Project 2 Group Members - Shreyas Waikar, Rishabh Kanodiya, Snigdha Pandit, Arya Satam

1.

```
# Load the necessary libraries
library(readr)
library(ggplot2)

# Read the data
oil_data <- read_csv("D:/UB/509 Sdm2/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.
```

```
# Convert the 'date' column to a date format
oil_data$date <- as.Date(oil_data$date)

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
```

2.

```
# Plot the time series as is
ggplot(data = oil_data, aes(x = date, y = dcoilwtico)) + geom_line(color = "blue") +
  labs(title = "Daily Oil Prices Over Time",
       x = "Date", y = "Oil Price (USD)") + theme_minimal()
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```

3. When working with time series data such as daily oil prices, it is essential to address any missing data points to ensure the accuracy of your analysis. Popular techniques for data imputation in time series include: Forward Filling (Last Observation Carried Forward): This approach uses the most recent non-null value to fill subsequent missing entries. Backward Filling: This method fills missing values by using the subsequent available data point, effectively the reverse of forward filling. Linear Interpolation: This technique calculates missing values by creating a linear path between existing data points, providing a smooth transition between them. Time Series Specific Methods (e.g., Seasonal Decomposition): For more intricate data sets, methods that analyze and account for seasonal and trend variations are employed. For the oil price data specifically, linear interpolation is a straightforward yet effective choice. It fills gaps by drawing straight lines between known values, thus maintaining a natural progression without the need for complex models.

```
# Load necessary libraries
library(tidyr)
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 4.3.2
```

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

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

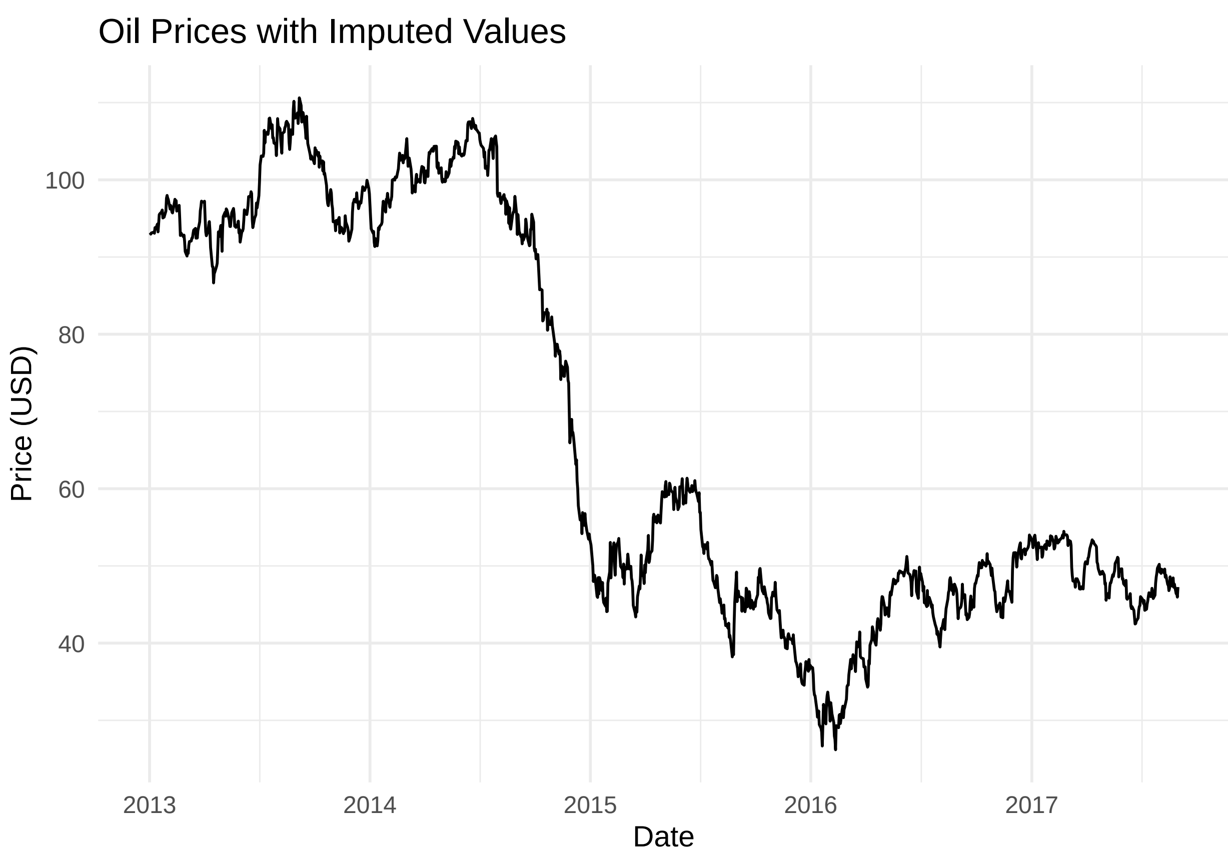
```r
# Convert 'date' to Date format
oil_data$date <- as.Date(oil_data$date)

# Impute missing data using linear interpolation
oil_data$dcoilwtico <- na.approx(oil_data$dcoilwtico, na.rm = FALSE)
```

4.

```r
ggplot(oil_data, aes(x = date, y = dcoilwtico)) + geom_line() +
  labs(title = "Oil Prices with Imputed Values", x = "Date", y = "Price (USD)") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```



The oil price data exhibits noticeable trends and potential seasonal variations: Trend There is a pronounced decline in prices from mid-2014 to early 2016, marking a significant drop. Post early 2016, the prices appear to stabilize and show a slow upward movement through 2017. Seasonality Determining seasonality from the provided plot alone is challenging. While oil prices might be affected by seasonal factors like higher travel demand during summer months or increased heating oil usage in winter, these patterns may not be immediately apparent from the plot. A more thorough analysis or reviewing data over an extended period might be necessary to confirm seasonality.

5. ETS models, short for Error, Trend, and Seasonality, are a class of time series forecasting techniques that extend exponential smoothing. These models are particularly effective for datasets displaying trends and seasonal patterns. Error Types: The model can incorporate either additive or multiplicative

errors, which affects how residuals are computed relative to the series level. Trend Types: Options include no trend, additive, or multiplicative trends. An additive trend implies linear changes over time, whereas a multiplicative trend suggests changes that scale with the series. Seasonality Types: These can also be non-existent, additive, or multiplicative, depending on whether the seasonal fluctuations are constant or vary in proportion to the time series. ETS models leverage likelihood-based methods to automatically tune the smoothing parameters and initial conditions. These models are available in several statistical software platforms, including R, where they are accessible through the forecast package.

The Holt-Winters method is an advancement of exponential smoothing designed to address seasonality in addition to capturing overall level and trend within a dataset. This method is particularly effective for handling data characterized by seasonal fluctuations. Level: Represents the average value within the series. Trend: Indicates the direction and rate of change within the data, either increasing or decreasing. Seasonal: Refers to the recurring short-term cycles observed in the series. The Holt-Winters method is available in two forms: Additive: Appropriate for when seasonal variations are consistent and relatively stable throughout the series. Multiplicative: Best used when seasonal variations vary in intensity proportionate to the time series. This approach employs three distinct types of smoothing—level, trend, and seasonal—which involve calculating weighted averages of past observations, where the weights decrease exponentially over time.

Implementation in R Both ETS and Holt-Winters models can be easily implemented in R using the forecast package. For instance:

```
# Assuming 'oil_data' is our data frame and 'date' and 'dcoilwtico' are the columns
# First, make sure that 'date' is in Date format and 'dcoilwtico' is numeric
oil_data$dcoilwtico <- as.numeric(oil_data$dcoilwtico)

# Create a ts object from the 'dcoilwtico' column
# Frequency could be 365 (daily data), 12 (monthly data), etc.
oil_ts <- ts(oil_data$dcoilwtico, frequency=365, start=c(2013,1))

# Check the ts object
print(head(oil_ts))
```

```
## [1]    NA 93.14 92.97 93.12 93.20 93.21
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.2
```

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

```
# Fit an ETS model
ets_fit <- ets(oil_ts)
```

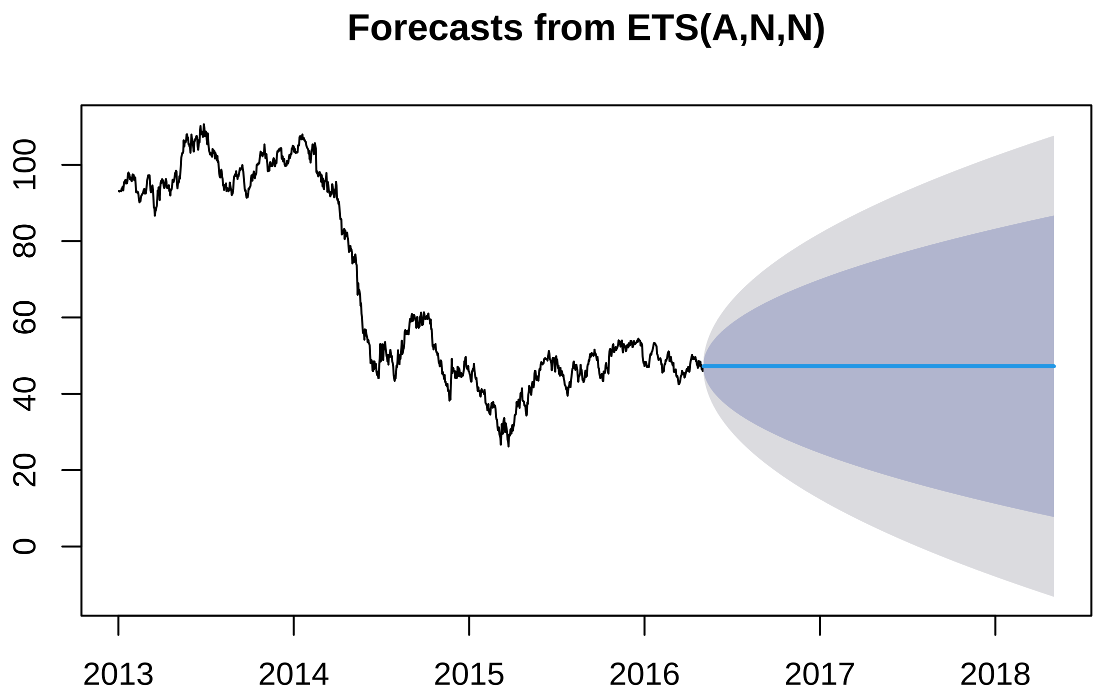
```
## Warning in ets(oil_ts): Missing values encountered. Using longest contiguous
## portion of time series
```

```
## 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.
```

```r
# Check the model summary
summary(ets_fit)
```

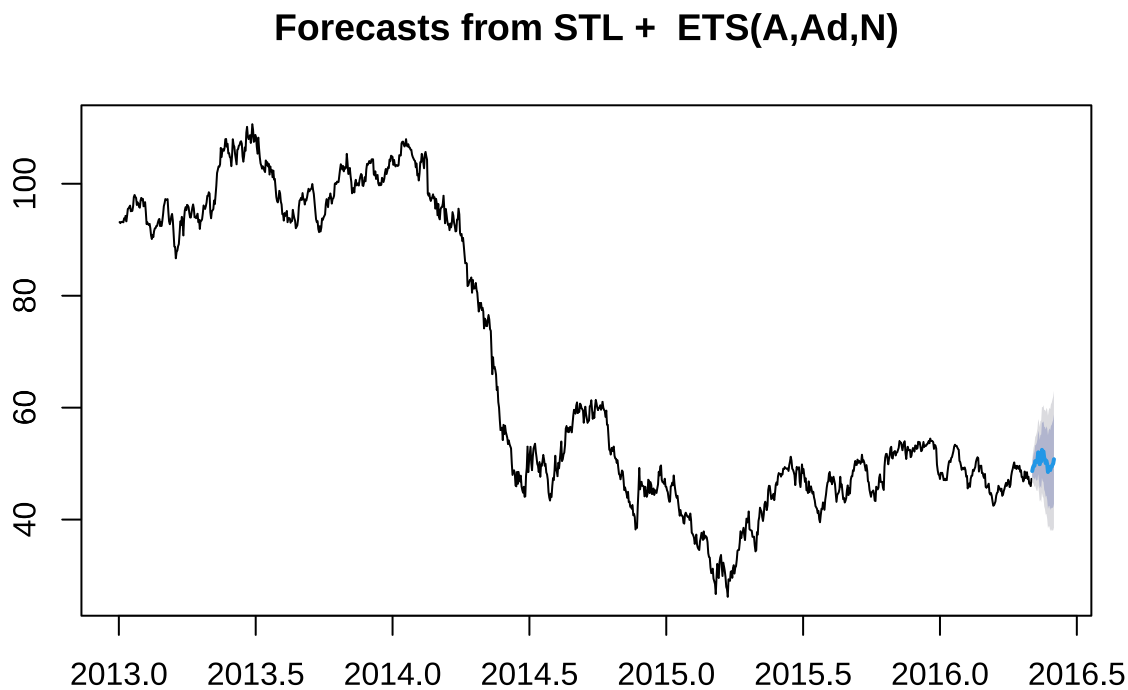
```
## ETS(A,N,N)
##
## Call:
##  ets(y = oil_ts)
##
##   Smoothing parameters:
##     alpha = 0.9686
##
##   Initial states:
##     l = 93.1255
##
##   sigma:  1.1776
##
##      AIC     AICc      BIC
## 9047.704 9047.724 9063.016
##
## Training set error measures:
##                      ME     RMSE       MAE         MPE     MAPE       MASE
## Training set -0.0389448 1.176659 0.8958282 -0.08226113 1.549556 0.03257094
##                      ACF1
## Training set -0.001020253
```

```r
# Optional: Plot the model with forecasts
plot(forecast(ets_fit))
```

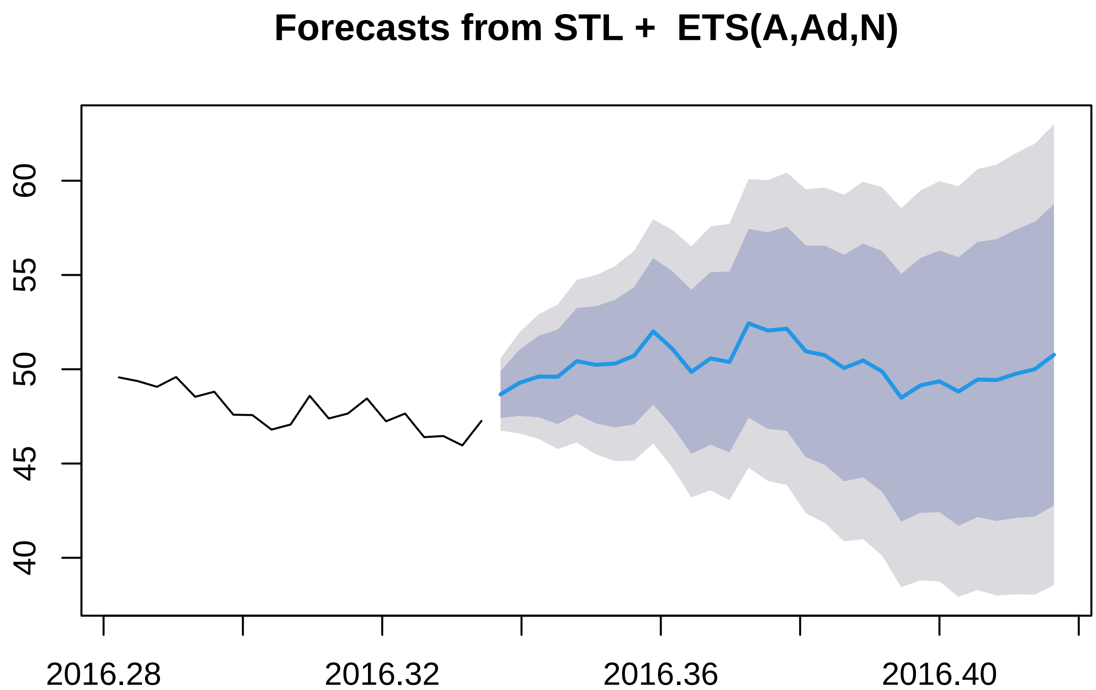



```
# Fitting a Holt-Winters model, we start with the default which includes both trend and seasonal compon
# This will use both additive trend and additive seasonal components by default
hw_model <- stlf(oil_ts)

# Plot the model with forecasts; let's forecast for the next 30 days as an example
hw_forecast <- forecast(hw_model, h=30)
plot(hw_forecast)
```




```
# Additional outputs can include confidence intervals on the forecasts
plot(hw_forecast, include=20)  # Including last 20 points of the historical data for context
```



6. Given the oil price data's evident trend and possible seasonality influenced by economic, political, or seasonal factors: Holt-Winters Model: This model is well-suited for the data due to its ability to handle both trend and potential seasonal variations. Depending on whether the seasonal changes are consistent or vary over time, you would select either the additive or multiplicative version of the model. ETS Model: This model might also be fitting, especially if the errors and seasonal patterns in the data are believed to fluctuate over time. The ETS model can automatically identify and apply the most suitable configurations of error, trend, and seasonality—whether additive or multiplicative. Both models are adept at managing the trends and seasonality observed in the oil price data. The selection between these models should be guided by the specific nuances of the data and your predictive objectives. If these models don't yield the desired outcomes, considering more sophisticated models might be necessary.

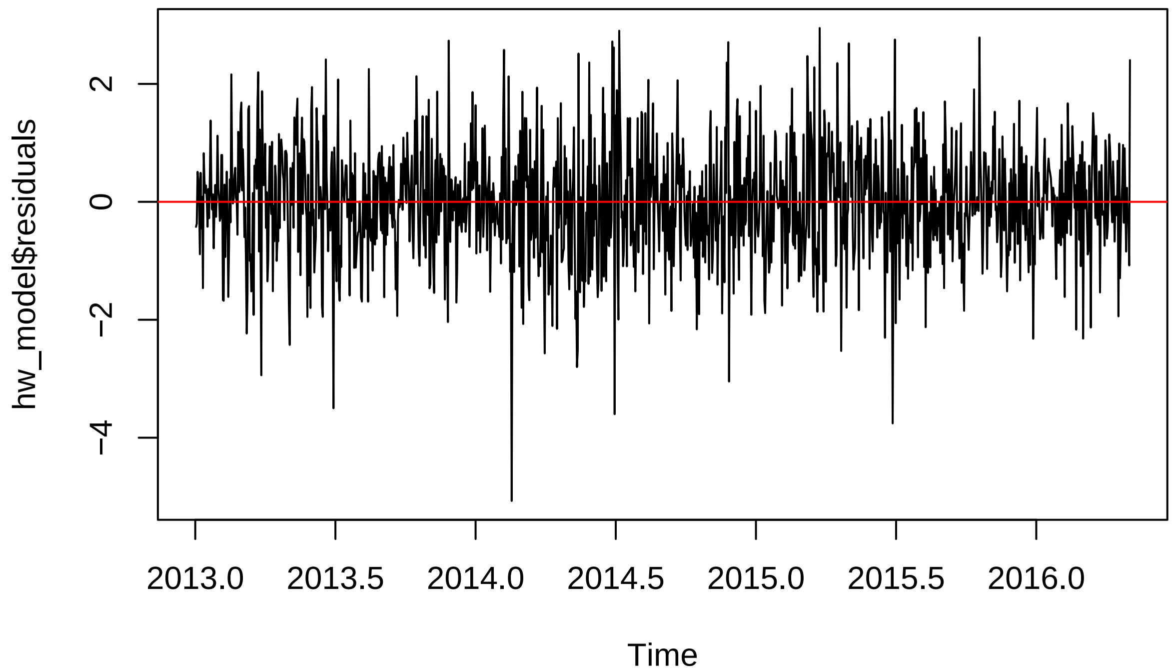
7.

```
library(forecast)
hw_model <- stlf(oil_ts)
ets_model <- ets(oil_ts)
```

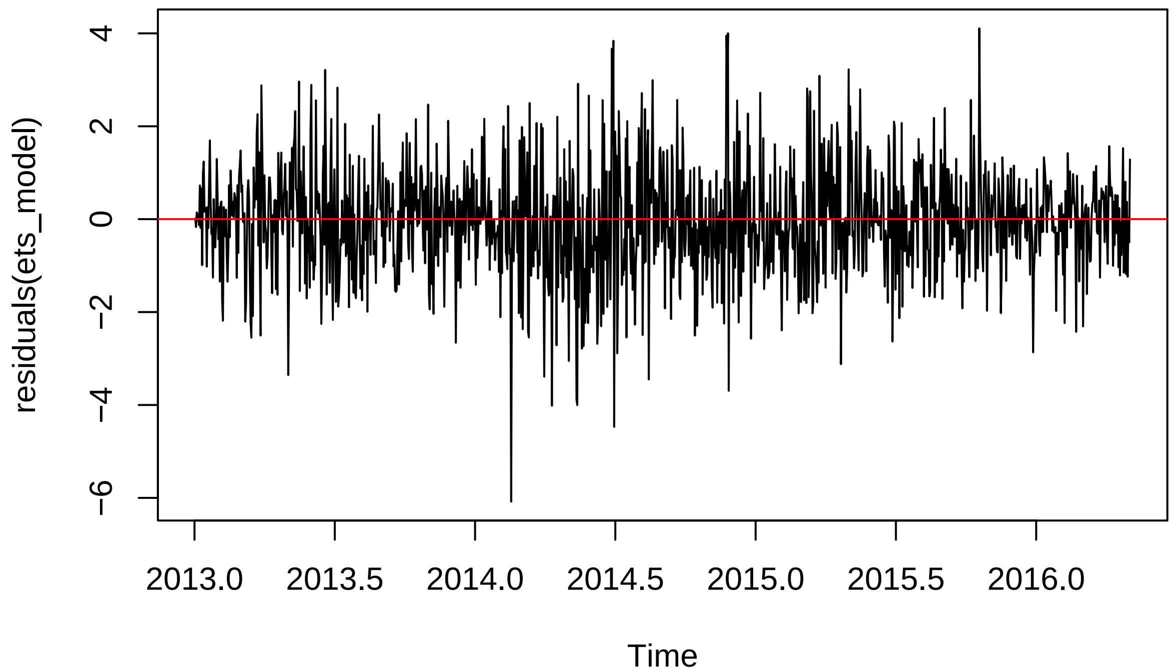
```
## Warning in ets(oil_ts): Missing values encountered. Using longest contiguous
## portion of time series
```

```
## 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.
```

```r
#Checking Model Adequacy
# Plotting residuals for Holt-Winters
plot(hw_model$residuals)
abline(h = 0, col = "red")
```




```r
# Plotting residuals for ETS
plot(residuals(ets_model))
abline(h = 0, col = "red")
```




```r
#Statistical Tests
Box.test(hw_model$residuals, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  hw_model$residuals
## X-squared = 16.381, df = 20, p-value = 0.6927
```

```r
Box.test(residuals(ets_model), lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals(ets_model)
## X-squared = 11.402, df = 20, p-value = 0.9351
```

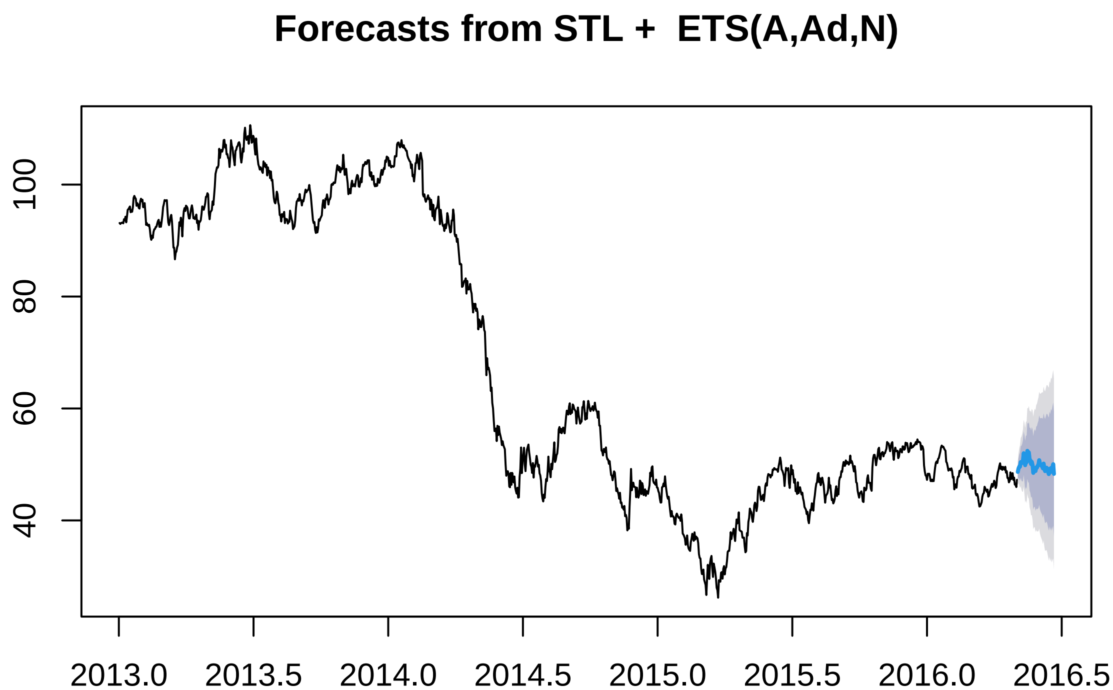
```r
# Accuracy of Holt-Winters
hw_accuracy <- accuracy(hw_model)
print(hw_accuracy)
```

```
##                       ME      RMSE       MAE         MPE     MAPE       MASE
## Training set -0.02033319 0.9765624 0.7550432 -0.03412894 1.300276 0.02745221
##                      ACF1
## Training set -0.000355573
```

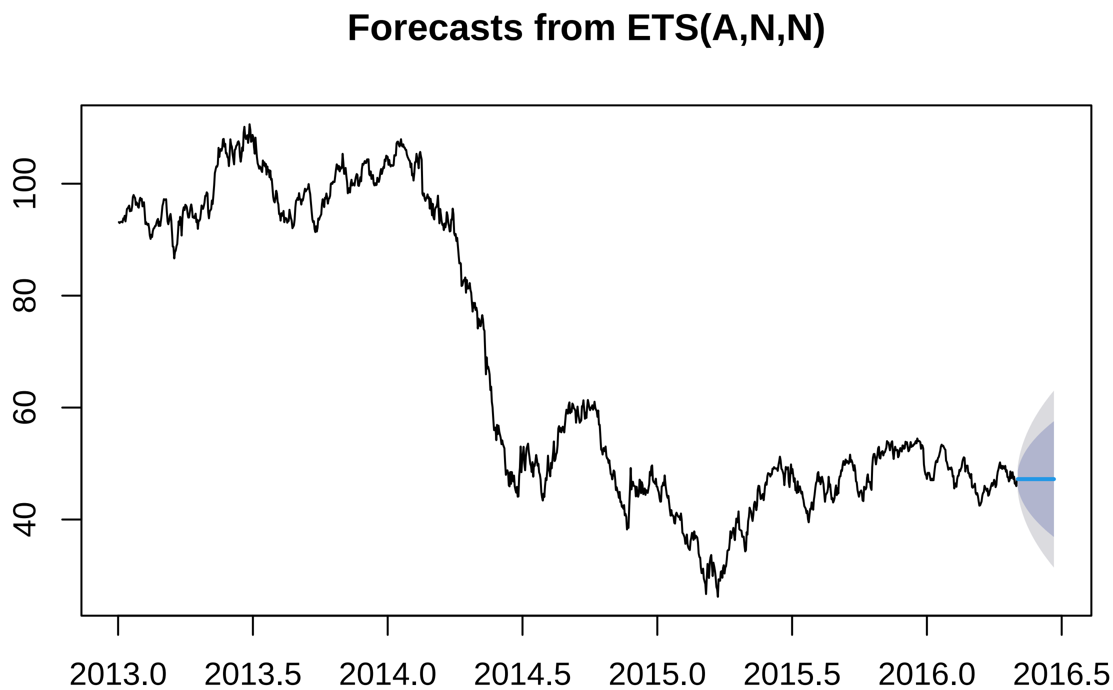
```
# Accuracy of ETS
ets_accuracy <- accuracy(ets_model)
print(ets_accuracy)
```

```
##                       ME     RMSE       MAE         MPE     MAPE       MASE
## Training set -0.0389448 1.176659 0.8958282 -0.08226113 1.549556 0.03257094
##                      ACF1
## Training set -0.001020253
```

```
# Forecasting with Holt-Winters
hw_forecast <- forecast(hw_model, h=50)
plot(hw_forecast)
```




```
# Forecasting with ETS
ets_forecast <- forecast(ets_model, h=50)
plot(ets_forecast)
```



8. Here's a comparison of their performance based on the metrics shown:

Root Mean Squared Error (RMSE) Holt-Winters: RMSE = 0.9765624 ETS: RMSE = 1.176659 Lower RMSE values indicate a model that better fits the data, minimizing the square root of average squared differences between predicted and actual observations. Here, the Holt-Winters model has a lower RMSE, suggesting it performs better in terms of prediction accuracy.

Mean Absolute Error (MAE) Holt-Winters: MAE = 0.7550432 ETS: MAE = 0.8958282 MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. Again, Holt-Winters shows a lower MAE, indicating it generally makes smaller errors in its predictions than the ETS model.

Mean Absolute Percentage Error (MAPE) Holt-Winters: MAPE = 1.300276 ETS: MAPE = 1.549556 MAPE expresses accuracy as a percentage, and lower values are better. The Holt-Winters model has a lower MAPE, signifying it has a smaller average percentage error.

Conclusion Based on these metrics, particularly focusing on RMSE, the Holt-Winters model outperforms the ETS model in this scenario. It provides more accurate and reliable forecasts according to the data provided. If reducing forecasting error is crucial for your application, Holt-Winters seems to be the preferable choice.