Introduction The two time series with monthly frequency provided are Capacity Utilisation Rate (TCU) and Consumer Price Index Inflation Rate (CPIAUCSL). TCU represents the extent to which the production capacity of firms is being used, while CPIAUCSL represents the changes in the cost of a basket of goods and services purchased by households. 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 output 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) from one year ago. CPI is a measure of the average price level of a basket of goods and services consumed by households. The percentage change is obtained by calculating the difference between the current month’s CPI and the CPI of the same month one year ago, divided by the CPI of the same month one year ago, and then multiplied by 100. Loading Packages

library(dynamac)   
library(forecast)

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

library(tseries)   
library(nlme)

##   
## Attaching package: 'nlme'

## The following object is masked from 'package:forecast':  
##   
## getResponse

library(pdfetch)   
library(zoo)

##   
## Attaching package: 'zoo'

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

library(urca)   
library(vars)

## Loading required package: MASS

## Loading required package: strucchange

## Loading required package: sandwich

## Loading required package: lmtest

library(car)

## Loading required package: carData

library(dynlm)   
library(tsDyn)   
library(gets)

## Loading required package: parallel

##   
## Attaching package: 'gets'

## The following object is masked from 'package:car':  
##   
## logit

library(readxl)   
library(aod)   
library(egcm)

## Loading required package: xts

library(aTSA)

##   
## Attaching package: 'aTSA'

## The following object is masked from 'package:vars':  
##   
## arch.test

## The following objects are masked from 'package:tseries':  
##   
## adf.test, kpss.test, pp.test

## The following object is masked from 'package:forecast':  
##   
## forecast

## The following object is masked from 'package:graphics':  
##   
## identify

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2  
## ──

## ✔ ggplot2 3.4.0 ✔ purrr 0.3.5   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.3 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ stringr::boundary() masks strucchange::boundary()  
## ✖ dplyr::collapse() masks nlme::collapse()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::first() masks xts::first()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ dplyr::last() masks xts::last()  
## ✖ dplyr::recode() masks car::recode()  
## ✖ dplyr::select() masks MASS::select()  
## ✖ purrr::some() masks car::some()

Clear the data buffer:

rm(list=ls())

Inflation data The CPI data is read from FRED using the pdfetch\_FRED() function from the quantmod package. The names() function is used to assign the name “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, which is a common measure of inflation. The resulting series is multiplied by 100 to convert it from a decimal to a percentage. The names() function is then used again to assign the name “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 begins in January 1948. The capacity utilization data is obtained using a similar process. The pdfetch\_FRED() function is used to download the data for “TCU” (total capacity utilization), which is then assigned the name “TCU” using names(). The resulting data is converted to a time series using the ts() function, with a start date of January 1967 and a frequency of 12 (monthly data).

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)

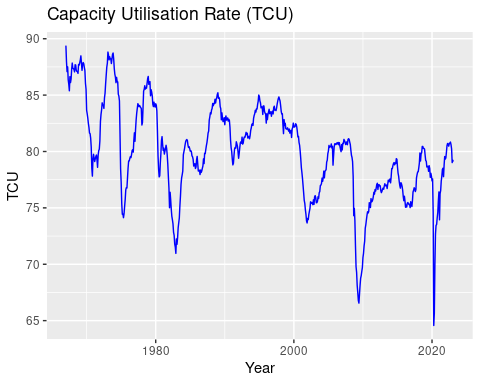
How the Data is Measured

The data consists of two time series: the Consumer Price Index (CPI) and the Capacity Utilization Rate (CUR). The CPI measures the average change in prices over time of a fixed basket of goods and services consumed by households. 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. The Capacity Utilization Rate (CUR) is a measure of the extent to which a firm is using its installed productive capacity. It is calculated as the ratio of actual output to potential output, where potential output is the maximum level of output that a firm can produce with its installed productive capacity. In this dataset, the CUR is measured in levels and represents the percentage of productive capacity that is being utilized in the manufacturing, mining, and electric and gas utilities industries in the United States.

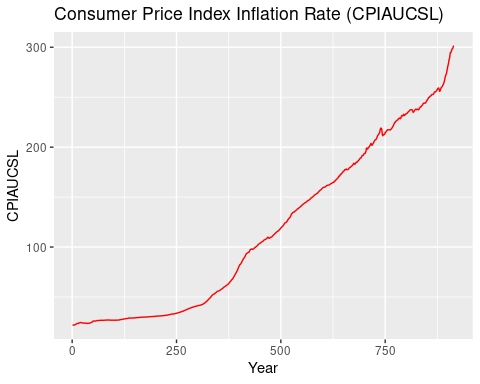
Plotting Data

# 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")

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

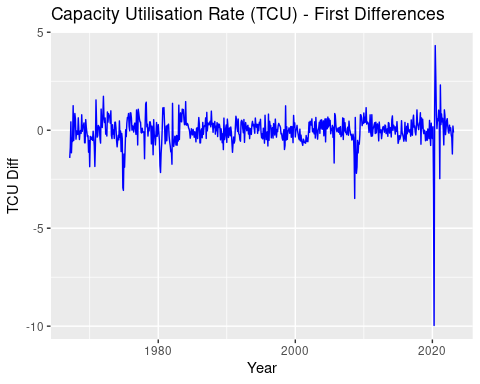
 Consumer Price Index Inflation Rate (CPIAUCSL)

# 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")

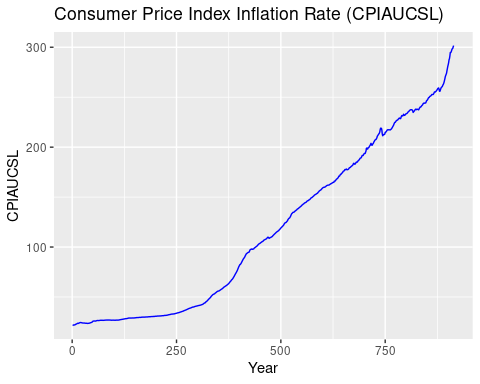
 Capacity Utilisation Rate (TCU) - First Differences

# 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")

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

 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")

 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 are used to 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 test 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 \*   
## z.lag.1 -0.011893 0.004354 -2.732 0.006427 \*\*   
## z.diff.lag1 0.345556 0.029059 11.892 < 2e-16 \*\*\*  
## z.diff.lag2 0.059477 0.030952 1.922 0.054981 .   
## z.diff.lag3 -0.011299 0.030884 -0.366 0.714571   
## z.diff.lag4 0.036098 0.030778 1.173 0.241175   
## z.diff.lag5 0.064922 0.030300 2.143 0.032415 \*   
## z.diff.lag6 0.016143 0.030336 0.532 0.594761   
## z.diff.lag7 0.087664 0.030261 2.897 0.003862 \*\*   
## z.diff.lag8 0.020324 0.030367 0.669 0.503491   
## z.diff.lag9 0.037002 0.030347 1.219 0.223061   
## z.diff.lag10 0.084945 0.029876 2.843 0.004570 \*\*   
## z.diff.lag11 0.100161 0.029170 3.434 0.000623 \*\*\*  
## z.diff.lag12 -0.458721 0.027523 -16.667 < 2e-16 \*\*\*  
## ---  
## 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 null hypothesis of the test is that the series has a unit root (i.e., it is 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 that we can reject the null hypothesis of a unit root and conclude that the TCU series in levels is stationary.

It is important to note that the PP test is similar to the ADF test in that it assumes that the errors are white noise. However, if the errors are serially correlated, the PP test may have low power and may not detect non-stationarity when it is present. Therefore, it is important to check for serial correlation in the errors using diagnostic tests such as the Ljung-Box test. Additionally, like the ADF test, the PP test assumes that the underlying data generating process is linear and time-invariant. If these assumptions are violated, the test results may not be reliable.

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   
## z.lag.1 -0.592255 0.057635 -10.276 < 2e-16 \*\*\*  
## z.diff.lag1 0.010934 0.053849 0.203 0.8391   
## z.diff.lag2 0.054194 0.053344 1.016 0.3099   
## z.diff.lag3 0.026421 0.052689 0.501 0.6162   
## z.diff.lag4 0.052259 0.051443 1.016 0.3100   
## z.diff.lag5 0.110522 0.049844 2.217 0.0269 \*   
## z.diff.lag6 0.113134 0.048202 2.347 0.0191 \*   
## z.diff.lag7 0.189033 0.045980 4.111 4.31e-05 \*\*\*  
## z.diff.lag8 0.194227 0.043760 4.439 1.02e-05 \*\*\*  
## z.diff.lag9 0.222688 0.040603 5.485 5.43e-08 \*\*\*  
## z.diff.lag10 0.296768 0.037375 7.940 6.18e-15 \*\*\*  
## z.diff.lag11 0.383413 0.034029 11.267 < 2e-16 \*\*\*  
## z.diff.lag12 -0.133037 0.031270 -4.254 2.32e-05 \*\*\*  
## ---  
## 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 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 \*\*\*  
## z.lag.1 -0.030736 0.007096 -4.332 1.71e-05 \*\*\*  
## z.diff.lag1 0.299697 0.038579 7.768 3.12e-14 \*\*\*  
## z.diff.lag2 -0.036453 0.040190 -0.907 0.36474   
## z.diff.lag3 0.050993 0.040271 1.266 0.20588   
## z.diff.lag4 0.075998 0.040268 1.887 0.05956 .   
## z.diff.lag5 0.013603 0.040320 0.337 0.73594   
## z.diff.lag6 0.072175 0.040160 1.797 0.07277 .   
## z.diff.lag7 0.009042 0.040176 0.225 0.82201   
## z.diff.lag8 -0.024710 0.040119 -0.616 0.53817   
## z.diff.lag9 0.060198 0.039991 1.505 0.13274   
## z.diff.lag10 0.114998 0.039966 2.877 0.00414 \*\*   
## z.diff.lag11 -0.056014 0.038782 -1.444 0.14912   
## ---  
## 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 is used to test the null hypothesis that a series is stationary around a deterministic trend, rather than testing for the presence of a unit root.

In the context of testing the stationarity of TCU in levels, the KPSS test can be used to determine if the TCU series has a trend component that is non-stationary. The test is based on regressing the time series on a constant and a deterministic trend, and testing the null hypothesis that the residuals from this regression are stationary. If the null hypothesis is rejected, then there is evidence that the series has a trend component that is non-stationary.

It is important to note that the KPSS test has some limitations. First, it assumes that the errors in the regression are serially uncorrelated, which may not hold in practice. Second, the test may suffer from low power in small samples, and may not detect non-stationarity when it is present. Therefore, it is important to use the KPSS test in conjunction with other tests, such as the ADF test, to make a definitive conclusion about the stationarity of a time series.

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

##   
## #######################   
## # KPSS Unit Root Test #   
## #######################   
##   
## Test is of type: tau with 6 lags.   
##   
## Value of 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

ADF test for TCU in first differences The Augmented Dickey-Fuller (ADF) test is a widely used test for testing the stationarity of time series data. In the case of the TCU variable in 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 takes into account the possibility of autocorrelation in the errors. The test is based on a regression of the form:

ΔTCU\_t = α + βt + γTCU\_t-1 + δ1ΔTCU\_t-1 + … + δkΔTCU\_t-k + ε\_t

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 are lagged 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 that 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.

If the test statistic is less than the critical value, then the null hypothesis of a unit root is rejected, and the conclusion is that the TCU variable is stationary. If the test statistic is greater than the critical value, then the null hypothesis cannot be rejected, and the conclusion is that the TCU variable is non-stationary.

In the case of the TCU variable in first differences, the null hypothesis is that the first differences of the TCU variable have a unit root, and the alternative hypothesis is that the first differences are stationary. If the null hypothesis is rejected, it suggests that the first differences of the TCU variable are stationary, and the variable has achieved a level of stability over time.

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   
## z.lag.1 -0.578597 0.086387 -6.698 4.6e-11 \*\*\*  
## z.diff.lag1 -0.121972 0.085776 -1.422 0.1555   
## z.diff.lag2 -0.169759 0.082679 -2.053 0.0405 \*   
## z.diff.lag3 -0.130129 0.079089 -1.645 0.1004   
## z.diff.lag4 -0.065957 0.075695 -0.871 0.3839   
## z.diff.lag5 -0.065383 0.072092 -0.907 0.3648   
## z.diff.lag6 -0.008407 0.067507 -0.125 0.9009   
## z.diff.lag7 -0.015737 0.062553 -0.252 0.8014   
## z.diff.lag8 -0.058333 0.055942 -1.043 0.2975   
## z.diff.lag9 -0.014925 0.047695 -0.313 0.7544   
## z.diff.lag10 0.082827 0.038891 2.130 0.0336 \*   
## ---  
## 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 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 the presence of a cointegrating relationship between two non-stationary time series. It involves running a regression of one variable on another and testing whether the residuals are stationary. The test consists of the following steps: Test the order of integration of the two variables using the ADF or KPSS test. If both variables are integrated of order 1, then they can be cointegrated. Regress one variable (say Y) on the other (say X) to obtain the estimated regression equation Y^ = a + bX^ + e^, where Y^ and X^ are the estimated values of Y and X, respectively, a and b are the estimated intercept and slope coefficients, and e^ is the estimated residual. Test the stationarity of the residual series e^ using the ADF or KPSS test. If the null hypothesis of non-stationarity is rejected, then there is evidence of cointegration between the two variables. If the null hypothesis of non-stationarity is not rejected, then there is no evidence of cointegration between the two variables. The Engle-Granger test has several limitations. First, it only tests for cointegration between two variables at a time, so it cannot be used to test for cointegration among multiple variables. Second, it assumes that the cointegrating relationship is linear and constant over time. Finally, it may suffer from low power in small samples, and it may not detect cointegration when it is present.

The “type” argument specifies the type of test to perform, and “K” specifies the maximum number of cointegrating vectors to consider. In this case, we set K=2 to allow for up to two cointegrating vectors. The output of the summary function shows the results of the test. If the p-value of the test statistic is less than 0.05, we can reject the null hypothesis of no cointegration and conclude that the variables are cointegrated.

# 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 teststatistic and critical values of 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, normalised to 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

Johansen Cointergration 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 of these variables 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 the estimation of the rank of the cointegration matrix, which represents the long-run equilibrium relationships between the variables in the system. The Johansen test allows for the estimation of multiple cointegrating vectors, which can be useful in identifying 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 levels of significance. If the test statistic is greater than the critical value, then 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 “longrun” to perform the Johansen test. The output of the summary function shows the results of the test. Again, if the p-value of the test statistic is less than 0.05, we can reject the null hypothesis of no cointegration and conclude that the variables are cointegrated.

# 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 teststatistic and critical values of 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, normalised to 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

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 2 and the model type as “both” (which means the model will include both levels and differences). The “season” argument is set to “NULL” as there is no seasonality in this example.

The summary() function gives you a summary 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.

If the Johansen cointegration test indicates that there is evidence of cointegration, then you can estimate the VEC model by using the ca.jo() function in the “vars” package to estimate the cointegrating vectors, and then using the vec2var() function to convert the VEC model to a VAR model with an error correction term.

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

VAR - First Differences In the VAR model in first differences, we difference both the dependent and independent variables to make them stationary. This eliminates the need to include an error correction mechanism (ECM) in the model. 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 first create a first-difference series of the data using the diff() function. Then, we estimate the VAR model using the VAR() function, specifying the appropriate lag order p and model type type. We set season = NULL to indicate that the data is not seasonal. Finally, we test for Granger causality in both directions using the causality() function. We specify the VAR model object model\_diff and the variable we want to test for causality, which is TCU in the first test and Inflation in the second test. 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. If the Johansen test suggests the presence of cointegration, we should estimate a VECM and use its impulse-response functions to test for Granger causality in the long-run.

# 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. If there is evidence of cointegration, then the code estimates the VEC model using the ca.jo function and specifies the “longrun” option to indicate that there are cointegrating relationships. The summary output of the VEC model provides information about the coefficients and significance of the error correction terms.

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 teststatistic and critical values of 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, normalised to 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

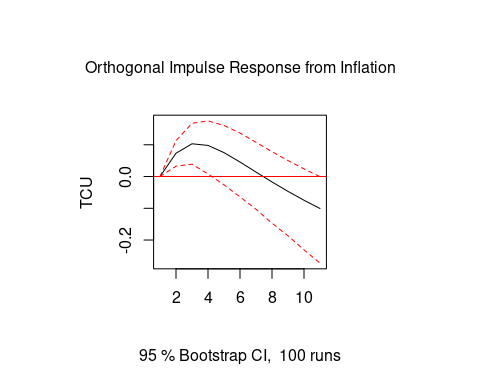
VEC Model

# 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 teststatistic and critical values of 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, normalised to 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

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 time periods for which 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)



ARIMA Model

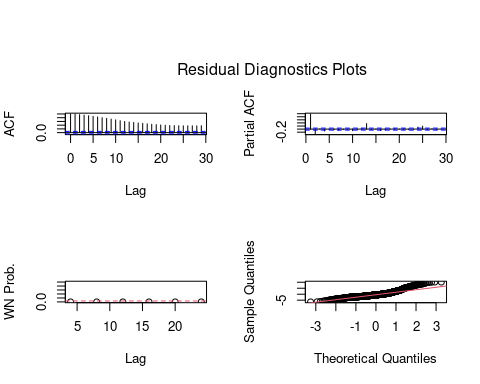
library(forecast)  
  
# assuming the time series is stored in the object `Inflation`  
# we can first use auto.arima to find the best ARIMA model  
fit <- auto.arima(Inflation)  
  
# view the summary of the ARIMA model  
summary(fit)

## 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 had the best fit for our data.

Using this model, we were able 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 S.E 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  
##   
## $residuals  
## Time Series:  
## Start = 1   
## End = 902   
## Frequency = 1   
## [1] 6.314521275 5.622628369 3.159464017 4.511978295 5.533994309  
## [6] 5.524883107 5.877707670 5.238759879 3.006573043 2.495096424  
## [11] 1.223553060 -0.739161257 -2.052372740 -2.427496968 -1.706696730  
## [16] -3.017396247 -3.853695638 -4.393277864 -6.347141178 -6.470016475  
## [21] -5.972323998 -6.104269242 -5.358665255 -5.282797308 -5.540787530  
## [26] -4.698975799 -4.571991655 -4.571514177 -4.023582772 -3.603696818  
## [31] -1.887212397 -1.348574260 -0.982477054 0.010136640 0.290806718  
## [36] 2.204182598 4.217227211 5.550292839 5.616694947 5.728842893  
## [41] 5.492423966 4.799598341 3.929941516 3.198141256 3.276530383  
## [46] 3.119508142 3.321933590 2.357318392 0.693136840 -1.215720842  
## [51] -1.484864674 -1.374404042 -1.606316004 -1.148775902 -0.507810115  
## [56] -0.277173755 -1.157464680 -1.430589079 -2.040347066 -2.533731674  
## [61] -2.720564025 -2.757084868 -2.531007831 -2.570852289 -2.571177845  
## [66] -2.535763824 -3.024886511 -2.838647008 -2.464725717 -2.466899368  
## [71] -2.838647008 -2.839093212 -2.316500231 -1.943210585 -2.316082062  
## [76] -2.801409994 -2.578598620 -2.803301417 -3.175382008 -3.436332762  
## [81] -3.734284565 -4.293427631 -3.697380831 -3.809189272 -4.069364107  
## [86] -4.068187680 -3.957555089 -3.697283516 -4.032237691 -4.293747148  
## [91] -3.809328346 -3.921679991 -3.287245891 -3.062779844 -3.063615223  
## [96] -3.063476253 -3.212452072 -3.287301438 -3.063754090 -2.915110435  
## [101] -2.469782486 -1.802430836 -1.475122816 -1.252274080 -1.591261971  
## [106] -0.896163054 -1.119626834 -0.647161131 -0.353519826 0.003450198  
## [111] 0.107417045 0.209714551 0.089382227 0.038501092 -0.191635722  
## [116] 0.053857792 0.048841234 -0.534463373 -0.217170669 -0.441456527  
## [121] 0.009222067 -0.250222555 0.124769770 0.116001363 -0.134311659  
## [126] -0.630115651 -0.983511827 -1.129344745 -1.374404042 -1.374404042  
## [131] -1.553431947 -1.695341713 -2.152707158 -2.396462040 -3.090550951  
## [136] -3.298211193 -3.091385887 -2.746912589 -2.540392930 -2.610450573  
## [141] -2.267130997 -1.925833738 -2.064098537 -1.928938684 -2.203018054  
## [146] -2.032440504 -1.928938684 -1.522398741 -1.627718371 -1.733294327  
## [151] -2.073449657 -1.973472860 -2.213075919 -2.082674190 -1.981884664  
## [156] -2.085417250 -1.848729750 -1.984830485 -1.984830485 -2.526469703  
## [161] -2.527388615 -2.662569440 -2.191991837 -2.328009074 -2.194497706  
## [166] -2.666196694 -2.766986220 -2.767657586 -2.768327606 -2.535576046  
## [171] -2.336504979 -2.103423991 -2.104755165 -2.204010756 -2.438652388  
## [176] -2.307127804 -1.979353346 -2.110932313 -2.110932313 -2.210948991  
## [181] -2.113562152 -2.214993930 -2.315688206 -2.546559220 -2.547438020  
## [186] -2.120957006 -1.893041583 -1.896076194 -2.454970617 -2.225783054  
## [191] -2.128269638 -1.803909963 -1.807101547 -2.035430230 -2.036798120  
## [196] -1.906105931 -1.907599136 -2.138034747 -2.366803850 -2.465451349  
## [201] -2.271271040 -2.240262256 -2.048989975 -2.245267660 -2.343425709  
## [206] -2.246416809 -2.247563743 -2.056557945 -1.835273906 -1.519953451  
## [211] -1.647147654 -1.838854701 -1.713802134 -1.747587834 -1.720915856  
## [216] -1.534532179 -1.536338938 -0.910946030 -0.695572140 -0.608623152  
## [221] -0.710173022 -1.029590560 -0.718689330 -0.009206089 0.074980268  
## [226] 0.285026872 0.060851723 -0.132033383 -0.286952377 -0.608854915  
## [231] -0.920090772 -0.927788864 -1.144406618 -0.634687888 -0.550787079  
## [236] -0.866274451 -0.874022273 -0.881723519 -0.381288342 -0.208317881  
## [241] 0.146139890 0.135475498 0.427446508 0.415995438 0.706271403  
## [246] 0.681896235 0.956760160 0.943929504 0.931173588 1.202179896  
## [251] 0.893347813 1.162178562 1.148997688 1.135892172 1.678418355  
## [256] 1.939784927 1.924612298 1.894522583 1.864768835 1.850016194  
## [261] 2.105251152 2.074703509 2.326578522 2.295112898 2.543709568  
## [266] 2.790306111 2.479370323 2.447717240 2.432017421 2.400867860  
## [271] 2.113307783 2.098676746 2.069644956 2.040916203 2.012485766  
## [276] 1.984349022 1.706188419 1.179871414 0.906623031 0.635476540  
## [281] 0.873586486 0.851839986 0.841048836 0.830311870 0.564200699  
## [286] 0.300091912 0.037962082 -0.222211841 -0.230139516 0.254122931  
## [291] 0.003809910 -0.004623470 -0.261462931 -0.516417293 -0.523505470  
## [296] -0.530559314 -0.299766834 -0.307316963 -0.078703209 -0.086737319  
## [301] 0.139733626 0.355590724 1.281523265 1.729134724 1.945082519  
## [306] 2.386033266 2.146512193 3.701333522 3.668601853 4.312416786  
## [311] 4.495342721 5.127455754 5.732095507 6.094685218 6.220287077  
## [316] 6.157074739 6.734588320 6.873218141 7.483596434 6.899518537  
## [321] 7.849116183 7.755458858 8.075336298 7.981350146 7.674984053  
## [326] 7.184249662 6.512222359 6.264640882 5.419006972 5.349802817  
## [331] 5.669663789 4.829657807 4.171897884 3.932438817 3.682777324  
## [336] 3.450108346 3.041417070 2.648493280 2.447717240 2.248057136  
## [341] 2.592890264 2.372922921 1.970389365 2.125638765 1.912535733  
## [346] 1.884070844 1.502942771 1.476936096 1.630252983 2.468159822  
## [351] 2.794055572 3.284542207 3.083986619 3.050582668 3.017519352  
## [356] 2.984791484 2.789394762 2.758115834 3.060539357 3.028220834  
## [361] 3.155839318 2.616209277 2.744495975 2.861147154 3.462954387  
## [366] 3.738057724 4.009702605 4.126324771 4.705877710 5.115884582  
## [371] 5.062988521 5.171128115 5.410897001 5.950280861 6.323846018  
## [376] 6.534745549 6.726931792 7.068944838 7.404597089 7.750103007  
## [381] 7.789066656 7.960385936 8.424241479 9.010651433 9.551175380  
## [386] 9.808366222 10.184687721 10.182035194 10.039280490 9.899491982  
## [391] 8.918691176 8.688122101 8.580634401 8.460104308 8.458861524  
## [396] 8.211847168 7.713217660 7.352563439 6.649267592 6.218218279  
## [401] 5.905343958 5.818822974 6.796596413 6.834945511 6.968524319  
## [406] 6.341241596 5.711624524 5.100704316 4.497341462 3.901385809  
## [411] 3.221881496 2.975422950 3.247188520 3.499780018 2.915007810  
## [416] 2.357810087 1.386404714 1.473269372 0.943929504 0.318018484  
## [421] 0.204214876 -0.010984914 0.091003876 0.485738553 -0.053129522  
## [426] -0.992219246 -1.104752231 -1.009520035 -0.710267941 -0.721233856  
## [431] -0.322068027 0.280820449 0.764284801 1.150712645 1.340694865  
## [436] 1.017644375 0.807027079 0.798667395 0.782046468 0.769666151  
## [441] 0.757358300 0.741059462 0.633996549 0.527453414 0.028884008  
## [446] 0.106402499 0.283695607 0.082813072 0.076130831 0.162540471  
## [451] -0.036571908 -0.138813081 -0.240572085 -0.252543252 0.017113547  
## [456] 0.285026872 0.460264091 -0.287924569 -1.305636131 -1.860036796  
## [461] -1.771169403 -1.684328520 -1.778836253 -1.873082537 -1.693968647  
## [466] -1.881660388 -2.160107601 -2.256110256 -2.080686154 -1.540113418  
## [471] -0.634522737 0.177429930 0.167660886 0.242861931 0.415464481  
## [476] 0.762586867 0.747633072 0.827190402 0.993089261 0.804585430  
## [481] 0.609953599 0.423795608 0.324495230 0.478912848 0.468718725  
## [486] 0.451845992 0.610711126 0.593359469 0.663301962 0.736260605  
## [491] 0.722094129 0.880884424 0.948855491 1.106079749 1.340444611  
## [496] 1.475182843 1.705761819 1.603974012 1.502942771 1.081889517  
## [501] 0.903275694 1.048723854 1.113599045 1.098857363 1.631096334  
## [506] 1.692996677 1.668443745 1.167654913 0.836469083 1.131390385  
## [511] 1.270418324 2.109797537 2.550686187 2.748017766 2.574721544  
## [516] 2.630786756 2.057039120 1.739860694 1.272205905 1.261500740  
## [521] 1.475875003 1.152663444 0.838783933 0.292660062 -0.096504722  
## [526] -0.627574581 -0.415845315 -0.499264082 -0.798817237 -0.656343934  
## [531] -0.296242717 -0.303107426 -0.457543686 -0.466176001 -0.328023256  
## [536] -0.407983103 -0.487539459 -0.209089622 -0.363565110 -0.512773386  
## [541] -0.234351639 -0.241172781 -0.461609172 -0.328738066 -0.265936903  
## [546] -0.482531518 -0.629130884 -0.635029139 -0.709844406 -0.721233856  
## [551] -0.728774102 -0.664152469 -1.014906950 -0.951732903 -0.819103485  
## [556] -1.099456661 -1.173637669 -0.972142368 -0.773150444 -0.577050163  
## [561] -0.513936550 -0.859919068 -0.866887825 -0.872089701 -0.605955146  
## [566] -0.613564700 -0.687249148 -0.359166895 -0.365330414 -0.439105409  
## [571] -0.645453950 -0.852552063 -0.922972950 -0.729003329 -0.866174816  
## [576] -0.936202542 -0.683865423 -0.755557648 -0.632105975 -0.643036506  
## [581] -0.648469878 -0.653882151 -0.593764112 -0.662856056 -0.476013632  
## [586] -0.420383931 -0.235035063 -0.113343312 -0.443431884 -0.449138932  
## [591] -0.708602591 -1.031153530 -1.223153589 -1.227344235 -1.293843754  
## [596] -1.172085087 -1.241199152 -1.371824029 -1.563618857 -1.753525834  
## [601] -1.818375874 -2.006404761 -2.069002572 -2.008180578 -1.761875035  
## [606] -1.826390869 -1.705758052 -1.832350194 -2.019616736 -1.961198453  
## [611] -1.963009611 -1.842184962 -1.783402567 -1.783402567 -1.722704538  
## [616] -1.180827211 -1.366873638 -1.489799283 -1.314398039 -1.197207445  
## [621] -0.840352393 -0.906077296 -0.849722391 -0.795122808 -0.681667567  
## [626] -0.269052775 0.256761141 -0.466993470 -0.351865227 0.230547602  
## [631] 0.099686777 -0.139977807 -0.038231943 -0.044194668 -0.050136501  
## [636] -0.058026638 0.217306444 0.032223037 -0.497486862 -0.268777417  
## [641] 0.064742175 -0.292318445 -0.751224290 -0.751224290 -0.877193955  
## [646] -1.330991842 -1.559678312 -1.845390115 -2.247527366 -2.306377237  
## [651] -2.082674190 -1.805709058 -2.203133277 -2.372790691 -1.981354661  
## [656] -1.703961972 -1.931706996 -1.429576457 -1.207828283 -0.986321575  
## [661] -0.716209462 -0.338739973 -0.455979723 -1.284525599 -1.559898015  
## [666] -1.506303181 -1.401618629 -1.244464191 -1.085855600 -1.414958284  
## [671] -1.526315625 -1.421562732 -1.430301817 -1.761975526 -1.711222595  
## [676] -1.169641006 -0.579765587 -0.317800479 -0.539134418 -0.920813448  
## [681] -0.928866102 -0.295270700 0.121249735 -0.148655962 -0.631173969  
## [686] -0.428983994 -0.279837147 -0.129812701 -0.607437993 -0.927053172  
## [691] -0.415269677 0.145674896 1.196508043 0.821812443 -0.152302397  
## [696] -0.152302397 0.503803406 0.137299168 -0.075527663 0.113735777  
## [701] 0.463883018 0.660320989 0.586331270 0.415105997 -1.444235013  
## [706] -2.039801337 -1.486758392 -0.943681274 -1.381818020 -1.044896137  
## [711] -0.676569677 -0.876454267 -0.762514286 -0.779184595 -1.144897240  
## [716] -1.556661316 -0.641916384 0.110707696 0.844006896 0.590312118  
## [721] 0.768699049 0.623104999 0.461604439 0.393158188 0.570715729  
## [726] 1.381680391 1.915385683 1.735603902 1.398216508 0.226805428  
## [731] -2.342420390 -3.458563235 -3.549983339 -3.427869980 -3.883811223  
## [736] -4.014324344 -4.457288318 -4.673124212 -5.414531737 -4.931287297  
## [741] -4.823857380 -3.660552240 -1.539843177 -0.661078556 -0.848983801  
## [746] -1.307811012 -1.175902633 -1.253558801 -1.452590751 -2.321015024  
## [751] -2.104463172 -2.292719479 -2.324227394 -2.276391024 -2.357626993  
## [756] -2.008777964 -1.749850664 -1.333695751 -0.850805937 -0.405495740  
## [761] -0.035838644 0.006049644 0.080959829 0.249879886 0.305404600  
## [766] 0.025322698 -0.043126500 -0.420211620 -0.471941892 -0.579357314  
## [771] -0.886250239 -1.188620768 -1.713319324 -1.795989803 -2.028774032  
## [776] -1.764451987 -1.505379346 -1.303560838 -1.656250944 -1.692127862  
## [781] -1.766294101 -1.438286819 -1.929003030 -2.303960320 -2.055521168  
## [786] -1.735092736 -1.568415660 -1.909242871 -2.347547491 -2.563355180  
## [791] -2.211000518 -1.934823675 -1.890582480 -2.322088944 -1.836503633  
## [796] -1.441242403 -1.292529630 -1.398261554 -1.481330212 -1.735776190  
## [801] -1.766304713 -1.839606848 -2.212329351 -2.785334966 -3.666528487  
## [806] -3.523402119 -3.458366474 -3.540417901 -3.401305679 -3.256921990  
## [811] -3.210900940 -3.195319634 -3.427490191 -3.308797562 -3.000963653  
## [816] -2.799639212 -2.206424731 -2.592624433 -2.548668092 -2.270528988  
## [821] -2.363630380 -2.362828954 -2.571718008 -2.386546491 -1.899557259  
## [826] -1.764461771 -1.766026891 -1.406279576 -0.956932165 -0.664737110  
## [831] -1.024457491 -1.283450995 -1.597009338 -1.809078837 -1.725936442  
## [836] -1.526563935 -1.279201801 -1.435721365 -1.287101234 -1.328768037  
## [841] -1.307828134 -1.198100187 -1.132134716 -0.995371798 -0.692403590  
## [846] -0.667471296 -0.622179345 -0.827730718 -1.131054559 -0.974845005  
## [851] -1.311734464 -1.453735840 -1.897555607 -1.927706175 -1.600158589  
## [856] -1.464117951 -1.658708407 -1.800109707 -1.672227506 -1.704633188  
## [861] -1.734282800 -1.682617344 -1.395105692 -1.148710736 -0.966666155  
## [866] -1.123960187 -1.905246455 -3.091722669 -3.210179500 -2.722859917  
## [871] -2.427302151 -2.135753776 -2.074171305 -2.260735199 -2.275537955  
## [876] -2.122958104 -2.051188886 -1.757151009 -0.839817222 0.611185638  
## [881] 1.361731197 1.710525278 1.653419080 1.621848714 1.807380007  
## [886] 2.614602972 3.200839962 3.511104298 3.884325665 4.217954440  
## [891] 4.735689156 4.470449411 4.723815184 5.119938038 4.641617341  
## [896] 4.470069528 4.458512579 4.039614997 3.455945687 2.809434717  
## [901] 2.717528784 2.377762497  
##   
## $call  
## arima(x = x, order = c(p, d, q), seasonal = season, include.mean = intercept,   
## transform.pars = tr.pars, fixed = fix, method = METHOD)  
##   
## $series  
## [1] "x"  
##   
## $code  
## [1] 0  
##   
## $n.cond  
## [1] 0  
##   
## $nobs  
## [1] 902  
##   
## $model  
## $model$phi  
## numeric(0)  
##   
## $model$theta  
## numeric(0)  
##   
## $model$Delta  
## numeric(0)  
##   
## $model$Z  
## [1] 1  
##   
## $model$a  
## [1] 2.377762  
##   
## $model$P  
## [,1]  
## [1,] 0  
##   
## $model$T  
## [,1]  
## [1,] 0  
##   
## $model$V  
## [,1]  
## [1,] 1  
##   
## $model$h  
## [1] 0  
##   
## $model$Pn  
## [,1]  
## [1,] 1  
##   
##   
## attr(,"class")  
## [1] "estimate"