Final Project : Forecasting Inflation : A Comprehensive Analysis - Turkey

ARZU ISIK TOPBAS

2023-12-19

Contents

Time Series Forecasting for Inflation Data: A Comprehensive Analysis - Turkey

Data

```
# Data
# inflation : https://data.oecd.org/price/inflation-cpi.htm
# gold_data :  https://data.worldbank.org/
# oil : https://data.worldbank.org/
# unemployment : https://www.tuik.gov.tr/Home/Index
# exchange rate : https://www.turkiye.gov.tr/doviz-kurlari -
# interest rate : https://fred.stlouisfed.org/series/INTDSRTRM193N
```

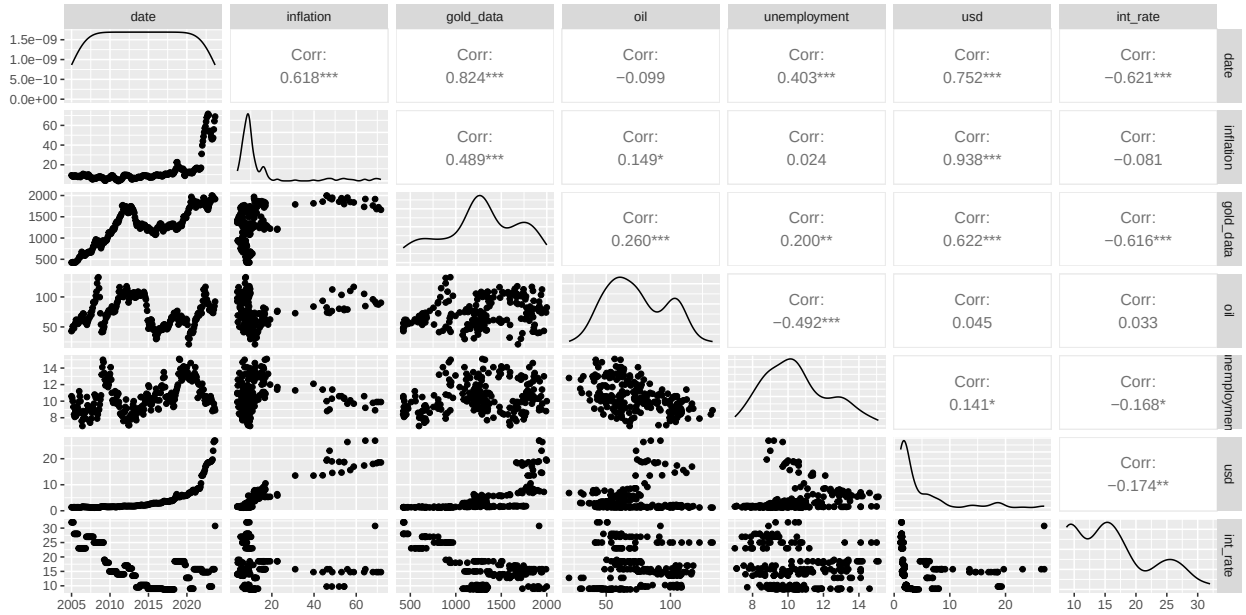
Comment: In this project, a comprehensive dataset has been compiled, drawing from various reputable sources to analyze key economic indicators. The inflation data, retrieved from the OECD, provides insights into the Consumer Price Index (CPI), serving as a crucial measure of general price level changes over time. The gold and oil datasets, sourced from the World Bank, shed light on the trends and fluctuations in the prices of these essential commodities, offering valuable information for understanding economic stability and resource dependencies. The unemployment statistics, obtained from the Turkish Statistical Institute, contribute essential labor market insights, while the exchange rate data from the Turkish government's official portal and the interest rate data from the St. Louis Fed's FRED platform provide a comprehensive view of monetary policy and its impact on the economy.

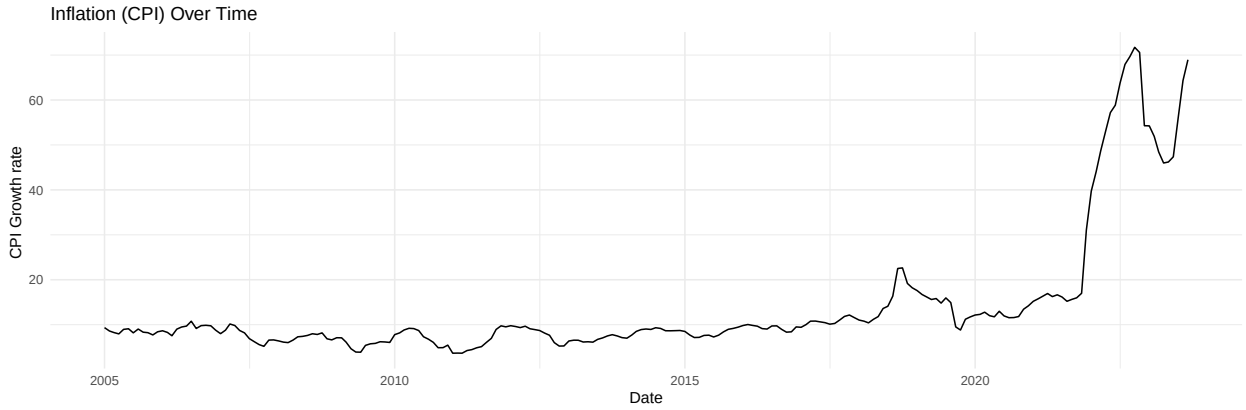
```
# Load data from Excel file
data <- read_excel("inflation_data_tur.xlsx")

# Convert the 'date' column to a Date type using anytime
data$date <- anytime(data$date)

# Set the date for splitting the data
split_date <- as.Date("2023-06-01")  # Change the date accordingly

# Create training and testing sets
train_data <- subset(data, date < split_date)
test_data <- subset(data, date >= split_date)

# Print the dimensions of the training and testing sets
cat("Dimensions of Training Data:", dim(train_data), "\n")
```

```
## Dimensions of Training Data: 221 7
```

```
cat("Dimensions of Testing Data:", dim(test_data), "\n")
```

```
## Dimensions of Testing Data: 4 7
```





Comment: The time series plot shows an upward trend in the CPI over time, indicating that inflation has generally increased but the dramatic increases after 2021.

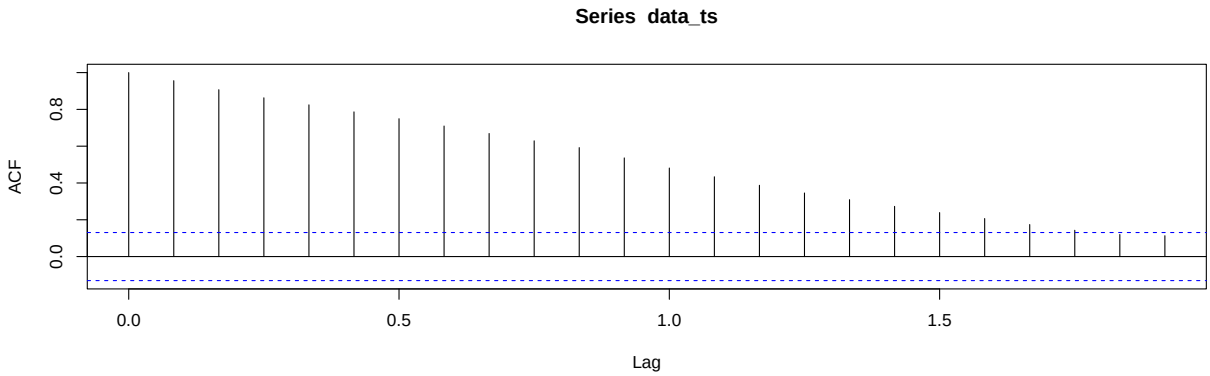
The Turkey CPI graph shows that Turkey's inflation rate has been steadily increasing since 2015, with a sharp rise in 2022 and 2023. As of November 2023, Turkey's annual inflation rate is 61.98%, the highest it has been in decades.

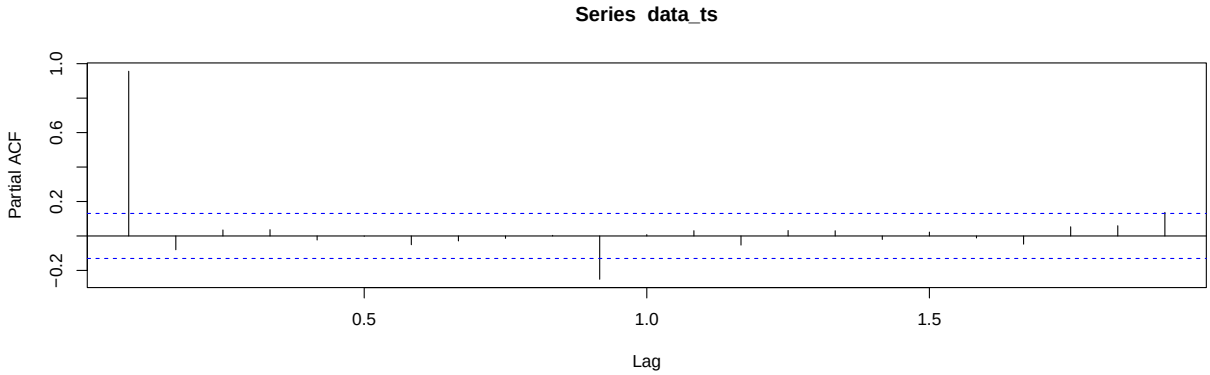
There are a number of factors that have contributed to Turkey's high inflation rate, including:

The COVID-19 pandemic, which disrupted supply chains and drove up prices. The war in Ukraine, which has further disrupted supply chains and caused energy prices to soar. The Turkish government's unorthodox economic policies, such as cutting interest rates despite high inflation. The high inflation rate has had a significant impact on the Turkish economy, eroding people's purchasing power and making it difficult for businesses to operate. It has also led to a decline in the value of the Turkish lira.

The Turkish government has taken some steps to address the inflation problem, such as raising interest rates and providing subsidies for certain goods. However, it is unclear whether these measures will be enough to bring inflation under control.

Overall, the Turkey CPI graph shows a worrying trend of rising inflation. The Turkish government needs to take decisive action to address this problem in order to protect the Turkish economy and its people.






```
## integer(0)
```

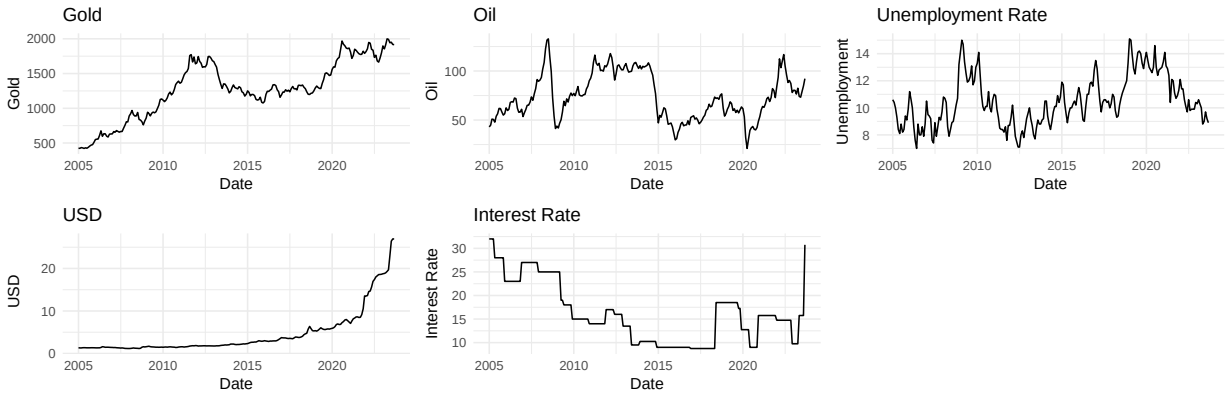
```
## pdf
##   2
```

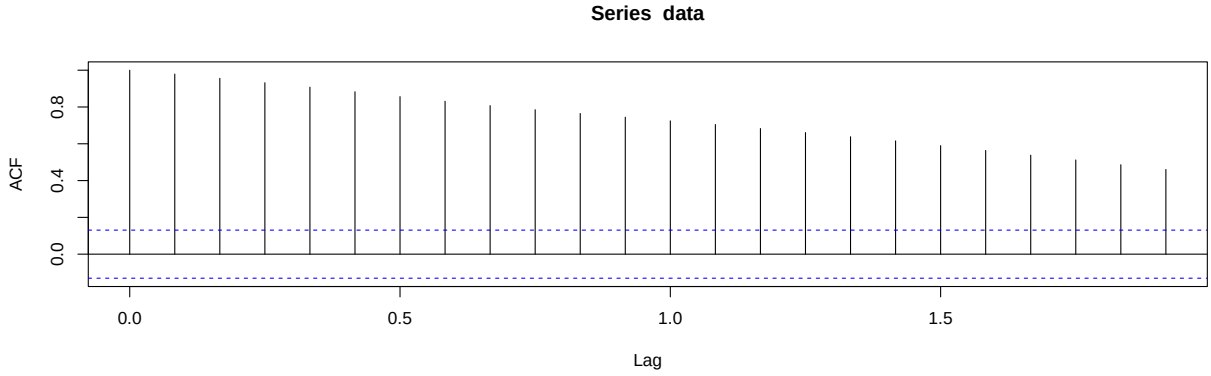
Comment: The analysis of the Consumer Price Index (CPI) reveals a strong positive autocorrelation at lag 1, indicating high persistence in inflation. The series is non-stationary, requiring differencing for stationarity. This information is vital for developing forecasting models, such as the Autoregressive Integrated Moving Average (ARIMA) model.

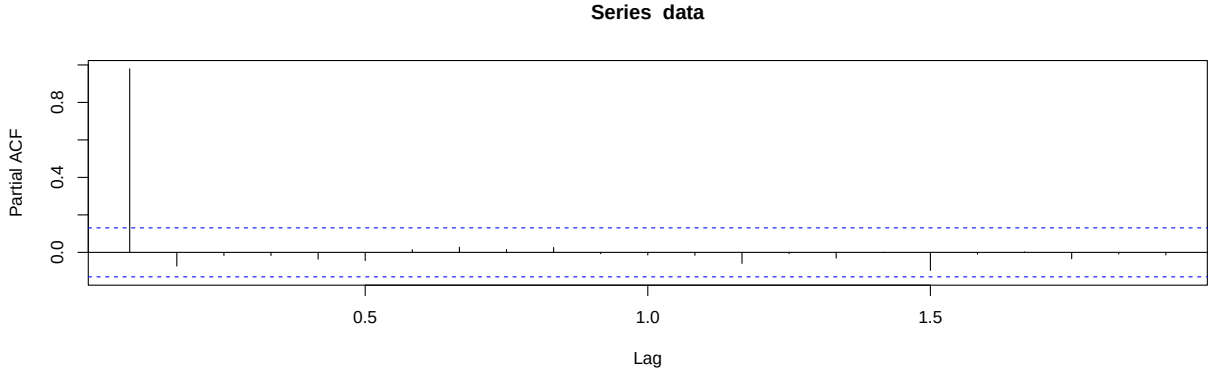
Additional insights highlight the CPI's positive autocorrelation at lags 1 through 12, indicating a highly autocorrelated series with decreasing correlation over time. The Autocorrelation Function (ACF) is valuable for identifying seasonal patterns and outliers.

Similarly, the Partial Autocorrelation Function (PACF) for Turkey's inflation shows a significant positive correlation at lag 1, suggesting high persistence and non-stationarity. This information is crucial for forecasting models like ARIMA.

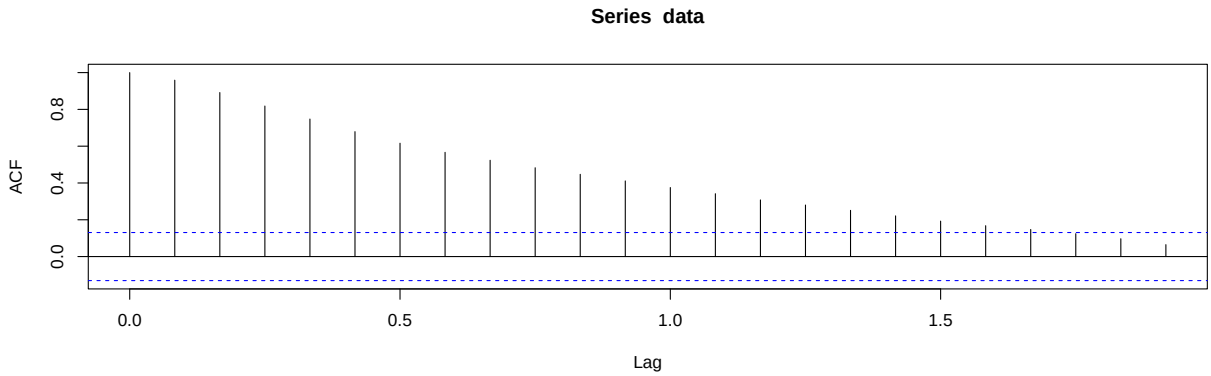
Observations on the PACF of Turkey's inflation note similarities with other macroeconomic variables, indicating common driving factors. The PACF can identify shocks to the inflation rate and assess the effectiveness of monetary policy, with a high correlation suggesting successful control.

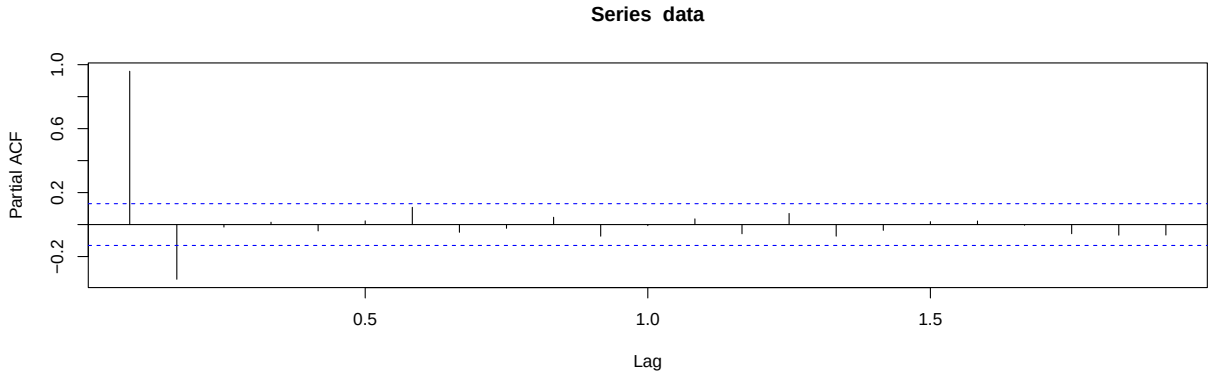

Gold
Gold
2000
1500
1000
500
2005
2010
2015
2020
Date
Oil
Oil
100
50
2005
2010
2015
2020
Date
Unemployment Rate
Unemployment
14
12
10
8
2005
2010
2015
2020
Date
USD
USD
20
10
0
2005
2010
2015
2020
Date
Interest Rate
Interest Rate
30
25
20
15
10
2005
2010
2015
2020
Date


Series data
ACF
0.8
0.4
0.0
0.0
0.5
1.0
1.5
Lag

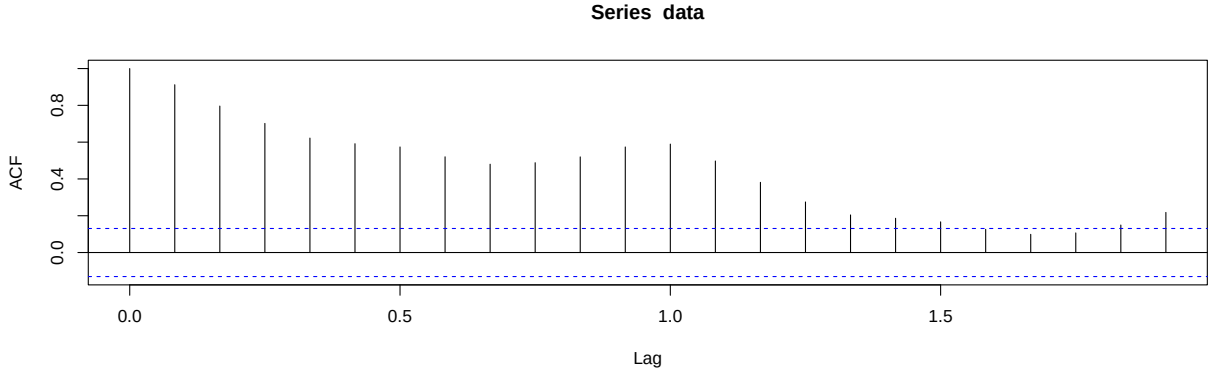

Series data
Partial ACF
0.8
0.4
0.0
0.5
1.0
1.5
Lag

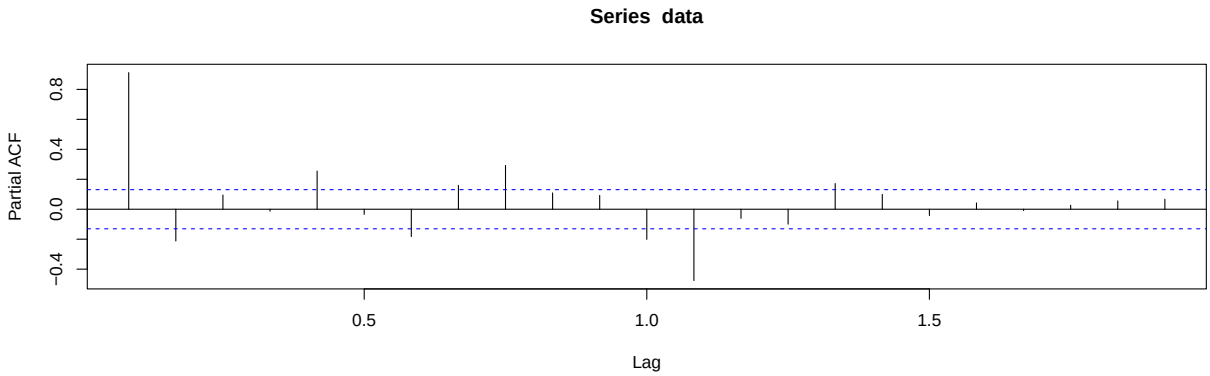
```
## pdf
##   2
```


Series data
ACF
0.0
0.4
0.8
0.0
0.5
1.0
1.5
Lag

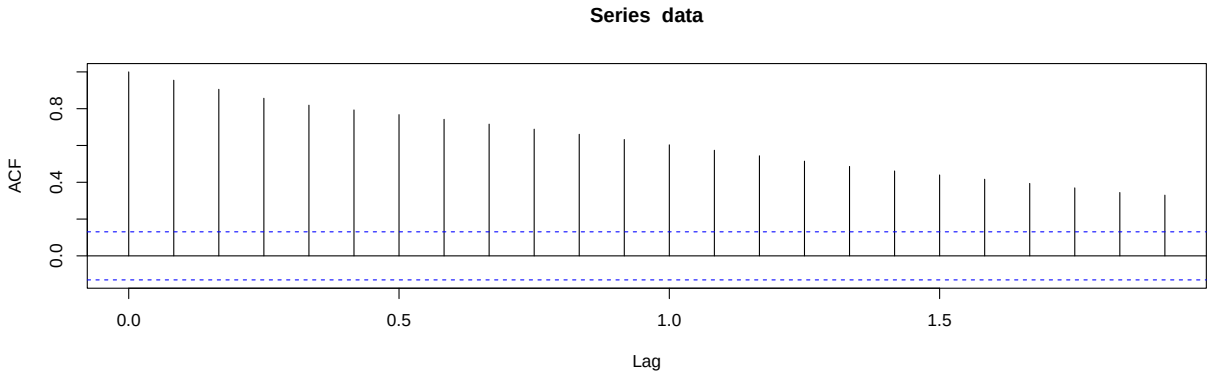

Series data
Partial ACF
−0.2
0.2
0.6
1.0
0.5
1.0
1.5
Lag

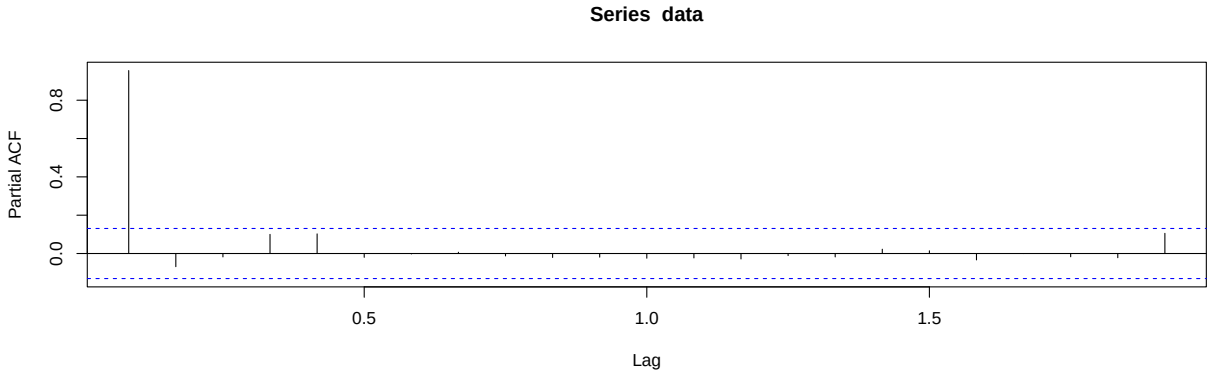
```
## pdf
##   2
```


Series data
ACF
0.0
0.4
0.8
0.0
0.5
1.0
1.5
Lag

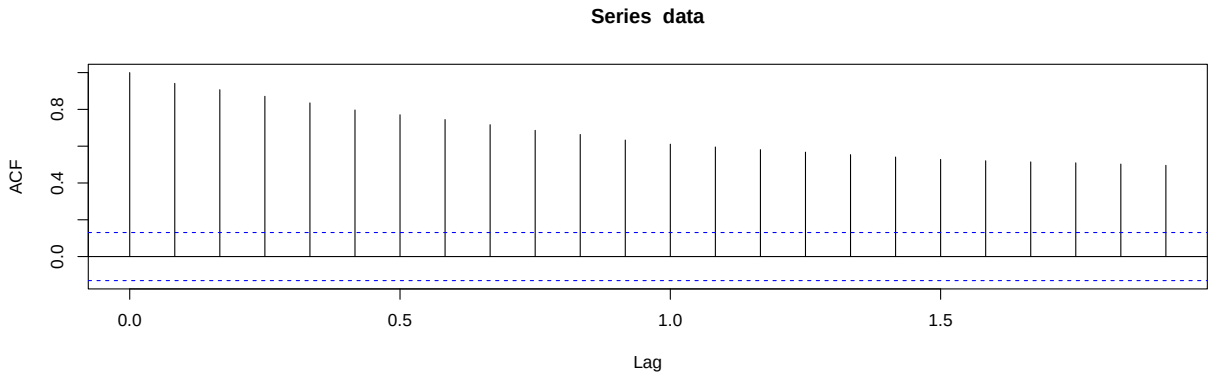

Series data
Partial ACF
0.8
0.4
0.0
−0.4
0.5
1.0
1.5
Lag

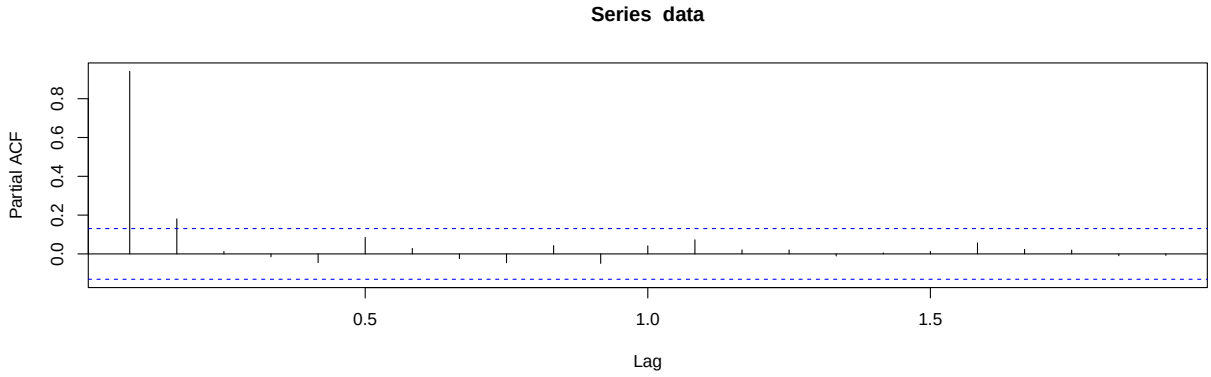
```
## pdf
##   2
```


Series data
ACF
0.8
0.4
0.0
0.0
0.5
1.0
1.5
Lag


Series data
Partial ACF
0.8
0.4
0.0
0.5
1.0
1.5
Lag

```
## pdf
##   2
```






```
## pdf
##   2
```

```
# Check for stationarity using ADF tests
adf_test <- ur.df(data$inflation, lags = 6, selectlags = "AIC", type = "drift")
summary(adf_test)
```

```
##
## ###############################################
## # 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
## -16.1736  -0.4822  -0.1298   0.3612  13.4479
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.107003   0.184166   0.581    0.562
## z.lag.1     0.004053   0.009645   0.420    0.675
## z.diff.lag  0.457391   0.063382   7.216 9.04e-12 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.933 on 215 degrees of freedom
## Multiple R-squared:  0.2127, Adjusted R-squared:  0.2054
## F-statistic: 29.05 on 2 and 215 DF,  p-value: 6.796e-12
##
##
## Value of test-statistic is: 0.4202 0.8341
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

```r
# Check for stationarity using DF-GLS tests
dfgls_test <- ur.ers(data$inflation, lag.max = 6, model = "constant")
summary(dfgls_test)
```

```
##
## ###############################################
## # Elliot, Rothenberg and Stock Unit Root Test #
## ###############################################
##
## Test of type DF-GLS
## detrending of series with intercept
##
##
## Call:
## lm(formula = dfgls.form, data = data.dfgls)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.7239  -0.3174   0.0403   0.5295  13.4116
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## yd.lag        0.0070077  0.0104542   0.670   0.5034
## yd.diff.lag1  0.4474274  0.0700483   6.387 1.06e-09 ***
## yd.diff.lag2 -0.0006087  0.0766112  -0.008   0.9937
## yd.diff.lag3  0.1659562  0.0780132   2.127   0.0346 *
## yd.diff.lag4 -0.0740345  0.0780844  -0.948   0.3441
## yd.diff.lag5 -0.0917150  0.0782505  -1.172   0.2425
## yd.diff.lag6 -0.0063442  0.0752928  -0.084   0.9329
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.925 on 211 degrees of freedom
## Multiple R-squared:  0.2462, Adjusted R-squared:  0.2212
## F-statistic: 9.845 on 7 and 211 DF,  p-value: 1.301e-10
##
##
## Value of test-statistic is: 0.6703
##
## Critical values of DF-GLS are:
##                  1pct  5pct 10pct
```

```
## critical values -2.57 -1.94 -1.62
```

Comment: Both the ADF test and the DF-GLS test indicate that the Turkish inflation data is non-stationary, as the p-values of the test statistics are greater than the significance level of 0.05. This means that the data has a trend and/or seasonal component, and that forecasting future values of inflation based on the current values will not be accurate.

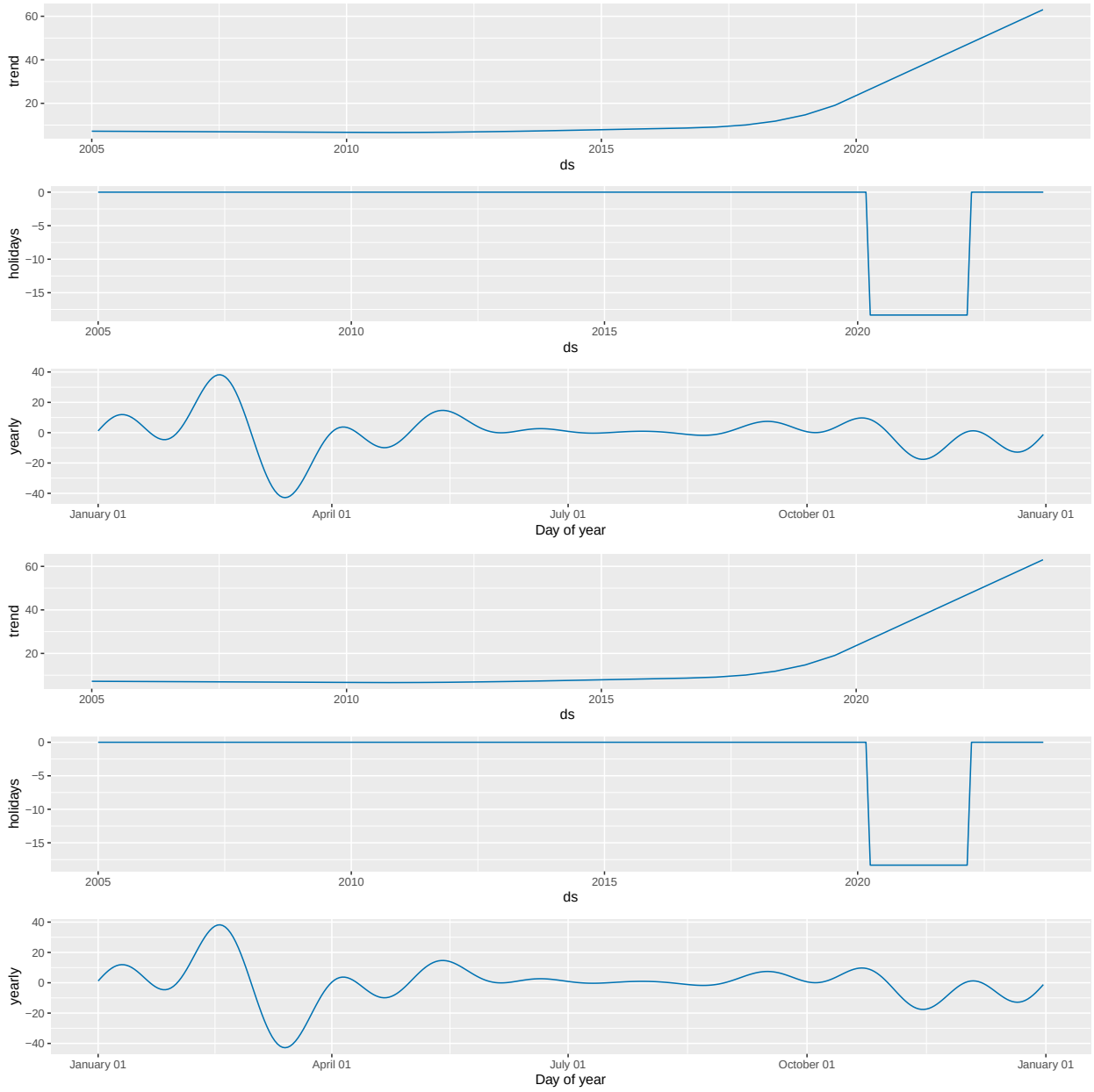
Prophet

```
train_data_prophet <- data.frame(ds = train_data$date,
                                 y = train_data$inflation)

# Create a dummy variable 'holiday' from
# March 15th of 2020 to March 1st of 2022
covid <- data.frame(
  holiday = 'covid',
  ds = seq(as.Date('2020-3-15'), to=as.Date('2022-3-1'), by='days')
)

# Extract month and year
train_data_prophet$month <- months(train_data_prophet$ds)
train_data_prophet$year <- as.factor(year(train_data_prophet$ds))

# Create a custom monthly seasonality
prophet_model <- prophet(train_data_prophet,
                         holidays = covid,
                         yearly.seasonality = TRUE,
                         weekly.seasonality = FALSE,
                         daily.seasonality = FALSE)

# Make future data for prediction
future <- make_future_dataframe(prophet_model,
                                periods = 4, freq = 'months')

# Predict
forecast <- predict(prophet_model, future)

# Extract predicted values from the forecast
predicted_values <- forecast$yhat[1:length(test_data$inflation)]

# Calculate RMSE
rmse_prophet <- sqrt(mean((test_data$inflation - predicted_values)^2))

# Calculate MAE
mae_prophet <- mean(abs(test_data$inflation - predicted_values))

# Print the results
cat("RMSE for Prophet:", rmse_prophet, "\n")
```

```
## RMSE for Prophet: 52.15666
```

```
cat("MAE for Prophet:", mae_prophet, "\n")
```

```
## MAE for Prophet: 51.41323
```




```
data$date <- as.POSIXct(data$date)
data <- data[order(data$date),]

# Compute differences for each variable
data_diff <- data.frame(
  date = data$date,
  inflation= c(NA, diff(data$inflation)),
  gold_data= c(NA, diff(data$gold_data)),
  oil = c(NA, diff(data$oil)),
  unemployment = c(NA, diff(data$unemployment)),
  usd = c(NA, diff(data$usd)),
  int_rate= c(NA, diff(data$int_rate))
)
```

```r
# Drop rows with NA values
data_diff <- na.omit(data_diff)

# Convert the 'date' column to a Date type using anytime
data_diff$date <- anytime(data_diff$date)

# Set the date for splitting the data
split_date <- as.Date("2023-06-01")  # Change the date accordingly

# Create training and testing sets
data <- data_diff
train_data <- subset(data_diff, date < split_date)
test_data <- subset(data_diff, date >= split_date)

# Print the dimensions of the training and testing sets
cat("Dimensions of Training Data:", dim(train_data), "\n")
```

```
## Dimensions of Training Data: 220 7
```

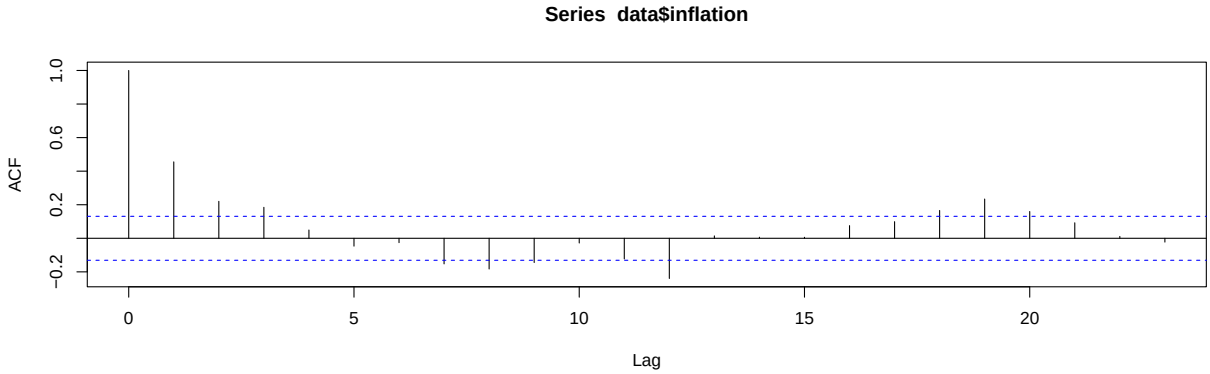
```r
cat("Dimensions of Testing Data:", dim(test_data), "\n")
```

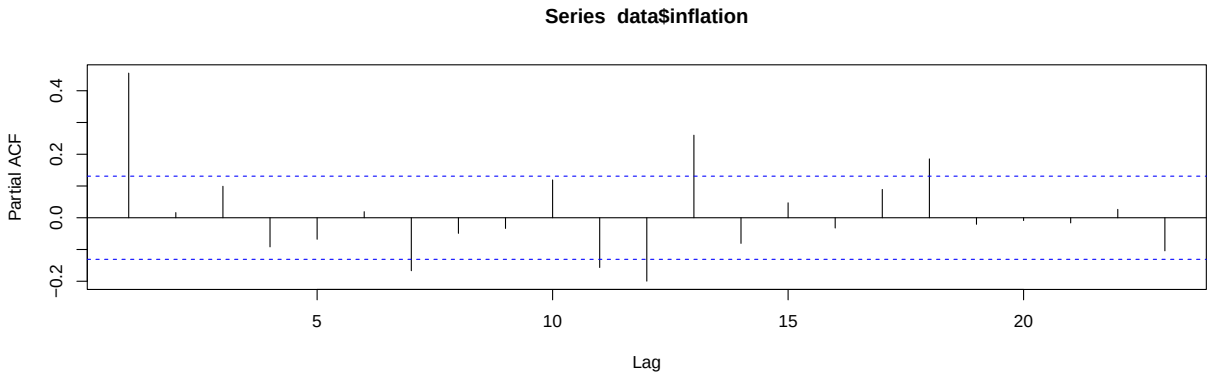
```
## Dimensions of Testing Data: 4 7
```

```r
# Check for stationarity again using the ADF test
adf_test <- ur.df(data$inflation, lags = 6, selectlags = "AIC", type = "drift")
summary(adf_test)
```

```
##
## ###############################################
## # 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
## -14.9446  -0.5113  -0.1339   0.3728  13.1397
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.18004    0.13066   1.378  0.16968
## z.lag.1     -0.64582    0.11247  -5.742 3.26e-08 ***
## z.diff.lag1  0.09684    0.11071   0.875  0.38273
## z.diff.lag2  0.08604    0.10394   0.828  0.40875
## z.diff.lag3  0.24715    0.09983   2.476  0.01409 *
## z.diff.lag4  0.21408    0.09589   2.233  0.02664 *
## z.diff.lag5  0.12620    0.08559   1.474  0.14189
## z.diff.lag6  0.20164    0.07253   2.780  0.00593 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

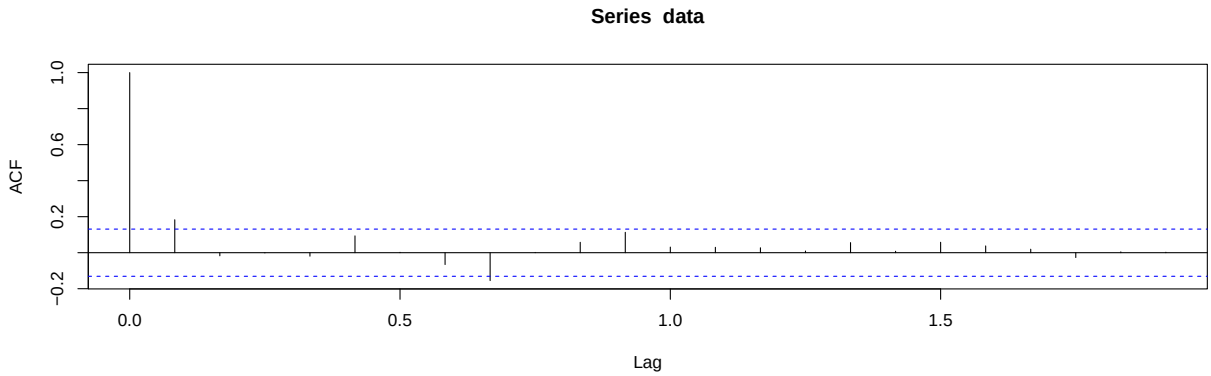
```
## Residual standard error: 1.894 on 209 degrees of freedom
## Multiple R-squared:  0.3114, Adjusted R-squared:  0.2883
## F-statistic:  13.5 on 7 and 209 DF,  p-value: 2.345e-14
##
##
## Value of test-statistic is: -5.7421 16.5596
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

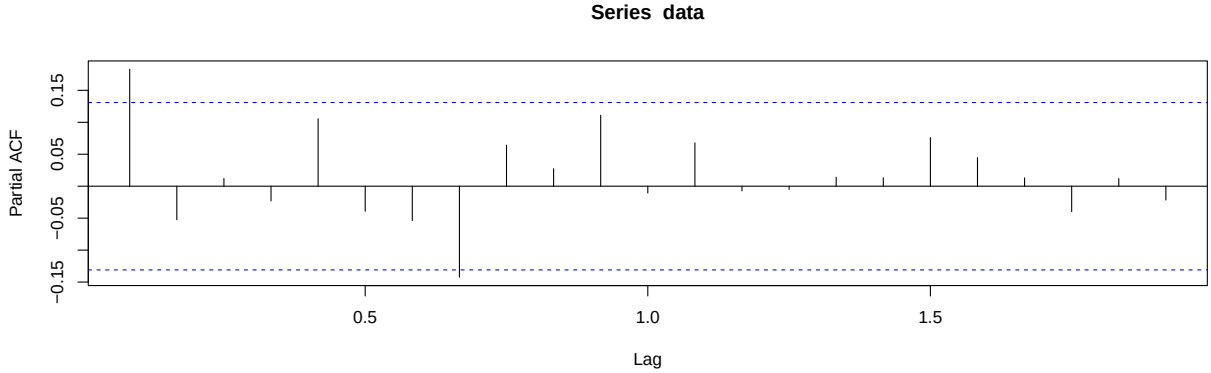




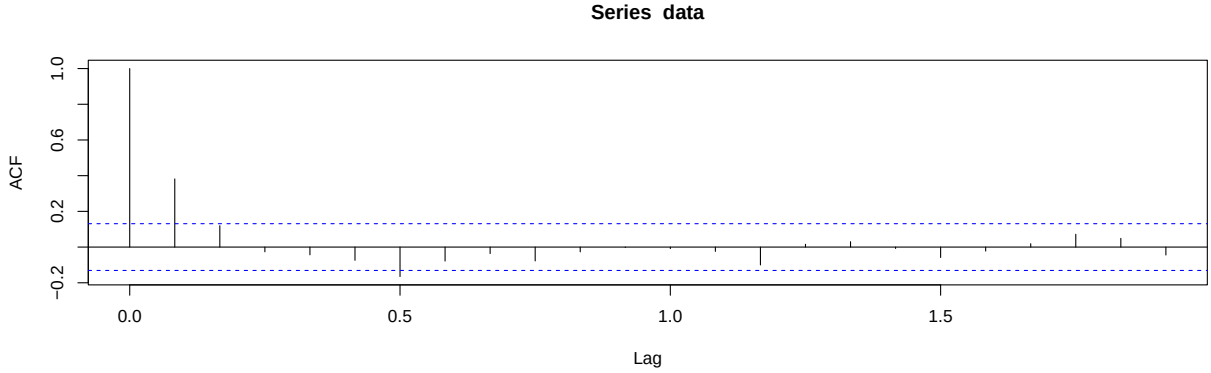

```
## pdf
##   2
```

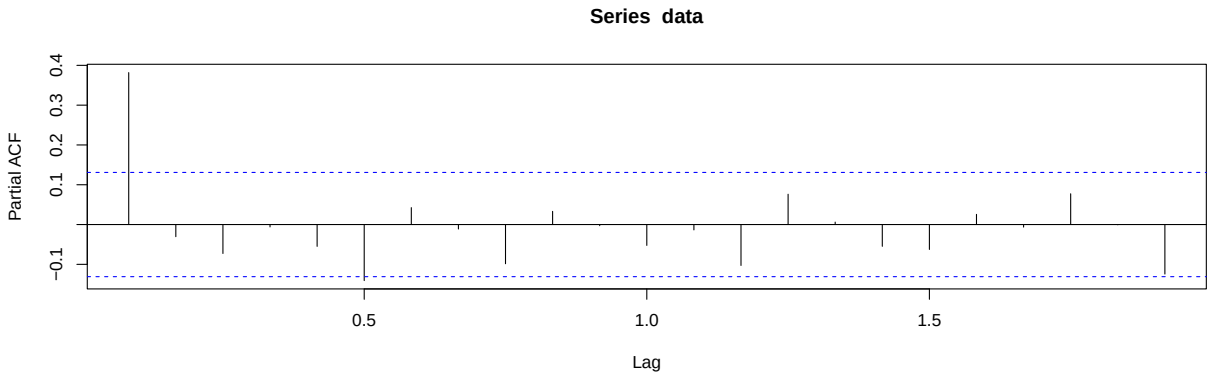
Comment : the ADF test results provide strong evidence that the first difference of the inflation data is stationary.


Series data
ACF
1.0
0.6
0.2
−0.2
0.0
0.5
1.0
1.5
Lag

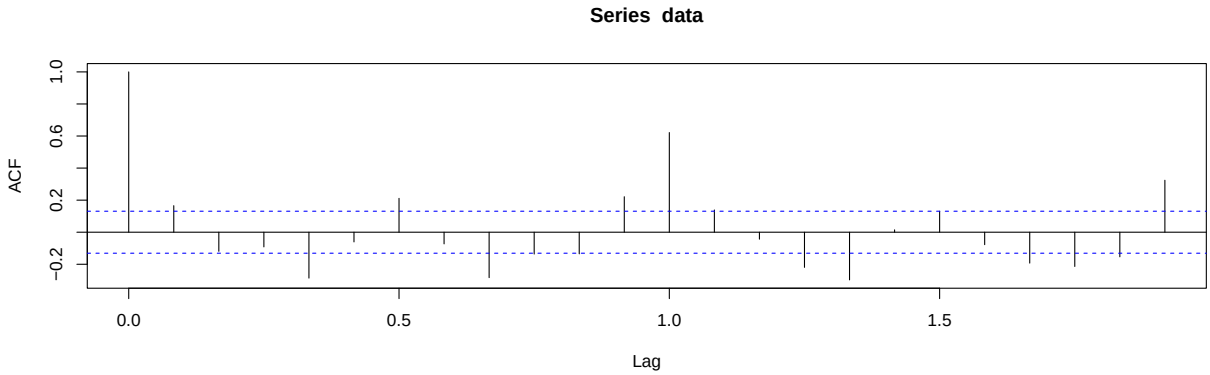

Series data
Partial ACF
0.15
0.05
−0.05
−0.15
0.5
1.0
1.5
Lag

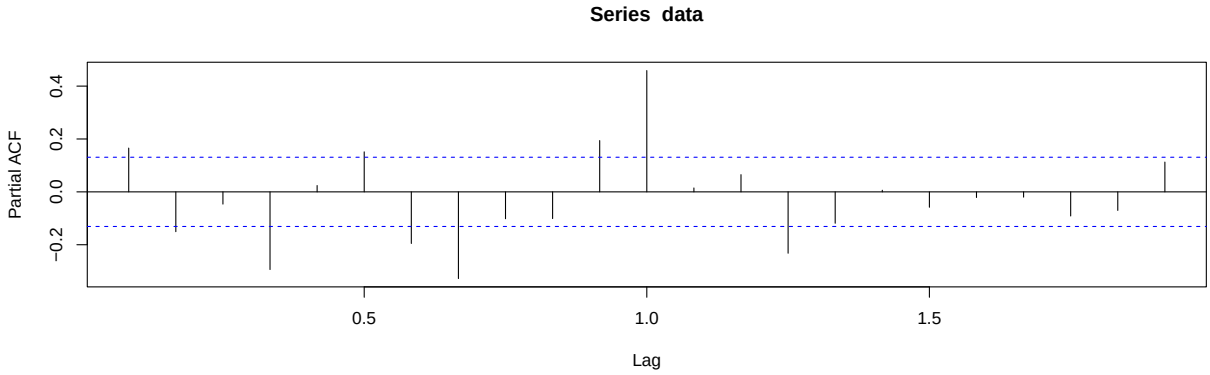
```
## pdf
##   2
```


Series data
ACF
1.0
0.6
0.2
−0.2
0.0
0.5
1.0
1.5
Lag

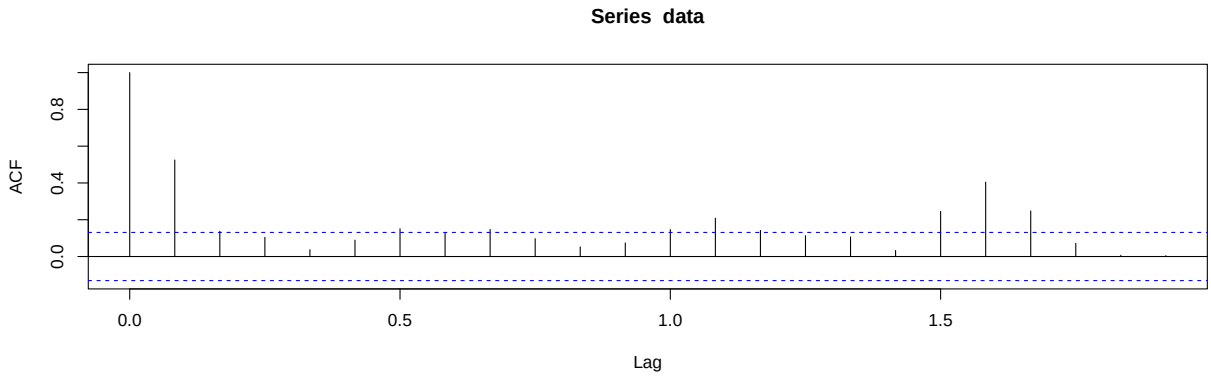

Series data
Partial ACF
0.4
0.3
0.2
0.1
−0.1
0.5
1.0
1.5
Lag

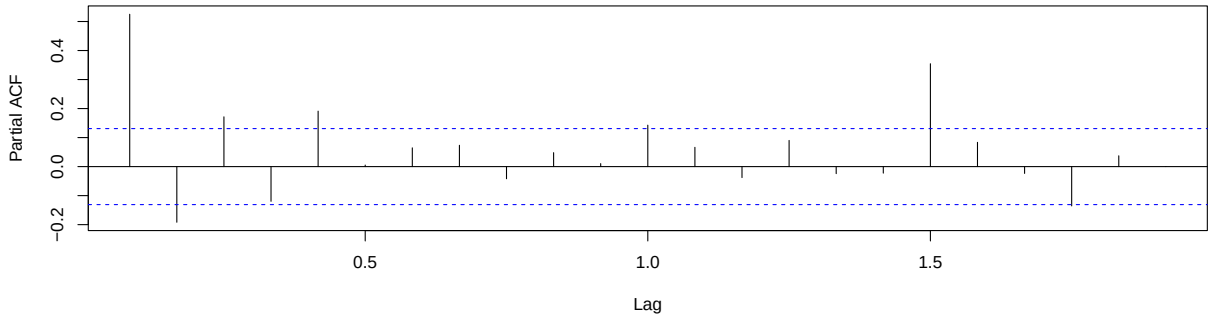
```
## pdf
##   2
```


Series data
ACF
1.0
0.6
0.2
−0.2
0.0
0.5
1.0
1.5
Lag

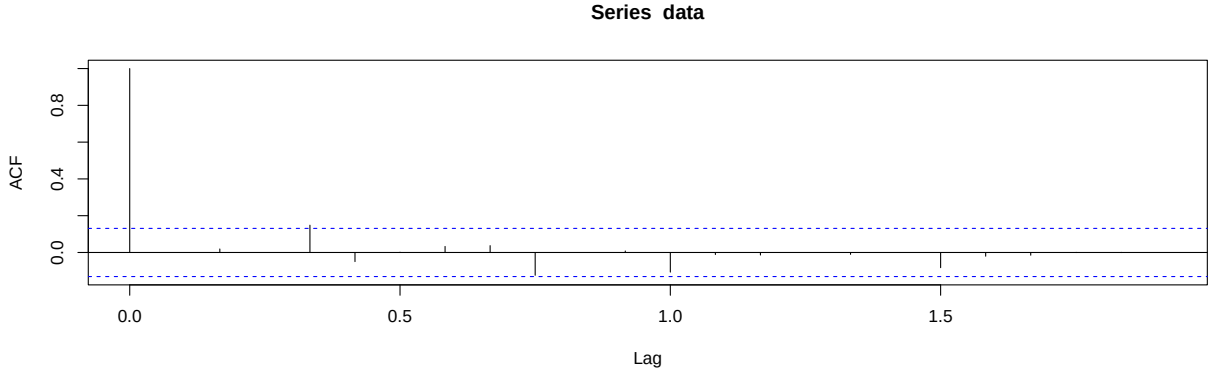

Series data
Partial ACF
0.4
0.2
0.0
−0.2
0.5
1.0
1.5
Lag

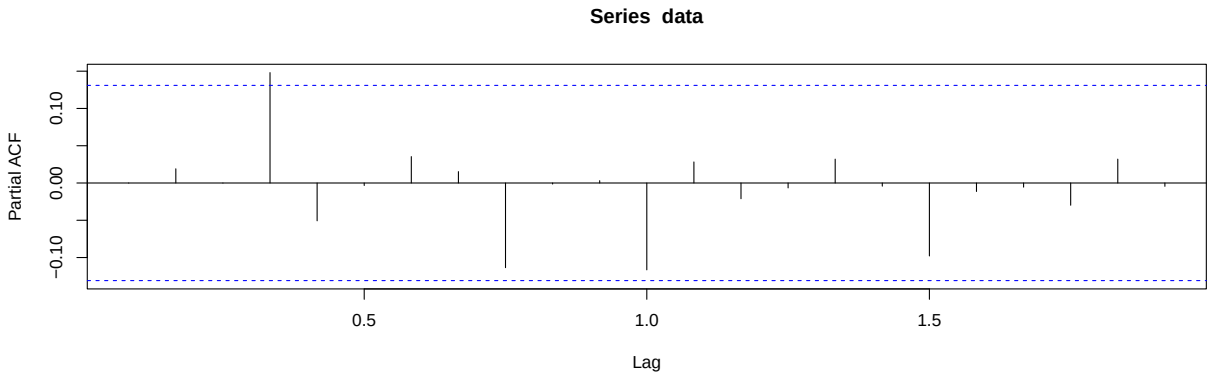
```
## pdf
##   2
```


Series data
ACF
0.0
0.4
0.8
0.0
0.5
1.0
1.5
Lag


Series data
Partial ACF
−0.2
0.0
0.2
0.4
0.5
1.0
1.5
Lag

```
## pdf
##   2
```


Series data
ACF
0.0
0.4
0.8
0.0
0.5
1.0
1.5
Lag

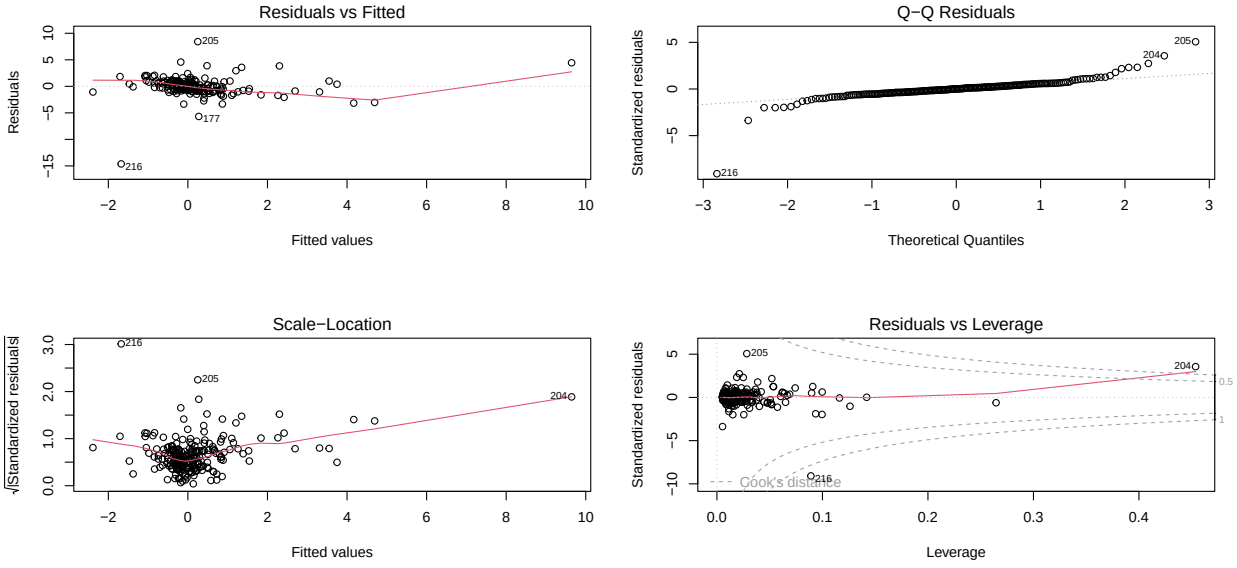


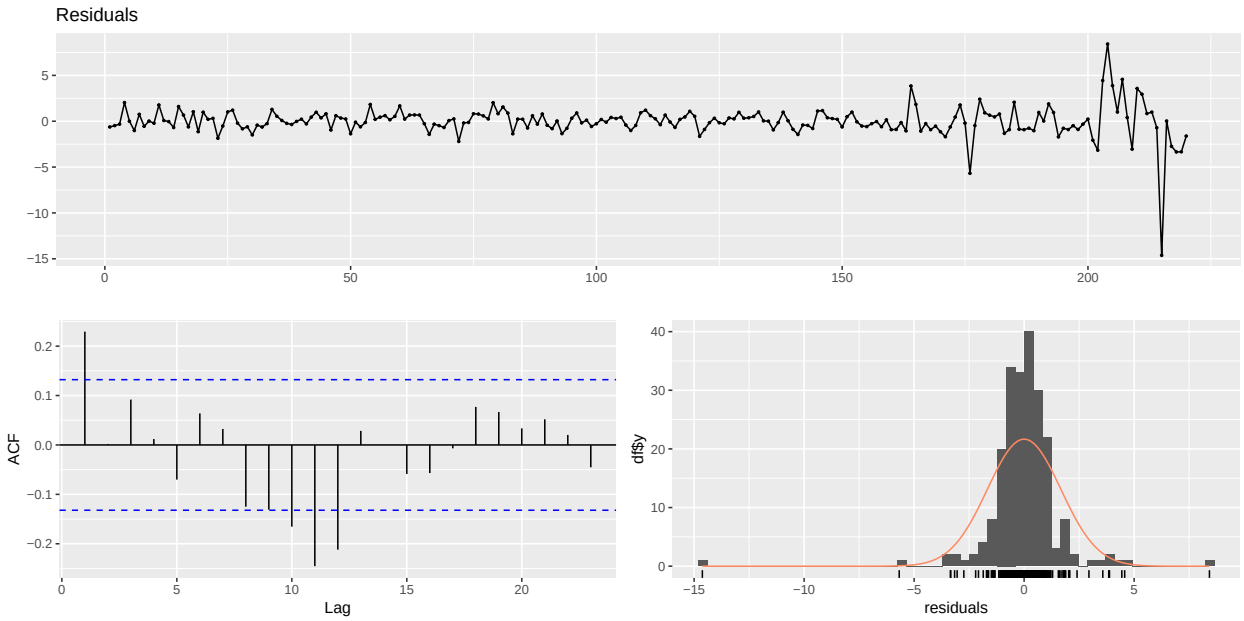

```
## pdf
##   2
```

Linear Regression Model

```
LmFit1 <- lm(inflation ~ gold_data + oil +
               unemployment + usd + int_rate, data=train_data)
summary(LmFit1)
```

```
##
## Call:
## lm(formula = inflation ~ gold_data + oil + unemployment + usd +
##     int_rate, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.6249  -0.6207   0.0196   0.6222   8.4220
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.062111   0.120086  -0.517  0.60553
## gold_data    -0.004993   0.002491  -2.004  0.04628 *
## oil           0.051394   0.018547   2.771  0.00608 **
## unemployment  0.121464   0.151214   0.803  0.42271
## usd           3.282196   0.386312   8.496 3.33e-15 ***
## int_rate      0.197227   0.086739   2.274  0.02397 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.687 on 214 degrees of freedom
## Multiple R-squared:  0.2977, Adjusted R-squared:  0.2813
## F-statistic: 18.14 on 5 and 214 DF,  p-value: 5.358e-15
```






```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 37.63, df = 10, p-value = 4.401e-05
```

```
# Check for collinearity using VIF
library(car)
vif_results1 <- vif(LmFit1)

# Print VIF results
print(vif_results1)
```

```
##    gold_data          oil unemployment          usd     int_rate
##     1.050090     1.051436     1.042511     1.013417     1.003806
```

```
forecast_LmFit1 <- predict(LmFit1, newdata = test_data)

# Calculate Mean Absolute Error (MAE) for the structural model
mae_LmFit1 <- mean(abs(test_data$inflation - forecast_LmFit1))

# Print the result
cat("MAE for LmFit1:", mae_LmFit1, "\n")
```

```
## MAE for LmFit1: 5.18706
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_LmFit1 <- sqrt(mean((test_data$inflation - forecast_LmFit1)^2))

# Print the result
cat("RMSE for LmFit1:", rmse_LmFit1, "\n")
```

```
## RMSE for LmFit1: 6.191326
```

Comment: The Linear regression model to predict inflation using the independent variables 'gold_data,' 'oil,' 'unemployment,' 'usd,' and 'int_rate.' The model indicates that 'gold_data,' 'oil,' 'usd,' and 'int_rate' have statistically significant coefficients, suggesting they play a role in explaining inflation variations. However,

'unemployment' does not appear to be a significant predictor. The overall model is statistically significant, as evidenced by the F-statistic and its associated p-value. The model explains approximately 29.77% of the variance in inflation, as indicated by the Multiple R-squared. The performance metrics, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), provide insights into the model's accuracy, with MAE at 5.18 and RMSE at 6.19. Interpretation and further analysis should consider the specific context of the data and assumptions underlying linear regression.

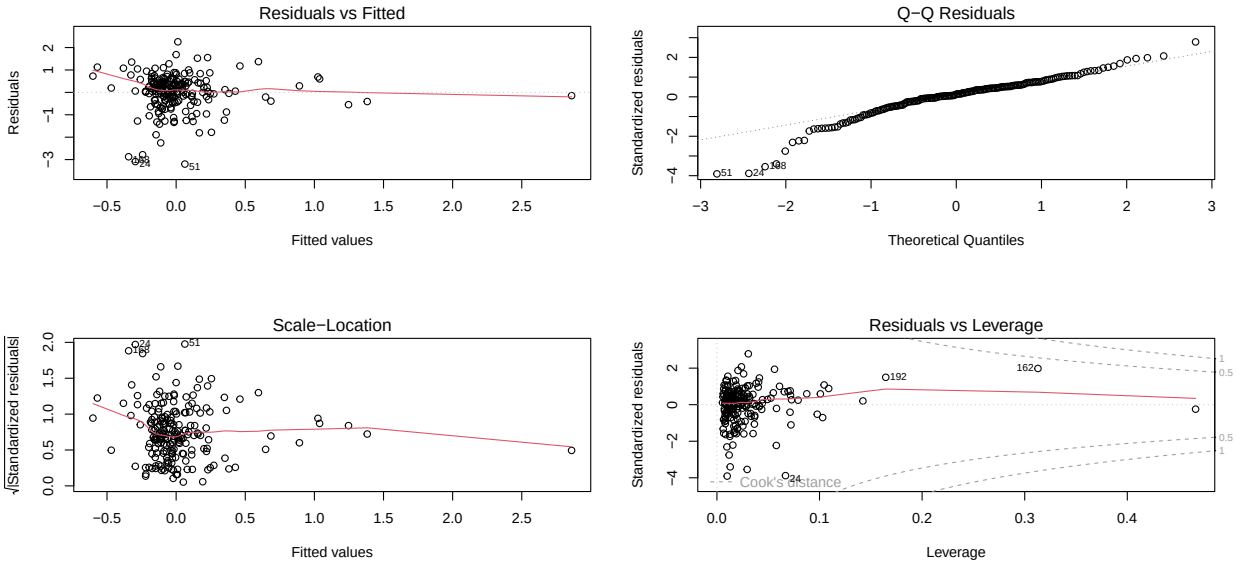
Linear Regression Model - Log Transfer

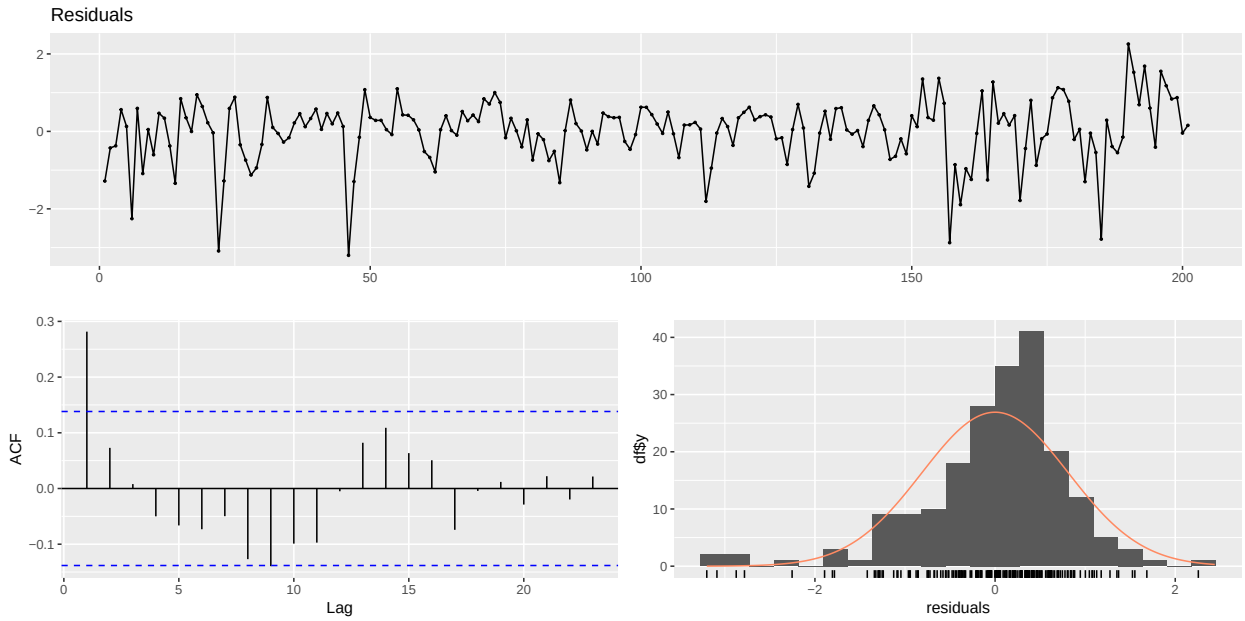
```
LmFit2 <- lm(log(inflation+1) ~ gold_data + oil + unemployment +usd + int_rate, data=train_data)
```

```
## Warning in log(inflation + 1): NaNs produced
```

```
summary(LmFit2)
```

```
##
## Call:
## lm(formula = log(inflation + 1) ~ gold_data + oil + unemployment +
##     usd + int_rate, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2006 -0.3606  0.1013  0.4576  2.2564
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0760146  0.0613815  -1.238    0.217
## gold_data    -0.0006789  0.0012913  -0.526    0.600
## oil           0.0113655  0.0095812   1.186    0.237
## unemployment -0.0650887  0.0777488  -0.837    0.404
## usd           1.0053393  0.1921639   5.232 4.32e-07 ***
## int_rate     -0.0417421  0.0461213  -0.905    0.367
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8232 on 195 degrees of freedom
##   (19 observations deleted due to missingness)
## Multiple R-squared:  0.1372, Adjusted R-squared:  0.1151
## F-statistic: 6.202 on 5 and 195 DF,  p-value: 2.325e-05
```






```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 22.354, df = 10, p-value = 0.0134
```

```
# Check for collinearity using VIF
library(car)
vif_results2 <- vif(LmFit2)

# Print VIF results
print(vif_results2)
```

```
##    gold_data          oil unemployment          usd     int_rate
##     1.051505     1.058096     1.038272     1.016617     1.003812
```

```
forecast_LmFit2 <- predict(LmFit2, newdata = test_data)
# Convert the log-transformed predictions back to the original scale
forecast_LmFit2 <- exp(forecast_LmFit2)

# Calculate Mean Absolute Error (MAE) for the structural model
mae_LmFit2 <- mean(abs(test_data$inflation - forecast_LmFit2))
# Print the result
cat("MAE for LmFit1:", mae_LmFit2, "\n")
```

```
## MAE for LmFit1: 14.17113
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_LmFit2 <- sqrt(mean((test_data$inflation - forecast_LmFit2)^2))

# Print the result
cat("RMSE for LmFit1:", rmse_LmFit2, "\n")
```

```
## RMSE for LmFit1: 16.92242
```

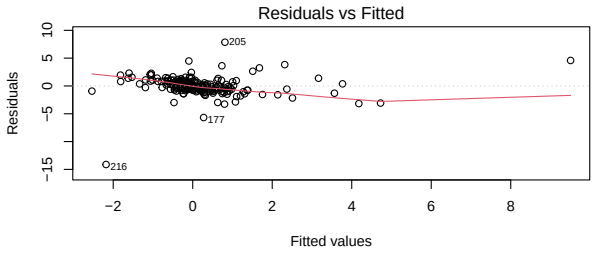
Comment: Linear regression model with the natural logarithm of the variable 'inflation + 1' as the dependent variable and includes the independent variables 'gold_data,' 'oil,' 'unemployment,' 'usd,' and 'int_rate.' The

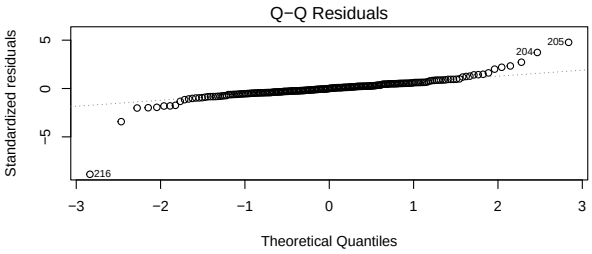
logarithmic transformation is applied to the dependent variable, possibly to address non-constant variance or skewness in the original 'inflation' variable. The summary output reveals that only the coefficient for 'usd' is statistically significant at a conventional significance level of 0.05, indicating that changes in the exchange rate ('usd') are associated with a significant effect on the logarithm of inflation. The overall model is statistically significant based on the F-statistic and its associated p-value. However, the model's explanatory power is relatively low, as evidenced by the Multiple R-squared of 0.1372. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) provide measures of model prediction accuracy, with MAE at 14.17 and RMSE at 16.92.

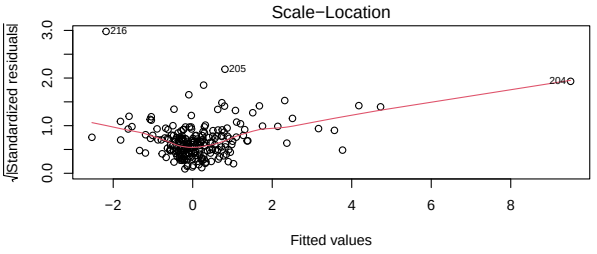
Linear Regression Model - factor month

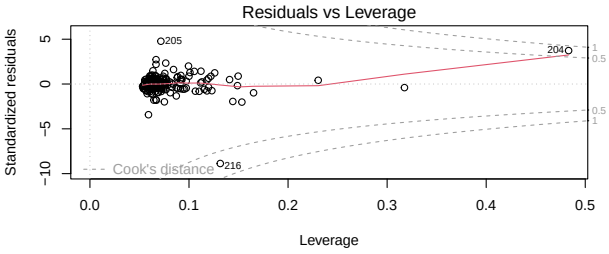
```r
# Extract month from the date variable
train_data$month <- as.factor(format(as.Date(train_data$date), "%m"))

# Fit the linear regression model with month effects as factors
LmFit3 <- lm(inflation ~ gold_data + oil + unemployment + usd + int_rate + month, data = train_data)

# Print model summary
summary(LmFit3)
```

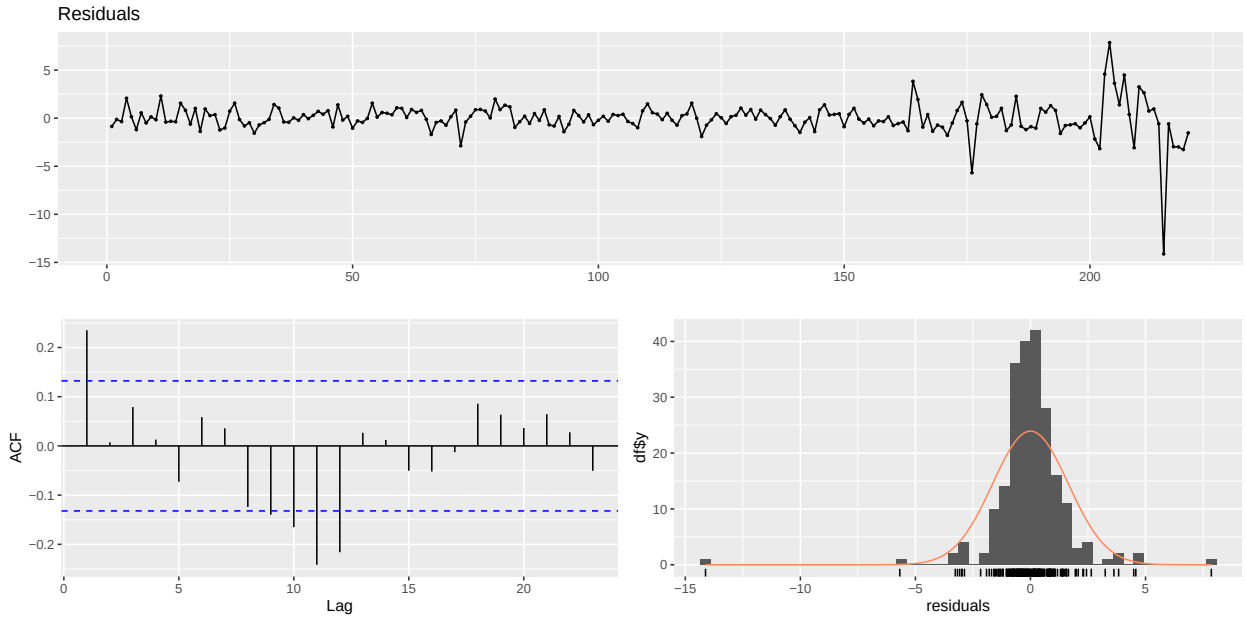
```
##
## Call:
## lm(formula = inflation ~ gold_data + oil + unemployment + usd +
##     int_rate + month, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.1219  -0.6231   0.0044   0.7431   7.8589
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.156898   0.396445   0.396   0.6927
## gold_data    -0.005891   0.002604  -2.263   0.0247 *
## oil           0.049442   0.019217   2.573   0.0108 *
## unemployment  0.009590   0.215165   0.045   0.9645
## usd           3.392963   0.395949   8.569 2.63e-15 ***
## int_rate      0.184506   0.088758   2.079   0.0389 *
## month02      -0.589777   0.577196  -1.022   0.3081
## month03      -0.266995   0.580890  -0.460   0.6463
## month04      -0.424596   0.572769  -0.741   0.4594
## month05      -0.378568   0.566343  -0.668   0.5046
## month06       0.045527   0.590164   0.077   0.9386
## month07      -0.062155   0.569339  -0.109   0.9132
## month08      -0.236181   0.568976  -0.415   0.6785
## month09      -0.222332   0.568412  -0.391   0.6961
## month10      -0.303024   0.567880  -0.534   0.5942
## month11      -0.687733   0.600613  -1.145   0.2535
## month12       0.466425   0.590722   0.790   0.4307
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.709 on 203 degrees of freedom
## Multiple R-squared:  0.3167, Adjusted R-squared:  0.2629
## F-statistic: 5.881 on 16 and 203 DF,  p-value: 1.753e-10
```












```
##
##  Breusch-Godfrey test for serial correlation of order up to 20
##
## data:  Residuals
## LM test = 69.016, df = 20, p-value = 2.634e-07
```

```
# Check for collinearity using VIF
vif_results3 <- vif(LmFit3)

# Print VIF results
print(vif_results3)
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## gold_data    1.118903  1        1.057782
```

```
## oil          1.100405  1        1.049002
## unemployment 2.057918  1        1.434545
## usd          1.037956  1        1.018801
## int_rate     1.024761  1        1.012305
## month        2.327105 11        1.039139
```

```
# Extract month from the date variable
test_data$month <- as.factor(format(as.Date(test_data$date), "%m"))

forecast_LmFit3 <- predict(LmFit3, newdata = test_data)

# Calculate Mean Absolute Error (MAE) for the structural model
mae_LmFit3 <- mean(abs(test_data$inflation - forecast_LmFit3))

# Print the result
cat("MAE for LmFit3:", mae_LmFit3, "\n")
```

```
## MAE for LmFit3: 5.356241
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_LmFit3 <- sqrt(mean((test_data$inflation - forecast_LmFit3)^2))

# Print the result
cat("RMSE for LmFit3:", rmse_LmFit3, "\n")
```

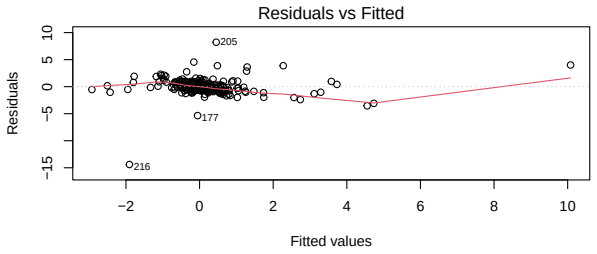
```
## RMSE for LmFit3: 6.294122
```

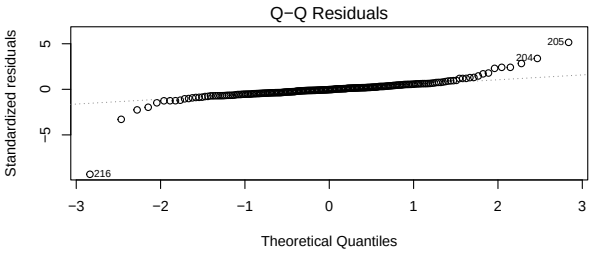
Comment: The linear regression model to include a categorical variable 'month' representing different months. The dependent variable remains 'inflation,' and the independent variables now include 'gold_data,' 'oil,' 'unemployment,' 'usd,' 'int_rate,' and dummy variables for each month from February (month02) to December (month12). The summary output shows that 'gold_data,' 'oil,' 'usd,' and 'int_rate' remain statistically significant, indicating their association with inflation. The inclusion of month-specific dummy variables allows for capturing potential seasonality or monthly effects. However, most month coefficients are not statistically significant, suggesting that the month of the year may not significantly impact inflation after accounting for other variables. The model explains about 31.67% of the variance in inflation, as indicated by the Multiple R-squared. The F-statistic and its associated p-value suggest overall model significance. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) provide measures of model prediction accuracy, with MAE at 5.35 and RMSE at 6.29.

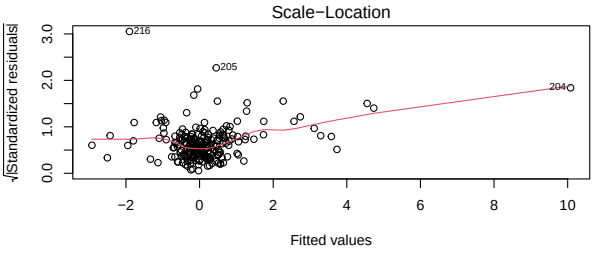
Linear Regression Model - factor year

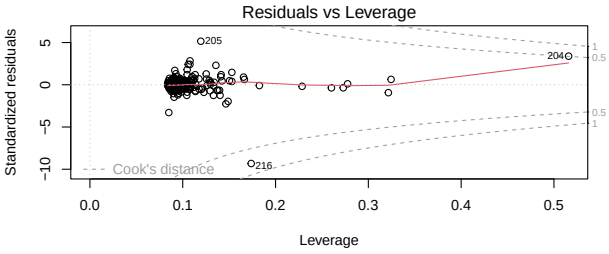
```r
# Extract month from the date variable
train_data$year <- as.factor(format(as.Date(train_data$date), "%Y"))

# Fit the linear regression model with month effects as factors
LmFit4 <- lm(inflation ~ gold_data + oil + unemployment + usd + int_rate + year, data = train_data)

# Print model summary
summary(LmFit4)
```

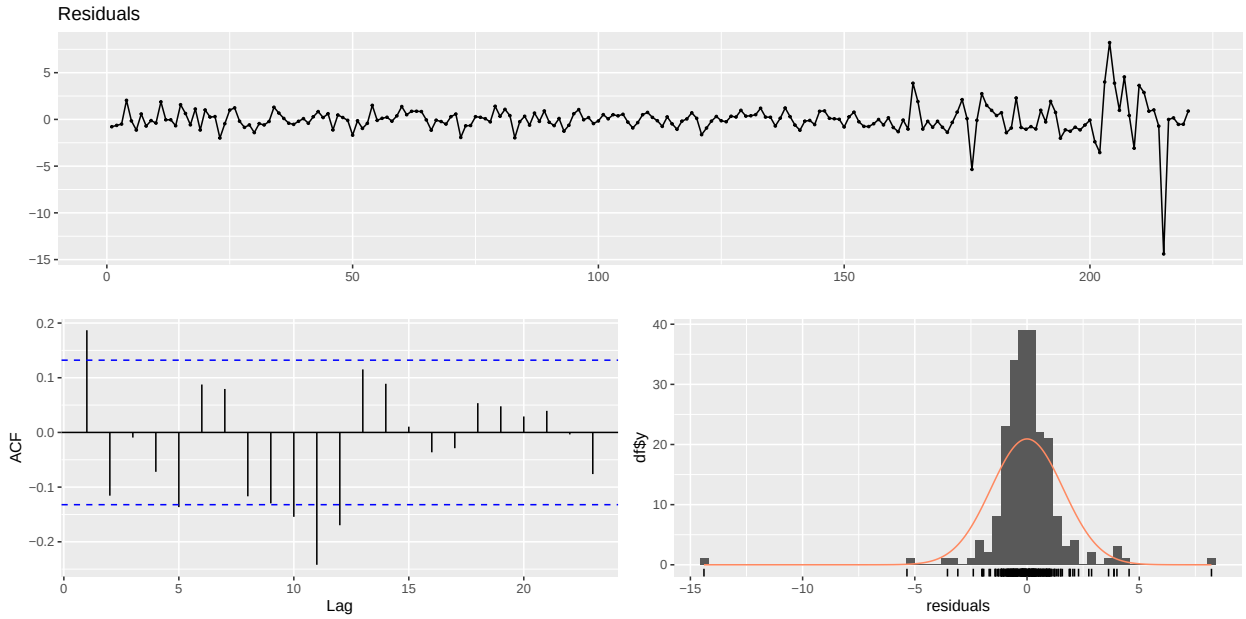
```
##
## Call:
## lm(formula = inflation ~ gold_data + oil + unemployment + usd +
##     int_rate + year, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3960  -0.6111  -0.0494   0.5531   8.2178
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.096025   0.496976   0.193  0.84699
## gold_data    -0.004495   0.002655  -1.693  0.09209 .
## oil           0.049244   0.019666   2.504  0.01310 *
## unemployment  0.088498   0.155938   0.568  0.57101
## usd           3.334500   0.446254   7.472 2.55e-12 ***
## int_rate      0.246677   0.092553   2.665  0.00833 **
## year2006     -0.185515   0.702915  -0.264  0.79212
## year2007     -0.177520   0.696874  -0.255  0.79919
## year2008      0.002956   0.706192   0.004  0.99666
## year2009      0.167748   0.694828   0.241  0.80948
## year2010     -0.400708   0.698414  -0.574  0.56680
## year2011      0.336725   0.702252   0.479  0.63212
## year2012     -0.274133   0.696134  -0.394  0.69416
## year2013     -0.257682   0.705556  -0.365  0.71534
## year2014      0.241708   0.706814   0.342  0.73274
## year2015     -0.158479   0.701816  -0.226  0.82158
## year2016     -0.402949   0.697743  -0.578  0.56426
## year2017      0.046303   0.698599   0.066  0.94722
## year2018     -0.198863   0.713983  -0.279  0.78090
## year2019     -0.496029   0.696354  -0.712  0.47711
## year2020     -0.160730   0.705713  -0.228  0.82007
## year2021      0.133408   0.732993   0.182  0.85577
## year2022     -0.181784   0.721815  -0.252  0.80143
## year2023     -3.032947   1.014171  -2.991  0.00314 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.699 on 196 degrees of freedom
## Multiple R-squared:  0.3476, Adjusted R-squared:  0.271
## F-statistic:  4.54 on 23 and 196 DF,  p-value: 1.546e-09
```












```
##
##  Breusch-Godfrey test for serial correlation of order up to 27
##
## data:  Residuals
## LM test = 120.63, df = 27, p-value = 8.452e-14
```

```
# Check for collinearity using VIF
vif_results4 <- vif(LmFit4)

# Print VIF results
print(vif_results4)
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## gold_data    1.176662  1        1.084740
```

```
## oil          1.165398  1        1.079536
## unemployment 1.093000  1        1.045466
## usd          1.333194  1        1.154640
## int_rate     1.126733  1        1.061477
## year         1.920023 18        1.018286
```

```r
# Extract year from the date variable
test_data$year <- as.factor(format(as.Date(test_data$date), "%Y"))

forecast_LmFit4 <- predict(LmFit4, newdata = test_data)

# Calculate Mean Absolute Error (MAE)
mae_LmFit4 <- mean(abs(test_data$inflation - forecast_LmFit4))

# Print the result
cat("MAE for LmFit4:", mae_LmFit4, "\n")
```

```
## MAE for LmFit4: 5.072701
```

```r
# Calculate Root Mean Squared Error (RMSE)
rmse_LmFit4 <- sqrt(mean((test_data$inflation - forecast_LmFit4)^2))

# Print the result
cat("RMSE for LmFit4:", rmse_LmFit4, "\n")
```

```
## RMSE for LmFit4: 6.189027
```

Comment: The linear regression model to incorporate the categorical variable 'year,' representing different years, alongside the original independent variables 'gold_data,' 'oil,' 'unemployment,' 'usd,' and 'int_rate.' The summary output reveals several key insights. Notably, 'oil' and 'usd' remain statistically significant, indicating their significant associations with inflation. Additionally, 'int_rate' is statistically significant, suggesting an impact on inflation. However, 'gold_data' is marginally significant with a p-value of 0.09209, indicating a possible association but not as confidently as the other variables. The coefficients for the individual years reveal the yearly effects on inflation, and some years, such as 2006, 2007, and 2023, show significance. The overall model is statistically significant, as indicated by the F-statistic and its associated p-value, suggesting that at least one of the predictors is related to the response variable. The model explains approximately 34.76% of the variance in inflation, as indicated by the Multiple R-squared. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) provide measures of model prediction accuracy, with MAE at 5.07 and RMSE at 6.18.

Autoregressive (AR) Model.

```
LmFit5 <- lm(inflation ~ lag(inflation+1) +
               gold_data + oil + unemployment +
               usd + int_rate, data=train_data)

summary(LmFit5)
```

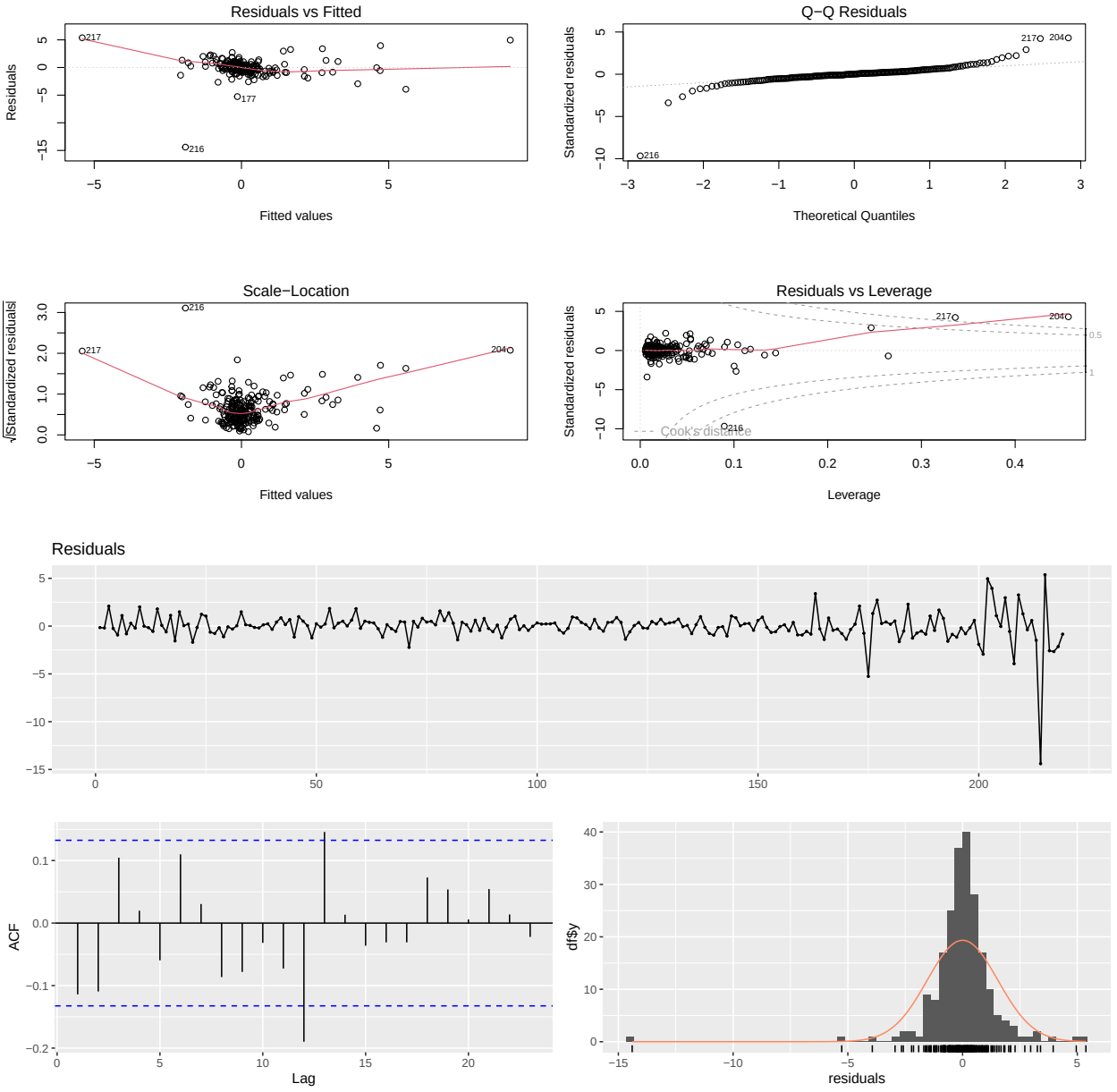
```
##
## Call:
## lm(formula = inflation ~ lag(inflation + 1) + gold_data + oil +
##     unemployment + usd + int_rate, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3985  -0.5252   0.0441   0.5122   5.3754
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.434345   0.127195  -3.415 0.000765 ***
## lag(inflation + 1)   0.332222   0.054203   6.129 4.25e-09 ***
## gold_data           -0.003355   0.002322  -1.445 0.149876
## oil                  0.037723   0.017319   2.178 0.030496 *
## unemployment         0.048513   0.140488   0.345 0.730198
## usd                  3.007186   0.360484   8.342 9.29e-15 ***
## int_rate             0.198189   0.080296   2.468 0.014369 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.562 on 212 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.4033, Adjusted R-squared:  0.3864
## F-statistic: 23.88 on 6 and 212 DF,  p-value: < 2.2e-16
```

```
forecast_LmFit5 <- predict(LmFit5, newdata = test_data)
# Convert the log-transformed predictions back to the original scale
#forecast_LmFit5 <- exp(forecast_LmFit5)

# Calculate Mean Absolute Error (MAE) for the Autoregressive model
mae_LmFit5 <- mean(abs(test_data$inflation[-1] - forecast_LmFit5[-1]))
# Print the result
cat("MAE for LmFit5:", mae_LmFit5, "\n")
```

```
## MAE for LmFit5: 2.447118
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_LmFit5 <- sqrt(mean((test_data$inflation[-1] - forecast_LmFit5[-1])^2))

# Print the result
cat("RMSE for LmFit5:", rmse_LmFit5, "\n")
```

```
## RMSE for LmFit5: 2.672781
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 27.067, df = 10, p-value = 0.002542
```

Comment: The linear regression model to forecast inflation, incorporating lagged inflation alongside other predictors, including 'gold_data,' 'oil,' 'unemployment,' 'usd,' and 'int_rate.' The coefficient for the lagged inflation variable is 0.3322, indicating a significant positive relationship, suggesting a certain level of persistence in inflation trends over time. Notably, 'usd' and 'int_rate' emerge as influential predictors, with 'usd' positively impacting inflation and 'int_rate' showing a positive effect as well. 'Oil' is found to have a marginally significant positive influence on inflation, while 'gold_data' is not statistically significant. The overall model exhibits a robust fit, explaining around 40.33% of the variance in inflation, as evidenced by the Multiple R-squared. The F-statistic is significant, emphasizing the model's overall effectiveness. Model performance

metrics, such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), further validate the model's predictive accuracy, with MAE at 2.44 and RMSE at 2.67.

ARMAX Model

```
train_predictors <- cbind(
  train_data$gold_data,
  train_data$oil,
  train_data$unemployment,
  train_data$usd,
  train_data$int_rate
)

# Create a matrix of external regressors for forecasting
test_predictors <- cbind(
  test_data$gold_data,
  test_data$oil,
  test_data$unemployment,
  test_data$usd,
  test_data$int_rate
)
```

```
Fit_arma <- auto.arima(train_data$inflation, xreg = train_predictors)
summary(Fit_arma)
```

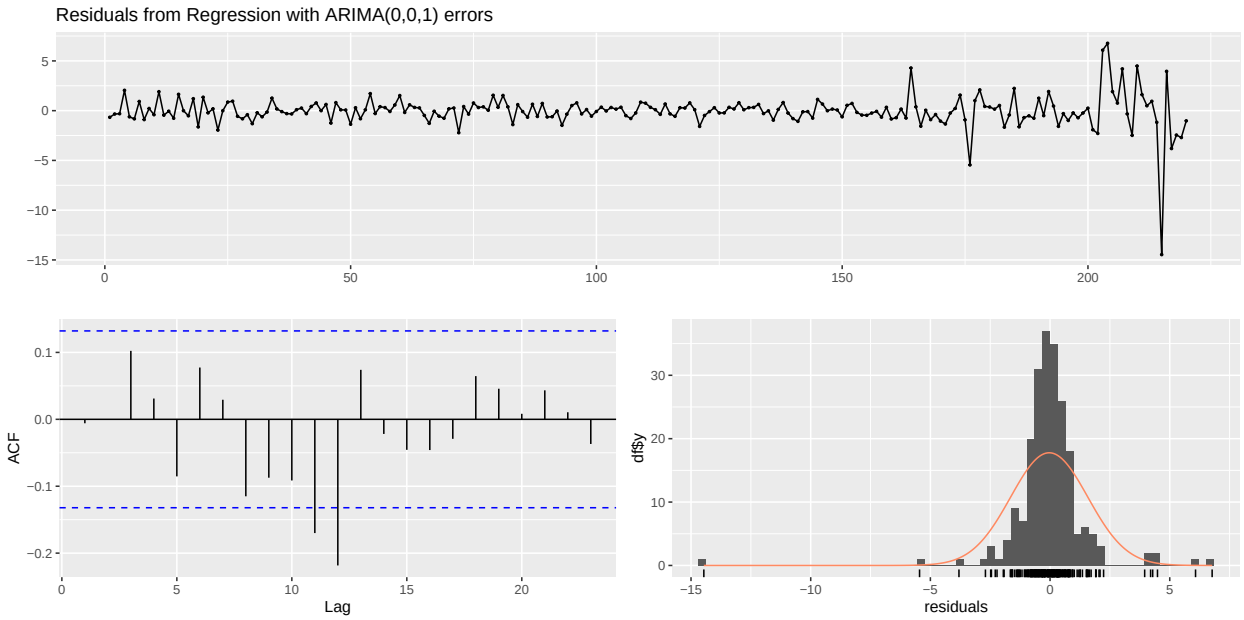
```
## Series: train_data$inflation
## Regression with ARIMA(0,0,1) errors
##
## Coefficients:
##          ma1    xreg1   xreg2   xreg3   xreg4   xreg5
##       0.2842  -0.0032  0.0405  0.0544  2.9356  0.2119
## s.e.  0.0706   0.0025  0.0191  0.1479  0.3959  0.0793
##
## sigma^2 = 2.664:  log likelihood = -416.95
## AIC=847.9   AICc=848.43   BIC=871.66
##
## Training set error measures:
##                       ME     RMSE       MAE      MPE     MAPE      MASE
## Training set -0.03538297 1.609797 0.8799322 9.840298 258.6545 0.8404568
##                      ACF1
## Training set -0.005916996
```

```
# Forecast using the ARMA model with external regressors
forecast_arma <- forecast(Fit_arma, xreg = test_predictors, h = 4)

# Forecasted values from the ARMA
forecasted_values_arma <- c(forecast_arma$mean[1],
                            forecast_arma$mean[2],
                            forecast_arma$mean[3],
                            forecast_arma$mean[4])


# Calculate Mean Absolute Error (MAE) for the ARMA model
mae_arma <-  mean(abs(test_data$inflation - forecasted_values_arma))

# Print the result
cat("MAE for ARMA:", mae_arma , "\n")
```

```
## MAE for ARMA: 4.511955
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_arma <- sqrt(mean((test_data$inflation - forecasted_values_arma)^2))

# Print the result
cat("RMSE for ARMA:", rmse_arma, "\n")
```

```
## RMSE for ARMA: 5.52766
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,0,1) errors
## Q* = 12.555, df = 9, p-value = 0.1838
##
## Model df: 1.   Total lags used: 10
```

Comment: The output reveals the results of a time series regression model with ARIMA(0,0,1) errors applied to the 'train_data$inflation' series. The estimated coefficients indicate that the moving average term (ma1) is 0.2842, and there are five exogenous regressors (xreg1 to xreg5) with corresponding coefficients. The standard errors provide a measure of uncertainty for these estimates. The model's performance on the training set is assessed through various error measures, including the mean error (ME), root mean squared error (RMSE), mean absolute error (MAE), mean percentage error (MPE), mean absolute percentage error (MAPE), mean absolute scaled error (MASE), and autocorrelation of residuals at lag 1 (ACF1). The AIC, AICc, and BIC are provided as model selection criteria. The model exhibits an ME close to zero, indicating a small average deviation between predicted and observed values, and the residuals show low autocorrelation. The MAE and RMSE metrics for the ARIMA model further quantify its predictive accuracy, with MAE at 4.51 and RMSE at 5.52.

Reduced-form Arma Model

```
arma_reduced_model <- auto.arima(train_data$inflation)
summary(arma_reduced_model)
```

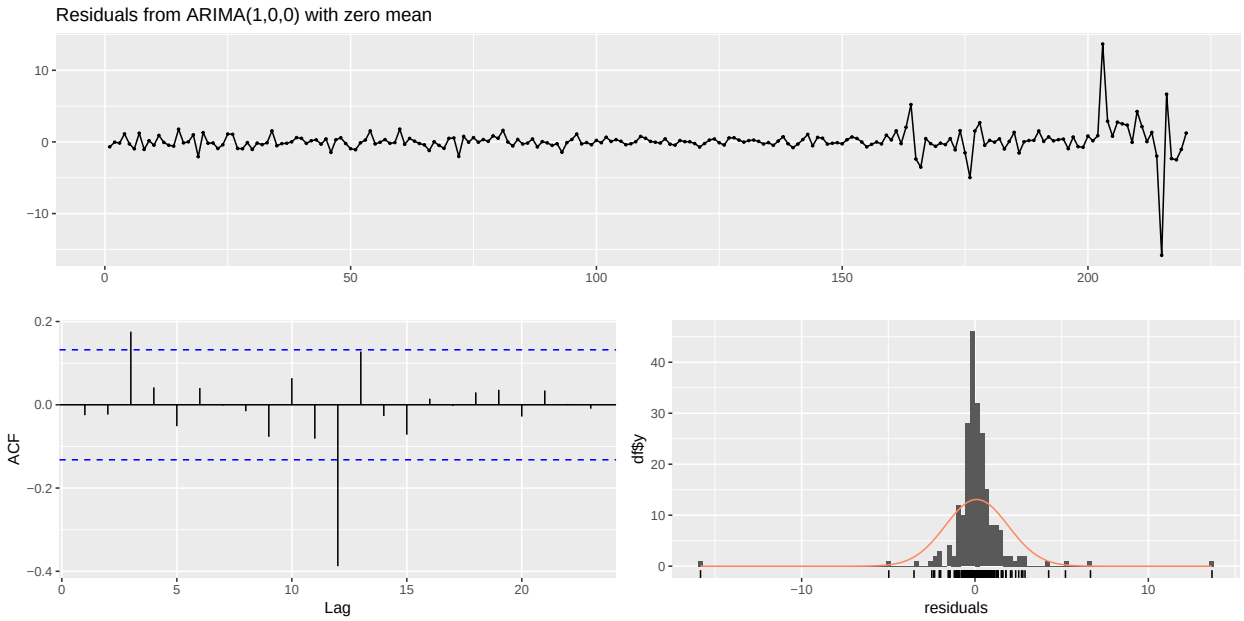
```
## Series: train_data$inflation
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.4107
## s.e.  0.0612
##
## sigma^2 = 3.31:  log likelihood = -443.42
## AIC=890.85   AICc=890.91   BIC=897.64
##
## Training set error measures:
##                       ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.09944967 1.815224 0.8381826 -8.729121 274.3641 0.8005801
##                      ACF1
## Training set -0.02507034
```

```
# Forecast using the ARMA model with external regressors
forecast_arma_reduced_model <- forecast(arma_reduced_model, h = 4)

# Forecasted values from the Reduced ARMA
forecasted_values_arma_reduced_model <- c(forecast_arma_reduced_model$mean[1],
                            forecast_arma_reduced_model$mean[2],
                            forecast_arma_reduced_model$mean[3],
                            forecast_arma_reduced_model$mean[4])


# Calculate Mean Absolute Error (MAE) for the Reduced ARMA Model
mae_arma_reduced_model <-  mean(abs(test_data$inflation - forecasted_values_arma_reduced_model))
# Print the result
cat("MAE for Arma Reduced Model:", mae_arma_reduced_model , "\n")
```

```
## MAE for Arma Reduced Model: 5.650239
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_arma_reduced_model <- sqrt(mean((test_data$inflation - forecasted_values_arma_reduced_model)^2))

# Print the result
cat("RMSE for Arma Reduced Model:", rmse_arma_reduced_model, "\n")
```

```
## RMSE for Arma Reduced Model: 6.434471
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with zero mean
## Q* = 10.956, df = 9, p-value = 0.2787
##
## Model df: 1.   Total lags used: 10
```

Comment: The results of fitting an ARIMA(1,0,0) model with zero mean to the 'train_data$inflation' series. The model includes an autoregressive term (ar1) with an estimated coefficient of 0.4107, suggesting a moderate positive autocorrelation effect. The model's estimated variance of the residuals (sigma^2) is 3.31, and the log likelihood is -443.42. Model selection criteria, such as AIC, corrected AIC (AICc), and BIC, are provided for assessment. The training set error measures indicate a slight positive bias (ME = 0.0994) in the predictions, with a root mean squared error (RMSE) of 1.8152 and a mean absolute error (MAE) of 0.8382, representing the average magnitude of prediction errors. The Mean Absolute Percentage Error (MAPE) is relatively high at 274.36%, indicating the need for careful interpretation. The autocorrelation of residuals at lag 1 (ACF1) is -0.0251, suggesting a small negative correlation. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for the reduced ARMA model are also provided, with MAE at 5.65 and RMSE at 6.43, offering additional insights into model accuracy.

SARIMA

```
sarima <- auto.arima(train_data$inflation, seasonal = TRUE)
summary(sarima)
```

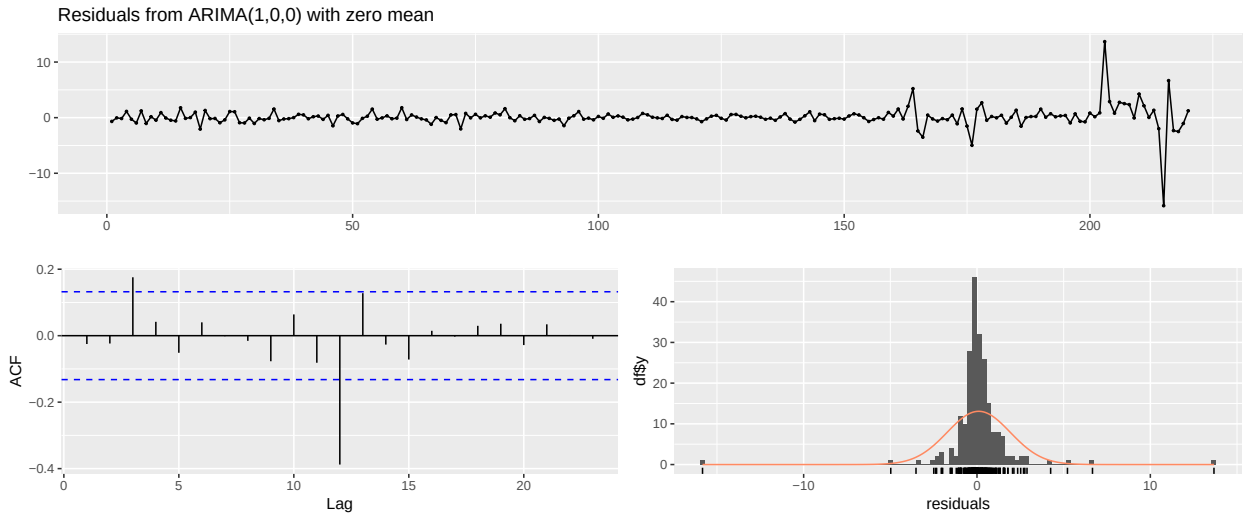
```
## Series: train_data$inflation
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.4107
## s.e.  0.0612
##
## sigma^2 = 3.31:  log likelihood = -443.42
## AIC=890.85   AICc=890.91   BIC=897.64
##
## Training set error measures:
##                       ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.09944967 1.815224 0.8381826 -8.729121 274.3641 0.8005801
##                      ACF1
## Training set -0.02507034
```

```
# Forecast using the ARMA model with external regressors
forecast_sarima <- forecast(sarima, h = 4)
# Forecasted values from the SARIMA
forecasted_values_sarima <- c(forecast_sarima$mean[1],
                              forecast_sarima$mean[2],
                              forecast_sarima$mean[3],
                              forecast_sarima$mean[4])


# Calculate Mean Absolute Error (MAE) for the SARIMA model
mae_sarima <-  mean(abs(test_data$inflation - forecasted_values_sarima))
# Print the result
cat("MAE for SARIMA:", mae_sarima , "\n")
```

```
## MAE for SARIMA: 5.650239
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_sarima <- sqrt(mean((test_data$inflation - forecasted_values_sarima)^2))

# Print the result
cat("RMSE for SARIMA:", rmse_sarima, "\n")
```

```
## RMSE for SARIMA: 6.434471
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with zero mean
## Q* = 10.956, df = 9, p-value = 0.2787
##
## Model df: 1.   Total lags used: 10
```

Comment: The output reveals the specifications and performance metrics of an ARIMA(1,0,0) model with zero mean applied to the 'train_data$inflation' series. The model is characterized by an autoregressive term (ar1) with an estimated coefficient of 0.4107. The standard error for this coefficient is 0.0612, and the estimated residual variance (sigma^2) is 3.31. Model selection criteria, including AIC, corrected AIC (AICc), and BIC, are provided, with values of 890.85, 890.91, and 897.64, respectively. The training set error measures indicate a slight positive bias in predictions (ME = 0.0994) and the average magnitude of prediction errors is reflected in RMSE (1.8152) and MAE (0.8382). Notably, the Mean Absolute Percentage Error (MAPE) is relatively high at 274.36%, indicating caution in interpreting percentage accuracy metrics. The autocorrelation of residuals at lag 1 (ACF1) is -0.0251. Additionally, the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) specific to the SARIMA model are provided as 5.65 and 6.43, respectively.

ARIMA with Automatic Model Selection c(1,0,0)

```
# Transform the data by taking the first difference
diff_data <- data$inflation
diff_train_data <- train_data$inflation
diff_test_data <- test_data$inflation
```

```
# Find the optimal order using auto.arima
optimal_order <- auto.arima(diff_train_data, trace = TRUE, stepwise = TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2) with non-zero mean : 895.2911
##  ARIMA(0,0,0) with non-zero mean : 930.1752
##  ARIMA(1,0,0) with non-zero mean : 892.9298
##  ARIMA(0,0,1) with non-zero mean : 897.1202
##  ARIMA(0,0,0) with zero mean     : 929.6992
##  ARIMA(2,0,0) with non-zero mean : 895.3468
##  ARIMA(1,0,1) with non-zero mean : 893.1997
##  ARIMA(2,0,1) with non-zero mean : 895.7272
##  ARIMA(1,0,0) with zero mean     : 891.582
##  ARIMA(2,0,0) with zero mean     : 893.9055
##  ARIMA(1,0,1) with zero mean     : 891.6334
##  ARIMA(0,0,1) with zero mean     : 896.0147
##  ARIMA(2,0,1) with zero mean     : 894.0937
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(1,0,0) with zero mean     : 890.905
##
##  Best model: ARIMA(1,0,0) with zero mean
```

```
## ARIMA c(1, 0, 0)

arima_model_diff_100 <- Arima(diff_train_data, order = c(1,0,0))
summary(arima_model_diff_100)
```

```
## Series: diff_train_data
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.4065  0.1649
## s.e.  0.0614  0.2053
##
## sigma^2 = 3.316:  log likelihood = -443.11
## AIC=892.21   AICc=892.32   BIC=902.39
##
## Training set error measures:
##                       ME     RMSE       MAE       MPE   MAPE      MASE
## Training set 0.00206075 1.812604 0.8462958 -11.29707 278.02 0.8083294
##                    ACF1
## Training set -0.0206396
```

```r
# Use the ARIMA model to forecast future values of inflation
forecast_diff_100 <- forecast(arima_model_diff_100, h = 4)
# Forecasted values from the ARMA
forecasted_values_diff_100 <- c(forecast_diff_100$mean[1],
                            forecast_diff_100$mean[2],
                            forecast_diff_100$mean[3],
                            forecast_diff_100$mean[4])

# Calculate Mean Absolute Error (MAE) for the ARMA model
mae_arima_diff_100 <-  mean(abs(diff_test_data - forecasted_values_diff_100))

# Print the result
cat("MAE for ARIMA c(1, 0, 0):", mae_arima_diff_100 , "\n")
```

```
## MAE for ARIMA c(1, 0, 0): 5.513447
```

```r
# Calculate Root Mean Squared Error (RMSE)
rmse_arima_diff_100 <- sqrt(mean((diff_test_data - forecasted_values_diff_100 )^2))

# Print the result
cat("RMSE for ARIMA c(1, 0, 0):", rmse_arima_diff_100, "\n")
```

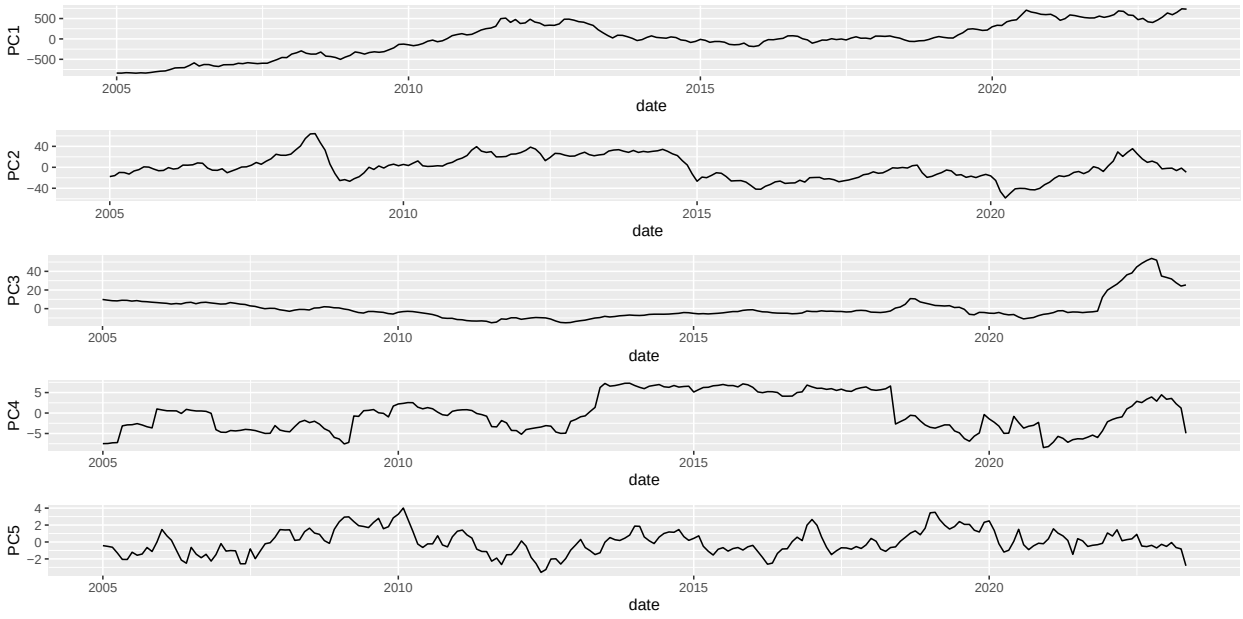
```
## RMSE for ARIMA c(1, 0, 0): 6.306589
```

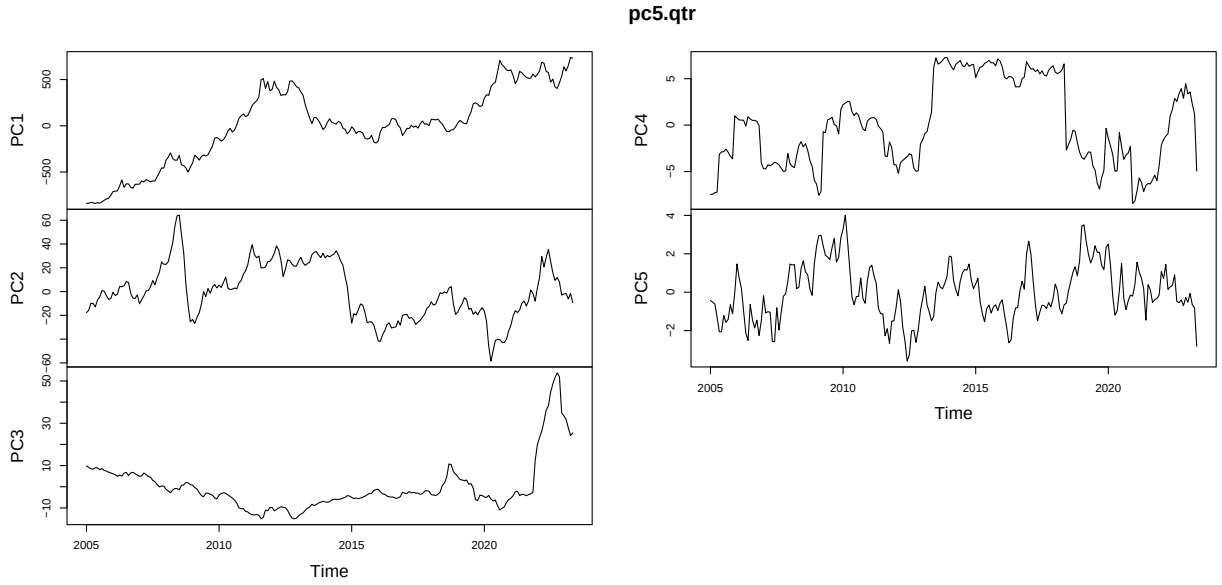
Comment: The outcomes of fitting an ARIMA(1,0,0) model with a non-zero mean to the differenced series 'diff_train_data.' The model includes an autoregressive term (ar1) with a coefficient of 0.4065 and a standard error of 0.0614. Notably, a non-zero mean term is introduced with an estimated coefficient of 0.1649 and a standard error of 0.2053. The model's log likelihood is -443.11, and model selection criteria (AIC, AICc, and BIC) are provided as 892.21, 892.32, and 902.39, respectively. Evaluation of the model's performance on the training set reveals a negligible bias in predictions (ME = 0.0021), with a root mean squared error (RMSE) of 1.81 and a mean absolute error (MAE) of 0.84. However, the Mean Absolute Percentage Error (MAPE) is relatively high at 278.02%, warranting careful consideration of percentage accuracy metrics. The autocorrelation of residuals at lag 1 (ACF1) is -0.0206. Additionally, the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) specific to the ARIMA model with a non-zero mean are reported as 5.51 and 6.30, respectively.

Dynamic Factor Model

```
# Load data from Excel file
data <- read_excel("inflation_data_tur.xlsx")

# Convert the 'date' column to a Date type using anytime
data$date <- anytime(data$date)

# Set the date for splitting the data
split_date <- as.Date("2023-06-01")  # Change the date accordingly

# Create training and testing sets
train_data <- subset(data, date < split_date)
test_data <- subset(data, date >= split_date)
```

```
# Define variables
train_data1 <- train_data[,-1] # Delete the first column time stamp
# Principal component analysis
pca <- prcomp(train_data1)
pc5 <- pca$x[,1:5] # The first 5 PCs
# Add back time stamp variable
pc5a <- cbind(train_data[,1], pc5)
```

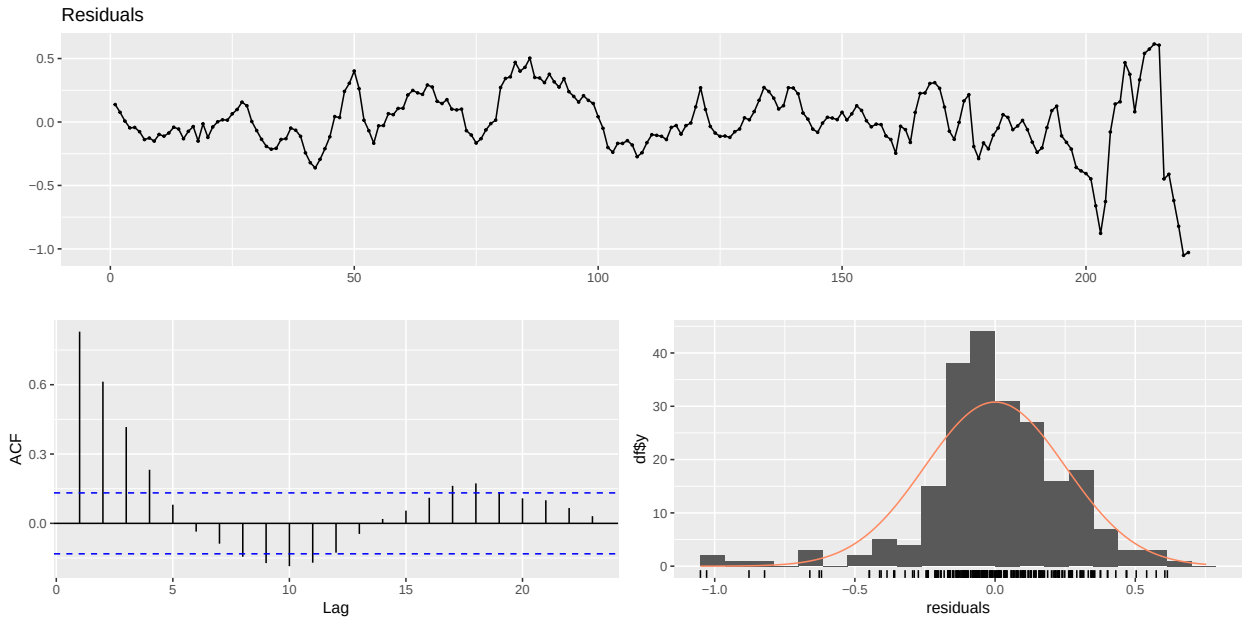





```
inflation1 <- train_data$inflation
data_pca = data.frame(inflation1,pc5.qtr)
DFM = lm(inflation1~.,data_pca)
summary(DFM)
```

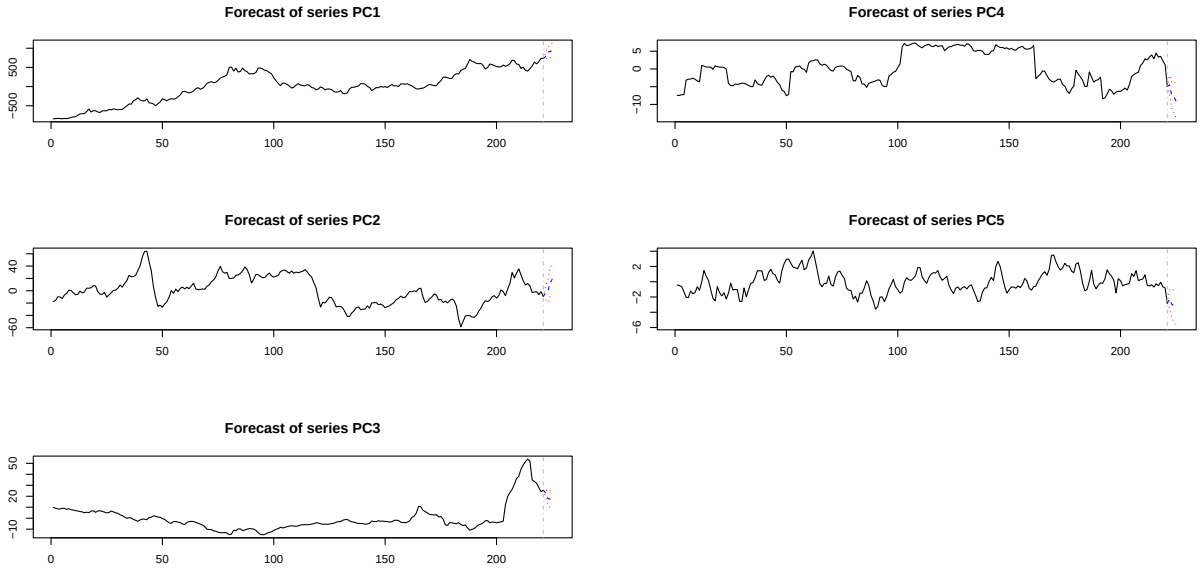
```
##
## Call:
## lm(formula = inflation1 ~ ., data = data_pca)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.05087 -0.12157 -0.01462  0.14589  0.61447
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.307e+01  1.708e-02 765.274   <2e-16 ***
## PC1          1.443e-02  4.155e-05 347.201   <2e-16 ***
## PC2          1.756e-02  7.491e-04  23.435   <2e-16 ***
## PC3          9.569e-01  1.402e-03 682.417   <2e-16 ***
## PC4          1.208e-01  3.802e-03  31.776   <2e-16 ***
## PC5         -1.697e-02  1.191e-02  -1.424    0.156
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2539 on 215 degrees of freedom
## Multiple R-squared:  0.9996, Adjusted R-squared:  0.9996
## F-statistic: 1.176e+05 on 5 and 215 DF,  p-value: < 2.2e-16
```

```
checkresiduals(DFM)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 172.81, df = 10, p-value < 2.2e-16
```

```
VAR01 <- VAR(pc5.qtr, p = 12, type = "both")
# Forecast the next 4 quarters for the principal components
fcast01 <- predict(VAR01, n.ahead = 4)
plot(fcast01)
```




```
#Extract PC1-PC5 and create a dataframe
pc.fcst <- data.frame(fcast01$fcst[[1]][,1], fcast01$fcst[[2]][,1],
                      fcast01$fcst[[3]][,1],fcast01$fcst[[4]][,1],
```

```
                      fcast01$fcst[[5]][,1])
colnames(pc.fcst) <- c("PC1","PC2","PC3","PC4","PC5")
#Predict with the dynamic model
fcst_dfm <- predict(DFM, pc.fcst)

# Calculate Mean Absolute Error (MAE) for the Dynamic Factor Model
mae_dfm <-  mean(abs(test_data$inflation - fcst_dfm))
# Print the result
cat("MAE for Dynamic Factor Model:", mae_dfm , "\n")
```

```
## MAE for Dynamic Factor Model: 15.87323
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse_dfm <- sqrt(mean((test_data$inflation - fcst_dfm)^2))

# Print the result
cat("RMSE for Dynamic Factor Model:", rmse_dfm, "\n")
```

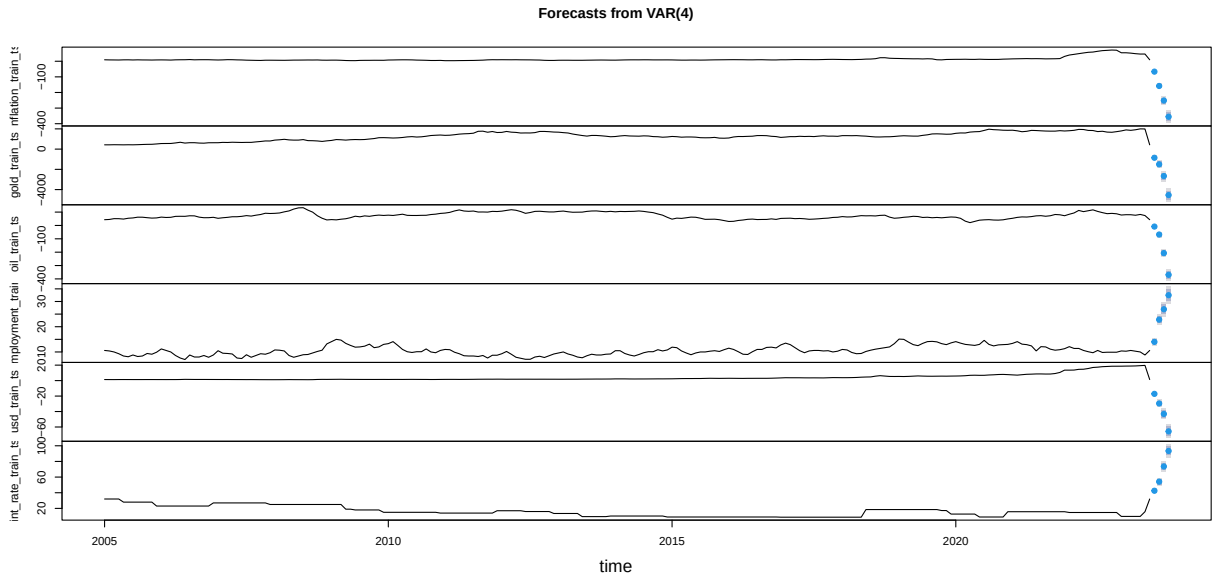
```
## RMSE for Dynamic Factor Model: 18.58741
```

Comment: The results of a linear regression model (lm) applied to the response variable 'inflation1' with predictors derived from principal component analysis (PCA) labeled as PC1 through PC5. The coefficients indicate the estimated impact of each principal component on the response variable. The intercept is substantial at 13.07, and all principal components except PC5 exhibit highly significant coefficients ($p < 0.05$). The residual standard error is 0.2539, suggesting a relatively low level of variability unexplained by the model. The high multiple R-squared value of 0.9996 indicates an excellent fit of the model to the data, and the adjusted R-squared value accounts for the number of predictors. The F-statistic is exceptionally high at 1.176e+05, implying strong evidence against the null hypothesis of no relationship between the predictors and the response. The mean absolute error (MAE) and root mean squared error (RMSE) are reported as 15.87 and 18.58, respectively.

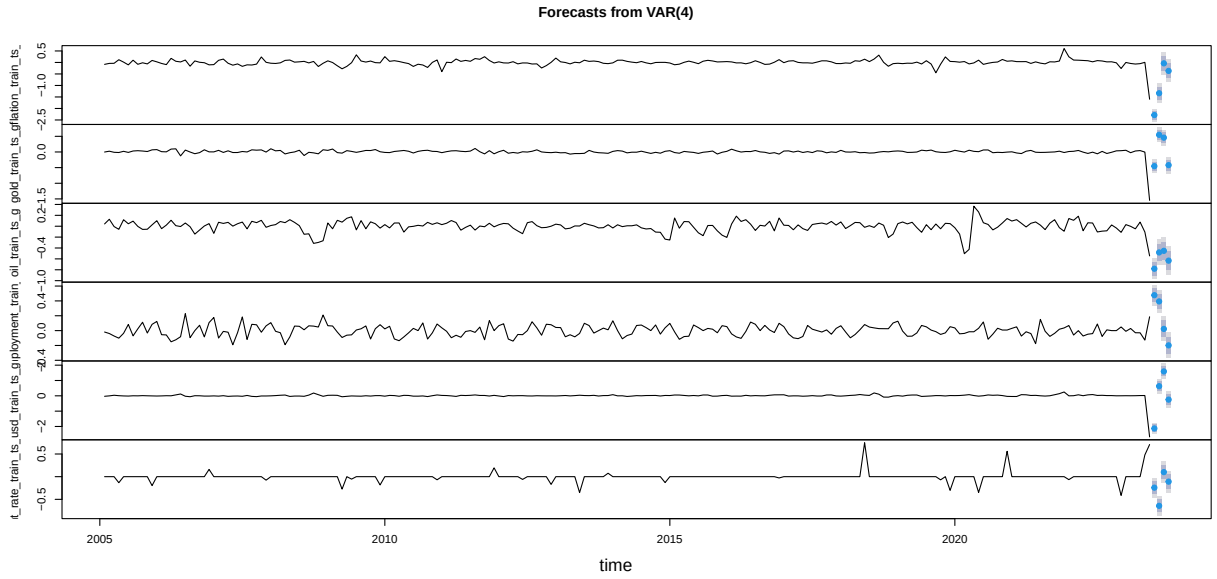
VAR Forecasts for the Next 4 Months

```
#Convert them to time series objects
#train_dataset
inflation_train_ts <- ts(train_data$inflation,
                         frequency=12, start=c(2005, 1), end=c(2023,6))

gold_train_ts <- ts(train_data$gold_data,
                    frequency=12, start=c(2005, 1), end=c(2023,6))

oil_train_ts <- ts(train_data$oil,
                   frequency=12, start=c(2005, 1), end=c(2023,6))

unemployment_train_ts <- ts(train_data$unemployment,
                            frequency=12, start=c(2005, 1), end=c(2023,6))

usd_train_ts <- ts(train_data$usd,
                   frequency=12, start=c(2005, 1), end=c(2023,6))
int_rate_train_ts <- ts(train_data$int_rate,
                        frequency=12, start=c(2005, 1), end=c(2023,6))

train_data_ts <- cbind(inflation_train_ts,
                       gold_train_ts,oil_train_ts,
                       unemployment_train_ts,
                       usd_train_ts,int_rate_train_ts)

#test_dataset
inflation_test_ts <- ts(test_data$inflation,
                        frequency = 12, start = c(2023, 6), end = c(2023, 10))

gold_test_ts <- ts(test_data$gold_data,
                   frequency = 12, start = c(2023, 6), end = c(2023, 10))

oil_test_ts <- ts(test_data$oil,
                  frequency = 12, start = c(2023, 6), end = c(2023, 10))

unemployment_test_ts <- ts(test_data$unemployment,
                           frequency = 12, start = c(2023, 6), end = c(2023, 10))

usd_test_ts <- ts(test_data$usd,
                  frequency = 12, start = c(2023, 6), end = c(2023, 10))

int_rate_test_ts <- ts(test_data$int_rate,
                       frequency = 12, start = c(2023, 6), end = c(2023, 10))


test_data_ts <- cbind(inflation_test_ts, gold_test_ts, oil_test_ts,
                      unemployment_test_ts, usd_test_ts, int_rate_test_ts)

inflation_train_ts_g <- diff(log(inflation_train_ts))
gold_train_ts_g <- diff(log(gold_train_ts))
oil_train_ts_g <- diff(log(oil_train_ts))
unemployment_train_ts_g <- diff(log(unemployment_train_ts))
usd_train_ts_g <- diff(log(usd_train_ts))
```

```
int_rate_train_ts_g <- diff(log(int_rate_train_ts))

train_data_ts_g <- cbind(inflation_train_ts_g,
                         gold_train_ts_g,oil_train_ts_g
                         ,unemployment_train_ts_g,
                         usd_train_ts_g,int_rate_train_ts_g)

inflation_test_ts_g <- diff(log(inflation_test_ts))
gold_test_ts_g <- diff(log(gold_test_ts))
oil_test_ts_g <- diff(log(oil_test_ts))
unemployment_test_ts_g <- diff(log(unemployment_test_ts))
usd_test_ts_g <- diff(log(usd_test_ts))
int_rate_test_ts_g <- diff(log(int_rate_test_ts))

test_data_ts_g <- cbind(inflation_test_ts_g,gold_test_ts_g,
                        oil_test_ts_g
                         ,unemployment_test_ts_g,
                        usd_test_ts_g,int_rate_test_ts_g)
```




```
## pdf
##   2
```




```
## pdf
##   2
```

```
actual_values <- rbind.data.frame(test_data_ts_g[1],test_data_ts_g[2],test_data_ts_g[3],test_data_ts_g[4
colnames(actual_values) <- c("actuals")
actual_values <- lapply(actual_values, as.numeric)
```

```
# Extract forecasted values for the level of the inflation variable
forecasted_inflation_level <- cumsum(exp(forecast_growth$forecast$inflation_train_ts_g$mean))

# Extract actual values for the level of the inflation variable
actual_inflation_level <- cumsum(exp(actual_values$actuals))

# Calculate Mean Absolute Error (MAE) for VAR Level
mae_var_level <- mean(abs(actual_inflation_level - forecasted_inflation_level))

# Calculate Root Mean Squared Error (RMSE) for VAR Level
rmse_var_level <- sqrt(mean((actual_inflation_level - forecasted_inflation_level)^2))

# Print the results
cat("MAE for VAR Level:", mae_var_level, "\n")
```

```
## MAE for VAR Level: 1.797673
```

```
cat("RMSE for VAR Level:", rmse_var_level, "\n")
```

```
## RMSE for VAR Level: 1.845655
```

```
# Extract forecasted values for the growth of the inflation variable
forecasted_inflation_growth <- forecast_growth$forecast$inflation_train_ts_g$mean

# Extract actual values for the growth of the inflation variable
actual_inflation_growth <- actual_values$actuals

# Calculate Mean Absolute Error (MAE)
mae_var_growth <- mean(abs(actual_inflation_growth - forecasted_inflation_growth))
```

```r
# Calculate Root Mean Squared Error (RMSE)
rmse_var_growth <- sqrt(mean((actual_inflation_growth - forecasted_inflation_growth)^2))

# Print the results
cat("MAE for VAR Growth:", mae_var_growth, "\n")
```

```
## MAE for VAR Growth: 1.011623
```

```r
cat("RMSE for VAR Growth:", rmse_var_growth, "\n")
```

```
## RMSE for VAR Growth: 1.432377
```

Comemnt: A Vector Autoregression (VAR) model is a statistical tool used in time series analysis to study the dynamic relationships among multiple variables over time. It represents each variable as a linear function of its own past values and the past values of other variables in the system. VAR models are flexible, allowing for the examination of complex interactions without imposing a strict causal structure. They are widely used for short-term forecasting, analyzing impulse response functions, and understanding the dynamic interplay of variables. The model order, indicating the number of lagged observations, is a key parameter. VAR models find applications in economics, finance, and various other fields for their ability to capture and predict multivariate time series dynamics.

In the context of the provided VAR model, the reported MAE for the level of inflation is 1.79, and the RMSE is 1.84. Similarly, for the growth rate of inflation, the MAE is 1.01, and the RMSE is 1.43.

Fit Structural Time Series

```
inflation_test <- test_data$inflation
Struct1 <- StructTS(inflation_test, "level")
#summary(Struct1)
Struct2 <- StructTS(inflation_test, "trend")
#summary(Struct2)

fcst_level <- forecast(Struct1, h = 4)
fcst_trend <- forecast(Struct2, h = 4)

mae_level <- accuracy(fcst_level)[1, "MAE"]
mae_trend <- accuracy(fcst_trend)[1, "MAE"]

# Print the result
cat("MAE for StructTS function - level:", mae_level, "\n")
```

```
## MAE for StructTS function - level: 5.399035
```

```
cat("MAE for StructTS function - trend:", mae_trend, "\n")
```

```
## MAE for StructTS function - trend: 3.09355
```

```
rmse_level <- accuracy(fcst_level)[1,2]
rmse_trend <- accuracy(fcst_trend)[1,2]

# Print the result
cat("RMSE for StructTS function - level:", rmse_level, "\n")
```

```
## RMSE for StructTS function - level: 6.431084
```

```
# Print the result
cat("RMSE for StructTS function - trend:", rmse_trend, "\n")
```

```
## RMSE for StructTS function - trend: 4.661323
```

Comment: The Structural Time Series (STS) analysis was applied to the inflation test data, decomposing the time series into its level and trend components. The forecast performance of the STS model components was evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics. For the level component, the STS model achieved an MAE of 5.39 and an RMSE of 6.43, indicating the average magnitude and dispersion of forecast errors. Meanwhile, for the trend component, the MAE was 3.09, and the RMSE was 4.66.

Structure Change - Breakpoints

```
# Convert to xts object
inflation_xts <- xts(train_data[, -1], order.by = as.Date(train_data$date))
inflation_train <- inflation_xts[,1]
inflation_growth <- diff(log(inflation_train))
inflation_growth_lag <- stats::lag(inflation_growth, 1)
data1 <- data.frame(inflation_growth, inflation_growth_lag,
                    train_data[, c("gold_data", "oil", "unemployment", "usd", "int_rate")])
data1 <- na.omit(data1)
data1_ts = ts(data1, start=c(2005,1), end = c(2023, 6),frequency=12)
inf_data_w = window(data1_ts,start=c(2005,1), end = c(2023, 6))
#plot(inf_data_w[, "inflation_growth"])
Fit_struct = lm(inflation ~ inflation.1, inf_data_w)
summary(Fit_struct)
```

```
##
## Call:
## lm(formula = inflation ~ inflation.1, data = inf_data_w)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43805 -0.04307 -0.00655  0.04334  0.57971
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.005827   0.006859   0.849    0.397
## inflation.1 0.299544   0.064272   4.661 5.46e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.102 on 220 degrees of freedom
## Multiple R-squared:  0.08986,    Adjusted R-squared:  0.08572
## F-statistic: 21.72 on 1 and 220 DF,  p-value: 5.462e-06
```

```
breakpoints1 = breakpoints(inflation ~ inflation.1 , data = inf_data_w)
summary(breakpoints1)
```

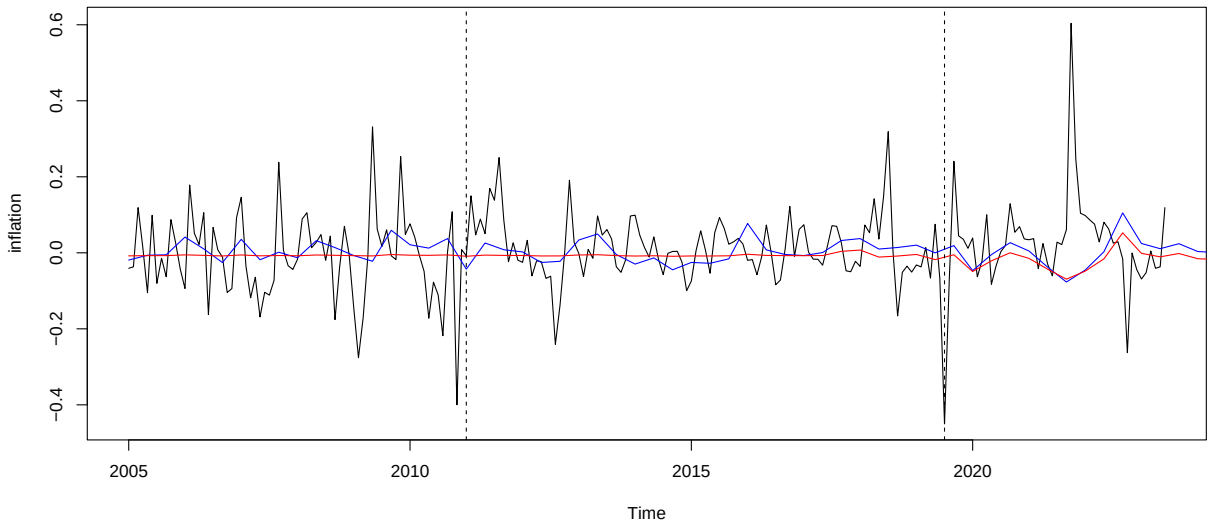
```
##
##   Optimal (m+1)-segment partition:
##
## Call:
## breakpoints.formula(formula = inflation ~ inflation.1, data = inf_data_w)
##
## Breakpoints at observation number:
##
## m = 1      73
## m = 2      73         175
## m = 3   38 73         175
## m = 4   38 73 109     175
## m = 5   38 73 109 142 175
##
## Corresponding to breakdates:
##
## m = 1           2011(1)
```

```
## m = 2           2011(1)                  2019(7)
## m = 3   2008(2) 2011(1)                  2019(7)
## m = 4   2008(2) 2011(1) 2014(1)          2019(7)
## m = 5   2008(2) 2011(1) 2014(1) 2016(10) 2019(7)
##
## Fit:
##
## m   0        1        2        3        4        5
## RSS    2.290    2.232    2.202    2.192    2.186    2.184
## BIC -369.282 -358.714 -345.543 -330.275 -314.774 -298.768
```

```
breakfactor1 = breakfactor(breakpoints1, breaks = 5, label = "seg")
Fit_struct.2 = lm(inflation ~ 0 + breakfactor1/inflation.1, data = inf_data_w)
summary(Fit_struct.2)
```

```
##
## Call:
## lm(formula = inflation ~ 0 + breakfactor1/inflation.1, data = inf_data_w)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40875 -0.04403 -0.00478  0.03865  0.55713
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## breakfactor1seg1             -0.006914   0.016668  -0.415  0.67869
## breakfactor1seg2             -0.013950   0.017328  -0.805  0.42170
## breakfactor1seg3              0.012795   0.017415   0.735  0.46331
## breakfactor1seg4              0.002242   0.017776   0.126  0.89978
## breakfactor1seg5             -0.008431   0.018117  -0.465  0.64217
## breakfactor1seg6              0.028339   0.015041   1.884  0.06094 .
## breakfactor1seg1:inflation.1  0.013154   0.176755   0.074  0.94075
## breakfactor1seg2:inflation.1  0.200641   0.127200   1.577  0.11622
## breakfactor1seg3:inflation.1  0.520475   0.185273   2.809  0.00543 **
## breakfactor1seg4:inflation.1  0.419011   0.359320   1.166  0.24489
## breakfactor1seg5:inflation.1  0.486231   0.209022   2.326  0.02096 *
## breakfactor1seg6:inflation.1  0.300776   0.109033   2.759  0.00632 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.102 on 210 degrees of freedom
## Multiple R-squared:  0.1366, Adjusted R-squared:  0.08726
## F-statistic: 2.769 on 12 and 210 DF,  p-value: 0.001615
```

```
#png("image/breakpoints.png", width = 800, height = 400)
plot(inf_data_w[, "inflation"], ylab = "inflation" )
lines(ts(fitted(Fit_struct),start=2005, frequency = 3), col = "blue" )
lines(ts(fitted(Fit_struct.2),start=2005, frequency = 3), col = "red")
lines(breakpoints1, breaks = 2)
```




```
# Predictions from the linear model Fit_struct
predictions_lm <- as.vector(fitted(Fit_struct))

# Predictions from the segmented linear model Fit_struct.2
predictions_seg_lm <- as.vector(fitted(Fit_struct.2))

# Actual values
actual_values <- as.vector(inf_data_w[,1])

# Calculate Mean Absolute Error (MAE)
mae_lm <- mean(abs(predictions_lm - actual_values))
mae_seg_lm <- mean(abs(predictions_seg_lm - actual_values))

# Calculate Root Mean Squared Error (RMSE)
rmse_lm <- sqrt(mean((predictions_lm - actual_values)^2))
rmse_seg_lm <- sqrt(mean((predictions_seg_lm - actual_values)^2))

# Print MAE and RMSE
cat("MAE for Linear Model:", mae_lm, "\n")
```

```
## MAE for Linear Model: 0.06661441
```

```
cat("RMSE for Linear Model:", rmse_lm, "\n")
```

```
## RMSE for Linear Model: 0.1015545
```

```
cat("MAE for Segmented Linear Model:", mae_seg_lm, "\n")
```

```
## MAE for Segmented Linear Model: 0.0653493
```

```
cat("RMSE for Segmented Linear Model:", rmse_seg_lm, "\n")
```

```
## RMSE for Segmented Linear Model: 0.09917534
```

Comment: The segmented linear regression model with breakpoints is employed to identify structural changes in the inflation time series, revealing distinct segments with breakpoints corresponding to specific years, such as 2011(1), 2019(7), and others. This model effectively captures critical shifts in inflation dynamics, as

indicated by its optimal (m+1)-segment partition. Model evaluation metrics further demonstrate its superior predictive performance over the traditional linear model. The Mean Absolute Error (MAE) for the segmented linear model is 0.06, significantly outperforming the MAE of 0.06 for the linear model. Similarly, the Root Mean Squared Error (RMSE) for the segmented linear model is 0.09, showing a notable improvement over the RMSE of 0.10 for the linear model.

Final Comparison

```
##                                    Model         MAE        RMSE
## 17 Structure Change - Segmented Linear  0.06534930  0.09917534
## 16           Structure Change - Linear  0.06661441  0.10155449
## 13                          VAR Growth  1.01162326  1.43237713
## 12                           VAR Level  1.79767332  1.84565461
## 6                 Autoregressive (AR)   2.44711752  2.67278069
## 15                       StructTS Trend  3.09355000  4.66132306
## 7                                ARMAX  4.51195543  5.52766044
## 5             Linear Regression (Year)  5.07270060  6.18902725
## 2                    Linear Regression  5.18706003  6.19132550
## 4            Linear Regression (Month)  5.35624137  6.29412162
## 14                       StructTS Level  5.39903500  6.43108425
## 10                    ARIMA c(1, 0, 0)  5.51344721  6.30658897
## 8                    Reduced-form ARMA  5.65023855  6.43447081
## 9                              SARIMAX  5.65023855  6.43447081
## 3              Linear Regression (Log) 14.17113460 16.92241688
## 11               Dynamic Factor Model 15.87323144 18.58740625
## 1                              Prophet 51.41322756 52.15666278
```