

Package ‘tseffects’

July 15, 2026

Title Dynamic Inferences from Time Series (with Interactions)

Version 0.3.1

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 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 (what we call the Generalized Dynamic Response Function). These effects 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)>). We also provide the formulae for these effects.

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, vdiffr, 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]

Contents

adl.dummy.checks	2
adl.plot	3
approval	4
GDRF.adl.plot	5
GDRF.dummy.checks	8
GDRF.gecm.plot	9
gecm.dummy.checks	12
gecm.plot	13
gecm.to.adl	15
general.calculator	16
get.value	17
interact.adl.plot	18
mpoly.subber	21
pulse.calculator	21
toy.ts.interaction.data	22
what.to.return	24
yhat.calculator	24
Index	28

adl.dummy.checks	<i>Do consistent dummy checks for functions that use an ADL model</i>
------------------	---

Description

Do consistent dummy checks for functions that use an ADL model

Usage

```
adl.dummy.checks(
  x.vrbl,
  y.vrbl,
  d.x,
  d.y,
  inferences.x,
  inferences.y,
  the.coef,
  se.type,
  type = NULL
)
```

Arguments

x.vrbl	a named vector of the x variables and corresponding lag orders in an ADL model
y.vrbl	a named vector of the y variables and corresponding lag orders in an ADL model
d.x	the order of differencing of the x variable in the ADL model
d.y	the order of differencing of the y variable in the ADL model
inferences.x	is the independent variable treated in levels or in differences?
inferences.y	are the inferences for the dependent variable expected in levels or in differences?

the.coef	the coefficient vector from the estimated ADL model
se.type	the type of standard error calculated
type	whether the effects are estimated in the context of a GDRF

Author(s)

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

adl.plot	<i>Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for an autoregressive distributed lag (ADL) model, assuming the underlying model is in levels ($d.x = d.y = 0$) and the user wants a marginal effect (the untransformed GDRF). (This is just a wrapper for <code>GDRF.adl.plot</code> with simplifying assumptions)</i>
----------	---

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for an autoregressive distributed lag (ADL) model, assuming the underlying model is in levels ($d.x = d.y = 0$) and the user wants a marginal effect (the untransformed GDRF). (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

model	the lm model containing the ADL estimates
x.vrbl	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
y.vrbl	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>	the desired shock history. <code>shock.history</code> determines the shock history (<code>h</code>) (which can be expressed as an integer) that will be applied to the independent variable. -1 represents a pulse (Impulse Response Function). 0 represents a step (Step Response Function). These can also be specified via <code>pulse</code> and <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. The default is <code>pulse</code>
<code>dM.level</code>	a numeric 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 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>binomials</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>

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 Saliency (Most Important Problem): economy
MIP_FOREIGN Saliency (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)

GDRF.adl.plot

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for an autoregressive distributed lag (ADL) model

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for an autoregressive distributed lag (ADL) model

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 = "marginal",
  prediction.values = NULL,
  baseline.y = NULL,
  baseline.y.se = 0,
  shock.size = 1,
```

```

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>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>	the desired shock history. <code>shock.history</code> determines the shock history (<code>h</code>) (which can be expressed as an integer) that will be applied to the independent variable. <code>-1</code> represents a pulse (Impulse Response Function). <code>0</code> represents a step (Step Response Function). These can also be specified via <code>pulse</code> and <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. The default is <code>pulse</code>
<code>inferences.y</code>	does the user want resulting inferences about the dependent variable in levels or in differences? (For <code>y</code> variables where <code>d.y</code> is <code>0</code> , this is automatically levels.) The default is <code>levels</code>
<code>inferences.x</code>	does the user want to apply the shock history to the independent variable in levels or in differences? (For <code>x</code> variables where <code>d.x</code> is <code>0</code> , this is automatically levels.) The default is <code>levels</code>
<code>effect.type</code>	whether to return marginal effects or fitted values. <code>marginal</code> returns the GDRF as a marginal effect. <code>fitted</code> returns the GDRF as a fitted value, relative to a baseline value of <code>y</code> . The default is <code>marginal</code>
<code>prediction.values</code>	a named list of values for non- <code>y</code> variables in the model, used to calculate a steady-state baseline when <code>effect.type = "fitted"</code> and <code>d.y = 0</code> and <code>baseline.y</code> is not supplied. This allows for the calculation of model-based uncertainty. If any differenced variables are included in the model, they should be set to <code>0</code> . Ignored when <code>d.y > 0</code>
<code>baseline.y</code>	a user-supplied baseline value of <code>y</code> in levels. For <code>d.y = 0</code> , this overrides the steady-state calculation from <code>prediction.values</code> if provided. For <code>d.y > 0</code> with <code>inferences.y = "levels"</code> , this is required (otherwise it is just marginal effects). Only used when <code>effect.type = "fitted"</code>
<code>baseline.y.se</code>	a user-supplied standard error for the baseline value of <code>y</code> (to suggest uncertainty around predictions). If supplied, this is added in quadrature to the standard errors of the GDRF estimates. Only used when <code>effect.type = "fitted"</code> and

	inferences.y = "levels". 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	the size of the shock to x in the units of x. Only used when effect.type = "fitted"; marginal effects are not scaled. Defaults to 1 (a marginal effect)
dM.level	a numeric 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 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.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 binomials (for the shock history). The default is FALSE
...	other arguments to be passed to the call to plot

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.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 = "step",
  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.dummy.checks	<i>Do consistent dummy checks for GDRF functions that might take fitted values</i>
-------------------	--

Description

Do consistent dummy checks for GDRF functions that might take fitted values

Usage

```
GDRF.dummy.checks(
  effect.type,
  prediction.values,
  baseline.y,
  baseline.y.se,
  shock.size,
  d.y,
  inferences.y
)
```

Arguments

<code>effect.type</code>	whether to return marginal effects or fitted values. <code>marginal</code> returns the GDRF as a marginal effect. <code>fitted</code> returns the GDRF as a fitted value, relative to a baseline value of <code>y</code> . The default is <code>marginal</code>
<code>prediction.values</code>	a named list of values for non- <code>y</code> variables in the model, used to calculate a steady-state baseline when <code>effect.type = "fitted"</code> and <code>d.y = 0</code> and <code>baseline.y</code> is not supplied. This allows for the calculation of model-based uncertainty. If any differenced variables are included in the model, they should be set to 0. Ignored when <code>d.y > 0</code>
<code>baseline.y</code>	a user-supplied baseline value of <code>y</code> in levels. For <code>d.y = 0</code> , this overrides the steady-state calculation from <code>prediction.values</code> if provided. For <code>d.y > 0</code> with <code>inferences.y = "levels"</code> , this is required (otherwise it is just marginal effects). Only used when <code>effect.type = "fitted"</code>

baseline.y.se	a user-supplied standard error for the baseline value of y (to suggest uncertainty around predictions). If supplied, this is added in quadrature to the standard errors of the GDRF estimates. Only used when effect.type = "fitted" and inferences.y = "levels". 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	the size of the shock to x in the units of x. Only used when effect.type = "fitted"; marginal effects are not scaled. Defaults to 1 (a marginal effect)
d.y	the order of differencing of the y variable in the ADL model
inferences.y	does the user want resulting inferences about the dependent variable in levels or in differences? (For y variables where d.y is 0, this is automatically levels.) The default is levels

Author(s)

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

GDRF.gecm.plot	<i>Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for a Generalized Error Correction Model (GECM)</i>
----------------	--

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for a Generalized Error Correction Model (GECM)

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 = "marginal",
  prediction.values = NULL,
  baseline.y = NULL,
  baseline.y.se = 0,
  shock.size = 1,
  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.vrbl</code>	a named numeric vector of the x variables (of the lower level of differencing, usually in levels $d = 0$) and corresponding lag orders in the GECM model
<code>y.vrbl</code>	a named numeric vector of the (lagged) y variables (of the lower level of differencing, usually in levels $d = 0$) and corresponding lag orders in the GECM model
<code>x.vrbl.d.x</code>	the order of differencing of the x variable (of the lower level of differencing, usually in levels $d = 0$) in the GECM model
<code>y.vrbl.d.y</code>	the order of differencing of the y variable (of the lower level of differencing, usually in levels $d = 0$) in the GECM model
<code>x.d.vrbl</code>	a named numeric vector of the x variables (of the higher level of differencing, usually first differences $d = 1$) and corresponding lag orders in the GECM model
<code>y.d.vrbl</code>	a named numeric vector of the y variables (of the higher level of differencing, usually first differences $d = 1$) and corresponding lag orders in the GECM model. Can be NULL if the model has no lags of the differenced dependent variables
<code>x.d.vrbl.d.x</code>	the order of differencing of the x variable (of the higher level of differencing, usually first differences $d = 1$) in the GECM model
<code>y.d.vrbl.d.y</code>	the order of differencing of the y variable (of the higher level of differencing, usually first differences $d = 1$) in the GECM model
<code>shock.history</code>	the desired shock history. <code>shock.history</code> determines the shock history (h) that will be applied to the independent variable. -1 represents a pulse. 0 represents a step. These can also be specified via <code>pulse</code> and <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. The default is pulse
<code>inferences.y</code>	does the user want resulting inferences about the dependent variable in levels or in differences? The default is levels
<code>inferences.x</code>	does the user want to apply the shock history to the independent variable in levels or in differences? The default is levels
<code>effect.type</code>	whether to return marginal effects or fitted values. <code>marginal</code> returns the GDRF as a marginal effect. <code>fitted</code> returns the GDRF as a fitted value, relative to a baseline value of y. The default is <code>marginal</code>
<code>prediction.values</code>	a named list of values for non-y variables in the model, used to calculate a steady-state baseline when <code>effect.type = "fitted"</code> and <code>d.y = 0</code> and <code>baseline.y</code> is not supplied. This allows for the calculation of model-based uncertainty. If any differenced variables are included in the model, they should be set to 0. Ignored when <code>d.y > 0</code>
<code>baseline.y</code>	a user-supplied baseline value of y in levels. For <code>d.y = 0</code> , this overrides the steady-state calculation from <code>prediction.values</code> if provided. For <code>d.y > 0</code> with <code>inferences.y = "levels"</code> , this is required (otherwise it is just marginal effects). Only used when <code>effect.type = "fitted"</code>


```

names(test.pulse)

x.d.vrbl.d.x = 1,
y.d.vrbl.d.y = 1,
shock.history = "pulse",
inferences.y = "levels",
inferences.x = "levels",
s.limit = 10,
return.plot = TRUE,
return.formulae = TRUE)

```

gecm.dummy.checks

*Do consistent dummy checks for functions that use a GECM model***Description**

Do consistent dummy checks for functions that use a GECM model

Usage

```

gecm.dummy.checks(
  x.vrbl,
  y.vrbl,
  x.d.vrbl,
  y.d.vrbl,
  x.vrbl.d.x,
  y.vrbl.d.y,
  x.d.vrbl.d.x,
  y.d.vrbl.d.y,
  inferences.x,
  inferences.y,
  the.coef,
  se.type,
  type = NULL
)

```

Arguments

<code>x.vrbl</code>	a named vector of the x variables and corresponding lag orders of lower order of integration (typically levels, 0) in a GECM model
<code>y.vrbl</code>	a named vector of the y variables and corresponding lag orders of lower order of integration (typically levels, 0) in a GECM model
<code>x.d.vrbl</code>	a named vector of the x variables and corresponding lag orders of higher order of integration (typically first differences, 1) in a GECM model
<code>y.d.vrbl</code>	a named vector of the y variables and corresponding lag orders of higher order of integration (typically first differences, 1) in a GECM model
<code>x.vrbl.d.x</code>	the order of differencing of the x variable of lower order of integration (typically levels, 0) in a GECM model
<code>y.vrbl.d.y</code>	the order of differencing of the y variable of lower order of integration (typically levels, 0) in a GECM model

<code>x.d.vrbl.d.x</code>	the order of differencing of the x variable of higher order of integration (typically first differences, 1) in a GECM model
<code>y.d.vrbl.d.y</code>	the order of differencing of the y variable of higher order of integration (typically first differences, 1) in a GECM model
<code>inferences.x</code>	is the independent variable treated in levels or in differences?
<code>inferences.y</code>	are the inferences for the dependent variable expected in levels or in differences?
<code>the.coef</code>	the coefficient vector from the estimated GECM model
<code>se.type</code>	the type of standard error calculated
<code>type</code>	whether the effects are estimated in the context of a GDRF

Author(s)

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

<code>gecm.plot</code>	<i>Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for a GECM(1,1) model, assuming the underlying model is in first differences ($x.vrbl.d.x = y.vrbl.d.y = 0$ and $x.d.vrbl.d.x = y.d.vrbl.d.y = 1$) and the user wants a marginal effect (the untransformed GDRF) and inferences about y in levels to a treatment applied to x in levels. (This is just a wrapper for <code>GDRF.gecm.plot</code> with simplifying assumptions)</i>
------------------------	---

Description

Evaluate (and possibly plot) the General Dynamic Response Function (GDRF) for a GECM(1,1) model, assuming the underlying model is in first differences ($x.vrbl.d.x = y.vrbl.d.y = 0$ and $x.d.vrbl.d.x = y.d.vrbl.d.y = 1$) and the user wants a marginal effect (the untransformed GDRF) and inferences about y in levels to a treatment applied to x in levels. (This is just a wrapper for `GDRF.gecm.plot` with simplifying assumptions)

Usage

```
gecm.plot(
  model = NULL,
  x.vrbl = NULL,
  y.vrbl = NULL,
  x.d.vrbl = NULL,
  y.d.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 GECM estimates
<code>x.vrbl</code>	a named numeric vector of the <code>x</code> variables (of the lower level of differencing, usually in levels $d = 0$) and corresponding lag orders in the GECM model
<code>y.vrbl</code>	a named numeric vector of the (lagged) <code>y</code> variables (of the lower level of differencing, usually in levels $d = 0$) and corresponding lag orders in the GECM model
<code>x.d.vrbl</code>	a named numeric vector of the <code>x</code> variables (of the higher level of differencing, usually first differences $d = 1$) and corresponding lag orders in the GECM model
<code>y.d.vrbl</code>	a named numeric vector of the <code>y</code> variables (of the higher level of differencing, usually first differences $d = 1$) and corresponding lag orders in the GECM model. Can be <code>NULL</code> if the model has no lags of the differenced dependent variables
<code>shock.history</code>	the desired shock history. <code>shock.history</code> determines the shock history (<code>h</code>) that will be applied to the independent variable. <code>-1</code> represents a pulse. <code>0</code> represents a step. These can also be specified via <code>pulse</code> and <code>step</code> . For others, see Vande Kamp, Jordan, and Rajan. The default is <code>pulse</code>
<code>dM.level</code>	a numeric significance level of the GDRF, calculated by the delta method. The default is <code>0.95</code>
<code>s.limit</code>	an integer for the number of periods to determine the GDRF (beginning at $s = 0$)
<code>se.type</code>	a 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>binomials</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>

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

depending on `return.data`, `return.plot`, and `return.formulae`, a list of elements relating to the GDRF

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	<i>Translate the coefficients from the General Error Correction Model (GECM) to the autoregressive distributed lag (ADL) model</i>
-------------	--

Description

Translate the coefficients from the General Error Correction Model (GECM) to the autoregressive distributed lag (ADL) model

Usage

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

Arguments

<code>x.vrbl</code>	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
<code>y.vrbl</code>	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
<code>x.d.vrbl</code>	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
<code>y.d.vrbl</code>	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 effect formulae for an autoregressive distributed lag (ADL) model, given pulse effects and shock history</i>
--------------------	--

Description

Generate the generalized effect formulae for an autoregressive distributed lag (ADL) model, given pulse effects and shock history

Usage

```
general.calculator(d.x, d.y, h, limit, 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>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: mpoly does not allow variable names with a .; variables passed to general.calculator should not include this character

Value

a list of limit + 1 mpoly formulae containing the generalized 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
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
```

get.value	<i>Find starting values for predicted values plots from the data in the model frame, if not supplied</i>
-----------	--

Description

Find starting values for predicted values plots from the data in the model frame, if not supplied

Usage

```
get.value(var, prediction.values, model)
```

Arguments

var	the variable to establish a prediction value for
prediction.values	(possible) user-supplied list of values for variables
model	the model containing the dataframe for mean estimation if values are not user-supplied (and warn the user if we're taking the mean)

Author(s)

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

interact.adl.plot	<i>Plot the interaction in a single-equation time series model estimated via lm.</i>
-------------------	--

Description

Plot the interaction in a single-equation time series model estimated via lm.

Usage

```
interact.adl.plot(
  model = NULL,
  x.vrbl = NULL,
  z.vrbl = NULL,
  x.z.vrbl = NULL,
  y.vrbl = NULL,
  shock.history = "impulse",
  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.data = FALSE,
  return.plot = TRUE,
  return.formulae = FALSE,
  ...
)
```

Arguments

model	the lm model containing the ADL estimates
x.vrbl	named numeric vector of the “main” x variables and corresponding lag orders in the ADL model
z.vrbl	named numeric vector of the “moderating” z variables and corresponding lag orders in the ADL model
x.z.vrbl	named numeric vector with the interaction variables and corresponding lag orders in the ADL model. IMPORTANT: enter the lag order that pertains to the “main” x variable. For instance, x_l_1_z (contemporaneous x times lagged z) would be 0 and l_1_x_z (lagged x times contemporaneous z) would be 1

<code>y.vrbl</code>	named numeric vector of the (lagged) y variables and corresponding lag orders in the ADL model. Can be NULL if the model has no lagged dependent variables
<code>shock.history</code>	whether impulse/pulse or cumulative/step effects should be calculated. <code>impulse</code> (or <code>pulse</code>) generates impulse effects (the Impulse Response Function), or short-run/instantaneous effects specific to each period. <code>cumulative</code> (or <code>step</code> : the Step Response Function) generates the accumulated, or long-run/cumulative effects up to each period (including the long-run multiplier). The default is <code>impulse</code>
<code>plot.type</code>	a string for whether to feature marginal effects at discrete values of s/z as lines, or across a range of values through a heatmap. The default is <code>lines</code>
<code>line.options</code>	if drawing lines, a string for whether the moderator should be values of z (<code>z.lines</code>) or values of s (<code>s.lines</code>). The default is <code>z.lines</code>
<code>heatmap.options</code>	if drawing a heatmap, a string for whether all marginal effects should be shown or just statistically significant ones. (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.) The default is <code>significant</code>
<code>line.colors</code>	a string for what color lines would you like for line plots? This defaults to the color-safe Okabe-Ito (<code>okabe-ito</code>) colors. There is also a grayscale option through <code>bw</code> . Users can also include whatever colors they like. The number of colors must match the number of lines drawn. This is passed to <code>scale_color_discrete</code>
<code>heatmap.colors</code>	a string for what color scale would you like for the heatmap? The default is <code>Blue-Red</code> . Alternate colors must be one of <code>hcl.pals()</code> . For grayscale plots, use <code>Grays</code> . This is passed to <code>scale_fill_gradientn</code>
<code>z.vals</code>	numeric values for the moderating variable. If <code>plot.type = lines</code> , these are treated as discrete levels of z. If <code>plot.type = heatmap</code> , these are treated as a lower and upper level of a range of values of z. If none are provided, <code>interact.adl.plot</code> will pick
<code>s.vals</code>	numeric values for the time since the shock. This is only used if <code>line.options = s.lines</code> , meaning s is treated as the moderator. The default is 0 (short-run) and the LRM
<code>z.label.rounding</code>	number of digits to round to for the z labels in the legend (if those values are automatically calculated)
<code>z.vrbl.label</code>	the name of the moderating z variable, used in plotting
<code>dM.level</code>	significance level of the (cumulative) marginal effects, calculated by the delta method. The default is 0.95
<code>s.limit</code>	an integer for the number of periods to determine the (cumulative) marginal effects (beginning at <code>s = 0</code>)
<code>se.type</code>	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 (cumulative) marginal 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) marginal 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) marginal effects as a list element under <code>formulae</code> (for the (cumulative) marginal effects) and <code>binomials</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>

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

depending on `return.data`, `return.plot`, and `return.formulae`, a list of elements relating to the interaction

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: marginal 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)

# Marginal 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 marginal 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)
```

mpoly.subber

Replace characters that mpoly does not take with underscores

Description

Replace characters that mpoly does not take with underscores

Usage

```
mpoly.subber(env = environment())
```

Arguments

env the environment with names to substitute. Defaults to the parent environment

Author(s)

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

pulse.calculator

Generate pulse effect formulae for a given autoregressive distributed lag (ADL) model

Description

Generate pulse effect formulae for a given autoregressive distributed lag (ADL) 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: `mpoly` does not allow variable names with a `.`; variables passed to `pulse.calculator` should not include this character

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 23 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

what.to.return	<i>Consistently return the correct objects after a GDRF ADL/GECM</i>
----------------	--

Description

Consistently return the correct objects after a GDRF ADL/GECM

Usage

```
what.to.return(
  return.plot,
  return.formulae,
  return.data,
  plot.out,
  dat.out,
  the.final.formulae
)
```

Arguments

return.plot	a TRUE/FALSE on whether the plot should be returned from the function
return.formulae	a TRUE/FALSE on whether the data from the effect should be returned from the function
plot.out	the created plot from the GDRF ADL/GECM
dat.out	the created data from the GDRF ADL/GECM
the.final.formulae	the created formulae from the GDRF ADL/GECM

Author(s)

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

yhat.calculator	<i>Transform the GDRF formulae to fitted value formulae</i>
-----------------	---

Description

Transform the GDRF formulae to fitted value formulae

Usage

```
yhat.calculator(
  formulae,
  d.y,
  model,
  the.coef,
  y.vrbl = NULL,
  inferences.y = NULL,
```



```

    prediction.values = NULL,
    baseline.y = NULL,
    shock.size = 1
  )

```

Arguments

<code>formulae</code>	the list of formulae from <code>general.calculator</code>
<code>d.y</code>	an integer for the order of differencing of the y variable in the ADL model
<code>model</code>	the lm model containing the model estimates
<code>the.coef</code>	the coefficient vector from the estimated model
<code>y.vrbl</code>	a named vector of the (lagged) y variables and corresponding lag orders in the ADL model
<code>inferences.y</code>	whether the inferences for the dependent variable are in levels or differences. Must be one of levels or differences
<code>prediction.values</code>	a named list of values for non-y variables in the model, used to calculate a steady-state baseline when <code>d.y = 0</code> and <code>baseline.y</code> is not supplied. If any differenced variables are included in the model, they should be set to 0. Ignored when <code>d.y > 0</code>
<code>baseline.y</code>	a user-supplied baseline value of y in levels. For <code>d.y = 0</code> , this overrides the steady-state calculation from <code>prediction.values</code> . For <code>d.y > 0</code> 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
<code>shock.size</code>	the size of the shock to x in the units of x. Defaults to 1 (the marginal effect)

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

Value

a list of `limit + 1` formula strings containing the fitted value formula in each period, for evaluation by `deltaMethod` in the calling function

Author(s)

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

Examples

```

# ADL model with y in levels
model.levels <- lm(y ~ x + l_1_x + l_1_y, data = toy.ts.interaction.data)

# set up formulae
pulses.levels <- pulse.calculator(x.vrbl = c("x" = 0, "l_1_x" = 1),

```

```

y.vrbl = c("l_1_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.diffs <- pulse.calculator(x.vrbl = c("x" = 0, "l_1_x" = 1),
  y.vrbl = c("l_1_d_y" = 1), limit = 5)
general.diffs <- general.calculator(d.x = 0, d.y = 1, h = -1,
  limit = 5, pulses = pulses.diffs)

## 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.diffs$formulae, d.y = 1,
  model = model.diffs, the.coef = coef(model.diffs),
  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 marginal effect (warns)
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 = "differences",
  baseline.y = NULL, shock.size = 1)

# inferences in differences, shock size of 2: scales the marginal effect
# Since we're asking for inferences.y in differences, the baseline will automatically be 0
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 = "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, [3](#)
 - GDRF.adl.plot, [5](#)
 - * **GDRF**
 - GDRF.adl.plot, [5](#)
 - GDRF.gecm.plot, [9](#)
 - gecm.plot, [13](#)
 - * **GECEM**
 - GDRF.gecm.plot, [9](#)
 - gecm.plot, [13](#)
 - * **datasets**
 - approval, [4](#)
 - toy.ts.interaction.data, [22](#)
 - * **interaction**
 - interact.adl.plot, [18](#)
 - * **internal**
 - adl.dummy.checks, [2](#)
 - GDRF.dummy.checks, [8](#)
 - gecm.dummy.checks, [12](#)
 - get.value, [17](#)
 - mpoly.subber, [21](#)
 - what.to.return, [24](#)
 - * **plot**
 - adl.plot, [3](#)
 - GDRF.adl.plot, [5](#)
 - GDRF.gecm.plot, [9](#)
 - gecm.plot, [13](#)
 - interact.adl.plot, [18](#)
 - * **utilities**
 - gecm.to.adl, [15](#)
 - general.calculator, [16](#)
 - pulse.calculator, [21](#)
 - yhat.calculator, [24](#)
- adl.dummy.checks, [2](#)
adl.plot, [3](#)
approval, [4](#)
- GDRF.adl.plot, [5](#)
GDRF.dummy.checks, [8](#)
GDRF.gecm.plot, [9](#)
gecm.dummy.checks, [12](#)
gecm.plot, [13](#)
gecm.to.adl, [15](#)
- general.calculator, [16](#)
get.value, [17](#)
- interact.adl.plot, [18](#)
- mpoly.subber, [21](#)
- pulse.calculator, [21](#)
- toy.ts.interaction.data, [22](#)
- what.to.return, [24](#)
- yhat.calculator, [24](#)