Symmetric and Asymmetric ARDL Modeling with kardl

Huseyin Karamelikli and Huseyin Utku Demir

2025-07-29

Contents

Introduction

The `kardl` package is an R tool for estimating symmetric and asymmetric Autoregressive Distributed Lag (ARDL) and Nonlinear ARDL (NARDL) models, designed for econometricians and researchers analyzing cointegration and dynamic relationships in time series data. It offers flexible model specifications, allowing users to include deterministic variables, asymmetric effects for short- and long-run dynamics, and trend components. The package supports customizable lag structures, model selection criteria (AIC, BIC, AICc, HQ), and parallel processing for computational efficiency. Key features include:

- **Flexible Formula Specification**: Use `asym()`, `asymL()`, and `asymS()` to model asymmetric effects in short- and long-run dynamics, and `deterministic()` for dummy variables.
- **Lag Optimization**: Choose between automatic lag selection (`"quick"`, `"grid"`, `"grid_custom"`) or user-defined lags.
- **Dynamic Analysis**: Compute long-run coefficients, perform cointegration tests (PSS F, PSS t, Narayan, Banerjee, ECM), assess parameter stability (CUSUM, CUSUMQ), and test for heteroskedasticity (ARCH).
- **Visualization and Reporting**: Generate plots for lag criteria, and stability tests.

This vignette demonstrates how to use the `kardl()` function to estimate an asymmetric ARDL model, perform diagnostic tests, and visualize results, using economic data from Turkey.

Installation

`kardl` in R can easily be installed from its CRAN repository:

```
install.packages("kardl")
library(kardl)
```

Alternatively, you can use the `devtools` package to load directly from GitHub:

```r
# Install required packages
install.packages(c("stats", "msm", "lmtest", "nlWaldTest", "car", "strucchange", "utils"))
# Install kardl from GitHub
install.packages("devtools")
devtools::install_github("karamelikli/kardl")
```

Load the package:

```r
library(kardl)
library(magrittr)
```

Example: Estimating an Asymmetric ARDL Model

This example estimates an asymmetric ARDL model to analyze the dynamics of exchange rate pass-through to domestic prices in Turkey, using a sample dataset (`imf_example_data`) with variables for Consumer Price Index (CPI), Exchange Rate (ER), Producer Price Index (PPI), and a COVID-19 dummy variable.

Step 1: Data Preparation

Assume `imf_example_data` contains monthly data for CPI, ER, PPI, and a COVID dummy variable. We prepare the data by ensuring proper formatting and adding the dummy variable. We retrieve data from the IMF's International Financial Statistics (IFS) dataset and prepare it for analysis.

Note: The `imf_example_data` is a placeholder for demonstration purposes. You should replace it with your actual dataset.

```r
# Load required packages
library(ggplot2)
library(dplyr)
library(tidyr)

 trdata <- imf_example_data

# Define the model formula
MyFormula <- CPI ~ ER + PPI + asym(ER) + deterministic(covid) + trend
```

Step 2: Model Estimation

We estimate the ARDL model using different `mode` settings to demonstrate flexibility in lag selection. The `kardl()` function supports various modes: `"grid"`, `"grid_custom"`, `"quick"`, or a user-defined lag vector.

Using `mode = "grid"` The `"grid"` mode evaluates all lag combinations up to `maxlag` and provides console feedback.

```r
# Set model options
kardl_set(criterion = "BIC", DifferentAsymLag = TRUE)
# Estimate model with grid mode
kardl_model <- kardl(trdata, MyFormula, maxlag = 4, mode = "grid")
```

```r
# View results
kardl_model
```

```
## ===============================================================
## KARDL Results
## The lags of the model is:
##                       AIC               BIC             AICc                HQ
## lag             2,1,0,3           1,1,0,3          1,0,0,0           2,1,0,3
## value -5.54912168145612 -5.41064350008761 -4.8778697960035 -5.49312125506803
##
## The estimated model's formula is:
## L0.d.CPI~L1.CPI+L1.ER_POS+L1.ER_NEG+L1.PPI+L1.d.CPI+L0.d.ER_POS+L1.d.ER_POS+L0.d.ER_NEG+L0.d.PPI+L1.
##
## The coeffients estimation's results are:
##   (Intercept)        L1.CPI     L1.ER_POS     L1.ER_NEG        L1.PPI
## -2.634023e-01 -1.627396e-02  1.471461e-02  3.218940e-02  5.764187e-02
##      L1.d.CPI   L0.d.ER_POS   L1.d.ER_POS   L0.d.ER_NEG      L0.d.PPI
##  3.473671e-01  1.055467e-01  8.628757e-02  1.759167e-02  2.110331e-02
##      L1.d.PPI      L2.d.PPI      L3.d.PPI         covid         trend
## -1.273313e-02 -3.633936e-02 -3.299246e-02  7.230885e-03 -6.473474e-05
## ================================================================
```

```r
# Display model summary
summary(kardl_model)
```

```
## ===============================================================
## KARDL Summary
##
##
## Call:
## lm(formula = as.formula(fmodel), data = data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.051809 -0.008060 -0.001136  0.006519  0.100863
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.634e-01  4.738e-02  -5.560 4.64e-08 ***
## L1.CPI      -1.627e-02  3.740e-03  -4.351 1.68e-05 ***
## L1.ER_POS    1.471e-02  4.881e-03   3.015 0.002715 **
## L1.ER_NEG    3.219e-02  6.097e-03   5.279 2.02e-07 ***
## L1.PPI       5.764e-02  1.156e-02   4.985 8.87e-07 ***
## L1.d.CPI     3.474e-01  3.931e-02   8.837  < 2e-16 ***
## L0.d.ER_POS  1.055e-01  1.821e-02   5.795 1.28e-08 ***
## L1.d.ER_POS  8.629e-02  1.836e-02   4.699 3.48e-06 ***
## L0.d.ER_NEG  1.759e-02  4.690e-02   0.375 0.707758
## L0.d.PPI     2.110e-02  8.603e-03   2.453 0.014542 *
## L1.d.PPI    -1.273e-02  1.093e-02  -1.165 0.244826
## L2.d.PPI    -3.634e-02  1.021e-02  -3.558 0.000413 ***
## L3.d.PPI    -3.299e-02  8.738e-03  -3.776 0.000181 ***
## covid        7.231e-03  3.886e-03   1.861 0.063397 .
## trend       -6.473e-05  1.312e-04  -0.494 0.621889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01489 on 451 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.6464, Adjusted R-squared:  0.6354
## F-statistic: 58.88 on 14 and 451 DF,  p-value: < 2.2e-16
##
## ================================================================
```

Using User-Defined Lags Specify custom lags to bypass automatic lag selection:

```
kardl_model2 <- kardl(trdata, MyFormula, mode = c(2, 1, 1, 3))
kardl_model2$properLag
```

```
##    CPI ER_POS ER_NEG    PPI
##      2      1      1      3
```

Using All Variables Use the `.` operator to include all variables except the dependent variable:

```
kardl(trdata, CPI ~ . + deterministic(covid), mode = "grid")
```

```
## ===============================================================
## KARDL Results
## The lags of the model is:
##                        AIC                BIC               AICc                 HQ
## lag                  1,2,3              1,0,0              1,0,0              1,2,0
## value -3.30318556115875 -3.22112627740211 -2.97831477517044 -3.26494116658086
##
## The estimated model's formula is:
## L0.d.ER~L1.ER+L1.CPI+L1.PPI+L1.d.ER+L0.d.CPI+L0.d.PPI+covid
##
## The coeffients estimation's results are:
##  (Intercept)        L1.ER       L1.CPI       L1.PPI      L1.d.ER     L0.d.CPI
## -0.069790957 -0.025177677  0.022530297 -0.003051024  0.078941014  0.742452528
##     L0.d.PPI        covid
## -0.019897958  0.020270715
## ================================================================
```

Step 3: Visualizing Lag Criteria

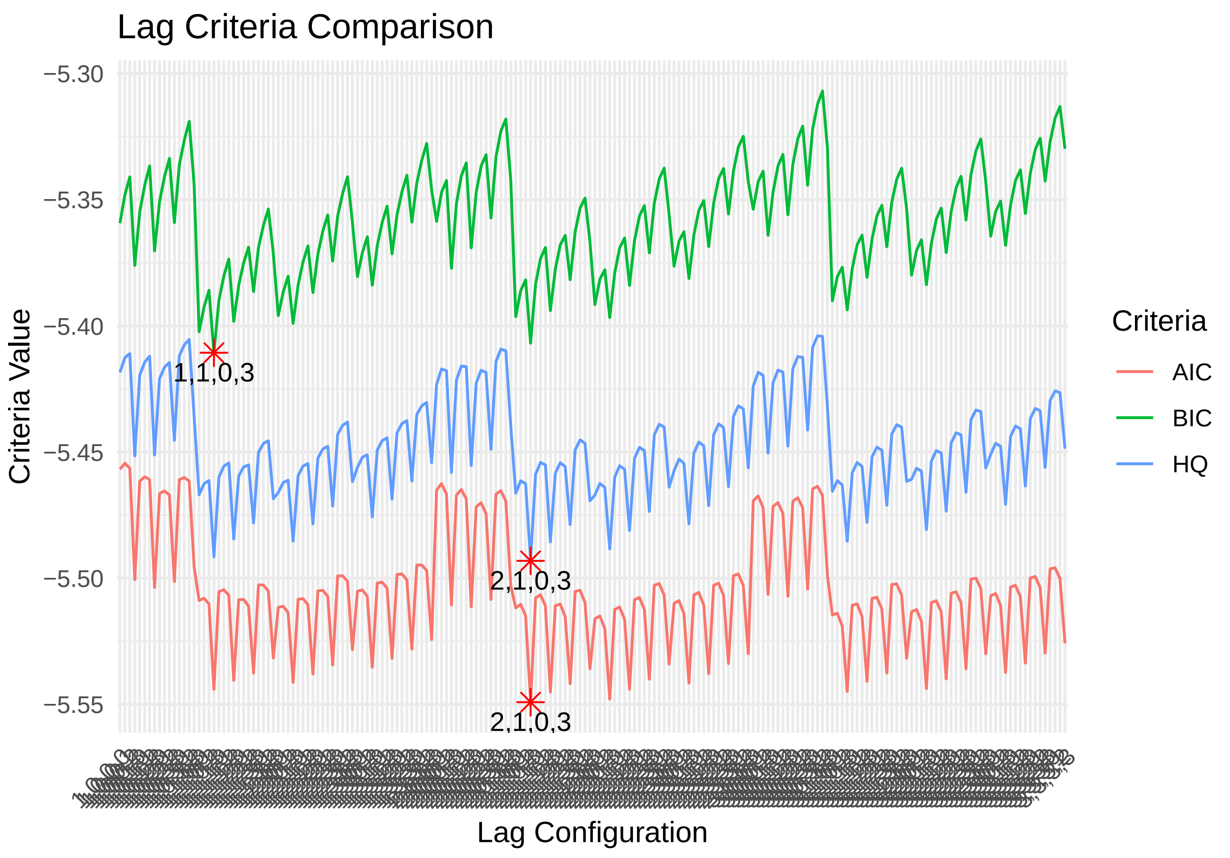
The `LagCriteria` component contains lag combinations and their criterion values. We visualize these to compare model selection criteria (AIC, BIC, HQ).

```
# Convert LagCriteria to a data frame
LagCriteria <- as.data.frame(kardl_model[["LagCriteria"]])
colnames(LagCriteria) <- c("lag", "AIC", "BIC", "AICc", "HQ")
LagCriteria <- LagCriteria %>% mutate(across(c(AIC, BIC, HQ), as.numeric))

# Pivot to long format
LagCriteria_long <- LagCriteria %>%
  select(-AICc) %>%
  pivot_longer(cols = c(AIC, BIC, HQ), names_to = "Criteria", values_to = "Value")

# Find minimum values
min_values <- LagCriteria_long %>%
  group_by(Criteria) %>%
  slice_min(order_by = Value) %>%
  ungroup()

# Plot
ggplot(LagCriteria_long, aes(x = lag, y = Value, color = Criteria, group = Criteria)) +
  geom_line() +
  geom_point(data = min_values, aes(x = lag, y = Value), color = "red", size = 3, shape = 8) +
  geom_text(data = min_values, aes(x = lag, y = Value, label = lag), vjust = 1.5, color = "black", size
  labs(title = "Lag Criteria Comparison", x = "Lag Configuration", y = "Criteria Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Step 4: Long-Run Coefficients

We calculate long-run coefficients using `kardl_longrun()`, which standardizes coefficients by dividing them by the negative of the dependent variable's long-run parameter.

```
# Long-run coefficients
mylong <- kardl_longrun(kardl_model)
```

```
mylong
```

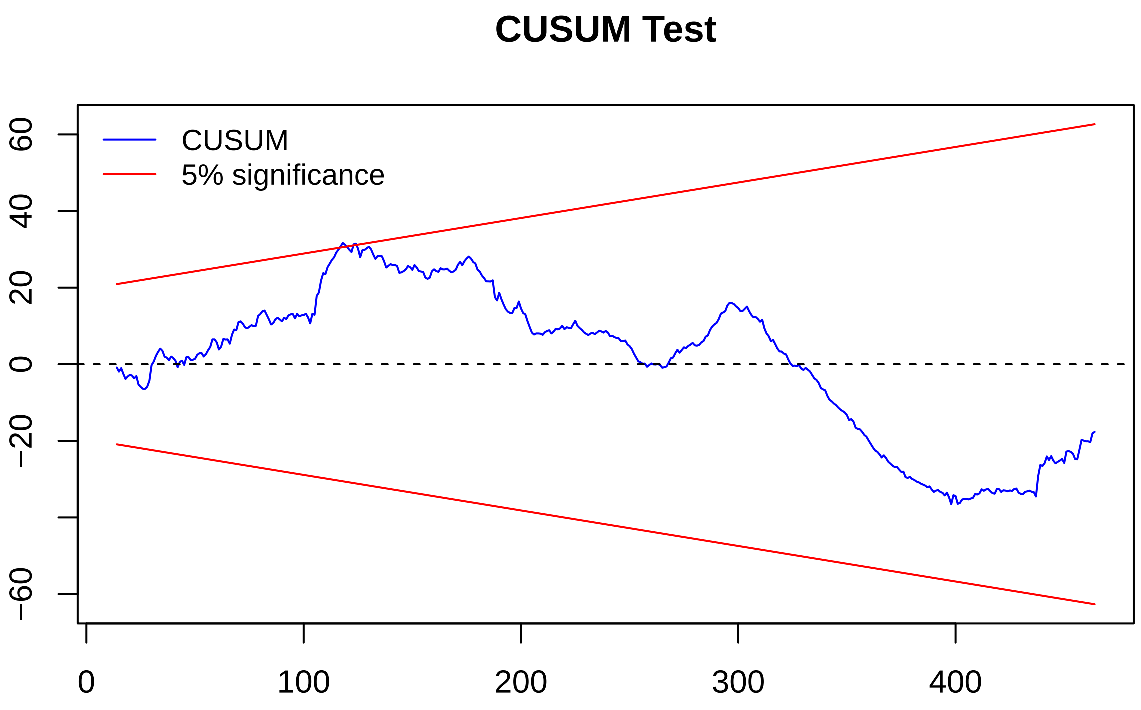
```
## ---------------------------------------------------
## Adjusted KARDL Results (long-run results)
##                  Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -16.1855020  3.1326537 -5.166706 3.584399e-07 ***
## L1.ER_POS     0.9041813  0.1554509  5.816506 1.139772e-08 ***
## L1.ER_NEG     1.9779693  0.4062688  4.868622 1.557148e-06 ***
## L1.PPI        3.5419692  0.9350204  3.788120 1.723626e-04 ***
## ---------------------------------------------------
##   Signif. codes: 0.001 = ***, 0.01 = **, 0.05 = *, 0.1 = .
## ---------------------------------------------------
```

Step 5: Parameter Stability Tests

We assess the stability of regression coefficients and variance using `cusum()` and `cusumq()`.

CUSUM Test The `cusum()` function plots the Cumulative Sum of recursive residuals to check for parameter stability.

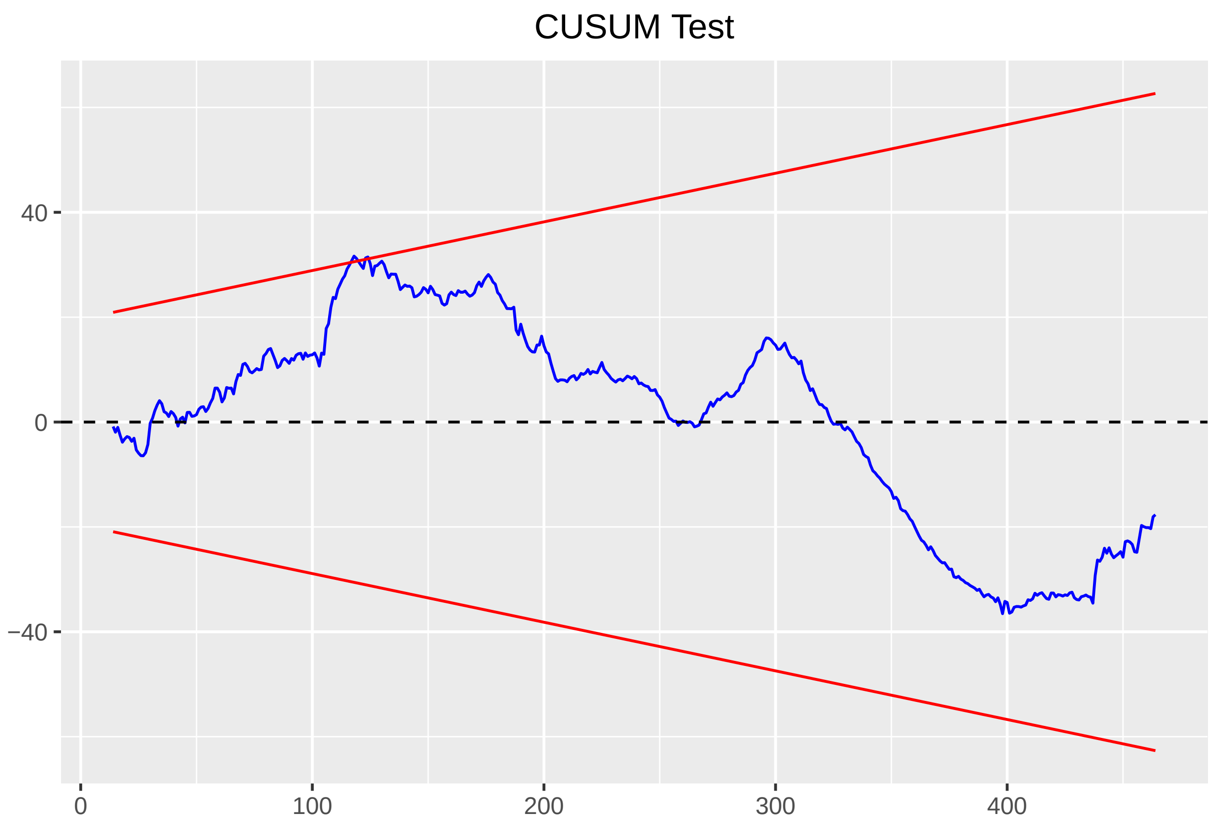
```
cDF <- cusum(kardl_model)
```



```
# View CUSUM data frame
head(cDF$dataframe)
```

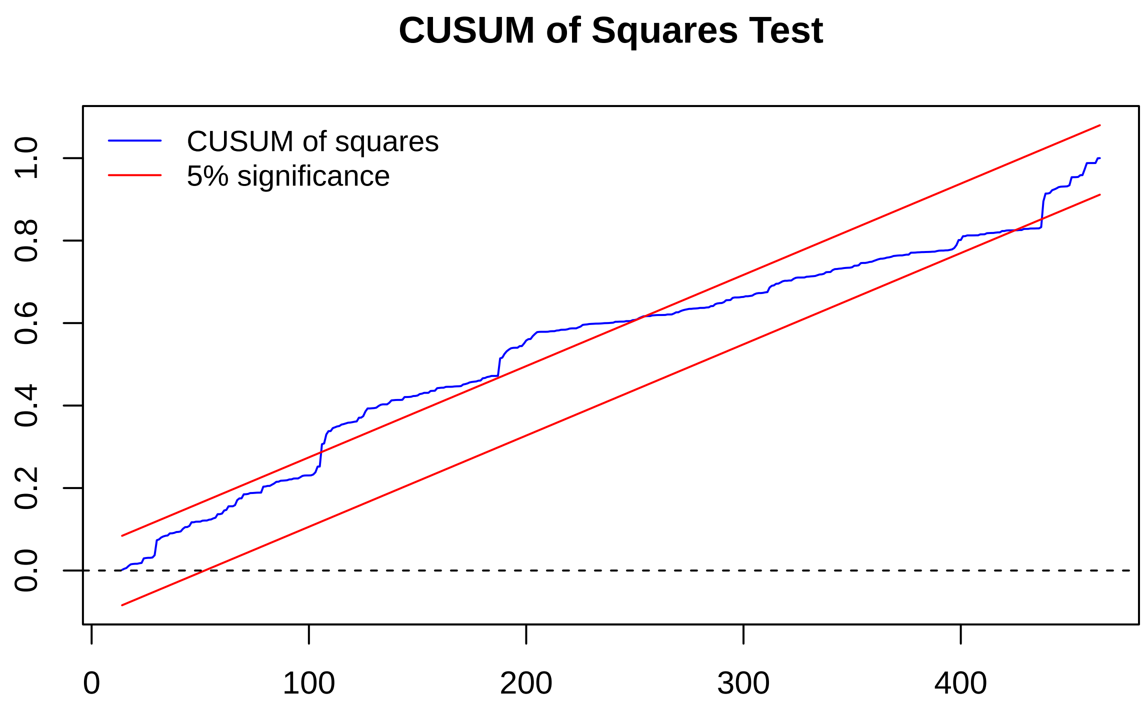
```
##    X     Lower          Y    Upper
## 1 14 -20.92013 -0.8485932 20.92013
## 2 15 -21.01290 -1.9379316 21.01290
## 3 16 -21.10567 -1.0150517 21.10567
## 4 17 -21.19844 -2.5043226 21.19844
## 5 18 -21.29122 -3.8463859 21.29122
## 6 19 -21.38399 -3.2121653 21.38399
```

```r
# Custom plot with ggplot2
ggplot(cDF$dataframe, aes(x = X, y = Y)) +
  geom_line(color = "blue") +
  geom_line(aes(y = Lower), color = "red") +
  geom_line(aes(y = Upper), color = "red") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  ggtitle("CUSUM Test") +
  xlab("") +
  ylab("") +
  ylim(range(cDF$dataframe$Lower, cDF$dataframe$Upper)) +
  scale_color_manual(values = c("blue", "red")) +
  guides(color = guide_legend(override.aes = list(linetype = c(1, 1)))) +
  theme(
    legend.position = "topleft",
    legend.title = element_blank(),
    plot.margin = margin(5, 4, 4, 1, "pt"),
    legend.background = element_blank(),
    legend.box.background = element_blank(),
    legend.key = element_blank(),
    plot.title = element_text(hjust = 0.5)
  )
```



CUSUM of Squares Test The `cusumq()` function plots the Cumulative Sum of Squares to assess variance stability.

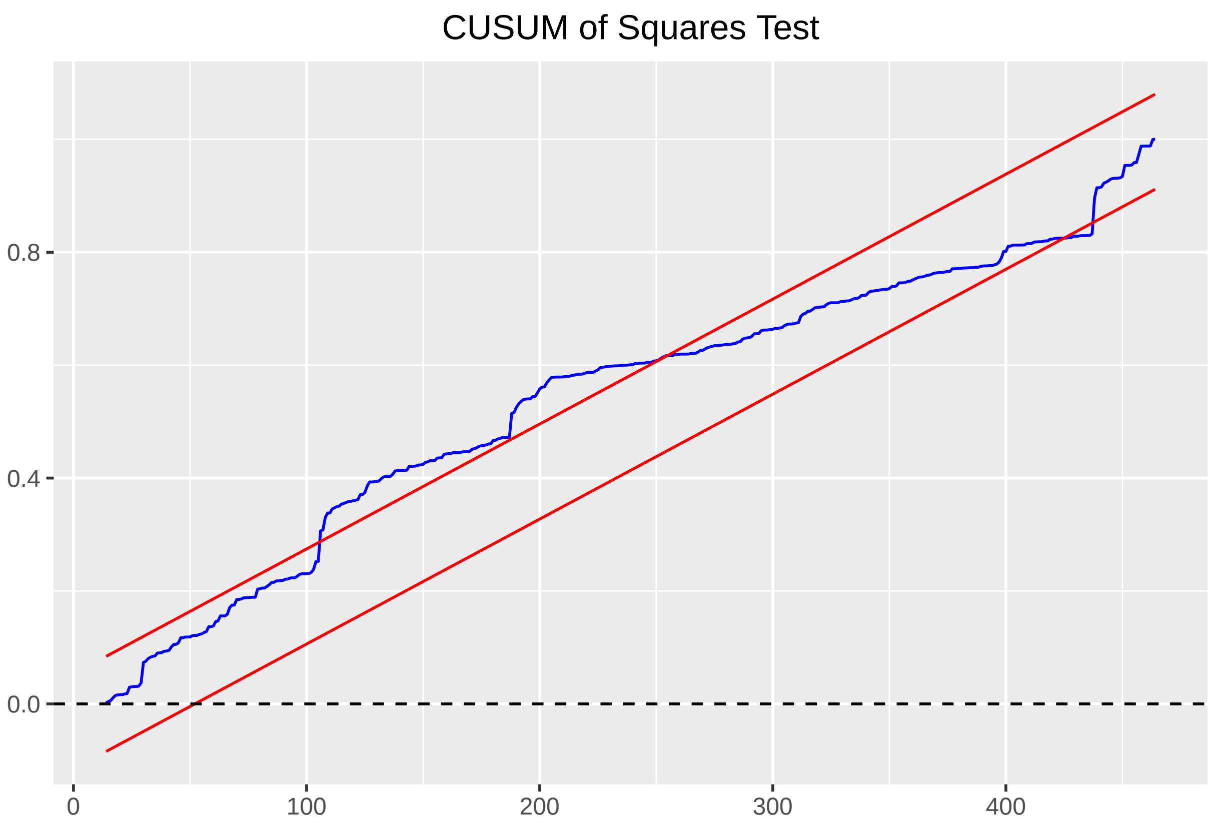
```
cDFq <- cusumq(kardl_model)
```



```
# View CUSUMQ data frame
head(cDFq$dataframe)
```

```
##    X      Lower           Y       Upper
## 1 14 0.08419278 0.001597786 -0.08419278
## 2 15 0.08640517 0.004230751 -0.08198039
## 3 16 0.08861756 0.006120525 -0.07976800
## 4 17 0.09082995 0.011041679 -0.07755561
## 5 18 0.09304234 0.015038047 -0.07534322
## 6 19 0.09525472 0.015930530 -0.07313083
```

```
# Custom plot with ggplot2
ggplot(cDFq$dataframe, aes(x = X, y = Y)) +
  geom_line(color = "blue") +
  geom_line(aes(y = Lower), color = "red") +
  geom_line(aes(y = Upper), color = "red") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  ggtitle("CUSUM of Squares Test") +
  xlab("") +
  ylab("") +
  ylim(range(cDFq$dataframe$Lower, cDFq$dataframe$Upper)) +
  scale_color_manual(values = c("blue", "red")) +
  guides(color = guide_legend(override.aes = list(linetype = c(1, 1)))) +
  theme(
    legend.position = "topleft",
    legend.title = element_blank(),
    plot.margin = margin(5, 4, 4, 1, "pt"),
    legend.background = element_blank(),
    legend.box.background = element_blank(),
    legend.key = element_blank(),
    plot.title = element_text(hjust = 0.5)
  )
```



Step 6: Asymmetry Test

The `asymmetrytest()` function performs Wald tests to assess short- and long-run asymmetry in the model.

```
ast <- trdata %>% kardl(CPI ~ ER + PPI + asym(ER + PPI) + deterministic(covid) + trend, mode = c(1, 2, 3
ast
```

```
## Asymmetries in the long run
##
##             F         P
## ER  2.1649064 0.1418984
## PPI 0.7152717 0.3981528
## ________________________________
## Asymmetries in the short run
##
##             F         P
## ER  1.6798112 0.1956198
## PPI 0.7582018 0.3843603
```

```r
# View long-run hypotheses
ast$Lhypotheses
```

```
## $H0
## [1] "- Coef(L1.ER_POS)/Coef(L1.CPI) = - Coef(L1.ER_NEG)/Coef(L1.CPI)"
## [2] "- Coef(L1.PPI_POS)/Coef(L1.CPI) = - Coef(L1.PPI_NEG)/Coef(L1.CPI)"
##
## $H1
## [1] "- Coef(L1.ER_POS)/Coef(L1.CPI)  - Coef(L1.ER_NEG)/Coef(L1.CPI)"
## [2] "- Coef(L1.PPI_POS)/Coef(L1.CPI)  - Coef(L1.PPI_NEG)/Coef(L1.CPI)"
```

```r
# Detailed results
summary(ast)
```

```
## Asymmetries in the long run
## H0: - Coef(L1.ER_POS)/Coef(L1.CPI) = - Coef(L1.ER_NEG)/Coef(L1.CPI)
## H1: - Coef(L1.ER_POS)/Coef(L1.CPI)  - Coef(L1.ER_NEG)/Coef(L1.CPI)
##
## H0: - Coef(L1.PPI_POS)/Coef(L1.CPI) = - Coef(L1.PPI_NEG)/Coef(L1.CPI)
## H1: - Coef(L1.PPI_POS)/Coef(L1.CPI)  - Coef(L1.PPI_NEG)/Coef(L1.CPI)
##
##             F         P df1 df2
## ER  2.1649064 0.1418984   1 446
## PPI 0.7152717 0.3981528   1 446
## ------------------------------
##
## Asymmetries in the long run
## H0: Coef(L0.d.ER_POS) + Coef(L1.d.ER_POS) + Coef(L2.d.ER_POS) = Coef(L0.d.ER_NEG) + Coef(L1.d.ER_NEG
## H1: Coef(L0.d.ER_POS) + Coef(L1.d.ER_POS) + Coef(L2.d.ER_POS)  Coef(L0.d.ER_NEG) + Coef(L1.d.ER_NEG)
##
## H0: Coef(L0.d.PPI_POS) = Coef(L0.d.PPI_NEG) + Coef(L1.d.PPI_NEG)
## H1: Coef(L0.d.PPI_POS)  Coef(L0.d.PPI_NEG) + Coef(L1.d.PPI_NEG)
##
##             F         P DF    Sum.of.Sq ResDF1 ResDF2      RSS1      RSS2
## ER  1.6798112 0.1956198  1 0.0003860722    447    446 0.1028906 0.1025045
## PPI 0.7582018 0.3843603  1 0.0001742581    447    446 0.1026788 0.1025045
```

Step 7: Cointegration Tests

We perform cointegration tests to assess long-term relationships using `pssf()`, `psst()`, `narayan()`, `banerjee()`, and `recmt()`.

PSS F Bound Test The `pssf()` function tests for cointegration using the Pesaran, Shin, and Smith F Bound test.

```r
A <- kardl_model %>% pssf(case = 3, signif_level = "0.05")
cat(paste0("The F statistic = ", A$statistic, " where k = ", A$k, "."))
```

```
## The F statistic = 12.7429667285825 where k = 2.
```

```r
cat(paste0("\nWe found '", A$Cont, "' at ", A$siglvl, "."))
```

```
##
## We found 'Cointegration' at 0.05.
```

```r
A$criticalValues
```

```
##      k  0.10L  0.10U  0.05L  0.05U 0.025L 0.025U  0.01L  0.01U
##   2.00   4.19   5.06   4.87   5.85   5.49   6.59   6.34   7.52
```

```r
summary(A)
```

```
## Method: PesaranF. Case: 5
## H0: Coef(L1.CPI) = Coef(L1.ER_POS) = Coef(L1.ER_NEG) = Coef(L1.PPI) = 0
## H1: Coef(L1.CPI)   Coef(L1.ER_POS)   Coef(L1.ER_NEG)   Coef(L1.PPI)  0
## The F statistics = 12.7429667285825 where k = 2.
## The proboblity is = 0.0000000008
## The results: There is Cointegration
## Significance level: 0.05
## ___________________________________________________
## Critique values are as follows:
##      k  0.10L  0.10U  0.05L  0.05U 0.025L 0.025U  0.01L  0.01U
##   2.00   4.19   5.06   4.87   5.85   5.49   6.59   6.34   7.52
```

PSS t Bound Test The `psst()` function tests the significance of the lagged dependent variable's coefficient.

```r
A <- kardl_model %>% psst(case = 3, signif_level = "0.05")
cat(paste0("The t statistic = ", A$statistic, " where k = ", A$k, "."))
```

```
## The t statistic = -4.35080566994952 where k = 3.
```

```r
cat(paste0("\nWe found '", A$Cont, "' at ", A$siglvl, "."))
```

```
##
## We found 'Cointegration' at 0.05.
```

```r
A$criticalValues
```

```
##      k  0.10L  0.10U  0.05L  0.05U 0.025L 0.025U  0.01L  0.01U
##   2.00  -3.13  -3.63  -3.41  -3.95  -3.65  -4.20  -3.96  -4.53
```

```r
summary(A)
```

```
## Method: Pesarant. Case: 5
## H0: Coef(L1.CPI) = 0
## H1: Coef(L1.CPI)  0
## The t statistics = -4.35080566994952 where k = 3.
## The results: There is Cointegration
## Significance level: 0.05
## ______________________________________________________
## Critique values are as follows:
##      k  0.10L  0.10U  0.05L  0.05U 0.025L 0.025U  0.01L  0.01U
##   2.00  -3.13  -3.63  -3.41  -3.95  -3.65  -4.20  -3.96  -4.53
```

Narayan Test The `narayan()` function is tailored for small sample sizes.

```
A <- kardl_model %>% narayan(case = 3, signif_level = "0.05")
cat(paste0("The F statistic = ", A$statistic, " where k = ", A$k, "."))
```

```
## The F statistic = 12.7429667285825 where k = 2.
```

```
cat(paste0("\nWe found '", A$Cont, "' at ", A$siglvl, "."))
```

```
##
## We found 'Cointegration' at 0.05.
```

```
A$criticalValues
```

```
##      k  0.10L  0.10U  0.05L  0.05U 0.025L 0.025U  0.01L  0.01U
##  2.000  4.307  5.223  5.067  6.103     NA     NA  6.730  8.053
```

```
summary(A)
```

```
## Method: Narayan. Case: 5
## H0: Coef(L1.CPI) = Coef(L1.ER_POS) = Coef(L1.ER_NEG) = Coef(L1.PPI) = 0
## H1: Coef(L1.CPI)   Coef(L1.ER_POS)   Coef(L1.ER_NEG)   Coef(L1.PPI)  0
## The F statistics = 12.7429667285825 where k = 2.
## The proboblity is = 0.0000000008
## The results: There is Cointegration
## Significance level: 0.05
## ______________________________________________________
## Critique values are as follows:
##      k  0.10L  0.10U  0.05L  0.05U 0.025L 0.025U  0.01L  0.01U
##  2.000  4.307  5.223  5.067  6.103     NA     NA  6.730  8.053
```

Banerjee Test The `banerjee()` function is designed for small datasets (100 observations).

```
A <- kardl_model %>% banerjee(signif_level = "0.05")
cat(paste0("The ECM parameter = ", A$coef, ", k = ", A$k, ", t statistic = ", A$statistic, "."))
```

```
## The ECM parameter = -0.013786158233982, k = 3, t statistic = -3.72124841419257.
```

```
cat(paste0("\nWe found '", A$Cont, "' at ", A$siglvl, "."))
```

```
##
## We found 'No Cointegration' at 0.05.
```

```
A$criticalValues
```

```
##  0.01  0.05  0.10  0.25
## -4.86 -4.19 -3.86 -3.30
```

```
summary(A)
```

```
## Method: Banerjee. Case: NULL
## H0: Coef(L1.CPI) = 0
## H1: Coef(L1.CPI)  0
## The coeffient of ECM is -0.013786158233982 , t=-3.72124841419257 k=3
## The results: There is No Cointegration
## Significance level: 0.05
## ___________________________________________________
## Critique values are as follows:
##  0.01  0.05  0.10  0.25
## -4.86 -4.19 -3.86 -3.30
```

Restricted ECM Test The `recmt()` function tests for cointegration using an Error Correction Model.

```
recmt_model <- trdata %>% recmt(MyFormula, mode = "grid_custom", case = 3)
recmt_model
```

```
##
## RESM test. Case: 5
## The t statistics = -0.692355032741378 where k = 3.
##
## Warnings:
## Warning: Trend is used in the model. The case was set to 5.
```

```
# View results
summary(recmt_model)
```

```
## Method: recmt. Case: 5
## H0: Coef(EcmRes) = 0
## H1: Coef(EcmRes)  0
## Attention: This function was performed in two steps.
## In the first step, the ECM was calculated and in the second step, the PSS test was performed.
##
## The coeffient of ECM is -0.00211093491158928 , t=-0.692355032741378 k=3
## The results: There is No Cointegration
## Significance level:
## ___________________________________________________
## Critique values are as follows:
##      k  0.10L  0.10U  0.05L  0.05U 0.025L 0.025U  0.01L  0.01U
##   2.00  -3.13  -3.63  -3.41  -3.95  -3.65  -4.20  -3.96  -4.53
```

Step 8: ARCH Test

The `archtest()` function checks for autoregressive conditional heteroskedasticity in the model residuals.

```r
arch_result <- archtest(kardl_model$finalModel$model$residuals, q = 2)
summary(arch_result)
```

```
## ARCH test
##
## F = 9.167386, p-value = 0.0001246777, df1 = 2, df2 = 461
##
##
## Call:
## lm(formula = as.formula(paste0("resid~", paste("resid", 1:q,
##     sep = "", collapse = "+"))), data = ndata)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0005732 -0.0001747 -0.0001381 -0.0000152  0.0099733
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.755e-04  3.061e-05   5.734 1.78e-08 ***
## resid1       1.984e-01  4.648e-02   4.268 2.40e-05 ***
## resid2      -2.290e-02  4.658e-02  -0.492    0.623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0006002 on 461 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.03825,    Adjusted R-squared:  0.03408
## F-statistic: 9.167 on 2 and 461 DF,  p-value: 0.0001247
```

Step 9: Customizing Asymmetric Variables

We demonstrate how to customize prefixes and suffixes for asymmetric variables using `kardl_set()`.

```r
# Set custom prefixes and suffixes
kardl_reset()
kardl_set(AsymPrefix = c("asyP_", "asyN_"), AsymSuffix = c("_PP", "_NN"))
kardl_custom <- kardl(trdata, MyFormula, mode = "grid_custom")
kardl_custom$properLag
```

```
##        CPI asyP_ER_PP asyN_ER_NN        PPI
##          2          1          0          3
```

Key Functions and Parameters

- **`kardl(data, model, maxlag, mode, ...)`**:
  - `data`: A time series dataset (e.g., a data frame with CPI, ER, PPI).
  - `model`: A formula specifying the long-run equation, e.g., `y ~ x + z + asym(z) + asymL(x2 + x3) + asymS(x3 + x4) + deterministic(dummy1 + dummy2) + trend`. Supports:
    - `asym()`: Asymmetric effects for both short- and long-run dynamics.
    - `asymL()`: Long-run asymmetric variables.

        * `asymS()`: Short-run asymmetric variables.
        * `deterministic()`: Fixed dummy variables.
        * `trend`: Linear time trend.
    - `maxlag`: Maximum number of lags (default: 4). Use smaller values (e.g., 2) for small datasets, larger values (e.g., 8) for long-term dependencies.
    - `mode`: Estimation mode:
        * `"quick"`: Verbose output for interactive use.
        * `"grid"`: Verbose output with lag optimization.
        * `"grid_custom"`: Silent, efficient execution.
        * User-defined vector (e.g., `c(1, 2, 4, 5)` or `c(CPI = 2, ER_POS = 3, ER_NEG = 1, PPI = 3)`).
    - Returns a list with components: `inputs`, `finalModel`, `start_time`, `end_time`, `properLag`, `TimeSpan`, `OptLag`, `LagCriteria`, `type` ("kardlmodel").

- **`kardl_set(...)`**: Configures options like `criterion` (AIC, BIC, AICc, HQ), `DifferentAsymLag`, `AsymPrefix`, `AsymSuffix`, `ShortCoef`, and `LongCoef`. Use `kardl_get()` to retrieve settings and `kardl_reset()` to restore defaults.

- **`kardl_longrun(model)`**: Calculates standardized long-run coefficients, returning `type` ("kardl_longrun"), `coef`, `delta_se`, `results`, and `starsDesc`.

- **`cusum(inputs_, saveToFile)`**: Plots the Cumulative Sum test for parameter stability, returning a list with a data frame for plotting.

- **`cusumq(inputs_, saveToFile)`**: Plots the Cumulative Sum of Squares test for variance stability, returning a similar list.

- **`asymmetrytest(model)`**: Performs Wald tests for short- and long-run asymmetry, returning `Lhypotheses`, `Lwald`, `Shypotheses`, `Swald`, and `type` ("asymmetrytest").

- **`pssf(model, case, signif_level)`**: Performs the Pesaran, Shin, and Smith F Bound test for cointegration, supporting cases 1–5 and significance levels ("auto", 0.01, 0.025, 0.05, 0.1, 0.10).

- **`psst(model, case, signif_level)`**: Performs the PSS t Bound test, focusing on the lagged dependent variable's coefficient.

- **`narayan(model, case, signif_level)`**: Conducts the Narayan test for cointegration, optimized for small samples (cases 2–5).

- **`banerjee(model, signif_level)`**: Performs the Banerjee cointegration test for small datasets (100 observations).

- **`recmt(data, model, maxlag, mode, case, signif_level, ...)`**: Conducts the Restricted ECM test for cointegration, with similar parameters to `kardl()` and case/significance level options.

- **`archtest(resid, q)`**: Tests for ARCH effects in model residuals, returning `type`, `statistic`, `parameter`, `p.value`, and `Fval`.

For detailed documentation, use `?kardl`, `?kardl_set`, `?kardl_longrun`, `?cusum`, `?cusumq`, `?asymmetrytest`, `?pssf`, `?psst`, `?narayan`, `?banerjee`, `?recmt`, or `?archtest`.

Conclusion

The `kardl` package is a versatile tool for econometric analysis, offering robust support for symmetric and asymmetric ARDL/NARDL modeling, cointegration tests, stability diagnostics, and heteroskedasticity checks. Its flexible formula specification, lag optimization, and support for parallel processing make it ideal for studying complex economic relationships. For more information, visit https://github.com/karamelikli/kardl or contact the authors at hakperest@gmail.com.