# Package 'kardl'

October 9, 2025

**Type** Package

**Title** Make Symmetric and Asymmetric ARDL Estimations

**Version** 0.1.1

**Maintainer** Huseyin Karamelikli <hakperest@gmail.com>

**Description** Implements estimation procedures for Autoregressive Distributed Lag (ARDL) and Nonlinear ARDL (NARDL) models, which allow researchers to investigate both short- and long-run relationships in time series data under mixed orders of integration. The package supports simultaneous modeling of symmetric and asymmetric regressors, flexible treatment of short-run and long-run asymmetries, and automated equation handling. It includes several cointegration testing approaches such as the Pesaran-Shin-Smith F and t bounds tests, the Banerjee error correction test, and the restricted ECM test, together with diagnostic tools including Wald tests for asymmetry, ARCH tests, and stability procedures (CUSUM and CUSUMQ). Methodological foundations are provided in Pesaran, Shin, and Smith (2001) <doi:10.1016/S0304-4076(01)00049-5> and Shin, Yu, and Greenwood-Nimmo (2014, ISBN:9780123855079).

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** stats, msm, lmtest, nlWaldTest, car, utils

**RoxygenNote** 7.3.3

**Suggests** knitr, rmarkdown, officer, flextable, equatags, magrittr, rlang, ggplot2, tidyr, dplyr, testthat (>= 3.0.0)

**NeedsCompilation** no

**VignetteBuilder** knitr

**Author** Huseyin Karamelikli [aut, cre] (ORCID: <https://orcid.org/0000-0001-7622-0972>), Huseyin Utku Demir [aut] (ORCID: <https://orcid.org/0000-0002-9140-0362>)

**Depends** R (>= 3.5.0)

**Config/testthat/edition** 3

**Repository** CRAN

**Date/Publication** 2025-10-09 12:10:11 UTC

# Contents



---

`archtest` *ARCH Test*

---

## Description

Autoregressive conditional heteroskedasticity ARCH(q)

$$\hat{\epsilon}_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \hat{\epsilon}_{t-i}^2$$

## Usage

```
archtest(resid, q = 1)
```

## Arguments

| | |
|---|---|
| `resid` | The residuals of the model |
| `q` | max lag |

## Value

A list of class "kardl" containing the following components:

- type: Type of the test
- statistic: The F-statistic of the test
- parameter: The degrees of freedom of the test
- p.value: The p-value of the test
- Fval: The F-value of the test

## References

Engle, Robert F. (1982). Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. Econometrica. 50 (4): 987–1007.

## See Also

bgtest bptest resettest

## Examples

```
kardl_model<-kardl(imf_example_data,
                   CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0))
archtest(kardl_model$finalModel$model$residuals,2)


# Summary of ARCH test
summary(archtest(kardl_model$finalModel$model$residuals))
```

---

# asymmetrytest *Wald tests for asymmetries*

---

## Description

The asymmetry test is a statistical procedure used to assess the presence of asymmetry in the relationship between variables in a model. It is particularly useful in econometric analysis, where it helps to identify whether the effects of changes in one variable on another are symmetric or asymmetric. The test involves estimating a model that includes both positive and negative components of the variables and then performing a Wald test to determine if the coefficients of these components are significantly different from each other. If the test indicates significant differences, it suggests that the relationship is asymmetric, meaning that the impact of increases and decreases in the variables differs.

The non-linear model with one asymmetric variables is specified as follows:

$$\Delta y_t = \psi + \eta_0 y_{t-1} + \eta_1^+ x_{t-1}^+ + \eta_1^- x_{t-1}^- + \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{j=0}^{q} \beta_j^+ \Delta x_{t-j}^+ + \sum_{j=0}^{m} \beta_j^- \Delta x_{t-j}^- + e_t$$

This function performs the asymmetry test both for long-run and short-run variables in a kardl model. It uses the `nlWaldtest` function from the **nlWaldTest** package for long-run variables and the `linearHypothesis` function from the **car** package for short-run variables. The hypotheses for the long-run variables are:

$$H_0 : -\frac{\eta_1^+}{\eta_0} = -\frac{\eta_1^-}{\eta_0}$$

$$H_1 : -\frac{\eta_1^+}{\eta_0} \neq -\frac{\eta_1^-}{\eta_0}$$

The hypotheses for the short-run variables are:

$$H_0 : \sum_{j=0}^{q} \beta_j^+ = \sum_{j=0}^{m} \beta_j^-$$

$$H_1 : \sum_{j=0}^{q} \beta_j^+ \neq \sum_{j=0}^{m} \beta_j^-$$

## Usage

```
asymmetrytest(model)
```

## Arguments

| | |
|---|---|
| `model` | The kardl obejct |

## Details

This function performs a Wald test to assess the validity of linearity or symmetry in both the short-run and long-run.

This function evaluates whether the inclusion of a particular variable in the model follows a linear relationship or exhibits a non-linear pattern. By analyzing the behavior of the variable, the function helps to identify if the relationship between the variable and the outcome of interest adheres to a straight-line assumption or if it deviates, indicating a non-linear interaction. This distinction is important in model specification, as it ensures that the variable is appropriately represented, which can enhance the model's accuracy and predictive performance.

## Value

A list with class "kardl" containing the following components:

- `Lhypotheses`: A list containing the null and alternative hypotheses for the long-run variables.
- `Lwald`: A data frame containing the Wald test results for the long-run variables, including F-statistic, p-value, degrees of freedom, and residual degrees of freedom.
- `Shypotheses`: A list containing the null and alternative hypotheses for the short-run variables.
- `Swald`: A data frame containing the Wald test results for the short-run variables, including F-statistic, p-value, degrees of freedom, residual degrees of freedom, and sum of squares.
- `type`: The type of the test, which is "asymmetrytest".

## References

Shin, Y., Yu, B., & Greenwood-Nimmo, M. (2014). Modelling asymmetric cointegration and dynamic multipliers in a nonlinear ARDL framework. Festschrift in honor of Peter Schmidt: Econometric methods and applications, 281-314.

## See Also

kardl, pssf, psst, banerjee, recmt, narayan

## Examples

```
kardl_model<-kardl(imf_example_data,
                   CPI~ER+PPI+asym(ER+PPI)+deterministic(covid)+trend,
                   mode=c(1,2,3,0,1))
ast<- asymmetrytest(kardl_model)
ast
ast$Lhypotheses

# Using magrittr package
library(magrittr)
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER+PPI)+deterministic(covid)+trend,
                   mode=c(1,2,3,0,1)) %>% asymmetrytest()

# Detailed results of the test:

summary(ast)
#Or, utilizing magrittr package
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER+PPI)+deterministic(covid)+trend,
                   mode=c(1,2,3,0,1)) %>% asymmetrytest() %>% summary()
```

---

# banerjee *Banerjee cointegration test*

---

## Description

The Banerjee t test is designated for small data length. It is not recommended to be utilized for data with more than 100 observations.

## Usage

```
banerjee(model, signif_level = "auto")
```

## Arguments

| | |
|---|---|
| model | The kardl obejct |
| signif_level | Character or numeric. Specifies the significance level to be used in the function. Acceptable values are "auto", "0.25", "0.10", "0.1", "0.05", and "0.01". If a numeric value is provided, it will be converted to a character string. If "auto" is chosen, the function determines the significance level automatically. Invalid values will result in an error. |

## Details

This function conducts the Banerjee cointegration test, which is specifically designed for datasets with a limited number of observations. The test assesses whether a long-term equilibrium relationship exists between variables by examining the residuals from a regression model. The Banerjee t-test is most effective for small sample sizes and is not recommended for datasets containing more than 100 observations, as its accuracy may decrease with larger data lengths. This test is useful for analyzing cointegration in small datasets, providing insights into the stability of relationships among variables over time.

## Value

A list containing the results of the Banerjee cointegration test, including:

- type: The type of test performed, which is "cointegration".
- case: The case number used in the test (1, 2, 3, 4, or 5).
- statistic: The t-statistic value calculated from the test.
- k: The number of long-run variables in the model.
- Cont: The conclusion of the test, indicating whether cointegration is present, inconclusive, or absent.
- BoundNum: A numeric representation of the conclusion, where 1 indicates cointegration, 0 indicates inconclusive, and -1 indicates no cointegration.
- siglvl: The significance level used in the test, either "auto" or one of the specified numeric levels.
- criticalValues: A vector of critical values for the test, corresponding to the significance levels.
- parameter: The names of the long-run variables in the model.
- FH0: The null hypothesis of the test, which includes the long-run variables set to zero.
- Fmodel: The linear hypothesis model used in the test.
- warnings: Any warnings generated during the test, such as sample size concerns.
- method: The method used for the test, which is "Narayan".

## References

Anindya Banerjee, Juan Dolado, Ricardo Mestre (1998) Error-correction Mechanism Tests for Cointegration in a Single-equation Framework, Journal of Time Series Analysis, Volume 19, Issue 3

## See Also

pssf psst recmt narayan

## Examples

```
kardl_model<-kardl(imf_example_data,
                   CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0))
A<- banerjee(kardl_model)
cat(paste0("The ECM parameter = ",A$coef,", k=",A$k," and the t statistics=",A$statistic,"."))
cat(paste0("\nWe found '",A$Cont, "' at ",A$siglvl,"."))


# Using magrittr
library(magrittr)
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                    mode=c(1,2,3,0)) %>% banerjee()

# critical Values are
A$criticalValues

# Getting details of the test.
 mySummary<-summary(A)
 mySummary

# The null hypothesis :
mySummary$H0

 # Using magrittr
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                    mode=c(1,2,3,0)) %>% banerjee() %>% summary()
```

---

# imf_example_data *IMF Example Data*

---

## Description

This is an example data set used for testing purposes.

## Usage

```
imf_example_data
```

## Format

A data frame with 470 rows and 4 variables:

**ER** Numeric. Exchange rate of Turkey.

**CPI** Numeric. CPI of Turkey.

**PPI** Numeric. PPI of Turkey.

**covid** Integer.Covid19 dummy variable.

## Details

These data obtained from **imf.data** package. The sample data is not updated and obtained by following codes.:

```
install.packages("imf.data")
library("imf.data")
IFS <- load_datasets("IFS")
```

- **PCPI_IX** Prices, Consumer Price Index, All items, Index
- **AIP_IX** Economic Activity, Industrial Production, Index
- **ENDE_XDC_USD_RATE** Exchange Rates, Domestic Currency per U.S. Dollar, End of Period, Rate

```
trdata<-IFS$get_series(freq = "M", ref_area = "TR", indicator = c("PCPI_IX","AIP_IX","ENDE_XDC_USD_RATE
= "1985-01",end_period = "2024-02")
PeriodRow<-trdata[,1]
trdata[,1]<-NULL
colnames(trdata)<-c("ER","CPI","PPI")
trdata<-log(as.data.frame(lapply(trdata, function(x) as.numeric(x))))
rownames(trdata)<-PeriodRow
```

**Inserting covid dummy variable**

```
trdata<-cbind(trdata,covid=0)
trdata[420:470,4]<-1
```

## See Also

load_datasets

## Examples

```
data(imf_example_data)
head(imf_example_data)
```

---

kardl *Estimation of ARDL and NARDL*

---

## Description

This function estimates an Autoregressive Distributed Lag (ARDL) or Nonlinear ARDL (NARDL) model based on the provided data and model formula. It allows for flexible specification of variables, including deterministic terms, asymmetric variables, and trend components. The function also supports automatic lag selection using various information criteria.

## Usage

```
kardl(
  data = NULL,
  model = NULL,
  maxlag = NULL,
  mode = NULL,
  criterion = NULL,
  differentAsymLag = NULL,
  batch = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | The data of analysis |
| `model` | A formula specifying the long-run model equation. This formula defines the relationships between the dependent variable and explanatory variables, including options for deterministic terms, asymmetric variables, and a trend component.<br>Example formula: `y ~ x + z + asym(z) + asymL(x2 + x3) + asymS(x3 + x4) + deterministic(dummy1 + dummy2) + trend`<br>***Details***<br>The formula allows flexible specification of variables and their roles in the model: - Deterministic variables (e.g., dummies) can be included using `deterministic()`. Multiple deterministic variables can be added with `+` (e.g., `deterministic(dummy1 + dummy2)`). These variables are considered fixed and are not associated with short-run or long-run dynamics. - Asymmetric variables can be included for both short-run and long-run dynamics:<br>• **asymS**: Specifies short-run asymmetric variables. For example, `asymS(x1 + x2)` includes variables `x1` and `x2` for short-run asymmetry.<br>• **asymL**: Specifies long-run asymmetric variables. For example, `asymL(x1 + x3)` includes variables `x1` and `x3` for long-run asymmetry.<br>• **asym**: Includes variables for both short-run and long-run asymmetry. For example, `asym(x1 + x3)` applies asymmetric decomposition for both dynamics.<br>A **trend** term can be added to the model to account for deterministic linear trends over time. Simply include `trend` in the formula.<br>These components can be combined flexibly in the formula to define a robust model tailored to your analysis. |
| `maxlag` | An integer specifying the maximum number of lags to be considered for the model. The default value is `4`. This parameter sets an upper limit on the lag length during the model estimation process.<br>***details***<br>The `maxlag` parameter is crucial for defining the maximum lag length that the model will evaluate when selecting the optimal lag structure based on the specified `criterion`. It controls the computational effort and helps prevent overfitting by restricting the search space for lag selection. |

- If the data has a short time horizon or is prone to overfitting, consider reducing `maxlag`. -
- If the data is expected to have long-term dependencies, increasing `maxlag` may be necessary to capture the relevant dynamics.

Setting an appropriate value for `maxlag` depends on the nature of your dataset and the context of the analysis:

- For small datasets or quick tests, use smaller values (e.g., `maxlag = 2`).
- For datasets with more observations or longer-term patterns, larger values (e.g., `maxlag = 8`) may be appropriate, though this increases computational time.

***examples***

Using the default maximum lag (4)

```
kardl(data, MyFormula, maxlag = 4)
```

Reducing the maximum lag to 2 for faster computation

```
kardl(data, MyFormula, maxlag = 2)
```

Increasing the maximum lag to 8 for datasets with longer dependencies

```
kardl(data, MyFormula, maxlag = 8)
```

`mode` Specifies the mode of estimation and output control. This parameter determines how the function handles lag estimation and what kind of feedback or control is provided during the process. The available options are:

- **"quick"** (default): Displays progress and messages in the console while the function estimates the optimal lag values. This mode is suitable for interactive use or for users who want to monitor the estimation process in real-time. It provides detailed feedback for debugging or observation but may use additional resources due to verbose output.
- **"grid"** : Displays progress and messages in the console while the function estimates the optimal lag values. This mode is suitable for interactive use or for users who want to monitor the estimation process in real-time. It provides detailed feedback for debugging or observation but may use additional resources due to verbose output.
- **"grid_custom"**: Suppresses most or all console output, prioritizing faster execution and reduced resource usage on PCs or servers. This mode is recommended for high-performance scenarios, batch processing, or when the estimation process does not require user monitoring. Suitable for large-scale or repeated runs where output is unnecessary.
- **User-defined vector**: A numeric vector of lag values specified by the user, allowing full customization of the lag structure used in model estimation. When a user-defined vector is provided (e.g., 'c(1, 2, 4, 5)'), the function skips the lag optimization process and directly uses the specified lags.
  - Users can define lag values directly as a numeric vector. For example: `mode = c(1, 2, 4, 5)` assigns lags of 1, 2, 4, and 5 to variables in the specified order. - Alternatively, lag values can be assigned to variables by name for clarity and control. For example: `mode = c(CPI = 2, ER_POS = 3, ER_NEG = 1, PPI = 3)` assigns lags to variables explicitly. - Ensure that the lags are correctly designated by verifying the result using `kardl_model$properLag` after estimation.

***Attention!*** -A function-based criterion or user-defined function can be specified for model selection, but this is only supported for `mode = "grid_custom"` and `mode = "quick"`. The `mode = "grid"` option is restricted to predefined criteria (e.g., AIC or BIC). For more information on available criteria, see the `modelCriterion` function documentation. - When using a numeric vector, ensure the order of lag values matches the variables in your formula. - If using named vectors, double-check the variable names to avoid mismatches or unintended results. - This mode bypasses the automatic lag optimization and assumes the user-defined lags are correct.

The ‘mode‘ parameter provides flexibility for different use cases: - Use ‘"grid"‘ mode for debugging or interactive use where progress visibility is important. - Use ‘"grid_custom"‘ mode to minimize overhead in computationally intensive tasks. - Specify a user-defined vector to customize the lag structure based on prior knowledge or analysis.

Selecting the appropriate mode can improve the efficiency and usability of the function depending on the user's requirements and the computational environment.

`criterion` A string specifying the information criterion to be used for selecting the optimal lag structure. The available options are:

- **"AIC"**: Akaike Information Criterion (default). This criterion balances model fit and complexity, favoring models that explain the data well with fewer parameters.
- **"BIC"**: Bayesian Information Criterion. This criterion imposes a stronger penalty for model complexity than AIC, making it more conservative in selecting models with fewer parameters.
- **"AICc"**: Corrected Akaike Information Criterion. This is an adjusted version of AIC that accounts for small sample sizes, making it more suitable when the number of observations is limited relative to the number of parameters.
- **"HQ"**: Hannan-Quinn Information Criterion. This criterion provides a compromise between AIC and BIC, favoring models that balance fit and complexity without being overly conservative.

The criterion can be specified as a string (e.g., `"AIC"`) or as a user-defined function that takes a fitted model object. Please visit the `modelCriterion` function documentation for more details on using custom criteria.

`differentAsymLag` A logical value indicating whether to allow different lag lengths for positive and negative decompositions.

`batch` A string specifying the batch processing configuration in the format "current_batch/total_batches". If a user utilize grid or grid_custom mode and want to split the lag search into multiple batches, this parameter can be used to define the current batch and the total number of batches. For example, "2/5" indicates that the current batch is the second out of a total of five batches. The default value is "1/1", meaning that the entire lag search is performed in a single batch.

`...` Additional arguments that can be passed to the function. These arguments can be used to

## Details

Note: All arguments of this function can be set using kardl_set function.

## Value

A list containing the final model, input parameters, and estimation results.

- inputs: The input parameters used for the model.
- finalModel: A list containing the final model formula, model object, number of parameters (k), sample size (n), start and end indices, and time span.
- start_time: The time when the model estimation started.
- end_time: The time when the model estimation ended.
- properLag: The optimal lag values for the short-run variables.
- TimeSpan: The total time span of the model.
- OptLag: A data frame containing the optimal lags and their corresponding criterion values.
- LagCriteria: A matrix containing the lag combinations and their corresponding criterion values.
- type: A string indicating the type of model, which is "kardlmodel".

## See Also

recmt, kardl_set, kardl_get, kardl_reset, modelCriterion

## Examples

```
# Sample article: THE DYNAMICS OF EXCHANGE RATE PASS-THROUGH TO DOMESTIC PRICES IN TURKEY
 library(dplyr)

  kardl_set(model=CPI~ER+PPI+asym(ER)+deterministic(covid)+trend ,
           data=imf_example_data,
           maxlag=2
           ) # setting the default values of the kardl function


 kardl_model_grid<-kardl( mode = "grid")
 kardl_model_grid
 kardl_model<- imf_example_data %>% kardl(mode = "grid_custom")
 kardl_model
 kardl_model2<-kardl(mode = c( 2    ,  1    ,  1   ,   3 ))

 # Getting the results
 kardl_model2

 # Getting the summary of the results
 summary(kardl_model)

 # OR
 imf_example_data %>% kardl(model=CPI~PPI+asym(ER)) %>% summary()
```

```
# using . in the formula means that all variables in the data will be used

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

# Setting max lag instead of default value [4]
kardl(imf_example_data,
      CPI~ER+PPI+asymL(ER),
      maxlag = 3, mode = "grid_custom")

# Using another criterion for finding the best lag#'
kardl_set(criterion = "HQ") # setting the criterion to HQ
kardl( mode = "grid_custom")

# using default values of lags
kardl( mode=c(1,2,3,0))

# summary( myNewStarSigns)
# For using different lag values for negative and positive decompositions of non-linear variables
# setting the same lags for positive and negative decompositions.
kardl_set(differentAsymLag = FALSE)

diffAsymLags<-kardl(model=CPI~asym(ER),maxlag=2, mode = "grid_custom")
diffAsymLags$properLag


# setting the different lags for positive and negative decompositions
kardl_set(differentAsymLag = TRUE)
sameAsymLags<-kardl(model=CPI~asym(ER),maxlag=2, mode = "grid_custom")
sameAsymLags$properLag

# Setting the preffixes and suffixes for non-linear variables
kardl_set(AsymPrefix = c("asyP_","asyN_"), AsymSuffix = c("_PP","_NN"))
kardl()

# For having the lags plot
library(ggplot2)

#  kardl_model_grid[["LagCriteria"]] is a matrix, convert it to a data frame
LagCriteria <- as.data.frame(kardl_model_grid[["LagCriteria"]])
# Rename columns for easier access and convert relevant columns to numeric
colnames(LagCriteria) <- c("lag", "AIC", "BIC", "AICc", "HQ")

LagCriteria <- LagCriteria %>%  mutate(across(c(AIC, BIC, HQ), as.numeric))

# Pivot the data to a long format excluding AICc
library(tidyr)

LagCriteria_long <- LagCriteria %>%
select(-AICc) %>%
pivot_longer(cols = c(AIC, BIC, HQ), names_to = "Criteria", values_to = "Value")
# Find the minimum value for each criterion
min_values <- LagCriteria_long %>%  group_by(Criteria) %>%
```

```
slice_min(order_by = Value) %>%  ungroup()

# Create the ggplot with lines, highlight minimum values, and add labels
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 = 3.5) +
   labs(title = "Lag Criteria Comparison ", x = "Lag Configuration",  y = "Criteria Value") +
 theme_minimal() +
 theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

---

kardl_get *Function to get settings*

---

## Description

This function retrieves options from the kardl package settings.

## Usage

```
kardl_get(...)
```

## Arguments

| | |
|---|---|
| ... | Names of the options to retrieve. If none provided, all options are returned. |

## Value

If no arguments are provided, returns all options as a list. If one option is requested, returns its value directly. If multiple options are requested, returns a list of those options.

## See Also

kardl_set, kardl_reset

## Examples

```
# Get specific options
kardl_get("criterion", "differentAsymLag")

# Get all options
# Note: In interactive use, avoid calling this directly to prevent cluttering the console.
a<-kardl_get()
a$AsymSuffix
```

kardl_longrun *Calculating the long-run parameters*

## Description

All long-run estimated parameters are standardized by dividing them by the negative of the long-run parameter of the dependent variable.

## Usage

```
kardl_longrun(model)
```

## Arguments

| | |
|---|---|
| model | the model produced by kardl |

## Details

This function is used to calculate the long-run parameters of a model produced by the kardl function.

It calculates the long-run multipliers and their standard errors, and returns a list containing the kardl_longrun coefficients, standard errors, and a data frame with the results.

## Value

a list of class kardl with the following elements:

- type: the type of the model, in this case "kardl_longrun"
- coef: the estimated coefficients of the long-run parameters
- delta_se: the standard errors of the long-run parameters
- results: a data frame with the estimated coefficients, standard errors, t-values, p-values, and significance stars
- starsDesc: a description of the significance stars used in the results

## See Also

kardl, pssf, psst

## Examples

```
kardl_model<-kardl(imf_example_data,
                   CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0))
kardl_longrun(kardl_model)

# Using magrittr
library(magrittr)
```

```
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                  mode=c(1,2,3,0)) %>% kardl_longrun()
```

---

kardl_reset *Function to reset kardl package settings*

---

## Description

This function resets all options in the kardl package to their default values.

## Usage

```
kardl_reset(. = FALSE)
```

## Arguments

| | |
|---|---|
| . | If provided and not 'FALSE', the function will return this value after resetting the settings. If not provided or set to 'FALSE', it will return the current settings. |

## Value

Returns the default options as a list.

## See Also

kardl_set, kardl_get

## Examples

```
# Set some options
kardl_set(criterion = "BIC", differentAsymLag = TRUE)
# Reset to default values
kardl_get("criterion")  # Check current settings
kardl_reset()
kardl_get("criterion")  # Check settings after reset

library(magrittr)
 MyFormula<-CPI~ER+PPI+asym(ER)+deterministic(covid)+trend
imf_example_data %>%
 kardl_set(LongCoef= "K1{lag}w1{varName}",differentAsymLag= FALSE ) %>% kardl(MyFormula ) %>%
    kardl_reset()
kardl_get()

imf_example_data %>%
  kardl_reset() %>%
   kardl_set(LongCoef= "K2{lag}w2{varName}",differentAsymLag=FALSE ) %>% kardl(MyFormula)

kardl_get(c("LongCoef","differentAsymLag","ShortCoef","batch"))
```

kardl_set *Function to get or set settings*

## Description

This function allows you to get or set various options related to the kardl package.

## Usage

```
kardl_set(. = FALSE, ...)
```

## Arguments

| | |
|---|---|
| . | If provided, the function will return this value. If not provided or set to 'FALSE', it will return the current settings. |
| ... | Named arguments to set options, or no arguments to get all options. |

## Value

If no arguments are provided, returns all options as a list. If named arguments are provided, sets those options and returns the updated list.

## See Also

kardl_get, kardl_reset

## Examples

```
# Set options
kardl_set(criterion = "BIC", differentAsymLag = TRUE)
# Get specific options
kardl_get("criterion", "differentAsymLag")
# Get all options.

# Note: In interactive use, avoid calling this directly to prevent cluttering the console.


kardl_get()


# Utilizing the magrittr pipe
library(magrittr)
# Set options and then get them

MyFormula<-CPI~ER+PPI+asym(ER)+deterministic(covid)+trend
kardl_set(ShortCoef = "L___{lag}.d.{varName}")
imf_example_data %>%   kardl(MyFormula)

kardl_reset()
```

```
kardl_get()

imf_example_data %>% kardl_set(LongCoef= "LK{lag}_{varName}",ShortCoef = "D{lag}.d.{varName}") %>%
kardl(MyFormula)
kardl_get(c("LongCoef","ShortCoef"))
```

---

`lmerge` *Merge two lists, accenting the first list.*

---

## Description

The first list of values takes precedence. When both lists have items with the same names, the values from the first list will be applied. In merging the two lists, priority is given to the left list, so if there are overlapping items, the corresponding value from the left list will be used in the merged result.

## Usage

```
lmerge(first, second, ...)
```

## Arguments

| | |
|---|---|
| `first` | list or vector |
| `second` | list or vector |
| `...` | other lists or vectors |

## Value

list

## See Also

append

## Examples

```
a<-list("a"="first a","b"="second a","c"=list("w"=12,"k"=c(1,3,6)))
b<-list("a"="first b","b"="second b","d"=14,"e"=45)
theResult<- lmerge(a,b)
unlist(theResult)

# for right merge
lmerge(b,a)

# Unisted return
theResult<- lmerge(a,b,c("v1"=11,22,3,"v5"=5))
theResult

m2<-list("m1"="kk2","m1.2.3"=list("m1.1.1"=333,"m.1.4"=918,"m.1.5"=982,"m.1.6"=981,"m.1.7"=928))
```

```
m3<-list("m1"="kk23","m2.3"=2233,"m1.2.4"=list("m1.1.1"=333444,"m.1.5"=982,"m.1.6"=91,"m.1.7"=928))
a<-c(32,34,542,"k"=35)
b<-c(65,"k"=34)

h1<-lmerge(a, m2)
unlist( h1)
h2<-lmerge(a,b,m2,m3,list("m1.1"=4))
unlist(h2)
```

---

`modelCriterion` *Model Selection Criterion*

---

## Description

Computes a model selection criterion (AIC, BIC, AICc, or HQ) or applies a user-defined function to evaluate a statistical model.

## Usage

```
modelCriterion(cr, model, ...)
```

## Arguments

| | |
|---|---|
| `cr` | A character string specifying the criterion to compute. Options are `"AIC"`, `"BIC"`, `"AICc"`, and `"HQ"`. Alternatively, a user-defined function can be provided. |
| `model` | An object containing the fitted model. The object should include at least:<br>• `model$model` – the actual fitted model object (e.g., from `lm`, `glm`).<br>• `k` – the number of estimated parameters.<br>• `n` – the sample size. |
| `...` | Additional arguments passed to the user-defined criterion function if `cr` is a function. |

## Details

This function returns model selection criteria used to compare the quality of different models. All criteria are defined such that **lower values indicate better models** (i.e., the goal is minimization).

If you wish to compare models using a maximization approach (e.g., log-likelihood), you can multiply the result by `-1`.

Note: The predefined string options (e.g., `"AIC"`) are **not** the same as the built-in R functions `AIC()` or `BIC()`. In particular, the values returned by this function are adjusted by dividing by the sample size `n` (i.e., normalized AIC/BIC), which makes it more comparable across datasets of different sizes.

The function returns:

- **"AIC"**: $\frac{2k-2\ell}{n}$ Akaike Information Criterion divided by `n`.

- **"BIC"**: $\frac{\log(n)\cdot k - 2\ell}{n}$ Bayesian Information Criterion divided by n.
- **"AICc"**: $\frac{2k(k+1)}{n-k-1} + \frac{2k-2\ell}{n}$ Corrected Akaike Information Criterion divided by n.
- **"HQ"**: $\frac{2\log(\log(n))\cdot k - 2\ell}{n}$ Hannan–Quinn Criterion divided by n.

where:

- $k$ is the number of parameters,
- $n$ is the sample size,
- $\ell$ is the log-likelihood of the model.

If cr is a function, it is called with the fitted model and any additional arguments passed through ....

## Value

A numeric value representing the selected criterion, normalized by the sample size if one of the predefined options is used.

## See Also

kardl

## Examples

```
model <- list(model = lm(mpg ~ wt + hp, data = mtcars), k = 3, n = nrow(mtcars))
modelCriterion(AIC, model)
#
modelCriterion("AIC", model)
modelCriterion("BIC", model)

# Using a custom criterion function
my_cr_fun <- function(mod, ...) { AIC(mod) / length(mod$model[[1]]) }
modelCriterion(my_cr_fun, model)
```

---

# narayan *NARAYAN test*

---

## Description

This function performs the Narayan test, which is designed to assess cointegration using critical values specifically tailored for small sample sizes. Unlike traditional cointegration tests that may rely on asymptotic distributions, the Narayan test adjusts for the limitations of small samples, providing more accurate results in such contexts. This makes the test particularly useful for studies with fewer observations, as it accounts for sample size constraints when determining the presence of a long-term equilibrium relationship between variables.

## Usage

```
narayan(model, case = 3, signif_level = "auto")
```

## Arguments

| | |
|---|---|
| model | The kardl obejct |
| case | Numeric or character. Specifies the case of the test to be used in the function. Acceptable values are 1, 2, 3, 4, 5, and "auto". If "auto" is chosen, the function determines the case automatically based on the model's characteristics. Invalid values will result in an error.<br>• 1: No intercept and no trend. This case is not supported by the Narayan test.<br>• 2: Restricted intercept and no trend.<br>• 3: Unrestricted intercept and no trend.<br>• 4: Unrestricted intercept and restricted trend.<br>• 5: Unrestricted intercept and unrestricted trend. |
| signif_level | Character or numeric. Specifies the significance level to be used in the function. Acceptable values are "auto", "0.10", "0.1", "0.05", "0.025", and "0.01". If a numeric value is provided, it will be converted to a character string. If "auto" is chosen, the function determines the significance level automatically. Invalid values will result in an error. |

## Value

A list with class "kardl" containing the following components:

- type: The type of the test, which is "cointegration".
- case: The case of the test, which is a numeric value (1, 2, 3, 4, or 5).
- statistic: The calculated F-statistic for the test.
- k: The number of long-run variables in the model.
- Cont: The conclusion of the test, indicating whether cointegration is found, inconclusive, or not found.
- BoundNum: A numeric value indicating the result of the test: 1 for cointegration, 0 for inconclusive, and -1 for no cointegration.
- siglvl: The significance level used in the test, which can be "auto", "0.10", "0.1", "0.05", "0.025", or "0.01".
- criticalValues: A numeric vector containing the critical values for the test based on the specified case and significance level.
- parameter: A character vector containing the names of the long-run parameters in the model.
- FH0: The null hypothesis of the test, which is a character string describing the hypothesis being tested.
- Fmodel: The detailed F test results, including the F-statistic, p-value, degrees of freedom, and residual degrees of freedom.
- warnings: A character vector containing any warnings generated during the test, such as issues with the model specification or data length.
- method: The method used for the test.

### Hypothesis testing

The null hypothesis (H0) of the F Bound test is that there is no cointegration among the variables in the model. In other words, it tests whether the long-term relationship between the variables is statistically significant. If the calculated F-statistic exceeds the upper critical value, we reject the null hypothesis and conclude that there is cointegration. Conversely, if the F-statistic falls below the lower critical value, we fail to reject the null hypothesis, indicating no evidence of cointegration. If the F-statistic lies between the two critical values, the result is inconclusive.

$$\Delta y_t = \psi + \varphi t + \eta_0 y_{t-1} + \sum_{i=1}^{k} \eta_i x_{i,t-1} + \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + e_t$$

**Cases 1, 3, 5:**

$$\mathbf{H_0}: \eta_0 = \eta_1 = \cdots = \eta_k = 0$$

$$\mathbf{H_1}: \eta_0 \neq \eta_1 \neq \ldots \neq \eta_k \neq 0$$

**Case 2:**

$$\mathbf{H_0}: \eta_0 = \eta_1 = \cdots = \eta_k = \psi = 0$$

$$\mathbf{H_1}: \eta_0 \neq \eta_1 \neq \ldots \neq \eta_k \neq \psi \neq 0$$

**Case 4:**

$$\mathbf{H_0}: \eta_0 = \eta_1 = \cdots = \eta_k = \varphi = 0$$

$$\mathbf{H_1}: \eta_0 \neq \eta_1 \neq \ldots \neq \eta_k \neq \varphi \neq 0$$

### References

Narayan, P. K. (2005). The saving and investment nexus for China: evidence from cointegration tests. Applied economics, 37(17), 1979-1990.

### See Also

pssf psst banerjee recmt

### Examples

```
kardl_model<-kardl(imf_example_data,
                   CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0))
A<-narayan(kardl_model)
A
cat(paste0("The F statistics=",A$statistic," where k=",A$k,"."))
cat(paste0("\nWe found '",A$Cont, "' at ",A$siglvl,"."))
cat(paste0("\nThe proboblity is = ",sprintf("%.10f",A$Fmodel[["Pr(>F)"]][2])))

# Utilising magrittr:

library(magrittr)
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0)) %>% narayan()
```

```
# Getting details of the test.
mySummary<-summary(A)
mySummary

# Utilising magrittr:
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0)) %>% narayan()  %>% summary()


# Critical Values are
A$criticalValues
# The null hypothesis :
A

# getting H0
mySummary$H0

# Detailed F test
A$Fmodel
```

---

`parseFormula` *Extract Variables from a Formula*

---

## Description

The `parseFormula()` unction extracts and lists variables from specific parts of a formula, especially those within parentheses of specified function types. This enables users to isolate and analyze segments of a model formula, such as variables within custom functions, with an option to ignore case sensitivity in pattern matching.

## Usage

```
parseFormula(formula_, matchPattern = "asym", ignore.case = FALSE)
```

## Arguments

| | |
|---|---|
| `formula_` | The initial formula for the model, typically specified using R's formula syntax (e.g., `y ~ x + f(x1 + x2)`). |
| `matchPattern` | A string or vector of strings that specifies the function(s) in the formula from which to extract variables. For example, `matchPattern = "f()"` will target variables within `f(...)` in the formula. |
| `ignore.case` | Logical; if `FALSE` (default), pattern matching is case-sensitive; if `TRUE`, case is ignored during matching.<br>A logical value indicating whether pattern matching should be case-sensitive (`FALSE`) or case-insensitive (`TRUE`). |

## Value

A list containing:

- Parsed variables: Variables that are located within the specified `matchPattern` part of the formula.
- Remaining model: The rest of the formula, excluding the parsed variables.

## See Also

`replaceValues`, `formula`

## Examples

```
# Identify variables within specific functions, such as 'asymS'
parsed<-parseFormula(y ~ x +det(s + d) + asymS(d + s),"asymS()")
parsed

 # Parse formulas containing various collection types like () or []
formula_ <- y ~ x +det(s -gg- d) + asymS(d2 -rr+ s)-mm[y1+y2+y3]+asym[k1+k2+k3]+trend-huseyin

# Extract variables in the 'asymS' function
parseFormula(formula_,"asymS")

# Use multiple functions to extract variables. If a specified function, such as "uuu",
# is not found, nothing is returned for it
a<-parseFormula(formula_,c("mm","det","uuu"))

 # To obtain variables not enclosed within any function, specify them directly
 parseFormula(formula_,c("trend","huseyin"))

# By default, 'parseFormula()' is case-sensitive.
# For case-insensitive matching, set 'ignore.case = TRUE'

parseFormula(formula_,"asyms", TRUE)
```

---

# pssf *PSS F Bound test*

---

## Description

This function performs the Pesaran et al. F Bound test

## Usage

```
pssf(model, case = 3, signif_level = "auto")
```

## Arguments

| | |
|---|---|
| `model` | The kardl obejct |
| `case` | Numeric or character. Specifies the case of the test to be used in the function. Acceptable values are 1, 2, 3, 4, 5, and "auto". If "auto" is chosen, the function determines the case automatically based on the model's characteristics. Invalid values will result in an error.<br>• `1`: No intercept and no trend<br>• `2`: Restricted intercept and no trend<br>• `3`: Unrestricted intercept and no trend<br>• `4`: Unrestricted intercept and restricted trend<br>• `5`: Unrestricted intercept and unrestricted trend |
| `signif_level` | Character or numeric. Specifies the significance level to be used in the function. Acceptable values are "auto", "0.10", "0.1", "0.05", "0.025", and "0.01". If a numeric value is provided, it will be converted to a character string. If "auto" is chosen, the function determines the significance level automatically. Invalid values will result in an error. |

## Details

This function performs the Pesaran, Shin, and Smith (PSS) F Bound test to assess the presence of a long-term relationship (cointegration) between variables in the context of an autoregressive distributed lag (ARDL) model. The PSS F Bound test examines the joint significance of lagged levels of the variables in the model. It provides critical values for both the upper and lower bounds, which help determine whether the variables are cointegrated. If the calculated F-statistic falls outside these bounds, it indicates the existence of a long-term equilibrium relationship. This test is particularly useful when the underlying data includes a mix of stationary and non-stationary variables.

## Value

A list with class "kardl" containing the following components:

- `type`: The type of the test, which is "cointegration".
- `case`: The case of the test, which is a numeric value (1, 2, 3, 4, or 5).
- `statistic`: The calculated F-statistic for the test.
- `k`: The number of long-run variables in the model.
- `Cont`: The conclusion of the test, indicating whether cointegration is found, inconclusive, or not found.
- `BoundNum`: A numeric value indicating the result of the test: 1 for cointegration, 0 for inconclusive, and -1 for no cointegration.
- `siglvl`: The significance level used in the test, which can be "auto", "0.10", "0.1", "0.05", "0.025", or "0.01".
- `criticalValues`: A numeric vector containing the critical values for the test based on the specified case and significance level.
- `parameter`: A character vector containing the names of the long-run parameters in the model.

- `FH0`: The null hypothesis of the test, which is a character string describing the hypothesis being tested.
- `Fmodel`: The detailed F test results, including the F-statistic, p-value, degrees of freedom, and residual degrees of freedom.
- `warnings`: A character vector containing any warnings generated during the test, such as issues with the model specification or data length.
- `method`: The method used for the test.

## Hypothesis testing

The null hypothesis (H0) of the F Bound test is that there is no cointegration among the variables in the model. In other words, it tests whether the long-term relationship between the variables is statistically significant. If the calculated F-statistic exceeds the upper critical value, we reject the null hypothesis and conclude that there is cointegration. Conversely, if the F-statistic falls below the lower critical value, we fail to reject the null hypothesis, indicating no evidence of cointegration. If the F-statistic lies between the two critical values, the result is inconclusive.

$$\Delta y_t = \psi + \varphi t + \eta_0 y_{t-1} + \sum_{i=1}^{k} \eta_i x_{i,t-1} + \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + e_t$$

**Cases 1, 3, 5:**

$$\mathbf{H_0}: \eta_0 = \eta_1 = \cdots = \eta_k = 0$$

$$\mathbf{H_1}: \eta_0 \neq \eta_1 \neq \ldots \neq \eta_k \neq 0$$

**Case 2:**

$$\mathbf{H_0}: \eta_0 = \eta_1 = \cdots = \eta_k = \psi = 0$$

$$\mathbf{H_1}: \eta_0 \neq \eta_1 \neq \ldots \neq \eta_k \neq \psi \neq 0$$

**Case 4:**

$$\mathbf{H_0}: \eta_0 = \eta_1 = \cdots = \eta_k = \varphi = 0$$

$$\mathbf{H_1}: \eta_0 \neq \eta_1 \neq \ldots \neq \eta_k \neq \varphi \neq 0$$

## References

Pesaran, M. H., Shin, Y. and Smith, R. (2001), "Bounds Testing Approaches to the Analysis of Level Relationship", Journal of Applied Econometrics, 16(3), 289-326.

## See Also

`psst` `banerjee` `recmt` `narayan`

## Examples

```
kardl_model<-kardl(imf_example_data,
                   CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0))
A<-pssf(kardl_model)
A
cat(paste0("The F statistics=",A$statistic," where k=",A$k,"."))
cat(paste0("\nWe found '",A$Cont, "' at ",A$siglvl,". "))
cat(paste0("\nThe proboblity is = ",sprintf("%.10f",A$Fmodel[["Pr(>F)"]][2])))

# Using magrittr :

library(magrittr)
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0)) %>% pssf()

# Getting details of the test.
mySummary<-summary(A)
mySummary

# Using magrittr:
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0)) %>% pssf() %>% summary()

# Critical Values are
A$criticalValues
# The null hypothesis :
A

# getting H0
mySummary$H0

# Detailed F test
A$Fmodel
```

---

`psst` *PSS t Bound test*

---

## Description

This function performs the Pesaran t Bound test

## Usage

```
psst(model, case = 3, signif_level = "auto")
```

## Arguments

| | |
|---|---|
| `model` | The kardl obejct |
| `case` | Numeric or character. Specifies the case of the test to be used in the function. Acceptable values are 1, 2, 3, 4, 5, and "auto". If "auto" is chosen, the function determines the case automatically based on the model's characteristics. Invalid values will result in an error.<br>• `1`: No intercept and no trend<br>• `2`: Restricted intercept and no trend<br>• `3`: Unrestricted intercept and no trend<br>• `4`: Unrestricted intercept and restricted trend<br>• `5`: Unrestricted intercept and unrestricted trend |
| `signif_level` | Character or numeric. Specifies the significance level to be used in the function. Acceptable values are "auto", "0.10", "0.1", "0.05", "0.025", and "0.01". If a numeric value is provided, it will be converted to a character string. If "auto" is chosen, the function determines the significance level automatically. Invalid values will result in an error. |

## Details

This function performs the Pesaran, Shin, and Smith (PSS) t Bound test, which is used to detect the existence of a long-term relationship (cointegration) between variables in an autoregressive distributed lag (ARDL) model. The t Bound test specifically focuses on the significance of the coefficient of the lagged dependent variable, helping to assess whether the variable reverts to its long-term equilibrium after short-term deviations. The test provides critical values for both upper and lower bounds. If the t-statistic falls within the appropriate range, it confirms the presence of cointegration. This test is particularly useful when working with datasets containing both stationary and non-stationary variables.

## Value

A list containing the results of the PSS t Bound test, including:

- `type`: The type of test performed, which is "cointegration".
- `case`: The case number used in the test (1, 2, 3, 4, or 5).
- `statistic`: The t-statistic value calculated from the test.
- `k`: The number of long-run variables in the model.
- `Cont`: The conclusion of the test, indicating whether cointegration is present, inconclusive, or absent.
- `BoundNum`: A numeric representation of the conclusion, where 1 indicates cointegration, 0 indicates inconclusive, and -1 indicates no cointegration.
- `siglvl`: The significance level used in the test, either "auto" or one of the specified numeric levels.
- `criticalValues`: A vector of critical values for the test, corresponding to the significance levels.
- `parameter`: The names of the long-run variables in the model.

- `FH0`: The null hypothesis of the test, which includes the long-run variables set to zero.
- `Fmodel`: The linear hypothesis model used in the test.
- `warnings`: Any warnings generated during the test, such as sample size concerns.
- `method`: The method used for the test, which is "Narayan".

## Hypothesis testing

The PSS t Bound test evaluates the null hypothesis that the long-run coefficients of the model are equal to zero against the alternative hypothesis that at least one of them is non-zero. The test is conducted under different cases, depending on the model specification.

$$\Delta y_t = \psi + \varphi t + \eta_0 y_{t-1} + \sum_{i=1}^{k} \eta_i x_{i,t-1} + \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + e_t$$

$$\mathbf{H_0} : \eta_0 = 0$$

$$\mathbf{H_1} : \eta_0 \neq 0$$

## References

Pesaran, M. H., Shin, Y. and Smith, R. (2001), "Bounds Testing Approaches to the Analysis of Level Relationship", Journal of Applied Econometrics, 16(3), 289-326.

## See Also

`pssf` `banerjee` `recmt` `narayan`

## Examples

```
kardl_model<-kardl(imf_example_data,
                   CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0))
A<- psst(kardl_model)
cat(paste0("The t statistics=",A$statistic," where k=",A$k,"."))
 cat(paste0("\nWe found '",A$Cont, "' at ",A$siglvl,"."))


# Using magrittr

library(magrittr)
imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                   mode=c(1,2,3,0)) %>% psst()

# critical Values are
A$criticalValues

# Getting details of the test.
mySummary<-summary(A)
mySummary
```

```
# The null hypothesis :
mySummary$H0

imf_example_data %>% kardl(CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,
                    mode=c(1,2,3,0)) %>% psst() %>% summary()
```

---

recmt *Restricted ECM test*

---

## Description

This function is used to perform the Error Correction Model (ECM) test, which is designed to determine whether there is cointegration in the model. Cointegration indicates a long-term equilibrium relationship between variables, despite short-term deviations. The ECM test helps identify if such a long-term relationship exists by examining the short-run dynamics and adjusting for deviations from equilibrium. If the test confirms cointegration, it suggests that the variables move together over time, maintaining a stable long-term relationship. This is critical for ensuring that the model properly captures both short-term fluctuations and long-term equilibrium behavior.

## Usage

```
recmt(
  data = NULL,
  model = NULL,
  case = 3,
  signif_level = "auto",
  maxlag = NULL,
  mode = NULL,
  criterion = NULL,
  differentAsymLag = NULL,
  batch = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | The data of analysis |
| `model` | A formula specifying the long-run model equation. This formula defines the relationships between the dependent variable and explanatory variables, including options for deterministic terms, asymmetric variables, and a trend component.<br>Example formula: `y ~ x + z + asym(z) + asymL(x2 + x3) + asymS(x3 + x4) + deterministic(dummy1 + dummy2) + trend`<br>***Details***<br>The formula allows flexible specification of variables and their roles in the model: - Deterministic variables (e.g., dummies) can be included using `deterministic()`. Multiple deterministic variables can be added with + (e.g., `deterministic(dummy1` |

+ dummy2)). These variables are considered fixed and are not associated with short-run or long-run dynamics. - Asymmetric variables can be included for both short-run and long-run dynamics:

- **asymS**: Specifies short-run asymmetric variables. For example, asymS(x1 + x2) includes variables x1 and x2 for short-run asymmetry.
- **asymL**: Specifies long-run asymmetric variables. For example, asymL(x1 + x3) includes variables x1 and x3 for long-run asymmetry.
- **asym**: Includes variables for both short-run and long-run asymmetry. For example, asym(x1 + x3) applies asymmetric decomposition for both dynamics.

A **trend** term can be added to the model to account for deterministic linear trends over time. Simply include trend in the formula.

These components can be combined flexibly in the formula to define a robust model tailored to your analysis.

case

Numeric or character. Specifies the case of the test to be used in the function. Acceptable values are 1, 2, 3, 4, 5, and "auto". If "auto" is chosen, the function determines the case automatically based on the model's characteristics. Invalid values will result in an error.

- 1: No intercept and no trend
- 2: Restricted intercept and no trend
- 3: Unrestricted intercept and no trend
- 4: Unrestricted intercept and restricted trend
- 5: Unrestricted intercept and unrestricted trend

signif_level

Character or numeric. Specifies the significance level to be used in the function. Acceptable values are "auto", "0.10", "0.1", "0.05", "0.025", and "0.01". If a numeric value is provided, it will be converted to a character string. If "auto" is chosen, the function determines the significance level automatically. Invalid values will result in an error.

maxlag

An integer specifying the maximum number of lags to be considered for the model. The default value is 4. This parameter sets an upper limit on the lag length during the model estimation process.

***details***

The maxlag parameter is crucial for defining the maximum lag length that the model will evaluate when selecting the optimal lag structure based on the specified criterion. It controls the computational effort and helps prevent overfitting by restricting the search space for lag selection.

- If the data has a short time horizon or is prone to overfitting, consider reducing maxlag. -
- If the data is expected to have long-term dependencies, increasing maxlag may be necessary to capture the relevant dynamics.

Setting an appropriate value for maxlag depends on the nature of your dataset and the context of the analysis:

- For small datasets or quick tests, use smaller values (e.g., maxlag = 2).

- For datasets with more observations or longer-term patterns, larger values (e.g., `maxlag = 8`) may be appropriate, though this increases computational time.

***examples***

Using the default maximum lag (4)

```
kardl(data, MyFormula, maxlag = 4)
```

Reducing the maximum lag to 2 for faster computation

```
kardl(data, MyFormula, maxlag = 2)
```

Increasing the maximum lag to 8 for datasets with longer dependencies

```
kardl(data, MyFormula, maxlag = 8)
```

`mode` Specifies the mode of estimation and output control. This parameter determines how the function handles lag estimation and what kind of feedback or control is provided during the process. The available options are:

- **"quick"** (default): Displays progress and messages in the console while the function estimates the optimal lag values. This mode is suitable for interactive use or for users who want to monitor the estimation process in real-time. It provides detailed feedback for debugging or observation but may use additional resources due to verbose output.
- **"grid"** : Displays progress and messages in the console while the function estimates the optimal lag values. This mode is suitable for interactive use or for users who want to monitor the estimation process in real-time. It provides detailed feedback for debugging or observation but may use additional resources due to verbose output.
- **"grid_custom"**: Suppresses most or all console output, prioritizing faster execution and reduced resource usage on PCs or servers. This mode is recommended for high-performance scenarios, batch processing, or when the estimation process does not require user monitoring. Suitable for large-scale or repeated runs where output is unnecessary.
- **User-defined vector**: A numeric vector of lag values specified by the user, allowing full customization of the lag structure used in model estimation. When a user-defined vector is provided (e.g., 'c(1, 2, 4, 5)'), the function skips the lag optimization process and directly uses the specified lags.
  - Users can define lag values directly as a numeric vector. For example: `mode = c(1, 2, 4, 5)` assigns lags of 1, 2, 4, and 5 to variables in the specified order. - Alternatively, lag values can be assigned to variables by name for clarity and control. For example: `mode = c(CPI = 2, ER_POS = 3, ER_NEG = 1, PPI = 3)` assigns lags to variables explicitly. - Ensure that the lags are correctly designated by verifying the result using `kardl_model$properLag` after estimation.
  ***Attention!*** -A function-based criterion or user-defined function can be specified for model selection, but this is only supported for `mode = "grid_custom"` and `mode = "quick"`. The `mode = "grid"` option is restricted to predefined criteria (e.g., AIC or BIC). For more information on available criteria, see the `modelCriterion` function documentation. - When using a numeric vector, ensure the order of lag values matches the variables in your formula. - If using named vectors, double-check the variable names to avoid mismatches

or unintended results. - This mode bypasses the automatic lag optimization and assumes the user-defined lags are correct.

The ‘mode‘ parameter provides flexibility for different use cases: - Use ‘"grid"‘ mode for debugging or interactive use where progress visibility is important. - Use ‘"grid_custom"‘ mode to minimize overhead in computationally intensive tasks. - Specify a user-defined vector to customize the lag structure based on prior knowledge or analysis.

Selecting the appropriate mode can improve the efficiency and usability of the function depending on the user's requirements and the computational environment.

`criterion` A string specifying the information criterion to be used for selecting the optimal lag structure. The available options are:

- **"AIC"**: Akaike Information Criterion (default). This criterion balances model fit and complexity, favoring models that explain the data well with fewer parameters.
- **"BIC"**: Bayesian Information Criterion. This criterion imposes a stronger penalty for model complexity than AIC, making it more conservative in selecting models with fewer parameters.
- **"AICc"**: Corrected Akaike Information Criterion. This is an adjusted version of AIC that accounts for small sample sizes, making it more suitable when the number of observations is limited relative to the number of parameters.
- **"HQ"**: Hannan-Quinn Information Criterion. This criterion provides a compromise between AIC and BIC, favoring models that balance fit and complexity without being overly conservative.

The criterion can be specified as a string (e.g., `"AIC"`) or as a user-defined function that takes a fitted model object. Please visit the `modelCriterion` function documentation for more details on using custom criteria.

`differentAsymLag` A logical value indicating whether to allow different lag lengths for positive and negative decompositions.

`batch` A string specifying the batch processing configuration in the format "current_batch/total_batches". If a user utilize grid or grid_custom mode and want to split the lag search into multiple batches, this parameter can be used to define the current batch and the total number of batches. For example, "2/5" indicates that the current batch is the second out of a total of five batches. The default value is "1/1", meaning that the entire lag search is performed in a single batch.

`...` Additional arguments that can be passed to the function. These arguments can be used to

## Value

A list containing the results of the PSS t Bound test and recm, including:

- `type`: The type of test performed, which is "cointegration".
- `case`: The case number used in the test (1, 2, 3, 4, or 5).

- `statistic`: The t-statistic value calculated from the test.
- `k`: The number of long-run variables in the model.
- `Cont`: The conclusion of the test, indicating whether cointegration is present, inconclusive, or absent.
- `BoundNum`: A numeric representation of the conclusion, where 1 indicates cointegration, 0 indicates inconclusive, and -1 indicates no cointegration.
- `siglvl`: The significance level used in the test, either "auto" or one of the specified numeric levels.
- `criticalValues`: A vector of critical values for the test, corresponding to the significance levels.
- `parameter`: The names of the long-run variables in the model.
- `coef`: The estimated coefficient of the error correction term.
- `FH0`: The null hypothesis of the test, which includes the long-run adjustment coefficient set to zero.
- `longrunEQ`: The long-run equation used in the test.
- `shortrunEQ`: The short-run equation used in the test.
- `ecmL`: The linear model fitted to the long-run equation.
- `ecmS`: The short-run model fitted to the error correction term.
- `ecmResiduals`: The residuals from the long-run model.
- `EcmResLagged`: The lagged residuals from the long-run model.
- `finalModel`: The final model used in the test, which includes the error correction model.
- `OptLag`: The optimal lag length determined for the model.
- `warnings`: Any warnings generated during the test, such as sample size concerns.
- `method`: The method used for the test, which is "recmt".

**Hypothesis testing**

The restricted ECM test, also known as the PSS t Bound test, is a statistical test used to assess the presence of cointegration in a model. Cointegration refers to a long-term equilibrium relationship between two or more time series variables. The PSS t Bound test is based on the work of Pesaran, Shin, and Smith (2001) and is particularly useful for models with small sample sizes.

The null and alternative hypotheses for the restricted ECM test are as follows:

$$\mathbf{H_0}: \theta = 0$$
$$\mathbf{H_1}: \theta \neq 0$$

The null hypothesis ($H_0$) states that there is no cointegration in the model, meaning that the long-run relationship between the variables is not significant. The alternative hypothesis ($H_1$) suggests that there is cointegration, indicating a significant long-term relationship between the variables.

The test statistic is calculated as the t-statistic of the coefficient of the error correction term ($\theta$) in the ECM model. If the absolute value of the t-statistic exceeds the critical value from the PSS t Bound table, we reject the null hypothesis in favor of the alternative hypothesis, indicating that cointegration is present.

The cases for the restricted ECM Bound test are defined as follows:

- `case 1`: No constant, no trend.

  This case is used when the model does not include a constant term or a trend term. It is suitable for models where the variables are stationary and do not exhibit any long-term trends.

  The model is specified as follows:

$$\Delta y_t = \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + \theta(y_{t-1} - \sum_{i=1}^{k} \alpha_i x_{i,t-1}) + e_t$$

- `case 2`: Restricted constant, no trend.

  This case is used when the model includes a constant term but no trend term. It is suitable for models where the variables exhibit a long-term relationship but do not have a trend component. The model is specified as follows:

$$\Delta y_t = \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + \theta(y_{t-1} - \alpha_0 - \sum_{i=1}^{k} \alpha_i x_{i,t-1}) + e_t$$

- `case 3`: Unrestricted constant, no trend.

  This case is used when the model includes an unrestricted constant term but no trend term. It is suitable for models where the variables exhibit a long-term relationship with a constant but do not have a trend component.

  The model is specified as follows:

$$\Delta y_t = \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + \theta(y_{t-1} - \alpha_0 - \sum_{i=1}^{k} \alpha_i x_{i,t-1}) + e_t$$

- `case 4`: Unrestricted Constant, restricted trend.

  This case is used when the model includes an unrestricted constant term and a restricted trend term. It is suitable for models where the variables exhibit a long-term relationship with a constant and a trend component.

  The model is specified as follows:

$$\Delta y_t = \phi + \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + \theta(y_{t-1} - \pi(t-1) - \sum_{i=1}^{k} \alpha_i x_{i,t-1}) + e_t$$

- `case 5`: Unrestricted constant, unrestricted trend.

The Error Correction Model (ECM) is specified as follows:

$$\Delta y_t = \phi + \varphi t + \sum_{j=1}^{p} \gamma_j \Delta y_{t-j} + \sum_{i=1}^{k} \sum_{j=0}^{q_i} \beta_{ij} \Delta x_{i,t-j} + \theta(y_{t-1} - \sum_{i=1}^{k} \alpha_i x_{i,t-1}) + e_t$$

## See Also

`kardl` `pssf` `psst` `recmt` `narayan`

## Examples

```
# Sample article: THE DYNAMICS OF EXCHANGE RATE PASS-THROUGH TO DOMESTIC PRICES IN TURKEY
library(magrittr)
kardl_set(model=CPI~ER+PPI+asym(ER)+deterministic(covid)+trend ,
          data=imf_example_data ,
          maxlag=3)

recmt_model_grid<-recmt(mode = "grid")
recmt_model_grid
recmt_model<- imf_example_data %>% recmt(mode = "grid_custom")
recmt_model
recmt_model2<-recmt(mode = c( 2    ,  1    ,  1   ,   3 ))
# Getting the results
recmt_model2
# Getting the summary of the results
summary(recmt_model2)
# OR
imf_example_data %>% recmt(CPI~PPI+asym(ER) +trend,case=4) %>% summary()

# For increasing the performance of finding the most fitted lag vector
recmt(mode = "grid_custom")
# Setting max lag instead of default value [4]
recmt(maxlag = 2, mode = "grid_custom")
# Using another criterion for finding the best lag
recmt(criterion = "HQ", mode = "grid_custom")



# summary( myNewStarSigns)
# For using different lag values for negative and positive decompositions of non-linear variables

# setting the same lags for positive and negative decompositions.
kardl_set(differentAsymLag = FALSE)

diffAsymLags<-recmt( mode = "grid_custom")
diffAsymLags$OptLag

# setting the different lags for positive and negative decompositions
sameAsymLags<-recmt(differentAsymLag = TRUE , mode = "grid_custom" )
sameAsymLags$OptLag


# Setting the preffixes and suffixes for non-linear variables
kardl_reset()
kardl_set(AsymPrefix = c("asyP_","asyN_"), AsymSuffix = c("_PP","_NN"))
customizedNames<-recmt(imf_example_data, CPI~ER+PPI+asym(ER) )
customizedNames$ecmS$finalModel$model

# For having the lags plot
library(ggplot2)
library(dplyr)
```

```
#  recmt_model_grid[["LagCriteria"]] is a matrix, convert it to a data frame
LagCriteria <- as.data.frame(recmt_model_grid$ecmS$LagCriteria)
# Rename columns for easier access and convert relevant columns to numeric
colnames(LagCriteria) <- c("lag", "AIC", "BIC", "AICc", "HQ")
LagCriteria <- LagCriteria %>%  mutate(across(c(AIC, BIC, HQ), as.numeric))

# Pivot the data to a long format excluding AICc
library(tidyr)

LagCriteria_long <- LagCriteria %>%  select(-AICc) %>%
pivot_longer(cols = c(AIC, BIC, HQ), names_to = "Criteria", values_to = "Value")
# Find the minimum value for each criterion
min_values <- LagCriteria_long %>%  group_by(Criteria) %>%
  slice_min(order_by = Value) %>%  ungroup()

# Create the ggplot with lines, highlight minimum values, and add labels
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 = 3.5) +
 labs(title = "Lag Criteria Comparison", x = "Lag Configuration",  y = "Criteria Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

---

`replaceValues` *Replace Patterns with Evaluated Values*

---

## Description

The `replaceValues()` function allows you to replace placeholders in text with variable values or evaluated expressions. Placeholders are marked by {} brackets, and the function supports capturing both global and non-global variables, making it useful for templating and dynamic string generation in R.

## Usage

```
replaceValues(original_text, x = FALSE, capture_warnings = TRUE)
```

## Arguments

| | |
|---|---|
| `original_text` | A string containing expressions within `{}` to be evaluated and replaced. |
| `x` | Non-global variables to be evaluated within the function's context. |
| `capture_warnings` | Logical. If `TRUE`, captures errors and warnings during evaluation and continues with replacements. If `FALSE`, errors and warnings are not captured. |

## Value

A string with all placeholders replaced by evaluated values.

## See Also

eval, parse, gregexpr, regmatches

## Examples

```
 # Basic example with global variables
orjTXT <- "My F value is {thisF} and  its criterion is {CrF}"
thisF <- 2.33
CrF <- 32.43
replaceValues(orjTXT)
cat(replaceValues("The summary ls: {summary(lm(mpg ~ wt + hp, data = mtcars))}\n This is example"))

 # Nested function example for non-global variables
globalY <- 10
bb <- function(x){
  y <-  x+2;
 secondFunc <- function(x){
   replaceValues("The global value of Y+3 is {globalY+3}.  The non global value is {x}",x)
  }
  secondFunc(y)
}
bb(5)


 # Nested variables example. Something like $$ in PHP
FirstVar <- "First"
SecondVar <- "Second"
FirstSecondVar <- "_____________"
orjTXT <- "First var is {FirstVar}. The combinations of var names \"FirstSecondVar\" "
orjTXT <- paste0(orjTXT,"is= {{FirstVar}{SecondVar}Var}. Where SecondVar = {SecondVar} .")
replaceValues( orjTXT)

# Using mathematical expressions
MyVar = 3
replaceValues( "First var multiplying to 12 is {MyVar*12} and its power is {MyVar^12}.")

# Using templates from external files
# file_contents <- readLines(system.file("template", templateName, package = getPackageName()))
file_contents <- readLines(system.file("template", "printkardl.txt", package = "kardl"))
data(imf_example_data)
 MyFormula<-CPI~ER+PPI+asym(ER)+deterministic(covid)+trend
 kardl_model<-kardl(imf_example_data,MyFormula,mode="quick")
output<- replaceValues(file_contents,kardl_model)
cat(output)
```

`writemath` *Make Equations of used model in Latex or Office*

## Description

The **writemath** function generates all necessary equations for analysis, including those for ARDL, NARDL, short-run and long-run nonlinearities, and dynamic multipliers. It produces these equations in multiple formats: as text and as files compatible with MS Word, Markdown, LaTeX, and LibreOffice. This function provides a comprehensive set of formulas essential for econometric analysis, facilitating easy access to equations across various document formats for efficient reporting and presentation.

## Usage

```
writemath(formula_, output = 1, verbose = FALSE)
```

## Arguments

`formula_` The **formula_** parameter allows a user to specify a model formula, such as `CPI~ER+PPI+asym(ER)+deterministic(covid)+trend`, to define the relationships in the analysis. Alternatively, if a user provide a kardl output object, the function will automatically extract all necessary variables and parameters from this object, simplifying the setup process. This flexibility enables users to define the model structure either through a formula or directly from an existing analysis object.

`output` The output parameter specifies the path and file name for saving output equations.

- **MS Word Output**: If output is set with a **.docx** extension, the equations will be saved as a **Word** document.
- **LaTeX Output**: If specified with a **.tex** extension, the equations will be saved as a **LaTeX** file.
- **Markdown Output**: If specified with a **.rm** extension, the equations will be saved as a **Markdown** file.
- **PDF Output**: If specified with a **.pdf** extension, the equations will be saved as a **PDF** file.
- **Display on terminal**
  - **1**: Using **1** as the output displays **LaTeX** formatted equations on the screen without saving to a file.
  - **2**: Setting output to **2** displays equations in **Markdown** on terminal.
  - **3**: Setting output to **3** displays the equations as **OpenOffice**-compatible math formulas on terminal.

If *output* is specified without directory paths (i.e., without / for Linux or \\ for Windows), the file will be saved in the current working directory. This feature ensures that, in the absence of a specified path, files are saved to a known default location.

See examples for details

verbose — A logical value indicating whether to print the generated formula to the console when not saving to a file. If set to TRUE, the function will display the formula; if FALSE, it will not print anything. This parameter is useful for users who want to see the output directly in the console without saving it to a file.

## Value

print or save

## See Also

kardl, kardl_longrun

## Examples

```
# Note: The file creations in this examples were deactivated in the current examples
# because of CRAN policies.

# For saving all equations in a MS word document, \strong{docx}.
# The directory can be added alongside of the file name.
# For Windows OS directories could be written as \ not /.

# Here the tempdir() function is used to get a temporary directory path.
# Users can replace this with their desired directory path.


writemath(y~x+z+asym(v)+asymL(q+a),tempfile(fileext = ".docx"))


# For Markdown file the extenssion of the file should be assigned as \strong{md}.

writemath(y~x+z+asym(v+w)+asymL(q+a)+asymS(m),tempfile(fileext = ".md"))


# All equations are able to be saved in a PDF file.

writemath(y~x+z+asym(v+w)+asymL(q+a)+asymS(m),tempfile(fileext = ".pdf"))




# Following commad saves LibreOffice formulas in terminal.

# Copy and paste formulas there and after selecting related formula click on formula icon.

kardl_model<-kardl(imf_example_data,CPI~ER+PPI+asym(ER)+deterministic(covid)+trend,mode=c(1,2,3,0))

# Following command writes all equations in a latex file in current directory.

# Note: The file creations in this examples were deactivated because of CRAN policies.
```

```
tmp <- tempfile(fileext = ".tex")
writemath(kardl_model,tmp)


# For printing the equations in Latex fomrat assignning of file name to 1 is required.
# A user's formula can be written directly in the function.
eq1 <- writemath(y~x+z+asym(v+w)+asymL(q+a)+asymS(m),1)

# To get all of the N/ARDL equations (in LaTeX format):
# Note: Here the output is limited to 1000 characters for better visibility.
print(eq1,1000)




# To saving outputs for different output type, in a list is available.
eq2 <- writemath(y~x+z+asym(v+w)+asymL(q+a)+asymS(m),1)
# To get the equations' subcomponents, here the long-run equation (in LaTeX format):
eq2$formulas$longRunEqFormula

# For printing the equations in Markdown fomrat assignning of file name to 2 is required.
eq3<- writemath(y~x+z+asym(v+w)+asymL(q+a)+asymS(m),2)
# To get the ECM equation (in Markdown format):
eq3$formulas$ECMEqFormula

# For printing the equations in OpenOffice fomrat assignning of file name to 3 is required.
eq4<-writemath(y~x+z+asym(v+w)+asymL(q+a)+asymS(m),3)
# To get the NARDL equation (in LibreOffice format):
eq4$formulas$NARDL_eq_Formula
```

# Index