APPENDIX B.
STATA SYNTAX

Many statistical software packages are available to researchers. Because STATA is prominent in the social sciences, we provide STATA-based syntax for readers to use in following our advice on interpreting and presenting results from linear models that include interaction terms.[1]

We advise creating a separate data set that contains simulated values for each of the variables in the regression analysis (it can be used for marginal effects and/or for predicted values, or separate ones can be used for each approach). A data set of simulated values enables the researcher to interpret effects along evenly spaced values of one or more of the variables, within a substantively useful range at which marginal effects, predicted values, and differences in predicted values can easily be interpreted (where the actual data set may not contain evenly spaced values).

Marginal Effects, Standard Errors, and Confidence Intervals

To begin, determine the number of observations that will be contained in the simulation data set. A researcher might want to calculate the estimated marginal effects of x as z ranges from its minimum to its maximum, at

1. This syntax is valid for STATA version 9.

evenly spaced increments (e.g., if the variable z ranges from 1 to 10, and the user wishes z to vary in 1-unit increments, this would imply 10 observations). We advise selecting a manageable number of values (5–100). Open STATA and create a new data set by setting the number of observations, v (e.g., "10"), to be included:

```
set obs v
```

One could manually enter each evenly spaced value into a data set (e.g., 1, 2, 3, etc.) using the data editor. A more efficient way of setting values of z is easily found:

```
egen z = fill(min(unit)max)
```

This command creates a variable **z** that ranges from *min* (e.g., "1") to *max* (e.g., "10") in *unit* increments (e.g., "1"). If, following our government-durability example, z is to range between 40 and 80 and rise by 5-unit increments, then we would need 9 observations, and we would run the following command lines to generate values of z:

```
set obs 9
```

```
egen z = fill(40(5)80)
```

Then, save the data set:

```
save dydxdata.dta
```

After you have created a data set that allows for a range of z values, return to the empirical data:

```
use realdata.dta
```

To estimate the following "standard model," given variables y, x, z and other covariates, w, in the data set:

$$y = \beta_0 + \beta_x x + \beta_z z + \beta_{xz} xz + \beta_w w + \varepsilon$$

Generate the multiplicative term, xz:

```
gen xz = x*z
```

Estimate the linear-regression model:

```
regress y x z xz w
```

Recall that the marginal effects of x and z on y are $\partial\hat{y}/\partial x = \hat{\beta}_x + \hat{\beta}_{xz} z$ and $\partial\hat{y}/\partial z = \hat{\beta}_z + \hat{\beta}_{xz} x$.

Marginal effects are calculated by adding the estimated $\hat{\beta}_x$ to the

product of each value of z with the estimated coefficient $\hat{\beta}_{xz}$. Open the simulation data set to calculate marginal effects:

```
use dydxdata.dta, clear
```

This command will call up the simulation data set and clear the empirical data set. The OLS estimates will remain in memory (although typing "clear" on its own *will* remove the estimates from memory).

One could take the estimated coefficients from the regression output and create a new variable:

```
gen dydx = β̂x + z* β̂xz
```

where the estimated value $\hat{\beta}_x$ from the regression output (e.g., "-2") is entered instead of "$\hat{\beta}_x$" and the estimated value of $\hat{\beta}_{xz}$ from the regression output (e.g., "10") is entered in place of "$\hat{\beta}_{xz}$." This command line would thus create v values corresponding with the marginal effects of x at various values of **z**. A disadvantage to this procedure is that it is possible to mistype the estimated values, creating grave errors in the calculated effects. A less error-prone way of calculating the marginal effects, then, is to use the estimates STATA stores in memory.

STATA stores the estimated coefficient $\hat{\beta}_x$ in memory as _b[x] and the coefficient $\hat{\beta}_{xz}$ in memory as _b[xz], and so a variable that consists of the marginal effects of x as z varies across the evenly incrementing values of z is generated as follows:

```
gen dydx=_b[x]+_b[xz]*z
```

Using the variable dydx, a table or plot of selected marginal effects for evenly spaced values of interest is now easily created.[2]

Presentations of marginal effects should also include an indication of our level of certainty or uncertainty regarding these marginal effects. Recall that the estimated variance of the marginal effects in this example would be $\widehat{V(\partial\hat{y}/\partial x)} = \widehat{V(\hat{\beta}_x)} + z^2\widehat{V(\hat{\beta}_{xz})} + 2z\widehat{C(\hat{\beta}_x,\hat{\beta}_{xz})}$. Calculating $\widehat{V(\partial\hat{y}/\partial x)}$ is straightforward from there. The estimated variance-covari-

2. Users can also take advantage of STATA's programmed postestimation commands. The command lincom will report estimates, standard errors, *t*-statistics, *p*-levels, and a 95 percent confidence interval for any linear combination of coefficients. So, lincom can be used to calculated marginal effects at selected values of *z*: lincom x *zvalue**xz will calculate $\hat{\beta}_x + \hat{\beta}_{xz}z$ at the *z*-value entered into the command line. For a handful of marginal effects, lincom is a useful shortcut; the disadvantage is that the *z*-values must be entered one at a time. If more than a handful of effects are desired (or if graphing is desired), then the procedure outlined in the text will be more serviceable. Alternatively, lincom can be written into a looping program and the results stored in a data set that will allow graphing. Appendix B contains this programming syntax.

ance matrix of estimated coefficients is retrieved in STATA by typing "vce" after an estimation command. The user could then simply generate a new variable by taking the specific values of $\widehat{V(\hat{\beta}_x)}$, $\widehat{V(\hat{\beta}_{xz})}$, and $\widehat{C(\hat{\beta}_x,\hat{\beta}_{xz})}$ acquired from viewing the values in the variance-covariance matrix.

gen vardydx = $\widehat{V(\hat{\beta}_x)}$+z*z*$\widehat{V(\hat{\beta}_{xz})}$+2*z*$\widehat{C(\hat{\beta}_x,\hat{\beta}_{xz})}$

where $\widehat{V(\hat{\beta}_x)}$, $\widehat{V(\hat{\beta}_{xz})}$, and $\widehat{C(\hat{\beta}_x,\hat{\beta}_{xz})}$, would be replaced by their estimated values (e.g., "2").

Again, although this "enter by hand" method is transparent, human error in data entry could be a problem. A less error-prone method uses the estimates that STATA stores in memory. The square root of $\widehat{V(\hat{\beta}_x)}$ is in "_se[x]," and the square root of $\widehat{V(\hat{\beta}_{xz})}$ is in "_se[xz]." The value $\widehat{C(\hat{\beta}_x,\hat{\beta}_{xz})}$ is stored as the element in the row and column corresponding to *x* and *xz* in the estimated variance-covariance matrix of the coefficient estimates, vce. In this particular case, given the order of the variables in the estimated equation, it is in the third row, first column (and, because the variance-covariance matrix is symmetric, also in the first row, third column).

Create a matrix V to represent the variance-covariance matrix of the coefficient estimates, VCE.

```
matrix V = get(VCE)
```

Generate a variable C_x_xz, which contains the covariance of interest, extracted from its position in the matrix V.

```
gen C_x_xz = V[3,1]
```

The estimated variance (and standard error) of each estimated marginal effect can thus be calculated as

```
gen vardydx=(_se[x]^2)+(z*z)*(_se[xz]^2)+2*z*C_x_xz
```

```
gen sedydx=sqrt(vardydx)
```

A table of marginal effects with accompanying standard errors could be generated as follows:

```
tabdisp z, cellvar(dydx sedydx)
```

This command line would present a table featuring all v values of z, with the appropriate marginal effect and standard error of the marginal effect. This table is likely to be useful for the researcher for interpretation,

but for presentational purposes, only a set of selected values of z might be incorporated into an abbreviated table.

Alternatively, marginal effects can be graphed. Recall that confidence intervals can be generated with the following formula: $\partial\hat{y}/\partial x \pm t_{df,p} \sqrt{\widehat{V(\partial\hat{y}/\partial x)}}$. STATA stores the degrees of freedom from the previous estimation as "e(df_r)," and the researcher can use the inverse *t*-distribution function to create $t_{df,p}$. For a 95 percent confidence interval, the lower and upper bounds are calculated as

```
gen LBdydx=dydx-invttail(e(df_r),.025)*sedydx
```

```
gen UBdydx=dydx+invttail(e(df_r),.025)*sedydx
```

This command graphs estimated marginal effects with confidence intervals, along values of z:

```
twoway connected dydx LBdydx UBdydx z
```

These procedures are summarized in table B1.

TABLE B1. STATA Commands for Calculating Marginal Effects of *x* on *y*, Standard Errors of Those Effects, and Confidence Intervals around Those Effects

Procedures	Command Syntax
Create simulation data set: v evenly spaced values for z from its minimum to its maximum. Save the data set.	set obs v egen z = fill(*min*(*unit*)*max*) save dydxdata.dta
Open the original data, generate the multiplicative term, and estimate the linear-regression model.	use realdata.dta gen xz = x*z regress y x z xz w
Open the simulation data set and calculate the estimated marginal effect.	use dydxdata.dta, clear gen dydx = _b [x] + _b [xz] *z
Create a matrix from the estimated covariance matrix of the coefficient estimates, pull out the stored element $C(\hat{\beta}_x, \hat{\beta}_{xz})$, and create a variable containing it.	matrix V = get (VCE) gen C_x_xz = V [3,1]
Calculate the estimated variance (and standard error) of each estimated marginal effect.	gen vardydx = (_se[x] ^2) + (z*z) * (_se[xz] ^2) + 2*z*C_x_xz gen sedydx = sqrt (vardydx)
Generate a table displaying estimated marginal effects and standard errors for all v values of **z**.	tabdisp z, cellvar(dydx sedydx)
Generate lower and upper confidence-interval bounds.	gen LBdydx = dydx-invttail (e(df_r),.025)*sedydx gen UBdydx = dydx + invttail(e(df_r),.025)*sedydx
Graph the estimated marginal effects and the upper and lower confidence intervals along values of **z**.	twoway connected dydx LBdydx UBdydx z

Differences in predicted values can be generated by multiplying the marginal effects calculated previously by Δx (recall that $\Delta y = \Delta x(\hat{\beta}_x + \hat{\beta}_{xz}z_0)$ so long as x enters linearly into the interaction). The estimated variance of these differences in predicted values, similarly, is calculated by multiplying the estimated variance of the estimated marginal effect by Δx^2. For example:

```
gen diffyhat = a*(dydx)
gen vardiffyhat = (a^2)*vardydx
```

where *a* is Δx, (e.g., "2").

Predicted Values, Standard Errors, and Confidence Intervals

Predicted values, $\hat{y}$, are generated by summing the products of the right-hand-side variables, set at particular values, and their corresponding coefficients: $\hat{y} = \mathbf{M}_h\hat{\boldsymbol{\beta}}$, where $\mathbf{M}_h$ is a matrix of values at which x, z, and any other variables in the model are set.

We advise creating a simulation data set that contains the values at which x, z, and any other variables in the model are to be set. Begin by determining the number of observations that will be contained in the data set. A researcher might want to calculate the estimated predicted values as z ranges from its minimum to its maximum, at evenly spaced increments (e.g., if the variable z ranges from 1 to 10, and the user wishes z to vary in 1-unit increments, this would imply 10 observations). We advise selecting a manageable number of values (5–100). Open STATA and create a new data set by setting the number of observations, *v*, to be included:

```
set obs v
```

One could manually enter each evenly spaced value into a data set (e.g., 1, 2, 3, etc.) using the data editor. A more efficient way of setting values of z is easily found:

```
egen z = fill(min(unit)max)
```

This command creates a variable z that ranges from *min* (e.g., "1") to *max* (e.g., "10") in *unit* increments (e.g., "1"). Set the other variables at the desired level, for example, the means, or modes, or substantively interesting values, using the generate command. To generate predicted probabilities based on a model that contains *k* regressors (including the

constant), k total variables must be created. In this example, we create variable x that takes the value *c1* (e.g., "10"), variable w that takes the value *c2* (e.g., "-2"), and variable col1 that takes the value of 1 (later to be multiplied by the intercept).

```
gen x=c1

gen w=c2

gen col1=1
```

Note that each of these variables in the data set will be set at a constant value: the only variable that will vary is z; all other variables (aside from the interaction term) are held constant.[3] Create the interaction term that reflects the values to which x and z are held and save the data set.

```
gen xz=x*z

save yhatdata.dta
```

Open the real data set:

```
use realdata.dta
```

To estimate the following "standard model," given variables y, x, and z in the data set.

$$y = \beta_0 + \beta_x x + \beta_z z + \beta_{xz} xz + \varepsilon$$

Generate the multiplicative term, xz:

```
gen xz = x*z
```

Estimate the linear-regression model:

```
regress y x z xz w
```

Open the simulation data set:

```
use yhatdata.dta, clear
```

This command will call up the simulation data set and clear the empirical data set. The OLS estimates will remain in memory (although typing "clear" on its own *will* remove the estimates from memory).

---

3. To generate several blocks of set values that allow z to range from its minimum to its maximum, but also allow x to take on different values, the user could take advantage of the expand command. Generate the first block of values following the preceding instructions and setting x to x_a. Then expand the data set by two: expand 2. This command line will create a block of v additional observations that will exactly match the first. Then replace the value of x in the new block of observations: replace x=*newvalue* in (v+1)/2v (e.g., replace x = 4 in 11/20).

Assemble the variables into a matrix:[4]

```
mkmat x z xz w col1, matrix(Mh)
```

This command creates matrix Mh, which contains the specified values at which our variables are set: x is fixed at the value *c1*; z varies at regular intervals between some minimum and maximum; the xz correspondingly varies, as it is the product of x (which is held at *c1*) and z (which varies). The covariate w is fixed at *c2*.

Recall that $\hat{y} = \mathbf{M_h}\hat{\boldsymbol{\beta}}$. Although $\hat{\boldsymbol{\beta}}$ is a column vector of coefficients, STATA stores the estimated coefficients as a $1 \times k$ row vector, e(b). So we want to create B, a column vector with $k \times 1$ dimensions, that takes the stored coefficients and transposes them into $\hat{\boldsymbol{\beta}}$:

```
matrix B=e(b)'
```

Calculating the predicted values is simply a matter of multiplying Mh by B:

```
matrix yhat=Mh*B
```

Then convert the resulting column vector into a variable, yhat:

```
svmat yhat, name(yhat)
```

Recall that $\widehat{\mathbf{V}(\hat{\mathbf{y}})} = \mathbf{M_h}\,\widehat{\mathbf{V}[\hat{\boldsymbol{\beta}}]}\mathbf{M_h'}$. STATA stores the estimated variance-covariance matrix of the estimated coefficients, $\widehat{\mathbf{V}[\hat{\boldsymbol{\beta}}]}$, as VCE in its memory. We create a matrix v consisting of $\widehat{\mathbf{V}[\hat{\boldsymbol{\beta}}]}$:

```
matrix V=get(VCE)
```

We can now calculate the variance of the predicted values as follows:

```
matrix VCEYH=Mh*V*Mh'
```

This command creates a matrix, VCEYH, that contains the variances and covariances of the predicted values. The diagonal elements in the variance-covariance matrix of predicted values are those of interest to us, as they correspond with the estimated variance of the predicted values. We want to extract these diagonal elements into a vector:

---

4. A shortcut is provided by the predict command. Estimate the regression on the original data, open the simulated data, and enter predict yhat (bypassing creation of **Mh**). The predict command line generates predicted values using the stored regression coefficients and the values of the variables in the current data set. As long as the variables in the simulation data set have the same name as the variables in the original data set, the predict command line will produce the desired results. Entering predict seyhat, stdp will generate standard errors around the predicted values.

```
matrix VYH= (vecdiag(VCEYH))'
```

The vecdiag command creates a row vector from the diagonal elements of the variance-covariance matrix of the predicted values. Because we want a column vector rather than a row vector, a transpose appears at the end of the command.

We then create a new variable, vyhat, which contains a unique estimated variance to correspond with each predicted value yhat:

```
svmat VYH, name(vyhat)
```

Taking the square root produces the estimated standard error of each predicted value yhat:

```
gen seyhat = sqrt(vyhat)
```

The researcher can next present a table of predicted values with corresponding standard errors:

```
tabdisp z, cellvar(yhat seyhat)
```

Predicted values are effectively displayed when graphed with confidence intervals. The confidence intervals around predicted values $\hat{y}$ can be constructed as $\hat{y} \pm t_{df,p} \sqrt{\widehat{V(\hat{y})}}$, where $\hat{y}$ corresponds with the values in yhat, $\sqrt{\widehat{V(\hat{y})}}$ corresponds with the values in seyhat, and $t_{df,p}$ refers to the relevant critical value from the *t*-distribution. STATA stores the degrees of freedom from the previous estimation as "e(df_r)," and the researcher can utilize the inverse *t*-distribution function to create the multiplier $t_{df,p}$.

For a 95 percent confidence interval, the lower and upper bounds are calculated as follows:

```
gen LByhat=yhat-invttail(e(df_r),.025)*seyhat
```

```
gen UByhat=yhat+invttail(e df_r),.025)*seyhat
```

The predicted values and confidence intervals can be graphed along values of z as follows:

```
twoway connected yhat LByhat UByhat z
```

These procedures are summarized in table B2.

TABLE B2. STATA Commands for Calculating Predicted Values of *y*, Standard Errors for Those Predicted Values, and Confidence Intervals around Those Predicted Values

Procedures	Command Syntax
Create simulation data set that contains *k* total variables. Begin with *v* evenly spaced values for *z* from its minimum to its maximum. Create variables that set the remaining covariates at meaningful values, including a column of 1s for the intercept. Create the interaction term. Save the data set.	set obs *v* egen z = fill(*min*(*unit*)*max*) gen x = *c1* gen w = *c2* gen col1 = 1 gen xz = xz save yhatdata.dta
Open the original data, generate the multiplicative term, and estimate the linear-regression model.	use realdata.dta gen xz = x*z regress y x z xz w
Open the simulation data set and assemble the matrix of set values.	use yhatdata.dta, clear mkmat x z xz w col1, matrix(Mh)
Create a column vector containing the coefficient estimates.	matrix B = e(b)'
Create a column vector of predicted values.	matrix Yhat = Mh*B
Convert the column vector into a variable.	svmat Yhat, name(yhat)
Create a matrix from the estimated covariance matrix of the coefficient estimates.	matrix V = get(VCE)
Calculate the variance of the predicted values.	matrix VCEYH = Mh*V*Mh'
Extract the diagonal elements of the variance-covariance matrix of predicted values into a column vector.	matrix VYH = (vecdiag(VCEYH))'
Convert the column vector into a variable.	svmat VYH, name(vyhat)
Calculate the estimated standard error of each predicted value.	gen seyhat = sqrt(vyhat)
Present a table of predicted values with corresponding standard errors.	tabdisp z, cellvar(yhat seyhat)
Generate lower and upper confidence-interval bounds.	gen LByhat = yhat-invttail(e(df_r),.025)*seyhat gen UByhat = yhat + invttail(e(df_r),.025)*seyhat
Graph the predicted values and the upper and lower confidence intervals along values of *z*.	twoway connected yhat LByhat UByhat z

Marginal Effects, Using "lincom"

The STATA command "lincom" provides a shortcut for calculating marginal effects and their estimated standard errors. It calculates a linear combination of estimators following regression. The disadvantage to "lincom" is that it can be cumbersome to use when the user desires to calculate marginal effects across several values. Here, we provide a looping command that applies lincom across a range of values. In the example, the marginal effects of x are calculated across values of z (for clarity, denoted as zvalues), as z takes values between 0 and 6. The programming loop will post four types of results to a data set called lincomresults.dta: the marginal effect estimates from lincom, the associated standard errors, the value of zvalues applied, and the degrees of freedom in the model (this will be constant throughout, but it helps to have STATA collect it).

```
program define lincomrange
        version 9
        tempname dydx
        postfile 'dydx' dydx sedydx zvalues df using lincomresults, replace
        quietly {
              forvalues z = 0/6 {
                    drop _all
                    use realdata.dta
                    reg y x z xz w
                    lincom x + 'zvalues'*xz
                    post 'dydx' (r(estimate)) (r(se)) ('zvalues') (e(df_r))
                          }
                    }
postclose 'dydx'
end
lincomrange
use lincomresults, clear
tabdisp zvalues, c(dydx sedydx)
gen LBdydx = dydx-invttail (df,.025)*sedydx
gen UBdydx = dydx+invtttail (df,.025)*sedydx
twoway connected dydx LBdydx UBdydx zvalues
```

REFERENCES

Achen, Christopher H. *Interpreting and Using Regression*. Thousand Oaks, CA: Sage, 1982.

Achen, Christopher H. "Two-Step Hierarchical Estimation: Beyond Regression Analysis." *Political Analysis* 13, no. 4 (2005): 447–56.

Allison, Paul D. "Testing for Interaction in Multiple Regression." *American Journal of Sociology* 83, no. 1 (1977): 144–53.

Althauser, Robert P. "Multicollinearity and Non-Additive Regression Models." In *Causal Models in the Social Sciences,* ed. H. M. Blalock, Jr., 453–72. Chicago: Aldine Atherton, 1971.

Amorim Neto, Octavio, and Gary W. Cox. "Electoral Institutions, Cleavage Structures, and the Number of Parties." *American Journal of Political Science* 41, no. 1 (1997): 149–74.

Baron, Reuben M., and David A. Kenny. "The Moderator-Mediator Variable Distinction in Social Psychological Research: Conceptual, Strategic, and Statistical Considerations." *Journal of Personality and Social Psychology* 51, no. 6 (1986): 1173–82.

Beck, Nathaniel, Kristian Gleditsch, and Kyle Beardsley. "Space Is More than Geography: Using Spatial Econometrics in the Study of Political Economy." *International Studies Quarterly* 50, no. 1 (2006): 27–44.

Berry, Frances Stokes, and William D. Berry. "State Lottery Adoptions as Policy Innovations: An Event History Analysis." *American Political Science Review* 84, no. 2 (1990): 395–415.

Berry, Frances Stokes, and William D. Berry. "Specifying a Model of State Policy Innovation." *American Political Science Review* 85, no. 2 (1991): 573–79.

Box-Steffensmeier, Janet M., Suzanna De Boef, and Tse-min Lin. "The Dynamics of the Partisan Gender Gap." *American Political Science Review* 98, no. 3 (2004): 515–28.

Bryk, Anthony S., and Stephen W. Raudenbush. *Hierarchical Linear Models: Applications and Data Analysis Methods*. 2nd ed. Newbury Park, CA: Sage, 2001.