Stats-Project-Phase-02

Team

2025-04-14

Loading the necessary Libraries :

```
library(readr)
library(dplyr)
```

```
##
## 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
```

```
library(tidyr)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.4.3

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

```
library(tseries)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

```r
library(imputeTS)
```

```
## Warning: package 'imputeTS' was built under R version 4.4.3
```

```
##
## Attaching package: 'imputeTS'
```

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

1.0 Loading the dataset

```r
oil_data <- read_csv("oil.csv")
```

```
## Rows: 1218 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): dcoilwtico
## date (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(oil_data)
```

```
## # A tibble: 6 x 2
##   date       dcoilwtico
##   <date>          <dbl>
## 1 2013-01-01       NA
## 2 2013-01-02       93.1
## 3 2013-01-03       93.0
## 4 2013-01-04       93.1
## 5 2013-01-07       93.2
## 6 2013-01-08       93.2
```

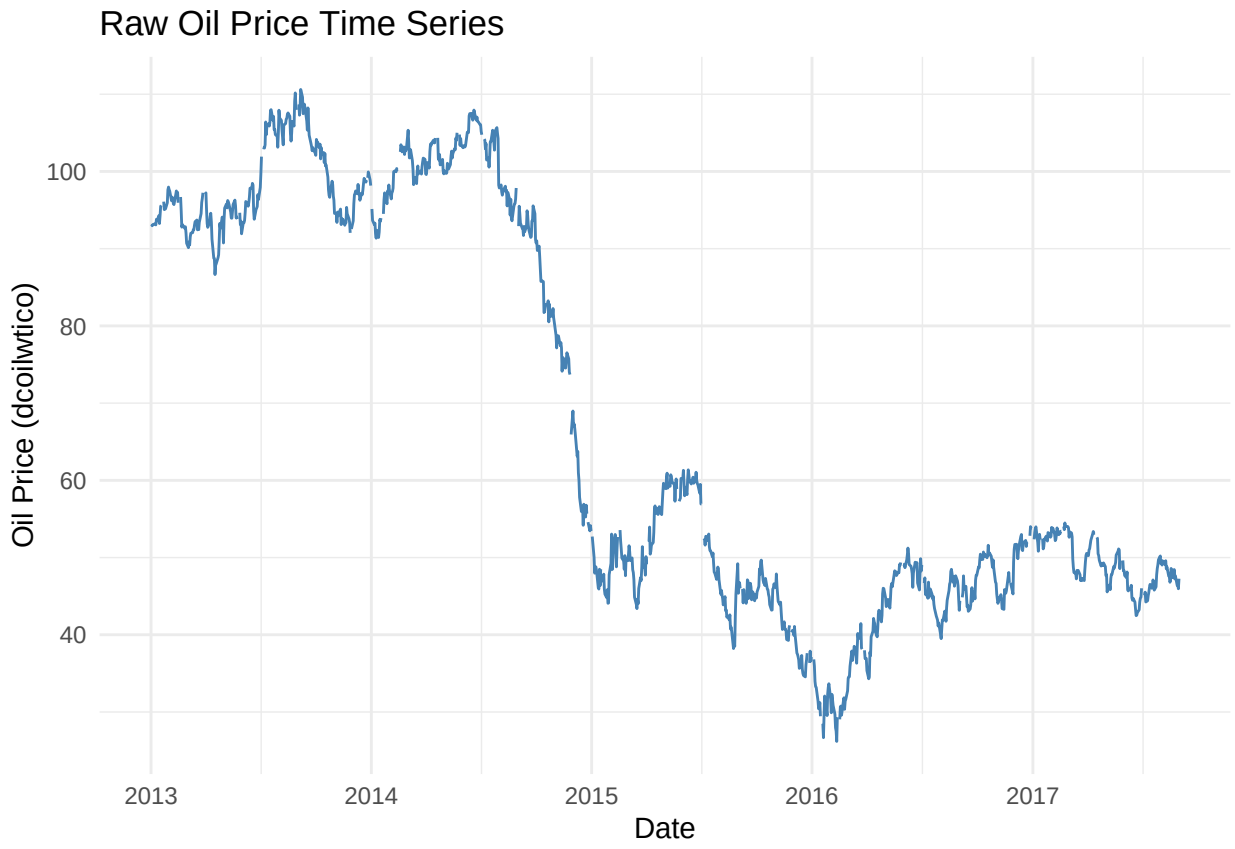
```r
str(oil_data)
```

```
## spc_tbl_ [1,218 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date      : Date[1:1218], format: "2013-01-01" "2013-01-02" ...
##  $ dcoilwtico: num [1:1218] NA 93.1 93 93.1 93.2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   date = col_date(format = ""),
##   ..   dcoilwtico = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

2.0 Plotting the Time Series Model :

```
ggplot(oil_data, aes(x = date, y = dcoilwtico)) +
  geom_line(color = "steelblue") +
  labs(
    title = "Raw Oil Price Time Series",
    x = "Date",
    y = "Oil Price (dcoilwtico)"
  ) +
  theme_minimal()
```

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



3.0 - MIssing data check and fill :

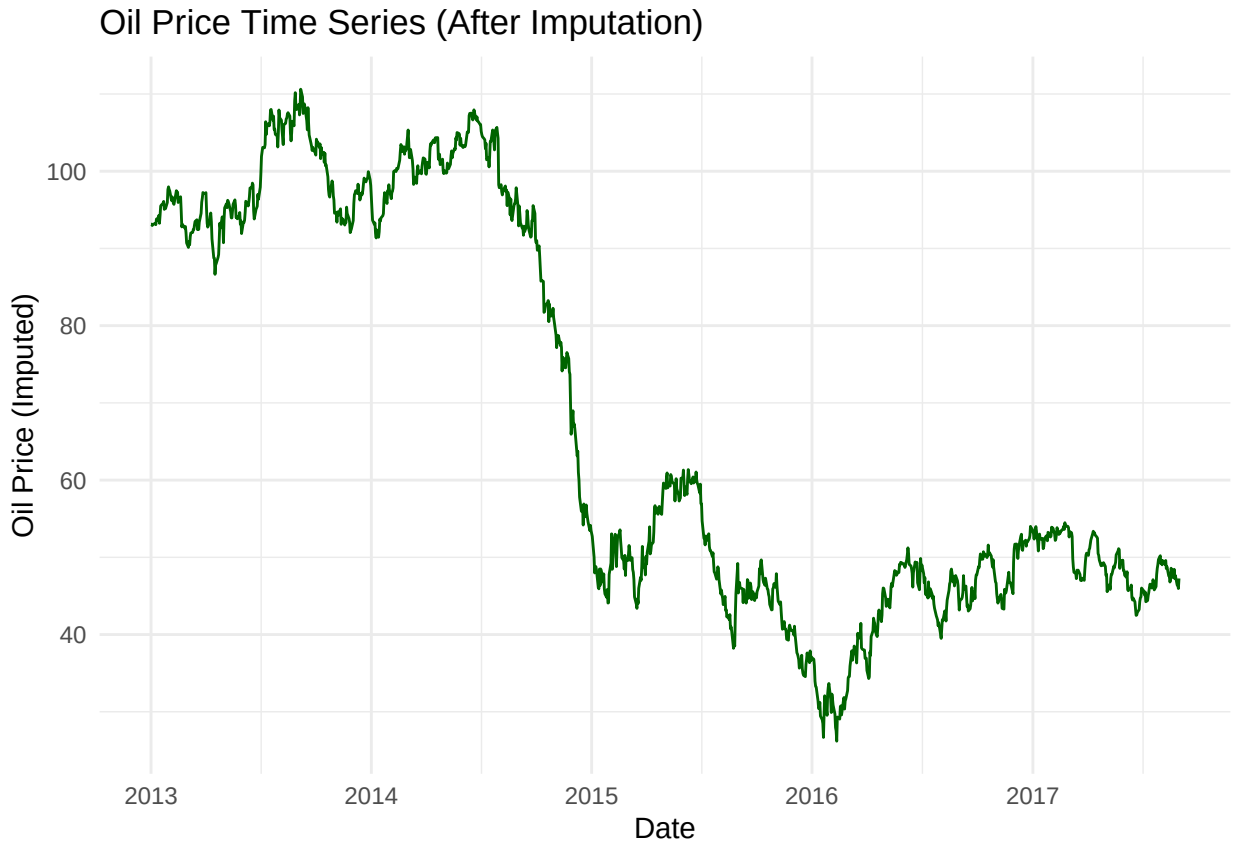
```
# Use of  linear interpolation to fill missing data
oil_data$imputed_dcoilwtico <- na_interpolation(oil_data$dcoilwtico, option = "linear")
```

Reason to choose linear interpolation : -

We chose **linear interpolation** for imputing missing values because it is simple, preserves the trend of the data, and is widely recommended for time series with small gaps. This method estimates missing values by connecting the nearest known values with a straight line, ensuring a smooth and realistic imputation.

4.0 Time series plot with imputed data and trend and/or seasonality check :

```
ggplot(oil_data, aes(x = date, y = imputed_dcoilwtico)) +
  geom_line(color = "darkgreen") +
  labs(
    title = "Oil Price Time Series (After Imputation)",
    x = "Date",
    y = "Oil Price (Imputed)"
  ) +
  theme_minimal()
```



4.1 Observation about a trend and/or seasonality in the data : -

Based on the two plots you provided:

Trend :- There is a clear downward trend in oil prices from mid-2014 to early 2016, where prices drop sharply from above $100 to below $40. After this period, the prices remain relatively low and stable, with some fluctuations but no strong upward or downward trend.

Seasonality:- There is no obvious seasonality (regular, repeating patterns at fixed intervals such as yearly or monthly) visible in the data. The fluctuations appear irregular and are not consistent in timing or magnitude.

4.2 Summary:

1. The data shows a strong downward trend from 2014 to 2016.
2. There is no clear evidence of seasonality in the oil price time series.

5.0 ETS models and Holt-Winters models :

5.1 ETS Models (Error, Trend, Seasonality)

Theoretical Aspects : -

1. ETS stands for Error, Trend, and Seasonality.
2. It is a family of exponential smoothing models for time series forecasting.

ETS models automatically select the best combination of:

1. Error type: Additive (A) or Multiplicative (M)
2. Trend type: None (N), Additive (A), Additive Damped (Ad), Multiplicative (M), Multiplicative Damped (Md)
3. Seasonality type: None (N), Additive (A), Multiplicative (M)

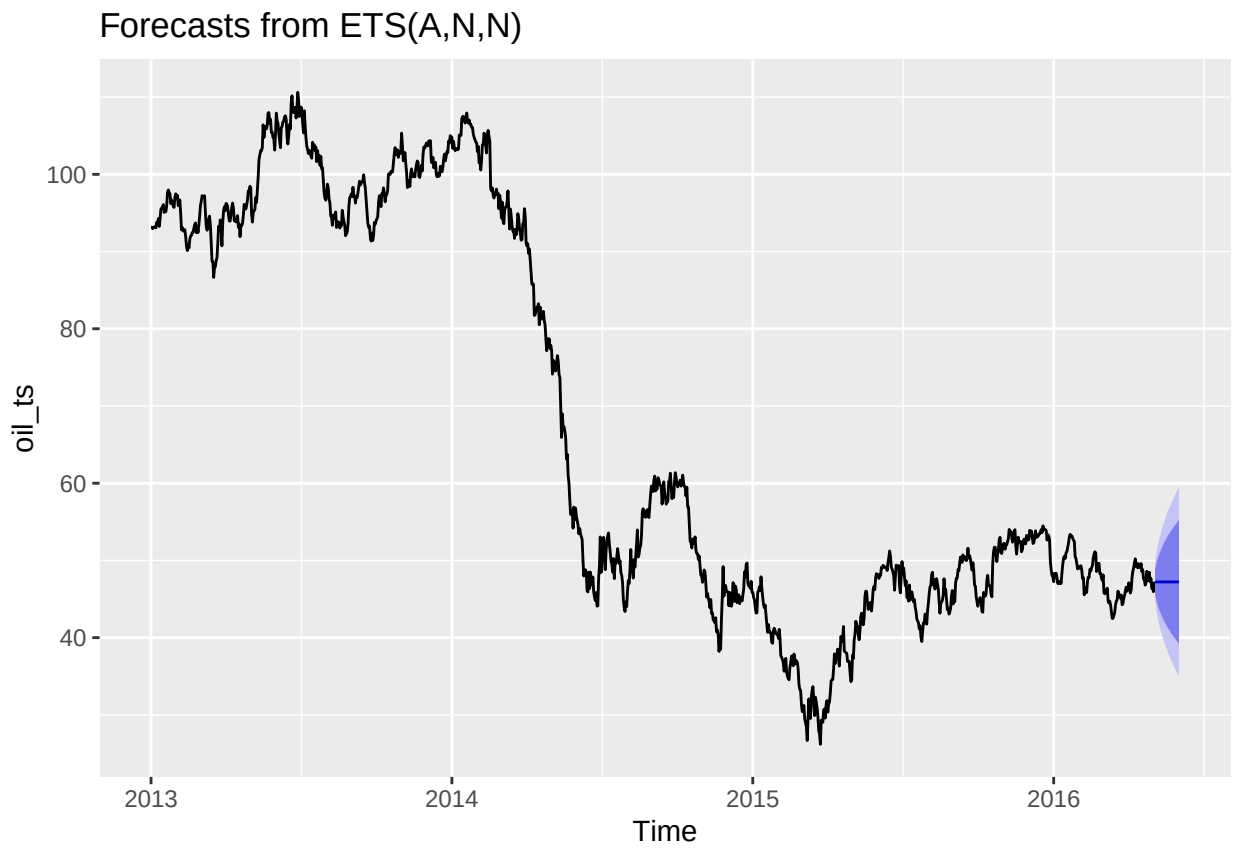
The model is denoted as ETS(Error, Trend, Seasonality), e.g.as : - ETS(A,Ad,N).

Strengths: Handles trend and seasonality, automatic model selection, good for univariate time series.

```
oil_ts <- ts(oil_data$imputed_dcoilwtico, start = c(2013, 1), frequency = 365)

# ETS model
ets_model <- ets(oil_ts)
```

```
## Warning in ets(oil_ts): I can't handle data with frequency greater than 24.
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.
```

```
# Forecasting next 30 days
ets_forecast <- forecast(ets_model, h = 30)

autoplot(ets_forecast)
```



5.2 Holt-Winters :-

Theoretical Aspects :-

Holt-Winters is a specific type of exponential smoothing model.

It extends simple exponential smoothing by adding:

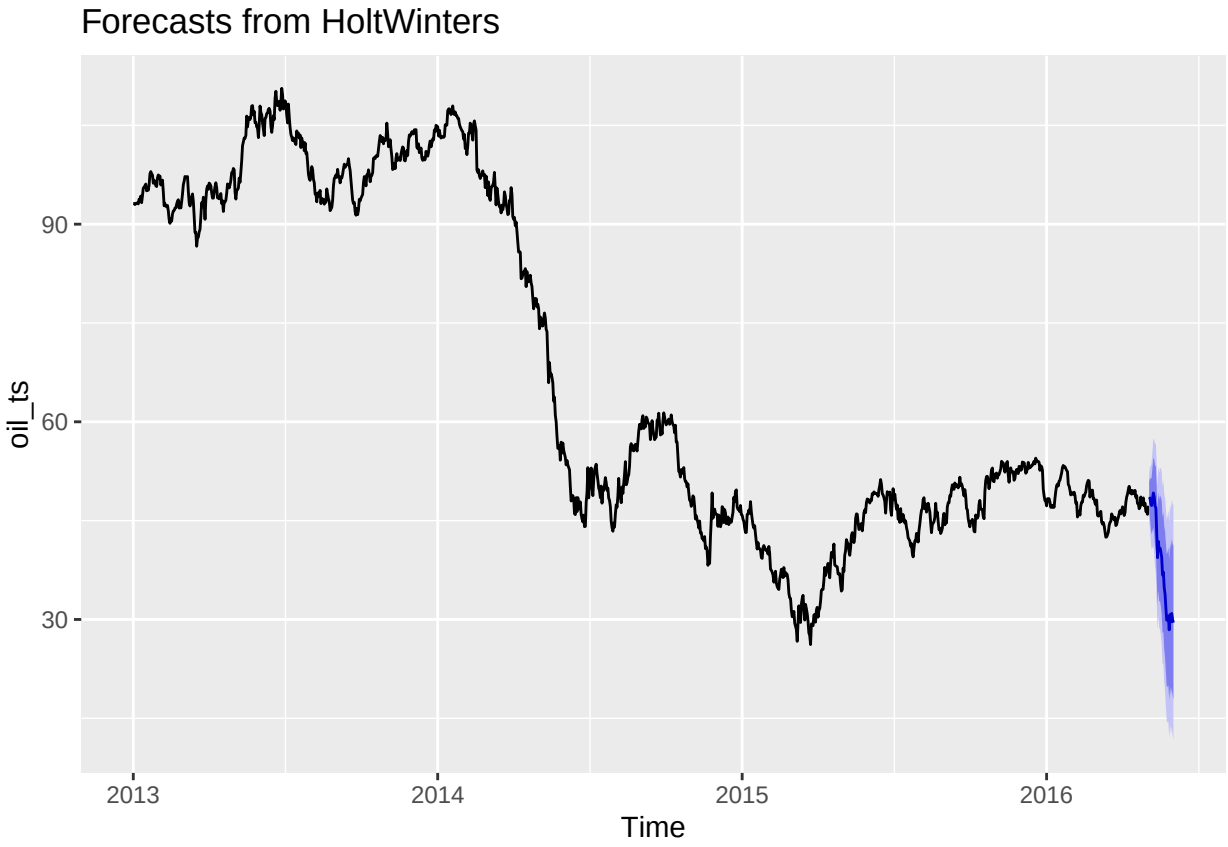
1. Trend (Holt's method)
2. Seasonality (Winters' method)

There are two main types:

1. Additive: For series with constant seasonal variation.
2. Multiplicative: For series with changing seasonal variation.

Strengths: Simple, effective for data with trend and/or seasonality.

```
#  Holt-Winters model
hw_model <- HoltWinters(oil_ts, seasonal = "additive")

hw_forecast <- forecast(hw_model, h = 30)
autoplot(hw_forecast)
```



5.3 When to Use Each Model

ETS :

Use when you want automatic selection of the best exponential smoothing model, especially if you're unsure about the presence of trend/seasonality.

Holt-Winters:

Use when you know your data has trend and/or seasonality and want a straightforward approach.

6.0 Suitable Model suggestion for given dataset

The oil price time series has a **clear downward trend** (2014-2016), but no seasonality. **Holt's Linear Trend Method (ETS(A,A,N))**, an exponential smoothing model that incorporates additive trend but not seasonality, is the best-suited model. Holt's Linear Trend Method extends Simple Exponential Smoothing, as it considers the trend components and is perfectly suited for a trending time series.

7.0 Models and their adequacy.

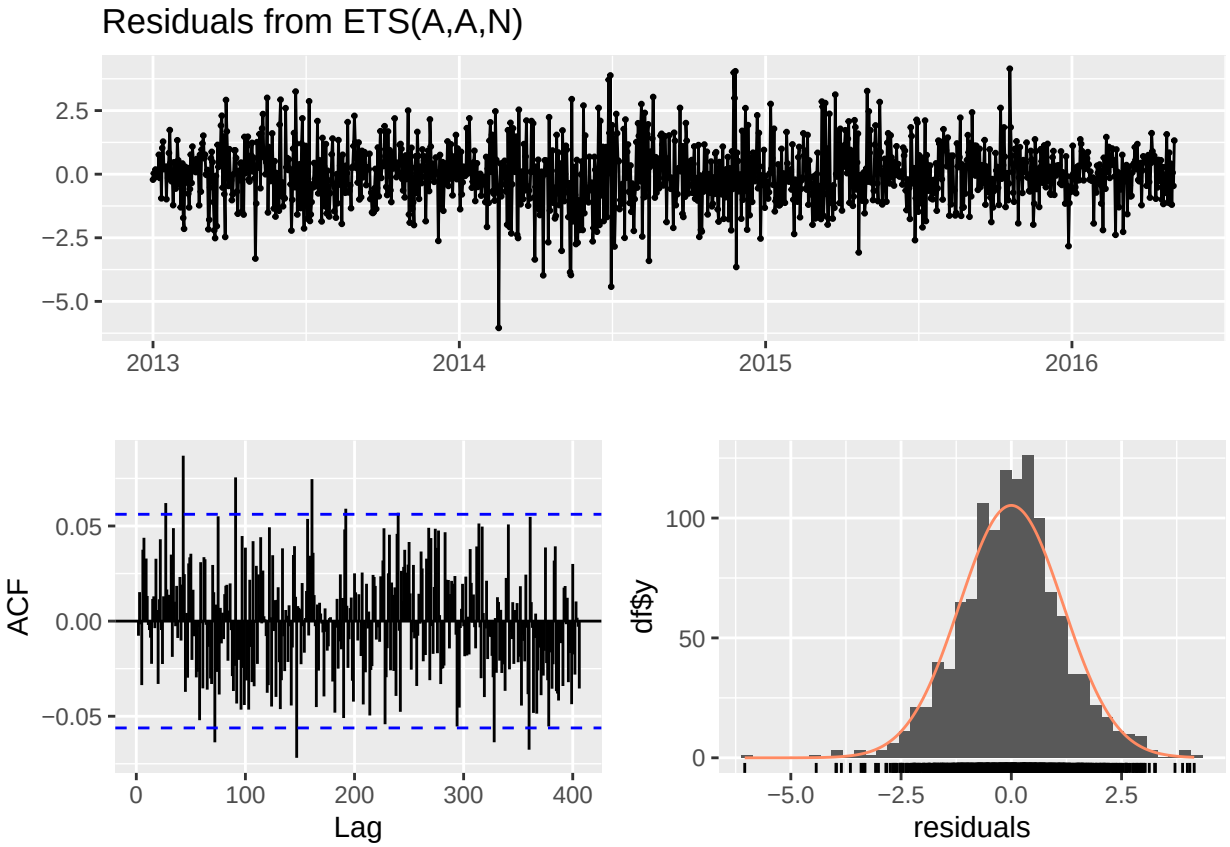
7.1 Holt's Linear Trend Method :

```
oil_ts <- ts(oil_data$imputed_dcoilwtico, start = c(2013, 1), frequency = 365)

# Fit the ETS(A,A,N) model
holt_model <- ets(oil_ts, model = "AAN")

summary(holt_model)
```

```
## ETS(A,A,N)
##
## Call:
## ets(y = oil_ts, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.9676
##     beta  = 1e-04
##
##   Initial states:
##     l = 93.3935
##     b = -0.0379
##
##   sigma:  1.1775
##
##      AIC     AICc      BIC
## 9057.952 9058.002 9083.477
##
## Training set error measures:
##                        ME     RMSE       MAE         MPE     MAPE       MASE
## Training set 0.0002500013 1.175605 0.8948867 -0.01426733 1.546043 0.03255872
##                       ACF1
## Training set -5.143552e-05
```

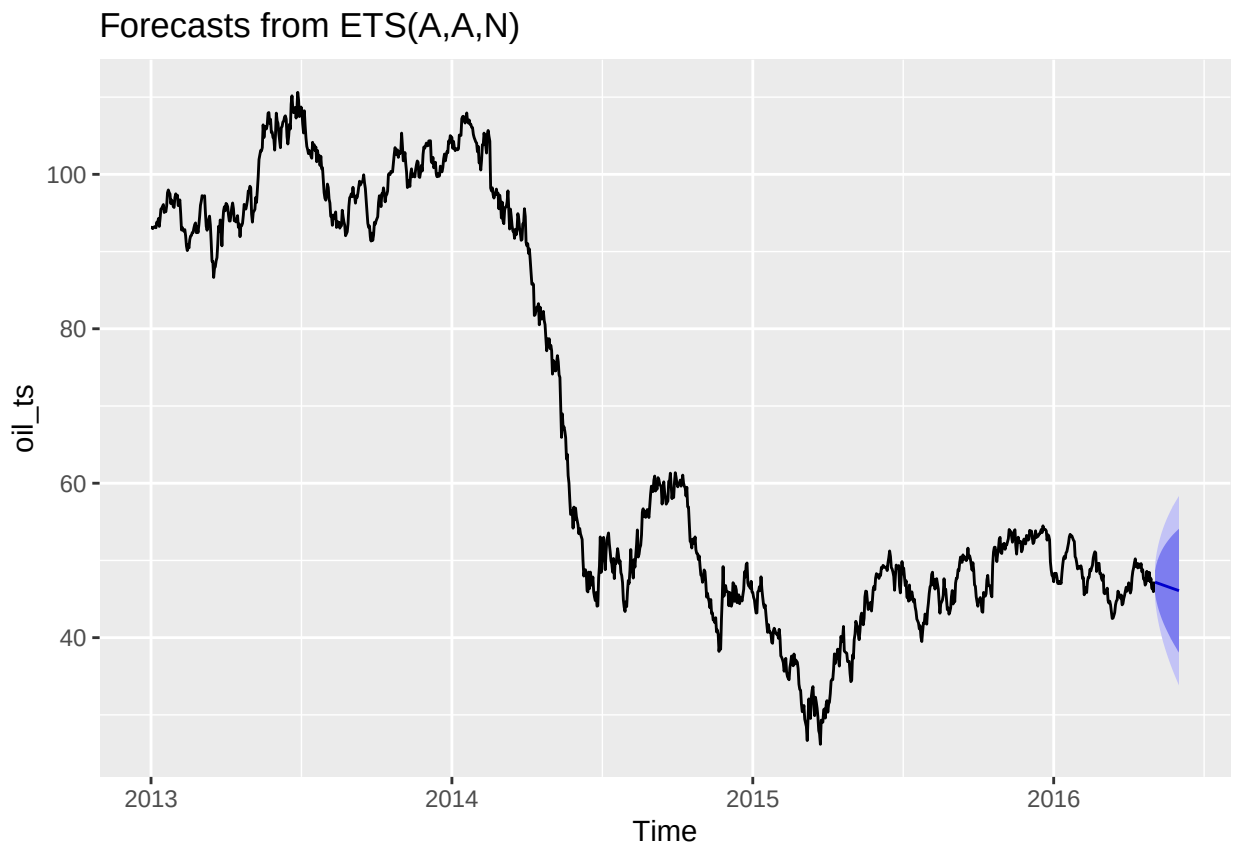
Model Adequacy

```
checkresiduals(holt_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 259.56, df = 244, p-value = 0.2358
##
## Model df: 0.   Total lags used: 244
```

```
holt_forecast <- forecast(holt_model, h = 30)
autoplot(holt_forecast)
```



7.2 ARIMA Model

Why ARIMA :

ARIMA (AutoRegressive Integrated Moving Average) is a flexible and powerful model for time series forecasting.

It can handle data with or without trend and seasonality (with proper parameterization).

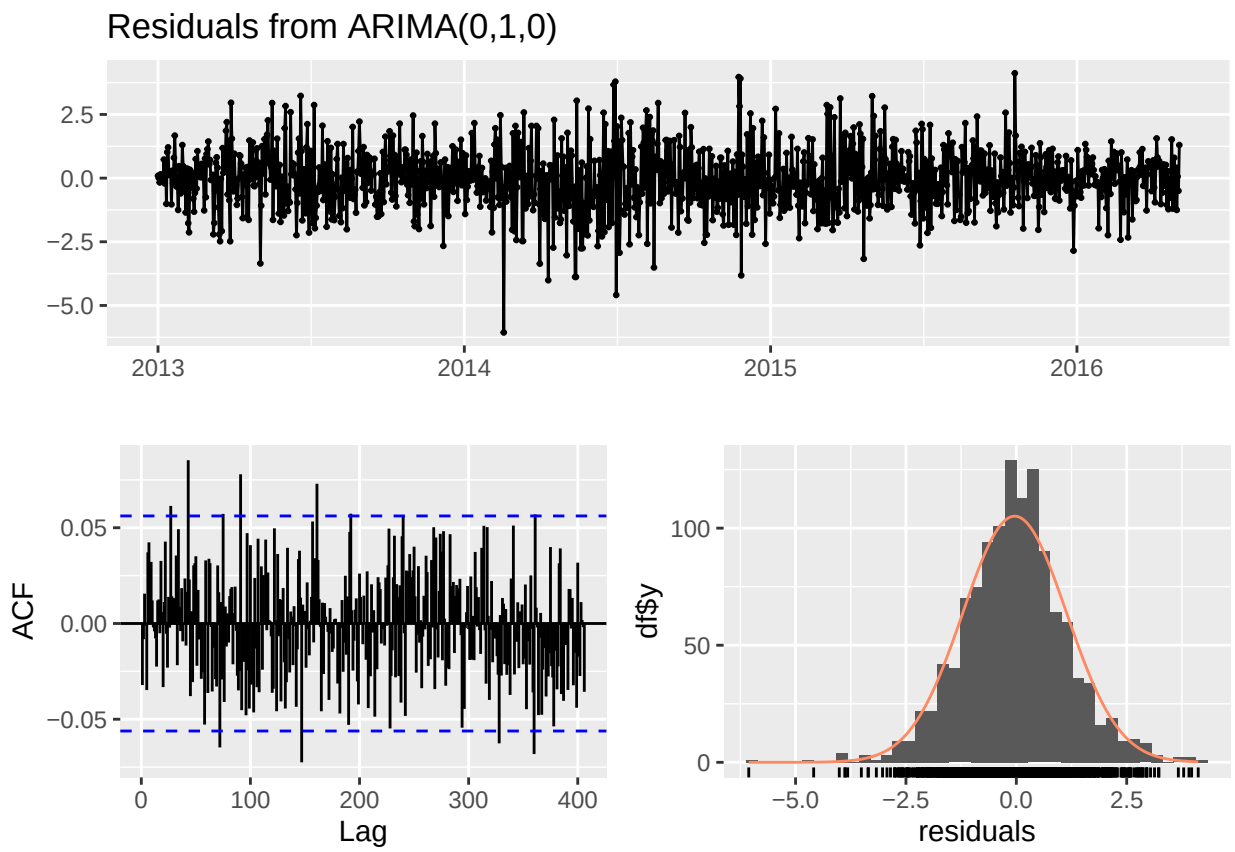
It is a standard benchmark for univariate time series forecasting and is often used alongside ETS for model comparison.

```
arima_model <- auto.arima(oil_ts)
summary(arima_model)
```

```
## Series: oil_ts
## ARIMA(0,1,0)
##
## sigma^2 = 1.386:  log likelihood = -1925.42
## AIC=3852.85   AICc=3852.85   BIC=3857.95
##
## Training set error measures:
##                      ME     RMSE       MAE         MPE     MAPE       MASE
```

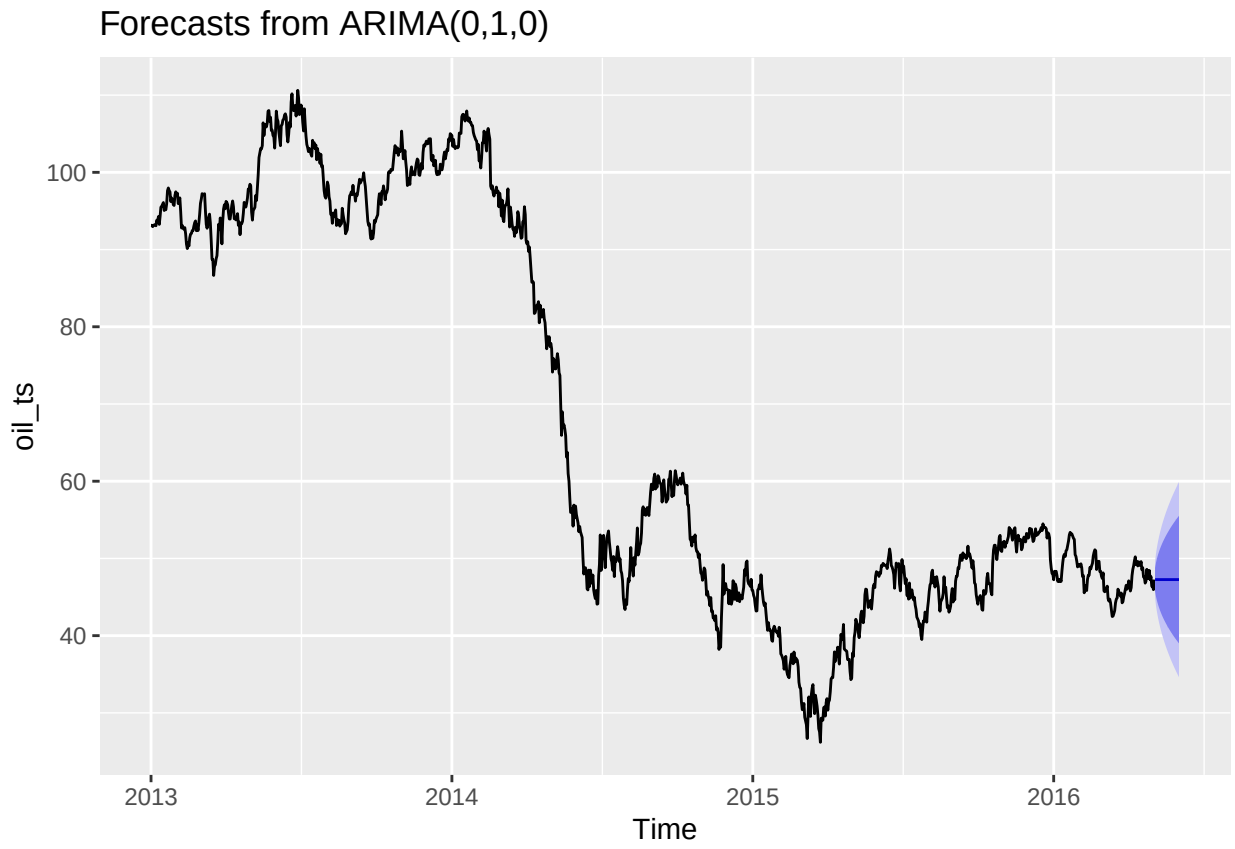
```
## Training set -0.03759184 1.176755 0.8949041 -0.07953015 1.547333 0.03255935
##                       ACF1
## Training set -0.03214235
```

```r
checkresiduals(arima_model)
```




```
##
##  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
```

```r
arima_forecast <- forecast(arima_model, h = 30)
autoplot(arima_forecast)
```



8.0 Models comparision based on performance by the metrics.

To compare the performance of the two models (ARIMA and ETS), we focus on the **Root Mean Squared Error (RMSE)**, which measures the average magnitude of the forecast errors. A lower RMSE indicates a better fit to the data.

ARIMA(0,1,0) Model —> RMSE: 1.1768 | AIC: 3852.85

ETS(A,A,N) Model —> RMSE: 1.17560

Interpretation : –

1. **ETS(A,A,N)** continues to show marginally better predictive accuracy (lower RMSE) than the ARIMA (0,1,0).
2. **ARIMA (0,1,0)** is the better model in terms of model efficiency (lower AIC).
3. Both models passed the following tests for residual diagnostics (Ljung-Box $p > 0.05$):

Recommendation : Prefer ETS(A,A,N)

because: - It explicitly models the observed downward trend. - It has slightly better forecast accuracy. - It has more interpretable smoothed forecasts.

```r
#comparison
comparison_df <- data.frame(
  Days_Ahead = 1:30,
  ARIMA = as.numeric(forecast(arima_model, h = 30)$mean),
  Holt_ETS = as.numeric(forecast(holt_model, h = 30)$mean)
)

ggplot(comparison_df, aes(x = Days_Ahead)) +
  geom_line(aes(y = ARIMA, color = "ARIMA(0,1,0)"), linewidth = 0.8) +
  geom_line(aes(y = Holt_ETS, color = "ETS(A,A,N)"), linewidth = 0.8, linetype = "dashed") +
  labs(
    title = "30-Day Forecast Comparison",
    x = "Days Ahead",
    y = "Oil Price",
    color = "Model"
  ) +
  scale_color_manual(values = c("ARIMA(0,1,0)" = "#0072B2", "ETS(A,A,N)" = "#D55E00")) +
  theme_minimal()
```