The cointARDL addon for gretl

Artur Tarassow

Version 0.8

Changelog

- Version 0.8 (April, 2021)
  - Return error code when calling both `setMod()` and/ or `cointARDL()` functions, respectively.
  - Fix bug with recent enforcement of the `const` operator.
  - Fix bug with setting up set of linear restrictions for case=2 and an added linear trend
  - Check for missing values.
  - Internal clean up and minor bug-fixes; make use of some functions from the "ADMBP" package.
- Version 0.7 (December, 2018)
  - small adaptions (but not of relevant code)
- Version 0.6 (April, 2018)
  - add the stationary block-bootstrap as as another re-sampling option
- Version 0.51 (May, 2017)
  - correction: following the literature, the wild bootstrap does not rely on resampled residuals but the initially estimated ones, instead.
- Version 0.5:
  - initial version

1 Introduction

The `cointARDL` package is a collection of gretl scripts to conduct single-equation bootstrap cointegration tests based on the autoregressive distributed lag (ARDL) model.

This package comprises the following two single-equation cointegration tests:

1. **BDM t-test**: Banerjee, A., J. Dolado, and R. Mestre (1998): "Error-correction Mechanism

Tests for Cointegration in a Single-equation Framework", *Journal of Time Series Analysis*, 19(3), 267-283.

2. **PSS bounds F-test**: Pesaran, M.H., Y. Shin and R.J. Smith (2001), "Bounds Testing Approaches to the Analysis of Level Relationships", *Journal of Applied Econometrics* Special Issue in honour of J.D. Sargan on the *Theme Studies in Empirical Macroeconometrics* D.F. Hendry and M.H. Pe- saran (eds.), 16, 289-326.

Critical values depend on the deterministic case considered, sample size and number of regressors. For all cases, the corresponding critical values are obtained via bootstrapping instead of relying on simulated asymptotic critical values. The package comprises 4 different bootstrap procedures for drawing the innovations, namely the parametric, non-parametric, wild uniform and wild Rademacher procedures.

2 The ARDL model

Begin with the *autoregressive distributed lag* ARDL(1,1) in levels model

$$y_t = \phi_1 y_{t-1} + c_0 x_t + c_1 x_{t-1} + u_t, \quad t = 1, ..., T$$

where ϕ_1, c_0 and c_1 are unknown parameters, and

$$x_t = x_{t-1} + e_t$$

The model can be re-written in the *error-correction* ECM(-ARDL) form:

$$\begin{aligned} \Delta y_t &= -(1-\phi_1)\left[y_{t-1} - \frac{c_0 + c_1}{1-\phi_1}\right] + c_0 \Delta x_t + u_t \\ &= \rho(y_{t-1} - \beta x_{t-1}) + c_0 \Delta x_t + u_t \end{aligned}$$

where $\rho = -(1-\phi_1)$ denotes the so called error-correction coefficient (speed of adjustment back to the long-run attractor) and $\beta = -\frac{c_0+c_1}{\rho}$ refers to the (non-linear ARDL-based) long-run multiplier. The model is dynamically stable as long as $|\rho| = |1-\phi_1| < 1$.

The general level ARDL(p,q) model can be written as

$$y_t = \sum_{j=1}^{p} \phi_j y_{t-j} + \sum_{j=0}^{q} c_j x_{t-j} + u_t$$

where p and q denote the respective lag length. In case the lowest lag of the exogenous regressors is $q^{min} > 0$ a so called *unconditional* ARDL is estimated, and if the contemporansous x_t value is included the *conditional* ARDL is specified (Pesaran et al., 2001).

The following assumptions are made:

1. $u_t \sim iid(0, \sigma_u^2)$

2. e_t is a general linear stationary process with zero mean and finite variance.

3. u_t and e_t are uncorrelated for all lags such that x_t is strictly exogenous w.r.t. u_t

4. $|\phi_1| < 1$, so that the model is dynamically stable.

2.1 The BDM test

Banerjee et al. (1998) propose to test the null of no cointegration (here illustrated for the level ARDL and ECM-ARDL representation, respectively)

$$H_0 : \phi_1 = 1 \quad \text{or} \quad \rho{=}0$$

against the alternative of cointegration

$$H_1 : \phi_1 < 1 \quad \text{or} \quad \rho{<}0$$

where it follows that there exists a long-run level relationship between y_t and x_t defined by $y_t = \beta x_t + \psi_t$ where $\beta = -\frac{(c_0+c_1)}{1-\phi_1}$ and ψ_t is a zero mean stationary process. The test on the null hypothesis is valid if assumptions 1 to 3 are fulfilled.

Note: In this package, we apply the level ARDL formulation instead of the ECM-ARDL form, and test the corresponding null hypothesis $H_0 : \phi_1 = 1$ for an ARDL(1,1). For a general ARDL(p,q) with p AR lags of the endogenous variable, the respective null hypothesis is

$$H_0 = \sum_{j=1}^{p} \phi_j - 1 = 0 \, .$$

2.2 The PSS bounds test

The BDM testing procedure (and the same holds for the residual-based Engle-Granger as well as the multivariate Johanson cointegration tests) requires all variables to be integrated of order 1. This involves some pre-testing via unit-root tests which introduces further uncertainty into the analysis of long-run relations. Pesaran et al. argue that stationary $I(0)$ variables which are not present in the set of $I(1)$ regressors x_t may also enter a long-run relationship. The PSS test is applicable irrespective of whether the underlying regressors are I(0), I(1) or mutually cointegrated.

PSS propose to test the null of no long-run relationship between the variables (here using the level ARDL(p,q) formulation again) by

$$H_0 : \sum_{j=1}^{p} \phi_j - 1 = \sum_{j=0}^{q} c_j = 0$$

which is equivalent to testing this null based on the ECM-ARDL representation

$$H_0 : \rho = \sum_{j=0}^{q} c_j = 0$$

by means of a Wald- or F-test. In this package, we apply the F-test version using again the level ARDL representation.

2.3 Deterministic components

We differentiate between three cases of interest regarding the specification of deterministic terms in the regression:

- **Case 0**: (unrestricted intercept and no time trend) The unrestricted ARDL is

$$y_t = \alpha + \sum_{j=1}^{p} \phi_j y_{t-1} + \sum_{j=0}^{q} c_j x_{t-j} + u_t$$

  and test for

$$\text{BDM } H_0 : \sum_{j=1}^{p} \phi_j - 1 = 0$$
$$\text{PSS } H_0 : \sum_{j=1}^{p} \phi_j - 1 = \sum_{j=0}^{q} c_j = 0$$

- **Case** 1: (restricted intercept and no time trend) The unrestricted ARDL is

$$y_t = \alpha + \sum_{j=1}^{p} \phi_j y_{t-1} + \sum_{j=0}^{q} c_j x_{t-j} + u_t$$

  and test for

$$\text{BDM } H_0 : \alpha = \sum_{j=1}^{p} \phi_j - 1 = 0$$
$$\text{PSS } H_0 : \alpha = \sum_{j=1}^{p} \phi_j - 1 = \sum_{j=0}^{q} c_j = 0$$

- **Case** 2: (restricted intercept and restricted time trend) The ARDL is

$$y_t = \alpha + \delta t + \sum_{j=1}^{p} \phi_j y_{t-1} + \sum_{j=0}^{q} c_j x_{t-j} + u_t$$

  and test for

$$\text{BDM } H_0 : \alpha = \delta = \sum_{j=1}^{p} \phi_j - 1 = 0$$
$$\text{PSS } H_0 : \alpha = \delta = \sum_{j=1}^{p} \phi_j - 1 = \sum_{j=0}^{q} c_j = 0$$

In principle further cases could be added.

All necessary restrictions based on the level ARDL(p,q) representation are imposed via their implicit form representation $R\theta = d$, and are automatically imposed by the sub function `prepRq()`.

3 Examples

3.1 Illustrating the equivalence of the ECM and the ARDL in levels formulation

To illustrate that the popular ECM representation and the ARDL in levels form are equivalent, we provide a working example of the PSS test on no cointegration in the following.

```
# turn extra output off
set verbose off
# Load the cointARDL package into memory
include cointARDL.gfn
# open the Johansen data and do some
# preliminary transformations
open denmark.gdt -q
d_LRM = diff(LRM)
d_LRY = diff(LRY)
d_IBO = diff(IBO)

# Estimate the ECM-ARDL and run PSS test on no cointegration
ols d_LRM 0 LRM(-1) LRY(-1) IBO(-1) d_LRY d_IBO --quiet
restrict
        b[2]=0
        b[3]=0
        b[4]=0
end restrict

# Run the equivalent ARDL in levels and run PSS test on no cointegration
ols LRM 0 LRM(-1 to -1) LRY(0 to -1) IBO(0 to -1) --quiet
restrict
        b[2]-1=0
        b[3]+b[4]=0
        b[5]+b[6]=0
end restrict
```

The F-statistics are the same as reported in the following gretl output.

```
# ARDL-ECM output
Restriction set
 1: b[LRM_1] = 0
 2: b[LRY_1] = 0
 3: b[IBO_1] = 0
Test statistic: F(3, 48) = 7.33271, with p-value = 0.000383251

# ARDL in levels output
Restriction set
 1: b[LRM_1] = 1
 2: b[LRY] + b[LRY_1] = 0
 3: b[IBO] + b[IBO_1] = 0
Test statistic: F(3, 48) = 7.33271, with p-value = 0.000383251
```

3.2 Run the bootstrap BDM t-test on no cointegration

For illustration, we will rely on the well-known Johansen dataset (quarterly frequency), and estimate a single-equation money demand relationship. The endogenous is the log of a monetary aggregate expressed in real terms (LRM). The regressors are the log of real GDP (LRY) and the bond rate (IBO). We assume all variables to be I(1). We will estimate a conditional ARDL(1,1,1) in levels including an intercept, run a non-parametric bootstrap with 999 iterations and test the restricted intercept **Case 1**.

```
# Setup the list of regressors
list xlist = LRY IBO
# set the list of additional deterministics or further I(0) vars.
list rxlist = const

# Model and test settings
scalar pq = 1   # Lag order (applied to all xlist vars. and the endog.)
scalar condARDL = 1     # 1=conditional ARDL
case = 1
btype = 2               # bootstrap type: 2=non-parametric
bootrep = 999
scalar failstop = 0.1   # max. 10% unstable iterations are allowed
scalar which = 1        # 1=BDM test
scalar verbose = 1      # 1=print details

# Setup the model via setMod()
bundle b = setMod(which, case, LRM, xlist, rxlist, \
pq, condARDL, btype, bootrep, failstop, verbose)

# Run bootstrap cointegration test via runCoint()
runCoint(&b)
```

The corresponding gretl output is

```
*****************************************************
*** BDM bootstrap t-Test based on ARDL(1,1) model ***
*****************************************************
Restriction case: Restricted constant
Bootstrap type: non-parametric
Bootstrap replications: 999
Initial test statistics = 8.329
Critical values:         1pct.     5pct.            10pct.
9.177        6.259        5.073
Bootstrap p-value = 0.016
Fraction of failed iterations = 0.06
*****************************************************
```

Only 6% of the bootstrap models were dynamically unstable indicating a reasonable ARDL specification. The bootstrap p-value is 1.6% and such that we can safely reject the null of no cointegration.

3.3 Run the bootstrap PSS bounds (F-)test on no cointegration

We run the same ARDL(1,1,1) model again but apply the PSS bounds test instead. For this, we only have to change the scalar `which` to `which=2`.

```
# Model and test settings
scalar pq = 1                    # Lag order (applied to all variables)
scalar condARDL = 1      # 1=conditional ARDL
case = 1
btype = 2                        # bootstrap type: 2=non-parametric
bootrep = 999
scalar failstop = 0.1    # max. 10% unstable iterations are allowed
scalar which = 2         # 2=PSS bounds test
scalar verbose = 1

# Setup the model via setMod()
bundle b = setMod(which, case, LRM, xlist, rxlist, \
pq, condARDL, btype, bootrep, failstop, verbose)

# Run bootstrap cointegration test via runCoint()
runCoint(&b)
```

The corresponding gretl output is

```
*****************************************************
*** PSS bootstrap F-Test based on ARDL(1,1) model ***
*****************************************************
Restriction case: Restricted constant
Bootstrap type: non-parametric
Bootstrap replications: 999
Initial test statistics = 6.047
Critical values:         1pct.    5pct.            10pct.
5.431       4.272        3.399
Bootstrap p-value = 0.007
Fraction of failed iterations = 0.04
*****************************************************
```

Again, we can safely reject the null; here even at the 1% significance level.

3.4 Access the bundle output

After running the `runCoint()` procedure, the user has access to various information generated by the `cointARDL` package. For details, we refer to the end of this document. As an example, we plot the test statistics of the PSS test obtained from the bootstrap and the 95% critical value

```
# Grab relevant information from bundle b
scalar cv5 = b.crit[2]          # 5 pct. bootstrap critical value
matrix bstats = b.matstat       # vector of bootstrap test statistics
matrix mplot = bstats~(ones(rows(bstats),1)*cv5)~seq(1,rows(bstats),)'
colnames(mplot,"Test-stats critval5 seq")       # attache column names
plot mplot
        options with-lines fit=none
end plot --output=display
```

This produces the following graph

`cointARDL_fig.png`

4 The various options

4.1 Test type

The user can select the test type by the integer `whichTest`:

`whichTest`	Test type
0	Banerjee/Dolado/Majestre t-test
1	Pesaran/Shin/Smith bounds F-test

4.2 Deterministic case

At the moment, the user can specify the following three different deterministic cases by the integer case:

`case`	Deterministic type
0	unrestricted intercept and no trend
1	restricted intercept and no trend
2	restricted intercept and restricted trend

The differences were already discussed above.

4.3 Specify the endogenous variable

The user needs to define the series `Y` referring to the endogenous variable. **Note**: The number of lags (`pq`) will be added automatically. At the moment it is not possible to set a gappy lag structure.

4.4 The list of exogenous regressors

The user needs to define the list `xlist` including the regressors. **Note**: For each `xlist`-member the same number of lags (`pq`) will be added automatically. At the moment it is not possible to set a gappy lag structure.

4.5 The list of additional exogenous deterministics and I(0) variables

The user can add additional exogenous variables such as deterministics or further exogenous regressors in the list `rxlist`. For members of this list no lags will be added automatically. **Note**: You always need to add the intercept to `rxlist` if you want to include it to the model.

4.6 Determining the lag length

The number of lags is set by the integer `pq`. The same selected lag length applies to the endogenous variable, `Y`, as well as all exogenous regressors of the `xlist` list.

4.7 Conditional or unconditional ARDL

The user can choose whether to include the contemporaneous effects of `xlist`-members (conditional model) or not. The latter type is equivalent of restricting the parameter $c_0 = 0$ in the levels ARDL model. It's meaning is shown below:

`condARDL`	Type
0	unconditional ARDL
1	conditional ARDL

4.8 Bootstrap variants

The user can choose between 5 bootstrap methods. In our the `cointARDL` package we generate for each replication an artificial data set of the same length as the original data set on the assumption that the estimated version of the core model is the true data-generating process, using the observed initial values of each variable, the estimated model, and a set of random innovations.

These innovations can be obtained using 5 different ways. The *parametric* bootstrap draws from a normal distribution. The *non-parametric* bootstrap re-samples with replacement from the estimated residuals.

To illustrate the two semiparametric wild bootstrap approaches, start from the wild bootstrap DGP $y_t^* = X_t\tilde{\beta} + f(\tilde{u}_t)v_t^*$ where $\tilde{\beta}$ denotes the least squares estimates, and $f(\tilde{u}_t)$ is a transformation of the t^{th} residual $\tilde{u}_t$, and v_t^* is a random variable with mean 0 and variance 1.[1]

The *wild uniform* bootstrap assumes that v_t^* is drawn from a uniform distribution. In contrast, the Rademacher two-point distribution is given by

$$v_t^* = \begin{cases} -1 & \text{with probability } 0.5 \\ 1 & \text{with probability } 0.5 \end{cases}$$

The stationary block-bootstrap was proposed by Politis/Romano (1994).[2] This bootstrap attempts to mimic time-dependency in the time-series. The mean block-size is determined by $b = round(1.75 \times T^{1/3})$ where T denotes the sample length.

`btype`	Bootstrap type
0	parametric
1	non-parametric
2	wild uniform
3	wild Rademacher
4	stationary block-bootstrap

[1] For further details, see e.g. russell-davidson.arts.mcgill.ca/e761/rd-jgm-bootstrap.pdf .

[2] The work was published as Dimitris N. Politis and Joseph P. Romano: The Stationary Bootstrap, *Journal of the American Statistical Association*, Vol. 89, No. 428 (Dec., 1994), pp. 1303-1313.

5 Contents of the model bundle

Basic setup	
`whichTest`	integer, test type
`case`	integer, deterministic case
`pq`	integer, ARDL order
`condARDL`	integer, unconditional or conditional ARDL
`btype`	integer, bootstrap type
`bootrep`	integer, no. of bootstrap iterations
`failstop`	scalar, max. fraction of failed (unstable) simulated ARDL models
`verb`	integer, print no details or do
cointARDL post-estimation	
`my`	T by 1 matrix of the obs. of the endogenous
`crit`	3 by 1 matrix containing the 99%, 95% and 90% critical values
`failed`	no. of failed (unstable) iterations
`pvboot`	estimated bootstrap p-value
`teststat`	test statistics of the initial cointegration test
`matstat`	bootrep by 1 vector containing all simulated test statistics
`addDET`	integer taking 1 if the model includes an intercept or 2
	if the model includes an intercept + trend
`error`	Error code: 0 in case of no error, otherwise 1