Interpreting Models for Categorical and Count Outcomes

Rose Medeiros

StataCorp LLC

Stata Webinar
March 21, 2019

Goals

- Learn how to fit models that include categorical variables and/or interactions using factor variable syntax
- Get an overview of tools available for investigating models
- Learn a bit about how Stata partitions model fitting and model testing tasks

A Logistic Regression Model

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

```
. webuse nhanes2
```

- We'll start with a model for high blood pressure (`highbp`) using age, body mass index (`bmi`) and sex (`female`)
- Before we fit the model, let's investigate the variables

```
. codebook highbp age bmi female
```

- Now we can fit the model

```
. logit highbp age bmi female
```

Working with Categorical Variables

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

```
. codebook region
```

- `region` 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

```
. logit highbp age bmi i.female i.region
```

Niceities

- Starting in Stata 13, 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
```

  - This means the base category will always be clearly documented in the output

Factor Notation as Operators

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

```
. logit highbp age bmi 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
- The base can also be permanently changed using `fvset`; see `help fvset` for more information

Playing with the Base

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

```
. logit highbp age bmi i.female b3.region
```

- We can use the most prevalent category as the base

```
. logit highbp age bmi i.female b(freq).region
```

- Factor variables can be distributed across many variables

```
. logit highbp age bmi b(freq).(female region)
```

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

```
. logit highbp age bmi i.female bn.region, noconstant
```

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

```
. logit highbp age bmi 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
- Variables involved in interactions are treated as categorical by default
  - Prefix a variable with `c.` to specify that a variable is continuous
- Here is our model with an interaction between age and `female`

```
. logit highbp bmi c.age##female i.region
```

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 model results
    - The main example here is after `logit` models, but these tools can be used with most estimation commands
- The usefulness of specific tools will depend on the types of hypotheses you wish to examine

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 showing the *coefficient legend*

```
. logit, 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
- The coefficient for the `female` by age interaction is stored as `_b[1.female#c.age]`

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 specify 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
```

Testing Sets of Coefficients

- If you are testing a large number of terms, typing them all out can be laborious
- `testparm` also performs Wald tests, but it accepts lists of variables, rather than coefficients in the model
- For example, to test all coefficients associated with `i.region`

```
. testparm i.region
```

Likelihood Ratio Tests

- Likelihood ratio tests provide an alternative method of testing sets of coefficients
- To test the coefficients associated with `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`

```
. logit highbp bmi c.age##female if e(sample)
```

- Adding `if e(sample)` 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` which includes `region` even though the terms are not collectively significant

```
. estimates restore m1
```

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

Tests of Differences

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

  ```
  . test 3.region = 4.region
  ```

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

  ```
  . lincom 3.region - 4.region
  ```

What are margins?

- Stata defines margins as "statistics calculated from predictions of a previously fit model at fixed values of some covariates and averaging or otherwise integrating over the remaining covariates."
  - Also known as counterfactuals, or when we fix a categorical variable, potential outcomes
- What sorts of predictions does `margins` work with?
  - Predicted means, probabilities, and counts
  - Derivatives
  - Elasticities
- We'll also see contrasts and pairwise comparisons of the above

Average Predictions

- Let's start with `margins` in its most basic form

```
. margins
```

- What happened here?
  1. The predicted probability of `highbp=1` was calculated for each case, using each case's observed values of `bmi`, `age`, `female`, and `region`
  2. The average of those predictions was calculated and displayed
- Unless we tell it to do otherwise, `margins` works with the estimation sample

Predictions at the Average

- An alternative is to calculate the predicted probability fixing all the covariates at some value, often the mean

```
. margins, atmeans
```

- What happened here?
  1. The mean of each independent variable was calculated
  2. The predicted probability of `highbp=1` was calculated using the means from step 1

Predictions at Each Level of a Factor Variable

- Adding a factor variable specifies that the predictions be repeated at each level of the variable, for example

```
. margins region
```

- What happened here?

1. The predicted probability is calculated treating all cases as if `region=1` and using each case's observed values of `bmi`, `age`, and `female`
2. The mean of the predictions from step 1 is calculated
3. Repeat steps 1 and 2 for each value of `region`

Multiple Factor Variables

- We can obtain `margins` for multiple variables

```
. margins region female
```

- Or combinations of values, for example each combination of `region` and `female`

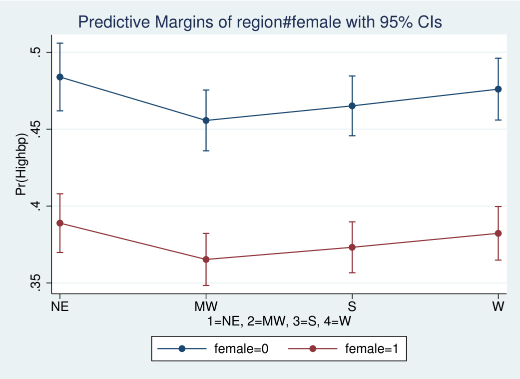
```
. margins region#female
```

- We can graph the resulting predictions using the `marginsplot` command

Graphing Predicted Probabilities

- For example to graph the last set of margins

```
. marginsplot
```



Predictions at Specified Values of Covariates

- The `at()` option is used to specify values at which margins should be calculated
- To obtain the average predicted probability setting age=40 specify

```
. margins, at(age=40)
```

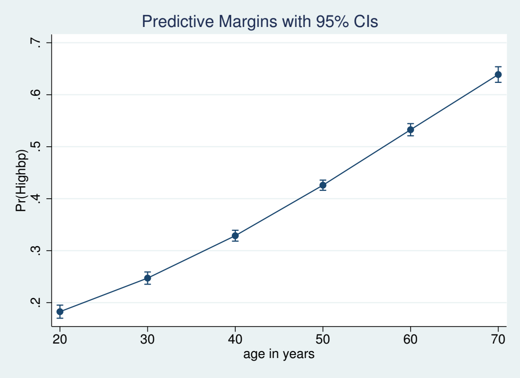
- `at()` accepts number lists, so we can obtain predictions setting age to 20, 30, ..., 70

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

- The `vsquish` option reduces the amount of vertical space the header for `margins` takes up

Graphing Across Values of Continuous Variables

```
. marginsplot
```



Specifying Values of Multiple Variables

- We can specify values of multiple variables using `at()`
- If we set values of all the independent variables in our model, we can ask very specific questions
- For example, what is the predicted probability of high blood pressure for an male who is age 40, with a bmi of 25 and living in the midwest (`region=2`)? What is the predicted probability if the person is female?

```
. margins female, at(age=40 bmi=25 region=2)
```

- We can use the contrast operator `r.` to compare the predicted probabilities for males and females

```
. margins r.female, at(age=40 bmi=25 region=2)
```

- We'll see more on contrasts below

Specifying Ranges of Multiple Variables

- We can also specify ranges of values for multiple variables, for example multiple values of age and bmi

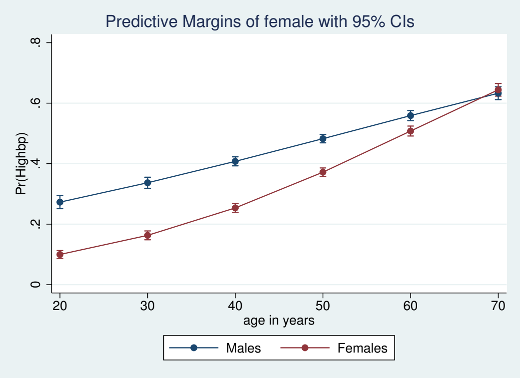
```
. margins, at(age=(20(10)70) bmi=(20(10)40))
```

- We can also combine the use of factor and continuous variables, for example

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

More Plots

```
. marginsplot, legend(order(3 "Males" 4 "Females"))
```



- The standard errors are drawn before the lines for the predictions, so we want the legend to show the third and fourth plots

More Predictions

- We can use at() with the generate() suboption to answer different sorts of questions
- For example, what would the averaged predicted probability be if everyone aged 5 years, while their values female and region remained the same?
- The generate(age+5) requests margins calculated at each observations value of age plus 5

```
.  margins, at(age=generate(age+5))
```

- We can specify at() multiple times, to obtain predictions under different scenarios

```
. margins, at(age=generate(age)) ///
     at(age=generate(age+5)) at(age=generate(age+10))
```

Predictions Over Groups

- The `over()` option produces predictions averaging within groups defined by the factor variable, for example, `female`

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

- What happened here?
  1. The predicted probability for each case is calculated, using the case's observed values on all variables
  2. The average predicted probability is calculated using only cases where `female=0`
  3. Repeat step 2 using only cases where `female=1`

Pairwise Comparisons of Predictions

- Earlier we obtained average predicted probabilities at each level of `region` using

```
. margins region
```

- For pairwise comparisons of these margins we can add the `pwcompare` option

```
. margins region, pwcompare
```

- Adding the `groups` option will allow us to see which levels are statistically distinguishable

```
. margins region, pwcompare(groups)
```

- The `pwcompare()` option can be used to specify other suboptions; see `help margins pwcompare` for more information

Contrasts of Predictions

- The `margins` command allows *contrast operators* which are used to request comparisons of the margins
  - In this case the margins are predicted probabilities
- For example, to compare average predicted probabilities setting `female=0` versus `female=1` add the `r.` prefix

  ```
  . margins r.female
  ```
- We can use the `@` operator to contrast `female` at each level of `region`

  ```
  . margins r.female@region
  ```
- This reports the differences in predicted probabilities when `female=1` versus `female=0` at each level of `region`

Contrasts of Predictions (Continued)

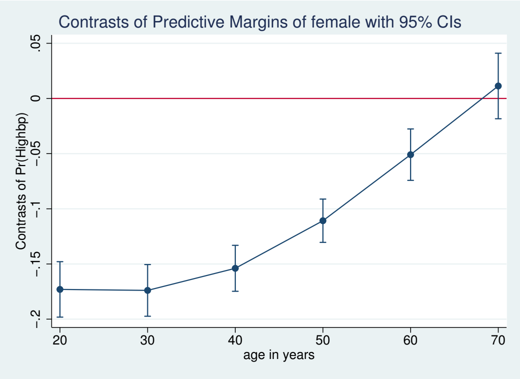
- To perform contrasts at different values of a continuous variable use the `at()` option

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

- The output gives tests of the differences in predicted probabilities for `female=1` versus `female=0` at each of the specified values of age
  - The joint test is statistically significant
  - The differences get smaller in absolute value as age increases

Plotting Contrasts

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



Contrast Operators

- A few common contrast operators are
    - `r.` differences from the base (a.k.a. reference) level
    - `a.` differences from the next (adjacent) level
    - `ar.` differences from the previous level (reverse adjacent)
    - `g.` differences from the balanced grand mean
    - `gw.` differences from the observeration-weighted grand mean
    - There are also operators for Helmert contrats and contrasts using orthogonal polynomials for balanced and unbalanced cases

contrast suboptions

- So far we've obtained contrasts using *contrast operators*, but `margins` also allows a `contrast()` option
- The `contrast()` option is particularly useful for specifying options to `contrast`
- For example, to obtain contrasts for continuous variables the `atcontrast()` suboption is used
    - The `effects` suboption requests a table showing the contrasts along with confidence intervals and p-values
    - In `atcontrast(a)` the `a` contrast operator requests comparisons of adjacent categories

```
. margins, at(age=(20(10)70)) contrast(atcontrast(a) effects) v
```

Contrasts with `generate()`

- Earlier we used the `generate()` suboption to obtain predicted probabilities modifying the observed values
- Specifically, we obtained predicted probabilities using each case's observed value of age and each case's observed value $+5$ years

```
. margins, at(age=generate(age)) at(age=generate(age+5))
```

- Using the `contrast` option, we can compare the two

```
. margins, at(age=generate(age)) ///
      at(age=generate(age+5)) contrast(atcontrast(r))
```

Contrasts of Differences

- We can also request contrasts of contrasts by combining contrast operators
- For example, to compare the differences between males and females across levels of `region` use

```
. margins r.female#r.region
```

Adjusting for Multiple Comparisons

- Use of `contrast` and `pwcompare` 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`

Using mcompare()

- To apply Bonferroni's adjustment to an earlier contrast

```
. margins r.female@region, mcompare(bonferroni)
```

- Specifying adjusted p-values with the pwcompare option

```
. margins region, mcompare(sidak) pwcompare
```

Marginal Effects

- In a straightforward linear model, the marginal effect of a variable is the coefficient b

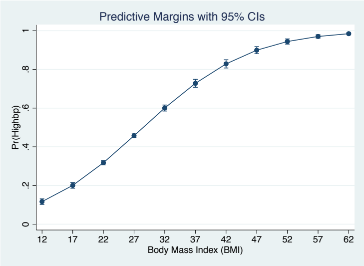
$$y = b_0 + b_1 x_1 + b_2 x_2 + e$$

- In more complex models, this is no longer true
  - models with interactions
  - models with polynomial terms
  - generalized linear models when the margin is not on the linear scale
- For example, in a logistic regression model, the marginal effect of covariates is not constant on the probability scale
- `margins` can be used to estimate the margins of the derivative of a response

A Closer Look at Slopes

- Here is a graph of predicted probabilities across values of `bmi`

```
. margins, at(bmi=(12(5)62))
. marginsplot
```



Average Marginal Effects

- The slope of `bmi` is not constant, but we might want to know what it is on average
- We can obtain the average marginal effect of `bmi`

```
. margins, dydx(bmi)
```

- What happened here?
  1. Calculate the derivative of the predicted probability with respect to `bmi` for each observaton
  2. Calculate the average of derivatives from step 1
- We can do the same for all variables in our model

```
. margins, dydx(*)
```

Marginal Effects Over the Response Surface

- It can also be informative to estimate the marginal effect of x at different values of x
- For example, we can obtain the derviative with respect to age at age=20, 30, ..., 70

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

- Here we do something similar, setting `female=0` and then `female=1`

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

Plots of Marginal Effects

- We can, of course, plot these marginal effects, to see how they change with different values of `female` and age

```
. marginsplot
```

`margins` with Other Estimation Commands

- `margins` works after most estimation commands
- The default prediction for `margins` is the same as the default prediction for `predict` after a given command
- See `help` *command* `postestimation` for information on postestimation commands and their defaults after a given command
- You can specify different predictions from `margins` using the `predict()` option

Modeling Household Size

- For the next set of examples we will model the number of individuals in a household (houssiz) using a Poisson model
- Our model will include covariates age, age^2, region, rural, and a region by rural interaction
- We've been working with age and region but we'll take a look at the new variables

```
. codebook houssiz rural
```

- Now we can fit our model

```
. poisson houssiz i.region##i.rural age c.age#c.age
```

margins after poisson

- predict's default after poisson is the predicted count
- To obtain the average predicted count, using the observed values of all covarites use

```
. margins
```

- As before, we can request predicted counts at specified values of factor variables

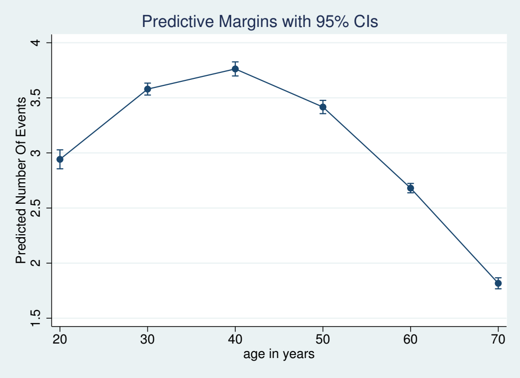
```
. margins region#rural
```

- And continuous variables

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

Plotting Predicted Counts

```
. marginsplot
```



Other Margins

- After `poisson`, `margins` can be used to predict the following
  - `n` number of events; the default
  - `ir` incidence rate, `exp(xb)`, `n` when the exposure variable $= 1$
  - `pr(`*n*`)` probability that `y=n`
  - `pr(`*a*,*b*`)` probability that `a` $\leq$ `y` $\leq$ `b`
  - `xb` the linear predcition
- Predicted probability that `houssiz=1`

```
. margins rural, predict(pr(1))
```

- Predicted probability that $3 \leq$ `houssiz` ≤ 5

```
. margins region#rural, predict(pr(3,5))
```

Multiple Responses

- Starting in Stata 14, `margins` can compute margins for multiple responses at the same time
  - After, for example, `ologit`, `mlogit`, `mvreg`
- To demonstrate this, we'll model self-rated health in a different version of the NHANES dataset

```
. webuse nhanes2f
. codebook health
```

- Our model is

```
. ologit health i.female age c.age#c.age
```

Specifying the Response

- By default `margins` will produce the average predicted probability of each value of `health`

```
. margins
```

- To request a single outcome we can use `predict(outcome(#))`

```
. margins, predict(outcome(2))
```

- For multiple responses from a single command, repeat the `predict()` option

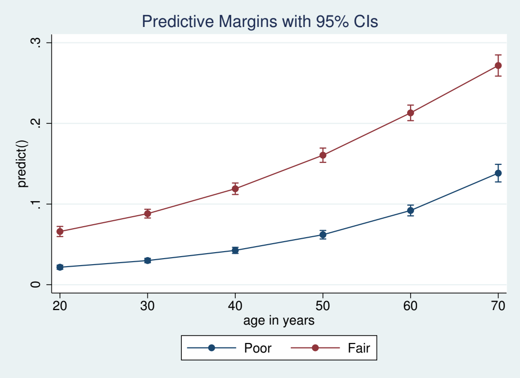
```
. margins, predict(outcome(1)) predict(outcome(2))
```

- To obtain predictions across values of age

```
. margins, at(age=(20(10)70)) pr(out(1)) pr(out(2)) vsquish
```

Plots with Multiple Responses

```
. marginsplot, legend(order(3 "Poor" 4 "Fair"))
```



Conclusion

- We've seen how to obtain a variety of predictions and marginal effects after regression models
- We now know how to perform contrasts of predictions and marginal effects
- We've also seen how to graph these results