Econometrics

TA Session 2

Lucia Sauer

2025-10-01

Overview

- Conditional means
- OLS in matrix algebra
- OLS using R and Python preset functions
- Plotting observations and fitted lines
- Verify some numerical property

Why These Topics?

> Conditional means
>
> Foundation of regression: OLS estimates the conditional mean of Y given X.

> OLS in matrix algebra
>
> Build intuition for how OLS works beyond formulas and preset functions.

> Numerical Conditions
>
> Check core properties of OLS to validate results and understand residual behavior.

Conditional Mean

Conditional Mean

Population concept:
The *conditional mean* of Y given $X = x$ is the expected value of Y in the sub-population where $X = x$:

$$E[Y|X = x]$$

Sample estimate:
The *sample conditional mean* is the average of all observed Y_i for which $X_i = x$:

$$\hat{E}[Y|X = x] = \frac{1}{N_x} \sum_{i:X_i=x} Y_i$$

where N_x is the number of observations with $X_i = x$.

The OLS estimator aims to model the *conditional mean function*, i.e., $E[Y|X]$, as a function of X.

Example: House Prices and Size

Description dataset

```
import wooldridge
df = wooldridge.data('hprice1')
#Look to a sample of 8 observations
df.sample(8)
```

	price	assess	bdrms	lotsize	sqrft	colonial	lprice	lassess	llotsize	lsqrft
70	215.000000	300.399994	3	11554.0	1694	0	5.370638	5.705115	9.354787	7.434848
34	361.000000	354.899994	4	9000.0	2066	1	5.888878	5.871836	9.104980	7.633369
38	209.001007	289.399994	4	6400.0	1854	1	5.342339	5.667810	8.764053	7.525101
42	248.000000	273.299988	4	7050.0	1656	1	5.513429	5.610570	8.860783	7.412160
41	713.500000	655.400024	5	28231.0	3331	1	6.570182	6.485246	10.248176	8.111028
29	350.000000	355.299988	4	9773.0	2051	1	5.857933	5.872962	9.187379	7.626083
17	285.000000	305.600006	3	7123.0	1774	1	5.652489	5.722277	8.871084	7.480992
33	235.000000	251.899994	4	6383.0	1840	1	5.459586	5.529032	8.761394	7.517521

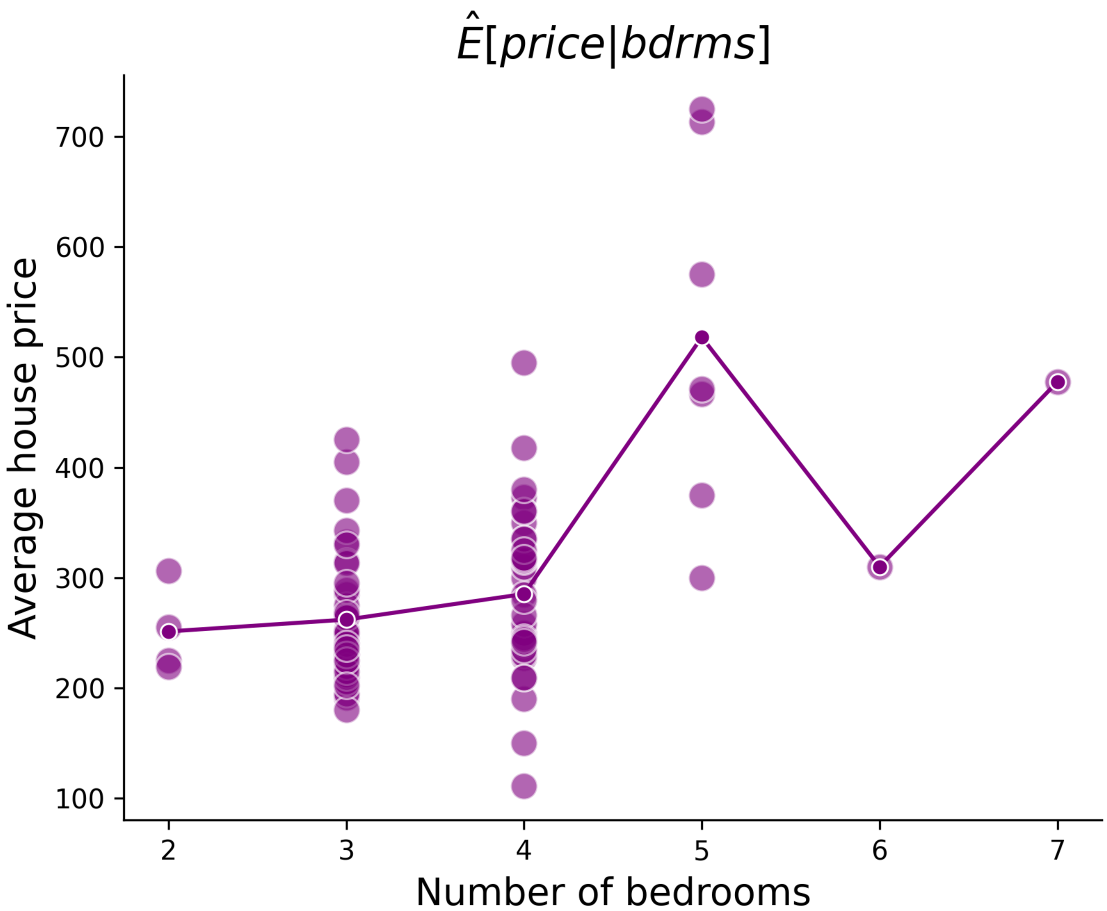
Compute Conditional Mean

```
import pandas as pd
df_grouped = df.groupby('bdrms')['price'].mean().reset_index()
df_grouped
```

	bdrms	price
0	2	251.250000
1	3	261.979167
2	4	285.163667
3	5	518.003571
4	6	310.000000
5	7	477.500000

Plot Conditional Mean

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(data=df, x='bdrms', y='price', color='purple', s=100)
sns.lineplot(data=df_grouped, x='bdrms', y='price', color='purple')
plt.show()
```



OLS in Matrix Algebra

Using our dataset, we can write the model as:

$$price\_i = \beta_1 + \beta_2 \cdot bdrms_i + \beta_3 \cdot sqrft_i + \beta_4 \cdot colonial + \varepsilon_i$$

We can express this model in matrix form as:

$$y = X\beta + \varepsilon$$

where:

$$\begin{bmatrix} price_1 \\ price_2 \\ \vdots \\ price_n \end{bmatrix}, \quad \begin{bmatrix} 1 & bdrms_1 & sqrft_1 & col_1 \\ 1 & bdrms_2 & sqrft_2 & col_2 \\ \vdots & \vdots & \vdots & \\ 1 & bdrms_n & sqrft_n & col_n \end{bmatrix}, \quad \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}, \quad \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Exercise: OLS in Matrix Algebra

Exercises

1. Estimate the OLS coefficients using matrix algebra.
2. Compute the fitted values and OLS residuals.
3. Calculate the Sum of Squared Errors (SSE).
4. Compute the R^2 statistic.

---

1. Estimate the OLS Coefficients

Starting from the OLS objective function:

$$\min_b \varepsilon'\varepsilon = \min_b (y - Xb)'(y - Xb)$$

The solution is given by:

$$\hat{\beta} = (X'X)^{-1}X'y$$

```python
import numpy as np
df['intercept'] = 1
X = df[['intercept', 'bdrms', 'sqrft', 'colonial']].values
y = df['price'].values
X_tX = X.T @ X
X_ty = X.T @ y
beta_hat = np.linalg.inv(X_tX) @ X_ty
print("Coefficients (beta_hat):",np.round(beta_hat,2))
```

```
Coefficients (beta_hat): [-21.55  12.49   0.13  13.08]
```

Fitted model:

$$\widehat{price}_i = -21.55 + 12.49 \cdot bdrms_i + 0.13 \cdot sqrft_i + 13.08 \cdot colonial_i$$

where the dependent variable `price` is in **$1000s**.

---

2. Compute the fitted values and OLS residuals.

$$\hat{y} = X\hat{\beta}$$

```python
# Fitted values
y_hat = X @ beta_hat
print(np.round(y_hat[:10], 2))
```

```
[358.05 298.55 194.32 217.01 367.92 411.57 297.39 253.76 245.35 261.32]
```

$$\hat{\varepsilon} = y - \hat{y}$$

```python
# Residuals
epsilon_hat = y - y_hat
print(np.round(epsilon_hat[:10], 2))
```

```
[-58.05  71.45  -3.32 -22.01   5.08  54.71  35.11  61.24 -39.35 -21.32]
```

Units

1. In what units are the fitted values $\hat{y}$?
2. In what units are the residuals $\hat{\varepsilon}$?

---

3. Calculate the Sum of Squared Errors (SSE).

$$SSE = \hat{\varepsilon}'\hat{\varepsilon}$$

Note that this is exactly the same as:

$$SSE = \sum_{i=1}^{n}(\hat{\varepsilon}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

```
#Sum of Squared Errors
SSE = epsilon_hat.T @ epsilon_hat
print(np.round(SSE, 2))
```

```
334982.95
```

SSE Units

Note: The SSE is in the squared units of the dependent variable (here, the price in 1000s of dollars).

---

4. Compute the R^2 statistic.

$$R^2 = 1 - \frac{SSE}{SST}$$

where

$$SST = (y - \bar{y}1)'(y - \bar{y}1)$$

```
# Total Sum of Squares
y_bar = np.mean(y)
SST = ((y - y_bar).T @ (y - y_bar))
r2 = 1 - (SSE / SST)
print(np.round(r2, 4))
```

```
0.635
```

R^2 Units

Note: The R^2 is unit free, and tells us that about 64% of the variation in house price is captured by the model.

---

3. Python and R Preset Functions

Preset Functions

All the operations we did in matrix algebra can be done using preset functions in Python and R.

- Python: `statsmodels` library, specifically the `OLS` class from `statsmodels.api`.
- R: `lm()` function.

Code example in Python

```
import statsmodels.api as sm
# Fit the model using statsmodels (OLS)
model = sm.OLS(y, X).fit()
# Print the coefficients and SSE
betas = model.params
y_hat = model.fittedvalues
epsilon_hat = model.resid
SSE = np.sum(epsilon_hat ** 2)
r2 = model.rsquared
print("Coefficients (betas):", np.round(betas, 2))
print("(SSE):", SSE)
print("(R^2):", r2)
```

```
Coefficients (betas): [-21.55  12.49   0.13  13.08]
(SSE): 334982.94691739464
(R^2): 0.6350369859143852
```

Code example in R

```r
library(wooldridge)
df <- wooldridge::hprice1
df$intercept <- 1
model <- lm(price ~ intercept + bdrms + sqrft + colonial, data = df)

summary(model)$coefficients
SSE <- sum(residuals(model)^2)
r2 <- summary(model)$r.squared

print(paste("Coefficients (betas):", round(coef(model), 2)))
print(paste("SSE:", round(SSE, 2)))
print(paste("R^2:", round(r2, 4)))
```

---

4. Plotting Observations and Fitted Line

For a simple model of $K = 2$, estimate the model and plot the observations and the fitted line.

$$price_i = \beta_1 + \beta_2 \cdot sqrft_i + \epsilon_i$$

```python
#estimate the model

X = df[['intercept', 'sqrft']].values
y = df['price'].values
#using statsmodels
model = sm.OLS(y, X).fit()
betas = model.params
y_hat = model.fittedvalues
print("Coefficients (betas):", np.round(betas, 2))
```

```
Coefficients (betas): [11.2   0.14]
```

```
# Scatter: data points
sns.scatterplot(x=df['sqrft'], y=df['price'], color='purple', s=50, ax=ax)
# Fitted line
sns.lineplot(x=df['sqrft'], y=y_hat, color='grey', ax=ax)
plt.show()
```

---

5. Numerical Property of OLS

Numerical Property of OLS

These properties are independent of the statistical assumptions, they are purely mathematical properties of the OLS estimator, that hold given a sample.

1.
$$\sum_{i=1}^{n} \hat{\varepsilon}_i = 0$$

2.
$$X'\hat{\varepsilon} = 0$$

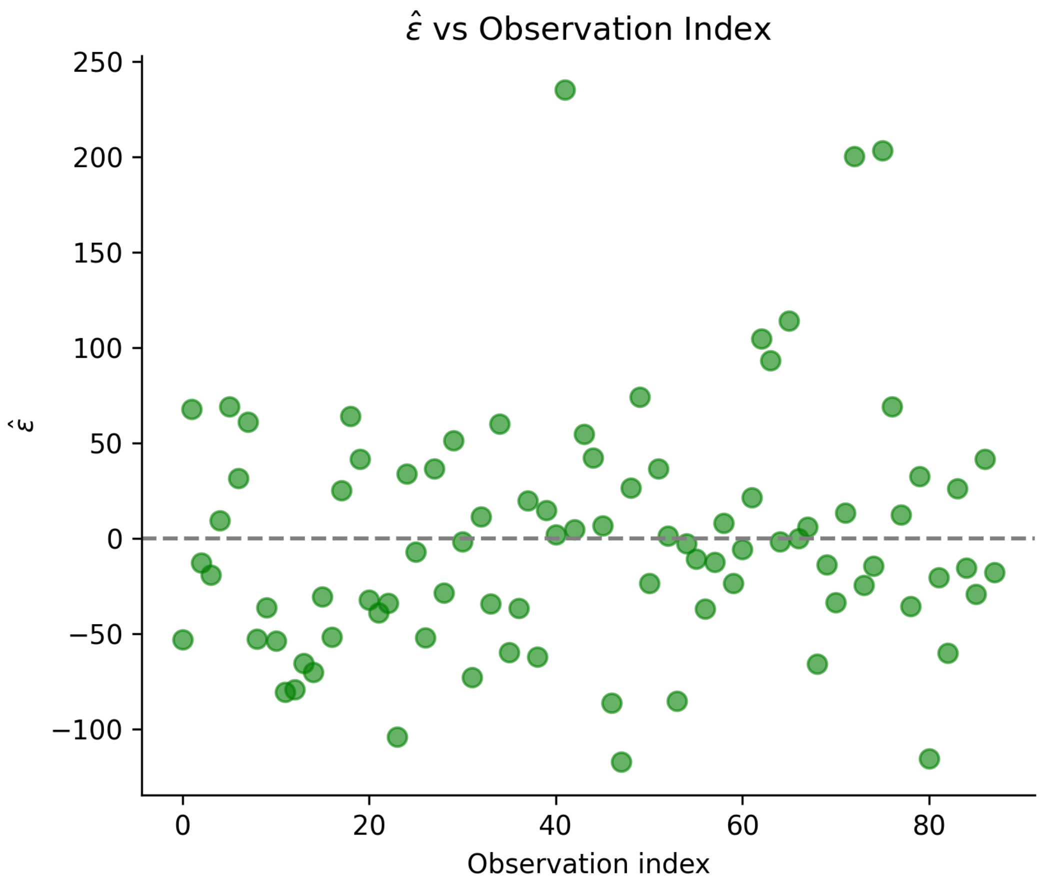
---

1. Sum of residuals is zero

```
epsilon_hat = y - y_hat
print(epsilon_hat.sum().round(8))
```

```
-0.0
```

Illustration:



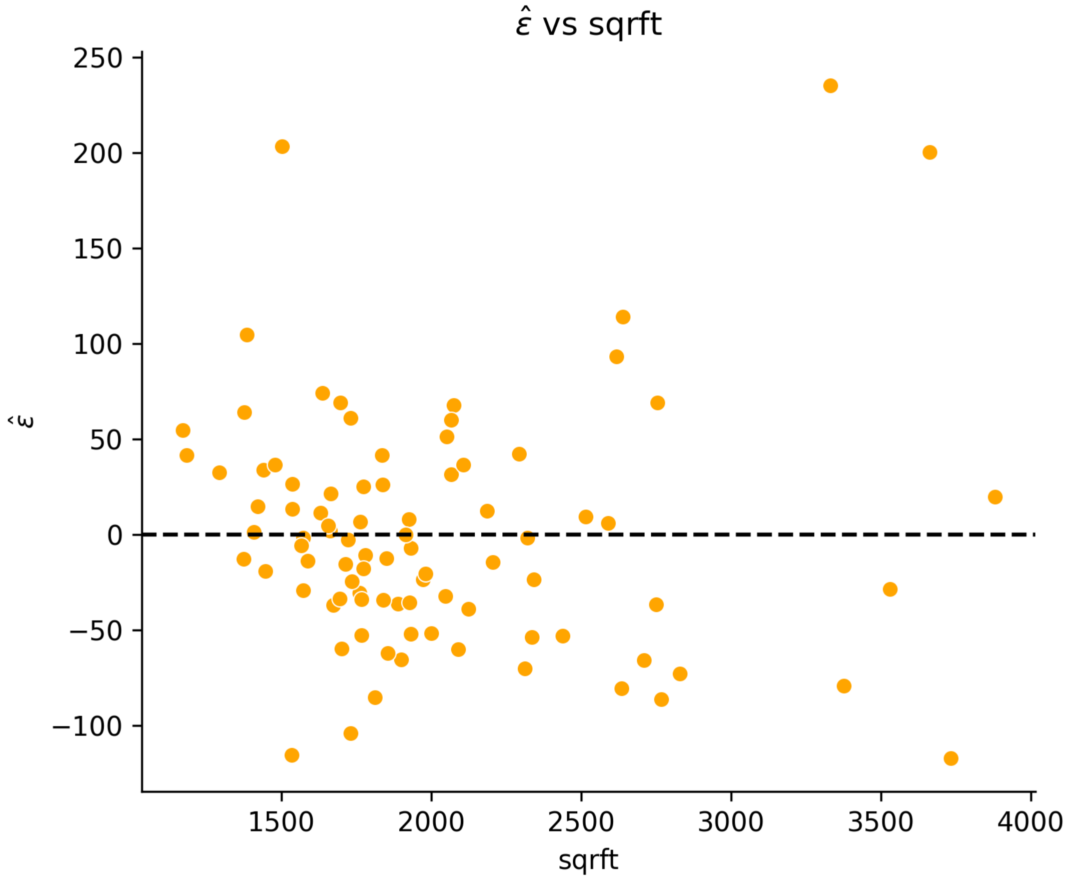
---

2. Residuals are orthogonal to regressors

```
X_t_epsilon = X.T @ epsilon_hat
print(np.round(X_t_epsilon, 5))
```

```
[-0. -0.]
```

Illustration:


$\hat{\varepsilon}$ vs sqrft
250
200
150
100
50
0
−50
−100
$\hat{\varepsilon}$
1500
2000
2500
3000
3500
4000
sqrft