

Package ‘tseffects’

October 2, 2026

Title Dynamic Effects from Single-Equation Time Series Models (with Interactions)

Version 0.4.1

Maintainer Soren Jordan <sorenjordanpols@gmail.com>

Description Autoregressive distributed lag (A[R]DL) models (and their reparameterized equivalent, the Generalized Error-Correction Model [GECM]) are the workhorse dynamic linear models in uncovering dynamic inferences. ADL models are simple to estimate; this is what makes them attractive. Once these models are estimated, what is less clear is how to uncover a rich set of dynamic inferences from these models. We provide tools for recovering those inferences. These tools apply to traditional time-series quantities of interest and are built from the Impulse Response Function and Step Response Function (sometimes described as a pulse effect or a cumulative effect). They also allow for a variety of shock histories to be applied to the independent variable (beyond just a one-time, one-unit increase) as well as the recovery of inferences in levels for shocks applied to (in)dependent variables in differences through the Generalized Dynamic Response Function. These tools are also available for the general conditional dynamic model advocated by Warner, Vande Kamp, and Jordan (2026 <[doi:10.1017/psrm.2026.10087](https://doi.org/10.1017/psrm.2026.10087)>).

URL <https://sorenjordan.github.io/tseffects/>,
<https://github.com/sorenjordan/tseffects>

BugReports <https://github.com/sorenjordan/tseffects/issues>

Imports mpoly, car, ggplot2, sandwich, stats, utils

Suggests knitr, rmarkdown, vdiff, ARDL, dynamac, kardl, tidyverse,
zoo, ggplotify, patchwork, testthat (>= 3.0.0)

Depends R (>= 3.5.0)

License GPL (>= 2)

Encoding UTF-8

LazyData true

BuildManual yes

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no
Author Soren Jordan [aut, cre, cph] (ORCID:
 <<https://orcid.org/0000-0003-4201-1085>>),
 Garrett N. Vande Kamp [aut],
 Reshi Rajan [aut]
Repository CRAN
Date/Publication 2026-10-02 19:30:02 UTC

Contents

adl.plot	2
approval	4
convolution.calculator	5
GDRF.adl.plot	6
GDRF.gecm.plot	10
gecm.plot	14
gecm.to.adl	16
general.calculator	17
interact.adl.plot	18
pulse.calculator	23
toy.ts.interaction.data	24
yhat.calculator	26
Index	30

adl.plot	<i>Evaluate and/or plot the General Dynamic Response Function (GDRF) for an ADL assumed in levels</i>
----------	---

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for an autoregressive distributed lag (ADL) model, making use of the GDRF framework (described below and in Vande Kamp, Jordan, and Rajan), assuming the underlying model is in levels ($d.x = d.y = 0$) and the user wants a discrete effect (the untransformed GDRF) with a shock size of 1. (This is just a wrapper for `GDRF.adl.plot` with simplifying assumptions)

Usage

```
adl.plot(  
  model = NULL,  
  x.vrbl = NULL,  
  y.vrbl = NULL,  
  shock.history = "pulse",  
  dM.level = 0.95,  
  s.limit = 20,
```

```

    se.type = "const",
    return.data = FALSE,
    return.plot = TRUE,
    return.formulae = FALSE,
    ...
)

```

Arguments

<code>model</code>	the <code>lm</code> model containing the ADL estimates
<code>x.vrbl</code>	a named numeric vector in which the names correspond to an independent variable and its lags and the numbers correspond to the specific lag order of each variable
<code>y.vrbl</code>	a named numeric vector in which the names correspond to lags of the dependent variable and the numbers correspond to the specific lag order of each variable. Can be <code>NULL</code> if the model has no lagged dependent variables
<code>shock.history</code>	either a character string, integer, or numeric vector for the desired shock history. <code>shock.history</code> determines the shock history (<code>h</code>) that will be applied to the independent variable. Vande Kamp, Jordan, and Rajan express this as the parameter <code>h</code> (an integer). -1 represents the Impulse Response Function; the equivalent string is <code>pulse</code> or <code>impulse</code> . 0 represents the Step Response Function; the equivalent string is <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. However, since any shock history can be decomposed into a series of dated Impulse Response Functions, users can also supply any numeric vector of the same length as <code>s.limit + 1</code> . The default is <code>pulse</code>
<code>dm.level</code>	a numeric value for the significance level of the GDRF, calculated by the delta method. The default is 0.95
<code>s.limit</code>	an integer for the number of periods to determine the GDRF (beginning at <code>s = 0</code>)
<code>se.type</code>	a character string for the type of standard error to extract from the model. The default is <code>const</code> , but any argument to <code>vcovHC</code> from the <code>sandwich</code> package is accepted
<code>return.data</code>	logical to return the raw calculated GDRFs as a list element under <code>estimates</code> . The default is <code>FALSE</code>
<code>return.plot</code>	logical to return the visualized GDRFs as a list element under <code>plot</code> . The default is <code>TRUE</code>
<code>return.formulae</code>	logical to return the formulae for the GDRFs as a list element under <code>formulae</code> (for the GDRFs) and <code>shocks</code> (for the shock history). The default is <code>FALSE</code>
<code>...</code>	other arguments to be passed to the call to <code>plot</code>

Value

if just one of `return.data`, `return.plot`, and `return.formulae` is `TRUE`, `adl.plot` will return just the requested output. If more than one is selected (or just `return.data` is selected), it will return a list with elements `plot` (for the plot), `estimates` (for the data), and `formulae` and `shocks` for the GDRF formula for each period as well as the applied shock history in integer form

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# ADL(1,1)
# Use the toy data to run an ADL. No argument is made this is well specified; it is just expository
model.toydata <- lm(y ~ l_1_y + x + l_1_x, data = toy.ts.interaction.data)

# Since this is in levels, we can quickly look at the adl.plot
# Pulse effect of x
adl.plot(model = model.toydata,
  x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_y" = 1),
  shock.history = "pulse",
  s.limit = 20)
```

approval

Data on US Presidential Approval

Description

A dataset from: Cavari, Amnon. 2019. "Evaluating the President on Your Priorities: Issue Priorities, Policy Performance, and Presidential Approval, 1981–2016." *Presidential Studies Quarterly* 49(4): 798-826.

Usage

```
data(approval)
```

Format

A data frame with 140 rows and 14 variables:

APPROVE Presidential approval

APPROVE_ECONOMY Presidential approval: economy

APPROVE_FOREIGN Presidential approval: foreign affairs

MIP_MACROECONOMICS Salience (Most Important Problem): economy

MIP_FOREIGN Salience (Most Important Problem): foreign affairs

PARTY_IN Macropartisanship (in-party)

PARTY_OUT Macropartisanship (out-party)

PRESIDENT Numeric indicator for president

DIVIDEDGOV Dummy variable for divided government

ELECTION Dummy variable for election years

HONEYMOON Dummy variable for honeymoon period

UMCSENT Consumer sentiment

UNRATE Unemployment rate

APPROVE_L1 Lagged presidential approval

Source

[doi:10.1111/psq.12594](https://doi.org/10.1111/psq.12594)

convolution.calculator

Generate the convolved arbitrary (im)pulses formulae for a dynamic linear model

Description

Generate the convolved arbitrary (im)pulse effect formulae for an autoregressive distributed lag (ADL) model, given pulse effects. Take the period-specific impulse response functions (presumably pulses from `pulse.calculator`) and construct an arbitrary set of effects, given an arbitrary shock history and order of differencing of the independent and/or dependent variable

Usage

```
convolution.calculator(d.x, d.y, shocks, pulses)
```

Arguments

<code>d.x</code>	an integer determining the order of differencing of the x variable before a shock is applied when parametrized as an ADL model. (Generally, this is the same x variable used in <code>pulse.calculator</code>)
<code>d.y</code>	an integer determining the order of differencing of the y variable when parametrized as an ADL model. (Generally, this is the same y variable used in <code>pulse.calculator</code>)
<code>shocks</code>	a numeric vector of dated impulses of the same length as the pulses, 0 to some limit <code>s</code> , determining the shock history applied to the independent variable in levels
<code>pulses</code>	a list comprising the formulae for Impulse Response Functions, typically generated using <code>pulse.calculator</code>

Details

`convolution.calculator` does no calculation. It generates a list of `mpoly` formulae that contain variable names that represent the effect in each period for an arbitrary shock history. This is possible because any arbitrary shock history can be decomposed into a sum of dated impulses, and any shock history in differences has an algebraic equivalence to one in levels. The expectation is that these will be evaluated using coefficients from an object containing an ADL model with corresponding variables. Note: avoid passing characters not allowed by `mpoly`

Value

a list with two elements: limit + 1 mpoly formulae containing the generalized effect formula (for calculating the GDRF) in each period (under formulae) as well as the actual shock history (under shocks)

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# ADL(1,1)
x.lags <- c("x" = 0, "l_1_x" = 1) # lags of x
y.lags <- c("l_1_y" = 1)
s <- 5
pulse.effects <- pulse.calculator(x.vrbl = x.lags, y.vrbl = y.lags, limit = s)
# Instead of a generic h-order shock history, suppose we want something bespoke
# like an interrupted trend. Note this is of length s+1
my.shocks <- c(1, 2, 3, 0, 0, 4)
custom.shock.formulae <- convolution.calculator(d.x = 0, d.y = 0,
  shocks = my.shocks, pulses = pulse.effects)
custom.shock.formulae
```

GDRF.adl.plot

Evaluate and/or plot the General Dynamic Response Function (GDRF) for an ADL

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for an autoregressive distributed lag (ADL) model, making use of the GDRF framework (described below and in Vande Kamp, Jordan, and Rajan)

Usage

```
GDRF.adl.plot(
  model = NULL,
  x.vrbl = NULL,
  y.vrbl = NULL,
  d.x = NULL,
  d.y = NULL,
  shock.history = "pulse",
  inferences.y = "levels",
  inferences.x = "levels",
  effect.type = "discrete",
  cumulative = FALSE,
  prediction.values = NULL,
  baseline.y = NULL,
```

```

baseline.y.se = 0,
shock.size = 1,
dM.level = 0.95,
s.limit = 20,
se.type = "const",
return.LRM = TRUE,
return.data = FALSE,
return.plot = TRUE,
return.formulae = FALSE,
...
)

```

Arguments

<code>model</code>	the <code>lm</code> model containing the ADL estimates
<code>x.vrbl</code>	a named numeric vector in which the names correspond to an independent variable and its lags and the numbers correspond to the specific lag order of each variable
<code>y.vrbl</code>	a named numeric vector in which the names correspond to lags of the dependent variable and the numbers correspond to the specific lag order of each variable. Can be <code>NULL</code> if the model has no lagged dependent variables
<code>d.x</code>	an integer describing how many times the independent variable was differenced before model estimation
<code>d.y</code>	an integer describing how many times the dependent variable was differenced before model estimation
<code>shock.history</code>	either a character string, integer, or numeric vector for the desired shock history. <code>shock.history</code> determines the shock history (<code>h</code>) that will be applied to the independent variable. Vande Kamp, Jordan, and Rajan express this as the parameter <code>h</code> (an integer). -1 represents the Impulse Response Function; the equivalent string is <code>pulse</code> or <code>impulse</code> . 0 represents the Step Response Function; the equivalent string is <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. However, since any shock history can be decomposed into a series of dated Impulse Response Functions, users can also supply any numeric vector of the same length as <code>s.limit + 1</code> . The default is <code>pulse</code>
<code>inferences.y</code>	a character string specifying whether the shock to the independent variable is interpreted in levels or differences (of the dependent variable). Valid arguments include <code>levels</code> or <code>differences</code> . (For <code>y</code> variables where <code>d.y</code> is 0, this is automatically <code>levels</code> .) The default is <code>levels</code>
<code>inferences.x</code>	a character string specifying whether the shock to the independent variable is applied in levels or differences. Valid arguments include <code>levels</code> or <code>differences</code> . (For independent variables where <code>d.x</code> is 0, this is automatically <code>levels</code> .) The default is <code>levels</code>
<code>effect.type</code>	a character string indicating whether to return discrete effects or fitted values. <code>discrete</code> returns the GDRF as a discrete effect estimate. <code>fitted</code> returns the GDRF as a fitted value, relative to a baseline value of the dependent variable. The default is <code>discrete</code>

cumulative	logical to return the cumulative effects or the period-specific effects of the given shock.history. For instance, to obtain the Cumulative Impulse Response Function, set shock.history to -1 and cumulative = TRUE. The default is FALSE
prediction.values	a named list of values for non-y variables in the model, used to calculate a steady-state baseline of the dependent variable when effect.type = "fitted" and d.y = 0 and baseline.y is not supplied. Model-based uncertainty will be calculated around the formula for this value. If any differenced variables are included in the model, they should be set to 0. If multiple lags of the same variable are included, they should be set to the same value. Ignored when d.y > 0
baseline.y	a numeric value for the user-supplied baseline value of y in levels (i.e. the baseline value for effects to accumulate from). Only used when effect.type = "fitted". For d.y = 0, this overrides the steady-state calculation from prediction.values if provided
baseline.y.se	a numeric value for the user-supplied standard error for the baseline value of y (to suggest uncertainty around predictions). Only used when effect.type = "fitted" and inferences.y = "levels". If supplied, this is added in quadrature to the standard errors of the GDRF estimates. The default is 0: in recognition that this is user-constructed uncertainty. Possible values would be the square root of the standard deviation of y (in levels)
shock.size	a numeric value for the size of the shock to x in the units of x. Defaults to 1 (a discrete effect similar to traditional marginal effects)
dm.level	a numeric value for the significance level of the GDRF, calculated by the delta method. The default is 0.95
s.limit	an integer for the number of periods to determine the GDRF (beginning at s = 0)
se.type	a character string for the type of standard error to extract from the model. The default is const, but any argument to vcovHC from the sandwich package is accepted
return.LRM	logical to return the LRM in the estimates and plots, should the LRM exist for that combination of shock.history and cumulative. The default is TRUE
return.data	logical to return the raw calculated GDRFs as a list element under estimates. The default is FALSE
return.plot	logical to return the visualized GDRFs as a list element under plot. The default is TRUE
return.formulae	logical to return the formulae for the GDRFs as a list element under formulae (for the GDRFs) and shocks (for the shock history). The default is FALSE
...	other arguments to be passed to the call to plot

Details

Internally, GDRF.adl.plot uses the separate subfunctions pulse.calculator, convolution.calculator/general.calculator and (possibly) yhat.calculator to generate the effect estimates representing the GDRF. These estimates represent the effect of the independent variable (in levels or differences, depending on d.x) on the dependent variable (in levels or differences, depending on d.y), given a shock applied

to the independent variable (in levels or differences, depending on `inferences.x`). The effect on the dependent variable can be calculated in its differenced form or levels form, depending on `inferences.y`. The shock history is determined by `shock.history`. For more information, see Vande Kamp, Jordan, and Rajan. We assume that the ADL model estimated is well specified, free of residual autocorrelation, balanced, and meets other standard time-series qualities

Value

if just one of `return.data`, `return.plot`, and `return.formulae` is TRUE, `GDRF.adl.plot` will return just the requested output. If more than one is selected (or just `return.data` is selected), it will return a list with elements `plot` (for the plot), `estimates` (for the data), and `formulae` and shocks for the GDRF formula for each period as well as the applied shock history in integer form

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# ADL(1,1)
# Use the toy data to run an ADL. No argument is made this is well specified; it is just expository
model.toydata <- lm(y ~ l_1_y + x + l_1_x, data = toy.ts.interaction.data)

# Pulse effect of x
GDRF.adl.plot(model = model.toydata,
  x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_y" = 1),
  d.x = 0,
  d.y = 0,
  shock.history = "pulse",
  inferences.y = "levels",
  inferences.x = "levels",
  s.limit = 20)

# Step effect of x. You can store the data to draw your own plot,
# if you prefer
test.step <- GDRF.adl.plot(model = model.toydata,
  x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_y" = 1),
  d.x = 0,
  d.y = 0,
  shock.history = "step",
  inferences.y = "levels",
  inferences.x = "levels",
  s.limit = 20)
test.step$plot

# Cumulative Impulse Response effect of x. For the dynamic linear model,
# this is equivalent to the step effect
test.cumulative <- GDRF.adl.plot(model = model.toydata,
  x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_y" = 1),
```

```

d.x = 0,
d.y = 0,
shock.history = "pulse",
cumulative = TRUE,
inferences.y = "levels",
inferences.x = "levels",
s.limit = 20)
test.cumulative$plot

# Fitted values: steady state baseline from prediction.values
GDRF.adl.plot(model = model.toydata,
x.vrbl = c("x" = 0, "l_1_x" = 1),
y.vrbl = c("l_1_y" = 1),
d.x = 0,
d.y = 0,
shock.history = "pulse",
inferences.y = "levels",
inferences.x = "levels",
effect.type = "fitted",
prediction.values = list("x" = 0, "l_1_x" = 0),
s.limit = 20)

```

GDRF.gecm.plot

Evaluate and/or plot the General Dynamic Response Function (GDRF) for a GECM

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for a Generalized Error Correction Model (GECM), making use of the GDRF framework (described below and in Vande Kamp, Jordan, and Rajan)

Usage

```

GDRF.gecm.plot(
  model = NULL,
  x.vrbl = NULL,
  y.vrbl = NULL,
  x.vrbl.d.x = NULL,
  y.vrbl.d.y = NULL,
  x.d.vrbl = NULL,
  y.d.vrbl = NULL,
  x.d.vrbl.d.x = NULL,
  y.d.vrbl.d.y = NULL,
  shock.history = "pulse",
  inferences.y = "levels",
  inferences.x = "levels",
  effect.type = "discrete",

```

```

    cumulative = FALSE,
    prediction.values = NULL,
    baseline.y = NULL,
    baseline.y.se = 0,
    shock.size = 1,
    dM.level = 0.95,
    s.limit = 20,
    se.type = "const",
    return.LRM = TRUE,
    return.data = FALSE,
    return.plot = TRUE,
    return.formulae = FALSE,
    ...
)

```

Arguments

<code>model</code>	the <code>lm</code> model containing the GECM estimates
<code>x.vrbl</code>	a named numeric vector in which the names correspond to an independent variable and its lags (of the lower order of differencing, usually in levels $d = 0$) and the numbers correspond to the specific lag order of each variable
<code>y.vrbl</code>	a named numeric vector in which the names correspond to lags of the dependent variable (of the lower order of differencing, usually in levels $d = 0$) and the numbers correspond to the specific lag order of each variable
<code>x.vrbl.d.x</code>	an integer describing how many times the independent variable (of the lower order of differencing, usually in levels $d = 0$) was differenced before model estimation
<code>y.vrbl.d.y</code>	an integer describing how many times the dependent variable (of the lower order of differencing, usually in levels $d = 0$) was differenced before model estimation
<code>x.d.vrbl</code>	a named numeric vector in which the names correspond to an independent variable and its lags (of the higher order of differencing, usually in first differences $d = 1$) and the numbers correspond to the specific lag order of each variable
<code>y.d.vrbl</code>	a named numeric vector in which the names correspond to lags of the dependent variable (of the higher order of differencing, usually in first differences $d = 1$) and the numbers correspond to the specific lag order of each variable
<code>x.d.vrbl.d.x</code>	an integer describing how many times the independent variable (of the higher order of differencing, usually in first differences $d = 1$) was differenced before model estimation
<code>y.d.vrbl.d.y</code>	an integer describing how many times the dependent variable (of the higher order of differencing, usually in first differences $d = 1$) was differenced before model estimation
<code>shock.history</code>	either a character string, integer, or numeric vector for the desired shock history. <code>shock.history</code> determines the shock history (h) that will be applied to the independent variable. Vande Kamp, Jordan, and Rajan express this as the parameter h (an integer). -1 represents the Impulse Response Function; the equivalent

	string is pulse or impulse. 0 represents the Step Response Function; the equivalent string is step. For others, see Vande Kamp, Jordan, and Rajan. However, since any shock history can be decomposed into a series of dated Impulse Response Functions, users can also supply any numeric vector of the same length as <code>s.limit + 1</code> . The default is pulse
<code>inferences.y</code>	a character string specifying whether the shock to the independent variable is interpreted in levels or differences (of the dependent variable). Valid arguments include levels or differences. (For y variables where <code>d.y</code> is 0, this is automatically levels.) The default is levels
<code>inferences.x</code>	a character string specifying whether the shock to the independent variable is applied in levels or differences. Valid arguments include levels or differences. (For independent variables where <code>d.x</code> is 0, this is automatically levels.) The default is levels
<code>effect.type</code>	a character string indicating whether to return discrete effects or fitted values. <code>discrete</code> returns the GDRF as a discrete effect estimate. <code>fitted</code> returns the GDRF as a fitted value, relative to a baseline value of the dependent variable. The default is <code>discrete</code>
<code>cumulative</code>	logical to return the cumulative effects or the period-specific effects of the given <code>shock.history</code> . For instance, to obtain the Cumulative Impulse Response Function, set <code>shock.history</code> to -1 and <code>cumulative = TRUE</code> . The default is <code>FALSE</code>
<code>prediction.values</code>	a named list of values for non-y variables in the model, used to calculate a steady-state baseline of the dependent variable when <code>effect.type = "fitted"</code> and <code>y.vrbl.d.y = 0</code> and <code>baseline.y</code> is not supplied. Model-based uncertainty will be calculated around the formula for this value. If any differenced variables are included in the model, they should be set to 0. If multiple lags of the same variable are included, they should be set to the same value. Ignored when <code>y.vrbl.d.y > 0</code>
<code>baseline.y</code>	a numeric value for the user-supplied baseline value of y in levels (i.e. the baseline value for effects to accumulate from). Only used when <code>effect.type = "fitted"</code> . For <code>y.vrbl.d.y = 0</code> , this overrides the steady-state calculation from <code>prediction.values</code> if provided
<code>baseline.y.se</code>	a numeric value for the user-supplied standard error for the baseline value of y (to suggest uncertainty around predictions). Only used when <code>effect.type = "fitted"</code> and <code>inferences.y = "levels"</code> . If supplied, this is added in quadrature to the standard errors of the GDRF estimates. The default is 0: in recognition that this is user-constructed uncertainty. Possible values would be the square root of the standard deviation of y (in levels)
<code>shock.size</code>	a numeric value for the size of the shock to x in the units of x. Defaults to 1 (a discrete effect similar to traditional marginal effects)
<code>dM.level</code>	a numeric value for the significance level of the GDRF, calculated by the delta method. The default is 0.95
<code>s.limit</code>	an integer for the number of periods to determine the GDRF (beginning at <code>s = 0</code>)
<code>se.type</code>	a character string for the type of standard error to extract from the model. The default is <code>const</code> , but any argument to <code>vcovHC</code> from the sandwich package is accepted

<code>return.LRM</code>	logical to return the LRM in the estimates and plots, should the LRM exist for that combination of <code>shock.history</code> and <code>cumulative</code> . The default is TRUE
<code>return.data</code>	logical to return the raw calculated GDRFs as a list element under <code>estimates</code> . The default is FALSE
<code>return.plot</code>	logical to return the visualized GDRFs as a list element under <code>plot</code> . The default is TRUE
<code>return.formulae</code>	logical to return the formulae for the GDRFs as a list element under <code>formulae</code> (for the GDRFs) and <code>shocks</code> (for the shock history). The default is FALSE
<code>...</code>	other arguments to be passed to the call to <code>plot</code>

Details

Internally, `GDRF.gecm.plot` uses the subfunction `gecm.to.adl` to convert the GECM coefficients to ADL coefficients before calculating the GDRF. From there, it uses the separate subfunctions `pulse.calculator`, `convolution.calculator/general.calculator`, and (possibly) `yhat.calculator` to generate the effect estimates representing the GDRF. The shock history is determined by `shock.history`. For more information, see Vande Kamp, Jordan, and Rajan. We assume that the GECM estimated is well specified, free of residual autocorrelation, balanced, and meets other standard time-series qualities

Value

if just one of `return.data`, `return.plot`, and `return.formulae` is TRUE, `GDRF.gecm.plot` will return just the requested output. If more than one is selected (or just `return.data` is selected), it will return a list with elements `plot` (for the plot), `estimates` (for the data), and `formulae` and `shocks` for the GDRF formula for each period as well as the applied shock history in integer form

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# GECM(1,1)
# Use the toy data to run a GECM. No argument is made this
# is well specified or even sensible; it is just expository
model <- lm(d_y ~ l_1_y + l_1_x + l_1_d_y + d_x + l_1_d_x, data = toy.ts.interaction.data)
test.pulse <- GDRF.gecm.plot(model = model,
                             x.vrbl = c("l_1_x" = 1),
                             y.vrbl = c("l_1_y" = 1),
                             x.vrbl.d.x = 0,
                             y.vrbl.d.y = 0,
                             x.d.vrbl = c("d_x" = 0, "l_1_d_x" = 1),
                             y.d.vrbl = c("l_1_d_y" = 1),
                             x.d.vrbl.d.x = 1,
                             y.d.vrbl.d.y = 1,
                             shock.history = "pulse",
                             inferences.y = "levels",
                             inferences.x = "levels",
```

```

names(test.pulse)

s.limit = 10,
return.plot = TRUE,
return.formulae = TRUE)

```

gecm.plot

Evaluate and/or plot the General Dynamic Response Function (GDRF) for a GECM assumed in levels/first differences

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for a GECM(1,1), making use of the GDRF framework (described below and in Vande Kamp, Jordan, and Rajan), assuming the underlying model is in first differences ($x.vrb1.d.x = y.vrb1.d.y = 0$ and $x.d.vrb1.d.x = y.d.vrb1.d.y = 1$), the user wants a discrete effect (the untransformed GDRF) and inferences about y in levels to a shock applied to x in levels, with a shock size of 1. (This is just a wrapper for `GDRF.gecm.plot` with simplifying assumptions)

Usage

```

gecm.plot(
  model = NULL,
  x.vrb1 = NULL,
  y.vrb1 = NULL,
  x.d.vrb1 = NULL,
  y.d.vrb1 = NULL,
  shock.history = "pulse",
  dM.level = 0.95,
  s.limit = 20,
  se.type = "const",
  return.data = FALSE,
  return.plot = TRUE,
  return.formulae = FALSE,
  ...
)

```

Arguments

<code>model</code>	the <code>lm</code> model containing the GECM estimates
<code>x.vrb1</code>	a named numeric vector in which the names correspond to an independent variable and its lags (of the lower order of differencing, in levels $d = 0$) and the numbers correspond to the specific lag order of each variable
<code>y.vrb1</code>	a named numeric vector in which the names correspond to lags of the dependent variable (of the lower order of differencing, in levels $d = 0$) and the numbers correspond to the specific lag order of each variable

<code>x.d.vrbl</code>	a named numeric vector in which the names correspond to an independent variable and its lags (of the higher order of differencing, in first differences $d = 1$) and the numbers correspond to the specific lag order of each variable
<code>y.d.vrbl</code>	a named numeric vector in which the names correspond to lags of the dependent variable (of the higher order of differencing, in first differences $d = 1$) and the numbers correspond to the specific lag order of each variable
<code>shock.history</code>	either a character string, integer, or numeric vector for the desired shock history. <code>shock.history</code> determines the shock history (h) that will be applied to the independent variable. Vande Kamp, Jordan, and Rajan express this as the parameter h (an integer). -1 represents the Impulse Response Function; the equivalent string is <code>pulse</code> or <code>impulse</code> . 0 represents the Step Response Function; the equivalent string is <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. However, since any shock history can be decomposed into a series of dated Impulse Response Functions, users can also supply any numeric vector of the same length as <code>s.limit + 1</code> . The default is <code>pulse</code>
<code>dM.level</code>	a numeric value for the significance level of the GDRF, calculated by the delta method. The default is 0.95
<code>s.limit</code>	an integer for the number of periods to determine the GDRF (beginning at $s = 0$)
<code>se.type</code>	a character string for the type of standard error to extract from the model. The default is <code>const</code> , but any argument to <code>vcovHC</code> from the <code>sandwich</code> package is accepted
<code>return.data</code>	logical to return the raw calculated GDRFs as a list element under <code>estimates</code> . The default is <code>FALSE</code>
<code>return.plot</code>	logical to return the visualized GDRFs as a list element under <code>plot</code> . The default is <code>TRUE</code>
<code>return.formulae</code>	logical to return the formulae for the GDRFs as a list element under <code>formulae</code> (for the GDRFs) and shocks (for the shock history). The default is <code>FALSE</code>
<code>...</code>	other arguments to be passed to the call to <code>plot</code>

Details

We assume that the GECM model estimated is well specified, free of residual autocorrelation, balanced, and meets other standard time-series qualities. Given that, to obtain inferences for the specified shock history, the user only needs a named vector of the x and y variables, as well as the order of the differencing. Internally, the GECM to ADL equivalences are used to calculate the GDRFs from the GECM

Value

if just one of `return.data`, `return.plot`, and `return.formulae` is `TRUE`, `gecm.plot` will return just the requested output. If more than one is selected (or just `return.data` is selected), it will return a list with elements `plot` (for the plot), `estimates` (for the data), and `formulae` and shocks for the GDRF formula for each period as well as the applied shock history in integer form

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# GECM(1,1). So we can use gecm.plot to quickly check dynamics
# Use the toy data to run a GECM. No argument is made this
# is well specified or even sensible; it is just expository
model <- lm(d_y ~ l_1_y + l_1_x + l_1_d_y + d_x + l_1_d_x, data = toy.ts.interaction.data)
test.pulse <- gecm.plot(model = model,
                        x.vrbl = c("l_1_x" = 1),
                        y.vrbl = c("l_1_y" = 1),
                        x.d.vrbl = c("d_x" = 0, "l_1_d_x" = 1),
                        y.d.vrbl = c("l_1_d_y" = 1),
                        shock.history = "pulse",
                        s.limit = 10,
                        return.plot = TRUE,
                        return.formulae = TRUE)

names(test.pulse)
```

gecm.to.adl

Use equivalencies to translate GECM coefficients to ADL coefficients

Description

Translate the coefficients from the General Error Correction Model (GECM) to the autoregressive distributed lag (ADL) model. The ADL and the GECM are isomorphic, so this translation loses no information from the model

Usage

```
gecm.to.adl(x.vrbl, y.vrbl, x.d.vrbl, y.d.vrbl)
```

Arguments

x.vrbl	a named numeric vector in which the names correspond to an independent variable (of the lower level of differencing, usually in levels $d = 0$) and its lags and the numbers correspond to the specific lag order of each variable in the GECM model
y.vrbl	a named numeric vector in which the names correspond to lags of the dependent variable (of the lower level of differencing, usually in levels $d = 0$) and the numbers correspond to the specific lag order of each variable in the GECM model
x.d.vrbl	a named numeric vector in which the names correspond to an independent variable (of the higher level of differencing, usually in first differences $d = 1$) and its lags and the numbers correspond to the specific lag order of each variable in the GECM model
y.d.vrbl	a named numeric vector in which the names correspond to lags of the dependent variable (of the higher level of differencing, usually in first differences $d = 1$) and the numbers correspond to the specific lag order of each variable in the GECM model

Details

gecm.to.adl utilizes the mathematical equivalence between the GECM and ADL models to translate the coefficients from one to the other. This way, we can apply a single function using the ADL math to calculate effects

Value

a list of named vectors of translated ADL coefficients for the x and y variables of interest

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# GECM(1,1)
the.x.vrbl <- c("l_1_x" = 1)
the.y.vrbl <- c("l_1_y" = 1)
the.x.d.vrbl <- c("d_x" = 0, "l_1_d_x" = 1)
the.y.d.vrbl <- c("l_1_d_y" = 1)
adl.coef <- gecm.to.adl(x.vrbl = the.x.vrbl, y.vrbl = the.y.vrbl,
  x.d.vrbl = the.x.d.vrbl, y.d.vrbl = the.y.d.vrbl)
adl.coef$x.vrbl.adl
adl.coef$y.vrbl.adl
```

general.calculator	<i>Generate the generalized (im)pulse formulae for a dynamic linear model</i>
--------------------	---

Description

Generate the generalized (im)pulse effect formulae for an autoregressive distributed lag (ADL) model, given pulse effects and shock history. Take the period-specific impulse response functions (presumably pulses from pulse.calculator or convolution.calculator) and generalize them according to a shock history h and order of differencing of the independent and/or dependent variable

Usage

```
general.calculator(d.x, d.y, h, limit, pulses)
```

Arguments

d.x	an integer determining the order of differencing of the x variable before a shock is applied when parametrized as an ADL model. (Generally, this is the same x variable used in pulse.calculator)
d.y	an integer determining the order of differencing of the y variable when parametrized as an ADL model. (Generally, this is the same y variable used in pulse.calculator)

<code>h</code>	an integer determining the shock history applied to the independent variable in levels. -1 represents the Impulse Response Function. 0 represents a Step Response Function. For others, see Vande Kamp, Jordan, and Rajan
<code>limit</code>	an integer for the number of periods (s) to determine the generalized effect (beginning at 0)
<code>pulses</code>	a list comprising the formulae for Impulse Response Functions, typically generated using <code>pulse.calculator</code>

Details

`general.calculator` does no calculation. It generates a list of `mpoly` formulae that contain variable names that represent the generalized effect in each period. The expectation is that these will be evaluated using coefficients from an object containing an ADL model with corresponding variables. Note: avoid passing characters not allowed by `mpoly`

Value

a list with two elements: `limit + 1` `mpoly` formulae containing the generalized effect formula (for calculating the GDRF) in each period (under formulae) as well as the actual shock history (under shocks)

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# ADL(1,1)
x.lags <- c("x" = 0, "l_1_x" = 1) # lags of x
y.lags <- c("l_1_y" = 1)
s <- 5
pulse.effects <- pulse.calculator(x.vrbl = x.lags, y.vrbl = y.lags, limit = s)
# Assume that both x and y are in levels and we want a pulse shock history
general.pulse.effects <- general.calculator(d.x = 0, d.y = 0,
  h = -1, limit = s, pulses = pulse.effects)
general.pulse.effects
# Apply a step shock response function
general.step.effects <- general.calculator(d.x = 0, d.y = 0,
  h = 0, limit = s, pulses = pulse.effects)
general.step.effects
```

interact.adl.plot

Evaluate and/or plot dynamic interactions from dynamic linear models

Description

Evaluate (and possibly plot) the dynamic conditional effect from an autoregressive distributed lag (ADL) model (following the framework of Warner, Vande Kamp and Jordan). The description here matches the prose of Warner, Vande Kamp, and Jordan

Usage

```

interact.adl.plot(
  model = NULL,
  x.vrbl = NULL,
  z.vrbl = NULL,
  x.z.vrbl = NULL,
  y.vrbl = NULL,
  shock.history = "impulse",
  cumulative = FALSE,
  effect.type = "discrete",
  prediction.values = NULL,
  baseline.y = NULL,
  baseline.y.se = 0,
  shock.size = 1,
  plot.type = "lines",
  line.options = "z.lines",
  heatmap.options = "significant",
  line.colors = "okabe-ito",
  heatmap.colors = "Blue-Red",
  z.vals = NULL,
  s.vals = c(0, "LRM"),
  z.label.rounding = 3,
  z.vrbl.label = names(z.vrbl)[1],
  dM.level = 0.95,
  s.limit = 20,
  se.type = "const",
  return.LRM = TRUE,
  return.data = FALSE,
  return.plot = TRUE,
  return.formulae = FALSE,
  ...
)

```

Arguments

<code>model</code>	the <code>lm</code> model containing the ADL estimates
<code>x.vrbl</code>	a named numeric vector in which the names correspond to the “main” x variables and its lags and the numbers correspond to the specific lag order of each variable
<code>z.vrbl</code>	a named numeric vector in which the names correspond to the “moderating” z variables and its lags and the numbers correspond to the specific lag order of each variable
<code>x.z.vrbl</code>	a named numeric vector in which the names correspond to the “interaction” x:z variables and its lags and the numbers correspond to the specific lag order of each variable. IMPORTANT: enter the lag order that pertains to the “main” x variable. For instance, <code>x_1_1_z</code> (contemporaneous x times lagged z) would be 0 and <code>l_1_x_z</code> (lagged x times contemporaneous z) would be 1

<code>y.vrbl</code>	a named numeric vector in which the names correspond to lags of the dependent variable and the numbers correspond to the specific lag order of each variable. Can be NULL if the model has no lagged dependent variables
<code>shock.history</code>	either a character string, integer, or numeric vector for the desired shock history. <code>shock.history</code> determines the shock history (h) that will be applied to the independent variable. Vande Kamp, Jordan, and Rajan express this as the parameter h (an integer). -1 represents the Impulse Response Function; the equivalent string is <code>pulse</code> or <code>impulse</code> . 0 represents the Step Response Function; the equivalent string is <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. However, since any shock history can be decomposed into a series of dated Impulse Response Functions, users can also supply any numeric vector of the same length as <code>s.limit + 1</code> . The default is <code>pulse</code>
<code>cumulative</code>	logical to return the cumulative effects or the period-specific effects of the given <code>shock.history</code> . For instance, to obtain the Cumulative Impulse Response Function, set <code>shock.history</code> to -1 and <code>cumulative = TRUE</code> . The default is <code>FALSE</code>
<code>effect.type</code>	a character string indicating whether to return discrete effects or fitted values. <code>discrete</code> returns a discrete effect estimate. <code>fitted</code> returns a fitted value, relative to a baseline value of the dependent variable. The default is <code>discrete</code>
<code>prediction.values</code>	a named list of values for non-y non-z variables in the model, used to calculate a steady-state baseline of the dependent variable when <code>effect.type = "fitted"</code> and <code>baseline.y</code> is not supplied. Model-based uncertainty will be calculated around the formula for this value. If any differenced variables are included in the model, they should be set to 0. If multiple lags of the same variable are included, they should be set to the same value (including for interactions). When working with interactions, these should be provided for any "main" x terms as well as the intended value of x for any "interaction" x:z terms. It is most sensible to use the same value of x for interaction terms with the same lag order: i.e. x and x:z would both be contemporaneous x, so <code>prediction.values = list("x" = 0, "x:z" = 0)</code> would set both values of contemporaneous x to 0. <code>prediction.values = list("x" = 0, "x:z" = 1)</code> would set one contemporaneous x to 0 and the other to 1; this is inconsistent. See the vignette for more examples. The "interaction" x.z terms will then be multiplied by a value of z, passed either by <code>z.vals</code> or internally defined by the function. If <code>return.formulae = TRUE</code> , a generic placeholder <code>z_val</code> is used in the formula to represent where the relevant <code>z.val</code> has been inserted to estimated the value
<code>baseline.y</code>	a numeric value for the user-supplied baseline value of y in levels (i.e. the baseline value for effects to accumulate from). Only used when <code>effect.type = "fitted"</code>
<code>baseline.y.se</code>	a numeric value for the user-supplied standard error for the baseline value of y (to suggest uncertainty around predictions). Only used when <code>effect.type = "fitted"</code> . If supplied, this is added in quadrature to the standard errors of the predictions. The default is 0: in recognition that this is user-constructed uncertainty. Possible values would be the square root of the standard deviation of y
<code>shock.size</code>	a numeric value for the size of the shock to x in the units of x. Defaults to 1 (a discrete effect similar to traditional marginal effects)

plot.type	a character string for the type of plot to create. heatmap features the combination of estimated effects across a set of values of z as well as a range of values of s. This is computationally intensive, as many effects are estimated. lines features the estimated effects at discrete values of s/z. The default is lines
line.options	a character string for the type of effect to calculate, if plot.type = "lines". z.lines evaluates the moderator as discrete values of z across a range of s. s.lines evaluates the moderator as discrete values of s across a range of z. The default is z.lines
heatmap.options	a character string for the type of heatmap to draw, if plot.type = "heatmap". all indicates that all discrete effects should be shown. significant indicates that only statistically significant discrete effects should be shown (since significance indicators cannot appear on the heatmap). Note: this just sets the insignificant effects to the numeric value of 0. If the middle value of your scale gradient is white, these will effectively "disappear." If another gradient is used, they will take on the color assigned to 0 values. If estimating fitted values, the function automatically switches to all, since significant fitted values are a bit nonsensical. The default is significant
line.colors	a character string for the line colors. This defaults to the color-safe Okabe-Ito (okabe-ito) colors. There is also a grayscale option through bw. Users can also include whatever colors they like; the number of colors must match the number of lines drawn. This is passed to scale_color_discrete
heatmap.colors	a character string for the heatmap colors. This defaults to the diverging gradient Blue-Red. Alternate colors must be one of hcl.pals(). For grayscale plots, use Grays. This is passed to scale_fill_gradientn
z.vals	a vector of numeric values for the moderating variable. If plot.type = "lines", these are treated as discrete levels of z. If plot.type = "heatmap", these are treated as a lower and upper limit of a range of values of z. If none are provided, interact.adl.plot will pick based on the plot.type
s.vals	a vector of values for the time since the shock at which to evaluate the discrete effects. This is only used if line.options = s.lines, meaning s is treated as the moderator. The default is 0 (short-run) and the LRM
z.label.rounding	a numeric value for the digits to round to for the z labels in the legend (if those values are automatically calculated). The default is 3
z.vrbl.label	an optional string to give as the name of the moderating z variable, used in plotting (to avoid ugly variable names in plot labels and legends)
dM.level	a numeric value for the significance level of the (cumulative) discrete effects, calculated by the delta method. The default is 0.95
s.limit	an integer for the number of periods to determine the (cumulative) discrete effects (beginning at s = 0)
se.type	a character string for the type of standard error to extract from the model. The default is const, but any argument to vcovHC from the sandwich package is accepted
return.LRM	logical to return the LRM in the estimates and plots, should the LRM exist for that combination of shock.history and cumulative. The default is TRUE

<code>return.data</code>	logical to return the raw calculated (cumulative) discrete effects as a list element under <code>estimates</code> . The default is <code>FALSE</code>
<code>return.plot</code>	logical to return the visualized (cumulative) discrete effects as a list element under <code>plot</code> . The default is <code>TRUE</code>
<code>return.formulae</code>	logical to return the formulae for the (cumulative) discrete effects as a list element under <code>formulae</code> (for the (cumulative) discrete effects) and shocks (for the shock history). For interactions, the placeholder of <code>z_val</code> is used to denote where a specific value of <code>z</code> is substituted. The default is <code>FALSE</code>
<code>...</code>	other arguments to be passed to the call to <code>plot</code>

Details

We assume that the ADL model estimated is well specified, free of residual autocorrelation, balanced, and meets other standard time-series qualities. It is imperative that you double-check you have referenced all `x`, `y`, `z`, and interaction terms through `x.vrbl`, `y.vrbl`, `z.vrbl`, and `x.z.vrbl`. You must also have their orders correctly entered. `interact.adl.plot` has no way of determining, from the variable list, which correspond with which

Value

if just one of `return.data`, `return.plot`, and `return.formulae` is `TRUE`, `interact.adl.plot` will return just the requested output. If more than one is selected (or just `return.data` is selected), it will return a list with elements `plot` (for the plot), `estimates` (for the data), and `formulae` and shocks for the (cumulative) conditional effects for each period as well as the applied shock history in integer form

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# Using Cavari's (2019) approval model
# Cavari's original model: APPROVE ~ APPROVE_ECONOMY + APPROVE_FOREIGN + MIP_MACROECONOMICS +
#   MIP_FOREIGN + APPROVE_ECONOMY*MIP_MACROECONOMICS + APPROVE_FOREIGN*MIP_FOREIGN +
#   APPROVE_L1 + PARTY_IN + PARTY_OUT + UNRATE +
#   DIVIDEDGOV + ELECTION + HONEYMOON + as.factor(PRESIDENT)

approval$ECONAPP_ECONMIP <- approval$APPROVE_ECONOMY*approval$MIP_MACROECONOMICS
approval$FPAPP_ECONFP <- approval$APPROVE_FOREIGN*approval$MIP_FOREIGN

cavari.model <- lm(APPROVE ~ APPROVE_ECONOMY + APPROVE_FOREIGN + MIP_MACROECONOMICS +
  MIP_FOREIGN + ECONAPP_ECONMIP + FPAPP_ECONFP +
  APPROVE_L1 + PARTY_IN + PARTY_OUT + UNRATE +
  DIVIDEDGOV + ELECTION + HONEYMOON + as.factor(PRESIDENT), data = approval)

# Now: discrete effect of X at different levels of Z
interact.adl.plot(model = cavari.model,
  x.vrbl = c("APPROVE_ECONOMY" = 0), y.vrbl = c("APPROVE_L1" = 1),
```

```

z.vrbl = c("MIP_MACROECONOMICS" = 0), x.z.vrbl = c("ECONAPP_ECONMIP" = 0),
shock.history = "impulse", plot.type = "lines", line.options = "z.lines")

# Use well-behaved simulated data (included) for even more examples,
# using the Warner, Vande Kamp, and Jordan general model
model.toydata <- lm(y ~ l_1_y + x + l_1_x + z + l_1_z +
  x_z + z_l_1_x +
  x_l_1_z + l_1_x_l_1_z, data = toy.ts.interaction.data)

# Discrete effect of z (not run: computational time)
# Be sure to specify x.z.vrbl orders with respect to x term
## Not run: interact.adl.plot(model = model.toydata, x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_y" = 1), z.vrbl = c("z" = 0, "l_1_z" = 1),
  x.z.vrbl = c("x_z" = 0, "z_l_1_x" = 1,
    "x_l_1_z" = 0, "l_1_x_l_1_z" = 1),
  z.vals = -2:2,
  shock.history = "impulse",
  plot.type = "lines",
  line.options = "z.lines",
  s.limit = 20)

## End(Not run)

# Heatmap of discrete effects, since X and Z are actually continuous
# (not run: computational time)
## Not run: interact.adl.plot(model = model.toydata, x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_y" = 1), z.vrbl = c("z" = 0, "l_1_z" = 1),
  x.z.vrbl = c("x_z" = 0, "z_l_1_x" = 1,
    "x_l_1_z" = 0, "l_1_x_l_1_z" = 1),
  z.vals = c(-2,2),
  shock.history = "impulse",
  plot.type = "heatmap",
  heatmap.options = "all",
  s.limit = 20)

## End(Not run)

```

pulse.calculator

Generate the (im)pulse formulae for a dynamic linear model

Description

Generate the (im)pulse effect formulae for a given autoregressive distributed lag (ADL) model. This is the Impulse Response Function (a one-unit, one-period increase) for the supplied lag structure, assumed to be from a dynamic linear model

Usage

```
pulse.calculator(x.vrbl, y.vrbl = NULL, limit)
```

Arguments

<code>x.vrbl</code>	a named numeric vector in which the names correspond to an independent variable and its lags and the numbers correspond to the specific lag order of each variable
<code>y.vrbl</code>	a named numeric vector in which the names correspond to lags of the dependent variable and the numbers correspond to the specific lag order of each variable. Can be NULL if the model has no lagged dependent variables
<code>limit</code>	an integer representing the number of periods after the initial shock (s) to calculate the Impulse Response Function

Details

`pulse.calculator` does no calculation. It generates a list of `mpoly` formulae that contain variable names that represent the pulse effect in each period. The expectation is that these will be evaluated using coefficients from an object containing an ADL model with corresponding variables. Note: avoid passing characters not allowed by `mpoly`

Value

a list of `limit + 1` `mpoly` formulae containing the pulse effect formula in each period

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# ADL(1,1)
x.lags <- c("x" = 0, "l_1_x" = 1) # lags of x
y.lags <- c("l_1_y" = 1)
s <- 5
pulses <- pulse.calculator(x.vrbl = x.lags, y.vrbl = y.lags, limit = s)
pulses
# Will also handle finite dynamics
x.lags <- c("x" = 0, "l_1_x" = 1) # lags of x
finite.pulses <- pulse.calculator(x.vrbl = x.lags, limit = s)
```

toy.ts.interaction.data

Simulated interactive time series data

Description

A simulated, well-behaved dataset of interactive time series data

Usage

```
data(toy.ts.interaction.data)
```

Format

A data frame with 50 rows and 32 variables:

time Indicator for time period

x Contemporaneous x

l_1_x First lag of x

l_2_x Second lag of x

l_3_x Third lag of x

l_4_x Fourth lag of x

l_5_x Fifth lag of x

d_x First difference of x

l_1_d_x First lag of first difference of x

l_2_d_x Second lag of first difference of x

l_3_d_x Third lag of first difference of x

z Contemporaneous z

l_1_z First lag of z

l_2_z Second lag of z

l_3_z Third lag of z

l_4_z Fourth lag of z

l_5_z Fifth lag of z

y Contemporaneous y

l_1_y First lag of y

l_2_y Second lag of y

l_3_y Third lag of y

l_4_y Fourth lag of y

l_5_y Fifth lag of y

d_y First difference of y

l_1_d_y First lag of first difference of y

l_2_d_y Second lag of first difference of y

d_2_y Second difference of y

l_1_d_2_y First lag of second difference of y

x_z Interaction of contemporaneous x and z

x_l_1_z Interaction of contemporaneous x and lagged z

z_l_1_x Interaction of lagged x and contemporaneous z

l_1_x_l_1_z Interaction of lagged x and lagged z

yhat.calculator

Transform GDRF formulae to fitted value formulae

Description

By default, the GDRF assumes the user is interested in effect estimates (the effect of an independent variable on a dependent variable, calculated in levels or differences, as appropriate). yhat.calculator “adds” these effects to a baseline value of the dependent variable to observe how its trajectory changes

Usage

```
yhat.calculator(
  formulae,
  d.y,
  model,
  the.coef,
  y.vrbl = NULL,
  inferences.y = NULL,
  prediction.values = NULL,
  baseline.y = NULL,
  shock.size = 1,
  z.val = NULL,
  z.vrbl = NULL,
  x.z.vrbl = NULL
)
```

Arguments

formulae	the list of formulae from general.calculator or convolution.calculator
d.y	an integer for the order of differencing of the y variable in the ADL model
model	the lm model containing the model estimates
the.coef	the coefficient vector from the estimated model
y.vrbl	a named vector of the (lagged) y variables and corresponding lag orders in the ADL model
inferences.y	a character string that determines whether the inferences for the dependent variable are in levels or differences. Must be one of levels or differences
prediction.values	a named list of values for non-y variables in the model, used to calculate a steady-state baseline when d.y = 0 and baseline.y is not supplied. If any differenced variables are included in the model, they should be set to 0. If multiple lags of the same variable are included, they should be set to the same value (including for interactions). Ignored when d.y > 0

baseline.y	a user-supplied numeric baseline value of y in levels. For $d.y = 0$, this overrides the steady-state calculation from <code>prediction.values</code> . For $d.y > 0$ with <code>inferences.y = "levels"</code> , this is required. Optional uncertainty around this baseline can be supplied through <code>baseline.y.se</code> in the calling function
shock.size	a numeric value for the size of the shock to x in the units of x. Defaults to 1 (a discrete effect similar to traditional marginal effects)
z.val	a numeric value for the current value of the moderating variable, if the baseline formula contains an interaction
z.vrbl	a named numeric vector in which the names correspond to the “moderating” z variables and its lags and the numbers correspond to the specific lag order of each variable, if the baseline formula contains an interaction
x.z.vrbl	a named numeric vector in which the names correspond to the “interaction” x:z variables and its lags and the numbers correspond to the specific lag order of each variable. IMPORTANT: enter the lag order that pertains to the “main” x variable. For instance, <code>x_l1_z</code> (contemporaneous x times lagged z) would be 0 and <code>l1_l_x_z</code> (lagged x times contemporaneous z) would be 1

Details

`yhat.calculator` does no calculation. It transforms the formulae from `general.calculator` into fitted value formulae by prepending a baseline value of y. For $d.y = 0$, the baseline is either a user-supplied value or a model-implied steady-state prediction, the latter of which incorporates model-based uncertainty. For $d.y > 0$, the baseline must be user-supplied through `baseline.y`, as the model in differences contains no information about the level of y. Optional uncertainty around a user-supplied baseline can be added through `baseline.y.se`, it is added as a post-processing step in the calling function. This ability to observe fitted values of y is most sensible when the underlying model is originally estimated in levels of the dependent variable (so that a pre-shock baseline can be calculated and model-based uncertainty can be added). When the dependent variable is differenced, the baseline is arbitrary (as the model is ignorant of any underlying history or level of the dependent variable). These formulae are evaluated by `deltaMethod` in the calling function

Value

a list of `limit + 1` formula strings containing the fitted value formula in each period

Author(s)

Soren Jordan, Garrett N. Vande Kamp, and Reshi Rajan

Examples

```
# ADL model with y in levels
model.levels <- lm(y ~ x + l1_x + l1_y, data = toy.ts.interaction.data)

# set up formulae
pulses.levels <- pulse.calculator(x.vrbl = c("x" = 0, "l1_x" = 1),
  y.vrbl = c("l1_y" = 1), limit = 5)
general.levels <- general.calculator(d.x = 0, d.y = 0, h = -1,
  limit = 5, pulses = pulses.levels)
```

```

# I(0) y: steady state from means (warns about differenced variables)
# Note this would mean different values for x and l_1_x, which might be undesirable
yhat.calculator(formulae = general.levels$formulae, d.y = 0,
  model = model.levels, the.coef = coef(model.levels),
  y.vrbl = c("l_1_y" = 1), inferences.y = "levels",
  prediction.values = NULL, baseline.y = 0, shock.size = 1)

# I(0) y: steady state from supplied prediction.values (same values for both x/l_1_x)
yhat.calculator(formulae = general.levels$formulae, d.y = 0,
  model = model.levels, the.coef = coef(model.levels),
  y.vrbl = c("l_1_y" = 1), inferences.y = "levels",
  prediction.values = list("x" = 1, "l_1_x" = 1),
  baseline.y = NULL, shock.size = 1)

# I(0) y: user-supplied baseline.y overrides prediction.values
yhat.calculator(formulae = general.levels$formulae, d.y = 0,
  model = model.levels, the.coef = coef(model.levels),
  y.vrbl = c("l_1_y" = 1), inferences.y = "levels",
  prediction.values = list("x" = 0, "l_1_x" = 1),
  baseline.y = 5, shock.size = 1)

# ADL model with differenced y
model.diffs <- lm(d_y ~ x + l_1_x + l_1_d_y, data = toy.ts.interaction.data)

# set up formulae
pulses.diffes <- pulse.calculator(x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_d_y" = 1), limit = 5)
general.diffes <- general.calculator(d.x = 0, d.y = 1, h = -1,
  limit = 5, pulses = pulses.diffes)

## Not run:
# inferences in differences, baseline.y != 0. warn that this makes no sense (implies the
# model is always changing) and change baseline.y to 0
yhat.calculator(formulae = general.diffes$formulae, d.y = 1,
  model = model.diffes, the.coef = coef(model.diffes),
  y.vrbl = c("l_1_y" = 1), inferences.y = "differences",
  baseline.y = 3, shock.size = 1)

# inferences in differences, shock size of 1: identical to discrete effect (warns)
yhat.calculator(formulae = general.diffes$formulae, d.y = 1,
  model = model.diffes, the.coef = coef(model.diffes),
  y.vrbl = c("l_1_y" = 1), inferences.y = "differences",
  baseline.y = NULL, shock.size = 1)

# inferences in differences, shock size of 2: scales the discrete effect
# Since we're asking for inferences.y in differences, the baseline will automatically be 0
yhat.calculator(formulae = general.diffes$formulae, d.y = 1,
  model = model.diffes, the.coef = coef(model.diffes),
  y.vrbl = c("l_1_y" = 1), inferences.y = "differences",
  baseline.y = NULL, shock.size = 2)

# inferences in levels with no baseline.y: stops with an error

```

```
yhat.calculator(formulae = general.diffs$formulae, d.y = 1,
  model = model.diffs, the.coef = coef(model.diffs),
  y.vrbl = c("l_1_y" = 1), inferences.y = "levels",
  baseline.y = NULL, shock.size = 2)

# inferences in levels with prediction.values but no baseline.y: warns and stops
yhat.calculator(formulae = general.diffs$formulae, d.y = 1,
  model = model.diffs, the.coef = coef(model.diffs),
  y.vrbl = c("l_1_y" = 1), inferences.y = "levels",
  prediction.values = list("x" = 1, "l_1_x" = 1),
  baseline.y = NULL, shock.size = 2)

## End(Not run)

# inferences in levels with a supplied baseline
yhat.calculator(formulae = general.diffs$formulae, d.y = 1,
  model = model.diffs, the.coef = coef(model.diffs),
  y.vrbl = c("l_1_y" = 1), inferences.y = "levels",
  prediction.values = list("x" = 1, "l_1_x" = 1),
  baseline.y = 5, shock.size = 2)
```

Index

- * **ADL**
 - adl.plot, [2](#)
 - GDRF.adl.plot, [6](#)
 - interact.adl.plot, [18](#)
- * **GDRF**
 - adl.plot, [2](#)
 - GDRF.adl.plot, [6](#)
 - GDRF.gecm.plot, [10](#)
 - gecm.plot, [14](#)
- * **GECM**
 - GDRF.gecm.plot, [10](#)
 - gecm.plot, [14](#)
- * **datasets**
 - approval, [4](#)
 - toy.ts.interaction.data, [24](#)
- * **interactions**
 - interact.adl.plot, [18](#)
- * **utilities**
 - convolution.calculator, [5](#)
 - gecm.to.adl, [16](#)
 - general.calculator, [17](#)
 - pulse.calculator, [23](#)
 - yhat.calculator, [26](#)

adl.plot, [2](#)
approval, [4](#)

convolution.calculator, [5](#)

GDRF.adl.plot, [6](#)
GDRF.gecm.plot, [10](#)
gecm.plot, [14](#)
gecm.to.adl, [16](#)
general.calculator, [17](#)

interact.adl.plot, [18](#)

pulse.calculator, [23](#)

toy.ts.interaction.data, [24](#)

yhat.calculator, [26](#)