

# Package ‘dynamac’

November 6, 2019

**Title** Dynamic Simulation and Testing for Single-Equation ARDL Models

**Version** 0.1.10

**Maintainer** Soren Jordan <sorenjordanpols@gmail.com>

**Description** While autoregressive distributed lag (ARDL) models allow for extremely flexible dynamics, interpreting substantive significance of complex lag structures remains difficult. This package is designed to assist users in dynamically simulating and plotting the results of various ARDL models. It also contains post-estimation diagnostics, including a test for cointegration when estimating the error-correction variant of the autoregressive distributed lag model (Pesaran, Shin, and Smith 2001 <doi:10.1002/jae.616>).

**URL** <https://github.com/andyphilips/dynamac/>

**BugReports** <https://github.com/andyphilips/dynamac/issues>

**Imports** MASS, lmtest

**Suggests** urca, knitr, rmarkdown, testthat

**Depends** R (>= 3.0.1)

**License** GPL (>=2)

**Encoding** UTF-8

**LazyData** true

**BuildManual** yes

**RoxygenNote** 6.1.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Soren Jordan [aut, cre, cph],  
Andrew Q. Philips [aut]

## R topics documented:

dshift	2
dynardl	3
dynardl.all.plots	5
dynardl.auto.correlated	6
dynardl.simulation.plot	7
france.data	9
ineq	10
ldshift	10

lshift . . . . .	11
pssbounds . . . . .	12
summary.dynardl . . . . .	13
supreme.sup . . . . .	14

<b>Index</b>	<b>15</b>
--------------	-----------

---

dshift	<i>Take first difference of a series</i>
--------	--

---

## Description

Take first difference of a series

## Usage

```
dshift(x)
```

## Arguments

x                      a series to be differenced

## Details

dshift assumes that the series are ordered, that there is no missing data, and that the time intervals are even

## Value

the differenced series

## Author(s)

Soren Jordan and Andrew Q. Philips

## Examples

```
x.var <- seq(0, 50, 5)
d.x.var <- dshift(x.var)
head(x.var)
head(d.x.var)
```

dynardl

*Estimate and simulate ARDL model***Description**

Estimate autoregressive distributed lag model and simulate interesting values (if desired)

**Usage**

```
dynardl(formula, data = list(), lags = list(), diffs = list(),
  lagdiffs = list(), levels = list(), ec = FALSE, trend = FALSE,
  constant = TRUE, modelout = FALSE, simulate = FALSE,
  shockvar = list(), shockval = sd(data[[shockvar]], na.rm = T),
  time = 10, qoi = "mean", forceset = NULL, range = 20,
  burnin = 20, sims = 1000, sig = 95, expectedval = FALSE,
  fullsims = FALSE)
```

**Arguments**

formula	a symbolic description of the model to be estimated. ARDL models are estimated using linear regression
data	an optional data frame or list containing the the variables in the model
lags	a list of variables and their corresponding lags to be estimated
diffs	a vector of variables to be differenced. Only first differences are supported
lagdiffs	a list of variables to be included in lagged differences
levels	a vector of variables to be included in levels
ec	estimate model in error-correction form, (i.e., y appears in first-differences). By default, ec is set to FALSE, meaning y will appear in levels.
trend	include a linear time trend. The default is FALSE
constant	include a constant. The default is TRUE
modelout	print the regression estimates in the console
simulate	simulate the reponse. Otherwise, just the regression model will be estimated. If simulate = FALSE, options shockvar, shockval, time, qoi, forceset, range, burnin, sims, sig, expectedval, and fullsims are ignored. The default is FALSE so that users can build models without needing to simulate the results each time. When simulate = TRUE, users are highly encouraged to set a seed before simulation, as with any stochastic exercise
shockvar	the variable to be shocked in the counterfactual simulation. There is no default
shockval	the amount by which the shockvar should be shocked. The default is one standard deviation of the shocked variable
time	the time period in the simulation for the variable to be shocked
qoi	summarize the response of the dependent variable with the mean or the median. Although the default is mean, if there is underlying skew in the distribution, it might be better summarized by median

forceset	by default, in the simulations, variables in levels will be set to their means; variables in differences will be set to 0. Alternatively, users can set any variable in the model to a different value using a list in forceset. These values can be any user-defined value, including means, medians, percentiles, or other values of interest
range	the range of the simulation to be conducted
burnin	the number of time periods to disregard before recording the values. These do not include the range; in other words, they take place before the range specified above. Users can increase the number of burnin periods, but probably should not decrease them. The default is 20
sims	the number of simulations to use in creating the quantities of interest (the response of the dependent variable). The default is 1000
sig	the significance level (1 - p) that the user wants for the simulations. The default level is 95% significance (sig = 95)
expectedval	if this is TRUE, the simulation will record the expected values of across the sims by averaging errors. The default is FALSE, since expected values do not account for stochastic error present in the model itself
fullsims	whether all of the raw simulations should be stored in the model object. These are required for some of the more advanced plotting functions, especially those that use the simulations to derive confidence intervals about the size of the period-over-period differences. The default is FALSE

## Details

Estimate an auto-regressive distributed lag model. Moreover, enable a graphical interpretation of the results (through [dynardl.simulation.plot](#)) by simulating the response of the dependent variable to shocks in one of the regressors

## Value

dynardl should always return an estimated model. It may or may not be simulated, according to the user. But the relevant regression output, model residuals (which can be tested for autocorrelation), and simulated response (if created) are stored in a list if the model is assigned to an object

## Author(s)

Soren Jordan and Andrew Q. Philips

## Examples

```
# Using the inequality data from dynamac
ardl.model <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  diffs = c("incshare10", "urate"),
  ec = TRUE, simulate = FALSE)
summary(ardl.model)

# Adding a lagged difference of the dependent variable
ardl.model.2 <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  diffs = c("incshare10", "urate"),
  lagdiffs = list("concern" = 1),
  ec = TRUE, simulate = FALSE)
```

```
summary(ardl.model.2)

# Does not work: levels and diffs must appear as a vector

ardl.model.3 <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  levels = list("urate" = 1),
  diffs = list("incshare10" = 1, "urate" = 1),
  lagdiffs = list("concern" = 1),
  ec = TRUE, simulate = FALSE)

ardl.model.3 <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  levels = c("urate"),
  diffs = c("incshare10", "urate"),
  lagdiffs = list("concern" = 1),
  ec = TRUE, simulate = FALSE)
```

---

dynardl.all.plots	<i>Combine all of the potential plots of a simulated response in a <a href="#">dynardl</a> model</i>
-------------------	--

---

## Description

Combine all of the potential plots of a simulated response in a [dynardl](#) model

## Usage

```
dynardl.all.plots(x, type = "area", bw = FALSE, last.period = NULL,
  start.period = 1, tol = (abs(x$model$ymean) * 0.01),
  abs.errors = "none", ...)
```

## Arguments

x	a dynardl model with a simulation to be plotted. Since all plots includes absolute cumulative differences, fullsims must be TRUE in the dynardl simulation
type	whether the plot should be an area plot (area) or a spike plot (spike)
bw	should the colors be in black and white (for publication)? The default is FALSE
last.period	when deciding when to stop calculating the absolute value of the shocks to the dependent variable, you can specify a specific period in which to stop calculating absolute cumulative differences. Specify a tol or a last.period. If both are specified, last.period overrides tol
start.period	which period of the simulation to begin the plot with. You can view the equilibrating behavior of the dependent variable, or you can skip forward in time (maybe to just before the shock). The default is 1 (the first period of the simulation)
tol	when deciding when to stop calculating the absolute value of the shocks to the dependent variable, you can specify the minimum amount of movement required to qualify as a non-noise change over time periods (for calculating absolute cumulative differences). The default is 0.1 percent of the mean of the dependent variable. Specify a tol or a last.period. If both are specified, last.period overrides tol

abs.errors	when calculating confidence for the absolute cumulative effect, should differences accumulate in each time time period (cumulate, which could be explosive if the error in the model is large), should differences be observed at each time (within.period, which will have smaller values in equilibrium than when changing), or should only the values be plotted (none)
...	other arguments to be passed to the call to plot. Use caution, as they will be passed to all plots

### Details

When running `dynardl`, `simulate` must be `TRUE` so that there is a simulation to plot. Also, `fullsims` must be `TRUE` as the plot will contain absolute cumulative differences. See [dynardl.simulation.plot](#) for arguments to the individual plotting types

### Value

a 2 x 3 grid of the plots of the simulated `dynardl` model effects plots

### Author(s)

Soren Jordan and Andrew Q. Philips

### Examples

```
# Using the ineq data in dynamac
# Shocking Income Top 10
set.seed(1)
ardl.model <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  diffs = c("incshare10", "urate"),
  lagdiffs = list("concern" = 1),
  ec = TRUE, simulate = TRUE, range = 30,
  shockvar = "incshare10", fullsims = TRUE)

# Shows all of the potential responses
dynardl.all.plots(ardl.model)
# Same plot, but with spikeplot
dynardl.all.plots(ardl.model, type = "spike")
# Grayscale plots
dynardl.all.plots(ardl.model, bw = TRUE)
```

---

`dynardl.auto.correlated`

*Run a variety of autocorrelation tests on the residuals from a [dynardl](#) model*

---

### Description

Run a variety of autocorrelation tests on the residuals from a [dynardl](#) model

### Usage

```
dynardl.auto.correlated(x, bg.type = "Chisq", digits = 3,
  order = NULL, object.out = FALSE)
```

**Arguments**

<code>x</code>	a dynardl model
<code>bg.type</code>	a character string for the type of Breusch-Godfrey test to run. The default is <code>Chisq</code> : the Chisq test statistic. The other option is <code>F</code> : the F-test statistic
<code>digits</code>	the number of digits to round to when showing output. The default is 3
<code>order</code>	the maximum order of serial autocorrelation to test when executing the Breusch-Godfrey test
<code>object.out</code>	if <code>TRUE</code> , and <code>dynardl.auto.correlated</code> is assigned to an object, the AIC, BIC, and results will be stored for the user's convenience

**Details**

This is a simple and convenient way to test whether the residuals from the dynardl model are white noise. As an aside, this is also why dynardl has a `simulate = FALSE` argument: users can ensure the model has white noise residuals before estimating a potentially time-intensive simulation. The output also reminds the user of the null hypotheses for the autocorrelation tests

**Value**

The results of autocorrelation tests

**Author(s)**

Soren Jordan and Andrew Q. Philips

**Examples**

```
# Using the ineq data from dynamac
ardl.model <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  diffs = c("incshare10", "urate"),
  lagdiffs = list("concern" = 1),
  ec = TRUE, simulate = FALSE)
dynardl.auto.correlated(ardl.model)
```

---

`dynardl.simulation.plot`

*Create a plot of a simulated response in a [dynardl](#) model*

---

**Description**

Create a plot of a simulated response in a [dynardl](#) model

**Usage**

```
dynardl.simulation.plot(x, type = "area", response = "levels",
  bw = FALSE, last.period = NULL, tol = (abs(x$model$ymean) * 0.01),
  start.period = 1, abs.errors = "none", ylab = "", xlab = "", ...)
```

**Arguments**

<code>x</code>	a dynardl model with a simulation to be plotted
<code>type</code>	whether the plot should be an area plot (area) or a spike plot (spike)
<code>response</code>	whether the plot of the response should be shown in levels of the dependent variable ( <code>levels</code> ), levels from the mean of the dependent variable ( <code>levels.from.mean</code> ), period-over-period changes in the dependent variable ( <code>diffs</code> ), the absolute value of the (decreasing) change in the dependent variable in each time period due to the shock ( <code>shock.effect.decay</code> ), the sum of the period-over-period changes ( <code>cumulative.diffs</code> ), or the absolute value of the cumulative differences (where negative effects are treated as positive) ( <code>cumulative.abs.diffs</code> ). The default is <code>levels</code>
<code>bw</code>	should the colors be in black and white (for publication)? The default is <code>FALSE</code>
<code>last.period</code>	when deciding when to stop calculating the absolute value of the shocks to the dependent variable, you can specify a specific period in which to stop calculating absolute cumulative differences. Specify a <code>tol</code> or a <code>last.period</code> . If both are specified, <code>last.period</code> overrides <code>tol</code>
<code>tol</code>	when deciding when to stop calculating the absolute value of the shocks to the dependent variable, you can specify the minimum amount of movement required to qualify as a non-noise change over time periods (for calculating absolute cumulative differences). The default is 0.1 percent of the mean of the dependent variable. Specify a <code>tol</code> or a <code>last.period</code> . If both are specified, <code>last.period</code> overrides <code>tol</code>
<code>start.period</code>	which period of the simulation to begin the plot with. You can view the equilibrating behavior of the dependent variable, or you can skip forward in time (maybe to just before the shock). The default is 1 (the first period of the simulation)
<code>abs.errors</code>	when calculating confidence for the absolute cumulative effect, should differences accumulate in each time time period ( <code>cumulate</code> , which could be explosive if the error in the model is large), should differences be observed at each time ( <code>within.period</code> , which will have smaller values in equilibrium than when changing), or should only the values be plotted ( <code>none</code> ). The default is <code>none</code>
<code>ylab</code>	a user-defined y-label to be used instead of the default
<code>xlab</code>	a user-defined x-label to be used instead of the default
<code>...</code>	other arguments to be passed to the call to <code>plot</code>

**Details**

When running `dynardl`, `simulate` must be `true` so that there is a simulation to plot. For types `cumulative.diffs` and `cumulative.abs.diffs`, `fullsims` must be `TRUE` in the `dynardl` simulation

**Value**

a plot of the simulated dynardl model

**Author(s)**

Soren Jordan and Andrew Q. Philips



## Examples

```
# Using the ineq data in dynamac
# Shocking Income Top 10
set.seed(1)
ardl.model <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  diffs = c("incshare10", "urate"),
  lagdiffs = list("concern" = 1),
  ec = TRUE, simulate = TRUE, range = 30,
  shockvar = "incshare10", fullsims = TRUE)

# Shows absolute levels
dynardl.simulation.plot(ardl.model)
# Shows changes from mean level
dynardl.simulation.plot(ardl.model, response = "levels.from.mean")
# Same plot, but with spikeplot
dynardl.simulation.plot(ardl.model, type = "spike", response = "levels.from.mean")
# Grayscale plots
dynardl.simulation.plot(ardl.model, bw = TRUE)
```

france.data

*Data on French Energy Consumption and GDP*

## Description

Data on GDP are from World Bank World Development Indicators. Data on energy consumption are from the PB Statistical Review of World Energy (June 2018).

## Usage

```
data(france.data)
```

## Format

A data frame with 53 rows and 4 variables:

**country** Country

**year** Year

**lnGDP\_cons2010USD** ln(GDP), constant 2010 US dollars

**lnenergy** ln(energy consumption), millions tons oil equivalent

ineq

*Data on public concern about economic inequality***Description**

A dataset from: Wright, Graham. 2017. "The political implications of American concerns about economic inequality." Political Behavior 40(2): 321-346.

**Usage**

```
data(ineq)
```

**Format**

A data frame with 49 rows and 9 variables:

**year** Year

**mood** Public mood liberalism

**urate** Unemployment rate

**concern** Concern about economic inequality

**demcontrol** Democratic control of congress

**incshare10** Proportion of income of top 10 percent

**csentiment** Consumer sentiment

**incshare01** Proportion of income of top 1 percent

**Source**

<http://dx.doi.org/10.7910/DVN/UYUU9G>

ldshift

*Take the lagged first difference of a series***Description**

Take the lagged first difference of a series

**Usage**

```
ldshift(x, 1)
```

**Arguments**

**x** a series to be differenced

**1** the number of lags

**Details**

ldshift assumes that the series are ordered, that there is no missing data, and that the time intervals are even

**Value**

the lagged differenced series

**Author(s)**

Soren Jordan and Andrew Q. Philips

**Examples**

```
x.var <- runif(50)
ld.1.x.var <- ldshift(x.var, 1)
ld.2.x.var <- ldshift(x.var, 2)
head(x.var)
head(ld.1.x.var)
head(ld.2.x.var)
```

---

lshift

*Take lag transformation of a series*

---

**Description**

Take lag transformation of a series

**Usage**

```
lshift(x, 1)
```

**Arguments**

x	a series to be lagged
1	the number of lags

**Details**

lshift assumes that the series are ordered, that there is no missing data, and that the time intervals are even

**Value**

the lagged series

**Author(s)**

Soren Jordan and Andrew Q. Philips

**Examples**

```
x.var <- runif(50)
l.1.x.var <- lshift(x.var, 1)
l.2.x.var <- lshift(x.var, 2)
head(x.var)
head(l.1.x.var)
head(l.2.x.var)
```

pssbounds

*Perform Pesaran, Shin and Smith (2001) cointegration test***Description**

Perform Pesaran, Shin and Smith (2001) cointegration test

**Usage**

```
pssbounds(data = list(), obs = NULL, fstat = NULL, tstat = NULL,
  case = NULL, k = NULL, digits = 3, object.out = FALSE)
```

**Arguments**

data	an optional <a href="#">dynardl</a> model. This option is highly recommended. Users are welcome to supply their own case, t-statistic, F-statistic, and observations, but it is easier to have the model determine these quantities
obs	number of observations
fstat	F-statistic of the joint test that variables in levels are equal to zero: the specific restriction tested is $1.y + 1.x1 + 1.x2 + \dots + 1.xk = 0$
tstat	t-statistic of the lagged dependent variable
case	specify certain restrictions on the constant and trend terms, since critical values differ by case. Case I: no intercept or trend, Case II: restricted intercept, no trend, Case III: unrestricted intercept with no trend, Case IV: unrestricted intercept and restricted trend, Case V: unrestricted intercept and trend. Case III is most frequently specified
k	number of regressors appearing in levels in the estimated model, not including the lagged dependent variable
digits	the number of digits to round to when showing output. The default is 3
object.out	if TRUE, and pssbounds is assigned to an object, the test quantities will be stored for the user's convenience

**Details**

pssbounds performs post-estimation cointegration testing using the bounds testing procedure from Pesaran, Shin, and Smith (2001). Since test statistics vary based on the number of k regressors, length of the series, these are required, in addition to F- and t-statistics

**Author(s)**

Soren Jordan and Andrew Q. Philips

**Examples**

```
# Using the ineq data from dynamac
# We can get all the values by hand
ardl.model <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  diffs = c("incshare10", "urate"),
  lagdiffs = list("concern" = 1),
```

```

      ec = TRUE, simulate = FALSE)
summary(ardl.model)
pssbounds(obs = 47, fstat = 7.01578, tstat = -3.223, case = 3, k = 1)

# Or just pass a dynardl model.
pssbounds(ardl.model)

```

summary.dynardl

*Enable summary calls to [dynardl](#) model objects***Description**

Enable summary calls to [dynardl](#) model objects

**Usage**

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

**Arguments**

object	a dynardl model
...	additional arguments in the generic summary call

**Details**

dynardl, by default, stores regression results in `foo$model`. This calls those results directly with `summary`

**Value**

A summary of the fitted ARDL model.

**Author(s)**

Soren Jordan and Andrew Q. Philips

**Examples**

```

# Using the ineq data from dynamac
ardl.model <- dynardl(concern ~ incshare10 + urate, data = ineq,
  lags = list("concern" = 1, "incshare10" = 1),
  diffs = c("incshare10", "urate"),
  lagdiffs = list("concern" = 1),
  ec = TRUE, simulate = FALSE)
summary(ardl.model)

```

---

supreme.sup

*Data on US Supreme Court Approval*


---

### Description

A dataset from: Durr, Robert H., Andrew D. Martin, and Christina Wolbrecht. 2000. "Ideological divergence and public support for the Supreme Court." *American Journal of Political Science* 44(4): 768-776.

### Usage

```
data(supreme.sup)
```

### Format

A data frame with 42 rows and 9 variables:

**dcalc** Supreme Court support

**l\_dcalc** Lagged Supreme Court support

**iddiv** Ideological divergence

**mooddev** Mean deviation of Mood

**dirdev** Mean deviation of percent liberal decisions

**sg** Rulings against Solicitor General's amicus briefs

**laws** Laws declared unconstitutional

**presapp** Approval of president

**congapp** Approval of Congress

### Source

<http://dx.doi.org/10.2307/2669280>

# Index

- \*Topic **ardl**
  - `dynardl`, [3](#)
- \*Topic **cointegration**
  - `pssbounds`, [12](#)
- \*Topic **datasets**
  - `france.data`, [9](#)
  - `ineq`, [10](#)
  - `supreme.sup`, [14](#)
- \*Topic **estimation**
  - `dynardl`, [3](#)
- \*Topic **simulation**
  - `dynardl`, [3](#)
- \*Topic **utilities**
  - `dshift`, [2](#)
  - `dynardl.all.plots`, [5](#)
  - `dynardl.auto.correlated`, [6](#)
  - `dynardl.simulation.plot`, [7](#)
  - `ldshift`, [10](#)
  - `lshift`, [11](#)
  - `summary.dynardl`, [13](#)

`dshift`, [2](#)  
`dynardl`, [3](#), [5–7](#), [12](#), [13](#)  
`dynardl.all.plots`, [5](#)  
`dynardl.auto.correlated`, [6](#)  
`dynardl.simulation.plot`, [4](#), [6](#), [7](#)

`france.data`, [9](#)

`ineq`, [10](#)

`ldshift`, [10](#)  
`lshift`, [11](#)

`pssbounds`, [12](#)

`summary.dynardl`, [13](#)  
`supreme.sup`, [14](#)