Multiple Regression Model

Juergen Meinecke

Roadmap

Functional Form Specifications

- Dummy Variable Trap

Let's say you have available G dummy variables that together are mutually exclusive and exhaustive of the population

Example: smoker with $G = 2$

Two dummies:

- `smoker` equal 1 if person is a smoker (zero otherwise)
- `nonsmoker` equal 1 if person is a non-smoker (zero otherwise)

If you are interested in the association between smoking and birthweight then you may want to consider the following three specifications

- $\texttt{birthweight} = \beta_0 + \beta_1 \texttt{smoker} + u_i$
- $\texttt{birthweight} = \beta_0 + \beta_2 \texttt{non-smoker} + u_i$
- $\texttt{birthweight} = \beta_0 + \beta_1 \texttt{smoker} + \beta_2 \texttt{non-smoker} + u_i$

Regression with both `smoker` and `nonsmoker` will throw an error (that's the dummy variable trap)

Python Code (output edited)

```
> reg1 = smf.ols('birthweight ~ smoker', data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> reg1.summary()
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   3432.0600     11.891    288.638      0.000    3408.755    3455.365
smoker      -253.2284     26.810     -9.445      0.000    -305.776    -200.681
==============================================================================

> df['nonsmoker'] = 1 - df.smoker
> reg2 = smf.ols('birthweight ~ nonsmoker', data=df, missing='drop')
>                                             .fit(cov_type='HC1', use_t=False)
> reg2.summary()
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   3178.8316     24.029    132.289      0.000    3131.735    3225.928
nonsmoker    253.2284     26.810      9.445      0.000     200.681     305.776
==============================================================================
```

In each regression, the group represented by 'zero' is the so-called *benchmark* or *default* group (represented by the constant term)

Absolute value of slope coefficient is identical

Example: number of prenatal visits with $G = 4$

Four dummies

- `tripre0` equal 1 if never went for prenatal health visits (presumably a problematic group)
- `tripre1` equal 1 if first prenatal health visit in 1st trimester (presumably the most common group)
- `tripre2` equal 1 if first prenatal health visit in 2nd trimester
- `tripre3` equal 1 if first prenatal health visit in 3rd trimester

We've just learned: only need to use a subset of three dummies

Which subset should we use?

It doesn't matter: as long as we use any three, we are not throwing out any information

However: the unused dummy dummy implicitly defines the benchmark group

From the week 8 lab: $G = 4$, what are the benchmark groups here:

Python Code (output edited)

```
> # benchmark: first prenatal health visit in 1st trimester
> formula_reg1 = 'birthweight ~ smoker + alcohol + tripre0 + tripre2 + tripre3'
> reg1 = smf.ols(formula_reg1, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> reg1.summary()
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   3454.5493     12.482    276.770      0.000    3430.086    3479.013
smoker      -228.8476     26.549     -8.620      0.000    -280.882    -176.813
alcohol      -15.1000     69.703     -0.217      0.828    -151.715     121.516
tripre0     -697.9687    146.579     -4.762      0.000    -985.258    -410.680
tripre2     -100.8373     31.553     -3.196      0.001    -162.680     -38.995
tripre3     -136.9553     67.696     -2.023      0.043    -269.637      -4.274
==============================================================================

> # benchmark: never went for prenatal health visit
> formula_reg2 = 'birthweight ~ smoker + alcohol + tripre1 + tripre2 + tripre3'
> reg2 = smf.ols(formula_reg2, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> reg2.summary()
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   2756.5806    146.077     18.871      0.000    2470.274    3042.887
smoker      -228.8476     26.549     -8.620      0.000    -280.882    -176.813
alcohol      -15.1000     69.703     -0.217      0.828    -151.715     121.516
tripre1      697.9687    146.579      4.762      0.000     410.680     985.258
tripre2      597.1315    149.102      4.005      0.000     304.897     889.366
tripre3      561.0135    160.945      3.486      0.000     245.566     876.461
==============================================================================
```

Multiple Regression Model

Juergen Meinecke

Roadmap

Functional Form Specifications

- Polynomials in X

Consider the following multiple regression model:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \ldots + \beta_r X_i^r + u_i$$

(Note: for simplicity we omit other regressors)

This is just the linear multiple regression model – except that the regressors are powers of X

Estimation, hypothesis testing, etc. proceeds as in the multiple regression model using OLS

The coefficients are difficult to interpret, but the regression function itself is interpretable

We will illustrate the use of polynomials using the textbook's data on test scores and student teacher ratios

Here we focus on the following two variables only

- $testscr_i$ is average test score in school district i
- $avginc_i$ is the average income in school district i (thousands of dollars per capita)

Quadratic specification:

$$testscr_i = \beta_0 + \beta_1 avginc_i + \beta_2 avginc_i^2 + u_i$$

Cubic specification:

$$testscr_i = \beta_0 + \beta_1 avginc_i + \beta_2 avginc_i^2 + \beta_3 avginc_i^3 + u_i$$

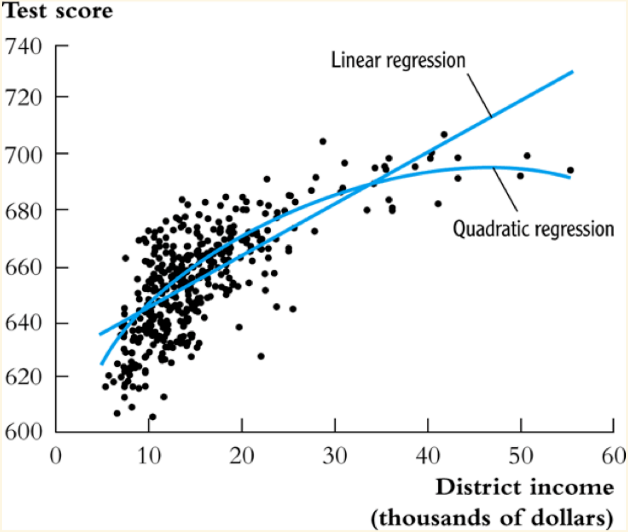
Estimation of the quadratic specification in `Python`

Python Code (output edited)

```
> formula = 'testscr ~ avginc + I(avginc**2)'
> reg1 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg1.summary())

OLS Regression Results
==================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept         607.3017      2.902    209.288      0.000     601.614     612.989
avginc              3.8510      0.268     14.364      0.000       3.326       4.376
I(avginc ** 2)     -0.0423      0.005     -8.851      0.000      -0.052      -0.033
==================================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

Compare the predicted values between linear and quadratic specification



How to interpret the estimated PRF?

Estimated PRF is

$$\widehat{testscr}_i = 607 + 3.85 avginc_i - 0.042 avginc_i^2$$

Predicted change in $testscr_i$ for a change in $avginc_i$ from \$5,000 to \$6,000 per capita (note: $avginc_i$ is in *thousands* of dollars):

$$\begin{aligned}\Delta\widehat{testscr}_i = & \, 607 + 3.85 \cdot 6 - 0.0423 \cdot 6^2 - \\ & (607 + 3.85 \cdot 5 - 0.0423 \cdot 5^2) \\ = & 3.4\end{aligned}$$

Predicted effects for different values of $avginc_i$

$\Delta avginc$	$\Delta testscr$
from \$5,000 to \$6,000	3.4
from \$25,000 to \$26,000	1.7
from \$45,000 to \$46,000	0.0

The effect of changing $avginc_i$ on $testscr_i$ is decreasing in $avginc_i$

The second derivative is negative (that's because the coefficient estimate on the quadratic term is negative)

Caution: do not extrapolate outside the range of the data

Estimation of the cubic specification in `Python`

Python Code (output edited)

```
> formula = 'testscr ~ avginc + I(avginc**2) + I(avginc**3)'
> reg2 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg2.summary())

OLS Regression Results
==================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept       600.0790      5.102    117.615      0.000     590.079     610.079
avginc            5.0187      0.707      7.095      0.000       3.632       6.405
I(avginc ** 2)   -0.0958      0.029     -3.309      0.001      -0.153      -0.039
I(avginc ** 3)    0.0007      0.000      1.975      0.048    5.25e-06       0.001
==================================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

Testing the null hypothesis of linearity, against the alternative that the population regression is quadratic and/or cubic, that is, it is a polynomial of degree up to 3:

$$H_0 : \text{population coefficients on } avginc2 \text{ and } avginc3 \text{ both } 0$$

$$H_1 : \text{at least one of these coefficients is nonzero}$$

Python Code

```
> ftest = reg2.f_test('I(avginc ** 2) = I(avginc ** 3) = 0')
> print(ftest)

                                                     tiny p-value

<F test: F=37.69077411568449, p=9.042596378792848e-16, df_denom=416, df_num=2>
```

The hypothesis that the population regression is linear is rejected at the 5% significance level against the alternative that it is a polynomial of (up to) third order

Multiple Regression Model

Juergen Meinecke

Roadmap

- Functional Form Specifications
  - Logarithmic functions of X or Y

Using logarithmic transformations of both the dependent and independent variables can be useful when estimating coefficients

Using the student test score example, let's focus on two variables:

- Y_i: test score in school district i
- X_i: average income in school district i
  (this is a proxy for socio economic status of the district)

Let's look at the simple regression model

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

We estimate β_1 by running a regression of Y_i on X_i

But what do we estimate when instead we

- run a regression of $\ln Y_i$ on X_i?
- run a regression of Y_i on $\ln X_i$?
- run a regression of $\ln Y_i$ on $\ln X_i$?

The logarithm has useful features based on calculus

Compare the independent variable at two values x_1 and x_0
(it works the same for the dependent variable)

Starting at x_0, you change the dependent variable by $\Delta x := x_1 - x_0$

Define the following: $\tilde{x}_1 = \ln(x_1)$ and $\tilde{x}_0 = \ln(x_0)$

The corresponding change in the logarithm captures:

$$\begin{aligned} \Delta\tilde{x} &:= \tilde{x}_1 - \tilde{x}_0 = \ln(x_1) - \ln(x_0) = \ln(x_0 + \Delta x) - \ln(x_0) \\ &= \ln\left(\frac{x_0 + \Delta x}{x_0}\right) = \ln\left(1 + \frac{\Delta x}{x_0}\right) \approx \frac{\Delta x}{x_0} = \text{percentage change} \end{aligned}$$

The difference in the logarithmic values of x_1 and x_0 is approximately equal to the percentage change between x_1 and x_0

The difference in logarithms approximates percentage changes

For example

$$
\begin{aligned}
x_0 &= 50 & \qquad \tilde{x}_0 &= \ln(x_0) = 3.91 \\
x_1 &= 52 & \qquad \tilde{x}_1 &= \ln(x_1) = 3.95 \\
\implies \frac{\Delta x}{x_0} &= 4\% & \qquad \implies \Delta\tilde{x} &= 0.04
\end{aligned}
$$

Another example:

If $\Delta\tilde{x} = 0.07$ then you know that x increased by 7%

In a few slides we will have:
$\Delta\tilde{x} = 1$ which means that x increased by 100%

(Aside: the log-approximation works best when the change from x_0 to x_1 is small)

Back to the regression model

You create log-versions of both X_i and Y_i

- $\widetilde{X}_i := \ln X_i$
- $\widetilde{Y}_i := \ln Y_i$

Now compare the following four specifications:

Specification	*Population regression function*
(1) linear–linear	$Y_i = \beta_0 + \beta_1 X_i$
(2) linear–log	$Y_i = \beta_0 + \beta_1 \widetilde{X}_i$
(3) log–linear	$\widetilde{Y}_i = \beta_0 + \beta_1 X_i$
(4) log–log	$\widetilde{Y}_i = \beta_0 + \beta_1 \widetilde{X}_i$

The interpretation of the slope coefficient β_1 differs in each case

The generic interpretation of the slope coefficient β_1 is:
By how much does the dependent variable change, on average, when the independent variable changes by one unit?

What does this mean in the different specifications?

(1) $\beta_1 = \dfrac{\Delta Y_i}{\Delta X_i}$ therefore $\Delta X_i = 1 \implies \Delta Y_i = \beta_1$

X up by 1 unit, Y up by β_1 units

(2) $\beta_1 = \dfrac{\Delta Y_i}{\Delta \widetilde{X}_i}$ therefore $\Delta \widetilde{X}_i = 1 \implies \Delta Y_i = \beta_1$

X up by 100%, Y up by β_1 units

(3) $\beta_1 = \dfrac{\Delta \widetilde{Y}_i}{\Delta X_i}$ therefore $\Delta X_i = 1 \implies \Delta \widetilde{Y}_i = \beta_1$

X up by 1 unit, Y up by $100 \cdot \beta_1\%$

(4) $\beta_1 = \dfrac{\Delta \widetilde{Y}_i}{\Delta \widetilde{X}_i}$ therefore $\Delta \widetilde{X}_i = 1 \implies \Delta \widetilde{Y}_i = \beta_1$

X up by 100%, Y up by $100 \cdot \beta_1\%$

Let's illustrate specifications (2), (3), and (4) in Python...

Linear-log specification

Python Code (output edited)

```
> import numpy as np
> formula = 'testscr ~ I(np.log(avginc))'
> reg3 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg3.summary())

OLS Regression Results
==================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept           557.8323      3.840    145.271      0.000     550.306     565.358
I(np.log(avginc))    36.4197      1.397     26.071      0.000      33.682      39.158
==================================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

Interpretation:

a 100% increase in `avginc` is associated with an increase in `testscr` by 36.42 points (the measurement units of `testscr`) on average

or alternatively:

a 1% increase in `avginc` is associated with an increase in `testscr` by 0.3642 points on average

Log-linear specification

Python Code (output edited)

```
> formula = 'I(np.log(testscr)) ~ avginc'
> reg4 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg4.summary())

OLS Regression Results
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      6.4394      0.003   2225.210      0.000       6.434       6.445
avginc         0.0028      0.000     16.244      0.000       0.003       0.003
==============================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

Interpretation:

an increase by $1 in `avginc` will increase `testscr` by 0.28% on average

Log-log specification

Python Code (output edited)

```
> formula = 'I(np.log(testscr)) ~ I(np.log(avginc))'
> reg5 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg5.summary())

OLS Regression Results
=====================================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept              6.3363      0.006   1069.501      0.000       6.325       6.348
I(np.log(avginc))      0.0554      0.002     25.841      0.000       0.051       0.060
=====================================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

Interpretation:
an increase by 100% in `avginc` will increase `testscr` by 5.5% on average

or alternatively:
an increase by 1% in `avginc` will increase `testscr` by 0.055% on average

The coefficient β_1 measures the *elasticity* of Y with respect to X

Multiple Regression Model

Juergen Meinecke

Roadmap

Functional Form Specifications

- Interaction Terms

Interactions Between two Binary Regressors

We will illustrate the use of interaction terms using the textbook's data on test scores and student teacher ratios

Consider the following multiple regression model:

$$testscr_i = \beta_0 + \beta_1 str_i + \beta_2 el\_pct_i + u_i,$$

where

- $testscr_i$ is average test score in school district i
- str_i is average student-teacher ratio in school district i
- el_pct_i is percent of English learners in school district i (remember, this data set is from California where many students are native Spanish speakers)

When you run this regression in `Python`, this is what you get:

Python Code (output edited)

```
> formula = 'testscr ~ str + el_pct'
> reg6 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg6.summary())

OLS Regression Results
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    686.0322      8.728     78.599      0.000     668.925     703.139
str           -1.1013      0.433     -2.544      0.011      -1.950      -0.253
el_pct        -0.6498      0.031    -20.939      0.000      -0.711      -0.589
==============================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

Interpreting the results

- If district i could decrease str_i by one unit while holding el_pct_i constant, it can expect an increase in average test scores of 1.10
- If district i could decrease el_pct_i by one percentage point while holding str_i constant, it can expect an increase in average test scores of 0.65
- Both effects a statistically significant at 5% level

Perhaps a class size reduction is more effective in some circumstances than in others

Perhaps the effect of student-teacher ratio on test scores varies with the percentage of English learners

This would be the case, for example, if English learners benefit disproportionately from smaller class sizes (and therefore lower student-teacher ratios)

More technically, $\frac{\Delta testscr}{\Delta str}$ might depend on el_pct

More generally, $\frac{\Delta Y}{\Delta X_1}$ might depend on X_2

How to model such *interactions* between X_1 and X_2?

Baseline model

$$Y_i = \beta_0 + \beta_1 D_{1i} + \beta_2 D_{2i} + u_i,$$

where D_{1i} and D_{2i} are binary regressors (dummy variables)

β_1 is the effect on Y_i of changing $D_{1i} = 0$ to $D_{1i} = 1$

In this specification, the effect does not depend on value of D_{2i}

To allow the effect of changing D_{1i} to depend on D_{2i}, include the *interaction term* $D_{1i} \times D_{2i}$ as a separate regressor:

$$Y_i = \beta_0 + \beta_1 D_{1i} + \beta_2 D_{2i} + \beta_3 (D_{1i} \times D_{2i}) + u_i$$

Interpreting the coefficients

Compare the PRF when D_{1i} changes from 0 to 1 while D_{2i} is fixed at $q \in \{0, 1\}$

$$\mathrm{E}[Y_i | D_{1i} = 0, D_{2i} = q] = \beta_0 + \beta_2 q$$

$$\mathrm{E}[Y_i | D_{1i} = 1, D_{2i} = q] = \beta_0 + \beta_1 + \beta_2 q + \beta_3 q$$

and their difference

$$\mathrm{E}[Y_i | D_{1i} = 1, D_{2i} = q] - \mathrm{E}[Y_i | D_{1i} = 0, D_{2i} = q] = \beta_1 + \beta_3 q$$

The effect of D_{1i} now depends on the value $q \in \{0, 1\}$ of D_{2i}

Interpretation of β_3:
increment to the effect of D_{1i} on Y_i when $D_{2i} = 1$

For illustration, define the following two dummy variables

$$HiSTR := \begin{cases} 1 & \text{if } str \geqslant 20 \\ 0 & \text{if } str < 20 \end{cases}$$

and

$$HiEL := \begin{cases} 1 & \text{if } el\_pct \geqslant 10 \\ 0 & \text{if } el\_pct < 10 \end{cases}$$

You want to estimate

$$testscr_i = \beta_0 + \beta_1 HiSTR_i + \beta_2 HiEL_i + \beta_3 (HiSTR_i \times HiEL_i) + u_i$$

Here is how you would program this in `Python`:

Python Code (output edited)

```
> df['Hi_str'] = df.str.apply(lambda x: 1 if x >= 20 else 0)
> df['Hi_el_pct'] = df.el_pct.apply(lambda x: 1 if x >= 10 else 0)

> formula = 'testscr ~ Hi_str * Hi_el_pct'
> reg7 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg7.summary())

OLS Regression Results
==================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept         664.1433      1.388    478.459      0.000     661.423     666.864
Hi_str             -1.9078      1.932     -0.987      0.323      -5.695       1.879
Hi_el_pct         -18.1629      2.346     -7.742      0.000     -22.761     -13.565
Hi_str:Hi_el_pct   -3.4943      3.121     -1.120      0.263      -9.612       2.623
==================================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

- Effect of $HiSTR_i$ when $HiEL_i = 0$ is -1.9
- Effect of $HiSTR_i$ when $HiEL_i = 1$ is $-1.9 - 3.5 = -5.4$
- Class size reduction is estimated to have a bigger effect when the percent of English learners is large
- However, the interaction term is not statistically significant

Interactions Between a Continuous and a Binary Regressor

Baseline model

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + u_i,$$

where D_i is binary and X_i is continuous

β_1 is the effect on Y_i of changing X_i

In this specification, the effect does not depend on value of D_i

To allow the effect of changing X_i to depend on D_i, include the *interaction term* $D_i \times X_i$ as a separate regressor:

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + \beta_3 (D_i \times X_i) + u_i$$

Interpreting the coefficients

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 D_i + \beta_3(D_i \times X_i) + u_i$$

Compare the PRF when X changes from x to $x+1$
while D_i is fixed at $q \in \{0,1\}$

$$\begin{aligned}
\mathrm{E}[Y_i|X_i = x, D_i = q] &= \beta_0 + \beta_1 x + \beta_2 q + \beta_3(q \times x) \\
\mathrm{E}[Y_i|X_i = x+1, D_i = q] &= \beta_0 + \beta_1(x+1) \\
&\quad + \beta_2 q + \beta_3(q \times (x+1))
\end{aligned}$$

and their difference

$$\mathrm{E}[Y_i|X_i = x+1, D_i = q] - \mathrm{E}[Y_i|X_i = x, D_i = q] = \beta_1 + \beta_3 q$$

The effect of X now depends on the value $q \in \{0,1\}$ of D_i

Interpretation of β_3: increment to effect of X_i on Y_i when $D_i = 1$

You could view these two cases as two different PRFs

- the intercept is different
- the slope is different

To see this, just rewrite

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 X_i + \beta_2 D_i + \beta_3 (D_i \times X_i) + u_i \\ &= (\beta_0 + \beta_2 D_i) + (\beta_1 + \beta_3 D_i) X_i + u_i, \end{aligned}$$

To make this more explicit, set $D_i = 0$ to obtain

$$Y_i = \beta_0 + \beta_1 X_i + u_i$$

and set $D_i = 1$ to obtain

$$Y_i = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_i + u_i$$

Python Code (output edited)

```
> formula = 'testscr ~ str * Hi_el_pct'
> reg8 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg8.summary())

OLS Regression Results
================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       682.2458     11.868     57.487      0.000     658.985     705.506
str              -0.9685      0.589     -1.644      0.100      -2.123       0.186
Hi_el_pct         5.6391     19.515      0.289      0.773     -32.609      43.887
str:Hi_el_pct    -1.2766      0.967     -1.320      0.187      -3.172       0.619
================================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

- Effect of str_i when $HiEL_i = 0$ is -0.97
- Effect of str_i when $HiEL_i = 1$ is $-0.97 - 1.28 = -2.25$
- Class size reduction is estimated to have a bigger effect when the percent of English learners is large
- But which effects are significant?

Comparing the two PRFs:

$$Y_i = \beta_0 + \beta_1 X_i + u_i \qquad D_i = 0$$

$$Y_i = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_i + u_i \qquad D_i = 1$$

Three hypotheses we could look at

1. The two PRFs are the same: $\beta_2 = 0$ and $\beta_3 = 0$

Python Code

```
> ftest = reg8.f_test('str:Hi_el_pct = Hi_el_pct = 0')
> print(ftest)

<F test: F=89.93943806333414, p=3.455817933875603e-33, df_denom=416, df_num=2>
```

   Rejected

2. The two PRFs have the same slope: $\beta_3 = 0$

   Coefficient on the interaction term has t-statistic of -1.32

   Not rejected

3. The two PRFs have the same intercept: $\beta_2 = 0$

   Coefficient on $HiEL$ has t-statistic of 0.289

   Not rejected

Interactions Between two Continuous Regressors

Baseline model

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + u_i,$$

where X_{1i}, X_{2i} are both continuous

β_1 is the effect on Y_i of changing X_{1i}

In this specification, the effect does not depend on value of X_{2i}

To allow the effect of changing X_{1i} to depend on X_{2i}, include the *interaction term* $X_{1i} \times X_{2i}$ as a separate regressor:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 (X_{1i} \times X_{2i}) + u_i$$

Interpreting the coefficients

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 (X_{1i} \times X_{2i}) + u_i$$

Compare the PRF when X_{1i} changes from x to $x + 1$
while X_{2i} is fixed at $q \in \mathbb{R}$

$$\begin{aligned} \mathrm{E}[Y_i | X_{1i} = x, X_{2i} = q] &= \beta_0 + \beta_1 x + \beta_2 q + \beta_3 (q \times x) \\ \mathrm{E}[Y_i | X_{1i} = x + 1, X_{2i} = q] &= \beta_0 + \beta_1 (x + 1) \\ &\quad + \beta_2 q + \beta_3 (q \times (x + 1)) \end{aligned}$$

and their difference

$$\mathrm{E}[Y_i | X_{1i} = x + 1, X_{2i} = q] - \mathrm{E}[Y_i | X_{1i} = x, X_{2i} = q] = \beta_1 + \beta_3 q$$

The effect of X_{1i} now depends on the value $q \in \mathbb{R}$ of X_{2i}

Interpretation of β_3: increment to effect of X_{1i} on Y_i when $X_{2i} = q$

Python Code (output edited)

```
> formula = 'testscr ~ str * el_pct'
> reg9 = smf.ols(formula, data=df, missing='drop').fit(cov_type='HC1', use_t=False)
> print(reg9.summary())

OLS Regression Results
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    686.3385     11.759     58.365      0.000     663.291     709.386
str           -1.1170      0.588     -1.901      0.057      -2.269       0.034
el_pct        -0.6729      0.374     -1.799      0.072      -1.406       0.060
str:el_pct     0.0012      0.019      0.063      0.950      -0.035       0.037
==============================================================================
Notes: [1] Standard Errors are heteroscedasticity robust (HC1)
```

Interpreting the results

Estimated effect of class size reduction is nonlinear because the size of the effect itself depends on el_pct_i

el_pct		Slope
value	location	of str
1.94	25th percentile	-1.12
8.85	median	-1.11
23.00	75th percentile	-1.09
43.92	90th percentile	-1.07

For example, at the median of el_pct_i (8.85% are English learners), the effect of str_i on test scores is -1.11

The effect of str_i is decreasing in el_pct_i (absolute value)

But the differences do not seem large

Checking statistical significance

- Interaction term is not significant at 5% level
- Neither is the coefficient on str
- But

Python Code

```
> ftest = reg9.f_test('str:el_pct = str = 0')
> print(ftest)

<F test: F=3.8896634079301577, p=0.021200264867197494, df_denom=416, df_num=2>
```

Rejected

- Yet another example in which one should not conduct a joint hypothesis by looking at the coefficients individually
- An F-test is required