R Appendix

```
knitr::opts_chunk$set(echo = FALSE)
knitr::opts_chunk$set(fig.width=6, fig.height=4)
library(forecast)
library(tseries)
library(tsoutliers)
# Read the CSV file while skipping the first row
data <- read.csv('/Users/tee._.thy/Desktop/ucd/davis y2/wq25/sta 137/sta 137 project/Popularity_trend.cs
# Summary Statistics
fivenum(data$philosophy...United.States.)
mean(data$philosophy...United.States.)
sd(data$philosophy...United.States.)
nrow(data)
# Rename columns properly
colnames(data) <- c("Month", "Value")

# Convert to time series object
ts_data <- ts(data$Value, start = c(2004, 1), frequency = 12)

# Print and plot
print(ts_data)
plot(ts_data, main = "Time Series Data", ylab = "Value", xlab = "Year")
plot(ts_data, type = "p",
     main = "Scatter Plot of Time Series Data",
     xlab = "Time", ylab = "Value",
     col = "blue", pch = 16)


# Find optimal lambda for Box-Cox
lambda <- BoxCox.lambda(ts_data)

# Apply Log Transformation
data$Transformed_Value <- log(data$Value)

# Convert transformed data to time series
ts_data_transformed <- ts(data$Transformed_Value, start = c(2004,1), frequency = 12)

# Plot the transformed time series
plot(ts_data_transformed, main = "Log Transformed Time Series",
     ylab = "Log(Value)", xlab = "Year", col="blue", lwd=2)

# QQ Plot for Original Data
qqnorm(ts_data, main = "QQ Plot of Original Data")
qqline(ts_data, col = "red", lwd = 2)

# QQ Plot for Transformed Data
qqnorm(ts_data_transformed, main = "QQ Plot of Log-Transformed Data")
qqline(ts_data_transformed, col = "red", lwd = 2)


# Before transformation
plot(ts_data, main = "Original Time Series", ylab = "Value", xlab = "Year", col="blue", lwd=2)
```

```r
# After transformation
plot(ts_data_transformed, main = "Log Transformed Time Series", ylab = "Transformed Value", xlab = "Yea


# Decompose the log-transformed time series (additive model)
decomposition <- decompose(ts_data_transformed, type = "additive")

# Extract and plot the seasonal component
s_hat <- decomposition$seasonal
plot(s_hat,
     main = "Seasonal Component (s_t)",
     ylab = "Seasonality",
     xlab = "Time", col = "blue", lwd = 2)

window_size <- 12

# Apply moving average smoothing
m_hat <- filter(ts_data_transformed, rep(1/window_size, window_size), sides = 2)

plot(ts_data_transformed, main = "Time Series with Smoothed Trend", ylab = "Value", xlab = "Time", col =
lines(m_hat, col = "blue", lwd = 2)  # Add trend line
legend("topleft", legend = c("Original Data", "Smoothed Trend"), col = c("gray", "blue"), lwd = c(1, 2))


# Deseasonalize and detrend the data by removing seasonality
deseasonalized_data <- ts_data_transformed - s_hat
detrended_data <- ts_data_transformed - m_hat
# Residuals:
residuals <- ts_data_transformed - m_hat - s_hat

# Plot the deseasonalized and detrended data and residuals
plot(deseasonalized_data,
     main = "Deseasonalized Data",
     ylab = "Deseasonalized Values",
     xlab = "Time", col = "red", lwd = 2)
plot(detrended_data,
     main = "Detrended Data",
     ylab = "Detrended Values",
     xlab = "Time", col = "red", lwd = 2)
plot(residuals,
     main = "Residuals",
     ylab = "Residual Values",
     xlab = "Time", col = "red", lwd = 2)

residuals_clean <- na.omit(residuals)
# Perform ADF test on the residual component
adf_test_res <- adf.test(residuals_clean, alternative = "stationary")
print(adf_test_res)


# Check normality
hist(residuals_clean, breaks = 20, main = "Histogram of Residuals", col = "lightblue")
qqnorm(residuals_clean)
```

```r
qqline(residuals_clean, col = "red", lwd = 2)

# Check Homoscedasticity (Constant Variance)
plot(residuals_clean, main = "Residuals Over Time", ylab = "Residuals", xlab = "Time")
abline(h = 0, col = "red", lwd = 2)

# Plot ACF and PACF of residuals
residuals_ts <- ts(residuals_clean)
acf(residuals_ts, main = "ACF of Residuals")
pacf(residuals_ts, main = "PACF of Residuals")

# Model selection
arima(residuals_ts)
arima(residuals_ts, order = (c(1,0,0)))
arima(residuals_ts, order = (c(1,0,1)))
arima(residuals_ts, order = (c(2,0,0)))
arima(residuals_ts, order = (c(2,0,1)))
arima(residuals_ts, order = (c(2,0,2)))
arima(residuals_ts, order = (c(3,0,1)))
arima(residuals_ts, order = (c(2,0,3)))



# Fit the ARIMA(2,0,1) model
arma_model <- arima(residuals_ts, order = c(2, 0, 1))

# Extract residuals from the fitted model
arma_residuals <- residuals(arma_model)

# Plot ACF of residuals
acf(arma_residuals, main = "ACF of ARIMA(2,0,1) Residuals")

# Plot PACF of residuals
pacf(arma_residuals, main = "PACF of ARIMA(2,0,1) Residuals")

# Plot residuals of arma model
plot(arma_residuals, type = "l", main = "Residuals Over Time", ylab = "Residuals", xlab = "Time")
abline(h = 0, col = "red", lwd = 2)


forecast_horizon <- 12
# Predict the trend component
time_index <- 1:length(m_hat)
trend_model <- lm(m_hat ~ time_index, na.action = na.exclude)
future_time <- (length(m_hat) + 1):(length(m_hat) + forecast_horizon)
trend_forecast <- predict(trend_model, newdata = data.frame(time_index = future_time))
trend_forecast <- ts(trend_forecast, start = c(2016, 1), frequency = 12)

# Predict the seasonal component
seasonal_forecast <- tail(s_hat, 12)
seasonal_forecast <- ts(seasonal_forecast, start = c(2016, 1), frequency = 12)

# Predict residuals using MLE
```

```r
arma_model_mle <- arima(residuals_ts, order = c(2, 0, 1), method = "ML")
sum_model_residual <- summary(arma_model_mle)
arma_forecast <- predict(arma_model_mle, n.ahead = forecast_horizon)

residuals_forecast <- arma_forecast$pred
residuals_forecast <- ts(residuals_forecast, start = c(2016, 1), frequency = 12)

# Combine all component and predict 2016
forecast_2016 <- trend_forecast + seasonal_forecast + residuals_forecast
print(forecast_2016)
forecast_original_2016 <- exp(forecast_2016)

# Extract phi_hat and se_phi from the model
phi_hat <- arma_model_mle$coef[1:2]
se_phi <- sqrt(diag(arma_model_mle$var.coef))[1:2]
theta_hat <- arma_model_mle$coef[3]
se_theta <- sqrt(diag(arma_model_mle$var.coef))[3]
# Compute 95% confidence intervals
conf_int_phi1 <- phi_hat[1] + c(-1.96, 1.96) * se_phi[1]
conf_int_phi2 <- phi_hat[2] + c(-1.96, 1.96) * se_phi[2]
conf_int_theta <- theta_hat + c(-1.96, 1.96) * se_theta


# Predict 2025
forecast_horizon_2025 <- 12
future_time_2025 <- (length(m_hat) + 13):(length(m_hat) + 24)
trend_forecast_2025 <- predict(trend_model, newdata = data.frame(time_index = future_time_2025))
seasonal_forecast_2025 <- seasonal_forecast
arma_forecast_2025 <- predict(arma_model_mle, n.ahead = forecast_horizon_2025)
residuals_forecast_2025 <- ts(arma_forecast_2025$pred, start = c(2025, 1), frequency = 12)
trend_forecast_2025 <- ts(trend_forecast_2025, start = c(2025, 1), frequency = 12)
seasonal_forecast_2025 <- ts(seasonal_forecast_2025, start = c(2025, 1), frequency = 12)

# Final Forecast for 2025
final_forecast_2025 <- trend_forecast_2025 + seasonal_forecast_2025 + residuals_forecast_2025
print(final_forecast_2025)
final_forecast_2025 <- exp(final_forecast_2025)
```