Dynamic Simulation and Testing for Single-Equation Cointegrating and Stationary Autoregressive Distributed Lag Models

by Soren Jordan and Andrew Q. Philips

Abstract While auto-regressive distributed lag models allow for extremely flexible dynamics, interpreting the substantive significance of complex lag structures remains difficult. In this paper we discuss **dynamac** (dynamic autoregressive and cointegrating models), an R package designed to assist users in estimating, dynamically simulating, and plotting the results of a variety of auto-regressive distributed lag models. It also contains a number of post-estimation diagnostics, including a test for cointegration for when researchers are estimating the error-correction variant of the auto-regressive distributed lag model.

Introduction

Many important social processes vary systematically over time. Models designed to account for such dynamics have become more common in the social sciences. Applied examples range from prime ministerial approval (Clarke et al., 2000), to the determinants of trust in government (Chanley et al., 2000), to preferences for different types of government spending (Wlezien, 1995).

Dynamic processes take many forms. Some of these processes are integrated, such that the current value ("realization") of a series is an expression of all past values in the series plus some new innovation (i.e. $y_t \sim I(1)$ if $y_t = y_{t-1} + \epsilon_t$). We call these series I(1), or non-stationary, or having a "unit root."[1] Other processes revert back to some mean level over time. Since they are not integrated, we often call these series stationary, or I(0). Stationary series are characterized by constant mean, variance, and covariance; non-stationary series violate one or more of these properties. An especially interesting relationship exists when two (or more) integrated variables are in a long-run relationship with each other, even if in the short term the series may move apart. Such series are said to be cointegrated.[2]

The job of properly modeling such a diverse set of dynamic processes is challenging. Researchers want to test whether a theoretical X regressor of interest has an effect on a dependent variable y, accounting for each series' own dynamics. Moreover, these effects might occur in both the short-run and long-run of the series. In cases where we have multiple I(1) variables, we must test whether the variables are in a cointegrated relationship. Then researchers need a way to interpret the effects of these complex models. When modeling dynamics, then, we might be particularly drawn to a strategy that contributes to each of these goals.

We advocate for one particular specification because of its generalizability, flexibility and robustness: auto-regressive distributed lag (ARDL) models. ARDL models can account for multiple lags of independent variables, either in levels or in first-differences, as well as multiple lags of the dependent variable. Moreover, utilizing the ARDL-bounds testing procedure of Pesaran et al. (2001), we can test whether variables in the ARDL model are also in a cointegrating relationship. However, the very flexibility of this modeling strategy produces its own challenges. If the researcher includes multiple lags (or differences, or lagged differences) of some independent variable, it becomes increasingly difficult to discern various effects of a change in X on y, independent of the individual coefficients estimated in the ARDL model. These are more complicated than static models (De Boef and Keele, 2008), since they often involve both short- and long-run effects of X on y.

We introduce a new R package, **dynamac** (Jordan and Philips, 2018), to help ameliorate these two challenges. The core of the package consists of two functions: `pssbounds`, which helps to implement the ARDL-bounds testing procedure for cointegration (Philips, 2018; Pesaran et al., 2001), and `dynardl`, which estimates ARDL models and simulates the effect of some X on y by way of dynamic simulations. The simulations provided by the latter function help provide substantive inferences of some X on y (1) in both the short run *and* long run, (2) when X appears in multiple forms (such as first-differences, lagged first-differences, and/or lagged levels), and (3) accounting for similar specifications of other

[1] A unit-root is one common form of non-stationarity; a series could be non-stationary due to a linear trend, for instance.

[2] Stationary variables cannot be in cointegrated relationships, as their long-run response is to return to their mean value, not a linear combination of some other series.

control variables Z. We also include functions to easily take the lag, first-difference, and lagged-first difference of a series, as well as a host of post-estimation diagnostics users can employ to ensure residuals from their resulting model are white noise. The manuscript proceeds to explain the context of ARDL models generally, cointegration testing and the ARDL-bounds test in particular, the **dynamac** functions to help estimate both the ARDL-bounds test as well as the ARDL models (and their stochastic simulations), concluding with illustrative examples of each function.

The general ARDL model

The auto-regressive distributed lag model has a very general form:

$$\begin{aligned} y_t, \Delta y_t =& \alpha_0 + \delta T + \sum_{p=1}^{P} \phi_p y_{t-p} + \sum_{l_1=0}^{L_1} \theta_{1l} x_{1,t-l_1} + \cdots + \sum_{l_k=0}^{L_k} \theta_{kl} x_{t-l_k} + \\ & \sum_{m=1}^{M} \alpha_m \Delta y_{t-m} + \sum_{q_1=0}^{Q_1} \beta_{1q_1} \Delta x_{1,t-q_1} + \cdots + \sum_{q_k=0}^{Q_k} \beta_{kq_k} \Delta x_{k,t-q_k} + \epsilon_t \end{aligned} \tag{1}$$

The left-hand side can be estimated either in levels (y_t) or in first-differences (Δy_t). This may be a function of a constant, a linear trend, up to p lags of the dependent variable, a series of lags for each of the k regressors—appearing either contemporaneously or with a lag—m lagged first-differences of the dependent variable, and contemporaneous and/or lagged first-differences of each of the regressors. Lagged first-differences may enter into Equation 1 for theoretical reasons, or to help ensure that the resulting residuals ϵ are white noise.[3] Since this model obviously results in a heavy parameterization, restrictions are often imposed. Two of the most common restrictions are the ARDL(1,1) model with all-stationary data:

$$y_t = \alpha_0 + \phi_1 y_{t-1} + \theta_{1,0} x_{1,t} + \cdots + \theta_{k,0} x_{k,t} + \theta_{1,1} x_{1,t-1} + \cdots + \theta_{k,1} x_{k,t-1} + \epsilon_t \tag{2}$$

As well as its non-stationary and cointegrating variant, often referred to as an error-correction model:

$$\Delta y_t = \alpha_0 + \phi_1 y_{t-1} + \theta_{1,1} x_{1,t-1} + \cdots + \theta_{k,1} x_{k,t-1} + \beta_1 \Delta x_{1,t} + \cdots + \beta_k \Delta x_{k,t} + \epsilon_t \tag{3}$$

For clarity, lagged first-differences are not shown, although they may be added to Equations 2 and 3, either for theoretical reasons or to purge autocorrelation from the residuals.[4]

The strength and drawback of the ARDL model should be immediately apparent: while researchers can specify a variety of lagged levels and differences to account for theory as well as ensure white-noise (random) residuals (that is, no residual autocorrelation), the same flexibility can make inferences regarding the total effect of any individual X variable difficult to discern. This is exacerbated even more when we consider that these variables might have short-run and long-run effects. In other words, an immediate change in an X variable might have an immediate impact in the contemporaneous period t, but if lagged values of X also appear in the model, this effect will persist for multiple time periods. Moreover, if the dependent variable is entered into the model in lagged levels through ϕ_p, the effects of the independent variable X in some time period t persist over time. There is a cumulative—or long-run—effect across time, given a change in X in a single time period. In simple cases this can be calculated as a non-linear combination of the lagged parameter estimate and the parameter on the lagged dependent variable (De Boef and Keele, 2008). Yet even these calculations become tedious with multiple lags of the regressors and the dependent variable.

We circumvent this difficulty by employing stochastic simulations rather than direct interpretation of coefficients in order to show statistical and substantive significance. Dynamic stochastic simulations provide an alternative to direct hypothesis testing of coefficients, instead focusing on simulating meaningful counterfactuals from model coefficients many times and drawing inferences from the central tendencies of the simulations. Such simulations are becoming increasingly popular with increased computing power, as demonstrated in the social sciences by recent methodological and applied work (Tomz et al., 2003; Choirat et al., 2018; Breunig and Busemeyer, 2012; Williams and Whitten, 2012; Philips et al., 2015, 2016; Gandrud et al., 2015).

Recall that the other challenge we wish to confront is that of identifying cointegration in auto-regressive distributed lag models in error-correction form. In instances of small-sample data, especially

[3] i.e., $\epsilon \sim N(0, \sigma^2)$.

[4] i.e., users could add: $\sum_{m=1}^{M} \alpha_m \Delta y_{t-m} + \sum_{q_1=0}^{Q_1} \beta_{1q_1} \Delta x_{1,t-q_1} + \cdots + \sum_{q_k=0}^{Q_k} \beta_{kq_k} \Delta x_{k,t-q_k}$.

of time points t of 80 or less, traditional tests like the Engle-Granger "two-step" method (Engle and Granger, 1987) or the Johansen (1991, 1995) approaches too often conclude cointegration when it does not exist (Philips, 2018). An alternative test, proposed by Pesaran et al. (2001), is more conservative, meaning it does not conclude cointegration when it does not exist, especially in small samples. This test, which we call the ARDL-bounds test, is desirable for the social sciences, where shorter time series (smaller samples) are quite common. The difficulty of the ARDL-bounds test is that it requires a specific form of the ARDL model and unique critical values. While these are not insurmountable obstacles, the usefulness of the test would be extended if software existed to get and test these critical values for the user.

The package **dynamac** works to solve both of these deficiencies. It includes a command, `pssbounds` (named for Pesaran, Shin and Smith, the authors of the test), which implements the ARDL-bounds test for cointegration. A second command, `dynardl`, estimates auto-regressive distributed lag models and implements the stochastic simulations we describe above. The commands also have the virtue of working together: estimating an ARDL model in error-correction form with `dynardl` by nature stores the values necessary to execute the ARDL-bounds test with `pssbounds`. Below, we use a substantive example to motivate our discussion of the package in greater detail.

Modeling using dynamac

We now illustrate the process—auto-regressive distributed lag modeling, testing for cointegration with `pssbounds`, and interpretation of X through stochastic simulations—using data originally from Wright (2017) on public concern about inequality in the United States.[5] For our example, assume that public concern about inequality in the US, Concern (`concern`), is a function of the share of income going to the top ten percent, Income Top 10 (`incshare10`). We also hypothesize that the unemployment rate, Unemployment (`urate`), affects concern over the short-run (i.e., is not cointegrating):

$$\text{Concern}_t = f(\text{Income Top 10}_t + \Delta\text{Unemployment}_t) \tag{4}$$

Before estimating any model using **dynamac**, users should first check for stationarity. A variety of unit root tests can be performed using the **urca** package (Pfaff et al., 2016). These suggest that all three series are integrated of order I(1), as they appear integrated in levels but stationary in first-differences (Δ), shown in Table 1.

Table 1: Unit root tests

	Augmented Dickey Fuller	Phillips-Perron	Dickey-Fuller GLS	KPSS
Concern	0.688	-3.437*	-0.893	0.642*
ΔConcern	-3.507*	-7.675*	-3.124*	0.814*
Unemployment	-0.612	-2.762	-2.802*	0.224
ΔUnemployment	-5.362*	-4.879*	-5.308*	0.064
Income Top 10	2.992	0.442	0.994	2.482*
ΔIncome Top 10	-3.170*	-6.244*	-4.032*	0.218

Note: One augmenting lag included for all tests. $*: p < 0.05$. ADF, PP, and DF-GLS have null hypothesis of a unit-root, while KPSS has a null of stationarity.

Given that all series appear to be I(1), we proceed with estimating a model in `dynardl` in error-correction form, and then testing for cointegration between concern about inequality and the share of income of the top 10 percent. In general, we suggest using this strategy—outlined in Philips (2018)—along with alternative tests for cointegration.[6] Our error-correction model appears as:

$$\begin{aligned}\Delta\text{Concern}_t = \alpha_0 + \phi_1\text{Concern}_{t-1} + \theta_1\text{Income Top 10}_{t-1} + \\ \beta_1\Delta\text{Income Top 10}_t + \beta_2\Delta\text{Unemployment}_t + \epsilon_t\end{aligned} \tag{5}$$

[5] Available at: http://dx.doi.org/10.7910/DVN/UYUU9G. For simplicity, we use a smaller version of Wright's dataset with missing values at the beginning of the series removed.

[6] These include the Johansen (1995) cointegration test (available in **urca**) or the proposed test by Engle and Granger (1987), among others.

where we assume $\epsilon_t \sim N(0, \sigma^2)$. Now we estimate this model with `dynardl`.

Estimation using `dynardl`

The syntax for `dynardl`, the main function in the package, is as follows:

- `formula`, a formula of the type $y \sim x_1 + x_2 + x_3 + \ldots x_k$. Note the variables do not need to be lagged or differenced in the specification; these transformations will be handled automatically by `dynardl`.
- `data`, an optional argument specifying a particular dataset in which the variables from `formula` can be found.
- `lags`, a *list* of variables to be lagged and their corresponding lagged levels. For instance, if an analyst had two variables, X_1 and X_2 that he or she wished to incorporate with a single lag, $t-1$, he or she would specify `lags = list("X1" = 1,"X2" = 1)`. If he or she wanted to incorporate multiple lags on X_2 from time periods $t-1$ and $t-2$, the code would appear as `lags = list("X1" = 1,"X2" = c(1,2,...)` , expanded for however many lags were desired.
- `diffs`, a *vector* of variables to be differenced, i.e., included as ΔX_t. Only first-differences are supported. For instance, if an analyst wanted to first-difference X_1 and X_3, he or she would include `diffs = c("X1",X3")`.
- `lagdiffs`, a *list* of variables to be included with lagged first-differences. The syntax is identical to `lags`, but instead of incorporating lagged levels of the relevant variable X_k at $t-1$, lagged differences $\Delta X_{k,t-1}$ are included. For instance, if the analyst wanted to include a first lagged difference of $X_{1,t}$ at $t-1$, he or she would include `lagdiffs = list("X1" = 1)`.
- `levels`, a *vector* of variables to be included in levels contemporaneously—i.e., at time t. If the analyst wanted to include $X_{2,t}$ and $X_{3,t}$ in levels at time t, he or she would include `levels = c("X2","X3")`.
- `ec`, a TRUE/FALSE option for whether the model should be estimated in error-correcting form. If `ec = TRUE`, the dependent variable will be run in differences. The default is FALSE.
- `trend`, a TRUE/FALSE option for whether a linear trend should be included in the regression. The default is FALSE.
- `constant`, a TRUE/FALSE option for whether a constant should be included in the regression. The default is TRUE.
- `modelout`, a TRUE/FALSE option for whether model out from the ARDL model should be printed in the console. The default is FALSE.
- `simulate`, a TRUE/FALSE option for whether any shocks should be simulated. Since simulations are potentially computationally intensive, if the analyst is simply using `dynardl` to test for cointegration using the bounds procedure, or wishes to just view the model output, he or she might not wish to estimate simulations in the interim. The default is TRUE.[7]

Reading the above, `dynardl()` is simply an engine for regression, but one that allows users to focus on theoretical specification rather than technical coding. All variables in the model are entered into the formula. In this sense, `dynardl()` can be used in *any* ARDL context, not just ones in which the user is also expecting cointegration testing or dynamic simulations.We estimate our example model shown in Equation 5 using `dynardl` as follows:

```
load("inequalitymood.rda")

res1 <- dynardl(concern ~ incshare10 + urate, data = ineq,
        lags = list("concern" = 1, "incshare10" = 1),
        diffs = c("incshare10", "urate"),
        ec = TRUE, simulate = FALSE )

[1] "Error correction (EC) specified; dependent variable to be run in differences."
```

We specify the formula by letting `dynardl` know what the dependent variable and the regressors are. Next, in the `lags` option, we let the program know we want a lag at $t-1$ for the dependent variable and for Income Top 10.[8] Since Unemployment did not appear in lag form in Equation 5, it

[7] For this first example, we will treat this option as FALSE.

[8] If we wanted further lags, we could specify it here, for instance `lags = list("concern" = c(1,2), "incshare10" = 2, "urate" = 2)` would add lags at $t-1$ and $t-2$ for concern, and at $t-2$ for `incshare10` and `urate`.

does not appear in `lags`. In `diffs`, we let `dynardl` know to include the first-difference for both Income Top 10 and Unemployment. The option `ec = TRUE` estimates the dependent variable in error-correction form (i.e., takes the first-difference of Concern), and the program will issue a message indicating to the user that such a transformation has taken place.[9] By setting `simulate = FALSE`, we save on computing time by just estimating the model without performing stochastic simulations needed to make a substantive inference through stochastic simulations, as shown in the next sections. This is useful if the residuals of our first model indicate residual autocorrelation and require respecification.

To see the usual regression summary, we recommend running `summary(foo$model)`, rather than `summary(foo)`:

```
summary(res1$model)

Call:
lm(formula = as.formula(paste(paste(dvnamelist), "~", paste(colnames(IVs),
    collapse = "+"), collapse = " ")))

Residuals:
      Min        1Q    Median        3Q       Max
-0.025848 -0.005293  0.000692  0.006589  0.031563

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.122043   0.027967   4.364 7.87e-05 ***
l.1.concern    -0.167655   0.048701  -3.443   0.0013 **
d.1.incshare10  0.800585   0.296620   2.699   0.0099 **
d.1.urate       0.001118   0.001699   0.658   0.5138
l.1.incshare10 -0.068028   0.031834  -2.137   0.0383 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.01169 on 43 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.3671,        Adjusted R-squared:  0.3083
F-statistic: 6.236 on 4 and 43 DF,  p-value: 0.0004755
```

As shown from the regression results, `dynardl` has included a constant, the lagged dependent variable, `l.1.concern`, the first difference of the two regressors (Income Top 10 and Unemployment), as well as the lag of Income Top 10.[10] While changes in Income Top 10 affect changes in Concern in the short-run, changes in Unemployment do not have a statistically significant effect in the short-run. The lag of Income Top 10 is negative and statistically significant. Moreover, the parameter on the lagged dependent variable is negative, between 0 and -1, and statistically significant, giving us cursory evidence of a cointegrating process taking place; we use a statistical test for this below.

Post-estimation diagnostics

An essential component of ARDL modeling is ensuring that the residuals from any ARDL estimation are white noise; this is especially important in regards to autocorrelation. In fact, it is necessary to adjust the model in order to ensure white-noise residuals before running the Pesaran et al. (2001) ARDL-bounds cointegration test. To assist users in model selection, we offer `dynardl.auto.correlated()`. This function takes the residuals from an ARDL model estimated by `dynardl()` and conducts two tests for autocorrelation—the Shaprio-Wilk test for normality and the Breusch-Godfrey test for higher-order serial correlation—as well as calculates the fit statistics for the AIC, BIC, and log-likelihood. The arguments for this function are:

- `x`, a `dynardl` model object.
- `bg.type`, a character string, either `"Chisq"` or `"F"` of the type of Breusch-Godfrey test for higher-order serial correlation. This returns either the Chi-squared test statistic or the F statistic. The default is `"Chisq"`.

[9] Setting this option to ‘`FALSE`’ would estimate the dependent variable in levels. If `ec = TRUE`, the saved object will also contain `$pssbounds`, described below.

[10] Lagged variables are denoted by `l.X.variable`, where `X` is the lag. First-differences are denoted by `d.variable`. Lagged first-differences are denoted by `l.d.X.variable`, where `X` is the lag.

- `digits`, an integer for the number of digits to round fit statistics to (by default, this is three digits).
- `order`, an integer for the highest possible order of autocorrelation to test in the data. By default, this is computed by the length of the series.
- `object.out`, a TRUE/FALSE option for whether to print this output into an object. This might be useful if the analyst is comparing multiple ARDL objects and testing for white-noise residuals on the basis of fit criteria like AIC. In this case, the analyst could assign the output to some object for later comparison.

To run this post-estimation diagnostics for our model above, we type:

```
dynardl.auto.correlated(res1)

-------------------------------------------------------
Breusch-Godfrey LM Test
Test statistic: 3.704
p-value: 0.054
H_0: no autocorrelation up to AR 1

-------------------------------------------------------
Shapiro-Wilk Test for Normality
Test statistic: 0.965
p-value: 0.158
H_0: residuals are distributed normal

-------------------------------------------------------
Log-likelihood: 148.094
AIC: -284.187
BIC: -272.96
Note: AIC and BIC calculated with k = 5 on T = 48 observations.
```

Given the results of the Breusch-Godfrey LM test for autocorrelation, there appears to be some autocorrelation still in the residuals. To mitigate this, we can add lagged first-differences to our model (Philips, 2018). For instance, to add a lagged first-difference of the dependent variable:

```
res2 <- dynardl(concern ~ incshare10 + urate, data = ineq,
        lags = list("concern" = 1, "incshare10" = 1),
        diffs = c("incshare10", "urate"),
        lagdiffs = list("concern" = 1),
        ec = TRUE, simulate = FALSE)
```

For brevity the results are not shown here, but they indicate that AR(1) autocorrelation is no longer present in the residuals after adding a lagged first-difference of the dependent variable.

Cointegration testing using `pssbounds`

Recall the definition of cointegration: two or more integrated series of order I(1) are cointegrated if some linear combination of them results in a stationary series.[11] This means that they have some long-run relationship; although temporary shocks may move the series apart, there is an attracting force that brings them back into a stable equilibrium relationship over the long-run. These movements are commonly referred to as being "corrected" or "re-equilibrated"; cointegrated series do not simply happen to move together, but are quite literally brought back into long-run relationship by an attracting force. Since not all I(1) series are cointegrating, failing to account for cointegration—in other words, including I(1) series in an error-correction model when they are not cointegrating—can lead to a drastic increase in Type I error (c.f., Grant and Lebo, 2016).

A variety of cointegration tests have been proposed, which we do not discuss here. Instead, we advocate for a test that has been shown to demonstrate strong small sample properties (Philips, 2018); the ARDL-bounds test for cointegration (Pesaran et al., 2001). The ARDL-bounds procedure proceeds as follows. First, analysts must ensure that the regressors are not of order I(2) or more, and that any seasonal component of the series has been removed. Second, analysts must ensure that the

[11] For instance, given $y_t = \kappa_0 + \kappa_1 x_t + z_t$, cointegration would exist if $z_t \sim$I(0). Higher orders of cointegration may also be cointegrating—something known as multicointegration—but these are not supported by the ARDL-bounds test.

dependent variable y is I(1). A wide variety of unit root tests are designed to assist in this, including the Dickey-Fuller, Elliott-Rothenberg-Stock, Phillips-Perron, and Kwiatkowski-Phillips-Schmidt-Shin tests. We reference these tests in Table 1.

Once the analyst ensures that the dependent variable y is I(1), and the independent variables are not of order I(2) and contain no seasonal components, he or she estimates an ARDL model in error-correction form.[12] That form resembles Equation 6, where the dependent variable is run in differences:

$$\begin{aligned}\Delta y_t =& \alpha_0 + \phi_1 y_{t-1} + \theta_{1,1} x_{1,t-1} + \cdots + \theta_{k,1} x_{k,t-1} + \beta_1 \Delta x_{1,t} + \cdots + \beta_k \Delta x_{k,t} + \\ & \sum_{m=1}^{M} \alpha_m \Delta y_{t-m} + \sum_{q_1=1}^{Q_1} \beta_{1q_1} \Delta x_{1,t-q_1} + \cdots + \sum_{q_k=1}^{Q_k} \beta_{kq_k} \Delta x_{k,t-q_k} + \epsilon_t \end{aligned} \tag{6}$$

Two points are key. First, any independent variables that are potentially I(1) must be entered in lagged levels at $t-1$. Second, the resulting residuals of the ARDL model must be white noise in order to perform the cointegration test (approximately normally distributed, with no residual autocorrelation). If they are not, the analyst should incorporate additional lags of the first difference of the variables until they are (Philips, 2018). A variety of tests exists to help determine whether the residuals are white noise, including information criteria (like SBIC and AIC), Breusch-Godfrey tests, Durbin's Alternative test, and Cook-Weisberg. Users can find most of these in `dynardl.auto.correlated`, discussed in the previous section.

The special case of the ARDL model in Equation 6 here is that the X variables appearing in levels ($\theta_1 \dots \theta_k$) are the only ones that can possibly be in a cointegrating relationship with the dependent variable y. The ARDL-bounds test for cointegration works by using a Wald or F-test on the following restriction from Equation 6

$$H_0 : \phi_1 = \theta_{1,1} = \cdots = \theta_{k,1} = 0 \tag{7}$$

In other words, that the coefficients on variables appearing in first lags are jointly equal to zero. The null hypothesis is that of no cointegration. Rejecting the null hypothesis indicates there is a cointegrating relationship between the series. In addition to the F-test, a one-sided t-test may be used to test the null hypothesis that the coefficient on the lagged dependent variable (in levels) is equal to zero, or

$$H_0 : \phi_1 = 0 \tag{8}$$

The alternative to this test is that $\phi_1 < 0$, or that y is cointegrating with the regressors. This is known as the bounds t-test.

The procedure and tests themselves are relatively straightforward to implement, but are complicated by two factors. The first is that critical values for these tests are non-standard, meaning that they are not readily testable in canned procedures. The correct (asymptotic) critical values are provided by Pesaran et al. (2001), and the small sample critical values for the F-statistic are given by Narayan (2005).[13] Moreover, the appropriate critical values depend on the "case" of the regression: a combination of whether the regression includes a (restricted) trend term (or not) and a (restricted) constant (or not). As Philips (2018) describes, whether the resulting F-statistic is higher than the I(1) critical value, below the I(0) critical value, or between the two, gives differential evidence on the integrated nature of the X variables and the potential cointegrating relationship between them.

Just looking at the above discussion: even if the test is powerful, it can be difficult to implement and interpret correctly. Accordingly, we offer two functions to help implement the test and get dynamic inferences. The first function, `dynardl`, estimates the ARDL relationship described above (to get the resulting F-statistic and t-statistic). The second function, `pssbounds`, takes the results of these ARDL models and implements the ARDL-bounds test for cointegration, providing the user with the correct critical values and an easy-to-understand description of hypothesis testing.

To summarize, if we find that the dependent variable and regressors are all I(1), they can only

[12] Note this does not require the analyst to make the sharp distinction between an I(0) or an I(1) regressor, a special benefit of the ARDL-bounds procedure. This is especially useful when "near integrated" series can appear to be integrated in small samples even though, if we were to observe more innovations, the series would ultimately return to some mean value (that is, it would be stationary).

[13] No small-sample critical values are given for the t-test, so it should not be used for decisions between cointegrating relationships or not, but rather for confirmatory evidence of a different test.

appear in lagged levels in the error-correction model if there is evidence of cointegration. **dynamac** contains the Pesaran et al. (2001) ARDL-bounds cointegration test to assist users in testing for this. To implement the bounds testing procedure, we introduce the function `pssbounds()`. This function has the following syntax:

- `data`, an optional argument expecting a `dynardl()` model, which calculates the following arguments for the user. If none is given, the user must supply each of the following:
- `obs`, an integer for the number of observations in the model time series.
- `fstat`, the *F* statistic on the restriction that each of the first lags in the model (except the lagged dependent variable) are jointly equal to zero.
- `tstat`, the *t* statistic on the lagged dependent variable in the error-correction model.
- `case`, a numeric 1-2-3-4-5 or numeral I-II-III-IV-V of the case of the regression.
  - 1: No intercept and no trend
  - 2: Restricted intercept and no trend
  - 3: Unrestricted intercept and no trend (the most common case)
  - 4: Unrestricted intercept and restricted trend
  - 5: Unrestricted intercept and unrestricted trend
- `k`, the number of regressors appearing in first lags.
- `digits`, an integer for the number of digits to round fit statistics to (by default, this is three digits).
- `object.out`, a TRUE/FALSE option for whether to print this output into an object. The default is FALSE.

This command can be implemented in two ways. First, users can run the appropriate model (following the instructions of Philips (2018)) and pass the relevant arguments—`obs`, `fstat`, `tstat`, `case`, and `k`—to `pssbounds`. Alternatively—and as we advocate—users can estimate the model using `foo <-dynardl(...)` and then, post-estimation, run `pssbounds(foo)`.[14]

Moving back to our example, to test for cointegration, we use the `pssbounds` command:[15]

```
res2 <- dynardl(concern ~ incshare10 + urate, data = ineq,
        lags = list("concern" = 1, "incshare10" = 1),
        diffs = c("incshare10", "urate"),
        lagdiffs = list("concern" = 1),
        ec = TRUE, simulate = FALSE)

pssbounds(res2)

PESARAN, SHIN AND SMITH (2001) COINTEGRATION TEST

 Observations: 47
 Number of Regressors (k): 1
 Case: 3

 ------------------------------------------------------
 -                      F-test                        -
 ------------------------------------------------------
                <------- I(0) ------------ I(1) ----->
 10% critical value       4.19            4.94
 5% critical value        5.22            6.07
 1% critical value        7.56            8.685


 F-statistic = 6.837
 ------------------------------------------------------
 -                      t-test                        -
 ------------------------------------------------------
```

[14] As described earlier, this only applies to models where `ec = TRUE`, as only error-correcting models can have cointegration in the ARDL-bounds procedure.

[15] To re-emphasize, this test is inapproprate if the residuals are not white-noise. Therefore it is crucial to users to first test for stationarity, run `dynardl`, and adjust their model as needed before testing for cointegration.

```
        <------- I(0) ------------ I(1) ----->
10% critical value       -2.57           -2.91
5% critical value        -2.86           -3.22
1% critical value        -3.43           -3.82


t statistic = -3.684
-----------------------------------------------------

t-statistic note: Small-sample critical values not provided for Case III.
Asymptotic critical values used.
```

The program displays the number of observations from the regression, the number of regressors appearing in lagged levels (i.e., the cointegrating equation), and the case (unrestricted intercept and no trend); all of these are necessary to obtain the special critical values used by Pesaran et al. (2001).[16] This situates the F-statistic and t-statistic with the correct critical values for the appropriate test. Users are then free to compare the test statistics to the critical values and make the appropriate deduction regarding cointegration in the sample model.

In our example, since the value of the F-statistic exceeds the critical value at the upper I(1) bound of the test at the 5% level, we may conclude that Income Top 10 and Concern about inequality are in a cointegrating relationship. As an auxiliary test, `pssbounds` displays a one-sided test on the t-statistic on the lagged dependent variable.[17] Since the t-statistic of -3.684 falls below the 5% critical value I(1) threshold, this lends further support for cointegration. Taken together, these findings indicate that there is a cointegrating relationship between concern about inequality and the income share of the top 10 percent, and that Equation 6 is appropriately specified.[18]

Gaining substantive and statistical significance of results using `dynardl`

Once users have decided on the appropriate theoretical model—accounting for the appropriate lags of the independent variables, first-differenced variables, and so on—and drawn conclusions regarding cointegration using `pssbounds` (if applicable), their models might be somewhat complicated. Our solution to inferences in the face of this complexity is the use of stochastic simulations.[19]

If a user runs `dynardl` with the argument `simulate = TRUE`, then the following options become available:

- `range`, an integer for how long the simulation is to run (the default is 20 periods)
- `sig`, an integer for any user-desired significance level in the form of $1 - p$. So if the user wants a p value of 0.10, `sig` would be 90. The default level is 95.
- `time`, the time period within the `range` of the simulation when to shock the independent variable of interest (the `shockvar`). The default is 10.
- `shockvar`, the independent variable to be shocked in the simulation. Analysis should enter this in quotes, like `shockvar = "X1"`. This is the only option required without a default: we must know which variable we want inferences about.
- `shockval`, the amount by which the independent variable should be shocked. The default is a standard deviation of the shock variable.
- `sims`, the number of simulations to run. The quantities of interest are created from these simulations. The default number is 1000.
- `forceset`, a *list* of variables and their values that should be set to particular values during the simulations. By default, variables in levels are held at their means, and variables in differences are held at 0. If the analyst wanted to investigate the change in y when some variables are at different values, he or she should do this with `forceset`. For instance, if X_1 was meant to be held at 3 (rather than the mean of X_1), the analyst would specify `forceset = list("X1" = 3)`.

[16] Pesaran et al. (2001) provide asymptotic critical values. We use small-sample critical values, calculated by Narayan (2005), when available.

[17] We call this an auxiliary test since only asymptotic critical values are available, while small sample critical values are available for most F-tests. When not available, `pssbounds` issues a warning note.

[18] Had we not found evidence of cointegration, (either the F-statistic fell between or below the I(1) and I(0) critical values), we could adjust our model accordingly, as outlined by Philips (2018).

[19] Note: if users are aiming for analysis to be able to be repeated, they should set a seed before engaging in stochastic simulations.

- `burnin`, the number of time periods before the range begins. `burnin` allows time for the variables to equilibriate. The default is 20. This means that 20 time periods are simulated, then the range of the simulation begins. Users are not shown the `burnin` time periods.
- `expectedval`, a TRUE/FALSE option for whether the expected values (which average errors within simulations) or predicted values (which do not) are desired. Unless the analyst has a strong reason to use expected values, we recommend using predicted values by including `expectedval = FALSE`, the default.
- `graph`, a TRUE/FALSE option for whether a graph of the simulation results should be produced upon estimation. The default is FALSE.
- `rarea`, a TRUE/FALSE option for whether an area plot should produced from the simulation. If `graph = TRUE` and `rarea = TRUE`, an area plot will be produced. `graph = TRUE` and `rarea = FALSE`, a spike plot will be produced. If `graph = FALSE`, `rarea` is ignored.

`dynardl()` takes this set of arguments and generates the appropriate auto-regressive distributed lag given the levels, lagged levels, first-differences, and lagged first-differences of the variables provided. Once this model is estimated using OLS, `dynardl()` simulates the quantities of interest. Specifically, the coefficients $\hat{\beta}$ are simulated as draws (with number of `sims`) from a multivariate normal distribution with mean of $\hat{\beta}$ and variance from the variance-covariance of the estimated model (using the **MASS** package from Ripley (2018)). We introduce uncertainty into these simulations by simulating $\hat{\sigma}^2$ scaled by random draws from the chi-squared distribution. We then use this estimate of $\hat{\sigma}^2$ to add back in fundamental random error to the predicted value of each simulation. These predictions come from set levels of the independent variables: the X variables in levels are held at their means and other variables in differences are held at 0. Users can override this default by specifying a different value for any variable using `forceset`. Note that the equation is given time to equilibriate (through the use of `burnin` periods) before the independent variable is shocked.

Perhaps the most important argument is `shockvar`. This is the variable for which the user wants some sort of inference regarding the effect of an independent variable on the dependent variable: the response in the dependent variable is created by shocking the `shockvar`. At time `time`, the `shockvar` is "shocked" by the amount of `shockval`. This shock is distributed appropriately, as defined by the model specified. For instance, if the `shockvar` is entered into the model in lagged levels at time $t-1$, that X variable would be shocked at `time` $+1$. Just like in an interactive model, one of the best ways to gain inferences regarding the effect of X on y is by varying a single X at a single point in time. Accordingly, only a single `shockvar` can be specified.

The only thing left to resolve is what to do with our `sims`. We are not particularly interested in any individual simulation in any individual time period. Rather, we can get a sense of the central effect of `shockvar` on y by doing some meaningful analysis across the simulations within any time period. Accordingly, in `foo$simulation`, `dynardl()` stores the mean value of y in each time period. It also stores the lower and upper percentiles for the 75%, 90% and 95% intervals. Users can add additional intervals with `sig`. Additionally, users can specify whether they want predicted values (the average across the simulations within a time period) or expected values (where each individual simulation essentially averages out fundamental error). This is through the option `expectedval = FALSE` or `TRUE`, respectively; we highly recommend the former.

We also advocate for a graphical interpretation of results. This can be accomplished in two ways, with two forms of graphs. `dynardl()` will oblige the user with a plot of the values of the simulations automatically if the option `graph = TRUE`. Two types of graphs are available: area graphs (`rarea = TRUE`) will show the 75%, 90%, and 95% likely values from the simulations using gradients of blue. Otherwise (`rarea = FALSE`), a spike graph will be presented. Both options are shown below.

If users prefer to create their own graphs, or if creating a graph for each auto-regressive distributed lag model estimated seems redundant, or if the user has multiple `shockvars` to investigate and wants to save each to their own object, he or she can turn `graph = FALSE` and save each model and set of simulations to a different `dynardl()` object. Then, for each object, the appropriate simulations and central values can be recovered under `object$simulation` and used later.

Moving back to our substantive example, now that we have our model and found the relationship to be cointegrating, we move onto interpreting the results through dynamic simulations. Although analytic calculations of short- and long-run effects are possible (c.f., De Boef and Keele, 2008), with a multitude of lags and lagged first differences, the ability to interpret the effect on y, given a change in a regressor, can become even more difficult. Therefore, we prefer the growing use of graphical presentations of results through simulation techniques (Tomz et al., 2003; Choirat et al., 2018; Breunig and Busemeyer, 2012; Williams and Whitten, 2012; Philips et al., 2015, 2016; Gandrud et al., 2015). All the researcher needs to do is imagine the counterfactual he or she wishes to investigate. For instance, what is the effect of a 7-percentage-point increase in Income Top 10 on Concern? To examine statistical and substantive significance of this counterfactual, we estimate the following model below and plot it.

Note that users may want to first set a seed in order to ensure that the plots are reproducible, since the simulations are stochastic. The resulting plot is shown in Figure 1.

```
set.seed(2059120)
res3 <- dynardl(concern ~ incshare10 + urate, data = ineq,
        lags = list("concern" = 1, "incshare10" = 1),
        diffs = c("incshare10", "urate"),
        lagdiffs = list("concern" = 1),
        ec = TRUE, simulate = TRUE,
        shockvar = c("incshare10"), shockval = .07)

area.simulation.graph(res3)
```

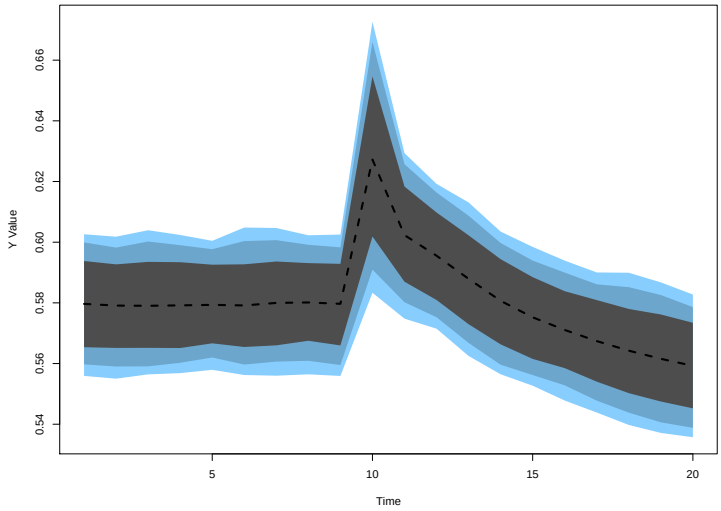

Figure 1: Area plot produced using `area.simulation.graph()`

The black dotted line shows the predicted value, while the colored areas, from darkest to lightest, show the 75, 90, and 95 percentiles of the predictions from the simulations, something akin to a credible interval or confidence interval. If instead we desire a spikeplot, we could type:[20]

```
spike.simulation.graph(res3)
```

The resulting plot is shown in Figure 2.

Figures 1 and 2 both illustrate that the immediate effect of a 7-percentage-point increase in Income Top 10 is to increase Concern by about 4 percentage points. Over time, this effect decays, eventually moving Concern to a new value of 0.56 from its pre-shock mean of 0.58. For clarity: these effects are produced by estimating the model formula given by the user, taking the estimated coefficients from the model, drawing a number of simulated coefficients from a multivariate normal distribution with mean and variance given by the estimated coefficients and variance-covariance from the model, and using them to predict values of the dependent variable (and incorporating fundamental error by drawing stochastic terms from $\hat{\sigma}^2$). It then tracks the response in these predicted values to simulated shifts in the independent variables. We might also note that the dependent variable *could* still be responding: users could increase the `range` of the simulation time periods to ensure that the dependent variable has enough time to respond.

Users may have noticed fluctuations in the confidence intervals shown in Figures 1 and 2. This is due to the stochastic sampling technique used to create these simulations. When using `dynardl`, users have the option of increasing the number of simulations performed using the `sims` option. This should have the effect of smoothing out period-to-period fluctuations in the confidence intervals, at the cost of increased computing time.

[20] Remember that we could create area plots and spikeplots in a single step in `dynardl` by adding the options `graph = TRUE` and `rarea = TRUE` or `rarea = FALSE`, respectively.

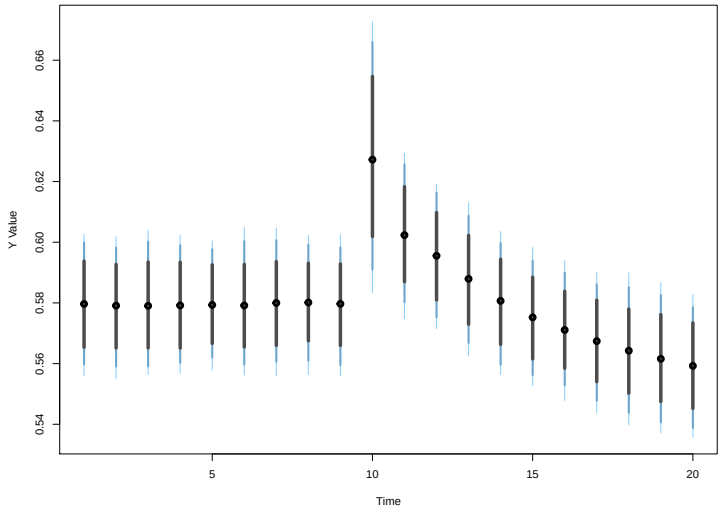

Figure 2: Spike plot produced using `spike.simulation.graph()`

Additional tools for dynamic analysis with dynamac

`dynardl` has a number of additional features users may find useful. First, not only is it suitable for error-correction models; `dynardl` can model a variety of stationary ARDL processes. To see this, we show a simple example using data from Durr et al. (2000), who examine how ideological movements in the United States affect aggregate public support for the Supreme Court. Our model is:

$$\begin{aligned}\text{Supreme Court Support}_t =& \alpha + \phi_1 \text{Supreme Court Support}_{t-1} + \beta_1 \text{Ideological Divergence}_t + \\ & \beta_2 \text{Congressional Approval}_t + \epsilon_t\end{aligned} \tag{9}$$

where support for the US Supreme Court (1973-1993) (`dcalc`) is a function of a constant, the lag of Supreme Court Support, the ideological divergence between the Supreme Court and the public (Ideological Divergence, `iddiv`), and the level of Congressional Approval (`congapp`).[21] To estimate this model using `dynardl`, we type:

```
res4 <- dynardl(dcalc ~ iddiv + congapp, data = supreme.sup,
        lags = list("dcalc" = 1),
        levels = list("iddiv", "congapp"),
        ec = FALSE, simulate = FALSE)

[1] "Dependent variable to be run in levels."
```

Anytime we specify `levels`, any variables appearing in the list will be included contemporaneously (i.e., appear at time t without any lags or differences) in the model. As with the error-correction specification, we can obtain regression results by typing:

```
summary(res4$model)

Call:
lm(formula = as.formula(paste(paste(dvnamelist), "~", paste(colnames(IVs),
        collapse = "+"), collapse = " ")))

Residuals:
     Min       1Q   Median       3Q      Max
-12.1912  -3.7482  -0.2058   2.6513  11.9549

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  41.6319    11.7367   3.547 0.001078 **
```

[21] As stressed in previous sections, stationarity testing is recommended on all dependent and independent variables before determining whether a stationary or error-correcting ARDL model should be estimated. This is a stylized example only meant to demonstrate additional capabilities of **dynamac**.

```
l.1.dcalc     0.2437     0.1352   1.802 0.079758 .
iddiv        -5.4771     2.5803  -2.123 0.040535 *
congapp       0.4732     0.1270   3.725 0.000649 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.626 on 37 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared:  0.4693,        Adjusted R-squared:  0.4263
F-statistic: 10.91 on 3 and 37 DF,  p-value: 2.833e-05
```

As the results show, the lag of Supreme Court Support is only weakly related to Supreme Court Support. Moreover, as Ideological Divergence grows, Supreme Court Support decreases. In contrast, as Congressional Approval increases, Supreme Court Support also increases. We can also bring a substantive interpretation to these effects by again using stochastic simulations. We can investigate the counterfactual of a 15-percentage-point decrease in Congressional Approval on Supreme Court Support. Moreover, we'll demonstrate the difference between the number of simulations used to generate predictions in Figure 3.

```
# 1000 sims
res4 <- dynardl(dcalc ~ iddiv + congapp, data = supreme.sup,
        lags = list("dcalc" = 1),
        levels = list("iddiv", "congapp"),
        ec = FALSE, simulate = TRUE,
        shockval = -15, shockvar = c("congapp"))

# 15000 sims
res5 <- dynardl(dcalc ~ iddiv + congapp, data = supreme.sup,
        lags = list("dcalc" = 1),
        levels = list("iddiv", "congapp"),
        ec = FALSE, simulate = TRUE,
        shockval = -15, shockvar = c("congapp"),
        sims = 15000)

par(mfrow = c(2, 1))
area.simulation.graph(res4)
area.simulation.graph(res5)
```

The effect of the 15-percentage-point decrease in Congressional Approval is about a 10-point decrease in Supreme Court Support. To uncover this effect, the upper plot uses 1000 simulations, while the bottom uses 15000 and results in a much smoother measure of uncertainty in predictions across the simulated timeframe, with only a slight increase in estimation time.[22] Of course, it is important to note that simulations do not *lower* uncertainty (simulating more values does not decrease the underlying variance). More simulations just improve and smooth our estimates within each time period.

While the default range for simulations is 20 time periods, with a shock occurring at time $t = 10$—as shown in Figure 3 for instance—users can change these using the `time` and `range` options. In addition, `dynardl` offers the ability to set variables at particular values other than their means throughout the simulation through the option `forceset`.[23] This would be ideal when the mean value is not particularly intuitive or of substantive interest, such as when dealing with a dichotomous variable. Instead, users can specify particular values using the option `forceset`. As an example of this, we re-estimate the model from above, but extend the total simulation range to $t = 50$, the shock time to $t = 20$, and set both Congressional Approval and Ideological Divergence to their 75th percentiles instead of their means:

```
res6 <- dynardl(dcalc ~ iddiv + congapp, data = supreme.sup,
        lags = list("dcalc" = 1),
        levels = c("iddiv", "congapp"),
        ec = FALSE, simulate = TRUE,
        shockval = -15, shockvar = c("congapp"),
        sims = 15000, time = 20, range = 50,
        forceset = list("iddiv" = 0.22, "congapp" = 72))
```

[22] For comparison, in this model, 1000 simulations took 3.98 seconds, while 15,000 took 4.12 seconds.

[23] Recall that, by default, variables in levels are set to their means throughout the simulation (unless shocked), while variables in differences are set to zero.

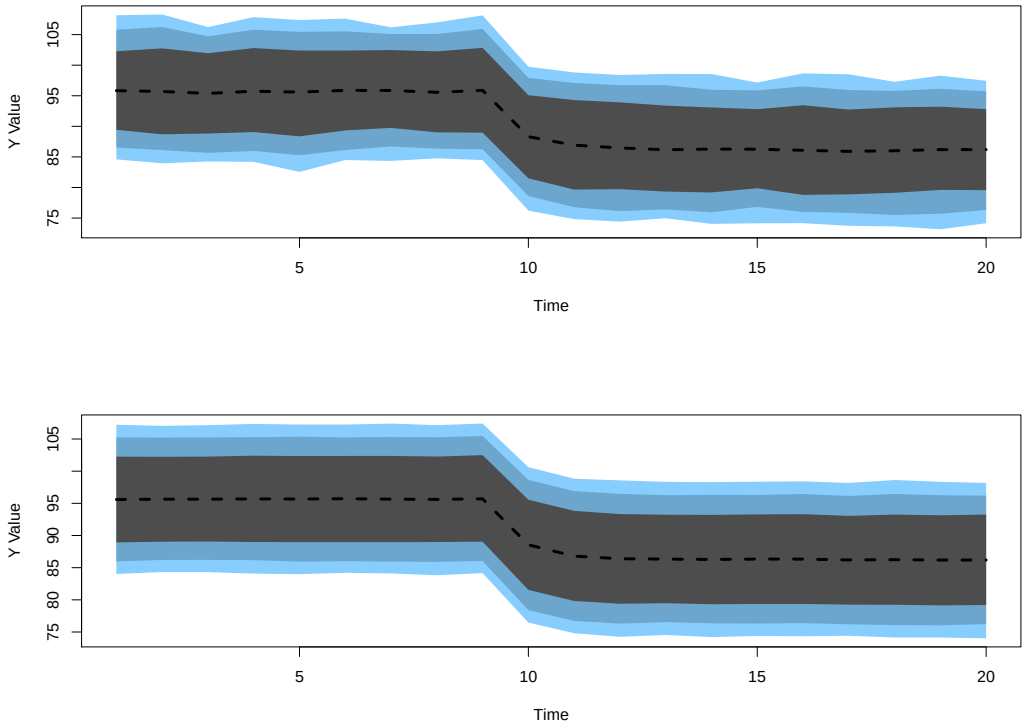

Figure 3: Effect of a 15-percentage-point decrease in Congressional Approval; 1000 simulations (default) (top) and 15000 (bottom)

```
spike.simulation.graph(res6)
```

The results of a 15-percentage-point drop in Congressional Approval on Supreme Court Support are shown in Figure 4. It is clear that there is a relatively quick negative effect on Supreme Court Support in response to a drop in Congressional Approval, and Supreme Court Support remains at around 88 for the rest of the time periods.

Finally, **dynamac** offers a few ancillary functions to help assist in dynamic modeling using auto-regressive distributed lag models and `dynardl()` in particular. These include three basic time-series operators to lag, difference, and lag-difference variables. Users are free to utilize these functions outside of estimating the models.

For lagging variables, we use `lshift(x,l)`, where `x` is the variable to be lagged and `l` is the number of periods to be lagged. If a user wanted a second lag (that is X_{t-2}, he or she would use `lshift(X,2)`:

```
head(ineq$incshare10)
[1] 0.3198 0.3205 0.3198 0.3182 0.3151 0.3175

head(lshift(ineq$incshare10, 2)) # second lag
[1]     NA     NA 0.3198 0.3205 0.3198 0.3182
```

For differencing variables, we use `dshift(x)`, where `x` is the variable to be first-differenced:

```
head(dshift(ineq$incshare10))
[1]      NA  0.0007 -0.0007 -0.0016 -0.0031  0.0024
```

For lag-differencing variables, we use `ldshift(x,l)`, where `x` is the variable to be lag-differenced and `l` is the number of periods for the difference to be lagged. If a user wanted for the difference of X from two time periods ago (ΔX_{t-2}) to be calculated, he or she would use `ldshift(X,2)`:

```
head(ldshift(ineq$incshare10, 2))
[1]      NA      NA      NA  0.0007 -0.0007 -0.0016
```

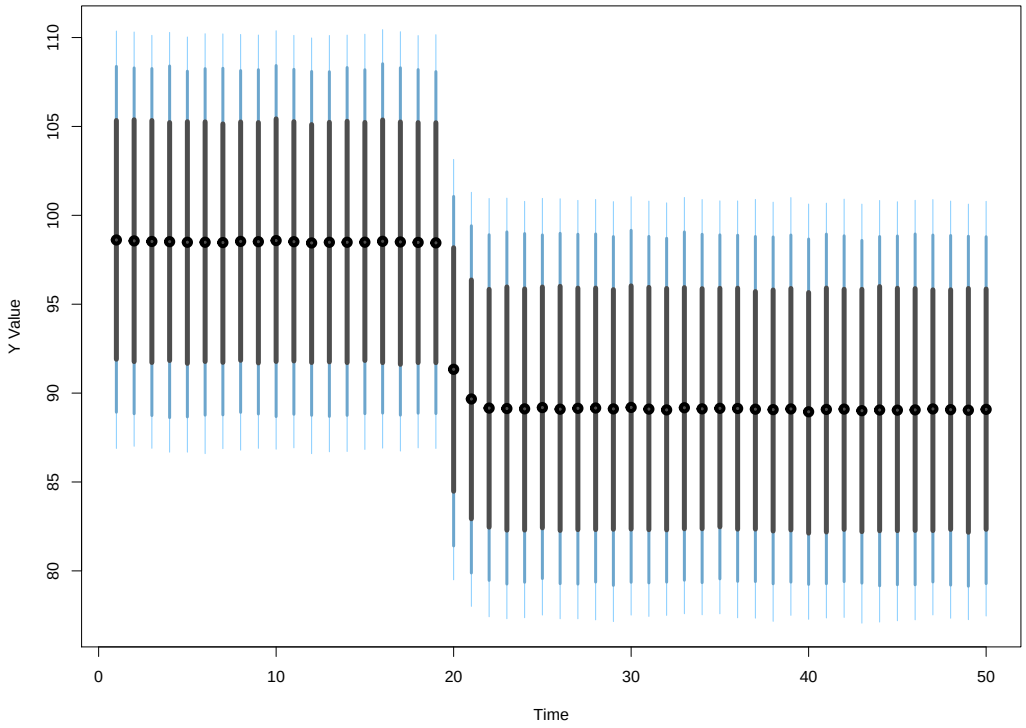

Figure 4: Extending the simulation range and using `forceset`

Conclusion

In this paper we have introduced **dynamac**, a suite of functions to assist applied time series analysts. This centers around `dynardl`, a multipurpose function to model autoregressive distributed lag models. This flexible program models both stationary ARDL relationships as well as cointegrating ones, and it allows users to easily change model specifications (e.g., lags, differences, and lag-differences). We also include two post-estimation diagnostic commands to check for white-noise residuals and cointegration. Last, we have shown the advantages of dynamic simulation for interpreting the substantive significance of the results of single-equation time series models, something users can easily do using **dynamac**.

Bibliography

C. Breunig and M. R. Busemeyer. Fiscal austerity and the trade-off between public investment and social spending. *Journal of European Public Policy*, 19(6):921–938, 2012. URL https://doi.org/10.1080/13501763.2011.614158. [p2, 10]

V. A. Chanley, T. J. Rudolph, and W. M. Rahn. The origins and consequences of public trust in government: A time series analysis. *Public opinion quarterly*, 64(3):239–256, 2000. URL https://doi.org/10.1086/317987. [p1]

C. Choirat, C. Gandrud, J. Honaker, K. Imai, G. King, and O. Lau. *Zelig: Everyone's Statistical Software*, 2018. URL https://CRAN.R-project.org/package=Zelig. R package version 5.1.6. [p2, 10]

H. D. Clarke, K. Ho, and M. C. Stewart. Major's lesser (not minor) effects: prime ministerial approval and governing party support in Britain since 1979. *Electoral Studies*, 19(2):255–273, 2000. URL https://doi.org/10.1016/S0261-3794(99)00051-7. [p1]

S. De Boef and L. Keele. Taking time seriously. *American Journal of Political Science*, 52(1):184–200, 2008. URL https://doi.org/10.1111/j.1540-5907.2007.00307.x. [p1, 2, 10]

R. H. Durr, A. D. Martin, and C. Wolbrecht. Ideological divergence and public support for the Supreme

Court. *American Journal of Political Science*, pages 768–776, 2000. URL https://doi.org/10.2307/2669280. [p12]

R. F. Engle and C. W. Granger. Co-integration and error correction: representation, estimation, and testing. *Econometrica*, 55(2):251–276, 1987. URL https://doi.org/10.2307/1913236. [p3]

C. Gandrud, L. K. Williams, and G. D. Whitten. *dynsim: Dynamic Simulations of Autoregressive Relationships*, 2015. URL https://CRAN.R-project.org/package=dynsim. R package version 1.2.1. [p2, 10]

T. Grant and M. J. Lebo. Error correction methods with political time series. *Political Analysis*, 24(1): 3–30, 2016. URL https://doi.org/10.1093/pan/mpv027. [p6]

S. Johansen. Estimation and hypothesis testing of cointegration vectors in Gaussian vector autoregressive models. *Econometrica: Journal of the Econometric Society*, 59(6):1551–1580, 1991. URL https://doi.org/10.2307/2938278. [p3]

S. Johansen. *Likelihood-based inference in cointegrated vector autoregressive models*. Oxford University Press, 1995. URL https://doi.org/10.1093/0198774508.001.0001. [p3]

S. Jordan and A. Q. Philips. *dynamac: Dynamic Simulation and Testing for Single-Equation ARDL Models*, 2018. URL https://CRAN.R-project.org/package=dynamac. R package version 0.1.4. [p1]

P. K. Narayan. The saving and investment nexus for China: Evidence from cointegration tests. *Applied Economics*, 37(17):1979–1990, 2005. URL https://doi.org/10.1080/00036840500278103. [p7, 9]

M. H. Pesaran, Y. Shin, and R. J. Smith. Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics*, 16(3):289–326, 2001. URL https://doi.org/10.1002/jae.616. [p1, 3, 5, 6, 7, 8, 9]

B. Pfaff, E. Zivot, and M. Stigler. *urca: Unit Root and Cointegration Tests for Time Series Data*, 2016. URL https://CRAN.R-project.org/package=urca. R package version 1.3-0. [p3]

A. Q. Philips. Have your cake and eat it too? Cointegration and dynamic inference from autoregressive distributed lag models. *American Journal of Political Science*, 62(1):230–244, 2018. URL https://doi.org/10.1111/ajps.12318. [p1, 3, 6, 7, 8, 9]

A. Q. Philips, A. Rutherford, and G. D. Whitten. The dynamic battle for pieces of pie—modeling party support in multi-party nations. *Electoral Studies*, 39:264–274, August 2015. URL https://doi.org/10.1016/j.electstud.2015.03.019. [p2, 10]

A. Q. Philips, A. Rutherford, and G. D. Whitten. Dynamic pie: A strategy for modeling trade-offs in compositional variables over time. *American Journal of Political Science*, 60(1):268–283, January 2016. URL https://doi.org/10.1111/ajps.12204. [p2, 10]

B. Ripley. *MASS: Support Functions and Datasets for Venables and Ripley's MASS*, 2018. URL https://CRAN.R-project.org/package=MASS. R package version 7.3-49. [p10]

M. Tomz, J. Wittenberg, and G. King. Clarify: Software for interpreting and presenting statistical results. *Journal of Statistical Software*, 8(1):1–30, 2003. URL https://doi.org/10.18637/jss.v008.i01. [p2, 10]

L. K. Williams and G. D. Whitten. But wait, there's more! Maximizing substantive inferences from TSCS models. *The Journal of Politics*, 74(03):685–693, 2012. URL https://doi.org/10.1017/s0022381612000473. [p2, 10]

C. Wlezien. The public as thermostat: Dynamics of preferences for spending. *American Journal of Political Science*, pages 981–1000, 1995. URL https://doi.org/10.2307/2111666. [p1]

G. Wright. The political implications of American concerns about economic inequality. *Political Behavior*, pages 1–23, 2017. URL https://doi.org/10.1007/s11109-017-9399-3. [p3]

Soren Jordan
Department of Political Science
Auburn University
United States
sorenjordanpols@gmail.com

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