Class 13 OLS Regression Advanced

Dr Wei Miao

UCL School of Management

November 15, 2023

Section 1

Categorical Variables

Categorical variables

- So far, the independent variables we have used are `Income` and `Kidhome`, which are **continuous variables**.
- Some variables are intrinsically not countable; we need to treat them as **categorical variables**
  - e.g., gender, education group, city.

Handling Categorical Variables in R using `factor()`

- In R, we need to use a function `factor()` to explicitly inform R that this variable is a categorical variable, such that statistical models will treat them differently from continuous variables.
  - e.g., we can use `factor(Education)` to indicate that, `Education` is a categorical variable.

```
data_full <- data_full %>%
  mutate(Education_factor = factor(Education))
```

- We can use `levels()` to check how many categories there are in the factor variable.
  - e.g., `Education` has 5 different levels.

```
# check levels of a factor
levels(data_full$Education_factor)
```

```
[1] "2n Cycle"   "Basic"      "Graduation" "Master"     "PhD"
```

Handling Categorical Variables using `factor()`

- `factor()` will check all levels of the categorical variables, and then choose the default level based on alphabetic order.
- If needed, we can revise the baseline group to another group using `relevel()` function.

```
# Create a new factor variable, with Basic as the baseline.
data_full <- data_full %>%
  mutate(Education_factor_2 = relevel(Education_factor,
                                      ref = "Basic") )

levels(data_full$Education_factor_2)
```

```
[1] "Basic"      "2n Cycle"   "Graduation" "Master"     "PhD"
```

Running Regression with Factor Variables

```r
pacman::p_load(fixest,modelsummary)
feols_categorical <- feols(data = data_full,
  fml = total_spending ~ Income + Kidhome + Education_factor_2)
modelsummary(feols_categorical,
             stars = T,
             gof_map = c('nobs','r.squared'))
```

	(1)
(Intercept)	−180.297**
	(56.305)
Income	0.020***
	(0.000)
Kidhome	−227.761***
	(16.961)
Education_factor_22n Cycle	−164.044**
	(60.448)
Education_factor_2Graduation	−119.695*
	(56.176)
Education_factor_2Master	−143.015*
	(58.443)
Education_factor_2PhD	−153.190**
	(57.751)
Num.Obs.	2000
R2	0.662

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

One-Hot Encoding of `factor()`

- In the raw data, Education is label-encoded with 5 levels.

```
     ID  Education
1  5524 Graduation
2  2174 Graduation
3  4141 Graduation
4  6182 Graduation
5  5324        PhD
6  7446     Master
7   965 Graduation
8  6177        PhD
9  4855        PhD
10 5899        PhD
```

- After factorizing education with "*Basic*" as the baseline group, internally, we have 4 binary indicators as follows. Because we have the intercept,"*Basic*" is omitted as the baseline group. Other groups represent the comparison relative to the baseline group.

```
      ID Edu_2n Cycle Edu_Graduation Edu_Master Edu_PhD
 1: 5524            0              1          0       0
 2: 2174            0              1          0       0
 3: 4141            0              1          0       0
 4: 6182            0              1          0       0
 5: 5324            0              0          0       1
 6: 7446            0              0          1       0
 7:  965            0              1          0       0
 8: 6177            0              0          0       1
 9: 4855            0              0          0       1
10: 5899            0              0          0       1
```

Interpretation of Coefficients for Categorical Variables

- In general, R uses **one-hot encoding** to encode factor variables with **K** levels into **K-1** binary variables.
  - As we have the intercept term, we can only have **K-1** binary variables.
- The interpretation of coefficients for factor variables: Ceteris paribus, compared with the ***[baseline group]***, the ***[outcome variable]*** of ***[group X]*** is higher/lower by ***[coefficient]***, and the coefficient is statistically ***[significant/insignificant]***.
  - Ceteris paribus, compared with the basic education group, the total spending of PhD group is lower by 153.190 dollars. The coefficient is statistically significant at the 1% level.
- Now please rerun the regression using `Education_factor` and interpret the coefficients. What's your finding?
  - Conclusion: factor variables can only measure the relative difference in outcome variable across different groups rather than the absolute levels.

Application of Categorical Variables in Marketing

- Analyze the treatment effects in A/B/N testing, where $Treatment_i$ is a categorical variable that specifies the treatment group customer i is in:

$$Outcome_i = \beta_0 + \delta Treatment_i + \epsilon$$

- Analyze the brand premiums or country-of-origin effects:

$$Sales_i = \beta_0 + \beta_1 Brand_i + \beta_2 Country_i + X\beta + \epsilon$$

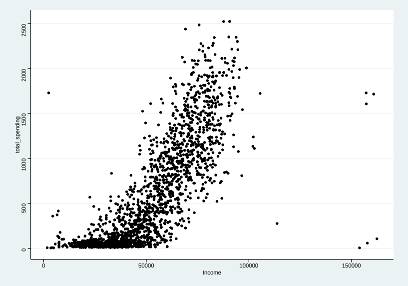
Section 2

Non-linear Effects

Quadratic Terms

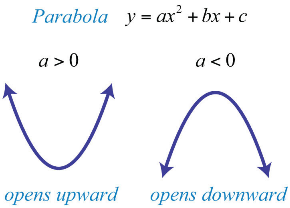
- If we believe the relationship between the outcome variable and explanatory variable is a quadratic function, we can include **an additional quadratic term** in the regression to model such non-linear relationship.

$$totalspending = \beta_0 + \beta_1 Income + \beta_2 Income^2 + \epsilon$$



Quadratic Terms

- If the coefficient for $Income^2$ is negative, then we have an downward open parabola. That is, as income increases, total spending first increases and then decreases, i.e., a non-linear, non-monotonic effect.
  - As income first increases, customers increase their spending with Tesco due to the **income effect**; however, as customers get even richer, they may switch to more premium brands such as Waitrose, so their spending may decrease due to the **substitution effect**.



Quadratic Terms in Linear Regression

- Let's run two regressions in the Quarto document, with and without the quadratic term.

```
# model 1: without quadratic term
feols_noquadratic <- feols(data = data_full,
  fml = total_spending ~ Income )

# model 2: with quadratic term
feols_quadratic <- feols(data = data_full%>%
                             mutate(Income_squared = Income^2 ),
  fml = total_spending ~ Income  + Income_squared )
```

Quadratic Terms in Linear Regression

```
modelsummary(list(feols_noquadratic,
        feols_quadratic),
        stars = T,
        fmt = fmt_sprintf("%.2e"),
        gof_map = c('nobs','r.squared'))
```

	(1)	(2)
(Intercept)	-5.57×10^{2}***	-6.27×10^{2}***
	(2.17×10^{1})	(3.65×10^{1})
Income	2.24×10^{-2}***	2.53×10^{-2}***
	(3.84×10^{-4})	(1.30×10^{-3})
Income_squared		-2.66×10^{-8}*
		(1.12×10^{-8})
Num.Obs.	2000	2000
R2	0.629	0.630

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Quadratic Terms: Compute the Vertex

- We can compute the vertex point where total spending is maximized by income

```
# extract the coeffcient vector using $ sign
feols_coefficient <- feols_quadratic$coefficients
feols_coefficient
```

```
  (Intercept)         Income Income_squared
-6.270403e+02   2.533276e-02  -2.663682e-08
```

```
# Use b / (-2a) to get the vertex
- feols_coefficient[2]/
  (2 * feols_coefficient[3])
```

```
  Income
475521.5
```

Section 3

Linear Probability Model

Linear Probability Model

- In Predictive Analytics, we learned how to use decision tree and random forest to make predictions for binary outcome variables.
- In fact, linear regression can also be used as another supervised learning model to predict binary outcomes. When the outcome variable is a binary variable, the linear regression model is also called linear probability model.
  - On the one hand, regression predicts the expectation of response Y conditional on X; that is
    $$E[Y] = E[X\beta + \epsilon] = X\beta$$
  - On the other hand, for a binary outcome variable, if the probability of outcome occurring is p, then we can write the expectation of Y is
    $$E[Y] = 1 * p + 0 * (1 - p) = p$$
  - As a result, we have the following equation
    $$p = X\beta$$
- Interpretation of LPM coefficients: Everything else equal, a unit change in x will change the **probability of the outcome occurring** by β.

Pros and Cons of LPM

- We use linear regression function `feols()` to train the LPM on the **training data** and make predictions using `predict(LPM, data_test)` to make predictions on the **test data**.
- Advantages
  - Fast to run, even with a large number of fixed effects and features
  - High interpretability: coefficients have clear economic meanings
- Disadvantages
  - Predicted probabilities of occurring may fall out of the [0,1] range
  - Accuracy tends to be low