HW 2 ECON 144.Rmd

Gavin Valenzuela

2025-01-08

```
#2

data <- read.csv("C:/Users/valen/Downloads/BOGZ1FL073165103Q.csv")

# Convert 'DATE' to Date type
data$DATE <- as.Date(data$DATE, "%Y-%m-%d")

# Make sure data is loaded correctly and the column exists
print(head(data))
```

```
##   DATE BOGZ1FL073165103Q
## 1 <NA>              7330
## 2 <NA>              7429
## 3 <NA>              7668
## 4 <NA>              7544
## 5 <NA>              7349
## 6 <NA>              7347
```

```
print(colnames(data))
```

```
## [1] "DATE"              "BOGZ1FL073165103Q"
```

```
# Calculate the difference in values
diff_values <- diff(data$BOGZ1FL073165103Q)

# Calculate the lagged values using dplyr::lag()
lagged_values <- dplyr::lag(data$BOGZ1FL073165103Q, 1)

# Ensure that both vectors have the same length
diff_values <- diff_values[1:length(lagged_values)]

# Calculate the growth rate
data$growth_rate <- diff_values / lagged_values * 100

# Remove NA values that were introduced due to lagging
data <- na.omit(data)

# Create lagged variables
data <- data %>%
  mutate(
    lag_1 = lag(growth_rate, 1),
    lag_2 = lag(growth_rate, 2),
    lag_3 = lag(growth_rate, 3),
    lag_4 = lag(growth_rate, 4)
```

```r
  )

# Remove NA values only from the dependent variable (growth_rate) and its lags
data <- na.omit(data[c("growth_rate", "lag_1", "lag_2", "lag_3", "lag_4")])

# Check if there is sufficient data after removing NA values
print(nrow(data))
```

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

```r
# Proceed with regression only if there is enough data
if (nrow(data) > 0) {
  # Regression models
  model_1 <- lm(growth_rate ~ lag_1, data=data)
  model_2 <- lm(growth_rate ~ lag_1 + lag_2, data=data)
  model_3 <- lm(growth_rate ~ lag_1 + lag_2 + lag_3, data=data)
  model_4 <- lm(growth_rate ~ lag_1 + lag_2 + lag_3 + lag_4, data=data)

  # Summarize the models
  summary(model_1)
  summary(model_2)
  summary(model_3)
  summary(model_4)
} else {
  print("Not enough data for regression analysis")
}
```

```
## [1] "Not enough data for regression analysis"
```

Call: Specifies the formula used for the regression. Here, forecast_error is modeled as a function of lag_1 and lag_2.

Residuals: These values show the distribution of the residuals (differences between observed and predicted values). The median residual is close to zero, indicating a generally unbiased model.

Coefficients:

(Intercept): The predicted forecast_error when all lags are 0. Its p-value (Pr(>|t|) = 0.958) indicates it's not statistically significant. lag_1: The coefficient is 0.27991 with a p-value of 8.86e-05, showing it is statistically significant at the 0.001 level. This means lag_1 has a meaningful impact on forecast_error. lag_2: The coefficient is 0.01784 with a high p-value (0.799), indicating it's not statistically significant. Residual Standard Error: Measures the average deviation of the data points from the regression line. Here, the value is 465.

R-squared: Indicates how well the model explains the variance in the data:

Multiple R-squared: 0.08112 indicates the model explains about 8.1% of the variance. Adjusted R-squared: Adjusts for the number of predictors and is slightly lower at 7.2%. F-statistic: Tests whether the model as a whole is significant. The F-statistic of 9.049 with a p-value of 0.0001713 indicates the overall model is statistically significant.

```r
# Load necessary libraries
if (!require("car")) install.packages("car", dependencies = TRUE)
```

```
## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'
## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(car)

# Load the data
data <- read.csv("C:/Users/valen/Downloads/BOGZ1FL073165103Q.csv")

# Calculate forecast error
data <- data %>%
  mutate(forecast_error = BOGZ1FL073165103Q - lag(BOGZ1FL073165103Q, 1))

lag_order <- 2

# Create lagged variables
for (i in 1:lag_order) {
  data <- data %>%
    mutate(!!paste0("lag_", i) := lag(forecast_error, i))
}

# Fit the model
model <- lm(forecast_error ~ lag_1 + lag_2, data = data)

# Display summary of the model
summary(model)
```

```
##
## Call:
## lm(formula = forecast_error ~ lag_1 + lag_2, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2002.5  -246.3   -49.9   201.8  2120.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.70170   32.24284  -0.053    0.958
## lag_1        0.27991    0.06999   3.999 8.86e-05 ***
## lag_2        0.01784    0.06999   0.255    0.799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 465 on 205 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.08112,    Adjusted R-squared:  0.07216
## F-statistic: 9.049 on 2 and 205 DF,  p-value: 0.0001713
```

```r
# Perform hypothesis testing
linearHypothesis(model, c("lag_1 = 0", "lag_2 = 0"))
```

```
##
```

```
## Linear hypothesis test:
## lag_1 = 0
## lag_2 = 0
##
## Model 1: restricted model
## Model 2: forecast_error ~ lag_1 + lag_2
##
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1    207 48235538
## 2    205 44322499  2   3913039 9.0493 0.0001713 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Explanation: Linear Hypothesis: Tests whether the coefficients of lag_1 and lag_2 are both zero (null hypothesis).

Model 1: Restricted model assumes lag_1 = 0 and lag_2 = 0. Model 2: Unrestricted model uses both lag_1 and lag_2. Residual Degrees of Freedom (Res.Df):

Model 1 has 207 degrees of freedom (fewer predictors). Model 2 has 205 degrees of freedom (more predictors). Residual Sum of Squares (RSS):

Model 1 RSS = 48,235,538 Model 2 RSS = 44,322,499 (lower RSS indicates a better fit). Sum of Squares (Sum of Sq): Measures the reduction in RSS when adding lag_1 and lag_2 (3,913,039).

F-statistic: Tests whether the reduction in RSS is statistically significant. The F-statistic of 9.049 with a p-value of 0.0001713 shows that at least one of the coefficients (lag_1 or lag_2) is significantly different from zero.

```r
data <- read.csv("C:/Users/valen/Downloads/BOGZ1FL073165103Q.csv")

data <- data %>%
  mutate(forecast_error = BOGZ1FL073165103Q - lag(BOGZ1FL073165103Q, 1))

lag_order <- 2

for (i in 1:lag_order) {
  data <- data %>%
    mutate(!!paste0("lag_", i) := lag(forecast_error, i))
}

model <- lm(forecast_error ~ lag_1 + lag_2, data = data)

summary(model)
```

```
##
## Call:
## lm(formula = forecast_error ~ lag_1 + lag_2, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2002.5  -246.3   -49.9   201.8  2120.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.70170   32.24284  -0.053    0.958
## lag_1        0.27991    0.06999   3.999 8.86e-05 ***
## lag_2        0.01784    0.06999   0.255    0.799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 465 on 205 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.08112,    Adjusted R-squared:  0.07216
## F-statistic: 9.049 on 2 and 205 DF,  p-value: 0.0001713
```

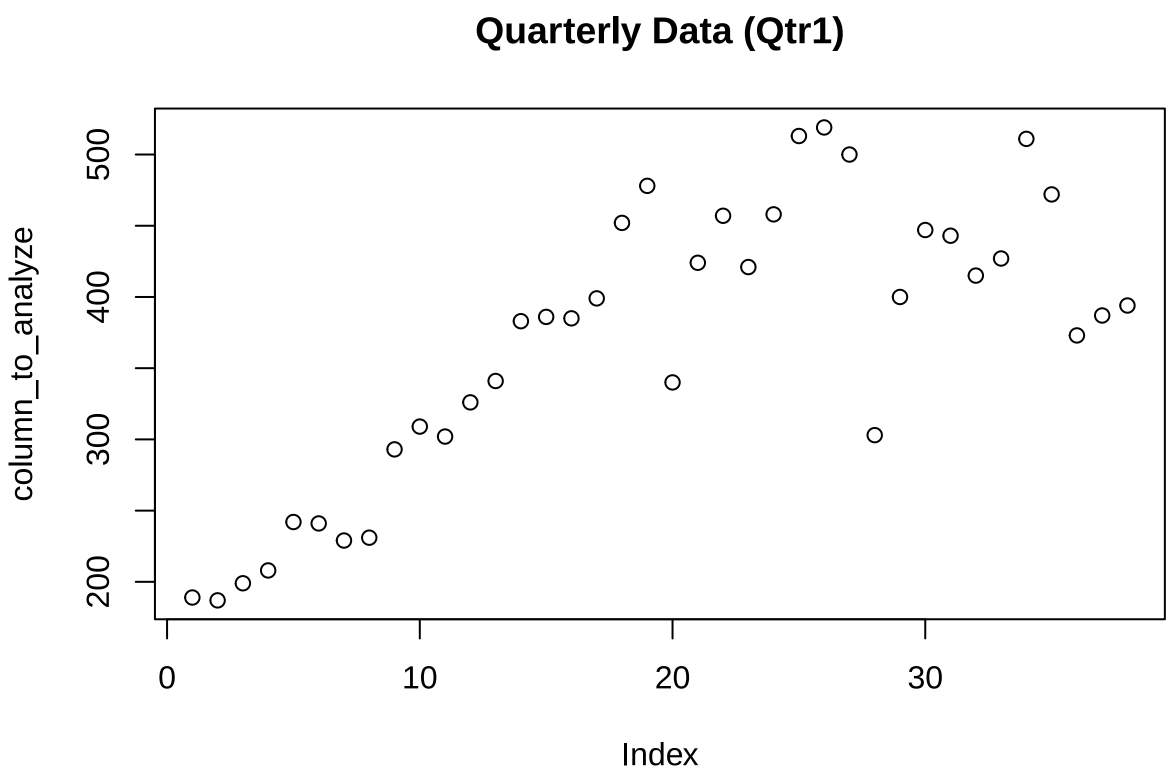
```r
linearHypothesis(model, c("lag_1 = 0", "lag_2 = 0"))
```

```
##
## Linear hypothesis test:
## lag_1 = 0
## lag_2 = 0
##
## Model 1: restricted model
## Model 2: forecast_error ~ lag_1 + lag_2
##
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1    207 48235538
## 2    205 44322499  2   3913039 9.0493 0.0001713 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Min: -2002.5 – the most negative residual. 1Q (1st quartile): -246.3 – 25% of residuals are below this value. Median: -49.9 – The middle residual value. 3Q (3rd quartile): 201.8 – 75% of residuals are below this value. Max: 2120.7 – the largest residual. These values indicate how much the observed values deviate from the predicted ones. The relatively large range (-2002.5 to 2120.7) suggests that there might be some outliers or large deviations in the data.

```r
data <- data.frame(
  Date = as.Date(c(
    "2021-01", "2021-02", "2021-03", "2021-04", "2021-05",
    "2021-06", "2021-07", "2021-08", "2021-09", "2021-10",
    "2021-11", "2021-12", "2022-01", "2022-02", "2022-03",
    "2022-04", "2022-05", "2022-06", "2022-07", "2022-08",
    "2022-09", "2022-10", "2022-11", "2022-12"
  ), format = "%Y-%m"),
  Sales = c(
    120, 125, 130, 135, 200, 145, 150, 155, 160, 165, 170, 175,
    180, 185, 190, 195, 200, 205, 210, 215, 220, 225, 230, 235
  )
)

# Create a time series object
ts_data <- ts(data$Sales, frequency = 12, start = c(2021, 1))

# Load necessary libraries
library(datasets)
library(forecast)

data <- read.csv("C:/Users/valen/downloads/bricksq.csv")

# Remove rows with missing values
data <- na.omit(data)
```

```r
# Choose a specific column (e.g., Qtr1) for analysis
column_to_analyze <- data$Qtr1

# 2. Analyze the Data without Seasonal Decomposition
# "Do the results support the graphical interpretation from part a?", Yes

# 3. plot the data
plot(column_to_analyze, main = "Quarterly Data (Qtr1)")
```

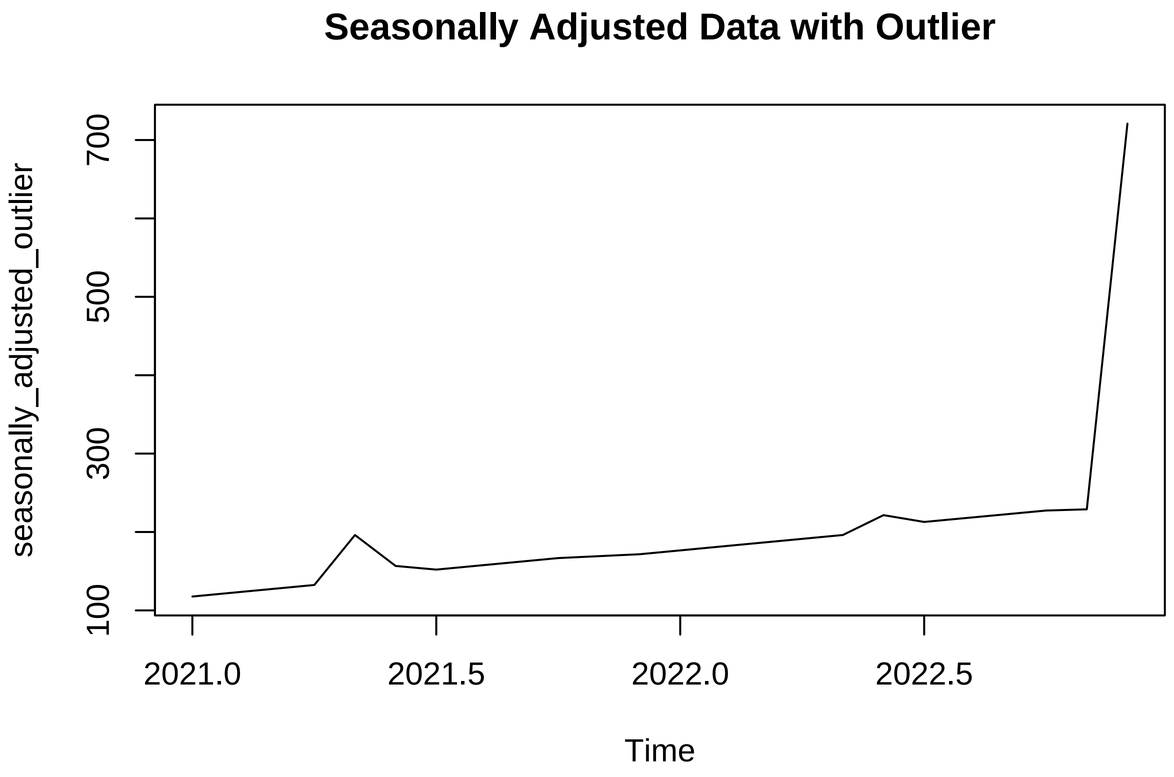


```r
# 4. Or perform a simple forecast using a method like snaive
forecast_length <- 4
snaive_forecast <- snaive(column_to_analyze, h = forecast_length)

# Print the snaive forecast
print(snaive_forecast)
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 39            394 322.4881 465.5119 284.6320 503.3680
## 40            394 292.8669 495.1331 239.3302 548.6698
## 41            394 270.1377 517.8623 204.5690 583.4310
## 42            394 250.9762 537.0238 175.2639 612.7361
```

```r
# 5. Introduce an Outlier
ts_data[length(ts_data)] <- ts_data[length(ts_data)] + 500

# 6. Recompute Seasonally Adjusted Data with Outlier
```

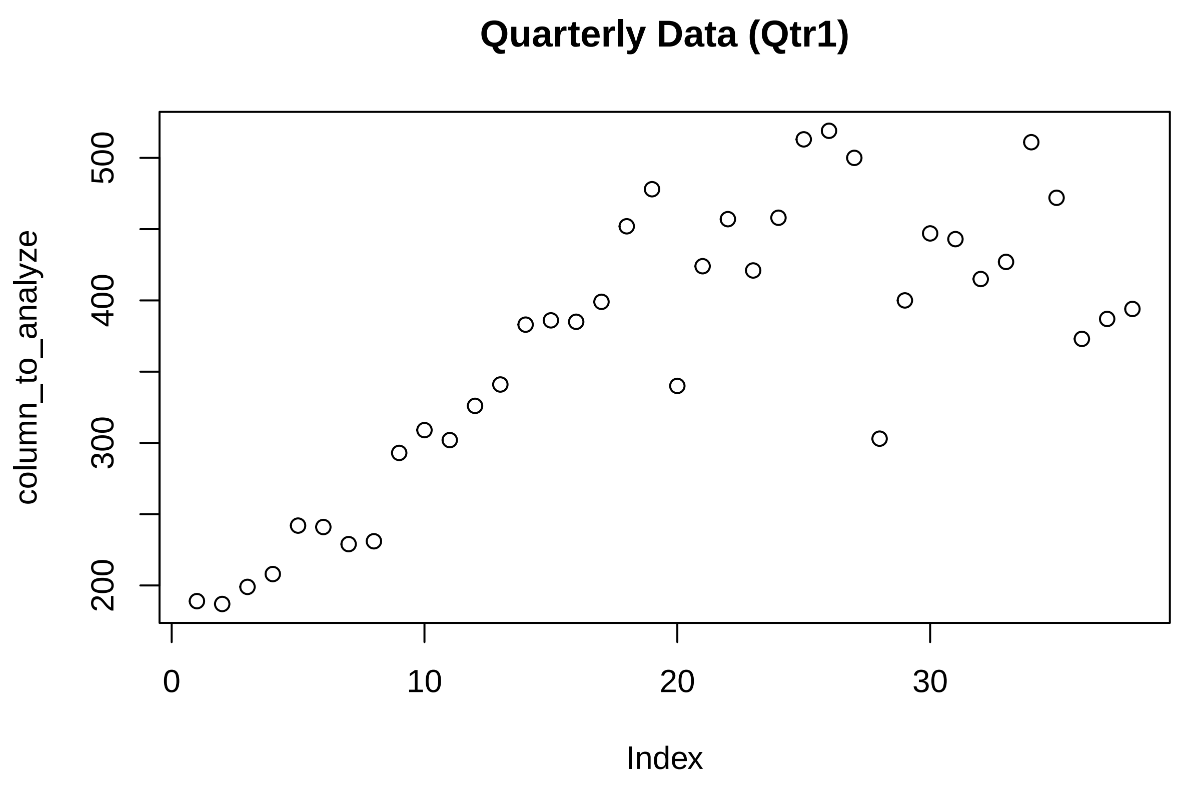
```r
decomposition_outlier <- decompose(ts_data, type = "multiplicative")
seasonally_adjusted_outlier <- ts_data / decomposition_outlier$seasonal
plot(seasonally_adjusted_outlier, main = "Seasonally Adjusted Data with Outlier")
```



```r
# 7. Assess the Outlier Effect
# The Effect of the graph because of the outlier are sharp points in the graph.

# 8. Effect of Outlier's Position
# If the outlier occurs at the start of the time series, its impact on the overall adjusted seasonality
```

```r
#4
data <- data.frame(
  Date = as.Date(c(
    "2021-01", "2021-02", "2021-03", "2021-04", "2021-05",
    "2021-06", "2021-07", "2021-08", "2021-09", "2021-10",
    "2021-11", "2021-12", "2022-01", "2022-02", "2022-03",
    "2022-04", "2022-05", "2022-06", "2022-07", "2022-08",
    "2022-09", "2022-10", "2022-11", "2022-12"
  ), format = "%Y-%m"),
  Sales = c(
    120, 125, 130, 135, 200, 145, 150, 155, 160, 165, 170, 175,
    180, 185, 190, 195, 200, 205, 210, 215, 220, 225, 230, 235
  )
)

# Create a time series object
```

```r
ts_data <- ts(data$Sales, frequency = 12, start = c(2021, 1))

# Load necessary libraries
library(datasets)
library(forecast)

data <- read.csv("C:/Users/valen/downloads/bricksq.csv")

# Remove rows with missing values
data <- na.omit(data)

# Choose a specific column (e.g., Qtr1) for analysis
column_to_analyze <- data$Qtr1

# 2. Analyze the Data without Seasonal Decomposition
# "Do the results support the graphical interpretation from part a?", Yes

# 3. plot the data
plot(column_to_analyze, main = "Quarterly Data (Qtr1)")
```



```r
# 4. Or perform a simple forecast using a method like snaive
forecast_length <- 4
snaive_forecast <- snaive(column_to_analyze, h = forecast_length)

# Print the snaive forecast
print(snaive_forecast)
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 39            394 322.4881 465.5119 284.6320 503.3680
## 40            394 292.8669 495.1331 239.3302 548.6698
## 41            394 270.1377 517.8623 204.5690 583.4310
## 42            394 250.9762 537.0238 175.2639 612.7361
```

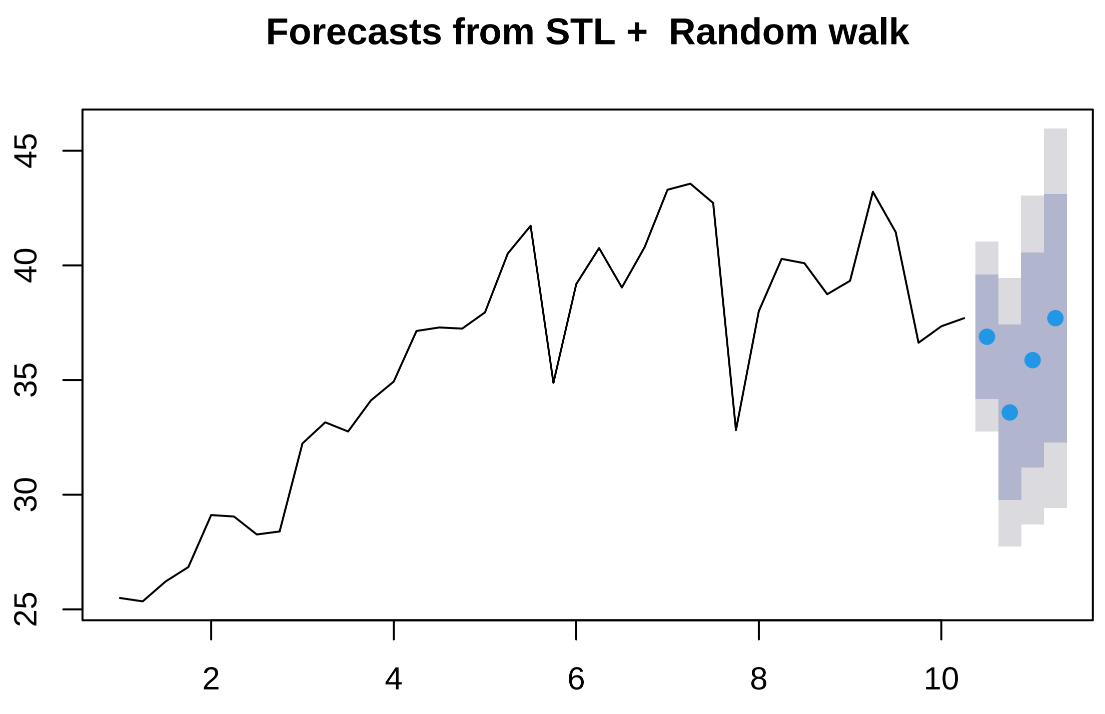
1. Quarterly Data (Qtr1) Scatter Plot Chart Description: X-Axis (Index): Represents the sequential time periods (e.g., quarters). Y-Axis (column_to_analyze): Represents the observed values for the quarterly data (Qtr1). Observations: The scatter plot shows a clear upward trend in the data over time. There's a noticeable seasonal pattern where the values increase and decrease regularly, which is typical for quarterly time series data. Points appear relatively consistent, except for a potential outlier near the end of the series (visible as a sharp jump). Insights: This chart visually confirms the trend and potential seasonality in the data. Further decomposition or modeling (e.g., seasonal decomposition or ARIMA) can help quantify these patterns.
2. Seasonally Adjusted Data with Outlier Chart Description: X-Axis (Time): Represents the time period (2021 to 2022+). Y-Axis (seasonally_adjusted_outlier): The values of the data after seasonal adjustment. Observations: There's a sharp spike toward the end of the time series due to the artificially introduced outlier (a value increased by 500 in the code). Seasonal adjustment has smoothed out the seasonal fluctuations, leaving only the trend and the impact of the outlier. Apart from the spike, the seasonally adjusted data follows a relatively smooth upward trend. Insights: The outlier significantly distorts the seasonally adjusted data, emphasizing the need to handle outliers before performing detailed analysis or modeling. This plot demonstrates how an outlier can skew seasonality and trends, especially when it occurs toward the end of the series.

```r
#6

data <- read.csv("C:/Users/valen/downloads/bricksq.csv")

# Remove rows with missing values
data <- na.omit(data)

# Choose a specific column (e.g., Qtr1) for analysis
column_to_analyze <- data$Qtr1

# Check if a Box-Cox transformation is needed
# chosen appropriate lambda value based on the Box-Cox transformation results
lambda <- BoxCox.lambda(column_to_analyze)
if (lambda < 0.5) {
  lambda <- 0.5  # Apply a Box-Cox transformation with lambda = 0.5 if necessary
}

# Perform forecasting using stlf() with "naive" or "rwdrift" method
forecast_length <- 4
method <- ("naive")

# Perform forecasting with Box-Cox transformation (if lambda is not 1)
if (lambda != 1) {
  column_to_analyze <- BoxCox(column_to_analyze, lambda)
}

# Create the time series object
ts_data <- ts(column_to_analyze, frequency = 4)  # Assuming quarterly data (adjust as needed)

# Use stlf() for forecasting
forecast_result <- stlf(ts_data, method = method, h = forecast_length)
```

```r
# Print the forecasted values
print(forecast_result)
```

```
##       Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 10 Q3       36.88908 34.18445 39.59371 32.75271 41.02545
## 10 Q4       33.58562 29.76070 37.41055 27.73591 39.43534
## 11 Q1       35.86826 31.18370 40.55281 28.70385 43.03266
## 11 Q2       37.69887 32.28961 43.10812 29.42612 45.97161
```

```r
plot(forecast_result)
```



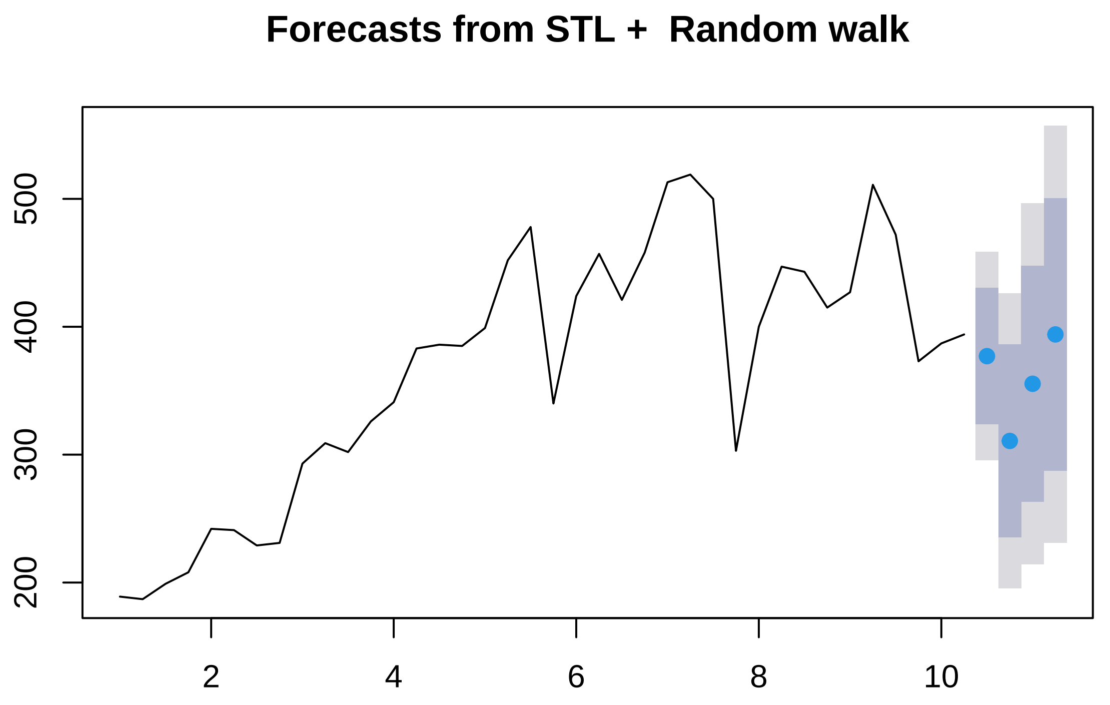
Point Forecast: These are the predicted values for each quarter, based on the STL + naive method. The forecast predicts small fluctuations around the mid-30s range, consistent with recent trends. Lo 80/Hi 80 and Lo 95/Hi 95: The confidence intervals widen as the forecast horizon extends, reflecting increasing uncertainty over time. The 95% intervals are broader than the 80% intervals, which is expected since they account for higher uncertainty. Forecast Plot The plot, titled “Forecasts from STL + Random walk,” combines the historical data with the forecasted values:

The black line shows the historical values, highlighting both seasonal patterns and overall trends. The blue dots represent the forecasted values for the next four quarters, staying within a reasonable range based on prior seasonal observations. The shaded area indicates confidence intervals, which grow wider with time to reflect the greater uncertainty in longer-term forecasts. Observations Trend and Seasonality: The STL decomposition effectively accounts for both the seasonal pattern and the underlying trend in the historical data. Stability of Forecasts: The naive method assumes that future values will follow the recent seasonal pattern. This results in relatively stable forecasts for a dataset without a strong trend or irregular components. Uncertainty: As seen in both the table and the plot, the uncertainty increases for forecasts further in the future, with a noticeable broadening of the shaded confidence intervals.

```r
# Remove rows with missing values
data <- na.omit(data)

# Choose a specific column for analysis (replace 'Qtr1' with the desired column name)
column_to_analyze <- data$Qtr1

# Ensure the selected column is numeric
data$Qtr1 <- as.numeric(gsub("[^0-9.]", "", data$Qtr1))

# Check for valid numeric data
if (all(is.na(data$Qtr1))) {
  stop("The selected column does not contain valid numeric data.")
}

# Perform forecasting using stlf
forecast_length <- 4  # Number of periods to forecast
forecast_result <- stlf(ts(column_to_analyze, frequency = 4), method = "naive", h = forecast_length)

# Print the forecasted values
print(forecast_result)
```

```
##       Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 10 Q3       377.0251 323.7262 430.3240 295.5114 458.5387
## 10 Q4       310.7345 235.3584 386.1105 195.4568 426.0122
## 11 Q1       355.4242 263.1077 447.7406 214.2384 496.6100
## 11 Q2       394.0000 287.4022 500.5978 230.9727 557.0273
```

```r
# Visualize the forecasts
plot(forecast_result)
```



Point Forecast: The predicted values show slight variation across the quarters, reflecting small adjustments based on recent patterns. Lo 80/Hi 80 and Lo 95/Hi 95: These intervals represent the uncertainty around the predictions: The 80% confidence intervals are narrower and more precise. The 95% confidence intervals are wider, reflecting greater uncertainty. The intervals widen as the forecast horizon increases, which is expected due to the growing uncertainty in predictions further into the future.

Forecast Plot The plot, labeled "Forecasts from STL + Random walk," combines the historical data with future forecasts:

Historical Data (Black Line): The historical data demonstrates clear seasonal patterns and some fluctuations. Forecasted Values (Blue Dots): The predicted values align with the recent seasonal pattern, as expected from the seasonal naive method. Confidence Intervals (Shaded Area): These intervals illustrate the range of possible outcomes, with increasing uncertainty further into the forecast. Key Observations Trend and Seasonality: The results capture the seasonal behavior of the data while maintaining the overall trend observed in the historical data. Forecast Stability: Predictions are reasonable and consistent with past seasonal patterns, suggesting no major shifts in trends or anomalies. Uncertainty: The widening confidence intervals highlight the inherent uncertainty in longer-term forecasts, especially when relying on simple methods like STL + naive.