Introduction
0000

Linear Algebra
00000000000

Coding
0000000000

P218 Econometrics I
TA Session 0

Gabriel Simões Gaspar

London Business School

Fall 2022

Introduction
0000

Linear Algebra
00000000000

Coding
0000000000

Roadmap

Introduction

Introduction
○●○○

Linear Algebra
○○○○○○○○○○○

Coding
○○○○○○○○○○

Welcome

Welcome to the P218 - Econometrics I TA sessions!

- **Email:** gsimoesgaspar@london.edu / gabrielsgaspar@gmail.com
- **Office:** P320 Plowden Building

Admin

Tutorials: Fridays from 12:45h to 15:30h London time.

- **Frequency:** Usually the week you have a problem set due. I will email you at the start of the week confirming the location.
- **Format:** In person discussion / presentation. If there are any topics you want me to emphasize, please let me know before the session.

Problem Sets: Due after every two sessions.

- **Hand in:** Submit self-contained PDF with answers, figures and tables (use LATEX).
- **Code:** Some code will be required for problem sets (not loads). Please hyperlink link to your Github repository with code.
  - E.g. "The code for the following question can be found in this Github repo."

Final exam: Scheduled for December 16 from 16:00h to 19:00h.

Introduction
Linear Algebra
Coding

○○○●
○○○○○○○○○○○
○○○○○○○○○○

Advice

This is a PhD level course, so take it seriously.

- Alexey will cover a lot of material and in great detail. It is perfectly fine if you don't understand everything in lecture.
- Look at different reference books if things are not clear.
- Feel free to email me with questions - I will try my best to help you.

Problem sets are lengthy.

- Start early. One week is not enough to properly complete a problem set (especially the later ones).
- Make sure you understand the concepts being covered.

Linear Algebra

Matrices & Vectors

Working with sums and multiplications of elements of matrices can get confusing very fast. Throughout the course, keep in mind the following representation for the outcome variable of interest:

$$Y_{n\times 1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

So that scalar y_i is the value for unit of observation i. This matrix holds information of the outcome variable of our sample with is of size n.

Introduction
0000

Linear Algebra
00●00000000

Coding
0000000000

Definitions

Matrices & Vectors

We will also use the following matrix to hold the values of our regressor variables:

$$X_{n \times k} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

A column k represents the value of variable k for all n units of observation. A row i represents the value of all of our variables for unit of observation i. We will define the **transpose** of the i-th row of matrix X as:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \cdots \\ x_{ik} \end{bmatrix}$$

Introduction
0000

Linear Algebra
00000000000

Coding
0000000000

Definitions

Matrix Operations as Summations

We can interpret a matrix operation using summations. For example (suppressing dimensionality subscript):

$$X'X = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1k} & x_{2k} & \cdots & x_{nk} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

$$= \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \sum_{i=1}^{n} x_i x_i'$$

Introduction
0000

Linear Algebra
0000●000000

Coding
0000000000

Definitions

Matrix Operations as Summations

Given the definition of Y, we can also write:

$$X'Y = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i$$

So, the OLS estimator can be written using summations as:

$$\hat{\beta}_{OLS} = (X'X)^{-1}X'Y = \left(\sum_{i=1}^{n} x_i x_i' \right)^{-1} \sum_{i=1}^{n} x_i y_i$$

Introduction
0000

Linear Algebra
00000●00000

Coding
0000000000

Projections

Projecting Vectors

A key concept to understand when dealing with linear regression is projection. Consider the following three-dimensional space representation.
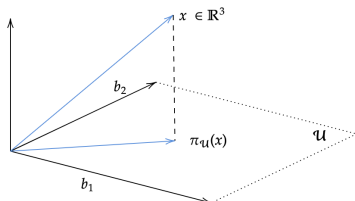
The vector $\pi_{\mathcal{U}}(x)$ is the orthogonal projection of vector x onto space $\mathcal{U} = [b_1, b_2]$, where b_1 and b_2 are the basis that span space $\mathcal{U}$.

Introduction
0000

Linear Algebra
0000000●0000

Coding
0000000000

Projections

Projecting Vectors

We can represent $\pi_{\mathcal{U}}(x)$ as a linear combination of the basis vectors that span $\mathcal{U}$ since $\pi_{\mathcal{U}}(x) \in \mathcal{U}$.

$$\pi_{\mathcal{U}}(x) = \lambda_1 b_1 + \lambda_2 b_2$$

Moreover, given that the difference vector $x - \pi_{\mathcal{U}}(x)$ is orthogonal to $\mathcal{U}$, its inner product with the basis vectors that span $\mathcal{U}$ must be zero.

$$\langle x - \pi_{\mathcal{U}}(x), b_1 \rangle = 0$$
$$\langle x - \pi_{\mathcal{U}}(x), b_2 \rangle = 0$$

This can be applied to an M-dimensional space.

Introduction
0000

Linear Algebra
0000000●0000

Coding
0000000000

Projections

Generalizing to *M*-Dimensional Spaces

In an *M*-dimensional, define the following:

$$\lambda_{M \times 1} = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \end{bmatrix}, \quad B_{D \times M} = \begin{bmatrix} b_1, \cdots, b_M \end{bmatrix}$$

We can then write the equations we had in the previous slide as:

$$\pi_{\mathcal{U}}(x) = \sum_{i=1}^{M} \lambda_i b_i$$

$$\langle x - \pi_{\mathcal{U}}(x), b_i \rangle = 0, \qquad i = 1, \cdots, M$$

where *x* now is a *D*-dimensional vector. But as we saw before, we can write the first equation as:

$$\pi_{\mathcal{U}}(x) = \sum_{i=1}^{M} \lambda_i b_i = B\lambda$$

Introduction
0000

Linear Algebra
0000000000000

Coding
0000000000

Projections

Generalizing to *M*-Dimensional Spaces

Plug in the matrix representation to get:

$$\langle x - B\lambda, b_i \rangle = 0$$

The inner product satisfies the linearity property, which means that:

$$\langle x - B\lambda, b_i \rangle = \langle x, b_i \rangle - \langle B\lambda, b_i \rangle = 0$$

Applying the dot-product as an inner product space we get:

$$\langle x, b_i \rangle - \langle B\lambda, b_i \rangle = 0$$
$$x^T b_i - \lambda^T B^T b_i = 0$$

The last line above can be seen as a set of conditions for all basis b_i in $i = 1, \cdots, M$, which allows us to write it as:

$$x^T B - \lambda^T B^T B = 0_{M \times 1}$$

Introduction
0000

Linear Algebra
0000000000000

Coding
0000000000

Projections

Generalizing to *M*-Dimensional Spaces

Remember, we are interested in finding λ that allows us to perform the orthogonal projection of an M-dimensional vector onto a D-dimensional space. We can re-arrange the above and play around with it:

$$x^T B - \lambda^T B^T B = 0_{M \times 1}$$
$$\lambda^T B^T B = x^T B$$
$$\lambda^T = x^T B (B^T B)^{-1}$$
$$\lambda = (B^T B)^{-1} B^T x$$

where above we use the fact that $(B^T B)^{-1} = [(B^T B)^{-1}]^T$ since $(B^T B)^{-1}$ is a symmetric matrix, so its inverse equals the original matrix. **Does this look familiar?**

Introduction
0000

Linear Algebra
0000000000●0

Coding
0000000000

Projections

Generalizing to M-Dimensional Spaces

To recap, we said that:

$$\pi_{\mathcal{U}}(x) = B\lambda$$
$$\lambda = (B^T B)^{-1} B^T x$$

Combine the two equations and we have that:

$$\pi_{\mathcal{U}}(x) = \underbrace{B(B^T B)^{-1} B^T}_{\text{projection matrix}} x$$

So the matrix $B(B^T B)^{-1} B^T$ projects the M-dimensional vector x onto the D-dimensional space spanned by the basis vectors in B.

Coding

Useful Tools

If you plan on working with data, you must know how to code.

- **Clean code:** Your code must be easy to understand.
- **Reproducible code:** More and more the profession requires code to be easily reproduced in any system (how can I trust a paper if I cannot replicate it?).

For most of us, knowing one or two high-level programming languages is enough, but you must know how to use it efficiently.

- **Efficiency:** Can I make this code run faster?

Before we dive into R, let's look at some useful results.

Introduction
0000

Linear Algebra
00000000000

Coding
0000000000

Leverage

Leverage

Depending on the data we are working with, we might want to know (probably do) which observations are more or less influential in computing our estimator. Recall that we derived the projection matrix before as:

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

We can get the diagonal elements of the projection matrix by looking at the i-th observation in matrix X to get

$$h_{ii} = x_i(X'X)^{-1}x_i'$$

Intuitively, this tells us how much the matrix P that projects Y onto the space spanned by the columns of X moves by a given observation. This is often called the leverage

Jackknife

Imagine you want to estimate the impact of income on some variable, but you have Mark Zuckerberg as an observation. This will throw off your estimate, so you might want to know what your estimator would be like without that observation.

You could simply remove it from the data and re-run your OLS, but what if you want to do this for each observation to see how much your estimator change? That's a lot of work. Instead, note that the <u>Sherman-Morrison formula</u> states that for a non-singular matrix A and vector b the following holds:

$$(A - bb')^{-1} = A^{-1} + (1 - b'A^{-1}b)^{-1}A^{-1}bb'A^{-1}$$

Introduction
0000

Linear Algebra
00000000000

Coding
0000●00000

Leverage

Jackknife

Using this formula, we know then that:

$$(X'X - x_i x_i') = (X'X)^{-1} + (1 - \underbrace{x_i'(X'X)^{-1} x_i}_{=h_{ii}})^{-1}(X'X)^{-1} x_i x_i'(X'X)^{-1}$$

We can calculate the **leave-one-out** estimator by removing observation i from our dataset:

$$\hat{\beta}_{-i} = (X'X - x_i x_i')^{-1}(X'Y - x_i y_i)$$

Plug in the result from the Sherman-Morrison formula (cont.)

Introduction
0000

Linear Algebra
00000000000

Coding
0000000000

Leverage

Jackknife

The leave-one-out estimator can be written as:

$$
\begin{aligned}
\hat{\beta}_{-i} &= \underbrace{(X'X)^{-1}X'Y}_{=\hat{\beta}} - (X'X)^{-1}x_i y_i \\
&+ (1 - h_{ii})^{-1}(X'X)^{-1}x_i x_i'(X'X)^{-1}(X'Y - x_i y_i) \\
&= \hat{\beta} - (X'X)^{-1}x_i y_i \\
&+ (1 - h_{ii})^{-1}(X'X)^{-1}(x_i x_i' \underbrace{(X'X)^{-1}X'Y}_{=\hat{\beta}} - x_i \underbrace{x_i'(X'X)^{-1}x_i}_{=h_{ii}} y_i) \\
&= \hat{\beta} - (1 - h_{ii})(X'X)^{-1}x_i \left((1 - h_{ii})y_i - x_i'\hat{\beta} + h_{ii}y_i \right) \\
&= \hat{\beta} - (1 - h_{ii})(X'X)^{-1}x_i \left(y_i - x_i'\hat{\beta} \right)
\end{aligned}
$$

Introduction
0000

Linear Algebra
00000000000

Coding
0000000●000

Leverage

Jackknife

But $y_i - x_i'\hat{\beta}$ is the residual $\hat{e}_i$ of the OLS regression, which then gives the final result

$$\hat{\beta}_{-i} = \hat{\beta} - (X'X)^{-1}x_i \underbrace{(1 - h_{ii})\hat{e}_i}_{\equiv \tilde{e}_i}$$

The term $\tilde{e}_i$ is the leave-one-out residual or prediction error. The expression above allows us to see whether $\hat{\beta}$ moves a lot or not if we take observation i out.

Importantly, this expression shows us that we don't have to run lots of regressions to get the the estimators we want.

Introduction
0000

Linear Algebra
00000000000

Coding
000000000000

Standard errors

OLS Variance

Recall that we can calculate the conditional variance of the OLS estimator as follows:

$$\text{Var}(\hat{\beta}_{OLS} \mid X) = \text{Var}\left((X'X)^{-1}X'Y \mid X\right)$$
$$= (X'X)^{-1}X' \text{Var}(Y \mid X)X(X'X)^{-1}$$

But $Y = X\beta + \varepsilon \implies \text{Var}(Y \mid X) = \text{Var}(\epsilon \mid X) = \Omega$, where β and ϵ are the true values. This means we can write the conditional variance of our estimator as:

$$\text{Var}(\hat{\beta}_{OLS} \mid X) = (X'X)^{-1}(X'\Omega X)(X'X)^{-1}$$

The middle term is simply

$$(X'\Omega X) = \sum_{i=1}^{n} x_i x_i' \sigma_i^2$$

Introduction
0000

Linear Algebra
00000000000

Coding
0000000000●0

Standard errors

OLS Variance

If we know the true values of ε, we can use them to calculate the variance:

$$\text{Var}(\hat{\beta}_{OLS} \mid X) = (X'X)^{-1}\left(\sum_{i=1}^{n} x_i x_i' \varepsilon_i^2\right)(X'X)^{-1}$$

This estimator is clearly unbiased:

$$\mathbb{E}[\text{Var}(\hat{\beta}_{OLS} \mid X)] = (X'X)^{-1}\left(\sum_{i=1}^{n} x_i x_i' \mathbb{E}[\varepsilon_i^2]\right)(X'X)^{-1}$$

$$= (X'X)^{-1}\left(\sum_{i=1}^{n} x_i x_i' \sigma_i^2\right)(X'X)^{-1}$$

But we don't really know the true value of ε.

Introduction
0000

Linear Algebra
00000000000

Coding
000000000●

Standard errors

Heteroskedasticity Consistent Estimates

In R, we have a few options we can use to compute our standard errors:

$$\text{Var}(\hat{\beta}_{OLS} \mid X)_{HC0} = (X'X)^{-1}\left(\sum_{i=1}^{n} x_i x_i' \hat{e}_i^2\right)(X'X)^{-1}$$

$$\text{Var}(\hat{\beta}_{OLS} \mid X)_{HC1} = (X'X)^{-1}\left(\sum_{i=1}^{n} x_i x_i' \hat{e}_i^2\right)(X'X)^{-1}\left(\frac{n}{n-k}\right)$$

$$\text{Var}(\hat{\beta}_{OLS} \mid X)_{HC2} = (X'X)^{-1}\left(\sum_{i=1}^{n} (1-h_{ii})^{-1} x_i x_i' \hat{e}_i^2\right)(X'X)^{-1}$$

$$\text{Var}(\hat{\beta}_{OLS} \mid X)_{HC3} = (X'X)^{-1}\left(\sum_{i=1}^{n} (1-h_{ii})^{-2} x_i x_i' \hat{e}_i^2\right)(X'X)^{-1}$$

The last two we compute by summing over the leave-one-out estimates.