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

```
library(wooldridge)
library(dplyr)
df <- wooldridge::hprice1
# Sample of 8 observations
df[sample(nrow(df), 8), ]
```

```
     price assess bdrms lotsize sqrft colonial   lprice  lassess llotsize
47 313.000  324.0     3    1000  2768        0 5.746203 5.780744 6.907755
72 240.000  250.7     3    6000  1536        1 5.480639 5.524257 8.699514
1  300.000  349.1     4    6126  2438        1 5.703783 5.855359 8.720297
4  195.000  231.8     3    4600  1448        1 5.273000 5.445875 8.433811
71 215.000  300.4     3   11554  1694        0 5.370638 5.705115 9.354787
18 285.000  305.6     3    7123  1774        1 5.652489 5.722277 8.871084
7  332.500  367.8     3    9000  2067        1 5.806640 5.907539 9.104980
82 268.125  254.0     3    5167  1980        1 5.591453 5.537334 8.550048
     lsqrft
```

```
47 7.925880
72 7.336937
1  7.798934
4  7.277938
71 7.434848
18 7.480992
7  7.633853
82 7.590852
```

Compute Conditional Mean in R

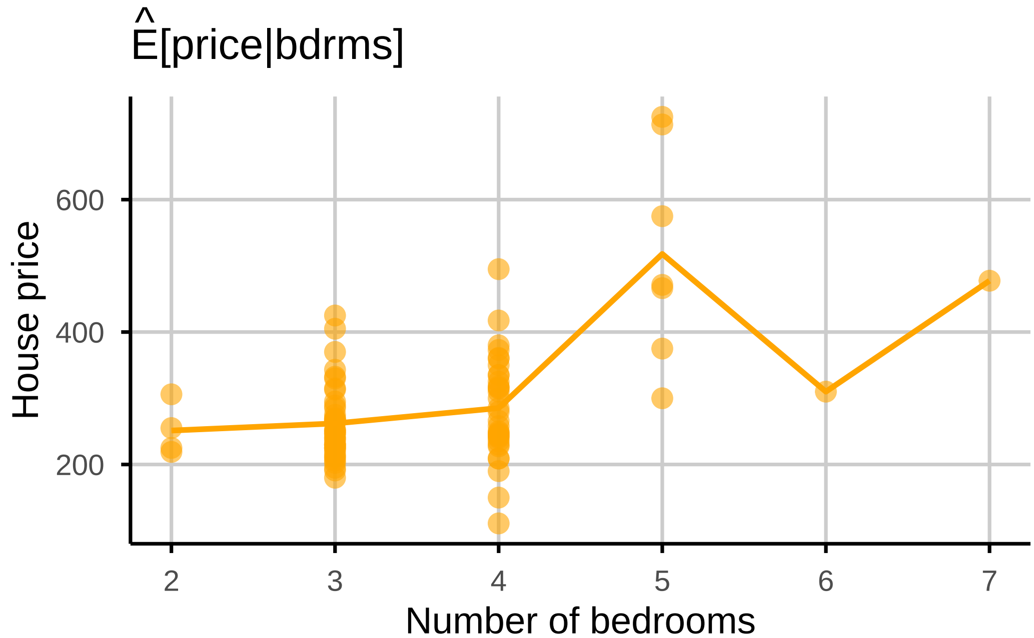
```r
df_grouped <- df %>%
  group_by(bdrms) %>%
  summarise(mean_price = mean(price))
df_grouped
```

```
# A tibble: 6 x 2
  bdrms mean_price
  <int>      <dbl>
1     2       251.
2     3       262.
3     4       285.
4     5       518.
5     6       310
6     7       478.
```

Plot Conditional Mean in R

```r
library(ggplot2)
ggplot() +
  geom_point(data = df, aes(x = bdrms, y = price),
             color = 'orange', alpha = 0.6, size = 3) +
  geom_line(data = df_grouped, aes(x = bdrms, y = mean_price),
            color = 'orange', size = 1) +
```

```r
  labs(title = expression(hat(E)[price ~ "|" ~ bdrms]),
       x = "Number of bedrooms",
       y = "Average house price")
```



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:

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

```r
df$intercept <- 1
X <- as.matrix(df[, c("intercept", "bdrms", "sqrft", "colonial")])
y <-df$price

#define X_tX and X_ty
X_tX <- t(X)%*%X
X_ty <- t(X)%*%y

#solve the system
beta_hat<- solve(X_tX)%*%X_ty
round(beta_hat, 2)
```

```
             [,1]
intercept -21.55
bdrms      12.49
sqrft       0.13
colonial   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.

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

```r
# Fitted values
y_hat <- X %*% beta_hat
```

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

```r
# Residuals
epsilon_hat <- y - y_hat
```

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:

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

```r
#Sum of Squared Errors
SSE <- t(epsilon_hat) %*% epsilon_hat
round(SSE, 2)
```

```
       [,1]
[1,] 334983
```

> 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

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

```r
# Total Sum of Squares
y_bar <- mean(y)
SST <- t(y - y_bar) %*% (y - y_bar)
r2 <- 1 - SSE/SST
round(r2, 4)
```

```
      [,1]
[1,] 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 R

```r
#Estimate the model with lm function
model <- lm(price ~ bdrms + sqrft + colonial, data = df)
beta_hat <- coef(model)
y_hat <- fitted(model)
epsilon_hat <- residuals(model)

sse <- sum(epsilon_hat^2)
r2 <- summary(model)$r.squared
cat("Coefficients (betas):\n", beta_hat)
```

```
Coefficients (betas):
 -21.55241 12.48749 0.1298488 13.07755
```

```r
cat("\nSSE:", round(sse, 2), "\n")
```

```

SSE: 334983
```

```r
cat("R^2:", round(r2, 4), "\n")
```

```
R^2: 0.635
```

---

4. Plotting Observations and Fitted Line

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

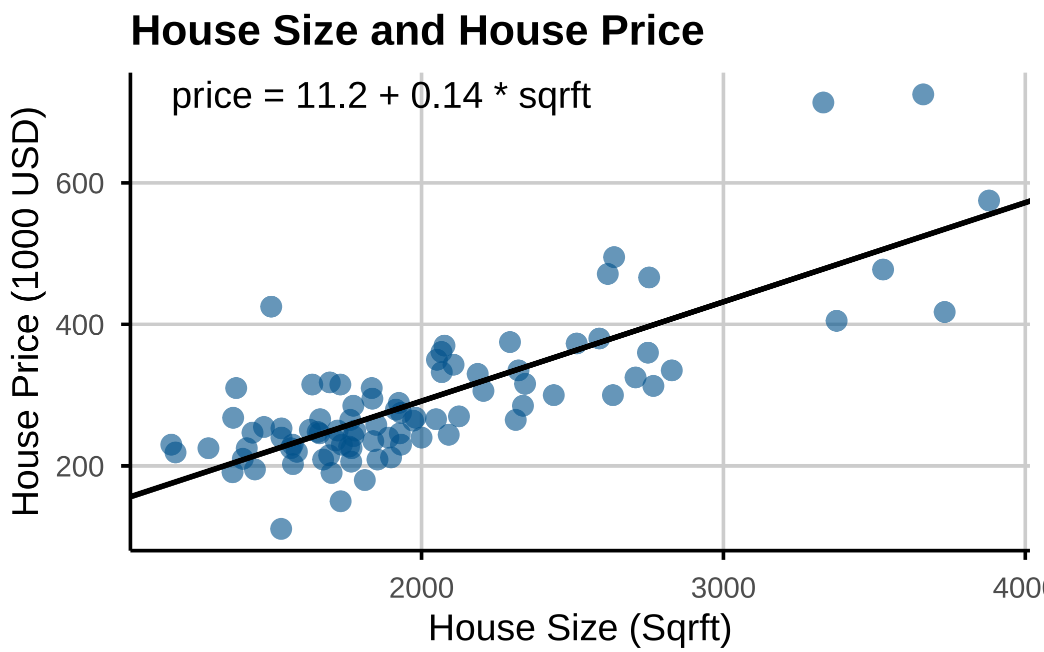
$$price_i = \beta_1 + \beta_2 \cdot sqrft_i + \epsilon_i$$

```
#estimate the model
#first we need to estimate simple model
model <- lm(price ~ sqrft, data = df)
y_hat <- fitted(model)
beta_hat <- coef(model)
beta_hat
```

```
(Intercept)       sqrft
  11.204145    0.140211
```

```
#plot observations with scatter and fitted line
ggplot() +
    #scatter with raw data
    geom_point(data = df, aes(x = sqrft, y = price),
    color = '#00518b') +

    geom_abline(intercept= beta_hat[1], slope = beta_hat[2],
    color = 'black', linewidth = 1) +
    #labels and styling
    labs(
        title = 'House size and price',
        x = 'sqrft',
        y='house price in 1000 usd'
    )
```



---

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. OLS is unitdependent, hence SSE is also unit independent.

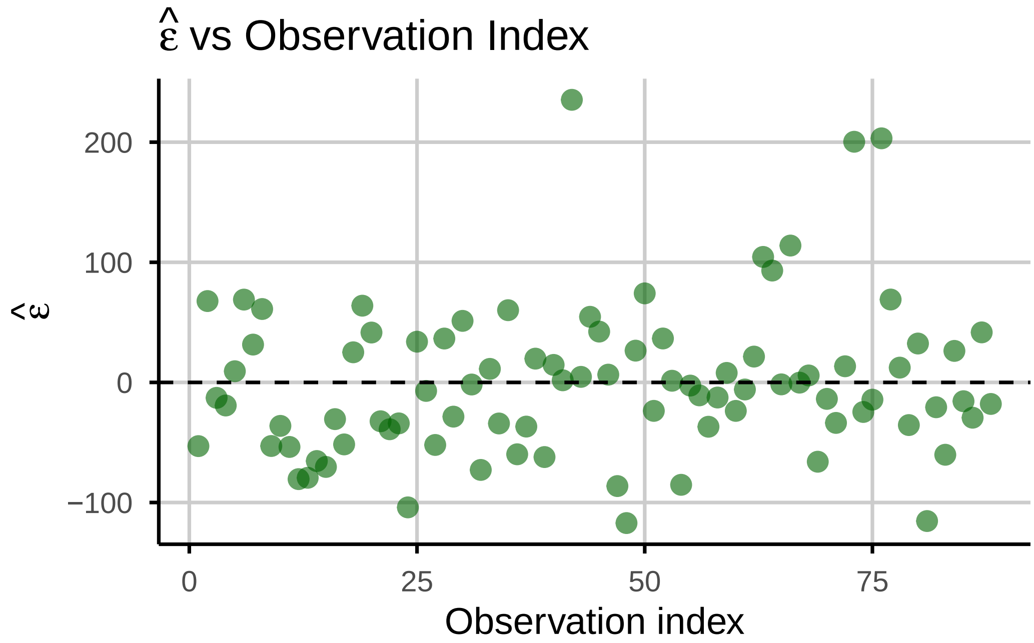
---

1. Sum of residuals is zero

```r
epsilon_hat <- residuals(model)
round(sum(epsilon_hat),2)
```

```
[1] 0
```

Illustration:



---

2. OLS unit dependent, hence SSE also unit dependent.

Our dependent variable is expressed in \$1000s, so let's scale it to actual dollars and see how the SSE changes.

```r
df$price <- df$price * 1000
model_dollars <- lm(price ~ sqrft, data = df)
beta_hat_dollars <- coef(model_dollars)
sse_dollars <- sum(residuals(model_dollars)^2)
cat("Coefficients in thousands of dollars:", round(beta_hat, 4), "\n")
```

```
Coefficients in thousands of dollars: 11.2041 0.1402
```

```r
cat("Coefficients in dollars:", round(beta_hat_dollars, 4), "\n")
```

```
Coefficients in dollars: 11204.15 140.211
```

```r
cat("SSE in thousands of dollars:", round(sse, 2), "\n")
```

```
SSE in thousands of dollars: 334983
```

```r
cat("SSE in dollars:", round(sse_dollars, 2), "\n")
```

```
SSE in dollars: 348053431609
```