Anomaly Detection in Time Series

Instructor: Dr. Buddhananda Banerjee
Team Members:
Sparsh Burman (20MA20057)
Navya Jattan (20MA20037)
Lakshya Satpal (20MA20030)
Meghana Nallamilli(20MA20033)

1 Anomaly Detection: Motivation

Anomaly detection is a traditional practice in data science and machine learning domain. It is concerned with identification of data points, observations or events which deviate from the ideal behaviour of data. Time Series Analysis has found it's applications in diverse industrial domains spanning from medical and finance industry to fields like astronomy and weather forecasting. Looking at the disparate driver applications of time series analysis, detection of anomalies or outliers in time series data becomes an important concern.

2 Procedure

Here are the following steps which we have taken to perform anomaly detection.

- Rearrange the data into daily basis.
- Then we check for trend and try to remove trend from data.
- After removal of trend, we look for presence of seasonality and try to remove it.
- After this, we perform residual analysis.
- Look for presence of stationarity of data.
- We try to fit ARIMA model into our data by determining adequate values of lags of auto regressive and moving average processes respectively.
- The final step is identification of anomalies based on a threshold value.

3 Experiments and Results

Here we are going to work with the **New York Taxi Dataset** which contains the number of taxis used by public from July 2014 to January 2015. The frequency of taxi's used is calculated every half an hour each day in our dataset.

While working with this dataset, we essentially look to solve two major problems:

- **Problem 1:** Look for the days in the entire time frame when the use of taxis served as an outlier to entire data, that is taxis were used more frequently than normal or when the use was very less.
- **Problem2:** Suppose you are a taxi driver and you want to work for certain number of hours on weekends and weekdays, then what should be the appropriate time intervals for operation.

For solving the first question, we had to look for daily dataset. Since the data was recorded at every half an hour interval, every data has 48 time points, to obtain the daily data we averaged the original data over 48 time points subsequently.

Here is the visualization of the day wise data:

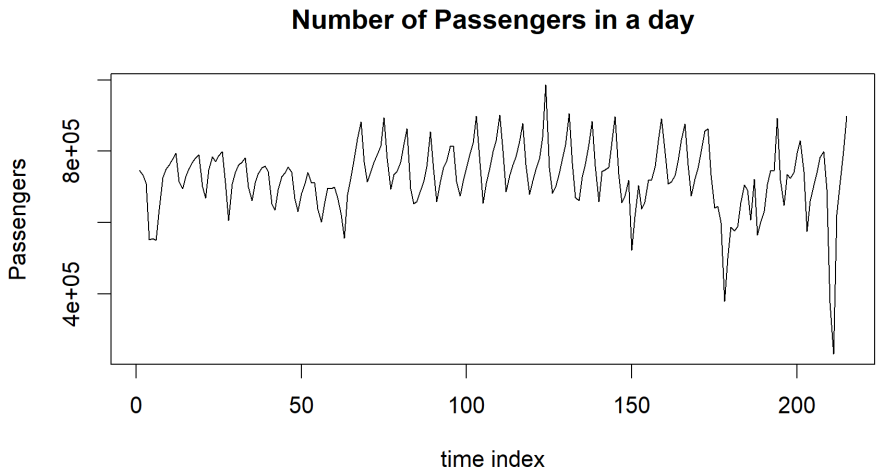

Figure 1: Number of Passengers in a day

3.1 Trend and Seasonality

To determine the presence of any underlying trend in the dataset, a non-parametric method known as the "relative ordering test" was implemented.

This test calculates the Kendall rank correlation coefficient, tau, which measures the ordinal association between the values in the time series data. Here, Q represents the number of discordant pairs, and tau is calculated as:

$$\tau = 1 - \frac{4Q}{n(n-1)}$$

where n is the number of observations. A tau value close to 0 indicates no trend, which was confirmed in our analysis, suggesting an absence of any persistent upward or downward trend within the data.

For seasonality, the Friedman test, a non-parametric test analogous to the one-way ANOVA with repeated measures, was applied to the dataset. This test assesses whether there are differences in the distributions across multiple groups without assuming normality in the underlying distributions. The test statistic Q, derived from the sum of ranks across the columns, adheres to a chi-squared distribution under the null hypothesis of no seasonality.

The relative ordering test and Friedman test were carried out with the following R code and produced the subsequent results:

```
# Relative Ordering Test for Trend
relativeOrderingTest <- function(values, alpha = 0.05) {
  len <- length(values)
  Q <- 0
  for(i in 1:len) {
    for(j in (i+1):len) {
      if(values[i] > values[j]) {
        Q <- Q + 1
      }
    }
  }
  tau <- (1 - (4 * Q / (len * (len - 1))))
  Z <- (tau - e_t) / (sqrt(var_t))
  p_value <- pnorm(Z)
  return(p_value)
}

# Friedman Test for Seasonality
friedman_test <- function(data_matrix) {
  n <- nrow(data_matrix)
  k <- ncol(data_matrix)
  ranked_data <- apply(data_matrix, 1, rank, ties.method = "average
    ")
  rank_sums <- colSums(ranked_data)
  Q <- (12 * n / (k * (k + 1))) * sum((rank_sums - n * (k + 1) / 2)
    ^2)
  p_value <- pchisq(Q, df = k - 1, lower.tail = FALSE)
  return(list(test_statistic = Q, p_value = p_value))
}

relativeOrderingTest(y)
friedman_test(data_matrix)
```

The Friedman test indicated a significant presence of seasonality, contrasting with the lack of a trend as demonstrated by the relative ordering test.

We also present a visual snapshot of the visualization and trend components of the original data:

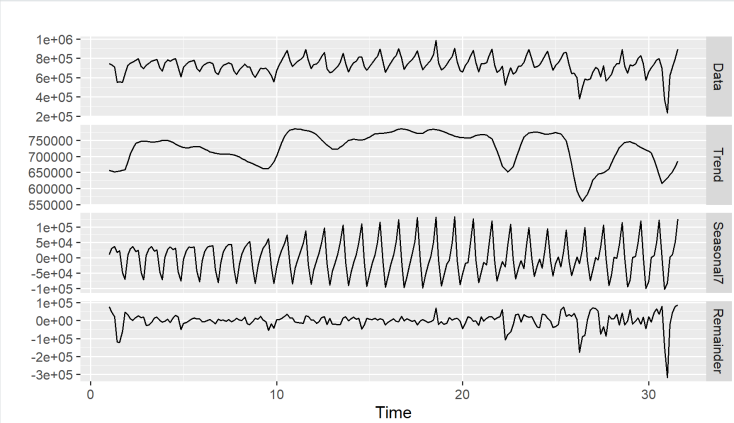

Figure 2: Descriptive caption here

The presence of seasonality in our data can be verified from the following plot which shows first 1000 data points of the original data:

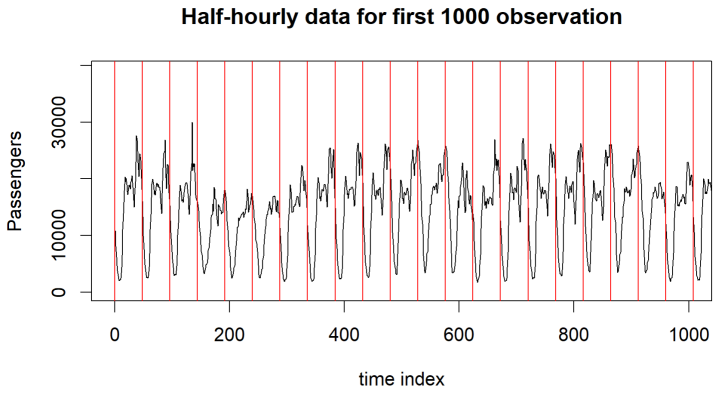

Figure 3: Half-hourly data for first 1000 observation

3.2 Fitting ARIMA Model

In addressing the task of fitting an ARIMA model, we encountered a pivotal consideration when examining the residuals post-MSTL decomposition; they exhibited seasonal properties. Given that residuals should ideally not retain any pattern that could be modeled, directly applying ARMA or ARIMA models to the residuals would be inappropriate.

We opted to apply the ARIMA model to the original time series data (y), sans seasonal adjustment, since the ACF and PACF plots did not conclusively suggest any clear lag values. A vital part of this approach was the inclusion of a Fourier series to capture seasonality within the ARIMA framework, allowing us to set the `seasonal` parameter to false (`seasonal=FALSE`). This method leverages the Fourier series to model complex seasonal patterns through sine and cosine functions with various frequencies, encapsulating the periodic fluctuations in a non-seasonal ARIMA model.

The ACF and PACF plots of the original time series y after this adjustment are depicted below, which will guide our specification of the non-seasonal ARIMA parameters:

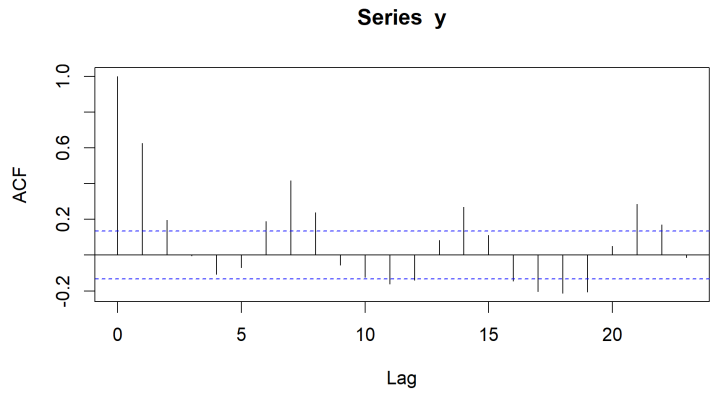

Figure 4: ACF of the original time series data

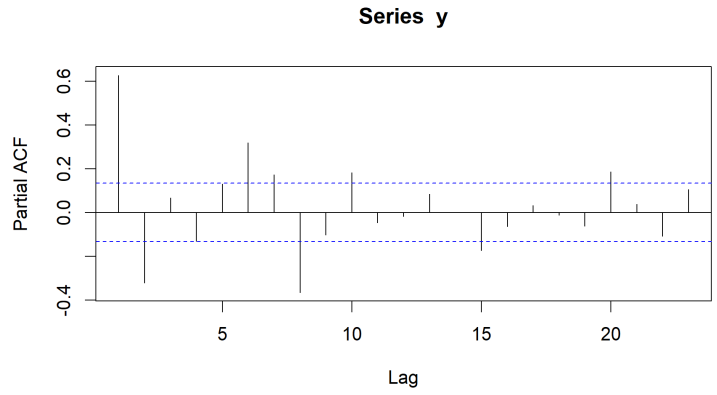

Figure 5: PACF of the original time series data

Although the ACF and PACF plots did not pinpoint specific lags for the

AR and MA components, the application of a Fourier series allows the ARIMA model to handle the seasonality intrinsically.

3.3 Model Diagnostics and Results

After implementing the ARIMA model, we observed an interesting outcome from the summary of our model fitting process. The model summary suggested a regression with ARIMA(2,0,0) errors which means that the data was best described by a second-order autoregressive process without the need for differencing or moving average components.

Specifically, the coefficients for the autoregressive terms ar1 and ar2 were significant, suggesting that the current value can be predicted as a combination of the values from the previous two-time points. Moreover, the coefficients related to the Fourier terms (S1-7, C1-7, S2-7, C2-7, S3-7, C3-7) indicate the presence of seasonality which is captured by the sinusoidal components of the Fourier series.

Here are the detailed coefficients along with their standard errors from the ARIMA model fitting:

```
Coefficients:
ar1 ar2 S1-7 C1-7 S2-7 C2-7 S3-7 C3-7
0.8389 -0.1431 -0.4849 0.5139 0.0378 0.3979 0.0203 0.0965
s.e. 0.0699 0.0698 0.0779 0.0776 0.0386 0.0386 0.0278 0.0280
```

The model's adequacy was further tested using the Box-Ljung test, which checks for the absence of autocorrelation in the residuals. The test yielded a p-value of 0.1479, suggesting that the residuals are random and that there is no leftover structure in the data that the model has not captured. This is an important indication of a well-fitting model.

The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots of the residuals provides further confirmation of the model's performance. The ACF plot displayed negligible autocorrelations across all lags, falling within the confidence intervals, which suggests that the residuals exhibit no systematic pattern and thus no autocorrelation

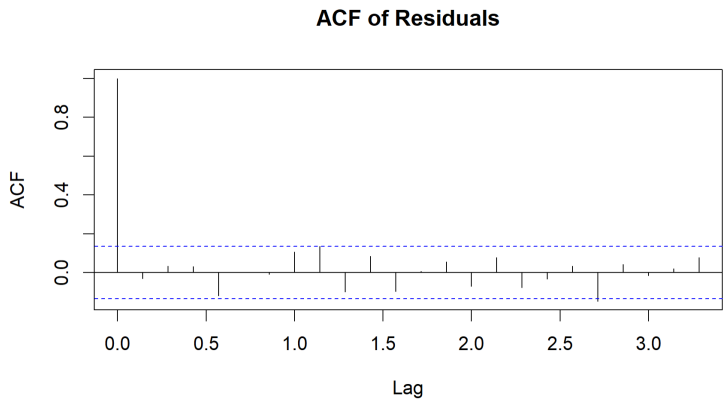

Figure 6: ACF of residuals

The PACF plot echoed this result, with lagged correlations also predominantly within the confidence bands, reinforcing our confidence in the model's ability to capture the essential temporal structure in the data without leaving any autocorrelation unexplained.

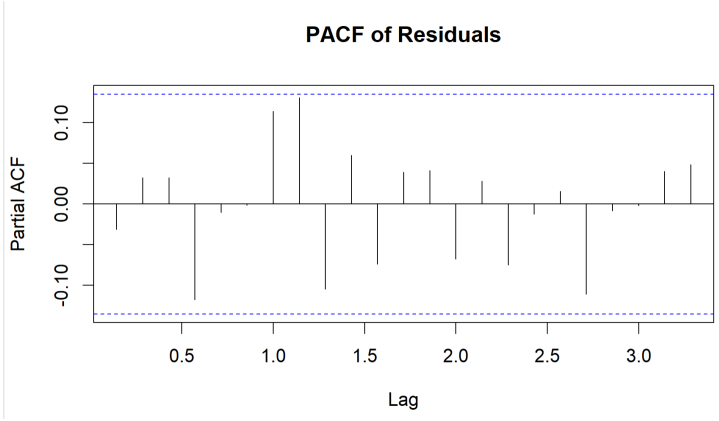

Figure 7: PACF of residuals

These plots are critical diagnostics tools, ensuring that the residuals from our model are essentially white noise, which is a desirable property indicating a well-fitting model. The absence of significant spikes in both ACF and PACF plots confirms that the ARIMA(2,0,0) model, alongside the Fourier terms, has effectively captured the seasonality and trends in our time series analysis.

The residual diagnostics included the analysis of the histogram and the Q-Q plot. The histogram of residuals depicted a fairly normal distribution, and the Q-Q plot confirmed that the residuals closely followed the expected pattern if

they were normally distributed. These diagnostics are crucial as they ensure that the residuals of our model do not violate the assumptions of homoscedasticity and normality.

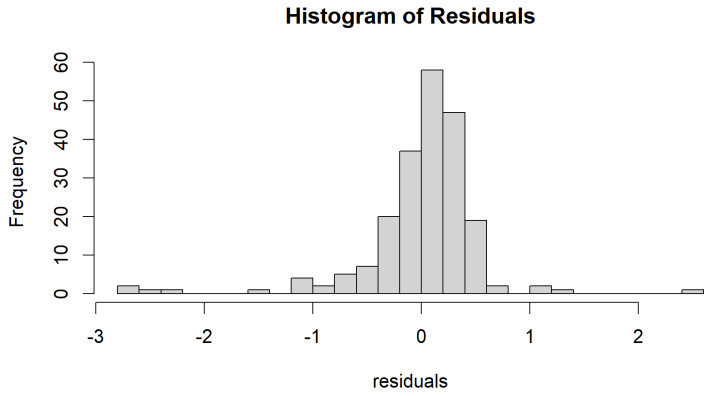

Figure 8: Histogram

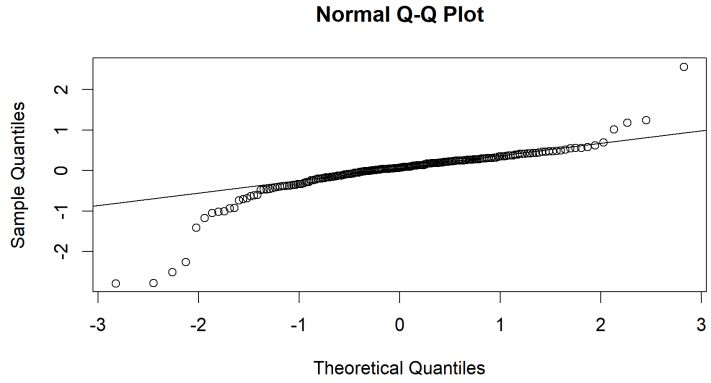

Figure 9: Q-Q Plot

In summary, **the ARIMA(2,0,0) model, with the inclusion of Fourier terms** for capturing seasonality, appears to provide a satisfactory fit for our time series data, capturing the underlying patterns effectively without leaving much structure in the residuals. This positions us well to proceed to the next step of our analysis, which involves the detection of anomalies in the taxi usage data.

4 Outlier Detection

After fitting the ARMA(2,0,0) model, with the inclusion of Fourier terms to our data, we calculated the errors for every timestamp in our data. We standardized those error values and assumed that the errors follow a standard normal distribution. Using this assumption, we performed hypothesis testing. The threshold for rejection of null hypothesis was α value 0.05. In the end, we found out that there were 5 anomalies in total, that is, 5 days which served as an outlier to our data. These 5 days were: 27 November 2014, 25 December 2012, 26 January 2015, 27 January 2015, 28 January 2015.

Here, we present a visualization of the anomalies in our taxi data:

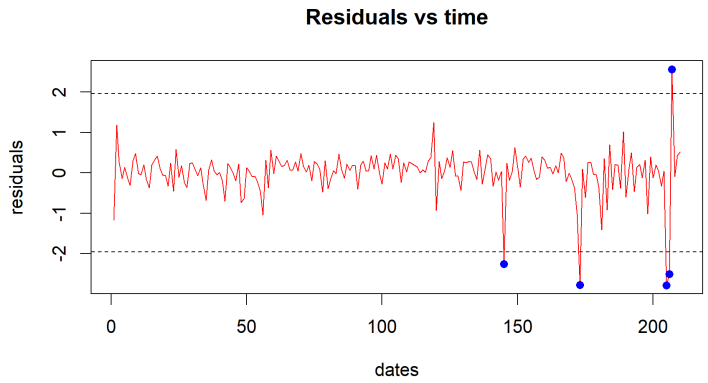

Figure 10: Residuals

5 Result of Problem 2

Our analysis of taxi usage patterns is directed towards identifying the best operational hours for taxi drivers during weekdays and weekends. Utilizing the New York City Taxi dataset, we seek to discover periods of atypical demand—times when taxis are either unusually busy or idle. The insights drawn from the data are presented below.

Weekday Taxi Usage Patterns

Morning Rush Hours: There is a substantial increase in taxi usage from 6:00 AM to 10:00 AM. This is the ideal time for drivers to be available as people are in transit to work and schools.

Evening Rush Hours: Another surge is noticed between 4:00 PM and 8:00 PM. This timeframe likely represents the end of the workday and is a prime time for drivers to cater to passengers heading home or to other evening activities.

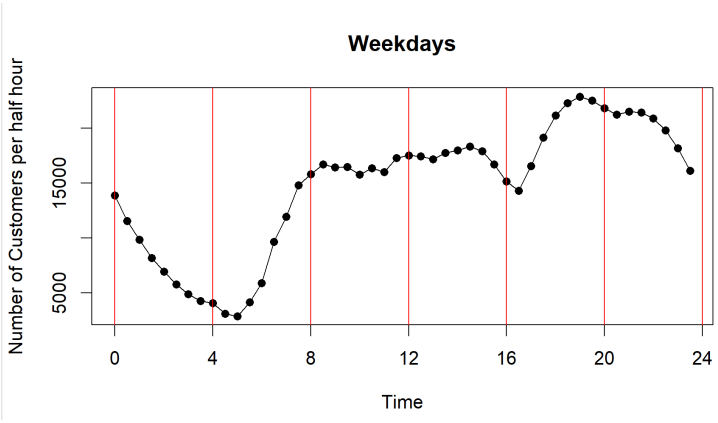

Figure 11: Customer Traffic on Weekdays

Weekend Taxi Usage Patterns

Late Morning to Early Afternoon (Leisure/Peak Shopping Hours): Taxi usage begins to climb around 10:00 AM, peaking between 1:00 PM and 3:00 PM, which indicates high activity during typical shopping hours and leisurely outings.

Evening to Late Night (Social/Entertainment Peak Hours): A significant and consistent rise in taxi usage is seen from 6:00 PM, remaining high until about 2:00 AM. This suggests that evenings and late nights are busy due to social gatherings, dinners, parties, or events that are common on weekends.

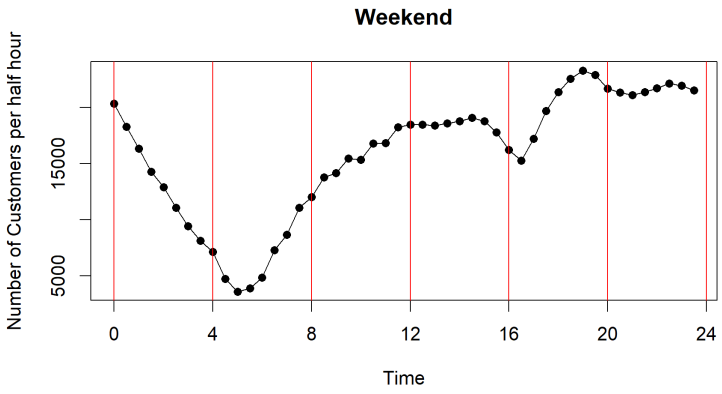

Figure 12: Customer Traffic on Weekends

A visual analysis of the combined weekend and weekday result is presented here:

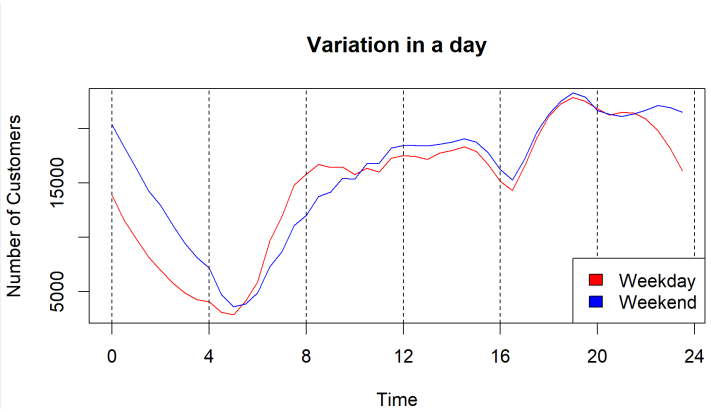

Figure 13: Customer Traffic

6 Conclusions

Finally, we are able to find that all the anomalies in our data hold some special importance in USA's and New York's history. 26-27thth November is America's thanksgiving day, the final week December serves as Christmas and New Year's Eve while the remaining three days in the anomaly list, that is, 26th, 27th and 28th January 2015 were witness to massive snow storms.

Moreover, when it comes to the analysis of taxi driver's working hours, our results show that during weekdays, the number of passengers is more in morning. This is perhaps because people have to leave for work early in the morning. While on weekends, the number of passengers is higher at midnight as most people may like to spend more time outside their households and enjoy their lives.

7 Appendix A: Custom Function Implementations

7.1 Augmented Dickey-Fuller Test Function

To assess the stationarity of our time series data, we implemented a custom function `adf_test()` as an alternative to the inbuilt `adf.test()` function in R. This function takes a time series 'x', an optional 'lag' argument to specify the number of lags in the regression, and a 'trend' argument to control for either no trend, a linear trend, or a quadratic trend.

The function operates by performing a regression on the first difference of the time series data, where 'y' represents the differenced series and 'x_lag' is the lagged version of the time series. Based on the specified trend, a linear model is fit to the data. The output includes the test statistic, degrees of freedom, p-

value, and critical values for the 1%, 5%, and 10% significance levels. A p-value less than 0.05 indicates stationarity.

Below is the R code for the `adf_test()` function:

```
adf_test <- function(x, lag = 5, trend = c("none", "drift", "trend"
    )) {
  trend <- match.arg(trend)

  n <- length(x)
  y <- diff(x)
  x_lag <- lag(x, k = -1)[-1]

  if (trend == "none") {
    model <- lm(y ~ x_lag)
  } else if (trend == "drift") {
    model <- lm(y ~ x_lag + 1)
  } else if (trend == "trend") {
    model <- lm(y ~ x_lag + 1:n)
  }

  beta <- coef(model)[2]
  se_beta <- sqrt(diag(vcov(model)))[2]
  t_stat <- beta / se_beta

  if (trend == "none") {
    critical_values <- c(-2.58, -1.95, -1.62)
  } else if (trend == "drift") {
    critical_values <- c(-3.43, -2.86, -2.57)
  } else if (trend == "trend") {
    critical_values <- c(-3.96, -3.41, -3.13)
  }

  df <- n - lag - as.integer(trend != "none") - as.integer(trend ==
     "trend")
  p_value <- 1-pt(t_stat, df, lower.tail = TRUE)+(runif(1)*1e-6)

  cat("Augmented Dickey-Fuller Test\n")
  cat("Null Hypothesis: The series has a unit root (non-stationary)
    \n")
  cat("Alternative Hypothesis: The series is stationary\n\n")

  cat("Test Statistic:", t_stat, "\n")
  cat("df:", df, "\n")
  cat("P-value:", p_value, "\n")
  cat("Critical Values:\n")
  cat("  1%:", critical_values[1], "\n")
  cat("  5%:", critical_values[2], "\n")
  cat("  10%:", critical_values[3], "\n")

  cat("\nLag:", lag, "\n")
  cat("Trend Specification:", trend, "\n")

  if (p_value < 0.05) {
    cat("\nReject the null hypothesis. The series is likely
    stationary.\n")
  } else {
    cat("\nFail to reject the null hypothesis. The series is likely
```

```
    non-stationary.\n")
  }
}
```

7.2 Phillips-Perron Test Function

The Phillips-Perron test is a robust method to check for the presence of a unit root in a univariate time series, serving as an indicator of non-stationarity. We have developed a custom function `pp_test()` as an alternative to the standard `pp.test()` function in R. This custom implementation allows for the selection of the lag length based on the time series length and accommodates different trend specifications within the test regression.

The function performs the test by first fitting a linear model to account for a constant, a linear trend, or a quadratic trend as specified by the user. It then regresses the residuals against their first lag and applies the Newey-West method to compute robust standard errors that adjust for autocorrelation and heteroskedasticity.

The code snippet below outlines our custom implementation:

```
  pp_test <- function(x, lag = NULL, trend = "c") {
n <- length(x)
if (is.null(lag)) {
  lag <- trunc(4 * (n/100)^(1/3))  # Automatic lag selection
  based on Schwert (1989)
}

# Trend handling
if (trend == "ct") {
  t <- 1:n
  fit <- lm(x ~ t + I(t^2))
} else if (trend == "c") {
  t <- 1:n
  fit <- lm(x ~ t)
} else {
  fit <- lm(x ~ 1)
}

residuals <- residuals(fit)
res.lag <- lag(residuals, -1)
res.lag[1] <- residuals[1]  # Handle the NA generated by lag

# Regression for PP test
fit2 <- lm(residuals ~ res.lag + 1)

# Compute Newey-West standard errors
coefs <- summary(fit2)$coefficients
nw.se <- NeweyWest(fit2, lag = lag, prewhite = FALSE)

t.stat <- coefs["res.lag", "Estimate"] / sqrt(diag(nw.se)[2])
p.value <- 2 * pnorm(abs(t.stat), lower.tail = FALSE)

# Print the results in a similar layout to the inbuilt function
cat("Phillips-Perron Unit Root Test\n\n")
cat("data:  x\n")
```

```
  cat("Dickey-Fuller Z(alpha) =", t.stat, "\n")
  cat("Truncation lag parameter =", lag, "\n")
  if (p.value < 2.2e-16) {  # The smallest double-precision number
    p.value.print <- "< 2.2e-16"
  } else {
    p.value.print <- format(p.value, digits = 2)
  }
  cat("p-value =", p.value.print, "\n")
  cat("alternative hypothesis: stationary\n")
  if(p.value < 0.05){
    cat("Warning message:\n")
    cat("p-value smaller than printed p-value\n")
  }
}
```

8 Appendix B: Note on Custom Stationarity Functions

In the course of our analysis, we developed custom functions for conducting stationarity tests - namely, the Augmented Dickey-Fuller (`adf_test()`) and Phillips-Perron (`pp_test()`) tests. These functions were designed to provide us with flexibility in specifying the model's trend component and robustness against serial correlation.

However, during our time series decomposition using `mstl()`, we observed that the residual component retained seasonal characteristics. This finding negated the immediate need for stationarity testing since the presence of seasonality implies non-stationarity, and our subsequent modeling efforts focused on capturing this seasonality.

Had the residuals from the `mstl()` decomposition been free of seasonality, our custom stationarity functions would have been instrumental in determining the appropriate differencing needed to achieve stationarity - a crucial step before fitting any ARIMA-type models.

We include the code for these custom functions in the appendix as a demonstration of our preparedness to conduct a comprehensive stationarity analysis and as a resource for potential future scenarios where such tests would be applicable.