Final Project

2025-03-02

1. Data Preparing

```r
# Load required libraries
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

## Loading required package: TTR

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

```r
library(forecast)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(fracdiff)
library(vars)
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## Loading required package: strucchange
## Loading required package: sandwich
##
## Attaching package: 'strucchange'
##
## The following object is masked from 'package:stringr':
##
##     boundary
##
## Loading required package: urca
## Loading required package: lmtest
```

```r
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
##
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```r
library(dplyr)
library(lmtest)
library(tseries)
library(FinTS)
```

```
##
## Attaching package: 'FinTS'
##
## The following object is masked from 'package:forecast':
##
##     Acf
```

```r
library(rugarch)
```

```
## Loading required package: parallel
##
## Attaching package: 'rugarch'
##
## The following object is masked from 'package:purrr':
```

```
##
##     reduce
##
## The following object is masked from 'package:stats':
##
##     sigma
```

```
cpi_data <- read.csv("CPIAUCNS.csv")
cpi_data$DATE <- as.Date(cpi_data$observation_date)
cpi_ts <- ts(cpi_data$CPIAUCNS, start = c(year(min(cpi_data$DATE)), month(min(cpi_data$DATE))), frequen

# Download data
getSymbols(c("DFF", "FEDFUNDS", "UNRATE", "GDP", "M2SL"), src = "FRED")
```

```
## [1] "DFF"      "FEDFUNDS" "UNRATE"   "GDP"      "M2SL"
```

```
# Convert to time series format, extracting data after 2000-01-01
fedfunds_ts <- window(ts(FEDFUNDS, start = c(1954, 7), frequency = 12), start = c(1999, 12), end = c(20
unrate_ts <- window(ts(UNRATE, start = c(1948, 1), frequency = 12), start = c(1999, 12), end = c(2025, 
# gdp_ts <- window(ts(GDP, start = c(1947, 1), frequency = 4), start = c(2000, 1))  # Quarterly
m2_ts <- window(ts(M2SL, start = c(1959, 1), frequency = 12), start = c(1999, 12), end = c(2025, 1))  #

# Merge all time series into a dataset
dataset <- data.frame(
  Date = seq(as.Date("2000-01-01"), by = "month", length.out = length(cpi_ts)),
  CPI = as.numeric(cpi_ts),
  FEDFUNDS = as.numeric(fedfunds_ts[-1]),
  UNRATE = as.numeric(unrate_ts[-1]),
  M2 = as.numeric(m2_ts[-1])
)

# Create lagged versions of external regressors (1-month lag)
dataset_lag <- data.frame(
  Date = seq(as.Date("2000-01-01"), by = "month", length.out = length(cpi_ts)),
  CPI = as.numeric(cpi_ts),
  FEDFUNDS = as.numeric(fedfunds_ts[-length(fedfunds_ts)]),
  UNRATE = as.numeric(unrate_ts[-length(unrate_ts)]),
  M2 = as.numeric(m2_ts[-length(m2_ts)])
)
```

```
summary(dataset)
```

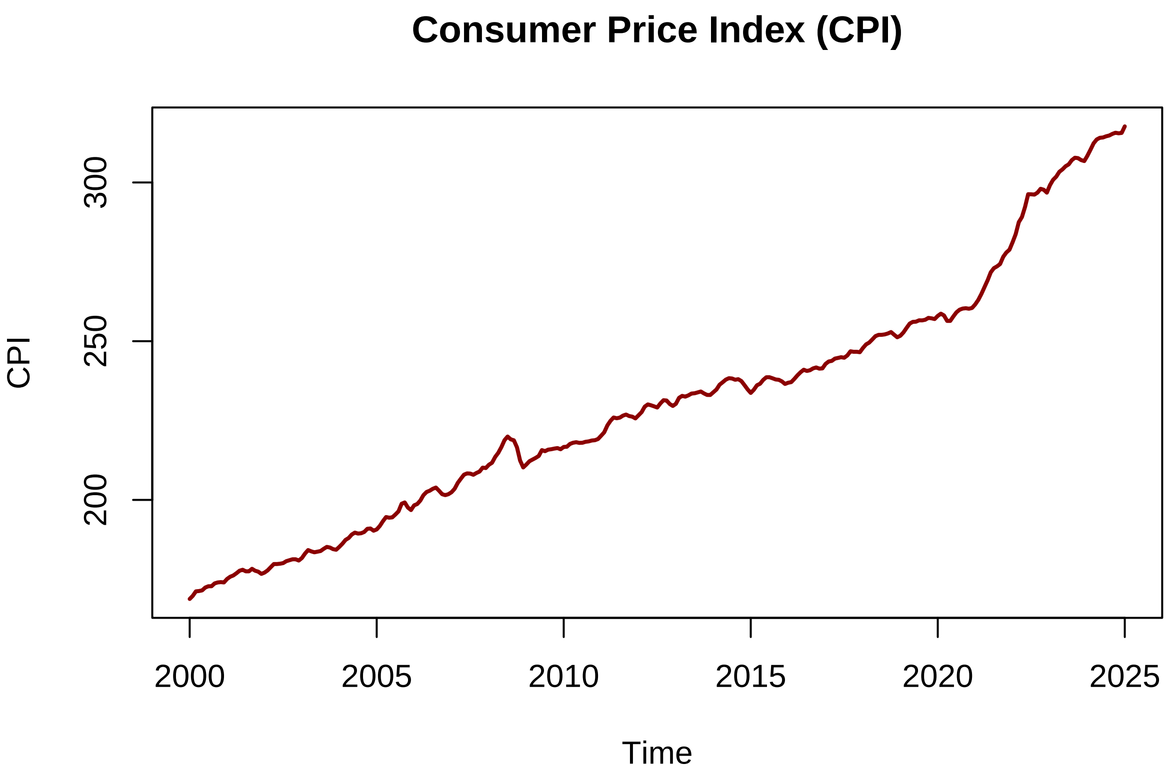
```
##       Date                 CPI           FEDFUNDS         UNRATE     
##  Min.   :2000-01-01   Min.   :168.8   Min.   :0.050   Min.   : 3.40  
##  1st Qu.:2006-04-01   1st Qu.:201.5   1st Qu.:0.140   1st Qu.: 4.20  
##  Median :2012-07-01   Median :229.8   Median :1.220   Median : 5.10  
##  Mean   :2012-07-01   Mean   :230.2   Mean   :1.928   Mean   : 5.69  
##  3rd Qu.:2018-10-01   3rd Qu.:252.0   3rd Qu.:3.620   3rd Qu.: 6.40  
##  Max.   :2025-01-01   Max.   :317.7   Max.   :6.540   Max.   :14.80  
##        M2       
##  Min.   : 4668  
##  1st Qu.: 6807  
```

```
##   Median :10066
##   Mean   :11399
##   3rd Qu.:14241
##   Max.   :21750
```

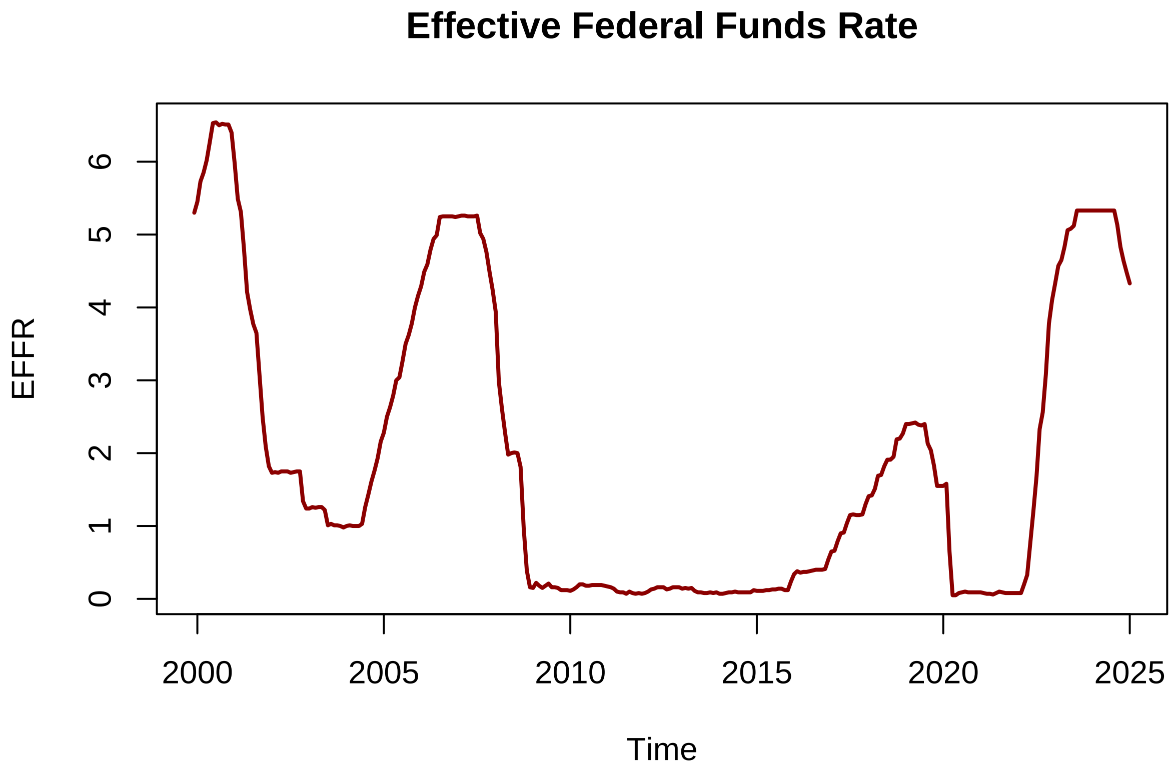
2. EDA

(1) Time Series Plots

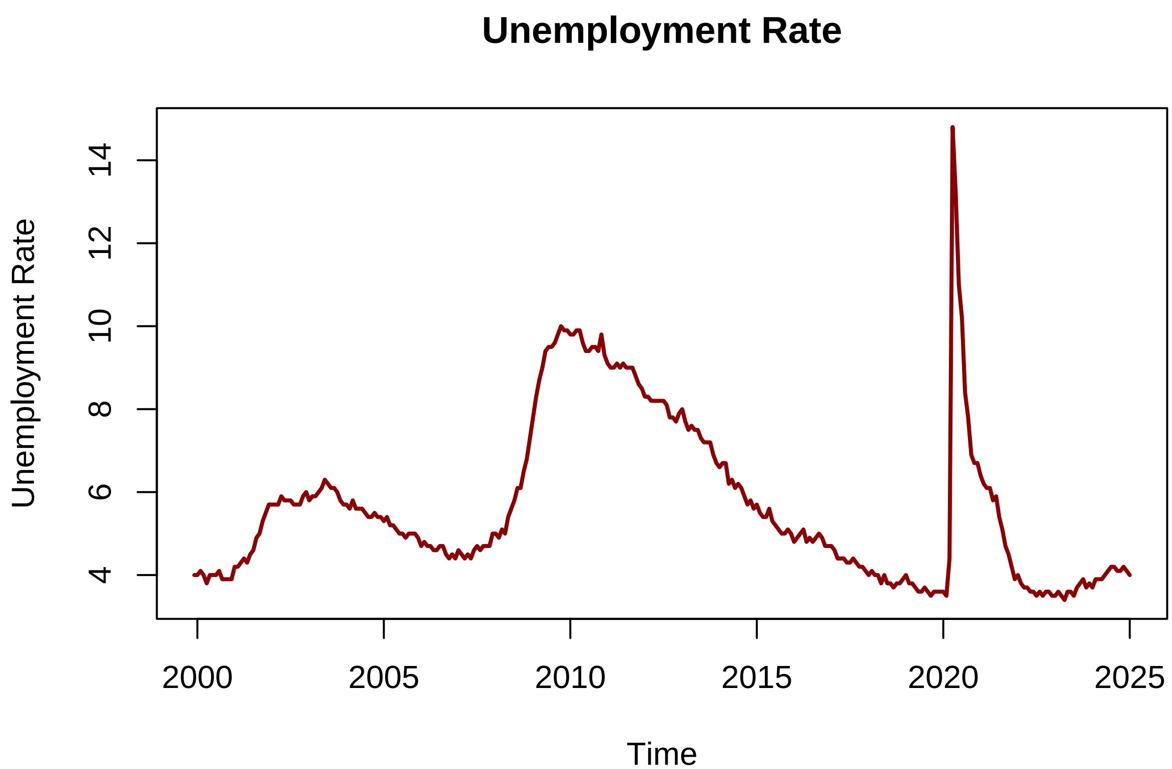
```r
# Plot time series trends
plot(cpi_ts, main = "Consumer Price Index (CPI)",
     ylab = "CPI", xlab = "Time", col = "darkred", lwd = 2)
```



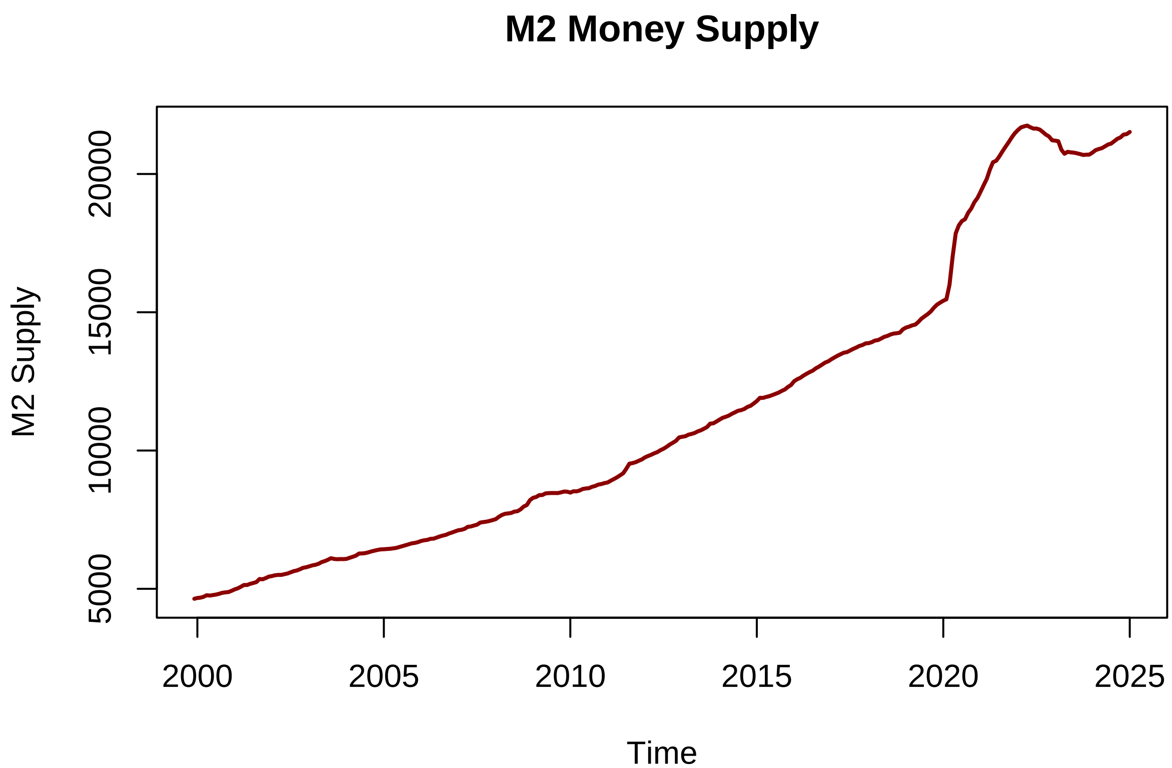

```r
plot(fedfunds_ts, main = "Effective Federal Funds Rate",
     ylab = "EFFR", xlab = "Time", col = "darkred", lwd = 2)
```




```r
plot(unrate_ts, main = "Unemployment Rate",
     ylab = "Unemployment Rate", xlab = "Time", col = "darkred", lwd = 2)
```

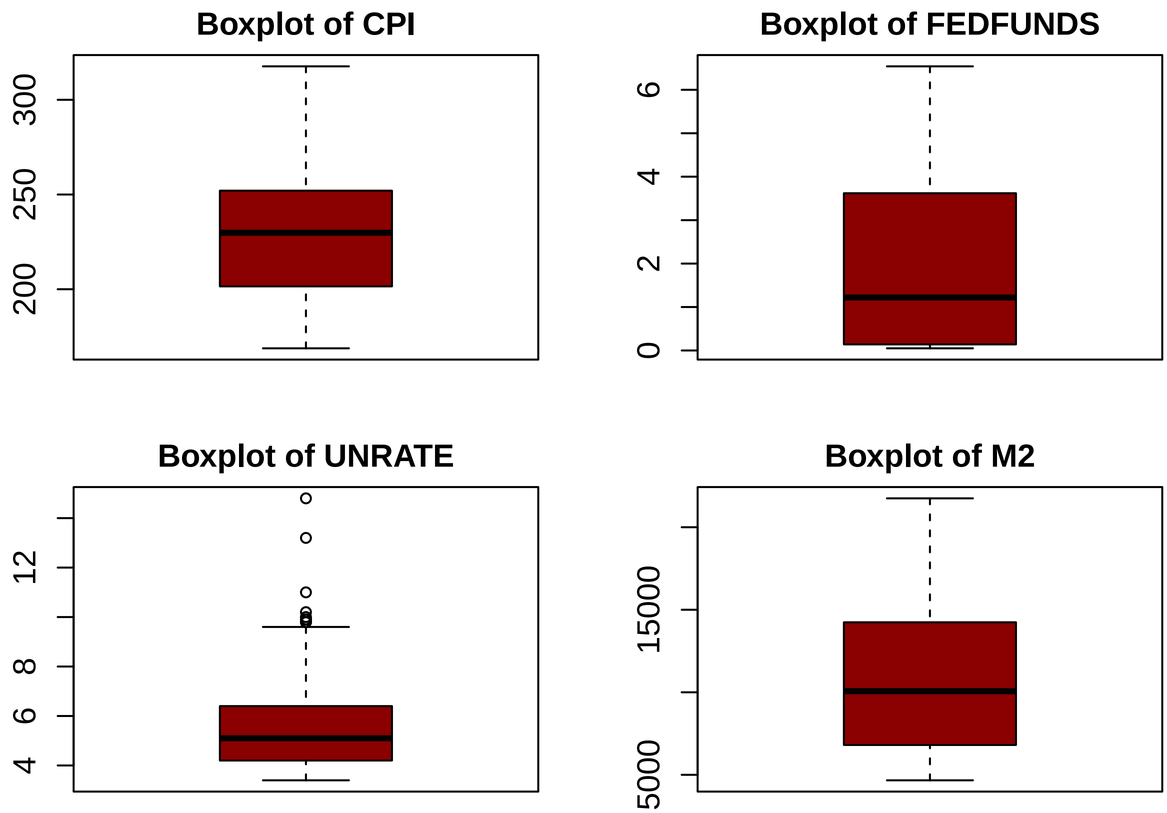


```r
plot(m2_ts, main = "M2 Money Supply",
     ylab = "M2 Supply", xlab = "Time", col = "darkred", lwd = 2)
```



(2) Boxplots

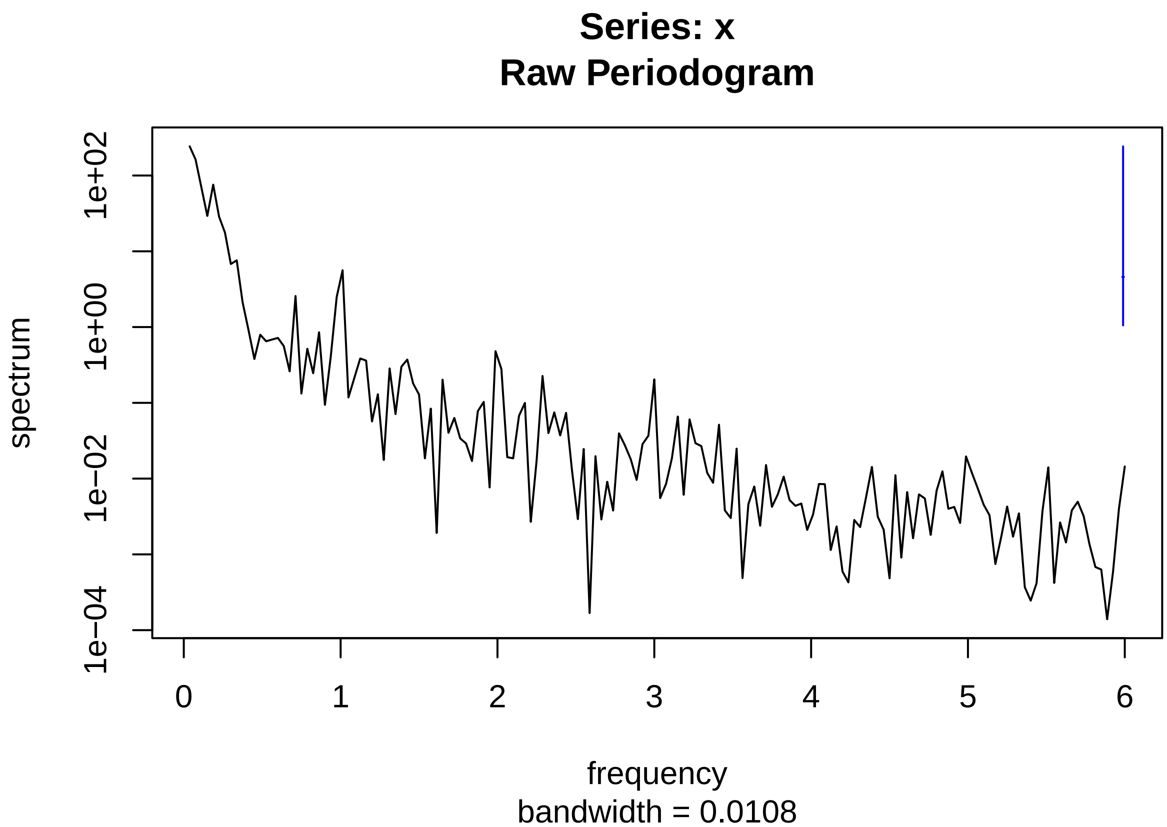
```r
# Boxplots for each feature
par(mfrow = c(2,2), mar = c(2,3,2,2), cex.lab = 1.2, cex.axis = 1.2, cex.main = 1.2)  # Arrange plots i
boxplot(dataset$CPI, main = "Boxplot of CPI", col = "darkred")
boxplot(dataset$FEDFUNDS, main = "Boxplot of FEDFUNDS", col = "darkred")
boxplot(dataset$UNRATE, main = "Boxplot of UNRATE", col = "darkred")
boxplot(dataset$M2, main = "Boxplot of M2", col = "darkred")
```




```r
par(mfrow = c(1,1))  # Reset layout
```

(3) Spectrum Analysis

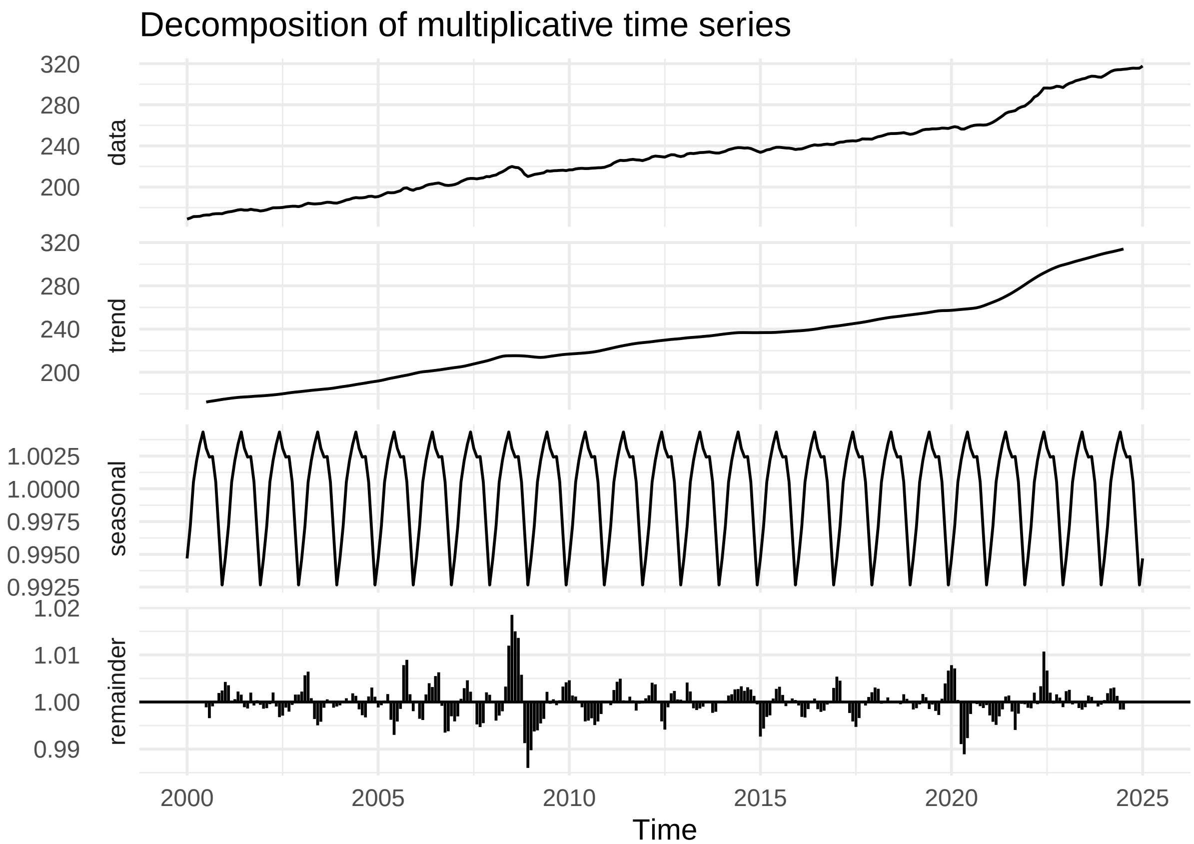
```r
# spectrum analysis
spectrum(cpi_ts)
```



(4) Decomposition

```
# Decompose the time series into trend, seasonal, and random components
decomp <- decompose(cpi_ts, type = "multiplicative")

# Plot the decomposition
autoplot(decomp)+theme_minimal()
```



(5) Check Stationarity

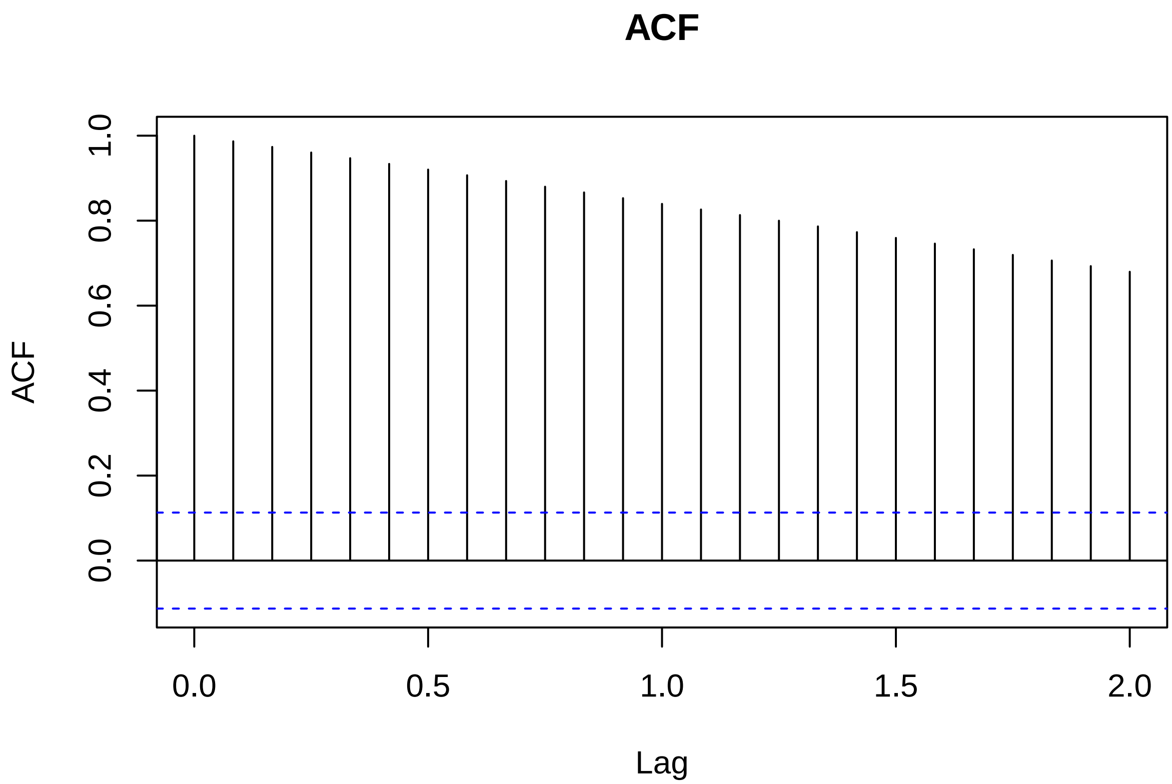
```r
library(tseries)
# ADF Test for stationarity
adf_test <- adf.test(cpi_ts)
```

```
## Warning in adf.test(cpi_ts): p-value greater than printed p-value
```

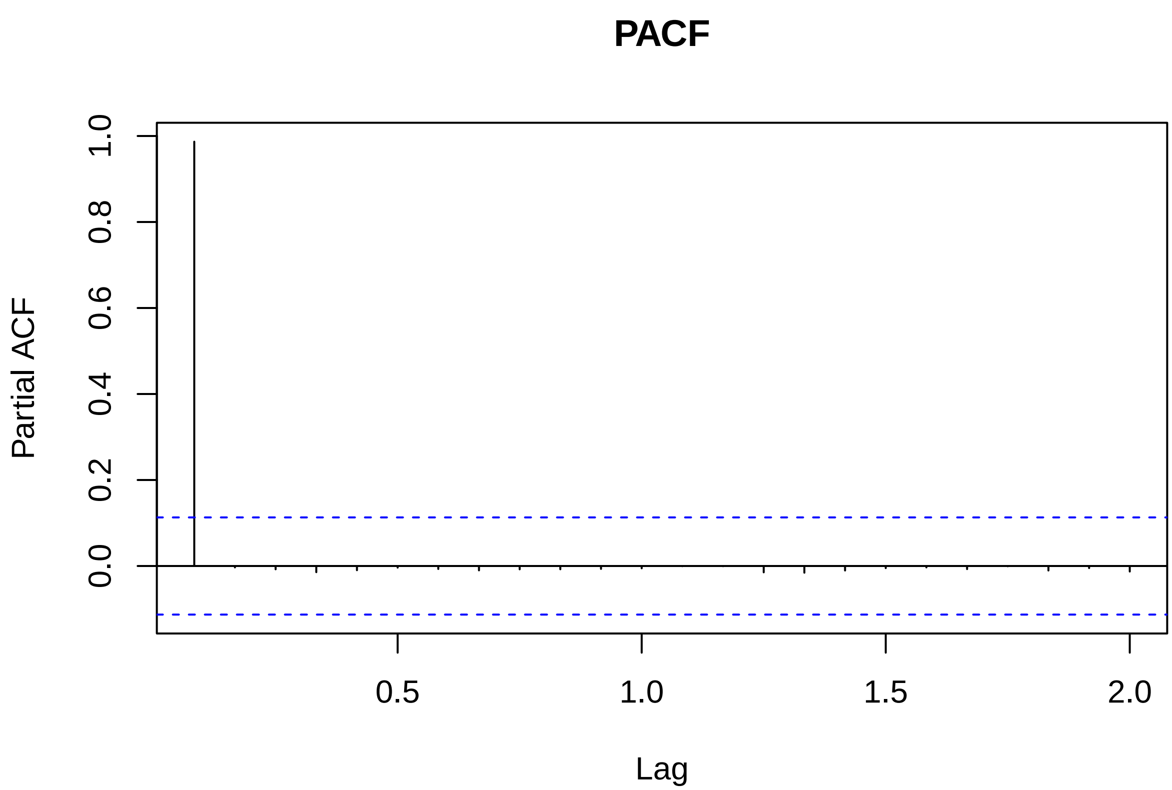
```r
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cpi_ts
## Dickey-Fuller = 0.59021, Lag order = 6, p-value = 0.99
## alternative hypothesis: stationary
```

```r
# Plot ACF and PACF
acf(cpi_ts, main = "ACF")
```




```r
pacf(cpi_ts, main = "PACF")
```



The data is non-stationary because P-value = 0.99 that significantly greater than 0.05.

3. Models

(1) Univariate

```r
cpi_ts_train <- window(cpi_ts, start = c(2000, 1), end = c(2023, 12))
cpi_ts_test <- window(cpi_ts, start = c(2024, 1))

evaluate_performance <- function(actual, predicted) {
  mse_value <- mse(actual, predicted)
  rmse_value <- rmse(actual, predicted)
  mae_value <- mae(actual, predicted)
  rss <- sum((actual - predicted)^2)  # Residual Sum of Squares
  tss <- sum((actual - mean(actual))^2)  # Total Sum of Squares
  r_squared <- 1 - (rss/tss)
  results <- list(
    MSE = mse_value,
    RMSE = rmse_value,
    MAE = mae_value,
    R_squared = r_squared
  )
  return(results)
}
```

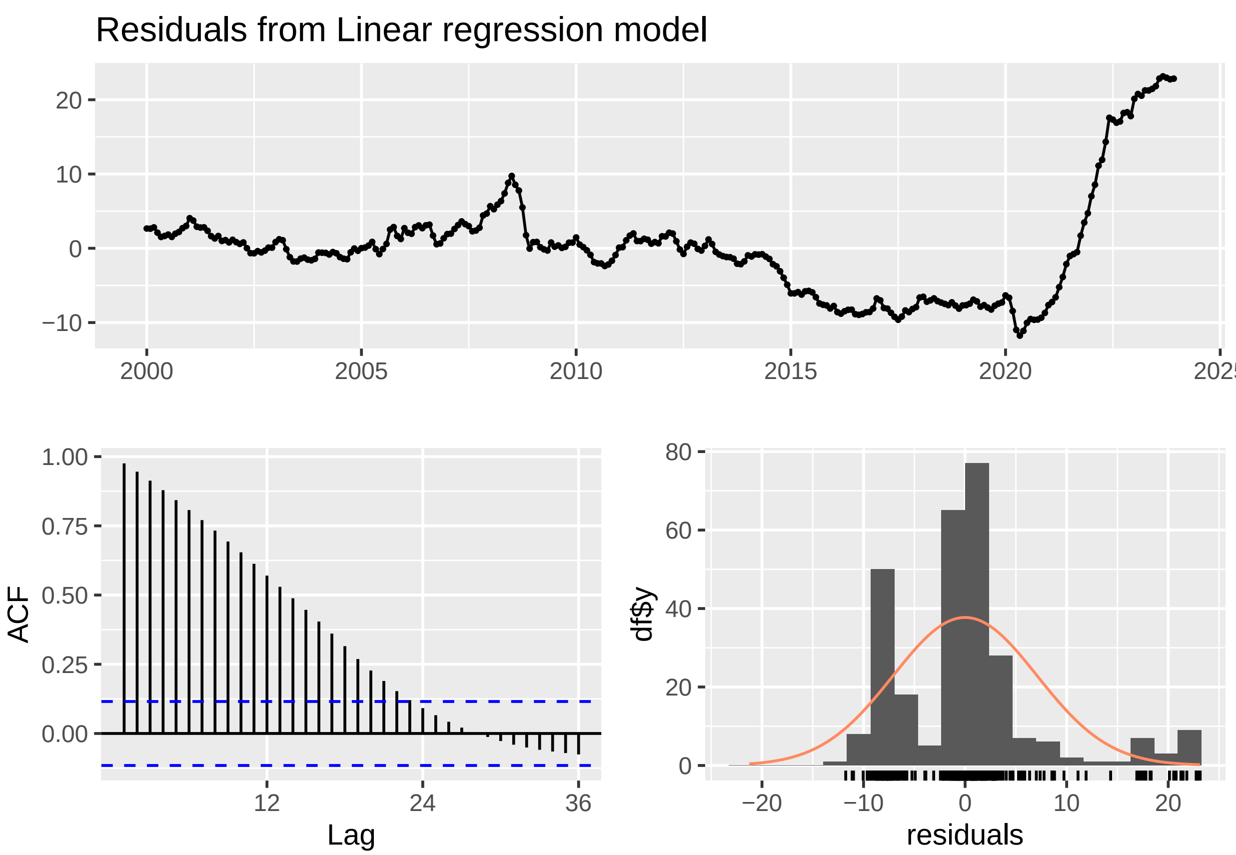
a. Regression Model

```r
regression <- tslm(cpi_ts_train ~ trend + season)
summary(regression)
```

```
##
## Call:
## tslm(formula = cpi_ts_train ~ trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.7568  -5.9115  -0.0911   1.9671  23.1474
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.657e+02  1.641e+00 100.968   <2e-16 ***
## trend       4.091e-01  5.136e-03  79.642   <2e-16 ***
## season2     6.107e-01  2.090e+00   0.292    0.770
## season3     1.430e+00  2.090e+00   0.684    0.494
## season4     1.832e+00  2.090e+00   0.876    0.382
## season5     2.196e+00  2.090e+00   1.051    0.294
## season6     2.555e+00  2.090e+00   1.222    0.223
## season7     2.362e+00  2.090e+00   1.130    0.260
## season8     2.264e+00  2.091e+00   1.083    0.280
## season9     2.324e+00  2.091e+00   1.111    0.267
## season10    1.981e+00  2.091e+00   0.948    0.344
## season11    1.165e+00  2.091e+00   0.557    0.578
```

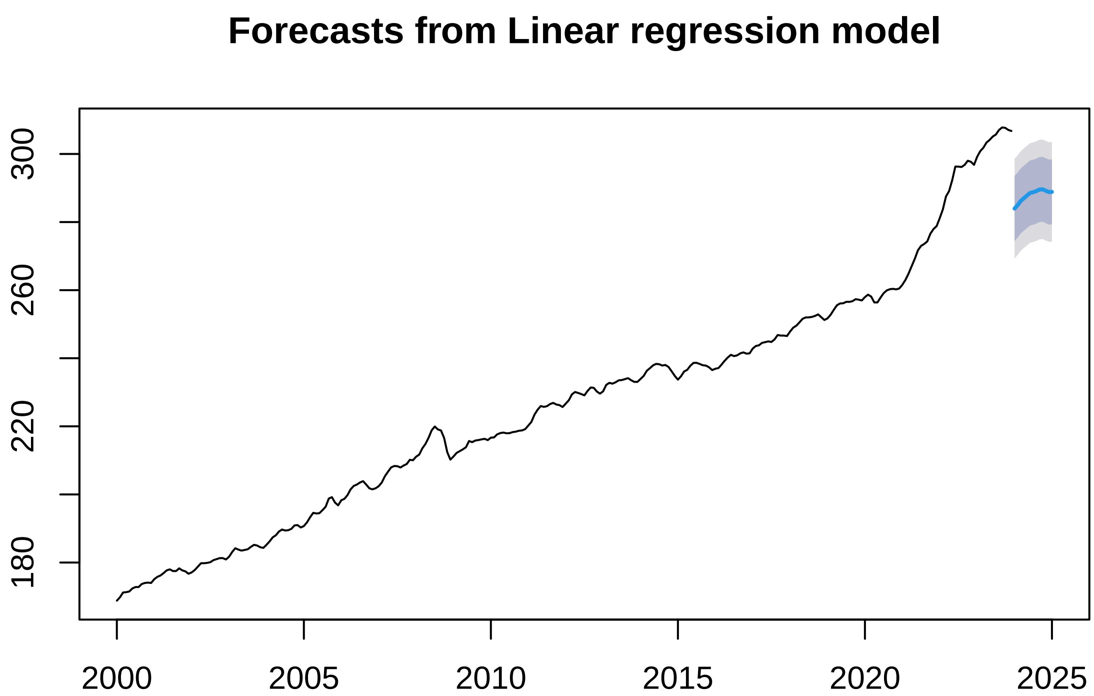
```
## season12    3.589e-01  2.091e+00   0.172    0.864
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.241 on 275 degrees of freedom
## Multiple R-squared:  0.9586, Adjusted R-squared:  0.9567
## F-statistic:   530 on 12 and 275 DF,  p-value: < 2.2e-16
```

```
checkresiduals(regression)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals from Linear regression model
## LM test = 285.97, df = 24, p-value < 2.2e-16
```

```
regression_forecast <- forecast(regression, h = 13)
plot(regression_forecast)
```




```
evaluate_performance(cpi_ts_test, regression_forecast$mean)
```

```
## $MSE
## [1] 683.1153
##
## $RMSE
## [1] 26.13648
##
## $MAE
## [1] 26.11893
##
## $R_squared
## [1] -123.4503
```

b. ARIMA Model

```
lambda <- BoxCox.lambda(cpi_ts_train)
print(lambda)
```

```
## [1] -0.9999242
```

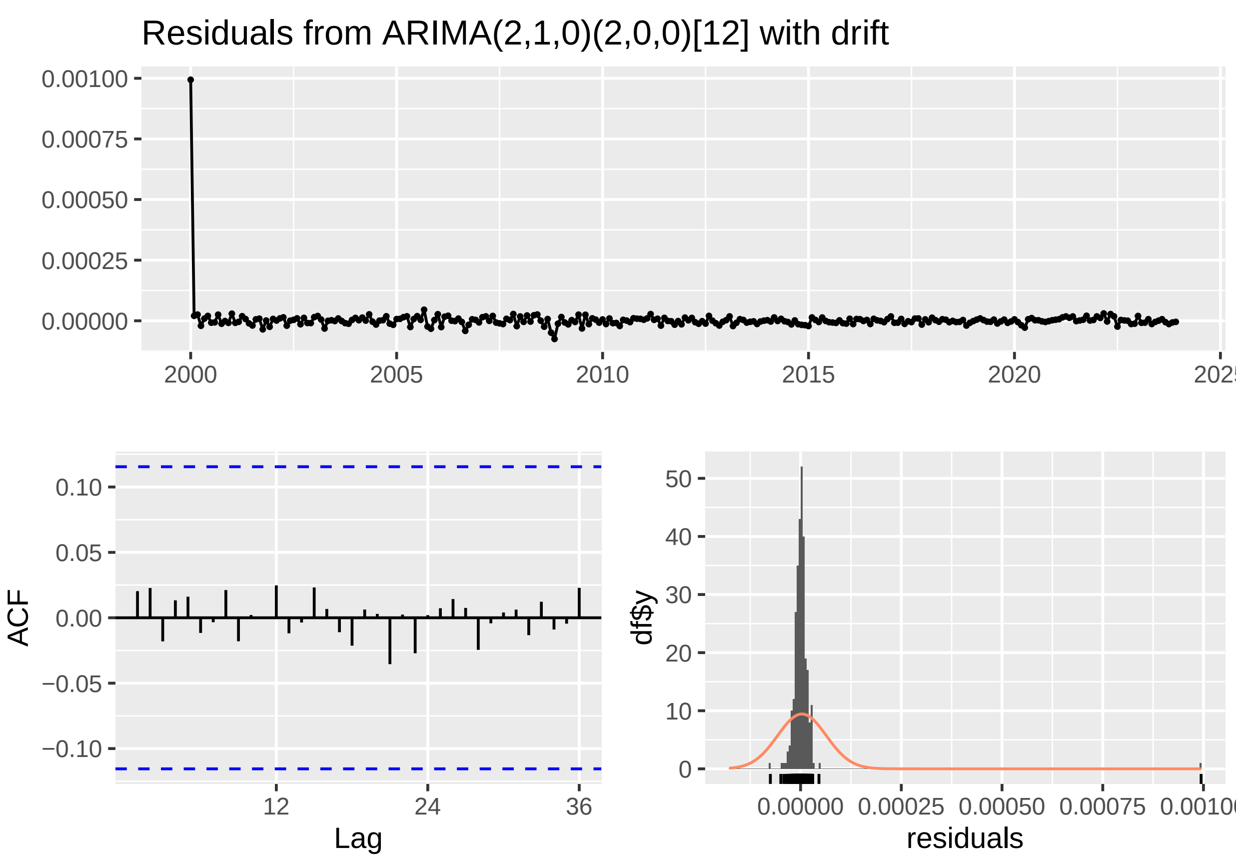
```
shapiro.test(cpi_ts_train)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cpi_ts_train
## W = 0.9639, p-value = 1.335e-06
```

```r
arima <- auto.arima(cpi_ts_train, lambda = lambda)
summary(arima)
```

```
## Series: cpi_ts_train
## ARIMA(2,1,0)(2,0,0)[12] with drift
## Box Cox transformation: lambda= -0.9999242
##
## Coefficients:
##          ar1      ar2    sar1    sar2  drift
##       0.5520  -0.2268  0.1884  0.2045  0e+00
## s.e.  0.0587   0.0581  0.0617  0.0709  1e-04
##
## sigma^2 = 3.709e-09:  log likelihood = 2795.75
## AIC=-5579.5   AICc=-5579.2   BIC=-5557.54
##
## Training set error measures:
##                      ME     RMSE       MAE        MPE      MAPE      MASE
## Training set 0.07664625 1.596256 0.6150133 0.04659053 0.2842446 0.1040759
##                    ACF1
## Training set 0.03222512
```

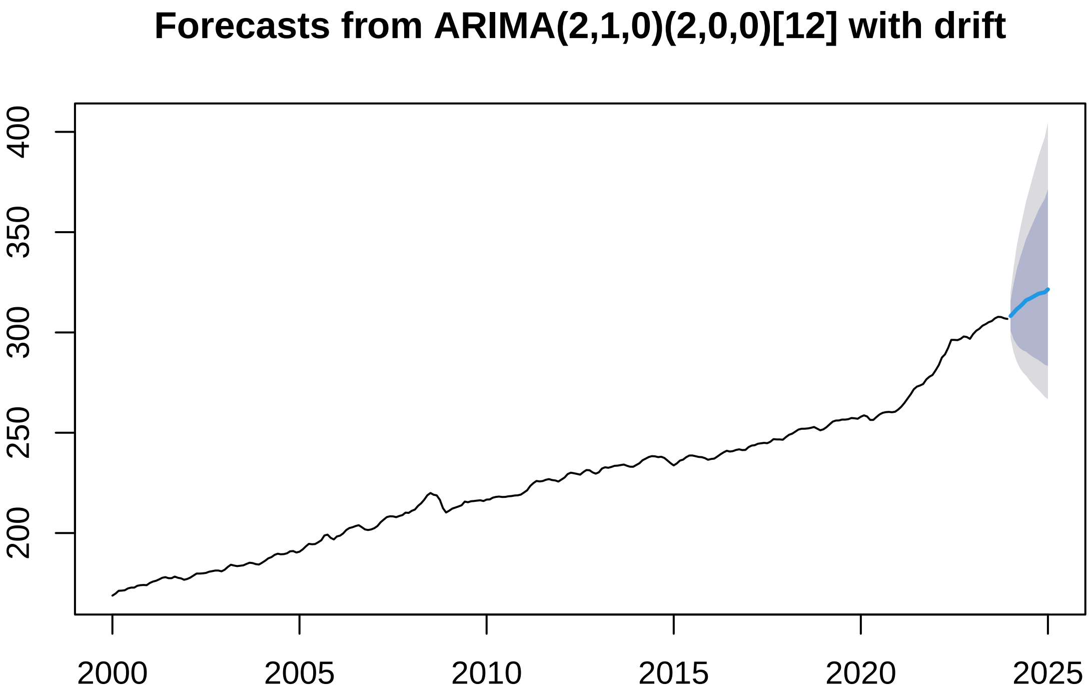
```r
checkresiduals(arima)
```




```
##
##  Ljung-Box test
```

```
##
## data:  Residuals from ARIMA(2,1,0)(2,0,0)[12] with drift
## Q* = 2.0032, df = 20, p-value = 1
##
## Model df: 4.   Total lags used: 24
```

```
arima_forecast <- forecast(arima, h = 13)
plot(arima_forecast)
```




```
evaluate_performance(cpi_ts_test, arima_forecast$mean)
```

```
## $MSE
## [1] 7.113968
##
## $RMSE
## [1] 2.667202
##
## $MAE
## [1] 2.194474
##
## $R_squared
## [1] -0.2960262
```

c. GARCH Model

```
residuals_arima <- residuals(arima)
# Perform ARCH-LM test (using 5 lags)
arch_test <- ArchTest(residuals_arima, lags = 5)
print(arch_test)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  residuals_arima
## Chi-squared = 16.65, df = 5, p-value = 0.005214
```

```r
# Define GARCH(1,1) Model for CPI
garch_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1,1)),
                         mean.model = list(armaOrder = c(2,2), include.mean = TRUE),
                         distribution.model = "norm")

# Fit GARCH model to CPI
garch_model <- ugarchfit(spec = garch_spec, data = cpi_ts_train)

# View results
summary(garch_model)
```

```
##    Length     Class      Mode
##         1 uGARCHfit        S4
```

```r
garch_forecast <- ugarchforecast(garch_model, n.ahead = 13)
evaluate_performance(cpi_ts_test, garch_forecast@forecast$seriesFor)
```

```
## $MSE
## [1] 4.751166
##
## $RMSE
## [1] 2.179717
##
## $MAE
## [1] 1.872261
##
## $R_squared
## [1] 0.1344301
```

(2) Multivariate

```r
# Split data into training (2000-01 to 2023-12) and testing (2024-01 onward)

train_end <- which(dataset$Date == as.Date("2023-12-01"))
dataset_train <- dataset[1:train_end, ]
dataset_test <- dataset[(train_end + 1):nrow(dataset), ]

dataset_lag_train <- dataset_lag[1:train_end, ]
dataset_lag_test <- dataset_lag[(train_end + 1):nrow(dataset), ]
```

```r
cor(dataset_train$CPI, dataset_train$FEDFUNDS, use = "complete.obs")
```

```
## [1] -0.2372331
```

```r
cor(dataset_train$CPI, dataset_train$UNRATE, use = "complete.obs")
```

```
## [1] -0.1512589
```

```r
cor(dataset_train$CPI, dataset_train$M2, use = "complete.obs")
```

```
## [1] 0.9661038
```

```r
granger_test <- grangertest(CPI ~ FEDFUNDS, order = 2, data = dataset_train)
print("FEDFUNDS:")
```

```
## [1] "FEDFUNDS:"
```

```r
print(granger_test)
```

```
## Granger causality test
##
## Model 1: CPI ~ Lags(CPI, 1:2) + Lags(FEDFUNDS, 1:2)
## Model 2: CPI ~ Lags(CPI, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    281
## 2    283 -2 0.9272 0.3969
```

```r
granger_test <- grangertest(CPI ~ UNRATE, order = 2, data = dataset_train)
print("UNRATE:")
```

```
## [1] "UNRATE:"
```

```r
print(granger_test)
```

```
## Granger causality test
##
## Model 1: CPI ~ Lags(CPI, 1:2) + Lags(UNRATE, 1:2)
## Model 2: CPI ~ Lags(CPI, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    281
## 2    283 -2 0.0448 0.9561
```

```r
granger_test <- grangertest(CPI ~ M2, order = 2, data = dataset_train)
print("M2:")
```

```
## [1] "M2:"
```

```r
print(granger_test)
```

```
## Granger causality test
##
## Model 1: CPI ~ Lags(CPI, 1:2) + Lags(M2, 1:2)
## Model 2: CPI ~ Lags(CPI, 1:2)
##   Res.Df Df      F    Pr(>F)
## 1    281
## 2    283 -2 8.3245 0.0003074 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

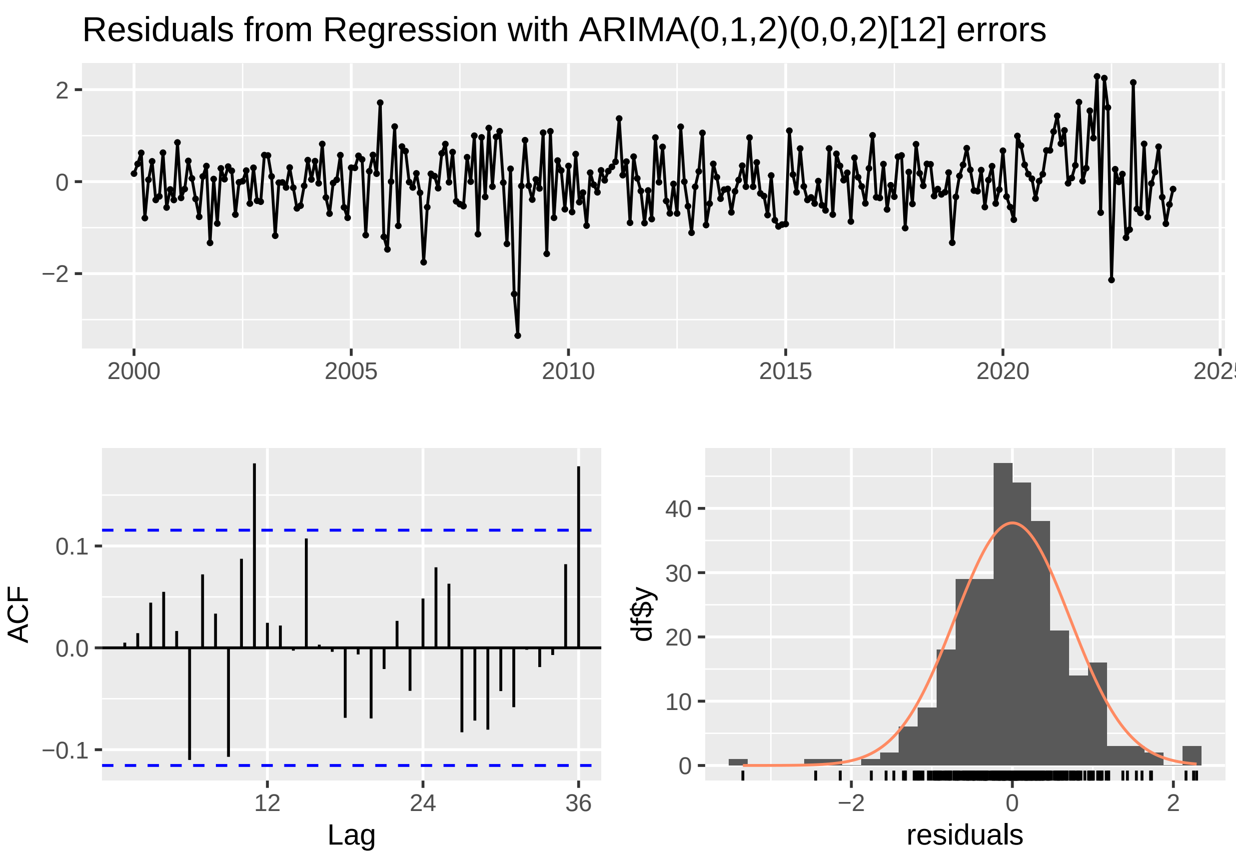
Reg-ARMA

```r
# Regression with ARMA Errors
reg_arma <- auto.arima(cpi_ts_train,
                              xreg = as.matrix(dataset_train[, c("M2")]),
                              seasonal = TRUE)

summary(reg_arma)
```

```
## Series: cpi_ts_train
## Regression with ARIMA(0,1,2)(0,0,2)[12] errors
##
## Coefficients:
##          ma1     ma2    sma1    sma2   drift     xreg
##       0.6040  0.1331  0.1931  0.1378  0.5519  -0.0013
## s.e.  0.0592  0.0621  0.0628  0.0578  0.1014   0.0006
##
## sigma^2 = 0.5222:  log likelihood = -311.54
## AIC=637.08   AICc=637.48   BIC=662.69
##
## Training set error measures:
##                        ME      RMSE       MAE          MPE      MAPE       MASE
## Training set 0.0007647766 0.7138165 0.5302602 -0.004068395 0.2349103 0.08973349
##                     ACF1
## Training set 0.005092052
```

```r
# Check residuals of the model
checkresiduals(reg_arma)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,2)(0,0,2)[12] errors
## Q* = 31.194, df = 20, p-value = 0.05268
##
## Model df: 4.   Total lags used: 24
```

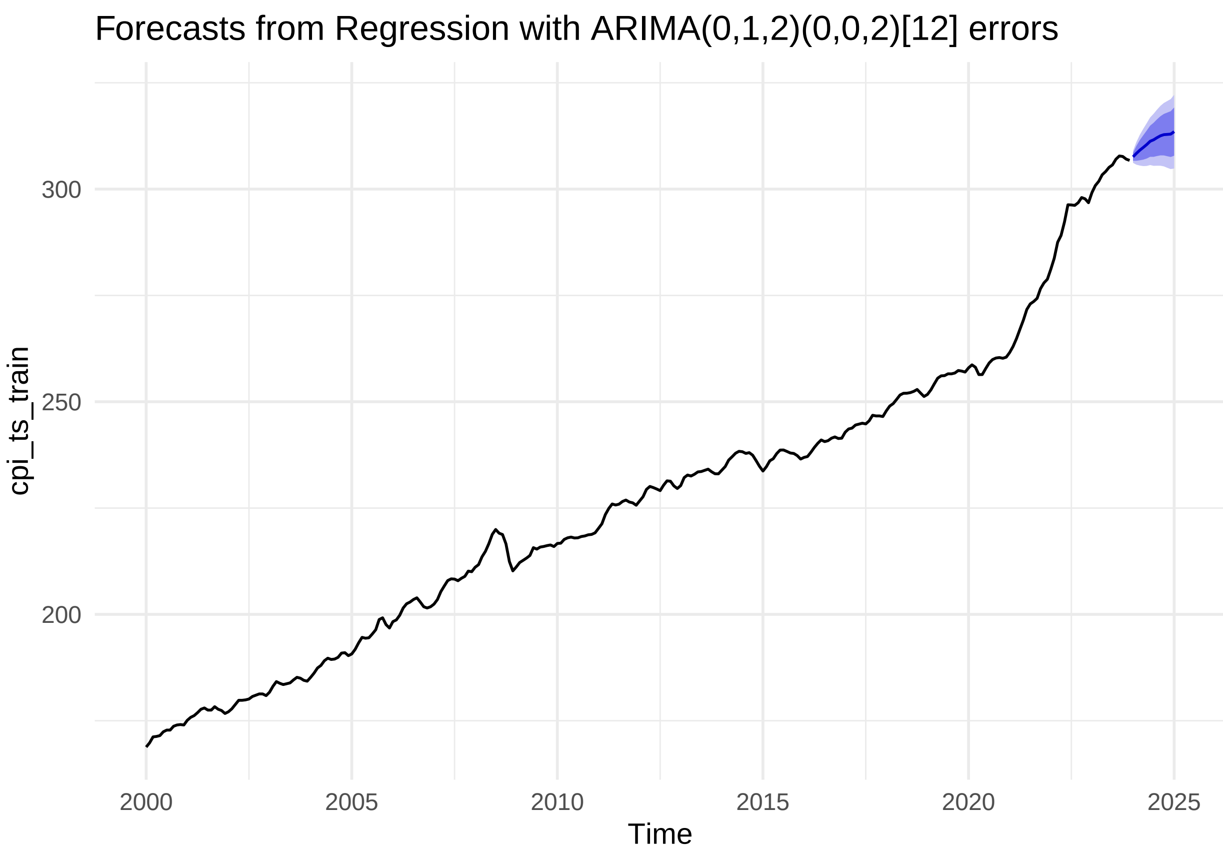
```
# Make predictions
arima_m2 <- auto.arima(dataset_train[, c("M2")], lambda = "auto")
m2_forecast <- forecast(arima_m2, h = 13)
```

```
reg_arma_forecast <- forecast(reg_arma,
                              xreg = as.matrix(m2_forecast$mean),
                              h = nrow(dataset_lag_test))
evaluate_performance(cpi_ts_test, reg_arma_forecast$mean)
```

```
## $MSE
## [1] 8.691107
##
## $RMSE
## [1] 2.948068
##
## $MAE
## [1] 2.838863
```

```
##
## $R_squared
## [1] -0.5833501
```

```r
autoplot(reg_arma_forecast)+theme_minimal()
```

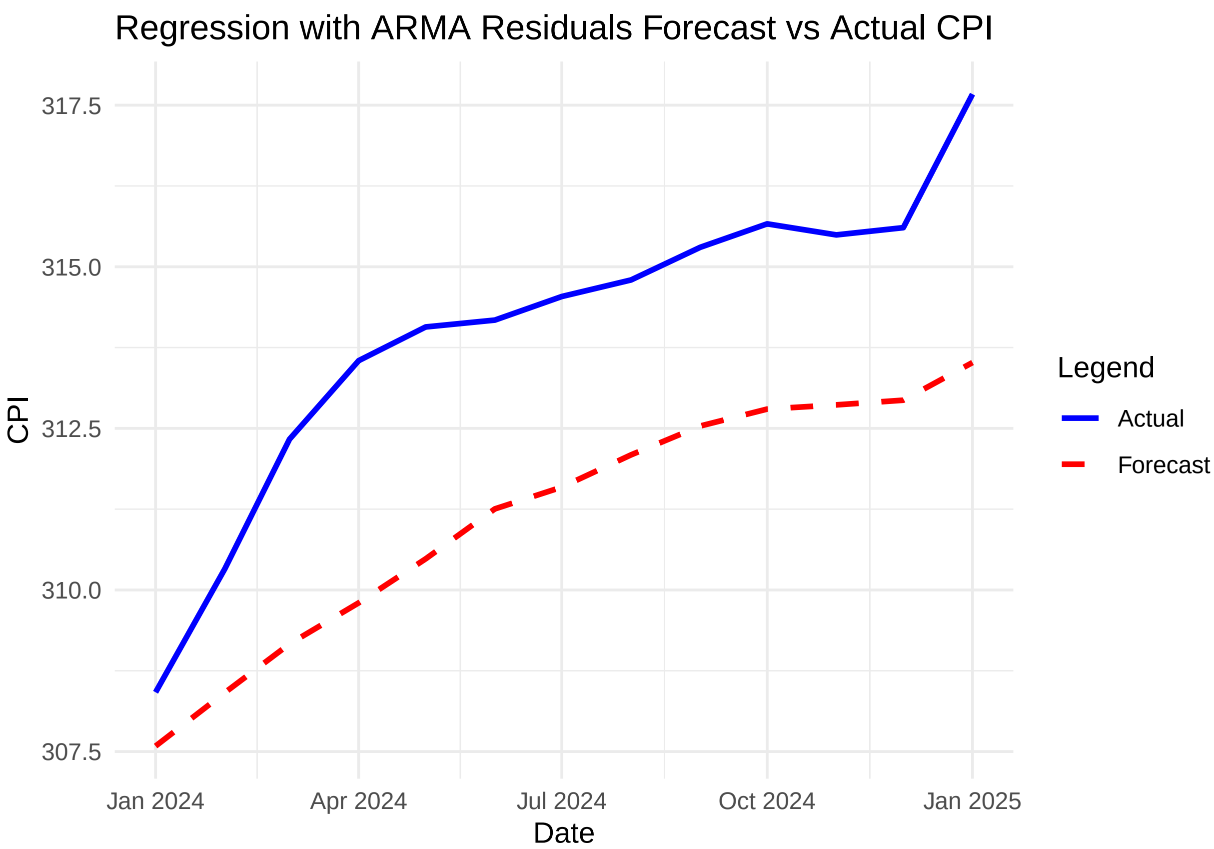



```r
# Create a dataframe for plotting
plot_data <- data.frame(
  Date = dataset_test$Date,
  Actual = cpi_ts_test,
  Predicted = reg_arma_forecast$mean
)

# Plot forecast vs actual
ggplot(plot_data, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual"), size = 1) +
  geom_line(aes(y = Predicted, color = "Forecast"), size = 1, linetype = "dashed") +
  labs(title = "Regression with ARMA Residuals Forecast vs Actual CPI", x = "Date", y = "CPI") +
  scale_color_manual(name = "Legend", values = c("Actual" = "blue", "Forecast" = "red")) +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



VAR

```r
# Check stationarity of each variable
adf.test(diff(log(dataset_train$CPI)))
```

```
## Warning in adf.test(diff(log(dataset_train$CPI))): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(log(dataset_train$CPI))
## Dickey-Fuller = -6.4423, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(diff(log(dataset_train$M2)))
```

```
## Warning in adf.test(diff(log(dataset_train$M2))): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(log(dataset_train$M2))
## Dickey-Fuller = -4.3114, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# Log transformation and differencing
dataset_train$CPI_log_diff <- c(NA, diff(log(dataset_train$CPI)))
dataset_test$CPI_log_diff <- c(NA, diff(log(dataset_test$CPI)))

# Log transformation and differencing on M2
dataset_train$M2_log_diff <- c(NA, diff(log(dataset_train$M2)))
dataset_test$M2_log_diff <- c(NA, diff(log(dataset_test$M2)))
```

```r
var_train <- cbind(
  dataset_train$CPI_log_diff,
  dataset_train$M2_log_diff
)

var_train <- na.omit(var_train)
colnames(var_train) <- c("CPI_log_diff", "M2_log_diff")

var_model <- VAR(var_train, p = 10, type = "both", season = 12)
var_forecast <- predict(var_model, n.ahead = nrow(dataset_test))

# Extract CPI forecasts and convert back from log differences
cpi_forecast_log_diff <- var_forecast$fcst$CPI_log_diff[,1]  # Forecasted log differences
cpi_forecast <- exp(log(tail(dataset_train$CPI, 1)) + cumsum(cpi_forecast_log_diff))
print(evaluate_performance(dataset_test$CPI, cpi_forecast))
```

```
## $MSE
## [1] 2.068678
##
## $RMSE
## [1] 1.43829
##
## $MAE
## [1] 1.262145
##
## $R_squared
## [1] 0.6231271
```

```r
# Create dataframe for plotting
plot_data <- data.frame(
  Date = time(cpi_ts_test),
  Actual = dataset_test$CPI,
  Forecast = cpi_forecast
)

# Plot actual vs forecasted CPI
ggplot(plot_data, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual"), size = 1) +
```

```r
  geom_line(aes(y = Forecast, color = "Forecast"), size = 1, linetype = "dashed") +
  labs(title = "VAR Model Forecast vs Actual CPI", x = "Time", y = "CPI") +
  scale_color_manual(name = "Legend", values = c("Actual" = "blue", "Forecast" = "red")) +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```