The Stata Journal (2018)
18, Number 4, pp. 902–923

Cointegration testing and dynamic simulations of autoregressive distributed lag models

Soren Jordan
Department of Political Science
Auburn University
Auburn, AL
sorenjordanpols@gmail.com

Andrew Q. Philips
Department of Political Science
University of Colorado Boulder
Boulder, CO
andrew.philips@colorado.edu

Abstract. In this article, we introduce `dynamac`, a suite of commands designed to assist users in modeling and visualizing the effects of autoregressive distributed lag models and in testing for cointegration. We discuss the bounds cointegration test proposed by Pesaran, Shin, and Smith (2001, *Journal of Applied Econometrics* 16: 289–326), which we have adapted into a command. Because the resulting models can be dynamically complex, we follow the advice of Philips (2018, *American Journal of Political Science* 62: 230–244) by introducing a flexible command designed to dynamically simulate and plot a variety of types of autoregressive distributed lag models, including error-correction models.

Keywords: st0545, dynamac, pssbounds, dynardl, cointegration, dynamic modeling, autoregressive distributed lag, error correction

1 Introduction

Time-series models using an autoregressive distributed lag (ARDL) are common in the social sciences. Whether the dependent variable is estimated in levels (for example, $y_t = \alpha_0 + \cdots$) or in first differences (for example, $\Delta y_t = \alpha_0 + \cdots$), these models can test a host of theoretically important theories such as the effect of public opinion on government response (Jennings and John 2009), the effect of domestic and international factors on defense expenditures (Whitten and Williams 2011) or tax rates (Swank and Steinmo 2002), or analyzing dynamic changes in partisan responsiveness over time (Ura and Ellis 2008).

When one uses an error-correction-style ARDL model, it becomes necessary to test for cointegration.[1] Philips (2018) shows that in small samples that are common in the social sciences—typically, when the number of time points is 80 or less—the ARDL bounds test for cointegration proposed by Pesaran, Shin, and Smith (2001) tends to be more conservative (that is, does not conclude cointegration when it does not exist) than either the popular Engle–Granger "two-step" approach (Engle and Granger 1987) or the Johansen (1991, 1995) approach to cointegration testing. However, there is no standard implementation of this cointegration test in common statistical software.

1. In general, error-correction models regress the first difference of the dependent variable on a constant, its own lag in levels, and the contemporaneous first difference and lagged levels of each of the independent variables. For instance, in a model with a single independent variable **x**, we might estimate $\Delta y_t = \alpha_0 + \theta_0 y_{t-1} + \theta_1 x_{t-1} + \beta_1 \Delta x_t$.

In addition to its error-correction form, ARDL models in general may have complex dynamic specifications, including multiple lags, first differences, and lagged first differences. This makes it more difficult to interpret the effects of changes—especially short- and long-run changes—in the independent variable. To mitigate this, we introduce a flexible command that allows users to dynamically simulate a variety of ARDL models, including the error-correction model. Dynamic simulations offer an alternative to hypothesis testing of model coefficients by instead conveying the substantive significance of the results through meaningful counterfactual scenarios. Such an approach has been gaining popularity in the social sciences (for example, Tomz, Wittenberg, and King [2003]; Choirat et al. [2018]; Williams and Whitten [2011, 2012]; Philips, Rutherford, and Whitten [2016a]).

Below, we briefly discuss the ARDL-bounds approach to cointegration testing. We then present `dynamac`,[2] a suite of two commands for dynamic ARDL modeling and cointegration testing. `dynamac` includes `pssbounds`, which provides the necessary critical values for the Pesaran, Shin, and Smith (2001) cointegration test, and `dynardl`, which produces dynamic simulations of a multitude of ARDL-style models.

2 The ARDL-bounds cointegration test

The concept of cointegration has been around for several decades. To understand cointegration, we briefly discuss integrated versus stationary series. Time series may have "full memory" such that current realizations are fully a function of all previous stochastic shocks, plus some new innovation. Such series are said to be integrated of order one [or $I(1)$], a form of nonstationarity.[3] For instance, the series in (1) is $I(1)$ because values at time t are a function of the prior value of y at time $t-1$, plus innovation ϵ_t.[4]

$$y_t = y_{t-1} + \epsilon_t \tag{1}$$

Most of the time, $I(1)$ series cannot be included in standard regression models, because the nature of these data makes it more likely that we find statistically significant relationships simply because of random chance alone; this is often referred to as the "spurious regression problem in time series" (Yule 1926; Grant and Lebo 2016). Despite this, two or more $I(1)$ series may still have short-run, long-run, or equilibrium relationships between one another. That is to say, while short-run perturbations may move the series apart, over time this disequilibrium is corrected because the series move back toward a stable long-run relationship. Such series are said to be cointegrating.

Because not all relationships between $I(1)$ series are cointegrating, it becomes necessary to test for cointegration. The earliest test comes from Engle and Granger (1987), who show that cointegration between k $I(1)$ regressors, $x_{1t}, x_{2t}, \ldots, x_{kt}$, and an $I(1)$

2. Users can find and download the most up-to-date version of this package at http://andyphilips.github.io/dynamac/.
3. Another term for an $I(1)$ series is that it contains a unit root. Series that are stationary are said to be $I(0)$.
4. This is commonly rewritten as $(y_t - y_{t-1}) = \Delta y_t = \epsilon_t$.

regressand, y_t, exists if the resulting residuals—from a regression of these variables entering into the equation in levels—are stationary:

$$y_t = \kappa_0 + \kappa_1 x_{1t} + \kappa_2 x_{2t} + \cdots + \kappa_k x_{kt} + z_t$$

Several other cointegration tests have since been proposed. Philips (2018) offers an in-depth discussion of how to apply the Pesaran, Shin, and Smith (2001) ARDL-bounds test for cointegration. Others include tests by Johansen (1991) and Phillips and Ouliaris (1990). However, the ARDL-bounds test offers several advantages. Chief among them is that users do not have to make the sharp $I(0)/I(1)$ distinction for the regressors.[5] Below, we briefly summarize this approach.[6]

First, the analyst must ensure that the dependent variable is $I(1)$. Many unit-root tests can be used to determine the order of integration of a series, including the Dickey–Fuller, Phillips–Perron, Elliott–Rothenberg–Stock, and Kwiatkowski–Phillips–Schmidt–Shin tests. Only an $I(1)$ dependent variable is a potential candidate for cointegration.

Second, the analyst must ensure that the regressors are not of integration higher than $I(1)$. While this means that the analyst does not have to make the potentially difficult $I(0)/I(1)$ decision, he or she must ensure that all regressors are not explosive or contain seasonal unit roots.

Third, the analyst fits an ARDL model in error-correction form. The model is

$$\begin{aligned}\Delta y_t = \alpha_0 + \theta_0 y_{t-1} + \theta_1 x_{1,t-1} + \cdots + \theta_k x_{k,t-1} + \sum_{i=1}^{p} \alpha_i \Delta y_{t-1} \\ + \sum_{j=0}^{q_1} \beta_{1j} \Delta x_{1,t-j} + \cdots + \sum_{j=0}^{q_k} \beta_{kj} \Delta x_{k,t-j} + \epsilon_t \end{aligned} \tag{2}$$

where the change in the dependent variable is a function of a constant, its value at $t-1$ (appearing in levels), values at $t-1$ of all regressors appearing in levels, and up to p and q_k lags of the first difference of the dependent variable and regressors, respectively. These may enter into (2) for theoretical reasons but also have the added benefit of helping to ensure white-noise residuals. While it is ideal to have well-behaved residuals in all models, this is a crucial step before running the ARDL-bounds test for cointegration. Information criteria such as Schwarz's Bayesian information criterion, the Akaike information criterion, and autocorrelation and heteroskedasticity tests (for example, Breusch–Godfrey, Durbin's Alternative, Cook–Weisberg, or Cumby–Huizinga tests) can be used to check for white-noise residuals. Many of these are available as canned Stata procedures.

5. However, users must ensure that regressors are not of order $I(2)$ or more and that seasonality has been removed from the series.
6. A more in-depth discussion can be found in Philips (2018).

While the ARDL-bounds test may be relatively easy to implement, the test uses special critical values. These critical values are available but cumbersome for users to map onto the test; Pesaran, Shin, and Smith (2001) provide asymptotic critical values, and Narayan (2005) provides finite sample critical values. To make the bounds test more accessible to users, we introduce `pssbounds` below, which provides the necessary critical values through an easy-to-use command.

2.1 The pssbounds command

Syntax

```
pssbounds, observations(#) fstat(#) k(#) case(#) [tstat(#)]
```

Options

`observations(#)` is the number of observations from the fitted ARDL model in error-correction form. `observations()` is required.[7]

`fstat(#)` is the value of the F statistic from the test that all parameters on the regressors appearing in levels plus the coefficient on the lagged dependent variable are jointly equal to 0: $H_0 = \theta_0 + \theta_1 + \cdots + \theta_k = 0$. After running the ARDL model in error-correction form, users should use Stata's `test` command to obtain the F statistic.[8] `fstat()` is required.

`k(#)` is the number of regressors, k, modeled in levels in the fitted ARDL model not including the lagged dependent variable.[9] The bounds F test is a test that the k parameters on the regressors appearing in levels (plus the coefficient on the lagged dependent variable, θ_0) are jointly equal to 0: $H_0 = \theta_0 + \theta_1 + \cdots + \theta_k = 0$. `k()` is required because critical values associated with the test differ based on the number of regressors.

`case(#)` identifies the type of case of the restrictions on the intercept, trend term, or both. The case type can be given in Roman numerals (`I`, `II`, `III`, `IV`, `V`) or numerically (`1`, `2`, `3`, `4`, `5`). Because the critical values of the bounds test depend on the assumptions placed on the intercept and trend, this option is required. See Pesaran, Shin, and Smith (2001) for more details; case 3 is the most common. The following five cases are possible:

7. As discussed below, if first using `dynardl` in error-correction form, users can simply run `pssbounds` afterward without having to specify the required options, because the latter command obtains the necessary stored values from the former.
8. For example, `test l.y l.x1 l.x2 ... = 0`.
9. The lagged dependent variable is not included in the count of `k` even though it is included in the restriction tested, because the lagged dependent variable is always present in the ARDL-bounds procedure. This is why special critical values are required. For instance, if the ARDL model was $\Delta y_t = \beta_0 - \theta_0 y_{t-1} + \beta_1 \Delta x_{1t} + \theta_1 x_{1,t-1} + \beta_2 \Delta x_{2t} + \theta_3 x_{2,t-1}$, then $k = 2$ because $x_{1,t-1}$ and $x_{2,t-1}$ appear in lagged levels.

- Case 1: No intercept and no trend, `case(1)`.
- Case 2: Restricted intercept and no trend, `case(2)`.
- Case 3: Unrestricted intercept and no trend, `case(3)`.
- Case 4: Unrestricted intercept and restricted trend, `case(4)`.
- Case 5: Unrestricted intercept and unrestricted trend, `case(5)`.

`tstat(#)` is the value of the t statistic for the coefficient on the lagged dependent variable. This serves as a one-sided auxiliary test to the bounds F test and should supplement—but not replace—the conclusions offered by the F statistic test above. If the tests are in conflict or are inconclusive, users should follow the full procedure outlined in Philips (2018, 235). Only asymptotic critical values are available for the t statistic, so this option is not required. Note that critical values do not currently exist for this test for cases 2 and 4 (see below).

Examples

Two examples of `pssbounds` are shown below. For the first example, we will use the Lutkepohl West German quarterly macroeconomic dataset available in Stata:

```
. webuse lutkepohl2
(Quarterly SA West German macro data, Bil DM, from Lutkepohl 1993 Table E.1)
. tsset
        time variable:  qtr, 1960q1 to 1982q4
                delta:  1 quarter
```

Following Philips (2018), the first step is to assess whether any of the three variables—log investment, log income, and log consumption—contain a unit root. Both the Phillips–Perron and Dickey–Fuller generalized least-squares unit-root tests (`pperron` and `dfgls`, respectively) fail to reject the null hypothesis of a unit root for all series, as shown in table 1:

```
. pperron ln_inv, lags(3)
. dfgls ln_inv, maxlag(4)

. pperron ln_inc, lags(3)
. dfgls ln_inc, maxlag(4)

. pperron ln_consump, lags(3)
. dfgls ln_consump, maxlag(4)
```

Table 1. Unit-root tests indicate $I(1)$ series

	ln(Investment)	ln(Income)	ln(Consumption)
Phillips–Perron	−1.25	−2.01	−1.54
Dickey–Fuller generalized least-squares	−2.05	−1.29	−1.60

NOTE: Three augmenting lags are included for all tests.
* $p < 0.05$

Next, we fit the following ARDL-bounds model in error-correction form:[10]

$$\Delta\ln(\text{Investment})_t = \alpha_0 + \theta_0\ln(\text{Investment})_{t-1} + \beta_1\Delta\ln(\text{Income})_t + \theta_1\ln(\text{Income})_{t-1} + \beta_2\Delta\ln(\text{Consumption})_t + \theta_2\ln(\text{Consumption})_{t-1}$$

In other words, we believe that investment is a function of income and consumption and that there is a cointegrating relationship between investment and the two regressors. The results are shown in model 1 in table 2.

```
. regress d.ln_inv l.ln_inv d.ln_inc l.ln_inc d.ln_consump l.ln_consump
```

10. Estimation is not the first step of the ARDL-bounds approach; we would need to conduct a unit-root test on the dependent variable and ensure that the independent variables were $I(1)$ or less (Philips 2018). We would also need to ensure that the resulting residuals from the model are white noise. This is simply a stylized example used to showcase the command using readily available Stata datasets.

Table 2. Lutkepohl example

	(1) Δln_inv$_t$	(2) Δln_inv$_t$
ln_inv$_{t-1}$	−0.140*	−0.152**
	(0.059)	(0.056)
Δln_inc$_t$	−0.201	−0.0985
	(0.459)	(0.423)
ln_inc$_{t-1}$	−0.209	
	(0.334)	
Δln_consump$_t$	1.548**	1.395**
	(0.538)	(0.459)
ln_consump$_{t-1}$	0.336	0.131**
	(0.333)	(0.048)
Constant	−0.008	
	(0.071)	
N	91	91
R^2	0.17	0.27

NOTE: The dependent variable is Δln_inv$_t$, and standard errors are in parentheses.
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

We then run an F test that the coefficients on the variables appearing in lagged levels (that is, ln_inv$_{t-1}$, ln_inc$_{t-1}$, and ln_consump$_{t-1}$) are jointly equal to 0:

```
. test l.ln_inv l.ln_inc l.ln_consump
 ( 1)  L.ln_inv = 0
 ( 2)  L.ln_inc = 0
 ( 3)  L.ln_consump = 0
       F(  3,    85) =    2.60
            Prob > F =    0.0573
```

Recall that the critical values are nonstandard, so we need only the value of the F statistic, which is 2.60. To test for cointegration, we use `pssbounds`. For the required option `fstat()`, we input the F statistic from above. From the fitted model, we tell `pssbounds` that the number of observations is `91`, that the case is case `3` (that is, unrestricted intercept with no trend, as shown in model 1), and that two regressors appear in levels (`k(2)`). The resulting output appears as follows:

```
. pssbounds, fstat(2.60) observations(91) case(3) k(2)
PESARAN, SHIN AND SMITH (2001) COINTEGRATION TEST
Obs: 91
No. Regressors (k): 2
Case: 3
─────────────────────────────────────────────────────────────
                              F-test
─────────────────────────────────────────────────────────────
                   <──── I(0) ──────────── I(1) ────>
10% critical value       3.170           4.140
5% critical value        3.790           4.850
1% critical value        5.150           6.360
F-stat. =     2.600
─────────────────────────────────────────────────────────────
F-statistic note: Asymptotic critical values used.
```

For this model, because the F statistic of 2.60 is below the $I(0)$ critical value—even at the 10% level—we can conclude that there is no cointegration and that all regressors appearing in levels are stationary. As discussed in Philips (2018), we would next want to exclude one of the regressors from appearing in lagged levels—thus removing it from the potentially cointegrating equation—to see whether either `ln_inc` or `ln_consump` on its own was in a cointegrating relationship with `ln_inv`.

Purely for illustrative purposes, let us now assume that we wanted to fit a model without a constant. These results are shown in model 2 in table 2. As with model 1, we next run an F test of all variables appearing in levels. For this example, note that the model also has `ln_inc` appearing in first differences but not in levels; thus, $k = 1$:

```
. regress d.ln_inv l.ln_inv d.ln_inc d.ln_consump l.ln_consump, noconstant
  (output omitted)
. test l.ln_inv l.ln_consump
 ( 1)  L.ln_inv = 0
 ( 2)  L.ln_consump = 0
       F(  2,    87) =    3.83
            Prob > F =    0.0254
```

We can account for a restricted constant by specifying the `case(1)` option (that is, no intercept and no trend). As an additional option, we add `tstat(-2.73)`, which is the t statistic on the coefficient on the lagged dependent variable.

```
. pssbounds, fstat(3.83) observations(91) case(1) k(1) tstat(-2.73)
PESARAN, SHIN AND SMITH (2001) COINTEGRATION TEST
Obs: 91
No. Regressors (k): 1
Case: 1
─────────────────────────────────────────────────────────────────
                              F-test
─────────────────────────────────────────────────────────────────
                   <──── I(0) ──────────── I(1) ────>
10% critical value       2.440           3.280
5% critical value        3.150           4.110
1% critical value        4.810           6.020
F-stat. =     3.830
─────────────────────────────────────────────────────────────────
                              t-test
─────────────────────────────────────────────────────────────────
                   <──── I(0) ──────────── I(1) ────>
10% critical value      -1.620          -2.280
5% critical value       -1.950          -2.600
1% critical value       -2.580          -3.220
t-stat. =    -2.730
─────────────────────────────────────────────────────────────────
F-statistic note: Asymptotic critical values used.
t-statistic note: Asymptotic critical values used.
```

The output of `pssbounds` now contains critical values for both the F test and one-sided t test. Based on the F statistic, we can conclude cointegration at the 10% level because the F statistic of 3.83 is above the $I(1)$ critical threshold of 3.28. However, there is not strong enough evidence to support cointegration at the five percent level. For the t test, the t statistic of -2.73 falls below the critical $I(1)$ threshold of -2.60, supporting the earlier conclusion of cointegration. Also note that for both tests, `pssbounds` issued a warning that asymptotic critical values are used. For all cases, only asymptotic critical values from Pesaran, Shin, and Smith (2001) are provided for the t statistic test.[11] Thus, interpreting the results of this test should be done with caution in small samples. Small-sample critical values for the F statistic are not available for case 1. Recall, though, that `pssbounds` implements only the test; when the results are inconclusive, users are encouraged to follow the process outlined in Philips (2018).

As a second example, we use data from Ura (2014), who examines public mood liberalism in the United States. Ura argues that in the short run, there will be a public "backlash" in response to liberal Supreme Court decisions, but that in the long run, the sentiments of the public tend to follow closely to those of the Court. Using the same dataset, Philips (2018, see supplemental materials page 110) finds that the dependent variable, public mood liberalism, is $I(1)$ and that Ura's ARDL model fit in error-correction form with an additional lagged first difference of unemployment produces the white-noise residuals needed to conduct the ARDL-bounds test. The model is shown in table 3.

11. Nor do critical values for the t statistic test exist for cases 2 and 4.

```
. use "supreme court mood replication.dta", clear
. tsset
        time variable:  year, 1955 to 2009
                delta:  1 unit
. regress d.mood l.mood d.policy l.policy d.unemployment dl.unemployment
> l.unemployment d.inflation l.inflation d.caselaw l.caselaw
```

(*output omitted*)

Table 3. Ura (2014) example

	(1) $\Delta\texttt{mood}_t$
$\texttt{mood}_{t-1}$	-0.241^{**}
	(0.076)
$\Delta\texttt{policy}_t$	0.051
	(0.069)
$\texttt{policy}_{t-1}$	-0.073^{***}
	(0.020)
$\Delta\texttt{unemployment}_t$	-0.106
	(0.265)
$\Delta\texttt{unemployment}_{t-1}$	-0.538^{*}
	(0.242)
$\texttt{unemployment}_{t-1}$	-0.024
	(0.198)
$\Delta\texttt{inflation}_t$	-0.306^{*}
	(0.123)
$\texttt{inflation}_{t-1}$	-0.299^{*}
	(0.120)
$\Delta\texttt{caselaw}_t$	-0.093^{*}
	(0.037)
$\texttt{caselaw}_{t-1}$	0.027^{*}
	(0.011)
Constant	15.65^{**}
	(5.050)
N	53
R^2	0.47

NOTE: The dependent variable is $\Delta\texttt{mood}_t$, and standard errors in parentheses.
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Next, we run an F test that all variables appearing in lagged levels ($\texttt{mood}_{t-1}$, $\texttt{policy}_{t-1}$, $\texttt{unemployment}_{t-1}$, $\texttt{inflation}_{t-1}$, and $\texttt{caselaw}_{t-1}$) are jointly equal to 0:

```
. test l.mood l.policy l.unemployment l.inflation l.caselaw
 ( 1)  L.mood = 0
 ( 2)  L.policy = 0
 ( 3)  L.unemployment = 0
 ( 4)  L.inflation = 0
 ( 5)  L.caselaw = 0
       F(  5,    42) =    5.15
            Prob > F =    0.0009
```

We then run `pssbounds` on the resulting F statistic of 5.15, where $k = 4$ (that is, 4 regressors: `policy`, `unemployment`, `inflation`, and `caselaw`) and where the value of the t statistic of the lagged dependent variable is -3.19:

```
. pssbounds, fstat(5.15) observations(53) case(3) k(4) tstat(-3.19)
PESARAN, SHIN AND SMITH (2001) COINTEGRATION TEST
Obs: 53
No. Regressors (k): 4
Case: 3
─────────────────────────────────────────────────────────────────
                              F-test
─────────────────────────────────────────────────────────────────
                   <──── I(0) ────────── I(1) ────>
10% critical value       2.578           3.710
5% critical value        3.068           4.334
1% critical value        4.244           5.726
F-stat. =     5.150
─────────────────────────────────────────────────────────────────
                              t-test
─────────────────────────────────────────────────────────────────
                   <──── I(0) ────────── I(1) ────>
10% critical value      -2.570          -3.660
5% critical value       -2.860          -3.990
1% critical value       -3.430          -4.600
t-stat. =    -3.190
─────────────────────────────────────────────────────────────────
F-statistic note:
t-statistic note: Small-sample critical values not provided
for Case III. Asymptotic critical values used.
```

We can conclude evidence of cointegration at the 5% level for the F test because the F statistic of 5.15 is above the $I(1)$ critical value of 4.334. However, note that we fail to clear the critical $I(1)$ threshold for the ARDL-bounds t test. However, the t-test values of this auxiliary test are asymptotic (`pssbounds` issues a note at the bottom of the output that warns the user of this) and not precisely tailored to small samples. Based on the small-sample F statistics, we have relatively strong evidence of cointegration.

3 Dynamic simulations of ARDL models

ARDL models may have a fairly complex lag structure, with lags, contemporaneous values, first differences, and lagged first differences of the independent (and sometimes the dependent) variable appearing in the model specification. While interpreting short- and long-run effects may be simple in something like an ARDL(1,1) model (that is, one lag of the dependent variable and contemporaneous and one-period lags of all independent variables), understanding the short-, medium-, and long-run effects becomes more difficult as the model specification grows in complexity.

To better interpret the substantive significance of our results, we introduce `dynardl` below, which is a command to dynamically simulate a variety of ARDL models. `dynardl` estimates, simulates, stores the results from, and automatically plots substantively interesting predictions from ARDL models. Users can even run `pssbounds` afterward as a postestimation command if simulating an error-correction model.

The output in `dynardl` helps us visualize the effect of a counterfactual change in one regressor at a single point in time, holding all else equal, using stochastic simulation techniques. Dynamic simulation approaches are gaining in popularity as a straightforward way to show the substantive results of time-series models, whose coefficients often have nonintuitive or "hidden" interpretations (Breunig and Busemeyer 2012; Williams and Whitten 2011; Philips, Rutherford, and Whitten 2016a,b; Gandrud, Williams, and Whitten 2016).[12] Before one uses the command, it is assumed that one has already determined the order of integration of the variables through unit-root testing, diagnosed and addressed other issues such as seasonal unit roots, and used information criteria (and theory) to identify the best-fitting lagged-difference structure, which is used to purge autocorrelation and to ensure the residuals are white noise. If an error-correction model is fit, users should use the ARDL-bounds test to determine whether there is cointegration and adjust the model accordingly if there is not.[13]

`dynardl` first runs a regression using ordinary least squares. Then, using a self-contained procedure similar to the popular `clarify` command for Stata (Tomz, Wittenberg, and King 2003), it takes 1,000 draws (or however many simulations a user desires) of the vector of parameters from a multivariate normal distribution. These distributions are assumed to have means equal to the estimated parameters from the regression. The variance of these distributions is equal to the estimated variance–covariance matrix from the regression. To reintroduce stochastic uncertainty back into the model when creating predicted values, `dynardl` simulates $\widehat{\sigma}^{2*}$ by taking draws from a scaled inverse χ^2 distribution. The distribution is scaled by the residual degrees of freedom $(n-k)$ and the estimated $\widehat{\sigma}^2$ from the regression (Gelman et al. 2013, 37, 582), which ensures that draws of $\widehat{\sigma}^{2*}$ are bounded by zero and one. Simulated parameters and sigma-squared

12. For instance, regressors in a stationary ARDL$(1,0)$ model have both a contemporaneous effect (given by the coefficient on the regressor, β) and a long-run or cumulative effect, given by $\beta/(1-\theta_0)$.
13. If the test is inconclusive, "[e]ach regressor should be tested for a unit root. Only $I(1)$ variables can appear in levels in the error-correction model. Stationary variables may still appear in first differences .... If the resulting statistic is still inconclusive, combinations of variables appearing in levels may need to be tested" (Philips 2018, 13–14). See Philips (2018) for a step-by-step example of this process for the ARDL model in general.

values are then used to create predicted values of the dependent variable over time, $\widehat{Y}_t$, for each of the simulations, by setting all covariates to certain values (typically means). Stochastic uncertainty is introduced into the prediction by taking a draw from a multivariate normal distribution with mean zero and variance $\widehat{\sigma}^{2*}$. The command then averages across the simulations, creating $\widehat{Y}_t^*$ (the predicted values plus stochastic uncertainty) and percentile confidence intervals of the distribution of simulated values at a particular point in time. These are then saved, allowing a user to make a table or (more commonly) a graph of the results over time.

3.1 The dynardl command

Syntax

`dynardl` *depvar indepvars*, `lags(`*numlist*`) shockvar(`*varname*`) shockval(#)`
`[diffs(`*numlist*`) lagdiffs(`*numlist*`) levels(`*numlist*`) ec trend noconstant`
`range(#) sig(#) time(#) saving(`*filename*`) forceset(`*numlist*`) sims(#)`
`burnin(#) [graph | graph rarea | graph change] expectedval]`

Options

`lags(`*numlist*`)` is a numeric list of the number of lags to include for each variable, separated by a comma. The number of desired lags is listed in exactly the same order in which the variables *depvar* and *indepvars* appear. `lags()` is required. For instance, the command

```
dynardl y x1 x2, lags(1, 2, 3) ...
```

would lag `y` by $t-1$, `x1` by $t-2$, and `x2` by $t-3$. Note that the lag on *depvar* (always the first entry in `lags()`) must always be specified. To fit a model without a lag for a particular variable, simply replace the number with a `.`; for instance, if we did not want a lag on the first regressor and wanted a lag of $t-1$ on the second regressor, we type `lags(1, ., 1)`. `dynardl` can accommodate consecutive lags by specifying the minimum lag, a forward slash, and then the maximum lag. For instance, `lags(1/3, ., .)` will introduce lags of y_t at $t-1$, $t-2$, and $t-3$ into the model. The command can also add nonconsecutive lags. For instance, to add a single lag of y_t at $t-1$ and $t-3$, specify `lags(1 3, ., .)`.

`shockvar(`*varname*`)` is a single independent variable from the list of *indepvars* that is to be shocked. It will experience a counterfactual shock of size `shockval(#)` at time `time(#)`. `shockvar()` is required.

`shockval(#)` is the amount to shock `shockvar(`*varname*`)` by. A common shock value is a +/- one standard-deviation shock, although any shock value can be used. `shockval()` is required.

`diffs(`*numlist*`)` is a numeric list of the number of contemporaneous first differences to include for each variable, separated by a comma. Note that the first entry (the placeholder for the *depvar*) will always be empty (denoted by `.`) because the first difference of the dependent variable cannot appear on the right-hand side of the model.[14] Only first differences can be taken using this option; for instance, `diffs(., 1, .)` would first difference only the first regressor in the equation.

`lagdiffs(`*numlist*`)` is a numeric list of the number of lagged first differences to include for each variable, separated by a comma. The syntax is similar to that of `lags()`. For instance, to include a lagged first difference at $t-2$ for *depvar* (that is, $\Delta y_{t-2} = y_{t-2} - y_{t-3}$), a lagged first difference at $t-1$ for the first regressor, and none for the second, specify `lagdiffs(2, 1, .)`. To include an additional lagged first difference for both the first and second lags of *depvar*, specify `lagdiffs(1/2, 1, .)`. Users can also include nonconsecutive lagged first differences.[15]

`levels(`*numlist*`)` is a numeric list of variables to appear in levels (that is, not lagged or differenced but appearing contemporaneously at time t), separated by a comma.[16] For example, `levels(., 1, .)` tells `dynardl` to include the first regressor contemporaneously at time t.

`ec` specifies that *depvar* will be estimated in first differences. If fitting an error-correction model, users will need to use this option.

`trend` specifies that the command will add a deterministic linear trend to the model.

`noconstant` specifies that the constant will be suppressed.

`range(#)` is the length of the scenario to simulate. The default is `range(20)`. Note that the range must be larger than `time()`.

`sig(#)` specifies the significance level for the percentile confidence intervals. The default is `sig(95)`.

`time(#)` is the scenario time in which the shock occurs to `shockvar()`. The default is `time(10)`.

`saving(`*filename*`)` specifies the name of the output file where the means of the predicted values and user-specified confidence intervals will be saved. The default is `saving(dynardl_results.dta)`.

`forceset(`*numlist*`)` allows the user to change the setting of the lagged (or unlagged if using `levels()`) levels of the variables. This could be useful when estimating a dummy variable, for instance, when we wish to see the effect of a movement from zero to one. By default, the command will fit the ARDL model in equilibrium; all lagged variables and variables appearing in levels are set to their sample means. All

14. However, it can appear in lagged first differences, as shown below.
15. For instance, `lagdiffs(1 3/4, ., .)` would add a first difference at $t-1$, $t-3$, and $t-4$ for the dependent variable.
16. If both `levels()` and `ec` are specified, `dynardl` will issue a warning message. Of course, users may have a valid reason to include a variable in levels, for instance, a dummy variable.

first differences and any lagged first differences are set to zero. For instance, to set the value of the first regressor to 5, specify `forceset(., 5, .)`.

`sims(#)` is the number of simulations. The default is `sims(1000)`. If confidence intervals are particularly noisy, it may help to increase this number. Note that you may also need to increase the `matsize` in Stata.

`burnin(#)` allows `dynardl` to run a number of early simulations that are dropped, resulting in more stable starting values. This option is rarely used. However, if using the option `forceset()`, the predicted values will not be in equilibrium at the start of the simulation and will take some time to converge on stable values. To get around this, one can use the `burnin()` option to specify a number of simulations to "throw away" at the start. The default is `burnin(20)`. Setting a burn-in does not change the simulation range or time; to simulate a range of 25 with a shock time at 10 and a burn-in of 30, specify `burnin(30) range(25) time(10)`.

`graph` can be specified to automatically plot the dynamic results using a spike plot. Two alternative plots are possible:

`rarea` creates an area plot. Predicted means and 75%, 90%, and 95% confidence intervals are shown with this option.

`change` shows predicted changes (from the sample mean) across time, starting with the time at which the shock occurs, similar to an impulse–response function.

`expectedval` calculates expected values of the dependent variable such that the average of 1,000 stochastic draws now becomes the estimate of the stochastic component for each of the simulations. This effectively removes stochastic uncertainty introduced in calculating $\widehat{Y}_t^*$ (Tomz, Wittenberg, and King 2003). Predicted values are more conservative than expected values. By default, `dynardl` will calculate predicted values of the dependent variable for a given number of simulations. For every simulation, the predicted value comes from a systematic component—which contains uncertainty surrounding the parameter estimates from the model—and a single draw from the stochastic component. Note that `dynardl` takes longer to run if calculating expected values. We recommend that unless users have a specific theoretical or substantive justification for using expected values, they instead use the default predicted values that account for the random uncertainty surrounding the predictions.

Examples

For the first example, we will once again use the results from model 1 in table 2 using the Lutkepohl data. We fit the following model:

$$\begin{aligned}\Delta \texttt{ln\_inv}_t = {} & \texttt{ln\_inv}_{t-1} + \Delta \texttt{ln\_inc}_t + \texttt{ln\_inc}_{t-1} + \Delta \texttt{ln\_consump}_t \\ & + \texttt{ln\_consump}_{t-1} \end{aligned} \tag{3}$$

Because `dynardl` uses Stata's matrix capabilities, we will also increase the maximum matrix size:

```
. set matsize 5000
. webuse lutkepohl2
(Quarterly SA West German macro data, Bil DM, from Lutkepohl 1993 Table E.1)
. tsset
        time variable:  qtr, 1960q1 to 1982q4
                delta:  1 quarter
```

To fit the model shown in (3) using `dynardl` and see the effect of a -1 shock to `ln_inc` (about two standard deviations), we specify the command below. In (3) we have two regressors and the lagged dependent variable, all of which are lagged one period. Therefore, we specify `lags(1, 1, 1)`. The first differences of all regressors appear, so we add `diff(., 1, 1)`. There are no lagged differences appearing in (3).

```
. dynardl ln_inv ln_inc ln_consump, lags(1, 1, 1) diffs(., 1, 1)
> shockvar(ln_inc) shockval(-1) time(10) range(30) graph ec
```

In the command, `lags(1, 1, 1)` tells `dynardl` to add lags (of $t-1$) for each of the variables, while `diffs(., 1, 1)` means that the second and third variables (`ln_inc` and `ln_consump`) also enter into the model as first differences. In `shockvar()`, we include the variable to be shocked and specify the amount to shock it by using `shockval()`. Additional options include `time(10)` to specify the time at which the shock occurs, `range(30)` to specify the total range of the simulations, and `ec` to include the dependent variable in first differences. Finally, because we specified the `graph` option, `dynardl` will produce a plot, which is shown in figure 1. As is clear from the figure, a -1 shock at $t = 10$ produces a small increase that is not statistically significant in the short run, which eventually increases to a predicted value of about 7.5 over the long run. This increase is statistically significant.

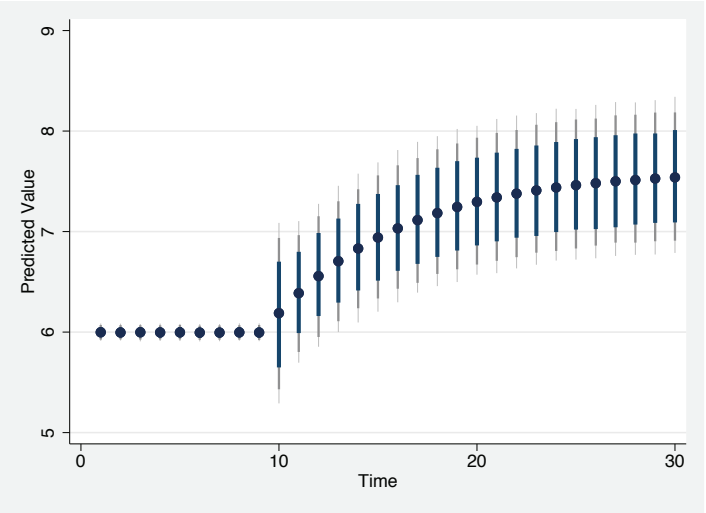

Figure 1. Plot produced from `dynardl` using the `graph` option. Dots show the average predicted value; shaded lines show (from darkest to lightest) the 75%, 90%, and 95% confidence intervals.

In addition to producing figures, `dynardl` also saves the prediction output, which can be used to create more customizable figures (for example, colors, lines, or labels) if users desire. Because we did not specify a filename to save using the `saving()` option, the results are automatically saved as `dynardl_results.dta`.

More complex dynamic specifications are possible using `dynardl`. For instance, perhaps we wanted to estimate the following equation:

$$\begin{aligned}\Delta \texttt{ln\_inv}_t = \texttt{ln\_inv}_{t-1} + \Delta \texttt{ln\_inv}_{t-1} + \texttt{ln\_inc}_{t-1} + \texttt{ln\_inc}_{t-2} + \texttt{ln\_inc}_{t-3} \\ + \Delta \texttt{ln\_consump}_t + \texttt{ln\_consump}_{t-1} \end{aligned} \tag{4}$$

Now the dependent variable appears not only in lagged-level form at $t-1$ but also as a lagged first difference. `ln_inc` no longer appears contemporaneously but at the first, second, and third lags, while `ln_consump` continues to appear in lagged and first-differenced forms. Equation (4) would appear as the following in `dynardl`:

```
. dynardl ln_inv ln_inc ln_consump, lags(1, 1/3, 1) diffs(., ., 1)
> lagdiffs(1, ., .) shockvar(ln_inc) shockval(-1) time(10) range(30)
> graph ec rarea sims(5000)
```

Note that, because the first through third lag of `ln_inc` are desired, we specify `lags(1, 1/3, 1)`.[17] Because there is a lagged first difference included for the dependent variable, we add `lagdiffs(1, ., .)`. The `.` is a placeholder for the other variables and must be included. Finally, we add the `rarea` option to produce an area plot, and

17. Equivalently, we could specify `lags(1, 1 2 3, 1)` instead of `lags(1, 1/3, 1)`.

we increase the number of simulations to 5,000 with `sims(5000)`. The resulting plot is shown in figure 2. As is clear from the figure, as a result of a negative shock to ln(Income) at time $t = 10$, investment decreases over the next few periods, though this does not appear to be statistically significantly lower than the average predicted value (shown when $t < 10$). After about three periods, investment increases in response to the negative income shock, resulting in a new equilibrium prediction just above 10. Figure 2 is appealing because it shows 75%, 90%, and 95% confidence intervals. Because it is an area plot, its continuous style may allow users to see changes slightly easier than the plot style in figure 1. Note that because we increased the number of simulations to 5,000, the confidence intervals in figure 2 are smoother from time point to time point than in figure 1.

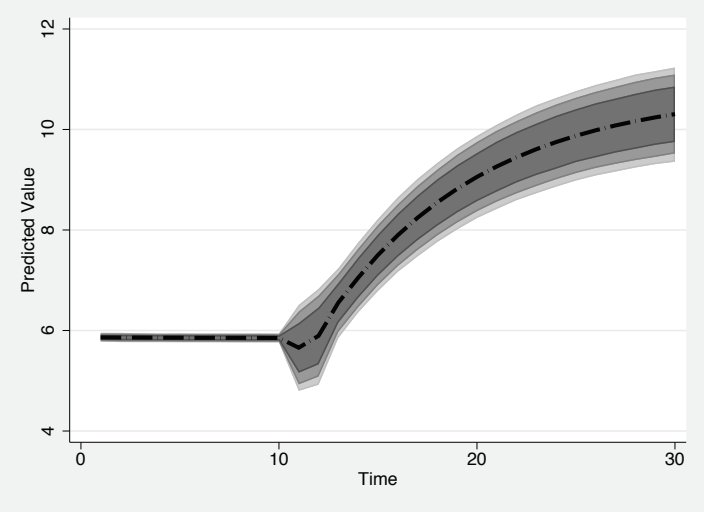

Figure 2. Plot produced from `dynardl` using the `graph` and `rarea` options. The black-dotted line shows the average predicted value; the shaded area shows (from darkest to lightest) the 75%, 90%, and 95% confidence intervals.

`dynardl` can also be used with `pssbounds` when fitting an error-correction model. For instance, directly after estimating (3), we can run `pssbounds` as a postestimation command to test for cointegration without having to specify any additional options; the program automatically obtains the necessary values for the F and t statistics, the number of observations, the case, and the number of regressors, k:

```
. dynardl ln_inv ln_inc ln_consump, lags(1, 1, 1) diffs(., 1, 1)
> shockvar(ln_inc) shockval(-1) time(10) range(30) ec
```

(*output omitted*)

```
. pssbounds
PESARAN, SHIN AND SMITH (2001) COINTEGRATION TEST
Obs: 91
No. Regressors (k): 2
Case: 3
```

F-test

	<——— I(0)	——— I(1) ———>
10% critical value	3.170	4.140
5% critical value	3.790	4.850
1% critical value	5.150	6.360

F-stat. = 2.602

t-test

	<——— I(0)	——— I(1) ———>
10% critical value	-2.570	-3.210
5% critical value	-2.860	-3.530
1% critical value	-3.430	-4.100

t-stat. = -2.372

```
F-statistic note: Asymptotic critical values used.
t-statistic note: Asymptotic critical values used.
```

In addition to error-correction style models, `dynardl` can handle ARDL models where the dependent variable is estimated in levels.[18] For instance, suppose we want to fit the following ARDL(1,1) model:

$$\texttt{ln\_inv}_t = \texttt{ln\_inv}_{t-1} + \texttt{ln\_inc}_t + \texttt{ln\_inc}_{t-1} + \texttt{ln\_consump}_t + \texttt{ln\_consump}_{t-1}$$

In `dynardl`, we add the `levels(., 1, 1)` to let the program know that the two independent variables are to appear contemporaneously in levels. If we wanted to see the effect of a change from `ln_inc` = 6 to `ln_inc` = 5, while holding `ln_consump` constant at 7, we can also use the `forceset()` option to force the program to evaluate the simulations at these values, not the sample means (by default).[19]

```
. dynardl ln_inv ln_inc ln_consump, lags(1, 1, 1) levels(., 1, 1) forceset(., 6, 7)
> shockvar(ln_inc) shockval(-1) time(10) range(30) graph change sims(5000)
```

Because we added the `change` option, the resulting plot is akin to an impulse–response function as shown in figure 3. In other words, we are looking not at the level of the predicted value but at the difference between the predictions at each point in time relative to the average predicted value before the shock. Figure 3 shows the change in

18. Although this is a stylized example, users should always first perform unit-root tests to determine the most appropriate model specification.
19. Note that while we can set some or all of the independent variables using `forceset()`, the lagged dependent variable cannot be forced to a fixed value.

predicted value, starting when the shock occurs.[20] As is clear from the figure, there is no statistically significant change in the predicted value in the short run as a result of the shock. However, over the long run, the change is statistically significant at the 90% level of confidence.

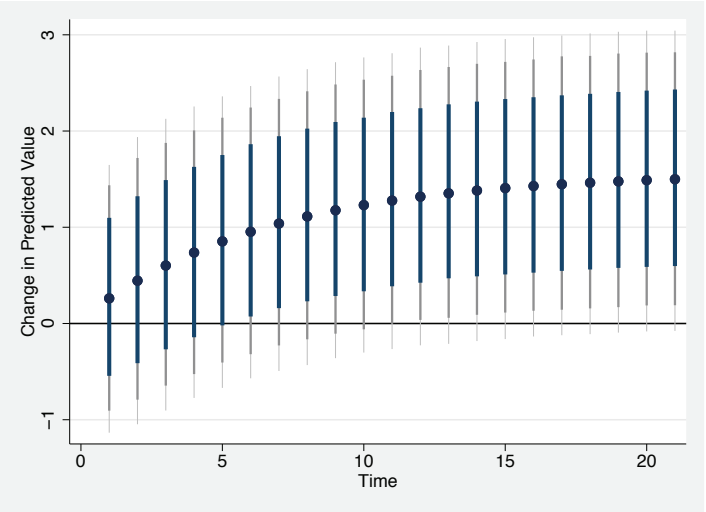

Figure 3. Plot produced from `dynardl` using the `graph` and `change` options. Dots show the mean change in predicted value from sample mean; the shaded area shows (from darkest to lightest) the 75%, 90%, and 95% confidence intervals.

4 Conclusion

In this article, we have introduced `dynamac`, a suite of commands for dynamic ARDL modeling and cointegration testing. `dynamac` consists of two commands designed to assist time-series analysts. `pssbounds` helps users test for cointegration by providing critical values from Pesaran, Shin, and Smith (2001) and Narayan (2005) automatically in a tabular format. The command `dynardl` helps users dynamically simulate a variety of ARDL models to gain a better understanding of the substantive significance of their results. Users can then graph or save their simulated predicted values for use elsewhere. Both commands make it easier for users to test and interpret their dynamic models.

20. In other words, an analyst might want to show the effect of a shock without showing the periods preceding it (like the first 10 periods of figure 2). In figure 3, because the shock occurred at $t = 10$ and the total range of the simulation was $t = 30$, the graph shows a total range of $t = 20$.

5 Acknowledgments

We thank Lorena Barberia, Natália Moreira, Paul Kellstedt, Guy Whitten, Joe Ura, Eric Guntermann, the editor, and a reviewer for thoughtful comments and suggestions on various versions of these commands. Of course, any errors remain our own.

6 References

Breunig, C., and M. R. Busemeyer. 2012. Fiscal austerity and the trade-off between public investment and social spending. *Journal of European Public Policy* 19: 921–938.

Choirat, C., C. Gandrud, J. Honaker, K. Imai, G. King, and O. Lau. 2018. *Zelig: Everyone's statistical software*. R package version 5.1.6. https://cran.r-project.org/web/packages/Zelig/.

Engle, R. F., and C. W. J. Granger. 1987. Co-integration and error correction: Representation, estimation, and testing. *Econometrica* 55: 251–276.

Gandrud, C., L. K. Williams, and G. D. Whitten. 2016. *dynsim: Dynamic simulations of autoregressive relationships*. R package version 1.2.1. https://cran.r-project.org/web/packages/dynsim/.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, eds. 2013. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: CRC Press.

Grant, T., and M. J. Lebo. 2016. Error correction methods with political time series. *Political Analysis* 24: 3–30.

Jennings, W., and P. John. 2009. The dynamics of political attention: Public opinion and the Queen's Speech in the United Kingdom. *American Journal of Political Science* 53: 838–854.

Johansen, S. 1991. Estimation and hypothesis testing of cointegration vectors in Gaussian vector autoregressive models. *Econometrica* 59: 1551–1580.

———. 1995. *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. Oxford: Oxford University Press.

Narayan, P. K. 2005. The saving and investment nexus for China: Evidence from cointegration tests. *Applied Economics* 37: 1979–1990.

Pesaran, M. H., Y. Shin, and R. J. Smith. 2001. Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics* 16: 289–326.

Philips, A. Q. 2018. Have your cake and eat it too? Cointegration and dynamic inference from autoregressive distributed lag models. *American Journal of Political Science* 62: 230–244.

Philips, A. Q., A. Rutherford, and G. D. Whitten. 2016a. dynsimpie: A command to examine dynamic compositional dependent variables. *Stata Journal* 16: 662–677.

———. 2016b. Dynamic pie: A strategy for modeling trade-offs in compositional variables over time. *American Journal of Political Science* 60: 268–283.

Phillips, P. C. B., and S. Ouliaris. 1990. Asymptotic properties of residual based tests for cointegration. *Econometrica* 58: 165–193.

Swank, D., and S. Steinmo. 2002. The new political economy of taxation in advanced capitalist democracies. *American Journal of Political Science* 46: 642–655.

Tomz, M., J. Wittenberg, and G. King. 2003. clarify: Software for interpreting and presenting statistical results. *Journal of Statistical Software* 8(1): 1–30.

Ura, J. D. 2014. Backlash and legitimation: Macro political responses to Supreme Court decisions. *American Journal of Political Science* 58: 110–126.

Ura, J. D., and C. R. Ellis. 2008. Income, preferences, and the dynamics of policy responsiveness. *PS: Political Science & Politics* 41: 785–794.

Whitten, G. D., and L. K. Williams. 2011. Buttery guns and welfare hawks: The politics of defense spending in advanced industrial democracies. *American Journal of Political Science* 55: 117–134.

Williams, L. K., and G. D. Whitten. 2011. Dynamic simulations of autoregressive relationships. *Stata Journal* 11: 577–588.

———. 2012. But wait, there's more! Maximizing substantive inferences from TSCS models. *Journal of Politics* 74: 685–693.

Yule, G. U. 1926. Why do we sometimes get nonsense-correlations between time-series? A study in sampling and the nature of time-series. *Journal of the Royal Statistical Society* 89: 1–63.

About the authors

Soren Jordan is an assistant professor in the Department of Political Science at Auburn University.

Andrew Q. Philips is an assistant professor in the Department of Political Science at the University of Colorado, Boulder.