Advanced Econometrics Project

Contents

Date: 22/04/2024

Module title: Advanced Econometrics

Course: BSc Economics with Econometrics

1 Introduction

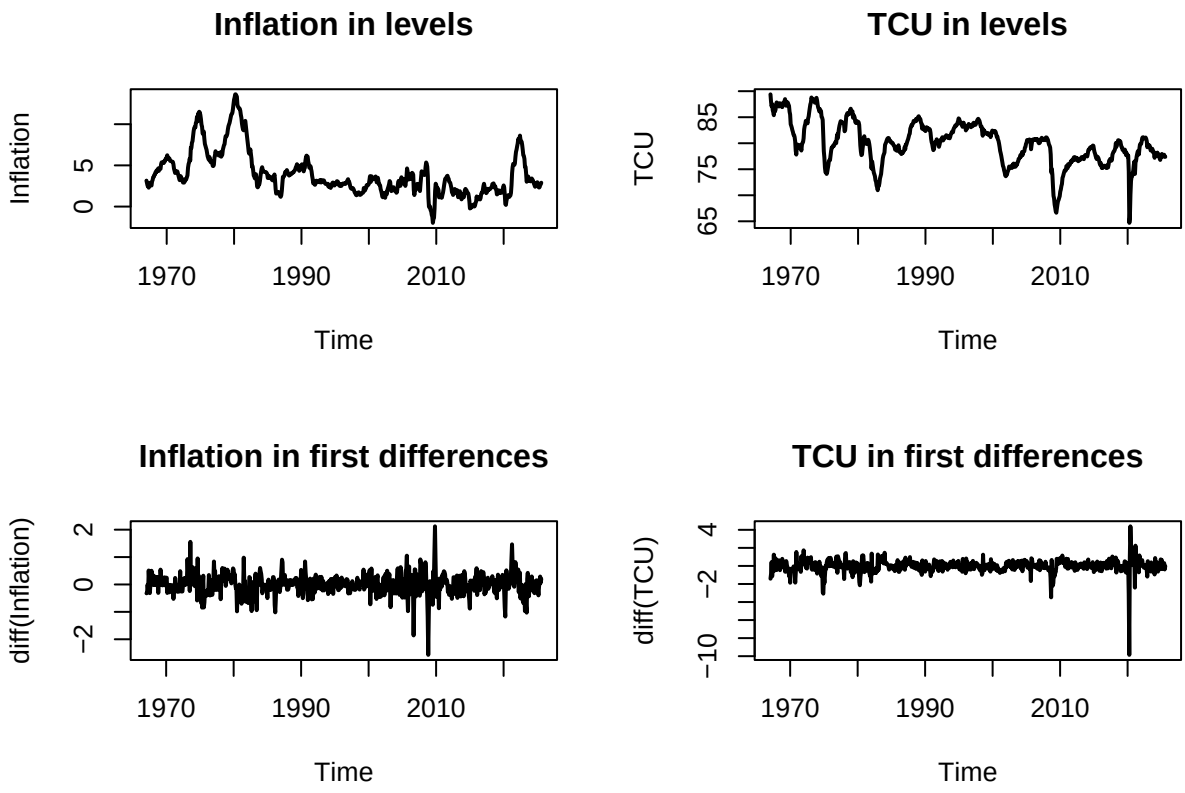
In this project we will use Inflation growth which is calculated from the Consumer Price Index Data set, and the TCU(capacity utilisation), which measures the extent to which a nation utilizes its productive capacity. We will try to understand whether these two variables cause one another, and we will do some forecasting for the inflation series. In the first part we plot the data, then we will check if the series have a unit root. Further we will check of these series are co integrated. If they are co integrated we will run the VEC model to understand their long run relationship. Prior to that we will run the ARDL model to understand how statistically significant are the past values to the present one for these two series.

```
rm(list=ls())
      # Inflation rate in January 1948 = (CPI in Jan 1948 - CPI in Jan 1947)/ (CPI in Jan 1947) = (CPI
      # Hence we need to set: "Inflation = diff(log(CPI), lag = 12)*100"

      # USA CPI for All Urban Consumers      - Seasonally Adjusted - Monthly (from January 1947):
      # USA CPI Inflation rate, year on year - Seasonally Adjusted - Monthly (from January 1948)

CPI = pdfetch_FRED("CPIAUCSL")
names(CPI) = "CPI"

Inflation = diff(log(CPI), lag = 12) * 100                    # Percent change from year ago
names(Inflation) = "Inflation"

Inflation = ts(Inflation, start=c(1947, 1), frequency=12)
Inflation = na.omit(Inflation)                                # inflation series begins in January 1948

      # Capacity Utilization: Total Index (TCU) - Percent of Capacity, Seasonally Adjusted - Monthly -

TCU = pdfetch_FRED("TCU")
names(TCU) = "TCU"

TCU = ts( TCU , start=c(1967, 1), frequency=12)
#Inflation = log( Inflation   * 100 )

# dataset in logs in logs
data.set = na.omit(
           ts.intersect(

        Inflation,     # inflation is already the log of CPI which is the inflation growth
        TCU,
           dframe=TRUE))

      Inflation   = ts( data.set$Inflation,    start=c(1967, 1),  frequency=12)
      TCU = ts( data.set$TCU,    start=c(1967, 1),  frequency=12)
```

```r
cor(Inflation,TCU)
```

```
## [1] 0.3659523
```

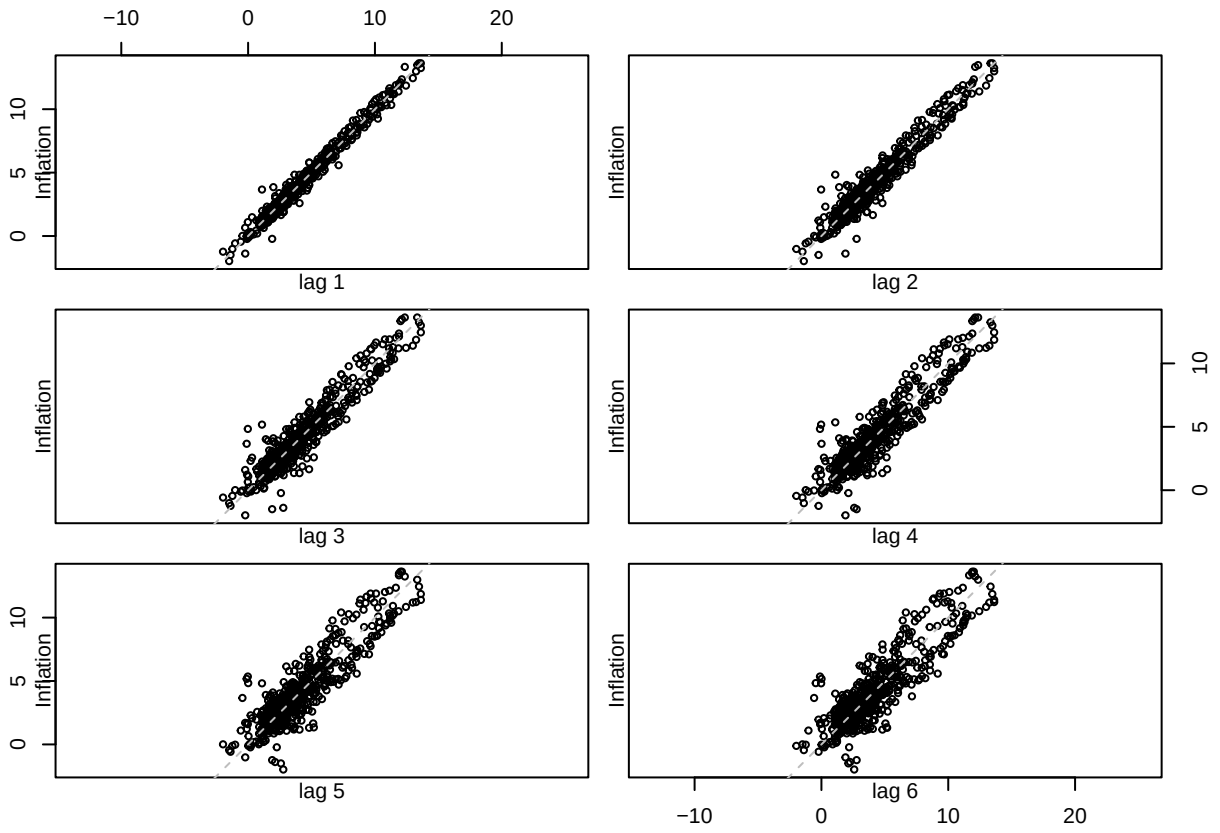
```r
    par( mfrow=c(2,2))
      plot( Inflation              , main = "Inflation in levels"                , lwd = 2 )
      plot( TCU                , main = "TCU in levels"                , lwd = 2 )
      plot( diff(Inflation)        , main = "Inflation in first differences"     , lwd = 2 )
      plot( diff(TCU)          , main = "TCU in first differences"     , lwd = 2 )
```



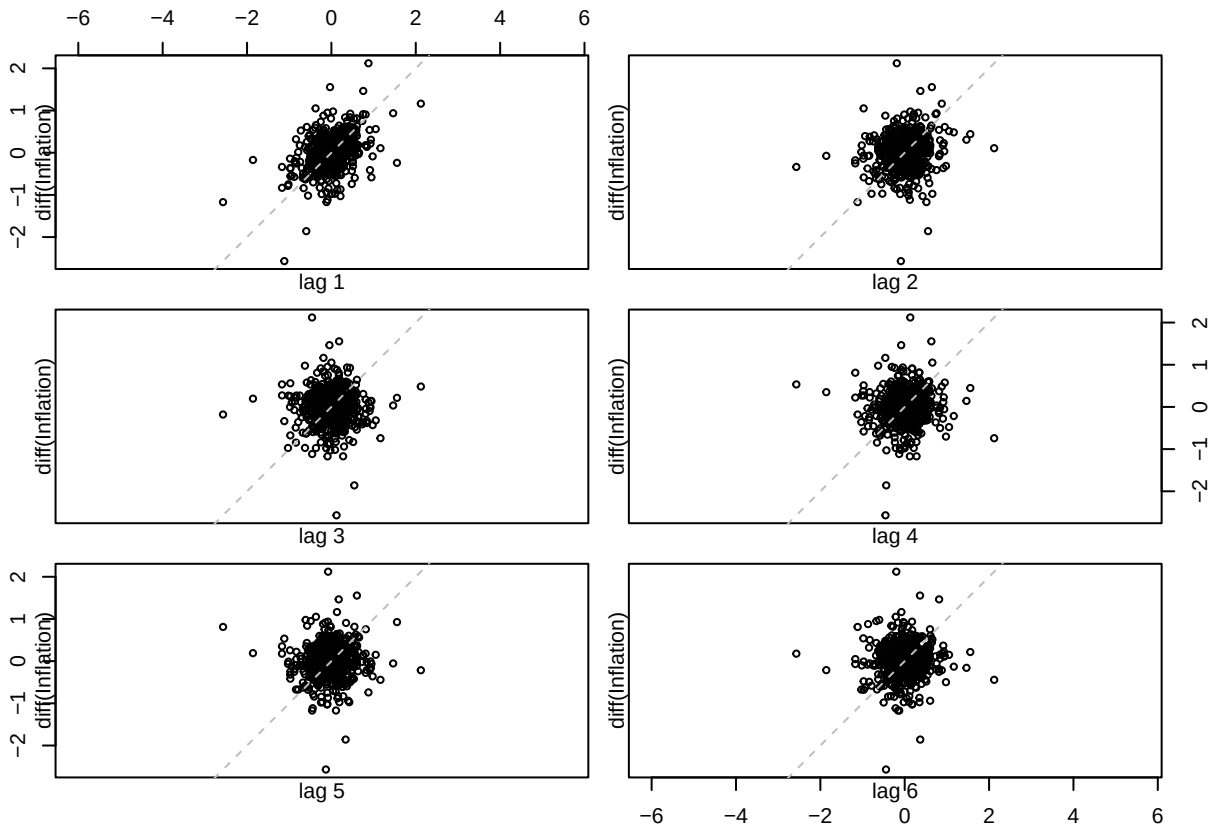
Analysing both plots we can see that TCU has a higher volatility during 2020. which creates residuals that are not normally distributed, hence it will affect the tests overall.

2 Correlation between Inflation in levels and its lags

```r
    lag.plot( Inflation,       6, do.lines=FALSE)       # levels
```

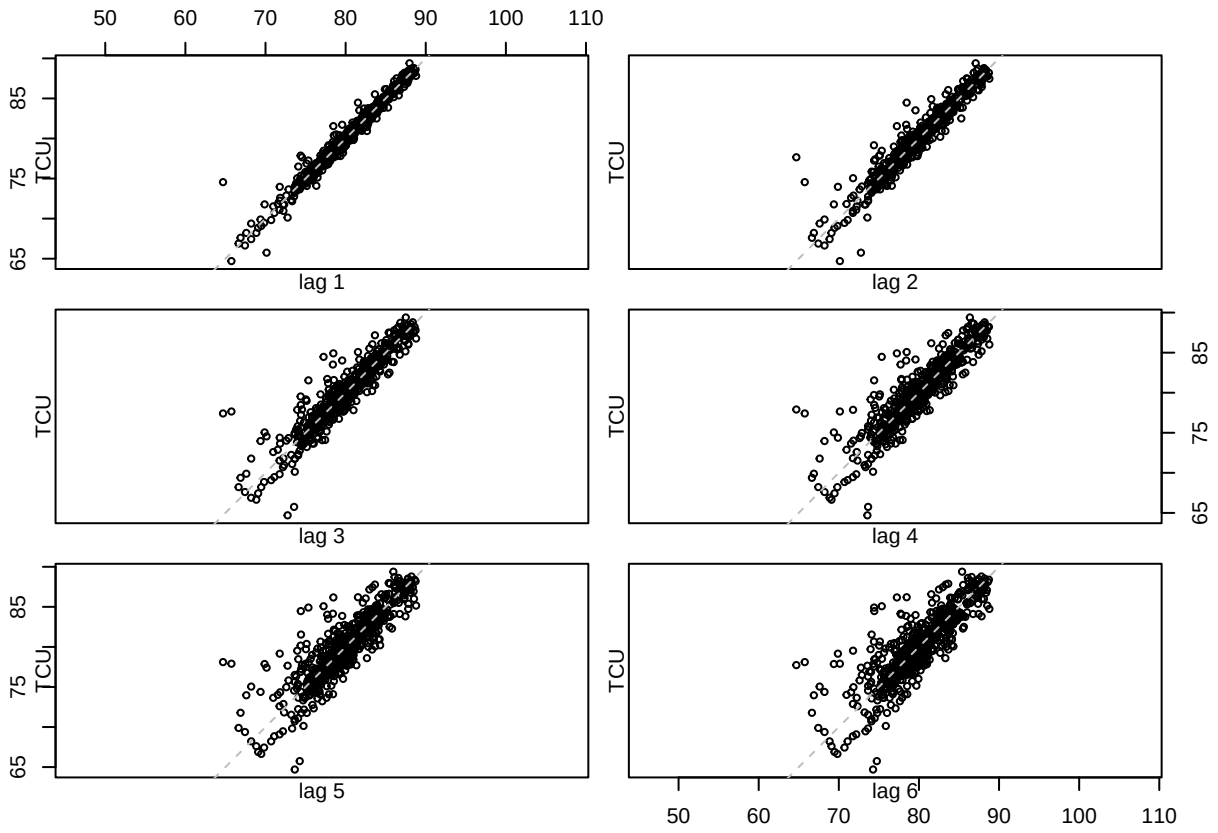



```
lag.plot( diff(Inflation), 6, do.lines=FALSE)       # first diference
```

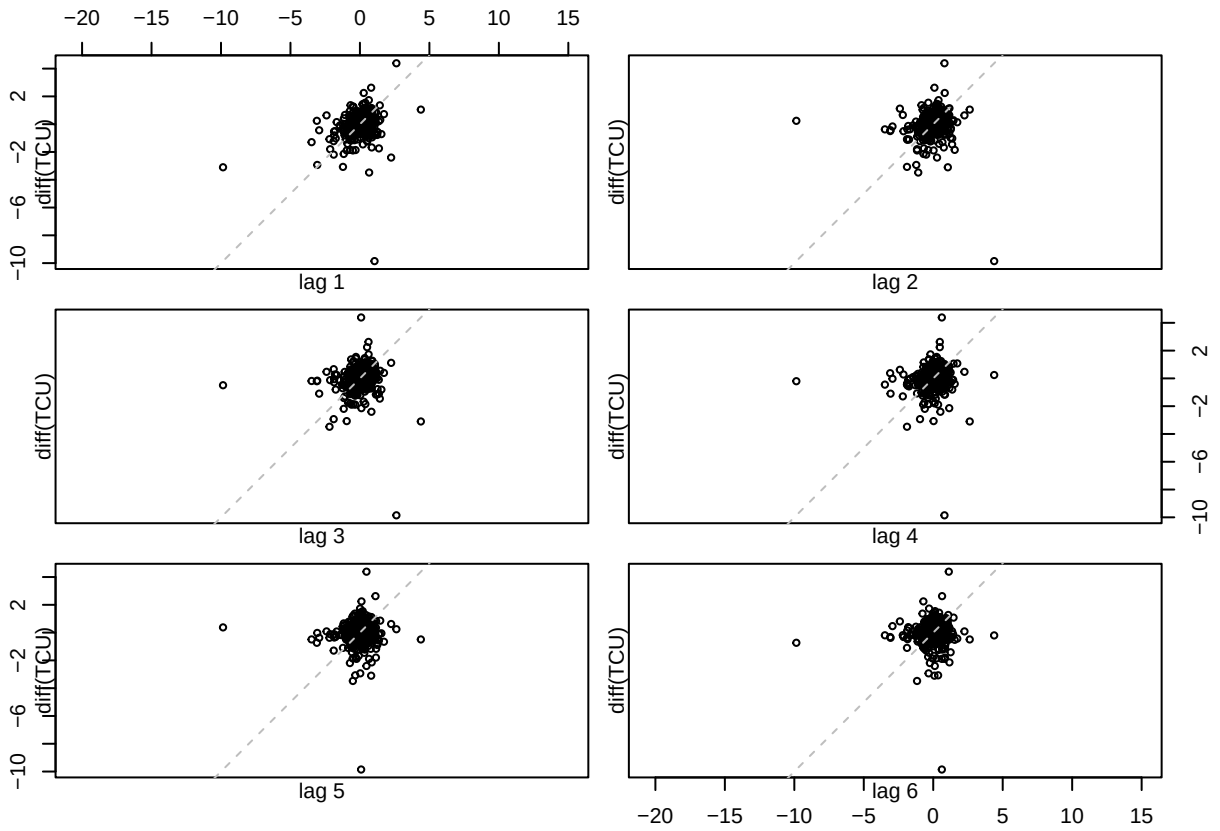


From these graphs we can infer that there is correlation between the inflation variable at time present and it's the past values. When there is correlation over time it could mean that there is persistence, hence the possibility of having a unit root.

```
    lag.plot( TCU,       6, do.lines=FALSE)       # levels
```

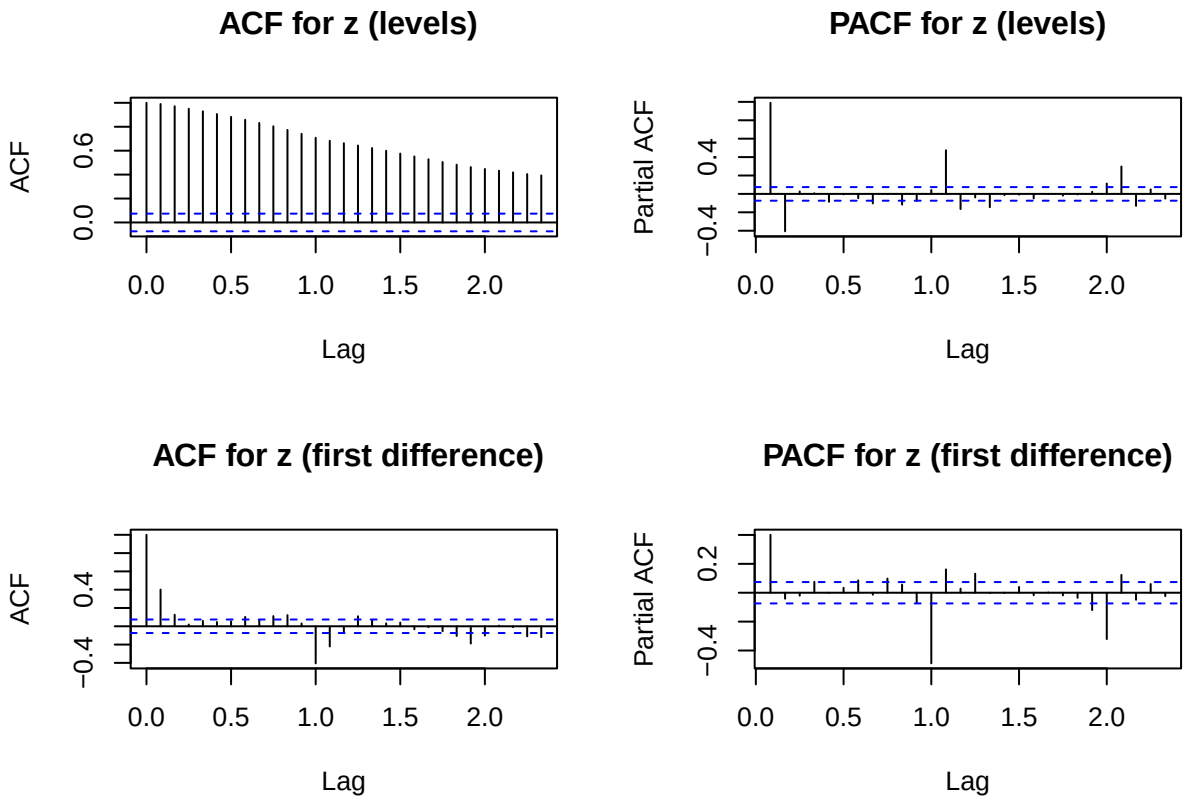



```
lag.plot( diff(TCU), 6, do.lines=FALSE)       # first diference
```

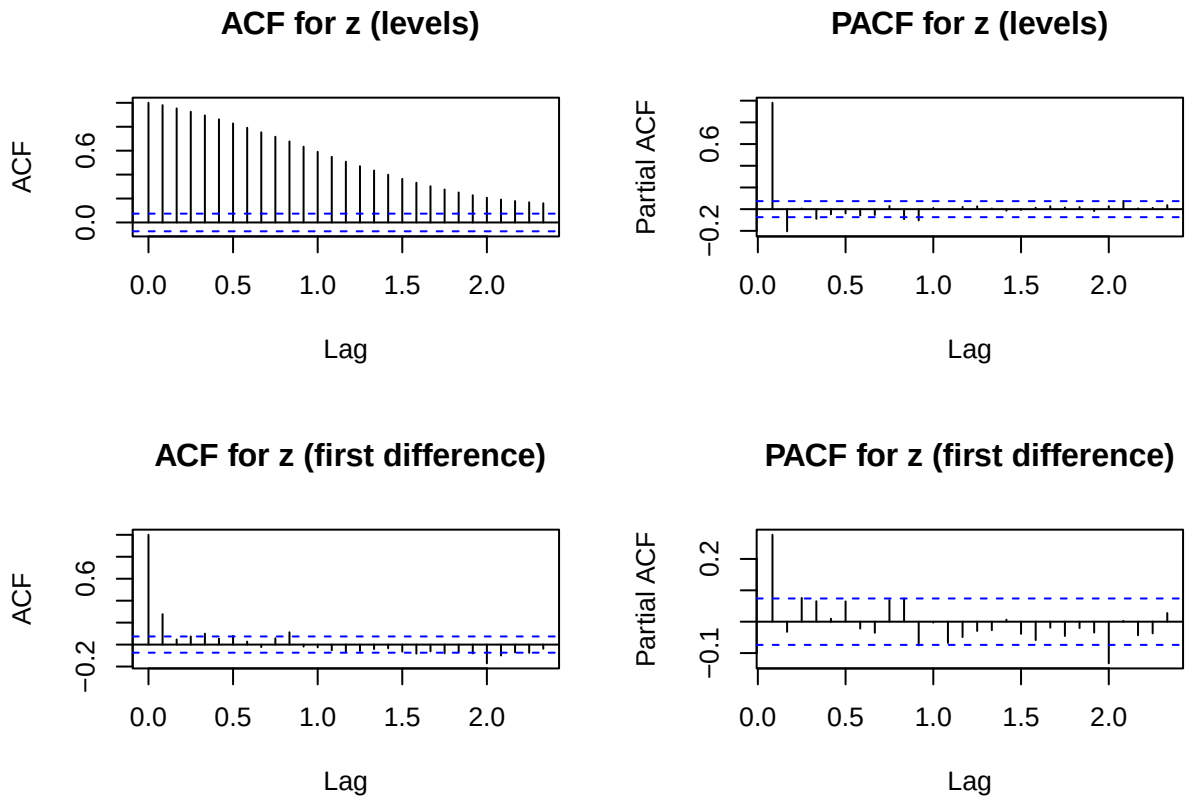


From these graphs we can infer that there is correlation between the TCU variable at time present and it's the past values. When there is correlation over time it could mean that there is persistence, hence the possibility of having a unit root.

```
par(mfrow=c(2,2))
   acf(       Inflation   , main=" ACF for z (levels)"           )
  pacf(       Inflation   , main="PACF for z (levels)"           )
   acf(  diff(Inflation)  , main=" ACF for z (first difference)" )
  pacf(  diff(Inflation)  , main="PACF for z (first difference)" )
```




```
par(mfrow=c(2,2))
   acf(       TCU   , main=" ACF for z (levels)"            )
  pacf(       TCU   , main="PACF for z (levels)"            )
   acf(  diff(TCU)  , main=" ACF for z (first difference)"  )
  pacf(  diff(TCU)  , main="PACF for z (first difference)"  )
```



These graphs indicate strong persistence. The Strong persistences disappears when taking first differences this could indicate that there is a unit root in both series.

3 Unit root tests.

In this part of the project we will run the unit root tests. We will run the Augmented Dickey-Fuller Test Unit Root Test, the Phillips-Perron Unit Root Test, and the KPSS Unit Root Test.

3.1 Unit root test on the inflation data in levels.

```
    # Maximum number of lags ("p max") to be used in the unit root tests:
    # Following formula 9.4.31 in Hayashi (2000, p.594, chapter 9)

  max.lags = (12*((length(Inflation)/100) ^(1/4))) |> trunc()
  max.lags
```

```
## [1] 19
```

```
    # Now run each unit root test:

    # Augmented Dickey-Fuller(ADF) tests on the chosen series:
    # Using BIC to determine the number of lags
```

```r
    # ADF:  Ho = series has a unit root = integrated I(1) process

  ur.df( Inflation,  type="none"   , selectlags="BIC",  lags=max.lags ) |> summary()    # from packag
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.64412 -0.13273  0.01993  0.18515  1.62682
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## z.lag.1      -0.004171   0.002440  -1.710 0.087799 .
## z.diff.lag1   0.461813   0.038198  12.090  < 2e-16 ***
## z.diff.lag2  -0.022019   0.042087  -0.523 0.601026
## z.diff.lag3   0.039233   0.041696   0.941 0.347087
## z.diff.lag4   0.034421   0.035887   0.959 0.337831
## z.diff.lag5   0.020324   0.035552   0.572 0.567741
## z.diff.lag6  -0.024310   0.035466  -0.685 0.493308
## z.diff.lag7   0.094854   0.035449   2.676 0.007638 **
## z.diff.lag8  -0.024307   0.035609  -0.683 0.495088
## z.diff.lag9   0.035948   0.035453   1.014 0.310964
## z.diff.lag10  0.071615   0.035427   2.021 0.043628 *
## z.diff.lag11  0.130440   0.035528   3.672 0.000260 ***
## z.diff.lag12 -0.546550   0.035853 -15.244  < 2e-16 ***
## z.diff.lag13  0.152671   0.041531   3.676 0.000256 ***
## z.diff.lag14 -0.030806   0.041936  -0.735 0.462851
## z.diff.lag15  0.139864   0.038126   3.668 0.000263 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2949 on 668 degrees of freedom
## Multiple R-squared:  0.4209, Adjusted R-squared:  0.4071
## F-statistic: 30.35 on 16 and 668 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -1.7096
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```r
    ur.df( Inflation,  type="drift"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # 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
## -1.66819 -0.15234  0.00396  0.16446  1.61904
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.055125   0.021229   2.597 0.009622 **
## z.lag.1      -0.014295   0.004594  -3.112 0.001939 **
## z.diff.lag1   0.463500   0.038040  12.185  < 2e-16 ***
## z.diff.lag2  -0.016265   0.041965  -0.388 0.698456
## z.diff.lag3   0.046360   0.041609   1.114 0.265599
## z.diff.lag4   0.036326   0.035741   1.016 0.309830
## z.diff.lag5   0.023970   0.035428   0.677 0.498907
## z.diff.lag6  -0.019984   0.035354  -0.565 0.572088
## z.diff.lag7   0.099287   0.035339   2.810 0.005106 **
## z.diff.lag8  -0.019249   0.035510  -0.542 0.587955
## z.diff.lag9   0.041706   0.035371   1.179 0.238779
## z.diff.lag10  0.077338   0.035344   2.188 0.029007 *
## z.diff.lag11  0.137145   0.035470   3.867 0.000121 ***
## z.diff.lag12 -0.538270   0.035842 -15.018  < 2e-16 ***
## z.diff.lag13  0.155910   0.041372   3.768 0.000179 ***
## z.diff.lag14 -0.026257   0.041794  -0.628 0.530052
## z.diff.lag15  0.148456   0.038107   3.896 0.000108 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2936 on 667 degrees of freedom
## Multiple R-squared:  0.4267, Adjusted R-squared:  0.413
## F-statistic: 31.03 on 16 and 667 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -3.1118 4.8452
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

```r
    ur.df( Inflation,  type="trend"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.65109 -0.15738  0.00436  0.16546  1.57803
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1478995  0.0440786   3.355 0.000838 ***
## z.lag.1      -0.0223817  0.0056847  -3.937 9.11e-05 ***
## tt           -0.0001689  0.0000704  -2.399 0.016709 *
## z.diff.lag1   0.4644612  0.0379072  12.253  < 2e-16 ***
## z.diff.lag2  -0.0121356  0.0418519  -0.290 0.771932
## z.diff.lag3   0.0512598  0.0415116   1.235 0.217329
## z.diff.lag4   0.0375915  0.0356185   1.055 0.291628
## z.diff.lag5   0.0265593  0.0353190   0.752 0.452326
## z.diff.lag6  -0.0168463  0.0352532  -0.478 0.632901
## z.diff.lag7   0.1027071  0.0352425   2.914 0.003685 **
## z.diff.lag8  -0.0155859  0.0354174  -0.440 0.660034
## z.diff.lag9   0.0458456  0.0352880   1.299 0.194330
## z.diff.lag10  0.0812330  0.0352564   2.304 0.021526 *
## z.diff.lag11  0.1417849  0.0353972   4.006 6.88e-05 ***
## z.diff.lag12 -0.5323335  0.0358007 -14.869  < 2e-16 ***
## z.diff.lag13  0.1581490  0.0412361   3.835 0.000137 ***
## z.diff.lag14 -0.0231787  0.0416657  -0.556 0.578191
## z.diff.lag15  0.1541305  0.0380457   4.051 5.70e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2926 on 666 degrees of freedom
## Multiple R-squared:  0.4316, Adjusted R-squared:  0.4171
## F-statistic: 29.75 on 17 and 666 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -3.9372 5.1717 7.754
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

We do not reject the null hypotheses in levels, This means that the time series is non stationary. Unit root I(1). A unit root implies that the series has a stochastic trend.

```
      # Philips-Perron(PP) tests on the chosen series:
      # PP:  Ho = series has a unit root = integrated I(1) process
      # PP test = DF test + robust to serial correlation by using the Newey-West HAC covariance matrix


    ur.pp( Inflation, type="Z-tau"   ,  model="constant" , lags="long" ) |> summary()    # from package
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.57122 -0.20779 -0.01104  0.20254  2.07970
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.038757   0.025294   1.532    0.126
## y.l1        0.989963   0.005349 185.061   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3797 on 701 degrees of freedom
## Multiple R-squared:  0.9799, Adjusted R-squared:  0.9799
## F-statistic: 3.425e+04 on 1 and 701 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -2.9074
##
##          aux. Z statistics
## Z-tau-mu            2.3873
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.442093 -2.866012 -2.569155
```

```
    ur.pp( Inflation, type="Z-tau"   ,  model="trend"    , lags="long" ) |> summary()
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
```

```
## Test regression with intercept and trend
##
##
## Call:
## lm(formula = y ~ y.l1 + trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.55416 -0.20849 -0.02132  0.20263  2.07990
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.717e-02  2.819e-02   2.028    0.043 *
## y.l1         9.853e-01  6.227e-03 158.212   <2e-16 ***
## trend       -1.210e-04  8.215e-05  -1.473    0.141
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3794 on 700 degrees of freedom
## Multiple R-squared:   0.98,  Adjusted R-squared:  0.9799
## F-statistic: 1.715e+04 on 2 and 700 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -3.5083
##
##            aux. Z statistics
## Z-tau-mu             -0.4002
## Z-tau-beta           -1.9629
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.975778 -3.418381 -3.131355
```

We reject the null hypothesis tha there is a unit root, so there is no unit root at the 5 percent confidence interval using the Philips-Perron(PP) test.

```
    # KPSS tests on the chosen series:
    # KPSS:   Ho = series does not have a unit root [opposite of ADF and PP]
    #
    # "mu"  = constant
    # "tau" = constant + linear trend


  ur.kpss( Inflation, type="mu"   ,  lags="long" ) |> summary()     # from package "urca"
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 19 lags.
##
## Value of test-statistic is: 1.3972
```

```
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
    ur.kpss( Inflation, type="tau" ,  lags="long" ) |> summary()
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: tau with 19 lags.
##
## Value of test-statistic is: 0.1986
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.119 0.146  0.176 0.216
```

We reject the null hypotheses which means that according to the test there is a unit root, thus being non stationary, having a stochastic trend.

Overall out of 3 tests tow of them show that there is a unit root, while the Philips-Perron(PP) test shows that there is no unit root at the 5% confidence intervall.

3.1.1 Unit root for inflation time series in first differences.

```
      # Augmented Dickey-Fuller(ADF) tests on the chosen series:
      # Using BIC to determine the number of lags
      # ADF:  Ho = series has a unit root = integrated I(1) process


    ur.df( diff(Inflation),  type="none"    , selectlags="BIC",  lags=max.lags ) |> summary()     # from
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.65692 -0.15126  0.00485  0.17156  1.60697
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## z.lag.1      -0.496997   0.078181  -6.357 3.81e-10 ***
## z.diff.lag1  -0.041855   0.076747  -0.545 0.585685
## z.diff.lag2  -0.065737   0.073100  -0.899 0.368835
## z.diff.lag3  -0.029342   0.067159  -0.437 0.662323
## z.diff.lag4   0.004643   0.066687   0.070 0.944517
## z.diff.lag5   0.023517   0.065461   0.359 0.719523
## z.diff.lag6  -0.002323   0.063600  -0.037 0.970878
## z.diff.lag7   0.090602   0.061377   1.476 0.140376
## z.diff.lag8   0.064430   0.059428   1.084 0.278683
## z.diff.lag9   0.097958   0.056322   1.739 0.082452 .
## z.diff.lag10  0.167438   0.053420   3.134 0.001798 **
## z.diff.lag11  0.294979   0.050423   5.850 7.69e-09 ***
## z.diff.lag12 -0.254977   0.047490  -5.369 1.09e-07 ***
## z.diff.lag13 -0.104004   0.043424  -2.395 0.016890 *
## z.diff.lag14 -0.136434   0.038180  -3.573 0.000378 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2955 on 668 degrees of freedom
## Multiple R-squared:  0.5113, Adjusted R-squared:  0.5003
## F-statistic: 46.59 on 15 and 668 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -6.357
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
    ur.df( diff(Inflation),  type="drift"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # 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
## -1.65606 -0.15041  0.00572  0.17242  1.60785
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0008639  0.0113172  -0.076 0.939175
## z.lag.1      -0.4970415  0.0782412  -6.353 3.92e-10 ***
## z.diff.lag1  -0.0418190  0.0768053  -0.544 0.586292
## z.diff.lag2  -0.0657110  0.0731555  -0.898 0.369384
## z.diff.lag3  -0.0293210  0.0672092  -0.436 0.662786
```

```
## z.diff.lag4   0.0046543  0.0667372   0.070 0.944421
## z.diff.lag5   0.0235226  0.0655101   0.359 0.719658
## z.diff.lag6  -0.0023217  0.0636470  -0.036 0.970912
## z.diff.lag7   0.0906033  0.0614231   1.475 0.140666
## z.diff.lag8   0.0644275  0.0594725   1.083 0.279060
## z.diff.lag9   0.0979558  0.0563643   1.738 0.082689 .
## z.diff.lag10  0.1674341  0.0534597   3.132 0.001812 **
## z.diff.lag11  0.2949767  0.0504607   5.846 7.90e-09 ***
## z.diff.lag12 -0.2549769  0.0475253  -5.365 1.12e-07 ***
## z.diff.lag13 -0.1040010  0.0434561  -2.393 0.016976 *
## z.diff.lag14 -0.1364360  0.0382088  -3.571 0.000381 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2957 on 667 degrees of freedom
## Multiple R-squared:  0.5113, Adjusted R-squared:  0.5003
## F-statistic: 46.52 on 15 and 667 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -6.3527 20.1787
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

```
    ur.df( diff(Inflation),  type="trend"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.65532 -0.15126  0.00612  0.17255  1.60638
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.023e-03  2.363e-02   0.043 0.965482
## z.lag.1      -4.972e-01  7.833e-02  -6.348 4.04e-10 ***
## tt           -5.228e-06  5.747e-05  -0.091 0.927545
## z.diff.lag1  -4.163e-02  7.689e-02  -0.541 0.588392
## z.diff.lag2  -6.552e-02  7.324e-02  -0.895 0.371309
## z.diff.lag3  -2.916e-02  6.728e-02  -0.433 0.664918
## z.diff.lag4   4.820e-03  6.681e-02   0.072 0.942505
## z.diff.lag5   2.368e-02  6.558e-02   0.361 0.718190
## z.diff.lag6  -2.170e-03  6.372e-02  -0.034 0.972847
```

```
## z.diff.lag7   9.075e-02  6.149e-02   1.476 0.140464
## z.diff.lag8   6.457e-02  5.954e-02   1.084 0.278538
## z.diff.lag9   9.808e-02  5.642e-02   1.738 0.082621 .
## z.diff.lag10  1.675e-01  5.351e-02   3.131 0.001819 **
## z.diff.lag11  2.951e-01  5.051e-02   5.842 8.07e-09 ***
## z.diff.lag12 -2.549e-01  4.757e-02  -5.359 1.15e-07 ***
## z.diff.lag13 -1.040e-01  4.349e-02  -2.390 0.017107 *
## z.diff.lag14 -1.364e-01  3.824e-02  -3.567 0.000387 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.296 on 666 degrees of freedom
## Multiple R-squared:  0.5113, Adjusted R-squared:  0.4996
## F-statistic: 43.55 on 16 and 666 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -6.3479 13.4352 20.1524
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
    # Philips-Perron(PP) tests on the chosen series:
    # PP:  Ho = series has a unit root = integrated I(1) process
    # PP test = DF test + robust to serial correlation by using the Newey-West HAC covariance matrix

  ur.pp( diff(Inflation), type="Z-tau"  ,  model="constant" , lags="long" ) |> summary()    # from pa
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09967 -0.17667  0.00672  0.19336  1.65369
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0003584  0.0131628   0.027    0.978
## y.l1        0.4011490  0.0346093  11.591   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3488 on 700 degrees of freedom
## Multiple R-squared:  0.161,  Adjusted R-squared:  0.1598
```

```
## F-statistic: 134.3 on 1 and 700 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -17.4612
##
##          aux. Z statistics
## Z-tau-mu            0.0268
##
## Critical values for Z statistics:
##                       1pct      5pct     10pct
## critical values -3.442105 -2.866017 -2.569158
```

```
    ur.pp( diff(Inflation), type="Z-tau"   ,  model="trend"     , lags="long" ) |> summary()
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept and trend
##
##
## Call:
## lm(formula = y ~ y.l1 + trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09745 -0.17523  0.00446  0.19271  1.65635
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0003662  0.0131717   0.028    0.978
## y.l1         0.4010430  0.0346354  11.579   <2e-16 ***
## trend       -0.0000157  0.0000650  -0.242    0.809
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.349 on 699 degrees of freedom
## Multiple R-squared:  0.1611, Adjusted R-squared:  0.1587
## F-statistic: 67.11 on 2 and 699 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -17.4469
##
##            aux. Z statistics
## Z-tau-mu              0.0271
## Z-tau-beta           -0.2434
##
## Critical values for Z statistics:
##                       1pct     5pct    10pct
## critical values -3.975795 -3.41839 -3.13136
```

```
    # KPSS tests on the chosen series:
    # KPSS:   Ho = series does not have a unit root [opposite of ADF and PP]
    #
    # "mu"  = constant
    # "tau" = constant + linear trend


  ur.kpss( diff(Inflation), type="mu"   ,  lags="long" ) |> summary()     # from package "urca"
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 19 lags.
##
## Value of test-statistic is: 0.0328
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
  ur.kpss( diff(Inflation), type="tau" ,  lags="long" ) |> summary()
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: tau with 19 lags.
##
## Value of test-statistic is: 0.0317
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.119 0.146  0.176 0.216
```

The Unit root tests show that there is a unit root in levels for the Inflation, while there is strong evidence that there is no unit root for the inflation in first differences.

3.2 Unit root test for TCU in levels.

```
    # Maximum number of lags ("p max") to be used in the unit root tests:
    # Following formula 9.4.31 in Hayashi (2000, p.594, chapter 9)

  max.lags = (12*((length(TCU)/100) ^(1/4))) |> trunc()
  max.lags
```

```
## [1] 19
```

```r
    # Augmented Dickey-Fuller(ADF) tests on the chosen series:
    # Using BIC to determine the number of lags
    # ADF:  Ho = series has a unit root = integrated I(1) process


  ur.df( TCU,  type="none"   , selectlags="BIC",  lags=max.lags ) |> summary()    # from package "urc
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9613 -0.2990  0.0159  0.3086  4.0994
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.0001839  0.0003344  -0.550    0.583
## z.diff.lag  0.2836280  0.0367118   7.726 3.98e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6989 on 682 degrees of freedom
## Multiple R-squared:  0.08103,    Adjusted R-squared:  0.07834
## F-statistic: 30.07 on 2 and 682 DF,  p-value: 3.055e-13
##
##
## Value of test-statistic is: -0.55
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```r
  ur.df( TCU,  type="drift"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # 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
## -9.0557 -0.2992  0.0256  0.3127  3.7613
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.835376   0.533794   3.438 0.000621 ***
## z.lag.1     -0.023117   0.006678  -3.462 0.000570 ***
## z.diff.lag   0.293691   0.036541   8.037 4.06e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6934 on 681 degrees of freedom
## Multiple R-squared:  0.09637,    Adjusted R-squared:  0.09372
## F-statistic: 36.31 on 2 and 681 DF,  p-value: 1.034e-15
##
##
## Value of test-statistic is: -3.4617 6.0648
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

```
    ur.df( TCU,  type="trend"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0062 -0.2930  0.0265  0.3075  3.7380
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.4422273  0.6478248   3.770 0.000178 ***
## z.lag.1     -0.0295579  0.0077293  -3.824 0.000143 ***
## tt          -0.0002563  0.0001554  -1.649 0.099642 .
## z.diff.lag   0.2976734  0.0365752   8.139  1.9e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6926 on 680 degrees of freedom
## Multiple R-squared:  0.09997,    Adjusted R-squared:  0.096
## F-statistic: 25.18 on 3 and 680 DF,  p-value: 1.869e-15
```

```
##
##
## Value of test-statistic is: -3.8241 4.9596 7.366
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
    # Philips-Perron(PP) tests on the chosen series:
    # PP:  Ho = series has a unit root = integrated I(1) process
    # PP test = DF test + robust to serial correlation by using the Newey-West HAC covariance matrix


  ur.pp( TCU, type="Z-tau"   ,  model="constant" , lags="long" ) |> summary()    # from package "urca"
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.9417 -0.2998  0.0583  0.3505  4.1302
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.508470   0.532335   2.834  0.00473 **
## y.l1        0.980936   0.006643 147.656  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7233 on 701 degrees of freedom
## Multiple R-squared:  0.9688, Adjusted R-squared:  0.9688
## F-statistic: 2.18e+04 on 1 and 701 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -3.8198
##
##          aux. Z statistics
## Z-tau-mu            3.7931
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.442093 -2.866012 -2.569155
```

```
    ur.pp( TCU, type="Z-tau"   ,  model="trend"     , lags="long" ) |> summary()
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept and trend
##
##
## Call:
## lm(formula = y ~ y.l1 + trend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.9164 -0.2916  0.0662  0.3413  4.1133
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.8959327  0.6333066   2.994  0.00285 **
## y.l1         0.9760958  0.0079060 123.463  < 2e-16 ***
## trend       -0.0001806  0.0001600  -1.129  0.25934
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7231 on 700 degrees of freedom
## Multiple R-squared:  0.9689, Adjusted R-squared:  0.9688
## F-statistic: 1.091e+04 on 2 and 700 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -4.3019
##
##            aux. Z statistics
## Z-tau-mu             -0.2593
## Z-tau-beta           -1.8757
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.975778 -3.418381 -3.131355
```

We reject the null hyphothesis so there is not a unit root in levels

```
    # KPSS tests on the chosen series:
    # KPSS:   Ho = series does not have a unit root [opposite of ADF and PP]
    #
    # "mu"  = constant
    # "tau" = constant + linear trend


   ur.kpss( TCU, type="mu"   ,  lags="long" ) |> summary()     # from package "urca"
```

```
##
```

```
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 19 lags.
##
## Value of test-statistic is: 1.3588
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
    ur.kpss( TCU, type="tau" ,  lags="long" ) |> summary()
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: tau with 19 lags.
##
## Value of test-statistic is: 0.0819
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.119 0.146  0.176 0.216
```

3.2.1 Unit root test for TCU in first differences.

```
      # Maximum number of lags ("p max") to be used in the unit root tests:
      # Following formula 9.4.31 in Hayashi (2000, p.594, chapter 9)

    max.lags = (12*((length(diff(TCU))/100) ^(1/4))) |> trunc()
    max.lags
```

```
## [1] 19
```

```
      # Augmented Dickey-Fuller(ADF) tests on the chosen series:
      # Using BIC to determine the number of lags
      # ADF:  Ho = series has a unit root = integrated I(1) process


    ur.df( diff(TCU),  type="none"    , selectlags="BIC",  lags=max.lags ) |> summary()    # from packag
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
```

```
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9340 -0.3265 -0.0030  0.2981  3.8379
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.74150    0.04583 -16.178   <2e-16 ***
## z.diff.lag  0.03554    0.03830   0.928    0.354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6992 on 681 degrees of freedom
## Multiple R-squared:  0.3588, Adjusted R-squared:  0.357
## F-statistic: 190.6 on 2 and 681 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -16.1784
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
    ur.df( diff(TCU),  type="drift"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # 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
## -8.9241 -0.3159  0.0075  0.3087  3.8456
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01054    0.02678  -0.394    0.694
## z.lag.1     -0.74194    0.04587 -16.173   <2e-16 ***
## z.diff.lag   0.03576    0.03833   0.933    0.351
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6996 on 680 degrees of freedom
## Multiple R-squared:  0.359,  Adjusted R-squared:  0.3571
```

```
## F-statistic: 190.4 on 2 and 680 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -16.1732 130.7861
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

```
    ur.df( diff(TCU),  type="trend"  , selectlags="BIC",  lags=max.lags ) |> summary()
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9374 -0.3151  0.0061  0.3098  3.8305
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0273361  0.0559092  -0.489    0.625
## z.lag.1     -0.7422821  0.0459154 -16.166   <2e-16 ***
## tt           0.0000465  0.0001359   0.342    0.732
## z.diff.lag   0.0359429  0.0383602   0.937    0.349
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7 on 679 degrees of freedom
## Multiple R-squared:  0.3591, Adjusted R-squared:  0.3563
## F-statistic: 126.8 on 3 and 679 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -16.1663 87.1165 130.6748
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
      # Philips-Perron(PP) tests on the chosen series:
      # PP:  Ho = series has a unit root = integrated I(1) process
      # PP test = DF test + robust to serial correlation by using the Newey-West HAC covariance matrix
```

```r
    ur.pp( diff(TCU), type="Z-tau"   ,  model="constant" , lags="long" ) |> summary()    # from package
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9855 -0.3068  0.0092  0.3092  4.1048
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01037    0.02633  -0.394    0.694
## y.l1         0.27690    0.03621   7.646 6.85e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6975 on 700 degrees of freedom
## Multiple R-squared:  0.07708,    Adjusted R-squared:  0.07576
## F-statistic: 58.46 on 1 and 700 DF,  p-value: 6.852e-14
##
##
## Value of test-statistic, type: Z-tau  is: -20.5802
##
##          aux. Z statistics
## Z-tau-mu           -0.4085
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.442105 -2.866017 -2.569158
```

```r
    ur.pp( diff(TCU), type="Z-tau"   ,  model="trend"    , lags="long" ) |> summary()
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept and trend
##
##
## Call:
## lm(formula = y ~ y.l1 + trend)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -8.9984 -0.3120  0.0076  0.3086  4.0929
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.039e-02  2.635e-02  -0.394    0.693
## y.l1         2.766e-01  3.625e-02   7.632 7.61e-14 ***
## trend        4.233e-05  1.300e-04   0.326    0.745
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.698 on 699 degrees of freedom
## Multiple R-squared:  0.07722,    Adjusted R-squared:  0.07458
## F-statistic: 29.25 on 2 and 699 DF,  p-value: 6.333e-13
##
##
## Value of test-statistic, type: Z-tau  is: -20.5654
##
##            aux. Z statistics
## Z-tau-mu             -0.3088
## Z-tau-beta            0.3398
##
## Critical values for Z statistics:
##                      1pct     5pct    10pct
## critical values -3.975795 -3.41839 -3.13136
```

```
    # KPSS tests on the chosen series:
    # KPSS:   Ho = series does not have a unit root [opposite of ADF and PP]
    #
    # "mu"  = constant
    # "tau" = constant + linear trend


  ur.kpss( diff(TCU), type="mu"   ,  lags="long" ) |> summary()     # from package "urca"
```

```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 19 lags.
##
## Value of test-statistic is: 0.0321
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
  ur.kpss( diff(TCU), type="tau" ,  lags="long" ) |> summary()
```

```
##
## ########################
## # KPSS Unit Root Test #
```

```
## ######################
##
## Test is of type: tau with 19 lags.
##
## Value of test-statistic is: 0.0188
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.119 0.146  0.176 0.216
```

The unit root tests show that we can reject the null hypothesis for the ADF and PP tests while we reject the null hypothesis for the KPSS, this means that there is a unit root in levels for the TCU series, while there is strong evidence that there is no unit root for the TCU time series in first differences.

4 Cointegration test for the dataset.

In this section we will check if the series are cointegrated. Cointegration means that the series could have a long run equilibrium.

```
  # Optimal Lag Selection
    # Check the optimal lag length according to the information criteria:
    # Using the "VARS" package


VARselect(data.set, lag.max=5, type="none",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=5, type="const", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=5, type="trend", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=5, type="both",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="none",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="const", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="trend", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="both",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=20, type="none",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
VARselect(data.set, lag.max=20, type="const", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
VARselect(data.set, lag.max=20, type="trend", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
VARselect(data.set, lag.max=20, type="both",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
        # Now set the number of lags that will be used in our estimations:

optimal.lags = 2
```

4.1 Cointegration Analysis: Engle-Granger Test

```
EGCM = egcm( TCU, Inflation, i1test="pp", urtest="pp", p.value=0.10)                # in levels
```

```
## Warning in pp.test(X): p-value smaller than printed p-value
```

```
EGCM
```

```
## Y[i] =   0.2386 X[i] -  15.1939 + R[i], R[i] =   0.9963 R[i-1] + eps[i], eps ~ N(0,  0.3844^2)
##         (0.0229)        (1.8447)                (0.0058)
##
## R[704] = -0.3680 (t = -0.148)
##
## WARNING: X does not seem to be integrated. Y does not seem to be integrated. X and Y do not appear t
```

```
EGCM = egcm( Y= TCU, X= Inflation, include.const   = TRUE)
```

```
## Warning in pp.test(X): p-value smaller than printed p-value
```

```
EGCM
```

```
## Y[i] =   0.5614 X[i] +  77.8341 + R[i], R[i] =   0.9872 R[i-1] + eps[i], eps ~ N(0,  0.7112^2)
##         (0.0539)        (0.4374)                (0.0070)
##
## R[704] = -2.0843 (t = -0.545)
##
## WARNING: Y does not seem to be integrated. X and Y do not appear to be cointegrated.
```

```
       # ------ Engle-Granger Methodology (using package "aTSA") --------------------

coint.test(Inflation, TCU,           d = 0, nlag = NULL, output = TRUE)                  # i
```

```
## Response: Inflation
## Input: TCU
## Number of inputs: 1
## Model: y ~ X + 1
## ------------------------------
## Engle-Granger Cointegration Test
## alternative: cointegrated
##
## Type 1: no trend
##     lag      EG p.value
##  6.0000 -3.1107  0.0329
## -----
##  Type 2: linear trend
##     lag      EG p.value
##   6.000  -0.571   0.100
## -----
##  Type 3: quadratic trend
##     lag      EG p.value
##  6.0000  0.0142  0.1000
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

```r
coint.test(Inflation,log(TCU),  d = 0, nlag = NULL, output = TRUE)          # in log
```

```
## Response: Inflation
## Input: log(TCU)
## Number of inputs: 1
## Model: y ~ X + 1
## ------------------------------
## Engle-Granger Cointegration Test
## alternative: cointegrated
##
## Type 1: no trend
##     lag      EG p.value
##  6.0000 -3.0904  0.0343
## -----
##  Type 2: linear trend
##     lag      EG p.value
##   6.000  -0.576   0.100
## -----
##  Type 3: quadratic trend
##     lag      EG p.value
##  6.0000  0.0109  0.1000
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

```r
coint.test(Inflation, TCU,                d = 0, nlag = NULL, output = TRUE)
```

```
## Response: Inflation
## Input: TCU
## Number of inputs: 1
## Model: y ~ X + 1
## ------------------------------
## Engle-Granger Cointegration Test
## alternative: cointegrated
##
## Type 1: no trend
##     lag      EG p.value
##  6.0000 -3.1107  0.0329
## -----
##  Type 2: linear trend
##     lag      EG p.value
##   6.000  -0.571   0.100
## -----
##  Type 3: quadratic trend
##     lag      EG p.value
##  6.0000  0.0142  0.1000
## -----------
## Note: p.value = 0.01 means p.value <= 0.01
##     : p.value = 0.10 means p.value >= 0.10
```

4.2 Cointegration Analysis: Johansen Test

```
        # Null hypothesis: Ho = no cointegration (r=0)
        # You need to reject the null to get cointegration (r=1).
        # r = number of cointegration relations


optimal.lags = 2



    # Linear trend outside of the cointegrating vector;
    # No constant and no trend in the cointegrating vector


johansen.none = ca.jo(data.set, type="eigen", ecdet="none",  K = optimal.lags, spec="longrun")
    summary(johansen.none)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.03393864 0.02325118
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.52  6.50  8.18 11.65
## r = 0  | 24.24 12.91 14.90 19.19
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.12      TCU.12
## Inflation.12     1.000000  1.00000000
## TCU.12          -2.171465 -0.04913232
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.12      TCU.12
## Inflation.d -0.005399723 -0.01494343
## TCU.d        0.009735001 -0.03219249
```

```
johansen.none = ca.jo(data.set, type="trace", ecdet="none",  K = optimal.lags, spec="longrun")
    summary(johansen.none)
```

```
##
```

```
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.03393864 0.02325118
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.52  6.50  8.18 11.65
## r = 0  | 40.75 15.66 17.95 23.52
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2
## Inflation.l2     1.000000  1.00000000
## TCU.l2          -2.171465 -0.04913232
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2
## Inflation.d -0.005399723 -0.01494343
## TCU.d        0.009735001 -0.03219249
```

The Johances test shows that the two series are cointegrated

```
    # With constant in the cointegrating vector:


johansen.const = ca.jo(data.set, type="eigen", ecdet="const",  K = optimal.lags, spec="longrun")
    summary(johansen.const)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegra
##
## Eigenvalues (lambda):
## [1] 3.410882e-02 2.332786e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.57  7.52  9.24 12.97
## r = 0  | 24.36 13.75 15.67 20.20
##
```

```
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2    constant
## Inflation.l2     1.000000  1.00000000    1.000000
## TCU.l2          -2.231647 -0.05781484    1.304402
## constant       174.072611  0.88402555 -180.476619
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      constant
## Inflation.d -0.005190549 -0.01515260  6.327124e-18
## TCU.d        0.009594073 -0.03205156 -1.097179e-17
```

```
johansen.const = ca.jo(data.set, type="trace", ecdet="const",  K = optimal.lags, spec="longrun")
    summary(johansen.const)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in cointegration
##
## Eigenvalues (lambda):
## [1] 3.410882e-02 2.332786e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.57  7.52  9.24 12.97
## r = 0  | 40.93 17.85 19.96 24.60
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2    constant
## Inflation.l2     1.000000  1.00000000    1.000000
## TCU.l2          -2.231647 -0.05781484    1.304402
## constant       174.072611  0.88402555 -180.476619
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      constant
## Inflation.d -0.005190549 -0.01515260  6.327124e-18
## TCU.d        0.009594073 -0.03205156 -1.097179e-17
```

The johansen test with a constant shows that they are cointegrated.

4.3 With trend in the cointegrating vector:

```
johansen.trend = ca.jo(data.set, type="eigen", ecdet="trend",  K = optimal.lags, spec="longrun")
    summary(johansen.trend)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1] 4.409662e-02 2.774420e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 19.75 10.49 12.25 16.26
## r = 0  | 31.66 16.85 18.96 23.65
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2       TCU.l2   trend.l2
## Inflation.l2   1.00000000  1.000000000  1.0000000
## TCU.l2         1.80514591 -0.316634329 -5.0156612
## trend.l2       0.02360683  0.003818288 -0.9186984
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      trend.l2
## Inflation.d  0.002347583 -0.02366188  6.086948e-21
## TCU.d       -0.020155929 -0.01737558 -6.409612e-20
```

```
johansen.trend = ca.jo(data.set, type="trace", ecdet="trend",  K = optimal.lags, spec="longrun")
    summary(johansen.trend)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1] 4.409662e-02 2.774420e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
```

```
## r <= 1 | 19.75 10.49 12.25 16.26
## r = 0  | 51.41 22.76 25.32 30.45
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2       TCU.l2   trend.l2
## Inflation.l2   1.00000000  1.000000000  1.0000000
## TCU.l2         1.80514591 -0.316634329 -5.0156612
## trend.l2       0.02360683  0.003818288 -0.9186984
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      trend.l2
## Inflation.d  0.002347583 -0.02366188  6.086948e-21
## TCU.d       -0.020155929 -0.01737558 -6.409612e-20
```

From these tow tests we can see that the series are not cointegrated. This means that the short term equilibrium of the series will tend to get back to its long term trend. Knowing if the series are correlated is important to understand which model to use to do forecasting or causal inference.

5 Auto-Regressive Distributed Lag (ARDL) Models

```
lags = 2

ARDL = dynardl(
                      TCU ~ Inflation,                                                    # Inflation is th
                                                                                               # TCU is th
               lags = list("Inflation" = 1,          "TCU" = 1             ),
              diffs =    c("Inflation"                                          ),
          lagdiffs = list("Inflation" = c(1:lags),  "TCU" = c(1:lags)     ),

                ec = TRUE,
          constant = TRUE,
             trend = FALSE,

          simulate = TRUE,              # Set this to "simulate = TRUE" when you want to plot the ARD
          shockvar = "Inflation",       # Select which variable will receive the exogenous shock in t
             range = 50,
              sims = 1000,
          fullsims = TRUE,

              data = data.set)
```

```
## [1] "Error correction (EC) specified; dependent variable to be run in differences."
## [1] "Inflation shocked by one standard deviation of Inflation by default."
## [1] "dynardl estimating ..."
##   |                                                                              |
```

```r
summary(ARDL)
```

```
##
## Call:
## lm(formula = as.formula(paste(paste(dvnamelist), "~", paste(colnames(IVs),
##     collapse = "+"), collapse = " ")))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7499 -0.3303  0.0249  0.3215  3.2137
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.778455   0.550810   3.229  0.00130 **
## l.1.TCU        -0.021531   0.007058  -3.051  0.00237 **
## ld.1.TCU        0.256975   0.037699   6.816 2.03e-11 ***
## ld.2.TCU       -0.038411   0.037737  -1.018  0.30910
## d.1.Inflation   0.304882   0.075091   4.060 5.46e-05 ***
## l.1.Inflation  -0.017110   0.010652  -1.606  0.10867
## ld.1.Inflation  0.127122   0.080759   1.574  0.11592
## ld.2.Inflation -0.064617   0.075757  -0.853  0.39398
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6796 on 693 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.1308, Adjusted R-squared:  0.122
## F-statistic: 14.89 on 7 and 693 DF,  p-value: < 2.2e-16
```

```r
    # Long-run relatiobship testing using the ARDL-bounds procedure and "pssbounds":


pssbounds(ARDL)
```

```
##
##  PESARAN, SHIN AND SMITH (2001) COINTEGRATION TEST
##
##  Observations: 701
##  Number of Lagged Regressors (not including LDV) (k): 1
##  Case: 3 (Unrestricted intercept; no trend)
##
##  ------------------------------------------------------
##  -                       F-test                       -
##  ------------------------------------------------------
##                  <------- I(0) ------------ I(1) ----->
##  10% critical value       4.04            4.78
##  5% critical value        4.94            5.73
##  1% critical value        6.84            7.84
##
##
##  F-statistic = 9.153
##  ------------------------------------------------------
##  -                       t-test                       -
```

```
##  ------------------------------------------------------
##                  <------- I(0) ------------ I(1) ----->
##  10% critical value       -2.57            -2.91
##  5% critical value        -2.86            -3.22
##  1% critical value        -3.43            -3.82
##
##
##  t statistic = -3.051
##  ------------------------------------------------------
##  F-statistic note: Asymptotic critical values used.
##  t-statistic note: Asymptotic critical values used.
```

```
pssbounds(ARDL, restriction=TRUE)
```

```
##
##  PESARAN, SHIN AND SMITH (2001) COINTEGRATION TEST
##
##  Observations: 701
##  Number of Lagged Regressors (not including LDV) (k): 1
##  Case: 2 (Intercept included in F-stat restriction; no trend)
##
##  ------------------------------------------------------
##  -                       F-test                       -
##  ------------------------------------------------------
##                  <------- I(0) ------------ I(1) ----->
##  10% critical value       3.02             3.51
##  5% critical value        3.62             4.16
##  1% critical value        4.94             5.58
##
##
##  F-statistic = 6.161
##
##  ------------------------------------------------------
##  F-statistic note: Asymptotic critical values used.
##  t-statistic note: Critical values do not currently exist for Case II.
```

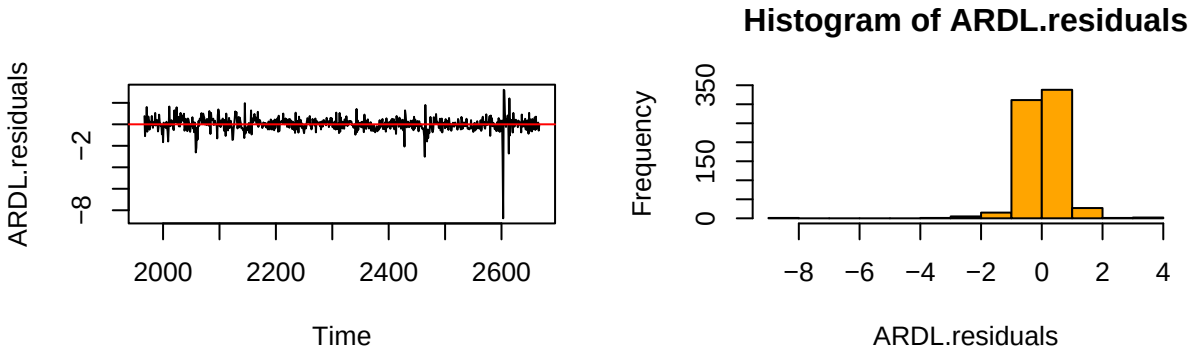
```
    # Philips (2018): if the BG test indicates the presence of residual autocorrelation,
    # add lagged first-differences of the endogenous variable on the right hand side of the ARDL.
    # The inclusion of lagged first-differences of the endogenous variable on the right hand side
    # of the ARDL should remove the autocorrelation from the ARDL residuals

    # ARDL-bounds test procedure requires the residuals to be white noise

dynardl.auto.correlated(ARDL)
```

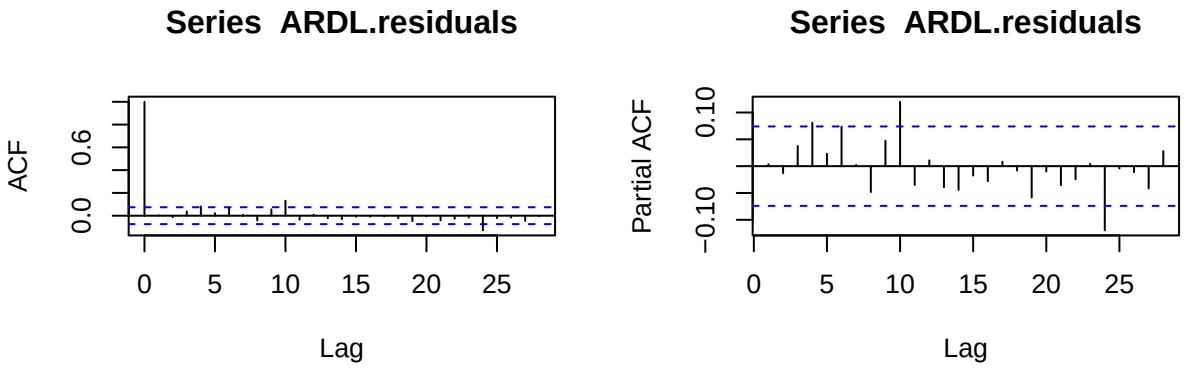
```
##
##  ------------------------------------------------------
##  Breusch-Godfrey LM Test
##  Test statistic: 2.838
##  p-value: 0.092
##  H_0: no autocorrelation up to AR 1
##
##  ------------------------------------------------------
```

```
##  Shapiro-Wilk Test for Normality
##  Test statistic: 0.8
##  p-value: 0
##  H_0: residuals are distributed normal
##
##  ------------------------------------------------------
##  Log-likelihood: -719.942
##  AIC: 1457.883
##  BIC: 1498.856
##  Note: AIC and BIC calculated with k = 8 on T = 701 observations.
##
##  ------------------------------------------------------
## Breusch-Godfrey test indicates we reject the null hypothesis of no autocorrelation at p < 0.10.
##  Add lags to remove autocorrelation before running dynardl simulations.
## Shapiro-Wilk test indicates we reject the null hypothesis of normality at p < 0.01.
```

5.1 ARDL Model Residuals and Additional Tests:

```
ARDL.residuals =    ARDL$model$residuals
ARDL.residuals = ts(ARDL$model$residuals,  start=c(1967, 1) ,  frequency=1)



  par(mfrow = c(2, 2))
      plot( ARDL.residuals )                # Plot the residuals over time
    abline( h=0, col="red" )
      hist( ARDL.residuals, col="orange")   # Plot histogram of the residuals
       acf( ARDL.residuals )                # Plot ACF
      pacf( ARDL.residuals )                # Plot PACF
```






```
jarque.bera.test(ARDL.residuals)      # Jarque-Bera test for normal residuals
```

```
##
##  Jarque Bera Test
##
## data:  ARDL.residuals
## X-squared = 52780, df = 2, p-value < 2.2e-16
```

```
bds.test(ARDL.residuals)              # BDS test Ho: series of i.i.d. random variable
```

```
##
##   BDS Test
##
## data:  ARDL.residuals
##
## Embedding dimension =  2 3
##
## Epsilon for close points =  0.3381 0.6762 1.0144 1.3525
##
## Standard Normal =
##       [ 0.3381 ] [ 0.6762 ] [ 1.0144 ] [ 1.3525 ]
## [ 2 ]     7.3318     8.2695     9.8458    11.4308
## [ 3 ]     7.9038     9.0543    10.8707    11.9591
##
```

```
## p-value =
##       [ 0.3381 ] [ 0.6762 ] [ 1.0144 ] [ 1.3525 ]
## [ 2 ]          0          0          0          0
## [ 3 ]          0          0          0          0
```

```
bptest(ARDL$model)                    # Breusch-Pagan test against heteroskedasticity, Ho: no heteroske
```

```
##
##  studentized Breusch-Pagan test
##
## data:  ARDL$model
## BP = 55.796, df = 7, p-value = 1.037e-09
```

Here the residuals are not serially correlated but they are not normally distributed as well , as we can see there is a long tail to the left, so the distribution is left skewed, since there are some extreme values. ## ARDL Model with robust standard errors:

```
      # Robust = corrected for heteroskedasticity and auto-correlation


coeftest(ARDL$model, vcov.=NULL   )   # Estimated coefficients with standard errors
```

```
##
## t test of coefficients:
##
##                   Estimate Std. Error t value  Pr(>|t|)
## (Intercept)     1.7784553  0.5508101  3.2288  0.001302 **
## l.1.TCU        -0.0215315  0.0070582 -3.0505  0.002371 **
## ld.1.TCU        0.2569745  0.0376993  6.8164 2.032e-11 ***
## ld.2.TCU       -0.0384108  0.0377367 -1.0179  0.309098
## d.1.Inflation   0.3048816  0.0750915  4.0601 5.464e-05 ***
## l.1.Inflation  -0.0171098  0.0106518 -1.6063  0.108667
## ld.1.Inflation  0.1271224  0.0807590  1.5741  0.115922
## ld.2.Inflation -0.0646169  0.0757567 -0.8530  0.393980
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(ARDL$model, vcov.=vcovHC )   # Estimated coefficients with HC standard errors
```

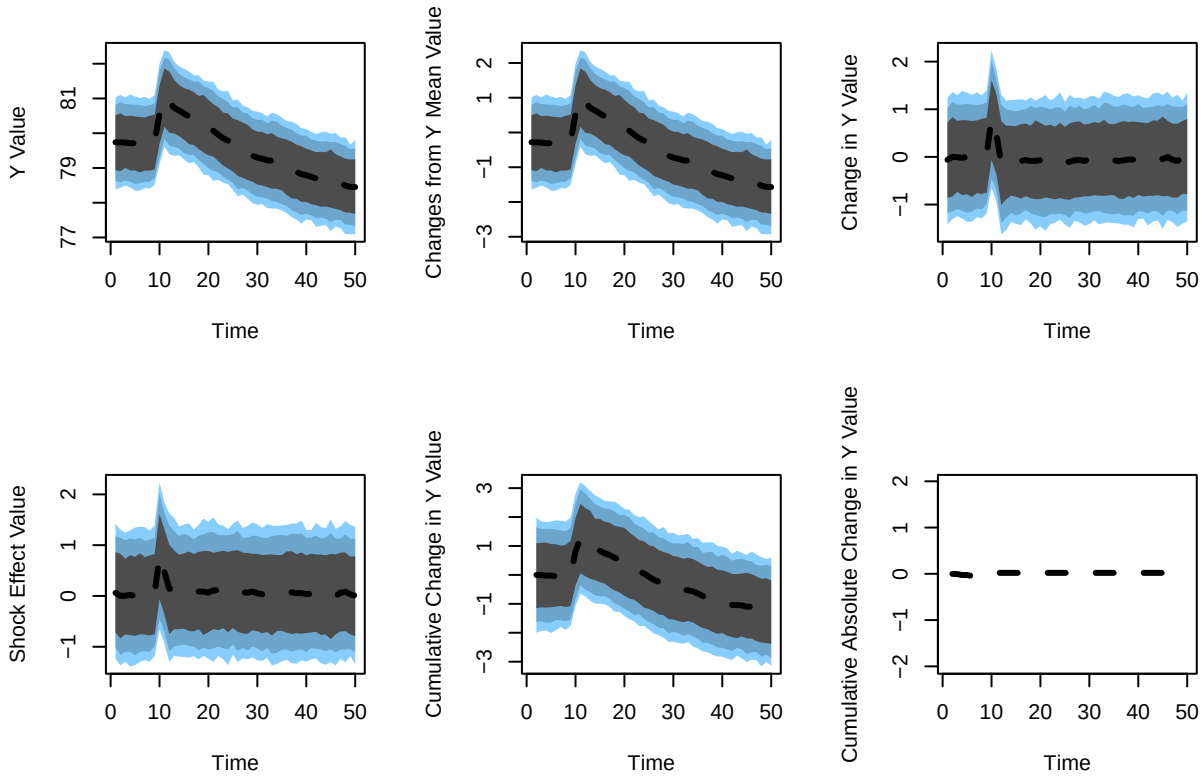
```
##
## t test of coefficients:
##
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.7784553  0.6460527  2.7528 0.006063 **
## l.1.TCU        -0.0215315  0.0078887 -2.7294 0.006506 **
## ld.1.TCU        0.2569745  0.1605751  1.6003 0.109979
## ld.2.TCU       -0.0384108  0.1468099 -0.2616 0.793680
## d.1.Inflation   0.3048816  0.1315991  2.3167 0.020808 *
## l.1.Inflation  -0.0171098  0.0139817 -1.2237 0.221471
## ld.1.Inflation  0.1271224  0.0795590  1.5978 0.110535
## ld.2.Inflation -0.0646169  0.0837414 -0.7716 0.440600
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
coeftest(ARDL$model, vcov.=vcovHAC)   # Estimated coeffficients with HAC standard errors
```

```
##
## t test of coefficients:
##
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.7784553  0.5710489  3.1144 0.001919 **
## l.1.TCU        -0.0215315  0.0070674 -3.0466 0.002402 **
## ld.1.TCU        0.2569745  0.1283899  2.0015 0.045726 *
## ld.2.TCU       -0.0384108  0.1058409 -0.3629 0.716782
## d.1.Inflation   0.3048816  0.1225919  2.4870 0.013118 *
## l.1.Inflation  -0.0171098  0.0131325 -1.3029 0.193055
## ld.1.Inflation  0.1271224  0.0762642  1.6669 0.095992 .
## ld.2.Inflation -0.0646169  0.0803823 -0.8039 0.421748
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5.2 Impulse-Response Functions for the ARDL model

```r
    # Counterfactual simulation of a response to a shock in some exogenous variable

    # simulate = TRUE
    # shockvar = exogenous variable that will receive the impulse
    # shockval = 1 standard deviation
    # time     = when the shock happens
    # sims     = 1,000 simulations to estimate the average response
    # forceset = set the values of any exogenous variables used = it hold an exogenous variable at a ce

    # If the exogenous variable is in levels or lagged levels: shock will last for many periods
    # If the exogenous variable is in differences or lagged differences: shock lasts for 1 period only
    # Other variables are kept at their averages, and differences are set to zero



    # Impulse-Response plots:



dynardl.all.plots(ARDL)  # all plots together
```




```
## Warning in dynardl.simulation.plot(x, response = "cumulative.abs.diffs", :
## Cumulative absolute effects assumed to be noise (by tolerance) at t = 1.
```

```
## Warning in dynardl.simulation.plot(x, response = "cumulative.abs.diffs", : Y
## does not move beyond the tolerance in the simulation. Reconsider the tolerance,
## or investigate if Y responds to the shockvar in the dynardl model.
```

According to this ARDL model there is a statistical significance that the there is a relationship between these two series in the long run. From the model we can understand that previous instances of the variables affect the current value of it. We used the error correction form in order check whether the the variables go back to its long run equilibrium, and we used it also because the two variables are cointegrated. We saw that the residuals of the model are not normally distributed, this due to the TCU dataset.

Breusch-Godfrey LM Test for autocorrelation and the Jarque Bera Test for normality of residuals indicate potential issues with the residuals,The Shapiro-Wilk Test similarly indicates that residuals do not follow a normal distribution. We have seen that this could have been caused by the TCU data, which had higher volatility in 2020.

From the plots we can see that Cumulative impact is positive, it could mean that the dependent variable TCU is affected when inflation increases, there will be an increase in output capacity, which will return to normality as the economy will adjust the new equilibrium.

5.3 Bivariate VAR(p) Model

```
VARselect(data.set, lag.max=5, type="none",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=5, type="const", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=5, type="trend", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=5, type="both",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="none",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="const", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="trend", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=10, type="both",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
```

```
VARselect(data.set, lag.max=20, type="none",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
VARselect(data.set, lag.max=20, type="const", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
VARselect(data.set, lag.max=20, type="trend", season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
VARselect(data.set, lag.max=20, type="both",  season = NULL, exogen = NULL)$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     16     14      2     16
```

```
    # Set optimal lag length using the code from the previous sections:


optimal.lags = 2
```

5.4 Reduced-form VAR(p) with p = optimal.lags:

```
    # Reduced-form VAR(p) with p = optimal.lags:


var.model.none  <- VAR(data.set,  p=optimal.lags, type="none",  exogen=NULL)
summary(var.model.none)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: Inflation, TCU
## Deterministic variables: none
## Sample size: 702
## Log Likelihood: -977.659
## Roots of the characteristic polynomial:
## 0.9996 0.9752 0.4345 0.2187
## Call:
## VAR(y = data.set, p = optimal.lags, type = "none", exogen = NULL)
##
##
## Estimation results for equation Inflation:
## ==========================================
## Inflation = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2
##
##              Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  1.38293    0.03515  39.345   <2e-16 ***
## TCU.l1        0.02488    0.01850   1.345    0.179
```

```
## Inflation.l2 -0.39788    0.03522 -11.297   <2e-16 ***
## TCU.l2       -0.02412    0.01853  -1.301    0.194
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.3464 on 698 degrees of freedom
## Multiple R-Squared: 0.9947,  Adjusted R-squared: 0.9946
## F-statistic: 3.256e+04 on 4 and 698 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation TCU:
## ====================================
## TCU = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2
##
##              Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  0.14798    0.07011   2.111   0.0352 *
## TCU.l1        1.24507    0.03691  33.732  < 2e-16 ***
## Inflation.l2 -0.18011    0.07025  -2.564   0.0106 *
## TCU.l2       -0.24369    0.03697  -6.592 8.55e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.6909 on 698 degrees of freedom
## Multiple R-Squared: 0.9999,  Adjusted R-squared: 0.9999
## F-statistic: 2.359e+06 on 4 and 698 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##           Inflation     TCU
## Inflation   0.11996 0.03266
## TCU         0.03266 0.47729
##
## Correlation matrix of residuals:
##           Inflation    TCU
## Inflation    1.0000 0.1365
## TCU          0.1365 1.0000
```

```r
var.model.const  <- VAR(data.set, p=optimal.lags, type="const", exogen=NULL)
summary(var.model.const)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: Inflation, TCU
## Deterministic variables: const
## Sample size: 702
## Log Likelihood: -965.649
## Roots of the characteristic polynomial:
## 0.9733 0.9733 0.4097 0.2357
## Call:
## VAR(y = data.set, p = optimal.lags, type = "const", exogen = NULL)
```

```
##
##
## Estimation results for equation Inflation:
## ==========================================
## Inflation = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2 + const
##
##              Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  1.35836    0.03564  38.113  < 2e-16 ***
## TCU.l1        0.03167    0.01848   1.714 0.087020 .
## Inflation.l2 -0.37870    0.03542 -10.692  < 2e-16 ***
## TCU.l2       -0.01921    0.01845  -1.041 0.298334
## const        -0.91693    0.27152  -3.377 0.000774 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.3438 on 697 degrees of freedom
## Multiple R-Squared: 0.9836,  Adjusted R-squared: 0.9836
## F-statistic: 1.048e+04 on 4 and 697 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation TCU:
## ====================================
## TCU = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2 + const
##
##              Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  0.19198    0.07120   2.696  0.00718 **
## TCU.l1        1.23293    0.03692  33.399  < 2e-16 ***
## Inflation.l2 -0.21444    0.07076  -3.030  0.00253 **
## TCU.l2       -0.25249    0.03687  -6.848 1.64e-11 ***
## const         1.64173    0.54246   3.026  0.00257 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.6869 on 697 degrees of freedom
## Multiple R-Squared: 0.9719,  Adjusted R-squared: 0.9718
## F-statistic:  6030 on 4 and 697 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##           Inflation     TCU
## Inflation   0.11820 0.03617
## TCU         0.03617 0.47179
##
## Correlation matrix of residuals:
##           Inflation    TCU
## Inflation    1.0000 0.1531
## TCU          0.1531 1.0000
```

```
var.model.trend  <- VAR(data.set, p=optimal.lags, type="trend", exogen=NULL)
summary(var.model.trend)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: Inflation, TCU
## Deterministic variables: trend
## Sample size: 702
## Log Likelihood: -974.962
## Roots of the characteristic polynomial:
## 0.9988 0.9667 0.4373 0.2189
## Call:
## VAR(y = data.set, p = optimal.lags, type = "trend", exogen = NULL)
##
##
## Estimation results for equation Inflation:
## ==========================================
## Inflation = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2 + trend
##
##                Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  1.3776498  0.0351356  39.209   <2e-16 ***
## TCU.l1        0.0236986  0.0184619   1.284   0.1997
## Inflation.l2 -0.3988216  0.0351273 -11.354   <2e-16 ***
## TCU.l2       -0.0219595  0.0185085  -1.186   0.2358
## trend        -0.0001543  0.0000703  -2.195   0.0285 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.3454 on 697 degrees of freedom
## Multiple R-Squared: 0.9947,  Adjusted R-squared: 0.9947
## F-statistic: 2.619e+04 on 5 and 697 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation TCU:
## ====================================
## TCU = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2 + trend
##
##                Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  0.1429936  0.0702723   2.035   0.0422 *
## TCU.l1        1.2439558  0.0369243  33.689  < 2e-16 ***
## Inflation.l2 -0.1809987  0.0702555  -2.576   0.0102 *
## TCU.l2       -0.2416517  0.0370174  -6.528 1.28e-10 ***
## trend        -0.0001457  0.0001406  -1.036   0.3004
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.6908 on 697 degrees of freedom
## Multiple R-Squared: 0.9999,  Adjusted R-squared: 0.9999
## F-statistic: 1.887e+06 on 5 and 697 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##           Inflation     TCU
```

```
## Inflation   0.11931 0.03193
## TCU         0.03193 0.47724
##
## Correlation matrix of residuals:
##           Inflation    TCU
## Inflation    1.0000 0.1338
## TCU          0.1338 1.0000
```

```
var.model.both   <- VAR(data.set, p=optimal.lags, type="both",  exogen=NULL)
summary(var.model.both)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: Inflation, TCU
## Deterministic variables: both
## Sample size: 702
## Log Likelihood: -960.32
## Roots of the characteristic polynomial:
## 0.965 0.965 0.4159 0.2347
## Call:
## VAR(y = data.set, p = optimal.lags, type = "both", exogen = NULL)
##
##
## Estimation results for equation Inflation:
## ==========================================
## Inflation = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2 + const + trend
##
##                Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  1.359e+00  3.571e-02  38.058  < 2e-16 ***
## TCU.l1        3.085e-02  1.859e-02   1.659  0.09755 .
## Inflation.l2 -3.805e-01  3.569e-02 -10.659  < 2e-16 ***
## TCU.l2       -1.912e-02  1.847e-02  -1.035  0.30091
## const        -8.424e-01  3.252e-01  -2.590  0.00979 **
## trend        -3.493e-05  8.382e-05  -0.417  0.67701
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.344 on 696 degrees of freedom
## Multiple R-Squared: 0.9837,  Adjusted R-squared: 0.9835
## F-statistic:  8375 on 5 and 696 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation TCU:
## ====================================
## TCU = Inflation.l1 + TCU.l1 + Inflation.l2 + TCU.l2 + const + trend
##
##                Estimate Std. Error t value Pr(>|t|)
## Inflation.l1  0.2044059  0.0708195   2.886 0.004019 **
## TCU.l1        1.2202120  0.0368694  33.095  < 2e-16 ***
## Inflation.l2 -0.2419374  0.0707815  -3.418 0.000667 ***
## TCU.l2       -0.2510947  0.0366177  -6.857 1.55e-11 ***
## const         2.7982882  0.6449552   4.339 1.65e-05 ***
```

```
## trend        -0.0005422  0.0001662  -3.262 0.001161 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.6822 on 696 degrees of freedom
## Multiple R-Squared: 0.9723,  Adjusted R-squared: 0.9721
## F-statistic:  4893 on 5 and 696 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##           Inflation     TCU
## Inflation   0.11834 0.03576
## TCU         0.03576 0.46535
##
## Correlation matrix of residuals:
##           Inflation    TCU
## Inflation    1.0000 0.1524
## TCU          0.1524 1.0000
```

```
    # Recursive VAR: Order the variables
    # Cholesky decompositions for orthogonal errors:



    # Ordering: Inflation --> TCU

  ordered.data.set = data.set[, c("Inflation","TCU")]




    # Estimate the ordered VARs:

var.ordered.none   <- VAR(ordered.data.set,  p=optimal.lags,   type="none",    exogen=NULL)
var.ordered.const  <- VAR(ordered.data.set,  p=optimal.lags,   type="const",   exogen=NULL)
var.ordered.trend  <- VAR(ordered.data.set,  p=optimal.lags,   type="trend",   exogen=NULL)
var.ordered.both   <- VAR(ordered.data.set,  p=optimal.lags,   type="both",    exogen=NULL)
```

6 Bivariate VEC(p) Model - Version 1: using package "urca"

```
    # Using package "urca" and function 'ca.jo'



optimal.lags = 2

        # Null hypothesis: Ho = no cointegration (r=0)
        # You need to reject the null to get cointegration (r=1).
        # r = number of cointegration relations
```

```r
    # Linear trend outside of the cointegrating vector;
    # No constant and no trend in the cointegrating vector


johansen.none = ca.jo(data.set, type="eigen", ecdet="none",  K = optimal.lags, spec="longrun")
    summary(johansen.none)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.03393864 0.02325118
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.52  6.50  8.18 11.65
## r = 0  | 24.24 12.91 14.90 19.19
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2
## Inflation.l2     1.000000  1.00000000
## TCU.l2          -2.171465 -0.04913232
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2
## Inflation.d -0.005399723 -0.01494343
## TCU.d        0.009735001 -0.03219249
```

```r
johansen.none = ca.jo(data.set, type="trace", ecdet="none",  K = optimal.lags, spec="longrun")
    summary(johansen.none)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.03393864 0.02325118
##
```

```
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.52  6.50  8.18 11.65
## r = 0  | 40.75 15.66 17.95 23.52
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2
## Inflation.l2     1.000000  1.00000000
## TCU.l2          -2.171465 -0.04913232
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2
## Inflation.d -0.005399723 -0.01494343
## TCU.d        0.009735001 -0.03219249
```

```
VECM = cajorls(johansen.none, r = 1)
VECM
```

```
## $rlm
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Coefficients:
##                Inflation.d  TCU.d
## ect1           -0.005400     0.009735
## constant       -0.916315     1.643051
## Inflation.dl1   0.364356     0.204895
## TCU.dl1         0.038493     0.247635
##
##
## $beta
##                   ect1
## Inflation.l2  1.000000
## TCU.l2       -2.171465
```

6.1 With constant in the cointegrating vector:

```
    # With constant in the cointegrating vector:


johansen.const = ca.jo(data.set, type="eigen", ecdet="const",  K = optimal.lags, spec="longrun")
    summary(johansen.const)
```

```
##
## ######################
```

```
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegra
##
## Eigenvalues (lambda):
## [1] 3.410882e-02 2.332786e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.57  7.52  9.24 12.97
## r = 0  | 24.36 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2    constant
## Inflation.l2     1.000000  1.00000000    1.000000
## TCU.l2          -2.231647 -0.05781484    1.304402
## constant       174.072611  0.88402555 -180.476619
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      constant
## Inflation.d -0.005190549 -0.01515260  6.327124e-18
## TCU.d        0.009594073 -0.03205156 -1.097179e-17
```

```
johansen.const = ca.jo(data.set, type="trace", ecdet="const",  K = optimal.lags, spec="longrun")
    summary(johansen.const)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in cointegration
##
## Eigenvalues (lambda):
## [1] 3.410882e-02 2.332786e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.57  7.52  9.24 12.97
## r = 0  | 40.93 17.85 19.96 24.60
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2    constant
## Inflation.l2     1.000000  1.00000000    1.000000
## TCU.l2          -2.231647 -0.05781484    1.304402
```

```
## constant       174.072611  0.88402555 -180.476619
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      constant
## Inflation.d -0.005190549 -0.01515260  6.327124e-18
## TCU.d        0.009594073 -0.03205156 -1.097179e-17
```

```
VECM = cajorls(johansen.const, r = 1)
VECM
```

```
## $rlm
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Coefficients:
##                Inflation.d  TCU.d
## ect1           -0.005191     0.009594
## Inflation.dl1   0.364679     0.205346
## TCU.dl1         0.038555     0.247501
##
##
## $beta
##                    ect1
## Inflation.l2   1.000000
## TCU.l2        -2.231647
## constant     174.072611
```

```
    # With trend in the cointegrating vector:

johansen.trend = ca.jo(data.set, type="eigen", ecdet="trend",  K = optimal.lags, spec="longrun")
    summary(johansen.trend)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1] 4.409662e-02 2.774420e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 19.75 10.49 12.25 16.26
## r = 0  | 31.66 16.85 18.96 23.65
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
```

```
##
##              Inflation.l2       TCU.l2   trend.l2
## Inflation.l2   1.00000000  1.000000000  1.0000000
## TCU.l2         1.80514591 -0.316634329 -5.0156612
## trend.l2       0.02360683  0.003818288 -0.9186984
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      trend.l2
## Inflation.d  0.002347583 -0.02366188  6.086948e-21
## TCU.d       -0.020155929 -0.01737558 -6.409612e-20
```

```
johansen.trend = ca.jo(data.set, type="trace", ecdet="trend",  K = optimal.lags, spec="longrun")
    summary(johansen.trend)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1] 4.409662e-02 2.774420e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 19.75 10.49 12.25 16.26
## r = 0  | 51.41 22.76 25.32 30.45
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2       TCU.l2   trend.l2
## Inflation.l2   1.00000000  1.000000000  1.0000000
## TCU.l2         1.80514591 -0.316634329 -5.0156612
## trend.l2       0.02360683  0.003818288 -0.9186984
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l2      TCU.l2      trend.l2
## Inflation.d  0.002347583 -0.02366188  6.086948e-21
## TCU.d       -0.020155929 -0.01737558 -6.409612e-20
```

```
VECM = cajorls(johansen.trend, r = 1)
VECM
```

```
## $rlm
##
## Call:
```

```
## lm(formula = substitute(form1), data = data.mat)
##
## Coefficients:
##                Inflation.d  TCU.d
## ect1            0.002348   -0.020156
## constant       -0.366815    3.146503
## Inflation.dl1   0.381938    0.221131
## TCU.dl1         0.035828    0.223870
##
##
## $beta
##                     ect1
## Inflation.l2 1.00000000
## TCU.l2       1.80514591
## trend.l2     0.02360683
```

6.2 Likelihood ratio test for no linear trend

```
    # This corresponds to the inclusion of a constant in the error correction term
    # Function "lttest" in package "urca"
    # Ho = no linear trend


linear.trend.test = lttest(johansen.none,  r=1)
```

```
## LR-test for no linear trend
##
## H0: H*2(r<=1)
## H1: H2(r<=1)
##
## Test statistic is distributed as chi-square
## with 1 degress of freedom
##         test statistic p-value
## LR test           0.06    0.81
```

```
linear.trend.test = lttest(johansen.const, r=1)
```

```
## LR-test for no linear trend
##
## H0: H*2(r<=1)
## H1: H2(r<=1)
##
## Test statistic is distributed as chi-square
## with 1 degress of freedom
##         test statistic p-value
## LR test           0.06    0.81
```

```
linear.trend.test = lttest(johansen.trend, r=1)
```

```
## LR-test for no linear trend
```

```
##
## H0: H*2(r<=1)
## H1: H2(r<=1)
##
## Test statistic is distributed as chi-square
## with 1 degress of freedom
##         test statistic p-value
## LR test           0.06    0.81
```

6.3 Transform VEC into VAR in levels, using package "vars"

```
    # Transform VEC into VAR in levels, using package "vars"
    # r = number of cointegrating vectors
    # It is then possible to compute and plot IRFs, FEVD, and Diagnosis Tests


VECM = vec2var(johansen.none,  r = 1)

VECM = vec2var(johansen.const, r = 1)

VECM = vec2var(johansen.trend, r = 1)
```

6.4 Diagnostic Checks:

```
    # Serial correlation test
    # Ho = residuals do not have serial correlation


serialtest <- serial.test(VECM, type = "PT.asymptotic")
serialtest
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object VECM
## Chi-squared = 275.46, df = 58, p-value < 2.2e-16
```

```
serialtest <- serial.test(VECM, type = "PT.adjusted")
serialtest
```

```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object VECM
## Chi-squared = 279.83, df = 58, p-value < 2.2e-16
```

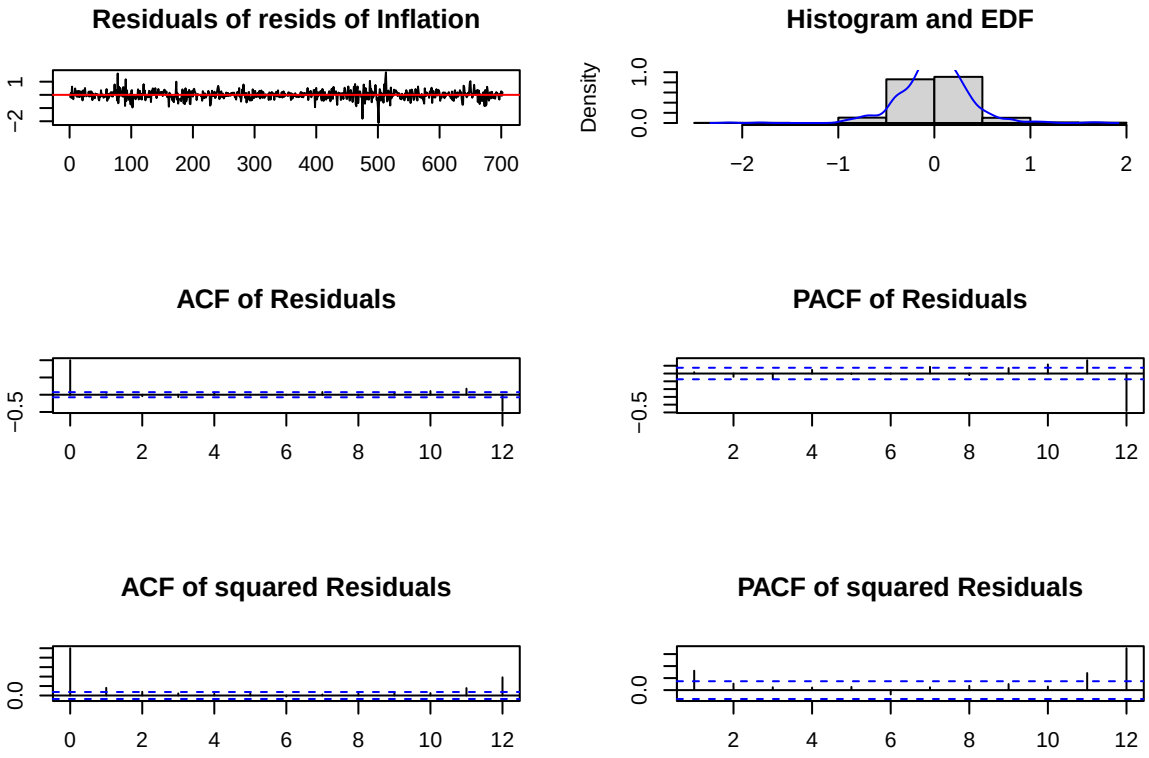
```
serialtest <- serial.test(VECM, type = "BG")
serialtest
```

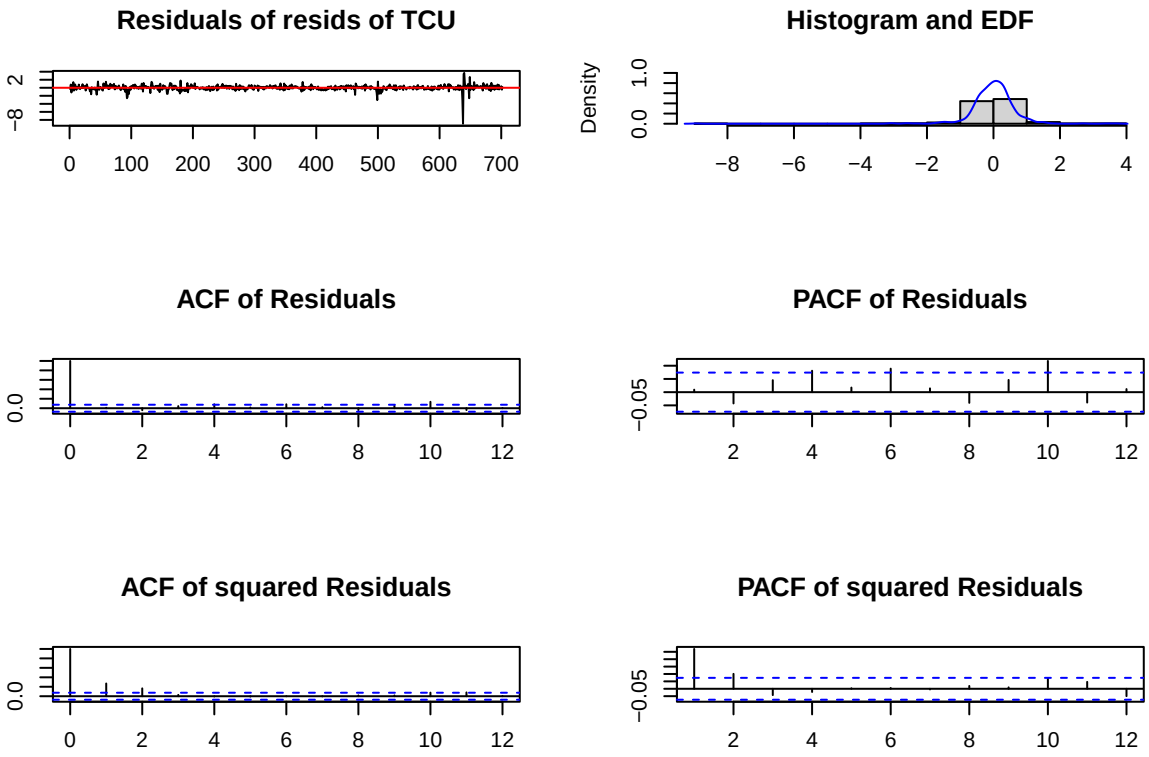
```
##
##  Breusch-Godfrey LM test
##
## data:  Residuals of VAR object VECM
## Chi-squared = 23.015, df = 20, p-value = 0.288
```

```
serialtest <- serial.test(VECM, type = "ES")
serialtest
```

```
##
##  Edgerton-Shukur F test
##
## data:  Residuals of VAR object VECM
## F statistic = 1.1422, df1 = 20, df2 = 1370, p-value = 0.2985
```

```
plot(serialtest)
```





6.5 Normality test

```
    # Jarque-Bera normality test of jointly normal residuals
    # Non-normal residuals distort estimates and confidence intervals
    # Test whether the multivariate skewness and kurtosis match a normal distribution
    # Joint Ho = skewness is zero, and excess kurtosis is zero
    # Non-normal residuals are a problem in small samples but less so in large samples (due to asymptot



normalitytest <- normality.test(VECM)
normalitytest
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object VECM
## Chi-squared = 54914, df = 4, p-value < 2.2e-16
##
##
## $Skewness
##
##  Skewness only (multivariate)
```

```
##
## data:  Residuals of VAR object VECM
## Chi-squared = 1146.2, df = 2, p-value < 2.2e-16
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object VECM
## Chi-squared = 53767, df = 2, p-value < 2.2e-16
```

6.6 Structural VAR: Order the variables

6.7 Cholesky decompositions (for orthogonal errors):

```
    # Ordering: Inflation --> TCU

ordered.data.set = data.set[, c("Inflation","TCU")]

    # Structural VECM using the Cholesky decomposition via the ordered data set:

johansen.const = ca.jo(ordered.data.set, type="eigen", ecdet="const",  K = optimal.lags, spec="longrun"
    summary(johansen.const)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegra
##
## Eigenvalues (lambda):
## [1] 3.410882e-02 2.332786e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.57  7.52  9.24 12.97
## r = 0  | 24.36 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l2      TCU.l2    constant
## Inflation.l2     1.000000  1.00000000    1.000000
## TCU.l2          -2.231647 -0.05781484    1.304402
## constant       174.072611  0.88402555 -180.476619
##
## Weights W:
## (This is the loading matrix)
##
```

```
##             Inflation.l2      TCU.l2      constant
## Inflation.d -0.005190549 -0.01515260  6.327124e-18
## TCU.d        0.009594073 -0.03205156 -1.097179e-17
```

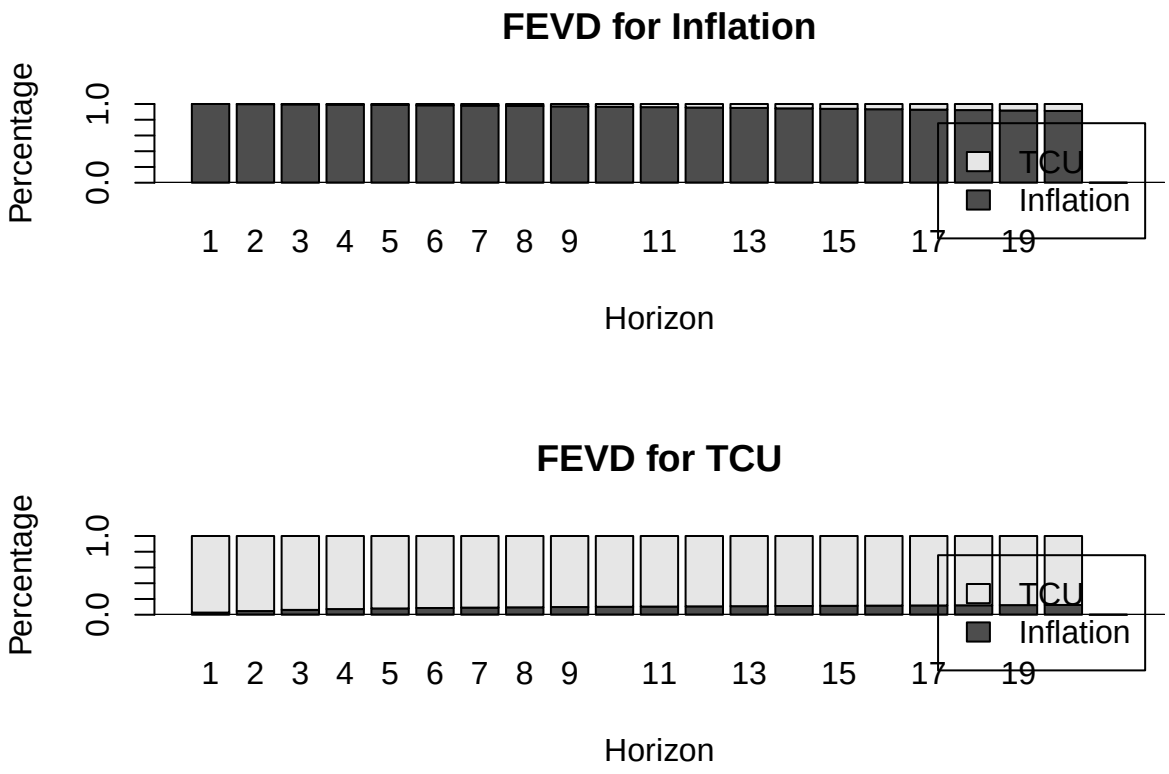
```
johansen.const = ca.jo(ordered.data.set, type="eigen", ecdet="const",  K = optimal.lags, spec="transito
    summary(johansen.const)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegra
##
## Eigenvalues (lambda):
## [1] 3.410882e-02 2.332786e-02 6.938894e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 16.57  7.52  9.24 12.97
## r = 0  | 24.36 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              Inflation.l1      TCU.l1    constant
## Inflation.l1     1.000000  1.00000000    1.000000
## TCU.l1          -2.231647 -0.05781484    1.304402
## constant       174.072611  0.88402555 -180.476619
##
## Weights W:
## (This is the loading matrix)
##
##             Inflation.l1      TCU.l1      constant
## Inflation.d -0.005190549 -0.01515260 -2.214712e-18
## TCU.d        0.009594073 -0.03205156  3.424577e-18
```

```
VECM = vec2var(johansen.const, r = 1)
```

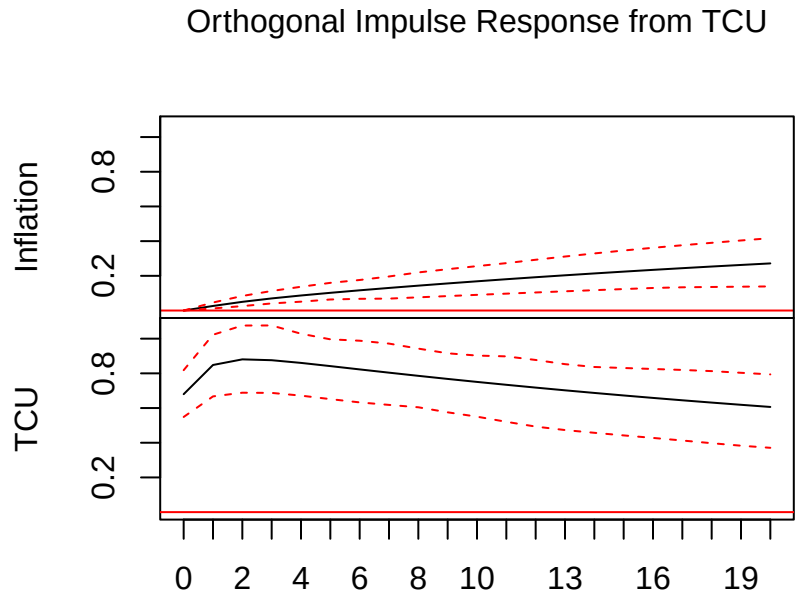
6.8 FEVD: Forecast error variance decomposition:

```
plot(fevd(VECM,  n.ahead=20))
```

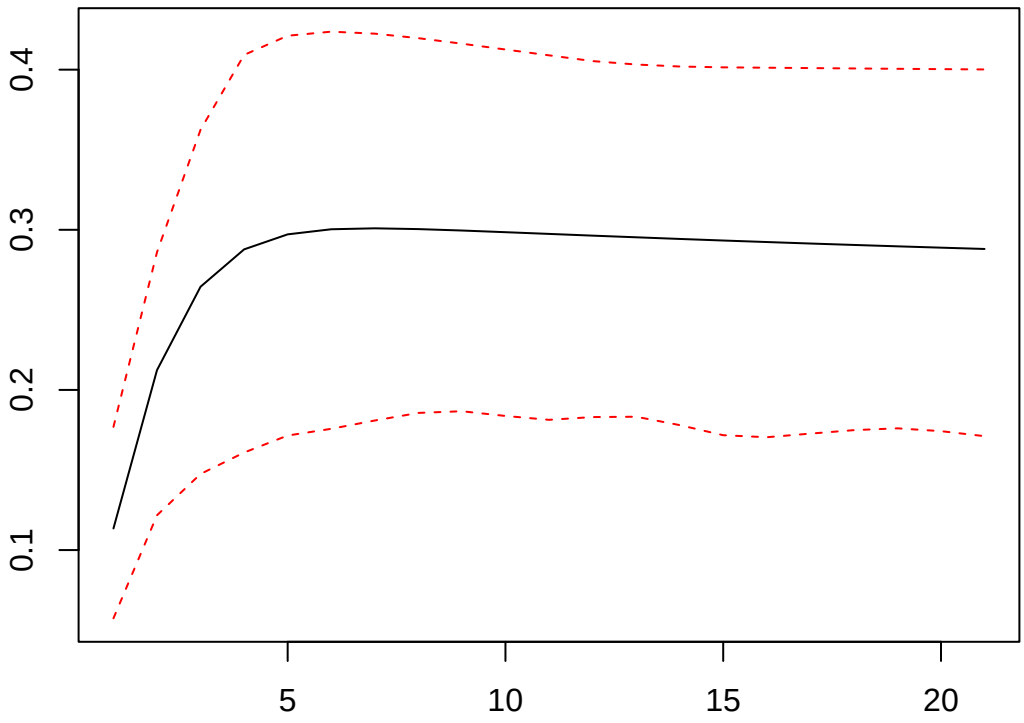


6.9 IRF: Impulse response functions:

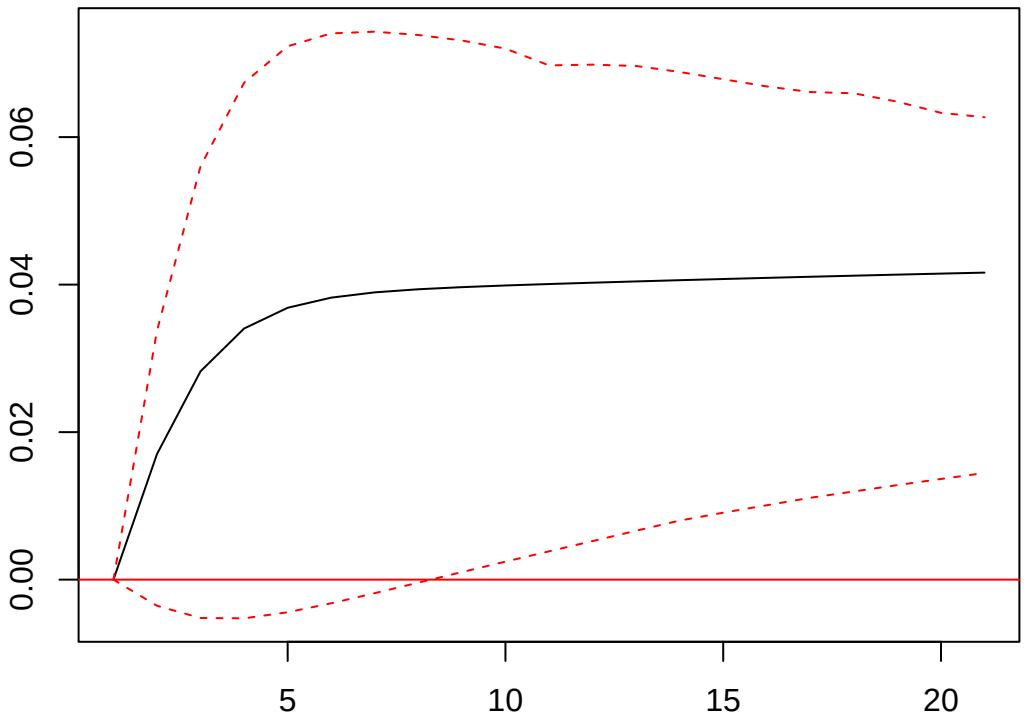
```
   #Short run (non-cumulative):

plot(irf(VECM, impulse="TCU", n.ahead=20, ortho=TRUE, cumulative=FALSE, boot=TRUE, ci=0.90, runs=100,
```



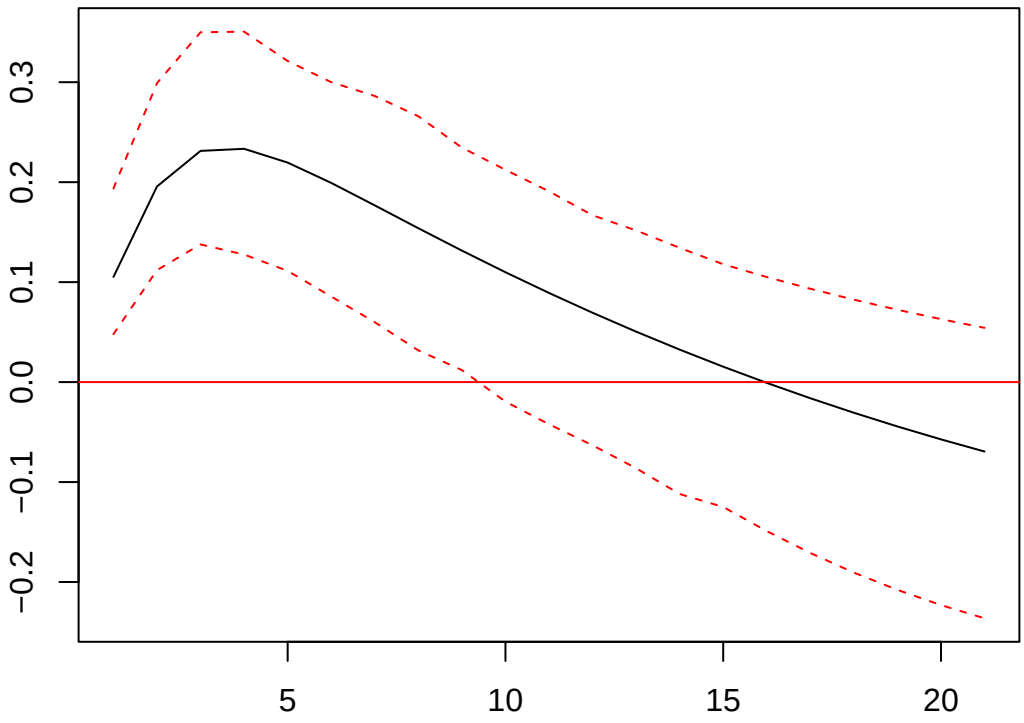

```
plot(irf(VECM,  impulse="Inflation", response="TCU", n.ahead=20, ortho=TRUE, cumulative=FALSE, boot=TR
          main="Inflation to TCU", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```



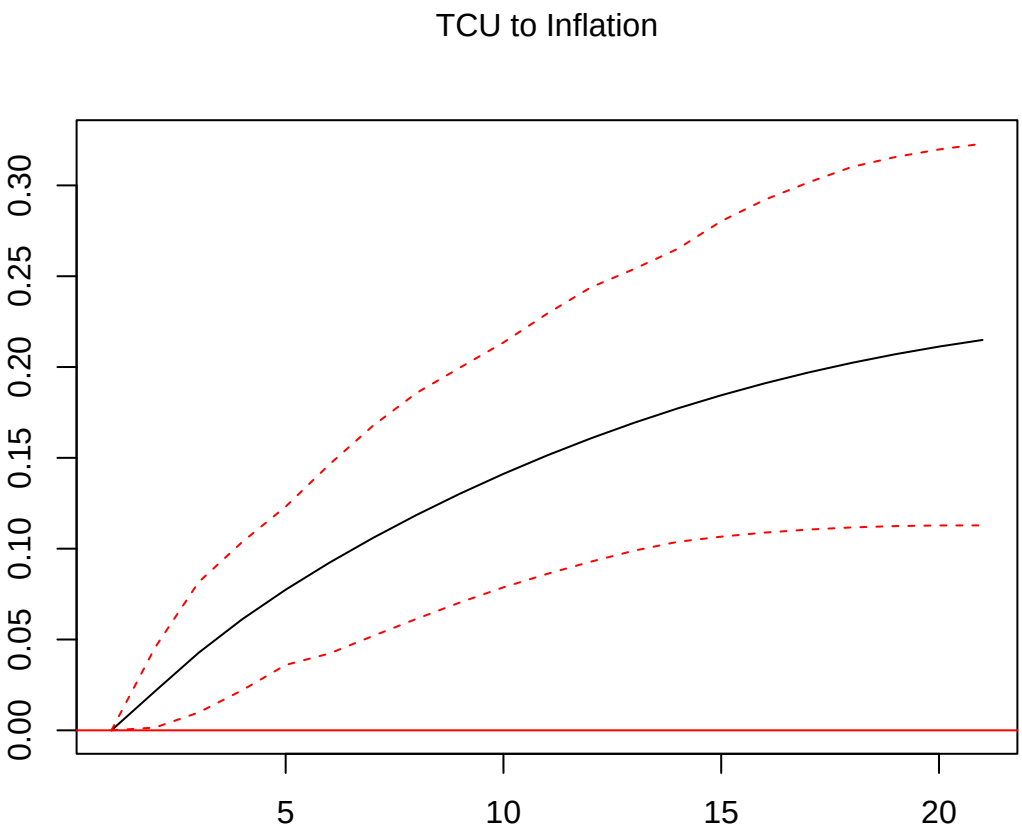

```
plot(irf(var.ordered.none,  impulse="TCU", response="Inflation", n.ahead=20, ortho=TRUE, cumulative=F
          main="TCU to Inflation", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```




```
plot(irf(var.ordered.const,  impulse="Inflation", response="TCU", n.ahead=20, ortho=TRUE, cumulative=
          main="Inflation to TCU", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```

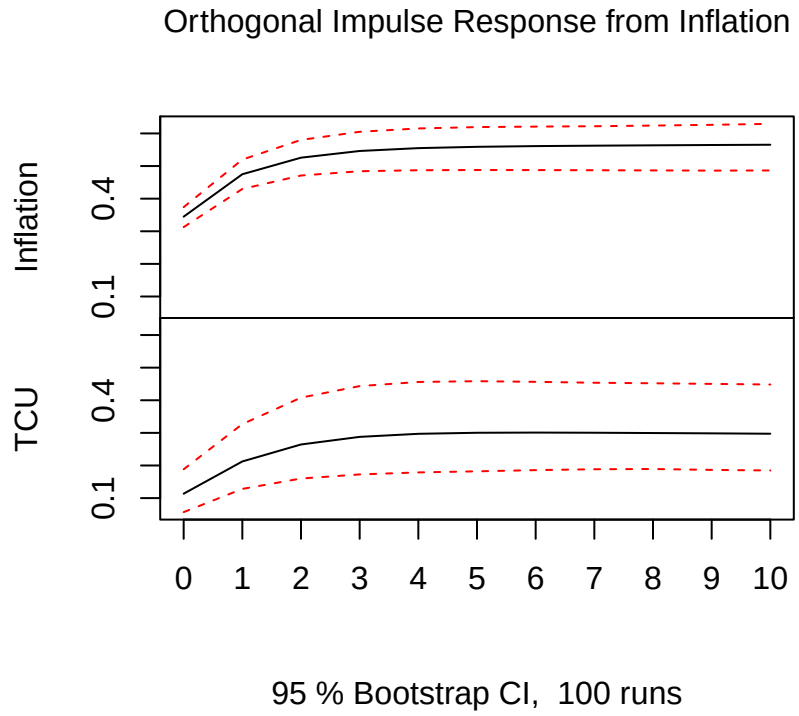


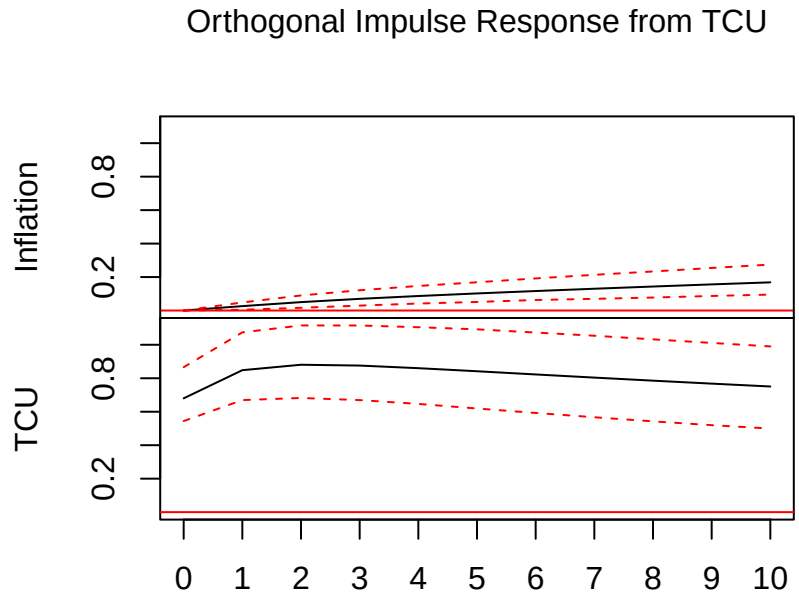

```
plot(irf(var.ordered.const,  impulse="TCU", response="Inflation", n.ahead=20, ortho=TRUE, cumulative=
          main="TCU to Inflation", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```



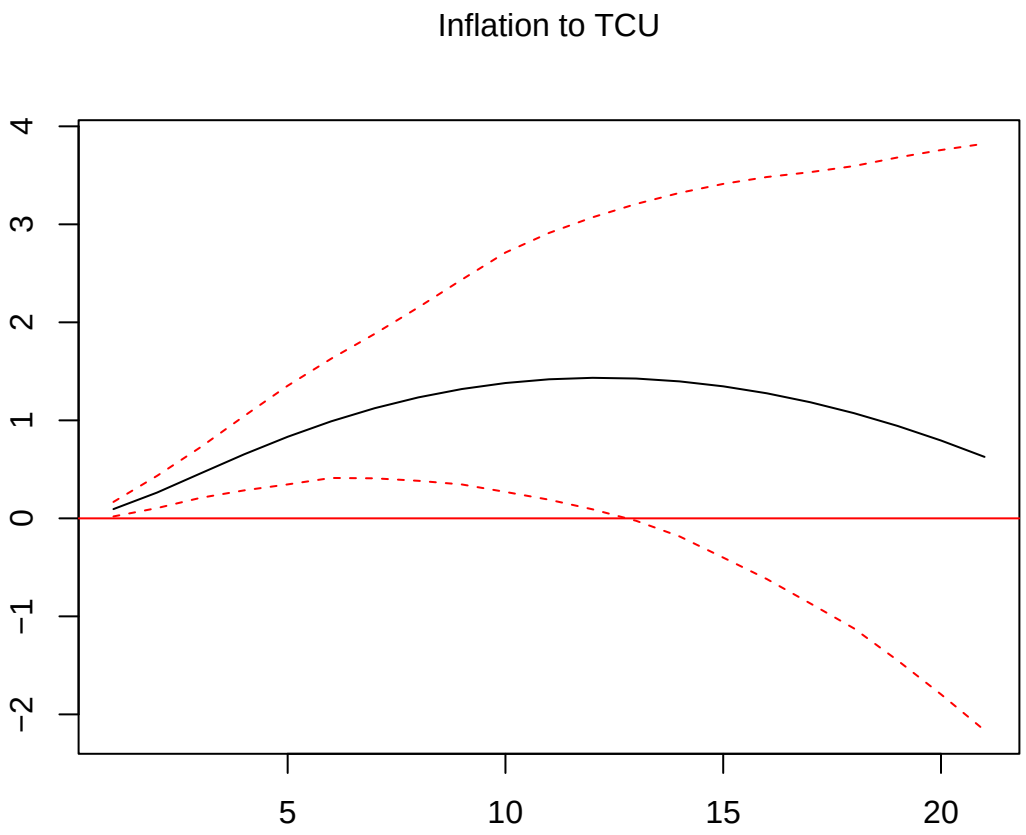
6.10 Long run (cumulative):

```
par( mfrow=c(1,1))
plot(irf(VECM), cumulative=TRUE)
```

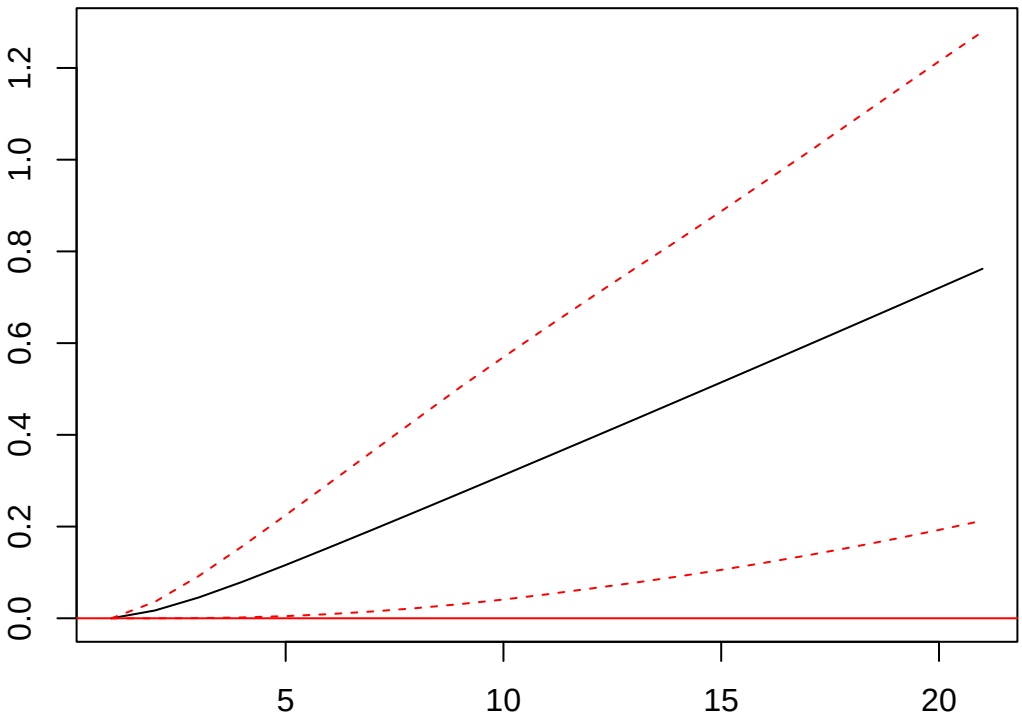

Orthogonal Impulse Response from Inflation
Inflation
TCU
0.1
0.4
0.1
0.4
0
1
2
3
4
5
6
7
8
9
10
95 % Bootstrap CI, 100 runs



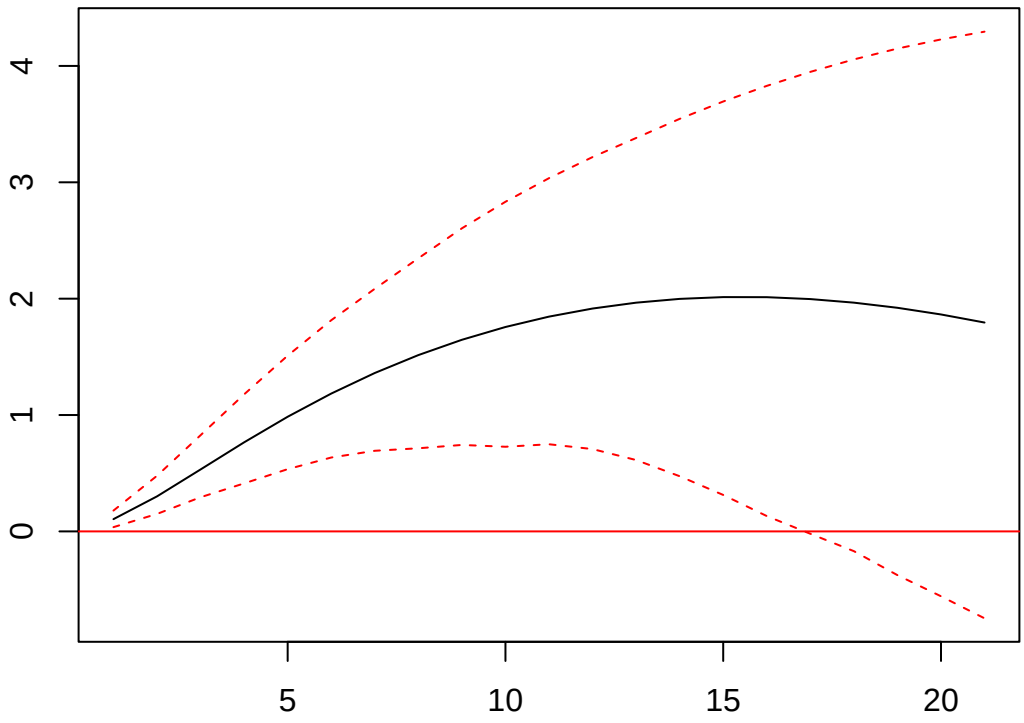

```
plot(irf(var.ordered.none,  impulse="Inflation", response="TCU", n.ahead=20, ortho=TRUE, cumulative=TRU
           main="Inflation to TCU", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```



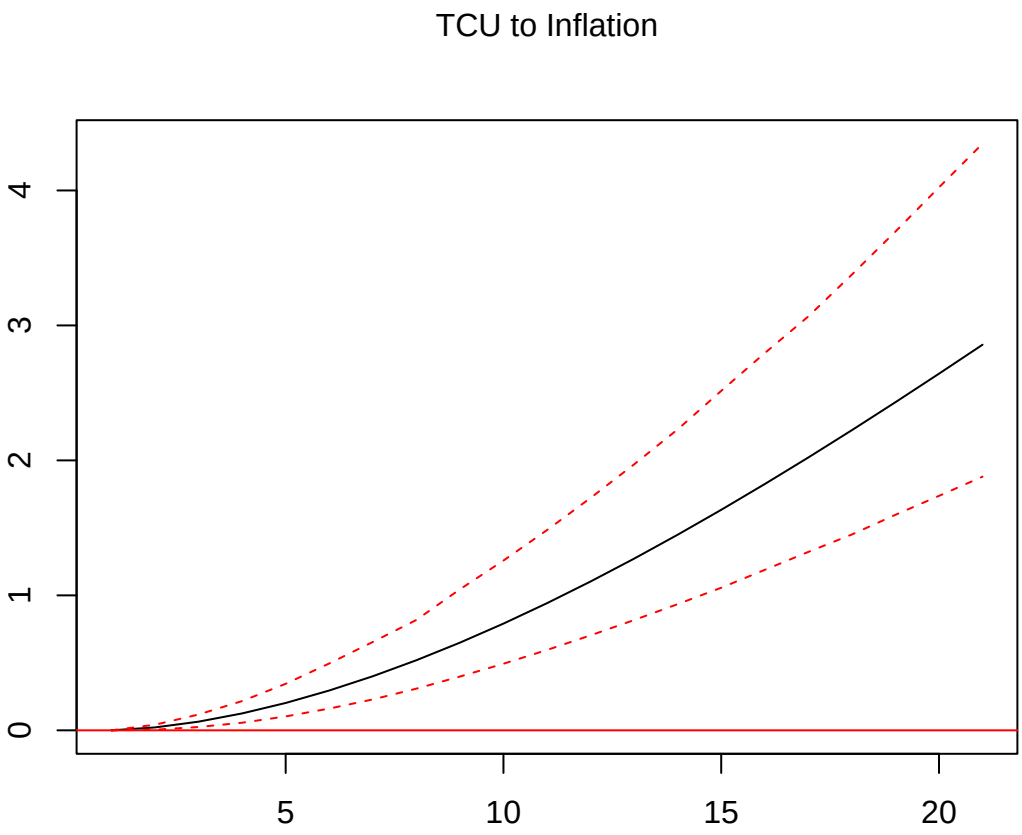

```
plot(irf(var.ordered.none,  impulse="TCU", response="Inflation", n.ahead=20, ortho=TRUE, cumulative=TRU
           main="Inflation to TCU", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```




```
plot(irf(var.ordered.const,  impulse="Inflation", response="TCU", n.ahead=20, ortho=TRUE, cumulative=TRU
           main="Inflation to TCU", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```




```
plot(irf(var.ordered.const,  impulse="TCU", response="Inflation", n.ahead=20, ortho=TRUE, cumulative=TRU
           main="TCU to Inflation", xlab="Lag", ylab="", sub="", oma=c(3,0,3,0))
```



6.11 Toda-Yamamoto Causality Test (Granger-causality test for non-stationary time series)

```
# Toda-Yamamoto Causality Test (Granger-causality test for non-stationary time series)


   Toda.Yamamoto = function(var) {

       # Add the extra lag to the VAR model:

     ty.df              =  eval(var$call$y);
     ty.varnames        =  colnames(ty.df);
     ty.lags            =  var$p + 1;
     ty.augmented_var   =  VAR(ty.df, ty.lags, type=var$type);

     ty.results = data.frame(predictor = character(0), causes = character(0), chisq = numeric(0), p = nu

     for (current_variable in ty.varnames) {

        # Construct the restriction matrix to test if "current_variable" causes any of the other variab
        # We test if the lagged values of the "current variable" (ignoring the extra lag) are jointly i

    ty.restrictions = as.matrix(Bcoef(ty.augmented_var))*0+1;
    ty.coefres      = head(grep(current_variable, colnames(ty.restrictions), value=T), -1);
```

```
    ty.restrictions[which(rownames(ty.restrictions) != current_variable), ty.coefres] = 0;

        # Estimate the restricted VAR:

    ty.restricted_var = restrict(ty.augmented_var, 'manual', resmat=ty.restrictions);

    for (k in 1:length(ty.varnames)) {

      if (ty.varnames[k] != current_variable) {

        my.wald    = waldtest(ty.augmented_var$varresult[[k]], ty.restricted_var$varresult[[k]], test='C

        ty.results = rbind(ty.results, data.frame(

                                          predictor  =  current_variable,
                                             causes  =  ty.varnames[k],
                                              chisq  =  as.numeric(my.wald$Chisq[2]),
                                                  p  =  my.wald$`Pr(>Chisq)`[2])
                                                  );
         }}}
      return(ty.results);
         }


   # Run the Toda-Yamamoto causality test:
   # Note that this part uses the VAR model you estimated before.

        var = var.model.const

        Toda.Yamamoto(var)
```

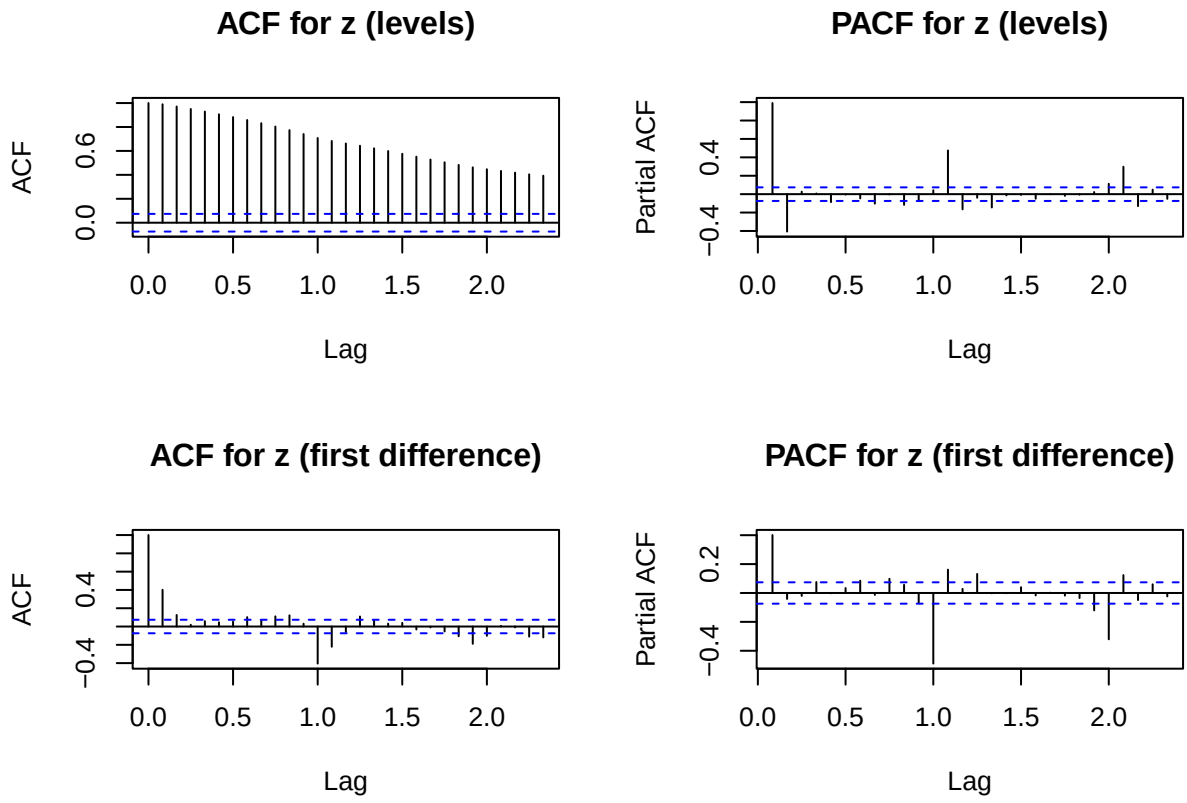
```
##   predictor    causes    chisq         p
## 1 Inflation       TCU 8.739786 0.0126526
## 2       TCU Inflation 7.211792 0.0271631
```

According to the Toda-Yamamoto Causality Test we can see that both series cause each other at the 5% confidence interval.

7 Arima model for forecasting

```
# Select a time series:
# We need to use the time series in levels, even if they have a unit root and thus are non-stationary
# No need to first-difference the data at this point


z = Inflation
```

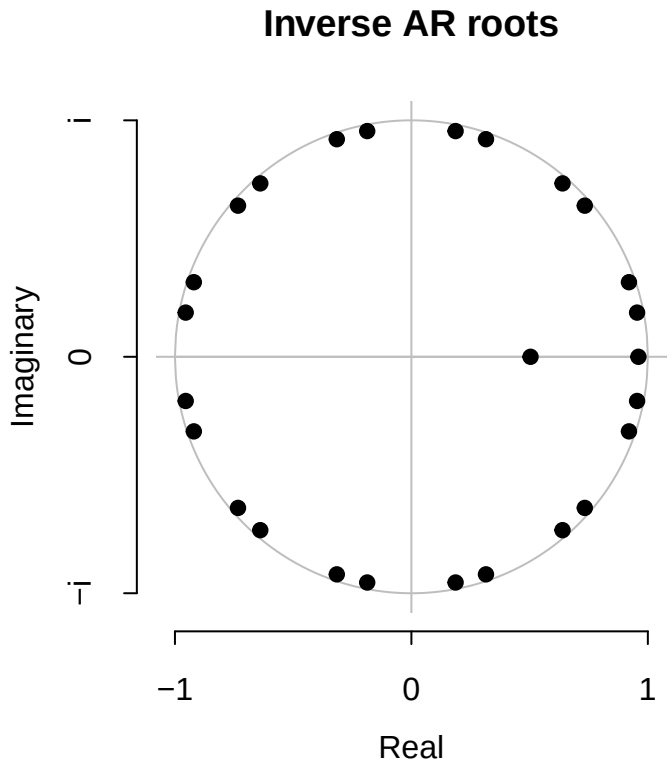
```r
#------- ACF and PACF ----------------------------------------------------------------------------

par(mfrow=c(2,2))
acf     (z,          main="ACF for z (levels)")
pacf    (z,          main="PACF for z (levels)")
acf     (diff(z),    main="ACF for z (first difference)")
pacf    (diff(z),    main="PACF for z (first difference)")
```



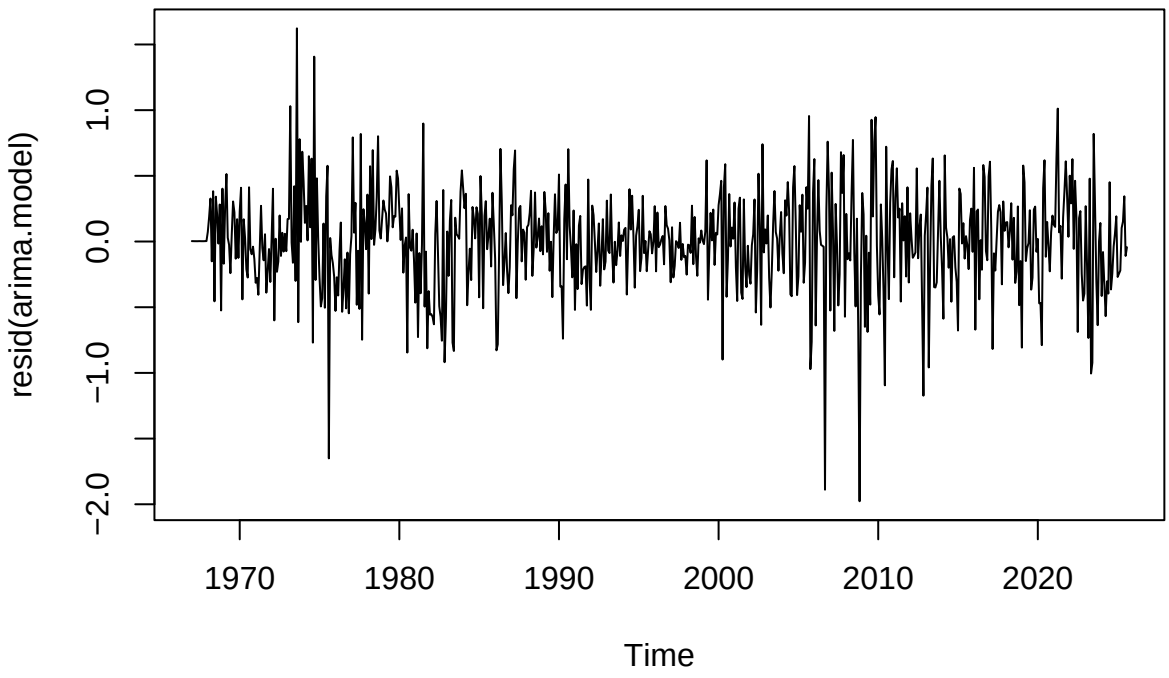

```r
# "forecast::auto.arima()" means you are using function "auto.arima()" from package "forecast"


arima.model = forecast::auto.arima(z,

                                   D = 1,
                                   stationary = FALSE,
                                   ic = c("aicc", "aic", "bic"),
                                   stepwise = FALSE,
                                   approximation = FALSE,
                                   seasonal = TRUE,
                                   allowdrift = TRUE   )


arima.model
```

```
## Series: z
## ARIMA(2,0,0)(2,1,0)[12]
##
## Coefficients:
##          ar1      ar2     sar1     sar2
##       1.4640  -0.4838  -0.9772  -0.5148
## s.e.  0.0338   0.0337   0.0331   0.0326
##
## sigma^2 = 0.1518:  log likelihood = -336
## AIC=682   AICc=682.09   BIC=704.7
```

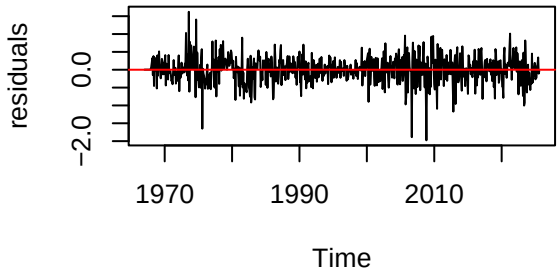
```
plot(arima.model)
```

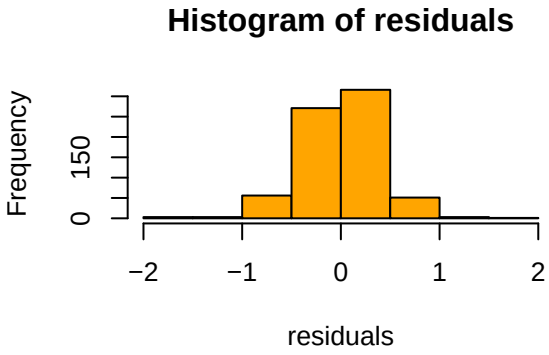


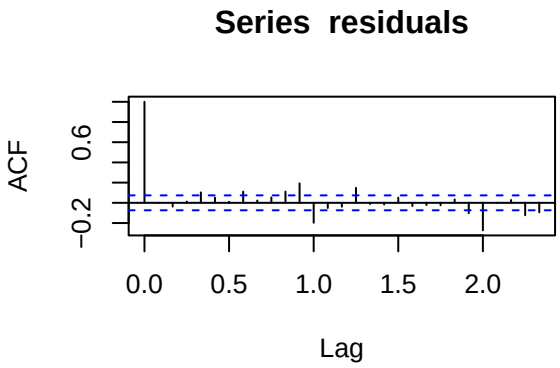

```
# Plot the residuals of the ARIMA model
# The model residuals must not have any systematic behavior
# Any systematic behavior indicates that the residuals are not white noise


plot(resid(arima.model))
```

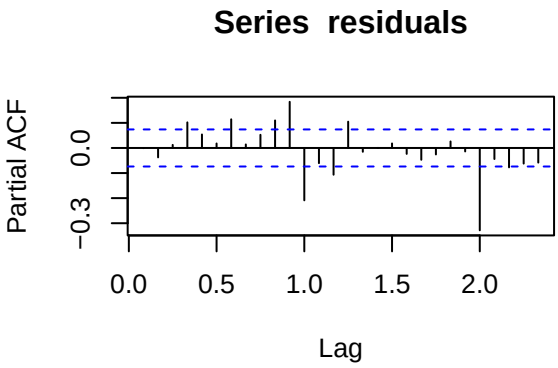



```
# Plot the histogram of the residuals, together with the ACF and PACF of the residuals:


residuals = resid(arima.model)



par(mfrow = c(2, 2))
plot( residuals )                  # Plot the residuals over time
abline( h=0, col="red" )
hist( residuals, col="orange")     # Plot histogram of the residuals
acf( residuals )                  # Plot ACF
pacf( residuals )                  # Plot PACF
```










```
#---- BDS test -----
#  Ho: series of i.i.d. random variable
#  Ha: series is not i.i.d.


bds.test(residuals)
```

```
##
##   BDS Test
##
## data:  residuals
##
## Embedding dimension =  2 3
##
## Epsilon for close points =  0.1927 0.3854 0.5781 0.7708
##
## Standard Normal =
##       [ 0.1927 ] [ 0.3854 ] [ 0.5781 ] [ 0.7708 ]
## [ 2 ]     4.8857     5.1733     5.2306     4.9963
## [ 3 ]     7.8294     7.5309     6.7870     5.9224
##
## p-value =
##       [ 0.1927 ] [ 0.3854 ] [ 0.5781 ] [ 0.7708 ]
## [ 2 ]          0          0          0          0
## [ 3 ]          0          0          0          0
```

```r
#---- Jarque-Bera Normality Test ------
# Ho = series is normaly distributed = kurtosis is zero and skewness is zero
# Ha = series is not normally distributed
# Using package "tseries"


jarque.bera.test(residuals)
```

```
##
##  Jarque Bera Test
##
## data:  residuals
## X-squared = 184.41, df = 2, p-value < 2.2e-16
```

```r
jarque.bera.test(resid(arima.model))
```

```
##
##  Jarque Bera Test
##
## data:  resid(arima.model)
## X-squared = 184.41, df = 2, p-value < 2.2e-16
```

```r
#---- Shapiro-Wilk Normality Test -----
# Ho = series is normaly distributed = kurtosis is zero and skewness is zero
# Ha = series is not normally distributed


shapiro.test(resid(arima.model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(arima.model)
## W = 0.97649, p-value = 3.292e-09
```

```r
#---- Ljung-Box -----
# Use lag > fitdf, in which fitdf=(p+q) because the series we are testing are the residuals from an est
# H0: series is independent; H1: series has dependence across lags
# Test statistic is distributed as a chi-squared random variable
# Desired result: p-value of the Ljung-Box test on the ARIMA residuals is high and, hence, the ARIMA re
# But if p-value is small then reject the null and conclude that the residuals are auto-correlated and,


Box.test(resid(arima.model), lag = 2, fitdf=0)
```

```
##
##  Box-Pierce test
##
## data:  resid(arima.model)
## X-squared = 0.99549, df = 2, p-value = 0.6079
```

```r
Box.test(resid(arima.model), lag = 2, type= "Ljung", fitdf=0)
```

```
##
##  Box-Ljung test
##
## data:  resid(arima.model)
## X-squared = 1.0012, df = 2, p-value = 0.6062
```

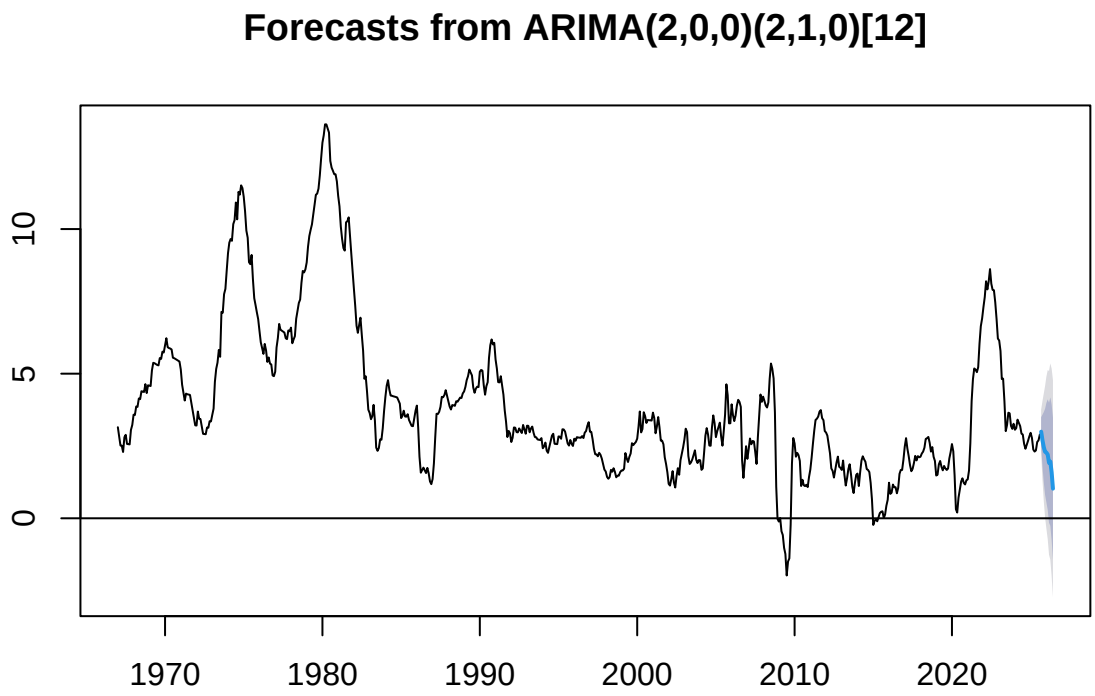
```r
Box.test(resid(arima.model), lag = 3, type= "Ljung", fitdf=0)
```

```
##
##  Box-Ljung test
##
## data:  resid(arima.model)
## X-squared = 1.1101, df = 3, p-value = 0.7746
```

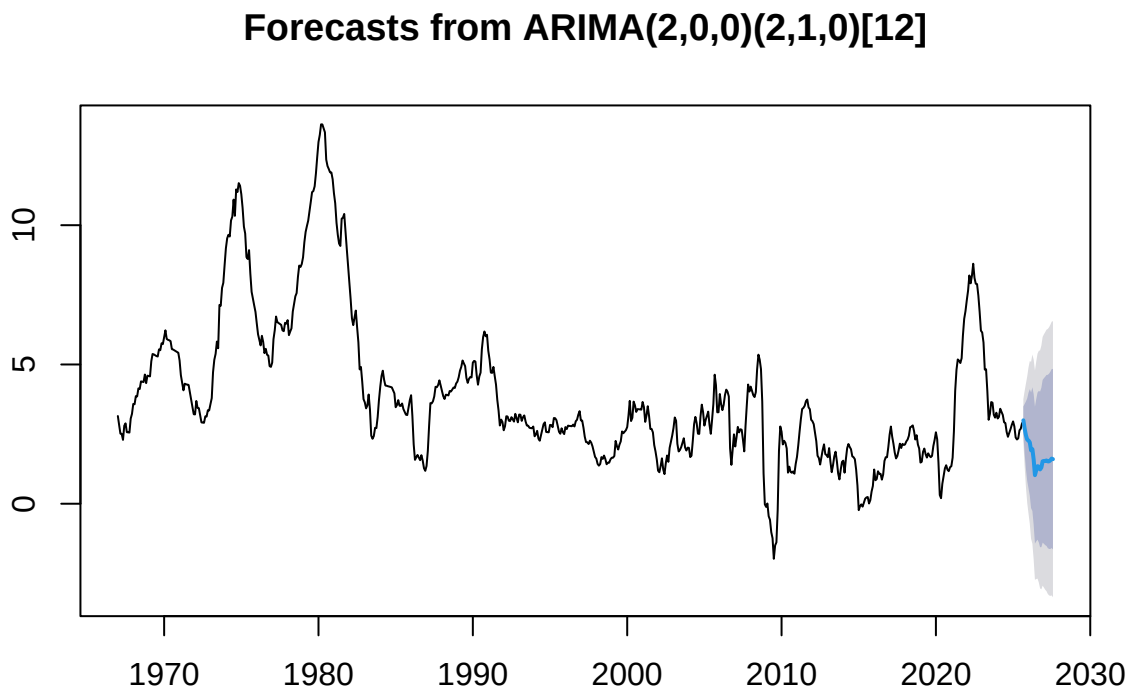
```r
Box.test(resid(arima.model), lag = 10, type= "Ljung", fitdf=0)
```

```
##
##  Box-Ljung test
##
## data:  resid(arima.model)
## X-squared = 30.744, df = 10, p-value = 0.0006467
```

```r
# Use the estimated ARIMA model to forecast future values of the time series:

# Out-of-sample forecasting:
# "forecast::forecast()" means you are using function forecast() from package "forecast"


ARIMA.forecast = forecast::forecast(arima.model, h = 10)

plot(ARIMA.forecast)
abline(h=0)
```




```
plot(forecast::forecast(arima.model))
```



From this forecast we can see that inflation should increase before decreasing to its previous level.

8 Conclusion

Overall we saw that the series are correlated to each other, we saw that the residuals of the TCU time series were not normally distributed. Both series have a unit root, which means that both have a stochastic trend, they didn't have a unit root in first differences. The cointegration tests proved that the two time series are cointegrated, which means that they have a long run relationship. We run the ARDL model in levels to understand the long run relationship between the two series, we used the the error correction form to test fo the significance of this long run relationship. the results show that Inflation has a positive cumulative effect on TCU. We used the VEC model which is the VAR model with the Error correction term. From the Impulse response function we saw that Inflation affects TCU in a positive war in both the short and long run. TCU also affects inflation in the short and long run as well. We then used the ARIMA model on the inflation series for forecasting. The out of sample ARIMA model forecasted an increase in inflation which would follow by a decrease decrease.