Package 'ardlverse'

October 31, 2026

Type Package

Title Comprehensive ARDL: Panel, Bootstrap and Fourier Methods

Version 2.1.2

Date 2026-10-30

Description A unified framework for Autoregressive Distributed Lag (ARDL) modeling and cointegration analysis. Implements Panel ARDL with Pooled Mean Group (PMG), Mean Group (MG), and Dynamic Fixed Effects (DFE) estimators following Pesaran, Shin and Smith (1999) <doi:10.1080/01621459.1999.10474156>. Provides the bounds test of Pesaran, Shin and Smith (2001) <doi:10.1002/jae.616> with the critical values of Kripfganz and Schneider (2020) <doi:10.1111/obes.12377> and with recursive bootstrap critical values following Bertelli, Vacca and Zoia (2022) <doi:10.1016/j.econmod.2022.105987> and McNown, Sam and Goh (2018) <doi:10.1080/00036846.2017.1366643>. Includes Quantile Nonlinear ARDL (QNARDL) combining distributional and asymmetric effects based on Shin, Yu and Greenwood-Nimmo (2014) <doi:10.1007/978-1-4899-8008-3_9>, Fourier ARDL for modeling smooth structural breaks following Enders and Lee (2012) <doi:10.1016/j.econlet.2012.04.081>, and Fourier (bootstrap) nonlinear ARDL. Features include Augmented ARDL (AARDL) with deferred t and F tests, Multiple-Threshold NARDL for complex asymmetries, Rolling/Recursive ARDL for time-varying relationships, and Panel NARDL for nonlinear panel cointegration. All methods include comprehensive diagnostics, publication-ready outputs, and visualization tools.

License GPL-3

URL https://github.com/muhammedalkhalaf/ardlverse

BugReports https://github.com/muhammedalkhalaf/ardlverse/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports stats, MASS, quantreg, lmtest, ggplot2, gridExtra, graphics, grDevices

Suggests testthat (>= 3.0.0), knitr, rmarkdown, covr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Muhammad Abdullah Alkhalaf [aut, cre] (ORCID:
<https://orcid.org/0009-0002-2677-9246>),
Yeleazar Levchenko [ctb] (Bug fixes and PMG/MG/DFE corrections for
panel_ardl in v2.0.0 (KSE))

Maintainer Muhammad Abdullah Alkhalaf <muhammedalkhalaf@gmail.com>

Repository CRAN

Date/Publication 2026-10-31 06:50:20 UTC

Contents

ardlverse-package . . . 3
aardl . . . 4
ardl_diagnostics . . . 7
asymmetry_test . . . 9
boot_ardl . . . 10
dynamic_multipliers . . . 13
fbnardl . . . 14
fourier_ardl . . . 18
fourier_bounds_test . . . 20
generate_oil_data . . . 21
generate_panel_data . . . 22
generate_ts_data . . . 23
hausman_test . . . 24
macro_data . . . 26
macro_panel . . . 27
mtnardl . . . 28
oil_data . . . 31
panel_ardl . . . 32
plot.boot_ardl . . . 34
plot.fbnardl . . . 34
pnardl . . . 35
print.boot_ardl . . . 37
print.panel_ardl . . . 37
pss_critical_values . . . 38
qnardl . . . 39
rardl . . . 41
summary.boot_ardl . . . 43
summary.panel_ardl . . . 43

Index **44**

`ardlverse-package` *Comprehensive ARDL Modeling Framework*

Description

A unified framework for Autoregressive Distributed Lag (ARDL) modeling and cointegration analysis. Implements Panel ARDL with Pooled Mean Group (PMG), Mean Group (MG), and Dynamic Fixed Effects (DFE) estimators. Provides bootstrap-based bounds testing for small samples, Quantile Nonlinear ARDL (QNARDL) combining distributional and asymmetric effects, and Fourier ARDL for modeling smooth structural breaks.

Details

The main functions are:

- `panel_ardl`: Panel ARDL with PMG, MG, DFE estimators
- `boot_ardl`: Bootstrap ARDL bounds test (recursive bootstrap of Bertelli, Vacca and Zoia, 2022, or McNown, Sam and Goh, 2018)
- `aardl`: Augmented ARDL bounds test
- `mtnardl`: Multiple-threshold nonlinear ARDL
- `fbnardl`: Fourier (bootstrap) nonlinear ARDL
- `qnardl`: Quantile Nonlinear ARDL
- `fourier_ardl`: Fourier ARDL for structural breaks
- `ardl_diagnostics`: Comprehensive model diagnostics

Supporting functions:

- `hausman_test`: PMG vs MG comparison
- `asymmetry_test`: Long-run asymmetry test
- `dynamic_multipliers`: Cumulative multipliers
- `pss_critical_values`: PSS (2001) critical values

Data generation for examples:

- `generate_panel_data`: Panel data
- `generate_ts_data`: Time series data
- `generate_oil_data`: Oil/gasoline price data

Author(s)

Muhammad Abdullah Alkhalaf <muhammedalkhalaf@gmail.com>

Maintainer: Muhammad Abdullah Alkhalaf <muhammedalkhalaf@gmail.com>

References

Pesaran, M. H., Shin, Y. and Smith, R. P. (1999). Pooled mean group estimation of dynamic heterogeneous panels. Journal of the American Statistical Association.

Pesaran, M. H., Shin, Y. and Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. Journal of Applied Econometrics.

Shin, Y., Yu, B. and Greenwood-Nimmo, M. (2014). Modelling asymmetric cointegration and dynamic multipliers in a nonlinear ARDL framework.

Cho, J. S., Kim, T. H. and Shin, Y. (2015). Quantile cointegration in the autoregressive distributed-lag modeling framework. Journal of Econometrics.

Banerjee, P., Arcabic, V. and Lee, H. (2017). Fourier ADL cointegration test. Economic Modelling.

Examples

```
# Panel ARDL
data <- generate_panel_data()
model <- panel_ardl(gdp ~ inflation + investment, data = data,
                    id = "country", time = "year", estimator = "pmg")
summary(model)

# Bootstrap bounds test
ts_data <- generate_ts_data()
boot <- boot_ardl(gdp ~ investment, data = ts_data, nboot = 99, seed = 1)
summary(boot)

# QNARDL
oil <- generate_oil_data()
qmodel <- qnardl(gasoline ~ oil_price, data = oil, tau = c(0.25, 0.5, 0.75))
asymmetry_test(qmodel, var = "oil_price")
```

`aardl` *Augmented ARDL Bounds Test (AARDL)*

Description

Augmented ARDL bounds test with the overall F test, the t test on the lagged dependent variable and the F test on the lagged independent variables, in linear, nonlinear (NARDL) and Fourier variants, with Kripfganz and Schneider (2020) bounds or bootstrap critical values.

Usage

```
aardl(
  formula,
  data,
  p = 1,
  q = 1,
  case = 3,
```

```
  type = c("linear", "nardl", "fourier", "fnardl", "bootstrap", "bnardl", "fbootstrap",
    "fbnardl"),
  nboot = 2000,
  fourier_k = 1,
  threshold = 0,
  seed = NULL,
  nulls = c("separate", "joint")
)

## S3 method for class 'aardl'
print(x, ...)

## S3 method for class 'aardl'
summary(object, ...)
```

Arguments

formula	A formula such as gdp ~ investment + trade; only plain variable names are accepted.
data	A data frame containing the time series (no missing values).
p	Integer. Number of lagged differences of the dependent variable (default: 1)
q	Integer or vector. Number of differences of each regressor, lags 0 to q - 1 (default: 1)
case	Integer from 1-5 specifying deterministic components
type	Character. Model type: "linear", "nardl", "fourier", "fnardl", "bootstrap", "bnardl", "fbootstrap", "fbnardl" (default: "linear")
nboot	Number of bootstrap replications (default: 2000)
fourier_k	Integer. Number of Fourier frequencies (default: 1, max: 3)
threshold	Numeric. Threshold value for NARDL decomposition (default: 0)
seed	Random seed; used locally, the session's random number generator state is restored afterwards.
nulls	Bootstrap nulls, "separate" (default) or "joint"; see boot_ardl.
x, object	An object of class "aardl"
...	Not used

Details

Three statistics are computed from the conditional error correction model:

- F_pss: F test that the lagged levels of y and of the regressors are jointly zero (with the intercept in case 2 and the trend in case 4; Pesaran, Shin and Smith, 2001)
- t_dep: t test on the lagged dependent variable
- F_ind: F test on the lagged independent variables (McNown, Sam and Goh, 2018)

The function supports 8 sub-models:

1. `"linear"`: ARDL, Kripfganz and Schneider (2020) bounds for F_pss and t_dep (F_ind has no tabulated bounds)
2. `"nardl"`: NARDL with positive and negative partial sums; the bounds use k = number of partial sums
3. `"fourier"`: ARDL with Fourier terms. No valid bounds exist for the bounds test with Fourier terms; no decision is reported and the bounds shown are those of the model without Fourier terms, for reference only (as in `fourier_bounds_test`)
4. `"fnardl"`: NARDL with Fourier terms; as `"fourier"`
5. `"bootstrap"`: ARDL with bootstrap critical values
6. `"bnardl"`: NARDL with bootstrap critical values
7. `"fbootstrap"`: Fourier ARDL with bootstrap critical values
8. `"fbnardl"`: Fourier NARDL with bootstrap critical values

The bootstrap types use the recursive bootstrap of `boot_ardl` (separate nulls of Bertelli, Vacca and Zoia, 2022, by default, or the Fov null for all statistics of McNown, Sam and Goh, 2018, applied to the conditional ECM, a package choice): y* and x* are generated recursively, the partial sums are rebuilt from x* with the same decomposition as the data, and the Fourier terms at the chosen `fourier_k` are held fixed. The decision requires F_pss, t_dep and F_ind to reject; the degenerate cases are labelled. With Fourier terms the bootstrap is valid for the given `fourier_k`; it does not account for a data-based choice of `fourier_k`. With Fourier terms and partial sums (type `"fbnardl"`) the F_ind test of the separate-null bootstrap over-rejects: in a Monte Carlo with independent random walks (n = 100, 200 samples, B = 199) it rejected in 16.0% (Monte Carlo s.e. 2.6%) of the samples at the 5% level, and an independent review Monte Carlo gave 11.3% (s.e. 1.8%); the combined decision rejected in 6.0% (s.e. 1.7%). The name follows the augmented ARDL bounds test of Sam, McNown and Goh (2019); that paper was not available for this release, so its bootstrap details are not reproduced here (pending verification): the bootstrap is the one of Bertelli, Vacca and Zoia (2022) or McNown, Sam and Goh (2018).

Value

An object of class "aardl" containing:

- `F_pss`: PSS F-statistic for bounds test
- `t_dep`: t-statistic for the lagged dependent variable
- `F_ind`: F-statistic for the lagged independent variables
- `critical_values`: Kripfganz and Schneider (2020) bounds (for the Fourier types: bounds of the model without Fourier terms, not valid for the estimated model)
- `boot_results`: bootstrap distributions, critical values and the engine output (bootstrap types)
- `conclusion`: list with `decision`, `message`, `method` (and `p_values` for the bootstrap types)
- `model`: The estimated ARDL model
- `long_run`: Long-run coefficients
- `short_run`: Short-run coefficients
- `diagnostics`: Model diagnostic tests

References

Bertelli, S., Vacca, G. and Zoia, M. (2022). Bootstrap cointegration tests in ARDL models. *Economic Modelling*, 116, 105987. doi:10.1016/j.econmod.2022.105987

Kripfganz, S. and Schneider, D. C. (2020). Response surface regressions for critical value bounds and approximate p-values in equilibrium correction models. *Oxford Bulletin of Economics and Statistics*, 82(6), 1456-1481. doi:10.1111/obes.12377

McNown, R., Sam, C. Y. and Goh, S. K. (2018). Bootstrapping the autoregressive distributed lag test for cointegration. *Applied Economics*, 50(13), 1509-1521. doi:10.1080/00036846.2017.1366643

Pesaran, M. H., Shin, Y. and Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326. doi:10.1002/jae.616

Sam, C. Y., McNown, R. and Goh, S. K. (2019). An augmented autoregressive distributed lag bounds test for cointegration. *Economic Modelling*, 80, 130-141. doi:10.1016/j.econmod.2018.11.001

Examples

```
data <- generate_ts_data(n = 80, seed = 1)
result <- aardl(gdp ~ investment + trade, data = data, p = 2, q = 2, case = 3)
result
boot <- aardl(gdp ~ investment, data = data, type = "bnardl", nboot = 29,
              seed = 1)
boot

summary(result)
result_fourier <- aardl(gdp ~ investment, data = data, type = "fbootstrap",
                        fourier_k = 1, nboot = 49, seed = 1)
summary(result_fourier)
```

`ardl_diagnostics` *ARDL Model Diagnostics*

Description

Comprehensive diagnostic tests for ARDL models.

Usage

```
ardl_diagnostics(model, lags = 4, arch_lags = 4)

## S3 method for class 'ardl_diagnostics'
summary(object, ...)

## S3 method for class 'ardl_diagnostics'
plot(x, which = 1:4, ...)
```

Arguments

`model`	An estimated model object (panel_ardl, boot_ardl, qnardl, or fourier_ardl)
`lags`	Integer. Number of lags for serial correlation tests (default: 4)
`arch_lags`	Integer. Number of lags for ARCH test (default: 4)
`object, x`	An object of class "ardl_diagnostics"
`which`	Integer vector. Which plots to display (1:4)
`...`	Additional arguments

Details

This function performs a battery of diagnostic tests:

- **Serial Correlation**: Breusch-Godfrey LM test and Ljung-Box test
- **Heteroskedasticity**: Breusch-Pagan and ARCH tests
- **Normality**: Jarque-Bera and Shapiro-Wilk tests
- **Functional Form**: RESET test
- **Stability**: CUSUM and CUSUMSQ tests

Value

An object of class "ardl_diagnostics" containing:

`serial_corr`	Breusch-Godfrey test results
`ljung_box`	Ljung-Box test results
`hetero_bp`	Breusch-Pagan test results
`arch`	ARCH-LM test results
`normality`	Jarque-Bera test results
`shapiro`	Shapiro-Wilk test results
`reset`	RESET test results
`cusum`	CUSUM test results
`cusumsq`	CUSUM of squares test results
`residuals`	Model residuals

See Also

`fourier_ardl`, `boot_ardl`

Examples

```
# Estimate a model
data <- generate_ts_data(n = 100)
model <- fourier_ardl(gdp ~ investment, data = data)

# Run diagnostics
diag <- ardl_diagnostics(model)
summary(diag)
plot(diag)
```

`asymmetry_test` *Test Asymmetry in QNARDL Model*

Description

Perform Wald test for long-run asymmetry in QNARDL models.

Usage

```
asymmetry_test(object, var)
```

Arguments

`object`	A qnardl object
`var`	Variable name to test asymmetry for

Details

Tests the null hypothesis that positive and negative long-run effects are equal (H0: theta+ = theta-) against the alternative of asymmetric effects (H1: theta+ != theta-).

The test is performed separately for each quantile.

Value

A data frame with test results by quantile:

`tau`	Quantile
`theta_pos`	Positive long-run coefficient
`theta_neg`	Negative long-run coefficient
`diff`	Difference (theta+ - theta-)
`wald_stat`	Wald test statistic
`p_value`	P-value

See Also

`qnardl`, `dynamic_multipliers`

Examples

```
data <- generate_oil_data(n = 200)

model <- qnardl(
  gasoline ~ oil_price,
  data = data,
  tau = c(0.25, 0.5, 0.75)
)

asymmetry_test(model, var = "oil_price")
```

`boot_ardl` *Bootstrap ARDL Bounds Test*

Description

Bounds test for cointegration in an ARDL model with bootstrap critical values from a recursive bootstrap under the null hypothesis.

Usage

```
boot_ardl(
  formula,
  data,
  p = 1,
  q = 1,
  case = 3,
  nboot = 2000,
  seed = NULL,
  parallel = FALSE,
  ncores = 2,
  nulls = c("separate", "joint"),
  xmodel = NULL,
  level = 0.05
)
```

Arguments

`formula`	A formula such as `gdp ~ investment + trade`; only plain variable names are accepted.
`data`	A data frame containing the time series (no missing values).
`p`	Integer. ARDL order of the dependent variable; the model has `p - 1` lagged differences of y (default: 1)
`q`	Integer or vector. Lags of the differenced regressors: lags 0 to `q` enter the model (default: 1)

`case`	Integer from 1-5 specifying deterministic components: • 1: No intercept, no trend • 2: Restricted intercept, no trend • 3: Unrestricted intercept, no trend (default) • 4: Unrestricted intercept, restricted trend • 5: Unrestricted intercept, unrestricted trend
`nboot`	Number of bootstrap replications (default: 2000)
`seed`	Random seed (default: NULL). The seed is used locally and the random number generator state of the session is restored afterwards.
`parallel`	Deprecated and ignored (the bootstrap runs serially).
`ncores`	Deprecated and ignored.
`nulls`	`"separate"` (default; Bertelli, Vacca and Zoia, 2022) or `"joint"` (Fov null for all statistics, McNown, Sam and Goh, 2018, applied to the conditional ECM); see Details.
`xmodel`	Model generating Δx^*: `"vecm"` (default with separate nulls), `"var"` (default with the joint null) or `"rw"` (random walk with the deterministic terms).
`level`	Significance level of the decision (default: 0.05).

Details

The model is the conditional error correction form of the ARDL(p, q) model of Pesaran, Shin and Smith (2001):

$$\Delta y_t = c + \rho y_{t-1} + \theta' x_{t-1} + \sum_{i=1}^{p-1} \phi_i \Delta y_{t-i} + \sum_{j=0}^{q} \pi_j' \Delta x_{t-j} + u_t.$$

Three statistics are computed:

- **Fov** (`F_stat`): F test that ρ and θ are jointly zero (with the intercept in case 2 and the trend in case 4). `F_overall` is the same statistic and is kept for backward compatibility.
- **t** (`t_stat`): t test that $\rho = 0$.
- **Find** (`Find_stat`): F test that $\theta = 0$ (McNown, Sam and Goh, 2018).

The bootstrap is recursive: y* and x* are generated from the estimated model, observation by observation, with resampled residual pairs, and the three statistics are recomputed with the same design on each bootstrap sample. With `nulls = "separate"` (default) the procedure follows Bertelli, Vacca and Zoia (2022, Section 3): each statistic has its own restricted model (eqs. 16-18), x* is generated from the marginal model of Δx_t on x_{t-1} and lagged differences (eqs. 19-20), the residuals are recentred after each draw (eqs. 21-22) and the initial values are a random block of the data. With `nulls = "joint"` the null of the Fov test generates the samples for all statistics (McNown, Sam and Goh, 2018, Steps 1-8) applied to the conditional ECM above, which is a package choice (MSG write the y equation without the contemporaneous Δx_t); Δx_t follows the unrestricted equation that also contains y_{t-1}, the residuals are recentred once (MSG eq. 13 subtracts the residual mean; no rescaling) and the initial values are the first observations. The x equation has `p - 1` lags of Δy and Δx, as in BVZ eq. 19. Bertelli, Vacca and Zoia (2022) treat cases 2 and 3; cases 1, 4 and 5

are handled by the same algorithm with the deterministic terms of the case (the intercept or trend is restricted under the Fov null in cases 2 and 4), which is a package choice.

Critical values are order statistics of the bootstrap distributions (BVZ eqs. 24-25). Cointegration is concluded only when Fov, t and Find all reject; Fov rejecting without t is reported as the degenerate case of the first type and Fov and t rejecting without Find as the degenerate case of the second type (Pesaran, Shin and Smith, 2001; McNown, Sam and Goh, 2018).

Value

An object of class "boot_ardl" containing:

- `F_stat`, `t_stat`, `Find_stat`: the statistics; `F_overall` equals `F_stat`
- `boot_F`, `boot_t`, `boot_Find`: bootstrap distributions (failed replications are `NA`)
- `cv_F`, `cv_t`, `cv_Find`: bootstrap critical values at the 10%, 5%, 2.5% and 1% levels
- `p_value_F`, `p_value_t`, `p_value_Find`: bootstrap p-values
- `decision`: one of `"COINTEGRATION"`, `"NO_COINTEGRATION"`, `"DEGENERATE_1"`, `"DEGENERATE_2"`, `"UNDETERMINED"`
- `conclusion`: a text version of the decision
- `model`: the estimated ARDL model
- `engine`: the full output of the bootstrap engine (including the number of valid replications and the reproduction check)

References

Bertelli, S., Vacca, G. and Zoia, M. (2022). Bootstrap cointegration tests in ARDL models. *Economic Modelling*, 116, 105987. doi:10.1016/j.econmod.2022.105987

McNown, R., Sam, C. Y. and Goh, S. K. (2018). Bootstrapping the autoregressive distributed lag test for cointegration. *Applied Economics*, 50(13), 1509-1521. doi:10.1080/00036846.2017.1366643

Pesaran, M. H., Shin, Y. and Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326. doi:10.1002/jae.616

Examples

```
data(macro_data)
boot_test <- boot_ardl(gdp ~ inflation, data = macro_data[1:60, ],
                       p = 2, q = 1, nboot = 49, seed = 1)
boot_test

summary(boot_test)
plot(boot_test)
```

`dynamic_multipliers` *Dynamic Multipliers for QNARDL*

Description

Compute and plot cumulative dynamic multipliers for QNARDL models.

Usage

```
dynamic_multipliers(object, var, tau = 0.5, horizon = 20)
```

Arguments

`object`	A qnardl object
`var`	Variable name
`tau`	Quantile to compute multipliers for (default: 0.5)
`horizon`	Number of periods (default: 20)

Details

Computes the cumulative dynamic multipliers showing how the effect of a unit change in x accumulates over time towards the long-run equilibrium.

Separate multipliers are computed for positive and negative changes, allowing visualization of asymmetric adjustment paths.

Value

A data frame with:

`horizon`	Time horizon (0 to horizon)
`multiplier`	Cumulative multiplier value
`type`	Either "Positive" or "Negative"

See Also

`qnardl`, `asymmetry_test`

Examples

```
data <- generate_oil_data(n = 200)

model <- qnardl(
  gasoline ~ oil_price,
  data = data,
  tau = c(0.25, 0.5, 0.75)
)
```

```
# Plot dynamic multipliers at median
dynamic_multipliers(model, var = "oil_price", tau = 0.5, horizon = 30)
```

`fbnardl` *Fourier (Bootstrap) Nonlinear ARDL*

Description

Nonlinear ARDL model with positive and negative partial sums (Shin, Yu and Greenwood-Nimmo, 2014), optional Fourier terms for smooth breaks, data-based selection of the Fourier frequency and of the lags, and cointegration tests with Kripfganz and Schneider (2020) bounds (no Fourier terms) or bootstrap critical values.

Usage

```
fbnardl(
  formula,
  data,
  decompose,
  exog = NULL,
  type = c("fnardl", "fbnardl"),
  maxlag = 4,
  maxk = 3,
  kgrid = c("integer", "fractional"),
  fourier = TRUE,
  kstar = NULL,
  lags = NULL,
  ic = c("aic", "bic"),
  bootstrap = c("bvz", "mcnown"),
  xdgp = c("vecm", "rw"),
  reps = 999,
  vcov = c("ols", "HC1", "HAC"),
  horizon = 20,
  bands = 500,
  level = 0.95,
  seed = NULL,
  graph = FALSE
)

## S3 method for class 'fbnardl'
print(x, ...)

## S3 method for class 'fbnardl'
summary(object, ...)

## S3 method for class 'fbnardl'
```

```
coef(object, ...)

## S3 method for class 'fbnardl'
vcov(object, ...)

## S3 method for class 'fbnardl'
nobs(object, ...)
```

Arguments

formula	A formula y ~ x1 + x2 + . . .; only plain variable names are accepted.
data	A data frame with consecutive observations (no missing values or gaps).
decompose	Names of the regressors decomposed into positive and negative partial sums.
exog	Optional names of exogenous variables in data that enter at time t in the short run only (for example dummies); held fixed in the bootstrap.
type	"fnardl" (bounds) or "fbnardl" (bootstrap).
maxlag	Largest lag considered (default 4).
maxk	Largest Fourier frequency (default 3).
kgrid	"integer" (default) or "fractional".
fourier	Logical; include Fourier terms (default TRUE).
kstar	Optional Fourier frequency fixed by the user; it is then not selected, neither on the data nor in the bootstrap.
lags	Optional list with elements p, q (one per decomposed variable) and r (one per other regressor) fixing the lags.
ic	"aic" or "bic".
bootstrap	"bvz" (default) or "mcnown".
xdgp	"vecm" (default) or "rw", the model of x* with bootstrap = "bvz".
reps	Bootstrap replications (default 999).
vcov	Covariance matrix for the long-run standard errors, asymmetry tests and multiplier bands: "ols", "HC1" or "HAC" (Newey-West with Bartlett weights and $\lfloor 4(n/100)^{2/9} \rfloor$ lags). The bounds and bootstrap statistics always use the OLS covariance.
horizon	Largest horizon of the dynamic multipliers (default 20).
bands	Number of parametric draws for the multiplier bands (default 500; 0 for none).
level	Confidence level of the bands (default 0.95).
seed	Optional seed for the bootstrap and the bands; the session's random number generator state is restored afterwards.
graph	Logical; plot the multipliers (default FALSE).
x, object	An object of class "fbnardl"
...	Not used

Details

The model (case 3, unrestricted intercept) is

$$\Delta y_t = c+\rho y_{t-1}+\sum_i(\beta_i^+ x_{i,t-1}^+ +\beta_i^- x_{i,t-1}^-)+\sum_j \delta_j z_{j,t-1}+\sum_{l=1}^{p}\phi_l \Delta y_{t-l}+\sum_i\sum_{l=0}^{q_i}(\theta_{il}^+ \Delta x_{i,t-l}^+ +\theta_{il}^- \Delta x_{i,t-l}^-)+\sum_j\sum_{l=0}^{r_j}\pi_{jl}\Delta$$

where $x_t^+ = \sum_{s\le t}\max(\Delta x_s, 0)$ and $x_t^- = \sum_{s\le t}\min(\Delta x_s, 0)$ start at zero at the first observation, z are the regressors that are not decomposed and e_t the exogenous short-run variables (`exog`).

Selection. When `fourier = TRUE` and `kstar` is not given, k^* minimises the sum of squared residuals of the model with every lag at `maxlag`; the grid is $1, \dots,$ `maxk` (`kgrid = "integer"`; integer frequencies chosen by minimum SSR as in Enders and Lee, 2012, Oxford Bulletin of Economics and Statistics) or $0.1, 0.2, \dots,$ `maxk` (`kgrid = "fractional"`). The default and the validity of the fractional grid are package choices pending verification against Yilanci, Bozoklu and Gorus (2020) and Omay (2015). Then p (1 to `maxlag`) and each q_i, r_j (0 to `maxlag`) minimise the AIC or BIC. Every candidate and the final model use the same sample, $t >$ `maxlag + 1`, so the criteria are comparable. `lags` fixes the lags instead.

Cointegration tests. Fov is the F test that ρ, all β and δ are zero, t the t test on ρ and Find the F test on the β and δ. Without Fourier terms (`fourier = FALSE`) and `type = "fnardl"`, the bounds of Kripfganz and Schneider (2020) are used for Fov and t, with k = 2 times the number of decomposed variables plus the number of other regressors, n the number of observations used and the number of short-run coefficients sr = p + sum 2(q_i + 1) + sum (r_j + 1) + number of `exog`; the bounds decision requires both statistics beyond their I(1) bounds. With Fourier terms no valid bounds exist: no decision is given for `type = "fnardl"` and the bounds printed are those of the model without Fourier terms, for reference only. Inference with Fourier terms is by the bootstrap (`type = "fbnardl"`).

Bootstrap. `type = "fbnardl"` uses the recursive bootstrap engine of `boot_ardl`: `bootstrap = "bvz"` has separate nulls for Fov, t and Find (Bertelli, Vacca and Zoia, 2022), x* from the marginal model of Δx_t on the intercept, x_{t-1}, p lags of Δy and Δx, the Fourier terms and `exog` (`xdgp = "vecm"`, default; BVZ eqs. 19-20), or from the regression of Δx_t on the intercept, the Fourier terms and `exog` only (`xdgp = "rw"`, a random walk with these deterministic terms; a package choice that is not the x model of BVZ), residuals recentred after each draw and a random block of initial values (the partial sums continue the data's levels in the block); `bootstrap = "mcnown"` uses the null of the Fov test for all statistics (McNown, Sam and Goh, 2018, Steps 1-8) applied to the conditional ECM, which is a package choice (MSG write the y equation without the contemporaneous differences), with the unrestricted equation for Δx that also contains y_{t-1}, residuals recentred once and the observed initial values. The partial sums are rebuilt from x*; the Fourier terms and `exog` are held fixed. On every bootstrap sample k^* (unless fixed by `kstar`) and the lags (unless fixed by `lags`) are selected again, so the bootstrap distribution accounts for the selection (package choice; BVZ step 5(c) re-estimates the unrestricted model but does not discuss re-selection). The decision requires Fov, t and Find to reject; the degenerate cases are labelled. Failed replications are `NA` and counted. In a Monte Carlo with independent random walks (n = 100, 200 samples, B = 199, maxlag = 1, integer grid) the 5% rejection rates were 7.0% (Monte Carlo s.e. 1.8%) for Fov, 11.5% (2.3%) for t and 11.5% (2.3%) for Find, and 3.5% (1.3%) for the combined decision. Find tends to over-reject with Fourier terms and partial sums (16.0%, s.e. 2.6%, for `aardl(type = "fbnardl")` in the same design).

Asymmetry and multipliers. The long-run coefficients $-\beta^{\pm}/\rho$ have delta-method standard errors, and the long-run symmetry test is the delta-method Wald test of $-\beta^+/\rho = -\beta^-/\rho$ (chi-squared, 1

df). The short-run symmetry test compares the sums over lags 0 to q_i, $\sum_l \theta_{il}^+ = \sum_l \theta_{il}^-$ (F(1, df)); testing the sums is a package choice pending verification against Shin, Yu and Greenwood-Nimmo (2014). The cumulative dynamic multipliers are the responses of the level of y to a permanent unit increase in x^+ or x^- at horizon 0, computed by the full recursion of the estimated ECM (all lagged differences included), with parametric confidence bands from `bands` draws of the coefficients from their estimated normal distribution. The sign convention of the multiplier of x^- (response to a unit increase in the negative partial sum) is pending verification against Shin, Yu and Greenwood-Nimmo (2014).

The R code is a port written for this package, following the Stata module fbnardl.

Value

An object of class `"fbnardl"` with elements `coefficients`, `vcov`, `se`, `residuals`, `fitted.values`, `lags`, `kstar`, `ic_value`, `ssr_by_k`, `multipliers` (short- and long-run, with standard errors), `asymmetry_tests`, `bounds_test` (statistics, bounds or bootstrap results and decision), `dynamic_multipliers` (paths and bands), `diagnostics`, `nobs`, `df.residual`, `r.squared`, `adj.r.squared`, `sigma`, `design`, `call` and the settings.

References

Bertelli, S., Vacca, G. and Zoia, M. (2022). Bootstrap cointegration tests in ARDL models. *Economic Modelling*, 116, 105987. doi:10.1016/j.econmod.2022.105987

Enders, W. and Lee, J. (2012). A unit root test using a Fourier series to approximate smooth breaks. *Oxford Bulletin of Economics and Statistics*, 74(4), 574-599. doi:10.1111/j.14680084.2011.00662.x

Kripfganz, S. and Schneider, D. C. (2020). Response surface regressions for critical value bounds and approximate p-values in equilibrium correction models. *Oxford Bulletin of Economics and Statistics*, 82(6), 1456-1481. doi:10.1111/obes.12377

McNown, R., Sam, C. Y. and Goh, S. K. (2018). Bootstrapping the autoregressive distributed lag test for cointegration. *Applied Economics*, 50(13), 1509-1521. doi:10.1080/00036846.2017.1366643

Omay, T. (2015). Fractional frequency flexible Fourier form to approximate smooth breaks in unit root testing. *Economics Letters*, 134, 123-126. doi:10.1016/j.econlet.2015.07.010

Pesaran, M. H., Shin, Y. and Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326. doi:10.1002/jae.616

Shin, Y., Yu, B. and Greenwood-Nimmo, M. (2014). Modelling asymmetric cointegration and dynamic multipliers in a nonlinear ARDL framework. In R. C. Sickles and W. C. Horrace (Eds.), *Festschrift in Honor of Peter Schmidt* (pp. 281-314). Springer. doi:10.1007/9781489980083_9

Yilanci, V., Bozoklu, S. and Gorus, M. S. (2020). Are BRICS countries pollution havens? Evidence from a bootstrap ARDL bounds testing approach with a Fourier function. *Sustainable Cities and Society*, 55, 102035. doi:10.1016/j.scs.2020.102035

Examples

```
d <- generate_oil_data(n = 100)
m <- fbnardl(gasoline ~ oil_price, data = d, decompose = "oil_price",
             maxlag = 2, fourier = FALSE)
m
```

```
mb <- fbnardl(gasoline ~ oil_price, data = d, decompose = "oil_price",
              type = "fbnardl", maxlag = 1, maxk = 2, reps = 49, bands = 50,
              seed = 1)
summary(mb)
```

`fourier_ardl` *Fourier ARDL Model*

Description

Estimate ARDL models with Fourier terms for smooth structural breaks.

Usage

```
fourier_ardl(formula, data, p = 1, q = 1, k = 1, case = 3,
             selection = c("fixed", "aic", "bic"))

## S3 method for class 'fourier_ardl'
summary(object, ...)

## S3 method for class 'fourier_ardl'
print(x, ...)

## S3 method for class 'fourier_ardl'
plot(x, which = "both", ...)
```

Arguments

`formula`	A formula specifying the model: gdp ~ investment + trade + ...
`data`	A data frame containing the time series data
`p`	Integer. Number of lags for dependent variable (default: 1)
`q`	Integer or vector. Number of lags for independent variables (default: 1)
`k`	Integer. Number of Fourier frequencies (default: 1, max: 3)
`case`	Integer from 1-5 specifying deterministic components (default: 3)
`selection`	Character. Method for selecting optimal k: "aic", "bic", or "fixed"
`object, x`	An object of class "fourier_ardl"
`which`	Character. "fourier", "fit", or "both" for plotting
`...`	Additional arguments

Details

This function implements the Fourier ARDL approach that captures smooth structural changes using trigonometric functions. The Fourier terms approximate unknown structural breaks without requiring prior specification of break dates.

The Fourier terms are defined as:

$$f_t = \sum_{k=1}^{K} [a_k \sin(2\pi kt/T) + b_k \cos(2\pi kt/T)]$$

where K is the number of frequencies and T is the sample size.

Value

An object of class "fourier_ardl" containing:

`coefficients`	Estimated coefficients
`long_run`	Long-run coefficients
`fourier_coefs`	Fourier term coefficients
`ec_coef`	Error correction coefficient
`bounds_test`	F and t statistics for bounds test
`k`	Number of Fourier frequencies used
`aic`	AIC values for k = 1, 2, 3
`bic`	BIC values for k = 1, 2, 3
`structural_breaks`	Detected structural break periods

References

Banerjee, P., Arcabic, V. and Lee, H. (2017). Fourier ADL cointegration test to approximate smooth breaks with new evidence from crude oil market. Economic Modelling, 67, 114-124.

Enders, W. and Lee, J. (2012). A unit root test using a Fourier series to approximate smooth breaks. Oxford Bulletin of Economics and Statistics, 74(4), 574-599.

See Also

`fourier_bounds_test`, `generate_ts_data`

Examples

```
# Generate example data
data <- generate_ts_data(n = 100)

# Estimate Fourier ARDL with automatic selection
model <- fourier_ardl(
  gdp ~ investment + trade,
  data = data,
  selection = "aic"
```

```
)
summary(model)
plot(model)
```

`fourier_bounds_test` *Fourier ARDL Bounds Test*

Description

Reports the bounds test statistics of a Fourier ARDL model.

Usage

```
fourier_bounds_test(object)
```

Arguments

`object`	A fourier_ardl object

Details

No critical values for the PSS bounds F test with Fourier terms have been tabulated in the literature cited by this package, and the Fourier terms change the null distribution. The function therefore returns no decision. For reference only, it also reports the Pesaran, Shin and Smith (2001) bounds for the model without Fourier terms, from the response surfaces of Kripfganz and Schneider (2020); these are not valid for the Fourier model. Use a bootstrap for inference.

Value

Invisibly, a list with `F_stat`, `t_stat`, `cv_F = NA` and `cv_pss_reference`.

References

Kripfganz, S. and Schneider, D. C. (2020). Response surface regressions for critical value bounds and approximate p-values in equilibrium correction models. *Oxford Bulletin of Economics and Statistics*, 82(6), 1456-1481. doi:10.1111/obes.12377

`generate_oil_data` *Generate Oil Price Data*

Description

Generate simulated oil and gasoline price data with asymmetric effects.

Usage

```
generate_oil_data(n = 200, seed = 789)
```

Arguments

`n`	Number of observations (default: 200)
`seed`	Random seed for reproducibility (default: 789)

Details

Generates a simulated dataset exhibiting the "rockets and feathers" phenomenon where gasoline prices respond more quickly to oil price increases than decreases. Suitable for demonstrating QNARDL methods and asymmetry testing.

Value

A data frame with columns:

`week`	Week number (1 to n)
`gasoline`	Retail gasoline price (cents/gallon)
`oil_price`	Crude oil price ($/barrel)
`exchange_rate`	USD exchange rate index

See Also

`qnardl`, `asymmetry_test`

Examples

```
# Generate oil price data
oil <- generate_oil_data(n = 200)
head(oil)

# Plot prices
plot(oil$week, oil$oil_price, type = "l", col = "blue",
     main = "Oil and Gasoline Prices")
lines(oil$week, oil$gasoline/4, col = "red")  # Scaled for comparison
```

`generate_panel_data` *Generate Example Panel Data*

Description

Generate simulated panel data for examples and testing.

Usage

```
generate_panel_data(n_groups = 10, n_time = 50, seed = 123)
```

Arguments

`n_groups`	Number of groups/countries (default: 10)
`n_time`	Number of time periods (default: 50)
`seed`	Random seed for reproducibility (default: 123)

Details

Generates a simulated panel dataset with cointegrated variables suitable for demonstrating panel ARDL methods. The data generating process includes country-specific fixed effects and error correction dynamics.

Value

A data frame with columns:

`country`	Country identifier (1 to n_groups)
`year`	Year (starting from 1970)
`gdp`	Log GDP per capita (I(1))
`inflation`	Inflation rate (I(0))
`investment`	Investment as % of GDP (I(1))
`trade`	Trade openness (I(1))

See Also

`panel_ardl`, `generate_ts_data`

Examples

```
# Generate panel data for 5 countries, 30 years
panel <- generate_panel_data(n_groups = 5, n_time = 30)
head(panel)

# Check dimensions
table(panel$country)
```

`generate_ts_data` *Generate Example Time Series Data*

Description

Generate simulated time series data for examples and testing.

Usage

```
generate_ts_data(n = 100, seed = 456)
```

Arguments

`n`	Number of observations (default: 100)
`seed`	Random seed for reproducibility (default: 456)

Details

Generates a simulated time series dataset with cointegrated variables and a smooth structural break (modeled via Fourier component). Suitable for demonstrating Fourier ARDL and bootstrap bounds testing methods.

Value

A data frame with columns:

`quarter`	Quarter number (1 to n)
`gdp`	Log real GDP (I(1) with structural break)
`inflation`	Inflation rate (I(0))
`investment`	Investment growth (I(1))
`trade`	Trade balance (I(1))

See Also

`fourier_ardl`, `boot_ardl`, `generate_panel_data`

Examples

```
# Generate 100 quarters of data
ts_data <- generate_ts_data(n = 100)
head(ts_data)

# Plot GDP series
plot(ts_data$quarter, ts_data$gdp, type = "l",
     main = "Simulated GDP with Structural Break")
```

hausman_test *Hausman Test for Panel ARDL Estimators*

Description

Compare two panel_ardl estimators (typically MG vs PMG) to test for long-run slope homogeneity.

Usage

```
hausman_test(
  inefficient,
  efficient,
  sigmamore = TRUE,
  method = c("strict", "ginv"),
  tol = 1e-08,
  quiet = FALSE
)
```

Arguments

inefficient	A panel_ardl object, the always-consistent estimator (an MG fit).
efficient	A panel_ardl object, the under-H0 efficient estimator (a PMG or DFE fit).
sigmamore	Logical. If TRUE (default), the MG covariance matrix is multiplied, before the difference is taken, by the factor $$(\hat{\sigma}_{eff}/\hat{\sigma}_{MG})^4,$$ where $\hat{\sigma} = \sqrt{\text{mean}(\hat{\varepsilon}^2)}$ is the root mean squared residual of each fit (component sigma); the efficient covariance matrix is not changed. The fourth power reproduces Stata's hausman, sigmamore after xtpmg: hausman multiplies the consistent covariance by (e(sigma)_eff/e(sigma)_MG)^2 and xtpmg stores the mean squared residual (a variance) under e(sigma), so the realised scale is the fourth power of the RMSE ratio. The fourth power is the Stata/xtpmg convention. Hausman (1978, eqs. 2.16-2.17, p. 1258; p. 1260) forms $V(q)$ with a single σ^2 estimated from either estimator, which is what the Stata manual describes for sigmamore (a common disturbance variance from the efficient estimator) and corresponds to the second power on the RMSE ratio; the fourth power is not in Hausman (1978). For MG versus DFE the same fourth power with the within-regression RMSE is a package choice (Stata's xtpmg DFE branch stores a standard deviation that includes the fixed-effect variance, so no exact Stata analogue exists). The scale actually applied is returned as scale (1 when sigmamore = FALSE).
method	Character. How to treat a difference matrix $V_{MG} - V_{eff}$ that is not positive definite. "strict" (default) returns NA for the statistic and p-value with a warning: the test is not defined. "ginv" uses the spectral pseudo-inverse over the

eigenvalues larger than `tol` times the largest absolute eigenvalue, with degrees of freedom equal to that rank. Eigenvalues at or below the threshold, negative ones included, are dropped. This rule is a package choice. It differs from Stata's `hausman`, whose generalised inverse (`MatGinv`) uses a singular value decomposition with tolerance `max(sv) * k * 1e-9` on the singular values: Stata drops only near-zero directions, keeps directions with negative eigenvalues and can then report a negative chi-squared with a warning.

`tol` Numeric. Relative tolerance for an eigenvalue to count as positive. Default `1e-8`.

`quiet` Logical. If `TRUE`, nothing is printed; warnings are still issued and the result is returned invisibly as usual. Default `FALSE`.

Details

Under the null of long-run homogeneity, the PMG (efficient) estimator is consistent and efficient; the MG (inefficient) estimator is consistent but less efficient. A significant test statistic rejects PMG in favour of MG.

The statistic is Hausman (1978, eq. 2.7) applied to the long-run coefficients only (Pesaran, Shin and Smith, 1999):

$$H = (\hat{\theta}_{MG} - \hat{\theta}_{eff})'[V(\hat{\theta}_{MG}) - V(\hat{\theta}_{eff})]^{-1}(\hat{\theta}_{MG} - \hat{\theta}_{eff}),$$

with k degrees of freedom, k being the number of long-run coefficients; the error-correction and short-run coefficients are not compared. The p-value is the upper tail of the chi-squared distribution.

Argument order in v2.0.0 follows Stata's convention: pass the **inefficient** estimator first (e.g. MG), then the **efficient** one (e.g. PMG). This is a breaking change from earlier versions.

The two fits must be comparable: same dependent variable and regressors, same lag orders `p` and `q`, same `trend`, same `id` and `time` variables, same number of observations and same number of groups. The function stops with an error otherwise.

Value

An (invisible) list with `statistic`, `df`, `p.value`, `theta_diff`, `eigenvalues` (of the symmetrised difference matrix), `n_nonpositive` (number of eigenvalues at or below the tolerance), `method`, `tol`, `sigmamore` and `scale` (the factor applied to the MG covariance matrix).

Non-positive-definite difference matrix

Hausman (1978, Corollary 2.6) shows that $V(\hat{\beta}_1) - V(\hat{\beta}_0)$ is non-negative definite asymptotically when $\hat{\beta}_0$ is efficient under the null, and warns (footnote 20) about inverting a nearly singular matrix. In finite samples the estimated difference can have negative eigenvalues; the quadratic form is then not a valid chi-squared statistic. Versions up to 2.1.0 returned a statistic of 0 with p-value 1 in that case, which reads as acceptance of homogeneity although nothing was tested. The default now returns `NA` with a warning. With `method = "ginv"` the statistic is computed on the positive eigen-subspace; discarding eigenvalues that are small but positive can inflate the statistic considerably, so the tolerance should be reported with the result.

The MG covariance matrix is the cross-section variance of the N group estimates divided by N (Pesaran, Shin and Smith, 1999), so it has rank at most $N - 1$. When $N - 1 < k$ the difference matrix can never be positive definite and the test on all k long-run coefficients is not defined for any

sample; neither `sigmamore` nor a different tolerance can change this, and the warning says so. With `method = "ginv"` a statistic is still reported, on a subspace of at most $N - 1$ dimensions (df at most $N - 1$), not on all k coefficients.

Small number of groups

In a Monte Carlo under the null (homogeneous long-run slopes, ARDL(1,1) with group-specific adjustment, I(1) regressors, 200 replications per cell, default settings) the test was undefined (returned `NA`) in 1 to 21 percent of samples at $N = 10$ and in 11.5 to 53 percent at $N = 5$, growing with T and k, and among the defined cases it rejected a true null at the nominal 5 percent level in 8.9 to 17.7 percent of samples at $N = 10$ and 7.9 to 25.5 percent at $N = 5$ (at $N = 20$: 0 to 2.5 percent undefined, 5.5 to 12.3 percent rejections). With $N \leq 10$ the p-value should be read with this in mind.

References

Hausman, J. A. (1978). Specification tests in econometrics. *Econometrica*, 46(6), 1251-1271.

Pesaran, M. H., Shin, Y. and Smith, R. P. (1999). Pooled mean group estimation of dynamic heterogeneous panels. *Journal of the American Statistical Association*, 94(446), 621-634.

Examples

```
## Not run:
fit_mg  <- panel_ardl(y ~ x, data = d, id = "i", time = "t", estimator = "mg")
fit_pmg <- panel_ardl(y ~ x, data = d, id = "i", time = "t", estimator = "pmg")
hausman_test(fit_mg, fit_pmg)
hausman_test(fit_mg, fit_pmg, method = "ginv")
h <- hausman_test(fit_mg, fit_pmg, quiet = TRUE)
h$statistic

## End(Not run)
```

---

macro_data *Time Series Macroeconomic Data*

Description

A simulated time series dataset of macroeconomic variables for a single country over 100 quarters.

Usage

```
macro_data
```

Format

A data frame with 100 rows and 5 variables: quarter, gdp, inflation, investment, trade.

Source

Simulated data for demonstration purposes

Examples

```
data(macro_data)
head(macro_data)
```

`macro_panel` *Simulated Macroeconomic Panel Data*

Description

A simulated panel dataset of macroeconomic variables for 10 countries over 50 years.

Usage

```
macro_panel
```

Format

A data frame with 500 rows and 6 variables: country, year, gdp, inflation, investment, trade.

Source

Simulated data for demonstration purposes

Examples

```
data(macro_panel)
head(macro_panel)

# Estimate Panel ARDL
model <- panel_ardl(
  gdp ~ inflation + investment,
  data = macro_panel,
  id = "country",
  time = "year",
  estimator = "pmg"
)
```

`mtnardl` *Multiple-Threshold Nonlinear ARDL (MT-NARDL)*

Description

Extends NARDL to allow multiple threshold decomposition for capturing complex asymmetric relationships.

Usage

```
mtnardl(
  formula,
  data,
  thresholds = c(0),
  p = 1,
  q = 1,
  case = 3,
  auto_select = FALSE,
  n_thresholds = 2,
  bootstrap = FALSE,
  nboot = 2000,
  seed = NULL,
  decompose = NULL,
  threshold_type = c("value", "quantile"),
  bounds_k = c("partial_sums", "original"),
  nulls = c("separate", "joint"),
  horizon = 30
)

## S3 method for class 'mtnardl'
print(x, ...)

## S3 method for class 'mtnardl'
summary(object, ...)

## S3 method for class 'mtnardl'
plot(x, type = c("multipliers", "asymmetry"), ...)
```

Arguments

`formula`	A formula such as `gasoline ~ oil_price`; only plain variable names are accepted (create transformed variables in `data`).
`data`	A data frame containing the time series (no missing values).
`thresholds`	Numeric vector of threshold values (default: c(0)), or probabilities when `threshold_type = "quantile"`.
`p`	Integer. Number of lagged differences of y (default: 1)

q	Integer or vector (one entry per regressor). Number of differences of each regressor, lags 0 to q - 1 (default: 1)
case	Integer from 1-5 specifying deterministic components (default: 3)
auto_select	Logical. Automatically select thresholds (default: FALSE). With n_thresholds = 1 the threshold minimising the AIC over 0 and the deciles of the changes of the first decomposed variable is chosen; with more thresholds, equally spaced quantiles of those changes are used.
n_thresholds	Integer. Number of thresholds to select if auto_select = TRUE
bootstrap	Logical. Use bootstrap inference (default: FALSE)
nboot	Number of bootstrap replications (default: 2000)
seed	Random seed; used locally, the session's random number generator state is restored afterwards.
decompose	Character vector of regressors to decompose (default: all regressors). The other regressors enter linearly.
threshold_type	"value" (default) or "quantile".
bounds_k	"partial_sums" (default) or "original"; the k used for the bounds, see Details.
nulls	Bootstrap nulls, "separate" (default) or "joint"; see boot_ardl.
horizon	Largest horizon of the dynamic multipliers (default: 30).
x, object	An object of class "mtnardl"
...	Not used
type	For plot: "multipliers" or "asymmetry".

Details

Each decomposed regressor x is split into regimes according to the size of its change Δx_t: with thresholds $c_1 < \cdots < c_{R-1}$, regime r collects Δx_t when $c_{r-1} < \Delta x_t \leq c_r$ ($c_0 = -\infty$, $c_R = \infty$), and its partial sum is the cumulative sum of these changes, starting at zero at the first observation. For example, with thresholds c(-0.02, 0, 0.02) a variable is decomposed into large decreases, small decreases, small increases and large increases. With threshold_type = "quantile" the thresholds are probabilities and are converted to the sample quantiles of Δx_t of each decomposed variable (a quantile partition of the changes). Regressors not listed in decompose enter linearly.

The conditional error correction model is

$$\Delta y_t = \rho y_{t-1} + \sum_m \theta_m w_{m,t-1} + \sum_{i=1}^{p} \phi_i \Delta y_{t-i} + \sum_m \sum_{l=0}^{q-1} \pi_{ml} \Delta w_{m,t-l} + d_t + u_t,$$

where w are the regime partial sums and the linear regressors and d_t holds the deterministic terms of the case.

Bounds test. The statistics are the F test on all lagged levels (Fov), the t test on y_{t-1} and the F test on the lagged regressors (Find). The Kripfganz and Schneider (2020) bounds are computed with k equal to the number of level regressors (partial sums plus linear regressors) by default (bounds_k = "partial_sums"), or the number of original regressors (bounds_k = "original"). This is a

package choice pending verification against Pal and Mitra (2016) and Shin, Yu and Greenwood-Nimmo (2014); in a Monte Carlo with independent random walks the first choice gave sizes close to the nominal level. The bounds decision requires both F and t to exceed their I(1) bounds. With `bootstrap = TRUE` the recursive bootstrap of `boot_ardl` is used (separate nulls of Bertelli, Vacca and Zoia (2022) by default): x* is generated recursively, the regime partial sums are rebuilt from x* with the same thresholds, and the decision requires Fov, t and Find to reject (degenerate cases labelled).

Long run. The long-run coefficients $-\theta_m/\rho$ have delta-method standard errors computed with the full covariance matrix. For each decomposed variable the joint Wald test of equal long-run coefficients across regimes is the linear test $\theta_1 = \cdots = \theta_R$ (chi-squared with R - 1 degrees of freedom); pairwise tests and the delta-method differences of the long-run coefficients are also reported.

Dynamic multipliers. `multipliers$cumulative` gives the response of the level of y at horizons 0 to `horizon` to a permanent unit increase in each partial sum (or linear regressor) at horizon 0, computed by the full recursion of the estimated ECM; it converges to the long-run coefficient. The definition of the multiplier follows the cumulative multiplier of the NARDL literature and is pending verification against Shin, Yu and Greenwood-Nimmo (2014).

Value

An object of class "mtnardl" containing:

- `model`: The estimated MT-NARDL model
- `F_stat`, `t_stat`, `Find_stat`: bounds test statistics
- `bounds_test`: decision, message, method (and bootstrap p-values)
- `critical_values`: Kripfganz and Schneider (2020) bounds, with `bounds_k_value` the k used
- `long_run`: Long-run coefficients for each regime; `long_run_table` adds delta-method standard errors
- `thresholds`: Threshold values used (a list per decomposed variable in `thresholds_by_var`)
- `asymmetry_tests`: for each decomposed variable, pairwise Wald tests and the joint test (element `joint`)
- `lr_differences`: delta-method differences of the long-run coefficients between regimes
- `multipliers`: long-run multipliers and cumulative dynamic multipliers (horizons 0 to `horizon`)

References

Bertelli, S., Vacca, G. and Zoia, M. (2022). Bootstrap cointegration tests in ARDL models. *Economic Modelling*, 116, 105987. doi:10.1016/j.econmod.2022.105987

Kripfganz, S. and Schneider, D. C. (2020). Response surface regressions for critical value bounds and approximate p-values in equilibrium correction models. *Oxford Bulletin of Economics and Statistics*, 82(6), 1456-1481. doi:10.1111/obes.12377

Pal, D. and Mitra, S. K. (2016). Asymmetric oil product pricing in India: Evidence from a multiple threshold nonlinear ARDL model. *Economic Modelling*, 59, 314-328. doi:10.1016/j.econmod.2016.08.003

Shin, Y., Yu, B. and Greenwood-Nimmo, M. (2014). Modelling asymmetric cointegration and dynamic multipliers in a nonlinear ARDL framework. In R. C. Sickles and W. C. Horrace (Eds.), *Festschrift in Honor of Peter Schmidt* (pp. 281-314). Springer. doi:10.1007/9781489980083_9

Examples

```
data <- generate_oil_data(n = 120)
result1 <- mtnardl(gasoline ~ oil_price, data = data)
result1

result2 <- mtnardl(gasoline ~ oil_price, data = data,
                   thresholds = c(-1, 0, 1))
summary(result2)
result3 <- mtnardl(gasoline ~ oil_price, data = data,
                   thresholds = c(1/3, 2/3), threshold_type = "quantile",
                   bootstrap = TRUE, nboot = 49, seed = 1)
result3$bounds_test
```

---

oil_data *Oil Price and Gasoline Data*

Description

Weekly data on oil prices and retail gasoline prices, suitable for demonstrating asymmetric price transmission.

Usage

```
oil_data
```

Format

A data frame with 200 rows and 4 variables: week, gasoline, oil_price, exchange_rate.

Source

Simulated data for demonstration purposes

Examples

```
data(oil_data)

# Test for asymmetric price transmission
model <- qnardl(
  gasoline ~ oil_price + exchange_rate,
  data = oil_data,
  tau = c(0.25, 0.5, 0.75)
)
asymmetry_test(model, var = "oil_price")
```

panel_ardl *Panel ARDL Estimation (PMG / MG / DFE)*

Description

Estimate Panel ARDL models with Pooled Mean Group (PMG), Mean Group (MG), and Dynamic Fixed Effects (DFE) estimators following Pesaran, Shin and Smith (1999). This implementation replicates Stata's `xtpmg` command (Blackburne and Frank 2007).

Usage

```
panel_ardl(
  formula,
  data,
  id,
  time,
  p = 1,
  q = 1,
  estimator = c("pmg", "mg", "dfe"),
  ec = TRUE,
  trend = FALSE,
  start_time = NULL,
  cluster = FALSE,
  maxiter = 100,
  tol = 1e-06
)
```

Arguments

`formula`	A formula `y ~ x1 + x2 + ...`.
`data`	A data frame containing panel data.
`id`	Character. Name of the panel/group identifier column.
`time`	Character. Name of the time variable column.
`p`	Integer. Lags for dependent variable (default 1).
`q`	Integer or numeric vector. Lags for independent variables (default 1).
`estimator`	Character. One of `"pmg"`, `"mg"`, `"dfe"`.
`ec`	Logical. If `TRUE`, display error-correction form (default `TRUE`).
`trend`	Logical. Include a deterministic time trend (default `FALSE`).
`start_time`	Optional. Restrict estimation to observations at or after this time.
`cluster`	Logical. If `TRUE`, use cluster-robust standard errors where applicable (default `FALSE`).
`maxiter`	Integer. Maximum iterations for PMG optimization (default 100).
`tol`	Numeric. Convergence tolerance (default 1e-6).

Details

The model is specified as:

$$\Delta y_{it} = \phi_i(y_{i,t-1} - \theta_i' x_{it}) + \sum_{j=1}^{p-1} \lambda_{ij} \Delta y_{i,t-j} + \sum_{j=0}^{q-1} \delta_{ij}' \Delta x_{i,t-j} + \mu_i + \varepsilon_{it}$$

Three estimators are available:

- **PMG**: Constrains long-run coefficients across groups while allowing short-run coefficients to vary. Efficient under long-run homogeneity.
- **MG**: Estimates separate regressions per group and averages. Consistent but less efficient than PMG.
- **DFE**: Traditional dynamic fixed effects with all coefficients pooled. Consistent only under slope homogeneity.

Value

An object of class `"panel_ardl"` with components:

- `long_run`: long-run coefficients (theta).
- `short_run`: short-run coefficient summaries.
- `ec_coef`: error-correction coefficient (phi).
- `vcov`: variance-covariance matrix of long-run coefficients.
- `sigma`: residual standard error.
- `estimator`: the estimator used.

References

Pesaran, M. H., Shin, Y. and Smith, R. P. (1999). Pooled mean group estimation of dynamic heterogeneous panels. *Journal of the American Statistical Association*, 94(446), 621-634.

Blackburne, E. F. and Frank, M. W. (2007). Estimation of nonstationary heterogeneous panels. *Stata Journal*, 7(2), 197-208.

Examples

```
## Not run:
fit <- panel_ardl(y ~ x1 + x2, data = my_panel,
                  id = "country", time = "year",
                  p = 1, q = 1, estimator = "pmg")
summary(fit)

## End(Not run)
```

plot.boot_ardl *Plot Bootstrap Distribution*

Description

Plot the bootstrap distribution of test statistics

Usage

```
## S3 method for class 'boot_ardl'
plot(x, which = "both", ...)
```

Arguments

x	An object of class "boot_ardl"
which	Character. "F", "t", or "both" (default)
...	Additional arguments passed to plotting functions

plot.fbnardl *Plot method for fbnardl objects*

Description

Plot method for fbnardl objects

Usage

```
## S3 method for class 'fbnardl'
plot(x, type = c("multipliers", "kstar"), ...)
```

Arguments

x	An object of class "fbnardl"
type	"multipliers" (dynamic multipliers with bands) or "kstar" (sum of squared residuals by Fourier frequency).
...	Not used

pnardl *Panel Nonlinear ARDL (Panel NARDL)*

Description

Panel data estimation with asymmetric/nonlinear ARDL specification using PMG, MG, or DFE estimators.

Usage

```
pnardl(formula, data, id, time, p = 1, q = 1,
       estimator = c("pmg", "mg", "dfe"),
       threshold = 0,
       effect = c("individual", "time", "twoways"),
       bootstrap = FALSE, nboot = 500)

## S3 method for class 'pnardl'
print(x, ...)
## S3 method for class 'pnardl'
summary(object, ...)
```

Arguments

formula	A formula specifying the model: gdp ~ investment + trade + ...
data	A data frame in long panel format
id	Character. Name of the group/panel identifier variable
time	Character. Name of the time variable
p	Integer. Number of lags for dependent variable (default: 1)
q	Integer or vector. Number of lags for independent variables (default: 1)
estimator	Character. One of "pmg", "mg", or "dfe" (default: "pmg")
threshold	Numeric. Threshold for asymmetric decomposition (default: 0)
effect	Character. Type of effects: "individual", "time", "twoways" (default: "individual")
bootstrap	Logical. Use bootstrap for inference (default: FALSE)
nboot	Number of bootstrap replications (default: 500)
x, object	An object of class "pnardl"
...	Additional arguments passed to methods

Details

The Panel NARDL model extends the Shin et al. (2014) NARDL framework to panel data settings. It allows for asymmetric short-run and long-run effects while controlling for cross-sectional heterogeneity.

Three estimators are available:

- **PMG**: Pooled Mean Group - constrains long-run coefficients to be homogeneous while allowing short-run heterogeneity
- **MG**: Mean Group - estimates separate models for each unit and averages coefficients
- **DFE**: Dynamic Fixed Effects - assumes all coefficients are homogeneous except fixed effects

Value

An object of class "pnardl" containing:

`coefficients`	Estimated coefficients
`long_run_pos`	Long-run coefficients for positive changes
`long_run_neg`	Long-run coefficients for negative changes
`short_run`	Short-run coefficients
`ec_coef`	Error correction coefficient
`asymmetry_test`	Wald test for long-run asymmetry
`unit_results`	Individual unit estimation results (for MG)
`hausman`	Hausman test comparing PMG vs MG

References

Shin, Y., Yu, B. and Greenwood-Nimmo, M. (2014). Modelling asymmetric cointegration and dynamic multipliers in a nonlinear ARDL framework.

Pesaran, M. H., Shin, Y. and Smith, R. P. (1999). Pooled mean group estimation of dynamic heterogeneous panels. Journal of the American Statistical Association.

See Also

`panel_ardl`, `qnardl`, `mtnardl`

Examples

```
# Generate panel data
data <- generate_panel_data(n_groups = 10, n_time = 40)

# Panel NARDL with PMG estimator
result <- pnardl(
  gdp ~ investment + trade,
  data = data,
  id = "country",
  time = "year",
  estimator = "pmg"
)
summary(result)

# Panel NARDL with MG estimator
result_mg <- pnardl(
  gdp ~ investment,
  data = data,
```

```
  id = "country",
  time = "year",
  estimator = "mg"
)

# Panel NARDL with bootstrap standard errors (nboot kept small here;
# use several hundred replications in applied work)
result_boot <- pnardl(
  gdp ~ investment,
  data = data,
  id = "country",
  time = "year",
  bootstrap = TRUE,
  nboot = 20
)
```

`print.boot_ardl` *Print method for boot_ardl*

Description

Print method for boot_ardl

Usage

```
## S3 method for class 'boot_ardl'
print(x, ...)
```

Arguments

`x`	An object of class "boot_ardl"
`...`	Not used

`print.panel_ardl` *Print Method for panel_ardl Objects*

Description

Concise display of a fitted `panel_ardl` model.

Usage

```
## S3 method for class 'panel_ardl'
print(x, ...)
```

Arguments

x	A panel_ardl object.
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

---

pss_critical_values *Critical Value Bounds for the PSS Bounds Test*

Description

Critical value bounds for the F and t statistics of the Pesaran, Shin and Smith (2001) bounds test, computed from the response surface regressions of Kripfganz and Schneider (2020). With n missing the asymptotic bounds are returned; with n (and sr) the finite-sample bounds.

Usage

```
pss_critical_values(k, case = 3, level = "5%", n = NULL, sr = 0)
```

Arguments

k	Number of regressors in levels (excluding the lagged dependent variable)
case	PSS case (1-5)
level	Significance level: "10%", "5%", "2.5%" or "1%"
n	Number of observations (optional)
sr	Number of short-run coefficients (regressors other than the deterministic terms and the lagged levels); used with n

Value

A list with F_bounds and t_bounds (each a list with I0 and I1), k, case and level. The t statistic is not tabulated for cases 2 and 4; the values of cases 3 and 5 are used, as in Kripfganz and Schneider (2020).

References

Pesaran, M. H., Shin, Y. and Smith, R. J. (2001). Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3), 289-326. doi:10.1002/jae.616

Kripfganz, S. and Schneider, D. C. (2020). Response surface regressions for critical value bounds and approximate p-values in equilibrium correction models. *Oxford Bulletin of Economics and Statistics*, 82(6), 1456-1481. doi:10.1111/obes.12377

Examples

```
pss_critical_values(k = 2, case = 3)
pss_critical_values(k = 2, case = 3, n = 80, sr = 4)
```

---

qnardl *Quantile Nonlinear ARDL (QNARDL)*

Description

Estimate Quantile Nonlinear ARDL models combining distributional and asymmetric effects.

Usage

```
qnardl(formula, data, tau = c(0.25, 0.5, 0.75),
       p = 1, q = 1, decompose = NULL, trend = FALSE)

## S3 method for class 'qnardl'
summary(object, ...)

## S3 method for class 'qnardl'
print(x, ...)

## S3 method for class 'qnardl'
plot(x, var = NULL, type = "long_run", ...)
```

Arguments

formula	A formula specifying the model: gdp ~ investment + trade + ...
data	A data frame containing the time series data
tau	Numeric vector of quantiles to estimate (default: c(0.25, 0.5, 0.75))
p	Integer. Number of lags for dependent variable (default: 1)
q	Integer or vector. Number of lags for independent variables (default: 1)
decompose	Character vector. Variables to decompose into +/- components. Default is all x variables.
trend	Logical. Include time trend (default: FALSE)
object, x	An object of class "qnardl"
var	Variable to plot (default: first decomposed variable)
type	"long_run" or "asymmetry"
...	Additional arguments

Details

This function implements the QNARDL model which extends the NARDL framework of Shin, Yu and Greenwood-Nimmo (2014) to a quantile regression setting.

The model allows for:

- **Asymmetric effects**: Positive and negative changes in X can have different impacts on Y
- **Distributional heterogeneity**: Effects can vary across quantiles of the conditional distribution

The partial sum decomposition separates each regressor into positive and negative components.

Value

An object of class "qnardl" containing:

`coefficients`	Estimated coefficients for each quantile
`long_run_pos`	Positive long-run coefficients by quantile
`long_run_neg`	Negative long-run coefficients by quantile
`short_run`	Short-run coefficients by quantile
`ec_coef`	Error correction coefficients by quantile
`asymmetry_test`	Wald test for long-run asymmetry
`tau`	Quantiles estimated

References

Cho, J. S., Kim, T. H. and Shin, Y. (2015). Quantile cointegration in the autoregressive distributed-lag modeling framework. Journal of Econometrics, 188(1), 281-300.

Shin, Y., Yu, B. and Greenwood-Nimmo, M. (2014). Modelling asymmetric cointegration and dynamic multipliers in a nonlinear ARDL framework. In Festschrift in Honor of Peter Schmidt (pp. 281-314). Springer.

See Also

`asymmetry_test`, `dynamic_multipliers`, `generate_oil_data`

Examples

```
# Generate oil price data
data <- generate_oil_data(n = 200)

# Estimate QNARDL
model <- qnardl(
  gasoline ~ oil_price + exchange_rate,
  data = data,
  tau = c(0.1, 0.25, 0.5, 0.75, 0.9)
)
summary(model)
plot(model, var = "oil_price")
```

`rardl` *Rolling and Recursive ARDL (R-ARDL)*

Description

Time-varying ARDL bounds test using rolling or recursive windows to detect structural changes in cointegration relationships.

Usage

```
rardl(formula, data, method = c("rolling", "recursive"),
      window = 50, min_obs = 40, p = 1, q = 1, case = 3,
      parallel = FALSE, ncores = 2)

## S3 method for class 'rardl'
print(x, ...)
## S3 method for class 'rardl'
summary(object, ...)
## S3 method for class 'rardl'
plot(x, type = c("F", "ec", "lr", "all"), ...)
```

Arguments

`formula`	A formula specifying the model: gdp ~ investment + trade + ...
`data`	A data frame containing the time series data
`method`	Character. Either "rolling" or "recursive" (default: "rolling")
`window`	Integer. Window size for rolling method (default: 50)
`min_obs`	Integer. Minimum observations for recursive method (default: 40)
`p`	Integer. Number of lags for dependent variable (default: 1)
`q`	Integer or vector. Number of lags for independent variables (default: 1)
`case`	Integer from 1-5 specifying deterministic components (default: 3)
`parallel`	Logical. Use parallel processing (default: FALSE)
`ncores`	Integer. Number of cores for parallel processing
`x, object`	An object of class "rardl"
`type`	Character. Plot type: "F", "ec", "lr", or "all"
`...`	Additional arguments passed to methods

Details

The Rolling ARDL (R-ARDL) approach implements time-varying bounds testing to detect structural changes in cointegration relationships. Two methods are available:

- **Rolling**: Fixed-size window that moves through the sample

- **Recursive**: Expanding window starting from initial observations

This is useful for detecting:

- Time-varying cointegration relationships
- Structural breaks in long-run parameters
- Changes in error correction speed

Value

An object of class "rardl" containing:

`F_stats`	Time series of F-statistics
`t_stats`	Time series of t-statistics (EC coefficient)
`cointegration`	Time series of cointegration decisions
`ec_coefs`	Time-varying error correction coefficients
`lr_coefs`	Time-varying long-run coefficients
`dates`	Index/dates for each window
`breaks`	Detected structural break points

References

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

See Also

`boot_ardl`, `aardl`

Examples

```
# Generate example data
data <- generate_ts_data(n = 300)

# Rolling ARDL with 60-observation window
roll_result <- rardl(gdp ~ investment + trade, data = data, method = "rolling", window = 60)
plot(roll_result, type = "all")
summary(roll_result)

# Recursive ARDL
rec_result <- rardl(gdp ~ investment + trade, data = data, method = "recursive", min_obs = 50)
plot(rec_result, type = "F")
```

summary.boot_ardl *Summary method for boot_ardl*

Description

Summary method for boot_ardl

Usage

```
## S3 method for class 'boot_ardl'
summary(object, ...)
```

Arguments

object	An object of class "boot_ardl"
...	Not used

summary.panel_ardl *Summary Method for panel_ardl Objects*

Description

Detailed summary of a fitted panel_ardl model with long-run, short-run, and error-correction coefficients.

Usage

```
## S3 method for class 'panel_ardl'
summary(object, digits = 6, ...)
```

Arguments

object	A panel_ardl object.
digits	Integer. Decimal digits to display (default 6).
...	Additional arguments (currently ignored).

Value

Invisibly returns the summary as a list.

Index

∗ **datagen**
  generate_oil_data, 21
  generate_panel_data, 22
  generate_ts_data, 23
∗ **datasets**
  macro_data, 26
  macro_panel, 27
  oil_data, 31
∗ **htest**
  asymmetry_test, 9
∗ **models**
  pnardl, 35
  rardl, 41
∗ **nonlinear**
  qnardl, 39
∗ **package**
  ardlverse-package, 3
∗ **panel**
  pnardl, 35
∗ **regression**
  ardl_diagnostics, 7
  dynamic_multipliers, 13
  fourier_ardl, 18
  qnardl, 39
∗ **structural breaks**
  rardl, 41
∗ **ts**
  ardl_diagnostics, 7
  fourier_ardl, 18

aardl, *3*, 4, *42*
ardl_diagnostics, *3*, 7
ardlverse *(*ardlverse-package*)*, 3
ardlverse-package, 3
asymmetry_test, *3*, 9, *13*, *21*, *40*

boot_ardl, *3*, *5, 6*, *8*, 10, *16*, *23*, *29, 30*, *42*

coef.fbnardl *(*fbnardl*)*, 14

dynamic_multipliers, *3*, *9*, 13, *40*

fbnardl, *3*, 14
fourier_ardl, *3*, *8*, 18, *23*
fourier_bounds_test, *6*, *19*, 20

generate_oil_data, *3*, 21, *40*
generate_panel_data, *3*, 22, *23*
generate_ts_data, *3*, *19*, *22*, 23

hausman_test, *3*, 24

macro_data, 26
macro_panel, 27
mtnardl, *3*, 28, *36*

nobs.fbnardl *(*fbnardl*)*, 14

oil_data, 31

panel_ardl, *3*, *22*, 32, *36*
plot.ardl_diagnostics
        *(*ardl_diagnostics*)*, 7
plot.boot_ardl, 34
plot.fbnardl, 34
plot.fourier_ardl *(*fourier_ardl*)*, 18
plot.mtnardl *(*mtnardl*)*, 28
plot.qnardl *(*qnardl*)*, 39
plot.rardl *(*rardl*)*, 41
pnardl, 35
print.aardl *(*aardl*)*, 4
print.boot_ardl, 37
print.fbnardl *(*fbnardl*)*, 14
print.fourier_ardl *(*fourier_ardl*)*, 18
print.mtnardl *(*mtnardl*)*, 28
print.panel_ardl, 37
print.pnardl *(*pnardl*)*, 35
print.qnardl *(*qnardl*)*, 39
print.rardl *(*rardl*)*, 41
pss_critical_values, *3*, 38

qnardl, *3*, *9*, *13*, *21*, *36*, 39

rardl, 41

summary.aardl *(*aardl*)*, 4
summary.ardl_diagnostics
*(*ardl_diagnostics*)*, 7
summary.boot_ardl, 43
summary.fbnardl *(*fbnardl*)*, 14
summary.fourier_ardl *(*fourier_ardl*)*, 18
summary.mtnardl *(*mtnardl*)*, 28
summary.panel_ardl, 43
summary.pnardl *(*pnardl*)*, 35
summary.qnardl *(*qnardl*)*, 39
summary.rardl *(*rardl*)*, 41

vcov.fbnardl *(*fbnardl*)*, 14