ARDL

Natsiopoulos, Kleanthis; Tzeremes, Nickolaos G.

DOI:
10.1007/s10614-023-10487-z

Publication date:
2023

Licence:
Other

Document Version
Peer reviewed version

ARDL: An R package for ARDL models and cointegration

Kleanthis Natsiopoulos[1] and Nickolaos G. Tzeremes[1*]

[1*]Department of Economics, University of Thessaly, 28th October street, 78, Volos, 38333, , Greece.

*Corresponding author(s). E-mail(s): bus9nt@econ.uth.gr;
Contributing authors: klnatsio@econ.uth.gr;

Abstract

This paper presents the `ARDL` package for the statistical language R, demonstrating its main functionalities in a step by step guide. Some of its main advantages over other related `R` packages are the intuitive API, and the fact that includes many important features missing from other packages that are essential for an in depth analysis. Additionally, it is designed in such a way that it can be combined with other packages for post regression diagnostics and tests. These characteristics are shown through an example, where we showcase part of the application demonstrated in the seminal work of Pesaran, Shin, and Smith (2001).

Keywords: R, R package, ARDL, bounds test, cointegration

JEL Classification: C22 , C87 , Y20

1 Introduction

Pesaran et al (2001) (henceforth PSS) introduced the bounds test for cointegration based on the previous work of Pesaran and Shin (1999) using the ARDL model as a platform for the test. Since then, the ARDL framework and the bounds test are used constantly by practitioners who seem to adopt every new advancement of the initial framework. A recent example combining various techniques, is Wu et al (2022) who applied bootstrap ARDL with a Fourier function.

This paper provides a smooth introduction to the ARDL package in R (R Core Team, 2023) and its main features and capabilities. It fills the gap between PSS and the recent study of Natsiopoulos and Tzeremes (2022) in which the validity and accuracy of the ARDL package is verified. The first one introduces the bounds test whereas the other proves that the ARDL package is capable to perform this type of modeling and tests. There are many recent publications using the ARDL package (Allen and McAleer, 2021; Baruah, 2022; Bertelli et al, 2022; Maravalle and Pandiella, 2022; Qiu et al, 2021), since it has already gained trust within the econometrics R users community.

Regarding proprietary software like EViews, although they are generally considered more user-friendly, they lack flexibility compared to programming languages such as R. Additionally, these software platforms are often slow to adopt the latest advancements in research and can be prohibitively expensive for many users.

On the other hand, open-source software do not provide any guarantees regarding the quality of results, and it is the responsibility of the user to verify the code. The problem lies in the fact that not everyone is an expert in the field, making it challenging to technically validate the code's implementation. Many practitioners simply seek reliable software they can trust.

During the development of the ARDL package, we tested various existing open-source packages in R. However, when comparing the open-source packages to other proprietary software options, we encountered inconsistent results, with the R packages producing outcomes that differed not only among themselves but also when compared to the proprietary software alternatives. Moreover, we observed that these open-source packages were developed following diverse programming practices. For instance, some packages present results in tabulated form, similar to the approach seen in menu-driven software.

Hence, these shortcomings served as the driving force behind the development of the ARDL package. This package has enhanced existing software by providing dependable and accurate results while maintaining a user-friendly experience that facilitates seamless interaction with other packages. The purpose of this paper is to serve as a step-by-step guide for users, offering them a starting point similar to a user manual found in proprietary software.

We demonstrate the main functionalities of the ARDL package (Natsiopoulos and Tzeremes, 2023, version 0.2.4) using the data from the PSS study. This way, researchers and authors interested in applying these methodologies, have a clear and step by step guide where they can utilize the codes and adjust them to their research needs.

2 ARDL, ECM and Bounds test

Below we present the ARDL equation and the conditional error correction model (ECM) which are equivalent representations of the same model, and they are the building blocks of the bounds test for cointegration which is also

presented. These are the most important equations that one needs in order to follow the code examples below[1].

An ARDL model, estimated using ordinary least squares (OLS), is a linear model that comprises two key components: the autoregressive part (AR) and the distributed lags (DL) of the independent variables. In the AR part, the dependent variable is considered in lagged form, while in the DL part, the independent variables are included in both their levels and lagged forms. Accordingly, an ARDL model denoted as $ARDL(3,0,2)$ would include, up to three lags of the dependent variable, one independent variable considered at the level, and another independent variable included in both its level and lagged forms up to two lags.

ARDL formula:

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

An ECM utilizes the first difference of the dependent variable, regressed on the first lags of both the dependent and independent variables. The remaining regressors in the model consist of the lags of the first differences of both the dependent and independent variables.

ECM formula:

$$\Delta y_t = c_0 + c_1 t + \pi_y y_{t-1} + \sum_{j=1}^{k} \pi_j x_{j,t-1} + \sum_{i=1}^{p-1} \psi_{y,i} \Delta y_{t-i} + \sum_{j=1}^{k} \sum_{l=1}^{q_j - 1} \psi_{j,l} \Delta x_{j,t-l} + \sum_{j=1}^{k} \omega_j \Delta x_{j,t} + \epsilon_t \tag{2}$$

Where $\psi_{j,l} = 0 \ \forall \ q_j \leq 1$, $\psi_{y,i} = 0 \ \ if \ \ p = 1$, and $x_{j,t-1}$ and $\Delta x_{j,t}$ cancel out becoming $x_{j,t} \ \forall \ q_j = 0$

The hypothesis test for the bounds F-test is based on the ECM and the terms c_0 and c_1 are included only in cases 2 and 4, respectively, where this essentially means that they enter the long-run relationship. The hypothesis is the following:

$$\begin{aligned} \mathbf{H_0} &: \pi_y = \pi_1 = \cdots = \pi_k = c_0 = c_1 = 0 \\ \mathbf{H_1} &: \pi_y \neq \pi_1 \neq \ldots \neq \pi_k \neq c_0 \neq c_1 \neq 0 \end{aligned} \tag{3}$$

3 The ARDL package

Below we present some examples that demonstrate the intuitive design of the package which follows common `R` practices in contrast to other arbitrary ways of object treatment and presentation of results. Reviewing the `ARDL` package it is observed that apart from providing a direct API, it also possesses two main

[1] For a detailed presentation of the mathematical formulas that the `ARDL` package uses, please see the official package manual https://cran.r-project.org/web/packages/ARDL/ARDL.pdf

properties. First, it provides a unified framework in order to allow the development of new features and expand the package without changing package's philosophy alongside with the API that the users are already familiar with. Based on this design, users are able to make small or big changes to the existing functions and expand or adjust them to their needs. Second, it allows the objects returned by the package to be used with other packages (using regular objects, not only console printed results) for post estimation tests or for any other econometric analysis. As a result, the `ARDL` package does not provide post estimation tests out of its core concept (e.g. normality, stationarity, autocorrelation and other tests).

On the other hand, there exist other related `R` packages such as `dLagM`, `dynamac`, and `ardl.nardl` (Demirhan, 2020; Jordan and Philips, 2022; Otoakhia, 2023) that share certain similarities with the `ARDL` package. However, these packages lack important functionalities that prevent them from performing the examples presented in this paper. Notably, `dynamac` and `ardl.nardl` do not offer auto ARDL procedures, while `dLagM` and `ardl.nardl` lack the crucial feature of fixed regressors, such as dummy variables. Furthermore, `dLagM` and `dynamac` do not provide long-run multipliers. These limitations prevent the construction of complex models. Specifically, we conducted an error correction model estimation, which is discussed later in subsection 3.1 (denoted as `m3`). However, for this estimation, the dummy variables were omitted for simplicity. The outcomes derived from the `ardl.nardl` package deviate from those produced by the other `R` packages and `EViews`. Furthermore, there are disparities in the bounds F-test, a crucial aspect of this analytical approach, across the three aforementioned packages and even when compared to proprietary software like `EViews`. To illustrate, in a particular model, we performed estimations using the `ARDL` package and `EViews`, yielding F-statistics of 2.87. In contrast, the `ardl.nardl` package and the `dLagM` package generated values of 2.53 and 3.44, respectively. In a different model, `ARDL` and `EViews` computed the F-statistic as 4.13, with lower and upper bounds at 2.56 and 3.49, respectively. In comparison, the `dynamac` package yielded values of 5.12, 3.1, and 3.87 for the same parameters[2].

In addition, it is important to highlight some additional features found in the aforementioned packages. For instance, `dLagM` enables the estimation of models such as Almon and Koyck, `dynamac` offers simulation capabilities by introducing shocks to independent variables, and `ardl.nardl` provides an option to estimate (Nonlinear ARDL) NARDL models. While these functionalities may be valuable to certain users, we have not compared them as they extend beyond the core concept of ARDL modeling and the bounds test. Therefore, we cannot make definitive claims regarding these features. However, it is worth mentioning that the `ARDL` package is continuously evolving, and future

[2]Please note that we have not included the code and results from this analysis in the paper to maintain its focus as an introduction to the `ARDL` package rather than a software comparison study. However, we have shared all the code and results with the anonymous reviewers of this paper, and they are also available upon request.

updates will include supplementary options, including the NARDL model, to enhance its functionality.

Furthermore, we present illustrative flowcharts that depict the relationships between the functions driving the construction of ARDL and ECM models in Figure 1, as well as the broader functionalities offered by the package in Figure 2. These visual aids serve as a cohesive visualization of the underlying processes and connections within the ARDL package. By offering this comprehensive visual guide, we aim to provide users with a holistic overview that reinforces the package's coherence and user-friendly approach. With these flowcharts in place, readers can better appreciate the interplay of functions, thus facilitating a more comprehensive grasp of the package's capabilities.

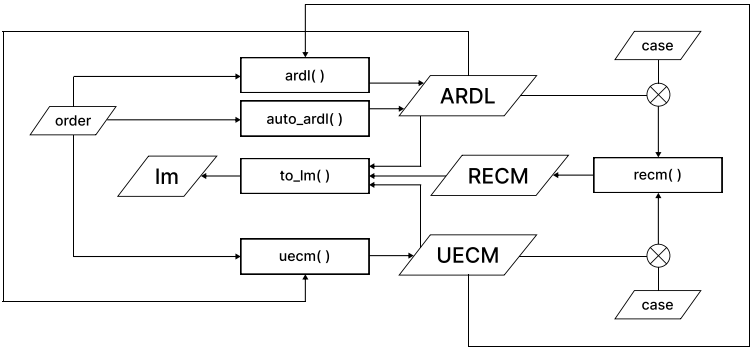

Fig. 1 Flowchart for building ARDL and ECM models using the `ARDL` R package. Rectangles represent functions, parallelograms stand for inputs or outputs depending on the direction of the arrow, and crossed circles show what two or more objects are used simultaneously as inputs. The model objects are shown in bold.

3.1 ARDL and EC models

The first functions we will need are `ardl()` and `uecm()`, where they estimate ARDL and ECM models respectively. These have different structures but are equivalent, thus, we will continue the analysis using the ECM form, as in PSS. We can estimate a conditional ECM model in two ways, either by `uecm(ardl())` or directly by `uecm()`.

In the subsequent examples, we aim to model real wage (w) as a function of various factors, including labor productivity ($Prod$), unemployment rate (UR), wedge effect ($Wedge$), and union power ($Union$). Our initial focus is on determining the suitable $ARDL(p, q_{Prod}, q_{UR}, q_{Wedge}, q_{Union})$ order, where p

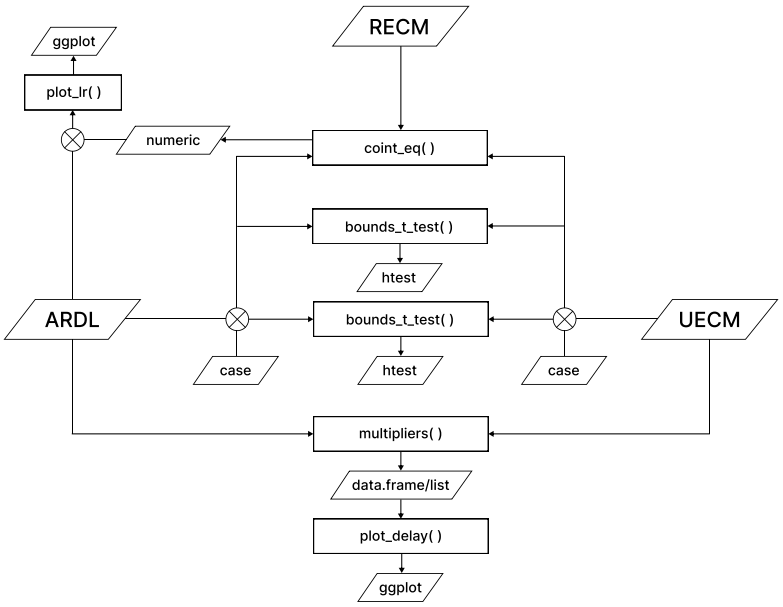

Fig. 2 Flowchart for the functionalities of the `ARDL` R package. Rectangles represent functions, parallelograms stand for inputs or outputs depending on the direction of the arrow, and crossed circles show what two or more objects are used simultaneously as inputs. The model objects are shown in bold

represents the number of lags utilized for the dependent variable, q_{Prod} denotes the number of lags for the variable *Prod*, and so forth.

Specifically, we estimate each model with the same number of lags as when we estimate a VAR model (i.e. $ARDL(p, p, p, p, p)$). In order to implement such a setting, we set `order = p` which is equivalent to `order = c(p,p,p,p,p)`, where p is a positive integer.

Then we set the dataset `PSS2001` and define the formula with the dependent and the independent variables, just like in a usual `lm()` model. Furthermore, we add two dummy variables that will take part in the short-run but not in the long-run relationship. This setting can be implemented by adding the variables after the symbol |. Additionally, we estimate models with linear trend, with, and without constant. To exclude the constant term we add `-1` just like in `lm()` and for the trend we can work in a similar manner as in `dynlm()`[3] (Zeileis, 2019).

PSS note that we should not compare models with different lags as they are not compatible, so we need to ensure that all regressions have the same number of observations. Therefore, we keep out the first 8 observations to be

[3] `dynlm()` is the package that drives every regression of the `ARDL` package

used for the lags. As a result, the remain of the analysis uses the sample period from 1972:Q1 to 1997:Q4. This can be executed by adding `start = c(1972, 01)`, which is also a feature from `dynlm()`.

We run the three forms of equations (without constant, with constant, and with constant and trend) and we estimate each one of them using `order = 1, order = 2, ..., order = 7` denoting the models as $ARDL(1,1,1,1,1)$, $ARDL(2,2,2,2,2)$, ..., $ARDL(7,7,7,7,7)$, respectively[4].

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

```
m <- uecm(ardl(w ~ Prod + UR + Wedge + Union -1| D7475 + D7579,
              order = 3, data = PSS2001, start = c(1972, 01)))
```

```
mc <- uecm(w ~ Prod + UR + Wedge + Union |D7475 + D7579,
          order = 5, data = PSS2001, start = c(1972, 01))
```

```
mt <- uecm(w ~ Prod + UR + Wedge + Union
          + trend(w, scale = FALSE) |D7475 + D7579,
          order = 7, data = PSS2001, start = c(1972, 01))
```

The returned object is a usual `dynlm` regression model caring some extra information.

Observing the results, PSS argue that the case of no constant or with a trend are not reasonable choices for the data, and they also note that the terms $\Delta Prod_{t-1}$, ..., $\Delta Prod_{t-p}$ are not statistically significant neither individually nor jointly, and we should exclude them from the equation, meaning that $q_{Prod} = 1$.

Below the model is re-estimated as:

```
m1 <- uecm(w ~ Prod + UR + Wedge + Union |D7475 + D7579,
          order = c(5, 1, 5, 5, 5), data = PSS2001,
          start = c(1972, 01))
```

After analyzing the results, we can determine the optimal $ARDL(p, q_{Prod}, q_{UR}, q_{Wedge}, q_{Union})$ order using the `auto_ardl()` function, which provides precise control over the model formulation. With the help of the `auto_ardl()` function, we can systematically evaluate numerous (or all) potential models based on a specified criterion, typically the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC). However, it's important to note that the `auto_ardl()` function allows the user to select any other suitable criterion.

We will use this functionality here, since PSS use a slightly different version of AIC, which has the property to assign a higher value to the best model, in contrast with the usual formula of criteria like AIC and BIC. This function is

[4]For space saving reasons we present only three models.

predefined in the package as `AIC_pss` $= ln(\hat{\mathcal{L}}) - K$, where $\hat{\mathcal{L}}$ is the maximized likelihood function $\mathcal{L}(\theta|X)$, and K is the number of estimated coefficients. Then we set `selection_minmax = "max"` so the best model to be the one with the highest value.

PSS suggest that 6 lags should be enough, so we restrict the `auto_ardl()` function to search only for up to 6 lags for each variable using `max_order = 6`.

In our analysis we include only one lag for the variable *Prod*, so we keep $q_{Prod} = 1$ fixed using `fixed_order = c(-1,1,-1,-1,-1)`, setting 1 in the place of q_{Prod} and -1 for the other variables for which we do not wish to set fixed lags.

Then, we can choose a starting order for the step-wise searching regression algorithm (e.g `starting_order = 5`) to avoid local optimum areas or set `grid = TRUE` to evaluate all possible combinations of orders. Although, note that the full search option is considerably more time consuming.

```
formula_c <- w ~ Prod + UR + Wedge + Union |D7475 + D7579
```

```
m2 <- auto_ardl(formula_c, data = PSS2001, start = c(1972, 01),
                max_order = 6, fixed_order = c(-1,1,-1,-1,-1),
                starting_order = 5, selection = "AIC_pss",
                selection_minmax = "max")
```

```
m2$best_order
```

```
   w  Prod   UR Wedge Union
   6     1    5     4     5
```

```
m3 <- m2$best_model
```

The optimization algorithm manages to find the global optimum model that the full search would have found, which is the $ARDL(6, 1, 5, 4, 5)$. The object `m2` contains the best order, the best model itself and a list of the top orders. Now we can save this model directly, without having to estimate it again using `ardl()` with `order = c(6,1,5,4,5)`.

Last, we can get the RECM which is the ECM where we have restricted the lagged variables in levels to form a single term, the error correction term (ECT). At this point we have to decide if the constant term is going to be part of the short-run relationship or the long-run (to be included in the ECT). For this example we use `case = 3` which leaves the intercept unrestricted and thus it is part of the short-run relationship. The model can be estimated in a similar manner as we have previously calculated the UECM, for example:

```
summary(recm(m3, case = 3))
```

```
Time series regression with "zooreg" data:
Start = 1972 Q1, End = 1997 Q4

Call:
dynlm::dynlm(formula = full_formula, data = data, start = ↩
    start,
    end = end)

Residuals:
      Min        1Q    Median        3Q       Max
-0.0175842 -0.0049998 0.0004342 0.0050892 0.0207731

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.619143   0.112409   5.508 4.28e-07 ***
d(L(w, 1))    -0.417479   0.090852  -4.595 1.59e-05 ***
d(L(w, 2))    -0.327934   0.096694  -3.391 0.001083 **
d(L(w, 3))    -0.523383   0.095549  -5.478 4.84e-07 ***
d(L(w, 4))    -0.133031   0.082024  -1.622 0.108767
d(L(w, 5))    -0.196693   0.075506  -2.605 0.010951 *
d(Prod)        0.315304   0.087619   3.599 0.000553 ***
d(UR)          0.003404   0.007024   0.485 0.629303
d(L(UR, 1))    0.015527   0.009777   1.588 0.116207
d(L(UR, 2))    0.003035   0.010193   0.298 0.766678
d(L(UR, 3))    0.028483   0.009982   2.853 0.005504 **
d(L(UR, 4))    0.026883   0.010119   2.657 0.009521 **
d(Wedge)      -0.297128   0.050195  -5.920 7.68e-08 ***
d(L(Wedge, 1)) -0.048266  0.051716  -0.933 0.353486
d(L(Wedge, 2)) -0.093438  0.053388  -1.750 0.083921 .
d(L(Wedge, 3)) -0.187797  0.053657  -3.500 0.000764 ***
d(Union)       -0.968727  0.713954  -1.357 0.178646
d(L(Union, 1)) -2.914759  0.808739  -3.604 0.000543 ***
d(L(Union, 2)) -0.021200  0.860420  -0.025 0.980404
d(L(Union, 3)) -0.101096  0.748143  -0.135 0.892849
d(L(Union, 4)) -1.994719  0.654656  -3.047 0.003131 **
D7475           0.029309  0.005085   5.764 1.48e-07 ***
D7579           0.016846  0.004598   3.664 0.000445 ***
ect            -0.229220  0.042411  -5.405 6.53e-07 ***
---
Signif. codes: 0 ‘***’0.001 ‘**’0.01 ‘*’0.05 ‘.’ 0.1 ‘’1

Residual standard error: 0.00805 on 80 degrees of freedom
  (0 observations deleted due to missingness)
Multiple R-squared: 0.6745,  Adjusted R-squared: 0.581
F-statistic: 7.209 on 23 and 80 DF, p-value: 1.288e-11
```

Usually we are interested in the coefficient of ECT which shows the speed of adjustment to the long-run equilibrium, which in this case is -0.229.

3.2 Bounds test

After estimating the models, the bounds test can be performed in order to test for the existence of levels relationship (cointegration) between the variables. Below we perform the bounds F-test on the object `m1`, which is the ECM form of the underlying $ARDL(5,1,5,5,5)$ model. Since an intercept is included in the estimation, we can choose between case II and case III. Under case II, the intercept is also included in the $H0$ and is restricted to be part of the long-run relationship, whereas under case III it is not. Setting a level of significance (e.g `alpha = 0.05`) we also get the critical value bounds for this level of significance.

```
f1 <- bounds_f_test(m1, case = 3, alpha = 0.05)
f1$tab
```

```
  statistic Lower-bound I(0) Upper-bound I(1) alpha    p.value
F 5.234323          2.894242          4.027319  0.05 0.007630228
```

Moreover, a typical `htest` object is returned, which includes some additional attributes, like the `tab`, which creates a data.frame summarizing the most useful information. The F-statistic is greater than the upper bound, so we can reject the $H0$ of no cointegration[5]. Of course, the same conclusion can be reached through the p-value which is less than the *alpha* value.

The generated values are the asymptotic critical value bounds and p-values, assuming $T = 1000$ (sample size), as referred to the tables of the original paper of PSS. We can calculate the exact sample critical value bounds and p-values using `exact = TRUE` (in our case the sample size is $T = 104$). This uses `R = 40000` (number of iterations) by default but this can be changed[6].

```
set.seed(2023)
f2 <- bounds_f_test(m1, case = 3, alpha = 0.05, exact = TRUE)
f2$tab
```

```
  statistic Lower-bound I(0) Upper-bound I(1) alpha    p.value
F  5.234323          2.946428           4.15907  0.05 0.01093268
```

The bounds t-test can be performed the same way. Note that the t-test is not applicable for case II and case IV where the intercept is restricted.

```
t1 <- bounds_t_test(m1, case = 3, alpha = 0.05)
t1$tab
```

[5] For double checking the results see PSS p.312

[6] It is important to set a seed beforehand so that our results to be reproducible.

```
  statistic Lower-bound I(0) Upper-bound I(1) alpha    p.value
t -3.996164        -2.864981        -4.000246  0.05 0.05044917
```

Our findings suggest that the t-statistic (-3.996164) is slightly smaller (in absolute terms) than the upper bound (-4.000246) which is consistent with the p-value being slightly greater than the alpha level $\alpha = 5\%$. In addition, we observe that the bounds as reported in PSS are rounded in the second decimal, which makes the comparison less precise[7] and in this case, the results are inverted. However, we can still get the traditional critical value bounds for comparison.

```
t1$PSS2001parameters
```

```
Lower-bound I(0) Upper-bound I(1)
           -2.86            -3.99
```

3.3 Long-run relationship and multipliers

The long-run relationship can be obtained based on the chosen model $ARDL(6, 1, 5, 4, 5)$, which is saved in the object `m3`. Therefore from the `ARDL` package we can obtain the long-run multipliers and the corresponding p-values, which appears as equation 31 in PSS.

```
multipliers(m3)
```

```
        Term    Estimate Std. Error   t value       Pr(>|t|)
1 (Intercept) 2.7010826 0.24148486 11.185308   1.000458e-17
2       Prod   1.0633250 0.04999031 21.270624   9.772143e-34
3         UR  -0.1046091 0.03367004 -3.106891   2.658463e-03
4      Wedge -0.9428506 0.26498172 -3.558172   6.474164e-04
5      Union  1.4806705 0.31061073  4.766965   8.809324e-06
```

From the estimated long-run multipliers we can compute the long-run relationship (cointegrating equation) and compare it graphically with the dependent variable.

```
ce <- coint_eq(m3, case = 2)
plot_lr(m3, coint_eq = ce, show.legend = TRUE)
```

The user also has the option to calculate short-run (impact), delay and interim multipliers. This is as simple as adding `type = "sr" or type = 0` for short-run or `type = h` for delay and interim multipliers, where h is a positive integer.

[7] See PSS p.312 or PSS Table CII(iii)

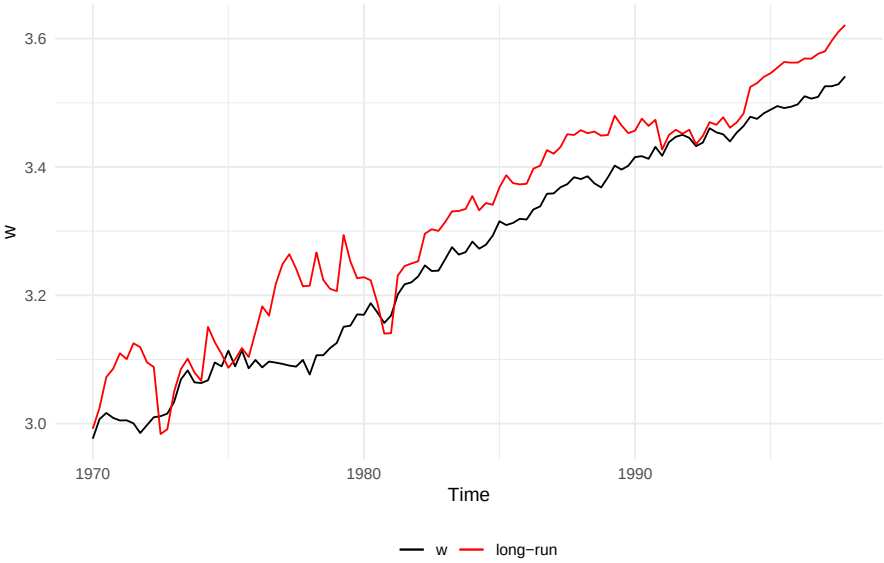

Fig. 3 Long-run relationship

In the following example, we compute the delay multipliers for 15 periods (quarters). These delay multipliers indicate the impact on the dependent variable in period $t+s$, resulting from an instant shock to an independent variable in period t. Figure 4 illustrates the responses of real wage (w) following the shock for each independent variable.

```
delay15 <- multipliers(m3, type = 15, se = TRUE)
plot_delay(delay15, interval = 0.95)
```

3.4 Post-estimation testing

After estimating an ARDL or ECM model using the `ARDL` package, it is essential to perform post-estimation testing to assess the validity of the model's assumptions and identify potential misspecifications. In this subsection, we present various post-estimation tests available through other packages, which can be used to examine the model's robustness and reliability.

In the following examples, we employ the `lmtest` package (Zeileis and Hothorn, 2002) to conduct post-estimation tests for serial correlation using the Breusch-Godfrey test, for heteroskedasticity using the Breusch-Pagan test, and for functional form misspecification using Ramsey's RESET test.

It's worth noting that when attempting to apply the RESET test to the `m3` model which was created using a data set of class `zoo`, we encounter an error. However, this is an uncommon occurrence since creating a model with a data set of another class, typically `data.frame` or `ts`, would not cause any issues. In such cases, a simple solution would be to recreate the model using the `uecm()`

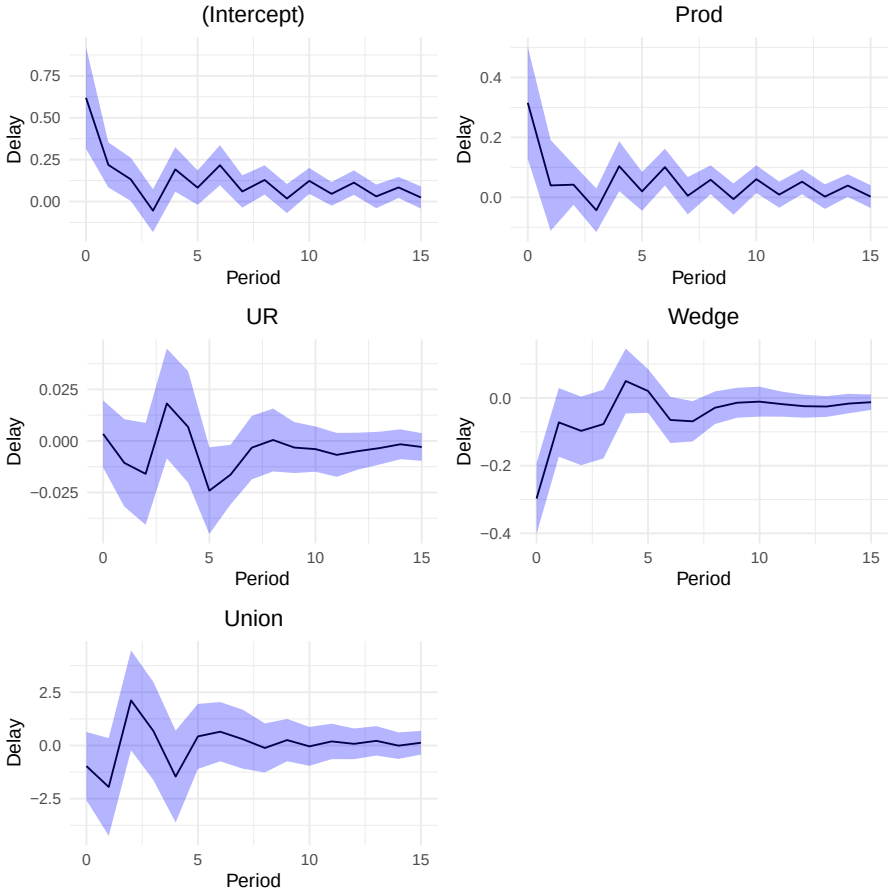

Fig. 4 Delay multipliers of model m3 for 15 periods with 95% CI

function while changing the class of the data set, for example, using `data = as.ts(PSS2001)`.

Alternatively, we can leverage the `to_lm()` function, which provides a convenient way to convert a model of class `ardl`, `uecm`, or `recm` into a regular `lms` model, as if it were created using the `lm()` function. The primary usage of the `to_lm()` function is to facilitate the conversion of the model's class, particularly because certain functions from other packages require models of `lm` class to function properly. In this case, we use the `data_class = "ts"` parameter to convert the class of the data used in the lm model.

```
library(lmtest) # for bgtest(), bptest(), and resettest()
library(tseries) # for jarque.bera.test()
library(strucchange) # for efp(), and sctest()
```

```
bgtest(m3, order = 4)
```

```
Breusch-Godfrey test for serial correlation of order up to 4

data: m3
LM test = 8.7447, df = 4, p-value = 0.06781
```

```
bptest(m3)
```

```
        studentized Breusch-Pagan test

data: m3
BP = 41.627, df = 27, p-value = 0.03581
```

```
m3_lm1 <- to_lm(m3, data_class = "ts")
resettest(m3_lm1, power = 2)
```

```
        RESET test

data: m3_lm1
RESET = 1.9196, df1 = 1, df2 = 75, p-value = 0.17
```

Next, we utilize the `tseries` (Trapletti and Hornik, 2023) and `strucchange` (Zeileis et al, 2002) packages to perform additional post-estimation tests. Specifically, we employ the Jarque-Bera test from the `tseries` package to assess normality and the CUSUM test from the `strucchange` package to identify structural changes.

During this process, we encounter an issue with the `efp()` function, which cannot handle variable names with special functions such as `d()` and `L()`. This limitation is not uncommon among many functions. To address this, we set the parameter `fix_names = TRUE` of the `to_lm()` function to convert the variable names accordingly.

It is important to note that, in none of the cases, did we explicitly require a model of class `lm`. However, we found the `to_lm()` function to be instrumental in resolving these issues and assisting us in conducting these post-estimation tests effectively. Its versatility proves beneficial in addressing various challenges that may arise during the testing process.

By employing these post-estimation testing functions from different packages, we gain valuable insights into the model's assumptions and identify potential sources of misspecification, further ensuring the reliability and robustness of the models.

```
jarque.bera.test(residuals(m3))
```

```
        Jarque Bera Test
```

```
data:  residuals(m3)
X-squared = 0.013978, df = 2, p-value = 0.993
```

```
m3_lm2 <- to_lm(m3, fix_names = TRUE)
fluctuation <- efp(m3_lm2$full_formula, data = m3_lm2$model)
sctest(fluctuation)
```

```
        Recursive CUSUM test

data:  fluctuation
S = 0.58002, p-value = 0.4527
```

```
plot(fluctuation)
```

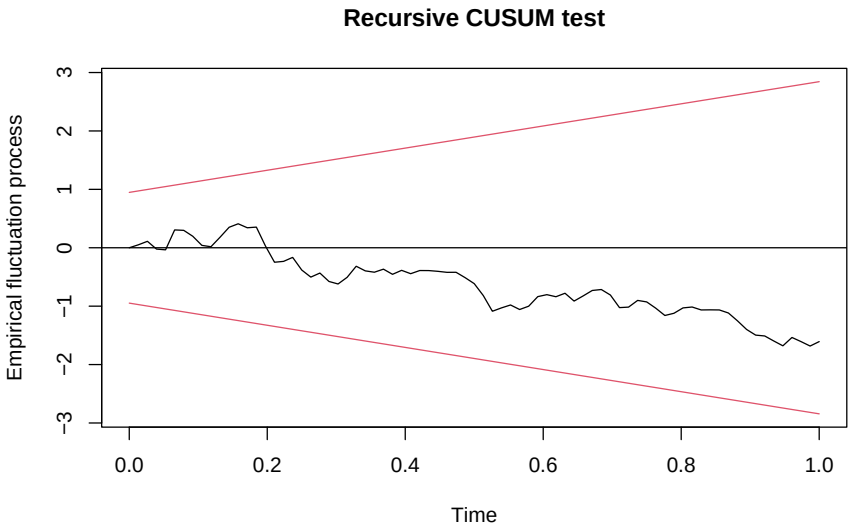

Fig. 5 Recursive CUSUM test

4 Conclusion

This paper serves as a comprehensive step-by-step guide, showcasing the core functionalities of the `ARDL` package, a versatile tool developed in the `R` language. In addition to explaining the package's capabilities, we provide insightful examples that end-users can readily adopt and tailor to suit their unique research requirements.

Throughout these illustrative examples, we highlight the user-friendly API of the ARDL package, which enables effortless estimation of even the most intricate models. The package's flexibility becomes evident as it easily accommodates the calculation of complex designs, making it a valuable asset for researchers seeking reliable and robust results.

By presenting this guide, we aim to empower researchers and practitioners to harness the full potential of the `ARDL` package, facilitating seamless model estimation and analysis. Whether it be for economic forecasting, policy evaluations, or academic research, the `ARDL` package equips users with an accessible and efficient toolkit to enhance their data analysis endeavors.

Ultimately, our hope is that this paper contributes to the advancement of empirical economic analysis and stimulates further exploration and utilization of the `ARDL` package in diverse research settings. As we continue to improve and expand the capabilities of the `ARDL` package through future updates, we anticipate its continued relevance and impact in the field of economic modeling and analysis.

Acknowledgments. We appreciate the anonymous referees' helpful and insightful comments, which greatly enhanced our paper. The usual disclaimer applies.

Declarations

Conflict of interest. All authors have stated that there are no conflicts of interest connected with this article.

References

Allen DE, McAleer M (2021) A nonlinear autoregressive distributed lag (nardl) analysis of the ftse and s&p500 indexes. Risks 9(11). https://doi.org/10.3390/risks9110195

Baruah P (2022) Investigating commodity price relations across wholesale markets: The case of paddy in chhattisgarh, india. Indian Journal of Agricultural Economics 77(1)

Bertelli S, Vacca G, Zoia M (2022) Bootstrap cointegration tests in ardl models. Economic Modelling 116:105,987. https://doi.org/10.1016/j.econmod.2022.105987

Demirhan H (2020) dLagM: An R package for distributed lag models and ARDL bounds testing. PLoS ONE 15(2):e0228,812. https://doi.org/10.1371/journal.pone.0228812, URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0228812

Jordan S, Philips AQ (2022) dynamac: Dynamic Simulation and Testing for Single-Equation ARDL Models. URL https://CRAN.R-project.org/package=dynamac, r package version 0.1.12

Maravalle A, Pandiella A (2022) The pass-through of the monetary policy rate into lending rates in mexico. Tech. Rep. 1734, OECD Economics Department Working Papers, Paris, https://doi.org/10.1787/acf23bc6-en

Natsiopoulos K, Tzeremes N (2023) ARDL: ARDL, ECM and Bounds-Test for Cointegration. URL https://CRAN.R-project.org/package=ARDL, r package version 0.2.4

Natsiopoulos K, Tzeremes NG (2022) ARDL bounds test for cointegration: Replicating the Pesaran et al. (2001) results for the UK earnings equation using R. Journal of Applied Econometrics 37(5):1079–1090. https://doi.org/https://doi.org/10.1002/jae.2919

Otoakhia EI (2023) ardl.nardl: Linear and Nonlinear Autoregressive Distributed Lag Models: General-to-Specific Approach. URL https://CRAN.R-project.org/package=ardl.nardl, r package version 1.2.3

Pesaran MH, Shin Y (1999) An Autoregressive Distributed-Lag Modelling Approach to Cointegration Analysis, Cambridge University Press, chap 11, p 371–413. Econometric Society Monographs, https://doi.org/10.1017/CCOL521633230.011

Pesaran MH, Shin Y, Smith RJ (2001) Bounds testing approaches to the analysis of level relationships. Journal of Applied Econometrics 16(3):289–326. https://doi.org/10.1002/jae.616

Qiu RT, Wu DC, Dropsy V, et al (2021) Visitor arrivals forecasts amid covid-19: A perspective from the asia and pacific team. Annals of Tourism Research 88:103,155. https://doi.org/https://doi.org/10.1016/j.annals.2021.103155

R Core Team (2023) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL https://www.R-project.org/

Trapletti A, Hornik K (2023) tseries: Time Series Analysis and Computational Finance. URL https://CRAN.R-project.org/package=tseries, r package version 0.10-54

Wu CF, Huang SC, Chiou CC, et al (2022) The relationship between economic growth and electricity consumption: Bootstrap ardl test with a fourier function and machine learning approach. Computational Economics 60(4):1197–1220. https://doi.org/https://doi.org/10.1007/s10614-021-10097-7

Zeileis A (2019) dynlm: Dynamic Linear Regression. URL https://CRAN.R-project.org/package=dynlm, r package version 0.3-6

Zeileis A, Hothorn T (2002) Diagnostic checking in regression relationships. R News 2(3):7–10. URL https://CRAN.R-project.org/doc/Rnews/

Zeileis A, Leisch F, Hornik K, et al (2002) strucchange: An r package for testing for structural change in linear regression models. Journal of Statistical Software 7(2):1–38. https://doi.org/10.18637/jss.v007.i02