# EAS 509 Project 2

jayyoges

2023-11-27

# Project 2

## Importing the data

```
data = read.csv('oil.csv')
head(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
```

```
str(data)
```

```
## 'data.frame':    1218 obs. of  2 variables:
##  $ date      : chr  "2013-01-01" "2013-01-02" "2013-01-03" "2013-01-04" ...
##  $ dcoilwtico: num  NA 93.1 93 93.1 93.2 ...
```

```
data <- data %>% setNames(c('Date', 'Price'))
data$Date <- as.Date(data$Date)
```

# Phase 1: EDA & Data Cleaning

## Checking for missing values in the data

```
# How many missing values?
missing_values <- sum(is.na(data))

# Print the number of missing values
cat("Number of missing values in the entire data frame:", missing_values, "\n")
```

```
## Number of missing values in the entire data frame: 43
```

```
# Check for missing values in each column
missing_values_per_column <- sapply(data, function(x) sum(is.na(x)))

# Print the number of missing values for each column
cat("Number of missing values per column:\n")
```

```
## Number of missing values per column:
```

```
print(missing_values_per_column)
```

```
##  Date Price
##     0    43
```

We have 43 missing values for oil price

```
# Identify rows with missing values in a specific column
rows_with_missing_values <- which(is.na(data$Price))

# Print the rows with missing values
cat("Rows with missing values in column 'Price':\n")
```

```
## Rows with missing values in column 'Price':
```

```
print(rows_with_missing_values)
```

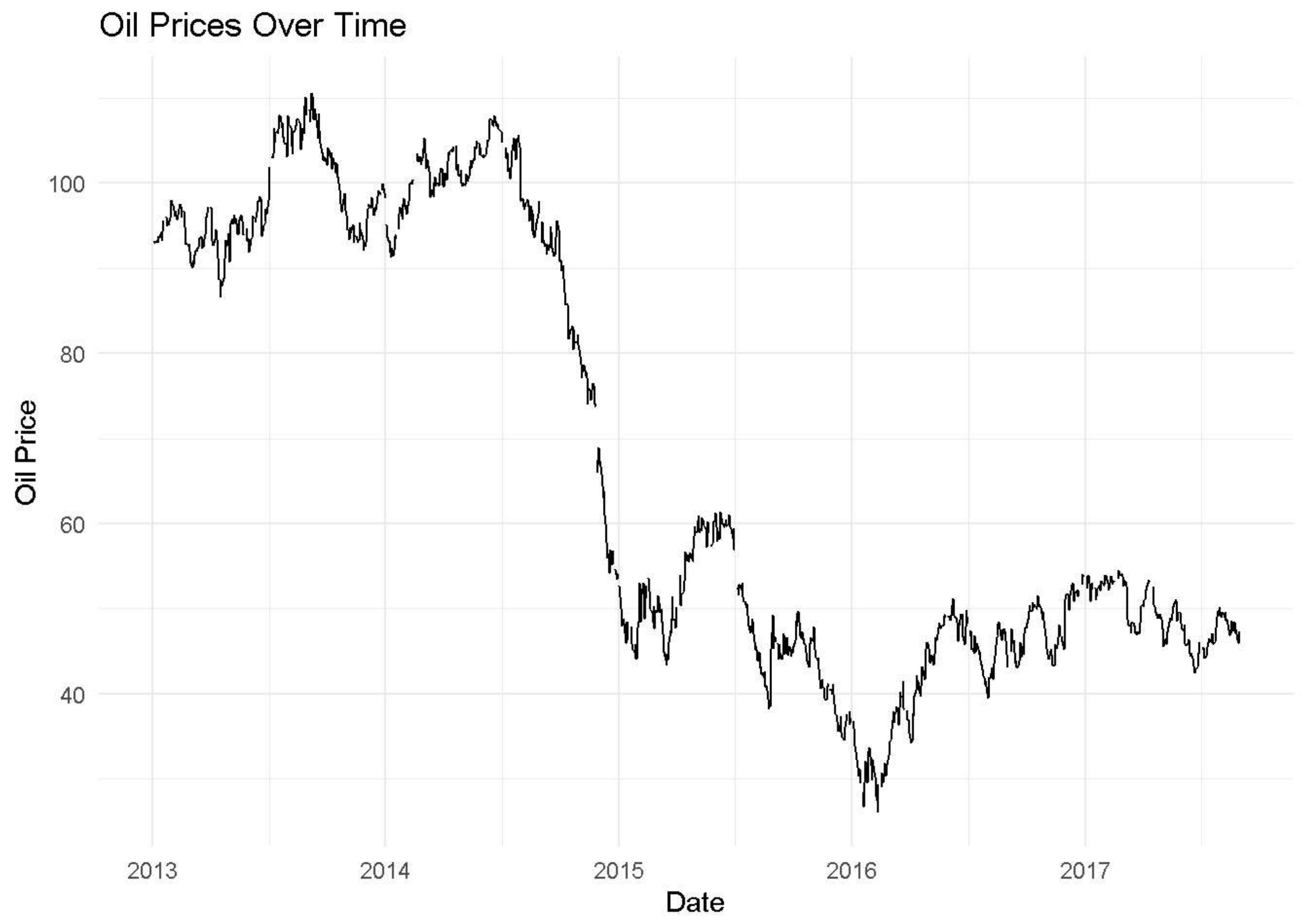
```
##  [1]    1   15   35   64  105  133  175  238  257  262  275  295  339  365  394
## [16]  435  498  518  523  535  555  589  625  654  700  758  779  784  795  815
## [31]  844  890  915  960 1018 1040 1045 1055 1080 1119 1150 1175 1176
```

## Plot before imputing

```
oilplot = function(data, title = "Oil Prices Over Time") {
  ggplot(data, aes(x = Date, y = Price)) +
    geom_line() +
    labs(title = title,
        x = "Date",
        y = "Oil Price") +
    theme_minimal()
}

oilplot(data)
```

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



## Imputing the data

```
# Forward Fill
data_ffill <- data
data_ffill$Price <- na.locf(data_ffill$Price, na.rm = FALSE)

# Backward Fill
data_bfill <- data
data_bfill$Price <- na.locf(data_bfill$Price, fromLast = TRUE)

# Linear Interpolation
data_linear <- data
data_linear$Price <- na.approx(data_linear$Price, na.rm = FALSE)

# Moving Average
k <- 5
data_ma <- data
data_ma$Price <- na.fill(data_ma$Price, fill = "extend")
data_ma$Price <- rollapply(data_ma$Price, k, FUN = mean, na.rm = TRUE, fill = NA, align = 'cente
r')
```

## Missing values after imputation

```
comparison_df <- data.frame(
   Date = data$Date
  ,Original = data$Price
  ,ForwardFill = data_ffill$Price
  ,BackwardFill = data_bfill$Price
  ,LinearInterpolation = data_linear$Price
  ,MovingAverage = data_ma$Price
)
```

```
sum(is.na(comparison_df$Original))
```

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

```
sum(is.na(comparison_df$ForwardFill))
```

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

```
sum(is.na(comparison_df$BackwardFill))
```

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

```
sum(is.na(comparison_df$LinearInterpolation))
```

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

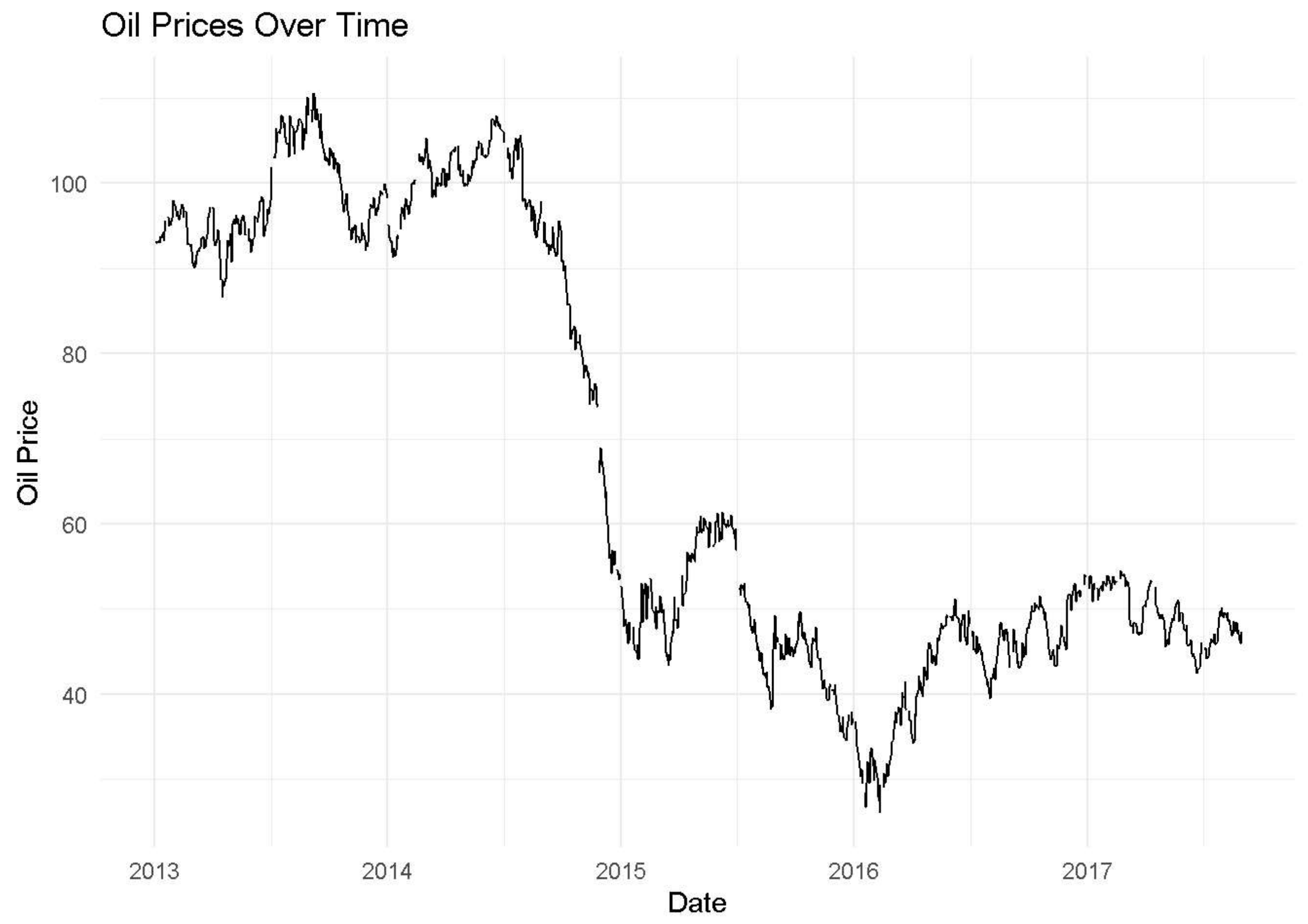
```
sum(is.na(comparison_df$MovingAverage))
```

```
## [1] 4
```

## Data Plots After Imputation

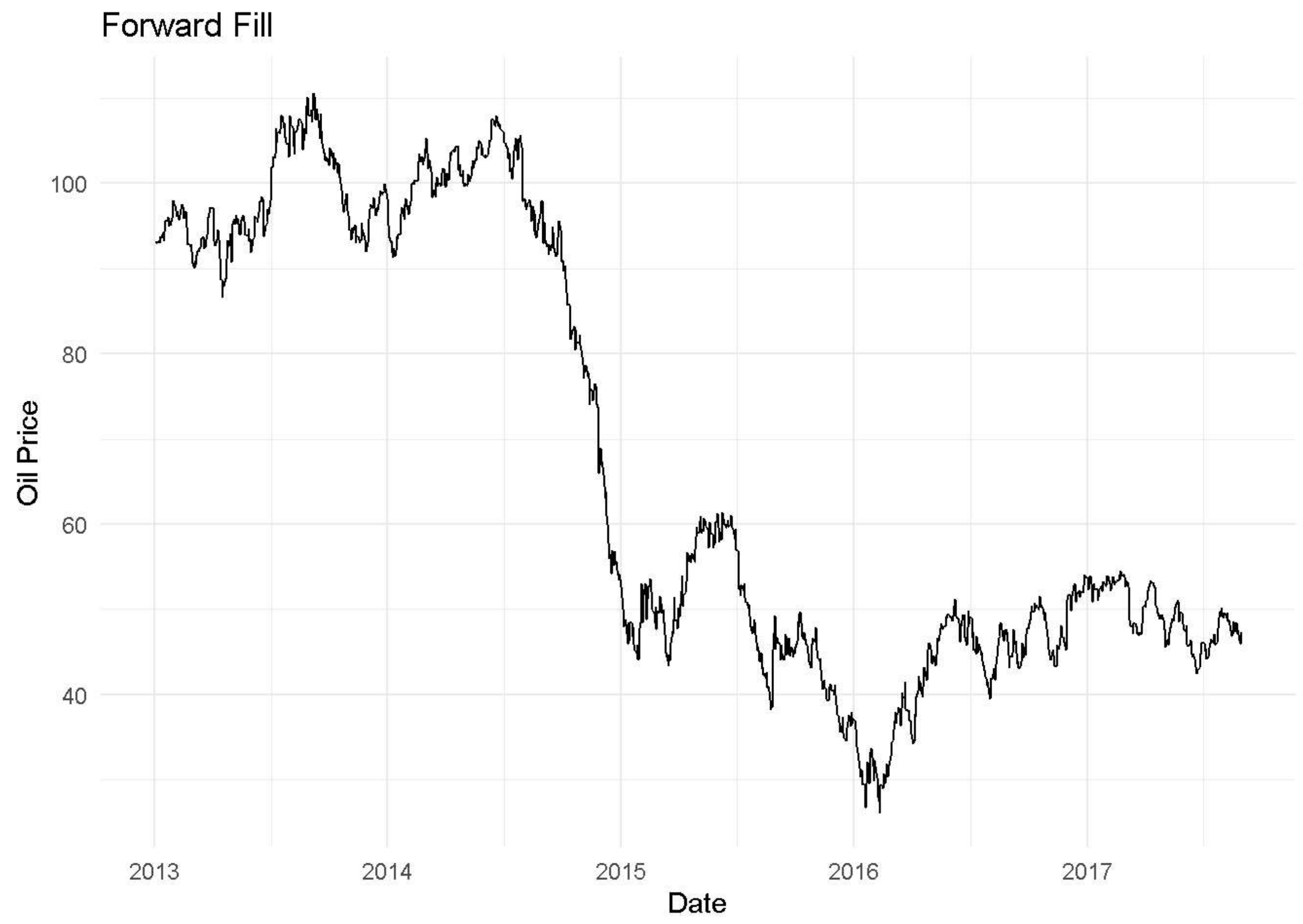
```
oilplot(data)
```

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

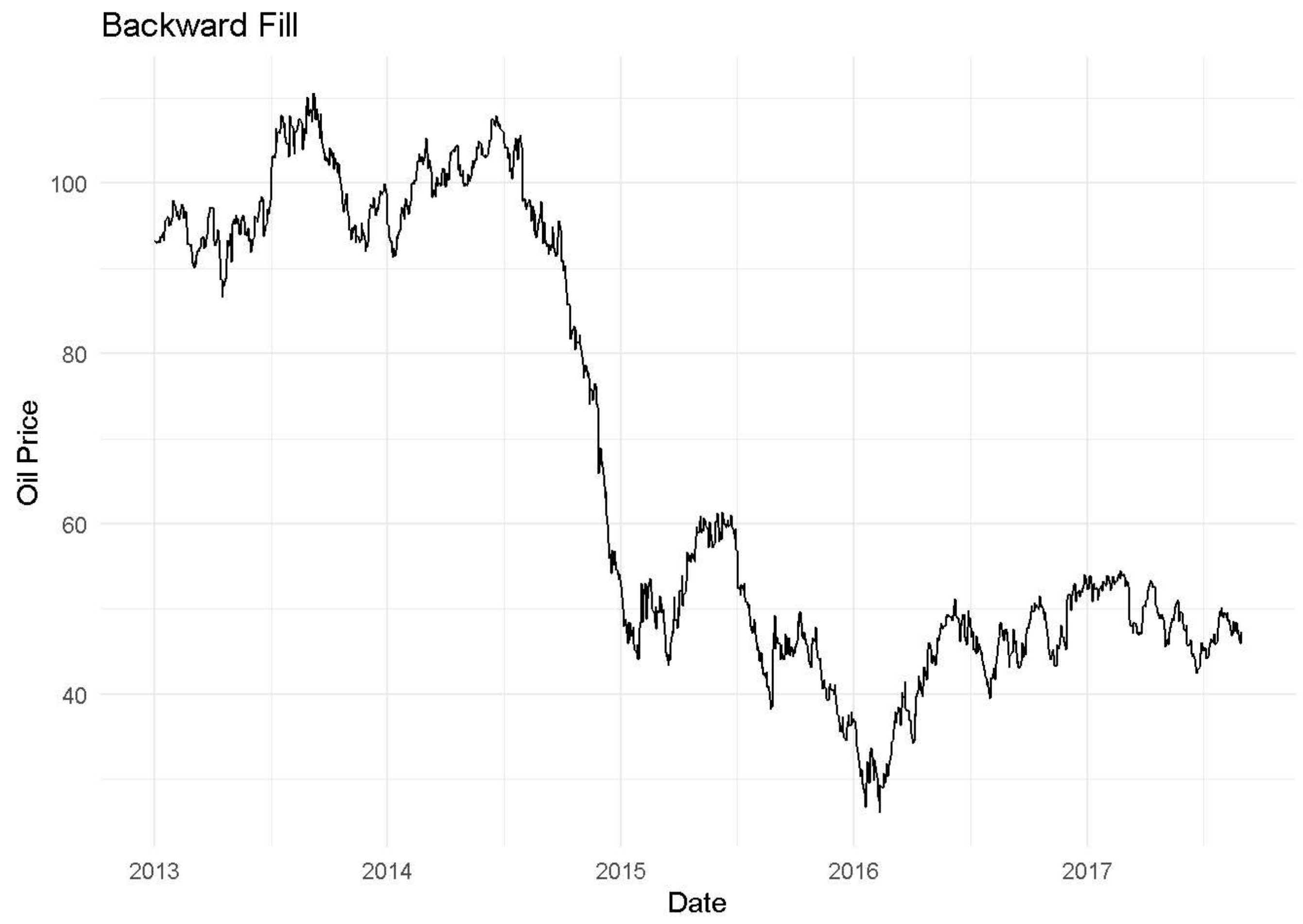


```
oilplot(data_ffill, "Forward Fill")
```

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

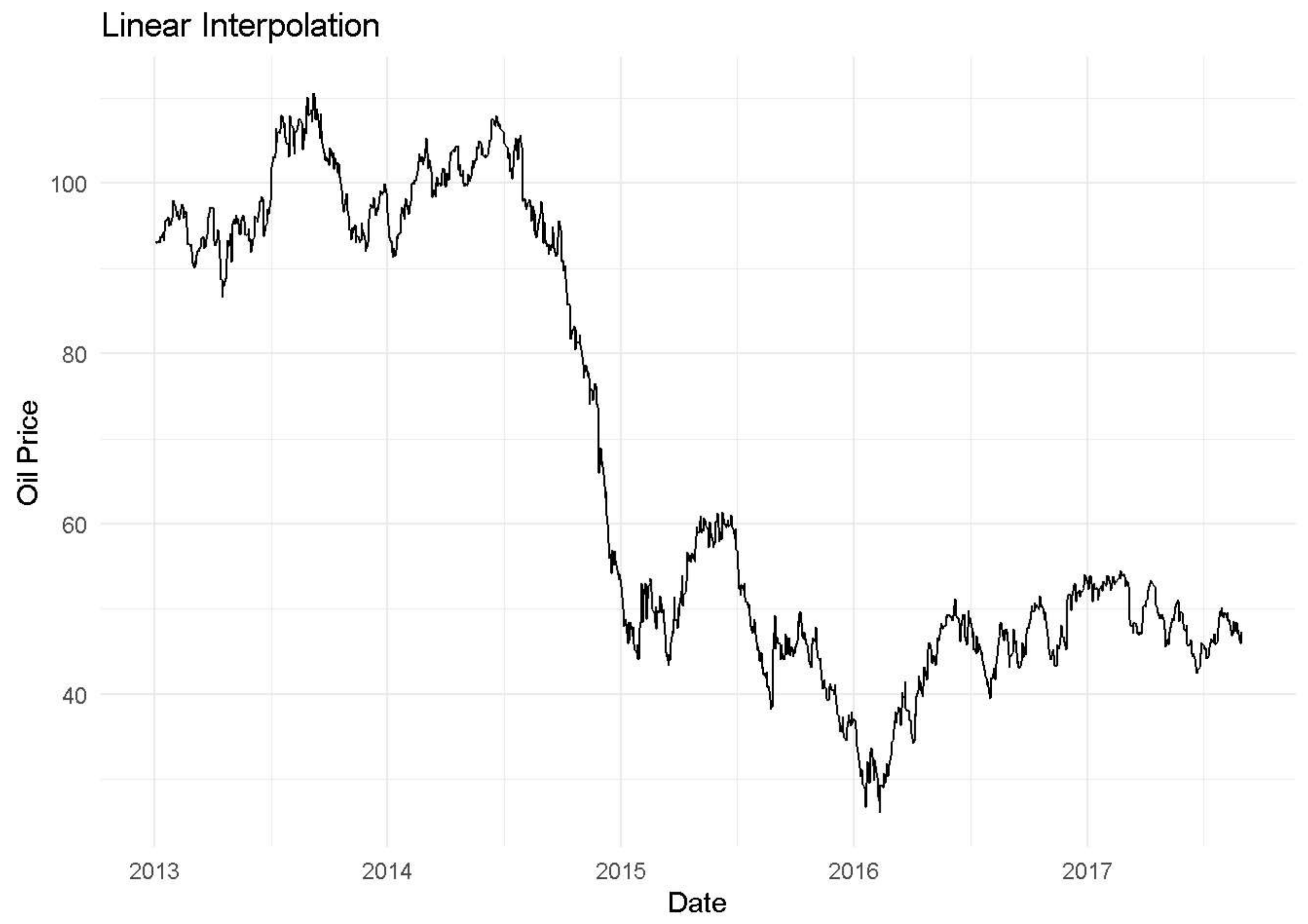


```
oilplot(data_bfill, "Backward Fill")
```



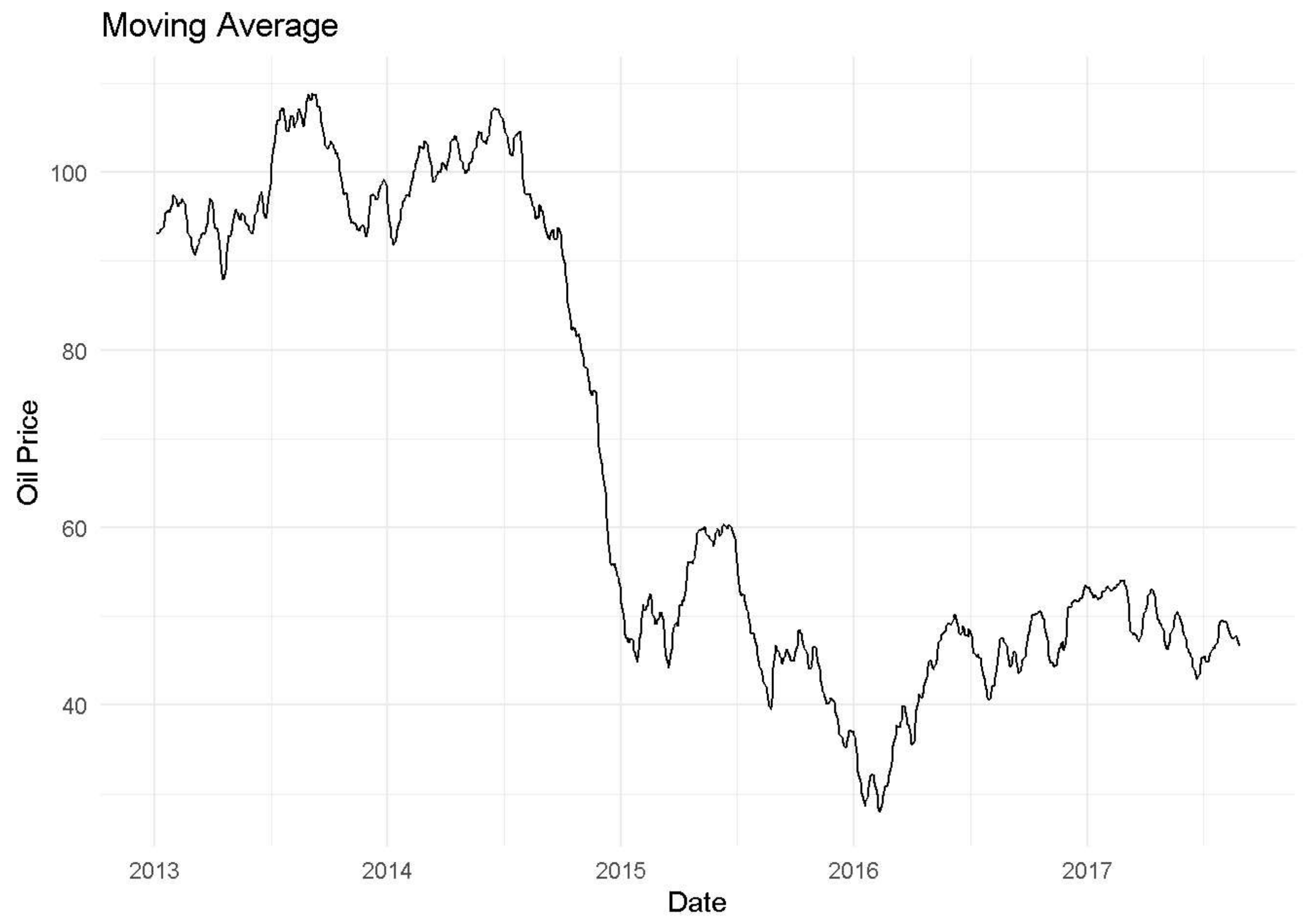
```
oilplot(data_linear, "Linear Interpolation")
```

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



```
oilplot(data_ma, "Moving Average")
```

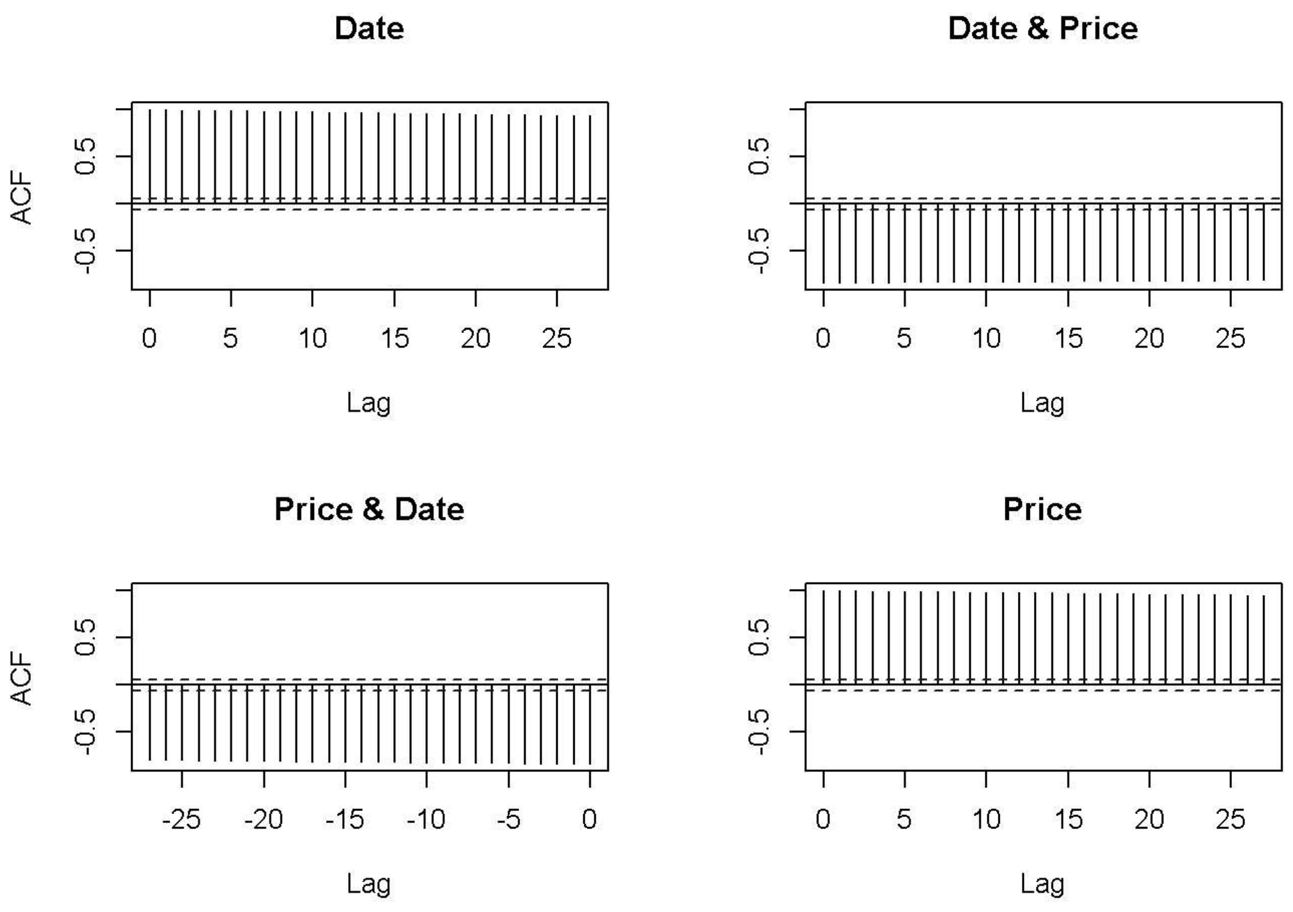
```
## Warning: Removed 4 rows containing missing values (`geom_line()`).
```



All imputation methods except backward fill are unable to fill in all missing values, thus we pick backward fill imputed data.

## Now Checking for seasonality in the time series data

```
# ACF plot
acf(data_bfill)
```



Here all the bars for our ACF plots are of same height. We can not see any spike in the graph for any lags which suggest that there is no strong seasonality in the data.

## Phase 2: Model Training & Evaluation

1. Simple Exponential Smoothing (ETS Model: ANN)

Theory: Simple Exponential Smoothing (SES) is used for forecasting univariate time series data. In this model, forecasts are calculated using weighted averages of past observations, where the weights decrease exponentially as observations get older, hence the name “exponential smoothing.”

Model Characteristics: ANN (Additive error, No trend, No seasonality): This is the simplest form of the ETS model, suitable for time series data with no clear trend or seasonal pattern. It’s effective for smoothing out noise and forecasting in the short term.

Error Type (Additive): In the additive error model, the forecast errors are assumed to be randomly distributed around zero.

2. ETS Model with Additive Error and Trend (AAN)

Theory: This variation of ETS introduces a trend component to the simple exponential smoothing model. Model Characteristics: AAN (Additive error, Additive trend, No seasonality): Suitable for data that shows a linear trend. The additive trend implies that the magnitude of the trend is constant over time.

Trend Component (Additive): The additive trend is a linear trend that can either be increasing or decreasing over time. The model adapts to changes in the trend but assumes that the trend’s impact on the forecast is constant in each period.

3. Auto ARIMA Model

Theory: The ARIMA model is a generalization of an autoregressive moving average (ARMA) model. ARIMA models are applied in cases where data shows evidence of non-stationarity, and they incorporate the differencing of raw observations (i.e., subtracting an observation from an observation at the previous time step) to make the time series stationary.

Model Characteristics: Components: ARIMA models are characterized by three primary parameters: p (AR terms), d (differencing order), and q (MA terms).

Auto ARIMA: This function automates the process of model selection by identifying the optimal combination of p, d, and q that provides the best fit to the time series.

4. Simple Exponential Smoothing (ZNN)

Theory: This approach involves allowing the ETS function to automatically select the most appropriate model based on the data.

Model Characteristics: Automatic Model Selection: The algorithm assesses various combinations of error, trend, and seasonality components and selects the model that best captures the underlying patterns in the data.

Flexibility: This method is beneficial when the user is unsure about the specific characteristics of the time series data.

5. Damped Trend Exponential Smoothing

Theory: This model is an extension of the ETS model that includes a damped trend component. It is particularly useful for long-term forecasting.

Model Characteristics: Damped Trend: Unlike a simple linear trend, a damped trend slows down over time, becoming flatter in the future. This makes the model less aggressive in projecting future trends and can yield more realistic long-term forecasts.

Applicability: Best suited for time series data where the trend is expected to lose momentum over time.

R script is designed for time series analysis, specifically for modeling and evaluating oil prices. It follows these steps:

Model Training: Various time series forecasting models are trained using the Price column from the data dataset. The models include: Simple Exponential Smoothing (ETS ANN model) ETS with Additive Error and Trend (AAN model) Auto ARIMA Model Simple Exponential Smoothing (ZNN model) Damped Trend Exponential Smoothing (AAN model with damping)

Model Visualization: The script uses the autoplot function (likely from the forecast package) to plot the fits of these models.

Model Evaluation (Optional): If the eval parameter is TRUE, the data_evaluation function is called to evaluate these models. This function prints summaries and accuracy metrics for each model.

When you run this script, it will:

Train multiple time series models on your data. Generate plots for each model to visualize their performance. Optionally, provide detailed evaluations of each model, including summaries and accuracy metrics.

```r
data_models <- function(data, eval = TRUE) {
  # 1. Simple Exponential Smoothing (ETS Model: ANN)
  ets_model1 <- ets(data$Price, model="ANN")

  #2. ETS Model with Additive Error and Trend (AAN)
  ets_model2 <- ets(data$Price, model="AAN")

  #3. Auto ARIMA Model
  best_arima_model <- auto.arima(data$Price)

  #4. Simple Exponential Smoothing
  ses_model <- ets(data$Price, model="ZNN")

  #5. Damped Trend Exponential Smoothing
  dtes_model <- ets(data$Price, model="AAN", damped=TRUE)


  autoplot(ets_model1) + xlab("Time") + ylab("Oil Prices") + ggtitle("ETS Model Fit")
  autoplot(ets_model2) + xlab("Time") + ylab("Oil Prices") + ggtitle("ETS Model Fit")
  autoplot(best_arima_model) + xlab("Time") + ylab("Oil Prices") + ggtitle("Auto ARIMA Model Fit")
  autoplot(ses_model) + xlab("Time") + ylab("Oil Prices") + ggtitle("Simple Exponential Smoothin
g")
  autoplot(dtes_model) + xlab("Time") + ylab("Oil Prices") + ggtitle("Damped Trend Exponential Smo
othing")

  if (eval) {
    data_evaluation(ets_model1, ets_model2, best_arima_model, ses_model, dtes_model)
  }
}

data_evaluation <- function(ets_model1, ets_model2, best_arima_model, ses_model, dtes_model) {
  cat("---- ETS Model 1 (ANN) ----\n")
  print(summary(ets_model1))
  print(accuracy(ets_model1))
  cat("\n--------------------------------\n")

  cat("---- ETS Model 2 (AAN) ----\n")
  print(summary(ets_model2))
  print(accuracy(ets_model2))
  cat("\n--------------------------------\n")

  cat("---- Auto ARIMA Model  ----\n")
  print(summary(best_arima_model))
  print(accuracy(best_arima_model))
  cat("\n--------------------------------\n")

  cat("---- Simple Exponential Smoothing ----\n")
  print(summary(ses_model))
  print(accuracy(ses_model))
  cat("\n--------------------------------\n")

  cat("---- Damped Trend Exponential Smoothing ----\n")
  print(summary(dtes_model))
  print(accuracy(dtes_model))
  cat("\n--------------------------------\n")
}
```

```
data_models(data_bfill)
```

```
## ---- ETS Model 1 (ANN) ----
## ETS(A,N,N)
##
## Call:
##  ets(y = data$Price, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.954
##
##   Initial states:
##     l = 93.1356
##
##   sigma:  1.2028
##
##      AIC     AICc      BIC
## 9107.716 9107.735 9123.031
##
## Training set error measures:
##                       ME    RMSE       MAE         MPE     MAPE     MASE
## Training set -0.03953028 1.20184 0.8965985 -0.08424058 1.551493 1.001156
##                       ACF1
## Training set -0.0008160611
##                       ME    RMSE       MAE         MPE     MAPE     MASE
## Training set -0.03953028 1.20184 0.8965985 -0.08424058 1.551493 1.001156
##                       ACF1
## Training set -0.0008160611
##
## --------------------------------
## ---- ETS Model 2 (AAN) ----
## ETS(A,A,N)
##
## Call:
##  ets(y = data$Price, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.9436
##     beta  = 1e-04
##
##   Initial states:
##     l = 92.6614
##     b = -0.0363
##
##   sigma:  1.2034
##
##      AIC     AICc      BIC
## 9110.806 9110.855 9136.331
##
## Training set error measures:
##                         ME     RMSE       MAE         MPE     MAPE     MASE
## Training set -0.0007681805 1.201391 0.8974959 -0.01745455 1.551229 1.002158
##                     ACF1
## Training set 0.009481971
##                        ME     RMSE       MAE         MPE     MAPE     MASE
## Training set -0.0007681805 1.201391 0.8974959 -0.01745455 1.551229 1.002158
##                     ACF1
## Training set 0.009481971
##
## --------------------------------
## ---- Auto ARIMA Model  ----
## Series: data$Price 
## ARIMA(0,1,0) 
## 
## sigma^2 = 1.449:  log likelihood = -1952.37
## AIC=3906.73   AICc=3906.73   BIC=3911.83
## 
## Training set error measures:
##                      ME     RMSE       MAE         MPE     MAPE      MASE
## Training set -0.03759184 1.203095 0.8949041 -0.08030733 1.547907 0.9992644
##                     ACF1
## Training set -0.04614975
##                      ME     RMSE       MAE         MPE     MAPE      MASE
## Training set -0.03759184 1.203095 0.8949041 -0.08030733 1.547907 0.9992644
##                     ACF1
## Training set -0.04614975
##
## --------------------------------
## ---- Simple Exponential Smoothing ----
## ETS(A,N,N) 
## 
## Call:
##  ets(y = data$Price, model = "ZNN") 
## 
##   Smoothing parameters:
##     alpha = 0.954 
## 
##   Initial states:
##     l = 93.1356 
## 
##   sigma:  1.2028
## 
##      AIC     AICc      BIC 
## 9107.716 9107.735 9123.031 
## 
## Training set error measures:
##                      ME    RMSE       MAE         MPE     MAPE     MASE
## Training set -0.03953028 1.20184 0.8965985 -0.08424058 1.551493 1.001156
##                       ACF1
## Training set -0.0008160611
##                      ME    RMSE       MAE         MPE     MAPE     MASE
## Training set -0.03953028 1.20184 0.8965985 -0.08424058 1.551493 1.001156
##                       ACF1
## Training set -0.0008160611
##
## --------------------------------
## ---- Damped Trend Exponential Smoothing ----
## ETS(A,Ad,N)
##
## Call:
##  ets(y = data$Price, model = "AAN", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9407
##     beta  = 0.0101
##     phi   = 0.976
##
##   Initial states:
##     l = 93.0539
##     b = 0.0824
##
##   sigma:  1.203
##
##      AIC     AICc      BIC
## 9111.022 9111.092 9141.652
##
## Training set error measures:
##                       ME     RMSE       MAE         MPE     MAPE     MASE
## Training set -0.02997243 1.200512 0.8958208 -0.06055127 1.548592 1.000288
##                      ACF1
## Training set 0.0006586526
##                       ME     RMSE       MAE         MPE     MAPE     MASE
## Training set -0.02997243 1.200512 0.8958208 -0.06055127 1.548592 1.000288
##                      ACF1
## Training set 0.0006586526
##
## --------------------------------
```

# Model Performance Evaluation

In our analysis, we evaluated several models to forecast oil prices, using key statistical metrics to assess their performance. These metrics include Mean Error (ME), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), Mean Absolute Percentage Error (MAPE), Mean Absolute Scaled Error (MASE), and the Autocorrelation of the first lag (ACF1). Each metric provides a different perspective on the accuracy and reliability of the forecasting models.

## Best Performing Model

The **Auto ARIMA Model** emerged as the best performing model in our analysis. It demonstrated the lowest RMSE (1.203095), MAE (0.8949041), and MASE (0.9992644), which are critical indicators of a model's accuracy and consistency. Furthermore, the ME (-0.03759184) and MPE (-0.08030733) values are among the lowest, suggesting a lower bias in the model's predictions. This model's superior performance across these metrics indicates its robustness in forecasting oil prices.

Specific details of the **Auto ARIMA Model** are as follows:

- Model Type: ARIMA(0,1,0)
- $sigma^2$ = 1.449
- Log Likelihood = -1952.37
- AIC = 3906.73
- AICc = 3906.73
- BIC = 3911.83

This information indicates a simple model without AR or MA components but with a differencing order of 1, suggesting that the model primarily focuses on capturing the changes rather than absolute levels in the oil price series.

## Poorest Performing Models

The **Damped Trend Exponential Smoothing** method was identified as the least effective model in our dataset. This model recorded a slightly higher RMSE (1.200512), MAE (0.8958208), and MASE (1.000288) compared to the Auto ARIMA Model. Its ME (-0.02997243) and MPE (-0.06055127) also suggested a greater degree of bias in forecasts. Despite providing reasonable forecasts, this model was outperformed by other models in our analysis.

## Other Models

The **ETS Model 2 (AAN)** displayed competitive performance with a slightly higher RMSE (1.201391), MAE (0.8974959), and MASE (1.002158) than the Auto ARIMA Model. It also showed the lowest MPE (-0.01745455), suggesting minimal bias in its predictions. The **ETS Model 1 (ANN)**, while competitive, slightly underperformed in comparison to the Auto ARIMA and ETS Model 2, as indicated by its RMSE, MAE, and MASE values.

## Conclusion

It’s important to note that the differences in performance among the models are relatively minor. However, Based on our analysis, we recommend the **Auto ARIMA Model** for forecasting oil prices, due to its superior performance across various statistical metrics.