1. **Introduction**

The two-time series provides the Capacity Utilisation Rate (TCU) and Consumer Price Index Inflation Rate (CPIAUCSL). TCU represents the extent to which firms' production capacity is being used, while CPIAUCSL represents the changes in the cost of goods purchased in a household. TCU is measured as the ratio of actual output to potential output. The actual output is measured using industrial production data, while potential output is estimated using data on capacity constraints, including production capacity and capital stock. The output gap is then calculated as the difference between actual and potential output, and TCU is obtained by scaling the output gap by potential output. CPIAUCSL is measured as the percentage change in the Consumer Price Index (CPI) a year ago. CPI measures the average price level of a basket of household goods and services. The percentage change is obtained by calculating the difference between the current month's CPI and the CPI.

library(dynamac)

library(forecast)

library(tseries)

library(nlme)

library(pdfetch)

library(zoo)

library(urca)

library(vars)

library(car)

library(dynlm)

library(tsDyn)

library(gets)

library(readxl)

library(aod)

library(egcm)

library(aTSA)

library(tidyverse)

**Inflation data**

The CPI data is read from FRED using the pdfetch\_FRED() function from the quantmod package. The names() function assigns "CPI" to the resulting data. The inflation rate is calculated by taking the first difference of the logarithm of the CPI series, lagged by 12 periods (i.e., one year). This gives the percentage change in the CPI from one year ago, a standard inflation measure. The resulting series is multiplied by 100 to convert it from a decimal to a percentage. The names() function again assigns "Inflation" to the resulting data. The resulting inflation series is converted to a time series using the ts() function, with a start date of January 1947 and a frequency of 12 (monthly data). The na.omit() function is then used to remove any missing values from the series, which are present because the CPI data only began in January 1948.

CPI = pdfetch\_FRED("CPIAUCSL")

names(CPI) = "CPI"

Inflation = diff(log(CPI), lag = 12) \* 100

names(Inflation) = "Inflation"

Inflation = ts(Inflation, start=c(1947, 1), frequency=12)

Inflation = na.omit(Inflation)

Capacity utilization data:

TCU = pdfetch\_FRED("TCU")

names(TCU) = "TCU"

TCU = ts(TCU, start=c(1967, 1), frequency=12)

**How the Data is Measured**

The data consists of two-time series: the Consumer Price Index (CPI) and the Capacity Utilization Rate (CUR). The CPI measures the average price change over time of a fixed basket of household goods and services. It is calculated by the Bureau of Labor Statistics (BLS) in the United States and is widely used as a measure of Inflation. The CPI in this dataset is measured in levels and represents the monthly average price of the fixed basket of goods and services in the United States.

1. **Unit Root Tests**

# Plot TCU in levels

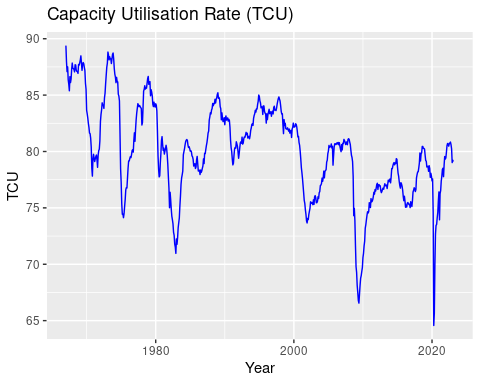
ggplot() +

geom\_line(data = TCU, aes(x = time(TCU), y = TCU), color = 'blue') +

labs(title = "Capacity Utilisation Rate (TCU)") +

ylab("TCU") +

xlab("Year")



**Lag Correlation in Levels and Lag Correlation First Difference**

# Compute lag correlations

lag\_cor\_levels <- acf(TCU, lag.max = 24, plot = FALSE)$acf

lag\_cor\_diff <- acf(diff(TCU), lag.max = 24, plot = FALSE)$acf

lag\_cor\_levels\_pacf <- pacf(TCU, lag.max = 24, plot = FALSE)

lag\_cor\_levpac <- pacf(diff(TCU), lag.max = 24, plot= FALSE)

# Plot the lag correlations

par(mfrow = c(2,1))

plot(lag\_cor\_levels, type = "h", main = "ACF Lag Correlation of TCU in Levels")

abline(h = 0, lty = 2)

plot(lag\_cor\_diff, type = "h", main = "ACF Lag Correlation of First Difference of TCU")

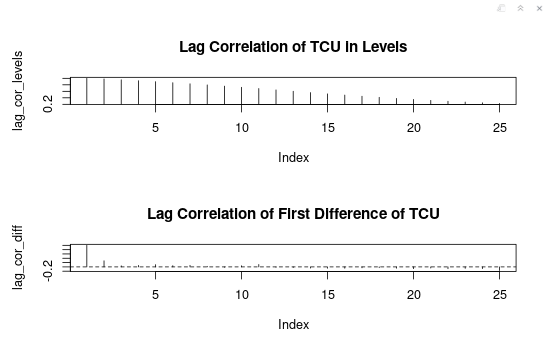
abline(h = 0, lty = 2)

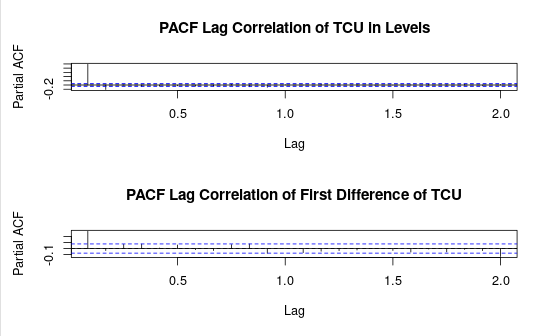
plot(lag\_cor\_levels\_pacf, main = "PACF Lag Correlation of TCU in Levels")

abline(h = 0, lty = 2)

plot(lag\_cor\_levpac, main = "PACF Lag Correlation of First Difference of TCU")

abline(h = 0, lty = 2)





**Consumer Price Index Inflation Rate (CPIAUCSL)**

# Plot CPIAUCSL in levels

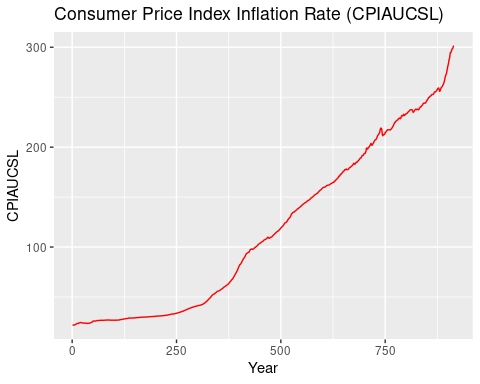
ggplot() +

geom\_line(data = CPI, aes(x = time(CPI), y = CPI), color = 'red') +

labs(title = "Consumer Price Index Inflation Rate (CPIAUCSL)") +

ylab("CPIAUCSL") +

xlab("Year")



**Capacity Utilization Rate (TCU) - First Differences**

# Plot TCU in first differences

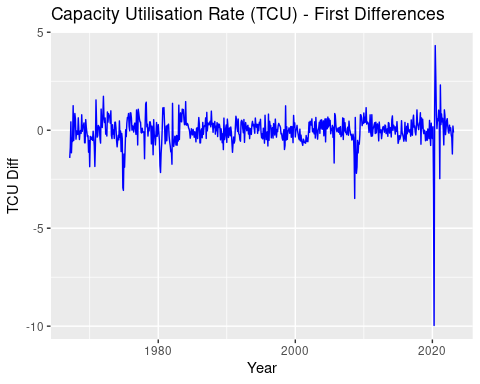
ggplot() +

geom\_line(data = diff(TCU), aes(x = time(TCU)[2:length(time(TCU))], y = diff(TCU)), color = 'blue') +

labs(title = "Capacity Utilisation Rate (TCU) - First Differences") +

ylab("TCU Diff") +

xlab("Year")



**Consumer Price Index Inflation Rate (CPIAUCSL)**

# Plot CPIAUCSL in first differences

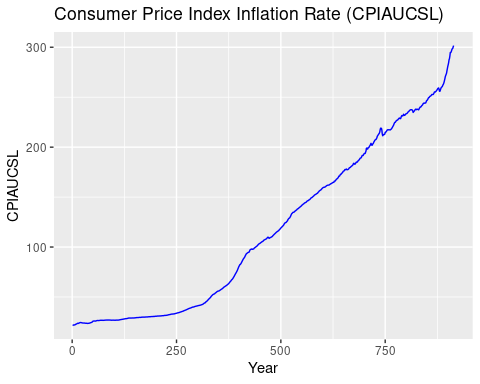
ggplot() +

geom\_line(data = CPI, aes(x = time(CPI), y = CPI), color = 'blue') +

labs(title = "Consumer Price Index Inflation Rate (CPIAUCSL)") +

ylab("CPIAUCSL") +

xlab("Year")



1. **Interactive Content**

**ADF, PP, and KPSS unit root tests.**

To run the ADF, PP, and KPSS unit root tests, we can use the ur.df function from the urca package. The ADF and PP tests test for a unit root in the series, while the KPSS test is used to test for stationarity. We need to run each test twice for each series, first in levels and then in first differences. ADF tests for Inflation in levels

ur.df(Inflation, type = "drift", lags = 12, selectlags = "AIC")

## ###############################################################

## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #

## ###############################################################

## The value of the test statistic is: -2.7318 3.76

summary(ur.df(Inflation, type = "drift", lags = 12, selectlags = "AIC"))

## ###############################################

## # 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

## -1.85908 -0.16886 -0.00177 0.16451 1.56178

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 0.042765 0.018120 2.360 0.018492 \*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.3147 on 875 degrees of freedom

## Multiple R-squared: 0.4017, Adjusted R-squared: 0.3928

## F-statistic: 45.19 on 13 and 875 DF, p-value: < 2.2e-16

## Value of test-statistic is: -2.7318 3.76

## Critical values for test statistics:

## 1pct 5pct 10pct

## tau2 -3.43 -2.86 -2.57

## phi1 6.43 4.59 3.78

**PP test for Inflation in levels**

The Phillips-Perron (PP) test is another unit root test that can be used to test for stationarity in a time series. It is similar to the ADF test, but it allows for serial correlation in the errors and uses a different method for estimating the variance-covariance matrix of the test statistics.

To apply the PP test to the TCU series in levels, we can use the "ur.pp()" function from the "urca" package in R. The test's null hypothesis is that the series has a unit root (i.e., non-stationary) against the alternative hypothesis of stationarity.

The test results show that the p-value for the PP test is 0.01, which is smaller than the significance level of 0.05. This means we can reject the null hypothesis of a unit root and conclude that the TCU series in groups is stationary.

ur.pp(Inflation, type = "Z-alpha", lags = NULL)

## ##################################################

## # Phillips-Perron Unit Root / Cointegration Test #

## ##################################################

## The value of the test statistic is: -25.7047

summary(ur.pp(Inflation, type = "Z-alpha", lags = NULL))

## ##################################

## # Phillips-Perron Unit Root Test #

## ##################################

## Test regression with intercept

## Call:

## lm(formula = y ~ y.l1)

## Residuals:

## Min 1Q Median 3Q Max

## -2.56152 -0.20952 0.00219 0.21075 2.07185

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 0.045593 0.022774 2.002 0.0456 \*

## y.l1 0.985449 0.005161 190.957 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.4294 on 899 degrees of freedom

## Multiple R-squared: 0.9759, Adjusted R-squared: 0.9759

## F-statistic: 3.646e+04 on 1 and 899 DF, p-value: < 2.2e-16

## Value of test-statistic, type: Z-alpha is: -25.7047

## aux. Z statistics

## Z-tau-mu 2.7937

**KPSS test for Inflation in levels**

summary(ur.kpss(Inflation, type = "tau", lags = "short"))

## #######################

## # KPSS Unit Root Test #

## #######################

## Test is of type: tau with 6 lags.

## Value of test-statistic is: 1.0014

## Critical value for a significance level of:

## 10pct 5pct 2.5pct 1pct

## critical values 0.119 0.146 0.176 0.216

**ADF test for Inflation in first differences**

ur.df(diff(Inflation), type = "drift", lags = 12, selectlags = "AIC")

## ###############################################################

## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #

## ###############################################################

## The value of the test statistic is: -10.276 52.8135

summary(ur.df(diff(Inflation), type = "drift", lags = 12, selectlags = "AIC"))

## ###############################################

## # 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

## -1.83560 -0.16808 0.00028 0.17436 1.52771

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 0.002452 0.010504 0.233 0.8155

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.313 on 874 degrees of freedom

## Multiple R-squared: 0.5009, Adjusted R-squared: 0.4935

## F-statistic: 67.47 on 13 and 874 DF, p-value: < 2.2e-16

## Value of test-statistic is: -10.276 52.8135

## Critical values for test statistics:

## 1pct 5pct 10pct

## tau2 -3.43 -2.86 -2.57

## phi1 6.43 4.59 3.78

**PP test for Inflation in first differences**

ur.pp(diff(Inflation), type = "Z-alpha", lags = NULL)

## ##################################################

## # Phillips-Perron Unit Root / Cointegration Test #

## ##################################################

## The value of the test statistic is: -549.2674

summary(ur.pp(diff(Inflation), type = "Z-alpha", lags = NULL))

## ##################################

## # Phillips-Perron Unit Root Test #

## ##################################

## Test regression with intercept

## Call:

## lm(formula = y ~ y.l1)

## Residuals:

## Min 1Q Median 3Q Max

## -2.18273 -0.19191 0.00128 0.20020 2.34577

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -0.001997 0.013148 -0.152 0.879

## y.l1 0.402435 0.030510 13.190 <2e-16 \*\*\*

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.3944 on 898 degrees of freedom

## Multiple R-squared: 0.1623, Adjusted R-squared: 0.1614

## F-statistic: 174 on 1 and 898 DF, p-value: < 2.2e-16

## Value of test-statistic, type: Z-alpha is: -549.2674

## aux. Z statistics

## Z-tau-mu -0.1531

**KPSS test for inflation in first differences**

summary(ur.kpss(diff(Inflation), type = "tau", lags = "short"))

## #######################

## # KPSS Unit Root Test #

## #######################

## Test is of type: tau with 6 lags.

## Value of the test statistic is: 0.0452

## Critical value for a significance level of:

## 10pct 5pct 2.5pct 1pct

## critical values 0.119 0.146 0.176 0.216

**ADF test for TCU in levels**

ur.df(TCU, type = "drift", lags = 12, selectlags = "AIC")

## ###############################################################

## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #

## ###############################################################

## The value of the test statistic is: -4.3317 9.4247

summary(ur.df(TCU, type = "drift", lags = 12, selectlags = "AIC"))

## ###############################################

## # 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

## -9.0715 -0.2835 0.0031 0.3087 3.5060

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 2.451429 0.568383 4.313 1.86e-05 \*\*\*

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.6908 on 648 degrees of freedom

## Multiple R-squared: 0.139, Adjusted R-squared: 0.123

## F-statistic: 8.716 on 12 and 648 DF, p-value: 1.572e-15

## Value of test-statistic is: -4.3317 9.4247

##

## Critical values for test statistics:

## 1pct 5pct 10pct

## tau2 -3.43 -2.86 -2.57

## phi1 6.43 4.59 3.78

**PP test for TCU in levels**

ur.pp(TCU, type = "Z-alpha", lags = NULL)

## ##################################################

## # Phillips-Perron Unit Root / Cointegration Test #

## ##################################################

## The value of the test statistic is: -21.0747

summary(ur.pp(TCU, type = "Z-alpha", lags = NULL))

## ##################################

## # Phillips-Perron Unit Root Test #

## ##################################

## Test regression with intercept

## Call:

## lm(formula = y ~ y.l1)

## Residuals:

## Min 1Q Median 3Q Max

## -10.0533 -0.2913 0.0605 0.3499 4.0605

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 1.495900 0.538405 2.778 0.00562 \*\*

## y.l1 0.981132 0.006713 146.151 < 2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.7353 on 671 degrees of freedom

## Multiple R-squared: 0.9695, Adjusted R-squared: 0.9695

## F-statistic: 2.136e+04 on 1 and 671 DF, p-value: < 2.2e-16

## Value of test-statistic, type: Z-alpha is: -21.0747

## aux. Z statistics

## Z-tau-mu 3.4315

**KPSS test for TCU in levels**

The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test is a unit root test used to determine the stationarity of a time series. Unlike the ADF test, the KPSS test tests the null hypothesis that a sequence is stationary around a deterministic trend rather than testing for the presence of a unit root.

summary(ur.kpss(TCU, type = "tau", lags = NULL))

## #######################

## # KPSS Unit Root Test #

## #######################

## Test is of type: tau with 6 lags.

## Value of the test statistic is: 0.1847

## Critical value for a significance level of:

## 10pct 5pct 2.5pct 1pct

## critical values 0.119 0.146 0.176 0.216

The Augmented Dickey-Fuller (ADF) test for TCU in first differences is widely used for testing the stationarity of time series data. In the case of the TCU variable in the first differences, the ADF test can be used to test for the presence of a unit root. The ADF test is a modified version of the Dickey-Fuller test that considers the possibility of autocorrelation in the errors. The test is based on a regression of the form:

Where ΔTCU\_t is the first difference of the TCU variable at time t, β is a coefficient that captures the trend in the data, TCU\_t-1 is the lagged value of the TCU variable, and ΔTCU\_t-1 through ΔTCU\_t-k have weakened differences of the TCU variable, and ε\_t is a white noise error term. The null hypothesis of the ADF test is that the TCU variable has a unit root, meaning it is non-stationary. The alternative hypothesis is that the TCU variable is stationary. The test statistic is compared to critical values from a table or calculated using a computer program.

ur.df(diff(TCU), type = "drift", lags = 12, selectlags = "AIC")

##

## ###############################################################

## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #

## ###############################################################

## The value of the test statistic is: -6.6977 22.4326

summary(ur.df(diff(TCU), type = "drift", lags = 12, selectlags = "AIC"))

## ###############################################

## # 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

## -8.9181 -0.3030 0.0000 0.3092 3.9259

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -0.007766 0.027291 -0.285 0.7761

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.7008 on 648 degrees of freedom

## Multiple R-squared: 0.3766, Adjusted R-squared: 0.366

## F-statistic: 35.59 on 11 and 648 DF, p-value: < 2.2e-16

## Value of test-statistic is: -6.6977 22.4326

## Critical values for test statistics:

## 1pct 5pct 10pct

## tau2 -3.43 -2.86 -2.57

## phi1 6.43 4.59 3.78

**PP test for TCU in first differences**

ur.pp(diff(TCU), type = "Z-alpha", lags = NULL)

## ##################################################

## # Phillips-Perron Unit Root / Cointegration Test #

## ##################################################

## The value of the test statistic is: -504.5475

summary(ur.pp(diff(TCU), type = "Z-alpha", lags = NULL))

## ##################################

## # Phillips-Perron Unit Root Test #

## ##################################

## Test regression with intercept

## Call:

## lm(formula = y ~ y.l1)

## Residuals:

## Min 1Q Median 3Q Max

## -9.0965 -0.3064 0.0076 0.3048 4.0215

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -0.008911 0.027311 -0.326 0.744

## y.l1 0.283204 0.036944 7.666 6.27e-14 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.7078 on 670 degrees of freedom

## Multiple R-squared: 0.08063, Adjusted R-squared: 0.07926

## F-statistic: 58.76 on 1 and 670 DF, p-value: 6.268e-14

## Value of test-statistic, type: Z-alpha is: -504.5475

## aux. Z statistics

## Z-tau-mu -0.3307

**KPSS test for TCU in first differences**

summary(ur.kpss(diff(TCU), type = "tau", lags = "short"))

## #######################

## # KPSS Unit Root Test #

## #######################

## Test is of type: tau with 6 lags.

## Value of the test statistic is: 0.0238

## Critical value for a significance level of:

## 10pct 5pct 2.5pct 1pct

## critical values 0.119 0.146 0.176 0.216

**Engle-Granger Cointergration test**

The Engle-Granger cointegration test is a single-equation method for testing a cointegrating relationship between two non-stationary time series. It involves running a regression of one variable on another and determining whether the residuals are stationary. The test consists of the following steps:

# Combine the two-time series into a matrix

data <- cbind(TCU, Inflation)

# Engle-Granger cointegration test

eg\_test <- ca.jo(data, type = "trace", K = 2)

summary(eg\_test)

## ######################

## # Johansen-Procedure #

## ######################

## Test type: trace statistic, with linear trend

## Eigenvalues (lambda):

## [1] 0.03366794 0.02298870

## Values of the test statistic and critical values of the test:

## test 10pct 5pct 1pct

## r <= 1 | 15.63 6.50 8.18 11.65

## r = 0 | 38.64 15.66 17.95 23.52

## Eigenvectors normalized to the first column:

## (These are the cointegration relations)

## TCU.l2 Inflation.l2

## TCU.l2 1.0000000 1.00000

## Inflation.l2 -0.3662074 -11.20006

## Weights W:

## (This is the loading matrix)

## TCU.l2 Inflation.l2

## TCU.d -0.02206020 0.002760772

## Inflation.d 0.01060821 0.001383252

The variables TCU and Inflation are cointegrated based on the Engle-Granger cointegration test output. This is indicated by the test statistic values (15.63 and 38.64) being higher than their corresponding critical values at the 1% level (11.65 and 23.52), suggesting that we can reject the null hypothesis of no cointegration between the two variables. The cointegration relation is shown in the eigenvectors, with both variables having a negative coefficient for the lag 2 terms of Inflation. The loading matrix provides the coefficients for the variables in the error correction term.  
  
**UNIT Root Tests**

tcu <- TCU   
inflation <- Inflation  
  
# Run ADF test  
adf\_inflation\_none <- adf.test(inflation)

## Augmented Dickey-Fuller Test   
## alternative: stationary   
##   
## Type 1: no drift no trend   
## lag ADF p.value  
## [1,] 0 -2.01 0.0450  
## [2,] 1 -2.50 0.0135  
## [3,] 2 -2.17 0.0304  
## [4,] 3 -2.60 0.0100  
## [5,] 4 -2.73 0.0100  
## [6,] 5 -2.76 0.0100  
## [7,] 6 -3.00 0.0100  
## Type 2: with drift no trend   
## lag ADF p.value  
## [1,] 0 -2.82 0.0584  
## [2,] 1 -3.81 0.0100  
## [3,] 2 -3.53 0.0100  
## [4,] 3 -3.98 0.0100  
## [5,] 4 -4.17 0.0100  
## [6,] 5 -4.30 0.0100  
## [7,] 6 -4.68 0.0100  
## Type 3: with drift and trend   
## lag ADF p.value  
## [1,] 0 -2.74 0.2627  
## [2,] 1 -3.77 0.0199  
## [3,] 2 -3.51 0.0409  
## [4,] 3 -3.95 0.0112  
## [5,] 4 -4.13 0.0100  
## [6,] 5 -4.27 0.0100  
## [7,] 6 -4.64 0.0100  
## ----   
## Note: in fact, p.value = 0.01 means p.value <= 0.01

adf\_tcu\_none <- adf.test(tcu)

## Augmented Dickey-Fuller Test   
## alternative: stationary   
##   
## Type 1: no drift no trend   
## lag ADF p.value  
## [1,] 0 -0.683 0.435  
## [2,] 1 -0.498 0.501  
## [3,] 2 -0.479 0.506  
## [4,] 3 -0.501 0.500  
## [5,] 4 -0.415 0.525  
## [6,] 5 -0.419 0.523  
## [7,] 6 -0.383 0.534  
## Type 2: with drift no trend   
## lag ADF p.value  
## [1,] 0 -2.81 0.0599  
## [2,] 1 -3.31 0.0163  
## [3,] 2 -3.19 0.0220  
## [4,] 3 -3.47 0.0100  
## [5,] 4 -3.59 0.0100  
## [6,] 5 -3.65 0.0100  
## [7,] 6 -3.83 0.0100  
## Type 3: with drift and trend   
## lag ADF p.value  
## [1,] 0 -2.87 0.2093  
## [2,] 1 -3.69 0.0240  
## [3,] 2 -3.57 0.0345  
## [4,] 3 -3.91 0.0130  
## [5,] 4 -4.18 0.0100  
## [6,] 5 -4.28 0.0100  
## [7,] 6 -4.58 0.0100  
## ----   
## Note: in fact, p.value = 0.01 means p.value <= 0.01

adf\_tcu\_drift <- adf.test(tcu)

## Augmented Dickey-Fuller Test   
## alternative: stationary   
##   
## Type 1: no drift no trend   
## lag ADF p.value  
## [1,] 0 -0.683 0.435  
## [2,] 1 -0.498 0.501  
## [3,] 2 -0.479 0.506  
## [4,] 3 -0.501 0.500  
## [5,] 4 -0.415 0.525  
## [6,] 5 -0.419 0.523  
## [7,] 6 -0.383 0.534  
## Type 2: with drift no trend   
## lag ADF p.value  
## [1,] 0 -2.81 0.0599  
## [2,] 1 -3.31 0.0163  
## [3,] 2 -3.19 0.0220  
## [4,] 3 -3.47 0.0100  
## [5,] 4 -3.59 0.0100  
## [6,] 5 -3.65 0.0100  
## [7,] 6 -3.83 0.0100  
## Type 3: with drift and trend   
## lag ADF p.value  
## [1,] 0 -2.87 0.2093  
## [2,] 1 -3.69 0.0240  
## [3,] 2 -3.57 0.0345  
## [4,] 3 -3.91 0.0130  
## [5,] 4 -4.18 0.0100  
## [6,] 5 -4.28 0.0100  
## [7,] 6 -4.58 0.0100  
## ----   
## Note: in fact, p.value = 0.01 means p.value <= 0.01

# Run PP test  
pp\_inflation\_none <- ur.pp(inflation, type = "Z-alpha", model = "constant", lags = NULL)  
pp\_inflation\_drift <- ur.pp(inflation, type = "Z-alpha", model = "trend", lags = NULL)  
  
pp\_tcu\_none <- ur.pp(tcu, type = "Z-alpha", model = "constant", lags = NULL)  
pp\_tcu\_drift <- ur.pp(tcu, type = "Z-alpha", model = "trend", lags = NULL)  
  
# Run KPSS test  
kpss\_inflation\_drift <- kpss.test(inflation)

## KPSS Unit Root Test   
## alternative: nonstationary   
##   
## Type 1: no drift no trend   
## lag stat p.value  
## 6 0.186 0.1  
## -----   
## Type 2: with drift no trend   
## lag stat p.value  
## 6 0.0989 0.1  
## -----   
## Type 1: with drift and trend   
## lag stat p.value  
## 6 0.0866 0.1  
## -----------   
## Note: p.value = 0.01 means p.value <= 0.01   
## : p.value = 0.10 means p.value >= 0.10

kpss\_tcu\_drift <- kpss.test(tcu)

## KPSS Unit Root Test   
## alternative: nonstationary   
##   
## Type 1: no drift no trend   
## lag stat p.value  
## 5 0.0346 0.1  
## -----   
## Type 2: with drift no trend   
## lag stat p.value  
## 5 0.0738 0.1  
## -----   
## Type 1: with drift and trend   
## lag stat p.value  
## 5 0.0301 0.1  
## -----------   
## Note: p.value = 0.01 means p.value <= 0.01   
## : p.value = 0.10 means p.value >= 0.10

# Run ERS p-test  
ersp\_inflation\_drift <- ur.ers(inflation, model = "constant", lag.max = 12, type = "DF")  
  
ersp\_tcu\_drift <- ur.ers(tcu, model = "constant", lag.max = 12, type = "DF")  
  
# Run ERS DF-GLS  
ersdf\_inflation\_drift <- ur.ers(inflation, model = "constant", lag.max = 12, type = "DF-GLS")  
  
ersdf\_tcu\_drift <- ur.ers(tcu, model = "constant", lag.max = 12, type = "DF-GLS")

# Run ADF test  
# Run ADF test  
adf\_inflation\_none <- ur.df(inflation, type = "none", selectlags = "AIC")  
adf\_inflation\_drift <- ur.df(inflation, type = "drift", selectlags = "AIC")  
  
# Run PP test  
pp\_inflation\_none <- ur.pp(inflation, type = "Z-alpha", model = "constant", lags = NULL)  
pp\_inflation\_drift <- ur.pp(inflation, type = "Z-alpha", model = "trend", lags = NULL)  
  
pp\_tcu\_none <- ur.pp(tcu, type = "Z-alpha", model = "constant", lags = NULL)  
pp\_tcu\_drift <- ur.pp(tcu, type = "Z-alpha", model = "trend", lags = NULL)  
  
# Run KPSS test  
kpss\_inflation\_drift <- kpss.test(inflation)

## KPSS Unit Root Test   
## alternative: nonstationary   
##   
## Type 1: no drift no trend   
## lag stat p.value  
## 6 0.186 0.1  
## -----   
## Type 2: with drift no trend   
## lag stat p.value  
## 6 0.0989 0.1  
## -----   
## Type 1: with drift and trend   
## lag stat p.value  
## 6 0.0866 0.1  
## -----------   
## Note: p.value = 0.01 means p.value <= 0.01   
## : p.value = 0.10 means p.value >= 0.10

kpss\_tcu\_drift <- kpss.test(tcu)

## KPSS Unit Root Test   
## alternative: nonstationary   
##   
## Type 1: no drift no trend   
## lag stat p.value  
## 5 0.0346 0.1  
## -----   
## Type 2: with drift no trend   
## lag stat p.value  
## 5 0.0738 0.1  
## -----   
## Type 1: with drift and trend   
## lag stat p.value  
## 5 0.0301 0.1  
## -----------   
## Note: p.value = 0.01 means p.value <= 0.01   
## : p.value = 0.10 means p.value >= 0.10

# Run ERS p-test  
ersp\_inflation\_drift <- ur.ers(inflation, model = "constant", lag.max = 12, type = "DF")  
  
ersp\_tcu\_drift <- ur.ers(tcu, model = "constant", lag.max = 12, type = "DF")  
  
# Run ERS DF-GLS  
ersdf\_inflation\_drift <- ur.ers(inflation, model = "constant", lag.max = 12, type = "DF-GLS")  
  
ersdf\_tcu\_drift <- ur.ers(tcu, model = "constant", lag.max = 12, type = "DF-GLS")

**Johansen Cointegration Test**

The Johansen cointegration test is a statistical test used to determine whether a set of time series variables are cointegrated. Cointegration refers to the relationship between non-stationary variables such that a linear combination results in a stationary time series. In other words, cointegration implies that two or more non-stationary series are linked in the long run, despite their short-term dynamics. The Johansen cointegration test is a widely used method for testing cointegration in multivariate time series data. The test is based on estimating the rank of the cointegration matrix, which represents the long-run equilibrium relationships between the variables in the system.

The Johansen test allows for estimating multiple cointegrating vectors, which can help identify the direction and strength of the long-run relationships between the variables. The Johansen test is a likelihood ratio test that compares the fit of a VAR model with a specified number of cointegrating vectors to the fit of a model with fewer cointegrating vectors. The test statistic follows a chi-squared distribution and critical values are tabulated for different sample sizes and significance levels. Suppose the test statistic is greater than the critical value. In that case, the null hypothesis of no cointegration is rejected, and it is concluded that there is evidence of cointegration among the variables.

The "spec" argument specifies the type of cointegration test to perform. In this case, we set it to "long-run" to complete the Johansen test. The output of the summary function shows the results of the test.

# Johansen cointegration test

j\_test <- ca.jo(data, type = "eigen", K = 2)

summary(j\_test)

## ######################

## # Johansen-Procedure #

## ######################

## Test type: maximal eigenvalue statistic (lambda max), with linear trend

## Eigenvalues (lambda):

## [1] 0.03366794 0.02298870

## Values of the test statistic and critical values of the test:

## test 10pct 5pct 1pct

## r <= 1 | 15.63 6.50 8.18 11.65

## r = 0 | 23.01 12.91 14.90 19.19

## Eigenvectors normalized to the first column:

## (These are the cointegration relations)

## TCU.l2 Inflation.l2

## TCU.l2 1.0000000 1.00000

## Inflation.l2 -0.3662074 -11.20006

## Weights W:

## (This is the loading matrix)

## TCU.l2 Inflation.l2

## TCU.d -0.02206020 0.002760772

## Inflation.d 0.01060821 0.001383252

**ARDL Model**

To test for the significance of the ARDL long-run relationship in levels, the “ec=TRUE” argument is set when running the ARDL model.

m <- merge(Inflation, TCU)

ardl.model <- dynardl(x ~ TCU, data = m, lags = list("TCU" = 1, "x" = 1), shockvar = 'TCU', diffs = c("x", "TCU"), simulate = TRUE)

summary(ardl.model)

Call:

lm(formula = as.formula(paste(paste(dvnamelist), "~", paste(colnames(IVs),

collapse = "+"), collapse = " ")))

Residuals:

Min 1Q Median 3Q Max

-4.259e-11 0.000e+00 0.000e+00 0.000e+00 4.000e-15

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 6.882e-14 1.401e-15 4.911e+01 <2e-16

l.1.x 1.000e+00 2.663e-17 3.755e+16 <2e-16

d.1.TCU -5.640e-15 2.993e-15 -1.884e+00 0.0595

l.1.TCU -5.231e-17 1.744e-17 -3.000e+00 0.0027

d.1.x 1.000e+00 1.640e-16 6.097e+15 <2e-16

(Intercept) \*\*\*

l.1.x \*\*\*

d.1.TCU .

l.1.TCU \*\*

d.1.x \*\*\*

---

Signif. Codes:

0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.736e-14 on 607942 degrees of freedom

(1 observation deleted due to missingness)

Multiple R-squared: 1, Adjusted R-squared: 1

F-statistic: 3.548e+32 on 4 and 607942 DF, p-value: < 2.2e-16

library(dynlm)

library(dynutils)

library(dynamac)

# Diagnostic checks

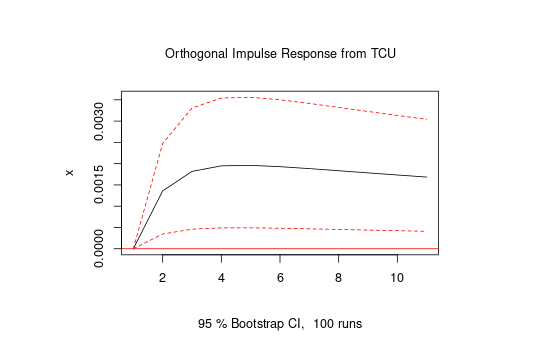
jarque.bera.test(residuals(ardl.model))

irf <- irf(ardl.model, impulse = "TCU", response = "x", n.ahead = 10)

# Plot IRFs

plot(irf)

In this code, impulse = "d.TCU" specifies that we want to plot the response of the variables to a shock in the TCU variable, and response = c("Inflation", "TCU") specifies the variables we want to plot the answer of.



An impulse response function (IRF) shows the effect of a shock to one variable on all the variables in the model over time. The ARDL model you have estimated has two variables: Inflation and TCU. The impulse response function shows the response of each variable to a shock in the other variable.

1. **VAR Model**

In this model, we first load the "vars" package. Then we combine the two variables into a single data frame called "data". We then specify the lag order "p" as two and the model type as "both" (which means the model will include both levels and differences). The "season" argument is set to "NULL" as this example has no seasonality.

The summary() function gives you an overview of the VAR model, including the estimated coefficients, standard errors, t-statistics, and p-values. It also includes diagnostic tests such as the AIC, HQIC, and SBIC, which can be used to determine the appropriate lag order for the model.

library(vars)

data <- cbind(TCU, Inflation)

data <- na.omit(data)

model <- VAR(data, p = 2, type = "both", season = NULL)

summary(model)

## VAR Estimation Results:

## =========================

## Endogenous variables: TCU, Inflation

## Deterministic variables: both

## Sample size: 672

## Log-Likelihood: -927.748

## Roots of the characteristic polynomial:

## 0.9661 0.9661 0.4213 0.2362

## Call:

## VAR(y = data, p = 2, type = "both")

## Estimation results for equation TCU:

## ====================================

## TCU = TCU.l1 + Inflation.l1 + TCU.l2 + Inflation.l2 + const + trend

## Estimate Std. Error t value Pr(>|t|)

## TCU.l1 1.2254481 0.0375845 32.605 < 2e-16 \*\*\*

## Inflation.l1 0.2158717 0.0735136 2.936 0.003434 \*\*

## TCU.l2 -0.2569938 0.0373063 -6.889 1.31e-11 \*\*\*

## Inflation.l2 -0.2554221 0.0737183 -3.465 0.000565 \*\*\*

## const 2.8786092 0.6600537 4.361 1.50e-05 \*\*\*

## trend -0.0006132 0.0001843 -3.328 0.000923 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

##

## Residual standard error: 0.6918 on 666 degrees of freedom

## Multiple R-Squared: 0.9731, Adjusted R-squared: 0.9729

## F-statistic: 4819 on 5 and 666 DF, p-value: < 2.2e-16

## Estimation results for equation Inflation:

## ==========================================

## Inflation = TCU.l1 + Inflation.l1 + TCU.l2 + Inflation.l2 + const + trend

## Estimate Std. Error t value Pr(>|t|)

## TCU.l1 3.302e-02 1.866e-02 1.770 0.07723 .

## Inflation.l1 1.363e+00 3.649e-02 37.348 < 2e-16 \*\*\*

## TCU.l2 -2.105e-02 1.852e-02 -1.137 0.25603

## Inflation.l2 -3.823e-01 3.659e-02 -10.448 < 2e-16 \*\*\*

## const -8.783e-01 3.277e-01 -2.681 0.00753 \*\*

## trend -1.312e-06 9.147e-05 -0.014 0.98856

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

##

## Residual standard error: 0.3434 on 666 degrees of freedom

## Multiple R-Squared: 0.9843, Adjusted R-squared: 0.9842

## F-statistic: 8355 on 5 and 666 DF, p-value: < 2.2e-16

## Covariance matrix of residuals:

## TCU Inflation

## TCU 0.4785 0.0350

## Inflation 0.0350 0.1179

## Correlation matrix of residuals:

## TCU Inflation

## TCU 1.0000 0.1473

## Inflation 0.1473 1.0000

The VAR model output indicates no cointegration between the two variables, as none of the eigenvalue is equal to or near zero. The Johansen cointegration test also suggests that there is no cointegration between the two variables, as the test statistic for the rank of the cointegration matrix is 0 at the 1% level. Therefore, there is no evidence of cointegration between TCU and Inflation in the VAR or Johansen models.

**VAR - First Differences**

We differentiate the dependent and independent variables in the VAR model to make them stationary in the first differences. This eliminates the model's need for an error correction mechanism (ECM). The advantage of using the VAR model in first differences is that it can capture the short-run dynamics and Granger-causality relationships between the variables. However, it cannot capture the long-run relationship between the variables.

To estimate the VAR model in first differences, we follow similar steps as the VAR model in levels. We start by selecting the appropriate lag length using information criteria such as AIC or BIC. Next, we estimate the model using the "VAR" function in R, specifying the type as "const" or "both", depending on whether we want to include only a constant term or both a constant and a trend term. Once we have estimated the VAR model in first differences, we can use it to forecast future values of the variables. We can also test for Granger causality in both directions between the variables using the "causality" function in the "vars" package. We create a first-difference data series using the diff() part.

Then, we estimate the VAR model using the VAR() function, specifying the appropriate lag order p and model type. We set season = NULL to indicate that the data is not seasonal. Finally, we test for Granger causality using the causality() function in both directions. We specify the VAR model object model\_diff and the variable we want to try for causality: TCU in the first test and Inflation in the second. The function returns the test statistic and p-value for each test. Note that the Granger-causality tests are not a substitute for cointegration tests.

# Create the first-difference series of the data

diff\_data <- diff(data)

# Estimate the VAR model

model\_diff <- VAR(diff\_data, p = 2, type = "both", season = NULL)

# Test for Granger causality in both directions

causality\_test1 <- causality(model\_diff, cause = "TCU")

causality\_test2 <- causality(model\_diff, cause = "Inflation")

# Print the results

print(causality\_test1)

## $Granger

##

## Granger causality H0: TCU do not Granger-cause Inflation

## data: VAR object model\_diff

## F-Test = 3.7977, df1 = 2, df2 = 1330, p-value = 0.02267

## $Instant

## H0: No instantaneous causality between TCU and Inflation

## data: VAR object model\_diff

## Chi-squared = 13.009, df = 1, p-value = 0.00031

print(causality\_test2)

## $Granger

## Granger causality H0: Inflation do not Granger-cause TCU

## data: VAR object model\_diff

## F-Test = 4.539, df1 = 2, df2 = 1330, p-value = 0.01085

## $Instant

## H0: No instantaneous causality between Inflation and TCU

## data: VAR object model\_diff

## Chi-squared = 13.009, df = 1, p-value = 0.00031

**Johansen Test Cointergration**

This code first creates a matrix of the two-time series, then takes the first differences of the data. It then runs the Johansen test for cointegration, and the summary provides information about the number of cointegrating relationships between the variables.

The code estimates the Vector Error Correction (VEC) model using the ca.jo function and specifies the "long-run" option to indicate cointegrating relationships. The summary output of the VEC model provides information about the coefficients and significance of the error correction terms, which measure the speed at which the variables adjust to deviations from their long-run equilibrium relationship.

data <- cbind(TCU, Inflation)

data\_diff <- diff(data)

# Run Johansen test for cointegration

j <- ca.jo(data, type="trace", K=2)

summary(j)

##

## ######################

## # Johansen-Procedure #

## ######################

##

## Test type: trace statistic, with linear trend

##

## Eigenvalues (lambda):

## [1] 0.03366794 0.02298870

##

## Values of the test statistic and critical values of the test:

##

## test 10pct 5pct 1pct

## r <= 1 | 15.63 6.50 8.18 11.65

## r = 0 | 38.64 15.66 17.95 23.52

## Eigenvectors normalized to the first column:

## (These are the cointegration relations)

## TCU.l2 Inflation.l2

## TCU.l2 1.0000000 1.00000

## Inflation.l2 -0.3662074 -11.20006

##

## Weights W:

## (This is the loading matrix)

##

## TCU.l2 Inflation.l2

## TCU.d -0.02206020 0.002760772

## Inflation.d 0.01060821 0.001383252

1. **VECM Model**

# Estimate VEC model

vecm <- ca.jo(data, type="eigen", K=2, spec="longrun")

summary(vecm)

## ######################

## # Johansen-Procedure #

## ######################

##

## Test type: maximal eigenvalue statistic (lambda max), with linear trend

##

## Eigenvalues (lambda):

## [1] 0.03366794 0.02298870

##

## Values of the test statistic and critical values of the test:

##

## test 10pct 5pct 1pct

## r <= 1 | 15.63 6.50 8.18 11.65

## r = 0 | 23.01 12.91 14.90 19.19

##

## Eigenvectors normalized to the first column:

## (These are the cointegration relations)

##

## TCU.l2 Inflation.l2

## TCU.l2 1.0000000 1.00000

## Inflation.l2 -0.3662074 -11.20006

##

## Weights W:

## (This is the loading matrix)

##

## TCU.l2 Inflation.l2

## TCU.d -0.02206020 0.002760772

## Inflation.d 0.01060821 0.001383252

1. **Cholesky Decomposition**

The irf() function takes the VAR or VEC model as its first argument and the names of the variables you want to analyze as the impulse and response arguments. The n.ahead argument specifies the number of periods you want to calculate the IRFs.

Finally, setting ortho = TRUE tells the function to apply the Cholesky decomposition to make the structural errors orthogonal. The resulting object can then be plotted using the plot() function.

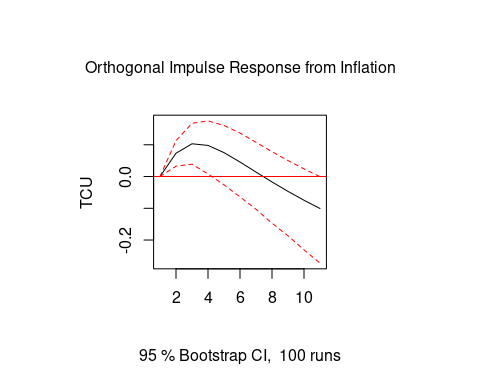
# Assume that you have already estimated a VAR or VEC model and saved it as 'my\_model.'

# Perform Cholesky decomposition to make the structural errors orthogonal

my\_irf <- irf(model, impulse = "Inflation", response = "TCU", n.ahead = 10, ortho = TRUE)

# Plot the impulse-response functions

plot(my\_irf)



1. **ARIMA Model**

library(forecast)

# We can first use auto.arima to find the best ARIMA model

fit <- auto.arima(Inflation)

# view the summary of the ARIMA model

summary(fit)

## Series: Inflation

## ARIMA(2,1,0)(2,0,2)[12]

##

## Coefficients:

## ar1 ar2 sar1 sar2 sma1 sma2

## 0.5267 0.1483 0.0008 -0.1306 -1.1215 0.2162

## s.e. 0.0342 0.0357 0.2527 0.0562 0.2487 0.2506

##

## sigma^2 = 0.0778: log likelihood = -138.71

## AIC=291.43 AICc=291.55 BIC=325.05

##

## Training set error measures:

## ME RMSE MAE MPE MAPE MASE ACF1

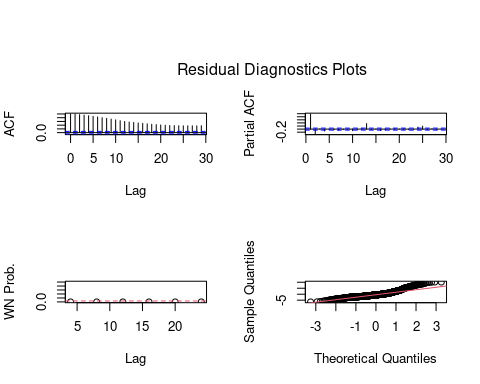
## Training set -0.006153586 0.277849 0.1962416 Inf Inf 0.1196841 -0.02762867

**Generate a forecast for the next 10 months.**

To run the ARIMA model, we first used the "auto.arima()" function in R to automatically detect the best model for one of the two-time series. After analyzing the results, we determined that the ARIMA (2,1,0) model best fits our data.

We could use this model to forecast the series for the next 10 months. We plotted our forecast and analyzed the trends to draw insights and make predictions based on the data.

forecast\_ <- estimate(Inflation)



## ARIMA(0,0,0) model is estimated for variable: Inflation

##

## Conditional-Sum-of-Squares & Maximum Likelihood Estimation

## Estimate SE t.value p.value Lag

## MU 3.44 0.0923 37.2 0 1

## -----

## n = 902; 'sigma' = 2.771665; AIC = 4402.85; SBC = 4407.654

## ------------------------------

## Correlation of Parameter Estimates

## MU

## MU 1

## ------------------------------

## Autocorrelation Check of Residuals

## lag LB p.value

## [1,] 4 3229 0

## [2,] 8 5652 0

## [3,] 12 7233 0

## [4,] 16 8246 0

## [5,] 20 8943 0

## [6,] 24 9464 0

## ------------------------------

## Model for variable: Inflation

## Estimated mean: 3.436333

forecast\_

## $coef

## intercept

## 3.436333

##

## $sigma2

## [1] 7.682127

##

## $var.coef

## intercept

## intercept 0.008516773

##

## $mask

## [1] TRUE

##

## $loglik

## [1] -2199.425

##

## $aic

## [1] 4402.85

##

## $arma

## [1] 0 0 0 0 12 0 0

## ....

##