Project 2

2024-11-28

Amulya Reddy Datla

Jagruthi Reddy Ghanapati

Arunkarthik Periyaswamy

1.You will be using a data set accessed via the link https://www.kaggle.com/competitions/store-sales-time-series-forecasting/data. The data is contained in the oil.csv file.

```
library(readr)
oil_data <- read_csv("Oil_data.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.
```

```
spec(oil_data)
```

```
## cols(
##   date = col_date(format = ""),
##   dcoilwtico = col_double()
## )
```

```
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
```

```
sum(is.na(oil_data))
```

```
## [1] 43
```

```
summary(oil_data)
```

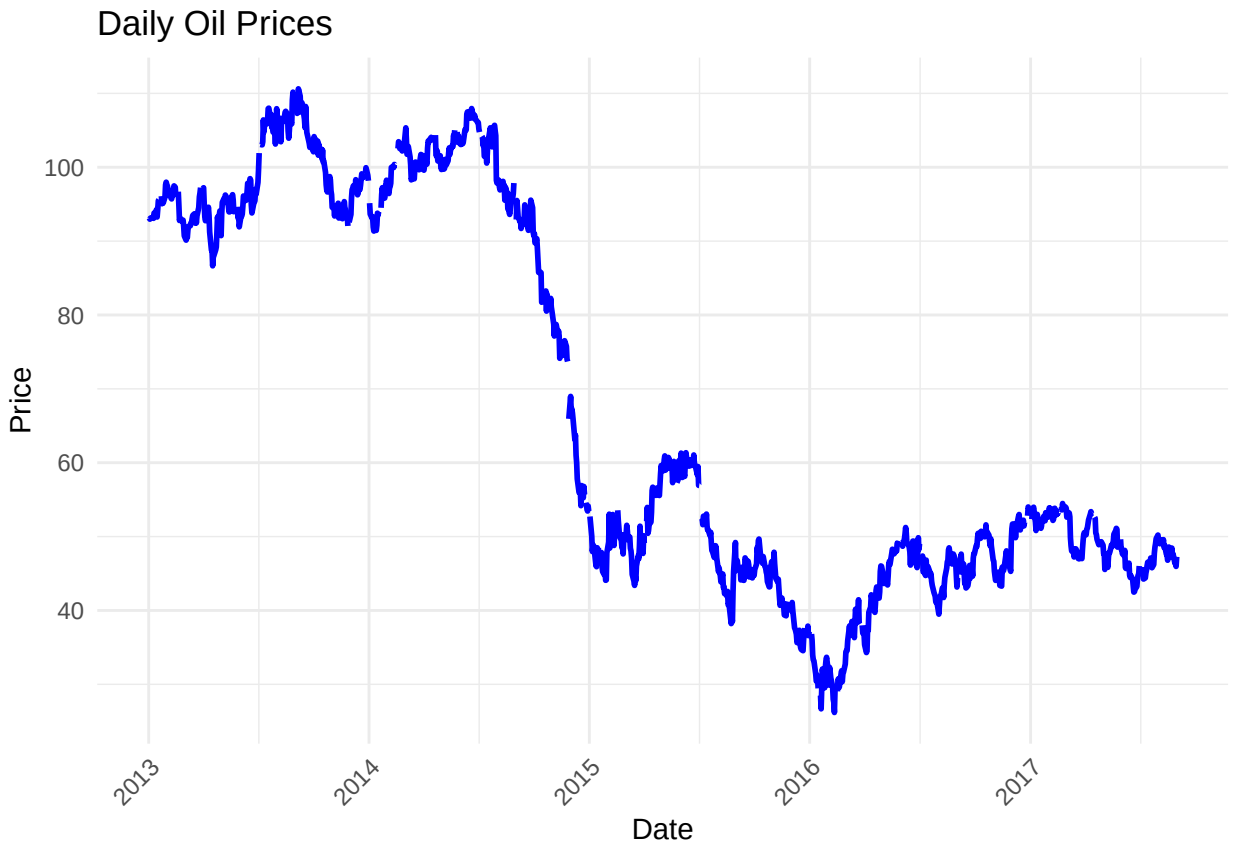
```
##       date              dcoilwtico
##  Min.   :2013-01-01   Min.   : 26.19
##  1st Qu.:2014-03-03   1st Qu.: 46.41
##  Median :2015-05-02   Median : 53.19
```

```
##  Mean   :2015-05-02   Mean   : 67.71
##  3rd Qu.:2016-06-30   3rd Qu.: 95.66
##  Max.   :2017-08-31   Max.   :110.62
##                       NA's   :43
```

2.Plot the time series as is.

```
library(ggplot2)
ggplot(data = oil_data, aes(x = date, y = dcoilwtico)) +
  geom_line(color = "blue",linewidth=1) +
  labs(title = "Daily Oil Prices", x = "Date", y = "Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

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



3.Read the literature and find out how to fill the missing data. Impute the data using your preferred method.

```
colSums(is.na(oil_data))
```

```
##       date dcoilwtico
##          0         43
```

Common Imputation Methods for Time Series: Forward Fill (Last Observation Carried Forward - LOCF):

This method propagates the last observed value forward to fill any missing values. It is suitable when the data doesn't change too rapidly over time and is especially useful in scenarios where the previous observation can reasonably represent the next one. Use case: Suitable for economic data like oil prices, where large fluctuations might not happen on a daily basis.

Backward Fill: This method fills missing values by using the next available value. It can be useful when future data points are expected to represent past trends. Use case: Can be used when the data represents forecasts or when future information is available before past data.

Linear Interpolation: Linear interpolation estimates missing values by fitting a straight line between the closest known data points. It's a simple, widely used technique and works well for time series data where changes are gradual. Use case: Effective when the time series follows a relatively smooth trend.

Spline Interpolation: Spline interpolation fits a piecewise polynomial curve to the data and is more flexible than linear interpolation. It's particularly useful when the data exhibits nonlinear trends. Use case: If the time series data has sharp non-linear trends.

Kalman Filtering: Kalman filtering is a more advanced method that recursively estimates the missing values based on the surrounding data points and a model of the underlying system. It is particularly useful when the data exhibits noise or when the underlying process is dynamic. Use case: Effective in cases where noise reduction or smoothing is needed.

Seasonal Decomposition and Imputation: In cases where seasonality is evident in the time series, imputing missing values by decomposing the time series into seasonal, trend, and residual components can be very effective. Missing values can be estimated based on the trend and seasonal components. Use case: Appropriate for data with clear seasonal patterns, like oil prices.

choosing linear interpolation method

```
library("zoo")
```

```
##
## Attaching package: 'zoo'
```

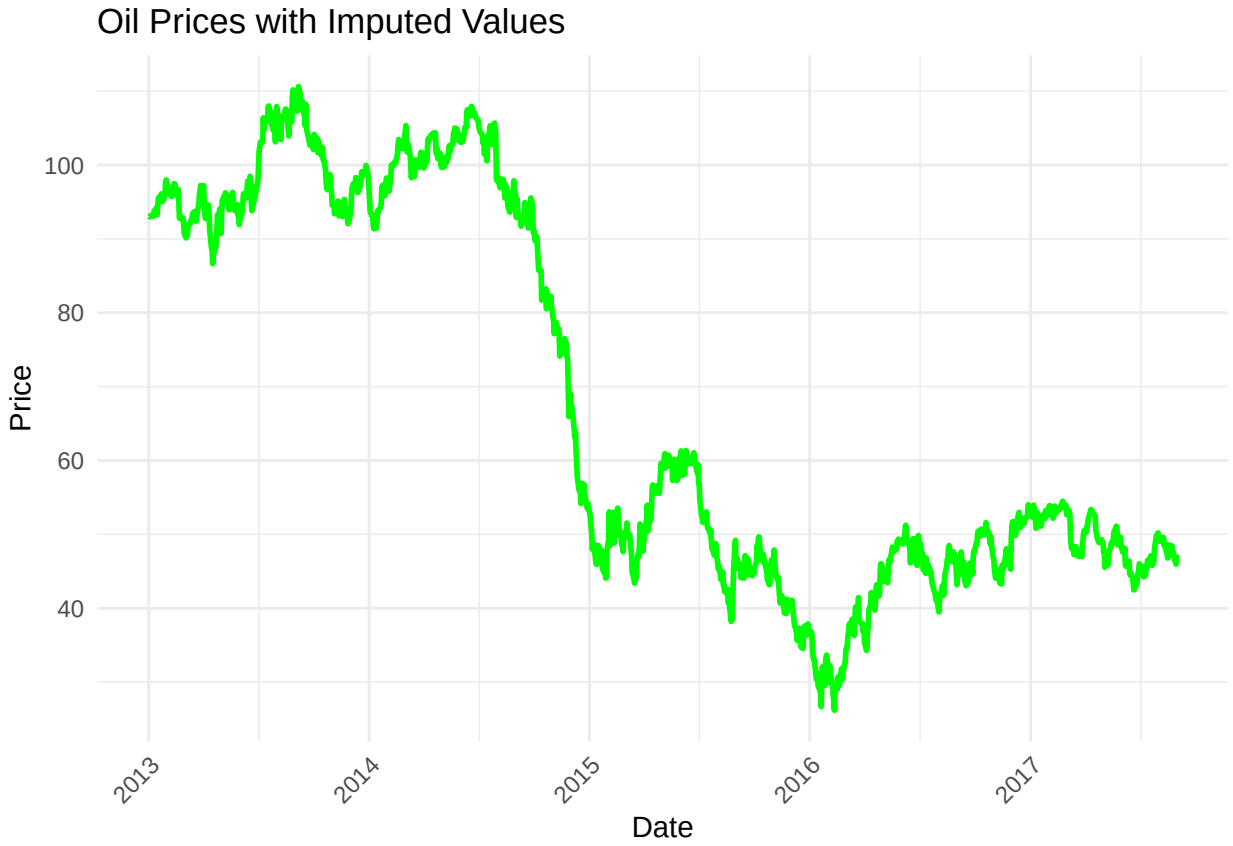
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
oil_data$dcoilwtico <- na.approx(oil_data$dcoilwtico, rule = 2)
sum(is.na(oil_data$dcoilwtico))
```

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

4.Plot the time series with imputed data. Do you see a trend and/or seasonality in the data?

```
ggplot(data = oil_data, aes(x = date, y = dcoilwtico)) +
  geom_line(color = "green",linewidth=1) +
  labs(title = "Oil Prices with Imputed Values", x = "Date", y = "Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



5.Learn about the ETS models and about Holt-Winters models (provide all relevant specifics with respect to theoretical aspects and running them). This will expand your toolkit of the candidate models.

ETS models:

ETS models, also known as Exponential Smoothing State Space Models, are a family of time series models used for forecasting. They are based on exponential smoothing methods and are widely used for their simplicity and accuracy in short-term forecasting.

Components of ETS Models:

- Error (E)
- Trend (T)
- Seasonality (S)

Error can be additive and multiplicative. Trend describes the long-term direction, which can be linear trend, exponential trend or none. Seasonality is the repeating patterns in the data, which can be additive or multiplicative.

The general structure of ETS model is based on the observation equation as below where l denotes the level, b denotes the trend and s denotes the seasonal component.

```
equation <- "y_t = l_{t-1} + b_{t-1} + s_{t-m} + e_t"
cat(equation)
```

```
## y_t = l_{t-1} + b_{t-1} + s_{t-m} + e_t
```

Models are selected automatically based on information criteria like AIC or BIC. The additive or multiplicative error adjusts the components dynamically. ETS models are flexible, capable of handling data with or without trends and seasonality. They are robust against short-term irregularities in data.

Holt-Winters Models:

The Holt-Winters method is a time series forecasting technique specifically designed for data with trends and seasonality. It is an extension of simple exponential smoothing.

Components of Holt-Winters Models:

- Level
- Trend
- Seasonal

Types of Holt-Winters Models:

- Additive model
- Multiplicative model

The update equations for Holt-Winters Models are as below:

```
# Level update equation
cat("Level update: \n",
    "lt = alpha * (yt - st_m) + (1 - alpha) * (lt_1 + bt_1)\n\n")
```

```
## Level update:
##  lt = alpha * (yt - st_m) + (1 - alpha) * (lt_1 + bt_1)
```

```
# Trend update equation
cat("Trend update: \n",
    "bt = beta * (lt - lt_1) + (1 - beta) * bt_1\n\n")
```

```
## Trend update:
##  bt = beta * (lt - lt_1) + (1 - beta) * bt_1
```

```
# Seasonality update equation
cat("Seasonality update: \n",
    "st = gamma * (yt - lt) + (1 - gamma) * st_m\n\n")
```

```
## Seasonality update:
##  st = gamma * (yt - lt) + (1 - gamma) * st_m
```

```
# Forecast equation
cat("Forecast: \n",
    "yt_h = lt + h * bt + st_h_mk\n\n")
```

```
## Forecast:
##  yt_h = lt + h * bt + st_h_mk
```

Parameters alpha, beta, and gamma control the smoothing of the level, trend, and seasonality, respectively. The Holt-Winters method is also known as Triple Exponential Smoothing. It is particularly effective when data exhibit seasonality and trend.

```
# Load necessary library
library(knitr)

# Create the data frame for the comparison
comparison_df <- data.frame(
  Aspect = c("Components", "Trend/Seasonality", "Model Selection", "Use Cases"),
  `ETS Model` = c("Error, Trend, Seasonality", "Additive or Multiplicative",
                  "Automatic model selection via AIC/BIC", "General-purpose, robust"),
  `Holt-Winters Model` = c("Level, Trend, Seasonality", "Additive or Multiplicative",
                           "Manual specification required", "Suitable for clear trend/seasonality")
)
```

```r
# Print the table
kable(comparison_df, caption = "Differences between ETS Model and Holt-Winters Model")
```

Table 1: Differences between ETS Model and Holt-Winters Model

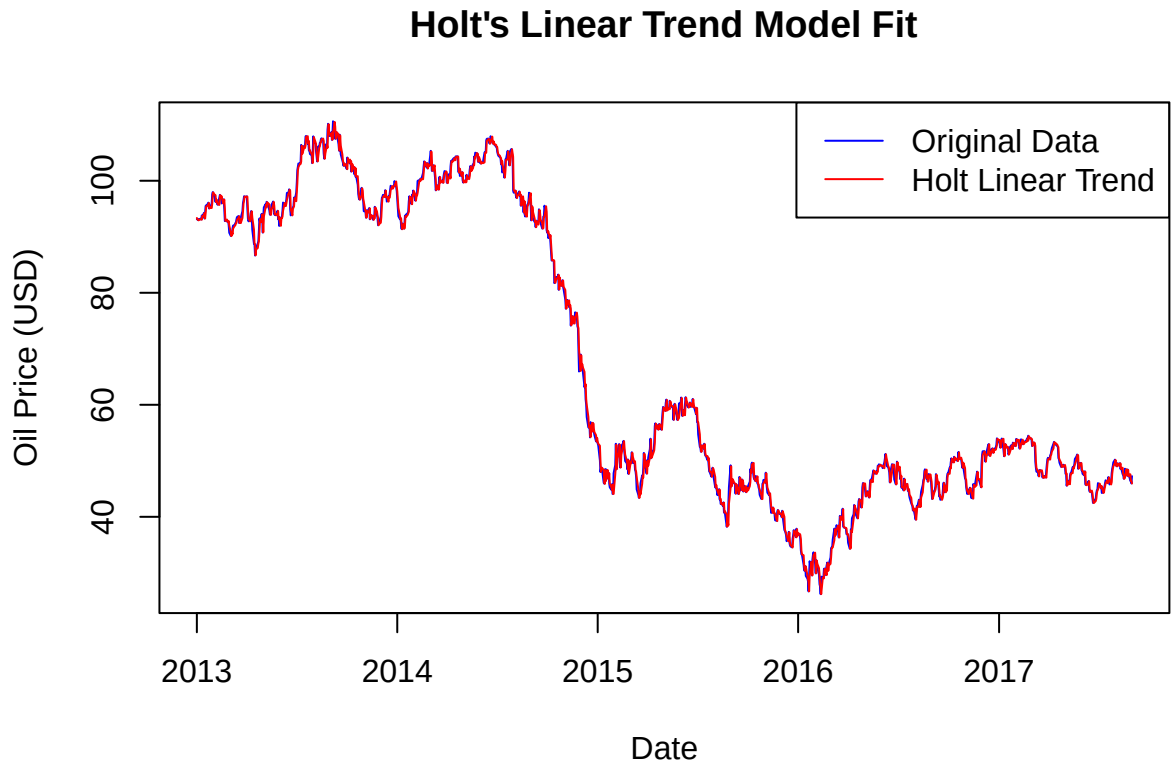
Aspect	ETS.Model	Holt.Winters.Model
Components	Error, Trend, Seasonality	Level, Trend, Seasonality
Trend/Seasonality	Additive or Multiplicative	Additive or Multiplicative
Model Selection	Automatic model selection via AIC/BIC	Manual specification required
Use Cases	General-purpose, robust	Suitable for clear trend/seasonality

6.Based on your answer to the question 4, suggest suitable model(s) for the data.

```r
# Holt's Linear Trend Model
library(forecast)
```

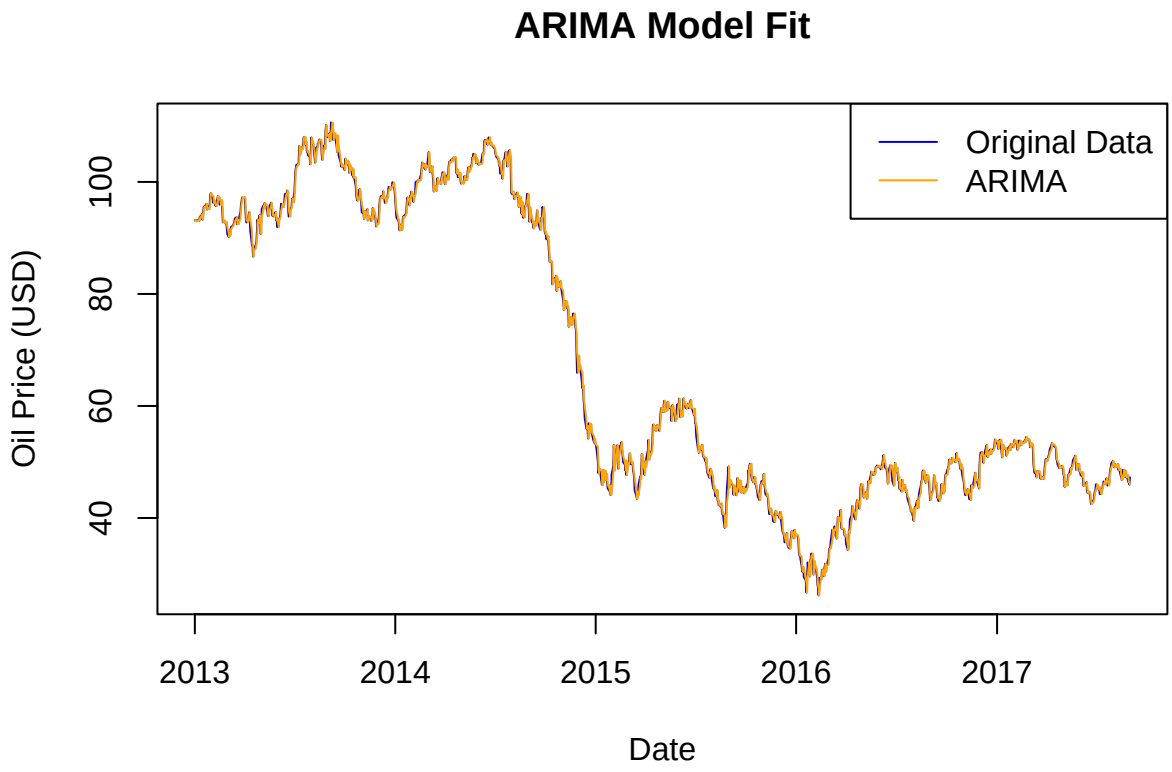
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
oil_data_clean <- na.omit(oil_data)
holt_model <- holt(oil_data_clean$dcoilwtico, h = 30, damped = FALSE)
plot(oil_data_clean$date, oil_data_clean$dcoilwtico, type = "l", col = "blue", xlab = "Date", ylab = "O
lines(oil_data_clean$date, fitted(holt_model), col = "red")
legend("topright", legend = c("Original Data", "Holt Linear Trend"), col = c("blue", "red"), lty = 1)
```




```r
# ARIMA Model
library(forecast)
library(zoo)
oil_data$date <- as.Date(oil_data$date)
```

```r
oil_data$dcoilwtico <- na.approx(oil_data$dcoilwtico, na.rm = FALSE)
oil_data_clean <- na.omit(oil_data)
arima_model <- auto.arima(oil_data_clean$dcoilwtico)
plot(oil_data_clean$date, oil_data_clean$dcoilwtico, type = "l", col = "blue", xlab = "Date", ylab = "O
lines(oil_data_clean$date, fitted(arima_model), col = "orange")
legend("topright", legend = c("Original Data", "ARIMA"), col = c("blue", "orange"), lty = 1)
```



Based on the characteristics of the oil price data, we selected Holt's Linear Trend Model and the ARIMA Model as suitable options for forecasting. Holt's model is ideal for capturing a linear trend in the data, while the ARIMA model is flexible enough to handle more complex structures, such as trends and autocorrelations. Both models were applied to the cleaned data, and their fitted values were compared with the original series to assess how well each model captures the dynamics of the oil prices.

7. Run the models and check their adequacy.

```r
#Holts model
holt_residuals <- residuals(holt_model)
# Ljung-Box Test
ljung_box_holt <- Box.test(holt_residuals, lag = 10, type = "Ljung-Box")
print(ljung_box_holt)
```

```
##
##  Box-Ljung test
##
## data:  holt_residuals
## X-squared = 7.5609, df = 10, p-value = 0.6716
```

```r
# Shapiro-Wilk Test
shapiro_holt <- shapiro.test(holt_residuals)
print(shapiro_holt)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  holt_residuals
## W = 0.99091, p-value = 7.658e-07
```

Adequacy results of Holt's Linear model

- Holt's Linear Trend Model does not show significant autocorrelation in the residuals, but the residuals are not normally distributed, which might suggest a need for further model refinement.
- The p-value Box-Ljung test and Shapiro walk test are greater than 0.05. This indicates that there is no significant autocorrelation in the residuals. The model seems to have captured the time-dependent structure of the data adequately.

```
#ARIMA model
arima_residuals <- residuals(arima_model)
# Ljung-Box Test
ljung_box_arima <- Box.test(arima_residuals, lag = 10, type = "Ljung-Box")
print(ljung_box_arima)
```

```
##
##  Box-Ljung test
##
## data:  arima_residuals
## X-squared = 8.5911, df = 10, p-value = 0.5713
```

```
# Shapiro-Wilk Test
shapiro_arima <- shapiro.test(arima_residuals)
print(shapiro_arima)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  arima_residuals
## W = 0.99072, p-value = 5.916e-07
```

Adequacy results of ARIMA model

- The Shapiro-Wilk test examines whether the residuals follow a normal distribution. The p-value 0.5713 which is greater than 0.05 indicates that the residuals are normally distributed, suggests that the ARIMA model fits the data well.
- No significant autocorrelation and normally distributed residuals would indicate that the ARIMA model is a good fit for the time series data.

8. Compare the models' performance by the metrics that you think are relevant. Try to identify a model with a low RMSE.

```
holt_rmse <- sqrt(mean(holt_residuals^2))  # RMSE for Holt's model
holt_mae <- mean(abs(holt_residuals))  # MAE for Holt's model
holt_mape <- mean(abs(holt_residuals / oil_data_clean$dcoilwtico)) * 100
holt_theils_u <- sqrt(mean(holt_residuals^2) / mean((oil_data_clean$dcoilwtico - lag(oil_data_clean$dco
cat("Theil's U-statistic for Holt's model:", holt_theils_u, "\n")
```

```
## Theil's U-statistic for Holt's model: Inf
```

```
cat("MAPE for Holt's model:", holt_mape, "\n")
```

```
## MAPE for Holt's model: 1.546043
```

```
cat("RMSE:", holt_rmse, "\n")
```

```
## RMSE: 1.175605
```

```
cat("MAE:", holt_mae, "\n")
```

```
## MAE: 0.8948867
```

```
arima_rmse <- sqrt(mean(arima_residuals^2))  # RMSE for ARIMA model
arima_mae <- mean(abs(arima_residuals))      # MAE for ARIMA model
arima_mape <- mean(abs(arima_residuals / oil_data_clean$dcoilwtico)) * 100
arima_theils_u <- sqrt(mean(arima_residuals^2) / mean((oil_data_clean$dcoilwtico - lag(oil_data_clean$d
cat("Theil's U-statistic for ARIMA model:", arima_theils_u, "\n")
```

```
## Theil's U-statistic for ARIMA model: Inf
```

```
cat("MAPE for ARIMA model:", arima_mape, "\n")
```

```
## MAPE for ARIMA model: 1.547333
```

```
cat("RMSE:", arima_rmse, "\n")
```

```
## RMSE: 1.176755
```

```
cat("MAE:", arima_mae, "\n")
```

```
## MAE: 0.8949041
```

- Holt's model has a marginally better RMSE (1.1756) compared to ARIMA (1.1768), which indicates it has a slightly better fit to the data.
- Both models show almost identical MAE (0.8949), meaning their average errors are very similar.
- The MAPE values are also very close (1.5460% for Holt's and 1.5473% for ARIMA), meaning both models have similar relative errors.
- Both models have an Inf value for Theil's U-statistic, which could indicate issues in comparing them to a naive model, but this is not a major issue when looking at the other metrics.

Conclusion:

Holt's Linear Trend Model appears to be the better choice based on RMSE, though both models perform quite similarly across the other metrics.