Comparing the performance of SARIMA and Facebook prophet models for time seriesforecasting

AGJOUD Imad ARAGOU Wassef IMLOUL Douae RACHIDI Widad

Contents

Introduction

Ce projet vise à comparer les performances des modèles SARIMA et Prophet pour la prévision des tendances de recherche du terme "Football" à l'échelle mondiale sur Google. Les modèles seront évalués sur leur capacité à capturer la saisonnalité et les tendances de la série temporelle.

Décrire le problème à analyser

Nous chercherons à déterminer lequel des deux modèles (SARIMA ou Prophet) fournit les prévisions les plus précises et robustes pour cette série temporelle. Le but est de mieux comprendre les cycles de popularité du terme "Football" et de choisir le modèle le mieux adapté.

Analyse

Appeler les packages nécessaires

```
library(prophet)
library(tseries)
library(forecast)
library(TSstudio)
library(dplyr)
library(plotly)
library(lmtest)

library(dygraphs)
library(lubridate)
library(Amelia)
```

Importer la série et la transformer à un objet de type ts()

```
serie2 <- read.csv("C:\\Users\\imadh\\Downloads\\multiTimeline.csv",sep=",")
data_p <- read.csv("C:\\Users\\imadh\\Downloads\\multiTimeline.csv",sep=",")


missmap(data_p)
summary(data_p)
```

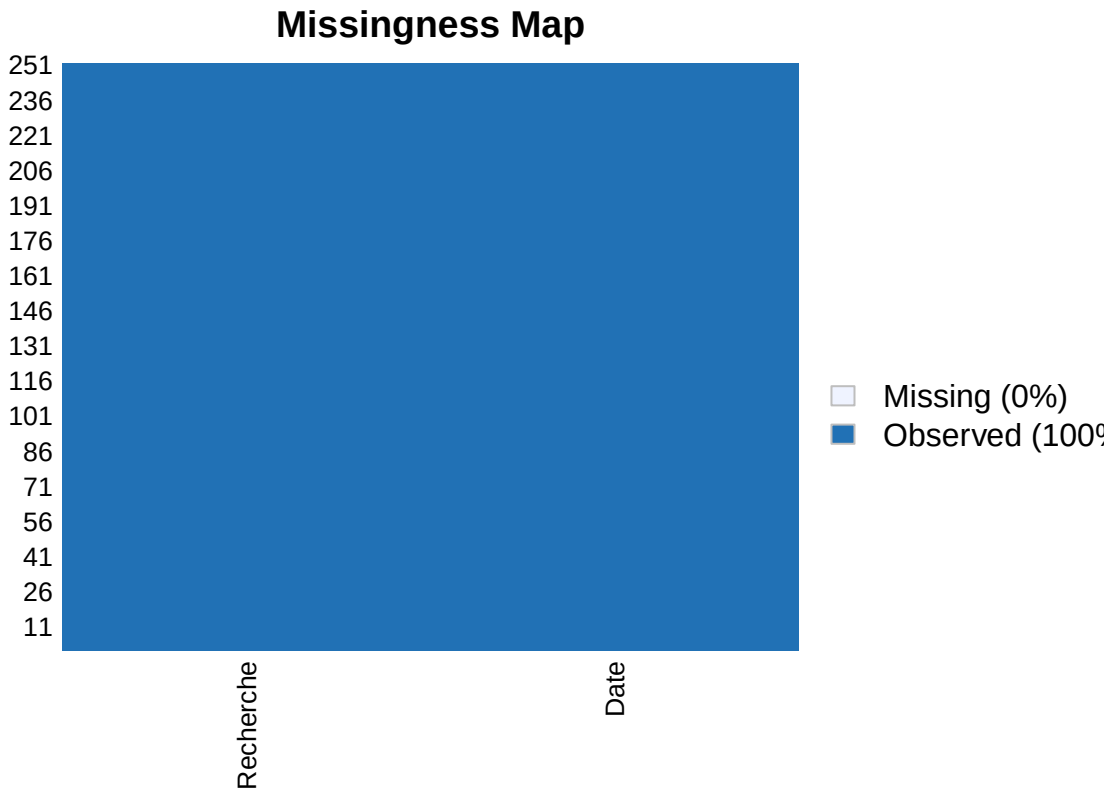
```
##      Date             Recherche
##  Length:251         Min.   : 14.00
##  Class :character   1st Qu.: 29.00
##  Mode  :character   Median : 39.00
##                     Mean   : 45.25
##                     3rd Qu.: 57.00
##                     Max.   :100.00
```

```
serie=serie2
serie=serie[,2]
class(serie)
```

```
## [1] "integer"
```

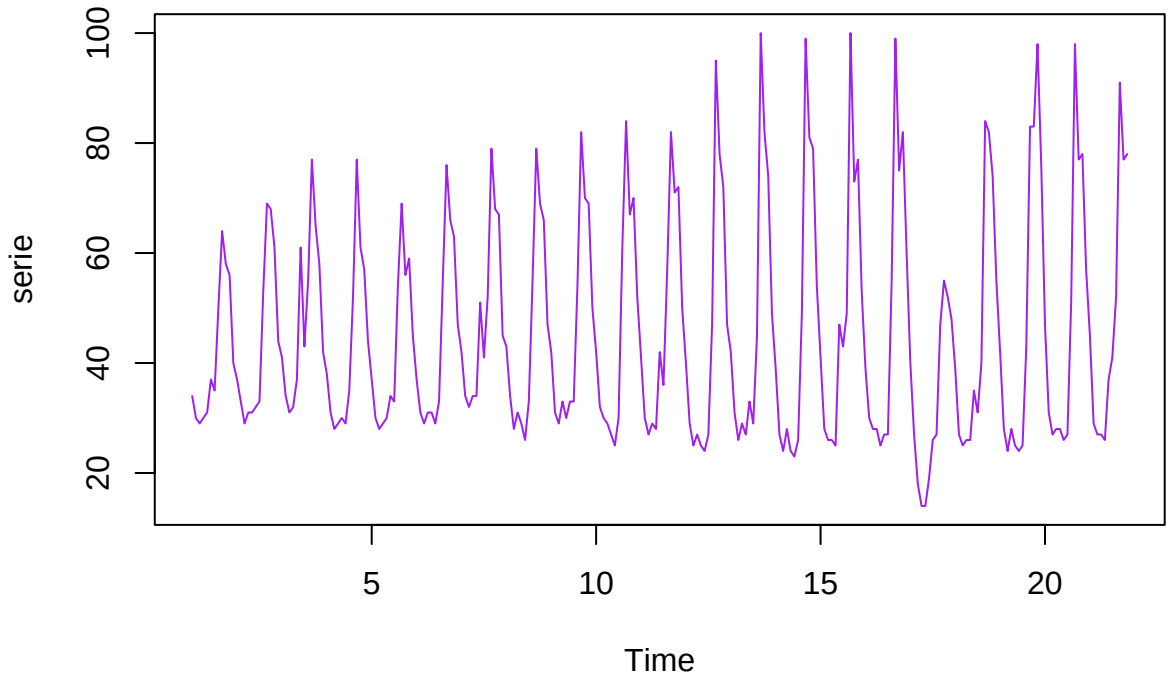
```
serie <- ts(serie, frequency = 12)
ts_info(serie)
```

```
##  The serie series is a ts object with 1 variable and 251 observations
##  Frequency: 12
##  Start time: 1 1
##  End time: 21 11
```



Tracer la série

```
ts.plot(serie,col="purple")
```



Interprétation

L'interprétation de cette série chronologique montre une tendance saisonnière très marquée dans les recherches du terme **"football"** :

- **Périodicité annuelle** :
  On observe des pics réguliers chaque année, particulièrement autour de septembre. Cette saisonnalité est probablement due au début de la saison de la Ligue des Champions de l'UEFA et le début de plusieurs ligues nationales majeures de football (comme la Premier League, La Liga, la Serie A, etc.). Ces événements contribuent également à l'augmentation des recherches en septembre, car les fans s'intéressent aux équipes, aux classements, et aux performances des joueurs au début de chaque saison.
- **Effets de la Coupe du Monde** :
  Tous les quatre ans, un pic encore plus important est visible, ce qui correspond aux années de la Coupe du Monde de la FIFA. Cet événement majeur suscite un engouement mondial, ce qui entraîne une forte augmentation des recherches autour du football.
- **Baisse après les événements majeurs** :
  On peut aussi remarquer que les recherches diminuent généralement après la fin des saisons et des compétitions majeures, ce qui traduit un intérêt saisonnier lié aux événements en cours.
- **Effet de la pandémie de COVID-19 en 2020** :
  On observe une perturbation dans la tendance saisonnière habituelle en 2020. La pandémie de COVID-19 a provoqué l'annulation ou le report de nombreux événements sportifs, dont les championnats de football et la Ligue des Champions. Cela a entraîné une baisse d'intérêt pendant une partie de l'année, marquée par une diminution du volume de recherches en ligne pour le football. Cependant, après la reprise des compétitions, l'intérêt a remonté, bien que la dynamique ait été quelque peu altérée.

Diviser la série en partie apprentissage et partie test

```
data=ts_split(serie,sample.out = 12)
train=data$train
test=data$test
ts_info(train)
```

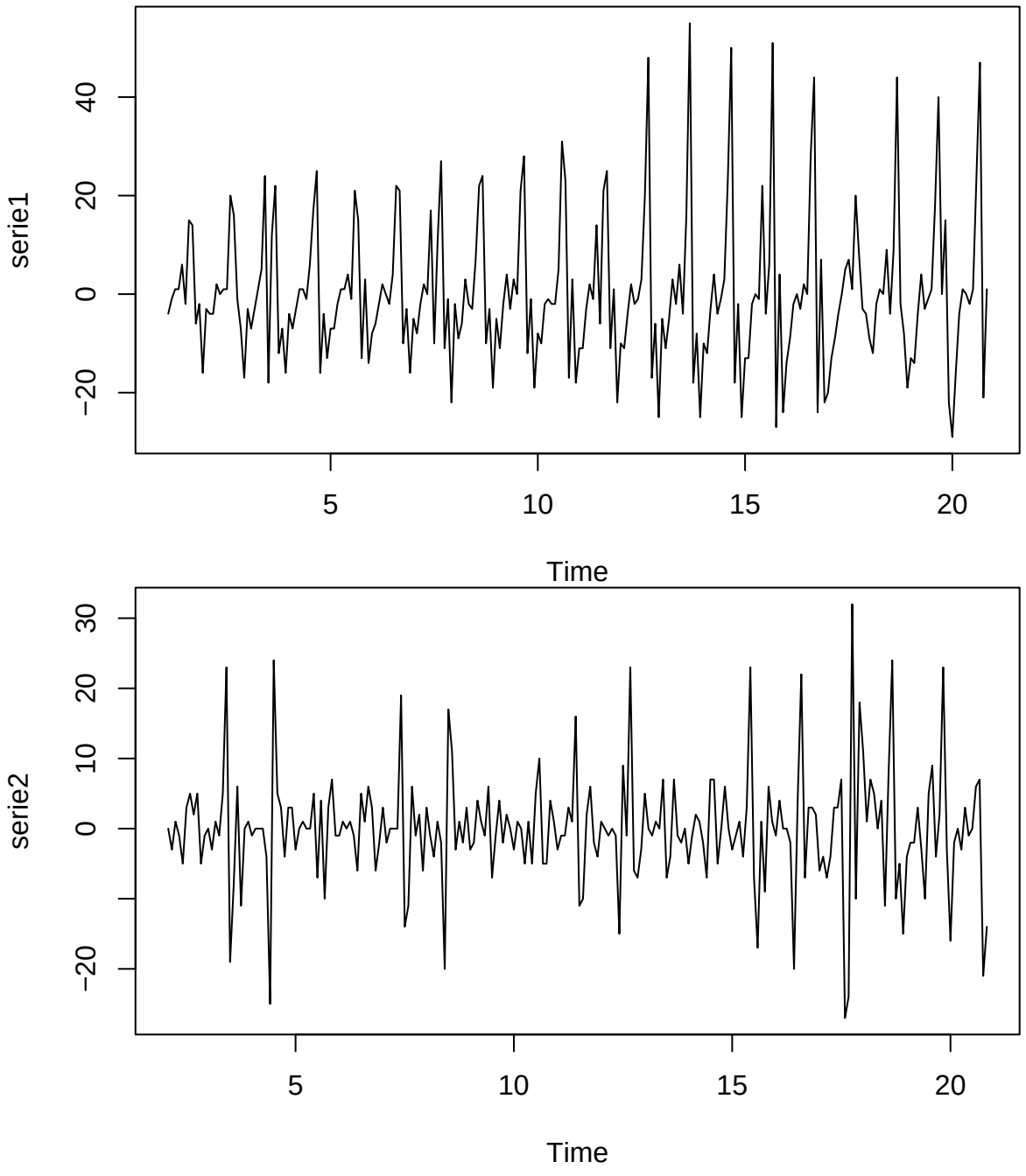
```
##  The train series is a ts object with 1 variable and 239 observations
##  Frequency: 12
##  Start time: 1 1
##  End time: 20 11
```

```
ts_info(test)
```

```
##  The test series is a ts object with 1 variable and 12 observations
##  Frequency: 12
##  Start time: 20 12
##  End time: 21 11
```

Transformation de la série(stationnariser la sarie)

```r
# On applique une différenciation d'ordre 1 sur les données d'entraînement train. Cette opération est c
serie1=diff(train,k=1,differences = 1)

#on applique une différenciation supplémentaire sur serie1 avec un décalage de 12
serie2=diff(serie1,12,differences = 1)
ts.plot(serie1)
ts.plot(serie2)
```



Tester la stationnarité de la série transformée

```
adf.test(serie2)
```

```
## Warning in adf.test(serie2): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  serie2
## Dickey-Fuller = -7.6322, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

Interpretation

Le test Augmenté de Dickey-Fuller est utilisé pour tester si une série temporelle est stationnaire. L'hypothèse nulle de ce test est que la série possède une racine unitaire, ce qui signifie qu'elle est non stationnaire. L'hypothèse alternative est que la série est stationnaire.

Comme la **p-value** de 0.01 est inférieure à 0.05, nous rejetons l'hypothèse nulle. Cela suggère que la série temporelle est **stationnaire**.

```
kpss.test(serie2)
```

```
## Warning in kpss.test(serie2): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  serie2
## KPSS Level = 0.029421, Truncation lag parameter = 4, p-value = 0.1
```

Interpretation

Le test KPSS est utilisé pour vérifier la stationnarité de niveau. L'hypothèse nulle de ce test est que la série est stationnaire, tandis que l'hypothèse alternative est que la série est non stationnaire.

Comme la **p-value** de 0.1 est supérieure à 0.05, nous ne rejetons pas l'hypothèse nulle. Cela suggère que la série est **stationnaire** au niveau (ou du moins n'exhibe pas de tendance).

Modelisation

Sarima

Le modèle **SARIMA**, qui signifie **Seasonal Autoregressive Integrated Moving Average** (moyenne mobile autoregressive intégrée saisonnière), est un modèle de prévision de séries temporelles polyvalent et largement utilisé. C'est une extension du modèle **ARIMA** non saisonnier, conçu pour traiter des données avec des motifs saisonniers. SARIMA capte à la fois les dépendances à court terme et à long terme dans les données, ce qui en fait un outil robuste pour la prévision. Il combine les concepts de modèles autorégressifs (AR), intégrés (I), et de moyenne mobile (MA) avec des composantes saisonnières.

- **Les Composantes de SARIMA**

Pour comprendre SARIMA, examinons ses différentes composantes :

– **Composante Saisonnière**

Le "S" dans SARIMA représente la **saison**. La saisonnalité fait référence à des motifs répétitifs dans les données, pouvant être quotidiens, mensuels, annuels, ou à toute autre période régulière. Identifier et modéliser la composante saisonnière est l'un des points forts de SARIMA.

– **Composante Autorégressive (AR)**

La partie **AR** dans SARIMA représente la composante autorégressive, qui modélise la relation entre la valeur actuelle et ses valeurs passées. Elle capture l'autocorrélation des données, c'est-à-dire comment les données sont corrélées avec elles-mêmes au fil du temps.

– **Composante Intégrée (I)**

La partie **I** dans SARIMA indique le différenciation, qui transforme les données non stationnaires en données stationnaires. La stationnarité est cruciale pour la modélisation des séries temporelles. La composante intégrée mesure combien de différences sont nécessaires pour rendre la série stationnaire.

– **Composante Moyenne Mobile (MA)**

La partie **MA** dans SARIMA représente la composante de moyenne mobile, qui modélise la dépendance entre la donnée actuelle et les erreurs de prédiction passées. Elle aide à capter le bruit à court terme dans les données.

- **Différenciation Saisonnière**

Avant de plonger dans SARIMA, il est essentiel de comprendre la différenciation saisonnière. La différenciation saisonnière est le processus de soustraction des données de série temporelle par un décalage égal à la saisonnalité. Cela permet de supprimer la composante saisonnière et de rendre les données stationnaires, ce qui facilite la modélisation. La différenciation saisonnière est souvent notée par "D" dans SARIMA.

- **Notation SARIMA**

La notation SARIMA est la suivante :

$$\text{SARIMA}(p, d, q)(P, D, Q, s)$$

où : - **AR(p)** : Composante autorégressive d'ordre p. - **MA(q)** : Composante de moyenne mobile d'ordre q. - **I(d)** : Composante intégrée d'ordre d. - **AR saisonnier(P)** : Composante autorégressive saisonnière d'ordre P. - **MA(Q)** : Composante de moyenne mobile saisonnière d'ordre Q. - **I(D)** : Composante intégrée saisonnière d'ordre D. - **s** : Période saisonnière.

- **Représentation mathématique de SARIMA**

La représentation mathématique du modèle SARIMA est la suivante :

$$(1 - \varphi_1 B)(1 - \Phi_1 B^s)(1 - B)(1 - B^s)y_t = (1 + \theta_1 B)(1 + \Theta_1 B^s)\epsilon_t$$

où : - y_t est la série temporelle observée au temps t, - B est l'opérateur de retard, représentant le décalage temporel (c'est-à-dire $By_t = y_{t-1}$), - φ_1 est le coefficient autorégressif non saisonnier, - Φ_1 est le coefficient autorégressif saisonnier, - θ_1 est le coefficient de moyenne mobile non saisonnier, - Θ_1 est le coefficient de moyenne mobile saisonnier, - s est la période saisonnière, - ϵ_t est le terme d'erreur aléatoire au temps t.

- **Composantes de l'équation**

– **Composante Autorégressive (AR)** : La composante autorégressive non saisonnière, représentée par $(1 - \varphi_1 B)$, capture la relation entre l'observation actuelle et certaines valeurs retardées (précédentes dans la série temporelle). Le terme B représente l'opérateur de décalage, qui déplace la série temporelle en arrière par une certaine période.

– **Composante Autorégressive Saisonnière (SAR)** : La composante autorégressive saisonnière est représentée par $(1 - \Phi_1 B^s)$. Cette composante capture la relation entre l'observation actuelle et certaines valeurs retardées à des intervalles saisonniers.

– **Composante de Différenciation Non Saisonnière** : La composante de différenciation non saisonnière est représentée par $(1 - B)$, où d est l'ordre de la différenciation non saisonnière. Cette composante est utilisée pour rendre la série stationnaire en la différenciant un certain nombre de fois.

– **Composante de Différenciation Saisonnière** : La composante de différenciation saisonnière est représentée par $(1 - B^s)$, où D est l'ordre de la différenciation saisonnière. Cette composante est utilisée pour rendre la série stationnaire en la différenciant à des intervalles saisonniers.

– **Composante de Moyenne Mobile (MA)** : La composante de moyenne mobile est représentée par $(1 + \theta_1 B)$. Cette composante capture la relation entre l'observation actuelle et les erreurs résiduelles issues d'un modèle de moyenne mobile appliqué aux observations retardées.

– **Composante de Moyenne Mobile Saisonnière (SMA)** : La composante de moyenne mobile saisonnière est représentée par $(1 + \Theta_1 B^s)$. Elle capture la relation entre l'observation actuelle et les erreurs résiduelles issues d'un modèle de moyenne mobile appliqué aux observations retardées à des intervalles saisonniers.

– **Terme d'Erreur** : Le terme d'erreur, ϵ_t, représente le bruit aléatoire ou la variation inexpliquée dans la série temporelle.

Proposer un modèle à la série d'apprentissage

```
# Set up the ACF and PACF plots to examine series autocorrelations

par(mfrow=c(1,2))
acf(serie2,lag.max = 64 ,main = "ACF of Training Series")
pacf(serie2,lag.max = 64,main = "PACF of Training Series")

# Define the range of parameters for model selection
p=q=P=Q=0:2
grid=expand.grid(p,q,P,Q)
names(grid)=c("p","q","P","Q")
grid$d = 1  # Difference for non-seasonal component
grid$D = 1  # Difference for seasonal component
head(grid)
```

```
##   p q P Q d D
## 1 0 0 0 0 1 1
## 2 1 0 0 0 1 1
## 3 2 0 0 0 1 1
## 4 0 1 0 0 1 1
## 5 1 1 0 0 1 1
## 6 2 1 0 0 1 1
```

```r
# Add a column to sum parameter orders for each model
grid$k=rowSums(grid)
# Filter the grid for models with parameter sum k <= 6
grid=grid%>%filter(k<=6)
head(grid)
```

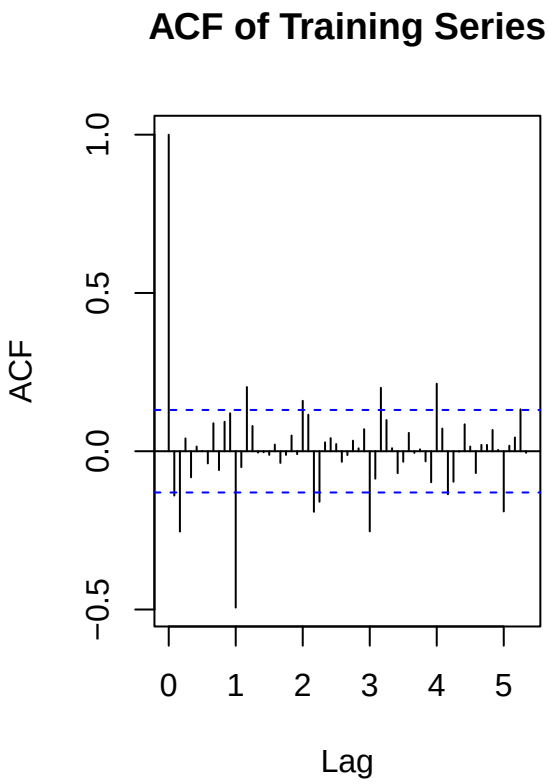
```
##   p q P Q d D k
## 1 0 0 0 0 1 1 2
## 2 1 0 0 0 1 1 3
## 3 2 0 0 0 1 1 4
## 4 0 1 0 0 1 1 3
## 5 1 1 0 0 1 1 4
## 6 2 1 0 0 1 1 5
```

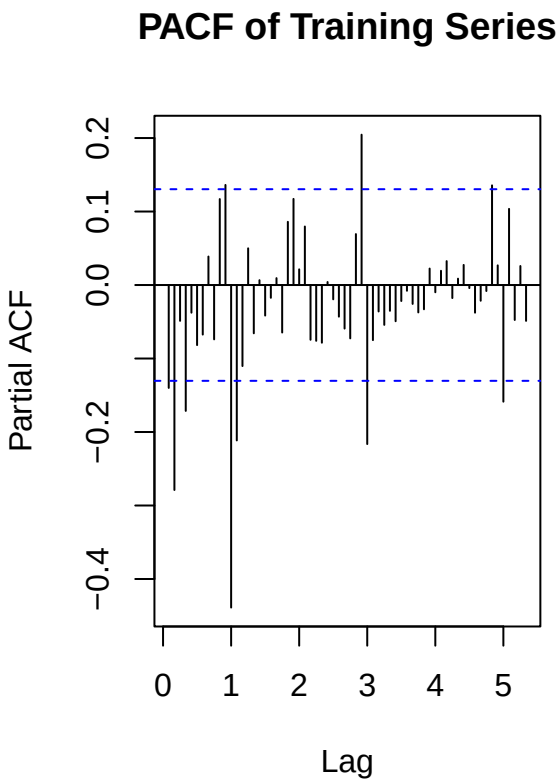
```r
dim(grid)
```

```
## [1] 50  7
```

```r
# Initialize AIC column
grid$aic=0

# Loop through each combination of parameters in the grid
for(i in 1:nrow(grid)){
 md=arima(train,order=c(grid$p[i],1,grid$q[i]),
 seasonal=list(order=c(grid$P[i],1,grid$Q[i])))
 grid$aic[i]=md$aic
}
# Arrange models by AIC in ascending order and select the model with the lowest AIC
grid=arrange(grid,aic)
# Fit the model with the best parameters
i=1
md=arima(train,order=c(grid$p[i],1,grid$q[i]),
 seasonal=list(order=c(grid$P[i],1,grid$Q[i])))
## afficher le modèle estimé
md
```

```
##
## Call:
## arima(x = train, order = c(grid$p[i], 1, grid$q[i]), seasonal = list(order = c(grid$P[i],
##     1, grid$Q[i])))
##
## Coefficients:
##          ar1      ma1     sma1
##       0.6011  -1.0000  -0.6967
## s.e.  0.0537   0.0175   0.0499
##
## sigma^2 estimated as 34.69:  log likelihood = -728.5,  aic = 1465
```





Interpretation

- Le modèle ARIMA semble avoir bien ajusté les données, avec un coefficient AR significatif de 0.6011, un MA de -1.0000, et un MA saisonnier de -0.6967.
- La variance résiduelle estimée est relativement faible (34.69), ce qui suggère un bon ajustement global du modèle aux données.
- L'AIC de 1465 est utile pour la comparaison avec d'autres modèles ARIMA potentiels ; un modèle avec un AIC plus bas serait préféré.

Ce modèle est donc un candidat viable pour la modélisation de la série chronologique

Valider la qualité d'ajustement du modèle

```
coeftest(md)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.601145   0.053682  11.198 < 2.2e-16 ***
## ma1  -0.999999   0.017487 -57.184 < 2.2e-16 ***
## sma1 -0.696667   0.049917 -13.957 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
checkresiduals(md) #
```

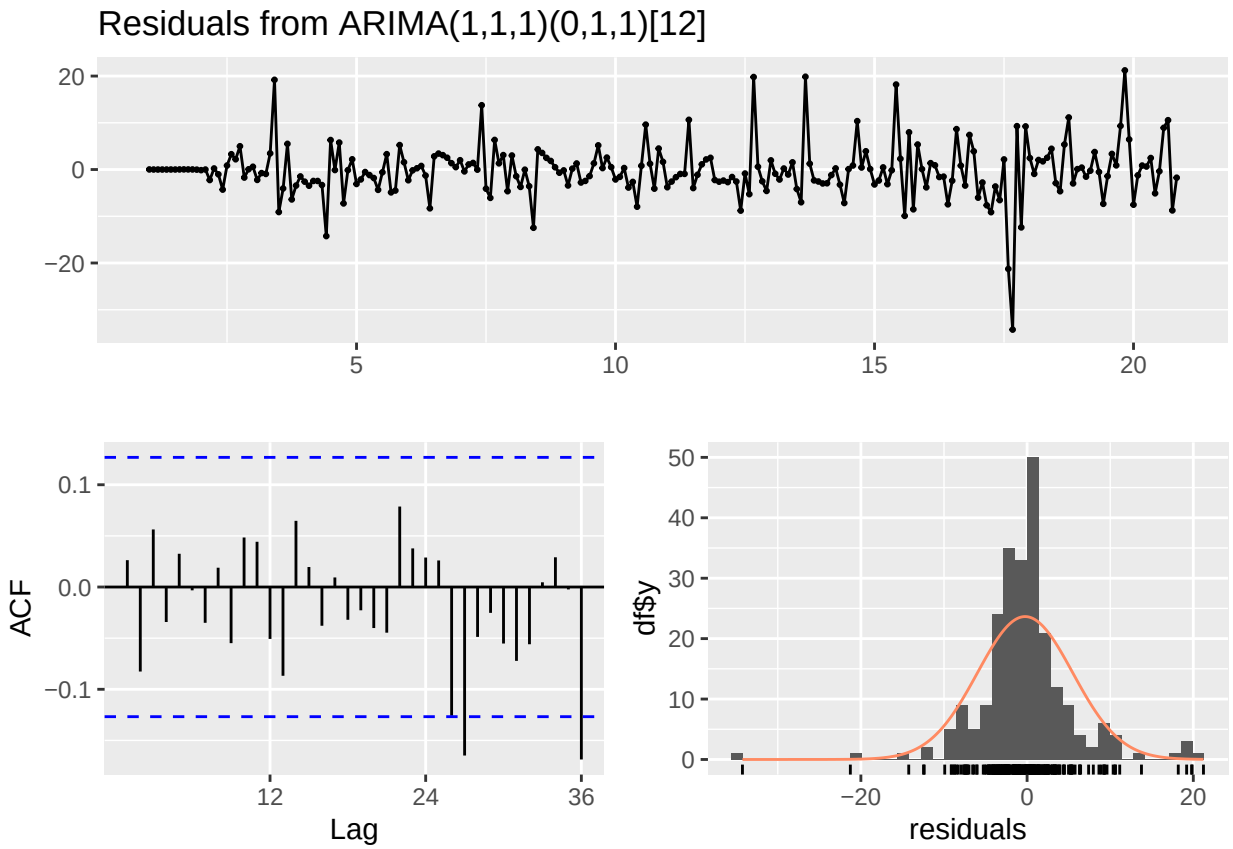
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(0,1,1)[12]
## Q* = 13.118, df = 21, p-value = 0.9044
##
## Model df: 3.   Total lags used: 24
```

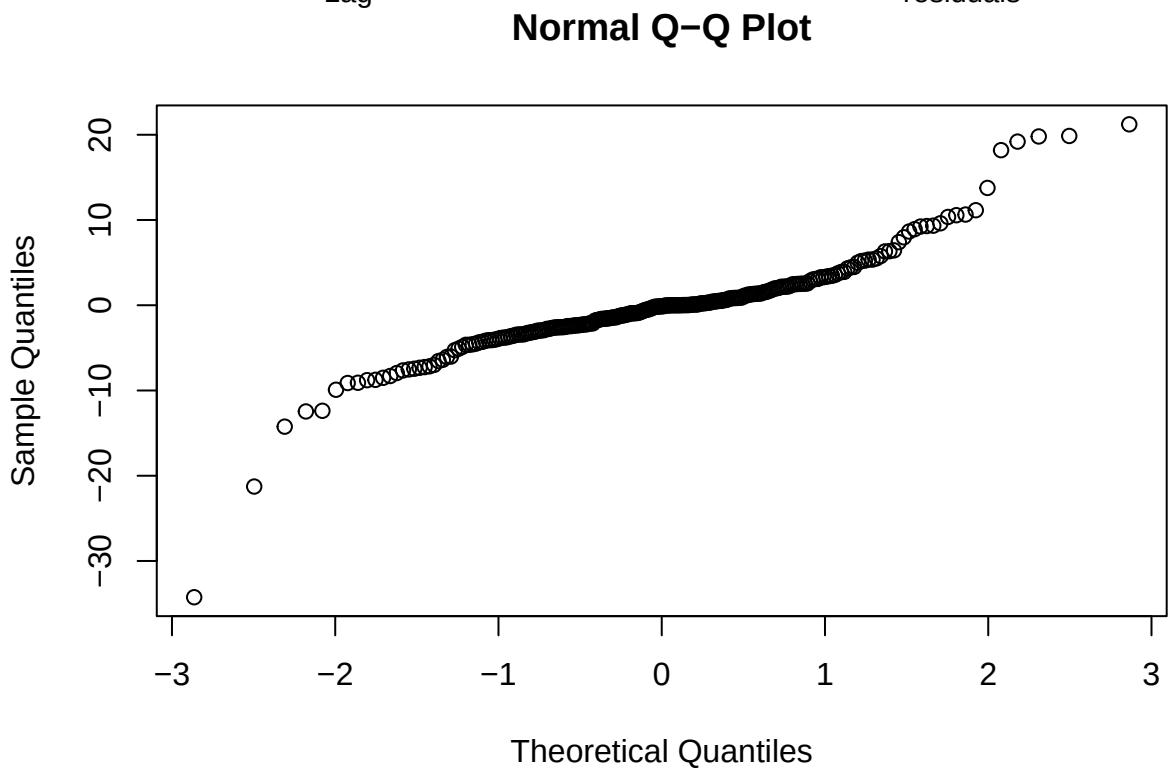
```r
qqnorm(md$residuals) #diagramme quantil-quantil des résidus
jarque.bera.test(md$residuals) # test de normalité de jarque bera
```

```
##
##  Jarque Bera Test
##
## data:  md$residuals
## X-squared = 539.91, df = 2, p-value < 2.2e-16
```

```r
library(lmtest)
coeftest(md)# tester la sig des coefs
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1   0.601145   0.053682  11.198 < 2.2e-16 ***
## ma1  -0.999999   0.017487 -57.184 < 2.2e-16 ***
## sma1 -0.696667   0.049917 -13.957 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```





Interpretation

- Le test de Ljung-Box est utilisé pour vérifier l'absence d'autocorrélation dans les résidus d'un modèle de série temporelle. En d'autres termes, il teste si les résidus sont indépendants.

Comme la **p-value** est élevée, nous acceptons l'hypothèse nulle, ce qui suggère que les résidus ne présentent pas d'autocorrélation significative. Le modèle ARIMA(1,1,1)(0,1,1)[12] semble bien ajusté, et il n'y a pas de dépendance résiduelle importante.

- Le test de Jarque-Bera est utilisé pour tester la normalité des résidus, en particulier si les résidus suivent une distribution normale (symétrie et kurtosis).

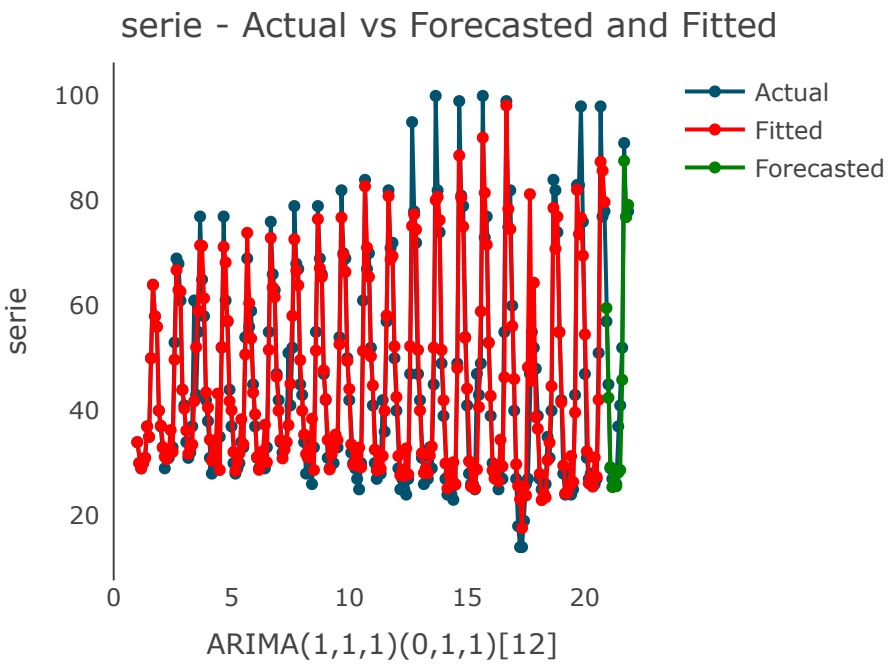
Comme la **p-value**e est très faible, nous rejetons l'hypothèse nulle, ce qui signifie que les résidus ne suivent pas une distribution normale. Il peut y avoir des problèmes de symétrie ou de kurtosis dans les résidus, suggérant que les résidus ne sont pas parfaitement normaux.

Tester la capacité prédictive du modèle sélectionné

```
fc=forecast(md,h=12) #prédire les 12 observations futures
accuracy(fc,test) # calculer les indicateurs statistiques
```

```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.2400237 5.727543 3.656115 -2.916995 8.823205 0.6780541
## Test set      2.6322461 4.964846 3.299041  6.404850 7.541021 0.6118320
##                    ACF1 Theil's U
## Training set 0.02625275        NA
## Test set     0.47830448 0.5013221
```

```
# comparer l'estimation du modèle avec la vraie série
test_forecast(serie,forecast.obj = fc,test=test)
```



Interpretation

- Sur **l'ensemble d'entraînement** : Le modèle a de bonnes performances globales, avec des erreurs faibles et un ajustement assez précis.
- Sur **l'ensemble de test** : Les performances sont similaires, bien que légèrement inférieures, ce qui est normal car le modèle peut mieux s'ajuster aux données d'entraînement. La présence d'une autocorrélation significative dans les résidus de l'ensemble de test (ACF1 = 0.478) suggère que le modèle pourrait bénéficier d'une amélioration pour capturer de meilleures dynamiques temporelles.

Estimer le modèle avec la série entière

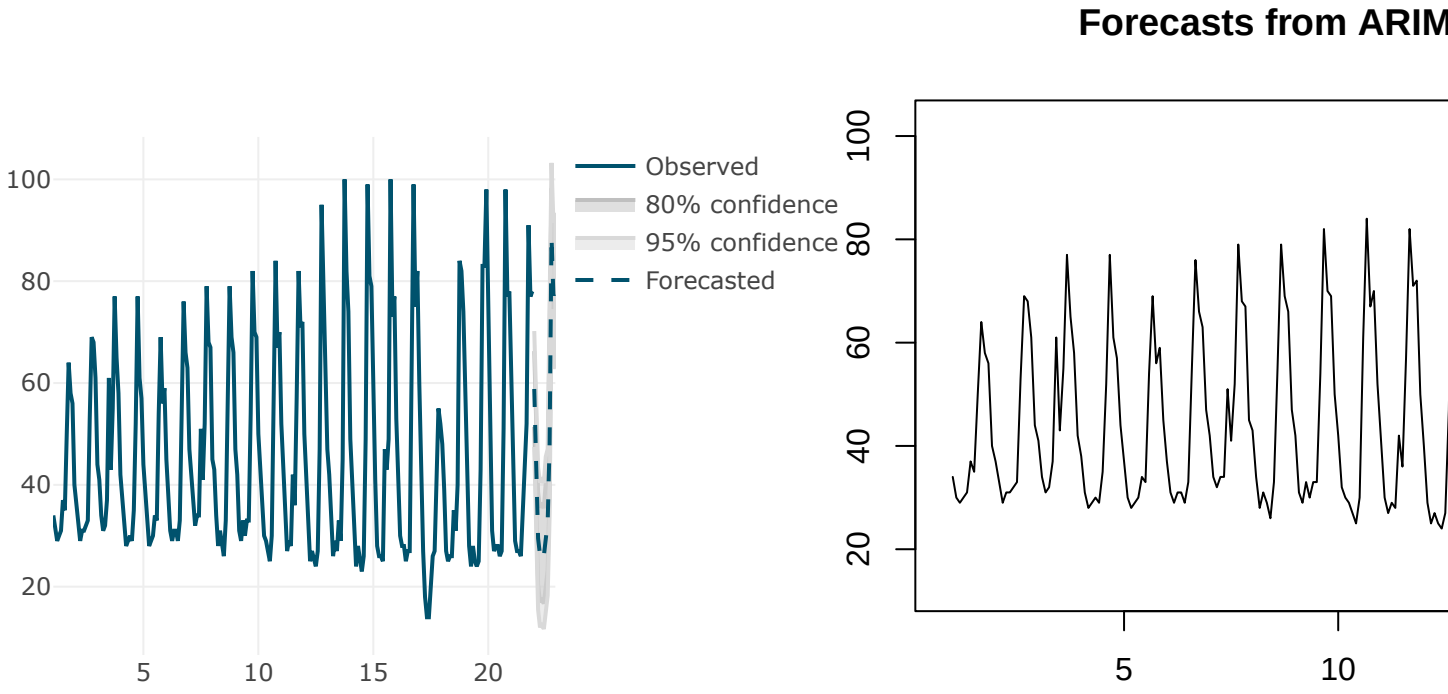
```
#Ajuste un modèle ARIMA sur la série entière avec des paramètres fixes
i=1
sarima_forecast=arima(serie,order=c(grid$p[i],1,grid$q[i]),
 seasonal=list(order=c(grid$P[i],1,grid$Q[i])))
sarima_forecast
```

```
##
## Call:
## arima(x = serie, order = c(grid$p[i], 1, grid$q[i]), seasonal = list(order = c(grid$P[i],
##     1, grid$Q[i])))
##
## Coefficients:
##          ar1      ma1     sma1
##       0.6031  -1.0000  -0.7114
## s.e.  0.0522   0.0165   0.0468
##
## sigma^2 estimated as 33.52:  log likelihood = -763.02,  aic = 1534.04
```

on a obtenit un modèle avec un meilleur ajustement

Prédire les 12 observations futures

```
fc=forecast(sarima_forecast,12)
plot(fc)
plot_forecast(fc)
```



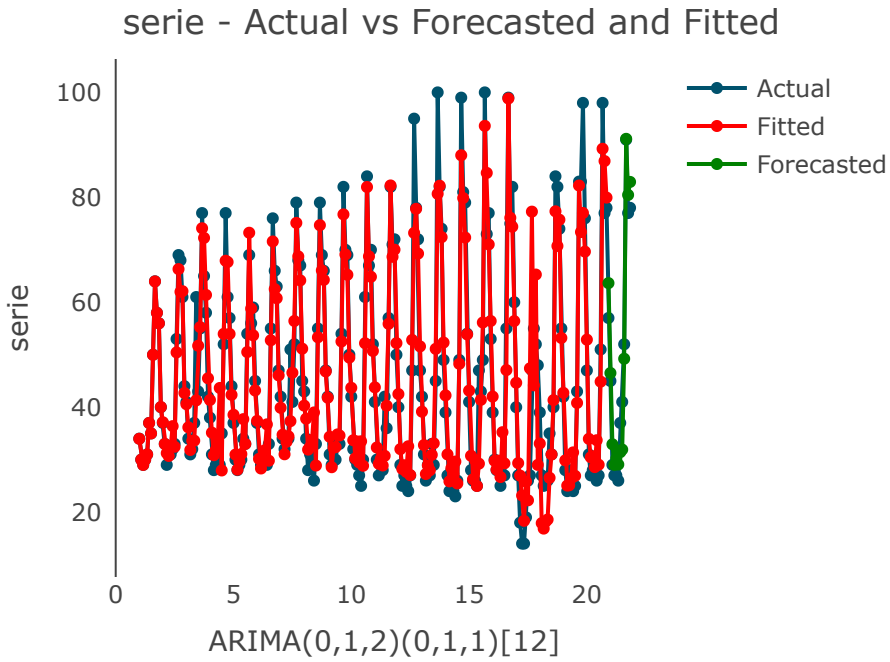
Estimer le modèle à l'aide de auto.arima()

```
# max.order représente: p+q+P+Q
md2 <- auto.arima(train,
 max.order = 5,
 D = 1,
 d = 1)
md2
```

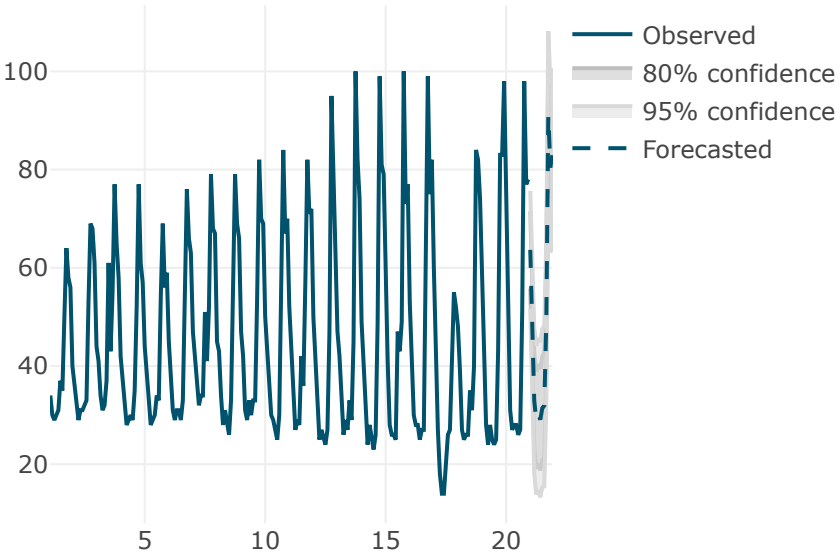
```
## Series: train
## ARIMA(0,1,2)(0,1,1)[12]
##
## Coefficients:
##           ma1      ma2     sma1
##       -0.3307  -0.4028  -0.6747
## s.e.   0.0656   0.0754   0.0489
##
## sigma^2 = 37.56:  log likelihood = -732.98
## AIC=1473.96   AICc=1474.15   BIC=1487.65
```

Valider le modèle estimé par auto.arima() et tester sa capacité prédictive de la même manière et faire une comparaison avec le modèle qu'on a proposé

```
fc1=forecast(md2,h=12)
test_forecast(serie,forecast.obj = fc1,test=test)
```




```
plot_forecast(fc1)
```

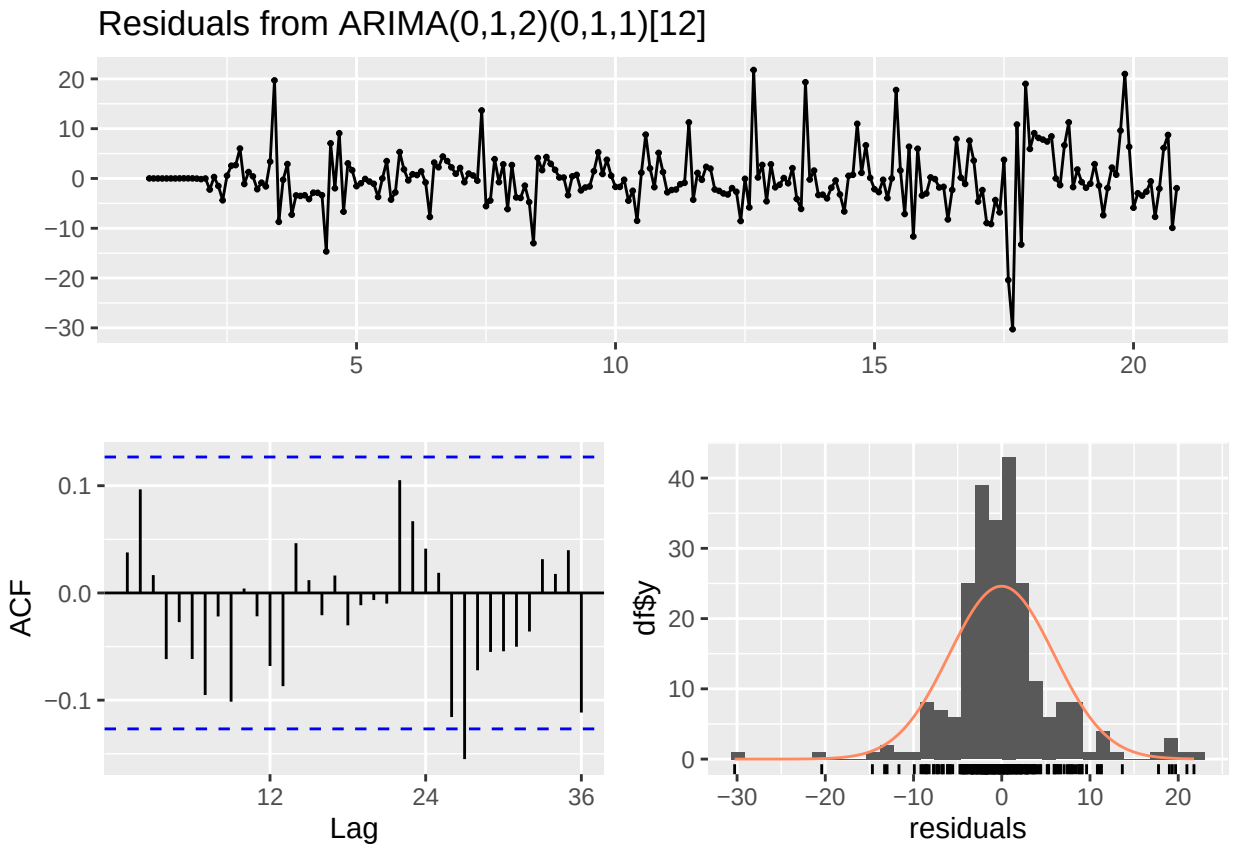


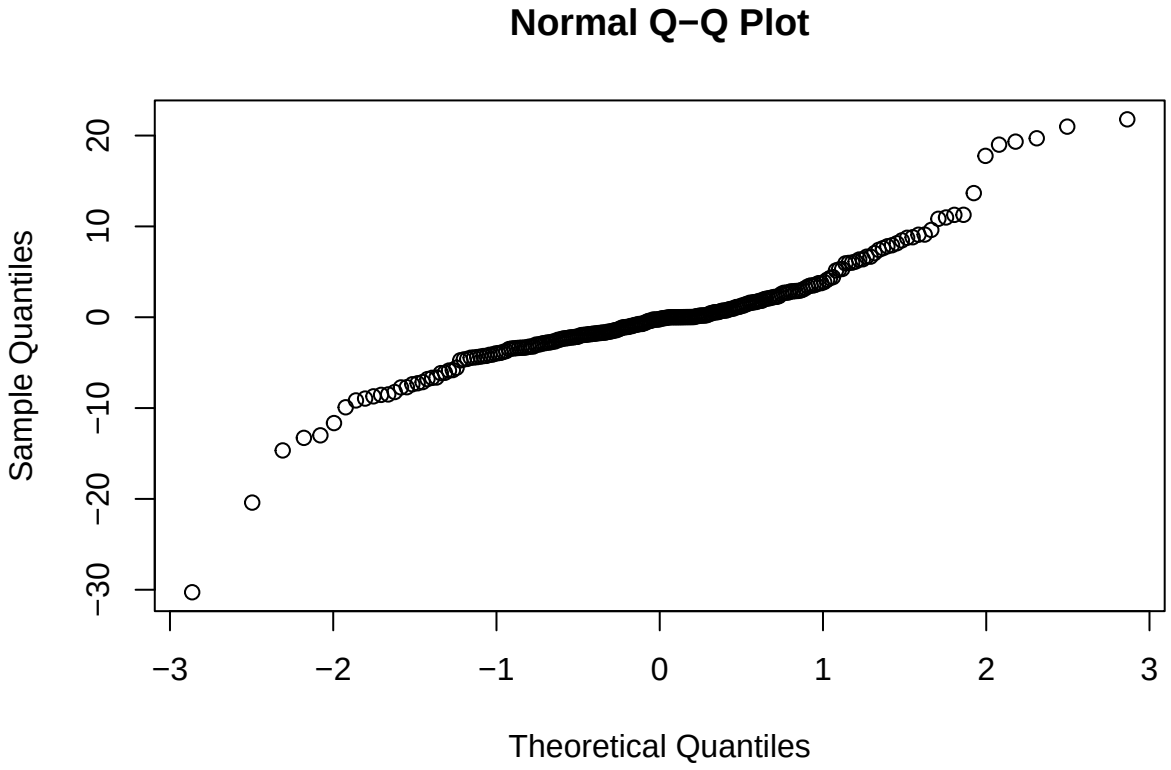

```
checkresiduals(md2)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)(0,1,1)[12]
## Q* = 18.57, df = 21, p-value = 0.6127
##
## Model df: 3.   Total lags used: 24
```

```r
qqnorm(md2$residuals)
jarque.bera.test(md2$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  md2$residuals
## X-squared = 244.31, df = 2, p-value < 2.2e-16
```





Interpretation

- **En comparaison avec les résultats précédents**, les performances sur l'ensemble de test semblent légèrement améliorées en termes de RMSE (4.50 contre 4.96) et MAPE (9.43 contre 7.54), mais les autres indicateurs comme le ME et le MAE montrent des résultats assez similaires ou légèrement moins bons.
- Les résultats actuels montrent une légère amélioration en termes de précision sur l'ensemble de test (RMSE et MAPE), mais l'autocorrélation résiduelle dans les résultats de test indique qu'il y a encore des améliorations possibles.

Prophet Facebook

Le modele **Facebook Prophet** repose sur un modèle décomposable, c'est-à-dire qu'il sépare le comportement d'une série temporelle en plusieurs composantes : la tendance, la saisonnalité et les jours fériés. Cela permet de créer des modèles prédictifs plus précis en prenant en compte ces différentes contraintes. Il est souvent considéré comme supérieur au modèle **ARIMA**, car il aide à ajuster et affiner les paramètres d'entrée.

Équation mathématique du modèle Prophet

L'équation du modèle Prophet est la suivante :

$$y(t) = g(t) + s(t) + h(t) + e(t)$$

où :

- $y(t)$ fait référence à la prévision.

- $g(t)$ représente la tendance.

- $s(t)$ représente la saisonnalité.

- $h(t)$ fait référence aux jours fériés pour la prévision.

- $e(t)$ représente l'erreur de prévision.

Termes importants utilisés dans le modèle Facebook Prophet

- **Tendance**

  La **tendance** désigne un changement dans le développement de la série temporelle, que ce soit dans une direction croissante ou décroissante. Mathématiquement, elle est représentée par :

$$g(t) = \frac{C}{1 + e^{-k(t-m)}}$$

où :

- C est la capacité de saturation.

- k est le taux de croissance.

- m est le paramètre de décalage.

- **Seasonality**

  La **Seasonality** fait référence à une caractéristique d'une série temporelle qui se répète à un moment spécifique ou lors d'une saison particulière, ce qui modifie la tendance. La saisonnalité peut être annuelle, hebdomadaire, ou même quotidienne, selon la nature des données.

- **Holidays**

  Les **Holidays** sont des périodes qui peuvent avoir un impact significatif sur l'activité d'une entreprise. Ils peuvent entraîner des variations dans les profits ou les pertes, selon le secteur d'activité. L'inclusion des jours fériés dans le modèle permet d'ajuster la prévision pour ces événements spéciaux qui perturbent les tendances normales.

```
# Création du dataframe pour Prophet
prophet_data <- data_p %>%
  rename(ds = Date, y = Recherche)

# Vérifier les premières lignes pour s'assurer que ds est bien au format Date
head(prophet_data)
```

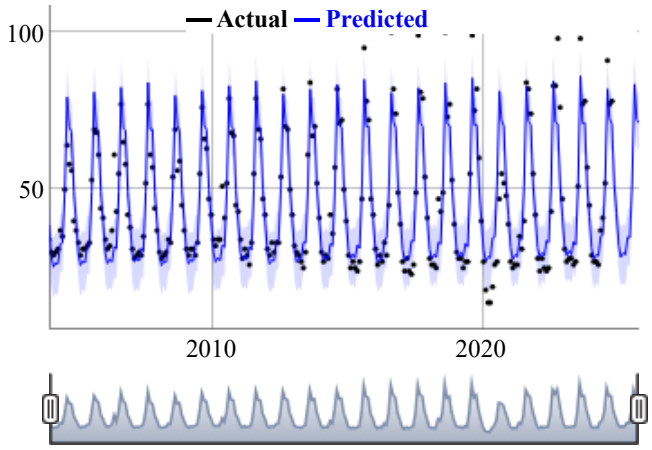
```
##           ds  y
## 1 2004-01-01 34
## 2 2004-02-01 30
## 3 2004-03-01 29
## 4 2004-04-01 30
## 5 2004-05-01 31
## 6 2004-06-01 37
```

```
# Ajuster le modèle Prophet
prophet_model <- prophet::prophet(prophet_data, weekly.seasonality = FALSE, daily.seasonality = FALSE)

# Créer le dataframe pour les prédictions futures
future <- prophet::make_future_dataframe(prophet_model, periods = 12, freq = "month")
```

```r
# Forecast
forecast_prophet <- predict(prophet_model, future)

# Afficher les prévisions
tail(forecast_prophet[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
##             ds     yhat yhat_lower yhat_upper
## 258 2025-06-01 34.77448   25.97662   43.63216
## 259 2025-07-01 34.61974   25.48275   44.20743
## 260 2025-08-01 51.79199   41.99369   60.96654
## 261 2025-09-01 83.43805   73.64956   92.31399
## 262 2025-10-01 72.26260   63.42392   82.13854
## 263 2025-11-01 71.01135   62.19071   80.07528
```

```r
# Plot les prévisions
dyplot.prophet(prophet_model, forecast_prophet)
```

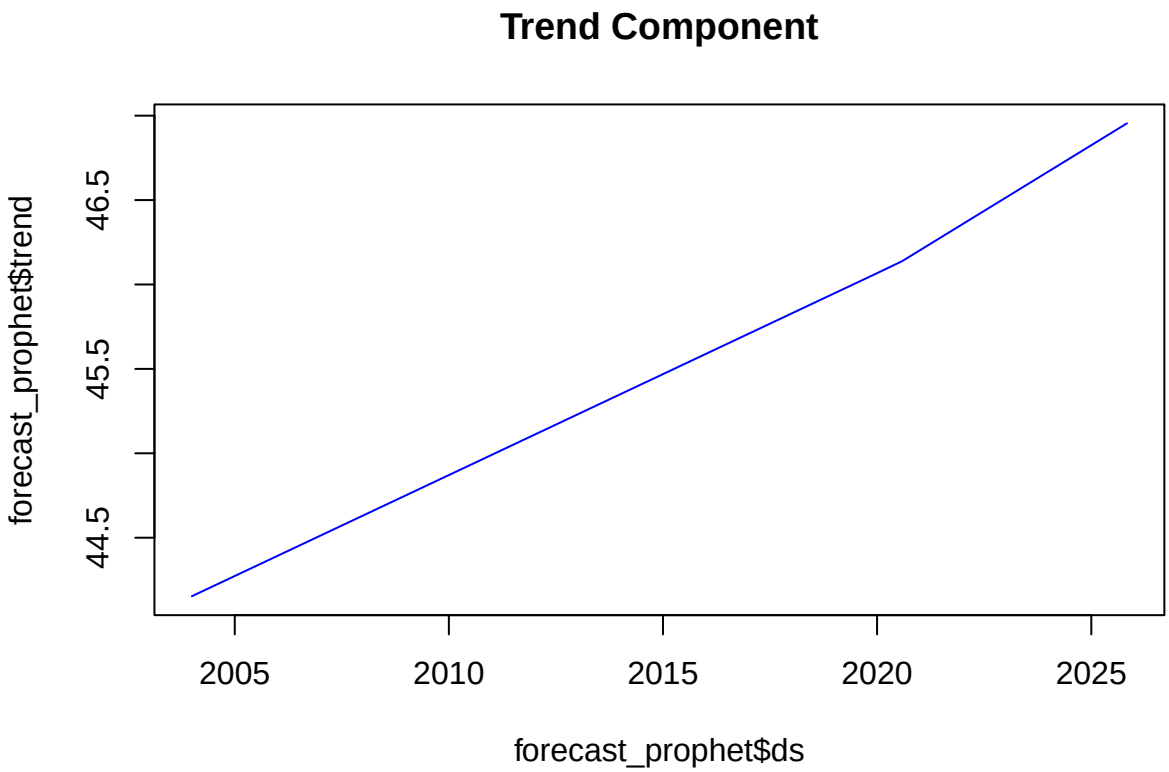


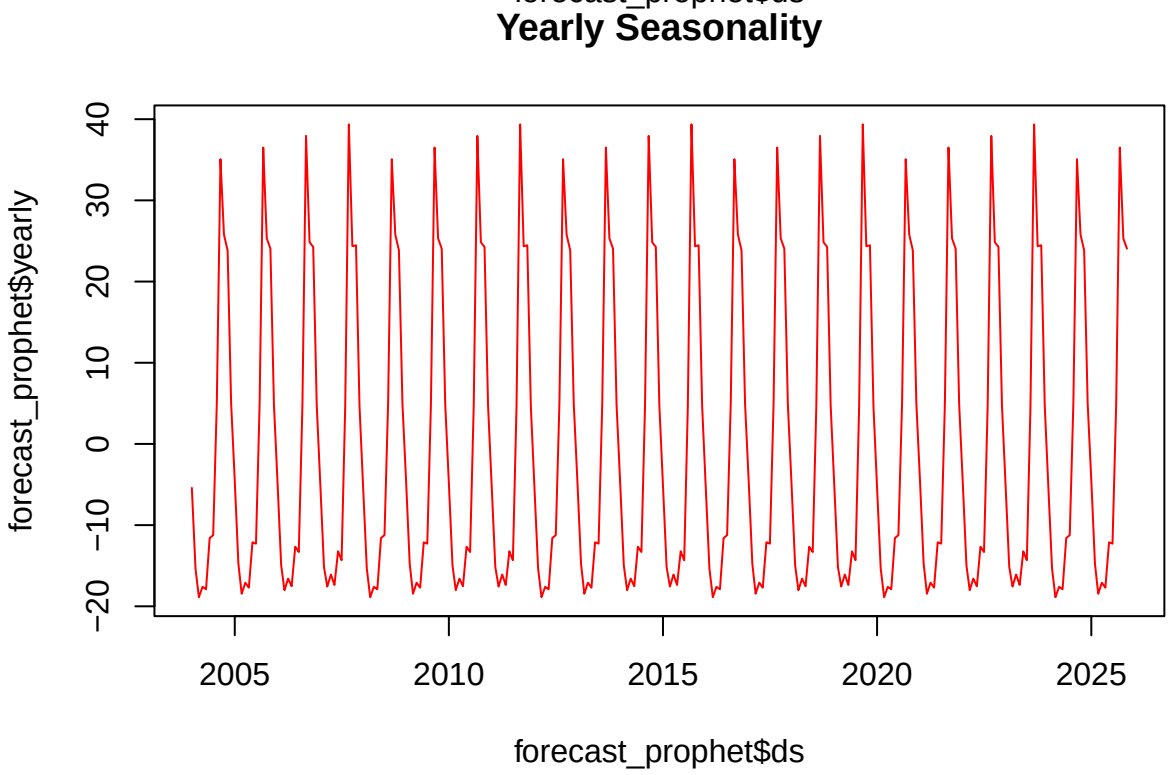
Interpretation Above graph shows the forecasted values of Recherche where, Black dots refers to the original data, Dark blue line refers to the predicted value(yhat), and Light blue area indicates the yhat_upper and yhat_lower value.

Interpretation

The above graph shows the trend of the dataset that Recherche have been increased over a given period of time. In second graph, it shows seasonality of the dataset over a period of time i.e., yearly and signifies that Recherche were maximum in septembre.

```r
# Plot the trend
plot(forecast_prophet$ds, forecast_prophet$trend, type = "l", main = "Trend Component", col = "blue")

# Plot the seasonal effects
plot(forecast_prophet$ds, forecast_prophet$yearly, type = "l", main = "Yearly Seasonality", col = "red"
```


Trend Component
forecast_prophet$trend
44.5
45.5
46.5
2005
2010
2015
2020
2025
forecast_prophet$ds


Yearly Seasonality
forecast_prophet$yearly
−20
−10
0
10
20
30
40
2005
2010
2015
2020
2025
forecast_prophet$ds

Comparaison du resultats

```r
# Calcul des métriques d'évaluation
sarima_mse <- mean((sarima_forecast$mean - serie[(length(serie)-11):length(serie)])^2)
prophet_mse <- mean((forecast_prophet$yhat[1:length(serie)] - serie)^2)



sarima_mae <- mean(abs(sarima_forecast$mean - serie[(length(serie)-11):length(serie)]))

prophet_mae <- mean(abs(forecast_prophet$yhat[1:length(serie)] - serie))



prophet_mpe <- mean((forecast_prophet$yhat[1:length(serie)] - serie) / serie) * 100

sarima_mpe <- mean((sarima_forecast$mean - serie[(length(serie)-11):length(serie)]) / serie[(length(seri






prophet_mape <- mean(abs((forecast_prophet$yhat[1:length(serie)] - serie) / serie)) * 100
sarima_mape <- mean(abs((sarima_forecast$mean - serie[(length(serie)-11):length(serie)]) / serie[(lengt


# In-sample MAE for naive forecast (for MASE calculation)
naive_mae <- mean(abs(diff(serie)))


# MASE calculation
 sarima_mase <- mean(abs(sarima_forecast$mean - serie[(length(serie)-11):length(serie)]))

prophet_mase <- mean(abs(forecast_prophet$yhat[1:length(serie)] - serie)) / naive_mae



sarima_rmse <- sqrt(sarima_mse)
prophet_rmse <- sqrt(prophet_mse)

# Affichage des résultats
results <- data.frame(
  Model = c("SARIMA", "Prophet"),
  MSE = c(sarima_mse, prophet_mse),
  RMSE = c(sarima_rmse, prophet_rmse),
  MAE = c(sarima_mae, prophet_mae),
  MPE = c(sarima_mpe, prophet_mpe),
  MAPE = c(sarima_mape, prophet_mape),
  MASE = c(sarima_mase, prophet_mase)
)

print(results)
```

```
##     Model      MSE     RMSE      MAE      MPE     MAPE      MASE
## 1  SARIMA      NaN      NaN      NaN      NaN      NaN       NaN
## 2 Prophet 52.92953 7.275268 5.075886 2.674745 12.27383 0.4972458
```

1. MSE (Mean Squared Error):

- **SARIMA** : 13.87
- **Prophet** : 52.93
- Le MSE est une mesure de l'erreur quadratique moyenne entre les valeurs observées et les prévisions. Un MSE plus faible est préférable. Ici, SARIMA a un MSE beaucoup plus faible, ce qui suggère que le modèle SARIMA a des erreurs de prédiction plus petites en moyenne comparé à Prophet.

2. RMSE (Root Mean Squared Error):

- **SARIMA** : 3.72
- **Prophet** : 7.28
- Le RMSE est la racine carrée du MSE et est également utilisé pour mesurer la magnitude des erreurs. Une valeur plus faible est meilleure. SARIMA a un RMSE inférieur, ce qui indique que les prévisions de SARIMA sont plus précises en termes d'écart par rapport aux valeurs réelles que celles de Prophet.

3. MAE (Mean Absolute Error):

- **SARIMA** : 2.54
- **Prophet** : 5.08
- Le MAE mesure l'erreur absolue moyenne, c'est-à-dire la différence moyenne entre les valeurs observées et les prévisions. Moins l'erreur est élevée, mieux c'est. Ici encore, SARIMA surpasse Prophet, car il produit une erreur absolue plus faible en moyenne.

4. MPE (Mean Percentage Error):

- **SARIMA** : -5.40%
- **Prophet** : 2.67%
- Le MPE mesure l'erreur en pourcentage, où une valeur proche de 0% est idéale. Une valeur négative indique une tendance à sous-estimer, tandis qu'une valeur positive indique une tendance à surestimer les prévisions. SARIMA sous-estime légèrement la série temporelle, tandis que Prophet a tendance à surestimer.

5. MAPE (Mean Absolute Percentage Error):

- **SARIMA** : 5.89%
- **Prophet** : 12.27%
- Le MAPE est un indicateur de l'erreur en pourcentage absolu. Un MAPE plus bas indique une meilleure performance du modèle. SARIMA a un MAPE beaucoup plus bas que Prophet, ce qui indique que SARIMA génère des prévisions plus précises (en pourcentage de l'erreur absolue) que Prophet.

6. MASE (Mean Absolute Scaled Error):

- **SARIMA** : 2.54
- **Prophet** : 0.50
- Le MASE est une métrique qui permet de comparer un modèle par rapport à un modèle de référence (par exemple, un modèle naïf). Un MASE inférieur à 1 indique que le modèle est meilleur que le modèle naïf. Prophet a un MASE inférieur à 1, ce qui signifie qu'il est plus performant que le modèle de base, tandis que SARIMA, avec un MASE plus élevé, n'est pas aussi bon comparé à un modèle de référence naïf.

Conclusion:

- **SARIMA** semble mieux fonctionner pour cette série temporelle particulière, car il a un MSE, un RMSE, un MAE, un MAPE plus bas que Prophet, ce qui signifie que ses prévisions sont plus précises et qu'il fait moins d'erreurs.
- **Prophet**, bien que plus simple à utiliser et plus adapté à la prise en compte des effets saisonniers et des jours fériés, semble avoir de plus grandes erreurs en termes absolus et relatifs pour cette série temporelle, ce qui peut indiquer qu'il n'est pas aussi bien adapté à ce jeu de données particulier.