project_2

Tejas Kolpek (50559893), Rahul Mannadiar (50560682), Farahath Tabassum (50559788), Prathamesh Ra

2024-11-27

Oil Price Time Series Forecasting

Introduction:

The oil price data from the oil.csv file, which has missing values, will be analyzed for this project. In order to investigate trends and seasonality, the tasks include plotting the initial time series with missing values, imputing missing values, and plotting the time series after imputing the missing values. Additionally, we study Holt-Winters and ETS models. We then apply appropriate models for our data based on the imputed time series, keeping an eye on trend and seasonality. After that, we run the models and assess their suitability. Lastly, we evaluate the model's performance using metrics such as the AIC, BIC, and Ljung-Box test. Then, we compare the models' performance using RMSE to determine which model performs best.

Step 1. Loading Required Libraries and Data

```
# Load required libraries
suppressWarnings({
library(ggplot2)
library(dplyr)
library(forecast)
library(imputeTS)
})
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

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

```r
# Load the dataset
oil_data <- read.csv("D:/fall 2024/statistical learning/oil.csv")
head(oil_data)
```

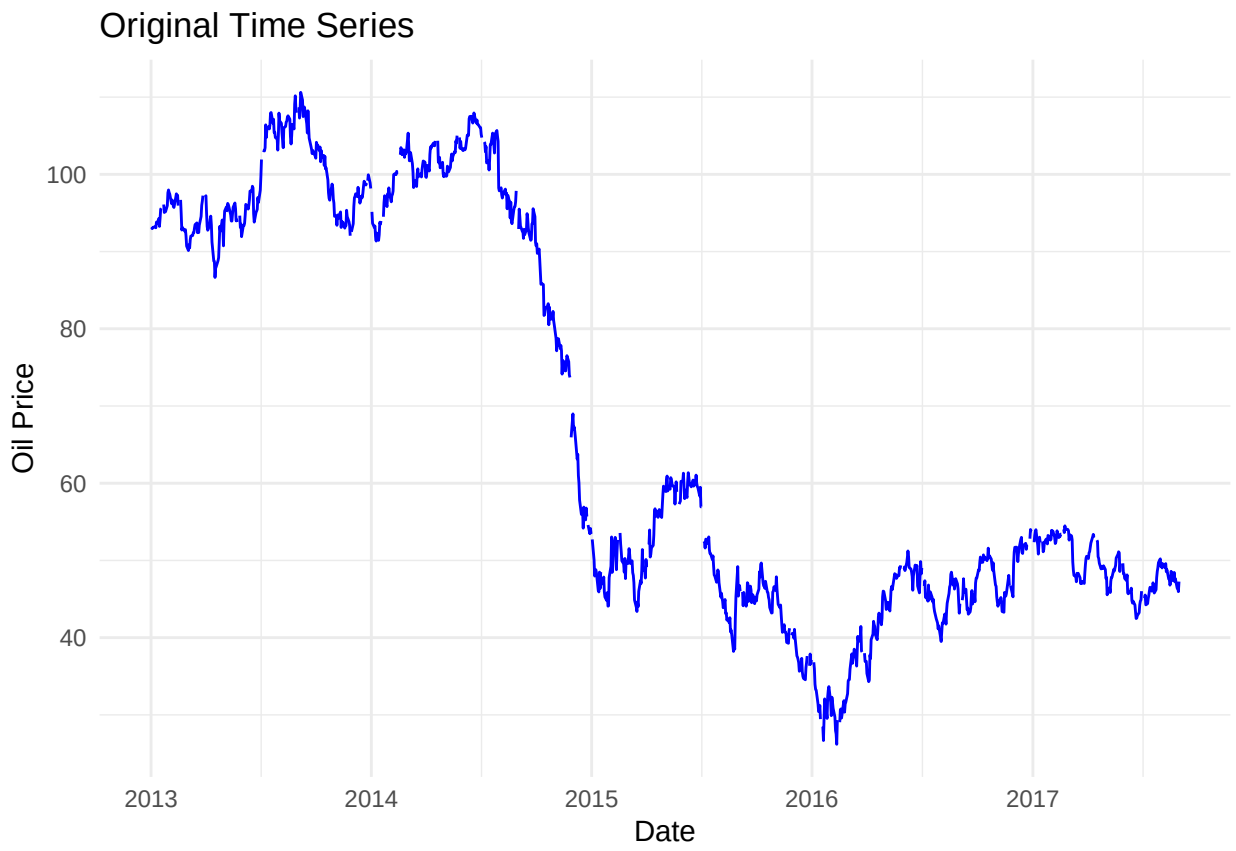
```
##         date dcoilwtico
## 1 2013-01-01         NA
## 2 2013-01-02      93.14
## 3 2013-01-03      92.97
## 4 2013-01-04      93.12
## 5 2013-01-07      93.20
## 6 2013-01-08      93.21
```

Step 2. Plotting the Original Time Series (with missing values)

The date column in the dataset was converted to the Date type to ensure proper handling of temporal data. This step is crucial for accurate visualization and analysis of time series data. Subsequently, the original time series of oil prices was plotted using the ggplot2 library, with the date column on the x-axis and the oil price (dcoilwtico) on the y-axis. The plot highlights the trends, seasonal patterns, and potential missing values in the raw data, providing an initial overview of the dataset's structure and behavior.

```r
# Convert date column to Date type
oil_data$date <- as.Date(oil_data$date)

# Plot the time series
ggplot(oil_data, aes(x = date, y = dcoilwtico)) +
  geom_line(color = "blue") +
  labs(title = "Original Time Series", x = "Date", y = "Oil Price") +
  theme_minimal()
```

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



Trend:

There is a clear downward trend around 2015 and stabilization afterward. However, this does not suggest that the seasonality scales with the trend.

Seasonality:

It appears that the seasonal fluctuations do not increase or decrease proportionally with the level of the trend. The amplitude of the fluctuations seems fairly consistent across the time series, despite the significant drop in oil prices around 2015.

Step 3. Handling Missing Data

We impute the missing data using linear interpolation. This method ensures that the imputed values are smooth and realistic.

```
# Impute missing values using linear interpolation
oil_data$dcoilwtico <- na_interpolation(oil_data$dcoilwtico, option = "linear")

# Verify that missing values are handled
sum(is.na(oil_data$dcoilwtico))
```

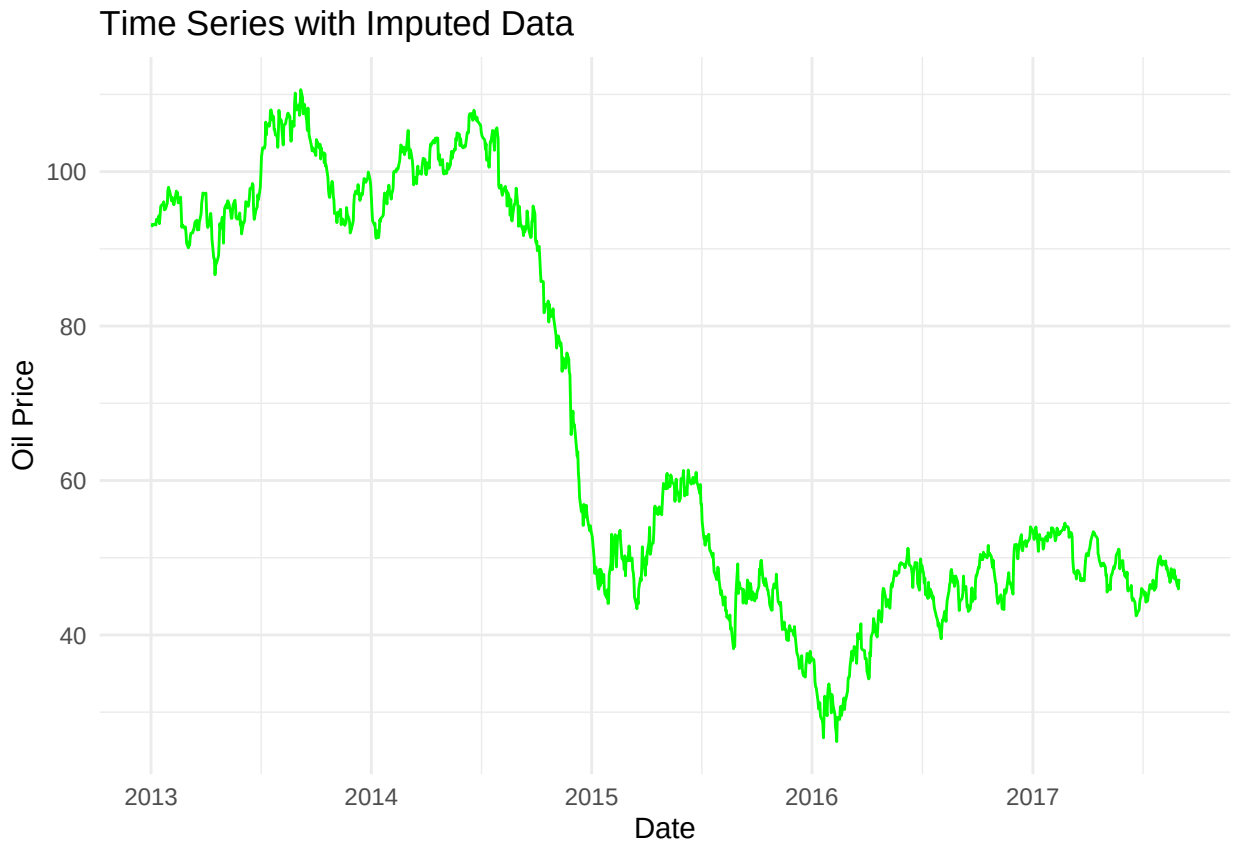
```
## [1] 0
```

Step 4.

(a) Plotting the Time Series After Imputation (without missing values)

After imputing the missing data, we plot the time series again to observe if there are trends or seasonality.

```
# Plot the imputed time series
ggplot(oil_data, aes(x = date, y = dcoilwtico)) +
  geom_line(color = "green") +
  labs(title = "Time Series with Imputed Data", x = "Date", y = "Oil Price") +
  theme_minimal()
```




```
# Convert data to time series
oil_ts <- ts(oil_data$dcoilwtico, start = c(2013, 1), frequency = 365)
```

The trend and seasonality remained same even after imputing the missing values in the original time series.

Trend:

There is a clear downward trend around 2015 and stabilization afterward. However, this does not suggest that the seasonality scales with the trend.

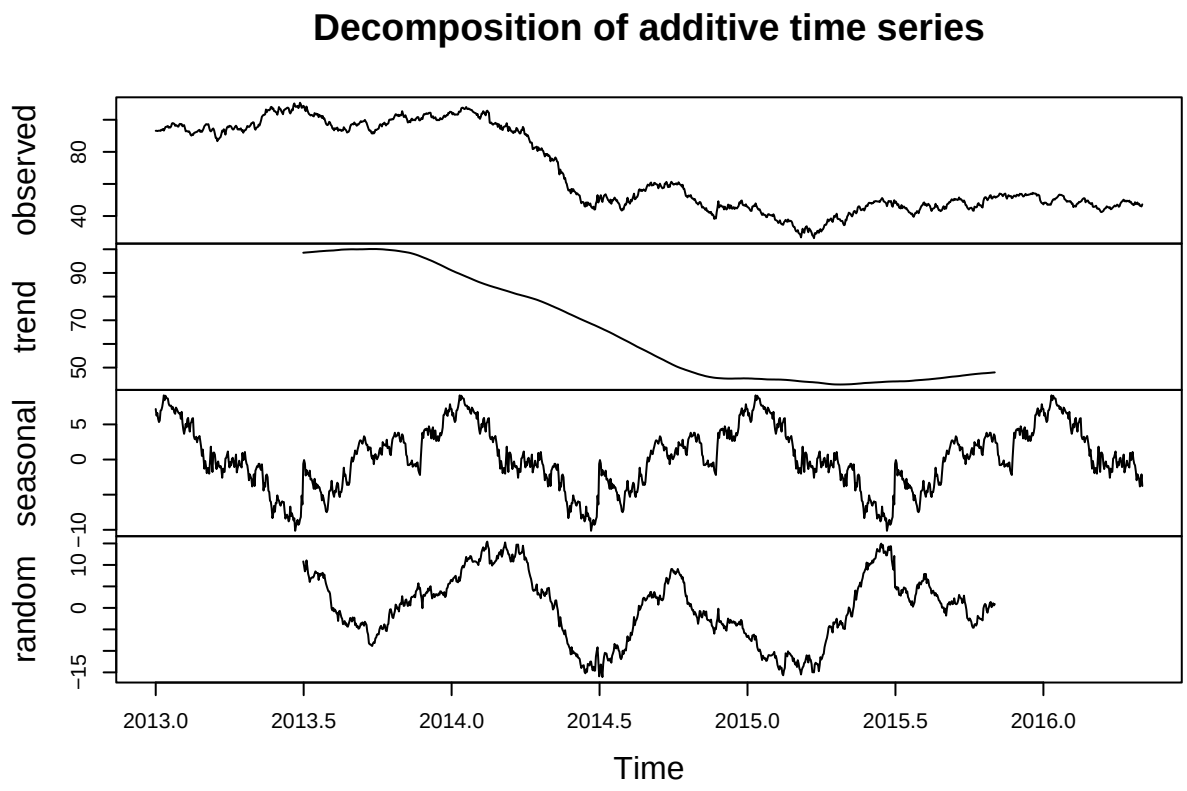
Seasonality:

It appears that the seasonal fluctuations do not increase or decrease proportionally with the level of the trend. The amplitude of the fluctuations seems fairly consistent across the time series, despite the significant drop in oil prices around 2015.

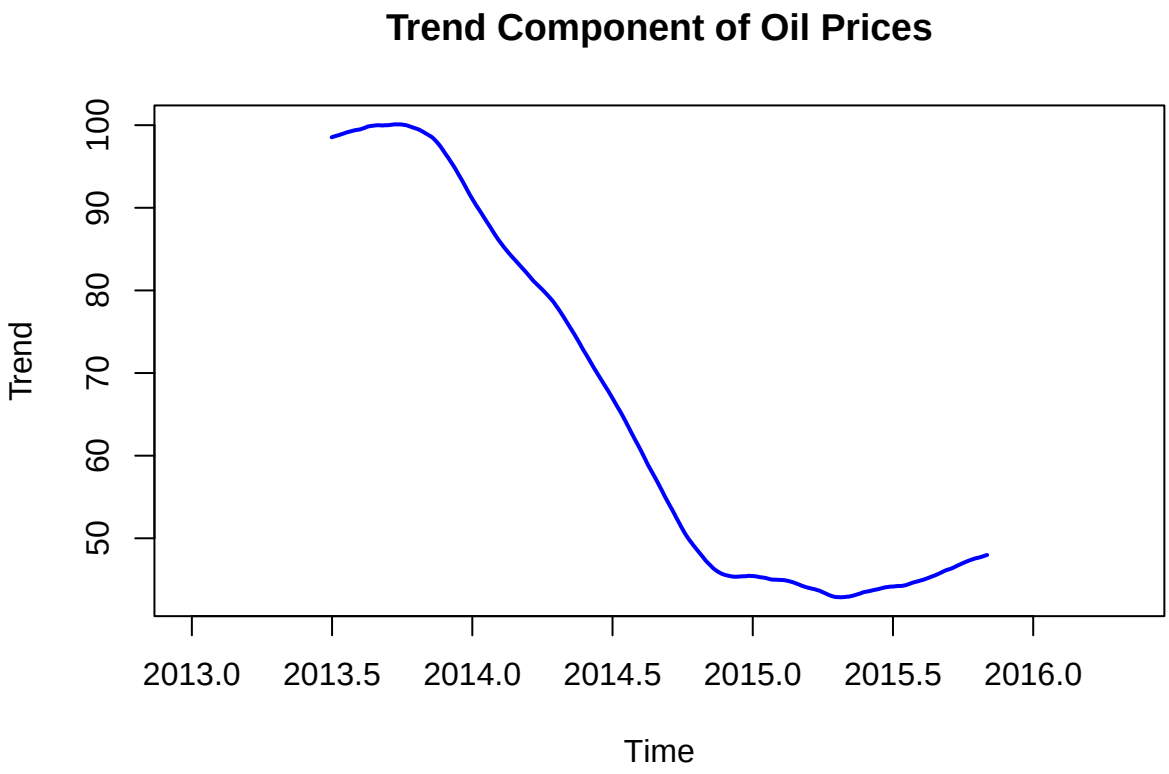
Step 4.

(b) Understanding Trend, Seasonality and Residual Components of Time Series

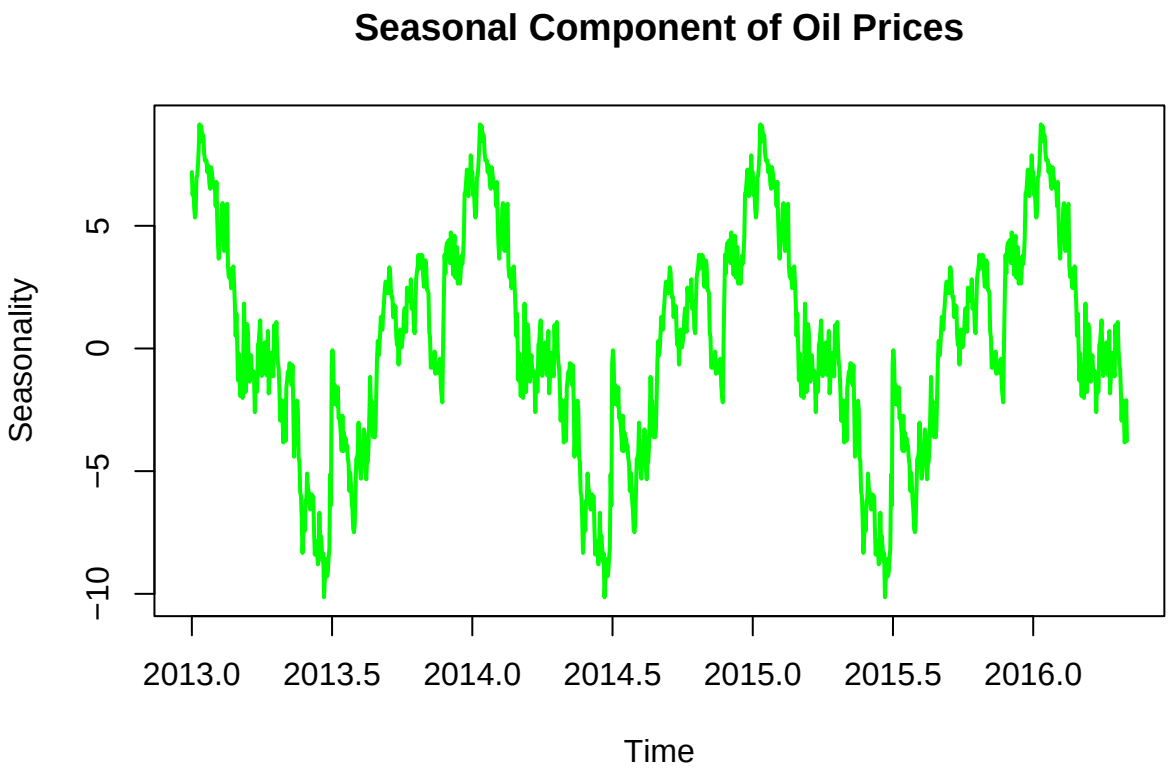
```r
# Decompose the time series into trend, seasonal, and residual components
decomposed_ts <- decompose(oil_ts, type = "additive")  # Since we have constant seasonal variation

# Plot the decomposed components (no 'main' argument)
plot(decomposed_ts)
```



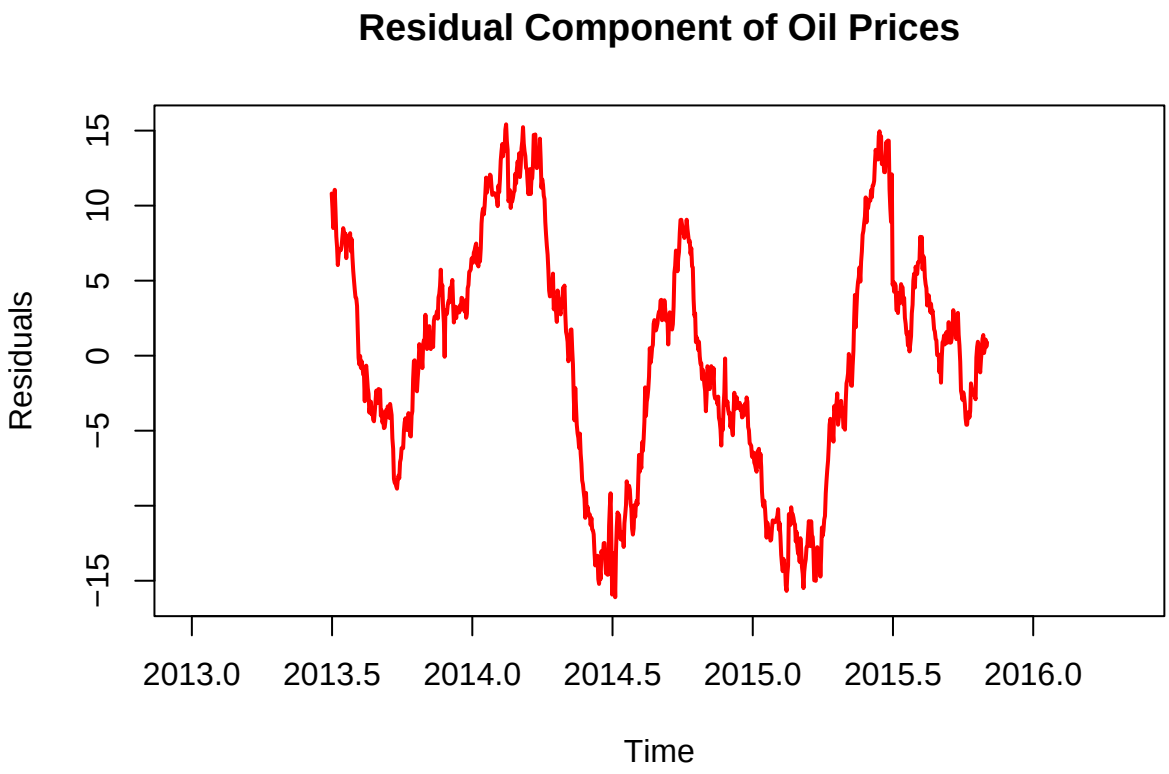

```r
# Add separate titles to each component manually (optional)
# Plot trend separately for clearer visualization
plot(decomposed_ts$trend,
     main = "Trend Component of Oil Prices",
     ylab = "Trend",
     xlab = "Time",
     col = "blue",
     lwd = 2)
```




```
# Similarly, you can manually plot seasonal and random components if needed:
plot(decomposed_ts$seasonal,
     main = "Seasonal Component of Oil Prices",
     ylab = "Seasonality",
     xlab = "Time",
     col = "green",
     lwd = 2)
```




```r
plot(decomposed_ts$random,
     main = "Residual Component of Oil Prices",
     ylab = "Residuals",
     xlab = "Time",
     col = "red",
     lwd = 2)
```



Analysis:

Trend Component: The trend component of oil prices shows a significant and consistent decline from 2013 to 2015, indicating a downward movement in overall oil price levels during this period. After hitting a low point around mid-2015, the trend stabilizes and shows slight signs of recovery toward the end of 2015 and into early 2016.

Seasonal Component: The seasonal component exhibits repetitive cyclical patterns across the years, suggesting a regular fluctuation in oil prices due to seasonal factors. Peaks and troughs recur at similar intervals, possibly linked to changes in energy demand based on weather conditions, holiday periods, or other predictable events. The amplitude of the seasonal variations remains fairly consistent throughout the analyzed period.

Residual Component:

The residual component captures the irregular fluctuations in oil prices that cannot be explained by the trend or seasonal components. These residuals display high volatility, with periods of sharp increases and decreases.

Stationarity Check

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.3.3
```

```
##
## Attaching package: 'tseries'

## The following object is masked from 'package:imputeTS':
##
##     na.remove
```

```r
# Perform the Augmented Dickey-Fuller Test
adf_test <- adf.test(oil_ts)

# Print the results
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  oil_ts
## Dickey-Fuller = -1.4636, Lag order = 10, p-value = 0.8054
## alternative hypothesis: stationary
```

```r
# Interpretation
if (adf_test$p.value <= 0.05) {
  cat("Conclusion: The time series is stationary (reject null hypothesis of unit root).")
} else {
  cat("Conclusion: The time series is not stationary (fail to reject null hypothesis of unit root).")
}
```

```
## Conclusion: The time series is not stationary (fail to reject null hypothesis of unit root).
```

Based on the ADF Test we can see the time series is not stationary since the p-value is greater than 0.05.

Step 5. ETS models and Holt-Winter models

ETS Models:

ETS models are a framework for exponential smoothing methods that decompose a time series into three components: Error (E), Trend (T), and Seasonality (S). These components can each be additive (A), multiplicative (M), or non-existent (N). The ETS model is selected based on the best fit to the data using criteria like AIC. Additive components imply constant change (linear), while multiplicative components account for proportional change. ETS models assume errors are independent and identically distributed, and they provide a probabilistic foundation for forecasting with uncertainty estimates. In R, ETS models can be implemented using the ets() function, which automatically selects the best model or allows manual specification.

Holt-Winters Models:

The Holt-Winters model is a specific type of exponential smoothing used for time series with trend and seasonality. It comes in two main variations: additive (for constant seasonality) and multiplicative (for proportional seasonality). The model extends simple exponential smoothing by adding equations to smooth the level, trend, and seasonality components. It's effective for data with clear seasonal patterns and assumes the seasonality repeats over a fixed period. In R, the Holt-Winters method is implemented using the HoltWinters() function, and it works best on stationary data, often requiring preprocessing like detrending or deseasonalizing for optimal results.

Step 6. Selecting suitable models based on data

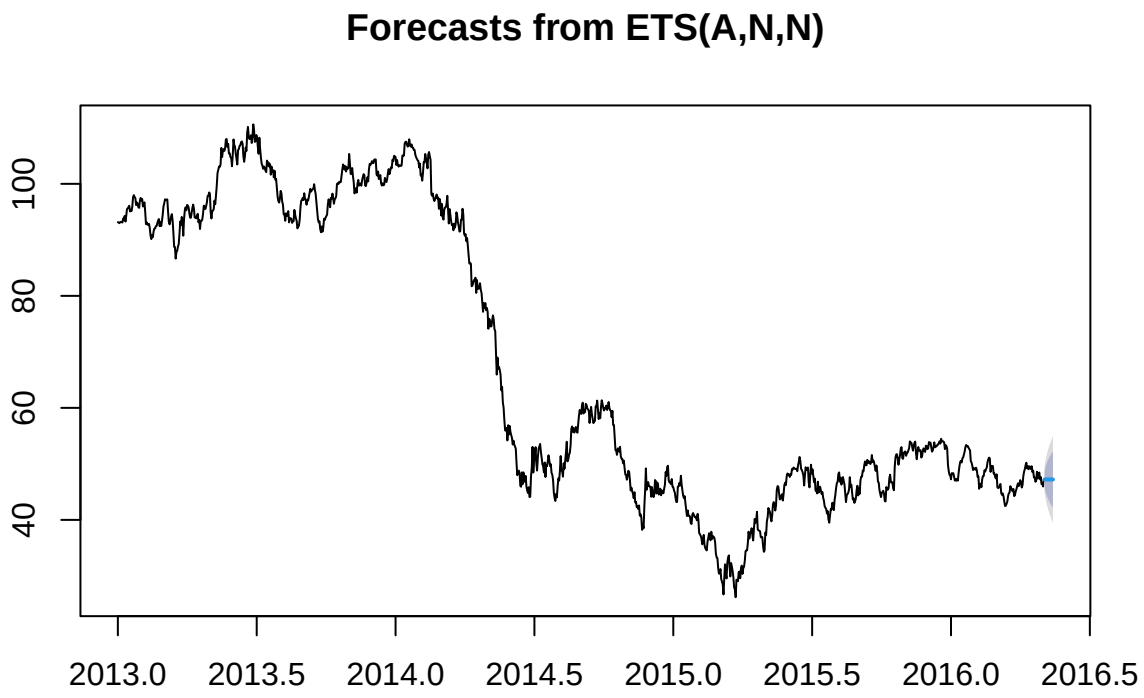
Based on the analysis of time series data, the ETS model (Error, Trend, Seasonal) with seasonal components (e.g., ETS(A,N,M) or ETS(M,A,M)) is a strong option if the data exhibits seasonality, as it automatically adjusts for seasonality and trends. The Holt-Winters model with multiplicative seasonality is another suitable choice, particularly if the seasonal variations are proportional to the level of the series, as it performed well in terms of residual autocorrelation. Finally, if the data shows complex seasonal patterns and trends, a SARIMA model with STLF (Seasonal and Trend decomposition using Loess) can provide a more robust solution by automatically handling intricate seasonality and trend decomposition. Testing and comparing these models based on forecast accuracy and residual analysis will help determine the most suitable model for your data.

Step 7. Implementing suitable models

(a) ETS Model

```
library(forecast)
suppressWarnings({
# Fit an ETS model to a time series
ets_model <- ets(oil_ts)

# Summary of the model
summary(ets_model)

# Print ETS Model
print(ets_model)

# Plot the forecast
forecast_ets <- forecast(ets_model, h = 12)  # Forecast for the next 12 periods
plot(forecast_ets)
})
```

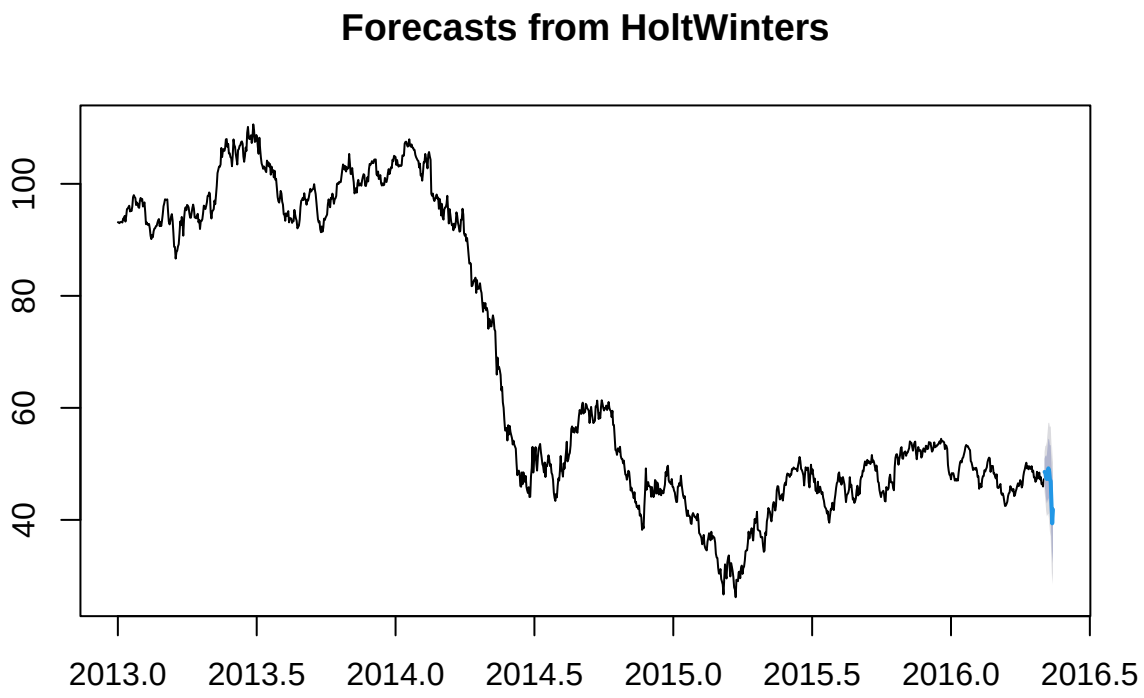
```
## ETS(A,N,N)
##
## Call:
## ets(y = oil_ts)
##
##   Smoothing parameters:
##     alpha = 0.9683
##
##   Initial states:
##     l = 93.1395
##
##   sigma:  1.1771
##
##      AIC     AICc      BIC
## 9055.133 9055.153 9070.448
```



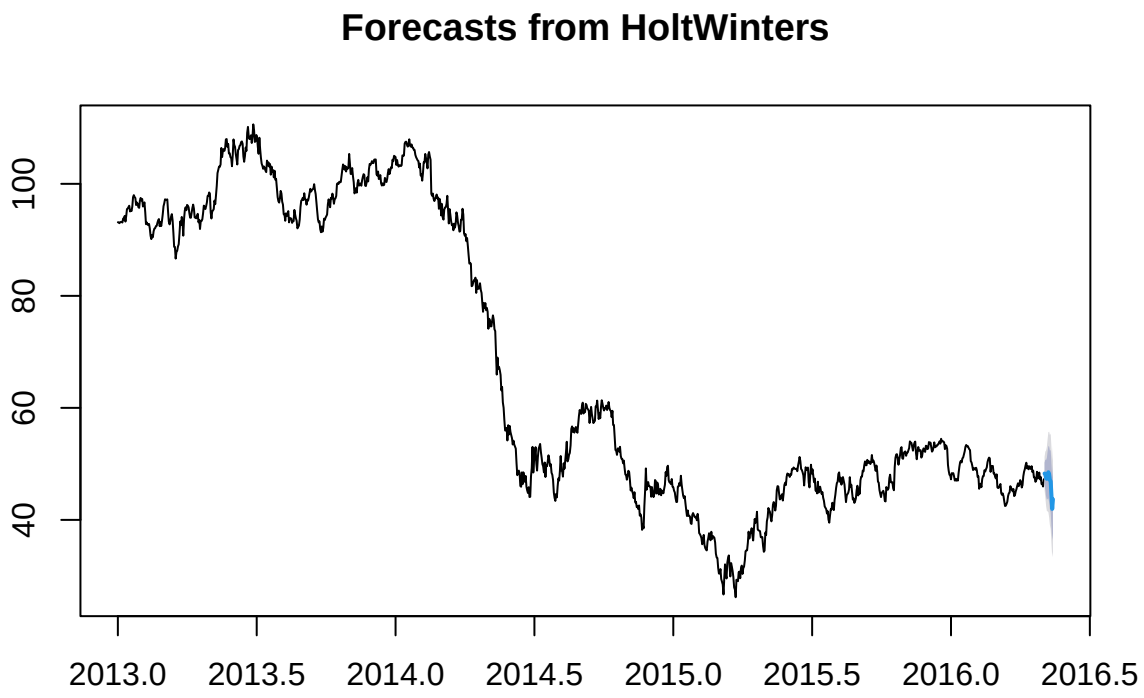
The output indicates that the ETS(A,N,N) model was selected for the time series oil_ts, which means the model uses additive error (A), no trend (N), and no seasonality (N). The smoothing parameter alpha is 0.9683, showing a high weight given to recent observations. The initial level (l) of the series is 93.1395, and the residual standard deviation (sigma) is 1.1771. The model's fit is quantified by several metrics: the AIC (9055.133), AICc (9055.153), and BIC (9070.448), which are used to assess model performance, with lower values indicating better fits. The training set error measures show a mean error (ME) close to zero (-0.0389), suggesting minimal bias. RMSE (1.1762) and MAE (0.8951) reflect the average deviations from the true values, while MAPE (1.5483%) suggests a small percentage error in forecasts. The ACF1 value (-0.00075) indicates no significant autocorrelation in the residuals. However, the warning about seasonality being ignored (due to a frequency greater than 24) suggests that seasonality is not being modeled, and the stlf() function may be needed for seasonal adjustments.

Step 5.(b) Holt-Winters Model

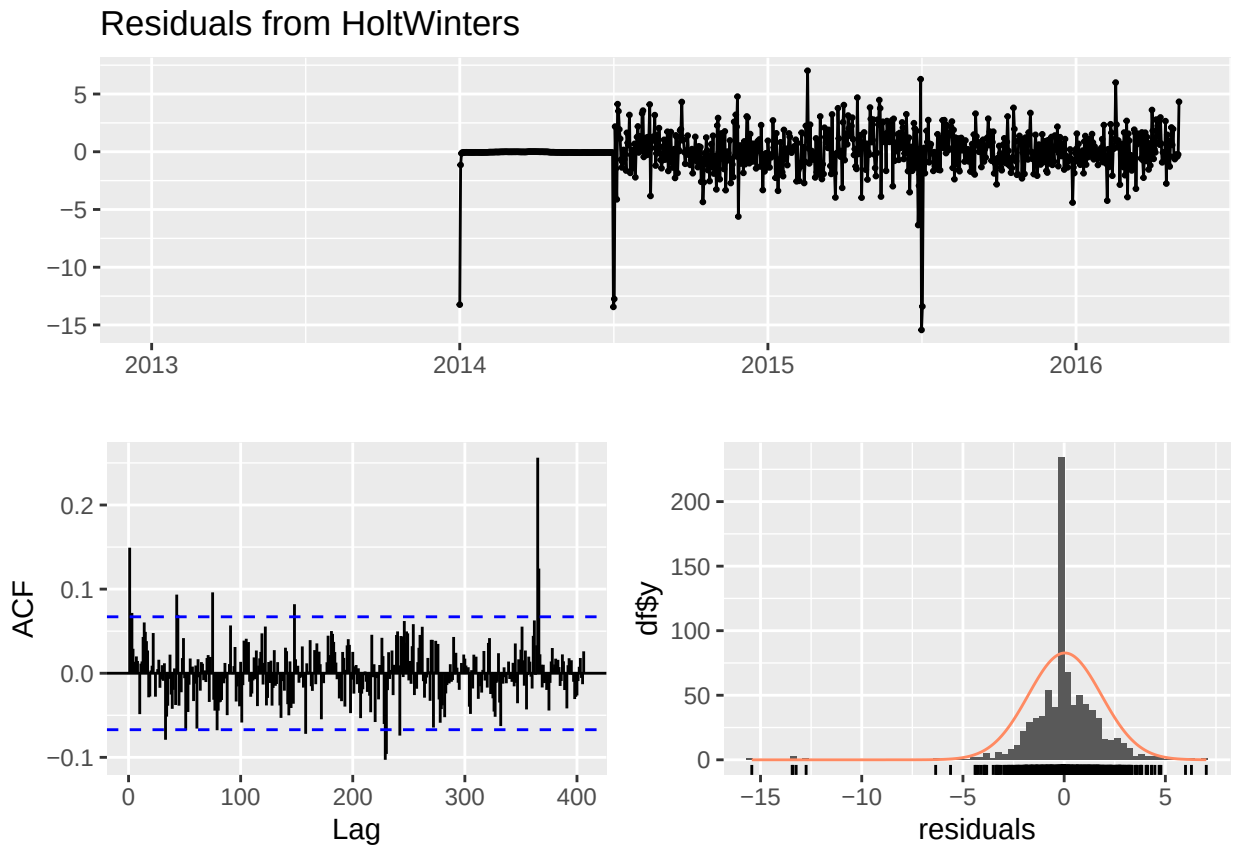
```
# Holt-Winters Additive Model
hw_additive <- HoltWinters(oil_ts, seasonal = "additive")

# Forecast for Additive Model
hw_additive_forecast <- forecast(hw_additive, h = 12)
plot(hw_additive_forecast)
```




```r
# Holt-Winters Multiplicative Model
hw_multiplicative <- HoltWinters(oil_ts, seasonal = "multiplicative")

# Forecast for Multiplicative Model
hw_multiplicative_forecast <- forecast(hw_multiplicative, h = 12)
plot(hw_multiplicative_forecast)
```

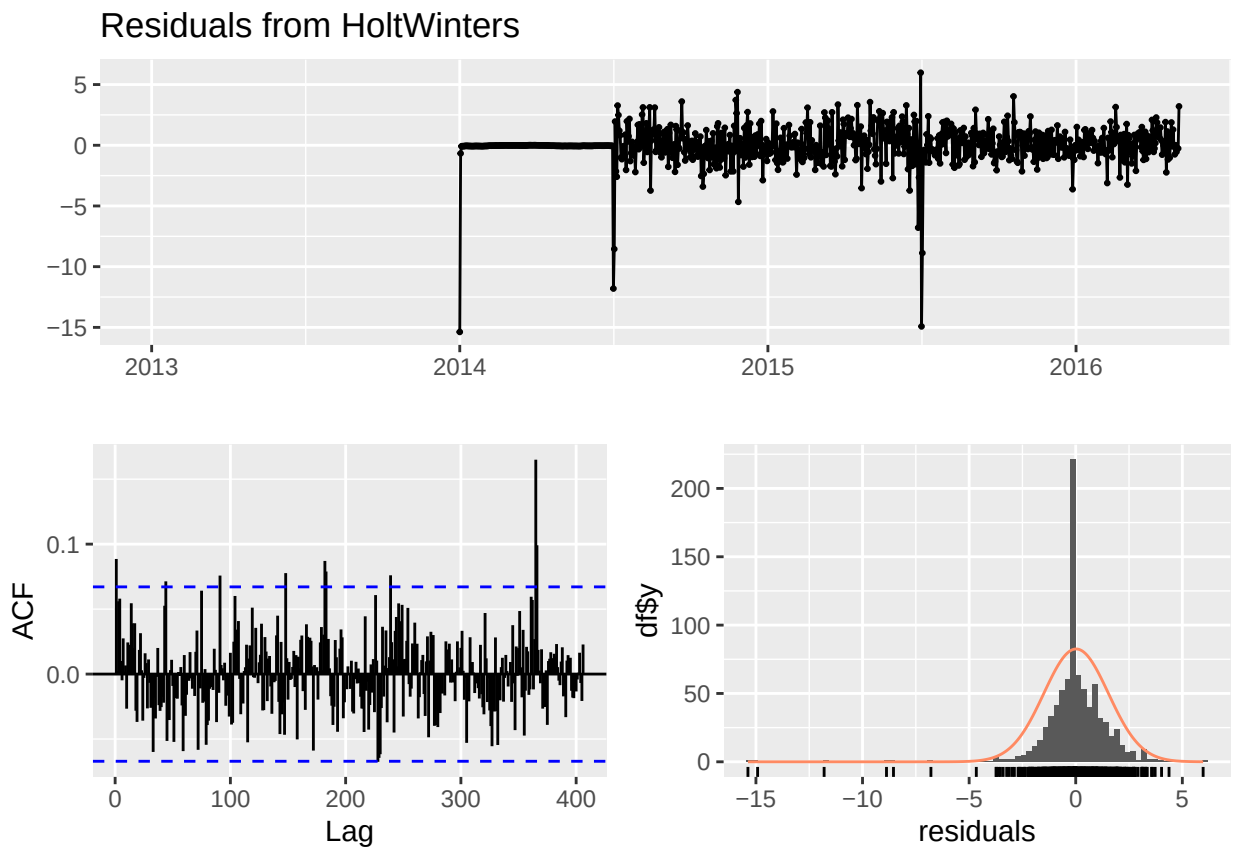



```
# Check residuals for the models
checkresiduals(hw_additive_forecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from HoltWinters
## Q* = 273.46, df = 244, p-value = 0.09457
##
## Model df: 0.   Total lags used: 244
```

```
checkresiduals(hw_multiplicative_forecast)
```



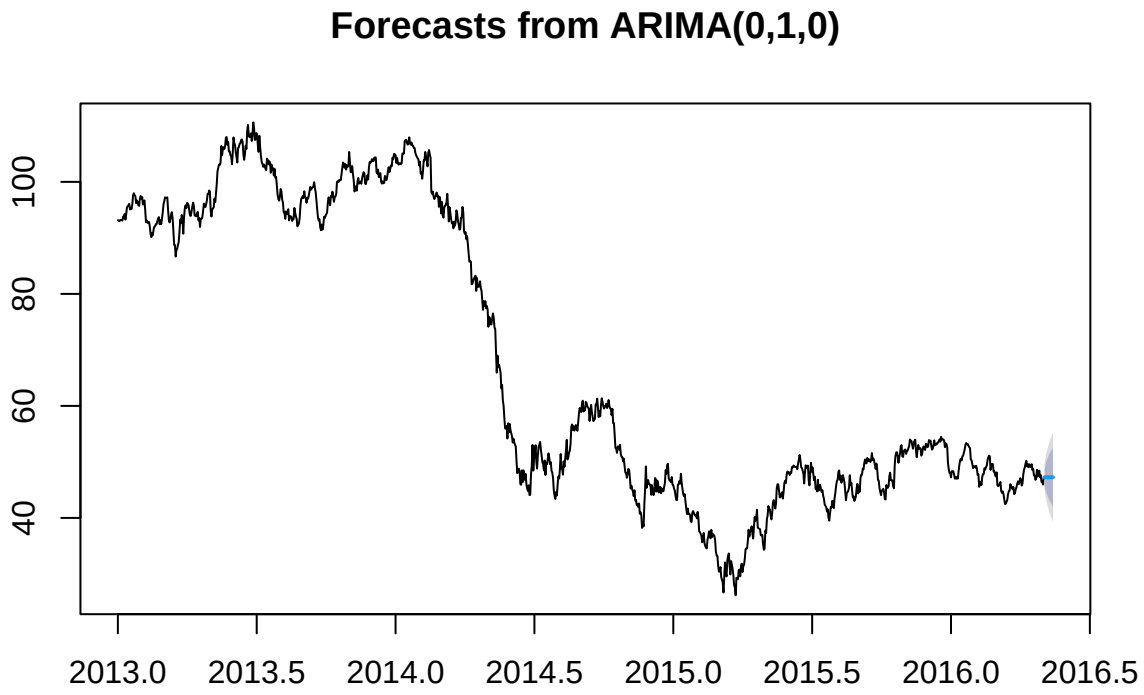

```
##
##  Ljung-Box test
##
## data:  Residuals from HoltWinters
## Q* = 219.84, df = 244, p-value = 0.8647
##
## Model df: 0.   Total lags used: 244
```

The Ljung-Box test for the residuals from the Holt-Winters additive model resulted in a p-value of 0.09457, which is slightly above the typical significance threshold of 0.05. While this suggests some evidence of autocorrelation in the residuals, the p-value is close to the threshold, indicating marginal or weak autocorrelation. Therefore, while the additive model may be generally capturing the time series dynamics, the presence of some residual correlation suggests that the model could potentially be improved, possibly by refining the seasonal component or adding more complexity.

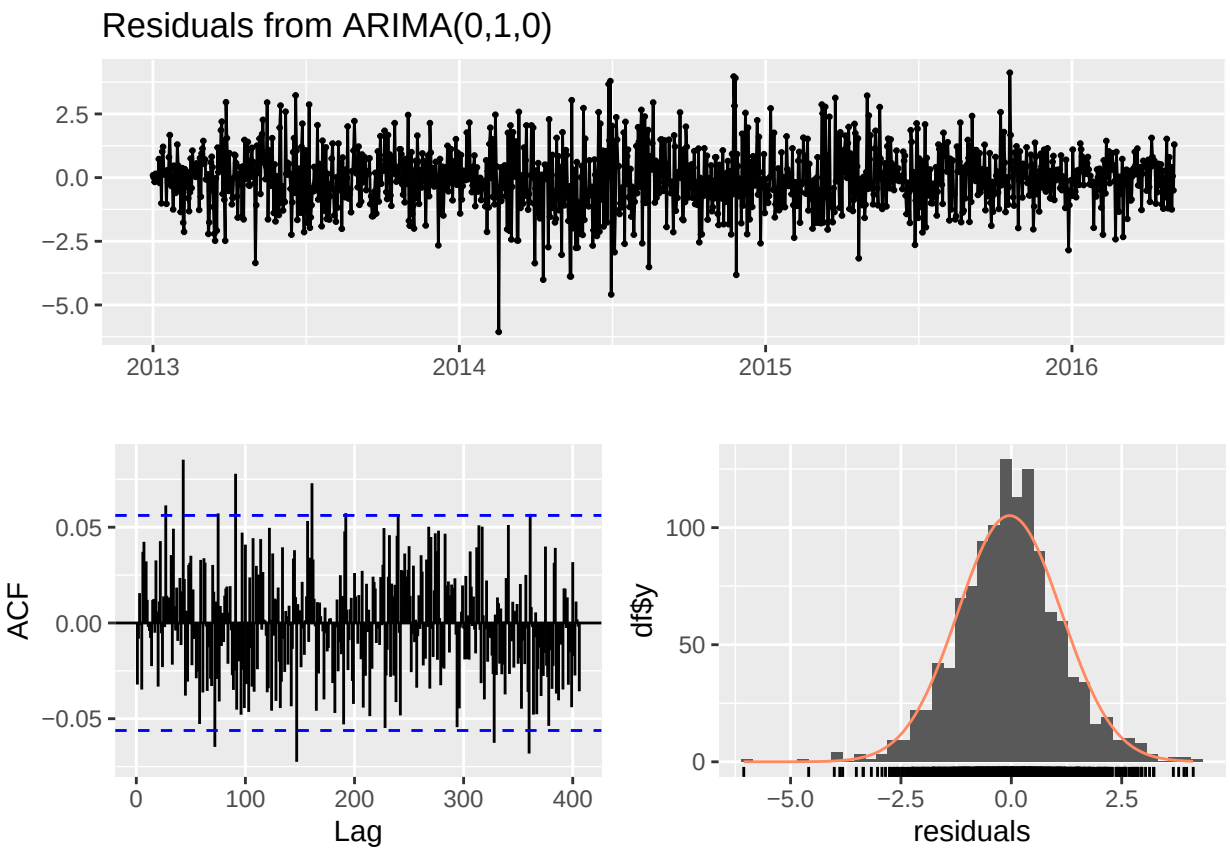
The Ljung-Box test for the residuals from the Holt-Winters multiplicative model yielded a p-value of 0.8647, which is well above the 0.05 threshold, indicating no significant autocorrelation in the residuals. This suggests that the multiplicative model adequately captures the seasonality and underlying structure of the data, with residuals behaving like white noise. The lack of significant autocorrelation supports the effectiveness of the multiplicative model in fitting the time series, indicating that it might be a better choice in comparison to the additive model based on these residuals.

(c) SARIMA Model

```
# Auto ARIMA with Seasonal Component
sarima_model <- auto.arima(oil_ts, seasonal = TRUE)

# Forecast
sarima_forecast <- forecast(sarima_model, h = 12)
plot(sarima_forecast)
```




```
# Check residuals
checkresiduals(sarima_forecast)
```



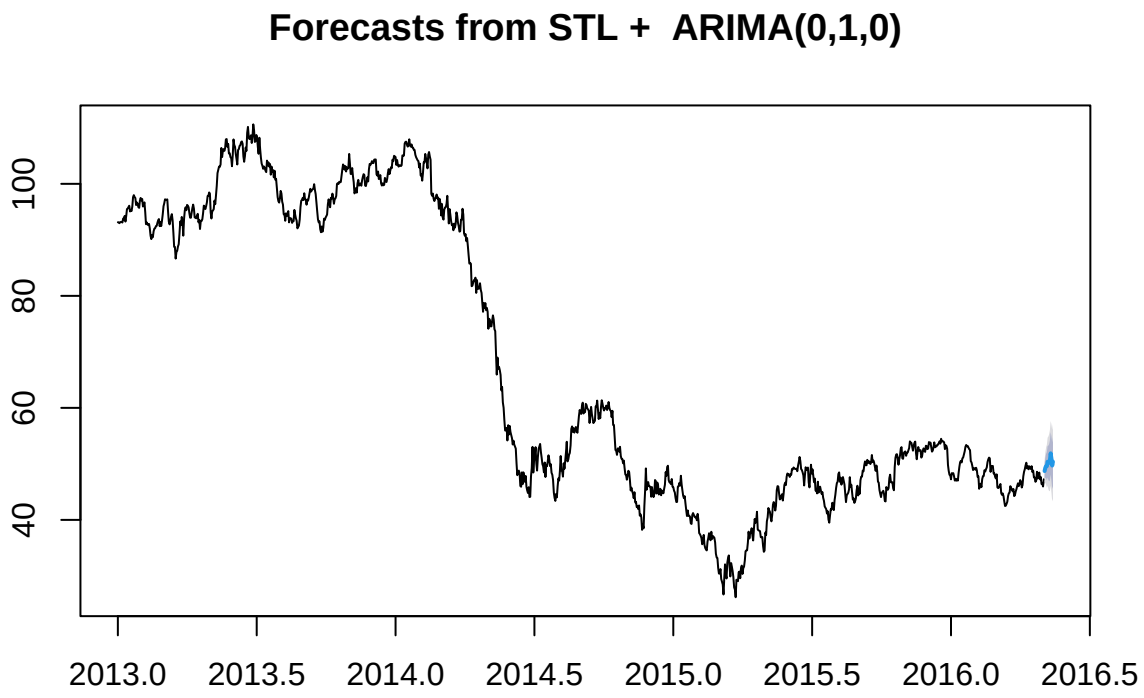

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 262.62, df = 244, p-value = 0.197
##
## Model df: 0.   Total lags used: 244
```

The Ljung-Box test for the residuals from the ARIMA(0,1,0) model yielded a p-value of 0.197, which is well above the typical significance threshold of 0.05. This indicates that there is no significant autocorrelation in the residuals, suggesting that the model adequately captures the underlying patterns in the data. The residuals appear to behave like white noise, meaning the model has effectively accounted for the time series dynamics, and no further improvements in the model specification are needed.

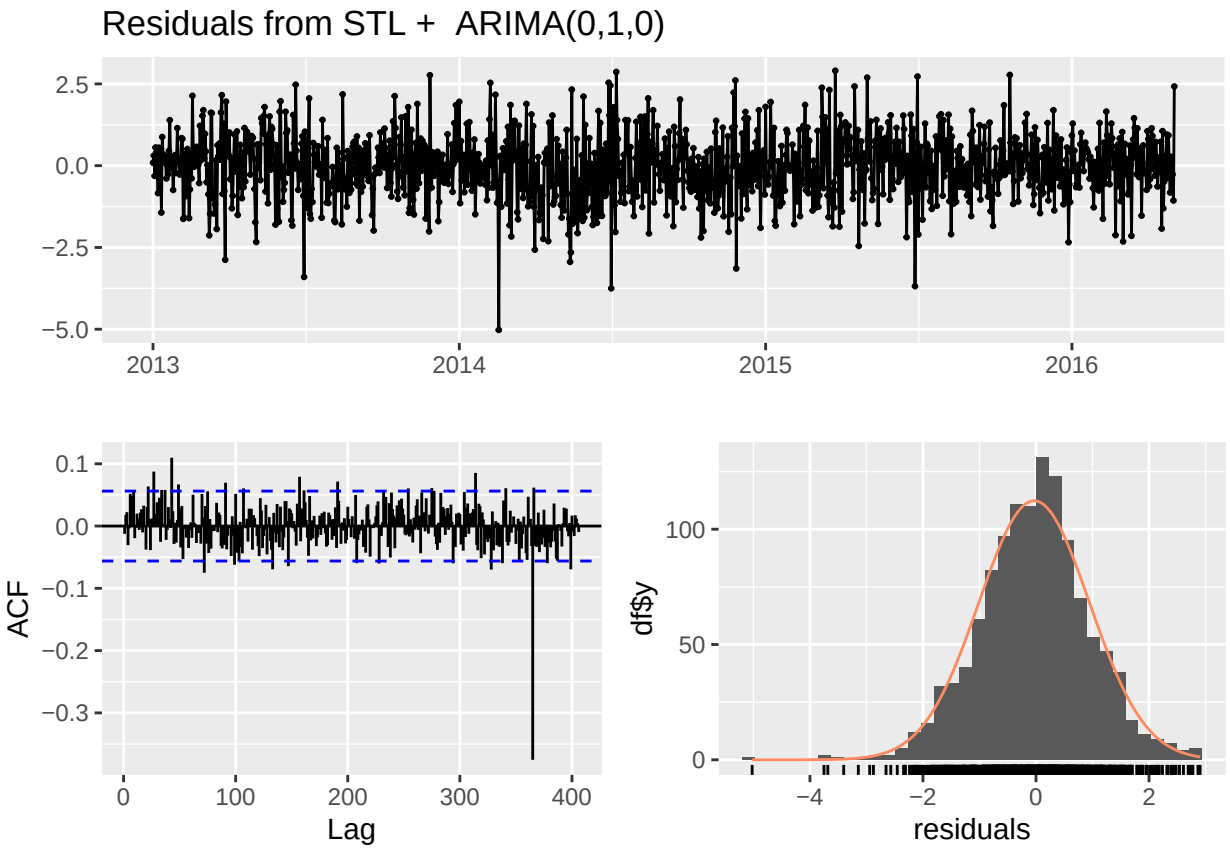
Additionally, the Auto ARIMA model with a seasonal component was fitted to the data, followed by forecasting for the next 12 periods and a residual check using checkresiduals(). This further confirms that the SARIMA model adequately handles both seasonality and trend in the data, with the residuals showing no significant autocorrelation.

(d) SARIMA Model with STLF Function

```r
# STL Decomposition and Forecast
stl_forecast <- stlf(oil_ts, h = 12, method = "arima")  # Use ARIMA for forecasting
plot(stl_forecast)
```




```
# Residual diagnostics
checkresiduals(stl_forecast)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ARIMA(0,1,0)
## Q* = 329.32, df = 244, p-value = 0.0002194
##
## Model df: 0.   Total lags used: 244
```

The Ljung-Box test for the residuals from the STL decomposition combined with ARIMA(0,1,0) model yielded a p-value of 0.0002194, which is highly significant and below the typical threshold of 0.05. This indicates the presence of significant autocorrelation in the residuals, suggesting that the model has not fully captured all the underlying patterns in the data. Therefore, the residuals are not behaving like white noise, and the model might need further refinement, such as adjustments to the ARIMA parameters or additional seasonal components.

Following this, the STL decomposition with ARIMA method was applied to forecast the next 12 periods, and residual diagnostics were performed using checkresiduals() to evaluate the adequacy of the model fit.

Step 8. Comparing the models based on AIC, BIC, Ljung-Box p-value and RMSE

```
suppressWarnings({
# Fit ETS model
ets_model <- ets(oil_ts)
```

```r
# Calculate metrics
rmse_ets <- sqrt(mean((oil_ts - fitted(ets_model))^2, na.rm = TRUE))  # RMSE
aic_ets <- AIC(ets_model)  # AIC
bic_ets <- BIC(ets_model)  # BIC

# Ljung-Box test
ljung_ets <- Box.test(residuals(ets_model), lag = 20, type = "Ljung-Box")

# Print results
cat("ETS Model:\n")
cat("RMSE:", rmse_ets, "\nAIC:", aic_ets, "\nBIC:", bic_ets, "\nLjung-Box p-value:", ljung_ets$p.value,
})
```

```
## ETS Model:
## RMSE: 1.176176
## AIC: 9055.133
## BIC: 9070.448
## Ljung-Box p-value: 0.9349022
```

```r
# Fit Holt-Winters Additive Model
hw_model <- HoltWinters(oil_ts, seasonal = "additive")

# Forecast
hw_forecast <- forecast(hw_model, h = 12)

# Calculate RMSE
rmse_hw <- sqrt(mean((oil_ts - fitted(hw_model))^2, na.rm = TRUE))  # RMSE

# For AIC and BIC
n_hw <- length(oil_ts)
rss_hw <- sum((residuals(hw_model))^2, na.rm = TRUE)  # Residual sum of squares
aic_hw <- n_hw * log(rss_hw / n_hw) + 2 * 3  # 3 params: alpha, beta, gamma
bic_hw <- n_hw * log(rss_hw / n_hw) + log(n_hw) * 3

# Ljung-Box test
ljung_hw <- Box.test(residuals(hw_model), lag = 20, type = "Ljung-Box")

# Print results
cat("Holt-Winters Additive Model:\n")
```

```
## Holt-Winters Additive Model:
```

```r
cat("RMSE:", rmse_hw, "\nAIC:", aic_hw, "\nBIC:", bic_hw, "\nLjung-Box p-value:", ljung_hw$p.value, "\n
```

```
## RMSE: 40.72244
## AIC: 1011.64
## BIC: 1026.955
## Ljung-Box p-value: 0.009803705
```

```r
# Fit Holt-Winters Multiplicative Model
hw_model_multiplicative <- HoltWinters(oil_ts, seasonal = "multiplicative")

# Forecast
hw_forecast_multiplicative <- forecast(hw_model_multiplicative, h = 12)

# Calculate RMSE
rmse_hw_multiplicative <- sqrt(mean((oil_ts - fitted(hw_model_multiplicative))^2, na.rm = TRUE))  # RMS

# For AIC and BIC
n_hw_mul <- length(oil_ts)
rss_hw_mul <- sum((residuals(hw_model_multiplicative))^2, na.rm = TRUE)  # Residual sum of squares
aic_hw_mul <- n_hw_mul * log(rss_hw_mul / n_hw_mul) + 2 * 3  # 3 params: alpha, beta, gamma
bic_hw_mul <- n_hw_mul * log(rss_hw_mul / n_hw_mul) + log(n_hw_mul) * 3

# Ljung-Box test
ljung_hw_mul <- Box.test(residuals(hw_model_multiplicative), lag = 20, type = "Ljung-Box")

# Print results
cat("Holt-Winters Multiplicative Model:\n")
```

```
## Holt-Winters Multiplicative Model:
```

```r
cat("RMSE:", rmse_hw_multiplicative, "\nAIC:", aic_hw_mul, "\nBIC:", bic_hw_mul, "\nLjung-Box p-value:"
```

```
## RMSE: 40.62907
## AIC: 581.9015
## BIC: 597.2164
## Ljung-Box p-value: 0.3246196
```

```r
# Fit Seasonal ARIMA Model
sarima_model <- auto.arima(oil_ts, seasonal = TRUE)

# Calculate metrics
rmse_sarima <- sqrt(mean((oil_ts - fitted(sarima_model))^2, na.rm = TRUE))  # RMSE
aic_sarima <- AIC(sarima_model)  # AIC
bic_sarima <- BIC(sarima_model)  # BIC

# Ljung-Box test
ljung_sarima <- Box.test(residuals(sarima_model), lag = 20, type = "Ljung-Box")

# Print results
cat("Seasonal ARIMA Model:\n")
```

```
## Seasonal ARIMA Model:
```

```r
cat("RMSE:", rmse_sarima, "\nAIC:", aic_sarima, "\nBIC:", bic_sarima, "\nLjung-Box p-value:", ljung_sari
```

```
## RMSE: 1.176755
## AIC: 3852.849
## BIC: 3857.953
## Ljung-Box p-value: 0.8994111
```

```r
# STL decomposition with ARIMA
stl_forecast <- stlf(oil_ts, h = 12, method = "arima")

# Calculate RMSE
fitted_stl <- oil_ts - residuals(stl_forecast$model)
rmse_stl <- sqrt(mean((oil_ts - fitted_stl)^2, na.rm = TRUE))  # RMSE

# AIC and BIC
aic_stl <- AIC(stl_forecast$model)  # AIC
bic_stl <- BIC(stl_forecast$model)  # BIC

# Ljung-Box test
ljung_stl <- Box.test(residuals(stl_forecast$model), lag = 20, type = "Ljung-Box")

# Print results
cat("STL with ARIMA Model:\n")
```

```
## STL with ARIMA Model:
```

```r
cat("RMSE:", rmse_stl, "\nAIC:", aic_stl, "\nBIC:", bic_stl, "\nLjung-Box p-value:", ljung_stl$p.value,
```

```
## RMSE: 0.9799459
## AIC: 3407.38
## BIC: 3412.484
## Ljung-Box p-value: 0.5681022
```

Based on the provided metrics, the STL with ARIMA Model has the lowest RMSE of 0.9799459, which indicates that it provides the best fit in terms of forecasting accuracy among all the models. This is followed closely by the ETS Model (RMSE = 1.176176), which also performs well in terms of forecast accuracy. In contrast, the Holt-Winters Additive Model (both versions) shows significantly higher RMSE values (40.72244 and 40.62907), indicating poorer accuracy compared to the other models.

Additionally, the Ljung-Box p-value values suggest that residuals from the STL with ARIMA Model (p-value = 0.5681022) and Seasonal ARIMA Model (p-value = 0.8994111) exhibit no significant autocorrelation, meaning these models adequately capture the underlying patterns in the data.

Thus, the STL with ARIMA Model is the preferred model based on the lowest RMSE, along with favorable residual diagnostics.