Package ‘dynamac’

April 3, 2020

Title  Dynamic Simulation and Testing for Single-Equation ARDL Models

Version  0.1.11

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]

Repository  CRAN

Date/Publication  2020-04-03 15:40:03 UTC
R topics documented:

dshift ................................................................. 2
dynardl .............................................................. 3
dynardl.all.plots ................................................ 5
dynardl.auto.correlated ........................................ 7
dynardl.simulation.plot ......................................... 8
france.data ......................................................... 10
ineq ................................................................. 10
ldshift .............................................................. 11
lshift ............................................................... 12
pssbounds ......................................................... 13
summary.dynardl ................................................ 14
supreme.sup ....................................................... 15

Index 16

dshift Take first difference of a series

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**

```r
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 models and simulate interesting values (if desired)

**Usage**

```r
dynardl(formula, data = list(), lags = list(), diffs = c(),
        lagdiffs = list(), levels = c(), ec = FALSE, trend = FALSE,
        constant = TRUE, modelout = FALSE, noLDV = 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
- `noLDV`: do not add a lagged dependent variable (LDV) to ARDL models when omitted in formula (special thanks to Hannes Datta). This is not recommended
- `simulate`: simulate the response. 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
The `dynardl` package allows you to estimate an auto-regressive distributed lag model. Moreover, it enables a graphical interpretation of the results through the `dynardl.simulation.plot` function by simulating the response of the dependent variable to shocks in one of the regressors, and the Pesaran, Shin, and Smith (2001) test for cointegration for error-correction models through the `pssbounds` function.

### 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

Combine all of the potential plots of a simulated response in a dynardl model

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", ylim = NULL, xlab = NULL, ylab = NULL, ...)
Arguments

- `x`: a `dynardl` model with a simulation to be plotted. Since all plots include 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 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`).
- `ylim`: a user-defined y-limit to be used instead of the default (for instance, for shared axes. Use caution, as it will be passed to all plots).
- `xlab`: a user-defined x-label to be used instead of the default (use caution, as it will be passed to all plots).
- `ylab`: a user-defined y-label to be used instead of the default (use caution, as it will be passed to all plots).
- `...`: 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

- **x**: a `dynardl` model
- **bg.type**: a character string for the type of Breusch-Godfrey test to run. The default is Chisq: the Chisq test statistic. The other option is F: the F-test statistic
- **digits**: the number of digits to round to when showing output. The default is 3
- **order**: the maximum order of serial autocorrelation to test when executing the Breusch-Godfrey test
- **object.out**: if TRUE, and `dynardl.auto.correlated` 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

```r
# Using the ineq data from dynamac
drl.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(drl.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

```r
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", ylim = NULL, ylab = NULL,
  xlab = NULL, ...)
```

### Arguments

- **x**
  - a *dynardl* model with a simulation to be plotted

- **type**
  - whether the plot should be an area plot (*area*) or a spike plot (*spike*)

- **response**
  - whether the plot of the response should be shown in levels of the dependent variable (*levels*), levels from the mean of the dependent variable (*levels.from.mean*), period-over-period changes in the dependent variable (*diffs*), the absolute value of the (decreasing) change in the dependent variable in each time period due to the shock (*shock.effect.decay*), the sum of the period-over-period changes (*cumulative.diffs*), or the absolute value of the cumulative differences (where negative effects are treated as positive) (*cumulative.abs.diffs*). The default is *levels*

- **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.

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.

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).

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). The default is none.

ylim a user-defined y-limit to be used instead of the default (for instance, for shared axes).

ylab a user-defined y-label to be used instead of the default.

xlab a user-defined x-label to be used instead of the default.

... other arguments to be passed to the call to plot.

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


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, l)`

**Arguments**

- **x**: a series to be differenced
- **l**: 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

```r
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)
```

---

**lshift**

Take lag transformation of a series

**Description**

Take lag transformation of a series

**Usage**

```r
lshift(x, l)
```

**Arguments**

- `x`: a series to be lagged
- `l`: 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**

```r
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)
```
Perform Pesaran, Shin, and Smith (2001) cointegration test

Usage

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

Arguments

data an optional dynardl model. This option is highly recommended. Users are welcome to supply their own case, k regressors, t-statistic, F-statistic, and observations, but it is easier to have the model determine these quantities. If a dynardl model is supplied, user-supplied arguments are ignored

obs number of observations

fstat F-statistic of the joint test that variables in first lags are equal to zero: the specific restriction tested is \(1.y + 1.1.x1 + 1.1.x2 + \ldots + 1.1.xk = 0\), except in cases II and IV (see restriction and case)

tstat t-statistic of the lagged dependent variable

case The case of the test, as per Pesaran, Shin, and Smith (2001). 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

restriction if you design to test case II or IV of pssbounds, where it is assumed that the constant (case 2) or trend (case 4) are restricted in the resulting F-test, indicate that restriction = TRUE. If restriction = TRUE and there is no trend in the regression (trend = FALSE in dynardl), the F-test will include the constant in addition to the lagged dependent variable and lagged regressors in order to test for cointegration under the assumption of a restricted constant (see Pesaran, Shin and Smith [2001], case II). If restriction = TRUE and there is a trend in the regression (trend = TRUE in dynardl), the F-test will include the trend term in addition to the lagged dependent variable and lagged regressors in order to test for cointegration under the assumption of a restricted trend (see Pesaran, Shin and Smith [2001], case IV). If you are estimating the regular unrestricted ECM (this is more common), restriction = FALSE. The default is FALSE

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
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

```r
# 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.
```r
pssbounds(ardl.model)
```

summary.dynardl

Enable summary calls to `dynardl` model objects

Description

Enable summary calls to `dynardl` model objects

Usage

```r
## 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 dynmac
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)

data(supreme.sup)

Description


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, 13
*Topic datasets
  france.data, 10
  ineq, 10
  supreme.sup, 15
*Topic estimation
  dynardl, 3
*Topic simulation
  dynardl, 3
*Topic utilities
  dshift, 2
  dynardl.all.plots, 5
  dynardl.auto.correlated, 7
  dynardl.simulation.plot, 8
  ldshift, 11
  lshift, 12
  summary.dynardl, 14

dshift, 2
dynardl, 3, 5, 7, 8, 13, 14
dynardl.all.plots, 5
dynardl.auto.correlated, 7
dynardl.simulation.plot, 4, 6, 8

france.data, 10
ineq, 10

ldshift, 11
lshift, 12

pssbounds, 4, 13
summary.dynardl, 14
supreme.sup, 15