Package: tseffects (via r-universe)

June 5, 2026

Title Dynamic Inferences from Time Series (with Interactions)

Version 0.2.1

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: especially instantaneous effects for any period and cumulative effects for any period (including the long-run 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 applies 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>). We also provide the actual 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, 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

Config/pak/sysreqs cmake libgmp3-dev make libicu-dev

Repository https://sorenjordan.r-universe.dev

Date/Publication 2026-02-05 17:01:04 UTC

RemoteUrl https://github.com/sorenjordan/tseffects

RemoteRef HEAD

RemoteSha 760711542cec8f70930a7dc3bb71815cfeae22b1

Contents

`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

`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",
  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 vector of the x variables and corresponding lag orders in the ADL model
y.vrbl	named vector of the (lagged) y variables and corresponding lag orders in the 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
shock.history	the desired shock history. shock.history 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 pulse and step. For others, see Vande Kamp, Jordan, and Rajan. The default is pulse
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
inferences.x	does the user want to apply the shock history to the independent variable in levels or in differences? (For x variables where d.x is 0, this is automatically levels.) The default is levels
dM.level	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	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	return the raw calculated GDRFs as a list element under estimates. The default is FALSE
return.plot	return the visualized GDRFs as a list element under plot. The default is TRUE
return.formulae	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 Reshikesav 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
```

`GDRF.gecm.plot` *Evaluate (and possibly plot) the General Dynamic Treatment Effect (GDRF) for a Generalized Error Correction Model (GECM)*

Description

Evaluate (and possibly plot) the General Dynamic Treatment Effect (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",
  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 GECM estimates
x.vrbl	a named vector of the x variables (of the lower level of differencing, usually in levels d = 0) and corresponding lag orders in the GECM model
y.vrbl	a named 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
x.vrbl.d.x	the order of differencing of the x variable (of the lower level of differencing, usually in levels d = 0) in the GECM model
y.vrbl.d.y	the order of differencing of the y variable (of the lower level of differencing, usually in levels d = 0) in the GECM model
x.d.vrbl	a named vector of the x variables (of the higher level of differencing, usually first differences d = 1) and corresponding lag orders in the GECM model
y.d.vrbl	a named vector of the y variables (of the higher level of differencing, usually first differences d = 1) and corresponding lag orders in the GECM model
x.d.vrbl.d.x	the order of differencing of the x variable (of the higher level of differencing, usually first differences d = 1) in the GECM model
y.d.vrbl.d.y	the order of differencing of the y variable (of the higher level of differencing, usually first differences d = 1) in the GECM model
shock.history	the desired shock history. shock.history 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 pulse and step. For others, see Vande Kamp, Jordan, and Rajan. The default is pulse
inferences.y	does the user want resulting inferences about the dependent variable in levels or in differences? The default is levels
inferences.x	does the user want to apply the shock history to the independent variable in levels or in differences? The default is levels
dM.level	level of significance 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	the type of standard error to extract from the GECM model. The default is const, but any argument to vcovHC from the sandwich package is accepted
return.data	return the raw calculated GDRFs as a list element under estimates. The default is FALSE
return.plot	return the visualized GDRFs as a list element under plot. The default is TRUE
return.formulae	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

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 GDTE

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",
                                  s.limit = 10,
                                  return.plot = TRUE,
                                  return.formulae = TRUE)
names(test.pulse)
```

`GDTE.adl.plot` *Evaluate (and possibly plot) the General Dynamic Treatment Effect (GDTE) for an autoregressive distributed lag (ADL) model*

Description

Evaluate (and possibly plot) the General Dynamic Treatment Effect (GDTE) for an autoregressive distributed lag (ADL) model

Usage

```
GDTE.adl.plot(
  model = NULL,
  x.vrbl = NULL,
  y.vrbl = NULL,
  d.x = NULL,
  d.y = NULL,
  te.type = "pte",
  inferences.y = "levels",
  inferences.x = "levels",
  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 vector of the x variables and corresponding lag orders in the ADL model
`y.vrbl`	a named vector of the (lagged) y variables and corresponding lag orders in the 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
`te.type`	the desired treatment history. `te.type` determines the counterfactual series (h) that will be applied to the independent variable. -1 represents a Pulse Treatment Effect (PTE). 0 represents a Step Treatment Effect (STE). These can also be specified via `pte`, `pulse`, `ste`, and `step`. For others, see Vande Kamp, Jordan, and Rajan. The default is `pte`
`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`
`inferences.x`	does the user want to apply the counterfactual treatment to the independent variable in levels or in differences? (For x variables where `d.x` is 0, this is automatically levels.) The default is `levels`
`dM.level`	level of significance of the GDTE, calculated by the delta method. The default is 0.95
`s.limit`	an integer for the number of periods to determine the GDTE (beginning at s = 0)
`se.type`	the type of standard error to extract from the ADL model. The default is `const`, but any argument to `vcovHC` from the `sandwich` package is accepted

`return.data` return the raw calculated GDTEs as a list element under `estimates`. The default is `FALSE`

`return.plot` return the visualized GDTEs as a list element under `plot`. The default is `TRUE`

`return.formulae` return the formulae for the GDTEs as a list element under `formulae` (for the GDTEs) and `binomials` (for the treatment history). The default is `FALSE`

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

Details

We assume that the ADL model estimated is well specified, free of residual autocorrelation, balanced, and meets other standard time-series qualities. Given that, to obtain causal inferences for the specified treatment history, the user only needs a named vector of the x and y variables, as well as the order of the differencing

Value

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

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 <- lm(y ~ l_1_y + x + l_1_x, data = toy.ts.interaction.data)
test.pulse <- GDTE.adl.plot(model = model,
                                  x.vrbl = c("x" = 0, "l_1_x" = 1),
                                  y.vrbl = c("l_1_y" = 1),
                                  d.x = 0,
                                  d.y = 0,
                                  te.type = "pulse",
                                  inferences.y = "levels",
                                  inferences.x = "levels",
                                  s.limit = 20,
                                  return.plot = TRUE,
                                  return.formulae = TRUE)
names(test.pulse)

# Using Cavari's (2019) approval model (without interactions)
# Cavari's original model: APPROVE ~ APPROVE_ECONOMY + APPROVE_FOREIGN +
#     APPROVE_L1 + PARTY_IN + PARTY_OUT + UNRATE +
#      MIP_MACROECONOMICS + MIP_FOREIGN +
#     DIVIDEDGOV + ELECTION + HONEYMOON + as.factor(PRESIDENT)

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

```
# What if there was a permanent, one-unit change in the salience of foreign affairs?
cavari.step <- GDTE.adl.plot(model = cavari.model,
                                  x.vrbl = c("MIP_FOREIGN" = 0),
                                  y.vrbl = c("APPROVE_L1" = 1),
                                  d.x = 0,
                                  d.y = 0,
                                  te.type = "ste",
                                  inferences.y = "levels",
                                  inferences.x = "levels",
                                  s.limit = 10,
                                  return.plot = TRUE,
                                  return.formulae = TRUE)
```

GDTE.gecm.plot *Evaluate (and possibly plot) the General Dynamic Treatment Effect (GDTE) for a Generalized Error Correction Model (GECM)*

Description

Evaluate (and possibly plot) the General Dynamic Treatment Effect (GDTE) for a Generalized Error Correction Model (GECM)

Usage

```
GDTE.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,
  te.type = "pte",
  inferences.y = "levels",
  inferences.x = "levels",
  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 GECM estimates
x.vrbl	a named vector of the x variables (of the lower level of differencing, usually in levels d = 0) and corresponding lag orders in the GECM model
y.vrbl	a named 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
x.vrbl.d.x	the order of differencing of the x variable (of the lower level of differencing, usually in levels d = 0) in the GECM model
y.vrbl.d.y	the order of differencing of the y variable (of the lower level of differencing, usually in levels d = 0) in the GECM model
x.d.vrbl	a named vector of the x variables (of the higher level of differencing, usually first differences d = 1) and corresponding lag orders in the GECM model
y.d.vrbl	a named vector of the y variables (of the higher level of differencing, usually first differences d = 1) and corresponding lag orders in the GECM model
x.d.vrbl.d.x	the order of differencing of the x variable (of the higher level of differencing, usually first differences d = 1) in the GECM model
y.d.vrbl.d.y	the order of differencing of the y variable (of the higher level of differencing, usually first differences d = 1) in the GECM model
te.type	the desired treatment history. te.type determines the counterfactual series (h) that will be applied to the independent variable. -1 represents a Pulse Treatment Effect (PTE). 0 represents a Step Treatment Effect (STE). These can also be specified via pte, pulse, ste, and step. For others, see Vande Kamp, Jordan, and Rajan. The default is pte
inferences.y	does the user want resulting inferences about the dependent variable in levels or in differences? The default is levels
inferences.x	does the user want to apply the counterfactual treatment to the independent variable in levels or in differences? The default is levels
dM.level	level of significance of the GDTE, calculated by the delta method. The default is 0.95
s.limit	an integer for the number of periods to determine the GDTE (beginning at s = 0)
se.type	the type of standard error to extract from the GECM model. The default is const, but any argument to vcovHC from the sandwich package is accepted
return.data	return the raw calculated GDTEs as a list element under estimates. The default is FALSE
return.plot	return the visualized GDTEs as a list element under plot. The default is TRUE
return.formulae	return the formulae for the GDTEs as a list element under formulae (for the GDTEs) and binomials (for the treatment history). The default is FALSE
...	other arguments to be passed to the call to plot

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 causal inferences for the specified treatment 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 GDTEs from the GECM

Value

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

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 <- GDTE.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,
                                  te.type = "pulse",
                                  inferences.y = "levels",
                                  inferences.x = "levels",
                                  s.limit = 10,
                                  return.plot = TRUE,
                                  return.formulae = TRUE)
names(test.pulse)
```

`gecm.to.adl` *Translate the coefficients from the General Error Correction Model (GECM) to the autoregressive distributed lag (ADL) model*

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

x.vrbl	a named vector of the x variables (of the lower level of differencing, usually in levels d = 0) and corresponding lag orders in the GECM model
y.vrbl	a named 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
x.d.vrbl	a named vector of the x variables (of the higher level of differencing, usually first differences d = 1) and corresponding lag orders in the GECM model
y.d.vrbl	a named vector of the y variables (of the higher level of differencing, usually first differences d = 1) and corresponding lag orders 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` *Generate the generalized effect formulae for an autoregressive distributed lag (ADL) model, given pulse effects and shock/treatment history*

Description

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

Usage

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

Arguments

`d.x`	the order of differencing of the x variable in the ADL model. (Generally, this is the same x variable used in `pulse.calculator`)
`d.y`	the order of differencing of the y variable in the ADL model. (Generally, this is the same y variable used in `pulse.calculator`)
`h`	an integer for the shock/treatment history. `h` determines the counterfactual series that will be applied to the independent variable. -1 represents a pulse. 0 represents a step. For others, see Vande Kamp, Jordan, and Rajan
`limit`	an integer for the number of periods (s) to determine the generalized effect (beginning at 0)
`pulses`	a list of pulse effect formulae used to construct the generalized effect formulae. We expect this will be provided by `pulse.calculator`

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 treatment
general.pulse.effects <- general.calculator(d.x = 0, d.y = 0,
       h = -1, limit = s, pulses = pulse.effects)
general.pulse.effects
# Apply a step treatment
general.step.effects <- general.calculator(d.x = 0, d.y = 0,
       h = 0, limit = s, pulses = pulse.effects)
general.step.effects
```

`interact.adl.plot` *Plot the interaction in a single-equation time series model estimated via* `lm`*. 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*

Description

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

Usage

```
interact.adl.plot(
  model = NULL,
  x.vrbl = NULL,
  z.vrbl = NULL,
  x.z.vrbl = NULL,
  y.vrbl = NULL,
  effect.type = "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 vector of the "main" x variables and corresponding lag orders in the ADL model
`z.vrbl`	named vector of the "moderating" z variables and corresponding lag orders in the ADL model
`x.z.vrbl`	named 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
`y.vrbl`	named vector of the (lagged) y variables and corresponding lag orders in the ADL model
`effect.type`	whether impulse or cumulative effects should be calculated. `impulse` generates impulse effects, or short-run/instantaneous effects specific to each period. `cumulative` generates the accumulated, or long-run/cumulative effects up to each period (including the long-run multiplier). The default is `impulse`
`plot.type`	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 `lines`
`line.options`	if drawing lines, whether the moderator should be values of z (`z.lines`) or values of s (`s.lines`). The default is `z.lines`
`heatmap.options`	if drawing a heatmap, 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 `significant`
`line.colors`	what color lines would you like for line plots? 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`	what color scale would you like for the heatmap? The default is `Blue-Red`. Alternate colors must be one of `hcl.pals()`. For grayscale plots, use `Grays`. This is passed to `scale_fill_gradientn`
`z.vals`	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 level of a range of values of z. If none are provided, `interact.adl.plot` will pick

`s.vals` values for the time since the shock. 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`
number of digits to round to for the z labels in the legend (if those values are automatically calculated)

`z.vrbl.label` the name of the moderating z variable, used in plotting

`dM.level` significance level of the (cumulative) marginal effects, calculated by the delta method. The default is 0.95

`s.limit` an integer for the number of periods to determine the (cumulative) marginal effects (beginning at s = 0)

`se.type` 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` return the raw calculated (cumulative) marginal effects as a list element under `estimates`. The default is `FALSE`

`return.plot` return the visualized (cumulative) marginal effects as a list element under `plot`. The default is `TRUE`

`return.formulae`
return the formulae for the (cumulative) marginal effects as a list element under `formulae` (for the (cumulative) marginal effects) 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 Reshikesav 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),
```

```
 effect.type = "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,
    effect.type = "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),
    effect.type = "impulse",
    plot.type = "heatmap",
    heatmap.options = "all",
    s.limit = 20)

## End(Not run)
```

`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

x.vrbl	a named vector of the x variables and corresponding lag orders in an ADL model
y.vrbl	a named vector of the (lagged) y variables and corresponding lag orders in an ADL model
limit	an integer for the number of periods (s) to determine the pulse effect (beginning at 0)

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

Index