# Package ‘ARDL’

April 10, 2020

**Type** Package

**Title** ARDL, ECM and Bounds-Test for Cointegration

**Description** Creates complex autoregressive distributed lag (ARDL) models providing just the order and automatically constructs the underlying unrestricted and restricted error correction model (ECM). It also performs the bounds-test for cointegration as described in Pesaran et al. (2001) <doi:10.1002/jae.616> and provides the multipliers and the cointegrating equation.

**Version** 0.1.0

**License** GPL-3

**URL** https://github.com/Natsiopoulos/ARDL

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.2.0)

**Suggests** qpcR, sandwich, xts

**Imports** aod, dplyr, dynlm, lmtest, msm, stringr, zoo

**RoxygenNote** 7.1.0

**NeedsCompilation** no

**Author** Kleanthis Natsiopoulos [aut, cre, dis] (<https://orcid.org/0000-0003-1180-2984>), Nickolaos Tzeremes [ths] (<https://orcid.org/0000-0002-6938-3404>), University of Thessaly, Department of Economics [dgg]

**Maintainer** Kleanthis Natsiopoulos <klnatsio@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-04-10 10:10:06 UTC

## R topics documented:

---

`ardl` *ARDL model regression*

---

## Description

A simple way to construct complex ARDL specifications providing just the model order additional to the model formula. It uses `dynlm` under the hood.

## Usage

```
ardl(formula, data, order, start = NULL, end = NULL, ...)
```

## Arguments

| | |
|---|---|
| `formula` | A "formula" describing the linear model. Details for model specification are given under 'Details'. |
| `data` | A time series object (e.g., "ts", "zoo" or "zooreg") or a data frame containing the variables in the model. In the case of a data frame, it is coerced into a `ts` object with `start = 1`, `end = nrow(data)` and `frequency = 1`. If not found in data, the variables are NOT taken from any environment. |
| `order` | A specification of the order of the ARDL model. A numeric vector of the same length as the total number of variables (excluding the fixed ones, see 'Details'). It should only contain positive integers or 0. An integer could be provided if all variables are of the same order. |
| `start` | Start of the time period which should be used for fitting the model. |
| `end` | End of the time period which should be used for fitting the model. |
| `...` | Additional arguments to be passed to the low level regression fitting functions. |

## Details

The `formula` should contain only variables that exist in the data provided through `data` plus some additional functions supported by `dynlm` (i.e., `trend()`).

You can also specify fixed variables that are not supposed to be lagged (e.g. dummies etc.) simply by placing them after `|`. For example, `y ~ x1 + x2 | z1 + z2` where `z1` and `z2` are the fixed variables and should not be considered in `order`. Note that the `|` notion should not be confused with the same notion in `dynlm` where it introduces instrumental variables.

## Value

`ardl` returns an object of `class` `c("dynlm","lm","ardl")`. In addition, attributes 'order', 'data', 'parsed_formula' and 'full_formula' are provided.

## Mathematical Formula

The general form of an $ARDL(p, q_1, \ldots, q_k)$ is:

$$y_t = c_0 + c_1 t + \sum_{i=1}^{p} b_{y,i} y_{t-i} + \sum_{j=1}^{k} \sum_{l=0}^{q_j} b_{j,l} x_{j,t-l} + \epsilon_t$$

## Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

## See Also

`uecm`, `recm`

## Examples

```
data(denmark)

## Estimate an ARDL(3,1,3,2) model ------------------------------------

ardl_3132 <- ardl(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))
summary(ardl_3132)

## Add dummies or other variables that should stay fixed ---------------

d_74Q1_75Q3 <- ifelse(time(denmark) >= 1974 & time(denmark) <= 1975.5, 1, 0)

# the date can also be setted as below
d_74Q1_75Q3_ <- ifelse(time(denmark) >= "1974 Q1" & time(denmark) <= "1975 Q3", 1, 0)
identical(d_74Q1_75Q3, d_74Q1_75Q3_)
den <- cbind(denmark, d_74Q1_75Q3)
ardl_3132_d <- ardl(LRM ~ LRY + IBO + IDE | d_74Q1_75Q3,
                    data = den, order = c(3,1,3,2))
summary(ardl_3132_d)
compare <- data.frame(AIC = c(AIC(ardl_3132), AIC(ardl_3132_d)),
                      BIC = c(BIC(ardl_3132), BIC(ardl_3132_d)))
rownames(compare) <- c("no ummy", "with dummy")
compare

## Estimate an ARDL(3,1,3,2) model with a linear trend -----------------

ardl_3132_tr <- ardl(LRM ~ LRY + IBO + IDE + trend(LRM),
                     data = denmark, order = c(3,1,3,2))

# Alternative time trend specifications:
# time(LRM)                 1974 + (0, 1, ..., 55)/4 time(data)
```

```
# trend(LRM)                (1, 2, ..., 55)/4        (1:n)/freq
# trend(LRM, scale = FALSE) (1, 2, ..., 55)          1:n

## Subsample ARDL regression (start after 1975 Q4) ---------------------

ardl_3132_sub <- ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                      order = c(3,1,3,2), start = "1975 Q4")

# the date can also be setted as below
ardl_3132_sub2 <- ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                       order = c(3,1,3,2), start = c(1975,4))
identical(ardl_3132_sub, ardl_3132_sub2)
summary(ardl_3132_sub)

## Ease of use ---------------------------------------------------------

# The model specification of the ardl_3132 model can be created as easy as order=c(3,1,3,2)
# or else, it could be done using the dynlm package as:
library(dynlm)
m <- dynlm(LRM ~ L(LRM, 1) + L(LRM, 2) + L(LRM, 3) + LRY + L(LRY, 1) + IBO + L(IBO, 1) +
           L(IBO, 2) + L(IBO, 3) + IDE + L(IDE, 1) + L(IDE, 2), data = denmark)
identical(m$coefficients, ardl_3132$coefficients)

# The full formula can be extracted from the ARDL model, and this is equal to
ardl_3132$full_formula
m2 <- dynlm(ardl_3132$full_formula, data = ardl_3132$data)
identical(m$coefficients, m2$coefficients)
```

---

auto_ardl    *Automatic ARDL model selection*

---

## Description

It searches for the best ARDL order specification, according to the selected criterion, taking into account the constraints provided.

## Usage

```
auto_ardl(
  formula,
  data,
  max_order,
  fixed_order = -1,
  starting_order = NULL,
  selection = "AIC",
  selection_minmax = c("min", "max"),
  grid = FALSE,
  search_type = c("horizontal", "vertical"),
  start = NULL,
```

```
    end = NULL,
    ...
)
```

## Arguments

`formula` A "formula" describing the linear model. Details for model specification are given under 'Details' in the help file of the `ardl` function.

`data` A time series object (e.g., "ts", "zoo" or "zooreg") or a data frame containing the variables in the model. In the case of a data frame, it is coerced into a `ts` object with `start = 1`, `end = nrow(data)` and `frequency = 1`. If not found in data, the variables are NOT taken from any environment.

`max_order` It sets the maximum order for each variable where the search is taking place. A numeric vector of the same length as the total number of variables (excluding the fixed ones, see 'Details' in the help file of the `ardl` function). It should only contain positive integers. An integer could be provided if the maximum order for all variables is the same.

`fixed_order` It allows setting a fixed order for some variables. The algorithm will not search for any other order than this. A numeric vector of the same length as the total number of variables (excluding the fixed ones). It should contain positive integers or 0 to set as a constraint. A -1 should be provided for any variable that should not be constrained. `fixed_order` overrides the corresponding `max_order` and `starting_order`.

`starting_order` Specifies the order for each variable from which each search will start. It is a numeric vector of the same length as the total number of variables (excluding the fixed ones). It should contain positive integers or 0 or only one integer could be provided if the starting order for all variables is the same. Default is set to NULL. If unspecified (`NULL`) and `grid = FALSE`, then all possible $VAR(p)$ models are calculated (constraints are taken into account), where $p$ is the minimum value in `max_order`. Note that where `starting_order` is provided, its first element will be the minimum value of $p$ that the searching algorithm will consider (think of it like a 'minimum p order' restriction) (see 'Searching algorithm' below). If `grid = TRUE`, only the first argument ($p$) will have an effect.

`selection` A character string specifying the selection criterion according to which the candidate models will be ranked. Default is `AIC`. Any other selection criterion can be used (a user specified or a function from another package) as long as it can be applied as `selection(model)`. The preferred model is the one with the smaller value of the selection criterion. If the selection criterion works the other way around (the bigger the better), `selection_minmax = "max"` should also be supplied (see 'Examples' below).

`selection_minmax` A character string that indicates whether the criterion in `selection` is supposed to be minimized (default) or maximized.

`grid` If `FALSE` (default), the stepwise searching regression algorithm will search for the best model by adding and subtracting terms corresponding to different ARDL orders. If `TRUE`, the whole set of all possible ARDL models (accounting for constraints) will be evaluated. Note that this method can be very time-consuming in case that max_order is big and there are many independent variables that create a very big number of possible combinations.

search_type — A character string describing the search type. If "horizontal" (default), the searching algorithm increases or decreases by 1 the order of each variable in each iteration. When the order of the last variable has been accessed, it begins again from the first variable until it converges. If "vertical", the searching algorithm increases or decreases by 1 the order of a variable until it converges. Then it continues the same for the next variable. The two options result to very similar top orders. The default ("horizontal"), sometimes is a little more accurate, but the "vertical" is almost 2 times faster. Not applicable if grid = TRUE.

start — Start of the time period which should be used for fitting the model.

end — End of the time period which should be used for fitting the model.

... — Additional arguments to be passed to the low level regression fitting functions.

## Value

auto_ardl returns a list which contains:

best_model — An object of class c("dynlm","lm","ardl")

best_order — A numeric vector with the order of the best model selected

top_orders — A data.frame with the orders of the top 20 models

## Searching algorithm

The algorithm performs the optimization process starting from multiple starting points concerning the autoregressive order $p$. The searching algorithm will perform a complete search, each time starting from a different starting order. These orders are presented in the tables below, for grid = FALSE and different values of starting_order.

starting_order = NULL:

| | | | | | | |
|---|---|---|---|---|---|---|
| VAR(p) | -> | p | q1 | q2 | ... | qk |
| VAR(1) | -> | 1 | 1 | 1 | ... | 1 |
| VAR(2) | -> | 2 | 2 | 2 | ... | 2 |
| : | -> | : | : | : | : | : |
| VAR(P) | -> | P | P | P | ... | P |

starting_order = c(3,0,1,2):

| | | | |
|---|---|---|---|
| p | q1 | q2 | q3 |
| 3 | 0 | 1 | 2 |
| 4 | 0 | 1 | 2 |
| : | : | : | : |
| P | 0 | 1 | 2 |

## Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

## See Also

ardl

## Examples

```
data(denmark)

## Find the best ARDL order --------------------------------------------

# Up to 5 for the autoregressive order (p) and 4 for the rest (q1, q2, q3)

# Using the defaults search_type = "vertical", grid = FALSE and selection = "AIC"
# ("Not run" indications only for testing purposes)
## Not run:
model1 <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                    max_order = c(5,4,4,4))
model1$top_orders

## Same, with search_type = "horizontal" -------------------------------

model1_h <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                      max_order = c(5,4,4,4), search_type = "horizontal")
model1_h$top_orders

## Find the global optimum ARDL order ----------------------------------

# It may take more than 10 seconds
model_grid <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                        max_order = c(5,4,4,4), grid = TRUE)

## Different selection criteria ----------------------------------------

# Using BIC as selection criterion instead of AIC
model1_b <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                      max_order = c(5,4,4,4), selection = "BIC")
model1_b$top_orders

# Using other criteria like adjusted R squared (the bigger the better)
adjr2 <- function(x) { summary(x)$adj.r.squared }
model1_adjr2 <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                           max_order = c(5,4,4,4), selection = "adjr2",
                           selection_minmax = "max")
model1_adjr2$top_orders

# Using functions from other packages as selection criteria
if (requireNamespace("qpcR", quietly = TRUE)) {

library(qpcR)
model1_aicc <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                          max_order = c(5,4,4,4), selection = "AICc")
model1_aicc$top_orders
adjr2 <- function(x){ Rsq.ad(x) }
```

```
model1_adjr2 <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                          max_order = c(5,4,4,4), selection = "adjr2",
                          selection_minmax = "max")
model1_adjr2$top_orders

## DIfferent starting order --------------------------------------------

# The searching algorithm will start from the following starting orders:
# p q1 q2 q3
# 1 1  3  2
# 2 1  3  2
# 3 1  3  2
# 4 1  3  2
# 5 1  3  2

model1_so <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                       max_order = c(5,4,4,4), starting_order = c(1,1,3,2))

# Starting from p=3 (don't search for p=1 and p=2)
# Starting orders:
# p q1 q2 q3
# 3 1  3  2
# 4 1  3  2
# 5 1  3  2

model1_so_3 <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                       max_order = c(5,4,4,4), starting_order = c(3,1,3,2))

# If starting_order = NULL, the starting orders for each iteration will be:
# p q1 q2 q3
# 1 1  1  1
# 2 2  2  2
# 3 3  3  3
# 4 4  4  4
# 5 5  5  5
}

## Add constraints -----------------------------------------------------

# Restrict only the order of IBO to be 2
model1_ibo2 <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                       max_order = c(5,4,4,4), fixed_order = c(-1,-1,2,-1))
model1_ibo2$top_orders

# Restrict the order of LRM to be 3 and the order of IBO to be 2
model1_lrm3_ibo2 <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
                       max_order = c(5,4,4,4), fixed_order = c(3,-1,2,-1))
model1_lrm3_ibo2$top_orders

## Set the starting date for the regression (data starts at "1974 Q1") -

# Set regression starting date to "1976 Q1"
model1_76q1 <- auto_ardl(LRM ~ LRY + IBO + IDE, data = denmark,
```

```
                                max_order = c(5,4,4,4), start = "1976 Q1")
start(model1_76q1$best_model)

## End(Not run)
```

---

bounds_f_test    *Bounds Wald-test for no cointegration*

---

## Description

bounds_f_test performs the Wald bounds-test for no cointegration *Pesaran et al. (2001)*. It is a Wald test on the parameters of a UECM (Unrestricted Error Correction Model) expressed either as a Chisq-statistic or as an F-statistic.

## Usage

```
bounds_f_test(
  object,
  case,
  alpha = NULL,
  pvalue = TRUE,
  exact = FALSE,
  R = 40000,
  test = c("F", "Chisq"),
  vcov_matrix = NULL
)
```

## Arguments

| | |
|---|---|
| object | An object of class 'ardl' or 'uecm'. |
| case | An integer from 1-5 or a character string specifying whether the 'intercept' and/or the 'trend' have to participate in the short-run or the long-run relationship (cointegrating equation) (see section 'Cases' below). |
| alpha | A numeric value between 0 and 1 indicating the significance level of the critical value bounds. If NULL (default), no critical value bounds for a specific level of significance are provide, only the p-value. See section 'alpha, bounds and p-value' below for details. |
| pvalue | A logical indicating whether you want the p-value to be provided. The default is TRUE. See section 'alpha, bounds and p-value' below for details. |
| exact | A logical indicating whether you want asymptotic (T = 1000) or exact sample size critical value bounds and p-value. The default is FALSE for asymptotic. See section 'alpha, bounds and p-value' below for details. |
| R | An integer indicating how many iterations will be used if exact = TRUE. Default is 40000. |
| test | A character vector indicating whether you want the Wald test to be expressed as 'F' or as 'Chisq' statistic. Default is "F". |

vcov_matrix
: The estimated covariance matrix of the random variable that the test uses to estimate the test statistic. The default is vcov(object) (when vcov_matrix = NULL), but other estimations of the covariance matrix of the regression's estimated coefficients can also be used (e.g., using vcovHC or vcovHAC). Only applicable if the input object is of class "uecm".

### Value

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

| | |
|---|---|
| method | a character string indicating what type of test was performed. |
| alternative | a character string describing the alternative hypothesis. |
| statistic | the value of the test statistic. |
| null.value | the value of the population parameters k (the number of independent variables) and T (the number of observations) specified by the null hypothesis. |
| data.name | a character string giving the name(s) of the data. |
| parameters | numeric vector containing the critical value bounds. |
| p.value | the p-value of the test. |
| tab | data.frame containing the statistic, the critical value bounds, the alpha level of significance and the p-value. |

### Hypothesis testing

$$\Delta y_t = c_0 + c_1 t + \pi_y y_{t-1} + \sum_{j=1}^{k} \pi_j x_{j,t-1} + \sum_{i=1}^{p-1} \psi_{y,i} \Delta y_{t-i} + \sum_{j=1}^{k} \sum_{l=1}^{q_j-1} \psi_{j,l} \Delta x_{j,t-l} + \sum_{j=1}^{k} \omega_j \Delta x_{j,t} + \epsilon_t$$

$$\mathbf{H_0} : \pi_y = \pi_1 = \ldots = \pi_k = 0$$

$$\mathbf{H_1} : \pi_y \neq \pi_1 \neq \ldots \neq \pi_k \neq 0$$

### alpha, bounds and p-value

In this section it is explained how the critical value bounds and p-values are obtained.

- If exact = FALSE, then the asymptotic (T = 1000) critical value bounds and p-value are provided.
- Only the asymptotic critical value bounds and p-values, and only for k <= 10 are precalculated, everything else has to be computed.
- Precalculated critical value bounds and p-values were simulated using set.seed(2020) and R = 70000.
- Precalculated critical value bounds exist only for alpha being one of the 0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.15 or 0.2, everything else has to be computed.
- If alpha is one of the 0.1, 0.05, 0.025 or 0.01 (and exact = FALSE and k <= 10), the critical value bounds are those presented in *Pesaran et al. (2001)*.

## Cases

According to *Pesaran et al. (2001)*, we distinguish the long-run relationship (cointegrating equation) (and thus the bounds-test and the Restricted ECMs) between 5 different cases. These differ in terms of whether the 'intercept' and/or the 'trend' are restricted to participate in the long-run relationship or they are unrestricted and so they participate in the short-run relationship.

**Case 1:**
- No *intercept* and no *trend*.
- `case` inputs: 1 or "n" where "n" stands for none.

**Case 2:**
- Restricted *intercept* and no *trend*.
- `case` inputs: 2 or "rc" where "rc" stands for restricted constant.

**Case 3:**
- Unrestricted *intercept* and no *trend*.
- `case` inputs: 3 or "uc" where "uc" stands for unrestricted constant.

**Case 4:**
- Unrestricted *intercept* and restricted *trend*.
- `case` inputs: 4 or "ucrt" where "ucrt" stands for unrestricted constant and restricted trend.

**Case 5:**
- Unrestricted *intercept* and unrestricted *trend*.
- `case` inputs: 5 or "ucut" where "ucut" stands for unrestricted constant and unrestricted trend.

Note that you can't restrict (or leave unrestricted) a parameter that doesn't exist in the input model. For example, you can't compute `recm(object,case=3)` if the object is an ARDL (or UECM) model with no intercept. The same way, you can't compute `bounds_f_test(object,case=5)` if the object is an ARDL (or UECM) model with no linear trend.

## References

Pesaran, M. H., Shin, Y., & Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326

## Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

## See Also

`bounds_t_test` `ardl` `uecm`

## Examples

```
data(denmark)

## How to use cases under different models (regarding deterministic terms)

## Construct an ARDL(3,1,3,2) model with different deterministic terms -

# Without constant
ardl_3132_n <- ardl(LRM ~ LRY + IBO + IDE -1, data = denmark, order = c(3,1,3,2))

# With constant
```

```
ardl_3132_c <- ardl(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))

# With constant and trend
ardl_3132_ct <- ardl(LRM ~ LRY + IBO + IDE + trend(LRM), data = denmark, order = c(3,1,3,2))

## F-bounds test for no level relationship (no cointegration) ----------

# For the model without a constant
bounds_f_test(ardl_3132_n, case = 1)
# or
bounds_f_test(ardl_3132_n, case = "n")

# For the model with a constant
# Including the constant term in the long-run relationship (restricted constant)
bounds_f_test(ardl_3132_c, case = 2)
# or
bounds_f_test(ardl_3132_c, case = "rc")

# Including the constant term in the short-run relationship (unrestricted constant)
bounds_f_test(ardl_3132_c, case = "uc")
# or
bounds_f_test(ardl_3132_c, case = 3)

# For the model with constant and trend
# Including the constant term in the short-run and the trend in the long-run relationship
# (unrestricted constant and restricted trend)
bounds_f_test(ardl_3132_ct, case = "ucrt")
# or
bounds_f_test(ardl_3132_ct, case = 4)

# For the model with constant and trend
# Including the constant term and the trend in the short-run relationship
# (unrestricted constant and unrestricted trend)
bounds_f_test(ardl_3132_ct, case = "ucut")
# or
bounds_f_test(ardl_3132_ct, case = 5)

## Note that you can't restrict a deterministic term that doesn't exist

# For example, the following tests will produce an error:
## Not run:
bounds_f_test(ardl_3132_c, case = 1)
bounds_f_test(ardl_3132_ct, case = 3)
bounds_f_test(ardl_3132_c, case = 4)

## End(Not run)

## Asymptotic p-value and critical value bounds (assuming T = 1000) ----

# Include critical value bounds for a certain level of significance

# F-statistic is larger than the I(1) bound (for a=0.05) as expected (p-value < 0.05)
bft <- bounds_f_test(ardl_3132_c, case = 2, alpha = 0.05)
```

```
bft
bft$tab

# F-statistic is slightly larger than the I(1) bound (for a=0.005)
# as p-value is slightly smaller than 0.005
bounds_f_test(ardl_3132_c, case = 2, alpha = 0.005)

## Exact sample size p-value and critical value bounds -----------------

# Setting a seed is suggested to allow the replication of results
# 'R' can be increased for more accurate resutls

# F-statistic is smaller than the I(1) bound (for a=0.01) as expected (p-value > 0.01)
# Note that the exact sample p-value (0.01285) is very different than the asymptotic (0.004418)
# It can take more than 30 seconds
## Not run:
set.seed(2020)
bounds_f_test(ardl_3132_c, case = 2, alpha = 0.01, exact = TRUE)

## End(Not run)

## "F" and "Chisq" statistics ------------------------------------------

# The p-value is the same, the test-statistic and critical value bounds are different but analogous
bounds_f_test(ardl_3132_c, case = 2, alpha = 0.01)
bounds_f_test(ardl_3132_c, case = 2, alpha = 0.01, test = "Chisq")
```

---

bounds_t_test *Bounds t-test for no cointegration*

---

## Description

bounds_t_test performs the t-bounds test for no cointegration *Pesaran et al. (2001).* It is a t-test on the parameters of a UECM (Unrestricted Error Correction Model).

## Usage

```
bounds_t_test(
  object,
  case,
  alpha = NULL,
  pvalue = TRUE,
  exact = FALSE,
  R = 40000,
  vcov_matrix = NULL
)
```

### Arguments

object
: An object of class 'ardl' or 'uecm'.

case
: An integer (1, 3 or 5) or a character string specifying whether the 'intercept' and/or the 'trend' have to participate in the short-run relationship (see section 'Cases' below). Note that the t-bounds test can't be applied for cases 2 and 4.

alpha
: A numeric value between 0 and 1 indicating the significance level of the critical value bounds. If `NULL` (default), no critical value bounds for a specific level of significance are provide, only the p-value. See section 'alpha, bounds and p-value' below for details.

pvalue
: A logical indicating whether you want the p-value to be provided. The default is `TRUE`. See section 'alpha, bounds and p-value' below for details.

exact
: A logical indicating whether you want asymptotic (T = 1000) or exact sample size critical value bounds and p-value. The default is `FALSE` for asymptotic. See section 'alpha, bounds and p-value' below for details.

R
: An integer indicating how many iterations will be used if `exact = TRUE`. Default is 40000.

vcov_matrix
: The estimated covariance matrix of the random variable that the test uses to estimate the test statistic. The default is `vcov(object)` (when `vcov_matrix = NULL`), but other estimations of the covariance matrix of the regression's estimated coefficients can also be used (e.g., using vcovHC or vcovHAC). Only applicable if the input object is of class "uecm".

### Value

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

method
: a character string indicating what type of test was performed.

alternative
: a character string describing the alternative hypothesis.

statistic
: the value of the test statistic.

null.value
: the value of the population parameters `k` (the number of independent variables) and `T` (the number of observations) specified by the null hypothesis.

data.name
: a character string giving the name(s) of the data.

parameters
: numeric vector containing the critical value bounds.

p.value
: the p-value of the test.

tab
: data.frame containing the statistic, the critical value bounds, the alpha level of significance and the p-value.

### Hypothesis testing

$$\Delta y_t = c_0 + c_1 t + \pi_y y_{t-1} + \sum_{j=1}^{k} \pi_j x_{j,t-1} + \sum_{i=1}^{p-1} \psi_{y,i} \Delta y_{t-i} + \sum_{j=1}^{k} \sum_{l=1}^{q_j-1} \psi_{j,l} \Delta x_{j,t-l} + \sum_{j=1}^{k} \omega_j \Delta x_{j,t} + \epsilon_t$$

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

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

### alpha, bounds and p-value

In this section it is explained how the critical value bounds and p-values are obtained.

- If `exact = FALSE`, then the asymptotic (T = 1000) critical value bounds and p-value are provided.
- Only the asymptotic critical value bounds and p-values, and only for k <= 10 are precalculated, everything else has to be computed.
- Precalculated critical value bounds and p-values were simulated using `set.seed(2020)` and `R = 70000`.
- Precalculated critical value bounds exist only for `alpha` being one of the 0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.15 or 0.2, everything else has to be computed.
- If `alpha` is one of the 0.1, 0.05, 0.025 or 0.01 (and `exact = FALSE` and k <= 10), the critical value bounds are those presented in *Pesaran et al. (2001)*.

### Cases

According to *Pesaran et al. (2001)*, we distinguish the long-run relationship (cointegrating equation) (and thus the bounds-test and the Restricted ECMs) between 5 different cases. These differ in terms of whether the 'intercept' and/or the 'trend' are restricted to participate in the long-run relationship or they are unrestricted and so they participate in the short-run relationship.

**Case 1:**
- No *intercept* and no *trend*.
- `case` inputs: 1 or "n" where "n" stands for none.

**Case 2:**
- Restricted *intercept* and no *trend*.
- `case` inputs: 2 or "rc" where "rc" stands for restricted constant.

**Case 3:**
- Unrestricted *intercept* and no *trend*.
- `case` inputs: 3 or "uc" where "uc" stands for unrestricted constant.

**Case 4:**
- Unrestricted *intercept* and restricted *trend*.
- `case` inputs: 4 or "ucrt" where "ucrt" stands for unrestricted constant and restricted trend.

**Case 5:**
- Unrestricted *intercept* and unrestricted *trend*.
- `case` inputs: 5 or "ucut" where "ucut" stands for unrestricted constant and unrestricted trend.

Note that you can't restrict (or leave unrestricted) a parameter that doesn't exist in the input model. For example, you can't compute `recm(object,case=3)` if the object is an ARDL (or UECM) model with no intercept. The same way, you can't compute `bounds_f_test(object,case=5)` if the object is an ARDL (or UECM) model with no linear trend.

### References

Pesaran, M. H., Shin, Y., & Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326

### Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

### See Also

bounds_f_test ardl uecm

### Examples

```
data(denmark)

## How to use cases under different models (regarding deterministic terms)

## Construct an ARDL(3,1,3,2) model with different deterministic terms -

# Without constant
ardl_3132_n <- ardl(LRM ~ LRY + IBO + IDE -1, data = denmark, order = c(3,1,3,2))

# With constant
ardl_3132_c <- ardl(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))

# With constant and trend
ardl_3132_ct <- ardl(LRM ~ LRY + IBO + IDE + trend(LRM), data = denmark, order = c(3,1,3,2))

## t-bounds test for no level relationship (no cointegration) ----------

# For the model without a constant
bounds_t_test(ardl_3132_n, case = 1)
# or
bounds_t_test(ardl_3132_n, case = "n")

# For the model with a constant
# Including the constant term in the short-run relationship (unrestricted constant)
bounds_t_test(ardl_3132_c, case = "uc")
# or
bounds_t_test(ardl_3132_c, case = 3)

# For the model with constant and trend
# Including the constant term and the trend in the short-run relationship
# (unrestricted constant and unrestricted trend)
bounds_t_test(ardl_3132_ct, case = "ucut")
# or
bounds_t_test(ardl_3132_ct, case = 5)

## Note that you can't use bounds t-test for cases 2 and 4, or use a wrong model

# For example, the following tests will produce an error:
## Not run:
bounds_t_test(ardl_3132_n, case = 2)
bounds_t_test(ardl_3132_c, case = 4)
bounds_t_test(ardl_3132_ct, case = 3)

## End(Not run)

## Asymptotic p-value and critical value bounds (assuming T = 1000) ----
```

```
# Include critical value bounds for a certain level of significance

# t-statistic is larger than the I(1) bound (for a=0.05) as expected (p-value < 0.05)
btt <- bounds_t_test(ardl_3132_c, case = 3, alpha = 0.05)
btt
btt$tab

# t-statistic doesn't exceed the I(1) bound (for a=0.005) as p-value is greater than 0.005
bounds_t_test(ardl_3132_c, case = 3, alpha = 0.005)

## Exact sample size p-value and critical value bounds -----------------

# Setting a seed is suggested to allow the replication of results
# 'R' can be increased for more accurate resutls

# t-statistic is smaller than the I(1) bound (for a=0.01) as expected (p-value > 0.01)
# Note that the exact sample p-value (0.009874) is very different than the asymptotic (0.005538)
# It can take more than 90 seconds
## Not run:
set.seed(2020)
bounds_t_test(ardl_3132_c, case = 3, alpha = 0.01, exact = TRUE)

## End(Not run)
```

---

`coint_eq` *Cointegrating equation (long-run level relationship)*

---

## Description

Creates the cointegrating equation (long-run level relationship) providing an 'ardl', 'uecm' or 'recm' model.

## Usage

```
coint_eq(object, case)

## S3 method for class 'recm'
coint_eq(object, ...)

## Default S3 method:
coint_eq(object, case)
```

## Arguments

| | |
|---|---|
| `object` | An object of `class` 'ardl', 'uecm' or 'recm'. |
| `case` | An integer from 1-5 or a character string specifying whether the 'intercept' and/or the 'trend' have to participate in the long-run level relationship (cointegrating equation) (see section 'Cases' below). If the input object is of class 'recm', case is not needed as the model is already under a certain case. |
| `...` | Currently unused argument. |

### Value

`coint_eq` returns an numeric vector containing the cointegrating equation.

### Cases

According to *Pesaran et al. (2001)*, we distinguish the long-run relationship (cointegrating equation) (and thus the bounds-test and the Restricted ECMs) between 5 different cases. These differ in terms of whether the 'intercept' and/or the 'trend' are restricted to participate in the long-run relationship or they are unrestricted and so they participate in the short-run relationship.

**Case 1:**
- No *intercept* and no *trend*.
- `case` inputs: 1 or "n" where "n" stands for none.

**Case 2:**
- Restricted *intercept* and no *trend*.
- `case` inputs: 2 or "rc" where "rc" stands for restricted constant.

**Case 3:**
- Unrestricted *intercept* and no *trend*.
- `case` inputs: 3 or "uc" where "uc" stands for unrestricted constant.

**Case 4:**
- Unrestricted *intercept* and restricted *trend*.
- `case` inputs: 4 or "ucrt" where "ucrt" stands for unrestricted constant and restricted trend.

**Case 5:**
- Unrestricted *intercept* and unrestricted *trend*.
- `case` inputs: 5 or "ucut" where "ucut" stands for unrestricted constant and unrestricted trend.

Note that you can't restrict (or leave unrestricted) a parameter that doesn't exist in the input model. For example, you can't compute `recm(object,case=3)` if the object is an ARDL (or UECM) model with no intercept. The same way, you can't compute `bounds_f_test(object,case=5)` if the object is an ARDL (or UECM) model with no linear trend.

### References

Pesaran, M. H., Shin, Y., & Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326

### Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

### See Also

`ardl` `uecm` `recm` `bounds_f_test` `bounds_t_test`

### Examples

```
data(denmark)
library(zoo) # for cbind.zoo()

## Estimate the Cointegrating Equation of an ARDL(3,1,3,2) model -------

# From an ARDL model (under case 2, restricted constant)
```

```
ardl_3132 <- ardl(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))
ce2_ardl <- coint_eq(ardl_3132, case = 2)

# From an UECM (under case 2, restricted constant)
uecm_3132 <- uecm(ardl_3132)
ce2_uecm <- coint_eq(uecm_3132, case = 2)

# From a RECM (under case 2, restricted constant)
# Notice that if a RECM has already been estimated under a certain case,
# the 'coint_eq()' can't be under different case, so no 'case' argument needed.
recm_3132 <- recm(uecm_3132, case = 2)
# The RECM is already under case 2, so the 'case' argument is no needed
ce2_recm <- coint_eq(recm_3132)

identical(ce2_ardl, ce2_uecm, ce2_recm)

## Check for a degenerate level relationship ---------------------------

# The bounds F-test under both cases reject the Null Hypothesis of no level relationship.
bounds_f_test(ardl_3132, case = 2)
bounds_f_test(ardl_3132, case = 3)

# The bounds t-test also rejects the NUll Hypothesis of no level relationship.
bounds_t_test(ardl_3132, case = 3)

# But when the constant enters the long-run equation (case 3)
# this becomes a degenerate relationship.
ce3_ardl <- coint_eq(ardl_3132, case = 3)
den <- cbind.zoo(LRM = denmark[,"LRM"], ce2_ardl, ce3_ardl)

if (requireNamespace("xts", quietly = TRUE)) {

library(xts)
den <- xts(den)
plot(den, legend.loc = "right")
plot(den[,-3], legend.loc = "right")

} else {

plot(den, col = c(1,2,3), screens = 1)
legend("right", lty = 1, legend = colnames(den), col = c(1:3))
plot(den[,-3], col = c(1,2), screens = 1)
legend("top", lty = 1, legend = colnames(den[,-3]), col = c(1:2))

}
```

## denmark *The Danish data on money income prices and interest rates*

## Description

This data set contains the series used by S. Johansen and K. Juselius for estimating a money demand function of Denmark.

## Usage

```
denmark
```

## Format

A data frame with 55 rows and 5 variables. Time period from 1974:Q1 until 1987:Q3.

**LRM** logarithm of real money, M2

**LRY** logarithm of real income

**LPY** logarithm of price deflator

**IBO** bond rate

**IDE** bank deposit rate

## Details

An object of class "zooreg" "zoo".

## Source

http://web.math.ku.dk/~sjo/data/danish_data.html

## References

Johansen, S. and Juselius, K. (1990), Maximum Likelihood Estimation and Inference on Cointegration – with Applications to the Demand for Money, *Oxford Bulletin of Economics and Statistics*, **52, 2**, 169–210.

---

`multipliers` *Multipliers estimation*

---

## Description

`multipliers` is a generic function used to estimate short-run (impact), interim and long-run (total) multipliers, along with their corresponding standard errors, t-statistics and p-values.

## Usage

```
multipliers(object, type = "lr", vcov_matrix = NULL)

## S3 method for class 'ardl'
multipliers(object, type = "lr", vcov_matrix = NULL)

## S3 method for class 'uecm'
multipliers(object, type = "lr", vcov_matrix = NULL)
```

## Arguments

| | |
|---|---|
| `object` | An object of `class` 'ardl' or 'uecm'. |
| `type` | A character string describing the type of multipliers. The default is "lr" for long-run (total) multipliers (see 'Details'). |
| `vcov_matrix` | The estimated covariance matrix of the random variable that the transformation function uses to estimate the standard errors (and so the t-statistics and p-values) of the multipliers. The default is `vcov(object)` (when `vcov_matrix = NULL`), but other estimations of the covariance matrix of the regression's estimated coefficients can also be used (e.g., using `vcovHC` or `vcovHAC`). |

## Details

The function invokes two different `methods`, one for objects of `class` 'ardl' and one for objects of `class` 'uecm'. This is because of the different (but equivalent) transformation functions that are used for each class/model ('ardl' and 'uecm') to estimate the multipliers.

Currently only the long-run (total) multipliers are supported (`type = "lr"`). Other choices, including short-run (impact) and interim multipliers are going to be available in future versions.

The delta method is used for approximating the standard errors (and thus the t-statistics and p-values) of the estimated multipliers.

## Value

`multipliers` returns a data.frame containing the independent variables (including possibly existing intercept or trend and excluding the fixed variables) and their corresponding standard errors, t-statistics and p-values.

## Mathematical Formula

**Long-Run Multipliers:**

**As derived from an ARDL:**

$$\mu = \frac{c_0}{1 - \sum_{i=1}^{p}}$$

$$\delta = \frac{c_1}{1 - \sum_{i=1}^{p}}$$

$$\theta_j = \frac{\sum_{l=0}^{q_j} b_{j,l}}{1 - \sum_{i=1}^{p}} \qquad \forall j = 1, \ldots, k$$

**As derived from an Unrestricted ECM:**

$$\mu = \frac{c_0}{-\pi_y}$$

$$\delta = \frac{c_1}{-\pi_y}$$

$$\theta_j = \frac{\pi_j}{-\pi_y} \qquad \forall j = 1, \ldots, k$$

### Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

### See Also

ardl, uecm

### Examples

```
data(denmark)

## Estimate the long-run multipliers of an ARDL(3,1,3,2) model ---------

# From an ARDL model
ardl_3132 <- ardl(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))
mult_ardl <- multipliers(ardl_3132)
mult_ardl

# From an UECM
uecm_3132 <- uecm(ardl_3132)
mult_uecm <- multipliers(uecm_3132)
mult_uecm

all.equal(mult_ardl, mult_uecm)
```

---

## recm *Restricted ECM regression*

---

### Description

Creates the Restricted Error Correction Model (RECM). This is the conditional RECM, which is the RECM of the underlying ARDL.

### Usage

```
recm(object, case)
```

### Arguments

| | |
|---|---|
| object | An object of class 'ardl' or 'uecm'. |
| case | An integer from 1-5 or a character string specifying whether the 'intercept' and/or the 'trend' have to participate in the short-run or the long-run relationship (cointegrating equation) (see section 'Cases' below). |

### Details

Note that the statistical significance of 'L(ect, 1)' in a RECM should not be tested using the corresponding t-statistic (or the p-value) because it doesn't follow a standard t-distribution. Instead, the bounds_t_test should be used.

### Value

`recm` returns an object of class `c("dynlm","lm","recm")`. In addition, attributes 'order', 'data', 'parsed_formula' and 'full_formula' are provided.

### Mathematical Formula

The formula of a Restricted ECM conditional to an $ARDL(p, q_1, \ldots, q_k)$ is:

$$\Delta y_t = c_0 + c_1 t + \sum_{i=1}^{p-1} \psi_{y,i} \Delta y_{t-i} + \sum_{j=1}^{k} \sum_{l=1}^{q_j-1} \psi_{j,l} \Delta x_{j,t-l} + \sum_{j=1}^{k} \omega_j \Delta x_{j,t} + \pi_y ECT_t + \epsilon_t$$

**Under Case 1:**
- $c_0 = c_1 = 0$
- $ECT = y_{t-1} - (\sum_{j=1}^{k} \theta_j x_{j,t-1})$

**Under Case 2:**
- $c_0 = c_1 = 0$
- $ECT = y_{t-1} - (\mu + \sum_{j=1}^{k} \theta_j x_{j,t-1})$

**Under Case 3:**
- $c_1 = 0$
- $ECT = y_{t-1} - (\sum_{j=1}^{k} \theta_j x_{j,t-1})$

**Under Case 4:**
- $c_1 = 0$
- $ECT = y_{t-1} - (\delta(t-1) + \sum_{j=1}^{k} \theta_j x_{j,t-1})$

**Under Case 5:**
- $ECT = y_{t-1} - (\sum_{j=1}^{k} \theta_j x_{j,t-1})$

### Cases

According to *Pesaran et al. (2001)*, we distinguish the long-run relationship (cointegrating equation) (and thus the bounds-test and the Restricted ECMs) between 5 different cases. These differ in terms of whether the 'intercept' and/or the 'trend' are restricted to participate in the long-run relationship or they are unrestricted and so they participate in the short-run relationship.

**Case 1:**
- No *intercept* and no *trend*.
- `case` inputs: 1 or "n" where "n" stands for none.

**Case 2:**
- Restricted *intercept* and no *trend*.
- `case` inputs: 2 or "rc" where "rc" stands for restricted constant.

**Case 3:**
- Unrestricted *intercept* and no *trend*.
- `case` inputs: 3 or "uc" where "uc" stands for unrestricted constant.

**Case 4:**
- Unrestricted *intercept* and restricted *trend*.
- `case` inputs: 4 or "ucrt" where "ucrt" stands for unrestricted constant and restricted trend.

**Case 5:**
- Unrestricted *intercept* and unrestricted *trend*.
- `case` inputs: 5 or "ucut" where "ucut" stands for unrestricted constant and unrestricted trend.

Note that you can't restrict (or leave unrestricted) a parameter that doesn't exist in the input model. For example, you can't compute `recm(object,case=3)` if the object is an ARDL (or UECM) model with no intercept. The same way, you can't compute `bounds_f_test(object,case=5)` if the object is an ARDL (or UECM) model with no linear trend.

## References

Pesaran, M. H., Shin, Y., & Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326

## Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

## See Also

ardl uecm

## Examples

```
data(denmark)

## Estimate the RECM, conditional to it's underlying ARDL(3,1,3,2) -----

# Indirectly from an ARDL
ardl_3132 <- ardl(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))
recm_3132 <- recm(ardl_3132, case = 2)

# Indirectly from an UECM
uecm_3132 <- uecm(ardl_3132)
recm_3132_ <- recm(uecm_3132, case = 2)
identical(recm_3132, recm_3132_)
summary(recm_3132)

## Error Correction Term (ect) & Speed of Adjustment -------------------

# The coefficient of the ect,
# shows the Speed of Adjustment towards equilibrium.
# Note that this can be also be obtained from an UECM,
# through the coefficient of the term L(y, 1) (where y is the dependent variable).
tail(recm_3132$coefficients, 1)
uecm_3132$coefficients[2]
```

---

uecm *Unrestricted ECM regression*

---

## Description

uecm is a generic function used to construct Unrestricted Error Correction Models (UECM). The function invokes two different methods. The default method works exactly like ardl. The other method requires an object of class 'ardl'. Both methods create the conditional UECM, which is the UECM of the underlying ARDL.

## Usage

```
uecm(...)

## S3 method for class 'ardl'
uecm(object, ...)

## Default S3 method:
uecm(formula, data, order, start = NULL, end = NULL, ...)
```

## Arguments

| | |
|---|---|
| `...` | Additional arguments to be passed to the low level regression fitting functions. |
| `object` | An object of `class` 'ardl'. |
| `formula` | A "formula" describing the linear model. Details for model specification are given under 'Details'. |
| `data` | A time series object (e.g., "ts", "zoo" or "zooreg") or a data frame containing the variables in the model. In the case of a data frame, it is coerced into a `ts` object with `start = 1`, `end = nrow(data)` and `frequency = 1`. If not found in data, the variables are NOT taken from any environment. |
| `order` | A specification of the order of the underlying ARDL model (e.g., for the UECM of an ARDL(1,0,2) model it should be `order = c(1,0,2)`). A numeric vector of the same length as the total number of variables (excluding the fixed ones, see 'Details'). It should only contain positive integers or 0. An integer could be provided if all variables are of the same order. |
| `start` | Start of the time period which should be used for fitting the model. |
| `end` | End of the time period which should be used for fitting the model. |

## Details

The `formula` should contain only variables that exist in the data provided through `data` plus some additional functions supported by `dynlm` (i.e., `trend()`).

You can also specify fixed variables that are not supposed to be lagged (e.g. dummies etc.) simply by placing them after `|`. For example, `y ~ x1 + x2 | z1 + z2` where `z1` and `z2` are the fixed variables and should not be considered in `order`. Note that the `|` notion should not be confused with the same notion in `dynlm` where it introduces instrumental variables.

## Value

`uecm` returns an object of `class` `c("dynlm","lm","uecm")`. In addition, attributes 'order', 'data', 'parsed_formula' and 'full_formula' are provided.

## Mathematical Formula

The formula of an Unrestricted ECM conditional to an $ARDL(p, q_1, \ldots, q_k)$ is:

$$\Delta y_t = c_0 + c_1 t + \pi_y y_{t-1} + \sum_{j=1}^{k} \pi_j x_{j,t-1} + \sum_{i=1}^{p-1} \psi_{y,i} \Delta y_{t-i} + \sum_{j=1}^{k} \sum_{l=1}^{q_j-1} \psi_{j,l} \Delta x_{j,t-l} + \sum_{j=1}^{k} \omega_j \Delta x_{j,t} + \epsilon_t$$

## Author(s)

Kleanthis Natsiopoulos, <klnatsio@gmail.com>

## See Also

ardl recm

## Examples

```
data(denmark)

## Estimate the UECM, conditional to it's underlying ARDL(3,1,3,2) -----

# Indirectly
ardl_3132 <- ardl(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))
uecm_3132 <- uecm(ardl_3132)

# Directly
uecm_3132_ <- uecm(LRM ~ LRY + IBO + IDE, data = denmark, order = c(3,1,3,2))
identical(uecm_3132, uecm_3132_)
summary(uecm_3132)
```

# Index