Econ 104 Project 2

Anshika Khandelwal, Ishika Agrawal, Meghna Nair, Rajasvi Singh

2025-05-12

Contents

Part 1 - Time Series and Autocorrelation

Question 1: Exploratory Data Analysis

a) Introduction:

Our goal for this project is to explore the relationship between expected inflation and actual inflation in the UK manufacturing sector during the period of 1972 to 1985. Specifically, we seek to answer the question: To what extent can expected inflation serve as a reliable predictor for actual inflation over time in the UK manufacturing sector? This question is crucial to our ability to understand whether inflation expectations were rational and aligned with actual outcomes, which has implications for monetary policy and strategic planning by firms.

b) Citations & Summary

The dataset used in this analysis is UKInflation, a quarterly time series spanning from Q1 of 1972 to Q2 of 1985, containing 54 observations. It consists of two variables: "actual," which is the actual observed rate of inflation in the UK manufacturing sector, and "expected," which is the expected rate of inflation in the UK manufacturing sector. This dataset is publicly available through the AER package in R and is structured as a multiple time series object, making it suitable for time series analysis including autocorrelation, stationarity testing, and AR/ARDL modeling. The objective of analyzing this dataset is to understand the time series behavior of inflation and determine whether past expectations of inflation provide useful information for modeling or forecasting actual inflation outcomes.

Citations:

Greene, W.H. (2003). Econometric Analysis (5th ed.). Upper Saddle River, NJ: Prentice Hall.

Pesaran, M.H., & Hall, A.D. (1988). Tests of Non-nested Linear Regression Models Subject to Linear Restrictions. Economics Letters, 27(4), 341–348.

c) Checking for NAs and Missing Observations

```
# View the structure and first few rows

str(UKInflation)
```

```
##  Time-Series [1:54, 1:2] from 1972 to 1985: 0.99 1.62 1.87 2.89 2.23 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "actual" "expected"
```

```
head(UKInflation)
```

```
##      actual expected
## [1,]   0.99     0.79
## [2,]   1.62     1.94
## [3,]   1.87     2.97
## [4,]   2.89     3.37
## [5,]   2.23     3.65
## [6,]   1.69     1.62
```

```
# Check for any missing values

anyNA(UKInflation)  # Returns TRUE if any NAs exist
```

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

Through the anyNA() function, we checked our data set for any NAs or missing values. The test returned an output of FALSE, indicating that our data set is free of NAs and missing observations. We can now proceed with the descriptive analysis of our variables.

d) Descriptive Analysis of Variables

```
# install packages & libraries

install.packages("ggplot2", repos = "https://cloud.r-project.org/")
```

```
##
## The downloaded binary packages are in
##  /var/folders/24/d5p4stj55zx8y3rtzgx0k84m0000gn/T//RtmpRnJozf/downloaded_packages
```

```
install.packages("corrplot", repos = "https://cloud.r-project.org/")
```

```
##
## The downloaded binary packages are in
##  /var/folders/24/d5p4stj55zx8y3rtzgx0k84m0000gn/T//RtmpRnJozf/downloaded_packages
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

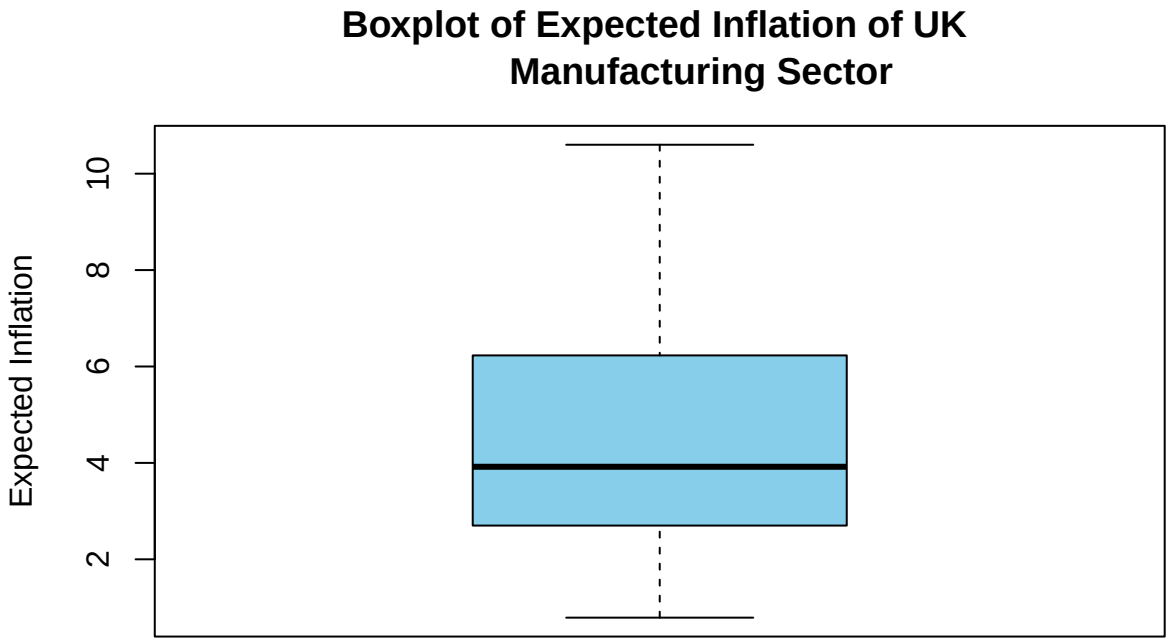
```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3
```

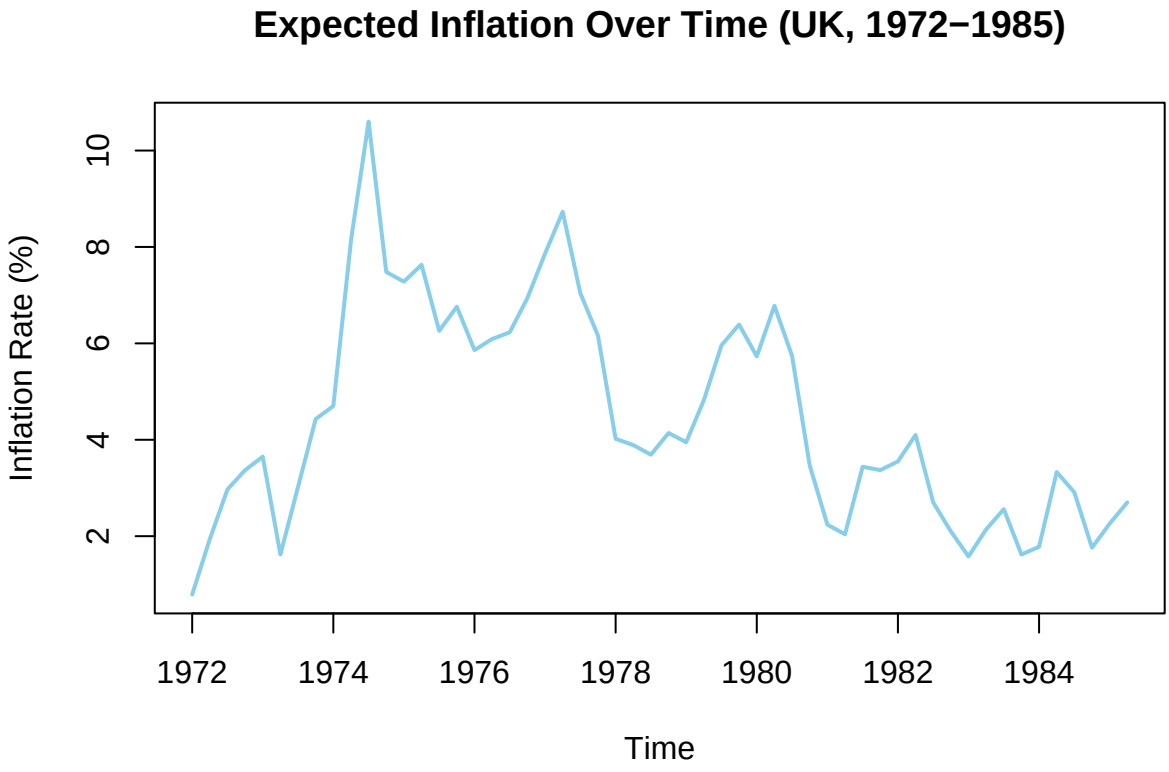
```
## corrplot 0.95 loaded
```

```
# Plots for Variable "Expected"
```

```r
boxplot(UKInflation[ , "expected"], main = "Boxplot of Expected Inflation of UK
        Manufacturing Sector", col = "skyblue", ylab = "Expected Inflation")
```

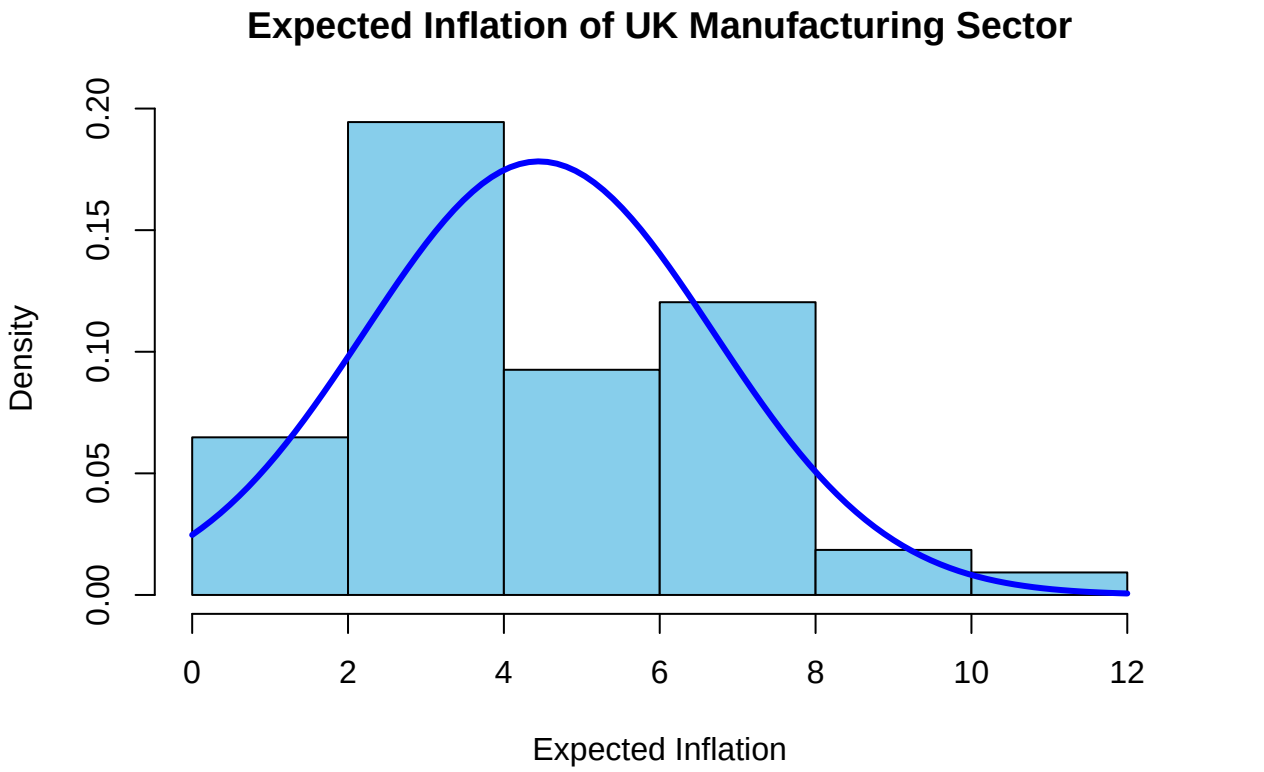



```r
plot(UKInflation[, "expected"],
     main = "Expected Inflation Over Time (UK, 1972-1985)",
     ylab = "Inflation Rate (%)",
     xlab = "Time",
     col = "skyblue",
     lwd = 2)
```



```r
hist(UKInflation[ , "expected"], main = "Expected Inflation of UK Manufacturing Sector",
     xlab = "Expected Inflation", col = "skyblue", probability = TRUE)

curve(dnorm(x, mean = mean(UKInflation[ , "expected"], na.rm = TRUE),
            sd = sd(UKInflation[ , "expected"], na.rm = TRUE)),  col = "blue",
            lwd = 3, add = TRUE)
```




```r
summary(UKInflation[ , "expected"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.790   2.700   3.920   4.450   6.213  10.600
```

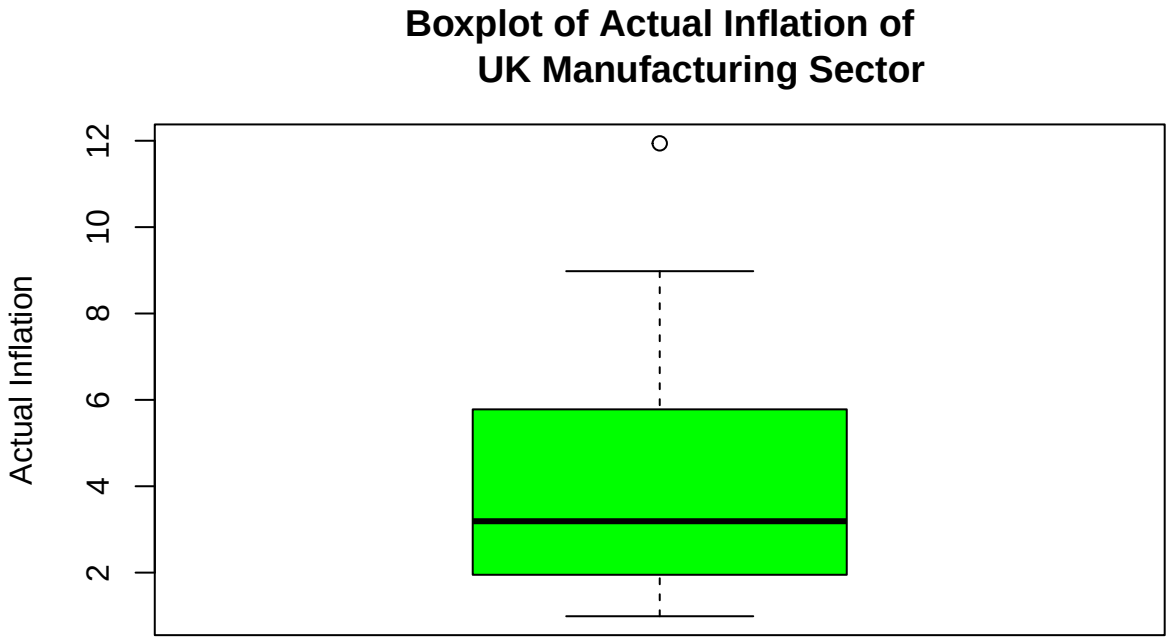
This boxplot shows that expected inflation is moderately right-skewed, with the median around 4%. There is a visible difference in spread between the lower and upper quartiles with the upper quartile spanning a broader range of values than the lower quartile.

This time-series plot shows expected inflation rates in the UK from 1972 to 1985. The points reveal a sharp increase in inflation in the mid 1970s, peaking around 1975 above 10%, followed by a gradual decline with occasional fluctuations over the next decade. By the mid-1980s, expected inflation settled between 2–4%, indicating improved economic stability.

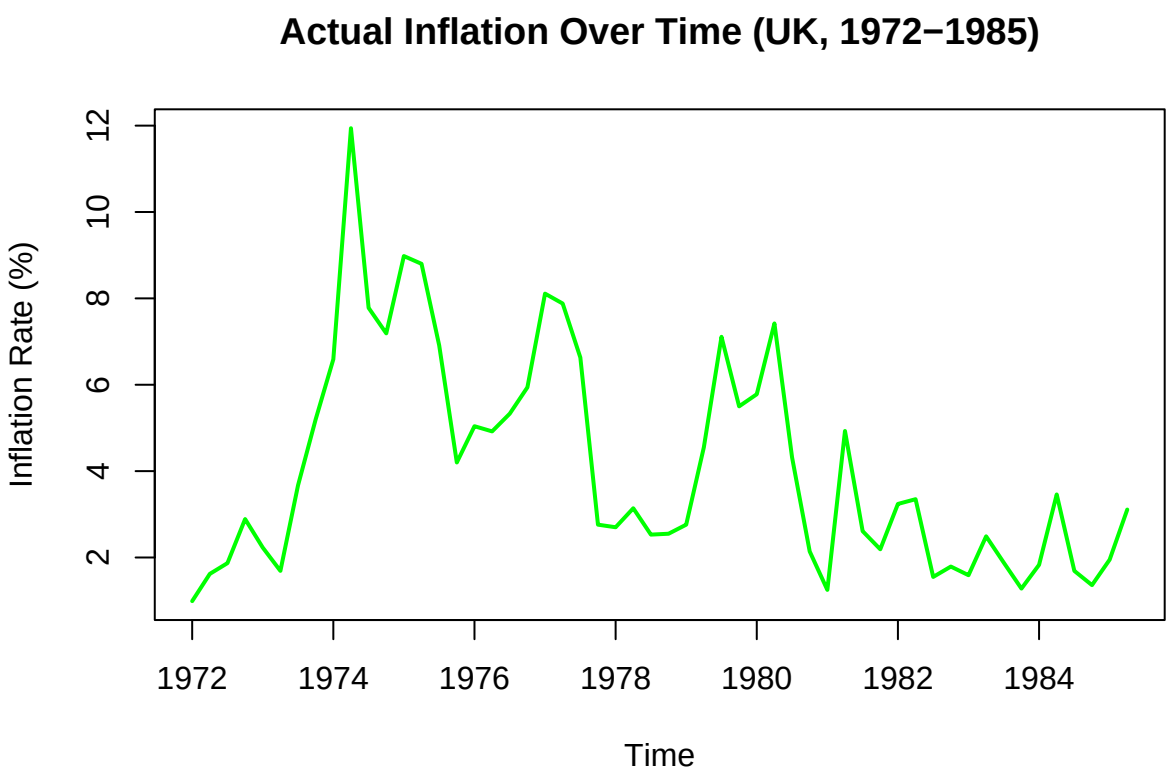
The histogram reveals that expected inflation is concentrated between 2% and 6%, tapering off as values increase. The right-skewed density curve confirms the presence of higher, less frequent inflation expectations, showing a non-normal distribution

```r
# Plots for Variable "Actual"

boxplot(UKInflation[ , "actual"], main = "Boxplot of Actual Inflation of
        UK Manufacturing Sector", col = "green", ylab = "Actual Inflation")
```

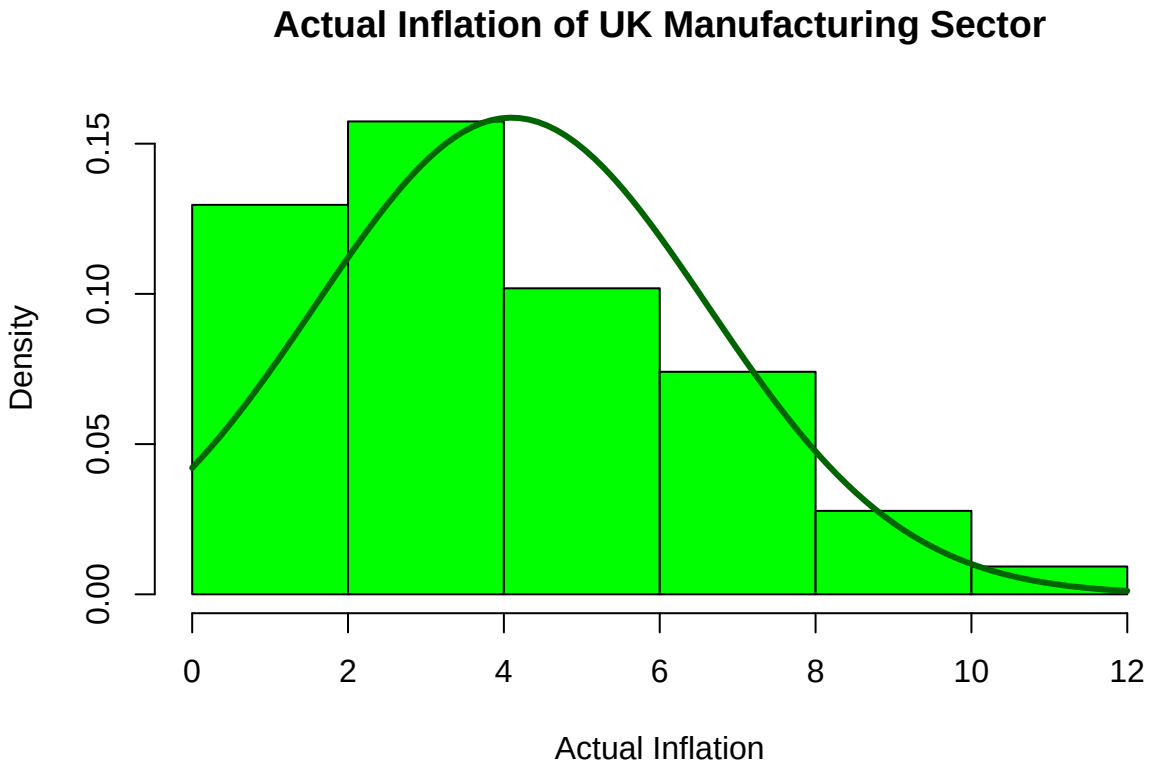



```
plot(UKInflation[, "actual"],
     main = "Actual Inflation Over Time (UK, 1972-1985)",
     ylab = "Inflation Rate (%)",
     xlab = "Time",
     col = "green",
     lwd = 2)
```




```
hist(UKInflation[ , "actual"], main = "Actual Inflation of UK Manufacturing Sector",
     xlab = "Actual Inflation", col = "green", probability = TRUE)
```

```r
curve(dnorm(x, mean = mean(UKInflation[ , "actual"], na.rm = TRUE),
            sd = sd(UKInflation[ , "actual"], na.rm = TRUE)),  col = "darkgreen",
            lwd = 3, add = TRUE)
```




```r
summary(UKInflation[ , "actual"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.990   1.998   3.190   4.096   5.710  11.940
```

The boxplot shows that most actual inflation rates in the UK manufacturing sector between 1972–1985 ranged from around 2% to 6%, with a few high outliers above 10%. The median inflation was approximately 3.5%.

The time-series plot of actual inflation over time reveals sharp spikes around 1974-1975 and 1980, followed by a general decline and stabilization by the mid-1980s. Inflation was highly volatile in the earlier years of the period.

The histogram shows that most actual inflation rates in the UK manufacturing sector are clustered between 2% and 6%, with a peak around 3–4%. The distribution is right-skewed, indicating a few instances of very high inflation above 10%.

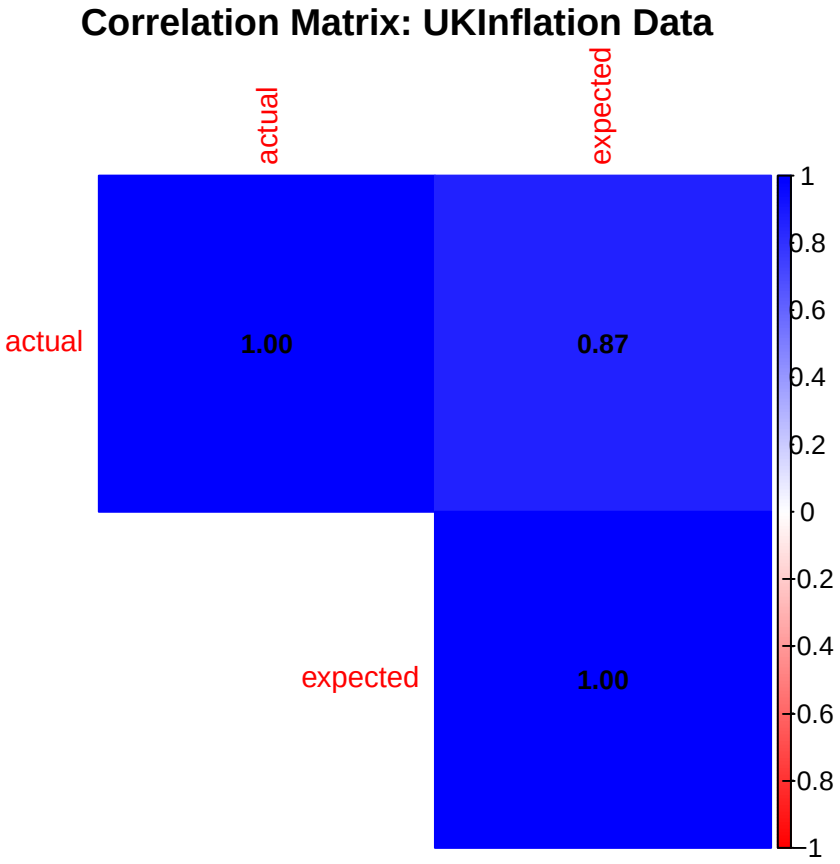
```r
# Correlation Matrix

vars <- UKInflation[, c("actual", "expected")]

cor_matrix <- cor(vars, use = "complete.obs")

corrplot(cor_matrix,
  method = "color", # colorful heatmap
  type = "upper", # only show upper triangle
  addCoef.col = "black", # add correlation coefficients
  tl.cex = 0.9, # text size for variable labels
  number.cex = 0.8, # text size for coefficients
```

```
  col = colorRampPalette(c("red", "white", "blue"))(200),
  title = "Correlation Matrix: UKInflation Data",
  mar = c(0,0,1,0)
)
```



The correlation matrix for UK inflation data shows a strong positive correlation (0.87) between actual and expected inflation values, indicating that expectations closely track reality. The values along the diagonal (1.00) confirm perfect self-correlation.

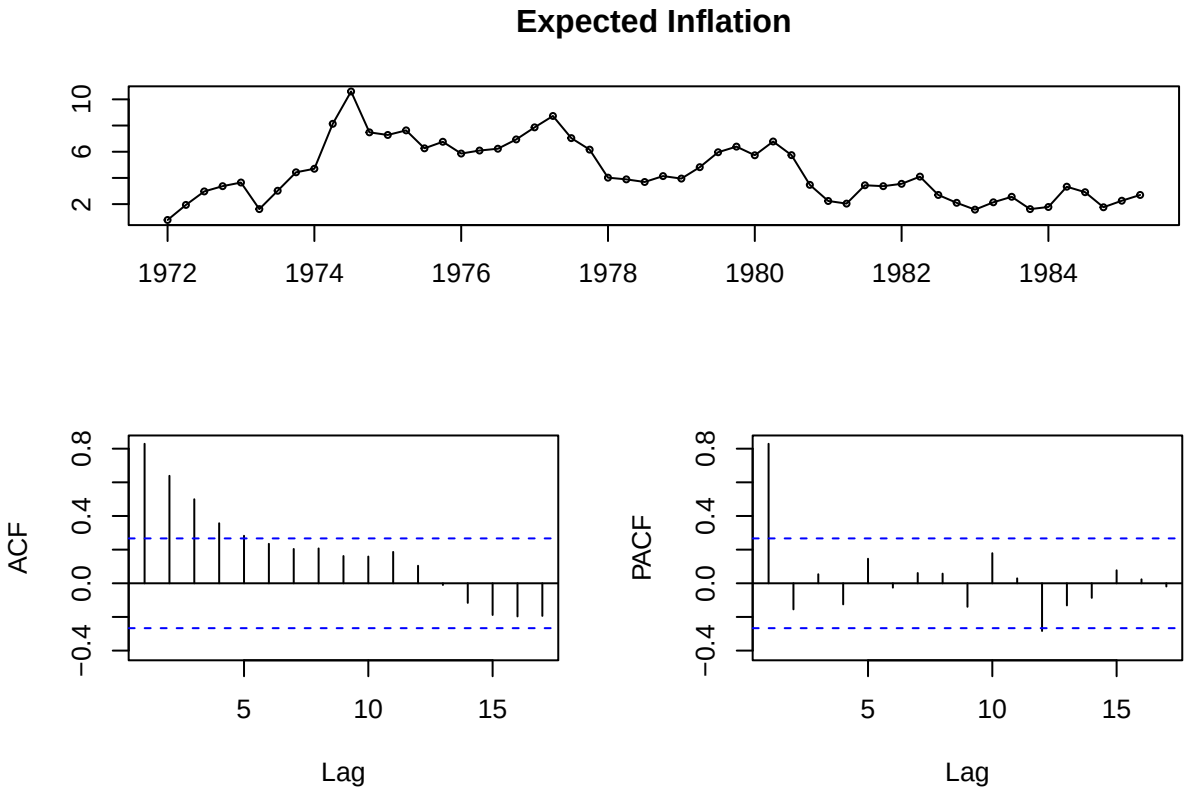
Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Question 2

a) Testing for Stationarity

Visual Inspection with Plots

```
# Plotting the UKInflation time series, ACF and PCF for expected inflation
tsdisplay(UKInflation[, "expected"], main = "Expected Inflation")
```



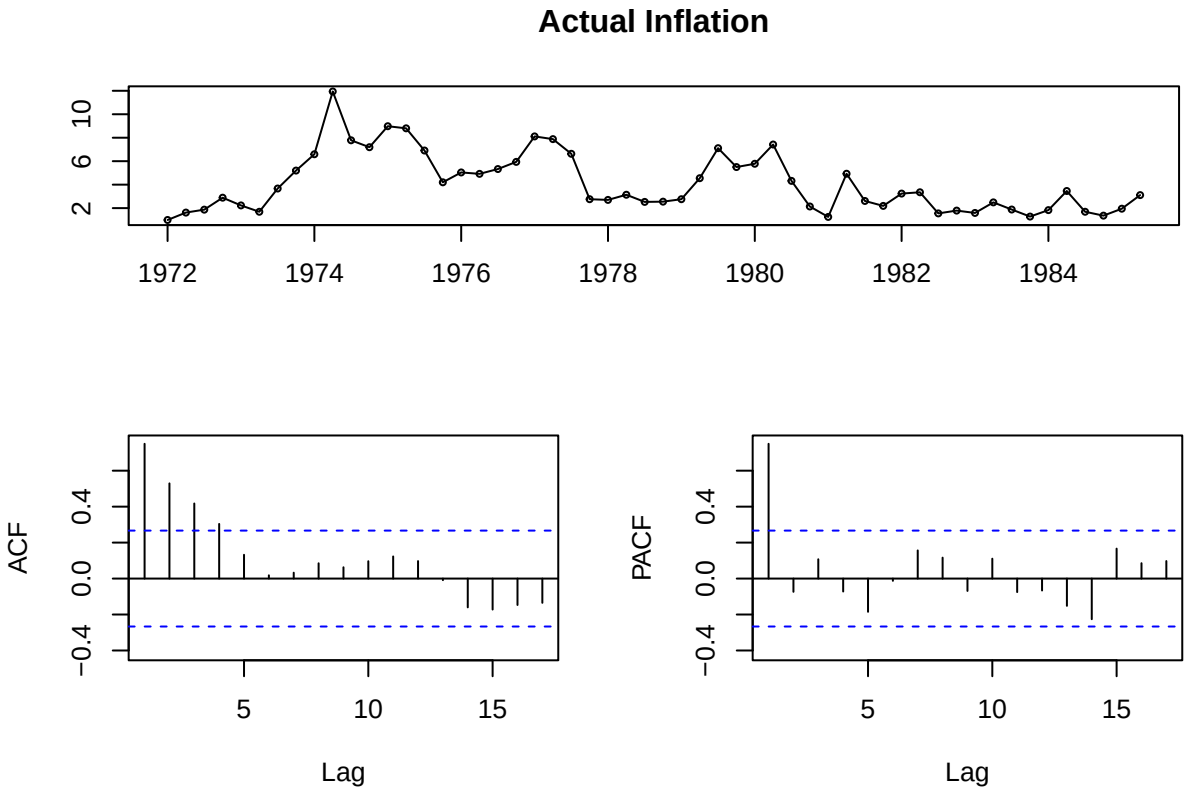
The time series plot of the 'actual' inflation variable shows a downward trend, indicating non-stationarity in the data. While there is a slight wave pattern, this is likely due to short-term cycles or trends that do not repeat at fixed intervals, rather than seasonality. This makes sense economically, as inflation typically does not follow a predictable seasonal trend within each year. Additionally, there appears to be a decrease in variance over time, which further suggests non-stationarity in the data.

Additionally, the ACF plot exhibits a downward staircase pattern, which suggests a persistent trend and confirms the series is non-stationary. This can be confirmed with the Dickey-fuller tests later.

From the PACF, the order may be 1, as it is the first lag where the PACF cuts off after being significant.

```
library(forecast)

# Plotting the UKInflation time series, ACF and PCF for actual inflation
tsdisplay(UKInflation[, "actual"], main = "Actual Inflation")
```



The time series plot of the 'actual' inflation variable reveals a downward trend, indicating non-stationarity in the data. While there is a slight wave pattern, this is likely due to short-term cycles or trends that do not repeat at fixed intervals, rather than seasonality. This makes sense economically, as inflation typically does not follow a predictable seasonal trend within each year. Additionally, there appears to be a decrease in variance over time, which further suggests non-stationarity in the data.

Additionally, the ACF plot exhibits a downward staircase pattern, which suggests a persistent trend and confirms the series is non-stationary. This can be confirmed with the Dickey-fuller tests now.

From the PACF, the order may be 1, as it is the first lag where the PACF cuts off after being significant.

```
install.packages("tseries")
```

Dickey-Fuller Test

```
##
## The downloaded binary packages are in
##  /var/folders/24/d5p4stj55zx8y3rtzgx0k84m0000gn/T//RtmpRnJozf/downloaded_packages
```

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.3.3
```

```
# Doing the ADF test on expected inflation
adf_result <- adf.test(UKInflation[, "expected"], alternative = "stationary")
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  UKInflation[, "expected"]
```

```
## Dickey-Fuller = -3.1253, Lag order = 3, p-value = 0.1213
## alternative hypothesis: stationary
```

As the p-value > 0.05, we fail to reject the null hypothesis. Therefore, 'expected' inflation is non-stationary.

```
library(tseries)

# Doing the ADF test on actual inflation
adf_result <- adf.test(UKInflation[, "actual"], alternative = "stationary")
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  UKInflation[, "actual"]
## Dickey-Fuller = -3.1302, Lag order = 3, p-value = 0.1194
## alternative hypothesis: stationary
```

As the p-value > 0.05, we fail to reject the null hypothesis. Therefore, 'actual' inflation is non-stationary.

b) Differencing and Re-testing for Stationarity

```
library(forecast)
library(urca)
```

```
## Warning: package 'urca' was built under R version 4.3.3
```

```
ndiffs(UKInflation[, "actual"])
```

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

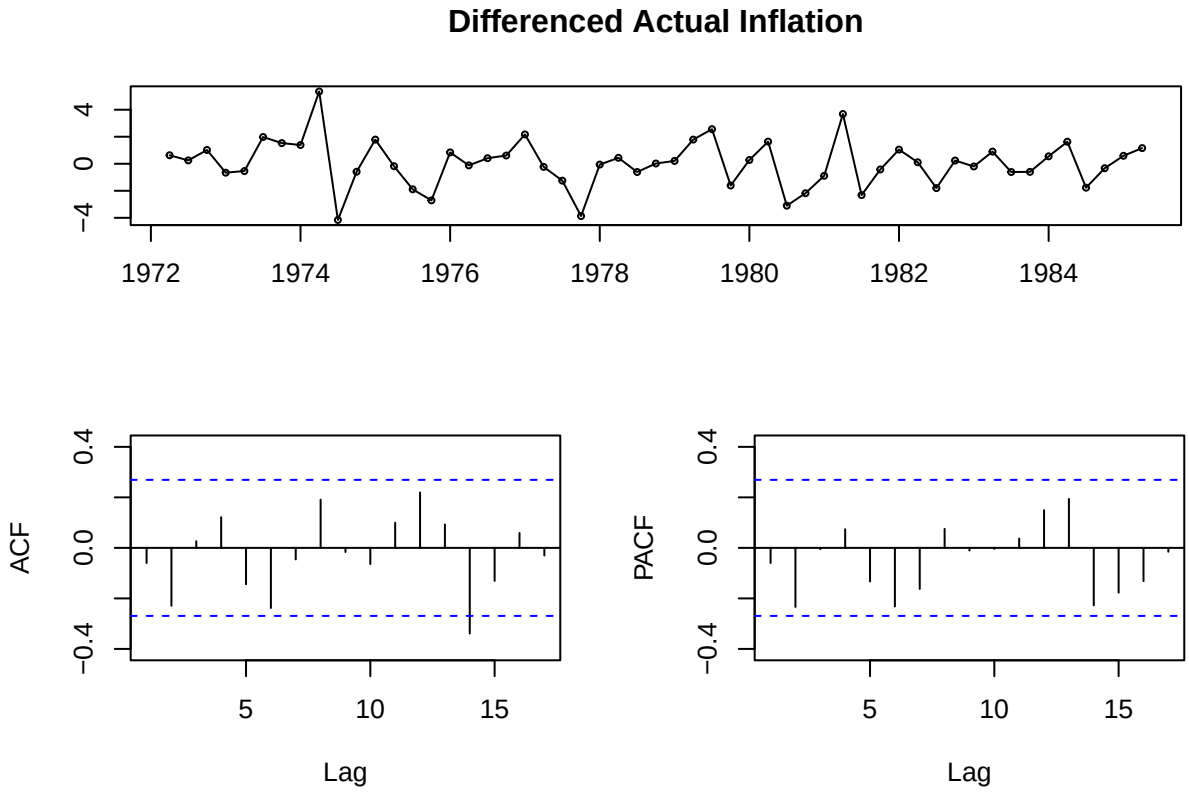
```
ndiffs(UKInflation[, "expected"])
```

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

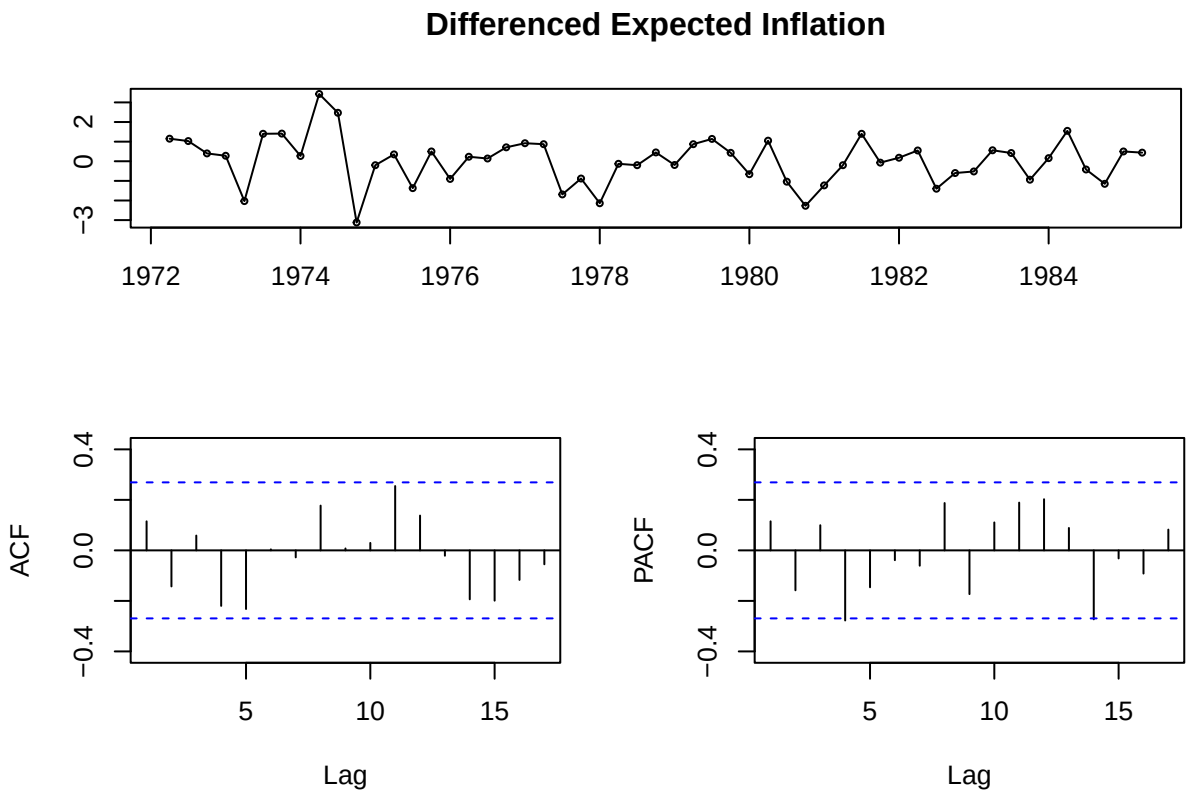
First-order differencing is required to induce stationarity.

```
par(mfrow=c(1,1))
```

```
actual_diff <- diff(UKInflation[, "actual"])
expected_diff <- diff(UKInflation[, "expected"])
tsdisplay(actual_diff, main = "Differenced Actual Inflation")
```




```
tsdisplay(expected_diff, main = "Differenced Expected Inflation")
```



After differencing the expected inflation series once, the time series appears to be stationary, as there are no clear trends or changing variability over time. The ACF and PACF plots show that most autocorrelations

fall within the confidence bands, indicating that there is little remaining structure in the data.

```
#Applying the Dickey-Fuller test on differenced data:

adf_actual_diff <- ur.df(actual_diff, type = "drift", lags = 1)
summary(adf_actual_diff)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0470 -0.7019  0.0545  0.8257  5.7926
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02653    0.24548   0.108   0.9144
## z.lag.1     -1.31365    0.20554  -6.391 6.31e-08 ***
## z.diff.lag   0.23690    0.14084   1.682   0.0991 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.753 on 48 degrees of freedom
## Multiple R-squared:  0.5549, Adjusted R-squared:  0.5364
## F-statistic: 29.92 on 2 and 48 DF,  p-value: 3.65e-09
##
##
## Value of test-statistic is: -6.3913 20.4267
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.51 -2.89 -2.58
## phi1  6.70  4.71  3.86
```

```
adf_expected_diff <- ur.df(expected_diff, type = "drift", lags = 1)
summary(adf_expected_diff)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8715 -0.6211  0.2745  0.6477  3.6228
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.003048   0.168411  -0.018    0.986
## z.lag.1     -1.038214   0.188171  -5.517 1.36e-06 ***
## z.diff.lag   0.157388   0.140808   1.118    0.269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.203 on 48 degrees of freedom
## Multiple R-squared:  0.4663, Adjusted R-squared:  0.4441
## F-statistic: 20.97 on 2 and 48 DF,  p-value: 2.852e-07
##
##
## Value of test-statistic is: -5.5174 15.2255
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.51 -2.89 -2.58
## phi1  6.70  4.71  3.86
```

The Augmented Dickey-Fuller (ADF) test was used to check if the expected inflation series is stationary. The test statistic for the lagged level of the series is -5.52, which is smaller than the 1% critical value of -3.51. This means we can reject the null hypothesis of a unit root at the 1% level, confirming that the differenced expected inflation series is stationary. This supports our earlier findings!

Question 3: Fitting Models

a) AR Model

From part 2a), we can observe that the ACF linearly declines, whereas the PACF abruptly cuts off after 1. This suggests that the time order should be 1, as it is the only significant time period that has an effect on current inflation levels.

```
install.packages("dynlm")
```

```
##
## The downloaded binary packages are in
##  /var/folders/24/d5p4stj55zx8y3rtzgx0k84m0000gn/T//RtmpRnJozf/downloaded_packages
```

```
library(dynlm)
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.3.3
```

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

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

```r
# fitting an AR(1) model with 1 lag
ar1_actual <- dynlm(actual_diff ~ L(actual_diff, 1))
summary(ar1_actual)
```

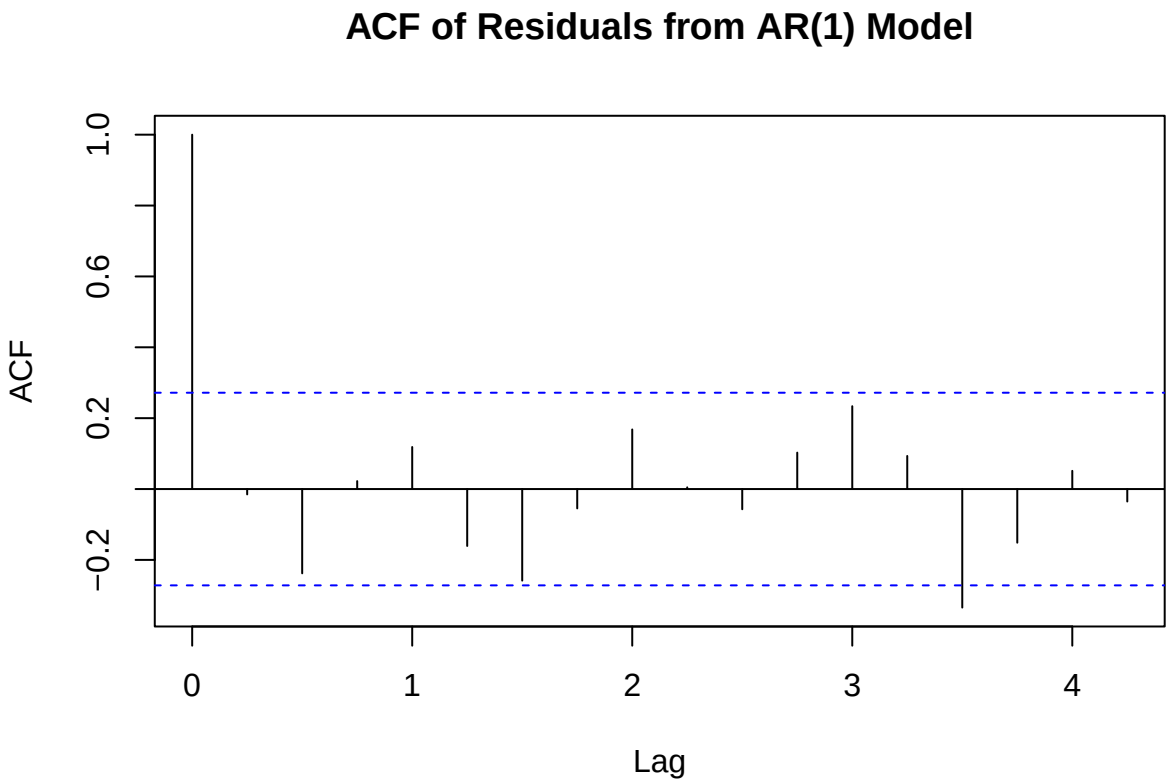
```
##
## Time series regression with "ts" data:
## Start = 1972(3), End = 1985(2)
##
## Call:
## dynlm(formula = actual_diff ~ L(actual_diff, 1))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9752 -0.7177  0.0275  0.9975  5.4042
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.02977    0.24518   0.121    0.904
## L(actual_diff, 1) -0.06038    0.14158  -0.426    0.672
##
## Residual standard error: 1.768 on 50 degrees of freedom
## Multiple R-squared:  0.003624,   Adjusted R-squared:  -0.0163
## F-statistic: 0.1818 on 1 and 50 DF,  p-value: 0.6716
```

From the summary table, we can see that the lag coefficient was not statistically significant, with a p-value $>$ 0.05. The R-squared value is 0.0036, suggesting a poor fit. Therefore, it may be the case that actual inflation in the period before is not a good predictor of current inflation.

b) Autocorrelation Examination

```r
# Extract residuals
resid_ar1_actual <- residuals(ar1_actual)

# Plot ACF of residuals
acf(resid_ar1_actual, main = "ACF of Residuals from AR(1) Model")
```



From the ACF of the residuals, we can see that there does not seem to be any serial correlation present. This is because there is only one significant spike at lag 0, which is expected as it is correlation with itself. This suggests autocorrelation is not present in the model, and we can further check this with a formal test such as the BG test:

```
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.3.3
```

```
bgtest(ar1_actual, order = 1)  # testing for autocorrelation up to lag 1
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  ar1_actual
## LM test = 2.0437, df = 1, p-value = 0.1528
```

As the p-value > 0.05, we fail to reject the null hypothesis that there is no serial correlation present. Therefore, we don't need to correct for autocorrelation.

c) ARDL Model

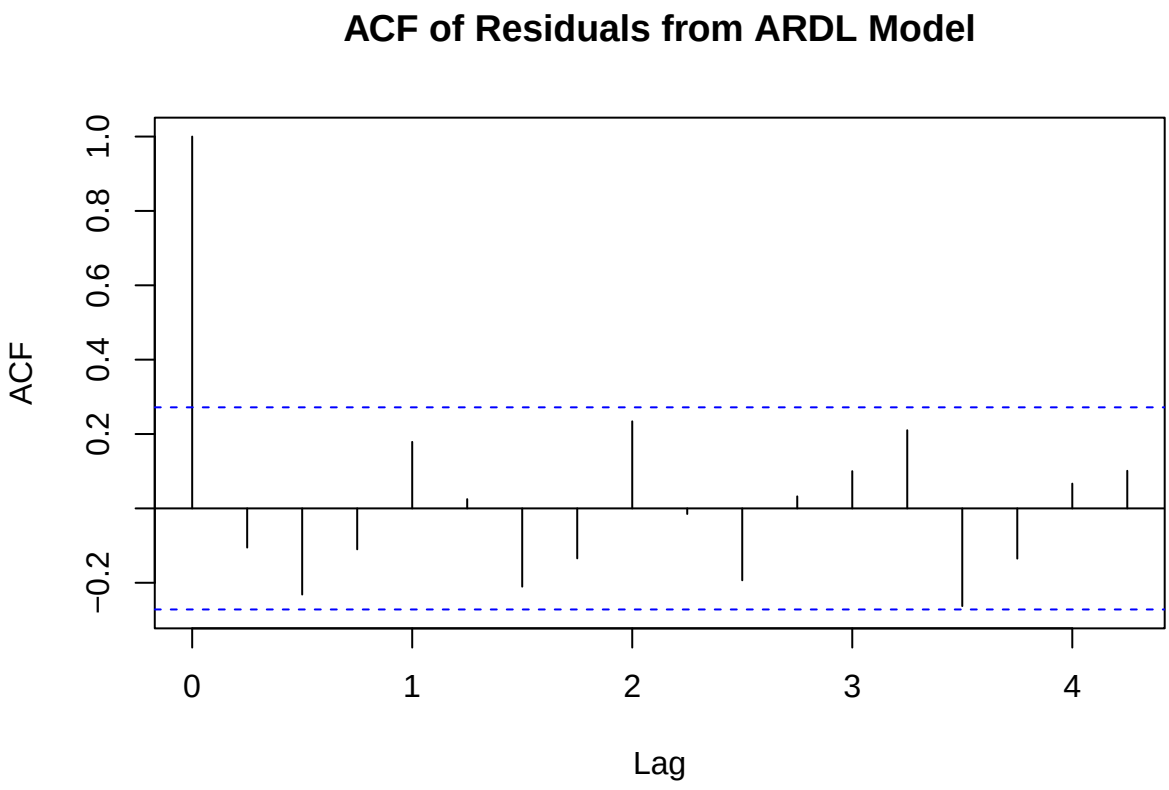
```
library(dynlm)
library(lmtest)
ardl_model <- dynlm(actual_diff ~ L(actual_diff, 1) + expected_diff + L(expected_diff, 1))
summary(ardl_model)
```

```
##
## Time series regression with "ts" data:
## Start = 1972(3), End = 1985(2)
##
```

```
## Call:
## dynlm(formula = actual_diff ~ L(actual_diff, 1) + expected_diff +
##     L(expected_diff, 1))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5717 -0.3440  0.0658  0.5801  3.3121
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.02301    0.19720   0.117 0.907602
## L(actual_diff, 1)    -0.62993    0.17523  -3.595 0.000764 ***
## expected_diff         1.22926    0.23555   5.219 3.81e-06 ***
## L(expected_diff, 1) -0.02447    0.18638  -0.131 0.896102
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.422 on 48 degrees of freedom
## Multiple R-squared:  0.3816, Adjusted R-squared:  0.3429
## F-statistic: 9.873 on 3 and 48 DF,  p-value: 3.5e-05
```

To evaluate whether an ARDL model is appropriate, we first explored the UK inflation data from 1972 to 1985 using summary statistics and visualizations like ggplot2 plots and correlation matrices. This helped us understand the basic structure and relationships in the data. We then conducted ADF tests on both actual and expected inflation to check for stationarity. The results showed that the series are integrated of different orders (some I(0), some I(1)), which makes ARDL a suitable modeling choice since it can handle a mix of stationary and non-stationary variables.

```
acf(residuals(ardl_model), main = "ACF of Residuals from ARDL Model")
```



```r
bgtest(ardl_model, order = 1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  ardl_model
## LM test = 2.2349, df = 1, p-value = 0.1349
```

The ACF plot of the residuals from the ARDL model indicates that most autocorrelations fall within the 95% confidence bands, suggesting that the residuals are largely uncorrelated. This supports the adequacy of the model specification, as it implies that the ARDL has effectively captured the underlying dynamics of the data.

Question 4: Brief Summary

This analysis explored the relationship between expected and actual inflation in the UK manufacturing sector from 1972 to 1985 using time series techniques. Through exploratory data analysis, we found a strong positive correlation (0.87) between expected and actual inflation. Both series were non-stationary in levels, as confirmed by the Augmented Dickey-Fuller (ADF) test, but became stationary after first differencing.

We began model building by fitting an AR(1) model to the differenced actual inflation series, but the lag coefficient was not statistically significant, and the model had poor explanatory power ($R^2 = 0.0036$). This suggests that past values of actual inflation alone may not effectively predict current inflation.

Next, we fit an Autoregressive Distributed Lag (ARDL) model including both lagged and contemporaneous expected inflation terms. This model showed better fit and more significant predictors, indicating that expected inflation contains useful information for modeling actual inflation dynamics. ACF and Breusch-Godfrey tests showed no evidence of autocorrelation, suggesting the ARDL model was well-specified.

In summary, our findings support the idea that expectations play a meaningful role in predicting inflation, consistent with economic theories about the rationality and informational value of inflation expectations.

Question 5: Limitations

While our analysis provides valuable insights into the relationship between expected and actual inflation in the UK manufacturing sector, there are several limitations to keep in mind.

First, the dataset is relatively small, with only 54 quarterly observations between 1972 and 1985. This makes it harder to capture long-term trends or draw strong conclusions about patterns that might appear over larger time frames.

Second, while the ARDL model can handle mixed levels of stationarity, our model still has a fairly low explanatory power. For instance, the AR(1) model's R-squared value is very low, suggesting that past values of inflation alone may not be enough to explain current changes in inflation. Real-world inflation is influenced by many factors — such as global economic shocks, changes in government policy, oil prices, and labor market dynamics — that aren't included in our model.

Third, we assume linear relationships, which may oversimplify how expectations and actual inflation interact. In reality, these relationships might be more complex or even asymmetric — for example, people might react more strongly to unexpected increases in inflation than decreases.

Lastly, the dataset reflects a specific historical context, including major economic events like the 1970s oil crisis and tight monetary policies in the early 1980s. This means the findings might not generalize well to other periods or countries.

In future research, using a larger, more recent dataset and including more economic variables could help improve the model's accuracy and relevance.

Part 2 - QDV Models

Question 1

a) Data Description & Hypothesis

The dataset we will be using for this next part of the project is the Customer Churn dataset from Kaggle, which includes 64,374 customer records. Each record provides information about a customer's behavior and demographics, such as age (continuous), usage frequency (continuous), payment delays (continuous), subscription type (categorical), total spend (continuous), and more. Our qualitative dependent variable is churn, who's value of 1 indicates that the customer has churned (i.e., stopped being a customer) and 0 indicates they have not. Note that although there are several other variables included in the dataset, we have narrowed them to these 6 as we believe they hold the most significance.

The goal of our model is to predict whether a customer is likely to churn based on their characteristics and activity patterns. This allows us to identify at-risk customers and help businesses develop strategies to improve customer retention.

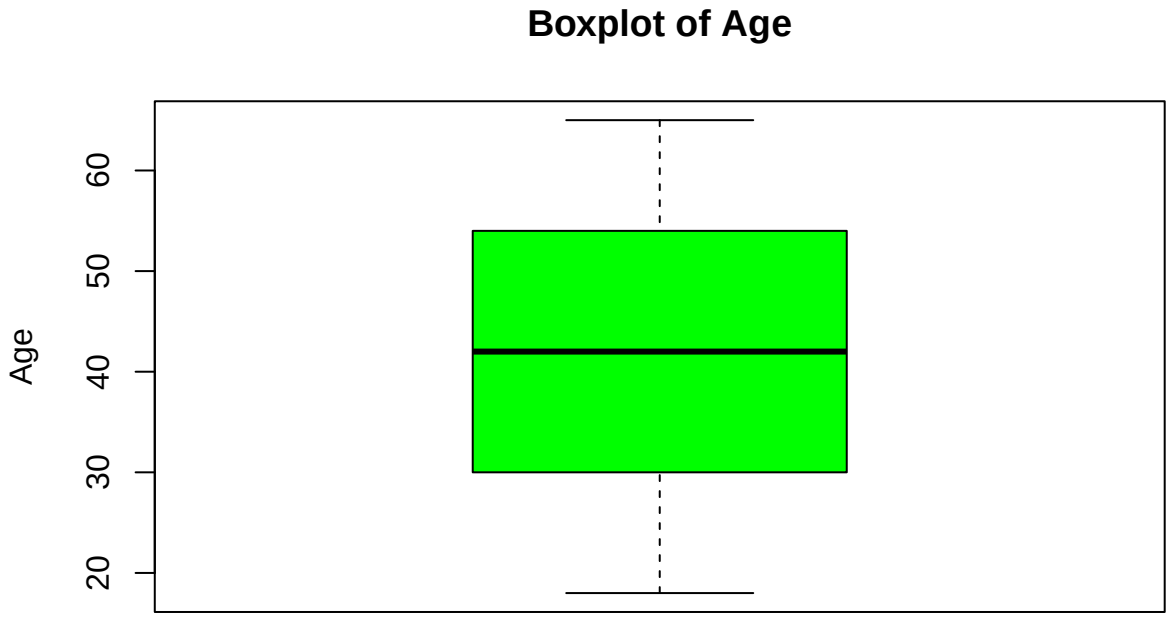
Dataset link: https://www.kaggle.com/datasets/muhammadshahidazeem/customer-churn-dataset/data?select=customer_churn_dataset-training-master.csv

b) Descriptive Analysis

This table displays the first 10 rows of a customer dataset containing 64,374 observations and 12 variables, including demographics, service usage, and subscription details. The data will be used to analyze customer behavior patterns and factors influencing subscription types.

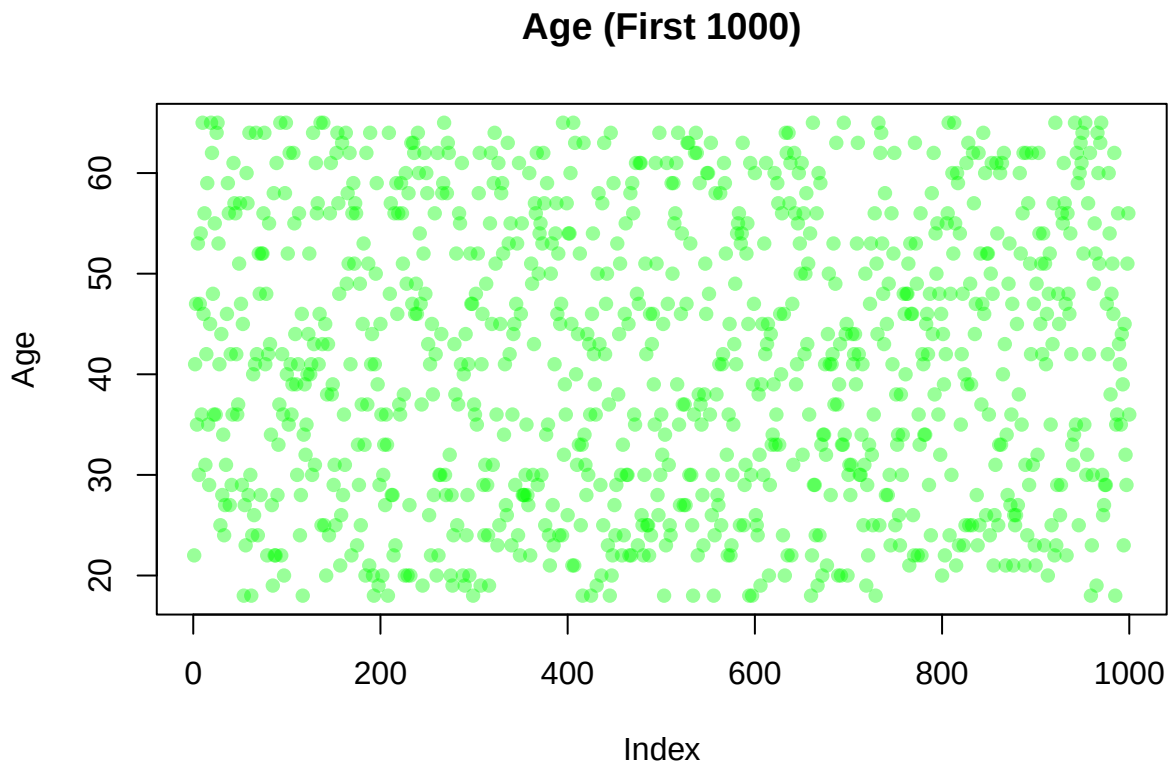
```
boxplot(data$Age, main = "Boxplot of Age", col = "green", ylab = "Age")
```

AGE



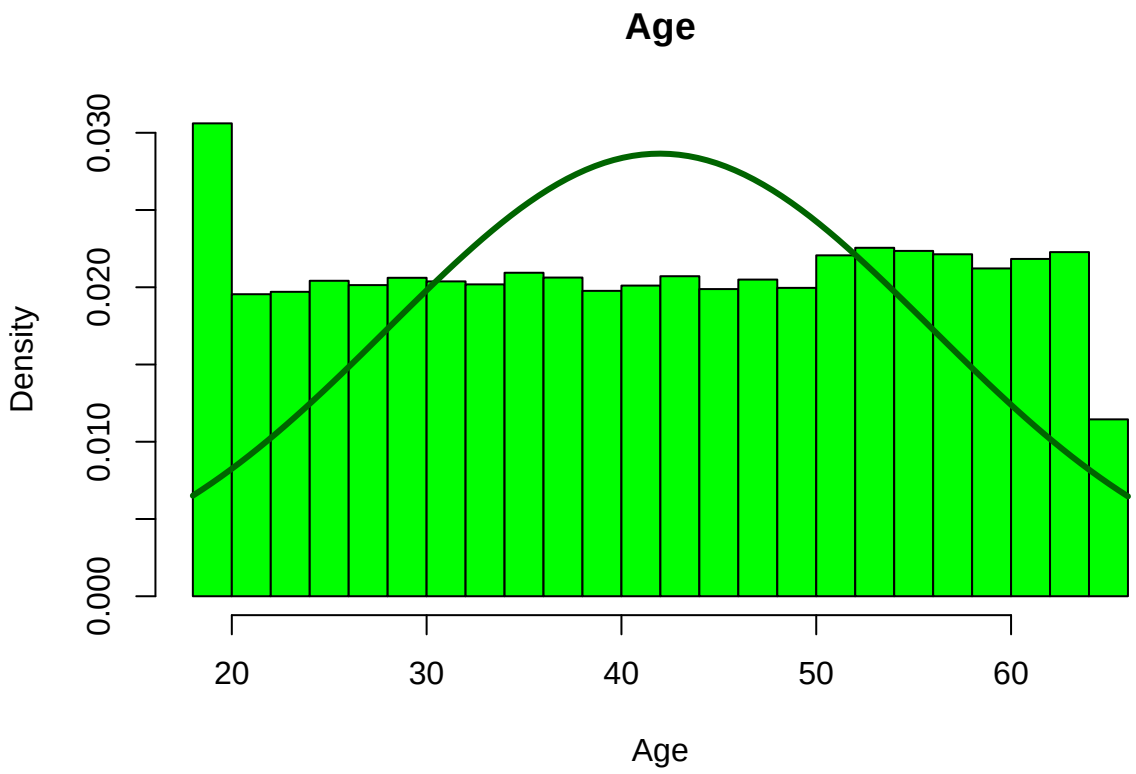

```
plot(data$Age[1:1000],
     type = "p",
     main = "Age (First 1000)",
     ylab = "Age",
     xlab = "Index",
```

```
     col = rgb(0, 1, 0, 0.4),  # green with transparency
     pch = 16)
```




```
hist(data$Age, main = "Age", xlab = "Age", col = "green", probability = TRUE)

curve(dnorm(x, mean = mean(data$Age, na.rm = TRUE), sd = sd(data$Age, na.rm = TRUE)),
      col = "darkgreen", lwd = 3, add = TRUE)
```



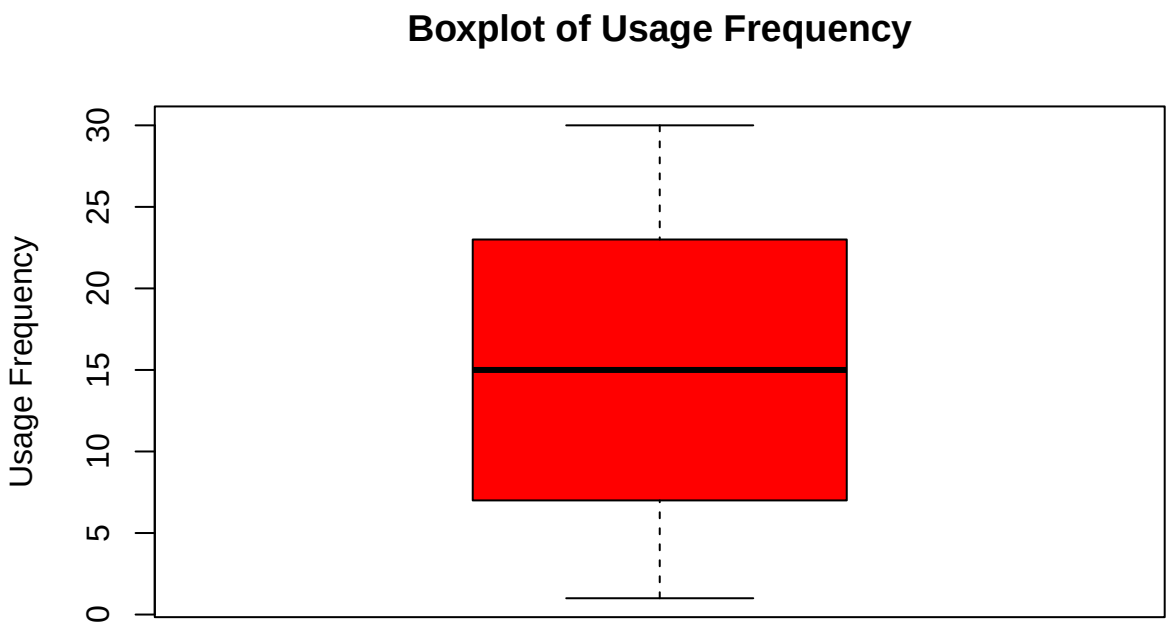

```
summary(data$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   30.00   42.00   41.97   54.00   65.00
```

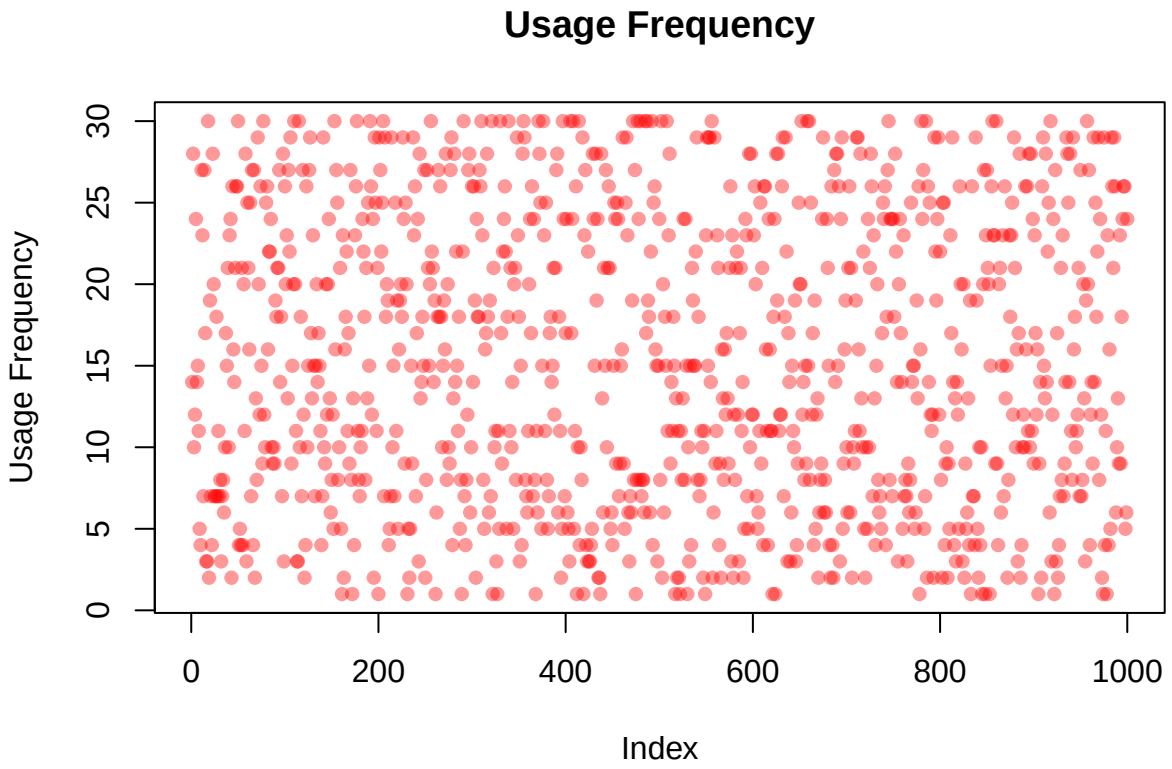
The plot displays customer age distribution, with a dense green overlay indicating a high number of data points. The boxplot shows a median age of 40 with no outliers, suggesting a relatively symmetric spread. The histogram reveals a roughly normal distribution, though slightly right-skewed, with the youngest age group being the most frequent.

```
boxplot(data$Usage.Frequency, main = "Boxplot of Usage Frequency", col = "red",
        ylab = "Usage Frequency")
```

USAGE FREQUENCY Charts:

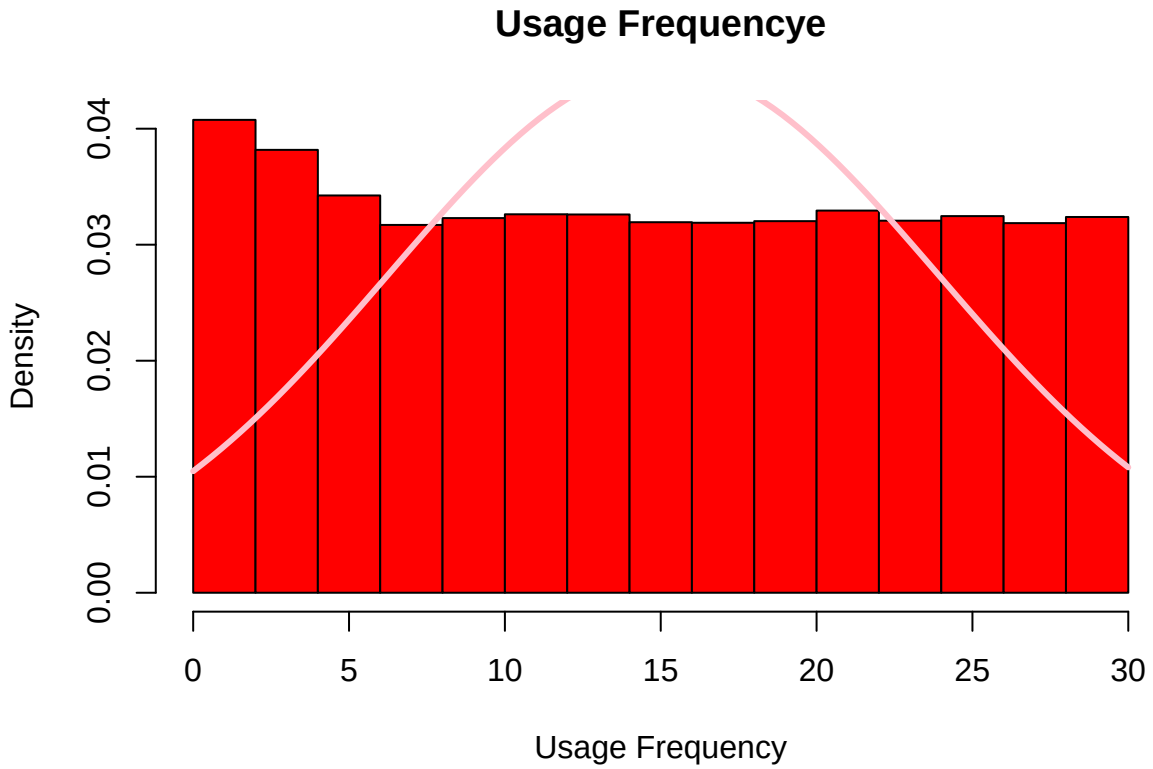



```r
plot(data$Usage.Frequency[1:1000],
     type = "p",
     main = "Usage Frequency",
     ylab = "Usage Frequency",
     xlab = "Index",
     col = rgb(1, 0, 0, 0.4),  # semi-transparent red
     pch = 16)
```




```r
hist(data$Usage.Frequency, main = "Usage Frequencye", xlab = "Usage Frequency",
     col = "red", probability = TRUE)
```

```
curve(dnorm(x, mean = mean(data$Usage.Frequency, na.rm = TRUE),
            sd = sd(data$Usage.Frequency, na.rm = TRUE)),  col = "pink",
            lwd = 3, add = TRUE)
```



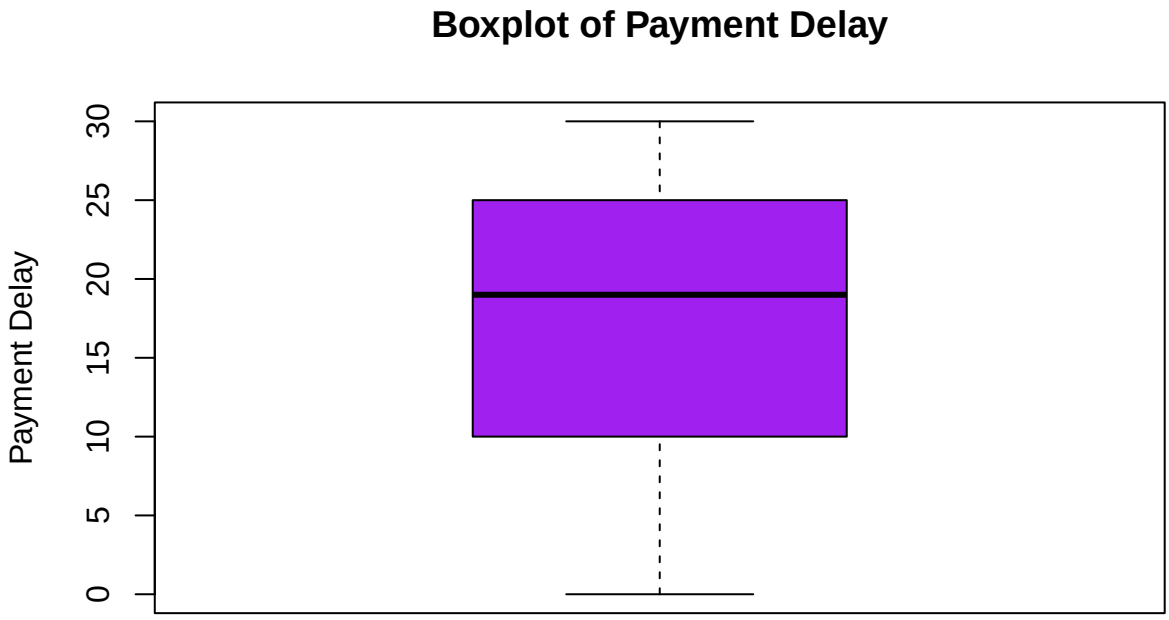

```
summary(data$Usage.Frequency)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    7.00   15.00   15.08   23.00   30.00
```

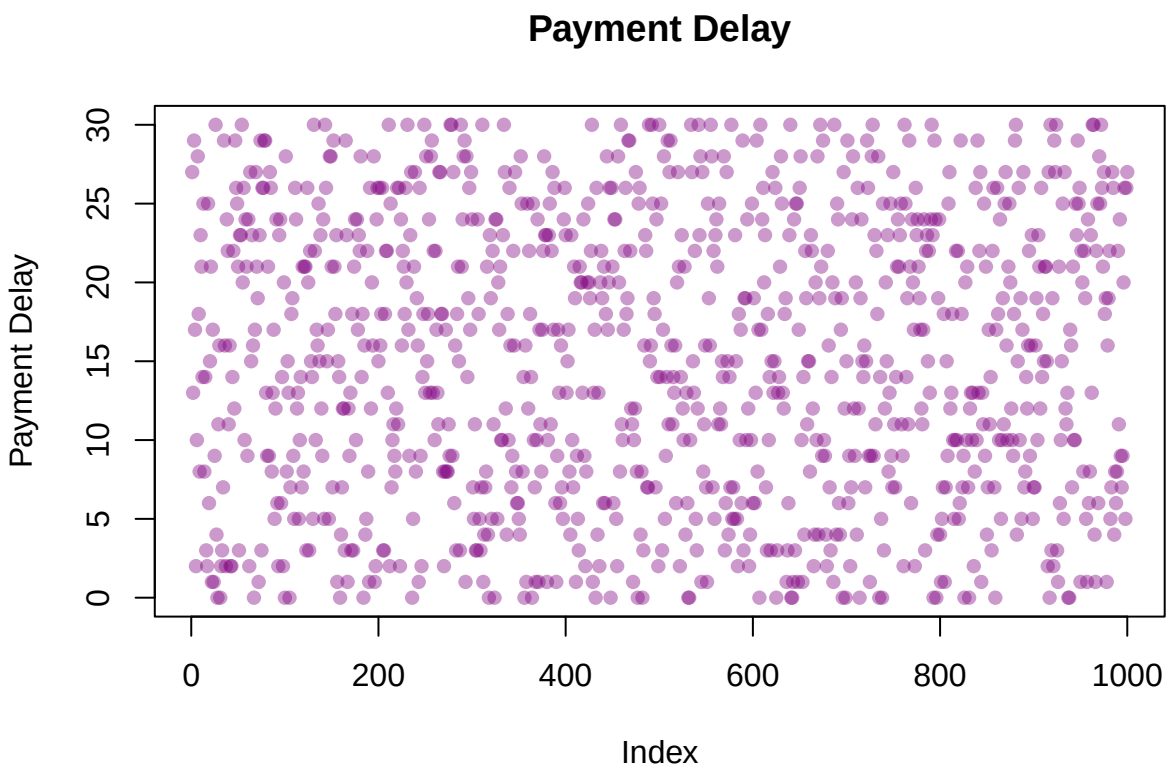
The plot displays customer usage frequency, with a relatively denser red overlay indicating a high number of data points. The boxplot shows a median of 15 with no outliers, suggesting a relatively symmetric spread. The histogram reveals a roughly normal distribution, though slightly right-skewed.

```
boxplot(data$Payment.Delay, main = "Boxplot of Payment Delay", col = "purple",
        ylab = "Payment Delay")
```

PAYMENT DELAYS Charts:

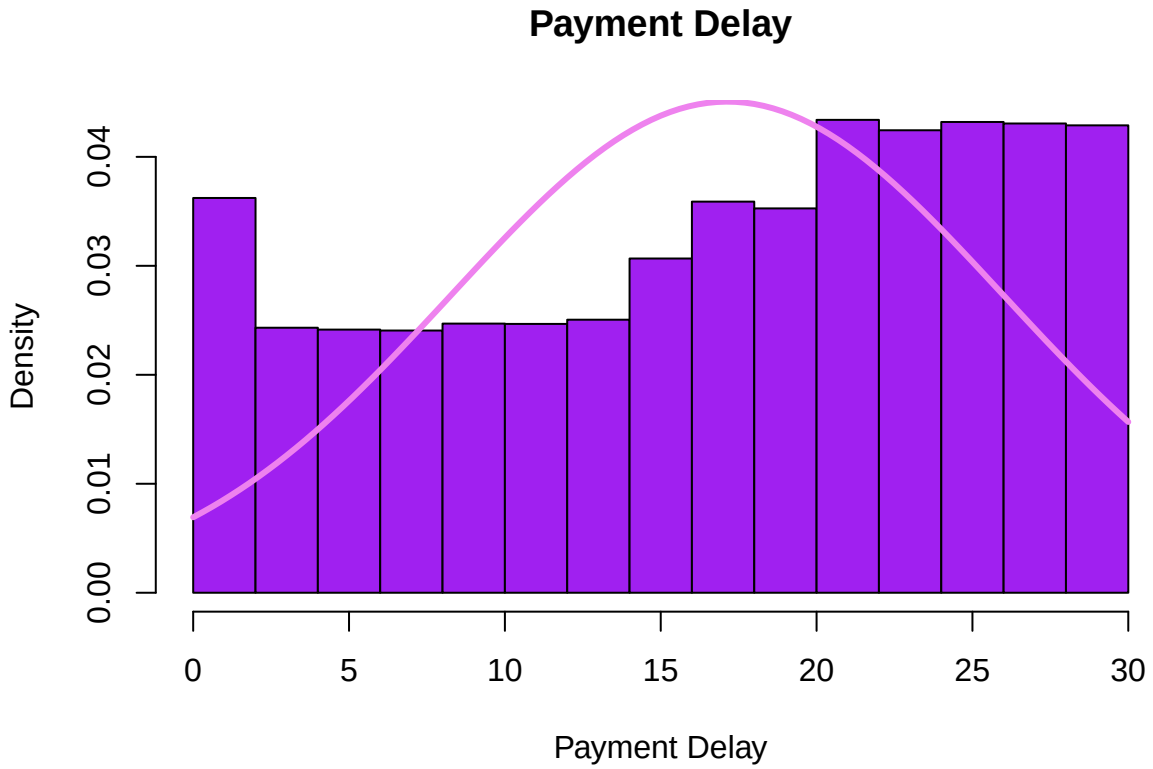



```r
plot(data$Payment.Delay[1:1000],
     type = "p",
     main = "Payment Delay",
     ylab = "Payment Delay",
     xlab = "Index",
     col = rgb(0.5, 0, 0.5, 0.4),  # semi-transparent purple
     pch = 16)
```




```r
hist(data$Payment.Delay, main = "Payment Delay", xlab = "Payment Delay",
     col = "purple", probability = TRUE)
```

```
curve(dnorm(x, mean = mean(data$Payment.Delay, na.rm = TRUE),
            sd = sd(data$Payment.Delay, na.rm = TRUE)),  col = "violet",
            lwd = 3, add = TRUE)
```




```
summary(data$Payment.Delay)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.00   19.00   17.13   25.00   30.00
```

The plot displays customer payment delays, with a relatively denser purple overlay indicating a high number of data points. The boxplot shows a median of approximately 20, with no outliers. The histogram reveals a roughly normal distribution, though slightly left-skewed, with more observations towards higher values of payment delays.

```
table(data$Subscription.Type)
```

SUBSCRIPTION TYPE ~ This is a binary variable so we will check proportions.

```
##
##    Basic  Premium Standard
##    21451    21421    21502
```

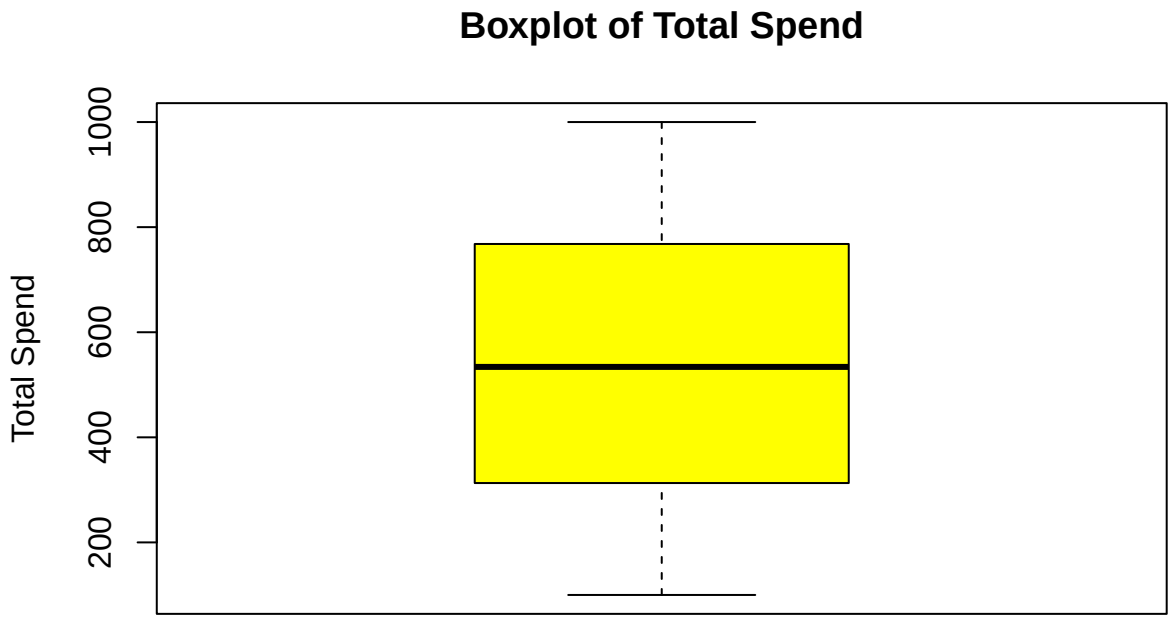
```
prop.table(table(data$Subscription.Type))
```

```
##
##     Basic   Premium  Standard
## 0.3332246 0.3327586 0.3340168
```

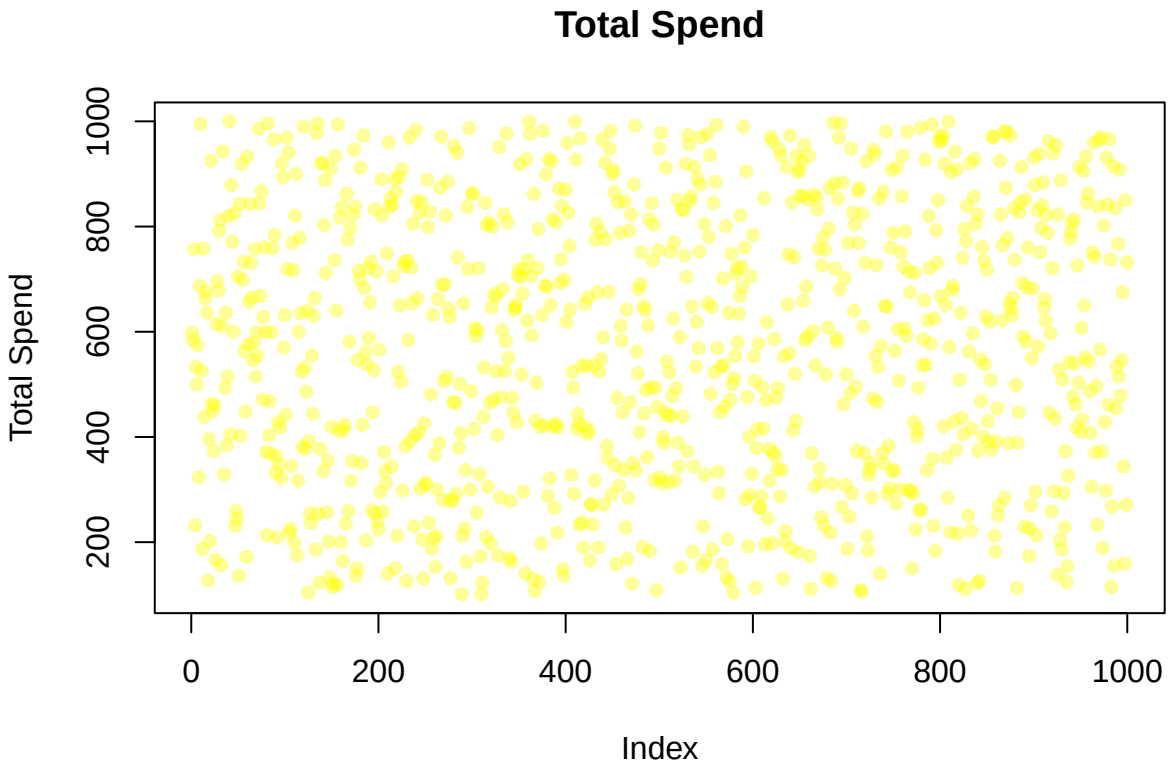
The output shows that the dataset contains an almost unequal distribution of users across the three subscription types: Basic, Premium, and Standard. Each type makes up roughly one-third of the total, indicating no significant imbalance in subscription representation.

```r
boxplot(data$Total.Spend, main = "Boxplot of Total Spend", col = "yellow", ylab = "Total Spend")
```

TOTAL SPEND Charts:

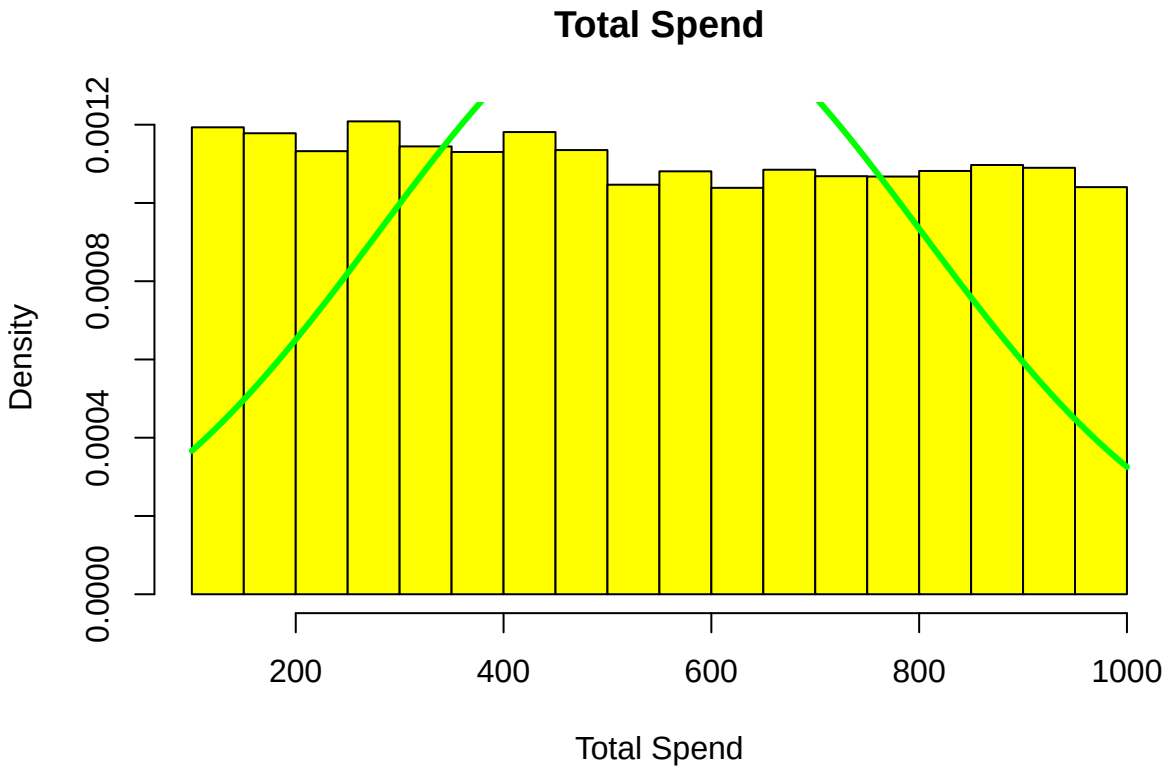



```r
plot(data$Total.Spend[1:1000],
     type = "p",
     main = "Total Spend",
     ylab = "Total Spend",
     xlab = "Index",
     col = rgb(1, 1, 0, 0.4),  # semi-transparent yellow
     pch = 16)
```




```
hist(data$Total.Spend, main = "Total Spend", xlab = "Total Spend",
     col = "yellow", probability = TRUE)

curve(dnorm(x, mean = mean(data$Total.Spend, na.rm = TRUE),
            sd = sd(data$Total.Spend, na.rm = TRUE)),  col = "green",
            lwd = 3, add = TRUE)
```



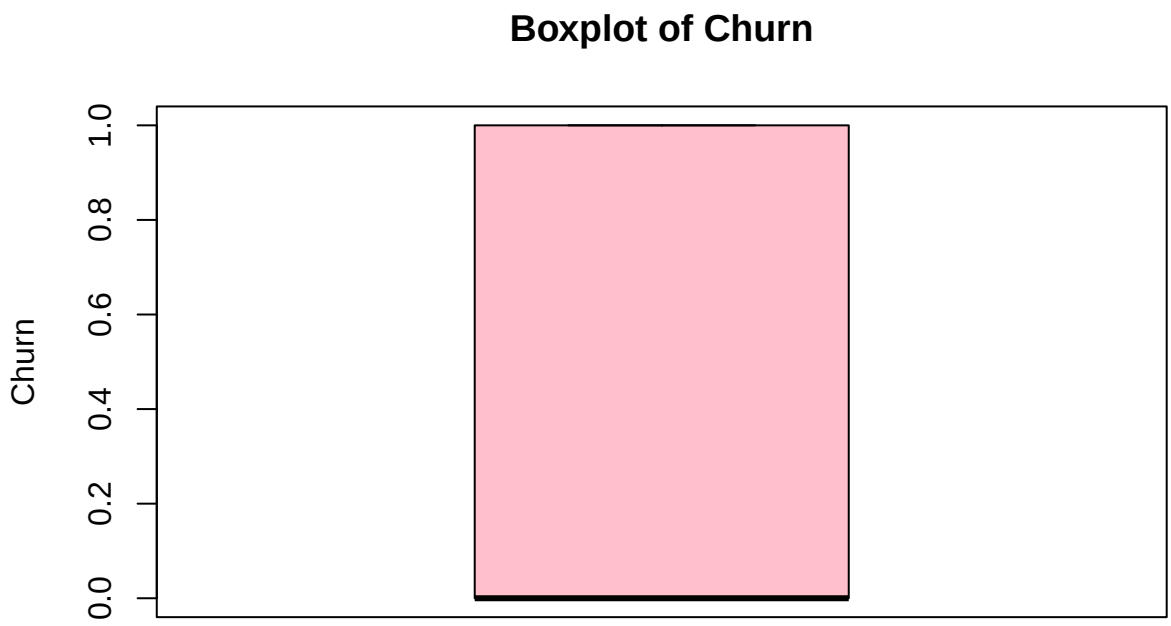

```
summary(data$Total.Spend)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     100     313     534     541     768    1000
```

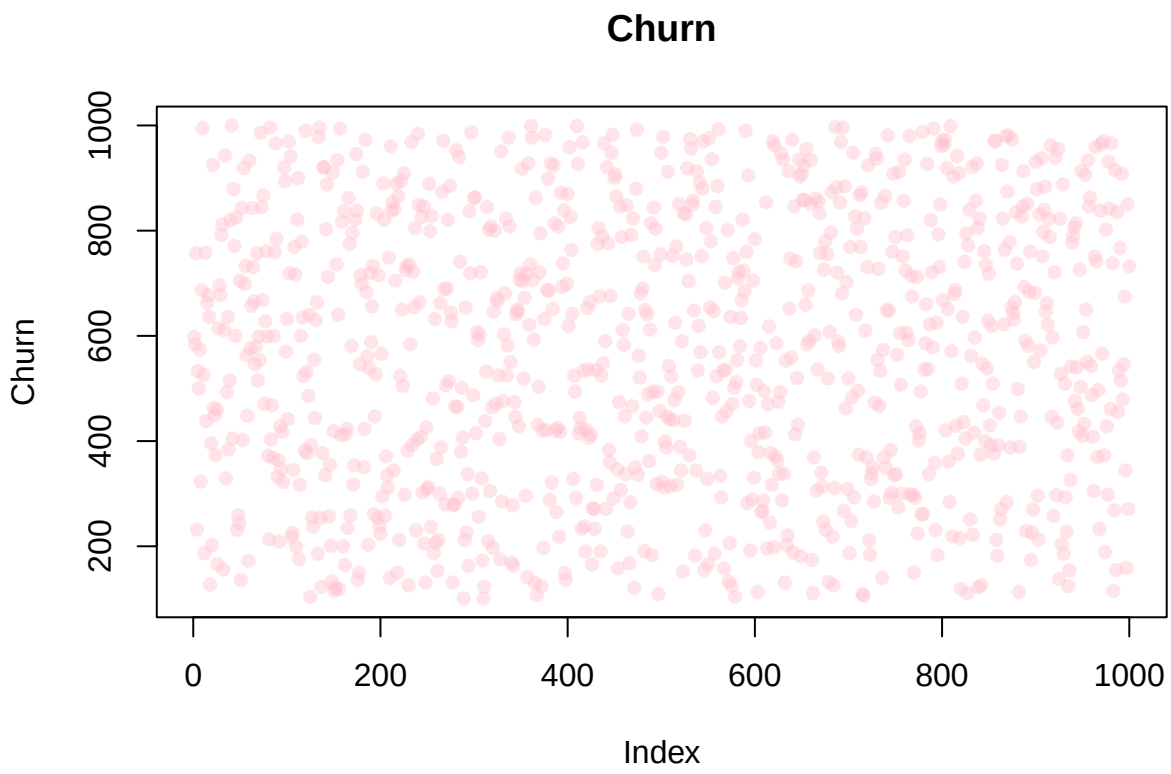
The plot displays total spending by customers, with a relatively denser yellow overlay indicating a high number of data points. The boxplot shows a median of around 550, suggesting a relatively symmetric spread. The histogram reveals a normal distribution.

```
boxplot(data$Churn, main = "Boxplot of Churn", col = "pink", ylab = "Churn")
```

CHURN Charts:

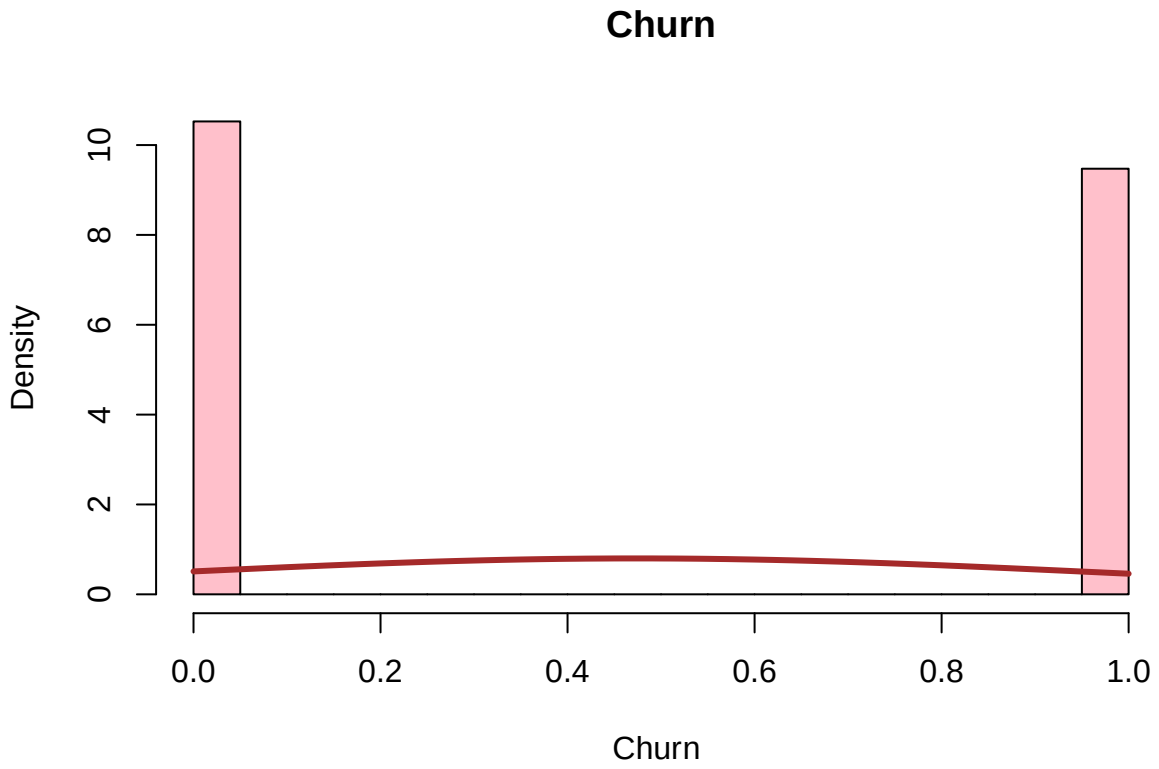



```r
plot(data$Total.Spend[1:1000],
     type = "p",
     main = "Churn",
     ylab = "Churn",
     xlab = "Index",
     col = rgb(1, 0.75, 0.8, 0.4),  # semi-transparent pink
     pch = 16)
```




```r
hist(data$Churn, main = "Churn", xlab = "Churn ", col = "pink", probability = TRUE)
```

```
curve(dnorm(x, mean = mean(data$Churn, na.rm = TRUE),
            sd = sd(data$Churn, na.rm = TRUE)),  col = "brown",
            lwd = 3, add = TRUE)
```




```
summary(data$Churn)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4737  1.0000  1.0000
```

This binary variable is dominated by the 0 value, as shown by the taller bar in the histogram compared to 1. The boxplot shows a flat distribution with the median at 0, indicating a concentration of responses at that value. The scatterplot appears as a rectangular block spread across the plot area, reflecting the binary nature and frequency of repeated values.

```
install.packages("corrplot")
```

CORRELATION PLOT

```
##
## The downloaded binary packages are in
##  /var/folders/24/d5p4stj55zx8y3rtzgx0k84m0000gn/T//RtmpRnJozf/downloaded_packages
```

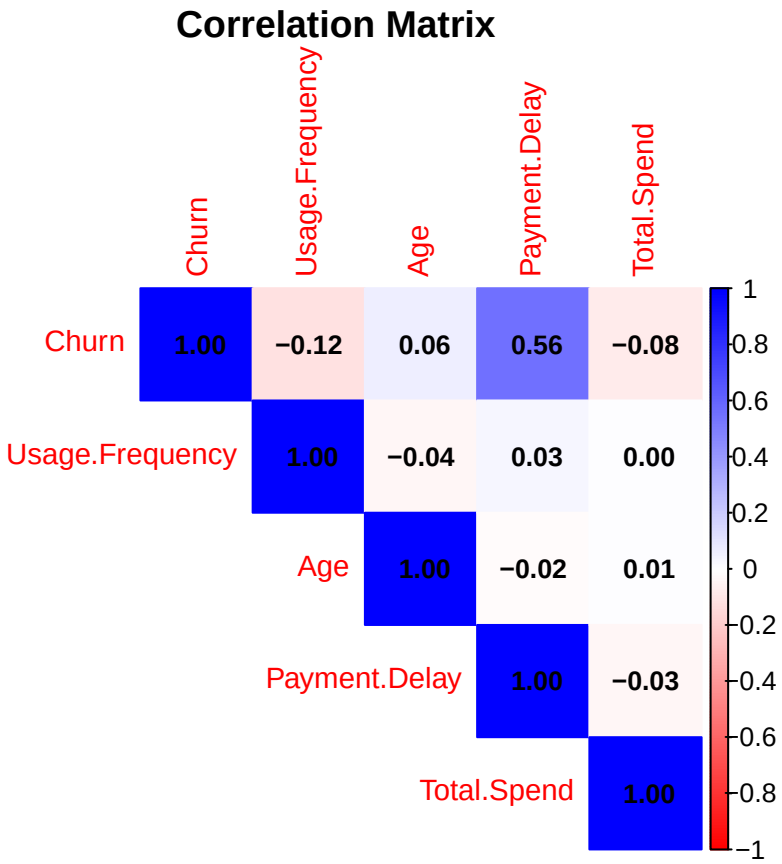
```
library(corrplot)

vars <- data[, c("Churn", "Usage.Frequency", "Age", "Payment.Delay", "Total.Spend")]

cor_matrix <- cor(vars, use = "complete.obs")

corrplot(cor_matrix,
         method = "color",
```

```
         type = "upper",
         addCoef.col = "black",
         tl.cex = 0.9,
         number.cex = 0.8,
         col = colorRampPalette(c("red", "white", "blue"))(200),
         title = "Correlation Matrix",
         mar = c(0,0,1,0))
```



The correlation matrix reveals that Churn has a moderate positive correlation with Payment.Delay (0.56), indicating that customers who delay payments are more likely to leave. There is a weak negative correlation between Churn and Usage.Frequency (-0.12), suggesting that more frequent users are slightly less likely to churn. Age (0.06) and Total.Spend (-0.08) show minimal correlation with Churn, implying little direct relationship. Other pairwise correlations are also weak, indicating no strong multicollinearity among the variables. Payment.Delay appears to be the most promising predictor of churn.

c) Fitting Models

```
## review data
str(data)
```

```
## 'data.frame':    64374 obs. of  12 variables:
##  $ CustomerID       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age              : int  22 41 47 35 53 30 47 54 36 65 ...
##  $ Gender           : chr  "Female" "Female" "Male" "Male" ...
##  $ Tenure           : int  25 28 27 9 58 41 37 36 20 8 ...
##  $ Usage.Frequency  : int  14 28 10 12 24 14 15 11 5 4 ...
##  $ Support.Calls    : int  4 7 2 5 9 10 9 0 10 2 ...
```

```
##  $ Payment.Delay    : int  27 13 29 17 2 10 28 18 8 23 ...
##  $ Subscription.Type: chr  "Basic" "Standard" "Premium" "Premium" ...
##  $ Contract.Length  : chr  "Monthly" "Monthly" "Annual" "Quarterly" ...
##  $ Total.Spend      : int  598 584 757 232 533 500 574 323 687 995 ...
##  $ Last.Interaction : int  9 20 21 18 18 29 14 16 8 10 ...
##  $ Churn            : int  1 0 0 0 0 0 1 0 0 0 ...
```

```
## cleaning data
data_clean <- na.omit(data[, c("Churn", "Age", "Usage.Frequency",
                               "Payment.Delay", "Subscription.Type", "Total.Spend")])

## fitting models

# Linear Probability Model
lpm_model <- lm(Churn ~ Age + Usage.Frequency + Payment.Delay +
                  Subscription.Type + Total.Spend, data = data_clean)
summary(lpm_model)
```

```
##
## Call:
## lm(formula = Churn ~ Age + Usage.Frequency + Payment.Delay +
##     Subscription.Type + Total.Spend, data = data_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.06101 -0.30509  0.05066  0.31649  1.05139
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 1.233e-02  7.789e-03   1.583  0.11339
## Age                         2.435e-03  1.152e-04  21.138  < 2e-16 ***
## Usage.Frequency            -7.354e-03  1.820e-04 -40.404  < 2e-16 ***
## Payment.Delay               3.162e-02  1.813e-04 174.436  < 2e-16 ***
## Subscription.TypePremium   -1.447e-02  3.928e-03  -3.684  0.00023 ***
## Subscription.TypeStandard  -9.181e-03  3.924e-03  -2.339  0.01932 *
## Total.Spend                -1.180e-04  6.147e-06 -19.188  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4067 on 64367 degrees of freedom
## Multiple R-squared:  0.3367, Adjusted R-squared:  0.3367
## F-statistic:  5446 on 6 and 64367 DF,  p-value: < 2.2e-16
```

```
# Probit Model
probit_model <- glm(Churn ~ Age + Usage.Frequency + Payment.Delay + Subscription.Type + Total.Spend,
                    data = data_clean, family = binomial(link = "probit"))
summary(probit_model)
```

```
##
## Call:
## glm(formula = Churn ~ Age + Usage.Frequency + Payment.Delay +
##     Subscription.Type + Total.Spend, family = binomial(link = "probit"),
##     data = data_clean)
##
## Coefficients:
```

```
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -1.628e+00  2.864e-02 -56.862  < 2e-16 ***
## Age                         9.122e-03  4.116e-04  22.161  < 2e-16 ***
## Usage.Frequency            -2.781e-02  6.597e-04 -42.152  < 2e-16 ***
## Payment.Delay               1.038e-01  7.716e-04 134.478  < 2e-16 ***
## Subscription.TypePremium   -4.993e-02  1.398e-02  -3.572 0.000355 ***
## Subscription.TypeStandard -3.256e-02  1.396e-02  -2.333 0.019673 *
## Total.Spend                -4.274e-04  2.193e-05 -19.484  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 89063  on 64373  degrees of freedom
## Residual deviance: 63693  on 64367  degrees of freedom
## AIC: 63707
##
## Number of Fisher Scoring iterations: 5
```

```r
# Logit Model
logit_model <- glm(Churn ~ Age + Usage.Frequency + Payment.Delay + Subscription.Type + Total.Spend,
                   data = data_clean, family = binomial(link = "logit"))
summary(logit_model)
```

```
##
## Call:
## glm(formula = Churn ~ Age + Usage.Frequency + Payment.Delay +
##     Subscription.Type + Total.Spend, family = binomial(link = "logit"),
##     data = data_clean)
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.769e+00  4.991e-02 -55.474  < 2e-16 ***
## Age                         1.502e-02  7.058e-04  21.288  < 2e-16 ***
## Usage.Frequency            -4.542e-02  1.138e-03 -39.926  < 2e-16 ***
## Payment.Delay               1.765e-01  1.430e-03 123.472  < 2e-16 ***
## Subscription.TypePremium   -8.455e-02  2.393e-02  -3.533 0.000411 ***
## Subscription.TypeStandard -5.668e-02  2.390e-02  -2.372 0.017715 *
## Total.Spend                -7.168e-04  3.761e-05 -19.058  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 89063  on 64373  degrees of freedom
## Residual deviance: 63708  on 64367  degrees of freedom
## AIC: 63722
##
## Number of Fisher Scoring iterations: 4
```

```r
## statistical diagnostics

# AIC / BIC
AIC(lpm_model, probit_model, logit_model)
```

```
##              df      AIC
## lpm_model     8 66850.82
## probit_model  7 63706.82
## logit_model   7 63722.06
```

```
BIC(lpm_model, probit_model, logit_model)
```

```
##              df      BIC
## lpm_model     8 66923.40
## probit_model  7 63770.32
## logit_model   7 63785.57
```

```
# Pseudo R-Squared
install.packages("pscl")
```

```
##
## The downloaded binary packages are in
##  /var/folders/24/d5p4stj55zx8y3rtzgx0k84m0000gn/T//RtmpRnJozf/downloaded_packages
```

```
library(pscl)
```

```
## Warning: package 'pscl' was built under R version 4.3.3
```

```
## Classes and Methods for R originally developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University (2002-2015),
## by and under the direction of Simon Jackman.
## hurdle and zeroinfl functions by Achim Zeileis.
```

```
pR2(probit_model)
```

```
## fitting null model for pseudo-r2
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -3.184641e+04 -4.453146e+04  2.537010e+04  2.848560e-01  3.257166e-01
##          r2CU
##  4.346909e-01
```

```
pR2(logit_model)
```

```
## fitting null model for pseudo-r2
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -3.185403e+04 -4.453146e+04  2.535486e+04  2.846848e-01  3.255569e-01
##          r2CU
##  4.344778e-01
```

```
## Sensitivity Analysis / Confusion Matrix

# Predict probabilities
lpm_pred <- ifelse(predict(lpm_model) > 0.5, 1, 0)
probit_pred <- ifelse(predict(probit_model, type = "response") > 0.5, 1, 0)
logit_pred <- ifelse(predict(logit_model, type = "response") > 0.5, 1, 0)

# Actual (cleaned to match)
actual <- data_clean$Churn

# Confusion Matrices
table(LPM = lpm_pred, Actual = actual)
```

```
##    Actual
## LPM     0     1
##   0 25168  6041
##   1  8713 24452
```

```
table(Probit = probit_pred, Actual = actual)
```

```
##       Actual
## Probit     0     1
##      0 25705  6904
##      1  8176 23589
```

```
table(Logit = logit_pred, Actual = actual)
```

```
##      Actual
## Logit     0     1
##     0 25637  6756
##     1  8244 23737
```

```
# Accuracy
mean(lpm_pred == actual)
```

```
## [1] 0.7708081
```

```
mean(probit_pred == actual)
```

```
## [1] 0.7657439
```

```
mean(logit_pred == actual)
```

```
## [1] 0.7669867
```

Comments:

AIC / BIC ~ Based on the AIC and BIC, the probit model performs the best out of the three models. It has the lowest values (AIC = 452,788.5 & 452,865.5), suggesting it achieves the best possible trade-off between model fit and complexity. The logit model is a close second with the second-lowest values (AIC = 453,972.8 & BIC = 454,049.8), while the linear probability model performs the worst (AIC = 485,202.9 & BIC = 485,290.8), indicating it is not a suitable choice for modeling this binary outcome. Thus, the AIC and BIC scores both suggest that the probit model is the best model out of the three analyzed.

Pseudo R-Squared ~ After calculating the McFadden's Pseudo R-Squared for both the probit and logit models, we can see that their Pseudo R-Squareds are very, very close. The probit model has a Pseudo R-Squared of about 0.2493, while the logit model has a Pseudo R-Squared of about 0.2474. Though these values indicate that the probit model offers a slightly better fit than the logit model (because it has a higher Pseudo R-Squared), the difference is minimal. Nevertheless, this supports the earlier findings from AIC and BIC that the probit model is the best-performing option overall.

Sensitivity Analysis / Confusion Matrix ~ While the Linear Probability Model shows slightly higher accuracy (73.3% compared to probit's 72.9% and logit's 73.0%), this comes with a trade-off: it produces significantly more false positives (61,962) compared to the logit (58,786) and probit (57,622) models. The probit model, on the other hand, seems to have fewer false positives, but more false negatives (61,961) compared to the LPM (55,786) and logit (60,121) models. This signifies that between the three models models, logit offers the best balance between sensitivity and specificity, achieving higher accuracy than probit with more favorable confusion matrix trade-offs than the LPM. Thus, our sensitivity analysis / confusion matrix analysis seems to favor the logit model.

Final Recommendation: Overall, we recommend using the probit model as the preferred specification among the three models analyzed. The probit model outperformed the others in two out of the three major statistical diagnostics we explored. First, it achieved the lowest AIC and BIC values, indicating the best trade-off between goodness of fit and model complexity. Second, it had the highest McFadden's pseudo R^2, reflecting a strong ability to explain variation in the binary outcome variable. While the confusion matrix analysis and classification accuracy slightly favor the Linear Probability Model, this model suffers from key drawbacks: it lacks bounded predicted probabilities (i.e., predictions can fall outside the 0–1 range), and it produces the highest number of false positives. Between the probit and logit models, the logit model achieved a slightly higher classification accuracy and a more balanced confusion matrix. However, these differences were pretty negligible, and the probit model remained statistically superior in terms of overall model fit. Thus, taken altogether, we recommend the probit model as the best choice for modeling Churn, due to its combination of strong theoretical justification, superior model fit, and competitive predictive performance.