REPUBLIQUE DU SENEGAL

Université Alioune Diop
L'Excellence est ma constance, l'Éthique ma vertu !

UFR : Science Appliquée de Technologie de l'Informatique et de la Communication (SATIC)

Master en Statistique et Informatique Décisionnelle (SID)

Projet de Séries temporelles

Sujet : « Etude de la tendance d'une série temporelle (production électrique mensuelle de 1985 à 2018), ensuite prédire la production électrique pour les années à venir en utilisant les processus ARIMA et la méthode de lissage. »

Présenté par :

Amadou BA

Mahmoud SIDIBE

Année académique :

2022-2023

Encadreur :

Pr Aba DIOP

Table des matières

Contexte:

On souhaite modéliser les données de la production électrique mensuelle de 1985 à 2018 d'un pays donné. L'objectif de cette étude consiste d'abord à étudier la tendance de la série, ensuite prédire la production électrique pour les années à venir en utilisant les processus ARIMA et la méthode de lissage.

Travail à faire

1) Décomposez la série et faîtes une description et un ajustement de la tendance en utilisant un modèle linéaire simple et multiple. Comparer les deux modèles.

Chargement des packages

```
# Chargeons les packages nécessaires sans les afficher le code dans le document
library(forecast)

library(TSstudio)

library(tseries)

library(plotly)

library(tidyverse)

library(stats)

library(dplyr)
library(ggplot2)
library(dygraphs)

library(lubridate)

library(datasets)

library(base)

library(Quandl

library(ggfortify)

library(urca)

library(zoo)

library(TSA)
```

```
library(readxl)
```

Importation de la base

```
#Importation de la base
mabase <- read_excel("C:/Users/sidibe/Documents/Downloads/Projet/Aba/Base-Production_Electrique.xlsx")

## Affichage des 10 premiers valeurs de la base
mabase_head <- head(mabase, 10)
mabase_head

# A tibble: 10 × 2
   DATE                PRODUCTION
   <dttm>              <chr>
 1 1985-01-01 00:00:00 72.5052
 2 1985-01-02 00:00:00 70.672
 3 1985-01-03 00:00:00 62.4502
 4 1985-01-04 00:00:00 57.4714
 5 1985-01-05 00:00:00 55.3151
 6 1985-01-06 00:00:00 58.0904
 7 1985-01-07 00:00:00 62.6202
 8 1985-01-08 00:00:00 63.2485
 9 1985-01-09 00:00:00 60.5846
10 1985-01-10 00:00:00 56.3154

attach(mabase)
# Afficher les dernières lignes de mabase
tail(mabase)

# A tibble: 6 × 2
  DATE                PRODUCTION
  <dttm>              <chr>
1 2017-01-08 00:00:00 108.9312
2 2017-01-09 00:00:00 98.6154
3 2017-01-10 00:00:00 93.6137
4 2017-01-11 00:00:00 97.3359
5 2017-01-12 00:00:00 114.7212
6 2018-01-01 00:00:00 129.4048

# Vérifier si mabase est une série temporelle
is.ts(mabase)

[1] FALSE

# Obtenir un résumé statistique de mabase
summary(mabase)
```

```
     DATE                      PRODUCTION
 Min.   :1985-01-01 00:00:00.000   Length:397
 1st Qu.:1993-01-04 00:00:00.000   Class :character
 Median :2001-01-07 00:00:00.000   Mode  :character
 Mean   :2001-01-21 18:19:02.569
 3rd Qu.:2009-01-10 00:00:00.000
 Max.   :2018-01-01 00:00:00.000
```

Conversion en série temporelle

Conversion de la colonne de dates en format de Date et la colone production en Numeric

```
mabase$DATE <- as.Date(mabase$DATE, format = "%m/%d/%Y")
mabase$PRODUCTION <- as.numeric(mabase$PRODUCTION)
```

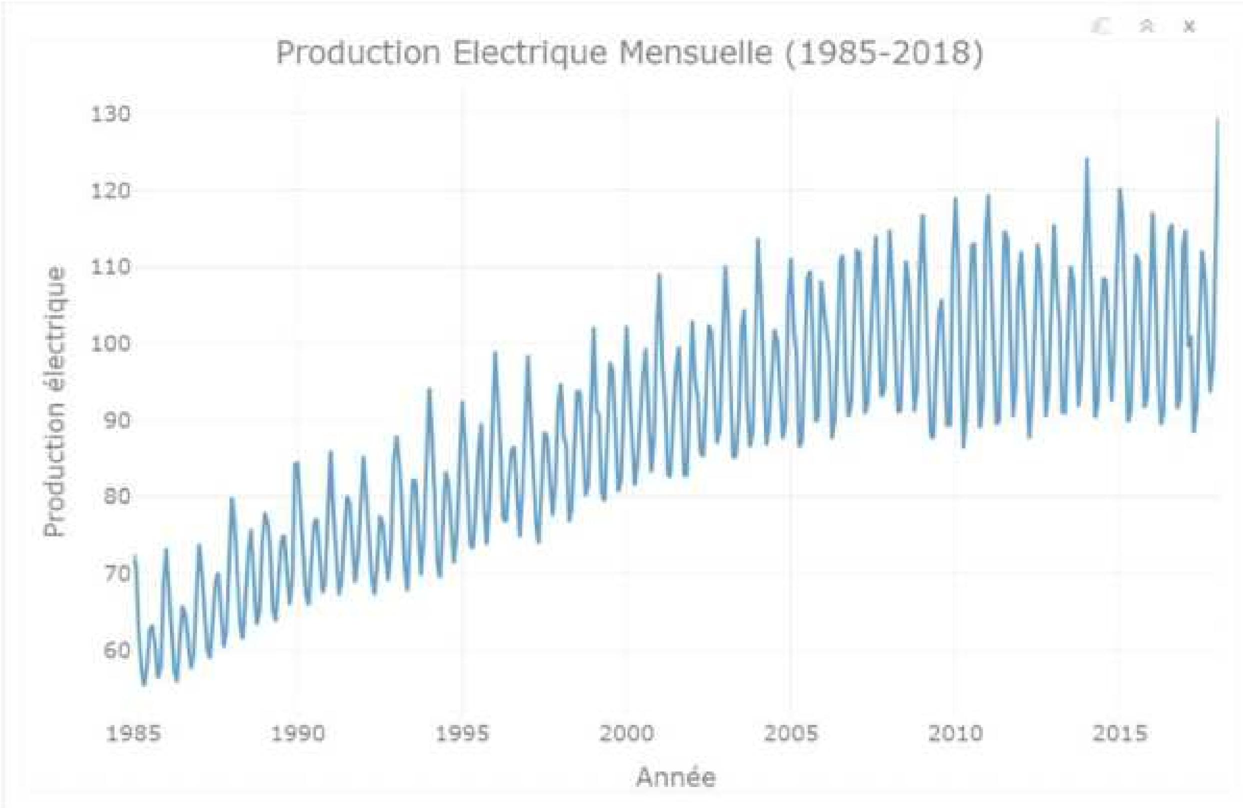
Création de la série temporelle

```
mabase_st <- ts(mabase$PRODUCTION, start = c(min(year(mabase$DATE)),
                                   min(month(mabase$DATE))), frequency = 12)
# Affichons les 50 premiers valeurs de notre base
mabase_st_head <- head(mabase_st, 50)
mabase_st_head

         Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep
1985 72.5052 70.6720 62.4502 57.4714 55.3151 58.0904 62.6202 63.2485 60.5846
1986 73.3057 67.9869 62.2221 57.0329 55.8137 59.9005 65.7655 64.4816 61.0005
1987 73.8152 70.0620 65.6100 60.1586 58.8734 63.8918 68.8694 70.0669 64.1151
1988 79.8703 76.1622 70.2928 63.2384 61.4065 67.1097 72.9816 75.7655 67.5152
1989 77.9188 76.6822
         Oct     Nov     Dec
1985 56.3154 58.0005 68.7145
1986 57.5322 59.3417 68.1354
1987 60.3789 62.4643 70.5777
1988 63.2832 65.1078 73.8631
1989
```

courbe de la production d’electricite en fonction des annees

```
# Conversion de la série temporelle en dataframe pour plotly
mabase_df <- data.frame(date = as.Date(time(mabase_st), origin = "1970-01-01"), production =
as.numeric(mabase_st))

# Création du graphique avec plotly
plot_ly(data = mabase_df,
        x = ~ date,
        y = ~ production,
        type = 'scatter',
        mode = 'lines') %>%
  layout(title = "Production Électrique Mensuelle (1985-2018)",
```

```
        yaxis = list(title = "Production électrique"),
        xaxis = list(title = "Année"))
```



La série temporelle montre une tendance globale à la hausse avec des variations saisonnières marquées. Il est possible de discerner des pics réguliers, ce qui suggère une forte saisonnalité.

Décomposons la série temporelle

La decomposition de la serie La décomposition consiste à séparer une série temporelle en plusieurs composantes :

Tendance : La tendance représente la variation à long terme de la série. Elle peut être linéaire ou non linéaire.

Saisonnalité : La saisonnalité est la variation périodique due à des facteurs saisonniers (par exemple, les saisons, les jours de la semaine, etc.).

Composante résiduelle : C’est la partie de la série qui ne peut pas être expliquée par la tendance et la saisonnalité. Elle contient le bruit aléatoire.

La fonction decompose sépare la série en tendance, saisonnalité et résidus.

```
base_decom <- decompose(mabase_st)
```

La décomposition de la série temporelle de la production électrique de 1985 à 2018 présente les composantes suivantes :

Tendance

```
# Afficher les 50 premières lignes de la tendence

trend_head <- head(base_decom$trend, 50)
trend_head

          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1985       NA       NA       NA       NA       NA       NA 62.19902 62.12050
1986 62.27651 62.45895 62.52765 62.59568 62.70227 62.73402 62.73112 62.83881
1987 64.18499 64.54704 64.90953 65.15792 65.40664 65.63851 65.99257 66.49904
1988 68.05069 68.45947 68.83859 69.10127 69.33243 69.57947 69.63505 69.57540
1989 70.35529 70.36895
          Sep      Oct      Nov      Dec
1985 61.99911 61.97134 61.97384 62.07004
1986 63.06644 63.33784 63.59556 63.88935
1987 66.94833 67.27178 67.50565 67.74527
1988 69.72455 69.92993 70.10295 70.25483
1989
```

Tendance (trend) : La composante de tendance montre une augmentation progressive de la production électrique sur la période étudiée. Cette croissance pourrait être due à une augmentation de la demande énergétique, des améliorations technologiques, ou des politiques énergétiques favorables. La tendance semble se stabiliser vers la fin de la période, indiquant une possible saturation ou un équilibre atteint dans la production.

Saisonnalité

```
# Afficher les 50 premières lignes de la saisonnalite
seasonal_head <- head(base_decom$seasonal, 50)
seasonal_head

            Jan        Feb        Mar        Apr        May        Jun
1985 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1986 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1987 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1988 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1989 13.2130393  5.7447473
            Jul        Aug        Sep        Oct        Nov        Dec
1985  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
1986  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
1987  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
```

```
1988  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
1989
```

Saisonnalité (seasonal) : La composante saisonnière révèle des variations récurrentes et régulières chaque année. Les pics et creux saisonniers sont constants en amplitude et en fréquence, ce qui suggère que certains mois ou saisons de l'année sont systématiquement associés à des augmentations ou des diminutions de la production électrique.

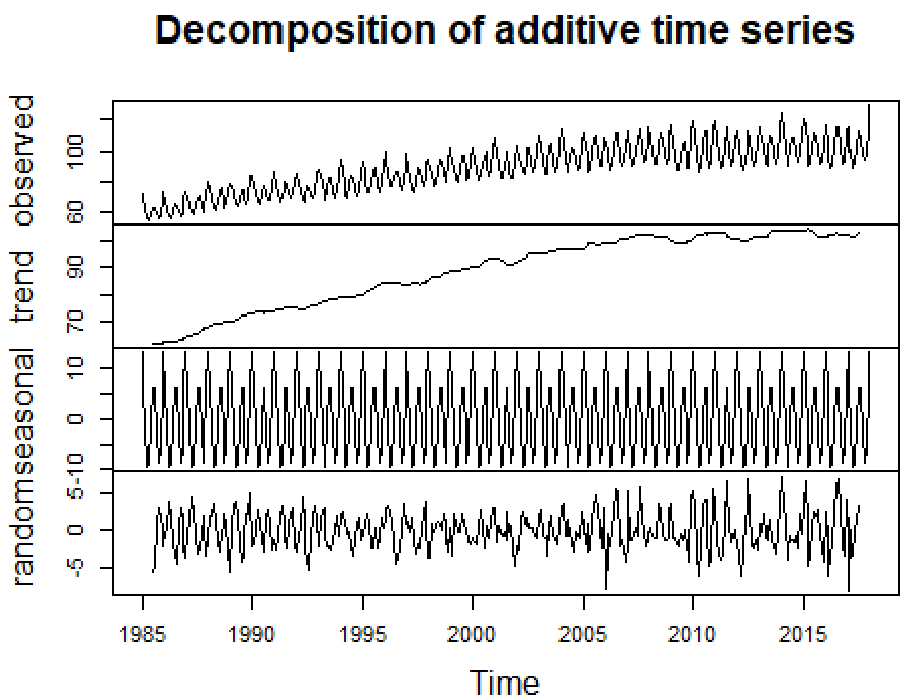
Résidus

```
# Afficher les 50 premières lignes des residus
random_head <- head(base_decom$random, 50)
random_head

            Jan         Feb         Mar         Apr         May         Jun
1985          NA          NA          NA          NA          NA          NA
1986 -2.18385184 -0.21679311  0.18812551  3.91557173  2.33359373 -1.66509728
1987 -3.58282684 -0.22978478  1.19414634  4.47903423  2.68891873 -0.57828895
1988 -1.39343101  1.95797772  1.94789217  3.61548423  1.29623123 -1.30134311
1989 -5.64952684  0.56850689
            Jul         Aug         Sep         Oct         Nov         Dec
1985 -5.72183472 -5.14644728  1.03331561  3.19741639  2.06619426  0.31637785
1986 -3.10863472 -4.63166395  0.38189061  3.04771639  1.78567342 -2.08203882
1987 -3.26618472 -2.70659311 -0.38540522  1.96047889  0.99819009 -3.49565548
1988 -2.79645972 -0.08435145  0.23848228  2.20662472  1.04439009 -2.71981382
1989
```

Résidus (random) : La composante résiduelle représente les fluctuations qui ne peuvent être expliquées ni par la tendance ni par la saisonnalité. Ces résidus semblent assez aléatoires, avec quelques variations importantes, ce qui peut indiquer des événements imprévus ou des perturbations temporaires dans la production électrique (par exemple, des pannes, des interventions de maintenance imprévues, ou des variations soudaines de la demande).

Representation des composantes de la serie

```
plot(base_decom)
```



Ajustement de la tendance en utilisant un modèle linéaire simple et multiple

Ajustement d'un modèle linéaire simple

Nous allons ajuster un modèle linéaire simple où la production dépend uniquement du temps:

Créons un dataframe avec les dates en tant que variable explicative

```
base_temps <- data.frame(   Time = as.numeric(time(mabase_st)),   Production =
as.numeric(mabase_st))

#Affichons les 10 premiers valeurs
base_temps_head <- head(base_temps, 10)
base_temps_head

       Time Production
1  1985.000    72.5052
2  1985.083    70.6720
3  1985.167    62.4502
4  1985.250    57.4714
5  1985.333    55.3151
6  1985.417    58.0904
7  1985.500    62.6202
8  1985.583    63.2485
9  1985.667    60.5846
10 1985.750    56.3154
```

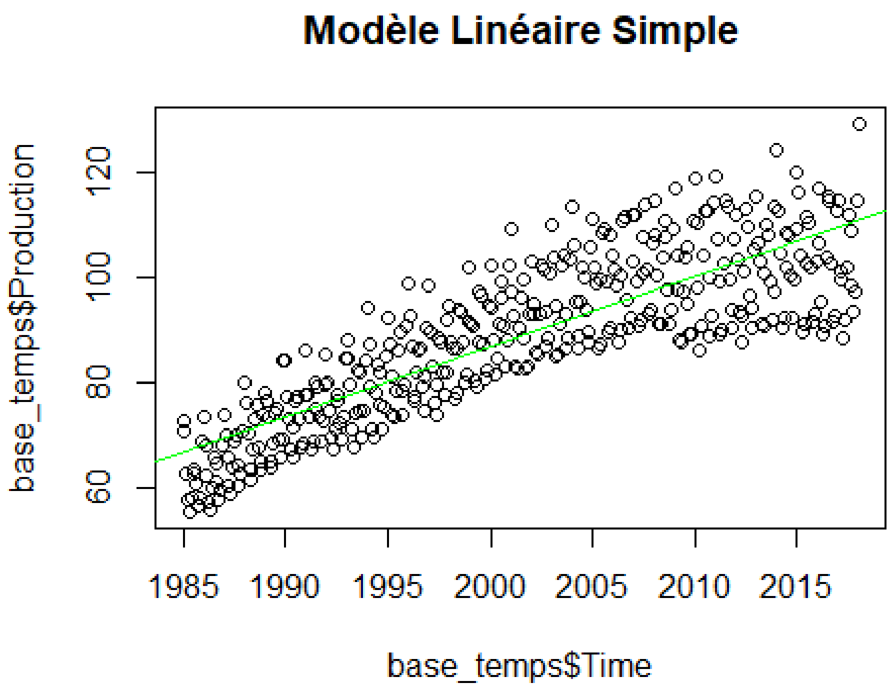
Ajustons maintenant le modèle linéaire simple

```
model_lineaire_simple <- lm(Production ~ Time, data = base_temps)
model_lineaire_simple


Call:
lm(formula = Production ~ Time, data = base_temps)

Coefficients:
(Intercept)        Time
  -2595.398       1.341

# Résumé du modèle
summary(model_lineaire_simple)


Call:
lm(formula = Production ~ Time, data = base_temps)

Residuals:
    Min       1Q   Median       3Q      Max
-21.6168  -6.3153  -0.4795   5.5767  21.5226

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.595e+03  8.957e+01  -28.98   <2e-16 ***
Time         1.341e+00  4.475e-02   29.97   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.515 on 395 degrees of freedom
Multiple R-squared:  0.6946,    Adjusted R-squared:  0.6938
F-statistic: 898.2 on 1 and 395 DF,  p-value: < 2.2e-16
```

Le modèle de régression linéaire indique que la production varie avec le temps. On a: **Production=−2595.398+1.341×Time**. Les résidus montrent une distribution centrée autour de zéro, suggérant un bon ajustement du modèle aux données. Avec un R-carré de 0.6946, environ 69,46 % de la variabilité de la production est expliquée par le temps, ce qui rend le modèle statistiquement significatif ($p < 2.2e-16$).

Visualisons la tendance ajustée pour voir

```
plot(base_temps$Time, base_temps$Production, main = "Modèle Linéaire Simple")
# Superposons la ligne de modèle linéaire
abline(model_lineaire_simple, col = "green")
```



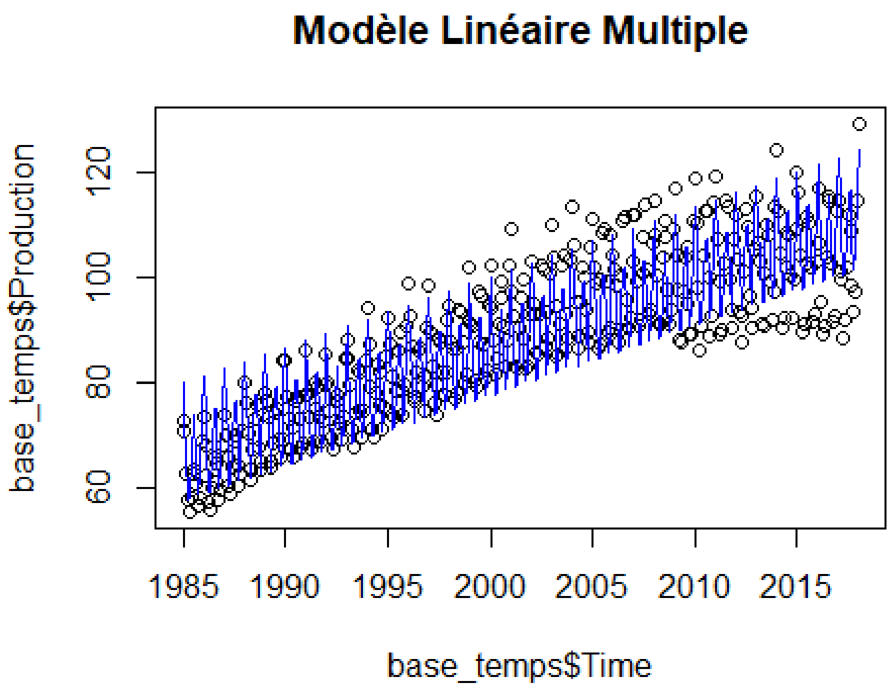
Ajustement de la Tendance avec un Modèle Linéaire Multiple

Pour un modèle linéaire multiple, on peut ajouter des termes saisonniers pour mieux capter les variations saisonnières.

On va ajouter des termes saisonniers (par exemple, des indicateurs pour les mois)

Ajoutons des termes saisonniers (par exemple, des indicateurs pour les mois)

```
base_temps$Month <- factor(cycle(mabase_st))
# Ajustons le modèle linéaire multiple
model_lineaire_multiple <- lm(Production ~ Time + Month, data = base_temps)
# Visualisualisation de la tendance ajustée
pred_values <- predict(model_lineaire_multiple, newdata=base_temps)
plot(base_temps$Time, base_temps$Production, main="Modèle Linéaire Multiple")
lines(base_temps$Time, pred_values, col="blue")
```



Comparaison des Modèles

Comparons les deux modèles en termes de performance, par exemple avec le R² ou l’AIC.

Les mesures R² et AIC fournissent des indications cruciales sur la performance et la pertinence des modèles linéaires dans l’analyse des données.

```
# Récupérons les valeurs d'ajustement
cds <- summary(model_lineaire_simple)$r.squared
cdm <- summary(model_lineaire_multiple)$r.squared
aic_simple <- AIC(model_lineaire_simple)
aic_multiple <- AIC(model_lineaire_multiple)
# Affichons les résultats
cat("R² du modèle linéaire simple: ", cds, "\n")

R² du modèle linéaire simple:  0.6945572

cat("R² du modèle linéaire multiple: ", cdm, "\n")

R² du modèle linéaire multiple:  0.9165092

cat("AIC du modèle linéaire simple: ", aic_simple, "\n")

AIC du modèle linéaire simple:  2831.257

cat("AIC du modèle linéaire multiple: ", aic_multiple, "\n")

AIC du modèle linéaire multiple:  2338.338
```

Un R² de 0.6945572 pour le modèle linéaire simple indique que près de 70% de la variance de la variable dépendante est expliquée par la variable indépendante utilisée. En revanche, le modèle linéaire multiple affiche un R² plus élevé à 0.9165092, suggérant qu'environ 92% de la variance est expliquée par toutes les variables indépendantes incluses. Le modèle linéaire multiple montre un score significativement plus bas à 2338.338 par rapport au modèle simple à 2831.257, indiquant ainsi une meilleure adéquation du modèle multiple à vos données observées.

2) Faisons une étude de la saisonnalité et analysons les résultats.

L'étude de la saisonnalité nous permet de comprendre les fluctuations récurrentes de notre série temporelle, tandis que l'analyse des résultats nous aide à interpréter les effets de la tendance et des variations saisonnières sur nos données. Utilisez ces informations pour prendre des décisions éclairées dans votre analyse de série temporelle.

Saisonnalité

```
# Afficher les 50 premières lignes de la saisonnalite
seasonal_head <- head(base_decom$seasonal,50)
seasonal_head

            Jan        Feb        Mar        Apr        May        Jun
1985 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1986 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1987 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1988 13.2130393  5.7447473 -0.4936797 -9.4783551 -9.2221604 -1.1684236
1989 13.2130393  5.7447473
            Jul        Aug        Sep        Oct        Nov        Dec
1985  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
1986  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
1987  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
1988  6.1430139  6.2744514 -2.4478281 -8.8533539 -6.0395359  6.3280847
1989
```

Les données de la composante saisonnière révèlent des variations distinctes mois par mois de la production électrique sur une période étendue de 1985 à 2018. Les résultats montrent les effets saisonniers constants sur une série temporelle étalée sur plusieurs années. Chaque mois, de janvier à décembre, présente des valeurs constantes au fil des années incluses dans nos données. Ces constantes indiquent l'impact moyen que chaque mois a sur la série temporelle par rapport à la tendance globale. Par exemple, janvier présente une valeur de 13.2130393, suggérant une augmentation constante chaque année à cette période.

Cette régularité dans les schémas saisonniers implique une prévisibilité relative dans la demande énergétique tout au long de l'année et Cette compréhension est cruciale pour la planification stratégique des capacités de production, l'optimisation des ressources et la réponse efficace aux fluctuations saisonnières de la demande, assurant ainsi une gestion stable et adaptative du réseau électrique.

En résumé, ces résultats fournissent une base solide pour comprendre et gérer les variations saisonnières dans nos données, améliorant ainsi la précision des prévisions et la capacité à réagir efficacement aux changements saisonniers.

3) Modéliser la série par la méthodologie de Box & Jenkins. Analyser et interpréter les sorties du logiciel R.

Modelisation Box & Jenkins

Pour modéliser la série temporelle par la méthodologie de Box & Jenkins, nous allons suivre les étapes suivantes dans R :

Tests de stationnarite

TEST AUGMENTE DE DICKEY-FULLER

Le test augmente de Dickey-FULLER (ADF) est utilisé pour vérifier la présence d'une racine unitaire dans une série temporelle, ce qui indique si la série est stationnaire ou non.

```
# H0: non stationnarite
adf.test(mabase_st)

Warning in adf.test(mabase_st): p-value smaller than printed p-value


   Augmented Dickey-Fuller Test

data:  mabase_st
Dickey-Fuller = -5.139, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

La p-value obtenue (0.01) est inférieure au seuil de signification typique de 0.05, ce qui signifie que nous rejetons l'hypothèse nulle selon laquelle la série a une racine unitaire. Par conséquent, l'hypothèse alternative selon laquelle la série est stationnaire est acceptée.

En d'autres termes, les résultats de ce test indiquent que notre série temporelle est stationnaire.

TEST PP

Le test de Phillips-Perron (PP) est également utilisé pour vérifier la présence d'une racine unitaire dans une série temporelle, similaire au test ADF.

```
# H0: non stationnarite
pp.test(mabase_st)

Warning in pp.test(mabase_st): p-value smaller than printed p-value


   Phillips-Perron Unit Root Test
```

```
data:  mabase_st
Dickey-Fuller Z(alpha) = -88.143, Truncation lag parameter = 5, p-value
= 0.01
alternative hypothesis: stationary
```

La p-value (0.01) est encore une fois inférieure au seuil de signification typique de 0.05, ce qui nous permet de rejeter l’hypothèse nulle de présence d’une racine unitaire. Par conséquent, nous acceptons l’hypothèse alternative selon laquelle la série est stationnaire. Les résultats de ce test confirment les conclusions du test ADF, indiquant que la série temporelle est stationnaire.

TEST KPSS

Le test KPSS (Kwiatkowski-Phillips-Schmidt-Shin) est utilisé pour tester l’hypothèse nulle selon laquelle une série temporelle est stationnaire autour d’une moyenne ou d’une tendance, contre l’hypothèse alternative selon laquelle elle est non stationnaire.

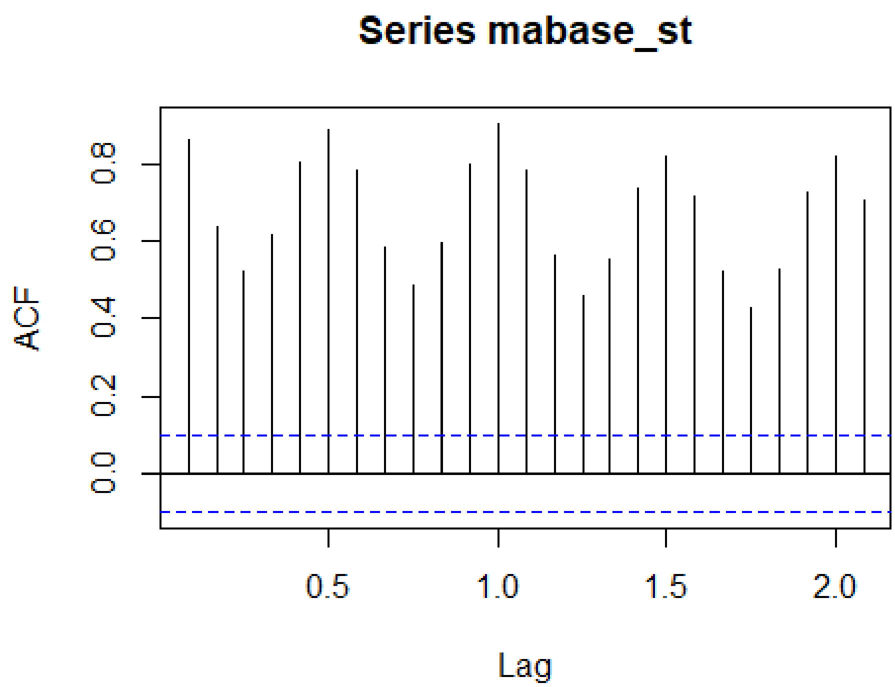
```
# H0: Stationnarite
kpss.test(mabase_st)

Warning in kpss.test(mabase_st): p-value smaller than printed p-value


   KPSS Test for Level Stationarity

data:  mabase_st
KPSS Level = 6.3058, Truncation lag parameter = 5, p-value = 0.01
```

La p-value (0.01) est inférieure au seuil de signification typique de 0.05, ce qui signifie que nous rejetons l’hypothèse nulle selon laquelle la série est stationnaire. Contrairement aux tests ADF et PP, le test KPSS suggère que notre série temporelle n’est pas stationnaire.

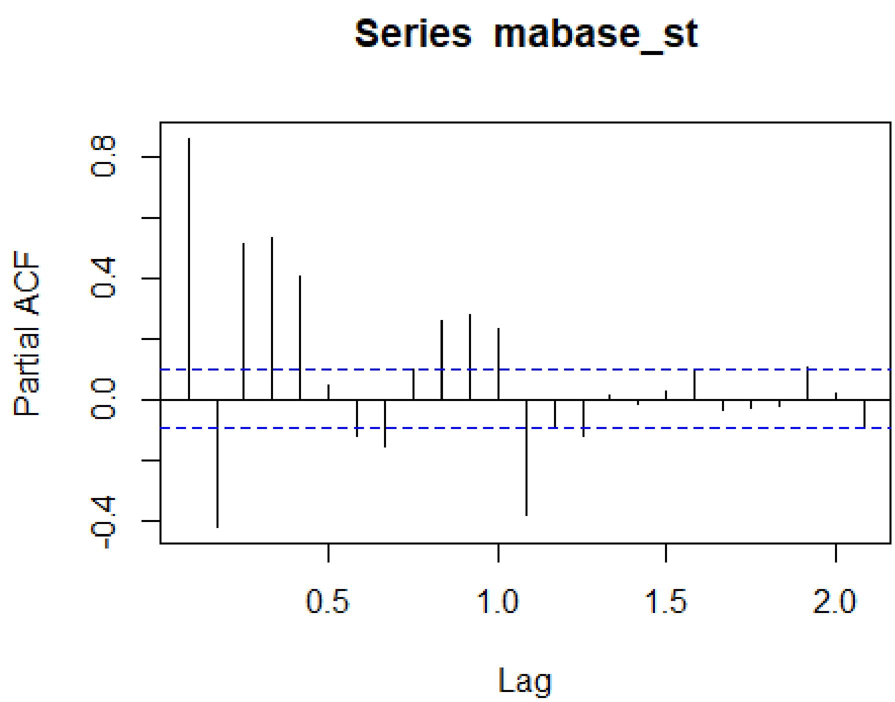
Identification du model

1. Pour identifier le modèle ARIMA approprié, nous devons analyser les autocorrélations (ACF) et les autocorrélations partielles (PACF) de la série.

```
# Tracé de l'ACF et de la PACF
acf(mabase_st)
```




```
pacf(mabase_st)
```



```
#Ensuite, nous utilisons auto.arima pour trouver les meilleurs paramètres (p, d, q)
automatiquement.
model <- auto.arima(mabase_st)
summary(model)

Series: mabase_st
ARIMA(2,1,1)(0,1,1)[12]

Coefficients:
        ar1      ar2      ma1     sma1
     0.5503  -0.0683  -0.9477  -0.7635
s.e.  0.0544   0.0549   0.0193   0.0331

sigma^2 = 5.838:  log likelihood = -888.05
AIC=1786.11   AICc=1786.27   BIC=1805.86

Training set error measures:
                     ME     RMSE      MAE        MPE     MAPE      MASE
Training set -0.1474299 2.363849 1.791855 -0.2115265 1.942956 0.6278541
                     ACF1
Training set -0.002715831

# Imprimer les détails du modèle
print(model)

Series: mabase_st
ARIMA(2,1,1)(0,1,1)[12]

Coefficients:
        ar1      ar2      ma1     sma1
     0.5503  -0.0683  -0.9477  -0.7635
s.e.  0.0544   0.0549   0.0193   0.0331

sigma^2 = 5.838:  log likelihood = -888.05
AIC=1786.11   AICc=1786.27   BIC=1805.86
```

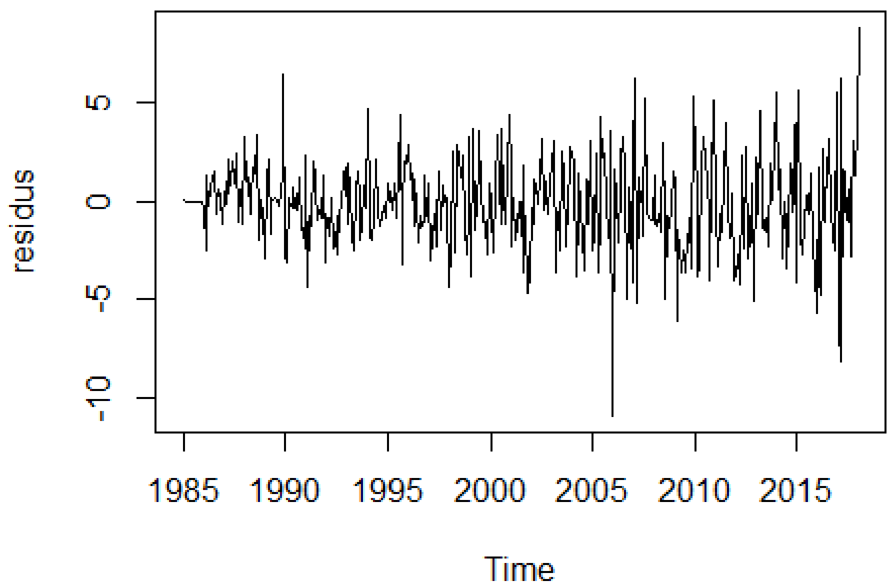
Le modèle ARIMA(2,1,1)(0,1,1)[12] est utilisé pour modéliser les données mensuelles. Les coefficients indiquent une influence significative des observations précédentes (ar1 = 0,5503) et une correction des erreurs de prévision passées (ma1 = -0,9477). Le terme saisonnier sma1 montre une correction saisonnière précédente. L'ajustement du modèle est raisonnable avec un sigma^2 de 5,838 et une log-vraisemblance de -888,05.

Les valeurs AIC, AICc, et BIC sont respectivement de 1786,11, 1786,27, et 1805,86, indiquant un ajustement acceptable du modèle mais laissant place à des améliorations potentielles.

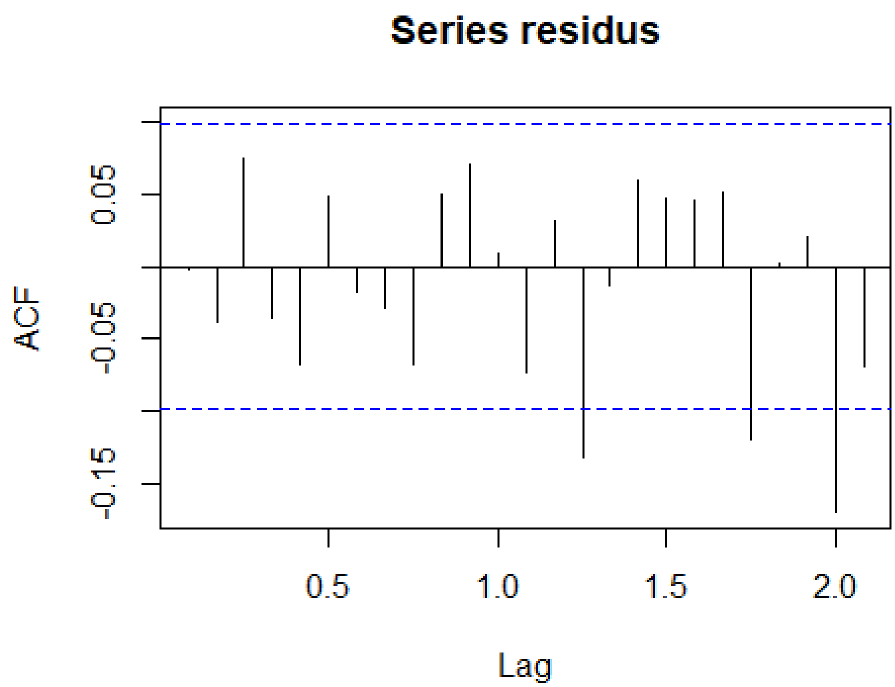
Diagnostic

Nous devons vérifier que les résidus du modèle sont un bruit blanc. Cela signifie que les résidus doivent être non corrélés et suivre une distribution normale.

```
# Résidus du modèle
residus <- residuals(model)

# Tracé des résidus
plot(residus)
```




```
# Tracé de l'ACF des résidus
acf(residus)
```




```
# Test de Ljung-Box pour l'indépendance des résidus
Box.test(residus, lag=20, type="Ljung-Box")


   Box-Ljung test

data:  residus
X-squared = 25.662, df = 20, p-value = 0.1773

# Test de normalité des résidus
shapiro.test(residus)


   Shapiro-Wilk normality test

data:  residus
W = 0.98648, p-value = 0.0009324
```

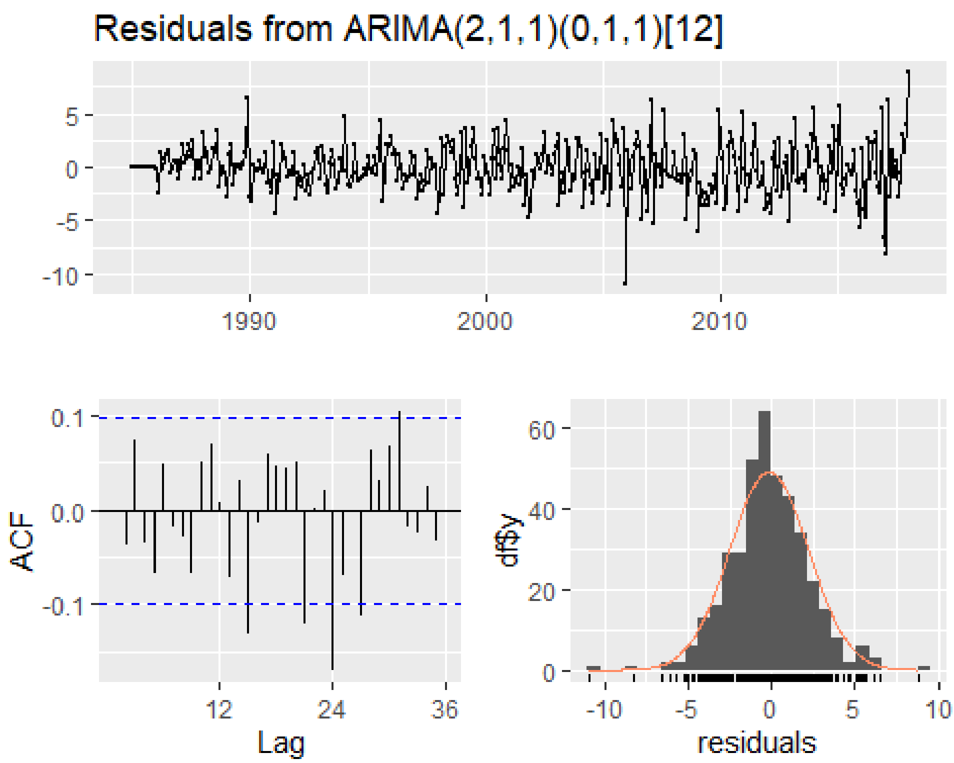
Le test de Box-Ljung indique une statistique X^2 de 25.662 avec 20 degrés de liberté et une valeur de p de 0.1773, suggérant qu'il n'y a pas suffisamment de preuves pour rejeter l'hypothèse nulle d'absence d'autocorrélation dans les résidus. Cela suggère que le modèle capture efficacement la structure temporelle des données.

Cependant, le test de normalité de Shapiro-Wilk montre un W de 0.98648 avec un p-value très faible de 0.0009324, indiquant que les résidus ne suivent pas une distribution normale. Cela pourrait suggérer que le modèle pourrait être amélioré pour mieux capturer la variabilité non expliquée par le modèle actuel.

Diagnostic des Résidus

Il est important de vérifier que les résidus du modèle sont proches d'un bruit blanc. Voici comment effectuer cette vérification.

```
# Plot des résidus
checkresiduals(model)
```




```
	Ljung-Box test

data:  Residuals from ARIMA(2,1,1)(0,1,1)[12]
Q* = 43.986, df = 20, p-value = 0.001511

Model df: 4.   Total lags used: 24

# Test de Ljung-Box
Box.test(residuals(model), lag = 24, type = "Ljung-Box")


	Box-Ljung test
```

```
data:  residuals(model)
X-squared = 43.986, df = 24, p-value = 0.007658
```

- On a Q* est de 43.986 avec 20 degrés de liberté et un p-value de 0.001511. Cequi indique une forte autocorrélation significative dans les résidus jusqu'au lag 20.
- On a X-squared est de 43.986 avec 24 degrés de liberté et un p-value de 0.007658. Ce résultat confirme également une autocorrélation significative dans les résidus.

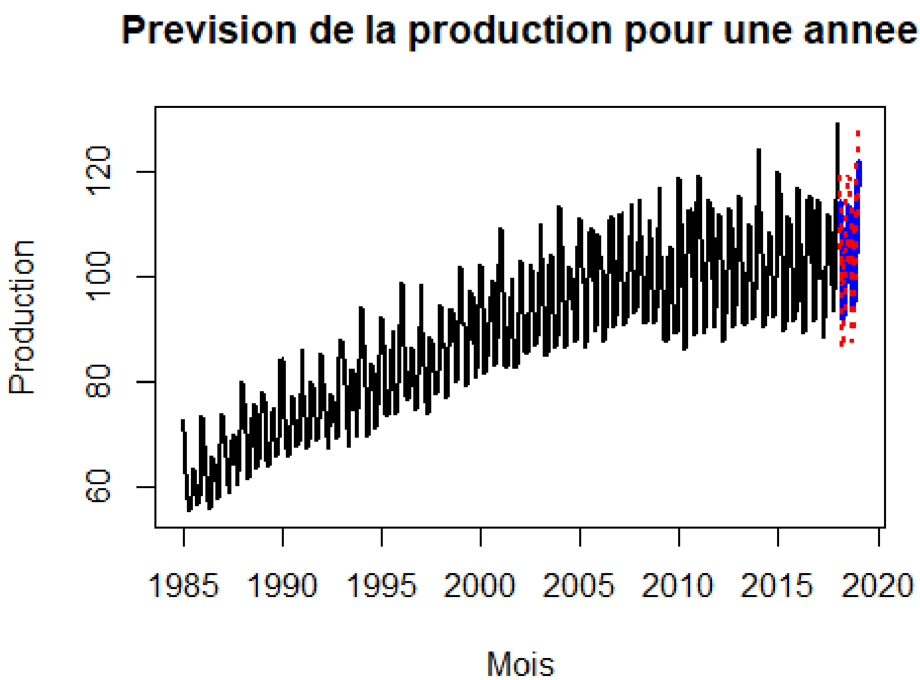
Le modèle ARIMA a un bon ajustement global, ces tests indiquent que les résidus présentent encore une autocorrélation non négligeable, suggérant des possibilités d'amélioration du modèle pour mieux expliquer la variabilité résiduelle des données.

4) Effectuer une prévision par ARIMA de la production électrique pour une période d'une année et une période de 3 années. Représenter graphiquement ces prévisions dans deux graphes différents avec leurs intervalles de confiance. Commenter.

prevision pour 1 an par arima

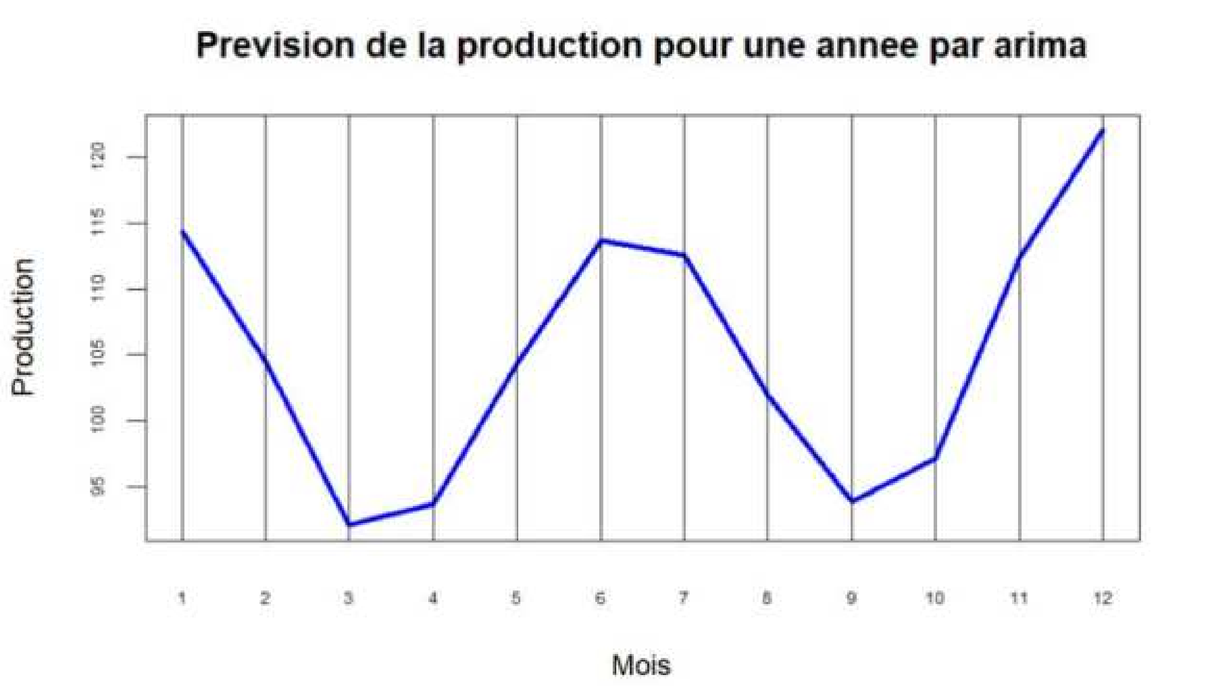
```
prev=predict(model,n.ahead=12)
prev

$pred
          Jan       Feb       Mar       Apr       May       Jun       Jul
2018           114.31111 104.45857  92.09910  93.63140 104.31821 113.65996
2019 122.04284
          Aug       Sep       Oct       Nov       Dec
2018 112.58325 101.93541  93.85642  97.12217 112.41629
2019

$se
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
2018          2.416146 2.821027 2.922366 2.956342 2.973647 2.986085 2.996905
2019 3.046708
          Sep      Oct      Nov      Dec
2018 3.007142 3.017149 3.027054 3.036902
2019

prev_arima2018=prev$pred
prev_arima2018

          Jan       Feb       Mar       Apr       May       Jun       Jul
2018           114.31111 104.45857  92.09910  93.63140 104.31821 113.65996
2019 122.04284
          Aug       Sep       Oct       Nov       Dec
```

```
2018 112.58325 101.93541  93.85642  97.12217 112.41629
2019

plot(mabase_st,xlim=c(1985,2019),ylim=range(c(mabase_st,prev_arima2018)),
    xlab='Mois',ylab='Production',main='Prevision de la production pour une annee',lwd=2)
lines(prev_arima2018,col="blue",lwd=3)
lines(prev_arima2018+2*prev$se,col="red",lty=3,lwd=2)
lines(prev_arima2018-2*prev$se,col="red",lty=3,lwd=2)
```



Graphe des previsions

```
mois=1:12

plot(mois,prev_arima2018, type='l', col='blue', lwd=3,
    cex.axis=0.6, xaxt='n', xlab='Mois',
    ylab='Production', main='Prevision de la production pour une annee par arima')

axis(1, 1:12, mois[1:12],tck=T,cex.axis=.6)
```



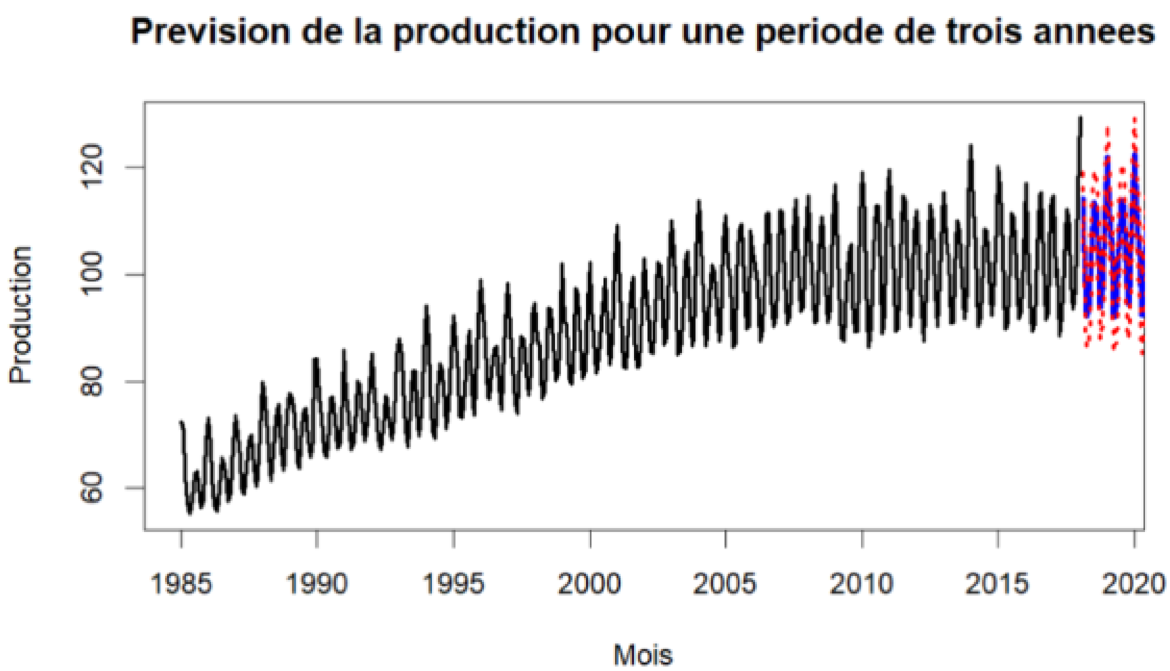
Graphe avec intervalles de confiance

prevision pour 3 ans par Arima

```
prev3an=predict(model,n.ahead=36)
prev_arima3ans=prev3an$pred
prev_arima3ans

          Jan       Feb       Mar       Apr       May       Jun       Jul
2018           114.31111 104.45857  92.09910  93.63140 104.31821 113.65996
2019 122.04284 110.69451 103.24842  91.95762  93.91376 104.76087 114.16190
2020 122.57875 111.23047 103.78440  92.49360  94.44975 105.29685 114.69789
2021 123.11473
          Aug       Sep       Oct       Nov       Dec
2018 112.58325 101.93541  93.85642  97.12217 112.41629
2019 113.10687 102.46692  94.39078  97.65757 112.95206
2020 113.64286 103.00290  94.92677  98.19356 113.48805
2021
```

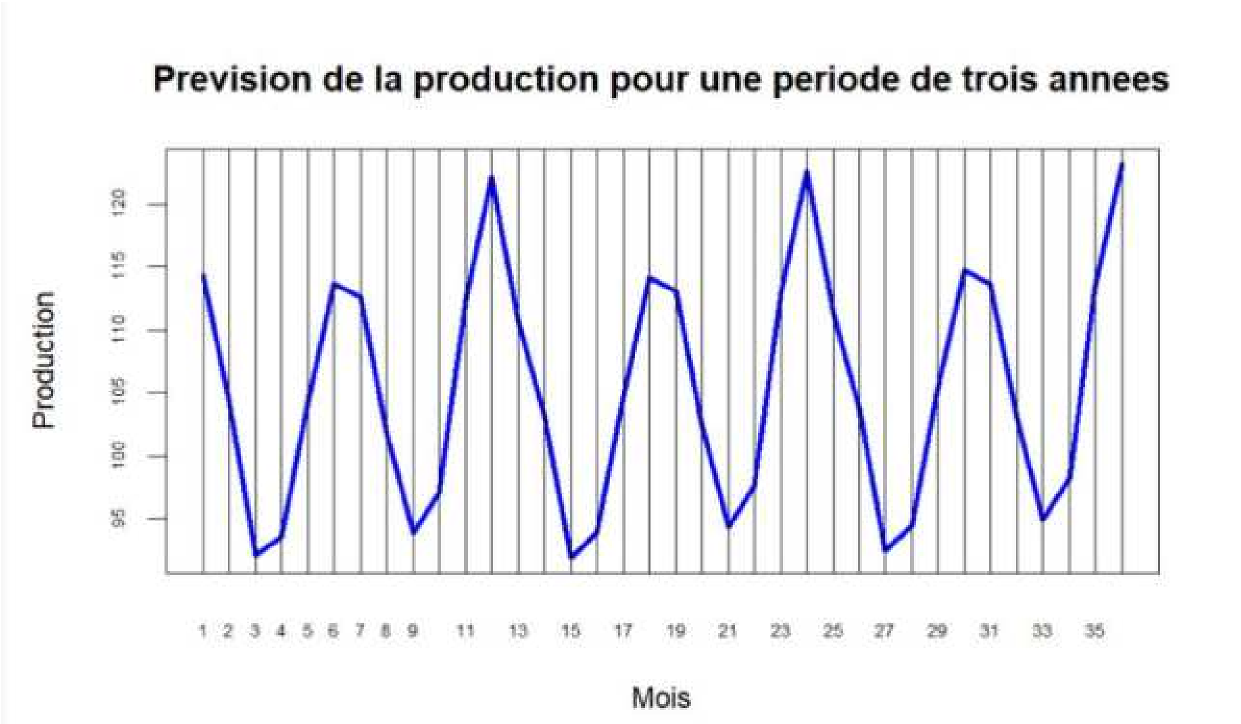
Prévisions basées sur un modèle ARIMA pour les années à venir (3 ans), couvrant janvier 2018 à décembre 2021.

```
plot(mabase_st,xlim=c(1985,2019),ylim=range(c(mabase_st,prev_arima3ans)),
     xlab='Mois',ylab='Production',main='Prevision de la production pour une periode de trois annees',lwd=2)
lines(prev_arima3ans,col="blue",lwd=3)
lines(prev_arima3ans+2*prev3an$se,col="red",lty=3,lwd=2)
lines(prev_arima3ans-2*prev3an$se,col="red",lty=3,lwd=2)
```



Graphe des previsions

```
mois=1:36

plot(mois,prev_arima3ans, type='l', col='blue', lwd=3,
     cex.axis=0.6, xaxt='n', xlab='Mois',
     ylab='Production', main='Prevision de la production pour une periode de trois annees')

axis(1, 1:36, mois[1:36],tck=T,cex.axis=.6)
```



Graphe avec intervalles de confiance

5) Effectuer une prévision pour une année et pour 3 années par lissage exponentiel et comparer graphiquement ces valeurs avec celles obtenues par la méthode ARIMA.

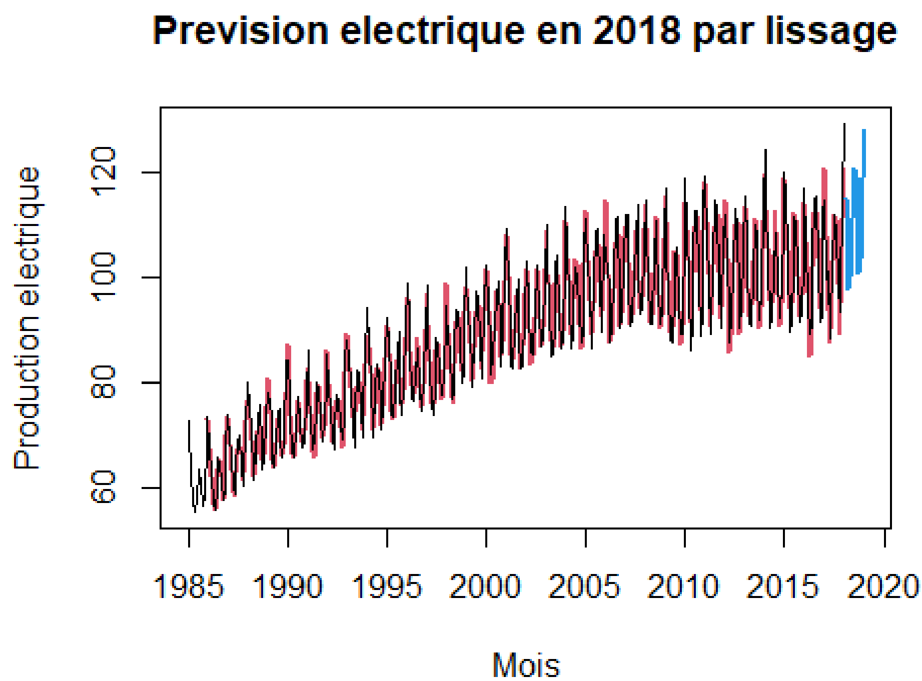
Lissage Exponentiel pour un an

```
lissage=HoltWinters(mabase_st)
prev_lis2018=predict(lissage,n.ahead=12)
prev_lis2018

       Jan       Feb       Mar       Apr       May       Jun       Jul
2018           115.12458 109.15159  97.69496 100.10703 111.28663 120.85122
2019 128.16038
       Aug       Sep       Oct       Nov       Dec
2018 119.52513 108.92212 100.62997 103.68895 118.96908
2019
```

Ici on a les prévisions par lissage pour les mois de janvier 2018 et janvier à septembre 2019.

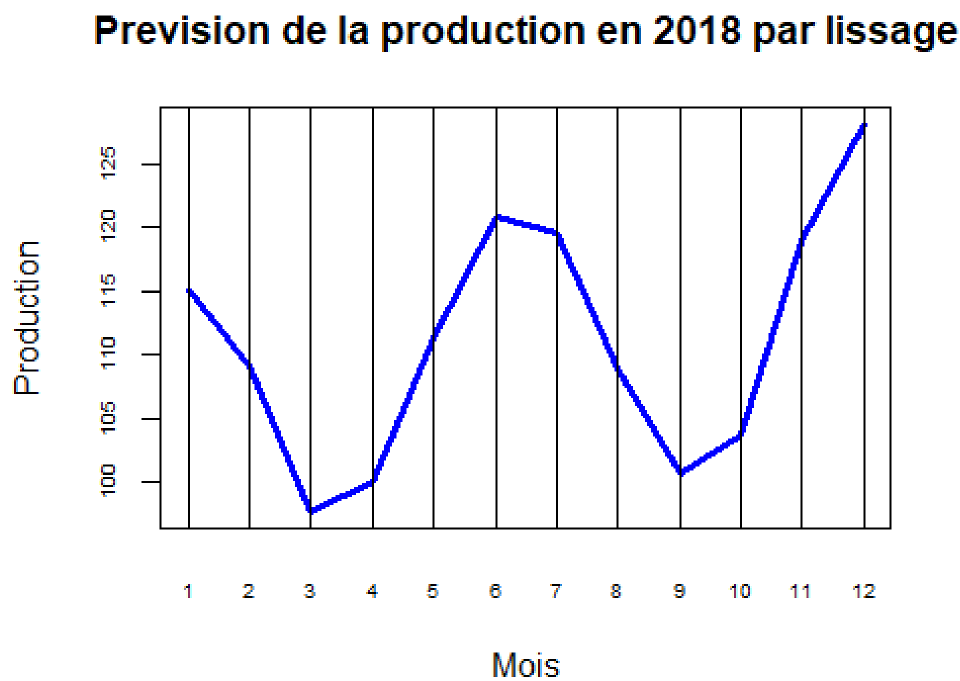
Representation graphique

```
plot(lissage, xlim=c(1985,2019), xlab='Mois',
    ylab='Production electrique', main='Prevision electrique en 2018 par lissage',lwd=2)

lines(prev_lis2018,col=4,lwd=2)
```



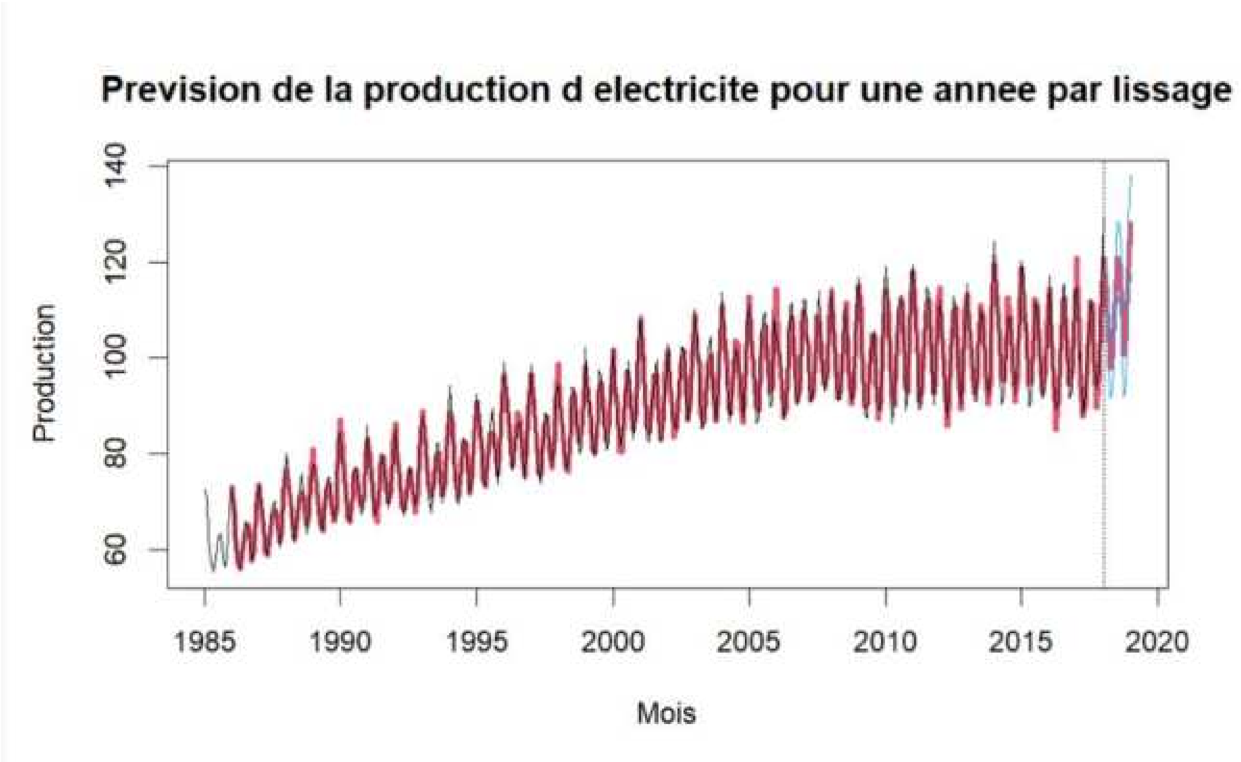
Graphe des previsions

```
mois=1:12

plot(mois,prev_lis2018, type='l', col='blue', lwd=3,
     cex.axis=0.6, xaxt='n', xlab='Mois',
     ylab='Production', main='Prevision de la production en 2018 par lissage')

axis(1, 1:12, mois[1:12],tck=T,cex.axis=.6)
```



Graphe avec intervalles de confiance

```
lissage_ic=predict(lissage,n.ahead=12, prediction.interval=TRUE)
lissage_ic

               fit      upr       lwr
Feb 2018 115.12458 120.1664 110.08274
Mar 2018 109.15159 114.8146 103.48856
Apr 2018  97.69496 103.9175  91.47244
May 2018 100.10703 106.8427  93.37135
Jun 2018 111.28663 118.4991 104.07420
Jul 2018 120.85122 128.5108 113.19165
Aug 2018 119.52513 127.6071 111.44313
Sep 2018 108.92212 117.4055 100.43869
Oct 2018 100.62997 109.4967  91.76327
Nov 2018 103.68895 112.9230  94.45487
Dec 2018 118.96908 128.5565 109.38170
Jan 2019 128.16038 138.0885 118.23225

plot(lissage,lissage_ic, xlim=c(1985,2019),lwd=3,xlab='Mois',
     ylab='Production',main='Prevision de la production d electricite pour une annee par lissage')
```



On a ici les prévisions par lissage avec les intervalles de prédiction pour chaque mois à partir de février 2018 jusqu'à janvier 2019.

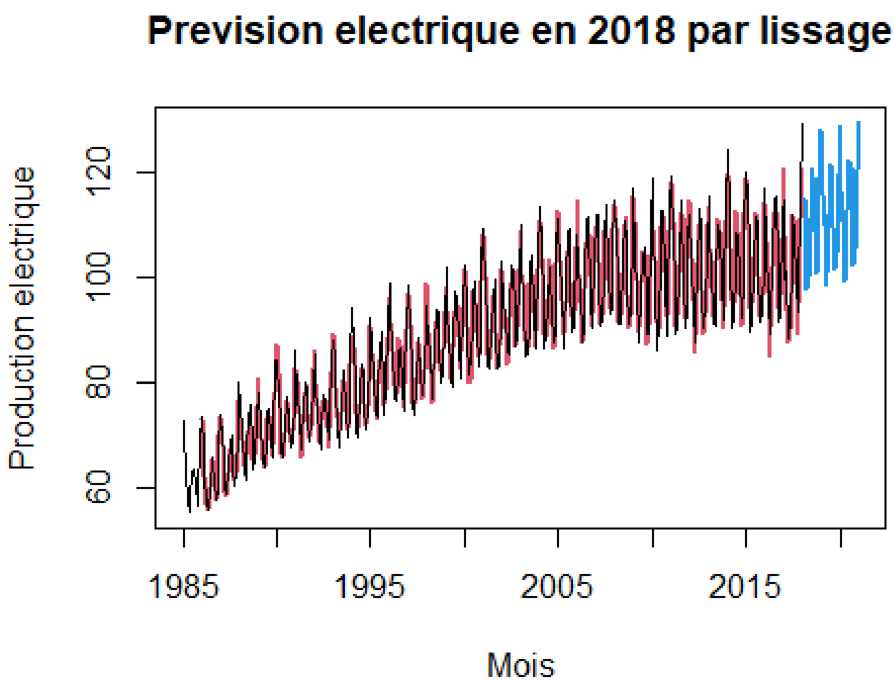
Lissage Exponentiel pour trois ans

```
lissage3ans=HoltWinters(mabase_st)
prev_lis3ans=predict(lissage,n.ahead=36)
prev_lis3ans

           Jan       Feb       Mar       Apr       May       Jun       Jul
2018           115.12458 109.15159  97.69496 100.10703 111.28663 120.85122
2019 128.16038 115.95284 109.97986  98.52323 100.93530 112.11490 121.67948
2020 128.98864 116.78111 110.80812  99.35149 101.76356 112.94316 122.50775
2021 129.81691
           Aug       Sep       Oct       Nov       Dec
2018 119.52513 108.92212 100.62997 103.68895 118.96908
2019 120.35340 109.75038 101.45824 104.51721 119.79734
2020 121.18166 110.57865 102.28650 105.34548 120.62561
2021
```

Representation graphique

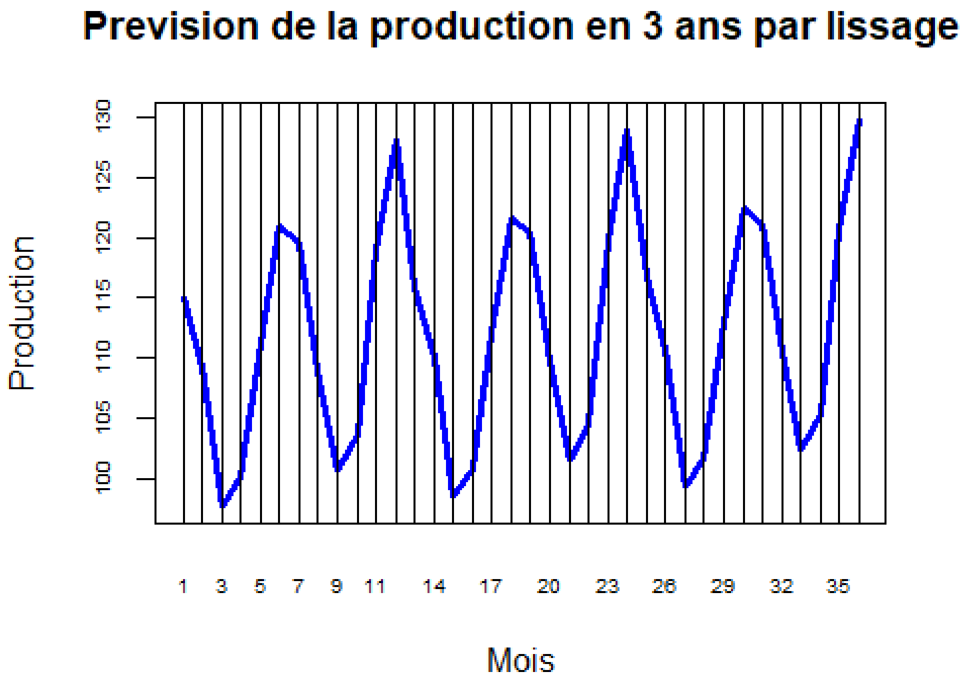
```
plot(lissage3ans, xlim=c(1985,2021), xlab='Mois',
     ylab='Production electrique', main='Prevision electrique en 2018 par lissage',lwd=2)
```

```
lines(prev_lis3ans,col=4,lwd=2)
```



Graphe des previsions

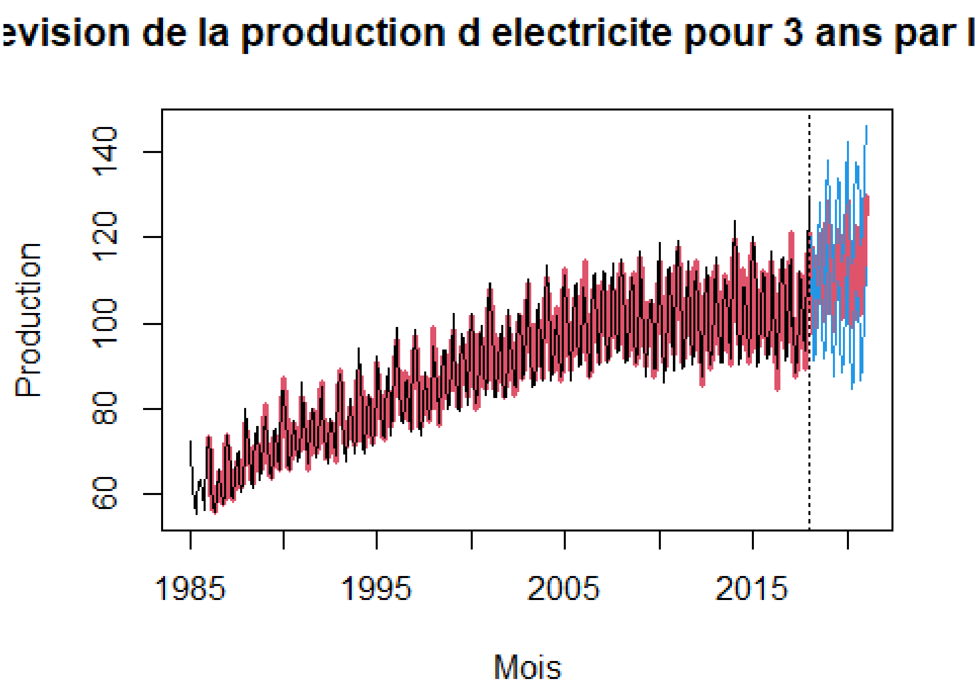
```
mois=1:36

plot(mois,prev_lis3ans, type='l', col='blue', lwd=3,
     cex.axis=0.6, xaxt='n', xlab='Mois',
     ylab='Production', main='Prevision de la production en 3 ans par lissage')

axis(1, 1:36, mois[1:36],tck=T,cex.axis=.6)
```



Graphe avec intervalles de confiance

```
lissage3ans_ic=predict(lissage3ans,n.ahead=36, prediction.interval=TRUE)
lissage3ans_ic

              fit      upr       lwr
Feb 2018 115.12458 120.1664 110.08274
Mar 2018 109.15159 114.8146 103.48856
Apr 2018  97.69496 103.9175  91.47244
May 2018 100.10703 106.8427  93.37135
Jun 2018 111.28663 118.4991 104.07420
Jul 2018 120.85122 128.5108 113.19165
Aug 2018 119.52513 127.6071 111.44313
Sep 2018 108.92212 117.4055 100.43869
Oct 2018 100.62997 109.4967  91.76327
Nov 2018 103.68895 112.9230  94.45487
Dec 2018 118.96908 128.5565 109.38170
Jan 2019 128.16038 138.0885 118.23225
Feb 2019 115.95284 126.5816 105.32413
Mar 2019 109.97986 120.9169  99.04279
Apr 2019  98.52323 109.7602  87.28627
May 2019 100.93530 112.4643  89.40625
Jun 2019 112.11490 123.9288 100.30098
Jul 2019 121.67948 133.7716 109.58740
Aug 2019 120.35340 132.7174 107.98941
Sep 2019 109.75038 122.3804  97.12034
```

```
Oct 2019 101.45824 114.3488  88.56763
Nov 2019 104.51721 117.6632  91.37120
Dec 2019 119.79734 133.1939 106.40080
Jan 2020 128.98864 142.6311 115.34617
Feb 2020 116.78111 130.9416 102.62063
Mar 2020 110.80812 125.2015  96.41476
Apr 2020  99.35149 113.9740  84.72895
May 2020 101.76356 116.6117  86.91539
Jun 2020 112.94316 128.0136  97.87272
Jul 2020 122.50775 137.7972 107.21828
Aug 2020 121.18166 136.6871 105.67625
Sep 2020 110.57865 126.2970  94.86027
Oct 2020 102.28650 118.2150  86.35800
Nov 2020 105.34548 121.4814  89.20958
Dec 2020 120.62561 136.9663 104.28496
Jan 2021 129.81691 146.3598 113.27404

plot(lissage3ans,lissage3ans_ic, xlim=c(1985,2021),lwd=3,xlab='Mois',
     ylab='Production',main='Prevision de la production d electricite pour 3 ans par lissage')
```



6) validation

Par apprentissage-test

Division de la base

```
# Nombre total de points de données
nombval <- length(mabase_st)
# Calculer l'index de division pour 85% des données
nombval_train <- floor(0.85 * nombval)
nombval_train

[1] 337
```

Diviser les données en base d'apprentissage

```
base_appr <- mabase_st[1:nombval_train]
# Affichons seulement les 50 premiers valeurs
base_appr_head <- head(base_appr, 50)
base_appr_head

 [1] 72.5052 70.6720 62.4502 57.4714 55.3151 58.0904 62.6202 63.2485 60.5846
[10] 56.3154 58.0005 68.7145 73.3057 67.9869 62.2221 57.0329 55.8137 59.9005
[19] 65.7655 64.4816 61.0005 57.5322 59.3417 68.1354 73.8152 70.0620 65.6100
[28] 60.1586 58.8734 63.8918 68.8694 70.0669 64.1151 60.3789 62.4643 70.5777
[37] 79.8703 76.1622 70.2928 63.2384 61.4065 67.1097 72.9816 75.7655 67.5152
[46] 63.2832 65.1078 73.8631 77.9188 76.6822
```

Diviser les données en base de test

```
base_test <- mabase_st[(nombval_train + 1):nombval]
# Affichons seulement les 50 premiers valeurs
base_test_head <- head(base_test, 50)
base_test_head

 [1] 106.7340 102.9948  91.0092  90.9634 100.6957 110.1480 108.1756  99.2809
 [9]  91.7871  97.2853 113.4732 124.2549 112.8811 104.7631  90.2867  92.1340
[17] 101.8780 108.5497 108.1940 100.4172  92.3837  99.7033 109.3477 120.2696
[25] 116.3788 104.4706  89.7461  91.0930 102.6495 111.6354 110.5925 101.9204
[33]  91.5959  93.0628 103.2203 117.0837 106.6688  95.3548  89.3254  90.7369
[41] 104.0375 114.5397 115.5159 102.7637  91.4867  92.8900 112.7694 114.8505
[49]  99.4901 101.0396
```

Ajuster le Modèle ARIMA sur la Base d'Apprentissage

```
# Ajuster le modèle ARIMA sur les données d'apprentissage
base_appr_arima <- auto.arima(base_appr)
summary(base_appr_arima)

Series: base_appr
ARIMA(2,1,2) with drift
```

```
Coefficients:
         ar1      ar2      ma1     ma2   drift
      0.9692  -0.8626  -1.0465  0.1415  0.1221
s.e.  0.0313   0.0289   0.0523  0.0498  0.0219

sigma^2 = 13.67:  log likelihood = -915.77
AIC=1843.55   AICc=1843.8   BIC=1866.45

Training set error measures:
                     ME     RMSE      MAE        MPE     MAPE      MASE
Training set -0.04635035 3.664873 2.822719 -0.1751415 3.268739 0.4494264
                    ACF1
Training set -0.02580367
```

Le modèle ARIMA(2,1,2) avec drift ajuste la série avec deux termes auto-régressifs, deux termes de moyenne mobile, et une dérive. Les coefficients AR1 et AR2 sont significatifs, montrant une forte corrélation avec les valeurs précédentes, tandis que MA1 et MA2 capturent la relation avec les erreurs passées. La dérive de 0.1221 indique une légère tendance positive. La variance résiduelle est de 13.67, et le log-likelihood est de -915.77. Les critères d'information AIC, AICc, et BIC sont 1843.55, 1843.8, et 1866.45 respectivement. Les mesures d'erreur sur le jeu d'apprentissage montrent un RMSE de 3.6649 et un MAPE de 3.2687%, indiquant une erreur de prévision acceptable. L'ACF1 des résidus est faible (-0.0258), suggérant que le modèle capture bien la structure des données.

Effectuons des Prévisions sur la Période de Test pour une annee

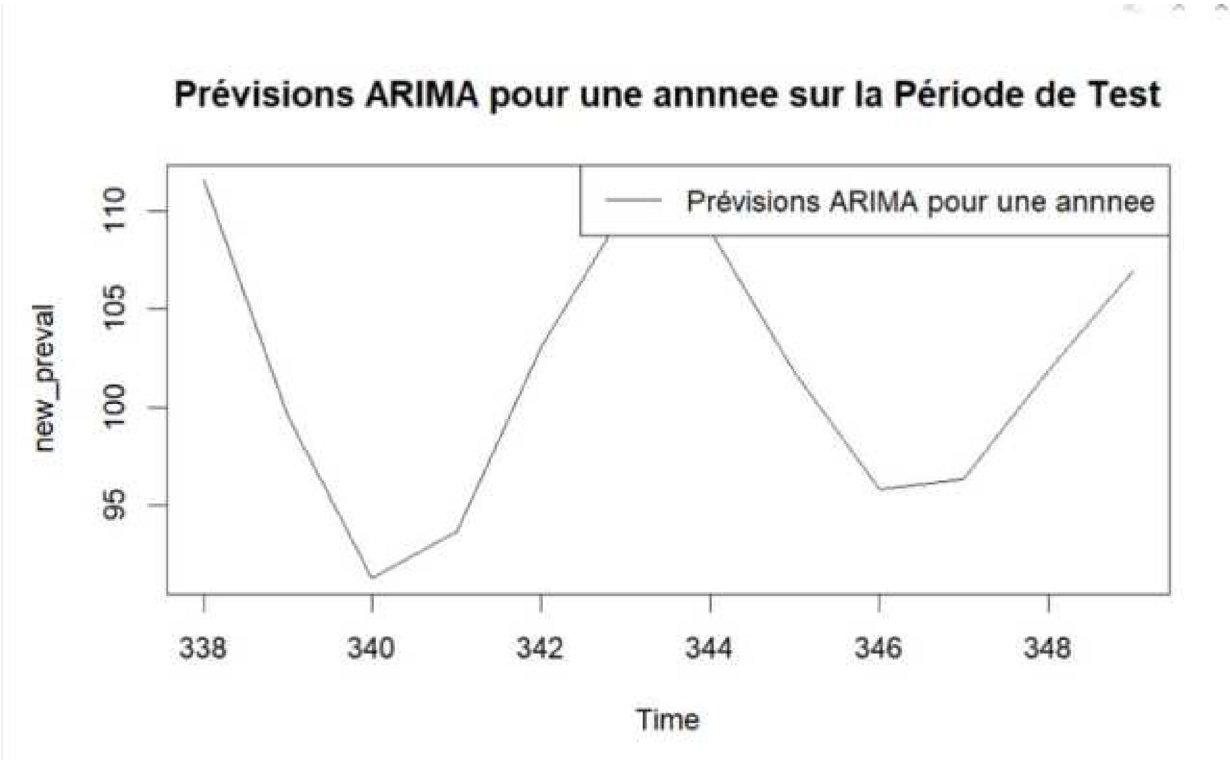
```
model_appr=arima(base_appr, order=c(2,1,2), seasonal=c(0,1,1))
model_appr


Call:
arima(x = base_appr, order = c(2, 1, 2), seasonal = c(0, 1, 1))

Coefficients:
         ar1      ar2      ma1     ma2     sma1
      0.9687  -0.8586  -1.0973  0.1339  -0.9574
s.e.  0.0319   0.0290   0.0683  0.0519   0.0445

sigma^2 estimated as 13.44:  log likelihood = -916.89,  aic = 1843.78

preval=predict(model_appr, n.ahead=12)
new_preval=preval$pred
new_preval
```

```
Time Series:
Start = 338
End = 349
Frequency = 1
 [1] 111.53957  99.51290  91.28890  93.67337 103.06969 110.15033 108.96746
 [8] 101.76766  95.83398  96.29288 101.85740 106.87933

summary(new_preval)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  91.29   96.18  101.81  101.74  107.40  111.54 

summary(base_test)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  88.35   93.02  102.40  103.00  110.26  129.40 
```

Le modèle s'ajuste bien aux données, avec des coefficients significatifs et des erreurs standard acceptables. La prévision montre une certaine variabilité dans la production électrique prévue pour les 12 prochains mois, ce qui est conforme aux données historiques. Le modèle peut être utilisé pour la planification future, mais il est important de continuer à surveiller les performances avec de nouvelles données

Graphique des previsions pour une annnee

```
# Visualiser les prévisions
plot(new_preval, main="Prévisions ARIMA pour une annnee sur la Période de Test")
lines(base_test, lty=2)
legend("topright", legend=c("Prévisions ARIMA pour une annnee"), lty=c(1, 2))
```



Effectuer des Prévisions sur la Période de Test pour une periode de trois ans

```
model_appr=arima(base_appr, order=c(2,1,2), seasonal=c(0,1,1))
model_appr


Call:
arima(x = base_appr, order = c(2, 1, 2), seasonal = c(0, 1, 1))

Coefficients:
         ar1      ar2      ma1     ma2     sma1
      0.9687  -0.8586  -1.0973  0.1339  -0.9574
s.e.  0.0319   0.0290   0.0683  0.0519   0.0445

sigma^2 estimated as 13.44:  log likelihood = -916.89,  aic = 1843.78

preval3ans=predict(model_appr, n.ahead=36)
new_preval3ans=preval3ans$pred
new_preval3ans

Time Series:
Start = 338
End = 373
Frequency = 1
 [1] 111.53957  99.51290  91.28890  93.67337 103.06969 110.15033 108.96746
 [8] 101.76766  95.83398  96.29288 101.85740 106.87933 106.99204 102.81487
[15]  98.69697  98.31967 101.51515 104.96006 105.57907 103.24640 100.48059
```

```
[22]  99.82947 101.59880 103.89725 104.63012 103.39208 101.58892 100.93051
[29] 101.86627 103.36347 104.03584 103.42712 102.28555 101.72773 102.19289
[36] 103.14784
```

```
summary(new_preval3ans)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  91.29  100.82  102.24  102.26  103.93  111.54 
```

```
length(new_preval3ans)
```

```
[1] 36
```

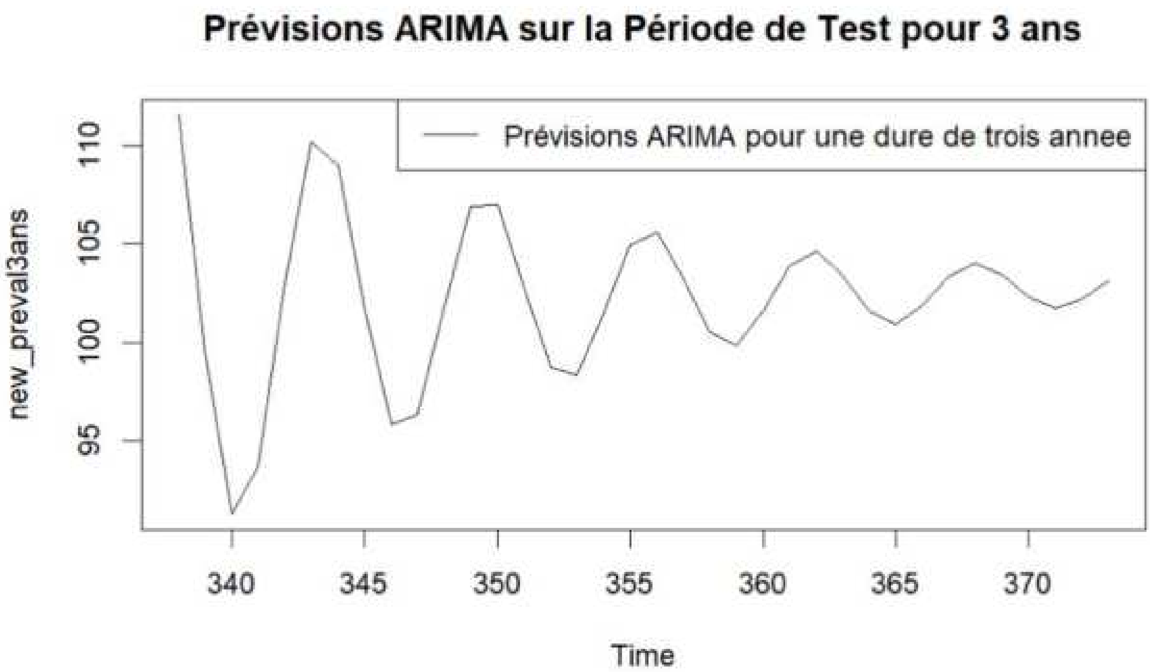
```
summary(base_test)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  88.35   93.02  102.40  103.00  110.26  129.40 
```

Le modèle s'ajuste bien aux données, avec des coefficients significatifs et des erreurs standard acceptables. La prévision montre une certaine variabilité dans la production électrique prévue pour les 36 prochains mois, ce qui est conforme aux données historiques. Le modèle peut être utilisé pour la planification future, mais il est important de continuer à surveiller les performances avec de nouvelles données.

Graphique des prévisions pour une dure de trois année

```
# Visualiser les prévisions
plot(new_preval3ans, main="Prévisions ARIMA sur la Période de Test pour 3 ans")
lines(base_test, lty=2)
legend("topright", legend=c("Prévisions ARIMA pour une dure de trois année"), lty=c(1, 2))
```



Nous avons fait aussi une validation par lissage pour en fin comparer les deux méthodes de validation.

Lissage exponentiel

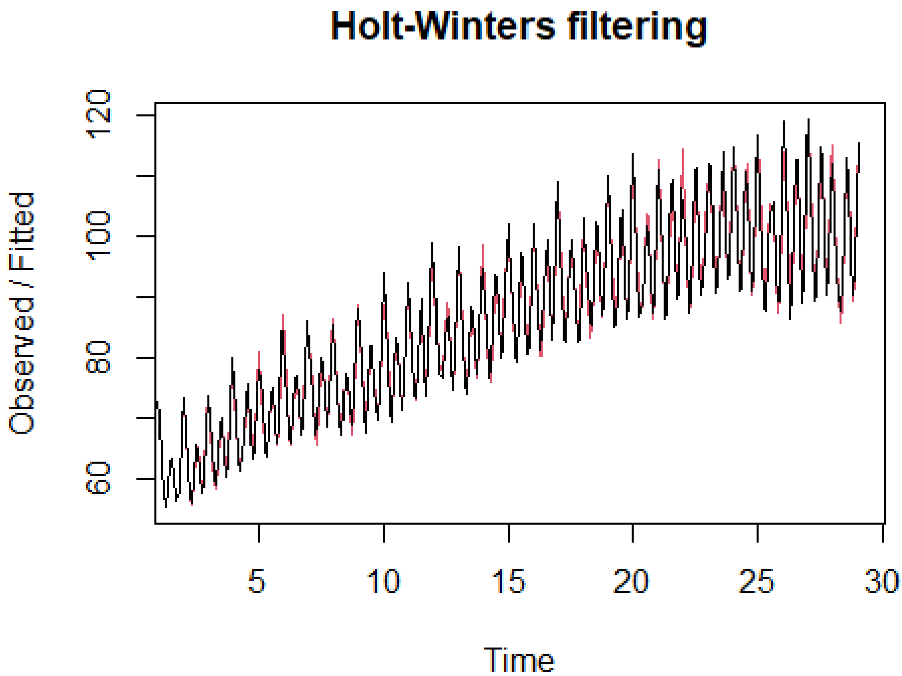
Prévision par lissage exponentiel

```
# Définir la fréquence de la série temporelle
base_appr_ts <- ts(base_appr, frequency = 12)
# Appliquer le modèle Holt-Winters
lissage_appr <- HoltWinters(base_appr_ts)
# Afficher les résultats
print(lissage_appr)

Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = base_appr_ts)

Smoothing parameters:
 alpha: 0.4641251
 beta : 0
 gamma: 0.4965931
```

```
Coefficients:
           [,1]
a    99.97092336
b     0.06902204
s1    5.59939904
s2   -2.81231065
s3  -10.95259277
s4   -8.22357864
s5    2.16471610
s6   11.54760703
s7    9.75536520
s8   -2.93185158
s9  -10.43265334
s10  -7.30950055
s11   7.55714043
s12  15.03143982
```

Les résultats du modèle de lissage exponentiel de Holt-Winters indiquent une prévision basée sur la tendance et la saisonnalité additive des données chronologiques. Les paramètres de lissage montrent un alpha de 0.464, suggérant que les observations récentes ont un poids significatif dans les prévisions futures. Le beta de 0 indique l'absence de variation de tendance à long terme dans ce modèle. Le gamma de 0.497 indique une variation saisonnière marquée. Les coefficients s1 à s12 représentent les effets saisonniers spécifiques pour chaque mois de l'année. Ensemble, ces résultats permettent de prévoir la série chronologique avec une prise en compte adaptée de la saisonnalité et des tendances récentes, tout en identifiant les variations périodiques significatives dans les données.

```
# Representation Graphique
plot(lissage_appr)
```



Prévision pour une année (12 mois)

```
prevision_1_an_lis <- predict(lissage_appr, n.ahead = 12, prediction.interval = TRUE)
prevision_1_an_lis

             fit       upr       lwr
Feb 29 105.63934 110.28625 100.99244
Mar 29  97.29666 102.41967  92.17364
Apr 29  89.22540  94.78389  83.66691
May 29  92.02343  97.98568  86.06119
Jun 29 102.48075 108.82109  96.14041
Jul 29 111.93266 118.62978 105.23554
Aug 29 110.20944 117.24528 103.17361
Sep 29  97.59125 104.95023  90.23227
Oct 29  90.15947  97.82798  82.49096
Nov 29  93.35164 101.31767  85.38561
Dec 29 108.28731 116.54014 100.03448
Jan 30 115.83063 124.36062 107.30064
```

Prévisions sur la production électrique par lissage appliqué sur une période d'un an, avec des intervalles de prédiction également fournis.

Prévision pour trois années (36 mois)

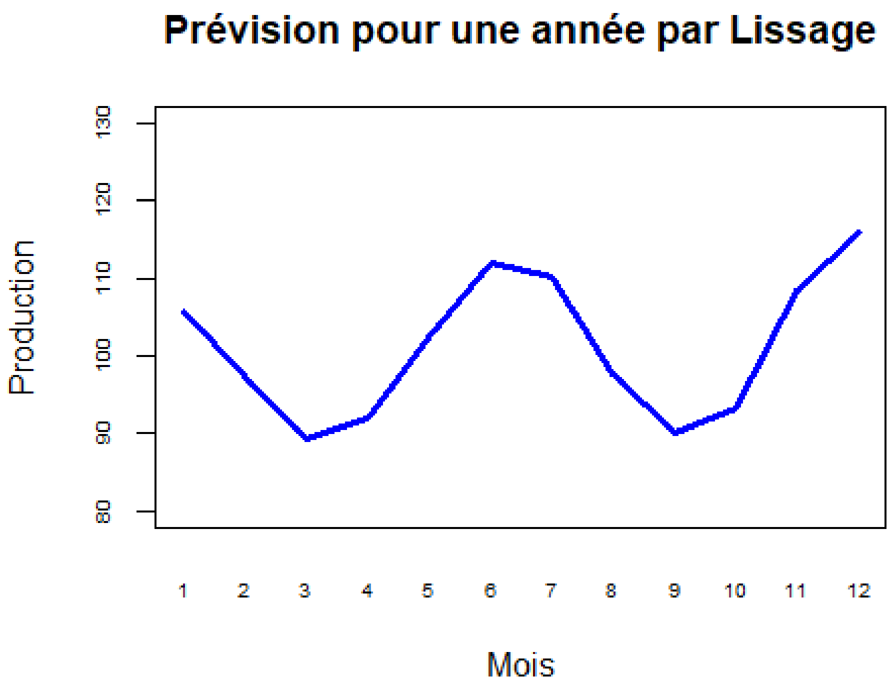
```
prevision_3_ans_lis <- predict(lissage_appr, n.ahead = 36, prediction.interval = TRUE)
prevision_3_ans_lis

             fit       upr       lwr
Feb 29 105.63934 110.28625 100.99244
Mar 29  97.29666 102.41967  92.17364
Apr 29  89.22540  94.78389  83.66691
May 29  92.02343  97.98568  86.06119
Jun 29 102.48075 108.82109  96.14041
Jul 29 111.93266 118.62978 105.23554
Aug 29 110.20944 117.24528 103.17361
Sep 29  97.59125 104.95023  90.23227
Oct 29  90.15947  97.82798  82.49096
Nov 29  93.35164 101.31767  85.38561
Dec 29 108.28731 116.54014 100.03448
Jan 30 115.83063 124.36062 107.30064
Feb 30 106.46761 115.64778  97.28744
Mar 30  98.12492 107.55503  88.69481
Apr 30  90.05366  99.72726  80.38006
May 30  92.85170 102.76281  82.94059
Jun 30 103.30901 113.45207  93.16595
Jul 30 112.76093 123.13075 102.39110
Aug 30 111.03771 121.62944 100.44598
Sep 30  98.41951 109.22860  87.61043
Oct 30  90.98773 102.00989  79.96558
Nov 30  94.17991 105.41109  82.94873
Dec 30 109.11557 120.55196  97.67918
Jan 31 116.65889 128.29687 105.02092
Feb 31 107.29587 119.41847  95.17328
Mar 31  98.95319 111.26614  86.64023
Apr 31  90.88193 103.38234  78.38151
May 31  93.67996 106.36507  80.99486
Jun 31 104.13728 117.00443  91.27013
Jul 31 113.58919 126.63584 100.54254
Aug 31 111.86597 125.08968  98.64226
Sep 31  99.24778 112.64621  85.84934
Oct 31  91.81600 105.38691  78.24509
Nov 31  95.00817 108.74939  81.26695
Dec 31 109.94384 123.85328  96.03439
Jan 32 117.48716 131.56282 103.41149
```

Prévisions sur la production electrique par lissage appliqué sur une période de trois ans, avec des intervalles de prédiction également fournis.

Graphiques des prévisions

Graphique pour une année

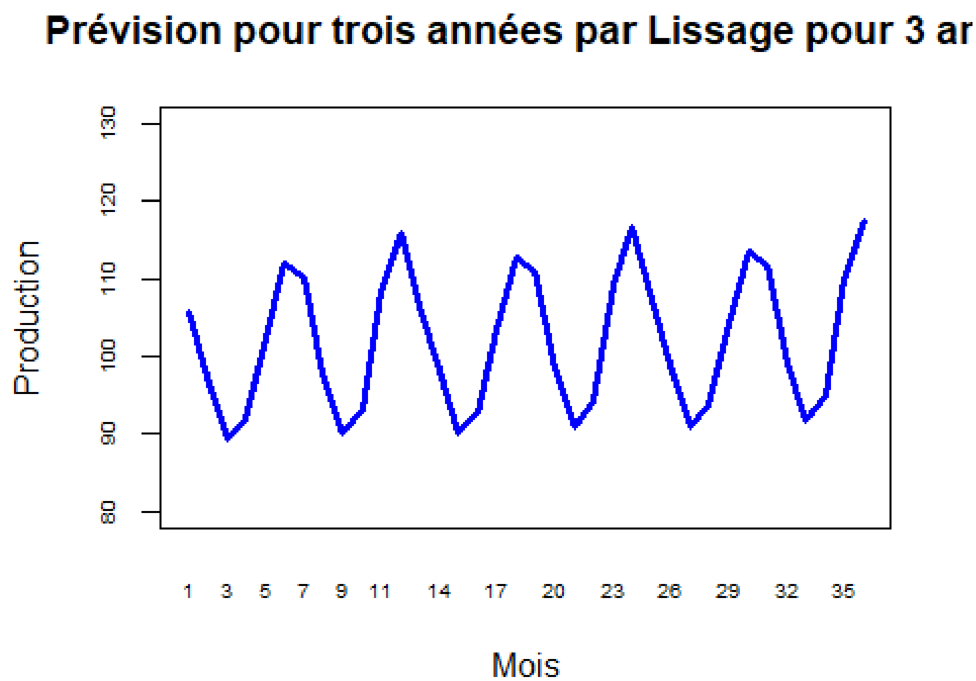
```
indice_1_an <- 1:12
plot(indice_1_an, prevision_1_an_lis[,1], type='l', col='blue', lwd=3, ylim=c(80, 130),
     cex.axis=0.6, xaxt='n', xlab='Mois', ylab='Production', main='Prévision pour une année par
Lissage')
axis(1, 1:12, indice_1_an, tck=F, cex.axis=0.6)
```



Les observations montrent que le modèle par lissage exponentiel est efficace pour des séries avec des motifs saisonniers réguliers.

Graphique pour trois années

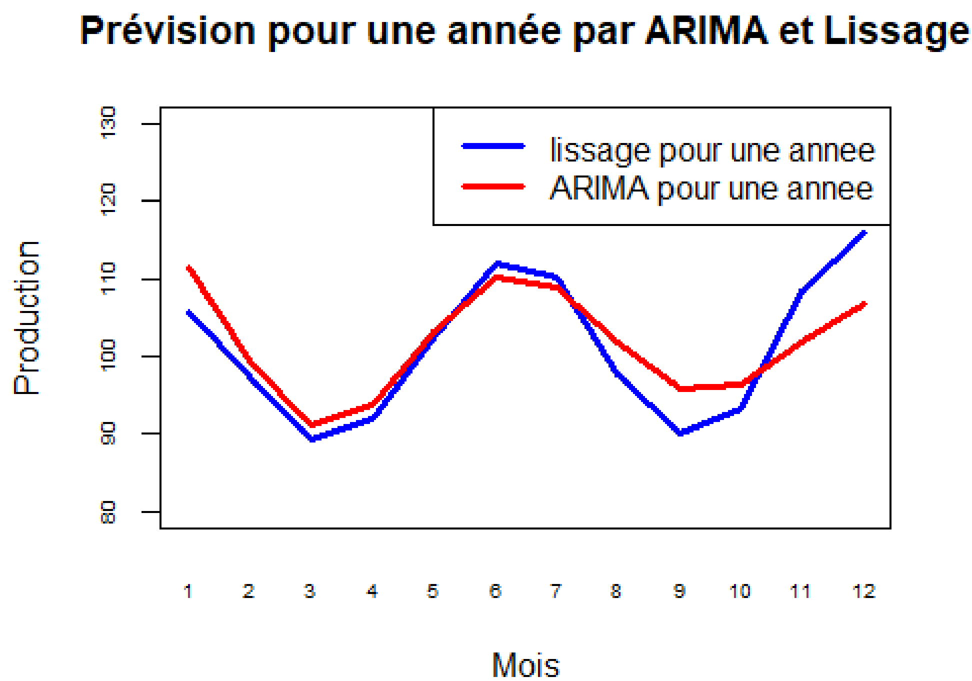
```
indice_3_ans <- 1:36
plot(indice_3_ans, prevision_3_ans_lis[,1], type='l', col='blue', lwd=3, ylim=c(80, 130),
     cex.axis=0.6, xaxt='n', xlab='Mois', ylab='Production', main='Prévision pour trois années par
Lissage pour 3 ans')
axis(1, 1:36, indice_3_ans, tck=F, cex.axis=0.6)
```



Ces observations suggèrent que le modèle Holt-Winters est efficace pour des séries avec des motifs saisonniers réguliers et sans tendances à long terme.

Graphique comparatif pour une année

```
plot(indice_1_an, prevision_1_an_lis[,1], type='l', col='blue', lwd=3, ylim=c(80, 130),
    cex.axis=0.6, xaxt='n', xlab='Mois', ylab='Production', main='Prévision pour une année par
ARIMA et Lissage')
axis(1, 1:12, indice_1_an, tck=F, cex.axis=0.6)
lines(indice_1_an, new_preval, col='red', lwd=3)
legend("topright", legend=c("lissage pour une annee", "ARIMA pour une annee"), col=c("blue",
"red"), lwd=3)
```



Ces deux modèles donnent des prévisions similaires, mais avec des différences notables dans le lissage et la réaction aux fluctuations. Le model arima montre une variabilité légèrement différente, avec des ajustements plus rapides aux changements.

Les prévisions ARIMA peuvent mieux s'ajuster aux fluctuations irrégulières et en bleu le model de lissage montre des prévisions plus lisses et plus régulières.