y = Indice_mm, color = "Moyenne Mobile"), linetype =
"dashed", size = 1) +
  labs(title = "CAC40 - Indice et Moyenne Mobile",
       x = "Date",
       y = "Valeur") +
  scale_color_manual(values = c("Indice" = "blue", "Moyenne Mobile" = "red"))
+
  theme_minimal()
```

```
## Warning: Removed 14 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```

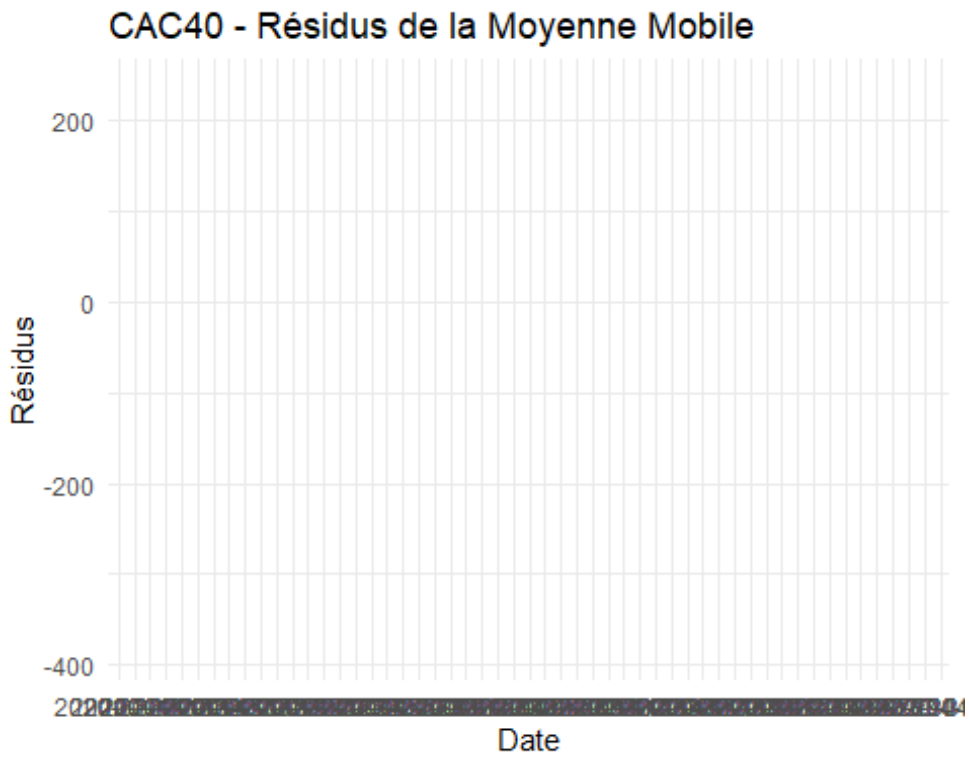
```
## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

```
## Warning: Removed 30 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```

```
## `geom_line()`: Each group consists of only one observation.
## ℹ Do you need to adjust the group aesthetic?
```




```
# Graphe des résidus
ggplot(data = CAC40_NA, aes(x = Date, y = Residus)) +
  geom_line(color = "purple") +
  labs(title = "CAC40 - Résidus de la Moyenne Mobile",
       x = "Date",
       y = "Résidus") +
  theme_minimal()

## Warning: Removed 30 rows containing missing values or values outside the
scale range
## (`geom_line()`).

## `geom_line()`: Each group consists of only one observation.
## ℹ Do you need to adjust the group aesthetic?
```



2.2 Lissage exponentielle

1. Fonction

```
#lissage_exponentielle <- function(serie, gamma) {
# Determination de la taille de la serie
  ##n <- length(serie)
# Definition du vecteur qui va recevoir le lissage exponentielle
  #le_gamma <- rep(NA, n)
# (Ce vecteur est plus grand que la serie brute afin de pouvoir y stocker
# une valeure initiale (necessaire pour la formule de recursivite du lissage
# exponentielle ).
  #

# Cette partie du code permet de prendre en charge les donnees manquantes
# dans la serie brute.
# Il n'est pas necessaire de comprendre cette partie.
#-------------------------------------------------------------------------
  #index <- NULL
  #if sum(is.na(serie))>0){
    #n_tmp <- n
    #n <- sum(!is.na(serie))
    #le_gamma <- rep(NA, n)


    #index <- which(!is.na(serie))
    #serie <- serie[index]
```

```
  #}

#-----------------------------------------------------------------------

# La valeur initiale de la serie lissee est la moyenne des trois premieres
valeurs
# de la serie brute.
#le_gamma[1] <- mean(serie[1:3])


# Faire un boucle pour appliquer succesivement la formule de recursivite
# du lissage exponentielle
#for(k in 1:(n-1)) {
  #le_gamma[k+1] <- gamma * serie[k+1] + (1-gamma)*le_gamma[k]}


# Cette partie du code permet de remettre des valeurs manquantes aux bons
indices
# s'il y en avait dans la serie brute.
# Il n'est pas necessaire de comprendre cette partie.

#-----------------------------------------------------------------------

#if(!is.null(index)){
  #le_gamma_tmp <- le_gamma
  #le_gamma <- rep(NA, n_tmp)
  #le_gamma[index] <- le_gamma_tmp
#}

#-----------------------------------------------------------------------

# Renvoi du resultat
#return(le_gamma)

#}
```

2. Calcul de la serie lissee par lissage exponentielle simple les

2.1 Les packages requis

```
#install.packages("forecast")
#install.packages("ggplot2")
#install.packages("stats")
library(forecast)
```

```
## Warning: le package 'forecast' a été compilé avec la version R 4.3.3
```

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

```r
library(ggplot2)
library(stats)
```

2.2 Traitement des valeures manquantes

```r
# Verification des valeurs NA, NaN
sum(is.na(CAC40_NA$Indice))  # Nombre de NA
```

```
## [1] 14
```

```r
sum(is.nan(CAC40_NA$Indice))  # Nombre de NaN
```

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

```r
# Nettoyage des données en supprimant les lignes contenant NA, NaN
CAC40_NA <- CAC40_NA[!is.na(CAC40_NA$Indice) & !is.nan(CAC40_NA$Indice), ]
```

2.3 Calcul de la serie lissee par lissage exponentielle simple Les

```r
# Application du lissage exponentiel simple avec HoltWinters

ts_data <- ts(CAC40_NA$Indice, frequency = 30)  # Ajustez la fréquence si nécessaire
lissage_exponentiel <- HoltWinters(ts_data, beta = FALSE, gamma = FALSE)

# Extraction des valeurs lissées
Indice_les <- fitted(lissage_exponentiel)[, "xhat"]

# Ajustement des longueurs
min_length <- min(length(CAC40_NA$Indice), length(Indice_les))
CAC40_NA$Indice_les <- NA  # Créer la colonne avec des NA
CAC40_NA$Indice_les[1:min_length] <- Indice_les[1:min_length]  # Remplacer les valeurs lissées

# la longueur
length(CAC40_NA$Indice)
```

```
## [1] 39
```

```r
length(CAC40_NA$Indice_les)
```

```
## [1] 39
```

3. Calcul de la serie lissee par lissage exponentielle double (LED)

```r
length(CAC40_NA$Indice)  # Longueur des données originales
```

```
## [1] 39
```

```r
lissage_exponentiel_double <- HoltWinters(CAC40_NA$Indice, beta = TRUE, gamma = FALSE)
length(fitted(lissage_exponentiel_double))  # Longueur des valeurs lissées
```

```
## [1] 111
```

```r
# Appliquer le lissage exponentiel double
library(forecast)
ts_data <- ts(CAC40_NA$Indice, frequency = 30)
lissage_exponentiel_double <- ets(ts_data, model = "AAN")

# Extraire les valeurs lissées
valeurs_led <- fitted(lissage_exponentiel_double)

# Vérifiez les valeurs lissées
#head(valeurs_lisse)

# Ajuster la longueur
min_length <- min(length(CAC40_NA$Indice), length(valeurs_led))
CAC40_NA$Indice_led <- NA  # Créer une colonne avec des NA
CAC40_NA$Indice_led[1:min_length] <- valeurs_led[1:min_length]

# Vérifiez les dimensions
length(CAC40_NA$Indice)
```

```
## [1] 39
```

```r
length(CAC40_NA$Indice_led)
```

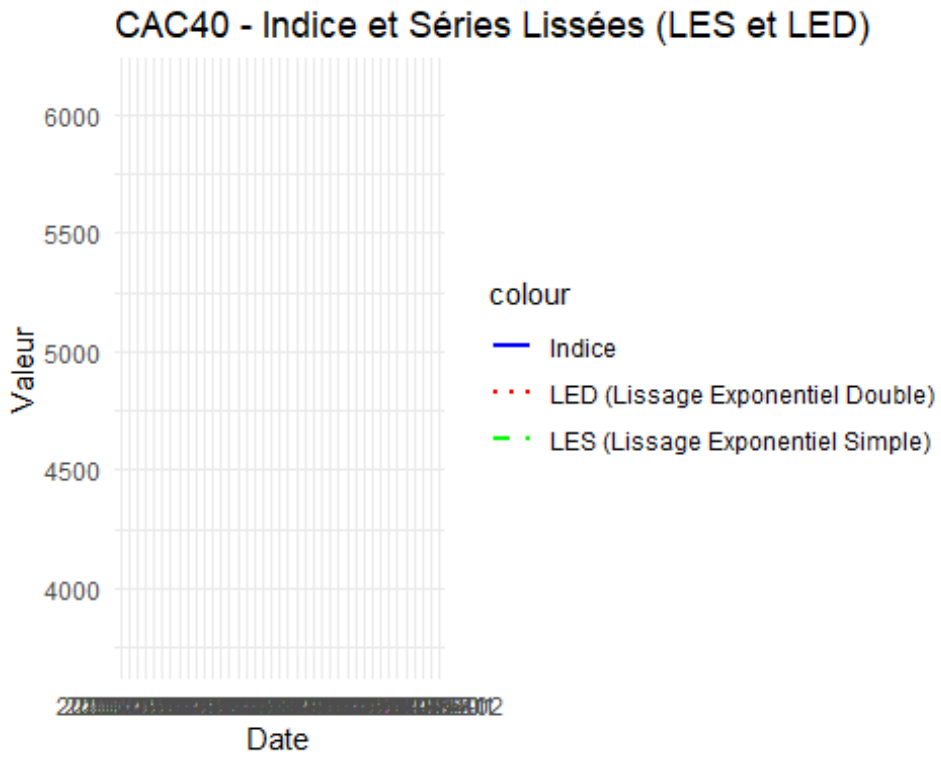
```
## [1] 39
```

4. Tracage de la serie chronologique et les series lissees par LES et LED.

```r
# Tracer le graphique
ggplot(data = CAC40_NA, aes(x = Date)) +
  geom_line(aes(y = Indice, color = "Indice"), size = 1) +
  geom_line(aes(y = Indice_les, color = "LES (Lissage Exponentiel Simple)"),
linetype = "dashed", size = 1) +
  geom_line(aes(y = Indice_led, color = "LED (Lissage Exponentiel Double)"),
linetype = "dotted", size = 1) +
  labs(title = "CAC40 - Indice et Séries Lissées (LES et LED)",
       x = "Date",
       y = "Valeur") +
  scale_color_manual(values = c("Indice" = "blue",
                                "LES (Lissage Exponentiel Simple)" = "green",
                                "LED (Lissage Exponentiel Double)" = "red"))
+
  theme_minimal()
```

```
## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?

## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_line()`).

## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

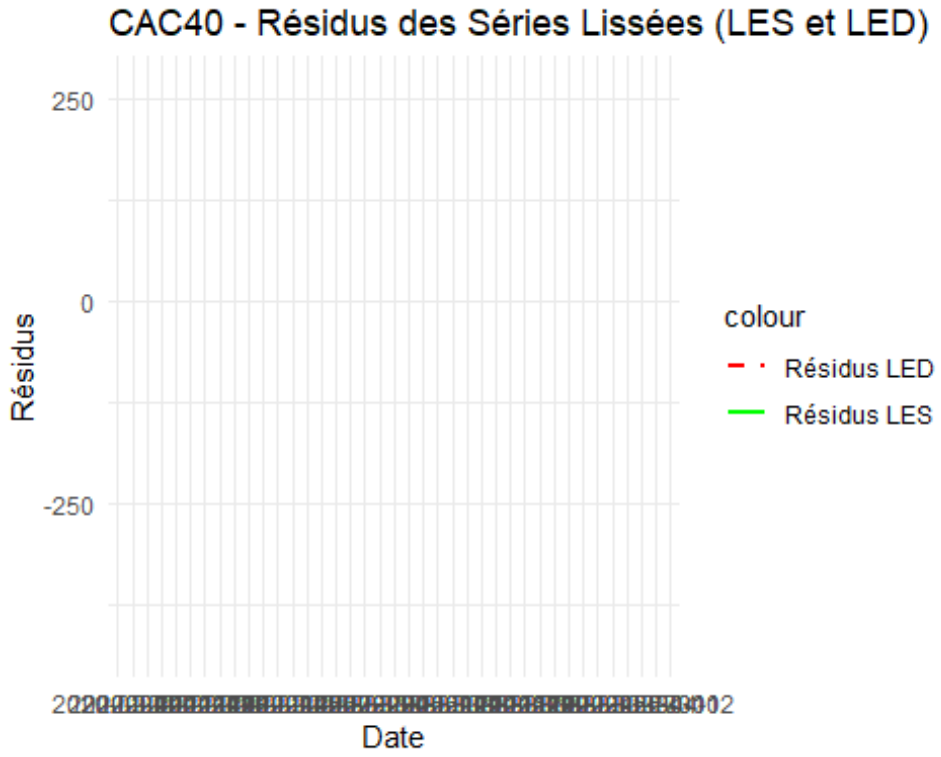
```
## `geom_line()`: Each group consists of only one observation.
## ℹ Do you need to adjust the group aesthetic?
```



5. Graphe des residus de chacune des series lissees

```
# Calcul des résidus
CAC40_NA$Residus_les <- CAC40_NA$Indice - CAC40_NA$Indice_les
CAC40_NA$Residus_led <- CAC40_NA$Indice - CAC40_NA$Indice_led

# Graphe des residus
# Tracer les résidus
ggplot(data = CAC40_NA, aes(x = Date)) +
  geom_line(aes(y = Residus_les, color = "Résidus LES"), size = 1) +
  geom_line(aes(y = Residus_led, color = "Résidus LED"), linetype = "dashed",
size = 1) +
  labs(title = "CAC40 - Résidus des Séries Lissées (LES et LED)",
       x = "Date",
       y = "Résidus") +
  scale_color_manual(values = c("Résidus LES" = "green", "Résidus LED" =
"red")) +
  theme_minimal()

## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_line()`).

## `geom_line()`: Each group consists of only one observation.
## ℹ Do you need to adjust the group aesthetic?
```

```
## `geom_line()`: Each group consists of only one observation.
## ℹ Do you need to adjust the group aesthetic?
```



3. TENDANCE ET PERIODICITE

1. Importation des prix du Bitcoin depuis le 12 Mars 2020 :

```
library(coindeskr)
bitcoin_data <- get_historic_price(start="2021-03-12", end = "2021-04-07")
bitcoin_data$Date <- 1:length(bitcoin_data$Price)
print(bitcoin_data)
```

```
##                Price Date
## 2021-01-01 29333.61    1
## 2021-01-02 32154.17    2
## 2021-01-03 33002.54    3
## 2021-01-04 31431.61    4
## 2021-01-05 34433.61    5
## 2021-01-06 36275.76    6
## 2021-01-07 39713.51    7
## 2021-01-08 40519.45    8
## 2021-01-09 40258.92    9
## 2021-01-10 38709.77   10
## 2021-01-11 34409.64   11
## 2021-01-12 34214.61   12
## 2021-01-13 37017.01   13
## 2021-01-14 38435.86   14
```

```
## 2021-01-15 36751.58   15
## 2021-01-16 36016.78   16
## 2021-01-17 36375.81   17
## 2021-01-18 36346.61   18
## 2021-01-19 36577.52   19
## 2021-01-20 35004.53   20
## 2021-01-21 30606.18   21
## 2021-01-22 33368.37   22
## 2021-01-23 32070.10   23
## 2021-01-24 32285.73   24
## 2021-01-25 32500.26   25
## 2021-01-26 32324.56   26
## 2021-01-27 30535.00   27
## 2021-01-28 33408.22   28
## 2021-01-29 34842.56   29
## 2021-01-30 34622.37   30
## 2021-01-31 33087.37   31
## 2021-02-01 33613.32   32
## 2021-02-02 35632.90   33
## 2021-02-03 37397.43   34
## 2021-02-04 37256.25   35
## 2021-02-05 37851.60   36
## 2021-02-06 40302.80   37
## 2021-02-07 38461.68   38
## 2021-02-08 44716.69   39
## 2021-02-09 46674.85   40
## 2021-02-10 45237.48   41
## 2021-02-11 47500.90   42
## 2021-02-12 47884.18   43
## 2021-02-13 47005.19   44
## 2021-02-14 49151.17   45
## 2021-02-15 48125.99   46
## 2021-02-16 48840.41   47
## 2021-02-17 52165.30   48
## 2021-02-18 51728.51   49
## 2021-02-19 55719.20   50
## 2021-02-20 54801.65   51
## 2021-02-21 57128.64   52
## 2021-02-22 54181.91   53
## 2021-02-23 48172.88   54
## 2021-02-24 48745.43   55
## 2021-02-25 48291.41   56
## 2021-02-26 45752.11   57
## 2021-02-27 46642.61   58
## 2021-02-28 45092.81   59
## 2021-03-01 49248.91   60
## 2021-03-02 47900.78   61
## 2021-03-03 50811.86   62
## 2021-03-04 48259.49   63
## 2021-03-05 49149.73   64
```

```
## 2021-03-06 48879.15   65
## 2021-03-07 50594.70   66
## 2021-03-08 51503.26   67
## 2021-03-09 54458.04   68
## 2021-03-10 56915.17   69
## 2021-03-11 57636.76   70
## 2021-03-12 57306.17   71
## 2021-03-13 60743.04   72
## 2021-03-14 60197.90   73
## 2021-03-15 56300.33   74
## 2021-03-16 56639.78   75
## 2021-03-17 58567.28   76
## 2021-03-18 57983.09   77
## 2021-03-19 58451.73   78
## 2021-03-20 58593.60   79
## 2021-03-21 57796.47   80
## 2021-03-22 54329.36   81
## 2021-03-23 54794.30   82
## 2021-03-24 52787.75   83
## 2021-03-25 52173.87   84
## 2021-03-26 54483.05   85
## 2021-03-27 56234.36   86
## 2021-03-28 55343.93   87
## 2021-03-29 57627.68   88
## 2021-03-30 58734.48   89
## 2021-03-31 58724.66   90
```

2. La tendance lineaire de cette serie chronologique avec la fonction lm et stockage des resultats dans un objet tendance

```
# Conversion en série temporelle
ts_data_bitcoin <- ts(bitcoin_data$Price, frequency = 30)



# Calcul de la tendance linéaire
# Nous devons créer une variable de temps pour l'ajustement du modèle
linéaire
time_index <- seq_along(ts_data_bitcoin)
tendance_model <- lm(ts_data_bitcoin ~ time_index)

# Stocker les résultats dans un objet `tendance`
bitcoin_data$tendance <- fitted(tendance_model)
```

3. Les residus de la tendance lineaire :

```
bitcoin_data$Price_detrend <- bitcoin_data$Price -
tendance_model$fitted.values
```

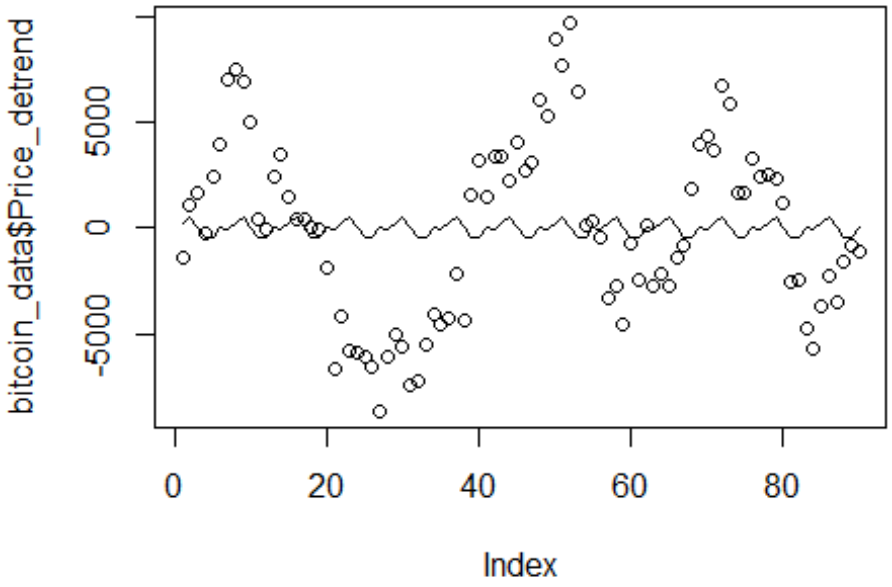
5. Application de la fonction sur la serie tendancee

```
calcul_periodicite <- function(bitcoin_data, p) {
  n <- length(bitcoin_data)
  N <- floor(n / p)
```

```
  periode <- rep(NA, p)

  for (i in 1:p) {
    indices <- i + (0:N) * p
    indices <- indices[indices <= n]  # Pour éviter les indices dépassant la
longueur de bitcoin_data
    periode[i] <- mean(bitcoin_data[indices], na.rm = TRUE)
  }

  if (n == N * p) {
    Date <- rep(periode, N)
  } else {
    Date <- c(rep(periode, N), periode[1:(n - N * p)])
  }

  return(Date)
}

periodicite <- calcul_periodicite(bitcoin_data$Price_detrend, 7)
```

6. Representation de la Periodicite de la serie detendancee

```
p <- 7
N <- floor(nrow(bitcoin_data) / p)


bitcoin_data$Jours <- as.Date(rownames(bitcoin_data))
plot(bitcoin_data$Price_detrend)
lines(periodicite)
for(i in 1:N) {
  rect(bitcoin_data$Jours[3+(i-1)*p], -6000, bitcoin_data$Jours[4+(i-1)*p],
6000,
       col = "gray", density = 20, border = NA)
  text(bitcoin_data$Jours[3.5+(i-1)*p], 500, labels = "Week-end", col =
"gray30")

}
```



4. MODELE ADDITIF POUR PREDIRE L'EVOLUTION DU PRIX DU BITCOIN

4.1 Rappel du model additif

```
# Le modèle additif avec une tendance linéaire et une périodicité de 7 jours
s'écrit comme suit :

# Yt = (β0 + β1t) + St + Et
# Ou St représente la composante saisonnière avec une période de 7 jours. Le
nombre total de paramètres à estimer est de 9 :
# * 2 pour la tendance lineaire (β0 et β1)
# * 7 pour la composante saisonnière (un pour chaque jour de la période).
```

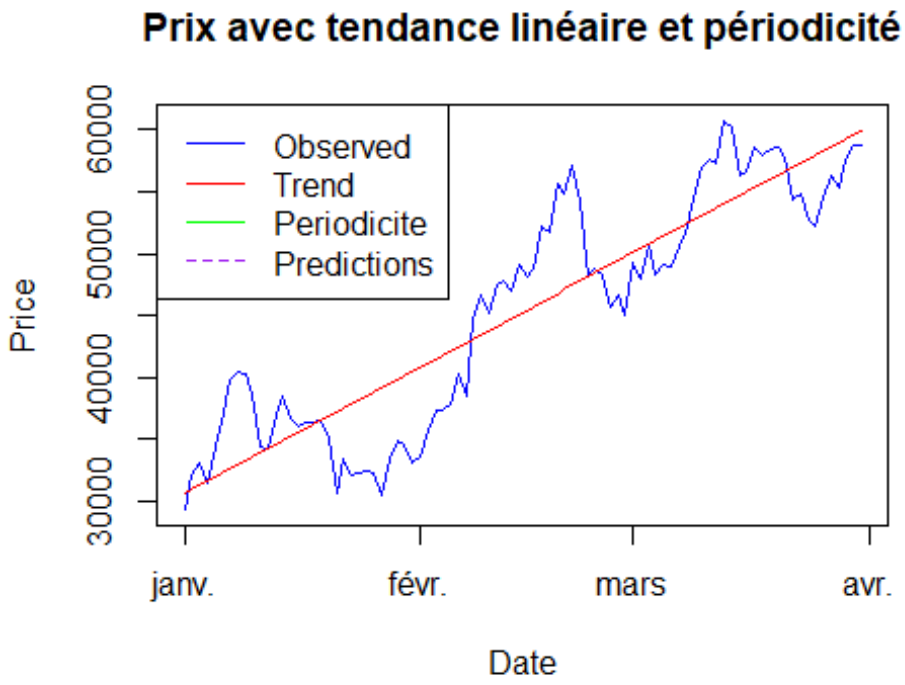
4.2 Calcul des predictions du modele additif et sa courbe

```
calcul_periodicite <- function(bitcoin_data, p) {
  n <- length(bitcoin_data)
  N <- floor(n / p)
  periode <- rep(NA, p)

  for (i in 1:p) {
    indices <- i + (0:N) * p
    indices <- indices[indices <= n]  # Pour éviter les indices dépassant la
longueur de bitcoin_data
    periode[i] <- mean(bitcoin_data[indices], na.rm = TRUE)
  }
```

```r
  if (n == N * p) {
    Date <- rep(periode, N)
  } else {
    Date <- c(rep(periode, N), periode[1:(n - N * p)])
  }

  return(Date)
}

# Calcule de la périodicité
p <- 7
periodicite <- calcul_periodicite(bitcoin_data$Price_detrend, p)
bitcoin_data$periodicite <- periodicite

print(bitcoin_data$periodicite)
```

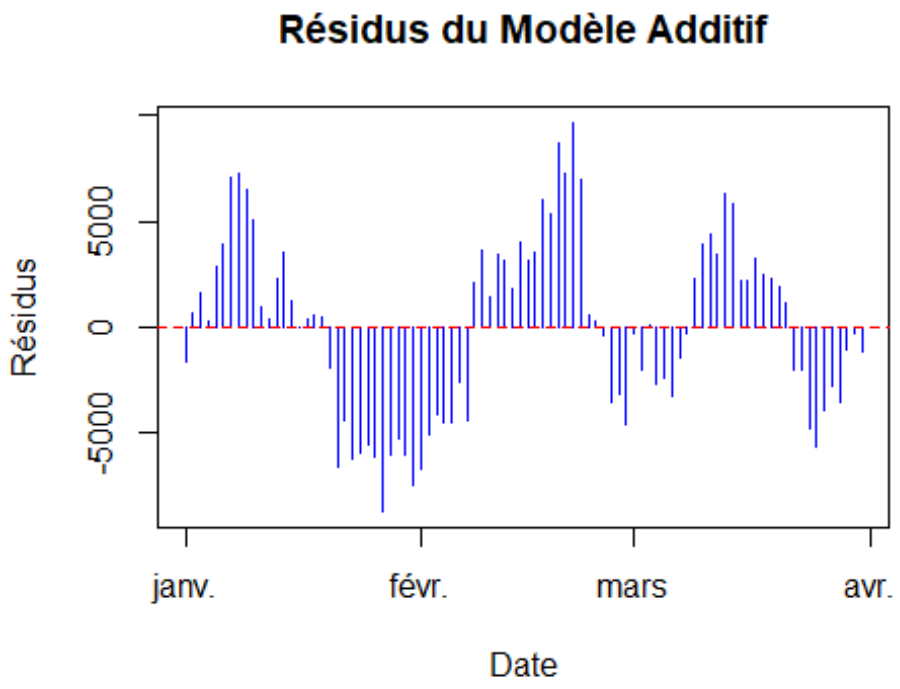
```
##  [1]  277.79741  481.83456   69.89927 -480.24264 -418.38023   83.69869
##  [7]  -15.82432  277.79741  481.83456   69.89927 -480.24264 -418.38023
## [13]   83.69869  -15.82432  277.79741  481.83456   69.89927 -480.24264
## [19] -418.38023   83.69869  -15.82432  277.79741  481.83456   69.89927
## [25] -480.24264 -418.38023   83.69869  -15.82432  277.79741  481.83456
## [31]   69.89927 -480.24264 -418.38023   83.69869  -15.82432  277.79741
## [37]  481.83456   69.89927 -480.24264 -418.38023   83.69869  -15.82432
## [43]  277.79741  481.83456   69.89927 -480.24264 -418.38023   83.69869
## [49]  -15.82432  277.79741  481.83456   69.89927 -480.24264 -418.38023
## [55]   83.69869  -15.82432  277.79741  481.83456   69.89927 -480.24264
## [61] -418.38023   83.69869  -15.82432  277.79741  481.83456   69.89927
## [67] -480.24264 -418.38023   83.69869  -15.82432  277.79741  481.83456
## [73]   69.89927 -480.24264 -418.38023   83.69869  -15.82432  277.79741
## [79]  481.83456   69.89927 -480.24264 -418.38023   83.69869  -15.82432
## [85]  277.79741  481.83456   69.89927 -480.24264 -418.38023   83.69869
```

```r
plot(bitcoin_data$Jours, bitcoin_data$Price, type = "l", col = "blue",
     main = "Prix avec tendance linéaire et périodicité", xlab = "Date", ylab
= "Price")
lines(bitcoin_data$Jours, bitcoin_data$tendance, col = "red")
lines(bitcoin_data$Jours, bitcoin_data$periodicite, col = "green")
lines(bitcoin_data$Jours, bitcoin_data$hat_Price, col = "purple", lty = 2)
legend("topleft", legend = c("Observed", "Trend", "Periodicite",
"Predictions"),
       col = c("blue", "red", "green", "purple"), lty = c(1, 1, 1, 2))
```



4.3. Calcul des residus de ce modele et sa graphe

```
# Calcul des prédictions du modèle additif
bitcoin_data$hat_Price <- bitcoin_data$tendance + bitcoin_data$periodicite

# Calcul des résidus
bitcoin_data$residus <- bitcoin_data$Price - bitcoin_data$hat_Price


# Plot des résidus
plot(bitcoin_data$Jours, bitcoin_data$residus, type = "h", col = "blue",
     main = "Résidus du Modèle Additif", xlab = "Date", ylab = "Résidus")
abline(h = 0, col = "red", lty = 2)
```



4.4 Previsions sur l'evolution des prix du bitcoin pour les 10 prochains jours

```
n <- length(bitcoin_data$Price)
nouveau_jours <- (n+1):(n+10)

# Conversion en série temporelle
ts_data_bitcoin <- ts(bitcoin_data$Price, nouveau_jours)


# Calcul de la tendance linéaire
# Nous devons créer une variable de temps pour l'ajustement du modèle
linéaire
time_index <- seq_along(ts_data_bitcoin)
tendance_model <- lm(ts_data_bitcoin ~ time_index)

# Stocker les résultats dans un objet `tendance`
bitcoin_data$tendance_10 <- fitted(tendance_model)


# Affichage des tendances pour les 10 prochains jours
head(bitcoin_data$tendance, 10)
```

```
##  [1] 30686.71 31014.32 31341.93 31669.54 31997.15 32324.76 32652.37
32979.98
##  [9] 33307.59 33635.20
```

4.5. La valeur de la periodicite pour les jours a venir

```
moment_periode <- nouveau_jours %% 7 + 1
prediction_periode <- periodicite[moment_periode]
print(prediction_periode)
```

```
##  [1]  277.79741  481.83456   69.89927 -480.24264 -418.38023   83.69869
##  [7]  -15.82432  277.79741  481.83456   69.89927
```

L'opération %% en R correspond à l'opérateur de modulo. Elle calcule le reste de la division entière de deux nombres.

L'utilisation de l'opérateur modulo (%%) dans la ligne moment_periode <- nouveau_jours %% 7 + 1 est nécessaire pour gérer la périodicité dans les données. Voici pourquoi :

- Détermination du Cycle Hebdomadaire : En appliquant l'opérateur modulo 7 à chaque valeur de nouveau_jours, vous obtenez un résultat qui indique la position du jour dans une semaine de 7 jours. Par exemple, un résultat de 1 correspond au premier jour du cycle (jour 1), un résultat de 2 correspond au deuxième jour (jour 2), et ainsi de suite jusqu'à 7, après quoi le cycle recommence à 1.

- Ajustement de l'Indexation : Le +1 après le modulo ajuste l'indexation pour qu'elle commence à 1 au lieu de 0 (ce qui serait le cas avec un simple modulo). Cela est utile si les éléments de periodicite sont indexés de 1 à 7 (plutôt que de 0 à 6).

- Utilisation de la Périodicité : moment_periode est ensuite utilisé pour accéder aux valeurs correspondantes dans periodicite, ce qui permet d'extraire la valeur de la périodicité pour chaque jour spécifié dans nouveau_jours.

En résumé, cette ligne de code est utilisée pour s'assurer que chaque jour dans nouveau_jours est correctement mappé à une position dans un cycle hebdomadaire de 7 jours, permettant ainsi d'extraire les valeurs correspondantes dans periodicite.

4.6. Calcul des previsions du modele pour les 10 jours a venir et stockage dans un objet `prevision_modele_add`

```
# Nombre de jours à prévoir
moment_periode <- 10

# Créer une séquence pour les nouveaux jours
nouveau_jours <- (nrow(bitcoin_data) + 1):(nrow(bitcoin_data) +
moment_periode)

# Calculer la tendance projetée
tendance_previsions <- predict(tendance_model, newdata = data.frame(Jours =
bitcoin_data$Jours[nrow(bitcoin_data)] + 1:moment_periode))
```

```
## Warning: 'newdata' avait 10 lignes mais les variables trouvées ont 90
lignes
```

```r
# Calculer le moment de la période pour chaque nouveau jour
moment_periode <- nouveau_jours %% 7 + 1

# Appliquer la périodicité aux prévisions
prediction_periode <- periodicite[moment_periode]

# Calcul des prévisions finales
prevision_modele_add <- tendance_previsions + prediction_periode

# Stocker les prévisions dans un objet
prevision_modele_add <- data.frame(Jours =
bitcoin_data$Jours[nrow(bitcoin_data)] + 1:moment_periode,
                                   Previsions = prevision_modele_add)

## Warning in 1:moment_periode: numerical expression has 10 elements: only
the
## first used

print(prevision_modele_add)

##         Jours Previsions
## 1  2021-04-01   30964.51
## 2  2021-04-01   31496.16
## 3  2021-04-01   31411.83
## 4  2021-04-01   31189.30
## 5  2021-04-01   31578.77
## 6  2021-04-01   32408.46
## 7  2021-04-01   32636.55
## 8  2021-04-01   33257.78
## 9  2021-04-01   33789.42
## 10 2021-04-01   33705.10
## 11 2021-04-01   34240.61
## 12 2021-04-01   34772.25
## 13 2021-04-01   34687.93
## 14 2021-04-01   34465.39
## 15 2021-04-01   34854.86
## 16 2021-04-01   35684.55
## 17 2021-04-01   35912.64
## 18 2021-04-01   36533.87
## 19 2021-04-01   37065.52
## 20 2021-04-01   36981.19
## 21 2021-04-01   37516.70
## 22 2021-04-01   38048.34
## 23 2021-04-01   37964.02
## 24 2021-04-01   37741.49
## 25 2021-04-01   38130.96
## 26 2021-04-01   38960.65
## 27 2021-04-01   39188.73
## 28 2021-04-01   39809.96
## 29 2021-04-01   40341.61
```

```
## 30 2021-04-01   40257.28
## 31 2021-04-01   40792.79
## 32 2021-04-01   41324.44
## 33 2021-04-01   41240.11
## 34 2021-04-01   41017.58
## 35 2021-04-01   41407.05
## 36 2021-04-01   42236.74
## 37 2021-04-01   42464.83
## 38 2021-04-01   43086.06
## 39 2021-04-01   43617.70
## 40 2021-04-01   43533.38
## 41 2021-04-01   44068.88
## 42 2021-04-01   44600.53
## 43 2021-04-01   44516.20
## 44 2021-04-01   44293.67
## 45 2021-04-01   44683.14
## 46 2021-04-01   45512.83
## 47 2021-04-01   45740.92
## 48 2021-04-01   46362.15
## 49 2021-04-01   46893.80
## 50 2021-04-01   46809.47
## 51 2021-04-01   47344.98
## 52 2021-04-01   47876.62
## 53 2021-04-01   47792.30
## 54 2021-04-01   47569.76
## 55 2021-04-01   47959.24
## 56 2021-04-01   48788.92
## 57 2021-04-01   49017.01
## 58 2021-04-01   49638.24
## 59 2021-04-01   50169.89
## 60 2021-04-01   50085.56
## 61 2021-04-01   50621.07
## 62 2021-04-01   51152.72
## 63 2021-04-01   51068.39
## 64 2021-04-01   50845.86
## 65 2021-04-01   51235.33
## 66 2021-04-01   52065.02
## 67 2021-04-01   52293.10
## 68 2021-04-01   52914.34
## 69 2021-04-01   53445.98
## 70 2021-04-01   53361.66
## 71 2021-04-01   53897.16
## 72 2021-04-01   54428.81
## 73 2021-04-01   54344.48
## 74 2021-04-01   54121.95
## 75 2021-04-01   54511.42
## 76 2021-04-01   55341.11
## 77 2021-04-01   55569.20
## 78 2021-04-01   56190.43
## 79 2021-04-01   56722.07
```

```
## 80 2021-04-01   56637.75
## 81 2021-04-01   57173.26
## 82 2021-04-01   57704.90
## 83 2021-04-01   57620.58
## 84 2021-04-01   57398.04
## 85 2021-04-01   57787.52
## 86 2021-04-01   58617.20
## 87 2021-04-01   58845.29
## 88 2021-04-01   59466.52
## 89 2021-04-01   59998.17
## 90 2021-04-01   59913.84
```

4.7. Visualisation de l'evolution du Prix du bitcoin

```
xlim <- c(1, n + 10)
ylim <- range(c(bitcoin_data$Price, prevision_modele_add$Previsions), na.rm =
TRUE)

plot(bitcoin_data$Price, type = 'l', xlim = xlim, ylim = ylim, ylab =
"Price", xlab = "Days")


lines(bitcoin_data$hat_Price, col = 'blue')


lines(prevision_modele_add$Previsions, type = 'p', pch = 17)
```