25 Working with categorical data and factor variables

Contents

25.1 Continuous, categorical, and indicator variables

Although to Stata a variable is a variable, it is helpful to distinguish among three conceptual types:

- A *continuous variable* measures something. Such a variable might measure a person's age, height, or weight; a city's population or land area; or a company's revenues or costs.
- A *categorical variable* identifies a group to which the thing belongs. You could categorize persons according to their race or ethnicity, cities according to their geographic location, or companies according to their industry. Sometimes, categorical variables are stored as strings.
- An *indicator variable* denotes whether something is true. For example, is a person a veteran, does a city have a mass transit system, or is a company profitable?

Indicator variables are a special case of categorical variables. Consider a variable that records a person's sex. Examined one way, it is a categorical variable. A categorical variable identifies the group to which a thing belongs, and here the thing is a person and the basis for categorization is anatomy. Looked at another way, however, it is an indicator variable. It indicates whether the person is female.

We can use the same logic on any categorical variable that divides the data into two groups. It is a categorical variable because it identifies whether an observation is a member of this or that group; it is an indicator variable because it denotes the truth value of the statement "the observation is in this group".

All indicator variables are categorical variables, but the opposite is not true. A categorical variable might divide the data into more than two groups. For clarity, let's reserve the term *categorical variable*

for variables that divide the data into more than two groups, and let's use the term *indicator variable* for categorical variables that divide the data into exactly two groups.

Stata can convert continuous variables to categorical and indicator variables and categorical variables to indicator variables.

25.1.1 Converting continuous variables to indicator variables

Stata treats logical expressions as taking on the values *true* or *false*, which it identifies with the numbers 1 and 0; see **[U] 13 Functions and expressions**. For instance, if you have a continuous variable measuring a person's age and you wish to create an indicator variable denoting persons aged 21 and over, you could type

```
. generate age21p = age>=21
```

The variable `age21p` takes on the value 1 for persons aged 21 and over and 0 for persons under 21.

Because `age21p` can take on only 0 or 1, it would be more economical to store the variable as a `byte`. Thus it would be better to type

```
. generate byte age21p = age>=21
```

This solution has a problem. The value of `age21` is set to 1 for all persons whose `age` is missing because Stata defines missing to be larger than all other numbers. In our data, we might have no such missing ages, but it still would be safer to type

```
. generate byte age21p = age>=21 if age<.
```

That way, persons whose age is missing would also have a missing `age21p`.

❑ Technical note

Put aside missing values and consider the following alternative to `generate age21p = age>=21` that may have occurred to you:

```
. generate age21p = 1 if age>=21
```

That does not produce the desired result. This statement makes `age21p` 1 (*true*) for all persons aged 21 and above but makes `age21p` missing for everyone else.

If you followed this second approach, you would have to combine it with

```
. replace age21p = 0 if age<21
```

❑

25.1.2 Converting continuous variables to categorical variables

Suppose that you wish to categorize persons into four groups on the basis of their age. You want a variable to denote whether a person is 21 or under, between 22 and 38, between 39 and 64, or 65 and above. Although most people would label these categories 1, 2, 3, and 4, there is really no reason to restrict ourselves to such a meaningless numbering scheme. Let's call this new variable `agecat` and make it so that it takes on the topmost value for each group. Thus persons in the first group will be identified with an `agecat` of 21, persons in the second with 38, persons in the third with 64, and persons in the last (drawing a number out of the air) with 75. Here is a way to create the variable that will work, but it is not the best method for doing so:

```
. use http://www.stata-press.com/data/r14/agexmpl
. generate byte agecat=21 if age<=21
(176 missing values generated)
. replace agecat=38 if age>21 & age<=38
(148 real changes made)
. replace agecat=64 if age>38 & age<=64
(24 real changes made)
. replace agecat=75 if age>64 & age<.
(4 real changes made)
```

We created the categorical variable according to the definition by using the `generate` and `replace` commands. The only thing that deserves comment is the opening `generate`. We (wisely) told Stata to `generate` the new variable `agecat` as a `byte`, thus conserving memory.

We can create the same result with one command using the `recode()` function:

```
. use http://www.stata-press.com/data/r14/agexmpl, clear
. generate byte agecat=recode(age,21,38,64,75)
```

`recode()` takes three or more arguments. It examines the first argument (here `age`) against the remaining arguments in the list. It returns the first element in the list that is greater than or equal to the first argument or, failing that, the last argument in the list. Thus, for each observation, `recode()` asked if `age` was less than or equal to 21. If so, the value is 21. If not, is it less than or equal to 38? If so, the value is 38. If not, is it less than or equal to 64? If so, the value is 64. If not, the value is 75.

Most researchers typically make tables of categorical variables, so we will `tabulate` the result:

```
. tabulate agecat
```

agecat	Freq.	Percent	Cum.
21	28	13.73	13.73
38	148	72.55	86.27
64	24	11.76	98.04
75	4	1.96	100.00
Total	204	100.00	

There is another way to convert continuous variables into categorical variables, and it is even more automated: `autocode()` works like `recode()`, except that all you tell the function is the range and the total number of cells that you want that range broken into:

```
. use http://www.stata-press.com/data/r14/agexmpl, clear
. generate agecat=autocode(age,4,18,65)
. tabulate agecat
```

agecat	Freq.	Percent	Cum.
29.75	82	40.20	40.20
41.5	96	47.06	87.25
53.25	16	7.84	95.10
65	10	4.90	100.00
Total	204	100.00	

In one instruction, we told Stata to break `age` into four evenly spaced categories from 18 to 65. When we `tabulate agecat`, we see the result. In particular, we see that the breakpoints of the four categories are 29.75, 41.5, 53.25, and 65. The first category contains everyone aged 29.75 years or less; the second category contains persons over 29.75 who are 41.5 years old or less; the third category contains persons over 41.5 who are 53.25 years old or less; and the last category contains all persons over 53.25.

❑ Technical note

We chose the range 18–65 arbitrarily. Although you cannot tell from the table above, there are persons in this dataset who are under 18, and there are persons over 65. Those persons are counted in the first and last cells, but we have not divided the age range in the data evenly. We could split the full age range into four categories by obtaining the overall minimum and maximum ages (by typing `summarize`) and substituting the overall minimum and maximum for the 18 and 65 in the `autocode()` function:

```
. use http://www.stata-press.com/data/r14/agexmpl, clear
. summarize age
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         age |        204    31.57353    10.28986         2         66
. generate agecat2=autocode(age,4,2,66)
```

We could also `sort` the data into ascending order of `age` and tell Stata to construct four categories over the range `age[1]` (the minimum) to `age[_N]` (the maximum):

```
. use http://www.stata-press.com/data/r14/agexmpl, clear
. sort age
. generate agecat2=autocode(age,4,age[1],age[_N])
. tabulate agecat2
    agecat2 |      Freq.     Percent        Cum.
------------+-----------------------------------
         18 |         10        4.90        4.90
         34 |        138       67.65       72.55
         50 |         41       20.10       92.65
         66 |         15        7.35      100.00
------------+-----------------------------------
      Total |        204      100.00
```

❑

25.2 Estimation with factor variables

Stata handles categorical variables as factor variables; see [U] **11.4.3 Factor variables**. Categorical variables refer to the variables in your data that take on categorical values, variables such as `sex`, `group`, and `region`. Factor variables refer to Stata's treatment of categorical variables. Factor variables create indicator variables for the levels (categories) of categorical variables and, optionally, for their interactions.

In what follows, the word level means the value that a categorical variable takes on. The variable `sex` might take on levels 0 and 1, with 0 representing male and 1 representing female. We could say that `sex` is a two-level factor variable.

The regressors created by factor variables are called indicators or, more explicitly, virtual indicator variables. They are called virtual because the machinery for factor variables seldom creates new variables in your dataset, even though the indicators will appear just as if they were variables in your estimation results.

To be used as a factor variable, a categorical variable must take on nonnegative integer values. If you have variables with negative values, recode them; see [D] **recode**. If you have string variables, you can use `egen`'s `group()` function to recode them,

```
. egen newcatvar= group(mystringcatvar)
```

If you also specify the `label` option, `egen` will create a value label for the numeric code it produces so that your output will be subsequently more readable:

```
. egen newcatvar= group(mystringcatvar), label
```

Alternatively, you can use `encode` to convert string categorical variables to numeric ones:

```
. encode mystringcatvar, generate(newcatvar)
```

`egen group(), label` and `encode` do the same thing. We tend to use `egen group(), label`. See [D] **egen** and [D] **encode**.

In the unlikely event that you have a noninteger categorical variable, use the `egen` solution. More likely, however, is that you need to read **[U] 25.1.2 Converting continuous variables to categorical variables**.

❑ Technical note

If you should ever need to create your own indicator variables from a string or numeric variable—and it is difficult to imagine why you would—type

```
. tabulate var, gen(newstub)
```

Typing that will create indicator variables named *newstub*1, *newstub*2, . . . ; see [R] **tabulate oneway**. ❑

We will be using linear regression in the examples that follow just because it is so easy to explain and to interpret. We could, however, just as well have used logistic regression, Heckman selectivity, or even Cox proportional-hazards regression with shared frailties. Stata's factor-variable features work with nearly every estimation command.

25.2.1 Including factor variables

The fundamental building block of factor variables is the treatment of each factor variable as if it represented a collection of indicators, with one indicator for each level of the variable. To treat a variable as a factor variable, you add `i.` in front of the variable's name:

```
. use http://www.stata-press.com/data/r14/fvex, clear
(Artificial factor variables' data)
. regress y  i.group age
```

Source	SS	df	MS	Number of obs	=	3,000
				F(3, 2996)	=	31.67
Model	42767.8126	3	14255.9375	Prob > F	=	0.0000
Residual	1348665.19	2,996	450.155272	R-squared	=	0.0307
				Adj R-squared	=	0.0298
Total	1391433.01	2,999	463.965657	Root MSE	=	21.217

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
group						
2	-2.395169	.9497756	-2.52	0.012	-4.257447	-.5328905
3	.2966833	1.200423	0.25	0.805	-2.057054	2.65042
age	-.318005	.039939	-7.96	0.000	-.3963157	-.2396943
_cons	83.2149	1.963939	42.37	0.000	79.3641	87.06571

In these data, variable `group` takes on the values 1, 2, and 3.

Because we typed

```
. regress y  i.group age
```

rather than

```
. regress y    group age
```

instead of fitting the regression as a continuous function of `group`'s values, `regress` fit the regression on indicators for each level of group included as a separate covariate. In the left column of the coefficient table in the output, the numbers 2 and 3 identify the coefficients that correspond to the values of 2 and 3 of the `group` variable. Using the more precise terminology of **[U] 11.4.3 Factor variables**, the coefficients reported for 2 and 3 are the coefficients for virtual variables `2.group` and `3.group`, the indicator variables for `group` = 2 and group = 3.

If `group` took on the values 2, 10, 11, and 125 rather than 1, 2, and 3, then we would see 2, 10, 11, and 125 below `group` in the table, corresponding to virtual variables `2.group`, `10.group`, `11.group`, and `125.group`.

We can use as many sets of indicators as we need in a varlist. Thus we can type

```
. regress y  i.group i.sex i.arm ...
```

25.2.2 Specifying base levels

In the above results, `group` = 1 was used as the base level and `regress` omitted reporting that fact in the output. Somehow, you are just supposed to know that, and usually you do. We can see base levels identified explicitly, however, if we specify the `baselevels` option, either at the time we estimate the model or, as we do now, when we replay results:

```
. regress, baselevels
      Source |       SS           df       MS      Number of obs   =     3,000
-------------+----------------------------------   F(3, 2996)      =     31.67
       Model |  42767.8126         3  14255.9375   Prob > F        =    0.0000
    Residual |  1348665.19     2,996  450.155272   R-squared       =    0.0307
-------------+----------------------------------   Adj R-squared   =    0.0298
       Total |  1391433.01     2,999  463.965657   Root MSE        =    21.217

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       group |
           1 |          0  (base)
           2 |  -2.395169   .9497756    -2.52   0.012    -4.257447   -.5328905
           3 |   .2966833   1.200423     0.25   0.805    -2.057054    2.65042
             |
         age |  -.318005    .039939    -7.96   0.000    -.3963157   -.2396943
       _cons |   83.2149    1.963939    42.37   0.000     79.3641    87.06571
------------------------------------------------------------------------------
```

The smallest value of the factor variable is used as the base by default. Using the notation explained in [U] **11.4.3.2 Base levels**, we can request another base level, such as `group` = 2, by typing

```
. regress y  ib2.group age
```

or, such as the largest value of `group`,

```
. regress y  ib(last).group age
```

Changing the base does not fundamentally alter the estimates in the sense that predictions from the model would be identical no matter which base levels we use. Changing the base does change the interpretation of coefficients. In the regression output above, the reported coefficients measure the differences from `group` = 1. Group 2 differs from group 1 by −2.4, and that difference is significant at the 5% level. Group 3 is not significantly different from group 1.

If we fit the above using `group` = 3 as the base,

```
. regress y  ib3.group age
(output omitted)
```

the coefficients on `group` = 1 and `group` = 2 would be −0.297 and −2.692. Note that the difference between group 2 and group 1 would still be $-2.692-(-0.296)=-2.4$. Results may look different, but when looked at correctly, they are the same. Similarly, the significance of `group` = 2 would now be 0.805 rather than 0.012, but that is because what is being tested is different. In the output above, the test against 0 is a test of whether group 2 differs from group 1. In the output that we omit, the test is whether group 2 differs from group 3. If, after running the `ib3.group` specification, we were to type

```
. test 2.group = 1.group
```

we would obtain the same 0.012 result. Similarly, after running the shown result, if we typed `test 3.group = 1.group`, we would obtain 0.805.

25.2.3 Setting base levels permanently

As explained directly above, you can temporarily change the base level by using the `ib.` operator; also see [U] **11.4.3.2 Base levels**. You can change the base level permanently by using the `fvset` command; see [U] **11.4.3.3 Setting base levels permanently**.

25.2.4 Testing significance of a main effect

In the example we have been using,

```
. use http://www.stata-press.com/data/r14/fvex
. regress y i.group age
```

many disciplines refer to the coefficients on the set of indicators for `i.group` as a main effect. Because we have no interactions, the main effect of `i.group` refers to the effect of the levels of `group` taken as a whole. We can test the significance of the indicators by using `contrast` (see [R] **contrast**):

```
. contrast group
Contrasts of marginal linear predictions
Margins      : asbalanced
```

	df	F	P>F
group	2	4.89	0.0076
Denominator	2996		

When we specify the name of a factor variable used in the previous estimation command in the `contrast` command, it will perform a joint test on the effects of that variable. Here we are testing whether the coefficients for the group indicators are jointly zero. We reject the hypothesis.

25.2.5 Specifying indicator (dummy) variables as factor variables

We are using the model

```
. use http://www.stata-press.com/data/r14/fvex
. regress y  i.group age
```

We are going to add `sex` to our model. Variable `sex` is a 0/1 variable in our data, a type of variable we call an indicator variable and which many people call a dummy variable. We could type

```
. regress y    sex i.group age
```

but we are going to type

```
. regress y  i.sex i.group age
```

It is better to include indicator variables as factor variables, which is to say, to include indicator variables with the `i.` prefix.

You will obtain the same estimation results either way, but by specifying `i.sex` rather than `sex`, you will communicate to postestimation commands that care that `sex` is not a continuous variable, and that will save you typing later should you use one of those postestimation commands. `margins` (see [R] **margins**) is an example of a postestimation command that cares.

Below we type `regress y i.sex i.group age`, and we will specify the `baselevels` option just to make explicit how `regress` is interpreting our request. Ordinarily, we would not specify the `baselevels` option.

```
. regress y  i.sex i.group age, baselevels
```

Source	SS	df	MS	Number of obs	=	3,000
				F(4, 2995)	=	136.51
Model	214569.509	4	53642.3772	Prob > F	=	0.0000
Residual	1176863.5	2,995	392.942737	R-squared	=	0.1542
				Adj R-squared	=	0.1531
Total	1391433.01	2,999	463.965657	Root MSE	=	19.823

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
sex						
male	0	(base)				
female	18.44069	.8819175	20.91	0.000	16.71146	20.16991
group						
1	0	(base)				
2	5.178636	.9584485	5.40	0.000	3.299352	7.057919
3	13.45907	1.286127	10.46	0.000	10.93729	15.98085
age	-.3298831	.0373191	-8.84	0.000	-.4030567	-.2567094
_cons	68.63586	1.962901	34.97	0.000	64.78709	72.48463

As with all factor variables, by default the first level of `sex` serves as its base, so the coefficient 18.4 measures the increase in `y` for `sex` $= 1$ as compared with `sex` $= 0$. In these data, `sex` $= 1$ represents females and `sex` $= 0$ represents males.

Notice that in the above output male and female were displayed rather than 0 and 1. The variable `sex` has the value label `sexlab` associated with it, so Stata used the value label in its output. Stata has three options, `nofvlabel`, `fvwrap(#)`, and `fvwrapon(word | width)`, that control how factor-variable value labels are displayed; see [R] **estimation options**.

25.2.6 Including interactions

We are using the model

```
. use http://www.stata-press.com/data/r14/fvex
. regress y  i.sex i.group age
```

If we are not certain that the levels of `group` have the same effect for females as they do for males, we should add to our model interactions for each combination of the levels in `sex` and `group`. We would need to add indicators for

`sex` $=$ male	and	`group` $= 1$
`sex` $=$ male	and	`group` $= 2$
`sex` $=$ male	and	`group` $= 3$
`sex` $=$ female	and	`group` $= 1$
`sex` $=$ female	and	`group` $= 2$
`sex` $=$ female	and	`group` $= 3$

Doing this would allow each combination of `sex` and `group` to have a different effect on `y`.

Interactions like those listed above are produced using the `#` operator. We could type

```
. regress y  i.sex i.group i.sex#i.group age
```

The # operator assumes that the variables on either side of it are factor variables, so we can omit the i. prefixes and obtain the same result by typing

```
. regress y  i.sex i.group sex#group age
```

We must continue to specify the prefix on the main effects i.sex and i.group, however.

In the output below, we add the allbaselevels option to that. The allbaselevels option is much like baselevels, except allbaselevels lists base levels in interactions as well as in main effects. Specifying allbaselevels will make the output easier to understand the first time, and after that, you will probably never specify it again.

```
. regress y  i.sex i.group sex#group age, allbaselevels
```

Source	SS	df	MS		
Model	217691.706	6	36281.9511	Number of obs	= 3,000
Residual	1173741.3	2,993	392.162145	F(6, 2993)	= 92.52
				Prob > F	= 0.0000
Total	1391433.01	2,999	463.965657	R-squared	= 0.1565
				Adj R-squared	= 0.1548
				Root MSE	= 19.803

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
sex						
male	0	(base)				
female	21.71794	1.490858	14.57	0.000	18.79473	24.64115
group						
1	0	(base)				
2	8.420661	1.588696	5.30	0.000	5.305615	11.53571
3	16.47226	1.6724	9.85	0.000	13.19309	19.75143
sex#group						
male#1	0	(base)				
male#2	0	(base)				
male#3	0	(base)				
female#1	0	(base)				
female#2	-4.658322	1.918195	-2.43	0.015	-8.419436	-.8972081
female#3	-6.736936	2.967391	-2.27	0.023	-12.55527	-.9186038
age	-.3305546	.0373032	-8.86	0.000	-.4036972	-.2574121
_cons	65.97765	2.198032	30.02	0.000	61.66784	70.28745

Look at the sex#group term in the output. There are six combinations of sex and group, just as we expected. That four of the cells are labeled base and that only two extra coefficients were estimated should not surprise us, at least after we think about it. There are 3×2 sex#age groups, and thus $3 \times 2 = 6$ means to be estimated, and we indeed estimated six coefficients, including a constant, plus a seventh for continuous variable age. Now look at which combinations were treated as base. Treated as base were all combinations that were the base of sex, plus all combinations that were the base of group. The combination of sex $= 0$ (male) and group $= 1$ was omitted for both reasons, and the other combinations were omitted for one or the other reason.

We entered a two-way interaction between sex and group. If we believed that the effects of sex#group were themselves dependent on the treatment arm of an experiment, we would want the three-way interaction, which we could obtain by typing sex#group#arm. Stata allows up to eight-way interactions among factor variables and another eight-ways of interaction among continuous covariates.

❑ Technical note

The virtual variables associated with the interaction terms have the names `1.sex#2.group` and `1.sex#3.group`.

❑

25.2.7 Testing significance of interactions

We are using the model

```
. use http://www.stata-press.com/data/r14/fvex
. regress y  i.sex i.group sex#group age
```

We can test the overall significance of the `sex#group` interaction by typing

```
. contrast sex#group
Contrasts of marginal linear predictions
Margins      : asbalanced
```

	df	F	P>F
sex#group	2	3.98	0.0188
Denominator	2993		

We can type the interaction term to be tested—`sex#group`—in the same way as we typed it to include it in the regression. The interaction is significant beyond the 5% level. That is not surprising because both interaction indicators were significant in the regression.

25.2.8 Including factorial specifications

We have the model

```
. use http://www.stata-press.com/data/r14/fvex
. regress y  i.sex i.group sex#group age
```

The above model is called a factorial specification with respect to `sex` and `group` because `sex` and `group` appear by themselves and an interaction. Were it not for `age` being included in the model, we could call this model a full-factorial specification. In any case, Stata provides a shorthand for factorial specifications. We could fit the model above by typing

```
. regress y  sex##group age
```

When you type *A*`##`*B*, Stata takes that to mean *A B A*`#`*B*.

When you type *A*`##`*B*`##`*C*, Stata takes that to mean *A B C A*`#`*B A*`#`*C B*`#`*C A*`#`*B*`#`*C*.

And so on. Up to eight-way interactions are allowed.

The `##` notation is just a shorthand. Estimation results are unchanged. This time we will not specify the `allbaselevels` option:

```
. regress y  sex##group age
      Source |       SS           df       MS      Number of obs   =     3,000
-------------+----------------------------------   F(6, 2993)      =     92.52
       Model |  217691.706         6  36281.9511   Prob > F        =    0.0000
    Residual |  1173741.3      2,993  392.162145   R-squared       =    0.1565
-------------+----------------------------------   Adj R-squared   =    0.1548
       Total |  1391433.01     2,999  463.965657   Root MSE        =    19.803

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |
      female |   21.71794   1.490858    14.57   0.000     18.79473    24.64115
             |
       group |
           2 |   8.420661   1.588696     5.30   0.000     5.305615    11.53571
           3 |   16.47226     1.6724     9.85   0.000     13.19309    19.75143
             |
   sex#group |
  female#2   |  -4.658322   1.918195    -2.43   0.015    -8.419436   -.8972081
  female#3   |  -6.736936   2.967391    -2.27   0.023    -12.55527   -.9186038
             |
         age |  -.3305546   .0373032    -8.86   0.000    -.4036972   -.2574121
       _cons |   65.97765   2.198032    30.02   0.000     61.66784    70.28745
------------------------------------------------------------------------------
```

25.2.9 Including squared terms and polynomials

`#` may be used to interact continuous variables if you specify the `c.` indicator in front of them. The command

```
. regress y  age c.age#c.age
```

fits `y` as a quadratic function of age. Similarly,

```
. regress y  age c.age#c.age c.age#c.age#c.age
```

fits a third-order polynomial.

Using the `#` operator is preferable to generating squared and cubed variables of `age` because when `#` is used, Stata understands the relationship between `age` and `c.age#c.age` and `c.age#c.age#c.age`. Postestimation commands can take advantage of this to produce smarter answers; see, for example, *Requirements for model specification* in [R] **margins**.

25.2.10 Including interactions with continuous variables

`#` and `##` may be used to create interactions of categorical variables with continuous variables if the continuous variables are prefixed with `c.`, such as `sex#c.age` in

```
. regress y  i.sex age sex#c.age
. regress y  sex##c.age
. regress y  i.sex sex#c.age
```

The result of fitting the first of these models (equivalent to the second) is shown below. We include `allbaselevels` to make results more understandable the first time you see them.

```
. regress y  i.sex age sex#c.age, allbaselevels
      Source |       SS           df       MS      Number of obs   =     3,000
-------------+----------------------------------   F(3, 2996)      =    139.91
       Model |  170983.675         3  56994.5583   Prob > F        =    0.0000
    Residual |  1220449.33     2,996  407.35959   R-squared       =    0.1229
-------------+----------------------------------   Adj R-squared   =    0.1220
       Total |  1391433.01     2,999  463.965657   Root MSE        =    20.183

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |
        male |          0  (base)
      female |   14.92308   2.789012     5.35   0.000     9.454508    20.39165
             |
         age |  -.4929608   .0480944   -10.25   0.000    -.5872622   -.3986595
             |
     sex#c.age |
        male |          0  (base)
      female |  -.0224116   .0674167    -0.33   0.740    -.1545994    .1097762
             |
       _cons |   82.36936   1.812958    45.43   0.000      78.8146    85.92413
------------------------------------------------------------------------------
```

The coefficient on the interaction (-0.022) is the difference in the slope of `age` for females (`sex` $= 1$) as compared with the slope for males. It is far from significant at any reasonable level, so we cannot distinguish the two slopes.

A different but equivalent parameterization of this model would be to omit the main effect of age, the result of which would be that we would estimate the separate slope coefficients of age for males and females:

```
. regress y  i.sex sex#c.age
      Source |       SS           df       MS      Number of obs   =     3,000
-------------+----------------------------------   F(3, 2996)      =    139.91
       Model |  170983.675         3  56994.5583   Prob > F        =    0.0000
    Residual |  1220449.33     2,996  407.35959   R-squared       =    0.1229
-------------+----------------------------------   Adj R-squared   =    0.1220
       Total |  1391433.01     2,999  463.965657   Root MSE        =    20.183

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         sex |
      female |   14.92308   2.789012     5.35   0.000     9.454508    20.39165
             |
     sex#c.age |
        male |  -.4929608   .0480944   -10.25   0.000    -.5872622   -.3986595
      female |  -.5153724   .0472435   -10.91   0.000    -.6080054   -.4227395
             |
       _cons |   82.36936   1.812958    45.43   0.000      78.8146    85.92413
------------------------------------------------------------------------------
```

It is now easier to see the slopes themselves, although the test of the equality of the slopes no longer appears in the output. We can obtain the comparison of slopes by using the `lincom` postestimation command:

```
. lincom 1.sex#c.age - 0.sex#c.age
 ( 1)  - 0b.sex#c.age + 1.sex#c.age = 0
```

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
(1)	-.0224116	.0674167	-0.33	0.740	-.1545994	.1097762

As noted earlier, it can be difficult at first to know how to refer to individual parameters when you need to type them on postestimation commands. The solution is to replay your estimation results specifying the `coeflegend` option:

```
. regress, coeflegend
```

Source	SS	df	MS		
Model	170983.675	3	56994.5583	Number of obs	= 3,000
Residual	1220449.33	2,996	407.35959	F(3, 2996)	= 139.91
				Prob > F	= 0.0000
Total	1391433.01	2,999	463.965657	R-squared	= 0.1229
				Adj R-squared	= 0.1220
				Root MSE	= 20.183

y	Coef.	Legend
sex		
female	14.92308	_b[1.sex]
sex#c.age		
male	-.4929608	_b[0b.sex#c.age]
female	-.5153724	_b[1.sex#c.age]
_cons	82.36936	_b[_cons]

The legend suggests that we type

```
. lincom _b[1.sex#c.age] - _b[0b.sex#c.age]
```

instead of `lincom 1.sex#c.age - 0.sex#c.age`. That is, the legend suggests that we bracket terms in `_b[]` and explicitly recognize base levels. The latter does not matter. Concerning bracketing, some commands allow you to omit brackets, and others do not. All commands will allow bracketing, which is why the legend suggests it.

25.2.11 Parentheses binding

Factor-variable operators can be applied to groups of variables if those variables are bound in parentheses. For instance, you can type

```
. regress y  sex##(group c.age c.age#c.age)
```

rather than

```
. regress y  i.sex i.group sex#group age sex#c.age c.age#c.age sex#c.age#c.age
```

Parentheses may be nested. The parenthetically bound notation does not let you specify anything you could not specify without it, but it can save typing and, as importantly, make what you type more understandable. Consider

```
. regress y  i.sex i.group sex#group age sex#c.age c.age#c.age sex#c.age#c.age
. regress y  sex##(group c.age c.age#c.age)
```

The second specification is shorter and easier to read. We can see that all the covariates have different parameters for males and females.

25.2.12 Including indicators for single levels

Consider the following regression of statewide marriage rates (marriages per 100,000) on the median age in the state of the United States:

```
. use http://www.stata-press.com/data/r14/censusfv
(1980 Census data by state)
. regress marriagert  medage
```

Source	SS	df	MS	Number of obs	=	50
				F(1, 48)	=	0.00
Model	148.944706	1	148.944706	Prob > F	=	0.9949
Residual	173402855	48	3612559.48	R-squared	=	0.0000
				Adj R-squared	=	-0.0208
Total	173403004	49	3538836.82	Root MSE	=	1900.7

marriagert	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
medage	1.029541	160.3387	0.01	0.995	-321.3531	323.4122
_cons	1301.307	4744.027	0.27	0.785	-8237.199	10839.81

There appears to be no effect of median age. We know, however, that couples from around the country flock to Nevada to be married in Las Vegas, which biases our results. We would like to add a single indicator for the state of Nevada. We `describe` our data, see the value label for state is `st`, and then type `label list st` to discover the label for Nevada. We find it is 30; thus we can now type

```
. regress marriagert  medage i30.state
```

Source	SS	df	MS	Number of obs	=	50
				F(2, 47)	=	2311.15
Model	171657575	2	85828787.6	Prob > F	=	0.0000
Residual	1745428.85	47	37136.784	R-squared	=	0.9899
				Adj R-squared	=	0.9895
Total	173403004	49	3538836.82	Root MSE	=	192.71

marriagert	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
medage	-61.23095	16.2825	-3.76	0.000	-93.98711	-28.47479
state						
Nevada	13255.81	194.9742	67.99	0.000	12863.57	13648.05
_cons	2875.366	481.5533	5.97	0.000	1906.606	3844.126

These results are more reasonable.

There is a subtlety to specifying individual levels. Let's add another indicator, this time for California. The following will not produce the desired results, and we specify the `baselevels` option to help you understand the issue. First, however, here is the result:

```
. regress marriagert  medage i5.state i30.state, baselevels
      Source |       SS           df       MS      Number of obs   =        50
-------------+----------------------------------   F(2, 47)        =   2311.15
       Model |  171657575          2  85828787.6   Prob > F        =    0.0000
    Residual |  1745428.85         47   37136.784   R-squared       =    0.9899
-------------+----------------------------------   Adj R-squared   =    0.9895
       Total |  173403004          49  3538836.82   Root MSE        =    192.71

------------------------------------------------------------------------------
  marriagert |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     medage  |  -61.23095    16.2825    -3.76   0.000    -93.98711   -28.47479
             |
       state |
  California |          0  (base)
      Nevada |   13255.81   194.9742    67.99   0.000     12863.57    13648.05
             |
       _cons |   2875.366   481.5533     5.97   0.000     1906.606    3844.126
------------------------------------------------------------------------------
```

Look at the result for `state`. Rather than obtaining a coefficient for `5.state` as we expected, Stata instead chose to omit it as the base category.

Stata considers all the individual specifiers for a factor variable together as being related. In our command, we specified that we wanted `i5.state` and `i30.state` by typing

```
. regress marriagert  medage i5.state i30.state
```

and Stata put that together as "include `state`, levels 5 and 30". Then Stata applied its standard logic for dealing with factor variables: treat the smallest level as the base category.

To achieve the desired result, we need to tell Stata that we want no base, which we do by typing the "base none" (`bn`) modifier:

```
. regress marriagert  medage i5bn.state i30.state
```

We need to specify `bn` only once, and it does not matter where we specify it. We could type

```
. regress marriagert  medage i5.state i30bn.state
```

and we would obtain the same result. We can specify `bn` more than once:

```
. regress marriagert  medage i5bn.state i30bn.state
```

The result of typing any one of these commands is

```
. regress marriagert  medage i5bn.state i30.state, baselevels
      Source |       SS           df       MS      Number of obs   =        50
-------------+----------------------------------   F(3, 46)        =   1529.59
       Model |  171681987          3  57227328.9   Prob > F        =    0.0000
    Residual |  1721017.33         46  37413.4203   R-squared       =    0.9901
-------------+----------------------------------   Adj R-squared   =    0.9894
       Total |  173403004          49  3538836.82   Root MSE        =    193.43

------------------------------------------------------------------------------
  marriagert |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     medage  |  -60.80985   16.35134    -3.72   0.001    -93.7234    -27.8963
             |
       state |
  California |  -157.9413   195.5294    -0.81   0.423    -551.5214    235.6389
      Nevada |    13252.3   195.7472    67.70   0.000     12858.28    13646.32
             |
       _cons |   2866.156   483.478     5.93   0.000     1892.965    3839.346
------------------------------------------------------------------------------
```

25.2.13 Including subgroups of levels

We just typed

```
. regress marriagert  medage i5bn.state i30.state
```

You can specify specific levels by using numlists. We could have typed

```
. regress marriagert  medage i(5 30)bn.state
```

There may be a tendency to think of `i(5 30)bn.state` as specifying something extra to be added to the regression. In the example above, it is just that. But consider

```
. regress y  i.arm i.agegroup arm#i(3/4)bn.agegroup
```

The goal may be to restrict the interaction term just to levels 3 and 4 of `agegroup`, but the effect will be to restrict `agegroup` to levels 3 and 4 throughout, which includes the term `i.agegroup`.

Try the example for yourself:

```
. use http://www.stata-press.com/data/r14/fvex
. regress y  i.arm i.agegroup arm#i(3/4)bn.agegroup
```

If you really wanted to restrict the interaction term `arm#agegroup` to levels 3 and 4 of `agegroup`, while leaving `i.agegroup` to include all the levels, you need to fool Stata:

```
. generate agegrp = agegroup
. regress y i.arm i.agegroup arm#i(3/4)bn.agegrp
```

In the above, we use `agegroup` for the main effect, but `agegrp` in the interaction.

25.2.14 Combining factor variables and time-series operators

You can combine factor-variable operators with the time-series operators `L.` and `F.` to lag and lead factor variables. Terms like `iL.group` (or `Li.group`), `cL.age#cL.age` (or `Lc.age#Lc.age`), and `F.arm#L.group` are all legal as long as the data are `tsset` or `xtset`. See [U] 11.4.3.6 Using factor variables with time-series operators.

25.2.15 Treatment of empty cells

Consider the following data:

```
. use http://www.stata-press.com/data/r14/estimability, clear
(margins estimability)
. table sex group
```

	group				
sex	1	2	3	4	5
male	2	9	27	8	2
female	9	9	3		

In these data, there are no observations for `sex` = female and `group` = 4, and for `sex` = female and `group` = 5. Here is what happens when you use these data to fit an interacted model:

```
. regress y  sex##group
note: 1.sex#4.group identifies no observations in the sample
note: 1.sex#5.group identifies no observations in the sample
```

Source	SS	df	MS	Number of obs	=	69
				F(7, 61)	=	4.88
Model	839.550121	7	119.935732	Prob > F	=	0.0002
Residual	1500.65278	61	24.6008652	R-squared	=	0.3588
				Adj R-squared	=	0.2852
Total	2340.2029	68	34.4147485	Root MSE	=	4.9599

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
sex						
female	-5.666667	3.877352	-1.46	0.149	-13.41991	2.086579
group						
2	-13.55556	3.877352	-3.50	0.001	-21.3088	-5.80231
3	-13	3.634773	-3.58	0.001	-20.26818	-5.731822
4	-12.875	3.921166	-3.28	0.002	-20.71586	-5.034145
5	-11	4.959926	-2.22	0.030	-20.91798	-1.082015
sex#group						
female#2	12.11111	4.527772	2.67	0.010	3.057271	21.16495
female#3	10	4.913786	2.04	0.046	.1742775	19.82572
female#4	0	(empty)				
female#5	0	(empty)				
_cons	32	3.507197	9.12	0.000	24.98693	39.01307

Stata reports that the results for `sex` = female and `group` = 4 and for `sex` = female and `group` = 5 are empty; no coefficients can be estimated. The notes refer to `1.sex#4.group` and `1.sex#5.group` because level 1 corresponds to female.

Empty cells are of no concern when fitting models and interpreting results. If, however, you subsequently perform tests or form linear or nonlinear combinations involving any of the coefficients in the interaction, you should be aware that those tests or combinations may depend on how you parameterized your model. See *Estimability of margins* in [R] **margins**.