

Package: dpm (via r-universe)

October 6, 2026

Type Package

Title Dynamic Panel Models Fit with Maximum Likelihood

Version 1.3.0

Author Jacob A. Long [aut, cre]

(<<https://orcid.org/0000-0002-1582-6214>>), Richard A. Williams
[aut], Paul D. Allison [aut]
(<<https://orcid.org/0000-0002-0646-5242>>)

Maintainer Jacob A. Long <jacob.long@sc.edu>

Description Implements the dynamic panel models described by Allison, Williams, and Moral-Benito (2017 <[doi:10.1177/2378023117710578](https://doi.org/10.1177/2378023117710578)>) in R. This class of models uses structural equation modeling to specify dynamic (lagged dependent variable) models with fixed effects for panel data. Additionally, models may have predictors that are only weakly exogenous, i.e., are affected by prior values of the dependent variable. Options also allow for random effects, dropping the lagged dependent variable, and a number of other specification choices.

License MIT + file LICENSE

BugReports <https://github.com/jacob-long/dpm/issues>

URL <https://github.com/jacob-long/dpm>, <https://dpm.jacob-long.com/>

Encoding UTF-8

LazyData true

Depends R (>= 2.10), lavaan, methods

Imports panelr, stringr, rlang, dplyr, crayon, jtools (>= 2.0.1),
Formula, stats4

Suggests broom, tibble, covr, testthat

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Config/pak/sysreqs cmake make libicu-dev

Repository <https://jacob-long.r-universe.dev>
Date/Publication 2026-04-06 14:50:28 UTC
RemoteUrl <https://github.com/jacob-long/dpm>
RemoteRef HEAD
RemoteSha 7e8ad1906464c485b077fdb7639e22660678e8c6

Contents

dpm	2
dpm-class	5
get_syntax	5
get_wide_data	6
lav_summary	6
summary,dpm-method	7
tidy.dpm	8
update	9

Index	11
--------------	-----------

dpm	<i>Dynamic panel models fit with maximum likelihood</i>
-----	---

Description

Estimate dynamic panel models with fixed effects via maximum likelihood estimation.

Usage

```
dpm(  
  formula,  
  data,  
  error.inv = FALSE,  
  const.inv = FALSE,  
  alpha.free = FALSE,  
  y.lag = 1,  
  y.free = FALSE,  
  x.free = FALSE,  
  fixed.effects = TRUE,  
  partial.pre = FALSE,  
  print.only = FALSE,  
  id = NULL,  
  wave = NULL,  
  err.inv = NULL,  
  weights = NULL,  
  ...  
)
```

Arguments

<code>formula</code>	Model formula. See details for instructions on specifying parameters properly.
<code>data</code>	Data frame in "long" format. Prefers a "panel_data" object.
<code>error.inv</code>	Constrain the error variance to be equal across waves. Default is FALSE.
<code>const.inv</code>	Constrain the dependent variable's variance to be equal across waves (or makes its intercept equal across waves). This removes cross-sectional dependence. Default is FALSE.
<code>alpha.free</code>	Estimate each wave of the dependent variable's loading on the alpha latent variable. Default is FALSE, meaning each wave has a loading of 1.
<code>y.lag</code>	Which lag(s) of the dependent variable to include in the regression. Default is 1, but any number or vector of numbers can be used.
<code>y.free</code>	If TRUE, allows the regression coefficient(s) for the lagged dependent variable to vary over time. Default is FALSE. You may alternately provide a number or vector of numbers corresponding to which lags should vary freely.
<code>x.free</code>	If TRUE, allows the regressions coefficient(s) for the predictor(s) to vary over time. Default is FALSE. If TRUE, the predictor regression coefficient(s) can vary over time. Alternately, you may provide a character vector of predictors to allow to vary if you only want a subset of predictors to vary.
<code>fixed.effects</code>	Fit a fixed effects model? Default is TRUE. If FALSE, you get a random effects specification instead.
<code>partial.pre</code>	Make lagged, predetermined predictors (i.e., they are surrounded by <code>pre()</code> in the model formula) correlated with the contemporaneous error term, as discussed in Allison (2022)? Default is FALSE.
<code>print.only</code>	Instead of estimating the model, print the lavaan model string to the console instead.
<code>id</code>	Name of the data column that identifies which individual the observation is. Not needed if data is a "panel_data" object.
<code>wave</code>	Name of the data column that identifies which wave the observation is from. Not needed if data is a "panel_data" object.
<code>err.inv</code>	Deprecated, same purpose as <code>error.inv</code> .
<code>weights</code>	Equivalent to the argument to <code>lm</code> , presumably the unquoted name of a variable in the data that represents the weight. It is passed to <code>lavaan()</code> 's <code>sampling.weights</code> argument.
<code>...</code>	Extra parameters to pass to sem . Examples could be <code>missing = "fiml"</code> for missing data or <code>estimator = "MLM"</code> for robust estimation.

Details

The right-hand side of the formula has two parts, separated by a bar (`|`). The first part should include the time-varying predictors. The second part, then, is for the time-invariant variables. If you put a time-varying variable in the second part of the formula, by default the first wave's value of that variable is treated as the constant.

You must include time-varying predictors. If you do not include a bar in the formula, all variables are treated as time-varying.

If you would like to include an interaction between time-varying and time-invariant predictors, you can add a third part to the formula to specify that term.

Predetermined variables:

To set a variable as predetermined, or weakly exogenous, surround the variable with a `pre` function. For instance, if you want the variable `union` to be predetermined, you could specify the formula like this: `wks ~ pre(union) + lwage | ed`, where `wks` is the dependent variable, `lwage` is a strictly exogenous time-varying predictor, and `ed` is a strictly exogenous time-invariant predictor.

To lag a predictor, surround the variable with a `lag` function in the same way. Note that the `lag` function used is specific to this package, so it does not work the same way as the built-in `lag` function (i.e., it understands that you can only lag values *within* entities).

Note: CFI and TLI model fit measures are computed using the same baseline model as Stata's `xtdepml`, so the reported values closely match that implementation for comparable specifications.

Value

An object of class `dpm` which has its own summary method.

The `dpm` object is an extension of the `lavaan` class and has all the capabilities of `lavaan` objects, with some extras.

It contains extra slots for:

- `mod_string`, the character object used to specify the model to `lavaan`. This is helpful if you want to fit the model yourself or wish to check that the specification is correct.
- `wide_data`, the widened data frame necessary to fit the SEM.

Author(s)

Jacob A. Long, in consultation with Richard A. Williams and Paul D. Allison. All errors are Jacob's.

References

Allison, P. D., Williams, R., & Moral-Benito, E. (2017). Maximum likelihood for cross-lagged panel models with fixed effects. *Socius*, 3, 1–17. <http://journals.sagepub.com/doi/10.1177/2378023117710578>

Examples

```
# Load example data
data("WageData", package = "panelr")
# Convert data to panel_data format for ease of use
wages <- panel_data(WageData, id = id, wave = t)

# Replicates Allison, Williams, & Moral-Benito (2017) analysis
fit <- dpm(wks ~ pre(lag(union)) + lag(lwage) | ed, data = wages,
          error.inv = TRUE, information = "observed")
# Note: information = "observed" only needed to match Stata/SAS standard errors
summary(fit)
```

dpm-class	<i>Dynamic Panel Model (dpm) class</i>
-----------	--

Description

Models fit using `dpm()` return values of this class, which inherits from `lavaan-class`.

Slots

`call_info` A list of metadata about the arguments used.
`call` The actual function call.
`mod_string` The model formula passed to lavaan.
`wide_data` The data provided to the data argument in the function
`formula` The `Formula::Formula()` object provided to `dpm()`. call.

<code>get_syntax</code>	<i>Retrieve lavaan model syntax from fitted dpm model</i>
-------------------------	---

Description

This helper function provides a simple way to retrieve the lavaan model syntax from a fitted `dpm()` object.

Usage

```
get_syntax(model, print = TRUE)
```

Arguments

<code>model</code>	A dpm object.
<code>print</code>	Print the syntax to the console so it is formatted properly? Default is TRUE.

Value

Returns a string with the lavaan model syntax for `model`. If `print` is TRUE, it is printed to the console as well.

Examples

```
data("WageData", package = "panelr")
wages <- panel_data(WageData, id = id, wave = t)
fit <- dpm(wks ~ pre(lag(union)) + lag(lwage), data = wages)
get_syntax(fit)
```

get_wide_data	<i>Retrieve wide-format data from fitted dpm model</i>
---------------	--

Description

This helper function provides a simple way to retrieve the widened data from a fitted `dpm()` object.

Usage

```
get_wide_data(model)
```

Arguments

model	A dpm object.
-------	---------------

Value

A `data.frame` with input data transformed from "long" to "wide" format, with just one row per person/entity. Internally, this is generated by calling `panelr::widen_panel()` after some preprocessing.

Examples

```
data("WageData", package = "panelr")
wages <- panel_data(WageData, id = id, wave = t)
fit <- dpm(wks ~ pre(lag(union)) + lag(lwage), data = wages)
get_wide_data(fit)
```

lav_summary	<i>lavaan-style summary for dpm objects</i>
-------------	---

Description

This is just a quick way to get lavaan's summary instead the more terse summary designed for dpm objects.

Usage

```
lav_summary(x, ...)
```

Arguments

x	The <code>dpm</code> object
...	Other arguments to the lavaan function.

Value

Returns a `lavaan.summary` object which contains various model data described in ?lavaan::summary, lavaan-method.

Examples

```
# Load example data
data("WageData", package = "panelr")
# Convert data to panel_data format for ease of use
wages <- panel_data(WageData, id = id, wave = t)

fit <- dpm(wks ~ pre(lag(union)) + lag(lwage) | ed, data = wages)
lav_summary(fit)
```

summary,dpm-method	<i>Summarize dpm objects</i>
--------------------	------------------------------

Description

The summary method is designed to offer similar arguments to lavaan's summary, but with shorter and more domain-specific output.

Usage

```
## S4 method for signature 'dpm'
summary(
  object,
  standardized = FALSE,
  ci = FALSE,
  se = TRUE,
  zstat = TRUE,
  pvalue = TRUE,
  ci.level = 0.95,
  boot.ci.type = c("perc", "norm", "basic", "bca.simple"),
  digits = getOption("dpm-digits", 3),
  ...
)
```

Arguments

<code>object</code>	A dpm object.
<code>standardized</code>	Use lavaan's method for standardizing coefficients? Default is FALSE.
<code>ci</code>	Show confidence intervals? Default is FALSE.
<code>se</code>	Show standard errors? Default is TRUE.
<code>zstat</code>	Show the z statistic? Default is TRUE.
<code>pvalue</code>	Show p values? Default is TRUE.

<code>ci.level</code>	How wide should the confidence intervals be? Ignored if <code>ci</code> is <code>FALSE</code> . Default is <code>.95</code> .
<code>boot.ci.type</code>	If the model was fit with bootstrapped standard errors and <code>ci</code> is <code>TRUE</code> , which method should be used for finding the intervals? Default is <code>"perc"</code> .
<code>digits</code>	How many digits should be printed in the model summary? Default is 3. You can set a default by setting the option <code>"dpm-digits"</code> .
<code>...</code>	Ignored.

Value

Returns a `summary.dpm` object, which is a list with three elements:

- `model`: The `dpm` object.
- `coefficients`: A data frame containing coefficient estimates, standard errors, p values, and so on.
- `fitmeasures`: A numeric vector containing model fit information.

The primary function of the object is to be printed to the console.

<code>tidy.dpm</code>	<i>Tidy methods for dpm</i>
-----------------------	-----------------------------

Description

`dpm` objects support the **broom** package's `tidy` method.

Usage

```
## S3 method for class 'dpm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'dpm'
glance(x, ...)
```

Arguments

<code>x</code>	A <code>dpm</code> object.
<code>conf.int</code>	Logical indicating whether or not to include a confidence interval in the tidy data frame.
<code>conf.level</code>	The confidence level to use for the confidence interval when <code>conf.int</code> is <code>TRUE</code> . Default is <code>.95</code> , corresponding to a 95% confidence interval.
<code>...</code>	Other arguments passed to summary.dpm .

Value

A `tibble::tibble()` with information about model components. These will be coefficient estimates (for `tidy()`) or model fit (for `glance()`), following the naming standards established by the **broom** package.

Examples

```
if (requireNamespace("broom")) {
  library(broom)
  # Load example data
  data("WageData", package = "panelr")
  # Convert data to panel_data format for ease of use
  wages <- panel_data(WageData, id = id, wave = t)

  fit <- dpm(wks ~ pre(lag(union)) + lag(lwage) | ed, data = wages)
  tidy(fit)
}
```

update	<i>Various methods for dpm objects</i>
--------	--

Description

R likes it when these things have documentation.

Usage

```
update(object, ...)

## S3 method for class 'dpm'
update(object, formula., ..., evaluate = TRUE)

## S4 method for signature 'dpm'
update(object, formula., ..., evaluate = TRUE)

## S4 method for signature 'dpm'
show(object)

coef(object, ...)

## S3 method for class 'dpm'
coef(object)

## S4 method for signature 'dpm'
coef(object)

## S4 method for signature 'dpm'
formula(x)
```

Arguments

<code>object</code>	A dpm object
<code>...</code>	Other arguments to <code>update</code> .
<code>formula.</code>	An updated formula (optional)
<code>evaluate</code>	If updating, should the updated model be updated or just return the call? Default is TRUE, re-run the model.
<code>x</code>	A dpm object

Value

`update.dpm()`: Returns an updated dpm object.

`coef.dpm()`: Returns a numeric vector of coefficients. If the model was fit with `x.free = TRUE` and/or `y.free = TRUE`, the coefficient names will be formatted with an underscore and the wave corresponding to which time period the coefficient is estimated for.

`formula.dpm()`: Returns the formula used to fit the model as a Formula object. The formula is the input to `dpm()`, not the lavaan syntax.

`show.dpm()`: Returns an invisible "NULL" while printing model info to console.

Examples

```
data("WageData", package = "panelr")
wages <- panel_data(WageData, id = id, wave = t)
fit <- dpm(wks ~ pre(lag(union)) + lag(lwage), data = wages)

# Re-run model without `lag(lwage)` term
update(fit, . ~ . - lag(lwage))
```

Index

`coef (update)`, [9](#)
`coef, dpm-method (update)`, [9](#)
`coef.dpm (update)`, [9](#)

`dpm`, [2](#), [6](#)
`dpm()`, [5](#), [6](#)
`dpm-class`, [5](#)

`formula, dpm-method (update)`, [9](#)
`Formula::Formula()`, [5](#)

`get_syntax`, [5](#)
`get_wide_data`, [6](#)
`glance.dpm (tidy.dpm)`, [8](#)

`lav_summary`, [6](#)

`sem`, [3](#)
`show, dpm-method (update)`, [9](#)
`summary, dpm-method`, [7](#)
`summary.dpm`, [8](#)

`tidy.dpm`, [8](#)

`update`, [9](#)
`update, dpm-method (update)`, [9](#)
`update.dpm (update)`, [9](#)