SDM Project 2

Shreya Valish Koushik, Naveen Kumar Kateghar, Vidush Bhardwaj, Simhavishnu Ramprasad

2023-12-05

Reading the file "oil.csv" and Understanding its structure:

```r
library(zoo)
```

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

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

```r
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
```

```r
sum(is.na(data))
```

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

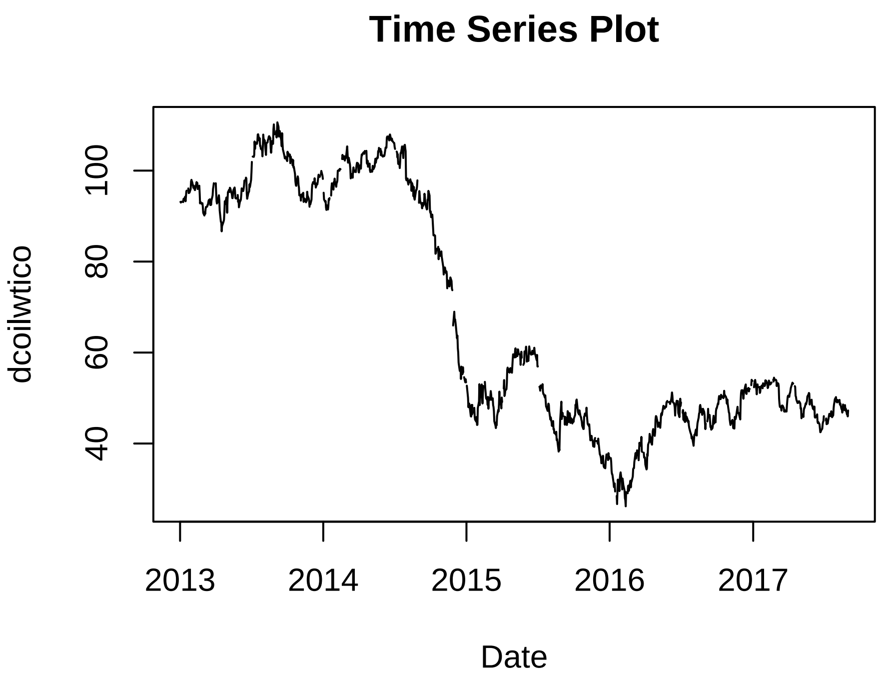
```r
nrow(data)
```

```
## [1] 1218
```

```r
data$date <- as.Date(data$date)
ts_data <- ts(data$dcoilwtico, start = min(data$date), frequency = 365)
```

Plotting the original time series data

```
plot(data$date, data$dcoilwtico, type = "l", xlab = "Date", ylab = "dcoilwtico", main = "Time Series Pl
```



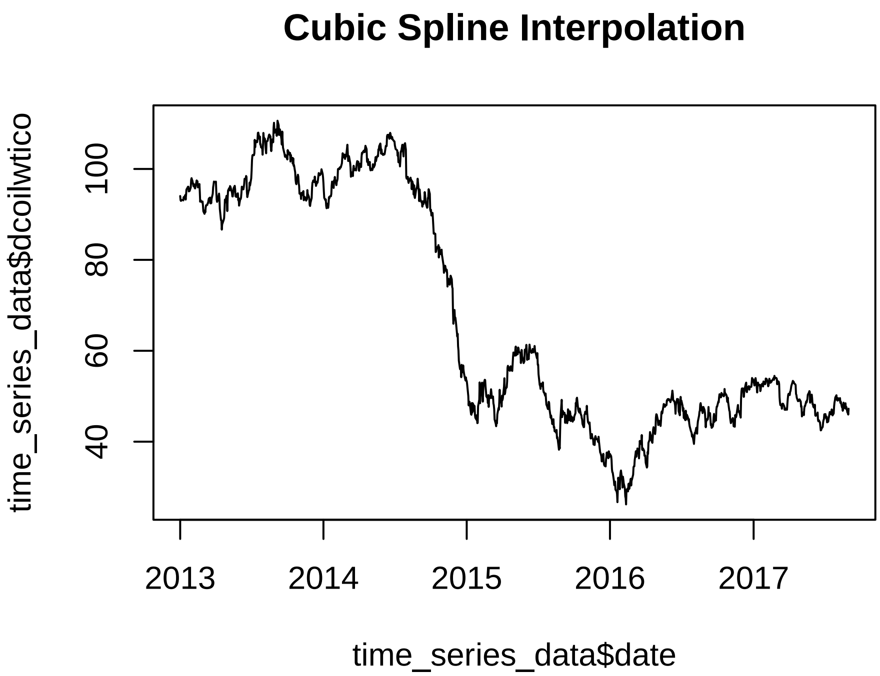

```
time_series_data <- data
time_series_data <- time_series_data[order(time_series_data$date),]
```

Imputing missing values with cubic spline

```
time_series_data$dcoilwtico <- na.spline(time_series_data$dcoilwtico)
```

Plotting the data after imputation

```
plot(time_series_data$date, time_series_data$dcoilwtico, type = "l", pch = 16, main = "Cubic Spline Inte
```



ADF Test

```
ts_data_imputed <- ts(time_series_data$dcoilwtico, start = min(data$date), frequency = 365)

library(tseries)
```

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

```
adf_test_result <- adf.test(ts_data_imputed)
print(adf_test_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_data_imputed
## Dickey-Fuller = -1.455, Lag order = 10, p-value = 0.809
## alternative hypothesis: stationary
```

###Inference from ADF Test:

Test Statistic (Dickey-Fuller): • The test statistic is -1.455. • In the context of the ADF test, a more negative test statistic provides stronger evidence against the null hypothesis (presence of a unit root, indicating non-stationarity).

Lag Order: • The number of lags considered in the test is 10.

P-value: • The p-value is 0.809. • The p-value is compared to a chosen significance level (commonly 0.05). In this case, the p-value is higher than 0.05.

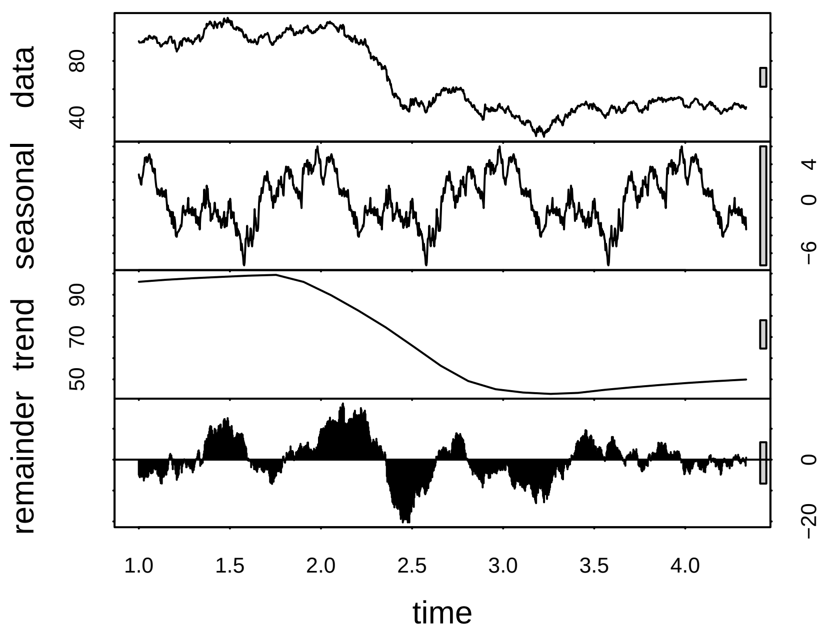
Alternative Hypothesis: • The alternative hypothesis is that the time series is stationary.

Interpretation: • Since the p-value is greater than the significance level (0.05), you fail to reject the null hypothesis. • The null hypothesis in the ADF test is that the time series has a unit root and is non-stationary. The high p-value suggests that there is not enough evidence to conclude that the time series is stationary. • In the context of seasonality, a non-stationary time series may indicate the presence of seasonality or trend.

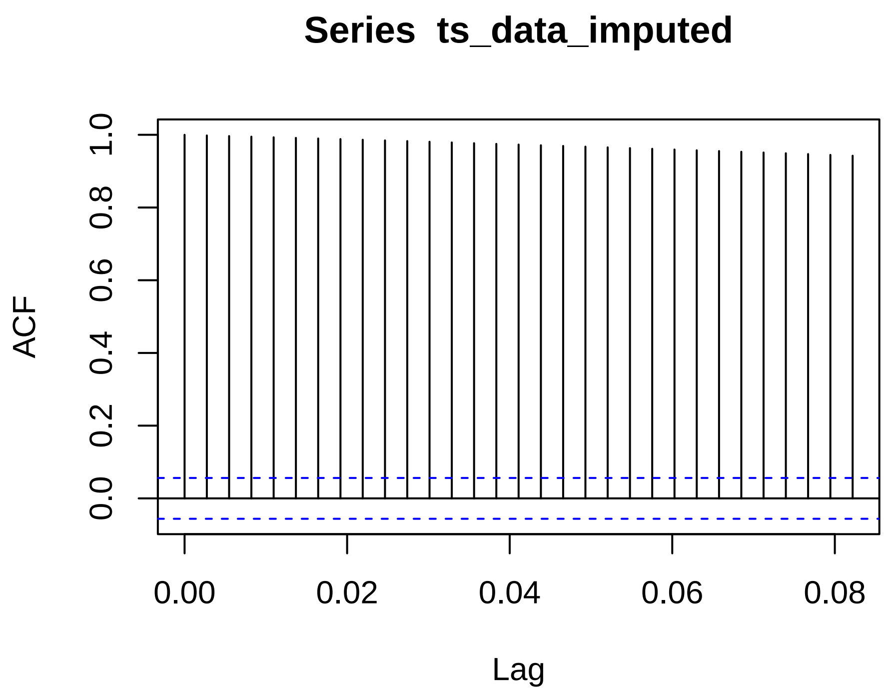
Based on the ADF Test we don't have enough evidence to say that the data is stationary, we accept the null hypothesis and say that the data is non-stationary and hence has the potential to have seasonality and trend, hence we will perform the Seasonality Decomposition for visual analysis.

Seasonal Decomposition

```r
library(ggplot2)
library(forecast)
time_series_data$date <- as.Date(time_series_data$date)
ts_data_imputed <- ts(time_series_data$dcoilwtico, frequency = 365)
stl_result <- stl(ts_data_imputed, s.window = "periodic")
plot(stl_result)
```




```r
acf(ts_data_imputed)
```



We see that there is a pattern in the seasonal component and hence we can say that the data has seasonality.

ETS (Error, Trend, Seasonality):

Error (E) Component: • Represents the residuals or errors after removing the trend and seasonality. • The nature of errors can be additive or multiplicative.

Trend (T) Component: • Captures the underlying trend in the time series. • Can be additive or multiplicative.

Seasonality (S) Component: • Represents repeating patterns or cycles in the data. • Can be additive or multiplicative.

The ETS (Error-Trend-Seasonality) framework includes several variations based on the combinations of error, trend, and seasonality components. The main types of ETS models are as follows:

ETS(AAA): Additive Error, Additive Trend, and Additive Seasonality

ETS(AAM): Additive Error, Additive Trend, and Multiplicative Seasonality

ETS(AMS): Additive Error, Multiplicative Trend, and Additive Seasonality

ETS(AMM): Additive Error, Multiplicative Trend, and Multiplicative Seasonality

ETS(MAA): Multiplicative Error, Additive Trend, and Additive Seasonality

ETS(MAM): Multiplicative Error, Additive Trend, and Multiplicative Seasonality

ETS(MSA): Multiplicative Error, Additive Trend, and Additive Seasonality

ETS(MMM): Multiplicative Error, Multiplicative Trend, and Multiplicative Seasonality

The ets function will automatically select the best-fitting ETS model based on the AIC (Akaike Information Criterion) value. Holt-Winters:

Holt-Winters:

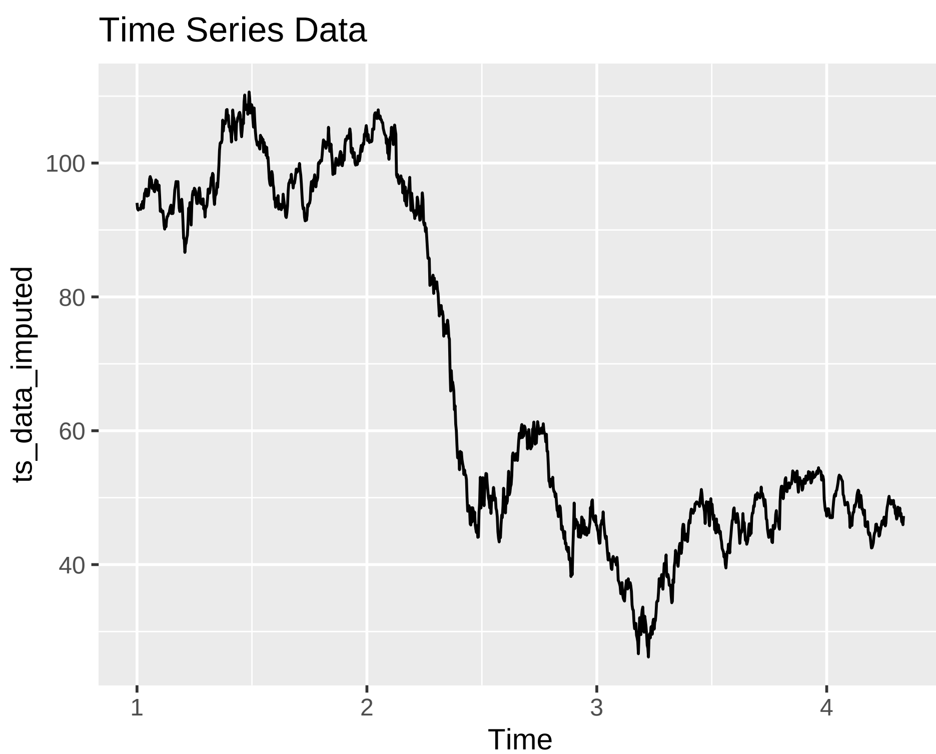
Level (L) Component: • Represents the baseline value around which the time series fluctuates.

Trend (T) Component: • Captures the direction and pattern of the time series over time.

Seasonality (S) Component: • Represents the repeating patterns or cycles with a fixed length.

Splitting into train and test and applying models

```
autoplot(ts_data_imputed) + ggtitle("Time Series Data")
```




```
train_size <- floor(length(ts_data_imputed) * 0.8)  # 80% for training
train_data <- head(ts_data_imputed, train_size)
test_data <- tail(ts_data_imputed, length(ts_data_imputed) - train_size)

model_results <- list()

model_names <- c("ARIMA", "STLF", "Holt-Winters","SARIMA","ETS")


for (model_name in model_names) {
  # Train the model
  if (model_name == "ARIMA") {
    model <- auto.arima(train_data,seasonal = FALSE)
  } else if (model_name == "STLF") {
    model <- stlf(train_data)
```

```r
  } else if (model_name == "Holt-Winters") {
    model <- HoltWinters(train_data)
  } else if (model_name == "SARIMA") {
    model <- auto.arima(train_data,seasonal = TRUE)
  } else if (model_name == "ETS") {
    model <- ets(train_data)
  }

  # Forecast using the trained model
  forecast_values <- forecast(model, h = length(test_data))

  # Evaluate the model performance
  model_accuracy <- accuracy(forecast_values, test_data)

  # Store the results in the list
  model_results[[model_name]] <- list(model = model, forecast_values = forecast_values, accuracy = model
}
```

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

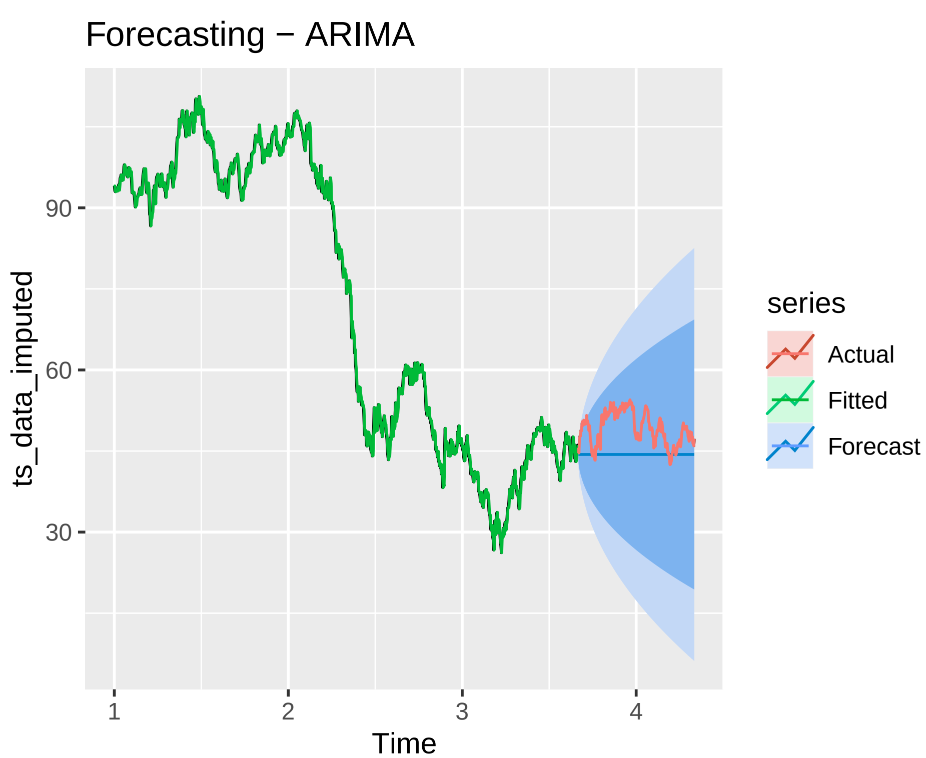
```r
for (model_name in model_names) {
  cat("Model:", model_name, "\n")
  print(model_results[[model_name]]$accuracy)
  cat("\n")
}
```

```
## Model: ARIMA
##                       ME     RMSE      MAE        MPE     MAPE       MASE
## Training set -0.05092203 1.247841 0.964288 -0.1031583 1.614943 0.02792738
## Test set      4.83131236 5.687002 4.913444  9.4813110 9.672341 0.14230147
##                     ACF1 Theil's U
## Training set -0.03003273        NA
## Test set      0.95460529  6.163439
##
## Model: STLF
##                     ME      RMSE       MAE        MPE      MAPE       MASE
## Training set -0.0299723 0.9940049 0.7616559 -0.0498944  1.260694 0.02205882
## Test set      8.9135236 9.5946115 8.9269635 17.9275391 17.958403 0.25853966
##                       ACF1 Theil's U
## Training set -0.0006375469        NA
## Test set      0.9357497077  10.72561
##
## Model: Holt-Winters
##                     ME     RMSE      MAE         MPE      MAPE       MASE
## Training set 0.00211874 1.932086 1.069184  0.05358621  2.404203 0.03096535
## Test set     7.11253135 9.486604 7.508847 14.38602281 15.218408 0.21746864
##                   ACF1 Theil's U
## Training set 0.08264281        NA
## Test set     0.95069158  10.79883
##
## Model: SARIMA
##                       ME     RMSE      MAE        MPE     MAPE       MASE
```

```
## Training set -0.05092203 1.247841 0.964288 -0.1031583 1.614943 0.02792738
## Test set      4.83131236 5.687002 4.913444  9.4813110 9.672341 0.14230147
##                    ACF1 Theil's U
## Training set -0.03003273        NA
## Test set      0.95460529  6.163439
##
## Model: ETS
##                      ME     RMSE       MAE       MPE     MAPE       MASE
## Training set -0.05251682 1.247309 0.9645263 -0.106237 1.616092 0.02793428
## Test set      4.77988901 5.643381 4.8667503  9.376379 9.578293 0.14094915
##                      ACF1 Theil's U
## Training set -0.0009021879        NA
## Test set      0.9546052906  6.114827
```

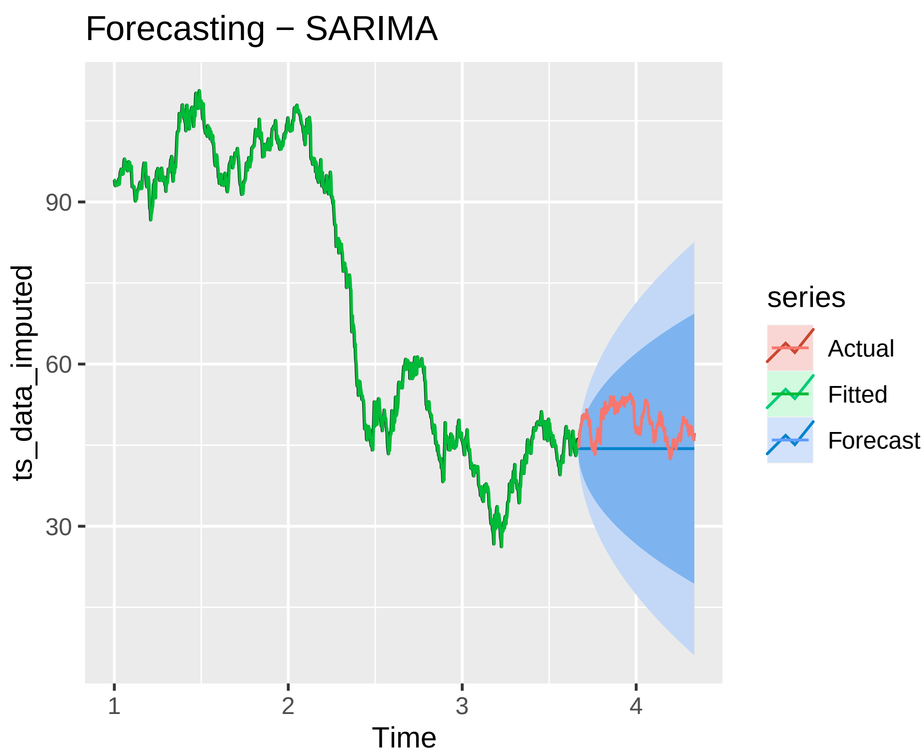
Plot for ARIMA

```r
autoplot(ts_data_imputed) +
  autolayer(fitted(model_results$ARIMA$model), series = paste("Fitted", model_results$ARIMA$name)) +
  autolayer(model_results$ARIMA$forecast_values, series = paste("Forecast", model_results$ARIMA$name)) +
  autolayer(test_data, series = "Actual") +
  ggtitle(paste("Forecasting - ARIMA"))
```



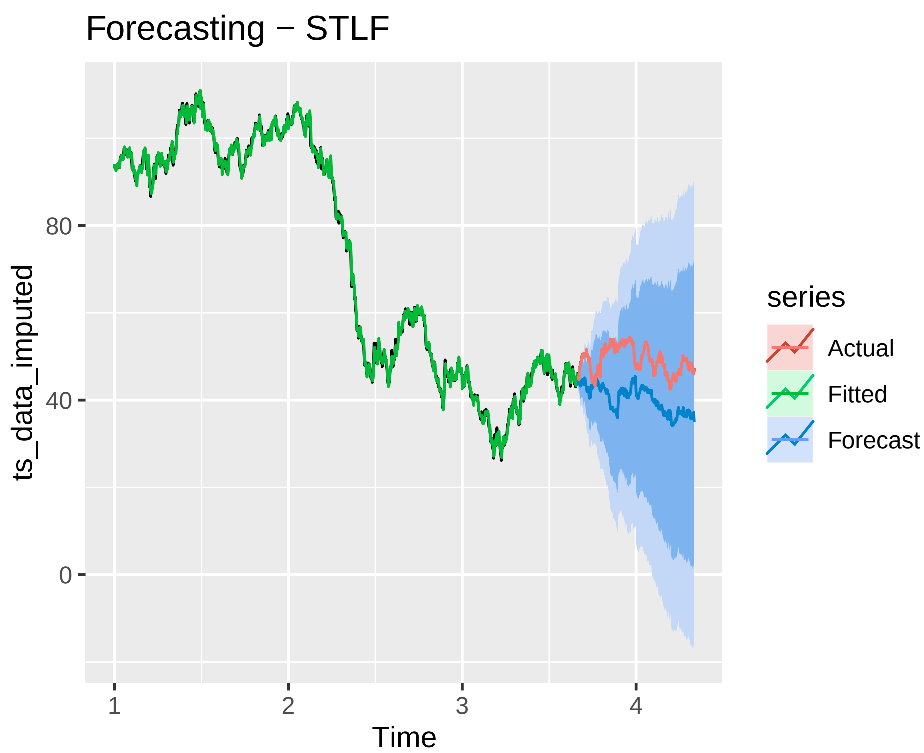
Plot for SARIMA

```r
autoplot(ts_data_imputed) +
  autolayer(fitted(model_results$SARIMA$model), series = paste("Fitted", model_results$SARIMA$name)) +
  autolayer(model_results$SARIMA$forecast_values, series = paste("Forecast", model_results$SARIMA$name)
  autolayer(test_data, series = "Actual") +
  ggtitle(paste("Forecasting - SARIMA"))
```



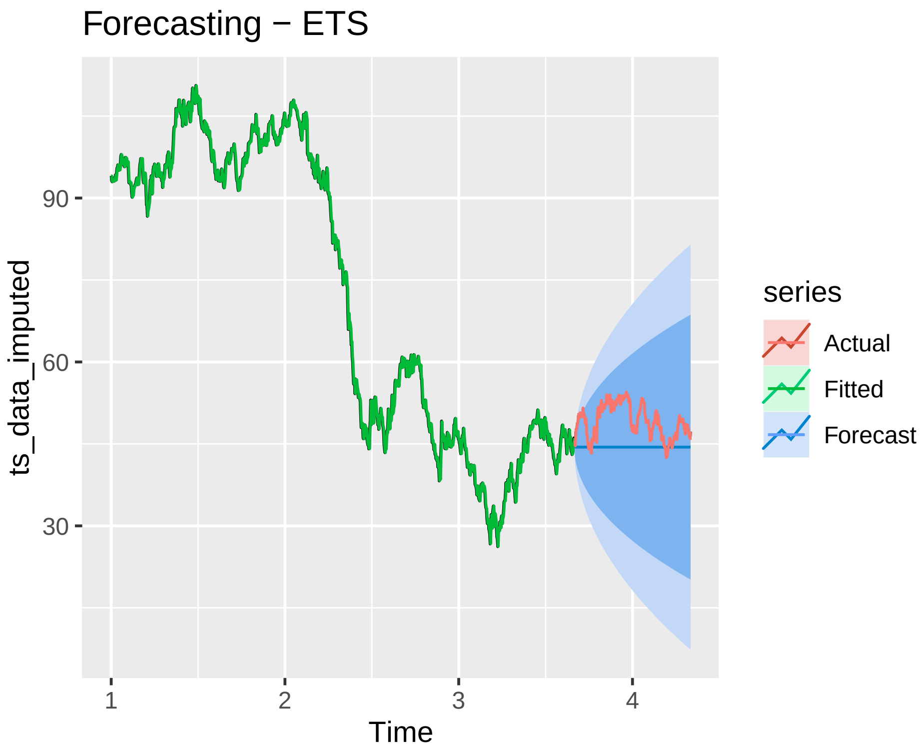
Plot for STLF

```r
autoplot(ts_data_imputed) +
  autolayer(fitted(model_results$STLF$model), series = paste("Fitted", model_results$STLF$name)) +
  autolayer(model_results$STLF$forecast_values, series = paste("Forecast", model_results$STLF$name)) +
  autolayer(test_data, series = "Actual") +
  ggtitle(paste("Forecasting - STLF"))
```



Plot for ETS

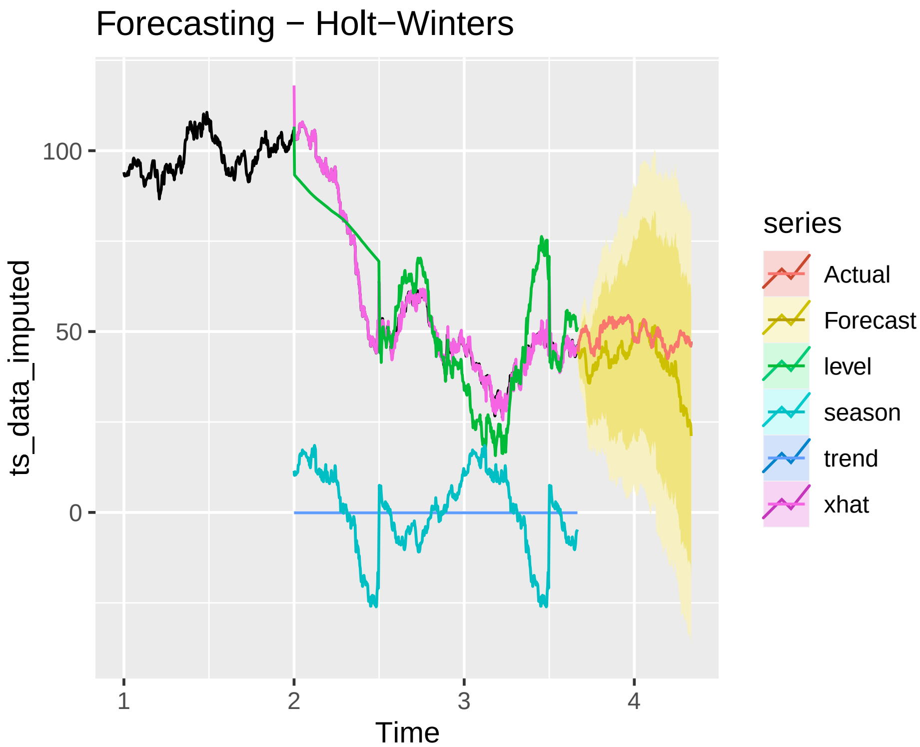
```r
autoplot(ts_data_imputed) +
  autolayer(fitted(model_results$ETS$model), series = paste("Fitted", model_results$ETS$name)) +
  autolayer(model_results$ETS$forecast_values, series = paste("Forecast", model_results$ETS$name)) +
  autolayer(test_data, series = "Actual") +
  ggtitle(paste("Forecasting - ETS"))
```



Plot for Holt-Winters

```
autoplot(ts_data_imputed) +
  autolayer(fitted(model_results$`Holt-Winters`$model), series = paste("Fitted", model_results$`Holt-Wi
  autolayer(model_results$`Holt-Winters`$forecast_values, series = paste("Forecast", model_results$`Hol
  autolayer(test_data, series = "Actual") +
  ggtitle(paste("Forecasting - Holt-Winters"))
```

```
## For a multivariate time series, specify a seriesname for each time series. Defaulting to column name
```



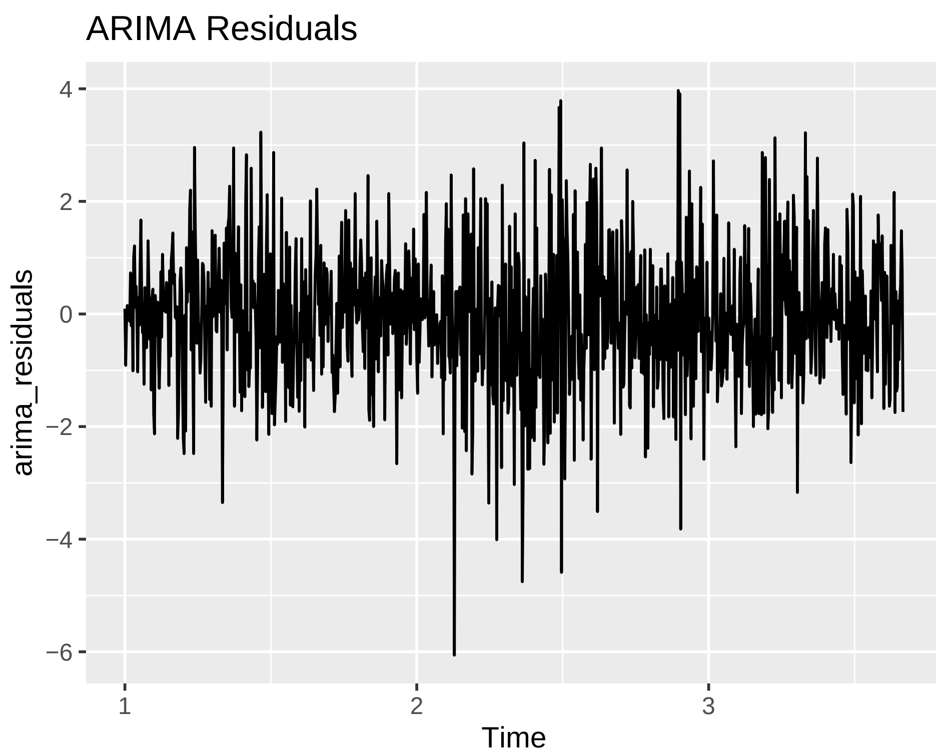
Checking the Adequacy by performing Residual Analysis

1. The Box-Ljung test is a statistical test used in time series analysis to determine if the residuals of a model exhibit significant autocorrelation at lags other than zero.

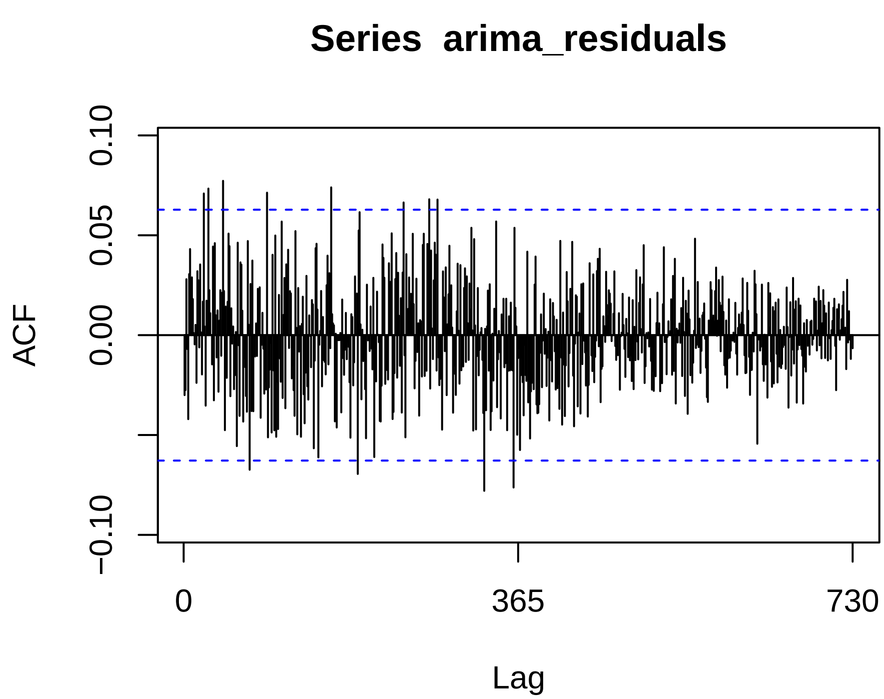
A low p-value indicates that the model's residuals deviate from the assumption of no autocorrelation. On the other hand, a high p-value suggests that the residuals are consistent with the assumption of no autocorrelation, providing support for the adequacy of the model in terms of capturing temporal dependencies in the data.

ARIMA Model Adequacy Check

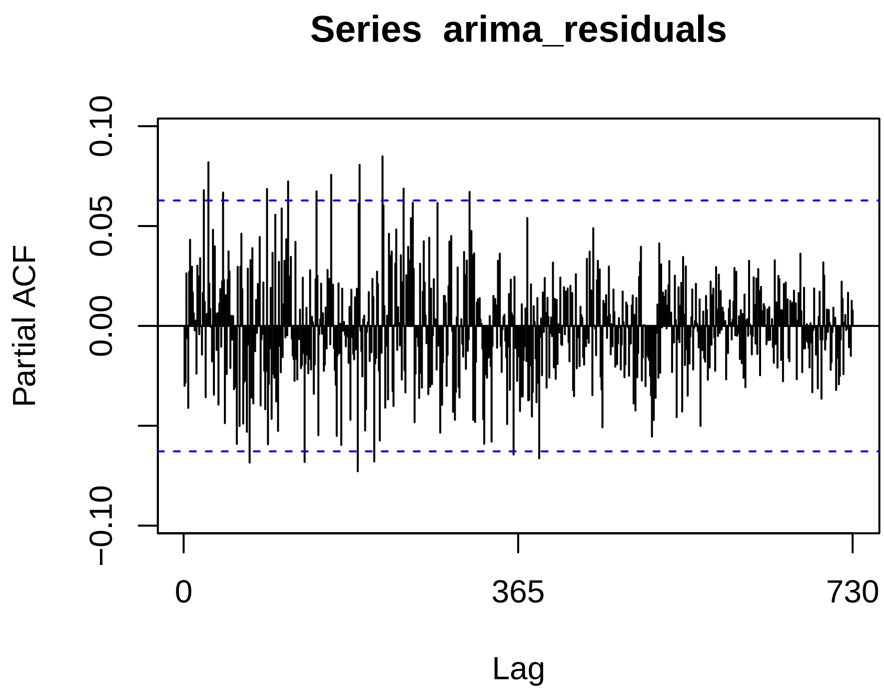
```r
library(forecast)
library(ggplot2)
arima_model <- auto.arima(train_data,seasonal = FALSE)
arima_residuals <- residuals(arima_model)

# Plot the Residuals
autoplot(arima_residuals) +
  ggtitle("ARIMA Residuals")
```




```r
# ACF and PACF of residuals
Acf(arima_residuals)
```



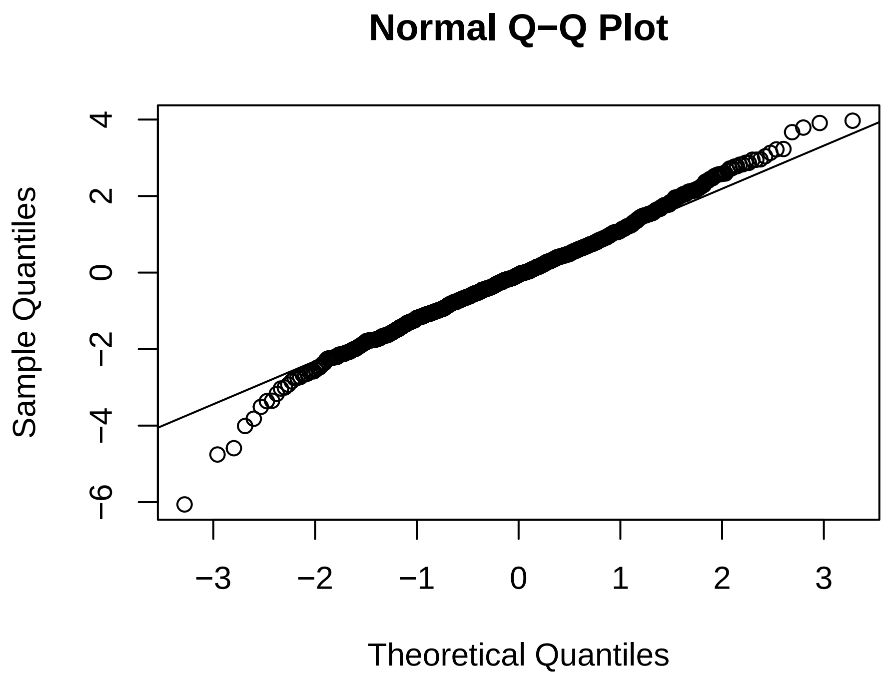
```r
Pacf(arima_residuals)
```




```r
# Ljung-Box Test
Box.test(arima_residuals, lag = 20, type = "Ljung-Box")
```

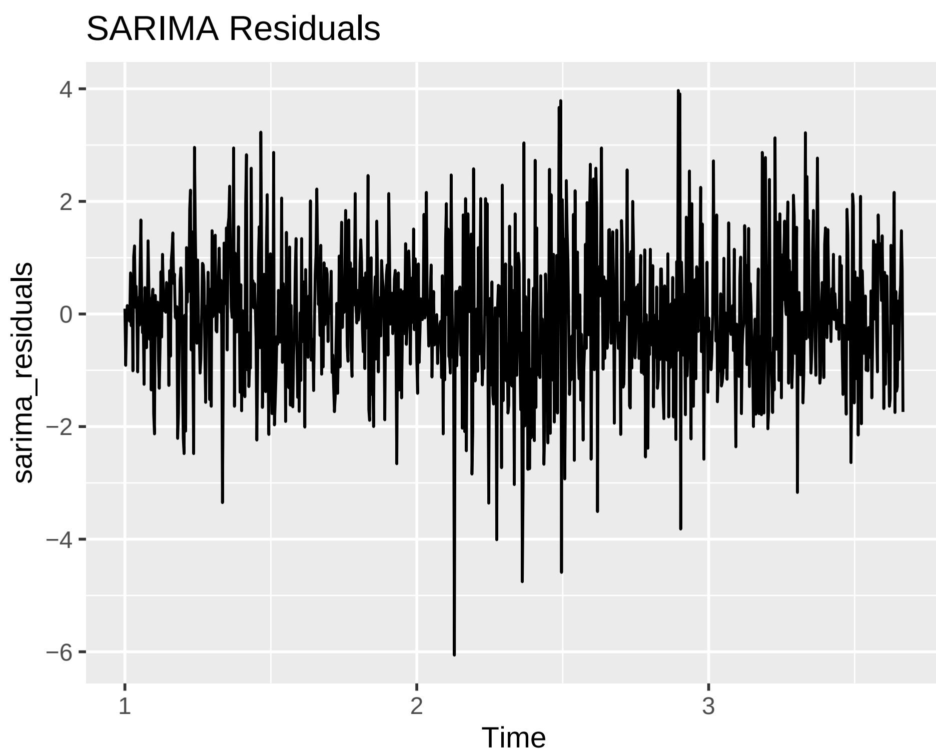
```
##
##  Box-Ljung test
##
## data:  arima_residuals
## X-squared = 12.125, df = 20, p-value = 0.9117
```

```r
# Normality Test

qqnorm(arima_residuals)
qqline(arima_residuals)
```

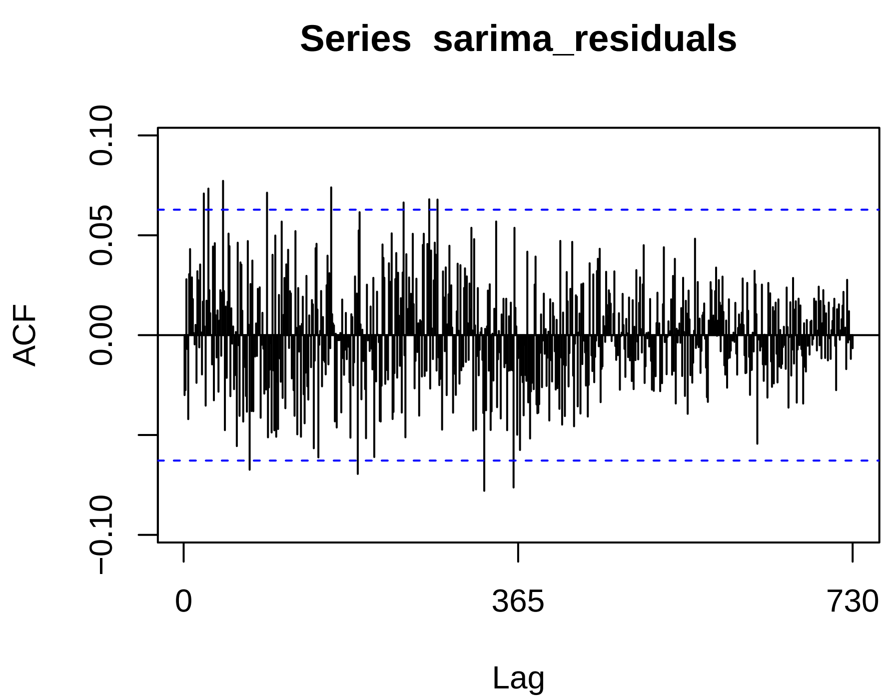


SARIMA Model Adequacy Check

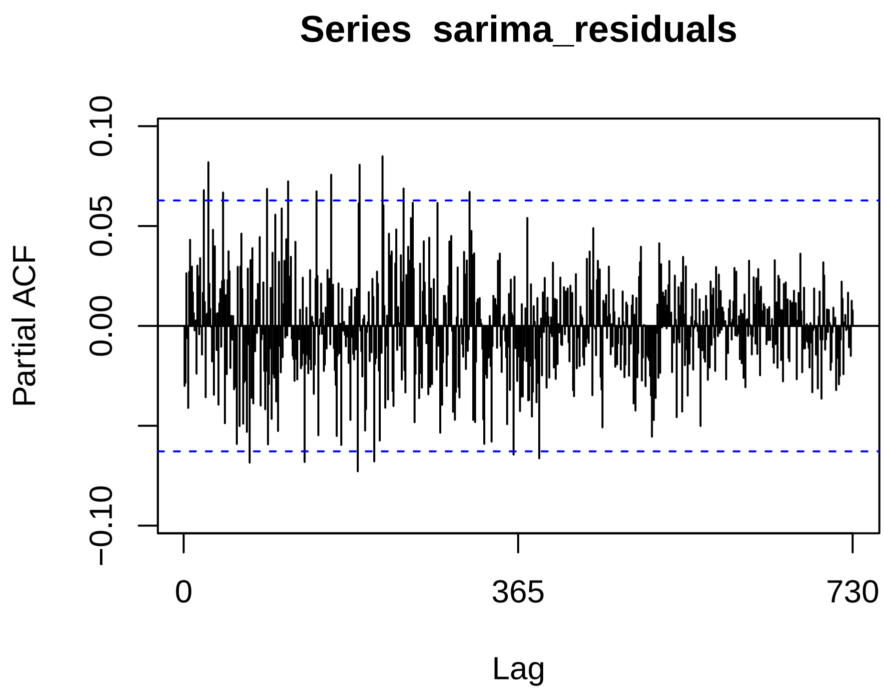
```r
sarima_model <- auto.arima(train_data,seasonal = TRUE)
sarima_residuals <- residuals(sarima_model)

# Plot residuals
autoplot(sarima_residuals) +
  ggtitle("SARIMA Residuals")
```




```r
# ACF and PACF of residuals
Acf(sarima_residuals)
```



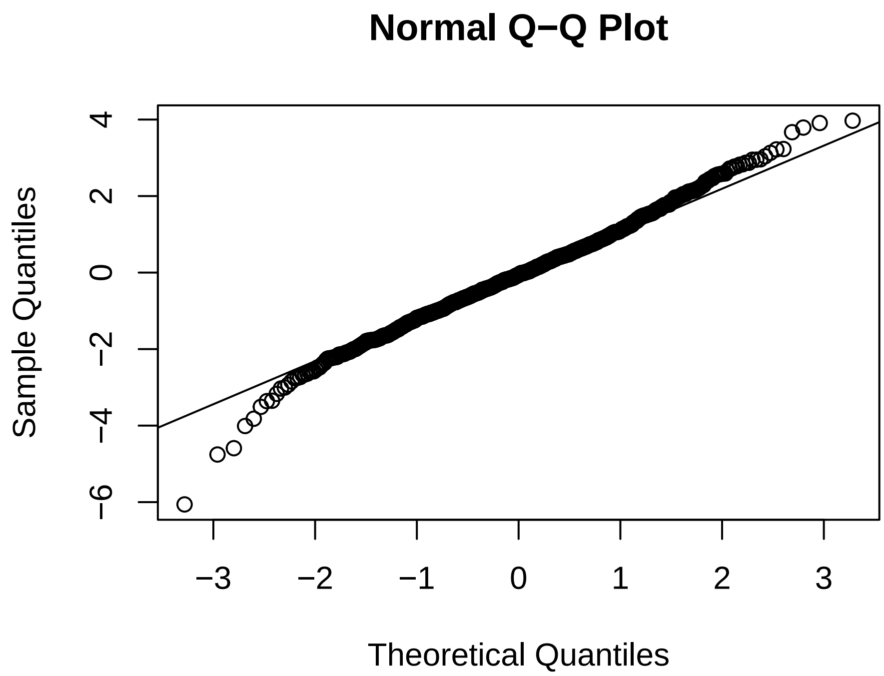
```r
Pacf(sarima_residuals)
```




```r
# Ljung-Box Test
Box.test(sarima_residuals, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  sarima_residuals
## X-squared = 12.125, df = 20, p-value = 0.9117
```

```r
# Normality Test
qqnorm(sarima_residuals)
qqline(sarima_residuals)
```

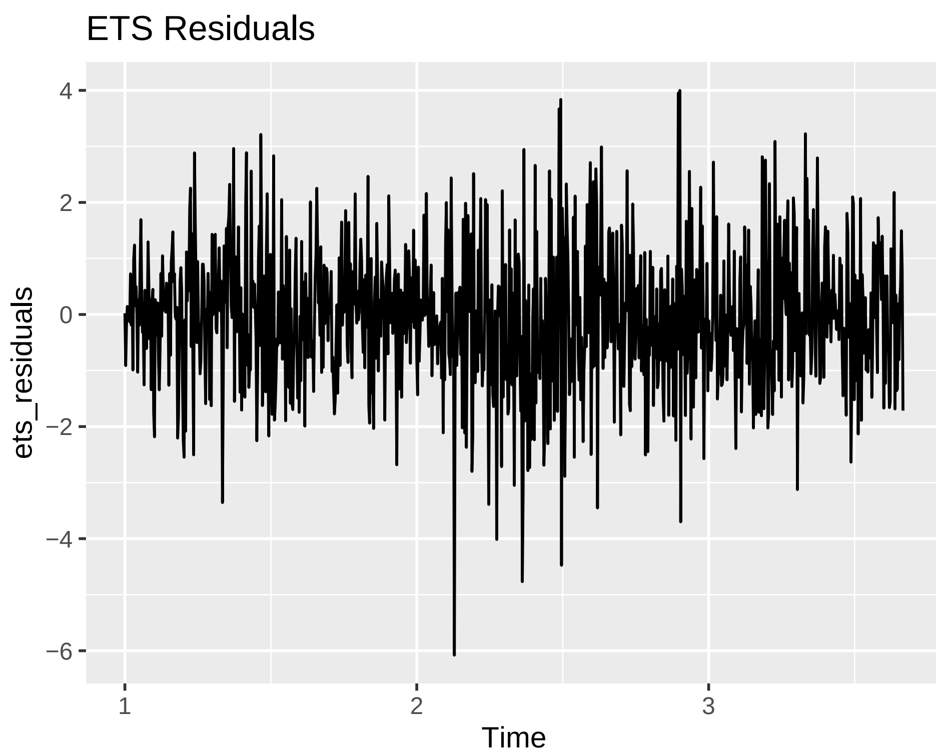


ETS Model Adequacy Check

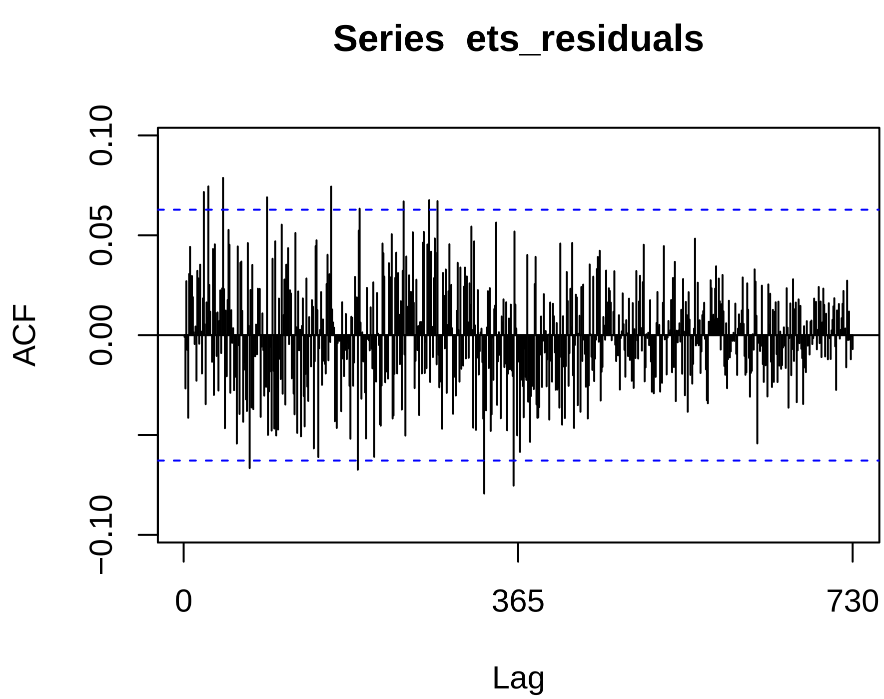
```
ets_model <- ets(train_data)
```

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

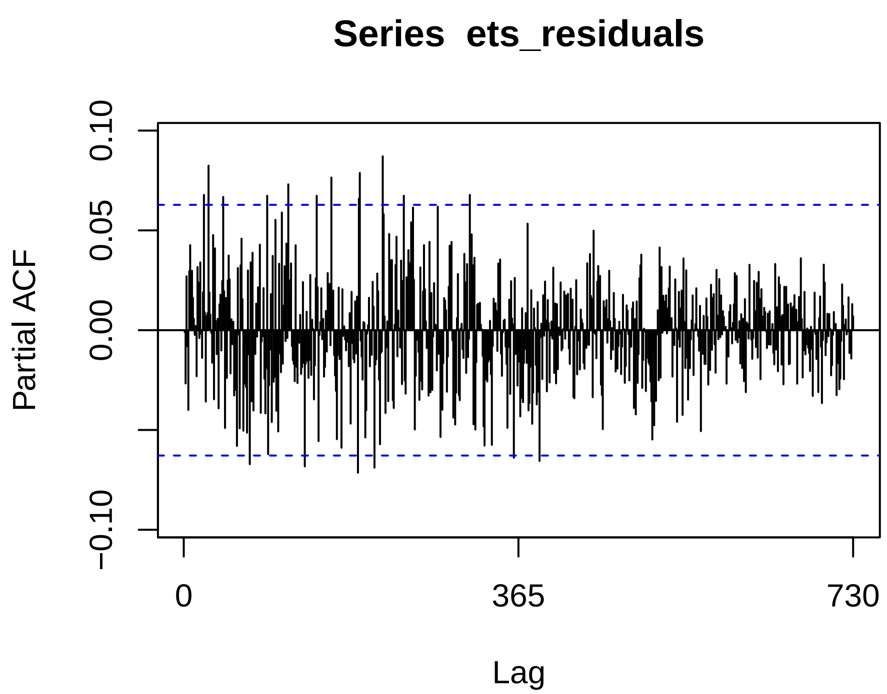
```
ets_residuals <- residuals(ets_model)

# Plot residuals
autoplot(ets_residuals) +
  ggtitle("ETS Residuals")
```




```r
# ACF and PACF of residuals
Acf(ets_residuals)
```



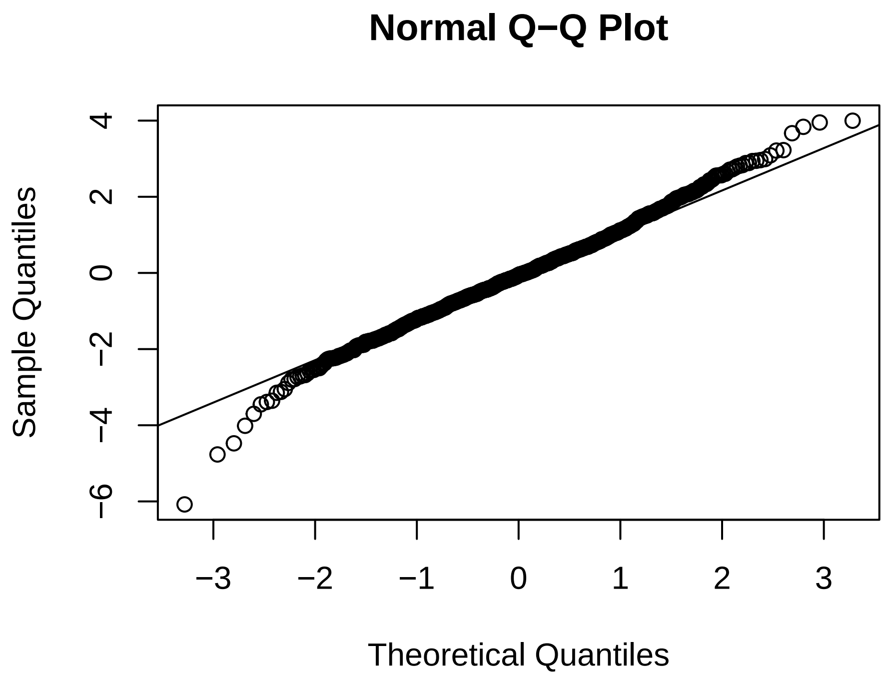
```r
Pacf(ets_residuals)
```




```r
# Ljung-Box Test
Box.test(ets_residuals, lag = 20, type = "Ljung-Box")
```

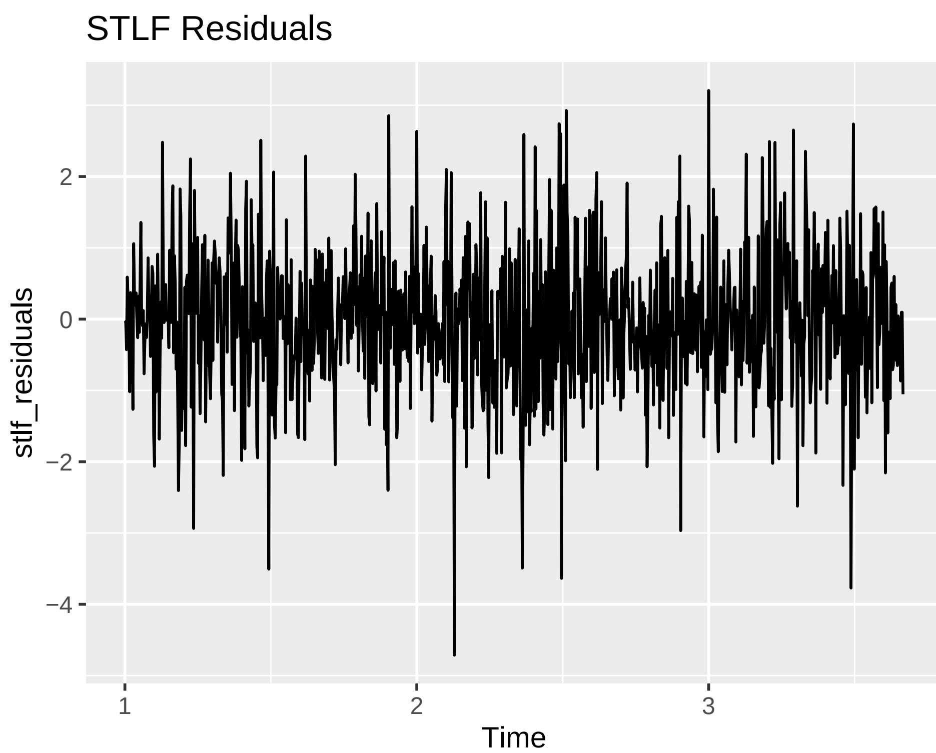
```
##
##  Box-Ljung test
##
## data:  ets_residuals
## X-squared = 11.251, df = 20, p-value = 0.9395
```

```r
# Normality Test
qqnorm(ets_residuals)
qqline(ets_residuals)
```

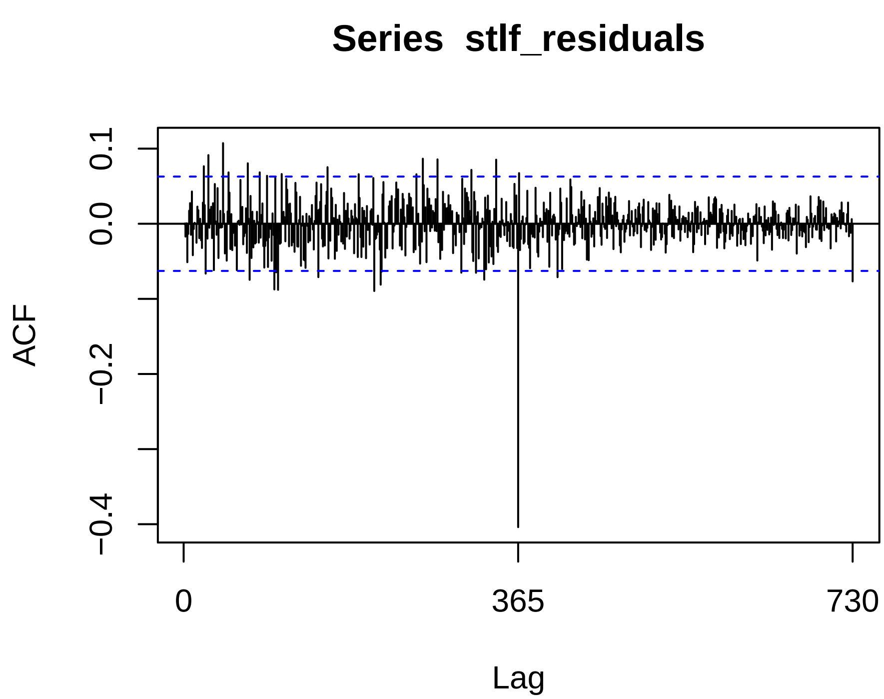


STLF Model Adequacy Check

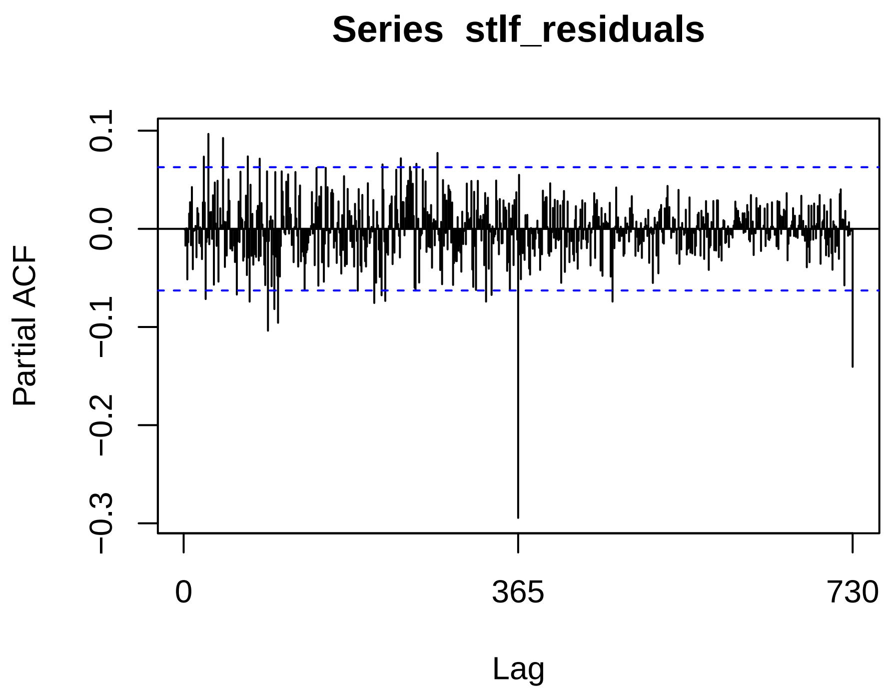
```
stlf_model <- stlf(train_data)
stlf_residuals <- residuals(stlf_model)

# Plot residuals
autoplot(stlf_residuals) +
  ggtitle("STLF Residuals")
```




```r
# ACF and PACF of residuals
Acf(stlf_residuals)
```



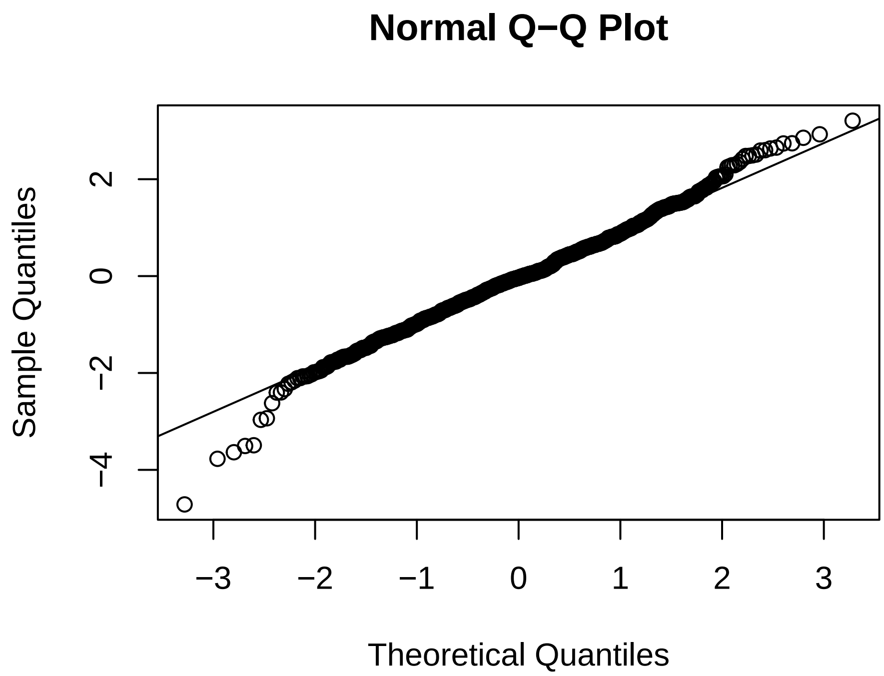
```r
Pacf(stlf_residuals)
```




```r
# Ljung-Box Test
Box.test(stlf_residuals, lag = 20, type = "Ljung-Box")
```

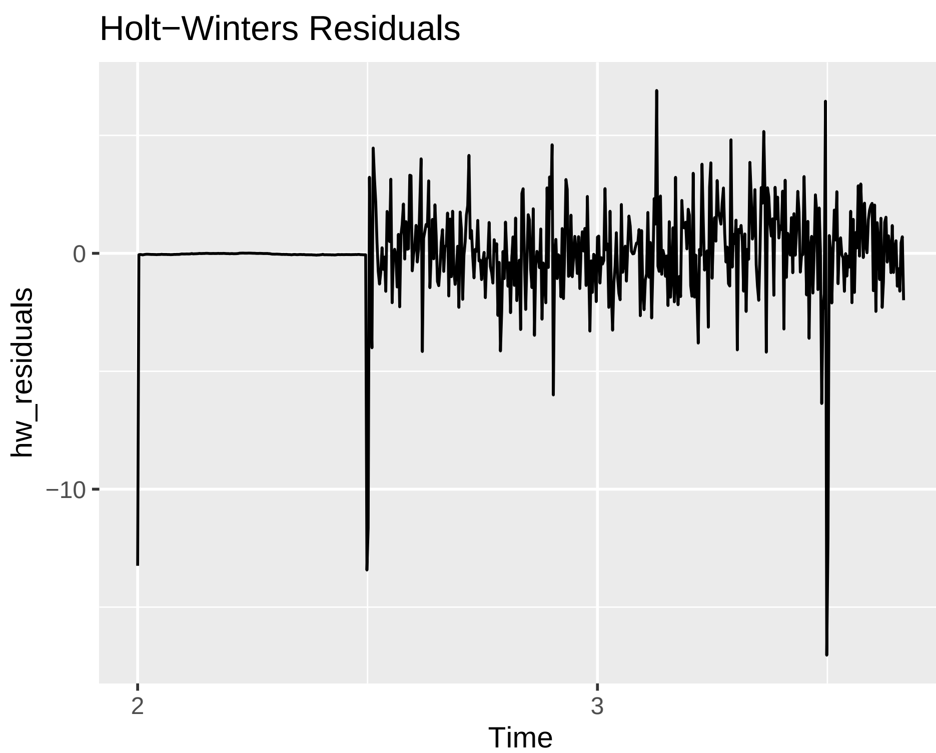
```
##
##  Box-Ljung test
##
## data:  stlf_residuals
## X-squared = 11.288, df = 20, p-value = 0.9384
```

```r
# Normality Test
qqnorm(stlf_residuals)
qqline(stlf_residuals)
```

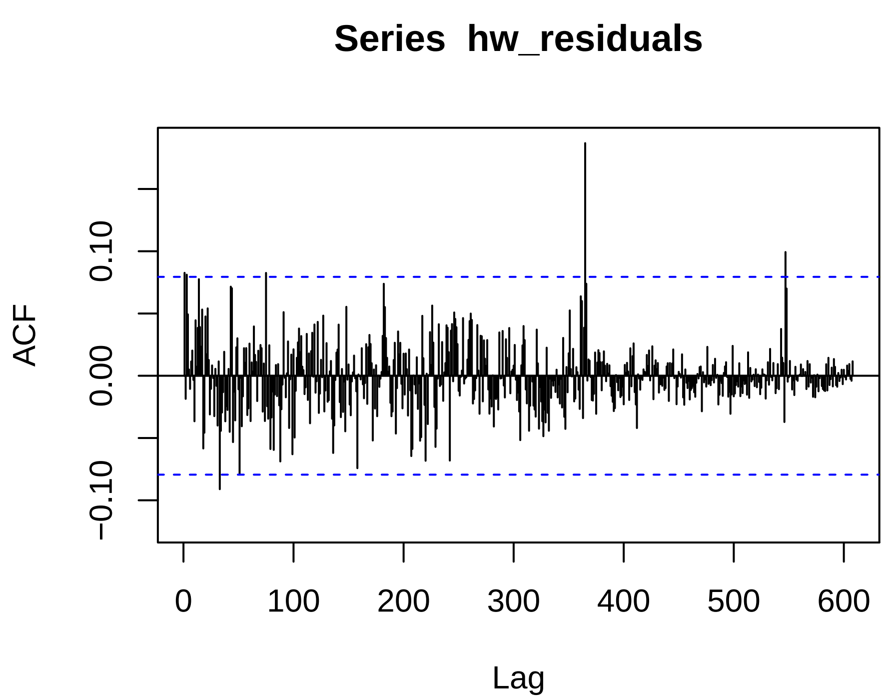


Holt-Winters Model Adequacy Check

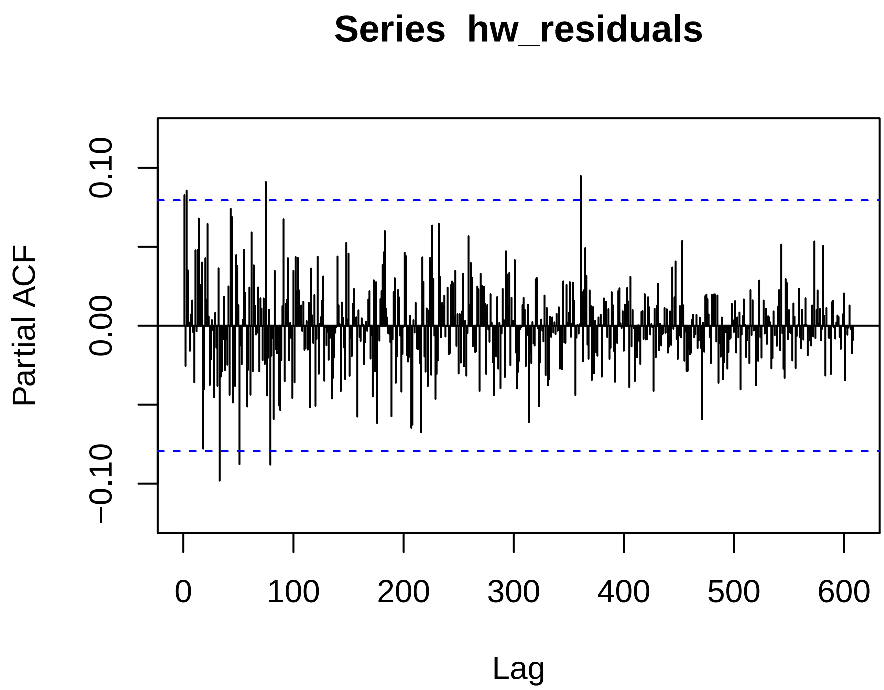
```
hw_model <- HoltWinters(train_data)
hw_residuals <- residuals(hw_model)

# Plot residuals
autoplot(hw_residuals) +
  ggtitle("Holt-Winters Residuals")
```




```r
# ACF and PACF of residuals
Acf(hw_residuals)
```



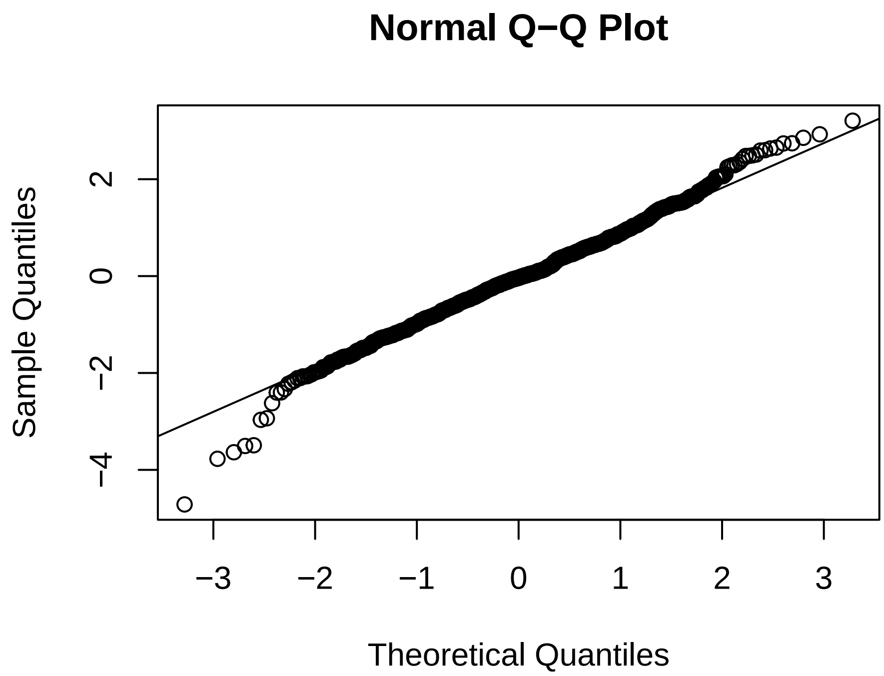
```r
Pacf(hw_residuals)
```




```r
# Ljung-Box Test
Box.test(hw_residuals, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  hw_residuals
## X-squared = 25.123, df = 20, p-value = 0.1968
```

```r
# Normality Test
qqnorm(stlf_residuals)
qqline(stlf_residuals)
```



The above specified models have a high p values for Ljung-Box test which suggests that there is no autocorrelation in the residuals at any lag for residuals and the residuals fall on the line in QQ plot which suggests that the models are appriate and are adequate to the given data.

Comparing MAE for each model

```
models <- list(
  arima_model = arima_model,
  sarima_model = sarima_model,
  ets_model = ets_model,
  stlf_model = stlf_model,
  hw_model = hw_model
)

# Assuming you have a validation dataset named validation_data
validation_data <- test_data

# Function to calculate MAE for a model on a validation dataset
calculate_mae <- function(model, validation_data) {
  forecast_values <- forecast(model, h = length(validation_data))
  actual_values <- validation_data
  mae <- mean(abs(forecast_values$mean - actual_values))
  return(mae)
}
```

```r
# Apply the function to each model
mae_values <- sapply(
  models,
  function(model) calculate_mae(model, validation_data)
)

# Display the MAE values
print(mae_values)
```

```
##  arima_model sarima_model    ets_model   stlf_model     hw_model
##     4.913444     4.913444     4.866750     8.926963     7.508847
```

Conclusion:

In the context of this particular project, it is noteworthy to highlight that the Exponential Triple Smoothing (ETS) model emerged with the most favorable results, exhibiting the lowest Root Mean Squared Error (RMSE)(for training set = 1.247309, for test set = 5.643381) and Mean Absolute Error (MAE)(4.866750) values among the models considered. This compelling performance leads us to conclude that, based on the metrics employed, the ETS model stands out as the most optimal and well-suited choice for this specific scenario.