Tutorial: Cointegration

by Kevin Kotzé

This tutorial will make use of the files in the `T13-coint.zip` folder. Thereafter, you will need to open the `T13-coint.Rproj` file to open the project.

1 Modelling the relationship between commodity prices

The first exercise considers the relationship between the price for gold and silver, when using the Engle-Granger procedure for cointegration. This example is contained in the file `tut13-EngleGranger.R`. To start off we can clear all the variables from the current environment and close all the plots.

```
rm(list = ls())
graphics.off()
```

Thereafter, we will need to make use of the `vars` package, so we make use of the `library` command.

```
library(tidyverse)
library(vars)
```

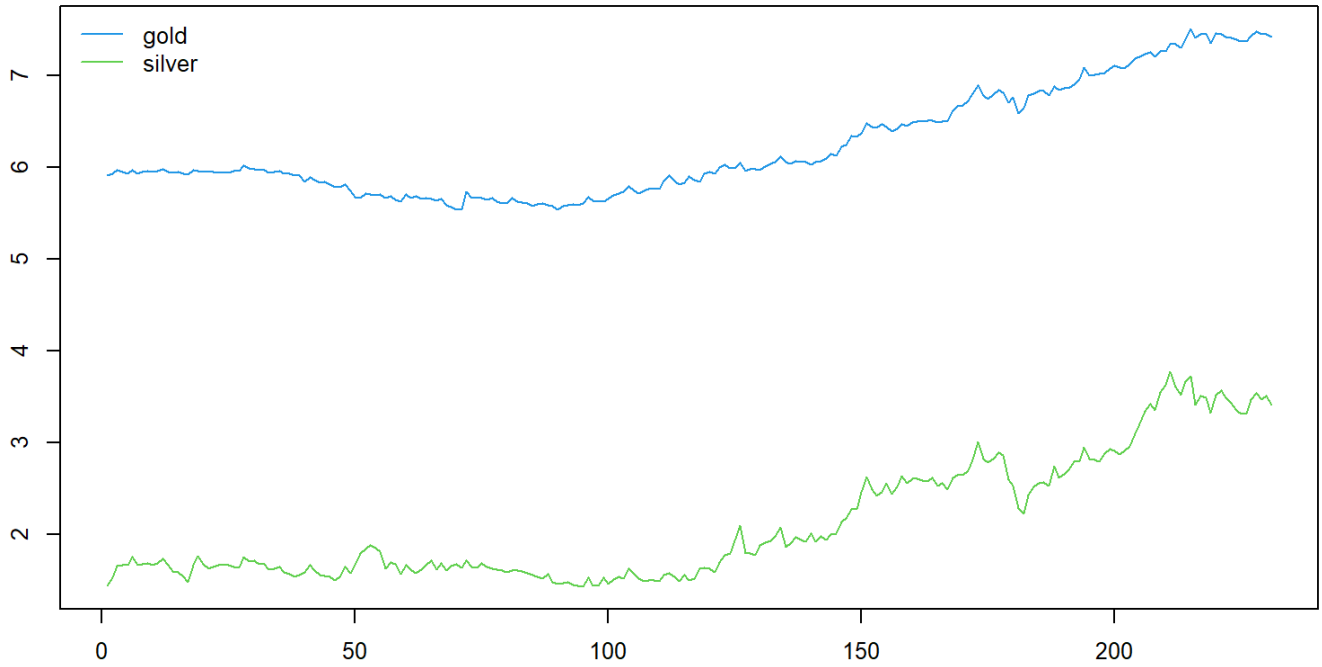
If you need install these packages run the following routine: `install.packages("vars")`.

As this data is contained in a `.csv` file we need to use the `read_csv()` command. This allows us to load the data and create the tibble object that contains the time series variables `gold` and `silver`, for the prices of these commodities. The dataset also includes information on platinum and palladium prices that you could play around with too. In what follows, we make use of the natural logarithm of these variables.

```
dat <- read_csv('comodity_price.csv')

dat <- dat %>%
  mutate(
    gold = log(gold),
    silver = log(silver),
    plat = log(plat),
    pall = log(pall)
  )
```

To ensure that this has all been completed correctly, we can plot the data.

```
par(mfrow = c(1, 1),
    mar = c(2.2, 2.2, 1, 2.2),
    cex = 0.8)
plot.ts(
  cbind(dat$gold, dat$silver),
  plot.type = "single",
  ylab = "",
  col = 4:3
)
legend(
  "topleft",
  legend = c("gold", "silver"),
  col = 4:3,
  lty = 1,
  bty = 'n'
)
```



These plots would suggest that the data is trending. To perform the respective unit root tests on these variables we proceed with the general to specific procedure for the augmented Dickey-Fuller test.

```
dat$gold %>%
  ur.df(., type = "trend", selectlags = c("BIC")) %>%
  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
## -0.188979 -0.026569 -0.001586  0.024103  0.188728
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.410e-02  5.049e-02   1.270  0.20555
## z.lag.1     -1.309e-02  9.335e-03  -1.403  0.16210
## tt           2.168e-04  8.273e-05   2.620  0.00939 **
## z.diff.lag  -1.717e-01  6.550e-02  -2.622  0.00934 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04515 on 225 degrees of freedom
## Multiple R-squared:  0.05981,    Adjusted R-squared:  0.04727
## F-statistic: 4.771 on 3 and 225 DF,  p-value: 0.003035
##
##
## Value of test-statistic is: -1.4027 5.041 4.439
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

```
dat$gold %>%
  ur.df(., type = "drift", selectlags = c("BIC")) %>%
  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
## -0.186504 -0.027319 -0.004936  0.023856  0.193189
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.037889   0.032573  -1.163   0.2460
## z.lag.1      0.007307   0.005217   1.401   0.1627
## z.diff.lag  -0.164912   0.066296  -2.488   0.0136 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04573 on 226 degrees of freedom
## Multiple R-squared:  0.03112,    Adjusted R-squared:  0.02255
## F-statistic: 3.629 on 2 and 226 DF,  p-value: 0.02809
##
##
## Value of test-statistic is: 1.4007 4.0244
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

```
dat$gold %>%
  ur.df(., type = "none", selectlags = c("BIC")) %>%
  summary()
```

```
##
## ###############################################
## # 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
## -0.184167 -0.029417 -0.004712  0.022142  0.188803
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1     0.0012656  0.0004895   2.586   0.0103 *
## z.diff.lag -0.1542830  0.0657140  -2.348   0.0197 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04577 on 227 degrees of freedom
## Multiple R-squared:  0.0444, Adjusted R-squared:  0.03598
## F-statistic: 5.274 on 2 and 227 DF,  p-value: 0.00577
##
##
## Value of test-statistic is: 2.5856
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

The first test with both components and lags that depend upon Bayesian Information Criteria suggests that we are unable to reject the null of a unit root as the calculated test statistic is -1.4027 and the critical values are, -3.99, -3.43, and -3.13 for the 1%, 5% and 10% significance levels. The F-test statsitics suggest that we can then drop the time trend. In the second test with just a constant, we are again unable to reject the null of a unit root as the calculated test statistic is 1.4007 and the critical values are, -3.46, -2.88, and -2.57 for the 1%, 5% and 10% significance levels. After considering the F-test statsitics the last case, where there are no deterministic elements provides a calculated test statistic of 2.5856 for critical values of -2.58, -1.95, and -1.62. Then it is worthwhile also checking that the variable is not $I2$ by performing the test on the first difference of the variable. In this case the first difference is clearly stationary as the calculated test statistic is -13.9207 and the critical values are, -2.58, -1.95, and -1.62. Hence gold is clearly integrated of order 1.

Similar evidence is provided for the price of silver.

```
dat$silver %>%
  ur.df(., type = "trend", selectlags = c("BIC")) %>%
  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
## -0.303769 -0.048608 -0.006425  0.051870  0.195662
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0333545  0.0219810   1.517   0.1306
## z.lag.1     -0.0306429  0.0167447  -1.830   0.0686 .
## tt           0.0003491  0.0001671   2.089   0.0378 *
## z.diff.lag  -0.0720055  0.0666563  -1.080   0.2812
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08551 on 225 degrees of freedom
## Multiple R-squared:  0.02589,    Adjusted R-squared:  0.01291
## F-statistic: 1.994 on 3 and 225 DF,  p-value: 0.1158
##
##
## Value of test-statistic is: -1.83 2.2871 2.1844
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

```
dat$silver %>%
  ur.df(., type = "drift", selectlags = c("BIC")) %>%
  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
## -0.315787 -0.046701 -0.004613  0.057160  0.207666
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0102511  0.0191368   0.536    0.593
## z.lag.1     -0.0006064  0.0086449  -0.070    0.944
## z.diff.lag  -0.0831583  0.0669348  -1.242    0.215
##
## Residual standard error: 0.08614 on 226 degrees of freedom
## Multiple R-squared:  0.007003,   Adjusted R-squared:  -0.001785
## F-statistic: 0.7969 on 2 and 226 DF,  p-value: 0.452
##
##
## Value of test-statistic is: -0.0701 1.2305
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

```
dat$silver %>%
  ur.df(., type = "none", selectlags = c("BIC")) %>%
  summary()
```

```
##
## ###############################################
## # 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
## -0.32183 -0.04435 -0.00334  0.05210  0.20662
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1     0.003812   0.002582   1.477    0.141
## z.diff.lag -0.086238   0.066583  -1.295    0.197
##
## Residual standard error: 0.08601 on 227 degrees of freedom
## Multiple R-squared:  0.01475,    Adjusted R-squared:  0.006074
## F-statistic:   1.7 on 2 and 227 DF,  p-value: 0.185
##
##
## Value of test-statistic is: 1.4768
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
dat$silver %>%
  diff() %>%
  ur.df(., type = "trend", selectlags = c("BIC")) %>%
  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
## -0.308047 -0.046231 -0.000964  0.048791  0.198855
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.418e-03  1.139e-02  -0.300   0.7643
## z.lag.1     -1.238e+00  9.743e-02 -12.702   <2e-16 ***
## tt           1.144e-04  8.606e-05   1.329   0.1853
## z.diff.lag   1.312e-01  6.603e-02   1.988   0.0481 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08511 on 224 degrees of freedom
## Multiple R-squared:  0.5555, Adjusted R-squared:  0.5495
## F-statistic:  93.3 on 3 and 224 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -12.7017 53.7927 80.67
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

1.1 Model selection and estimation

To create a bivariate object for the two time series we can make use of the `dplyr::select()` command.

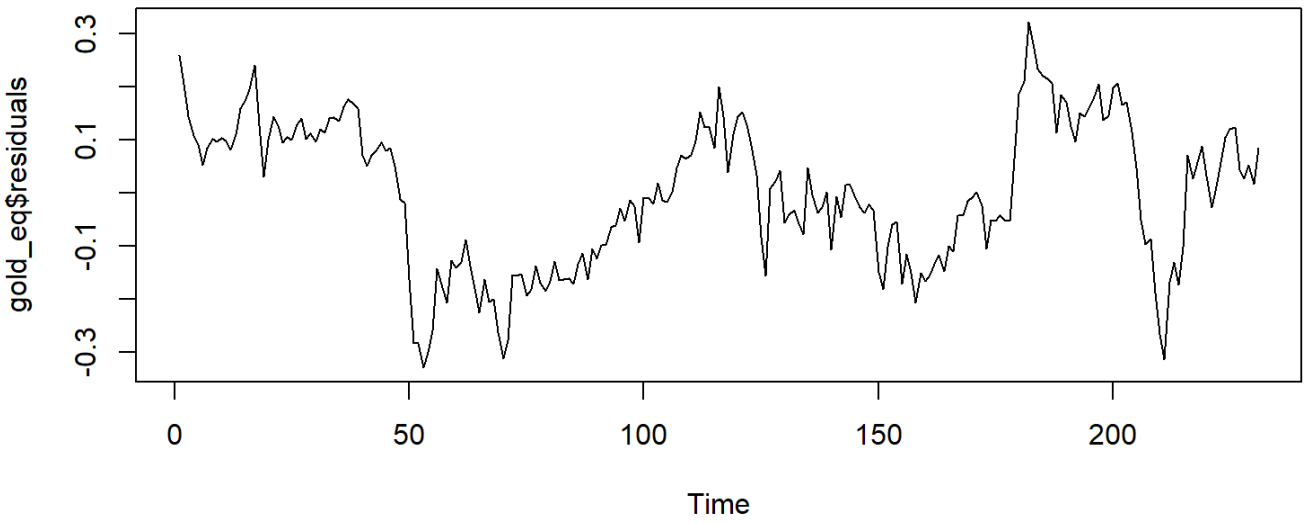
```
dat_lr <- dat %>%
  dplyr::select(gold, silver)
```

We then estimate long run regression with constant, by using the `lm` command. Note that we are primarily interested in the residuals from this equation, which would suggest that the variables are cointegrated if the residuals are stationary.

```
gold_eq <- lm(gold ~ silver, data = dat_lr)
summary(gold_eq)
```

```
##
## Call:
## lm(formula = gold ~ silver, data = dat_lr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32883 -0.11073  0.00267  0.10937  0.32204
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.40958    0.03000  146.98   <2e-16 ***
## silver       0.85885    0.01348   63.72   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1367 on 229 degrees of freedom
## Multiple R-squared:  0.9466, Adjusted R-squared:  0.9464
## F-statistic:  4061 on 1 and 229 DF,  p-value: < 2.2e-16
```

```
plot.ts(gold_eq$residuals)
```



These residuals are clearly not trending, but to confirm that they are stationary we subject them to a unit root test.

```
gold_eq$residuals %>%
  ur.df(., lags = 1, type = "none") %>%
  summary()
```

```
##
## ###############################################
## # 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
## -0.145218 -0.030291  0.002356  0.028725  0.159969
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.08179    0.02536  -3.224  0.00145 **
## z.diff.lag  0.05215    0.06596   0.791  0.43001
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05125 on 227 degrees of freedom
## Multiple R-squared:  0.04403,    Adjusted R-squared:  0.03561
## F-statistic: 5.228 on 2 and 227 DF,  p-value: 0.006031
##
##
## Value of test-statistic is: -3.2245
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

To evaluate the test statistics we should use the critical values from Engle & Granger (1987) or Engle & Yoo (1987). These would suggest that the residuals are indeed stationary. This would imply that they may be cointegrated.

The next part of the procedure involves constructing the error correction model. Note that we need to select the number of lags of the dependent variable and then we can make sure that all the variables conform to the same shape. In this case we are going to use a single lag of the dependent variable in the construction of the model `gold.d = const + error.ecm1 + gold.d1 + silver.d1`. By taking the first difference of the variable, we lose the first observation. In addition, the first lag of the first difference would shorten the time-series by two observations. Hence, we perform the necessary variable transformations as follows:

```
dat_ecm <- tibble(
  gold_d = diff(dat$gold)[-1],
  silver_d = diff(dat$silver)[-1],
  error_ecm1 = gold_eq$residuals[-1:-2],
  gold_d1 = diff(dat$gold)[-(length(dat$gold) - 1)],
  silver_d1 = diff(dat$silver)[-(length(dat$silver) - 1)]
)
```

And to estimate the error correction model with a linear regression model, we specify:

```
ecm_gold <-
  lm(gold_d ~ error_ecm1 + gold_d1 + silver_d1, data = dat_ecm)
summary(ecm_gold)
```

```
##
## Call:
## lm(formula = gold_d ~ error_ecm1 + gold_d1 + silver_d1, data = dat_ecm)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.182090 -0.029358 -0.005345  0.023016  0.189071
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.007538   0.003074   2.452   0.0150 *
## error_ecm1   0.002260   0.022936   0.099   0.9216
## gold_d1     -0.166119   0.094907  -1.750   0.0814 .
## silver_d1    0.011437   0.051358   0.223   0.8240
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04603 on 225 degrees of freedom
## Multiple R-squared:  0.02294,    Adjusted R-squared:  0.009912
## F-statistic: 1.761 on 3 and 225 DF,  p-value: 0.1555
```

Note that the alpha is insignificant and close to zero. Therefore, gold is weakly exogenous with respect to the cointegrating parameters since it does not adjust to past deviations from long-run equilibrium.

In a similar fashion we could specify the error correction model, where the first difference of silver is the dependent variable.

```
ecm_silver <-
  lm(silver_d ~ error_ecm1 + gold_d1 + silver_d1, data = dat_ecm)
summary(ecm_silver)
```

```
##
## Call:
## lm(formula = silver_d ~ error_ecm1 + gold_d1 + silver_d1, data = dat_ecm)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.294970 -0.048732  0.000863  0.052758  0.234587
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.009787   0.005681   1.723   0.0863 .
## error_ecm1  -0.079381   0.042387  -1.873   0.0624 .
## gold_d1     -0.263448   0.175394  -1.502   0.1345
## silver_d1    0.003645   0.094913   0.038   0.9694
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08506 on 225 degrees of freedom
## Multiple R-squared:  0.036,  Adjusted R-squared:  0.02314
## F-statistic: 2.801 on 3 and 225 DF,  p-value: 0.0408
```

Note that in this case the speed of adjustment coefficient is negative and significant, so silver does all the work to get the two variables back towards the equilibrium path. This implies that there is Granger causality from gold to silver and that it takes about $1/0.079$ periods to return to equilibrium.

Then finally note that at least one of the signs for the speed of adjustment coefficients should be negative and significant if we are to conclude that the variables are to be cointegrated!

2 Deriving a measure of the real equilibrium exchange rate

In the next example, we make use of the Johansen model to derive a model for the South African real equilibrium exchange rate. To do so we are going to try to replicate the results of an article that appeared in the *South African Journal of Economics*, by MacDonald & Ricci (2004). This file has been labelled `tut13_MacRic` To start off we can clear all the variables from the current environment and close all the plots.

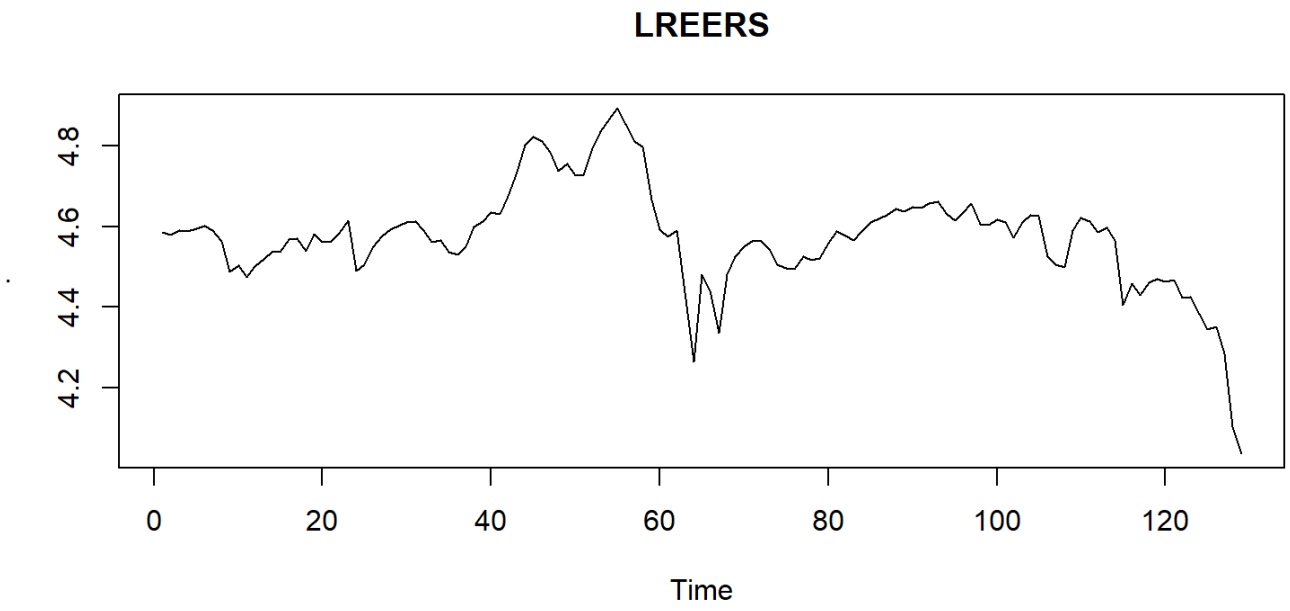
```
rm(list = ls())
graphics.off()
```

Thereafter, we will need to make use of the `vars` package, so we make use of the `library` command.

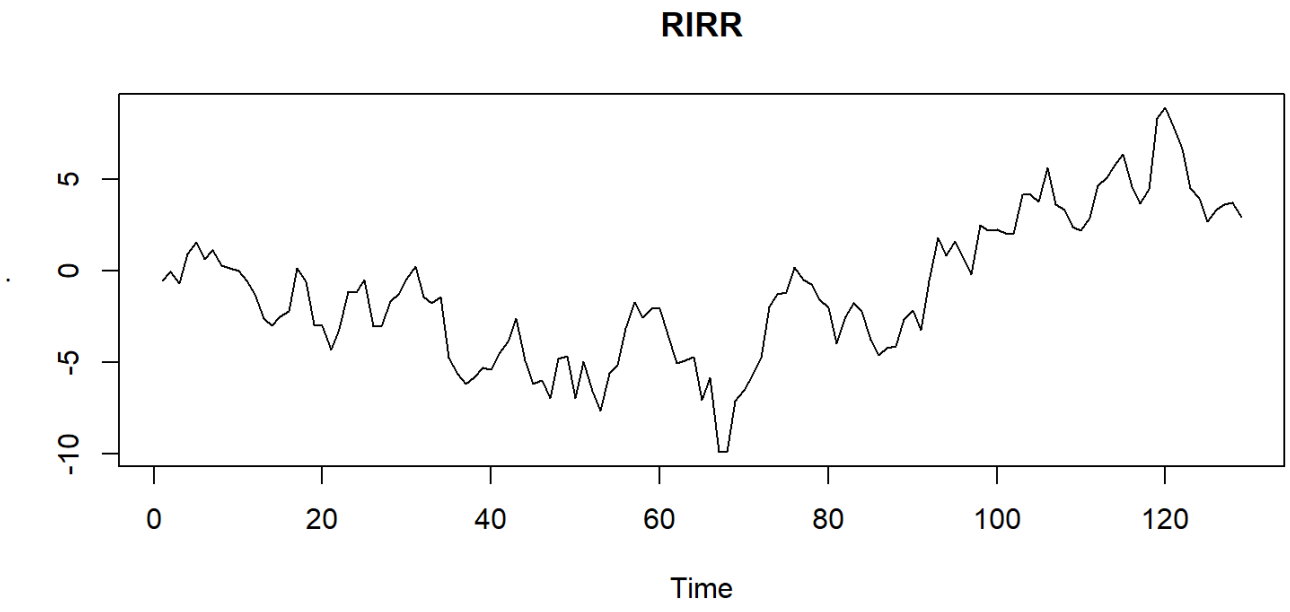
```
library(tidyverse)
library(vars)
```

As this data is contained in a `.csv` file we need to use the `read_csv()` command. This allows us to load the data, which we plot to inspect the general properties of the variables.

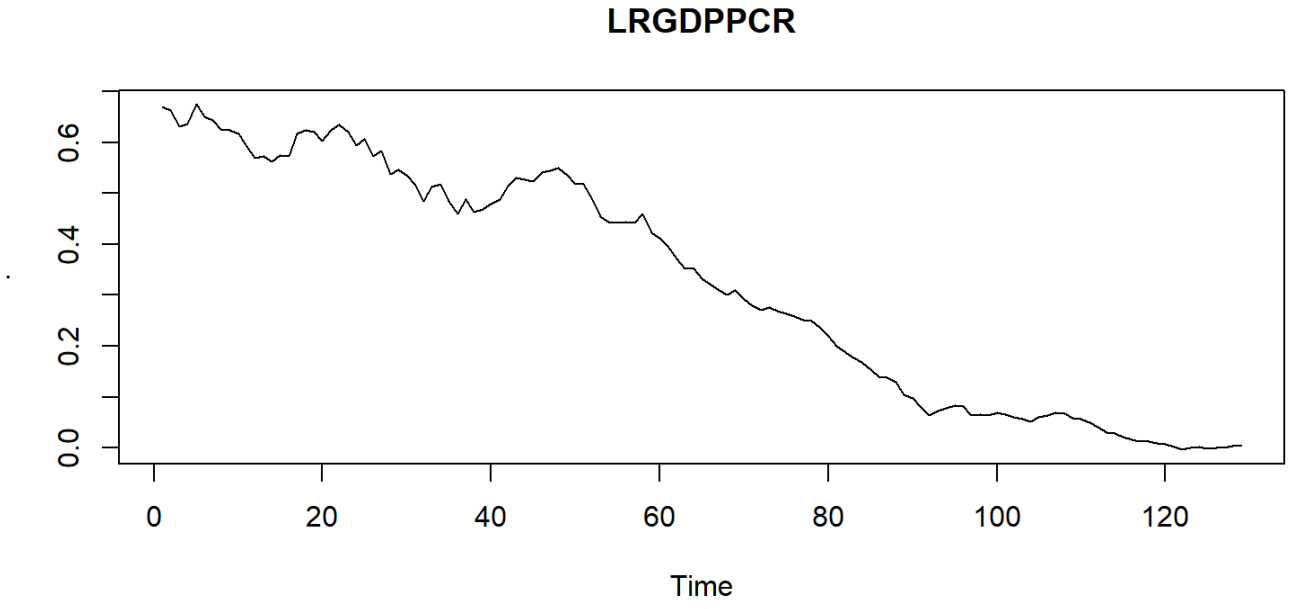
```
dat <- read_csv("Mac_Ric.csv")

dat$LREERS %>%
  plot.ts(., main = "LREERS")
```



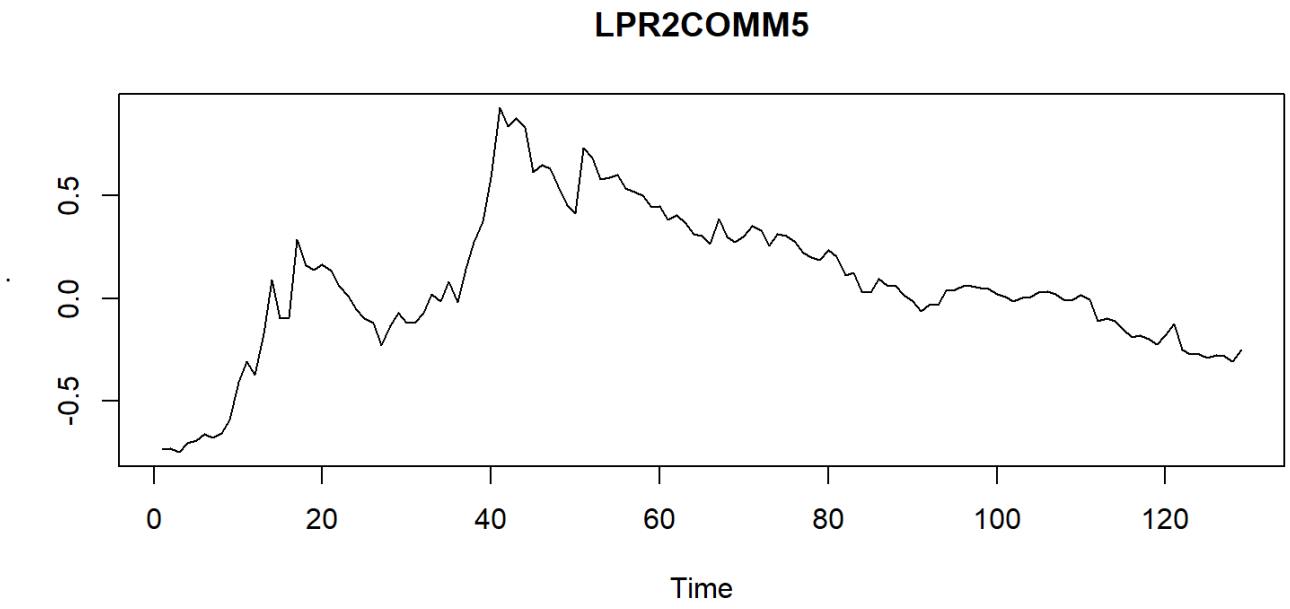

```
dat$RIRR %>%
  plot.ts(., main = "RIRR")
```



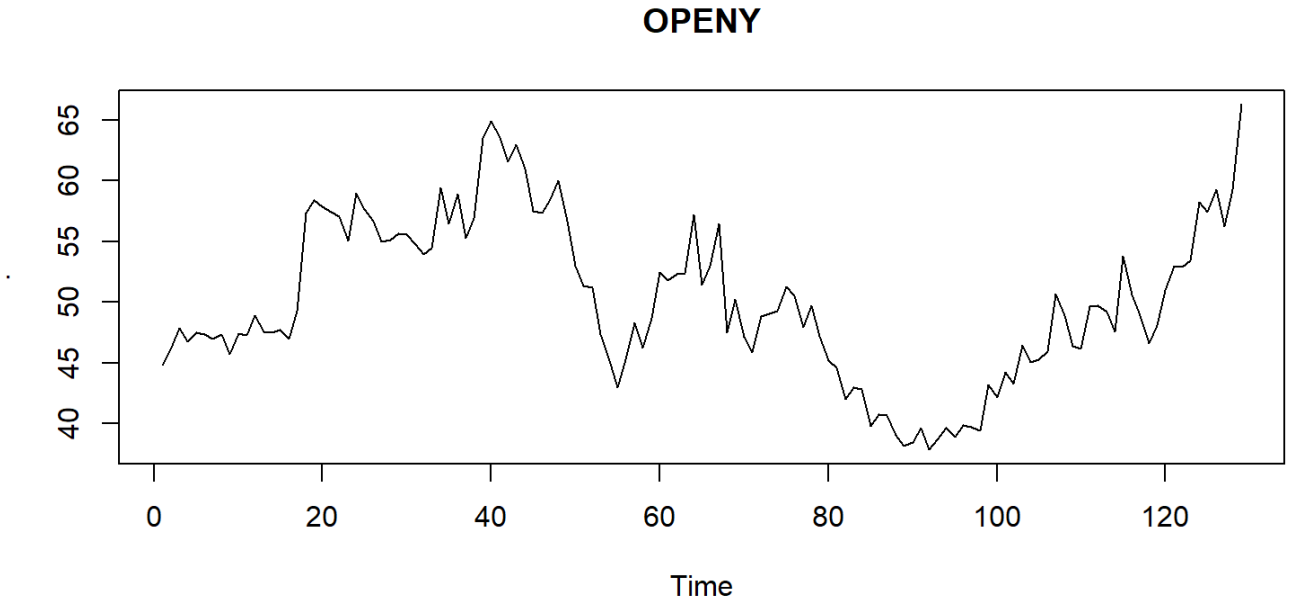

```
dat$LRGDPPCR %>%
  plot.ts(., main = "LRGDPPCR")
```



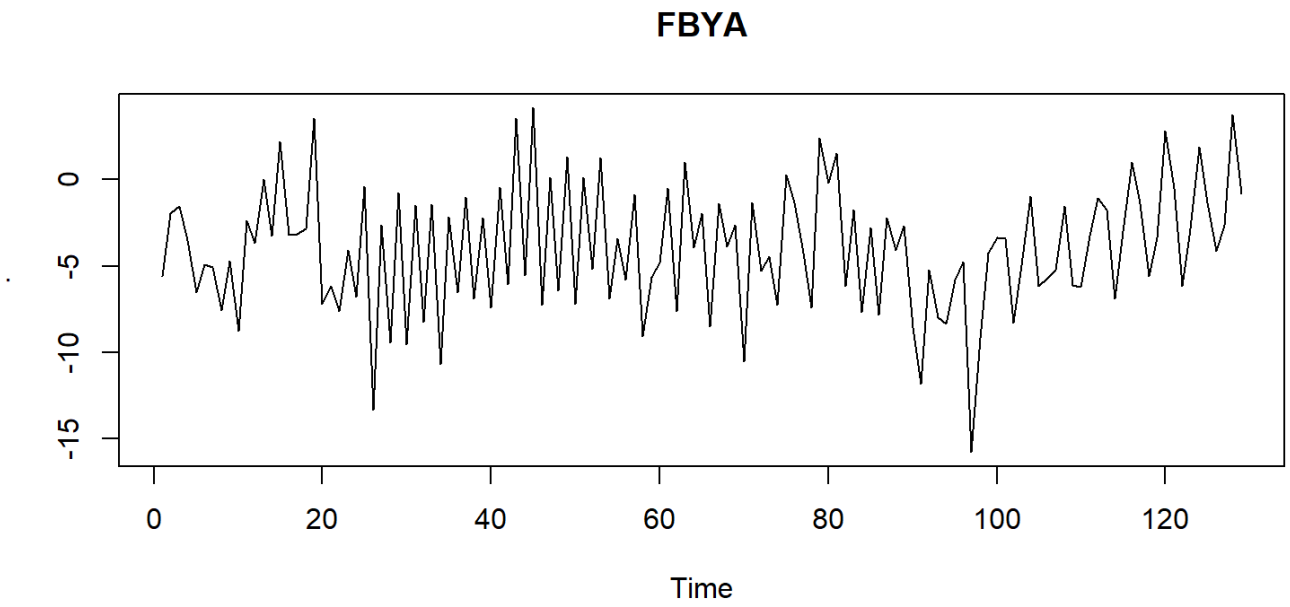

```
dat$LPR2COMM5 %>%
  plot.ts(., main = "LPR2COMM5")
```



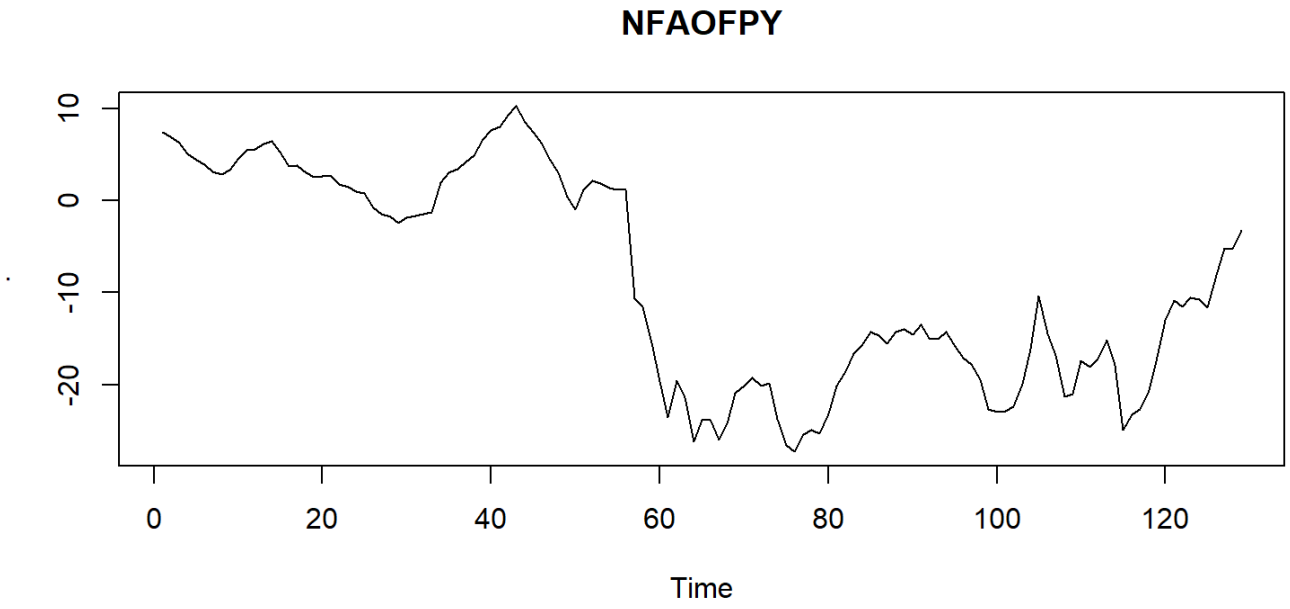

```
dat$OPENY %>%
  plot.ts(., main = "OPENY")
```




```
dat$FBYA %>%
  plot.ts(., main = "FBYA")
```




```
dat$NFAOFPY %>%
  plot.ts(., main = "NFAOFPY")
```



This dataset includes measures of the logarithm of real effective exchange rate, real interest rate relative to trading partners, logarithm of real GDP per capita relative to trading partners, real commodity prices, openness (ratio to GDP of exports and imports), ratio of fiscal balance to GDP, and the ratio to GDP of net foreign assets of the banking system. As is noted in the paper, all the necessary tests have been performed on these variables to ensure that they are of the correct integration order.

2.1 Estimate the model

To estimate the model we combine all the endogenous variables with the `dplyr::select()` command. The paper also makes use of a number of dummy variables, which are also placed in a separate `tibble()`.

```
dat_VAR <- dat %>%
  dplyr::select(LREERS,
                RIRR,
                LRGDPPCR,
                LPR2COMM5,
                OPENY,
                FBYA,
                NFAOFPY)

dat_EXO <- dat %>%
  dplyr::select(SDUMC1,
                SDUMC2,
                SDUMC3,
                DUMRER1,
                DUMRER2,
                DUMFBYA,
                DUMNFAOFPY)
```

We can then estimate the model parameters as follows:

```
VAR_est <- VAR(
  dat_VAR,
  p = 4,
  type = "const",
  season = NULL,
  exog = dat_EXO
)
summary(VAR_est)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: LREERS, RIRR, LRGDPPCR, LPR2COMM5, OPENY, FBYA, NFAOFPY
## Deterministic variables: const
## Sample size: 125
## Log Likelihood: -117.888
## Roots of the characteristic polynomial:
## 1.015 1.015 0.9641 0.8839 0.8822 0.8582 0.8582 0.8444 0.8444 0.799 0.799 0.7556 0.7556 0.7
401 0.7401 0.6946 0.6946 0.615 0.615 0.5544 0.5544 0.5321 0.5321 0.5174 0.4191 0.2592 0.2592
0.1678
## Call:
## VAR(y = dat_VAR, p = 4, type = "const", exogen = dat_EXO)
##
##
## Estimation results for equation LREERS:
## =======================================
## LREERS = LREERS.l1 + RIRR.l1 + LRGDPPCR.l1 + LPR2COMM5.l1 + OPENY.l1 + FBYA.l1 + NFAOFPY.l
1 + LREERS.l2 + RIRR.l2 + LRGDPPCR.l2 + LPR2COMM5.l2 + OPENY.l2 + FBYA.l2 + NFAOFPY.l2 + LREE
RS.l3 + RIRR.l3 + LRGDPPCR.l3 + LPR2COMM5.l3 + OPENY.l3 + FBYA.l3 + NFAOFPY.l3 + LREERS.l4 +
RIRR.l4 + LRGDPPCR.l4 + LPR2COMM5.l4 + OPENY.l4 + FBYA.l4 + NFAOFPY.l4 + const + SDUMC1 + SDU
MC2 + SDUMC3 + DUMRER1 + DUMRER2 + DUMFBYA + DUMNFAOFPY
##
##                Estimate Std. Error t value Pr(>|t|)
## LREERS.l1     1.176e+00  1.264e-01   9.299 9.03e-15
## RIRR.l1      -4.128e-03  3.814e-03  -1.082 0.282084
## LRGDPPCR.l1   1.847e-01  3.698e-01   0.499 0.618685
## LPR2COMM5.l1  6.426e-02  5.352e-02   1.201 0.233096
## OPENY.l1      4.056e-05  2.302e-03   0.018 0.985983
## FBYA.l1       1.554e-03  1.621e-03   0.959 0.340272
## NFAOFPY.l1   -6.449e-03  2.569e-03  -2.510 0.013863
## LREERS.l2    -1.708e-01  1.711e-01  -0.998 0.320942
## RIRR.l2      -1.640e-03  5.157e-03  -0.318 0.751196
## LRGDPPCR.l2  -3.499e-02  4.905e-01  -0.071 0.943291
## LPR2COMM5.l2  4.321e-02  6.822e-02   0.633 0.528119
## OPENY.l2     -2.781e-03  2.848e-03  -0.976 0.331505
## FBYA.l2      -1.992e-03  1.629e-03  -1.223 0.224695
## NFAOFPY.l2    6.500e-03  3.924e-03   1.657 0.101131
## LREERS.l3     1.568e-01  1.623e-01   0.966 0.336660
## RIRR.l3       4.297e-03  5.076e-03   0.847 0.399454
## LRGDPPCR.l3  -4.541e-01  4.613e-01  -0.984 0.327550
## LPR2COMM5.l3  1.130e-02  6.767e-02   0.167 0.867755
## OPENY.l3      6.635e-04  2.895e-03   0.229 0.819265
## FBYA.l3      -8.705e-04  1.590e-03  -0.547 0.585417
## NFAOFPY.l3    1.828e-04  4.096e-03   0.045 0.964508
## LREERS.l4    -1.940e-01  1.190e-01  -1.631 0.106520
## RIRR.l4       1.885e-03  3.673e-03   0.513 0.609062
## LRGDPPCR.l4   3.453e-01  3.428e-01   1.007 0.316536
## LPR2COMM5.l4 -1.064e-01  5.851e-02  -1.818 0.072496
## OPENY.l4      2.191e-03  2.326e-03   0.942 0.348840
## FBYA.l4       6.546e-05  1.636e-03   0.040 0.968166
## NFAOFPY.l4   -1.328e-03  2.635e-03  -0.504 0.615621
## const         1.092e-01  5.155e-01   0.212 0.832698
## SDUMC1        2.406e-02  1.612e-02   1.493 0.139035
## SDUMC2        1.190e-02  1.561e-02   0.763 0.447633
```

```
## SDUMC3        6.306e-04  1.557e-02   0.041 0.967781
## DUMRER1       1.713e-01  5.202e-02   3.293 0.001423
## DUMRER2       1.868e-01  5.474e-02   3.413 0.000969
## DUMFBYA      -1.440e-03  4.914e-02  -0.029 0.976687
## DUMNFAOFPY   -3.642e-02  4.912e-02  -0.741 0.460412
##
## LREERS.l1    ***
## RIRR.l1
## LRGDPPCR.l1
## LPR2COMM5.l1
## OPENY.l1
## FBYA.l1
## NFAOFPY.l1   *
## LREERS.l2
## RIRR.l2
## LRGDPPCR.l2
## LPR2COMM5.l2
## OPENY.l2
## FBYA.l2
## NFAOFPY.l2
## LREERS.l3
## RIRR.l3
## LRGDPPCR.l3
## LPR2COMM5.l3
## OPENY.l3
## FBYA.l3
## NFAOFPY.l3
## LREERS.l4
## RIRR.l4
## LRGDPPCR.l4
## LPR2COMM5.l4 .
## OPENY.l4
## FBYA.l4
## NFAOFPY.l4
## const
## SDUMC1
## SDUMC2
## SDUMC3
## DUMRER1      **
## DUMRER2      ***
## DUMFBYA
## DUMNFAOFPY
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.04427 on 89 degrees of freedom
## Multiple R-Squared: 0.9183,  Adjusted R-squared: 0.8861
## F-statistic: 28.56 on 35 and 89 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation RIRR:
## ====================================
## RIRR = LREERS.l1 + RIRR.l1 + LRGDPPCR.l1 + LPR2COMM5.l1 + OPENY.l1 + FBYA.l1 + NFAOFPY.l1
+ LREERS.l2 + RIRR.l2 + LRGDPPCR.l2 + LPR2COMM5.l2 + OPENY.l2 + FBYA.l2 + NFAOFPY.l2 + LREER
```

```
S.l3 + RIRR.l3 + LRGDPPCR.l3 + LPR2COMM5.l3 + OPENY.l3 + FBYA.l3 + NFAOFPY.l3 + LREERS.l4 + R
IRR.l4 + LRGDPPCR.l4 + LPR2COMM5.l4 + OPENY.l4 + FBYA.l4 + NFAOFPY.l4 + const + SDUMC1 + SDUM
C2 + SDUMC3 + DUMRER1 + DUMRER2 + DUMFBYA + DUMNFAOFPY
##
##                Estimate Std. Error t value Pr(>|t|)
## LREERS.l1      3.955793   3.631479   1.089   0.2790
## RIRR.l1        0.925187   0.109553   8.445 5.27e-13
## LRGDPPCR.l1   -1.844009  10.621952  -0.174   0.8626
## LPR2COMM5.l1  -0.304953   1.537184  -0.198   0.8432
## OPENY.l1      -0.111122   0.066121  -1.681   0.0963
## FBYA.l1       -0.087243   0.046558  -1.874   0.0642
## NFAOFPY.l1    -0.009058   0.073779  -0.123   0.9026
## LREERS.l2     -1.619794   4.915302  -0.330   0.7425
## RIRR.l2       -0.106833   0.148113  -0.721   0.4726
## LRGDPPCR.l2    5.243873  14.086570   0.372   0.7106
## LPR2COMM5.l2   0.617517   1.959364   0.315   0.7534
## OPENY.l2       0.063488   0.081789   0.776   0.4397
## FBYA.l2        0.014495   0.046797   0.310   0.7575
## NFAOFPY.l2    -0.123726   0.112698  -1.098   0.2752
## LREERS.l3      8.003958   4.661839   1.717   0.0895
## RIRR.l3        0.170119   0.145778   1.167   0.2463
## LRGDPPCR.l3   -7.749777  13.248856  -0.585   0.5601
## LPR2COMM5.l3   0.168857   1.943411   0.087   0.9310
## OPENY.l3       0.044747   0.083154   0.538   0.5918
## FBYA.l3        0.060098   0.045663   1.316   0.1915
## NFAOFPY.l3     0.076829   0.117640   0.653   0.5154
## LREERS.l4     -7.479200   3.417346  -2.189   0.0312
## RIRR.l4       -0.180240   0.105487  -1.709   0.0910
## LRGDPPCR.l4    0.680962   9.844281   0.069   0.9450
## LPR2COMM5.l4  -2.724991   1.680554  -1.621   0.1085
## OPENY.l4       0.058747   0.066808   0.879   0.3816
## FBYA.l4       -0.006385   0.046975  -0.136   0.8922
## NFAOFPY.l4     0.047263   0.075685   0.624   0.5339
## const        -14.852632  14.805000  -1.003   0.3185
## SDUMC1         0.043306   0.462935   0.094   0.9257
## SDUMC2         0.019189   0.448234   0.043   0.9659
## SDUMC3        -0.456341   0.447166  -1.021   0.3102
## DUMRER1       -1.855256   1.494023  -1.242   0.2176
## DUMRER2       -0.795473   1.572291  -0.506   0.6142
## DUMFBYA       -0.986094   1.411330  -0.699   0.4866
## DUMNFAOFPY     1.168305   1.410723   0.828   0.4098
##
## LREERS.l1
## RIRR.l1      ***
## LRGDPPCR.l1
## LPR2COMM5.l1
## OPENY.l1     .
## FBYA.l1      .
## NFAOFPY.l1
## LREERS.l2
## RIRR.l2
## LRGDPPCR.l2
## LPR2COMM5.l2
## OPENY.l2
## FBYA.l2
## NFAOFPY.l2
```

```
## LREERS.l3    .
## RIRR.l3
## LRGDPPCR.l3
## LPR2COMM5.l3
## OPENY.l3
## FBYA.l3
## NFAOFPY.l3
## LREERS.l4    *
## RIRR.l4      .
## LRGDPPCR.l4
## LPR2COMM5.l4
## OPENY.l4
## FBYA.l4
## NFAOFPY.l4
## const
## SDUMC1
## SDUMC2
## SDUMC3
## DUMRER1
## DUMRER2
## DUMFBYA
## DUMNFAOFPY
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.271 on 89 degrees of freedom
## Multiple R-Squared: 0.9257,  Adjusted R-squared: 0.8965
## F-statistic: 31.69 on 35 and 89 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation LRGDPPCR:
## ========================================
## LRGDPPCR = LREERS.l1 + RIRR.l1 + LRGDPPCR.l1 + LPR2COMM5.l1 + OPENY.l1 + FBYA.l1 + NFAOFP
Y.l1 + LREERS.l2 + RIRR.l2 + LRGDPPCR.l2 + LPR2COMM5.l2 + OPENY.l2 + FBYA.l2 + NFAOFPY.l2 + L
REERS.l3 + RIRR.l3 + LRGDPPCR.l3 + LPR2COMM5.l3 + OPENY.l3 + FBYA.l3 + NFAOFPY.l3 + LREERS.l4
 + RIRR.l4 + LRGDPPCR.l4 + LPR2COMM5.l4 + OPENY.l4 + FBYA.l4 + NFAOFPY.l4 + const + SDUMC1 + S
DUMC2 + SDUMC3 + DUMRER1 + DUMRER2 + DUMFBYA + DUMNFAOFPY
##
##                Estimate Std. Error t value Pr(>|t|)
## LREERS.l1     3.610e-02  3.758e-02   0.961  0.33933
## RIRR.l1       4.033e-04  1.134e-03   0.356  0.72288
## LRGDPPCR.l1   9.448e-01  1.099e-01   8.595 2.59e-13
## LPR2COMM5.l1  2.170e-02  1.591e-02   1.364  0.17597
## OPENY.l1     -1.739e-04  6.843e-04  -0.254  0.79994
## FBYA.l1      -7.606e-04  4.818e-04  -1.579  0.11800
## NFAOFPY.l1   -2.004e-03  7.636e-04  -2.625  0.01021
## LREERS.l2    -4.406e-02  5.087e-02  -0.866  0.38870
## RIRR.l2       3.971e-04  1.533e-03   0.259  0.79620
## LRGDPPCR.l2   1.861e-03  1.458e-01   0.013  0.98984
## LPR2COMM5.l2 -3.125e-03  2.028e-02  -0.154  0.87789
## OPENY.l2     -6.375e-04  8.465e-04  -0.753  0.45336
## FBYA.l2       3.006e-04  4.843e-04   0.621  0.53644
## NFAOFPY.l2    2.550e-03  1.166e-03   2.186  0.03144
## LREERS.l3     6.394e-02  4.825e-02   1.325  0.18848
```

```
## RIRR.l3       3.166e-04  1.509e-03   0.210  0.83426
## LRGDPPCR.l3  -2.033e-02  1.371e-01  -0.148  0.88247
## LPR2COMM5.l3 -1.819e-03  2.011e-02  -0.090  0.92815
## OPENY.l3      2.431e-03  8.606e-04   2.825  0.00583
## FBYA.l3       7.445e-04  4.726e-04   1.575  0.11870
## NFAOFPY.l3   -2.219e-04  1.218e-03  -0.182  0.85580
## LREERS.l4    -1.037e-01  3.537e-02  -2.932  0.00428
## RIRR.l4      -7.542e-04  1.092e-03  -0.691  0.49148
## LRGDPPCR.l4   4.584e-02  1.019e-01   0.450  0.65388
## LPR2COMM5.l4 -6.126e-03  1.739e-02  -0.352  0.72550
## OPENY.l4     -1.872e-03  6.914e-04  -2.707  0.00814
## FBYA.l4      -7.043e-05  4.862e-04  -0.145  0.88513
## NFAOFPY.l4    4.135e-04  7.833e-04   0.528  0.59888
## const         2.414e-01  1.532e-01   1.576  0.11869
## SDUMC1        1.338e-02  4.791e-03   2.792  0.00641
## SDUMC2        2.784e-03  4.639e-03   0.600  0.54993
## SDUMC3       -7.807e-04  4.628e-03  -0.169  0.86642
## DUMRER1      -2.377e-02  1.546e-02  -1.537  0.12774
## DUMRER2      -1.680e-02  1.627e-02  -1.033  0.30463
## DUMFBYA      -2.245e-02  1.461e-02  -1.537  0.12784
## DUMNFAOFPY   -1.139e-03  1.460e-02  -0.078  0.93799
##
## LREERS.l1
## RIRR.l1
## LRGDPPCR.l1  ***
## LPR2COMM5.l1
## OPENY.l1
## FBYA.l1
## NFAOFPY.l1   *
## LREERS.l2
## RIRR.l2
## LRGDPPCR.l2
## LPR2COMM5.l2
## OPENY.l2
## FBYA.l2
## NFAOFPY.l2   *
## LREERS.l3
## RIRR.l3
## LRGDPPCR.l3
## LPR2COMM5.l3
## OPENY.l3     **
## FBYA.l3
## NFAOFPY.l3
## LREERS.l4    **
## RIRR.l4
## LRGDPPCR.l4
## LPR2COMM5.l4
## OPENY.l4     **
## FBYA.l4
## NFAOFPY.l4
## const
## SDUMC1       **
## SDUMC2
## SDUMC3
## DUMRER1
## DUMRER2
```

```
## DUMFBYA
## DUMNFAOFPY
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.01316 on 89 degrees of freedom
## Multiple R-Squared: 0.9976,  Adjusted R-squared: 0.9966
## F-statistic:  1055 on 35 and 89 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation LPR2COMM5:
## ==========================================
## LPR2COMM5 = LREERS.l1 + RIRR.l1 + LRGDPPCR.l1 + LPR2COMM5.l1 + OPENY.l1 + FBYA.l1 + NFAOFP
Y.l1 + LREERS.l2 + RIRR.l2 + LRGDPPCR.l2 + LPR2COMM5.l2 + OPENY.l2 + FBYA.l2 + NFAOFPY.l2 + L
REERS.l3 + RIRR.l3 + LRGDPPCR.l3 + LPR2COMM5.l3 + OPENY.l3 + FBYA.l3 + NFAOFPY.l3 + LREERS.l4
 + RIRR.l4 + LRGDPPCR.l4 + LPR2COMM5.l4 + OPENY.l4 + FBYA.l4 + NFAOFPY.l4 + const + SDUMC1 + S
DUMC2 + SDUMC3 + DUMRER1 + DUMRER2 + DUMFBYA + DUMNFAOFPY
##
##                Estimate Std. Error t value Pr(>|t|)
## LREERS.l1    -0.1549461  0.2448711  -0.633 0.528509
## RIRR.l1      -0.0024334  0.0073872  -0.329 0.742619
## LRGDPPCR.l1   0.0998745  0.7162395   0.139 0.889415
## LPR2COMM5.l1  0.9263565  0.1036525   8.937 5.07e-14
## OPENY.l1      0.0019200  0.0044585   0.431 0.667779
## FBYA.l1      -0.0011213  0.0031394  -0.357 0.721810
## NFAOFPY.l1    0.0032745  0.0049749   0.658 0.512108
## LREERS.l2     0.3753613  0.3314394   1.133 0.260458
## RIRR.l2      -0.0031351  0.0099872  -0.314 0.754322
## LRGDPPCR.l2  -1.1141743  0.9498591  -1.173 0.243930
## LPR2COMM5.l2 -0.1433147  0.1321201  -1.085 0.280971
## OPENY.l2      0.0014569  0.0055151   0.264 0.792260
## FBYA.l2      -0.0009612  0.0031555  -0.305 0.761365
## NFAOFPY.l2   -0.0068419  0.0075993  -0.900 0.370373
## LREERS.l3    -0.3280467  0.3143484  -1.044 0.299508
## RIRR.l3       0.0059782  0.0098299   0.608 0.544623
## LRGDPPCR.l3  -0.2017872  0.8933720  -0.226 0.821819
## LPR2COMM5.l3  0.4676671  0.1310445   3.569 0.000581
## OPENY.l3     -0.0013780  0.0056071  -0.246 0.806434
## FBYA.l3      -0.0031603  0.0030791  -1.026 0.307492
## NFAOFPY.l3    0.0040689  0.0079325   0.513 0.609261
## LREERS.l4     0.0437857  0.2304320   0.190 0.849730
## RIRR.l4      -0.0088832  0.0071130  -1.249 0.214988
## LRGDPPCR.l4   1.0927100  0.6638010   1.646 0.103262
## LPR2COMM5.l4 -0.3451541  0.1133200  -3.046 0.003053
## OPENY.l4     -0.0004697  0.0045049  -0.104 0.917194
## FBYA.l4       0.0025595  0.0031675   0.808 0.421213
## NFAOFPY.l4    0.0019270  0.0051035   0.378 0.706639
## const         0.2573820  0.9983029   0.258 0.797141
## SDUMC1        0.0106416  0.0312158   0.341 0.733980
## SDUMC2        0.0157523  0.0302244   0.521 0.603537
## SDUMC3        0.0092950  0.0301525   0.308 0.758601
## DUMRER1       0.0036826  0.1007422   0.037 0.970922
## DUMRER2      -0.0274394  0.1060198  -0.259 0.796376
## DUMFBYA       0.0209035  0.0951662   0.220 0.826645
```

```
## DUMNFAOFPY   -0.0475721  0.0951252  -0.500 0.618239
##
## LREERS.l1
## RIRR.l1
## LRGDPPCR.l1
## LPR2COMM5.l1 ***
## OPENY.l1
## FBYA.l1
## NFAOFPY.l1
## LREERS.l2
## RIRR.l2
## LRGDPPCR.l2
## LPR2COMM5.l2
## OPENY.l2
## FBYA.l2
## NFAOFPY.l2
## LREERS.l3
## RIRR.l3
## LRGDPPCR.l3
## LPR2COMM5.l3 ***
## OPENY.l3
## FBYA.l3
## NFAOFPY.l3
## LREERS.l4
## RIRR.l4
## LRGDPPCR.l4
## LPR2COMM5.l4 **
## OPENY.l4
## FBYA.l4
## NFAOFPY.l4
## const
## SDUMC1
## SDUMC2
## SDUMC3
## DUMRER1
## DUMRER2
## DUMFBYA
## DUMNFAOFPY
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.08573 on 89 degrees of freedom
## Multiple R-Squared: 0.9504,  Adjusted R-squared: 0.9309
## F-statistic:  48.7 on 35 and 89 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation OPENY:
## ======================================
## OPENY = LREERS.l1 + RIRR.l1 + LRGDPPCR.l1 + LPR2COMM5.l1 + OPENY.l1 + FBYA.l1 + NFAOFPY.l1
+ LREERS.l2 + RIRR.l2 + LRGDPPCR.l2 + LPR2COMM5.l2 + OPENY.l2 + FBYA.l2 + NFAOFPY.l2 + LREER
S.l3 + RIRR.l3 + LRGDPPCR.l3 + LPR2COMM5.l3 + OPENY.l3 + FBYA.l3 + NFAOFPY.l3 + LREERS.l4 + R
IRR.l4 + LRGDPPCR.l4 + LPR2COMM5.l4 + OPENY.l4 + FBYA.l4 + NFAOFPY.l4 + const + SDUMC1 + SDUM
C2 + SDUMC3 + DUMRER1 + DUMRER2 + DUMFBYA + DUMNFAOFPY
##
```

```
##                Estimate Std. Error t value Pr(>|t|)
## LREERS.l1    -12.945436   6.654095  -1.945   0.0549
## RIRR.l1        0.522581   0.200739   2.603   0.0108
## LRGDPPCR.l1   13.071814  19.463001   0.672   0.5036
## LPR2COMM5.l1   3.659613   2.816641   1.299   0.1972
## OPENY.l1       0.631093   0.121155   5.209 1.21e-06
## FBYA.l1       -0.032751   0.085310  -0.384   0.7020
## NFAOFPY.l1     0.153049   0.135188   1.132   0.2606
## LREERS.l2      3.558809   9.006492   0.395   0.6937
## RIRR.l2       -0.344859   0.271392  -1.271   0.2071
## LRGDPPCR.l2   -8.223115  25.811350  -0.319   0.7508
## LPR2COMM5.l2  -1.948112   3.590216  -0.543   0.5887
## OPENY.l2       0.219454   0.149866   1.464   0.1466
## FBYA.l2       -0.076945   0.085747  -0.897   0.3720
## NFAOFPY.l2    -0.070022   0.206501  -0.339   0.7353
## LREERS.l3     -9.042131   8.542063  -1.059   0.2927
## RIRR.l3        0.081247   0.267115   0.304   0.7617
## LRGDPPCR.l3   14.662227  24.276377   0.604   0.5474
## LPR2COMM5.l3  -4.217900   3.560986  -1.184   0.2394
## OPENY.l3      -0.027948   0.152366  -0.183   0.8549
## FBYA.l3        0.022747   0.083670   0.272   0.7864
## NFAOFPY.l3     0.017058   0.215556   0.079   0.9371
## LREERS.l4      5.750914   6.261731   0.918   0.3609
## RIRR.l4       -0.068862   0.193288  -0.356   0.7225
## LRGDPPCR.l4  -19.389526  18.038045  -1.075   0.2853
## LPR2COMM5.l4   5.945296   3.079342   1.931   0.0567
## OPENY.l4      -0.003776   0.122414  -0.031   0.9755
## FBYA.l4       -0.016931   0.086074  -0.197   0.8445
## NFAOFPY.l4     0.029007   0.138681   0.209   0.8348
## const         68.062433  27.127756   2.509   0.0139
## SDUMC1        -0.382364   0.848253  -0.451   0.6533
## SDUMC2        -0.300689   0.821315  -0.366   0.7152
## SDUMC3         0.625164   0.819360   0.763   0.4475
## DUMRER1       -4.553020   2.737554  -1.663   0.0998
## DUMRER2       -3.462473   2.880968  -1.202   0.2326
## DUMFBYA        0.176502   2.586033   0.068   0.9457
## DUMNFAOFPY     3.426125   2.584921   1.325   0.1884
##
## LREERS.l1    .
## RIRR.l1      *
## LRGDPPCR.l1
## LPR2COMM5.l1
## OPENY.l1     ***
## FBYA.l1
## NFAOFPY.l1
## LREERS.l2
## RIRR.l2
## LRGDPPCR.l2
## LPR2COMM5.l2
## OPENY.l2
## FBYA.l2
## NFAOFPY.l2
## LREERS.l3
## RIRR.l3
## LRGDPPCR.l3
## LPR2COMM5.l3
```

```
## OPENY.l3
## FBYA.l3
## NFAOFPY.l3
## LREERS.l4
## RIRR.l4
## LRGDPPCR.l4
## LPR2COMM5.l4 .
## OPENY.l4
## FBYA.l4
## NFAOFPY.l4
## const        *
## SDUMC1
## SDUMC2
## SDUMC3
## DUMRER1      .
## DUMRER2
## DUMFBYA
## DUMNFAOFPY
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 2.33 on 89 degrees of freedom
## Multiple R-Squared: 0.9117,  Adjusted R-squared: 0.8769
## F-statistic: 26.24 on 35 and 89 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation FBYA:
## =====================================
## FBYA = LREERS.l1 + RIRR.l1 + LRGDPPCR.l1 + LPR2COMM5.l1 + OPENY.l1 + FBYA.l1 + NFAOFPY.l1
+ LREERS.l2 + RIRR.l2 + LRGDPPCR.l2 + LPR2COMM5.l2 + OPENY.l2 + FBYA.l2 + NFAOFPY.l2 + LREER
S.l3 + RIRR.l3 + LRGDPPCR.l3 + LPR2COMM5.l3 + OPENY.l3 + FBYA.l3 + NFAOFPY.l3 + LREERS.l4 + R
IRR.l4 + LRGDPPCR.l4 + LPR2COMM5.l4 + OPENY.l4 + FBYA.l4 + NFAOFPY.l4 + const + SDUMC1 + SDUM
C2 + SDUMC3 + DUMRER1 + DUMRER2 + DUMFBYA + DUMNFAOFPY
##
##                Estimate Std. Error t value Pr(>|t|)
## LREERS.l1    -13.696157   6.629877  -2.066  0.04175
## RIRR.l1        0.008582   0.200008   0.043  0.96587
## LRGDPPCR.l1  -18.724738  19.392163  -0.966  0.33687
## LPR2COMM5.l1   6.583088   2.806389   2.346  0.02121
## OPENY.l1      -0.117302   0.120714  -0.972  0.33382
## FBYA.l1       -0.003884   0.084999  -0.046  0.96365
## NFAOFPY.l1     0.096540   0.134696   0.717  0.47542
## LREERS.l2     13.126561   8.973712   1.463  0.14705
## RIRR.l2        0.340437   0.270404   1.259  0.21133
## LRGDPPCR.l2   28.670511  25.717408   1.115  0.26793
## LPR2COMM5.l2   5.373374   3.577149   1.502  0.13660
## OPENY.l2      -0.054553   0.149320  -0.365  0.71573
## FBYA.l2        0.127188   0.085435   1.489  0.14010
## NFAOFPY.l2    -0.306162   0.205750  -1.488  0.14028
## LREERS.l3     -6.877112   8.510973  -0.808  0.42123
## RIRR.l3       -0.057417   0.266143  -0.216  0.82969
## LRGDPPCR.l3  -25.114091  24.188021  -1.038  0.30195
## LPR2COMM5.l3  -5.547584   3.548025  -1.564  0.12147
## OPENY.l3       0.019524   0.151811   0.129  0.89796
```

```
## FBYA.l3       -0.143496   0.083366  -1.721  0.08867
## NFAOFPY.l3     0.240730   0.214772   1.121  0.26536
## LREERS.l4     -2.853336   6.238941  -0.457  0.64854
## RIRR.l4        0.130850   0.192585   0.679  0.49862
## LRGDPPCR.l4   17.008004  17.972394   0.946  0.34654
## LPR2COMM5.l4  -1.150794   3.068134  -0.375  0.70849
## OPENY.l4       0.134242   0.121969   1.101  0.27403
## FBYA.l4        0.361738   0.085761   4.218 5.91e-05
## NFAOFPY.l4    -0.009746   0.138176  -0.071  0.94393
## const         45.021458  27.029023   1.666  0.09930
## SDUMC1         0.185993   0.845166   0.220  0.82632
## SDUMC2        -2.161154   0.818326  -2.641  0.00976
## SDUMC3        -0.100161   0.816378  -0.123  0.90263
## DUMRER1        0.551087   2.727591   0.202  0.84035
## DUMRER2       -2.270303   2.870482  -0.791  0.43110
## DUMFBYA      -11.481528   2.576621  -4.456 2.42e-05
## DUMNFAOFPY     0.936860   2.575513   0.364  0.71690
##
## LREERS.l1    *
## RIRR.l1
## LRGDPPCR.l1
## LPR2COMM5.l1 *
## OPENY.l1
## FBYA.l1
## NFAOFPY.l1
## LREERS.l2
## RIRR.l2
## LRGDPPCR.l2
## LPR2COMM5.l2
## OPENY.l2
## FBYA.l2
## NFAOFPY.l2
## LREERS.l3
## RIRR.l3
## LRGDPPCR.l3
## LPR2COMM5.l3
## OPENY.l3
## FBYA.l3      .
## NFAOFPY.l3
## LREERS.l4
## RIRR.l4
## LRGDPPCR.l4
## LPR2COMM5.l4
## OPENY.l4
## FBYA.l4      ***
## NFAOFPY.l4
## const        .
## SDUMC1
## SDUMC2       **
## SDUMC3
## DUMRER1
## DUMRER2
## DUMFBYA      ***
## DUMNFAOFPY
## ---
## Signif. codes:
```

```
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 2.321 on 89 degrees of freedom
## Multiple R-Squared: 0.7171,  Adjusted R-squared: 0.6059
## F-statistic: 6.446 on 35 and 89 DF,  p-value: 6.356e-13
##
##
## Estimation results for equation NFAOFPY:
## ========================================
## NFAOFPY = LREERS.l1 + RIRR.l1 + LRGDPPCR.l1 + LPR2COMM5.l1 + OPENY.l1 + FBYA.l1 + NFAOFPY.
l1 + LREERS.l2 + RIRR.l2 + LRGDPPCR.l2 + LPR2COMM5.l2 + OPENY.l2 + FBYA.l2 + NFAOFPY.l2 + LRE
ERS.l3 + RIRR.l3 + LRGDPPCR.l3 + LPR2COMM5.l3 + OPENY.l3 + FBYA.l3 + NFAOFPY.l3 + LREERS.l4 +
RIRR.l4 + LRGDPPCR.l4 + LPR2COMM5.l4 + OPENY.l4 + FBYA.l4 + NFAOFPY.l4 + const + SDUMC1 + SDU
MC2 + SDUMC3 + DUMRER1 + DUMRER2 + DUMFBYA + DUMNFAOFPY
##
##                Estimate Std. Error t value Pr(>|t|)
## LREERS.l1      3.004834   5.294277   0.568   0.5718
## RIRR.l1       -0.018344   0.159716  -0.115   0.9088
## LRGDPPCR.l1   -7.579836  15.485578  -0.489   0.6257
## LPR2COMM5.l1   1.011844   2.241037   0.452   0.6527
## OPENY.l1      -0.054923   0.096396  -0.570   0.5703
## FBYA.l1        0.160912   0.067876   2.371   0.0199
## NFAOFPY.l1     1.062818   0.107562   9.881 5.64e-16
## LREERS.l2      0.999385   7.165942   0.139   0.8894
## RIRR.l2       -0.125167   0.215931  -0.580   0.5636
## LRGDPPCR.l2    8.570097  20.536591   0.417   0.6775
## LPR2COMM5.l2   0.859806   2.856526   0.301   0.7641
## OPENY.l2       0.125960   0.119240   1.056   0.2937
## FBYA.l2        0.056537   0.068224   0.829   0.4095
## NFAOFPY.l2    -0.144814   0.164301  -0.881   0.3805
## LREERS.l3     -0.472946   6.796423  -0.070   0.9447
## RIRR.l3       -0.216762   0.212528  -1.020   0.3105
## LRGDPPCR.l3  -11.628321  19.315301  -0.602   0.5487
## LPR2COMM5.l3  -3.190095   2.833269  -1.126   0.2632
## OPENY.l3      -0.022107   0.121228  -0.182   0.8557
## FBYA.l3       -0.055776   0.066572  -0.838   0.4044
## NFAOFPY.l3     0.069826   0.171505   0.407   0.6849
## LREERS.l4     -0.409634   4.982095  -0.082   0.9347
## RIRR.l4        0.284280   0.153788   1.849   0.0678
## LRGDPPCR.l4    9.678967  14.351823   0.674   0.5018
## LPR2COMM5.l4  -1.405772   2.450053  -0.574   0.5676
## OPENY.l4       0.055477   0.097398   0.570   0.5704
## FBYA.l4        0.008059   0.068484   0.118   0.9066
## NFAOFPY.l4    -0.061807   0.110341  -0.560   0.5768
## const        -19.052132  21.583978  -0.883   0.3798
## SDUMC1         0.075773   0.674906   0.112   0.9109
## SDUMC2        -0.088599   0.653473  -0.136   0.8925
## SDUMC3        -0.134567   0.651917  -0.206   0.8369
## DUMRER1        5.565573   2.178113   2.555   0.0123
## DUMRER2        2.018009   2.292219   0.880   0.3810
## DUMFBYA       -0.715007   2.057556  -0.348   0.7290
## DUMNFAOFPY    -9.539638   2.056671  -4.638 1.20e-05
##
## LREERS.l1
## RIRR.l1
```

```
## LRGDPPCR.l1
## LPR2COMM5.l1
## OPENY.l1
## FBYA.l1      *
## NFAOFPY.l1   ***
## LREERS.l2
## RIRR.l2
## LRGDPPCR.l2
## LPR2COMM5.l2
## OPENY.l2
## FBYA.l2
## NFAOFPY.l2
## LREERS.l3
## RIRR.l3
## LRGDPPCR.l3
## LPR2COMM5.l3
## OPENY.l3
## FBYA.l3
## NFAOFPY.l3
## LREERS.l4
## RIRR.l4      .
## LRGDPPCR.l4
## LPR2COMM5.l4
## OPENY.l4
## FBYA.l4
## NFAOFPY.l4
## const
## SDUMC1
## SDUMC2
## SDUMC3
## DUMRER1      *
## DUMRER2
## DUMFBYA
## DUMNFAOFPY   ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1.854 on 89 degrees of freedom
## Multiple R-Squared: 0.9808,  Adjusted R-squared: 0.9733
## F-statistic:   130 on 35 and 89 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##               LREERS      RIRR   LRGDPPCR  LPR2COMM5
## LREERS     1.960e-03 -0.006745 -2.127e-05  0.0000311
## RIRR      -6.745e-03  1.616455  6.173e-03 -0.0010310
## LRGDPPCR  -2.127e-05  0.006173  1.731e-04  0.0002231
## LPR2COMM5  3.110e-05 -0.001031  2.231e-04  0.0073497
## OPENY     -4.554e-02  0.525501  5.614e-03  0.0342141
## FBYA      -1.235e-02  0.093924  1.343e-03  0.0212078
## NFAOFPY    3.830e-02  0.041634  3.159e-03  0.0341195
##                OPENY      FBYA   NFAOFPY
## LREERS    -0.045536 -0.012347  0.038304
```

```
## RIRR       0.525501  0.093924  0.041634
## LRGDPPCR   0.005614  0.001343  0.003159
## LPR2COMM5  0.034214  0.021208  0.034119
## OPENY      5.427185 -0.229673 -0.384976
## FBYA      -0.229673  5.387751 -0.510952
## NFAOFPY   -0.384976 -0.510952  3.435657
##
## Correlation matrix of residuals:
##              LREERS      RIRR LRGDPPCR LPR2COMM5
## LREERS     1.000000 -0.119840 -0.03652  0.008195
## RIRR      -0.119840  1.000000  0.36897 -0.009459
## LRGDPPCR  -0.036522  0.368967  1.00000  0.197787
## LPR2COMM5  0.008195 -0.009459  0.19779  1.000000
## OPENY     -0.441548  0.177421  0.18315  0.171310
## FBYA      -0.120161  0.031827  0.04398  0.106575
## NFAOFPY    0.466817  0.017667  0.12954  0.214715
##              OPENY     FBYA  NFAOFPY
## LREERS    -0.44155 -0.12016  0.46682
## RIRR       0.17742  0.03183  0.01767
## LRGDPPCR   0.18315  0.04398  0.12954
## LPR2COMM5  0.17131  0.10658  0.21471
## OPENY      1.00000 -0.04247 -0.08915
## FBYA      -0.04247  1.00000 -0.11876
## NFAOFPY   -0.08915 -0.11876  1.00000
```

Now to ensure that the variables are cointegrated we need to run the Johansen test. To perform the trace test we use the command:

```
H1_trace <-
  ca.jo(
    dat_VAR,
    ecdet = c("const"),
    type = "trace",
    K = 4,
    season = NULL,
    dumvar = dat_EXO
  )
summary(H1_trace)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in cointegration
##
## Eigenvalues (lambda):
## [1] 3.247102e-01 2.010480e-01 1.678423e-01 1.370459e-01
## [5] 1.024100e-01 5.126687e-02 3.871036e-02 2.339826e-16
##
## Values of teststatistic and critical values of test:
##
##            test  10pct   5pct   1pct
## r <= 6 |   4.93   7.52   9.24  12.97
## r <= 5 |  11.51  17.85  19.96  24.60
## r <= 4 |  25.02  32.00  34.91  41.07
## r <= 3 |  43.44  49.65  53.12  60.16
## r <= 2 |  66.41  71.86  76.07  84.45
## r <= 1 |  94.47  97.18 102.14 111.01
## r = 0  | 143.54 126.58 131.70 143.09
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                 LREERS.l4     RIRR.l4  LRGDPPCR.l4
## LREERS.l4     1.000000000  1.00000000  1.000000000
## RIRR.l4      -0.037028340 -0.02217104 -0.105105691
## LRGDPPCR.l4  -0.260966973 -0.10323394 -3.569082136
## LPR2COMM5.l4 -0.502400503 -0.44785975 -1.305169783
## OPENY.l4      0.008606964  0.01699213  0.068590778
## FBYA.l4       0.033559327 -0.01146055 -0.004770043
## NFAOFPY.l4   -0.003229541 -0.01485047  0.011050699
## constant     -4.814001799 -5.48990544 -7.239143044
##              LPR2COMM5.l4     OPENY.l4      FBYA.l4
## LREERS.l4      1.00000000  1.000000000   1.00000000
## RIRR.l4        0.02037655  0.000948174   0.09470681
## LRGDPPCR.l4    0.19399009  0.361369746   2.25553392
## LPR2COMM5.l4  -0.16715504  0.041715850  -1.08072468
## OPENY.l4       0.01258497  0.008682122   0.11580183
## FBYA.l4        0.07676276 -0.009360877  -0.06554968
## NFAOFPY.l4    -0.01652846 -0.011515669  -0.03947573
## constant      -5.09320016 -5.305405605 -11.65527118
##                NFAOFPY.l4     constant
## LREERS.l4     1.000000000  1.000000000
## RIRR.l4       0.007182922  0.017250095
## LRGDPPCR.l4  -0.512302672  0.401414985
## LPR2COMM5.l4 -0.112993207 -0.230814968
## OPENY.l4      0.012319514 -0.004755849
## FBYA.l4      -0.006259900 -0.004889778
## NFAOFPY.l4    0.005434476 -0.009041014
## constant     -4.964366434 -4.555807751
##
## Weights W:
## (This is the loading matrix)
```

```
##
##                LREERS.l4     RIRR.l4  LRGDPPCR.l4
## LREERS.d     -0.04484616  0.02196090  0.005413644
## RIRR.d        0.99789298  2.86710818  0.560773554
## LRGDPPCR.d   -0.01430708 -0.03215539  0.008197594
## LPR2COMM5.d   0.16352494 -0.03133001  0.019930732
## OPENY.d      -3.45761020 -2.05340358 -0.541817011
## FBYA.d      -14.30927136  2.93507656  0.400169292
## NFAOFPY.d     0.56615157  3.04548400  0.408284003
##             LPR2COMM5.l4    OPENY.l4       FBYA.l4
## LREERS.d     0.007488375  0.04076695  0.0008130576
## RIRR.d      -0.527850319 -1.02165639 -0.2339214929
## LRGDPPCR.d   0.002735189 -0.01198082 -0.0006979307
## LPR2COMM5.d -0.110558996 -0.10117622  0.0155488960
## OPENY.d     -1.015069179 -5.84090300 -0.1809113421
## FBYA.d      -1.485970498  0.72175416  0.2171050504
## NFAOFPY.d    2.277091161 -1.97445970  0.2228442411
##                NFAOFPY.l4      constant
## LREERS.d    -0.0638506942 -6.894604e-15
## RIRR.d       0.2184101388  1.838642e-12
## LRGDPPCR.d   0.0004879649 -1.606515e-14
## LPR2COMM5.d -0.0197851350 -1.979458e-14
## OPENY.d      0.4118697424 -3.698376e-12
## FBYA.d       1.2210928819 -3.109541e-12
## NFAOFPY.d   -1.4237566328  2.360315e-12
```

Note that for $r = 0$ the test statistic of 143.54 is greater than critical values of 126.58, 131.70, and 143.09 and the 10%, 5% and 1% levels. This would imply that we are able to reject the null of no cointegration. For the case $r <= 1$ we note that the calculated test statistic of 94.47 is below the critical values of 126.58, 131.70 and 143.09. Hence we are unable to reject the null the number of cointegrating vectors is between 0 and 1.

We can also consider the results of the maximum eigenvalue statistics, which may be calculated as:

```
H1_eigen <-
  ca.jo(
    dat_VAR,
    ecdet = c("const"),
    type = "eigen",
    K = 4,
    season = NULL,
    dumvar = dat_EXO
  )
summary(H1_eigen)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant i
n cointegration
##
## Eigenvalues (lambda):
## [1] 3.247102e-01 2.010480e-01 1.678423e-01 1.370459e-01
## [5] 1.024100e-01 5.126687e-02 3.871036e-02 2.339826e-16
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 6 |  4.93  7.52  9.24 12.97
## r <= 5 |  6.58 13.75 15.67 20.20
## r <= 4 | 13.51 19.77 22.00 26.81
## r <= 3 | 18.42 25.56 28.14 33.24
## r <= 2 | 22.97 31.66 34.40 39.79
## r <= 1 | 28.06 37.45 40.30 46.82
## r = 0  | 49.08 43.25 46.45 51.91
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                 LREERS.l4     RIRR.l4  LRGDPPCR.l4
## LREERS.l4     1.000000000  1.00000000  1.000000000
## RIRR.l4      -0.037028340 -0.02217104 -0.105105691
## LRGDPPCR.l4  -0.260966973 -0.10323394 -3.569082136
## LPR2COMM5.l4 -0.502400503 -0.44785975 -1.305169783
## OPENY.l4      0.008606964  0.01699213  0.068590778
## FBYA.l4       0.033559327 -0.01146055 -0.004770043
## NFAOFPY.l4   -0.003229541 -0.01485047  0.011050699
## constant     -4.814001799 -5.48990544 -7.239143044
##              LPR2COMM5.l4     OPENY.l4      FBYA.l4
## LREERS.l4      1.00000000  1.000000000   1.00000000
## RIRR.l4        0.02037655  0.000948174   0.09470681
## LRGDPPCR.l4    0.19399009  0.361369746   2.25553392
## LPR2COMM5.l4  -0.16715504  0.041715850  -1.08072468
## OPENY.l4       0.01258497  0.008682122   0.11580183
## FBYA.l4        0.07676276 -0.009360877  -0.06554968
## NFAOFPY.l4    -0.01652846 -0.011515669  -0.03947573
## constant      -5.09320016 -5.305405605 -11.65527118
##                NFAOFPY.l4     constant
## LREERS.l4     1.000000000  1.000000000
## RIRR.l4       0.007182922  0.017250095
## LRGDPPCR.l4  -0.512302672  0.401414985
## LPR2COMM5.l4 -0.112993207 -0.230814968
## OPENY.l4      0.012319514 -0.004755849
## FBYA.l4      -0.006259900 -0.004889778
## NFAOFPY.l4    0.005434476 -0.009041014
## constant     -4.964366434 -4.555807751
##
## Weights W:
```

```
## (This is the loading matrix)
##
##                LREERS.l4     RIRR.l4  LRGDPPCR.l4
## LREERS.d     -0.04484616  0.02196090  0.005413644
## RIRR.d        0.99789298  2.86710818  0.560773554
## LRGDPPCR.d   -0.01430708 -0.03215539  0.008197594
## LPR2COMM5.d   0.16352494 -0.03133001  0.019930732
## OPENY.d      -3.45761020 -2.05340358 -0.541817011
## FBYA.d      -14.30927136  2.93507656  0.400169292
## NFAOFPY.d     0.56615157  3.04548400  0.408284003
##             LPR2COMM5.l4    OPENY.l4       FBYA.l4
## LREERS.d     0.007488375  0.04076695  0.0008130576
## RIRR.d      -0.527850319 -1.02165639 -0.2339214929
## LRGDPPCR.d   0.002735189 -0.01198082 -0.0006979307
## LPR2COMM5.d -0.110558996 -0.10117622  0.0155488960
## OPENY.d     -1.015069179 -5.84090300 -0.1809113421
## FBYA.d      -1.485970498  0.72175416  0.2171050504
## NFAOFPY.d    2.277091161 -1.97445970  0.2228442411
##                NFAOFPY.l4      constant
## LREERS.d    -0.0638506942 -6.894604e-15
## RIRR.d       0.2184101388  1.838642e-12
## LRGDPPCR.d   0.0004879649 -1.606515e-14
## LPR2COMM5.d -0.0197851350 -1.979458e-14
## OPENY.d      0.4118697424 -3.698376e-12
## FBYA.d       1.2210928819 -3.109541e-12
## NFAOFPY.d   -1.4237566328  2.360315e-12
```

In this case we are able to reject the null of there being no cointegrating vectors as the calculated test statistic is 49.08 which is greater than the crtical values of 43.25, 46.45, and 51.91.

This cointegrating vector is then used to describe the real equilibrium exchange rate in the paper. To consider the relationship between the real exchange rate and this measure of the real equilibrium exchange rate, we can construct a similar exogenity test. When doing so, we need to extract some information from the trace statistic, which relates to the speed of adjustment and long-run coefficients.

```
beta <- H1_trace@V
alpha <- H1_trace@W
```

The test for exogenity may then be used to test whether the real exchange rate responds to measure of the real equilibrium exchange rate. To consider whether the coefficient for this relationship is different from zero we construct a matrix of coefficients, where the first line (which relates to the real exchange rate - see ordering above) only contains zeros. This matrix is assigned the name `A1`.

```
A1 <- diag(7)[,-1]
print(A1)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    0    0    0
## [2,]    1    0    0    0    0    0
## [3,]    0    1    0    0    0    0
## [4,]    0    0    1    0    0    0
## [5,]    0    0    0    1    0    0
## [6,]    0    0    0    0    1    0
## [7,]    0    0    0    0    0    1
```

We can then perform the test as follows:

```
H2 <- alrtest(z = H1_trace, A = A1, r = 1)
summary(H2)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    0    0    0
## [2,]    1    0    0    0    0    0
## [3,]    0    1    0    0    0    0
## [4,]    0    0    1    0    0    0
## [5,]    0    0    0    1    0    0
## [6,]    0    0    0    0    1    0
## [7,]    0    0    0    0    0    1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.3204 0.1996 0.1650 0.1365 0.0955 0.0512 0.0000
## [8] 0.0000
##
## The value of the likelihood ratio test statistic:
## 0.79 distributed as chi square with 1 df.
## The p-value of the test statistic is: 0.37
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##                    [,1]
## RK.LREERS.l4     1.0000
## RK.RIRR.l4      -0.0370
## RK.LRGDPPCR.l4  -0.2505
## RK.LPR2COMM5.l4 -0.5017
## RK.OPENY.l4      0.0089
## RK.FBYA.l4       0.0329
## RK.NFAOFPY.l4   -0.0038
## RK.constant     -4.8422
##
## Weights W of the restricted VAR:
##
##          [,1]
## [1,]   0.0000
## [2,]   0.9911
## [3,]  -0.0149
## [4,]   0.1649
## [5,]  -4.8601
## [6,] -14.7250
## [7,]   1.5405
```

In this case the p-value is 0.37, which would suggest that we cannot reject the null that the first variable (i.e. the real exchange rate) is weakly exogenous. This would suggest that the real exchange rate does not react to movements in the equilibrium exchange rate, which is an illogical result.

3 Testing additional restrictions on the cointegrated VAR

In this example we are going to replicate the model from the book *Applied Time Series Econometrics*, which is contained in the chapter "Structural Vector Autoregressive Modeling and Impulse Responses", by Breitung, Bruggermann & Lutkepohl (2004). It makes use of data on the Canadian economy and the program is called `tut13_Canada.R`. To start off we can clear all the variables from the current environment and close all the plots.

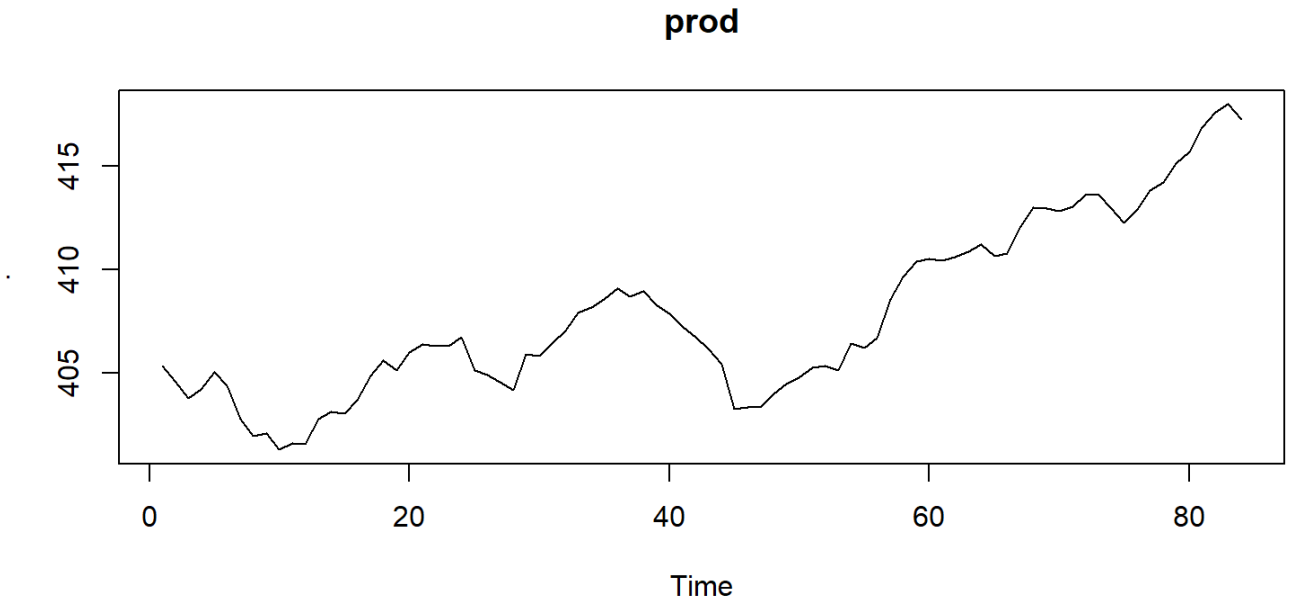
```
rm(list = ls())
graphics.off()
```

Thereafter, we will need to make use of the `vars` package, so we make use of the `library` command.

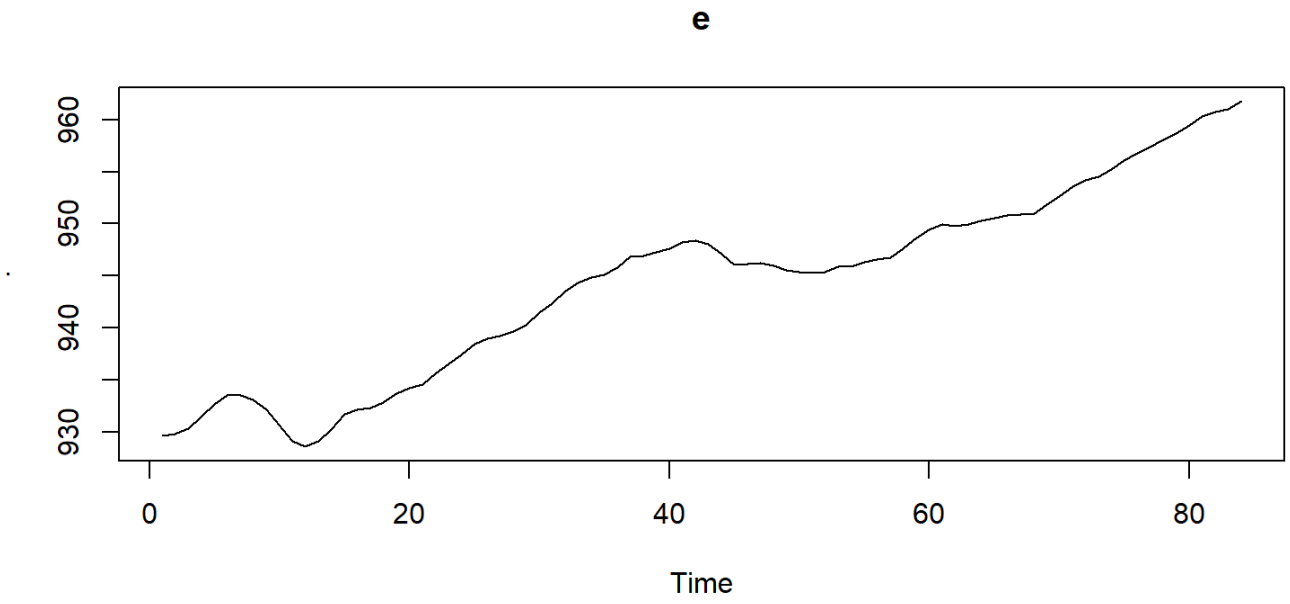
```
library(tidyverse)
library(vars)
```

As this data is contained in a `.csv` file we need to load the data, which we plot to inspect the general properties of the variables.

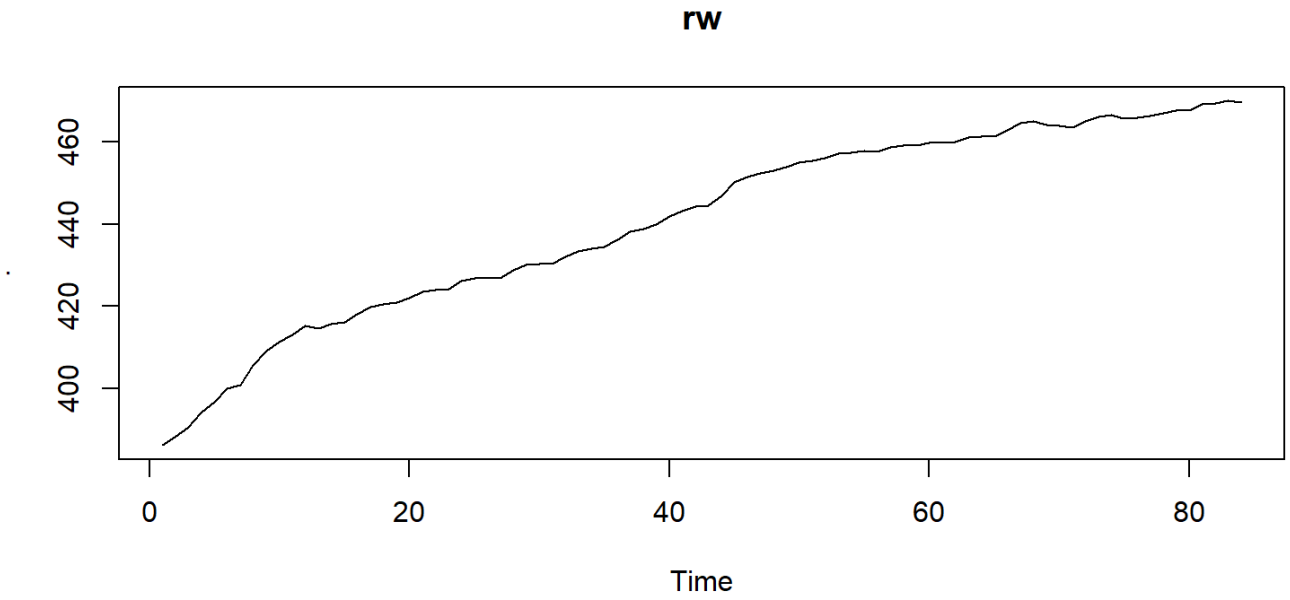
```
dat <- read_csv('Canada.csv')

dat$prod %>%
  plot.ts(., main = "prod")
```



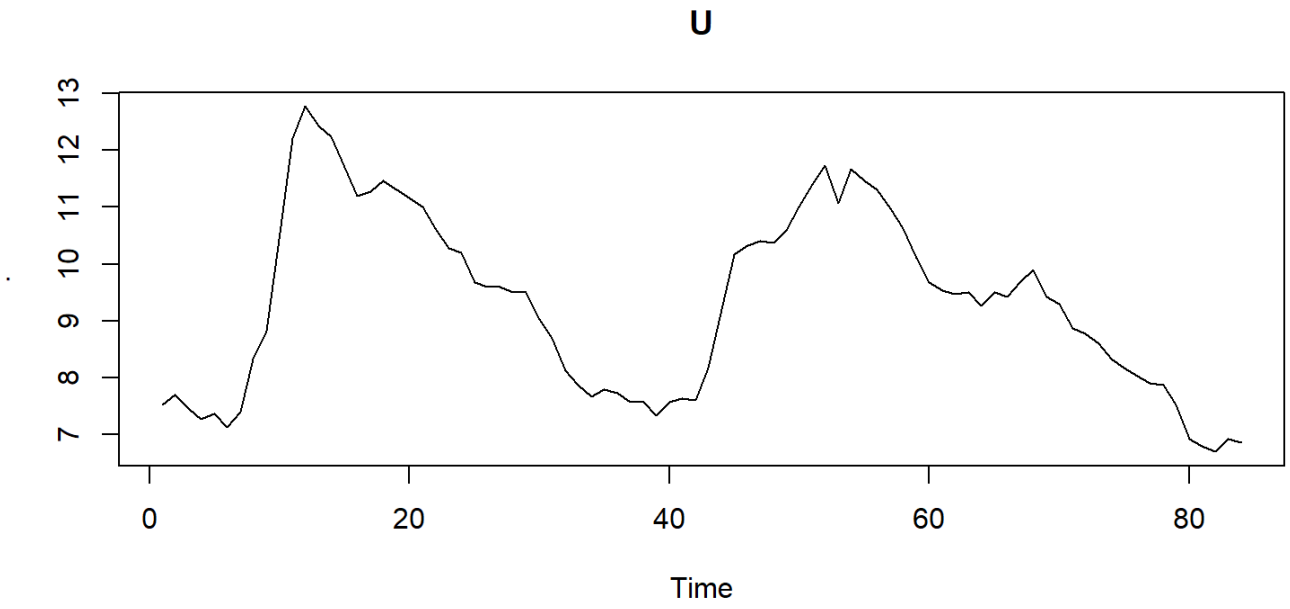

```
dat$e %>%
  plot.ts(., main = "e")
```




```
dat$rw %>%
  plot.ts(., main = "rw")
```




```
dat$U %>%
  plot.ts(., main = "U")
```



This dataset includes measures of the logarithm of employment, `e`, the real wage `rw`, productivity, `prod` and the unemployment rate, `U`. To consider the stationarity of the variables, we can perform a number of unit root tests, where we conclude the all of the variables are integrated of the order $I1$.

```
dat$prod %>%
  ur.df(., type = "trend", lags = 2) %>%
  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
## -2.19924 -0.38994  0.04294  0.41914  1.71660
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.415228  15.309403   1.987   0.0506 .
## z.lag.1     -0.075791   0.038134  -1.988   0.0505 .
## tt           0.013896   0.006422   2.164   0.0336 *
## z.diff.lag1  0.284866   0.114359   2.491   0.0149 *
## z.diff.lag2  0.080019   0.116090   0.689   0.4927
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6851 on 76 degrees of freedom
## Multiple R-squared:  0.1354, Adjusted R-squared:  0.08993
## F-statistic: 2.976 on 4 and 76 DF,  p-value: 0.02438
##
##
## Value of test-statistic is: -1.9875 2.3 2.3817
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$prod %>%
  diff() %>%
  ur.df(., type = "trend", lags = 1) %>%
  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
## -2.08255 -0.41492  0.03547  0.42292  1.77919
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.007671   0.160585  -0.048    0.962
## z.lag.1     -0.723368   0.139237  -5.195 1.63e-06 ***
## tt           0.003058   0.003456   0.885    0.379
## z.diff.lag  -0.024356   0.114799  -0.212    0.833
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6981 on 77 degrees of freedom
## Multiple R-squared:  0.3679, Adjusted R-squared:  0.3433
## F-statistic: 14.94 on 3 and 77 DF,  p-value: 9.378e-08
##
##
## Value of test-statistic is: -5.1952 9.1153 13.6696
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$e %>%
  ur.df(., type = "trend", lags = 2) %>%
  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
## -0.80266 -0.21963  0.01558  0.28686  0.73058
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.571570  18.067987   1.913   0.0595 .
## z.lag.1     -0.037139   0.019458  -1.909   0.0601 .
## tt           0.014646   0.007209   2.032   0.0457 *
## z.diff.lag1  0.928088   0.107620   8.624 7.02e-13 ***
## z.diff.lag2 -0.251322   0.112917  -2.226   0.0290 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3849 on 76 degrees of freedom
## Multiple R-squared:  0.597,  Adjusted R-squared:  0.5758
## F-statistic: 28.15 on 4 and 76 DF,  p-value: 2.378e-14
##
##
## Value of test-statistic is: -1.9087 3.804 2.0874
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$e %>%
  diff() %>%
  ur.df(., type = "trend", lags = 1) %>%
  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
## -0.9077 -0.2442 -0.0408  0.3000  0.7114
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.086813   0.092323   0.940   0.3500
## z.lag.1     -0.364701   0.080222  -4.546    2e-05 ***
## tt           0.001341   0.001870   0.717   0.4756
## z.diff.lag   0.322536   0.108388   2.976   0.0039 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3914 on 77 degrees of freedom
## Multiple R-squared:  0.2273, Adjusted R-squared:  0.1972
## F-statistic: 7.549 on 3 and 77 DF,  p-value: 0.0001715
##
##
## Value of test-statistic is: -4.5462 6.9125 10.3669
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$U %>%
  ur.df(., type = "trend", lags = 1) %>%
  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
## -0.67301 -0.21595 -0.03097  0.18204  1.20367
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.683200   0.264371   2.584   0.0116 *
## z.lag.1     -0.062507   0.025355  -2.465   0.0159 *
## tt          -0.002430   0.001709  -1.422   0.1591
## z.diff.lag   0.573870   0.092038   6.235 2.16e-08 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3517 on 78 degrees of freedom
## Multiple R-squared:  0.376,  Adjusted R-squared:  0.352
## F-statistic: 15.66 on 3 and 78 DF,  p-value: 4.585e-08
##
##
## Value of test-statistic is: -2.4652 2.3458 3.5068
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$U %>%
  diff() %>%
  ur.df(., type = "trend", lags = 0) %>%
  summary()
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81883 -0.20853 -0.05538  0.17026  1.27845
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.061241   0.082123   0.746    0.458
## z.lag.1     -0.453414   0.094261  -4.810 7.09e-06 ***
## tt          -0.001623   0.001731  -0.938    0.351
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3628 on 79 degrees of freedom
## Multiple R-squared:  0.2266, Adjusted R-squared:  0.207
## F-statistic: 11.57 on 2 and 79 DF,  p-value: 3.914e-05
##
##
## Value of test-statistic is: -4.8102 7.7158 11.5712
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$rw %>%
  ur.df(., type = "trend", lags = 4) %>%
  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
## -2.56910 -0.49348 -0.05407  0.53899  2.67852
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 20.17108    9.38785   2.149  0.03503 *
## z.lag.1     -0.04740    0.02305  -2.056  0.04342 *
## tt           0.03090    0.02045   1.511  0.13521
## z.diff.lag1  0.17198    0.10812   1.591  0.11608
## z.diff.lag2 -0.02622    0.10898  -0.241  0.81056
## z.diff.lag3 -0.08185    0.10919  -0.750  0.45594
## z.diff.lag4  0.32104    0.10758   2.984  0.00388 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8257 on 72 degrees of freedom
## Multiple R-squared:  0.3609, Adjusted R-squared:  0.3076
## F-statistic: 6.776 on 6 and 72 DF,  p-value: 1.009e-05
##
##
## Value of test-statistic is: -2.0558 3.2715 3.5018
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$rw %>%
  diff() %>%
  ur.df(., type = "trend", lags = 3) %>%
  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
## -2.62696 -0.48428 -0.09464  0.50459  3.09229
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.879815   0.410209   2.145  0.03530 *
## z.lag.1     -0.551650   0.181454  -3.040  0.00328 **
## tt          -0.009484   0.005816  -1.631  0.10724
## z.diff.lag1 -0.259314   0.162188  -1.599  0.11417
## z.diff.lag2 -0.270944   0.136800  -1.981  0.05141 .
## z.diff.lag3 -0.333664   0.109752  -3.040  0.00328 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8437 on 73 degrees of freedom
## Multiple R-squared:  0.4737, Adjusted R-squared:  0.4376
## F-statistic: 13.14 on 5 and 73 DF,  p-value: 3.983e-09
##
##
## Value of test-statistic is: -3.0402 3.4769 4.8506
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

```
dat$rw %>%
  diff() %>%
  ur.df(., type = "trend", lags = 0) %>%
  summary()
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2891 -0.5759 -0.0601  0.4996  3.4295
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.49355    0.29058   5.140 1.94e-06 ***
## z.lag.1     -0.77455    0.10979  -7.055 5.87e-10 ***
## tt          -0.01759    0.00479  -3.673 0.000435 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8829 on 79 degrees of freedom
## Multiple R-squared:  0.3865, Adjusted R-squared:  0.371
## F-statistic: 24.89 on 2 and 79 DF,  p-value: 4.147e-09
##
##
## Value of test-statistic is: -7.0548 16.6214 24.8884
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

3.1 Estimate the model

To estimate the model we combine all the endogenous variables in the order that is consistent with Breitung, et al. (2004), where we have `prod`, `e`, `U`, `rw`. We then consider the information criteria that is used to determine the lag order.

```
dat_can <- dat %>%
  dplyr::select(prod, e, U, rw)
VARselect(dat_can, lag.max = 8, type = "both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      2      1      3
##
## $criteria
##                   1            2            3
## AIC(n) -6.272579094 -6.636669729 -6.771176832
## HQ(n)  -5.978429479 -6.146420370 -6.084827731
## SC(n)  -5.536558039 -5.409967971 -5.053794371
## FPE(n)  0.001889842  0.001319462  0.001166019
##                   4            5            6
## AIC(n) -6.634609161 -6.398132194 -6.307704732
## HQ(n)  -5.752160316 -5.319583606 -5.033056400
## SC(n)  -4.426545996 -3.699388326 -3.118280161
## FPE(n)  0.001363175  0.001782055  0.002044202
##                   7           8
## AIC(n) -6.070727128 -6.06159666
## HQ(n)  -4.599979053 -4.39474884
## SC(n)  -2.390621854 -1.89081068
## FPE(n)  0.002768551  0.00306012
```

The results suggest between 1 and 3 lags, where Breitung, et al. (2004) make use of 3 and test for serial correlation in errors.

```
# Estimate models
p1ct <- VAR(dat_can, p = 1, type = "both")
p2ct <- VAR(dat_can, p = 2, type = "both")
p3ct <- VAR(dat_can, p = 3, type = "both")

# Serial
serial.test(p3ct, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object p3ct
## Chi-squared = 173.97, df = 208, p-value = 0.9587
```

```
serial.test(p2ct, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object p2ct
## Chi-squared = 209.74, df = 224, p-value = 0.7443
```

```
serial.test(p1ct, lags.pt = 16, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object p1ct
## Chi-squared = 233.5, df = 240, p-value = 0.606
```

```
serial.test(p3ct, lags.pt = 16, type = "PT.adjusted")
```

```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object p3ct
## Chi-squared = 198.04, df = 208, p-value = 0.6785
```

```
serial.test(p2ct, lags.pt = 16, type = "PT.adjusted")
```

```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object p2ct
## Chi-squared = 236.08, df = 224, p-value = 0.2769
```

```
serial.test(p1ct, lags.pt = 16, type = "PT.adjusted")
```

```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object p1ct
## Chi-squared = 256.88, df = 240, p-value = 0.2167
```

```
# JB
normality.test(p3ct)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object p3ct
## Chi-squared = 9.665, df = 8, p-value = 0.2893
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object p3ct
## Chi-squared = 4.3714, df = 4, p-value = 0.3581
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object p3ct
## Chi-squared = 5.2936, df = 4, p-value = 0.2585
```

```
normality.test(p2ct)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object p2ct
## Chi-squared = 2.2882, df = 8, p-value = 0.9709
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object p2ct
## Chi-squared = 1.2998, df = 4, p-value = 0.8614
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object p2ct
## Chi-squared = 0.9884, df = 4, p-value = 0.9115
```

```
normality.test(p1ct)
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object p1ct
## Chi-squared = 9.9189, df = 8, p-value = 0.2708
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object p1ct
## Chi-squared = 6.356, df = 4, p-value = 0.1741
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object p1ct
## Chi-squared = 3.5629, df = 4, p-value = 0.4684
```

```
# ARCH
arch.test(p3ct, lags.multi = 5)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object p3ct
## Chi-squared = 512.04, df = 500, p-value = 0.3451
```

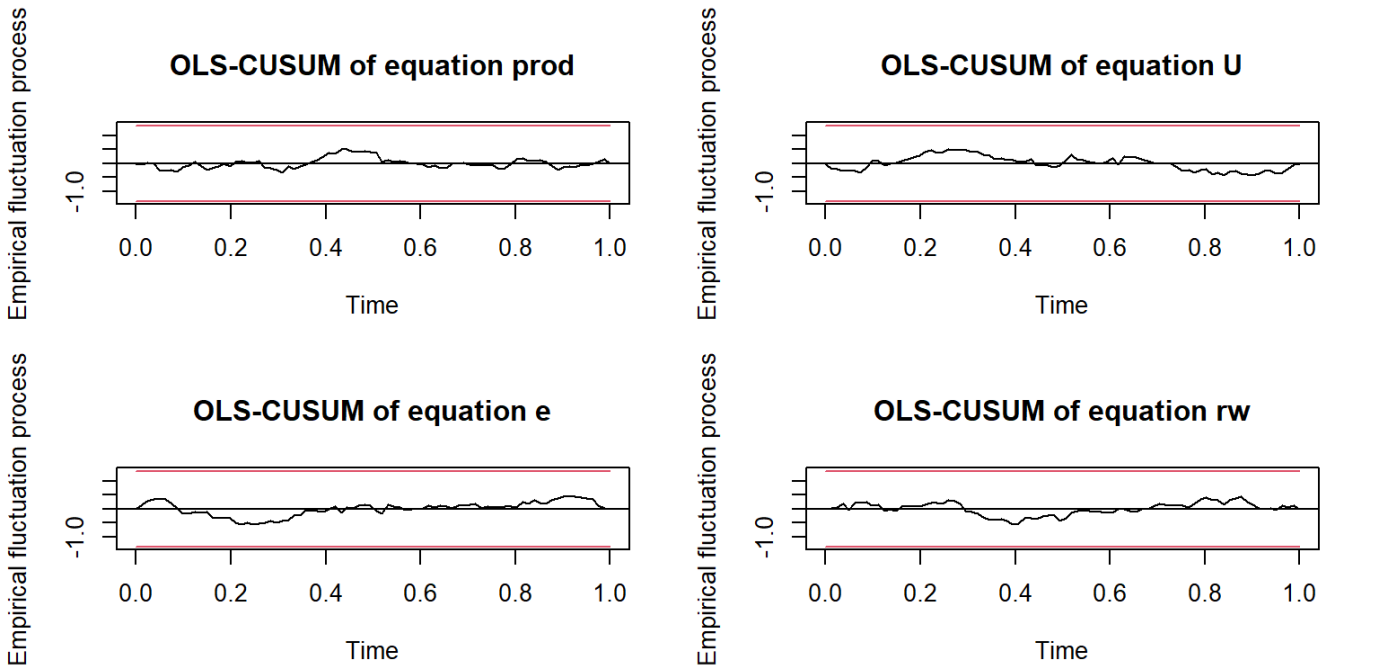
```
arch.test(p2ct, lags.multi = 5)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object p2ct
## Chi-squared = 528.14, df = 500, p-value = 0.1855
```

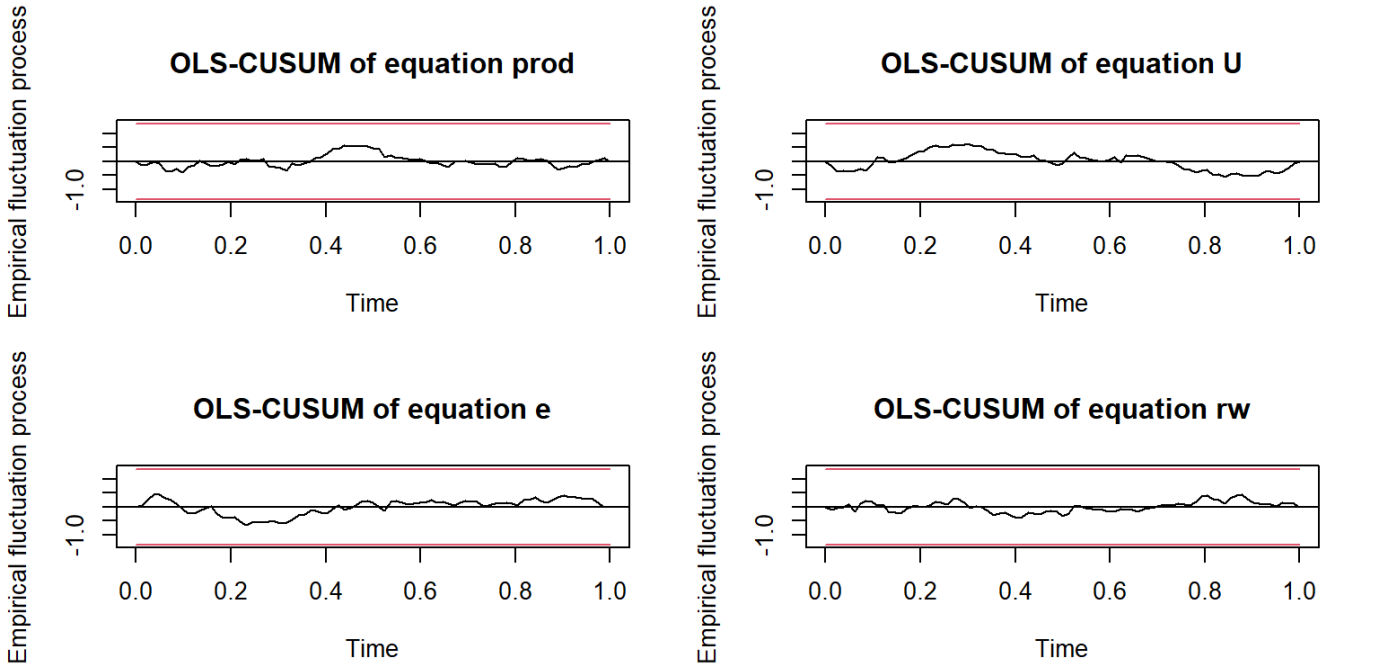
```
arch.test(p1ct, lags.multi = 5)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object p1ct
## Chi-squared = 570.14, df = 500, p-value =
## 0.01606
```

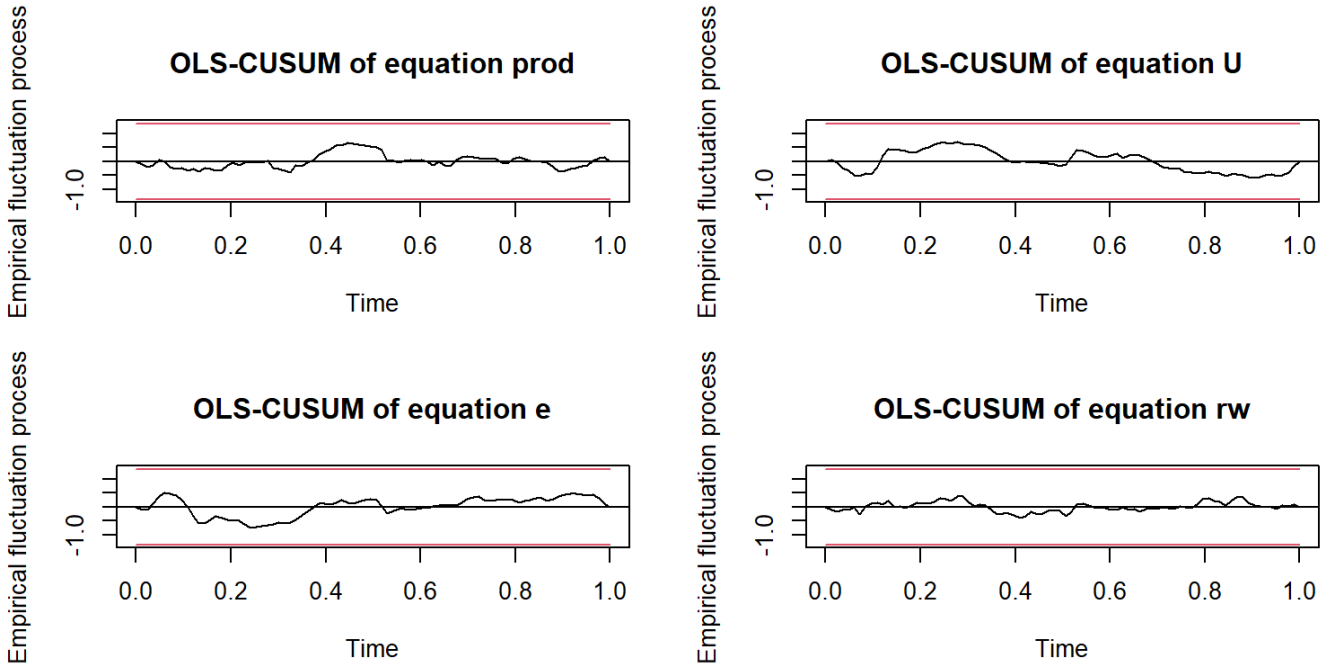
```
# Stability (Recursive CUSUM)
plot(stability(p3ct), nc = 2)
```




```
plot(stability(p2ct), nc = 2)
```




```
plot(stability(p1ct), nc = 2)
```



In terms of these results there may have a problem the heteroskedasticity when using only one lag. We can then use both the VAR(2) and/or VAR(3) form of the model to consider whether the variables are cointegrated or not. In terms of the race test:

```
trace_3 <- ca.jo(dat_can,
                 type = "trace",
                 ecdet = "trend",
                 K = 3) # 3 lags
summary(trace_3)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1] 4.505013e-01 1.962777e-01 1.676668e-01 4.647108e-02
## [5] 1.378552e-17
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 3 |  3.85 10.49 12.25 16.26
## r <= 2 | 18.72 22.76 25.32 30.45
## r <= 1 | 36.42 39.06 42.44 48.45
## r = 0  | 84.92 59.14 62.99 70.05
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              prod.l3       e.l3       U.l3      rw.l3
## prod.l3   1.00000000  1.0000000  1.0000000  1.0000000
## e.l3     -0.02385107  1.2946683 -2.8831549  4.2418095
## U.l3      3.16874537  3.4036737 -7.4261487  6.8413571
## rw.l3     1.83528094 -0.3330946  1.3978784 -0.1393999
## trend.l3 -1.30156066 -0.2302803 -0.5093217 -1.5925921
##             trend.l3
## prod.l3    1.0000000
## e.l3      -8.2903948
## U.l3     -12.5578447
## rw.l3      2.4466502
## trend.l3   0.2831079
##
## Weights W:
## (This is the loading matrix)
##
##             prod.l3        e.l3         U.l3
## prod.d -0.006535284 -0.02763445 -0.070975315
## e.d    -0.008503350  0.11414012 -0.008156655
## U.d    -0.004718575 -0.06154307  0.020719434
## rw.d   -0.046213364 -0.14579642 -0.016945115
##               rw.l3      trend.l3
## prod.d -0.014754350 -1.352490e-11
## e.d     0.003988049 -7.645184e-12
## U.d    -0.006557247  4.041018e-12
## rw.d    0.011896044 -1.113020e-11
```

```
trace_2 <- ca.jo(dat_can,
                 type = "trace",
                 ecdet = "trend",
                 K = 2) # 2 lags
summary(trace_2)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1]  4.483918e-01  2.323995e-01  1.313250e-01
## [4]  4.877894e-02 -5.413042e-17
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 3 |  4.10 10.49 12.25 16.26
## r <= 2 | 15.65 22.76 25.32 30.45
## r <= 1 | 37.33 39.06 42.44 48.45
## r = 0  | 86.12 59.14 62.99 70.05
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##             prod.l2        e.l2        U.l2     rw.l2
## prod.l2   1.0000000   1.0000000  1.00000000  1.000000
## e.l2      2.7132130  -6.3190317  0.49616476 16.333916
## U.l2      8.8369216 -15.2682862  1.48062670 25.774260
## rw.l2    -0.3716324   3.1817251 -0.04085216 -2.546391
## trend.l2 -0.4177976  -0.9335587 -0.26592659 -3.413555
##            trend.l2
## prod.l2    1.000000
## e.l2     -10.368561
## U.l2     -16.048486
## rw.l2      4.927457
## trend.l2  -1.753059
##
## Weights W:
## (This is the loading matrix)
##
##            prod.l2        e.l2        U.l2
## prod.d  0.02315564 -0.02832697 -0.10914770
## e.d     0.00560244 -0.01739149  0.08679397
## U.d    -0.01927714  0.01381763 -0.03696148
## rw.d   -0.08461896 -0.02739057 -0.07798404
##               rw.l2      trend.l2
## prod.d -0.006295988 -3.253597e-14
## e.d    -0.001019323  1.601749e-14
## U.d    -0.002276871 -2.523488e-14
## rw.d    0.003985020 -2.273140e-14
```

For both models it suggests that there is one cointegrating vector since when comparing the test statistic to the critical value at the 5% level of significance, we note that $36.42 < 42.44$ and $37.33 < 42.44$. Similarly, when considering the case where the null is that there are no cointegrating vectors, we note that for the calculated and test statistics, $84.92 > 62.99$ and $86.12 > 62.99$. To confirm these results we make use of the maximum eigenvalue test.

```
eigen_3 <- ca.jo(dat_can,
                 type = "eigen",
                 ecdet = "trend",
                 K = 3) # 3 lags
summary(eigen_3)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1] 4.505013e-01 1.962777e-01 1.676668e-01 4.647108e-02
## [5] 1.378552e-17
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 3 |  3.85 10.49 12.25 16.26
## r <= 2 | 14.87 16.85 18.96 23.65
## r <= 1 | 17.70 23.11 25.54 30.34
## r = 0  | 48.50 29.12 31.46 36.65
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              prod.l3       e.l3       U.l3      rw.l3
## prod.l3   1.00000000  1.0000000  1.0000000  1.0000000
## e.l3     -0.02385107  1.2946683 -2.8831549  4.2418095
## U.l3      3.16874537  3.4036737 -7.4261487  6.8413571
## rw.l3     1.83528094 -0.3330946  1.3978784 -0.1393999
## trend.l3 -1.30156066 -0.2302803 -0.5093217 -1.5925921
##             trend.l3
## prod.l3    1.0000000
## e.l3      -8.2903948
## U.l3     -12.5578447
## rw.l3      2.4466502
## trend.l3   0.2831079
##
## Weights W:
## (This is the loading matrix)
##
##             prod.l3        e.l3         U.l3
## prod.d -0.006535284 -0.02763445 -0.070975315
## e.d    -0.008503350  0.11414012 -0.008156655
## U.d    -0.004718575 -0.06154307  0.020719434
## rw.d   -0.046213364 -0.14579642 -0.016945115
##               rw.l3      trend.l3
## prod.d -0.014754350 -1.352490e-11
## e.d     0.003988049 -7.645184e-12
## U.d    -0.006557247  4.041018e-12
## rw.d    0.011896044 -1.113020e-11
```

```
eigen_2 <- ca.jo(dat_can,
                 type = "eigen",
                 ecdet = "trend",
                 K = 2) # 2 lags
summary(eigen_2)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend in cointegration
##
## Eigenvalues (lambda):
## [1]  4.483918e-01  2.323995e-01  1.313250e-01
## [4]  4.877894e-02 -5.413042e-17
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 3 |  4.10 10.49 12.25 16.26
## r <= 2 | 11.54 16.85 18.96 23.65
## r <= 1 | 21.69 23.11 25.54 30.34
## r = 0  | 48.78 29.12 31.46 36.65
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##             prod.l2        e.l2        U.l2     rw.l2
## prod.l2   1.0000000   1.0000000  1.00000000  1.000000
## e.l2      2.7132130  -6.3190317  0.49616476 16.333916
## U.l2      8.8369216 -15.2682862  1.48062670 25.774260
## rw.l2    -0.3716324   3.1817251 -0.04085216 -2.546391
## trend.l2 -0.4177976  -0.9335587 -0.26592659 -3.413555
##            trend.l2
## prod.l2    1.000000
## e.l2     -10.368561
## U.l2     -16.048486
## rw.l2      4.927457
## trend.l2  -1.753059
##
## Weights W:
## (This is the loading matrix)
##
##            prod.l2        e.l2        U.l2
## prod.d  0.02315564 -0.02832697 -0.10914770
## e.d     0.00560244 -0.01739149  0.08679397
## U.d    -0.01927714  0.01381763 -0.03696148
## rw.d   -0.08461896 -0.02739057 -0.07798404
##               rw.l2      trend.l2
## prod.d -0.006295988 -3.253597e-14
## e.d    -0.001019323  1.601749e-14
## U.d    -0.002276871 -2.523488e-14
## rw.d    0.003985020 -2.273140e-14
```

Once again, for both models it suggests that there is one cointegrating vector since for $r = 1$ we note that $17.70 < 25.54$ and $21.69 < 25.54$, while for $r = 0$, $48.50 > 31.46$ and $48.78 > 31.46$.

To derive an expression for the VECM that is normalised for `rw`, which was previosuly in the 4th column.

```
vecm <- dat_can %>%
  dplyr::select(rw, prod, e, U) %>%
  ca.jo(., type = "trace", ecdet = "trend", K = 3)

vecm_r1 <- cajorls(vecm, r = 1)

alpha <- coef(vecm_r1$rlm)[1, ]
beta <- vecm_r1$beta

resids <- resid(vecm_r1$rlm)
N <- nrow(resids)
sigma <- crossprod(resids) / N
```

The *t*-statistics for the speed of adjustment, α, coefficients may then be calculated as follows:

```
alpha_se <-
  sqrt(solve(crossprod(cbind(
    vecm@ZK %*% beta, vecm@Z1
  )))[1, 1] * diag(sigma))

alpha_t <- alpha / alpha_se
```

Note that `prod.d` and `U.d` would appear insignificant. Then the *t*-statistics for the coefficients that describe the long-run relationship (i.e. β) may be calculated as:

```
beta_se <-
  sqrt(diag(kronecker(
    solve(crossprod(vecm@RK[,-1])), solve(t(alpha) %*% solve(sigma) %*% alpha)
  )))
beta_t <- c(NA, beta[-1] / beta_se)
```

To view these statistics in a convenient manner:

```
print(rbind(alpha, alpha_t))
```

```
##                rw.d      prod.d         e.d
## alpha   -0.08481451 -0.01199408 -0.01560604
## alpha_t -5.71174163 -0.91861484 -2.15794372
##                  U.d
## alpha   -0.008659911
## alpha_t -1.486898975
```

```
print(t(cbind(beta, beta_t)))
```

```
##        rw.l3   prod.l3        e.l3     U.l3   trend.l3
## ect1       1 0.5448757 -0.01299587 1.726572 -0.7091888
## beta_t    NA 0.9004435 -0.01917210 1.193281 -2.5694062
```

4 Test exogeniety

To test for exogeniety we need to construct appropriate matrices. Where we want to consider if the speed of adjustment for the first variable is significant different from zero.

```
A1 <- diag(4)[, -1]
print(A1)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    1    0    0
## [3,]    0    1    0
## [4,]    0    0    1
```

And similarly so for the second, third and fourth variable.

```
A2 <- diag(4)[, -2]
print(A2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    0    0
## [3,]    0    1    0
## [4,]    0    0    1
```

```
A3 <- diag(4)[, -3]
print(A3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    0
## [4,]    0    0    1
```

```
A4 <- diag(4)[, -4]
print(A4)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
## [4,]    0    0    0
```

The exogeniety tests the be executed with the following commands.

```
alrtest(z = vecm, A = A1, r = 1) %>%
  summary()
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    1    0    0
## [3,]    0    1    0
## [4,]    0    0    1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.2999 0.1700 0.0852 0.0000 0.0000
##
## The value of the likelihood ratio test statistic:
## 19.62 distributed as chi square with 1 df.
## The p-value of the test statistic is: 0
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##                [,1]
## RK.rw.l3     1.0000
## RK.prod.l3  -1.3956
## RK.e.l3     -1.4890
## RK.U.l3     -3.0050
## RK.trend.l3 -0.1005
##
## Weights W of the restricted VAR:
##
##         [,1]
## [1,]  0.0000
## [2,]  0.0003
## [3,] -0.0386
## [4,]  0.0027
```

```
alrtest(z = vecm, A = A2, r = 1) %>%
  summary()
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    0    0
## [3,]    0    1    0
## [4,]    0    0    1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.4459 0.1957 0.0894 0.0000 0.0000
##
## The value of the likelihood ratio test statistic:
## 0.67 distributed as chi square with 1 df.
## The p-value of the test statistic is: 0.41
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##                [,1]
## RK.rw.l3     1.0000
## RK.prod.l3   0.4440
## RK.e.l3      0.1750
## RK.U.l3      2.5146
## RK.trend.l3 -0.7130
##
## Weights W of the restricted VAR:
##
##         [,1]
## [1,] -0.0731
## [2,]  0.0000
## [3,] -0.0142
## [4,] -0.0080
```

```
alrtest(z = vecm, A = A3, r = 1) %>%
  summary()
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    0
## [4,]    0    0    1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.4282 0.1687 0.0602 0.0000 0.0000
##
## The value of the likelihood ratio test statistic:
## 3.23 distributed as chi square with 1 df.
## The p-value of the test statistic is: 0.07
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##                [,1]
## RK.rw.l3     1.0000
## RK.prod.l3   1.7394
## RK.e.l3      1.5566
## RK.U.l3      6.5030
## RK.trend.l3 -1.2447
##
## Weights W of the restricted VAR:
##
##         [,1]
## [1,] -0.0636
## [2,] -0.0096
## [3,]  0.0000
## [4,] -0.0122
```

```
alrtest(z = vecm, A = A4, r = 1) %>%
  summary()
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
## [4,]    0    0    0
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.4388 0.1810 0.0949 0.0000 0.0000
##
## The value of the likelihood ratio test statistic:
## 1.7 distributed as chi square with 1 df.
## The p-value of the test statistic is: 0.19
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##                [,1]
## RK.rw.l3     1.0000
## RK.prod.l3   0.2250
## RK.e.l3     -0.6721
## RK.U.l3     -0.2810
## RK.trend.l3 -0.5132
##
## Weights W of the restricted VAR:
##
##         [,1]
## [1,] -0.0997
## [2,] -0.0169
## [3,] -0.0302
## [4,]  0.0000
```

In the case of `rw` the p-value is 0.00, which would suggest that we can reject the null of this restriction. However this is not the case for `prod` where the p-value is 0.41. For the case of `e` we can also reject the null of this restriction as the p-value is 0.07. The lastly, for `U` the p-value is 0.19, which would suggest taht we cannot reject the null of this restriction. Hence, it would appear as if most of the short run relationship is between the real wage and employment, as production and unemployment are weakly exogenous.

Then finally, to consider the possibility of a sign restriction on the long-run coefficient matrix, we can test whether wages and productivity are negatively related in long-run. In this case, we set up the variables and the restrictions of the `B` matrix as follows:

```
H1 <- dat_can %>%
  dplyr::select(rw, U, prod, e) %>%
  ca.jo(
    .,
    type = "trace",
    ecdet = "const",
    K = 2,
    spec = "transitory"
  )

B1 <- diag(5)[,-2]
B1[2,1] <- -1
print(B1)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]   -1    0    0    0
## [3,]    0    1    0    0
## [4,]    0    0    1    0
## [5,]    0    0    0    1
```

```
blrtest(z = H1, H = B1, r = 1) %>%
  summary()
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Estimation and testing under linear restrictions on beta
##
## The VECM has been estimated subject to:
## beta=H*phi and/or alpha=A*psi
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]   -1    0    0    0
## [3,]    0    1    0    0
## [4,]    0    0    1    0
## [5,]    0    0    0    1
##
## Eigenvalues of restricted VAR (lambda):
## [1] 0.5205 0.1528 0.0829 0.0243
##
## The value of the likelihood ratio test statistic:
## 3.26 distributed as chi square with 1 df.
## The p-value of the test statistic is: 0.07
##
## Eigenvectors, normalised to first column
## of the restricted VAR:
##
##           [,1]      [,2]      [,3]      [,4]
## [1,]    1.0000    1.0000    1.0000    1.0000
## [2,]   -1.0000   -1.0000   -1.0000   -1.0000
## [3,]   -0.0819    2.3349  -28.2836    0.3720
## [4,]   -1.8068   -3.1143    8.6924   -4.6704
## [5,] 1288.5129 1561.2633 2897.6613 3822.0671
##
## Weights W of the restricted VAR:
##
##           [,1]    [,2]    [,3]    [,4]
## rw.d   -0.0666 -0.0203  0.0015  0.0015
## U.d    -0.0104 -0.0024  0.0004 -0.0022
## prod.d -0.0023  0.0207  0.0028  0.0011
## e.d    -0.0075  0.0123 -0.0010  0.0018
```

Note that a p-value of 0.07 would suggest that we should reject the null of the sign restriction at 10% level.

5 Autoregressive Distributed Lag Model

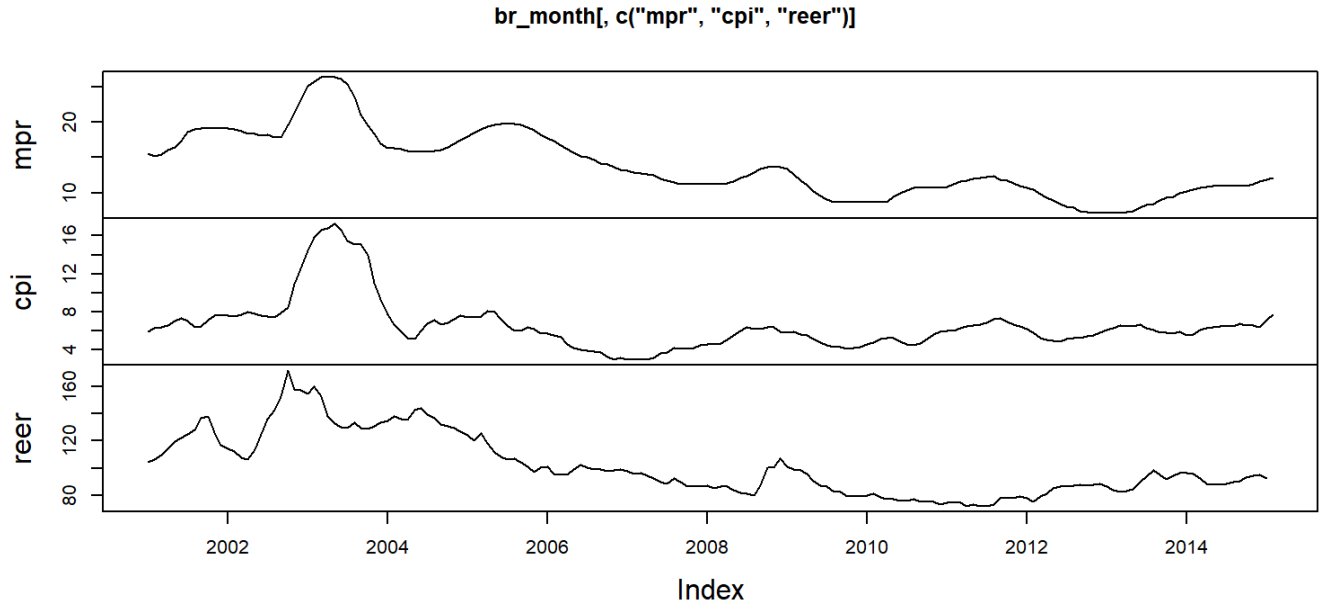
The package must be installed from a `GitHub` repository, which requires the use of the `devtools` package. The necessary commands are:

```
devtools::install_github("fcbarbi/ardl")
```

Thereafter, we can use the library command to access the routines in this package. The data pertains to measures for the interest rate, i_t, inflation, π_t, and the exchange rate, s_t, for Brazil between January 2001 up to December 2014.

```
library(tidyverse)
library(ardl)
library(vars)
data(br_month)
```

The variables have the following code names for the monetary policy rate `mpr`, prices `cpi`, and the exchange rate `reer`. The exogenous term used in this model is a dummy, `d_lula`, which is used to control for the first year in power of President Lula in 2003, when interest rates were increased to prevent a significant devaluation of the local currency. To consider the behaviour of the endogenous variables:



Note that `reer`, the real effective exchange rate, and the other regressors `cpi` and `mpr` look like unit-root processes. We can test for this with the aid of the Dickey-Fuller tests

```
br_month$cpi %>%
  ur.df(., type = "none", selectlags = c("BIC")) %>%
  summary()

br_month$mpr %>%
  ur.df(., type = "none", selectlags = c("BIC")) %>%
  summary()

br_month$reer %>%
  na.omit() %>%
  ur.df(.,
        type = "none",
        selectlags = c("BIC")) %>%
  summary()
```

All these results suggest that the variables are $I(1)$.

An ARDL(2,1,1) model structure for the monetary policy rate `mpr` with two regressors: prices `cpi` and the exchange rate `reer` with at most one lag each may then be expressed as:

$$i_t = \alpha + \phi_1 i_{t-1} + \phi_2 i_{t-2} + \beta_1 \pi_t + \beta_2 \pi_{t-1} + \beta_3 s_t + \beta_4 s_{t-1} + \varepsilon_t$$

To code this model we use the following command:

```
m1 <-
  ardl(
    mpr ~ cpi + reer |
      d_lula,
    data = br_month,
    ylag = 2,
    xlag = c(1, 1),
    case = 3
  )
```

```
##
## Dataset adjustment to the common sample of all regressors:
## Original dataset from  2001(1) to 2015(2)
## Adjusted dataset from  Jan 2001 to Jan 2015
```

The ARDL methodology allows the estimation in levels of a common long-term relation between the regressors and the explained variable. In function `coint()` a stationary specification is tested after controlling for the lag of the long-term relation, expressed as `L(coint)`.

To get model details on the coefficients and the usual tests use the traditional `summary()` function

```
summary(m1)
```

```
##
## Time series regression with "ts" data:
## Start = 2001(3), End = 2014(13)
##
## Call:
## dynlm::dynlm(formula = formula(fm), data = data, subset = subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93690 -0.11711 -0.01103  0.12217  1.17197
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.402581   0.121226  -3.321  0.00111 **
## L(mpr, 1)    1.700397   0.051446  33.052  < 2e-16 ***
## L(mpr, 2)   -0.733307   0.050578 -14.499  < 2e-16 ***
## cpi          0.069484   0.050493   1.376  0.17072
## L(cpi, 1)   -0.035500   0.050810  -0.699  0.48578
## reer         0.008958   0.005118   1.751  0.08196 .
## L(reer, 1)  -0.002349   0.005380  -0.437  0.66293
## d_lula      -0.541249   0.169134  -3.200  0.00166 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2683 on 159 degrees of freedom
## Multiple R-squared:  0.9969, Adjusted R-squared:  0.9967
## F-statistic:  7209 on 7 and 159 DF,  p-value: < 2.2e-16
```

To test the existence of the cointegration relation with the bounds test. The bounds test checks the existence of a long-term relation with critical values for I(0) and I(1) regressors.

```
bounds.test(m1)
```

```
##
## Bounds Test:
## mpr ~ +1 + L(mpr, 1) + L(mpr, 2) + cpi + L(cpi, 1) + reer + L(reer,      1) + d_lula
##
## PSS case 3  ( unrestricted intercert, no trend )
## Regressors (K) 2
##
## d(y_t) = alpha + pi (y_t-1,x_t)' + phi (d(y_t),d(x_t))' + epsilon_t
## Null hypothesis (H0): No long-run relation exist, ie H0:pi=0
##
##          I(0)   I(1)
##   10%   3.17  4.14
##    5%   3.79  4.85
##  2.5%   4.41  5.52
##    1%   5.15  6.36
##
## Wald test to compare the models:
## d(mpr) ~ +1+L(d(mpr)) +d(cpi)+d(reer)+d_lula
## d(mpr) ~ +1+L(d(mpr)) +L(mpr,1)+cpi+reer+d(cpi)+d(reer)+d_lula
##
## F statistic  7.716854
##
## Existence of a Long Term relation is not rejected at 5%
```

To visualize the long-term coefficients use the function `coint()`

```
coint(m1)
```

```
## AutoRegressive Distributed Lag model
## Dependent variable:  mpr
##
## Call:
## mpr ~ +1 + L(mpr, 1) + L(mpr, 2) + cpi + L(cpi, 1) + reer + L(reer,
##     1) + d_lula
## ____________________________________________________________
## Short-Run Coefficients. Dependent variable is d(mpr)
## ____________________________________________________________
##              Estimate   Std.Err Z value   Pr(>z)
## (Intercept) -0.406627  0.086541  -4.699 2.62e-06 ***
## L(d(mpr))    0.712596  0.048939  14.561  < 2e-16 ***
## d(cpi)       0.035800  0.049685   0.721   0.4712
## d(reer)      0.008433  0.005055   1.668   0.0953 .
## d_lula      -0.059326  0.093817  -0.632   0.5272
## L(coint)    -0.033565  0.006607  -5.080 3.77e-07 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## ____________________________________________________________
## Long-Run Coefficients. Dependent variable is mpr
## ____________________________________________________________
##         Estimate   Std.Err Z value   Pr(>z)
## cpi      1.03261   0.22795   4.530  5.9e-06 ***
## reer     0.20082   0.02826   7.106  1.2e-12 ***
## d_lula -16.44606   4.90103  -3.356 0.000792 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the SR coefficients (second panel) come from a model with regressors in first difference, which would imply that the variables are first-difference stationary.

This modelling framework allows for an automated model selection process, which selects the maximum lag for each regressor. If no such information is provided, it is assumed that a single lag is the maximum.

```
m2 <-
  auto.ardl(
    mpr ~ cpi + prod + reer |
      d_lula,
    data = br_month,
    ymax = 2,
    xmax = c(2, 2, 2),
    ic = "bic"
  )
summary(m2)
```

```
##
## Time series regression with "ts" data:
## Start = 2003(3), End = 2014(12)
##
## Call:
## dynlm::dynlm(formula = formula(fm), data = data, subset = subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89576 -0.09533 -0.00244  0.09232  0.73642
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.858712   0.681348   1.260  0.20973
## L(mpr, 1)    1.733216   0.047925  36.165  < 2e-16 ***
## L(mpr, 2)   -0.775263   0.048079 -16.125  < 2e-16 ***
## cpi          0.046507   0.020828   2.233  0.02720 *
## prod        -0.006345   0.003904  -1.625  0.10642
## reer         0.002320   0.001845   1.257  0.21094
## d_lula      -0.561685   0.180928  -3.104  0.00232 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2178 on 135 degrees of freedom
## Multiple R-squared:  0.9976, Adjusted R-squared:  0.9975
## F-statistic:  9532 on 6 and 135 DF,  p-value: < 2.2e-16
```

The selection process involves estimating the best fit for each regressor in the order they are included in the canonical equation. The algorithm will first adjust the best lag for the dependent variable and than proceed to test each regressor following the maximum lags dictated by the `xmax=c(2,0,1)` command that means “test up to the second lag of `cpi`, do not lag `prod` and test only one lag for `reer`”. By choosing `verbose=TRUE` you can follow all the tests.

```
m3 <-
  auto.ardl(
    mpr ~ cpi + reer,
    data = br_month,
    ymax = 2,
    xmax = c(1, 1),
    verbose = TRUE,
    case = 1
  )
```

```
##
## ARDL automatic model selection using bic with ymax= 2 and xmax= 1 1
## Model mpr ~ -1 + L(mpr, 1) + cpi + reer has bic = 276.6923
## Model mpr ~ -1 + L(mpr, 1) + L(mpr, 2) + cpi + reer has bic = 68.17748
## Model mpr ~ -1 + L(mpr, 1) + L(mpr, 2) + cpi + L(cpi, 1) + reer has bic = 71.63689
## Model mpr ~ -1 + L(mpr, 1) + L(mpr, 2) + cpi + reer + L(reer, 1) has bic = 72.84517
## Best model is mpr ~ -1 + L(mpr, 1) + L(mpr, 2) + cpi + reer chosen by bic = 68.17748
```

The selection algorithm relies on the user to choose the case to test. The following table describes what is inferred by the respective cases.

By default the choice is `case=3` (intercept only) but you can specify other cases to test.