A Program to Estimate and Interpret Models of Dynamic Compositional Dependent Variables: New Features for Dynsimpie

Yoo Sun Jung[1], Flávio D. S. Souza[2], Andrew Q. Philips[3], Amanda Rutherford[4], and Guy D. Whitten[5]

[1,2,5]*Department of Political Science, Texas A&M University*

[3]*Department of Political Science, University of Colorado Boulder*

[4]*School of Public and Environmental Affairs, Indiana University*

Ungated Version

May contain minor discrepancies from publisher's PDF.

Please Cite As:

Jung, Y. S., Souza, F. D. S., Philips, A. Q., Rutherford, A., & Whitten, G. D. (2020). A Program to Estimate and Interpret Models of Dynamic Compositional Dependent Variables: New Features for Dynsimpie. *The Stata Journal*, 20(3).

Abstract

Philips, Rutherford and Whitten (2016, *Stata Journal* 16 (3): 662-677) introduced `dynsimpie`, a program to examine dynamic compositional dependent variables. In this article, we present an update to `dynsimpie` and three new ado-files: and `dynsimpie`, `cfbplot`, `effectsplot`, and `dynsimpiecoef`. These updates greatly enhance the range of models that can be estimated and the ways in which model results can now be presented. The program `dynsimpie` has been updated so that users can obtain both prediction plots and change-from-baseline plots using post-estimation commands. With the new command `dynsimpiecoef`, various types of coefficient plots can also be obtained. We illustrate these improvements using monthly data on support for political parties in the United Kingdom.

***Keywords*:** time series, seemingly unrelated regression, cointegration, dynamic modeling, dynamic composition, error correction, lagged dependent variable

1 Introduction

Compositional variables, which represent data with the property of a constant sum, are common to a wide range of empirical fields. Most commonly, these variables are made up of values which sum to 100% for each unit of analysis. Examples include party shares of the popular vote, budgetary expenditures across spending categories, and the percentages by weight of minerals in rocks. The properties of compositional variables are such that they require special treatment when they are modeled as dependent variables (Aitchison, 1982). But, despite the prevalence of compositional variables, there are few software tools available for modeling them, especially in cases where the data are measured for the same unit across multiple points in time.

Philips, Rutherford and Whitten (2016*b*) introduced `dynsimpie`, a program to help estimate and interpret results from models of dynamic compositional dependent variables which we can write as y_{tj}, measured across categories $j = 1, 2, ..., J$ and over time points $t = 1, 2, ..., T$. Following the notation used by Philips, Souza and Whitten (2019), this program first transforms the dependent variable into $J-1$ logged compositions

$$s_{tj} = ln\left(\frac{y_{tj}}{y_{t1}}\right) \ \forall j \neq 1 \tag{1}$$

where the choice of the baseline category, $j = 1$, is irrelevant to subsequent inferences as long as it never takes on a value of 0. As discussed in Philips, Souza and Whitten (2019), $J-1$ error correction model (ECM) equations are then estimated with the following specification:

$$\Delta s_{tj} = \beta_{0j} - \alpha_j s_{jt-1} + \boldsymbol{\beta}_{Lj}\boldsymbol{x}_{t-1} + \boldsymbol{\beta}_{Sj}\Delta\boldsymbol{x}_t + \boldsymbol{\varepsilon}_{jt} \tag{2}$$

where:

- Δs_{tj} is the change in the logged ratio of dependent variable category "j" for $j > 1$ to baseline category $j = 1$ from time "t-1" to time "t",

- β_{0j} is a constant term,
- $\boldsymbol{x}_t$ is a vector of independent variable values at time "t",
- $-\alpha_j$ is a vector of adjustment parameters that measure the long-run error correction processes,
- $\boldsymbol{\beta}_{Sj}$ is a vector of short-run effects,
- $\boldsymbol{\beta}_{Lj}$, is a vector of parameters that can be combined with $-\alpha_j$ to calculate estimated long-run effects of changes in each independent variable,
- and ε_{jt} is a stochastic disturbance term that may be correlated across the "j" equations.

Because of the extensive number of parameters estimated and the non-linear nature of the resulting inferences, the original `dynsimpie` command also produced a graph to help with model interpretation. This graph, called a "change-from-baseline" plot, shows the results from a user-specified scenario in which there is a one-time change (shock) to the value of a chosen independent variable.

In this article, we present an update to `dynsimpie`. The updated `dynsimpie` suite of commands now comprises four ado-files: `dynsimpie`, `cfbplot`, `effectsplot`, and `dynsimpiecoef`. This suite of commands requires the installation of Tomz, Whittenberg and King's (2003) `clarify` package and Jann's (2014) `coefplot` package. The base command, `dynsimpie`, has been substantially enhanced.[1] Users may now specify multiple shock variables within a single option. They can similarly specify multiple shock values within only one option. With the updated command, users may choose between an error correction model specification such as that in Equation 2 or they may specify a lagged dependent variable model specification such as

$$s_{tj} = \phi_j s_{jt-1} + \beta_{0j} + \boldsymbol{\beta}_{sj}\boldsymbol{x}_t + \varepsilon_{jt} \tag{3}$$

where:

[1] It has also been simplified to avoid unnecessary lines of code and computing time.

- ϕ_j is the parameter on the lagged dependent variable which, together with the values of β_{sj} may be used to calculate long-run changes.

- all other terms are as defined above in Equation 2.

Where users find that a lagged dependent variable (LDV) model is not appropriate, the new `ecm` option allows for an error correction model (ECM). Additionally, users can estimate models with an interaction between explanatory variables of interest by specifying the `interaction(`*`string`*`)` and `intype` options.

With the post-estimation commands—`cfbplot` and `effectsplot`—users can now easily obtain change-from-baseline plots and prediction plots. These post-estimation commands produce plots displaying the model-based simulations of both the short- and long-run effects on each category of the compositional dependent variable. Short for "change-from-baseline" plot, `cfbplot` displays a plot of the simulated output. The expected proportion of each of the compositional dependent variables is plotted across time, along with the associated confidence intervals. This was the only plot type available with the original version of `dynsimpie`. In this updated `dynsimpie` package, the post-estimation command `effectsplot` displays effects plots (prediction plots) from estimated `dynsimpie` results.

The new command `dynsimpiecoef` produces various plots of the seemingly unrelated regression (SUR) coefficient results used in `dynsimpie`. In doing this, `dynsimpiecoef` allows two different ways of presenting the same information: displaying coefficients organized by dependent variable category or by independent variable. We illustrate these improvements using compositional data on monthly voter support for the political parties in the United Kingdom.

2 The dynsimpie command

2.1 Syntax

The syntax of the `dynsimpie` command is as follows:

`dynsimpie` *indepvars* [if] [in], `dvs`(*varlist*) [`ecm` `ar`(*numlist*) `shockvars`(*varlist*) `shockvals`(*numlist*) `dummy`(*varlist*) `dummyset`(*numlist*) `dummyshockvar`(*varname*) `interaction`(*varname*) `intype`(*string*) `time`(*#*) `sigs`(*numlist*) `range`(*#*) `percentage` `pv` `killtable`]

indepvars is a list of continuous independent variables to be included in the model that do not experience a counterfactual shock. Notice that `dynsimpie` automatically transforms these variables (and any additional variables specified in other options) into lags and first differences for estimation in error-correction form when `ecm` is specified. By default, `dynsimpie` runs a lagged dependent variable (LDV) model. Other options allow for the addition of dummy variables to the model, provide the ability to shock multiple independent variables at the same point in time, and estimate multiplicative interactions.

2.2 Options

`dvs`(*varlist*) is a list of the component parts of the compositional dependent variable. Each of these should be expressed as either proportions (summing to 1) or as percents (summing to 100). This program will issue an error message if neither of these criteria is met. The program takes the log of the proportion of each category relative to the proportion of an arbitrary "baseline" category. For instance, if there were J dependent variable categories in `dvs(varlist)`, these programs would create $J-1$ variables $s_{tj} = ln(y_{tj}/y_{t1})$, where the first category, y_{t1}, is the baseline. This option is required.

`ecm` specifies the choice of an error correction model. It automatically transforms the dependent variables and independent variables into lags and first differences for estimation in error-correction form. The default is to estimate a lagged dependent variable model.

`ar(#)` is a numeric list of dependent variable lags to be included in the model. Up to three values may be inputted following Stata's *numlist* conventions and they need not be consecutive. This option is not allowed with `ecm` but is required in its absence.

dummy(*varlist*) if specified, the program will include these in a vector of dummy variables in the model. In dynsimpie, dummy variables enter the estimation equation in lagged form and first-differenced form. This is not the case in dynsimpieldv, which takes dummy( ) exactly as specified.

interaction(*varname*) is a dichotomous variable, not included as a control or shock variable elsewhere, to enter a multiplicative interaction. In dynsimpie, the variable entered in interaction is interacted with the first variable listed in shockvars( ).

sigs(*numlist*) is a numeric list of the significance levels for the percentile confidence intervals. The default is for 95% confidence intervals. At most, two significance levels may be listed in sigs().

killtable suppresses the automatic generation of the seemingly unrelated regression (SUR) results. By default, a table of estimates is displayed in the results window.

shockvars(*varlist*) is a list of continuous independent variables that are subject to counterfactual one-period shocks specified in shockvals(). The time in which all shocks enter the model is specified in time(). The variables listed in shockvars( ) must not be simultaneously included in *indepvars*. When an error correction modeling technique is chosen by specifying ecm, the shock first affects the first-differenced values of the chosen variable at time t for one time period. This shock then affects the lag of the variable specified in shockvars(), starting at $t+1$ and lasting through the rest of the simulated period. Failure to specify both this option and dummyshockvar will lead to an all-baseline simulation. At most, users can specify three variables in shockvars().

shockvals(*numlist*) is a numeric list of the amount to shock each respective variable in shockvars() by for one period at time *t*, specified in time(). The number of values in shockvals() must be equal to the number of variables in shockvars().

intype(*string*) may be specified intype(*on*) or intype(*off*). The variable specified in interaction

takes the value 1 when `intype( )` is *on* and 0 when `intype` is *off*. This option is not allowed with `ecm`.

`time`(*#*) is the time in which the variables specified in `shockvars()` and `dummyshockvar()` experience a one-period shock. The default is `time(5)`.

`range`(*#*) gives the length of the scenario to simulate. `range`(*#*) must always be more than the `time`(*#*) at which the shock occurs. The default is `range(20)`.

`dummyset`(*numlist*) specifies alternative values for each of the dummy variables specified in `dummy()` to be used throughout the simulation. By default, each of the dummy variables in `dummy()` will be set to 0 throughout the simulation.

`dummyshockvar`(*varname*) is a dummy variable that is subject to a counterfactual one-period shock at time *t*, specified in `time()`. Variables specified in `dummyshockvar( )` must not be simultaneously specified in `dummy( )`. When `ecm` is specified, the shock first affects the first-differenced `dummyshockvar()` at time *t* for one time period. This shock then affects the lagged `dummyshockvar()`, starting at $t+1$ and lasts through the rest of the period. Failure to specify both this option and `shockvars()` will lead to an all-baseline simulation.

`pv` by default, this program will calculate expected proportions. This option will calculate predicted proportions instead.

`percentage` by default, this program produces plots of expected proportions. This option produces plots of percentages instead.

Note that users may also specify time lags or differences of independent variables in `shockvars` and `indepvars`. Time lags (but not time differences) may also be specified for dummy variables in `dummyshockvar`, `dummy`, and `interaction`. To specify lags or differences, users must employ Stata's time-series operators (e.g., *L1., L2., D.,* etc). Manually generating and saving lagged or differenced variables and then entering them in the program will lead to incorrectly

induced shocks, since the program will not be able to recognize the type of time-series transformation previously performed on the variable. In no instance should users employ time-series operators when `ecm` is specified (because the program automatically performs the relevant time-series transformations in this case).

2.3 dynsimpie postestimation

The postestimation commands available after `dynsimpie` are `cfbplot` and `effectsplot`. `cfbplot` runs as a post-estimation command when using the program `dynsimpie` and displays a change-from-baseline plot of the simulated output. The expected proportion of each of the compositional dependent variables is plotted against time, along with the associated confidence intervals. `effectsplot` runs as a post-estimation command when using the program `dynsimpie` and displays effects plots (prediction plots) from estimated results. This produces plots which visualize long- and short-run effects of each of the compositional dependent variables, along with the associated confidence intervals. This is akin to a marginal effects plot.

3 The dynsimpiecoef command

3.1 Syntax

The syntax for `dynsimpiecoef` is

`dynsimpiecoef` *indepvars* [if] [in], `dvs`(*varlist*) [`ecm` `ar`(*numlist*) `dummy`(*varlist*) `interaction`(*varname*) `smooth` `range`(*#*) `sigs`(*numlist*) `row`(*numlist*) `xsize`(*numlist*) `all` `vertical` `angle`(*numlist*) `killtable`]

indepvars is a list of continuous independent variables to be included in the model that do not experience a counterfactual shock.

3.2 Options

`dvs`(*varlist*) is a list of the component parts of the compositional dependent variable. Each of these should be expressed as either proportions (summing to 1) or as percents (summing to 100). The program takes the log of the proportion of each category relative to the proportion of an arbitrary "baseline" category. This option is required.

`ecm` specifies the choice of an error correction model. It automatically transforms the dependent variables and independent variables into lags and first differences for estimation in error-correction form. The default is to estimate a lagged dependent variable model.

`ar(#)` is a numeric list of dependent variable lags to be included in the model. Up to three values may be inputted following Stata's *numlist* conventions and they need not be consecutive. This option is not allowed with `ecm` but is required in its absence.

`dummy`(*varlist*) if specified, the program will include these in a vector of dummy variables in the model. In `dynsimpie`, dummy variables enter the estimation equation in lagged form and first-differenced form. This is not the case in `dynsimpieldv`, which takes `dummy( )` exactly as specified.

`interaction`(*varname*) is a dichotomous variable, not included as a control or shock variable elsewhere, to enter a multiplicative interaction. In `dynsimpiecoef`, the variable entered in `interaction` is interacted with the first variable listed in *indepvars*. `interaction` is optional but, when specified, must be coded 0/1. This option is not allowed with `ecm`.

`sigs`(*numlist*) is a numeric list of the significance levels for the percentile confidence intervals. The default is for 95% confidence intervals. At most, two significance levels may be listed in `sigs()`.

`killtable` suppresses the automatic generation of the seemingly unrelated regression (SUR) results. By default, a table of estimates is displayed in the results window.

`smooth` by default, `dynsimpiecoef` will generate coefficient plots with 95 percent confidence intervals. This option adds confidence intervals for "50 equally spaced levels (1,3, ..., 99) with graduated color intensities and varying line widths," as in `coefplot` (Jann, 2014, p.729). `sigs()` must not be specified with `smooth`.

`row`(*numlist*) specifies the number of rows of the combined graph. Depending on the number of categories of the dependent variable and the number of independent variables in the model, a combined graph with multiple rows might be desirable. The default is `row(1)` if `vertical` is not specified. The default is `row(3)` with the `vertical` option.

`xsize`(*numlist*) specifies the width of the combined graph in inches. Depending on the number of categories of the dependent variable and the number of independent variables in the model, it might be desirable to make the combined graph wider than usual using this option. The default is `xsize(5)`.

`all` if specified, `dynsimpiecoef` automatically produces coefficients plots for all possible compositions of dependent variable categories regardless of the baseline category. This option allows users to compare SUR results for all pairs of dependent variable categories. Thus, the order of dependent variable categories specified in `dvs`(*varlist*) does not matter in `dynsimpiecoef`.

`vertical` if specified, `dynsimpiecoef` creates coefficient plots for a particular independent variable across all possible pairwise compositions of dependent variable categories. With this option, `dynsimpiecoef` produces a collection of coefficient plots for each independent variable and saves coefficient plots for such variables automatically and separately in the working directory.

`angle`(*angle*) specifies the angle for the labels on the *x*-axis in the combined graph. This must be specified with the `vertical` option. The default is `angle(90)`, meaning that all labels are plotted at a 90-degree angle from the *x*-axis.

4 Examples: `dynsimpie`

4.1 `LDV Models`

By default, the `dynsimpie` command allows users to estimate effects of explanatory variables on compositional outcomes through a lagged dependent variable (LDV) model specification such as Equation 3 that is estimated as a system of seemingly unrelated regressions. This command allows users to include lags of dependent and independent variables as well as a multiplicative interaction between a dichotomous independent variable of choice and a continuous shock variable. To illustrate the functionality of the new programs, we use the same monthly UK party support data as Philips, Rutherford and Whitten (2016*a*,*b*). These data are from the 2004-2010 period when the Labour Party was in government and include the switch in Labour leader and thus Prime Minister from Tony Blair to Gordon Brown. During this period, the three main political parties were Labour, the Conservatives, and the Liberal Democrats. In our examples, support for these three main political parties are the categories of our dependent variable.[2]

Imagine that we are interested in estimating the effect of a one-standard-deviation increase in the average national economic retrospective evaluations on UK party support, while controlling for the percentage of respondents identifying with the party of the prime minister. Further imagine that we have theoretical reason to believe that the previous period's composition of party support affects the current period. Finally, we expect the effect to differ between Gordon Brown's and Tony Blair's tenures.

The `dynsimpie` command can model these expectations well by including a multiplicative interaction and a lagged dependent variable as shown in the code. Since `intype( )` is *on*, the

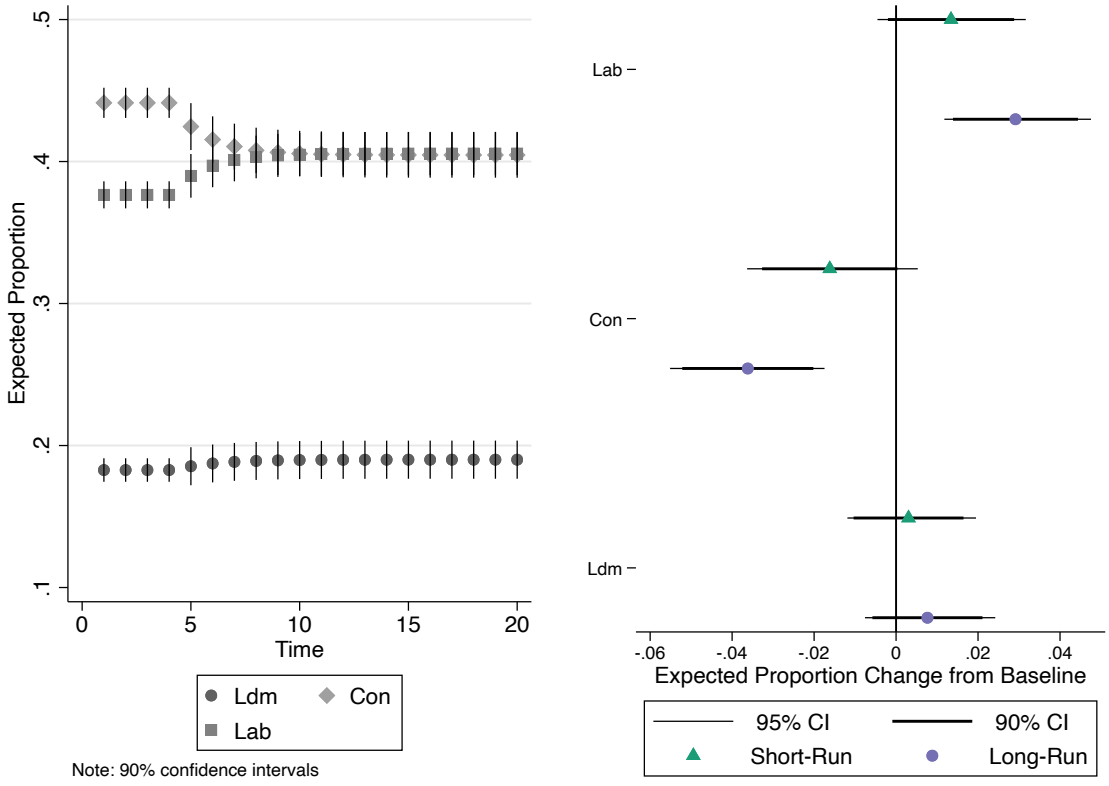
[2]Following the led of Philips, Rutherford and Whitten (2016*a*,*b*), we restrict our analyses to these three main political parties in order to have a more simple example to showcase the capabilities of these commands. We are thus modeling support for the three main political parties and ignoring fluctuations in support for what were then smaller parties, such as the Scottish Nationalist Party and the UK Independence Party. As demonstrated by Philips, Rutherford and Whitten (2015), this modeling approach and these commands can easily be extended to include dependent variables with more than three categories.

interaction variable is assumed to be 1 (Brown is in power). The option `killtable` is included in order to omit a regression table from the results window in Stata. Notice that we use the `cfbplot` postestimation command to produce a change-from-baseline plot.

```
dynsimpie all_pidW, dvs(Lab Ldm Con)  ar(1)   ///
shockvars(all_nat_retW) shockvals(.4882241) interaction(brown)  ///
time(5) range(20)  intype(on) sigs(90 95) killtable

cfbplot
effectsplot
```

Figure 1: Left panel: the simulated expected values of compositions over time. Right panel: the simulated effect of a 1-SD increase in the monthly average retrospective evaluation of the economy using the `sigs(90 95)` option in `dynsimpie`.



This plot is shown on the left panel in Figure 1. An effects plot can be produced with a different

postestimation command: `effectsplot`, shown on the right panel of Figure 1. Notice that the short- and long-run effects depicted here are the same or very close to each other.[3] This is because they are essentially two different ways of displaying the effects from the same shock.

In order to evaluate interaction effects between any two variables, researchers often produce figures and estimates of the effects of one of these variables while holding the other variable at specific values. In our case, it makes sense to evaluate simulation results from shocks to a primary shock variable, listed first in `shockvals( )`, at two distinct values of the dichotomous interaction variable. The `intype` option helps with this. When `intype` is set to *on*, as in Figure 1, we observe the effect of a shock to `shockvars` while the interaction variable is held at 1 (Brown government, in our running example). In Figure 2 we reproduce these figures with `intype` set to *off*—where the interaction variable is held at 0 (Blair government). The code for the plots on the left and right panels of Figure 2 is as follows:

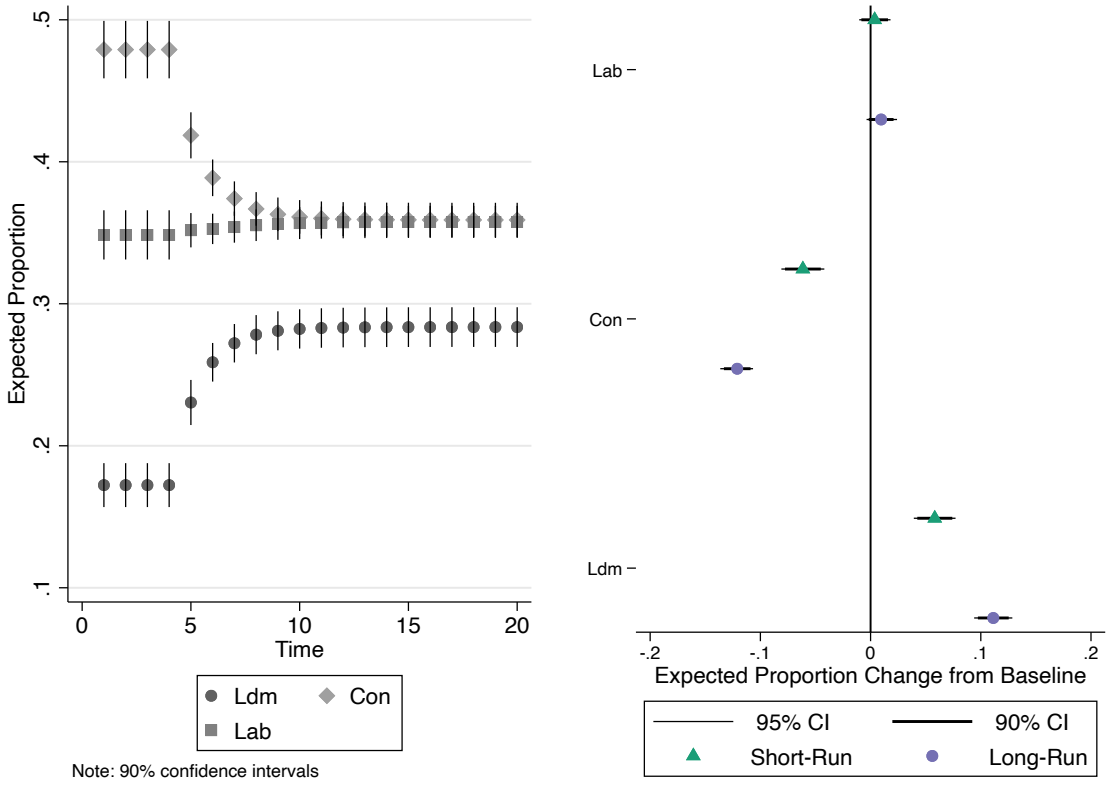
```
dynsimpie all_pidW, dvs(Lab Ldm Con) ar(1)   ///
shockvars(all_nat_retW) shockvals(.4882241) interaction(brown)  ///
time(5) range(20)  intype(off) sigs(90 95) killtable

cfbplot
effectsplot
```

Updating `dynsimpie` to allow for non-error-correction modeling contributes to a vastly more flexible program. First, users can now use `dynsimpie` when an error correction model is not warranted (such as when cointegration is not present). As a result, users can experiment with adding or removing time differences and lags among the independent variables as they see fit. Additionally, users can introduce a multiplicative interaction with one of the shock variables, which is useful in many theoretical contexts. Together, these additions to the program largely expand the domain of modeling choices available to users.

[3]The long-run effects from a particular scenario will sometimes be slightly different when the full long-run change depicted in an effects plot has not come into place by the end of the change-from-baseline figure.

Figure 2: Left panel: the simulated expected values of compositions over time. Right panel: the simulated effect of a 1-SD increase in the monthly average retrospective evaluation of the economy using the `sigs(90 95)` option in `dynsimpie`.



4.2 ECM

Users who find that an error correction model (ECM) is statistically and theoretically warranted, can specify the option `ecm` to estimate an ECM-based `dynsimpie`. We again use the UK party support data to illustrate this. We list five control variables in *indepvars* and an additional shock variable in `shockvars( )`: the percentage of those who view Labour as the best manager of the most important issue. Labour, Conservatives, and Liberal Democrats are the three dependent variable categories listed in the required `dvs( )` option. In our dynamic simulations we induce a one-standard-deviation increase (+0.054) in the percentage of those who think Labour is the best manager of the most important issue at time $t = 5$.

Given the three dependent variable categories, `dynsimpie` automatically performs log-ratio transformations and creates two log-ratio compositions: $ln(\frac{Lib\ Dems}{Conservative})$ and $ln(\frac{Labour}{Conservative})$.[4] The seemingly unrelated regression results, produced by default when the option `killtable` is not specified, is shown in Table 1. The coefficients in Table 1 are plotted side by side across dependent variable categories.

```
dynsimpie  all_pidW all_LabLeaderEval_W all_ConLeaderEval_W  ///
all_LDLeaderEval_W all_nat_retW, dvs(Con Ldm Lab)            ///
shockvals(0.054) shockvars(all_b_mii_lab_pct) sigs(95)  ecm

cfbplot
```

We also present the predicted probabilities, shown in Figure 3, generated using the `cfbplot` post-estimation command. The figure shows both short- and long-run effects of the counterfactual shock on the expected proportion of party support in the UK across the simulated time period.

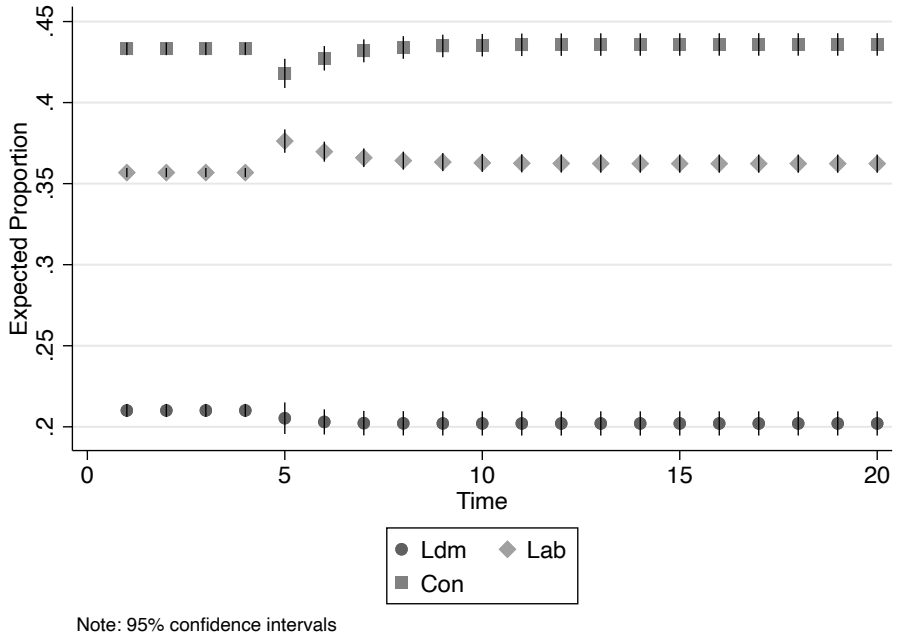
[4] While simulation results are the same regardless of the order in which dependent variable categories are entered into `dvs( )`, it should be noted that the first category entered will be chosen by the program as the baseline category.

Table 1: Table of Results from `dynsimpie` Output

Variable	$ln(\frac{Lib\ Dems}{Conservative})$		$ln(\frac{Labour}{Conservative})$	
Lagged Dependent Variable	-0.58*	(0.10)	-0.52*	(0.09)
ΔParty ID$_t$	-2.24*	(1.07)	-0.37	(0.62)
ΔLabour Leader Eval.$_t$	-0.03	(0.09)	0.24*	(0.05)
ΔConservative Leader Evaluation$_t$	-0.28*	(0.06)	-0.15*	(0.03)
ΔLib. Dem. Leader Evaluation$_t$	0.48*	(0.08)	0.01	(0.04)
ΔNatl. Retrospective Evaluation$_t$	0.15	(0.14)	0.08	(0.08)
ΔLabour "Best Manager"$_t$	0.27	(0.57)	1.66*	(0.31)
Party ID$_{t-1}$	1.25	(1.02)	1.73*	(0.62)
Labour Leader Eval.$_{t-1}$	-0.05	(0.07)	0.17*	(0.05)
Conservative Leader Eval.$_{t-1}$	-0.09	(0.07)	-0.01	(0.04)
Lib. Dem. Leader Eval.$_{t-1}$	0.18*	(0.05)	0.02	(0.02)
Natl. Retrospective Eval.$_{t-1}$	0.10	(0.05)	0.09*	(0.03)
Labour "Best Manager"$_{t-1}$	-0.49	(0.43)	0.09	(0.23)
Constant	-1.07	(0.63)	-1.54*	(0.37)

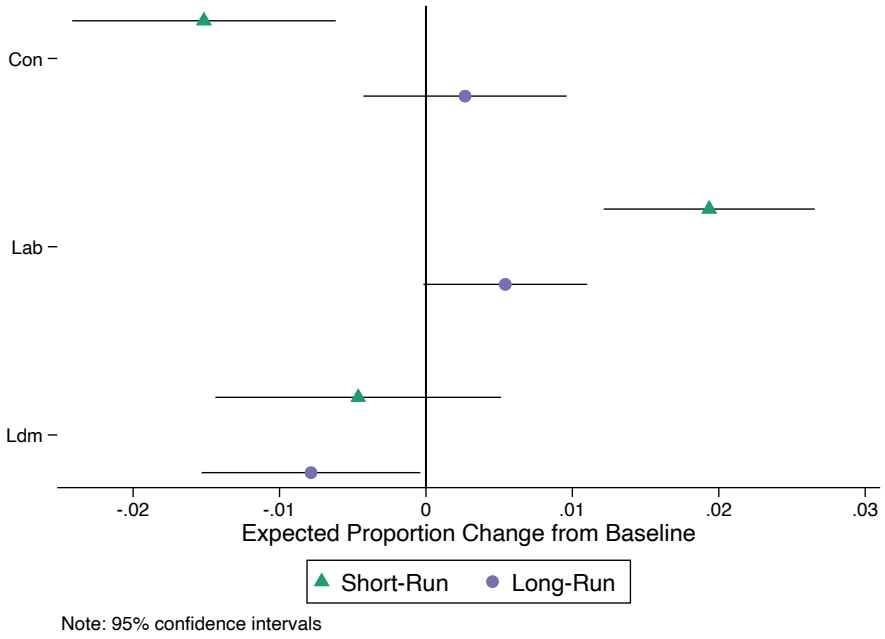
Note: Coefficients from a seemingly unrelated regression with standard errors in parentheses. Two-tailed test statistics. * $p < .05$.

Figure 3: The simulated effect of a 1-SD increase in the percentage of those who think Labour is the best manager of the most important issue



We now use another post-estimation command, `effectsplot`, to create and effects plot (prediction plots) from the estimated `dynsimpie` results. This post-estimation command allows users to easily produce plots which display the long- and short-run effects from the specified shock. The resulting effects plot is shown in Figure 4 below.

Figure 4: The `effectsplot` simulated effect of a 1-SD increase in the percentage of those who think Labour is the best manager of the most important issue



As discussed above, the change-from-baseline and effects plots are two different ways of viewing the estimated effects of any shock. In either figure, it can be seen that Labour received a two percentage point boost in the short run in response to the shock. This comes mainly at the expense of Conservative support. Nevertheless, over the long run it is in fact the Liberal Democrats that experience a drop in support, whereas support for Conservatives returns to its starting value. Labour support diminishes and eventually settles just above its starting value. This long-run effect on support for Labour, however, turns out to be statistically insignificant at the 95% confidence level.

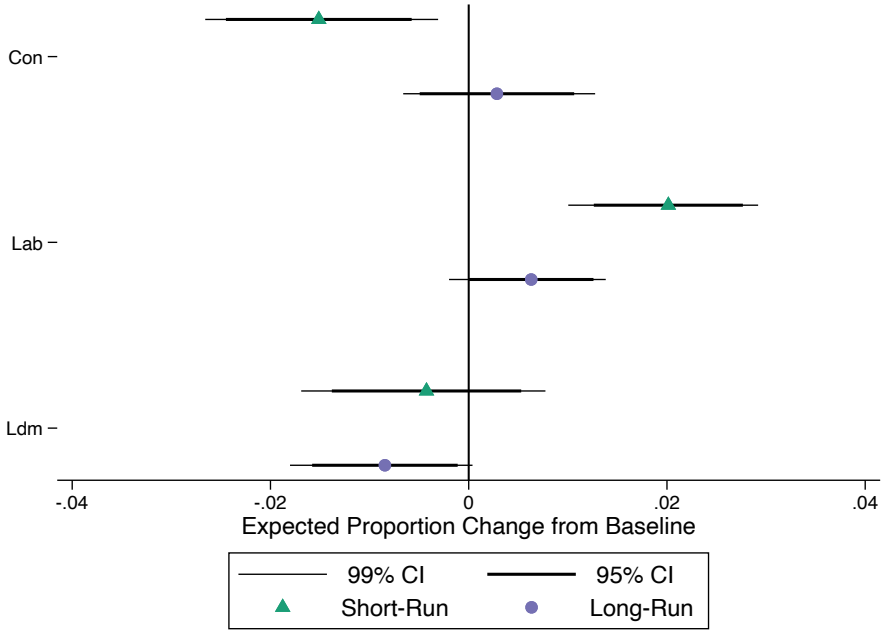
Additional confidence intervals can now be added to provide more information about the statistical and substantive significance of the simulation results. In the following example, we specify

`sigs(95 99)` for both 95% and 99% confidence intervals. This is shown in Figure 5.

```
dynsimpie all_pidW  all_LabLeaderEval_W all_ConLeaderEval_W   ///
all_LDLeaderEval_W  all_nat_retW,  dvs(Con Ldm Lab)           ///
shockvals(0.054) shockvars(all_b_mii_lab_pct)  sigs(95 99)  ecm

effectsplot
```

Figure 5: The simulated effect of a 1-SD increase in the percentage of those who think Labour is the best manager of the most important issue, created using the `sigs(95 99)` option



We will now briefly describe the other available options. For more detailed information, please refer to Philips, Rutherford and Whitten (2016*a*,*b*). The `time( )` option changes the time in the simulated scenario at which the independent variable receives a shock, while the `range( )` option specifies the length of the scenario to simulate. Users can add one or more dummy variables to their model. For example, users may include a dummy variable that is equal to 1 during the months of the Great Recession using the `dummy( )` option. By default, any dummy variables are set to zero in the simulations. Users can set this variable to 1 (or other values) using the option `dummyset( )`.

Users may include up to three continuous shocks to occur at the same point in time using the `shockvars()` and `shockvals()` options. Also, users may include a dummy shock variable that experiences some counterfactual one-period shock of 1 at time t using `dummyshockvar()`. Dummy shock variables are set to 0 before the shock occurs. Since this is within an error correction framework, the shock first affects the first difference of `dummyshockvar` at time t for one period, then will move into the lagged `dummyshockvar` for the rest of the simulation.

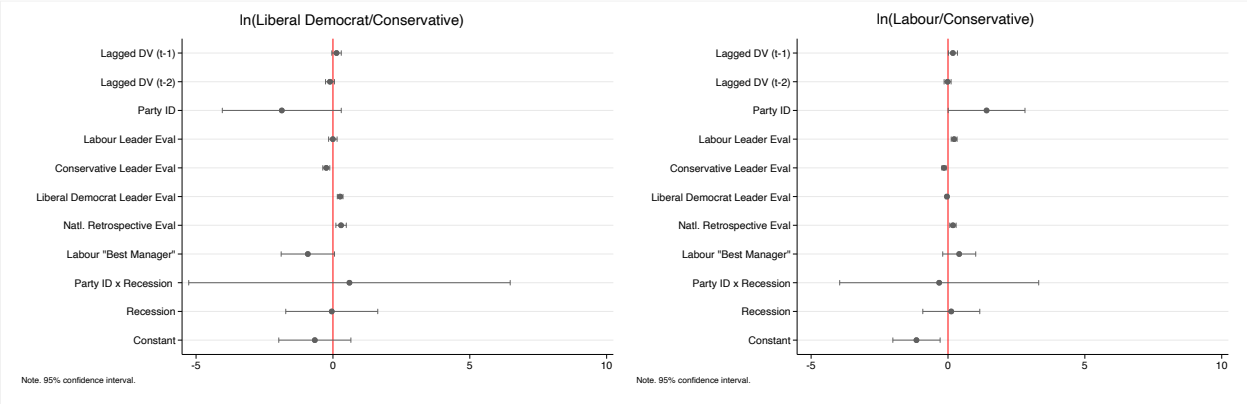
Users wishing to analyze both fundamental and estimation uncertainty can do so by generating predicted values with the `pv` option. By default, *Clarify* simulations produce expected values. These average out fundamental variability, keeping only estimation uncertainty. Users may also specify the option `percentage` to produce plot values in percentages, instead of proportions.

5 Example: `dynsimpiecoef`

We use `dynsimpiecoef` to create coefficient plots of the SUR results employed in `dynsimpie`. `dynsimpiecoef` displays two sets of coefficient plots for each log-compositional ratio as shown in the tabular plot presented in Table 1. By default, `dynsimpiecoef` also produces a table of estimates. The `killtable` option suppresses the automatic generation of the SUR results, just as in `dynsimpie`. Keep in mind that the option `ecm` can be used to specify an error correction model instead of the default lagged dependent variable model. When `ecm` is omitted, users have to specify the option `ar`(*numlist*)—the consecutive or nonconsecutive lags of the dependent variable to be included among the explanatory variables. Furthermore, when users specify `interaction`(*varname*), a multiplicative interaction is produced between this variable and the first variable listed among *indepvars*. Figure 6 below depicts an LDV(2) model with a multiplicative interaction between a recession dummy and the first variable listed in *indepvars*.

```
dynsimpiecoef all_pidW all_LabLeaderEval_W all_ConLeaderEval_W   ///
all_LDLeaderEval_W all_nat_retW all_b_mii_lab_pct,               ///
dvs(Con Ldm Lab) ar(1 2) interaction(recession_dum) xsize(8)
```

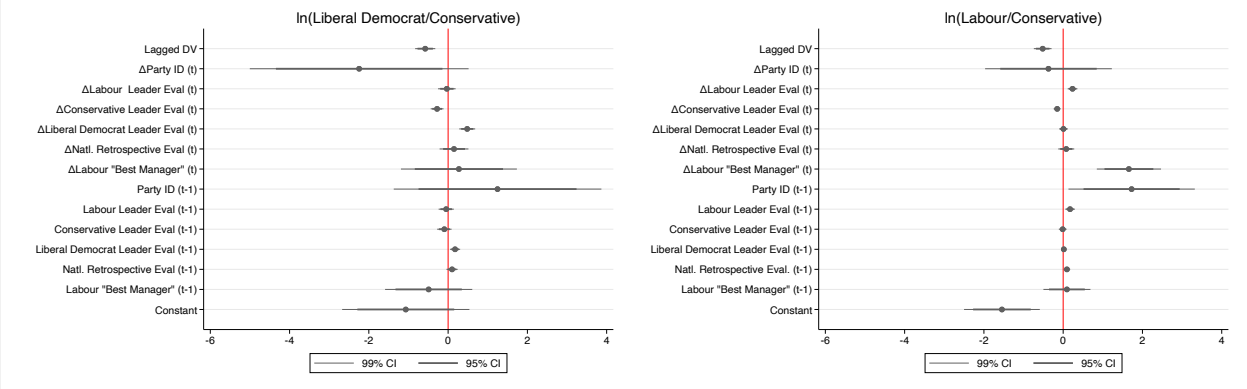
Figure 6: Coefficient plots from the lagged-dependent-variable `dynsimpiecoef` results with two lags of the dependent variable and a multiplicative interaction



In Figure 7, similar graphs are produced but now using the `ecm` and `sigs(`*`numlist`*`)` options. Additionally, we use `xsize(#)` to specify the width of the combined graph in inches. Depending on the number of categories of the dependent variable and the number of independent variables in the model, it might be desirable to make the combined graph wider than usual using this option. The default is `xsize(5)`.

```
dynsimpiecoef all_pidW all_LabLeaderEval_W all_ConLeaderEval_W   ///
all_LDLeaderEval_W all_nat_retW all_b_mii_lab_pct,               ///
dvs(Con Ldm Lab) sigs(95 99) xsize(8) ecm
```

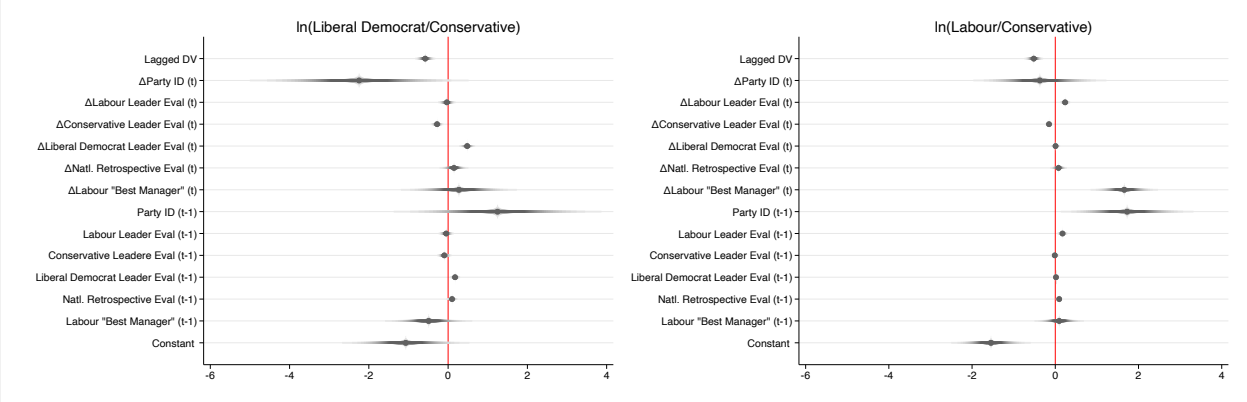
Figure 7: Coefficient plots from the `dynsimpiecoef` results, created using the `sigs(95 99)` option



While the plots in Figure 7 are generated using specific confidence intervals, users may wish to produce confidence intervals using a wide range of confidence levels. With the `smooth` option, users can add "50 equally spaced levels (1,3, ..., 99) with graduated color intensities and varying line widths" (Jann, 2014, p.729). This is shown in Figure 8.

```
dynsimpiecoef all_pidW all_LabLeaderEval_W all_ConLeaderEval_W ///
all_LDLeaderEval_W all_nat_retW all_b_mii_lab_pct,             ///
dvs(Con Ldm Lab)  smooth xsize(8) ecm
```

Figure 8: Coefficient plots from the `dynsimpiecoef` results, created using the `smooth` option

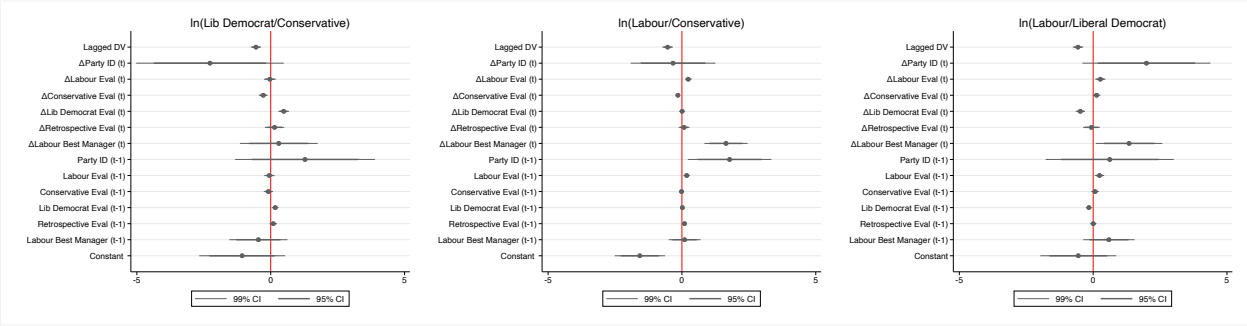


With the `all` option, the `dynsimpiecoef` command will produce coefficient plots for all possible pairs of dependent variable categories, regardless of the baseline category. Thus, the order of dependent variable categories specified in `dvs()` does not matter in `dynsimpiecoef`. An example of the output when using this option is shown in Figure 9.

```
dynsimpiecoef all_pidW all_LabLeaderEval_W all_ConLeaderEval_W  ///
all_LDLeaderEval_W all_nat_retW all_b_mii_lab_pct,              ///
dvs(Con Ldm Lab) sigs(95 99) xsize(10) all ecm
```

While the coefficient plots of each model are useful for judging the significance of coefficients, the `dynsimpiecoef` command can also create coefficient plots for a particular independent

Figure 9: Coefficient plots from the `dynsimpiecoef` results, created using the `sigs(95 99)`, and `all` options

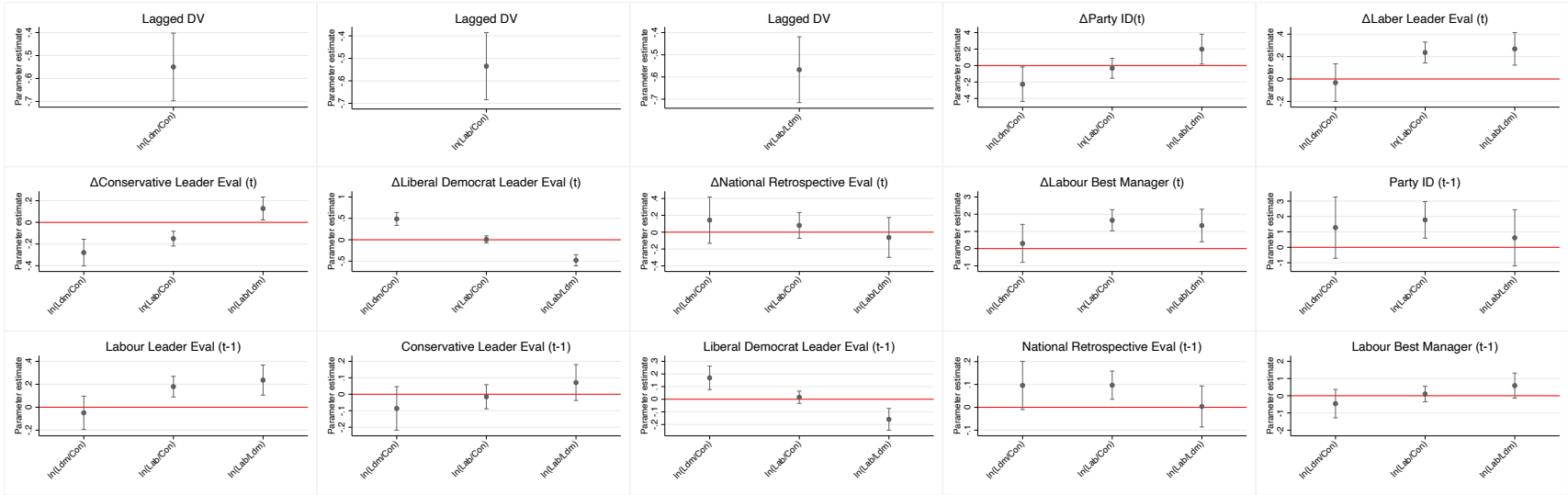


variable across all possible pairwise compositions of dependent variable categories. Using the `vertical` option, users can easily compare the impact of each covariate in the model across dependent variable categories. While this option produces a collection of coefficient plots for each independent variable, users can find coefficient plots for such variables saved automatically and separately in the working directory.[5] Using the `angle(#)` option, users can specify the angle for the labels on the *x*-axis in the combined graph. This option must be specified with the `vertical` option. The default is `angle(90)`, for 90 degrees from the *x*-axis. This is shown in Figure 10. Additionally, the `row(#)` option allows users to specify the number of rows of the combined graph. Depending on the number of categories of the dependent variable and independent variables in the model, a combined graph with multiple rows might be desirable. By default, `dynsimpiecoef` displays the resulting graph in one row when the `vertical` option is not specified. When the `vertical` option is specified, `dynsimpiecoef` displays the resulting graph in three rows, as shown in Figure 10.

```
dynsimpiecoef all_pidW all_LabLeaderEval_W all_ConLeaderEval_W  ///
all_LDLeaderEval_W all_nat_retW all_b_mii_lab_pct,              ///
dvs(Con Ldm Lab)  xsize(8) vertical angel(45) ecm
```

[5] `allg` is prefixed to those graphs.

Figure 10: Coefficient plots from the `dynsimpiecoef` results, created using the `vertical` and `angle(45)` options



6 Conclusion

We have substantially enhanced the `dynsimpie` suite of commands for Stata. With it, users will be able to estimate and interpret a wide range of dynamic models of compositional dependent variables. In this update, we incorporate considerable flexibility in both model specification choices and graphical visualization of results.

References

Aitchison, John. 1982. "The statistical analysis of compositional data." *Journal of the Royal Statistical Society: Series B (Methodological)* 44(2):139–160.

Jann, Ben. 2014. "Plotting regression coefficients and other estimates." *The Stata Journal* 14(4):708–737.

Philips, Andrew Q, Amanda Rutherford and Guy D Whitten. 2015. "The dynamic battle for pieces of pie—Modeling party support in multi-party nations." *Electoral Studies* 39:264–274.

Philips, Andrew Q, Amanda Rutherford and Guy D Whitten. 2016*a*. "Dynamic Pie: A Strategy for Modeling Trade-Offs in Compositional Variables over Time." *American Journal of Political Science* 60(1):268–283.

Philips, Andrew Q, Amanda Rutherford and Guy D Whitten. 2016*b*. "dynsimpie: A program to examine dynamic compositional dependent variables." *Stata Journal* 16(3):662–77.

Philips, Andrew Q, Flávio DS Souza and Guy D Whitten. 2019. "Globalization and comparative compositional inequality." *Political Science Research and Methods* pp. 1–17.

Tomz, Michael, Jason Whittenberg and Gary King. 2003. "Clarify: Software for Interpretng and Presenting Statistical Results." *Journal of Statistical Software* 8(1):1–30.