Linear Regression Models with Interaction/Moderation

Rose Medeiros

StataCorp LLC

Stata Webinar
March 19, 2019

Goals

- Learn how to use factor variable notation when fitting models involving
  - Categorical variables
  - Interactions
  - Polynomial terms
- Learn how to use postestimation tools to interpret interactions
  - Tests for group differences
  - Tests of slopes
  - Graphs

A Linear Model

- We'll use data from the National Health and Nutrition Examination Survey (NHANES) for our examples

```
. webuse nhanes2
```

- We'll start with a basic a model for `bmi` using age and sex (`female`).
- Before we fit the model, let's investigate the variables using `codebook`

```
. codebook bmi age female
```

- Now we can fit the model

```
. regress bmi age female
```

Working with Categorical Variables

- We would now like to include `region` in the model, let's take a look at this variable

```
. codebook region
```

  - It cannot simply be added to the list of covariates because it has 4 categories

- To include a categorical variable, put an `i.` in front of its name—this declares the variable to be a categorical variable, or in Stataese, a *factor variable*
- For example, to add `region` to our model we use

```
. regress bmi age i.female i.region
```

Niceities

- Value labels associated with factor variables are displayed in the regression table
- We can tell Stata to show the base categories for our factor variables

```
. set showbaselevels on
```

Factor Notation as Operators

- The `i.` operator can be applied to many variables at once:

```
. regress bmi age i.(female region)
```

- In other words, it understands the distributive property
  - This is useful when using variable ranges, for example
- For the curious, factor variable notation works with wildcards
  - If there were many variables starting with `u`, then `i.u*` would include them all as factor variables

Using Different Base Categories

- By default, the smallest-valued category is the base category
- This can be overridden within commands
  - `b#.` specifies the value # as the base
  - `b(##).` specifies the #'th largest value as the base
  - `b(first).` specifies the smallest value as the base
  - `b(last).` specifies the largest value as the base
  - `b(freq).` specifies the most prevalent value as the base
  - `bn.` specifies there should be no base

Playing with the Base

- We can use `region=3` as the base class on the fly:

```
. regress bmi age i.female b3.region
```

- We can use the most prevalent category as the base

```
. regress bmi age i.female b(freq).region
```

- Factor variables can be distributed across many variables

```
. regress bmi age b(freq).(female region)
```

- The base category can be omitted (with some care here)

```
. regress bmi age i.female bn.region, noconstant
```

- We can also include a term for `region=4` only

```
. regress bmi age i.female 4.region
```

Specifying Interactions

- Factor variables are also used for specifying interactions
    - This is where they really shine
- To include both main effects and interaction terms in a model, put ## between the variables
- To include only the interaction terms, put # between the terms

Categorical by Categorical Interactions

- For example, to fit a model that includes main effects for age, `female`, and `region`, as well as the interaction of `female`, and `region`

```
. regress bmi age female##region
```

- Variables involved in interactions are assumed to be categorical, so no `i.` is needed
- To see all the omitted terms we can add the `allbaselevels` option

```
. regress bmi age female##region, allbaselevels
```

Categorical by Continuous Interactions

- To include continuous variables in interactions use `c.` to specify that a variable is continuous
    - Otherwise it will be assumed to be categorical
- Here is our model with an interaction between age and `region`

```
. regress bmi c.age##region i.female
```

Continuous by Continuous Interactions

- Prefix both variables in the interaction with `c.` to fit models with continuous by continuous variable interactions
- For example, we can interact age with serum vitamin c levels (`vitaminc`)

```
. regress bmi c.age##c.vitaminc i.female i.region
```

- To include polynomial terms, interact a variable with itself
- For example, a model that includes both age and age^2

```
. regress bmi c.age##c.age i.female i.region
```

  - The coefficient for age-squared is next to `c.age#c.age`

Higher Order Interactions

- Factor variable syntax can be used to specify higher order interactions
- If the interactions are specified using ## all lower order terms are included
- For example, here we fit a model for `bmi` using a model that includes the three-way interaction of continuous variables `age` and `vitaminc` and categorical variable `female`

```
. regress bmi c.age##c.vitaminc##female
```

Some Factor Variable Notes

- If you plan to look at marginal effects of any kind, it is best to
  - Explicitly mark all categorical variables with `i.`
  - Specify all interactions using # or ##
  - Specify powers of a variable as interactions of the variable with itself
- There can be up to 8 categorical and 8 continuous interactions in one expression
  - Have fun with the interpretation

Introduction to Postestimation

- In Stata jargon, postestimation commands are commands that can be run after a model is fit, for example
  - Predictions
  - Additional hypothesis tests
  - Checks of assumptions
- We'll explore postestimation tools that can be used to help interpret the results of models that include interactions
- The usefulness of specific tools will depend on the types of hypotheses you wish to examine

Estimating a Model

- Lets begin by running a model with main effects for age, `female` and `region`, and the interaction of `female` and `region`

```
. regress bmi age female##region
```

- How might we begin?
    - Perform joint tests of coefficients
    - Estimate and test hypotheses about group differences

Finding the Coefficient Names

- Some postestimation commands require that you know the names used to store the coefficients
- To see these names we can replay the model and showing the *coefficient legend*

```
. regress, coeflegend
```

- From here, we can see the full specification of the factor levels:

  `_b[2.region]` corresponds to `region=2` which is "MW" or midwest
  `_b[3.region]` corresponds to `region=3` which is "S" or south

- We can also see the terms for the interaction:

  `_b[1.female#2.region]` corresponds to the term for the interaction of `region=2` and `female=1`
  `_b[1.female#3.region]` corresponds to the term for the interaction of `region=3` and `female=1`

Joint Tests

- The `test` command performs a Wald test of the specified null hypothesis
  - The default test is that the listed terms are equal to 0
- `test` takes a list of terms, which may be variable names, but can also be terms associated with factor variables
- To perform a joint test of the null hypothesis that the coefficients for the levels of `region` are all equal to 0

```
. test 2.region 3.region 4.region
```

  - Since the model contains an interaction, this is a test of the effect of `region` when `female=0`

Testing Sets of Coefficients

- To test that all of the coefficients associated with the interaction of `female` and `region` we would need to give the full name of all the coefficients

```
. test 1.female#2.region 1.female#3.region 1.female#4.region
```

- `testparm` also performs Wald tests, but it accepts lists of variables, rather than coefficients in the model
- So we can perform joint tests with less typing, for example

```
. testparm i.region#i.female
```

An Alternative Test

- Likelihood ratio tests provide an alternative method of testing sets of coefficients
- To test the coefficients associated with the interaction of `female` and `region` we need to store our model results. The name is arbitrary, we'll call them `m1`

```
. estimates store m1
```

- Now we can rerun our model without `region`

```
. regress bmi age i.female i.region
```

- If we were removing one of these variables entirely, we would want to add `if e(sample)` to makes sure the same sample, what Stata calls the *estimation sample*, is used for both models

Likelihood Ratio Tests (Continued)

- Now we store the second set of estimates

```
. estimates store m2
```

- And use the `lrtest` command to perform the likelihood ratio test

```
. lrtest m1 m2
```

- We'll restore the results from `m1`

```
. estimates restore m1
```

- Now it's as if we just ran the model stored in `m1`

Tests of Differences

- `test` can also be used to the equality of coefficients

```
. test 3.region#1.female = 4.region#1.female
```

- A likelihood ratio test can also be used; see `help constraint` for information on setting the necessary constraints
- The `lincom` command can be used to calculate linear combinations of coefficients, along with standard errors, hypothesis tests, and confidence intervals
- For example, to obtain the difference in coefficients

```
. lincom 3.region#1.female - 4.region#1.female
```

Contrasts

- The `contrast` command allows us to test a wide variety of comparisons across groups
- For example comparing regions separately for men and women

```
. contrast region@female, effects
```

  - The `@` symbol requests comparisons of the levels of `region` at each value of `female`
  - The `effects` option requests that individual contrasts be displayed along with their standard errors, hypothesis tests, and confidence intervals

Adjusting for Multiple Comparisons

- Use of `contrast` can result in a large number of hypothesis tests
- The `mcompare()` option can be used to adjust p-values and confidence intervals for multiple comparisons within factor variable terms
- The available methods are
  - `noadjust`
  - `bonferroni`
  - `sidak`
  - `scheffe`
- To apply Bonferroni's adjustment to our previous contrast

```
. contrast region@female, effects mcompare(bonferroni)
```

Average Predicted Values

- We might want to explore predictions based on our model and data
- Predictions for individual observations can be made using the `predict` command, see `help predict`
- To find out about our model more generally, we may be more interested in average predicted values
  - Also known as predictive margins or recycled predictions
- To obtain the average predicted value of `bmi`

```
. margins
```

Predictions at Specified Values of Factor Variables

- Stata calls the list of variables that follow the `margins` command the *marginslist*
  - To appear in the *marginslist* a variable must have been specified as factor variable in the model
- To obtain the average predicted value of `bmi` at different values of `region`

```
. margins region
```

- How were these values generated?
  1. Calculate the predicted value of `bmi` setting `region=1` and using each case's observed values of `female` and `age`
  2. Find the mean of the predicted values
  3. Repeat steps 1 and 2 for each value of `region`

Predicted Values with Multiple Factor Variables

- We can obtain `margins` for multiple variables

```
. margins region female
```

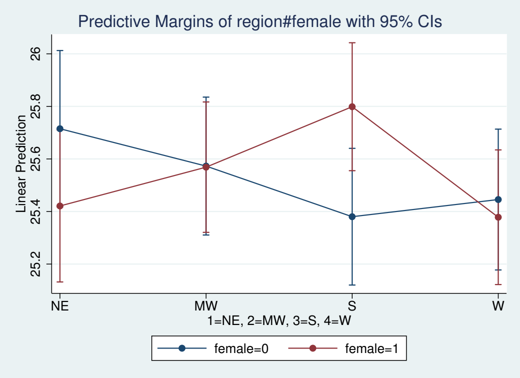
- Or we can oobtain predicted values of `bmi` at each combination of `region` and `female`

```
. margins region#female
```

- We might prefer to graph these results, we can do so using the `marginsplot` command

Graphing Predicted Values

```
. marginsplot
```



- If our model did not include a region by female interaction, the lines would be parallel

Predicted Values for Specific Groups

- When we specify the variables in the *marginslist* Stata calculates predicted values treating each case as though it belonged to each group
- The `over()` option allows us to obtain predictions separately for each group, for example

```
. margins, over(female)
```

- This time the table shows
  - The average predicted value of `bmi` for cases where `female=0` using each case's observed values of `age` and `region`
  - The average predicted value of `bmi` for cases where `female=1` using each case's observed values of `age` and `region`
- This can be useful when we want to compare groups

A Categorical by Continuous Interaction

- For this set of examples, we'll fit a model that includes an interaction between the continuous variable age and the categorical variable `region`

```
. regress bmi c.age##region i.female
```

- Let's take a look at how the coefficients are stored

```
. regress, coeflegend
```

test and testparm

- As before, we can test the null hypothesis that all of the coefficients associated with the interaction of age and region are equal to 0 using testparm

```
. testparm c.age#i.region
```

- We could also use lrtest
- We can test specific hypotheses about the slopes
- For example we might want to test whether the slope of age is significantly different in the south (region=3) versus the west (region=4)

```
. test 3.region#c.age = 4.region#c.age
```

Estimated Slopes

- We can use `lincom` to estimate the slope of age for the south (`region=3`)

  ```
  . lincom c.age + 3.region#c.age
  ```

- We can also use `margins` with the `dydx()` option to calculate the slope of age for each `region`

  ```
  . margins region, dydx(age)
  ```

- The `dydx()` option calculates derivative of the predicted values with respect to the specified variable, also known as the marginal effect

Predictions at Specified Values

- To obtain margins at set values of continuous variables use the `at()` option
- For example, the predicted value of `bmi` at each level of `region` setting age=20

```
. margins region, at(age=20) vsquish
```

  - The `vsquish` option reduces the vertical space in the output
- The `at()` option accepts *numlists* so we aren't restricted to a single value of age

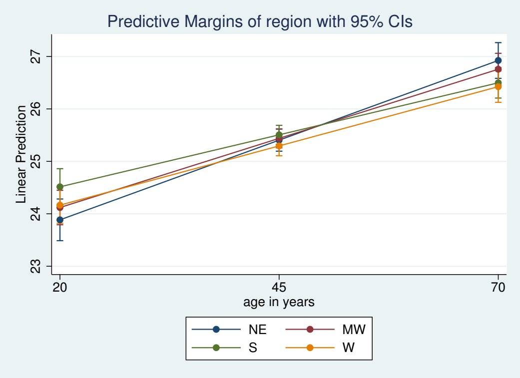
```
. margins region, at(age=(20(25)70)) vsquish
```

  - The observed values of age are from 20 to 74

Graphing Predicted Values

- And we can plot the results

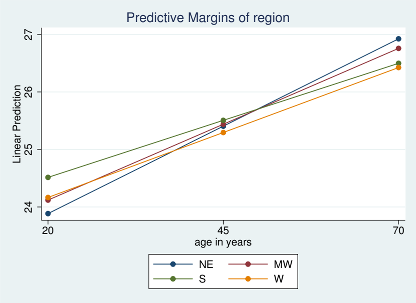
```
. marginsplot
```



Suppressing Confidence Intervals

- The confidence intervals can make the graph appear messy; we can suppress them

```
. marginsplot, noci
```



- This is dangerous because it makes the predictions look more precise than they are

Testing for Differences

- We might want to perform tests of differences at different levels of the continuous variable
- To obtain tests of differences between levels of `region` at each level of `age`

```
. margins region, at(age=(20(10)70)) vsquish contrast
```

Predicted Values Over Groups

- As with *marginslist*, when we specify `at()` Stata calculates predicted values treating each case as though they belong to each group or combination of values
- As before, we can use the `over()` option after models with categorical by continuous interactions
- For example, to obtain predicted values for each `region` using the observed values of `female` and `age` in that `region`

```
. margins, over(region)
```

A Continuous by Continuous Interaction

- For this example we'll use a similar model for `bmi` but we'll add a main effect of serum vitamin c (`vitaminc`), and an interaction between age and `vitaminc`
- Before we fit the model, let's take a closer look at `vitaminc`

```
. summ vitaminc, detail
```

  - The distribution has a long tail, but most observations are between .2 and 2.

- Now lets fit the model

```
. regress bmi c.age##c.vitaminc i.female i.region
```

- We can replay the model using `coeflegend`

```
. regress, coeflegend
```

Estimating Slopes

- We can use `lincom` to calculate the slope for `vitaminc` when `age=49` (it's median)

```
. lincom vitaminc + c.vitaminc#c.age*49
```

- We could also calculate the slope of `age` when `vitaminc=1` (it's median)

```
. lincom age + c.vitaminc#c.age*1
```

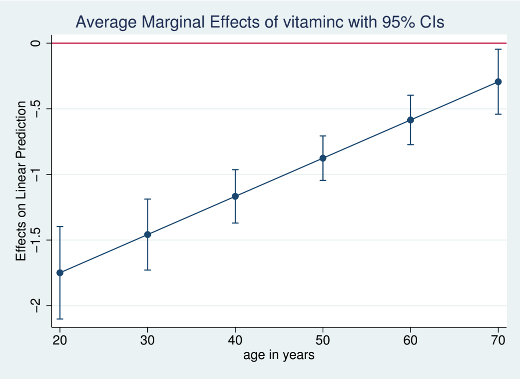
- `margins` can produce estimates of the slopes for a range of values

```
. margins, dydx(vitaminc) at(age=(20(10)70)) vsquish
```

Graphing Slopes

- We can graph the slopes of `vitaminc` across `age`

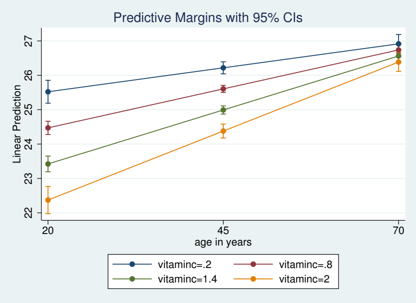
```
. marginsplot, yline(0)
```



Predicted Values

- Specifying multiple variables in the at() option results in predictions at each combination of values

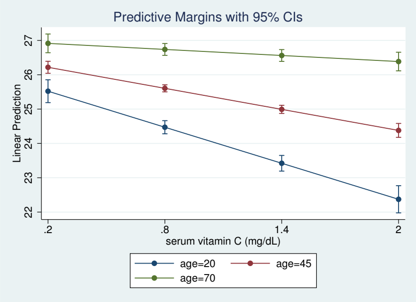
```
. margins , at(age=(20(25)70) vitaminc=(.2(.6)2)) vsquish
. marginsplot
```



Changing the X-axis Variable

- We can select which variable appears on the x-axis using the `xdimension()` option

```
. marginsplot, xdimension(vitaminc)
```



Models with Polynomial Terms

- We'll start by fitting a model that includes age and age^2

```
. regress bmi c.age##c.age i.female i.region
```

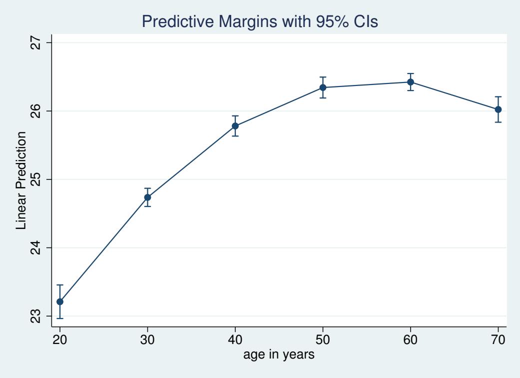
- Graphs can be particularly useful in understanding models with polynomial terms
- Here we predict values of `bmi` at different values of age

```
. margins, at(age=(20(10)70)) vsquish
```

Graphing Predicted Values

- And graph the predictions

```
. marginsplot
```



Slopes

- We can also obtain estimates of the slope of age across its range
- To do so we'll include age in both the `dyed()` and `at()` options

```
. margins, dydx(age) at(age=(20(10)70)) vsquish
```

Adding a Cubic Term

- The same process can be used with higher order polynomials, here we add a cubic term for age

```
. regress bmi c.age##c.age##c.age i.female i.region
```

- As before we can predict slopes at specified values of age

```
. margins, dydx(age) at(age=(20(10)70)) vsquish
```

- Or predict `bmi` at different values of age

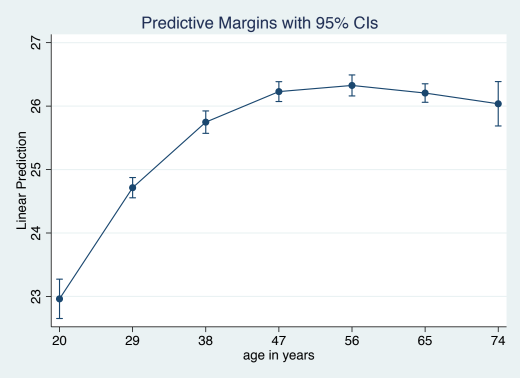
```
. margins, at(age=(20(9)74)) vsquish
```

  - Here, we get predictions across the full range of ages in the dataset (i.e. 20-74)

Graphing the Cubic Term

- And we can easily graph this as well

```
. marginsplot
```

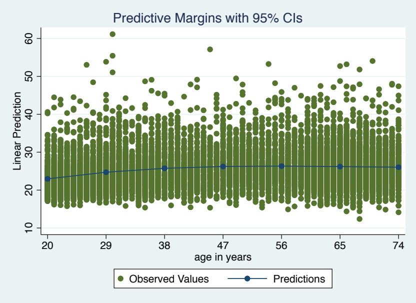


Adding Additional Plots

- We can add other types of `twoway` plots to the plots drawn by `marginsplots`
  - Continuing with our cubic example
- The `addplot` option allows us to add additional plots to our `marginsplots`
- We do want to be careful about the order in which graphs are drawn, we usually want the most dense graphs, for example individual data points, drawn first
  - Specifying `addplot(..., below)` draws the added plot below the `marginsplot`

Adding Observed Data

```
. marginsplot, addplot(scatter bmi age, below ///
    legend(order(3 "Observed Values" 2 "Predictions")) ///
    xlabel(20(9)74))
```



- Note: The confidence intervals are in the plot, they're just small relative to the scale of the y-axis, so they're hard to see.

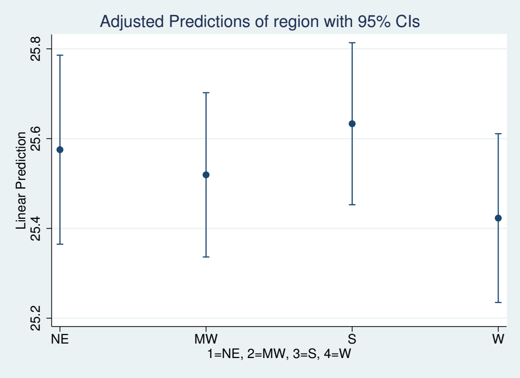
Changing the Plot Type

- We can change the plots drawn by `marginsplot` to another `twoway` plot type
  - See `help twoway` for a list
- The `recast()` option changes the plot for the predictions
  - `recastci()` changes how the CIs are plotted
- Let's run a simple model to demonstrate

```
. regress bmi i.region
. margins region
```

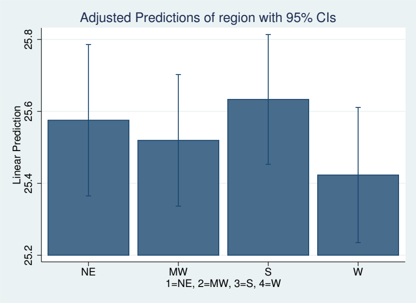
Estimates as a Scatterplot

```
. marginsplot, recast(scatter)
```



Estimates as a Bar plot

```
. marginsplot, recast(bar) plotopts(barwidth(.9))
```



- The `plotopts()` option allows you to specify options for the plots
- `barwidth()` specifies the width of the bars in units of the x variable

Conclusion

- We've seen how to fit models that include interactions
- We've learned how to use Stata's postestimation tools to explore the resulting models
- We've learned how to graph predictions and how to modify those graphs