VE472 Lecture 2

Jing Liu

UM-SJTU Joint Institute

Summer

Small Vs Big

Q: What is the major difference between traditional and contemporary analysis?

- To understand why there is a big difference, one must see the difference in

1. The origin of the dataset
2. The size of the dataset
3. The purpose of investigating the dataset

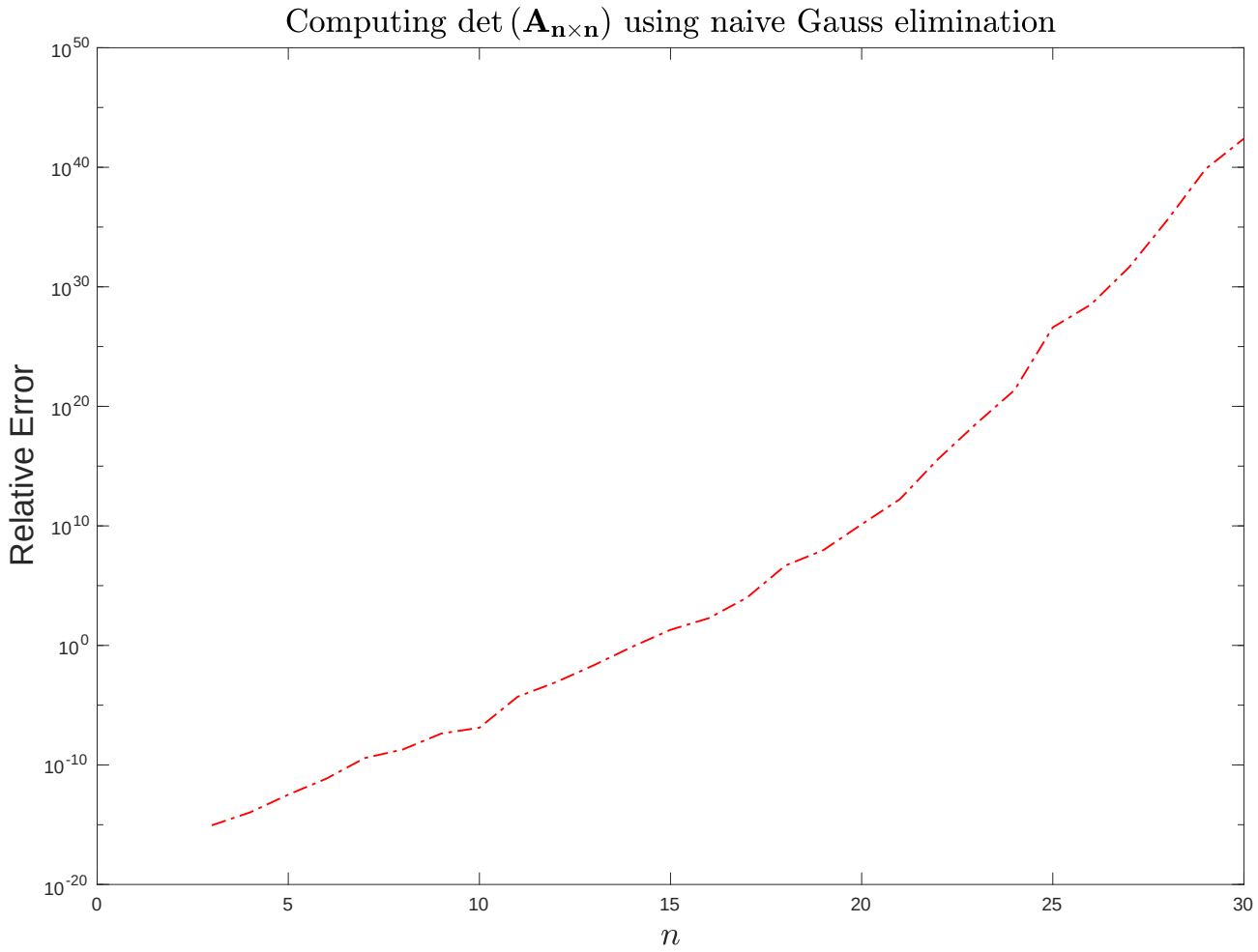
- Traditionally, data are often some kind of random sample collected from a carefully designed experiment, thus it is naturally very small due to the cost.
- A typical big dataset nowadays often is an opportunistic sample.
- Traditional methods are based on hypothesise-and-tests, which are heavily assumption-oriented, and the purpose is to explain/discover relationships.
- Contemporary methods are predictive in nature, and the purpose is more quantitive, so are heavily algorithm-oriented due to the size of the datasets.

- Recall determinant, $\det(\mathbf{A})$, where $\mathbf{A}$ is an $n \times n$ matrix, is a useful concept.
- Below gives the number of arithmetic operations involved in each method.

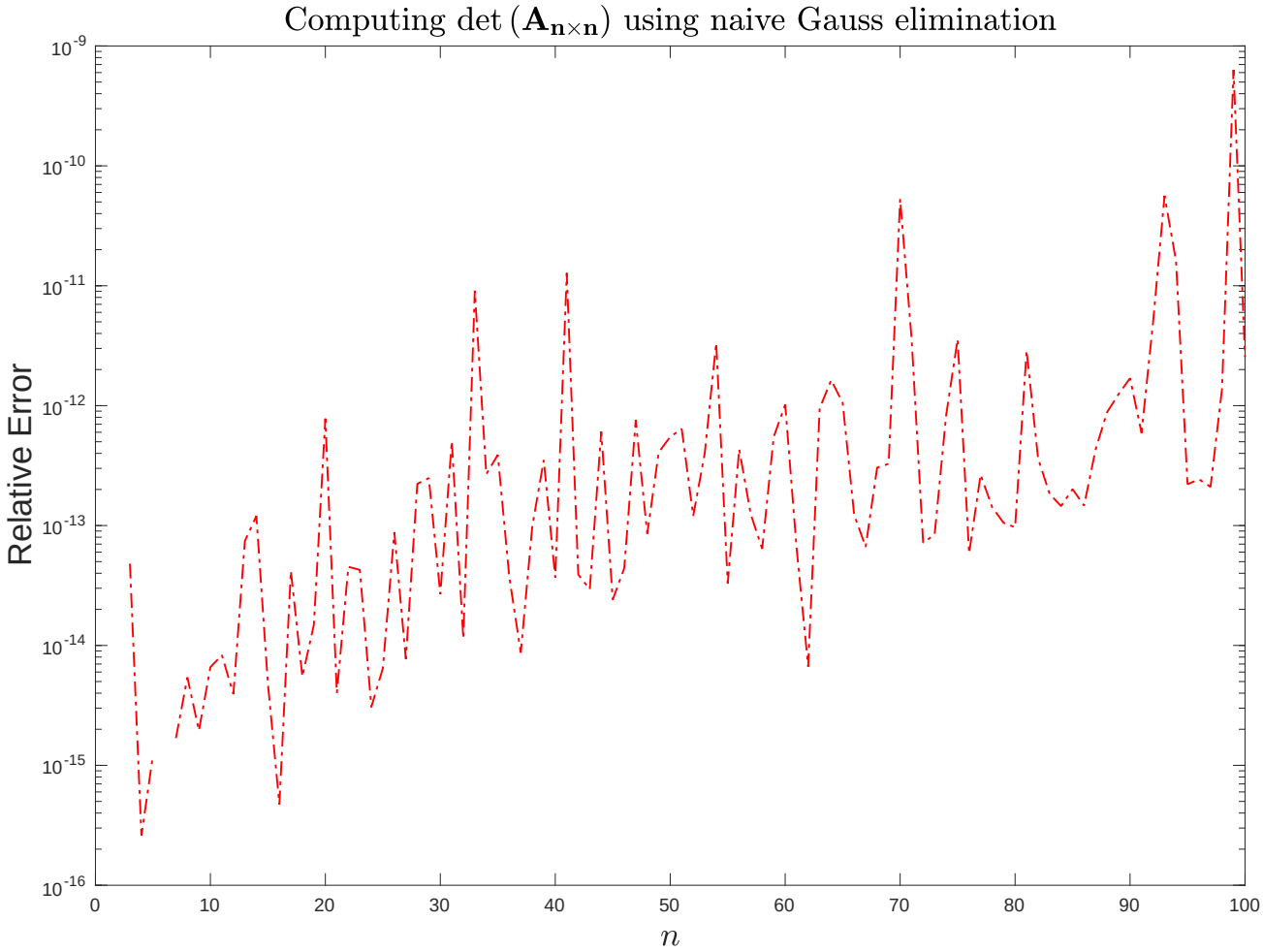
	Leibiniz		Laplace		Gauss	
n	$+$	$\times$	$+$	$\times$	$+$	$\times$
2	1	2	1	2	1	3
3	5	12	5	9	5	10
4	23	72	23	40	14	23
5	119	480	119	205	30	44
10	3,628,799	32,659,200	3,628,799	6,235,300	285	339

- Note computing the determinant of a 50×50 matrix directly would take a computer 10^{40} years! that is, more than 10^{30} times the age of the universe!
- The point here is the computational cost of using the wrong method may grow surprising fast as the data size increases.
- We will discuss in terms of computational cost but not in a rigorous way.
- In additional to cost, we will occasionally discuss the stability of a method.

Vandermonde matrix



Random invertible matrix



- Being unstable for a big dataset is often due to the fact our computer does floating-point computation and the machine precision is limited, but...
- To illustrate the idea in a simple context, consider solving the followings,

$$\mathbf{A}\mathbf{x}_1 = \mathbf{b} \qquad \text{and} \qquad \mathbf{A}\mathbf{x}_2 = \mathbf{c}$$

where $\mathbf{A} = \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1/3 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 3/2 \\ 1 \end{bmatrix}$ and $\mathbf{c} = \begin{bmatrix} 3/2 \\ 5/6 \end{bmatrix}$, without any rounding.

Q: Do you expect $\mathbf{x}_1$ and $\mathbf{x}_2$ to be related? or perhaps close to one another?

- In fact, the solutions are $\mathbf{x}_1 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ and $\mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, which are not close at all!

Q: What is the significance of this observation?

- The stability of a method used, thus our conclusion based on the data and the method, is not only depended on numerical rounding/machine precision.
- As our data become big as it being increasingly complex, i.e. k grows, the stability of many methods tends to deteriorate even without rounding.

Q: What do we mean by an unbiased estimator?

Definition

The bias of a point estimator $\hat{\theta}$ of a θ is the difference

$$\mathbb{E}\left[\hat{\theta}\right] - \theta$$

An estimator whose bias is identically zero is called unbiased, that is,

$$\mathbb{E}\left[\hat{\theta}\right] = \theta \qquad \text{for all} \quad \theta$$

Q: How about being consistent? Why is it more relevant for large datasets?

Definition

A consistent estimator $\hat{\theta}$ of θ gives estimates $\hat{\theta}_n$ that converge in probability to θ

$$\lim_{n\to\infty} \Pr\left[\left|\hat{\theta}_n - \theta\right| \geq \epsilon\right] = 0 \qquad \text{for all} \quad \epsilon > 0$$

Bashing Unbiasedness

- Let $\mathcal{S} = \{X_1, X_2, \ldots, X_n\}$ be a random sample from a population with

$$\mathbb{E}[X_i] = \mu \qquad \text{and} \qquad \mathrm{Var}[X_i] = \sigma^2 < \infty \qquad \text{for all } i.$$

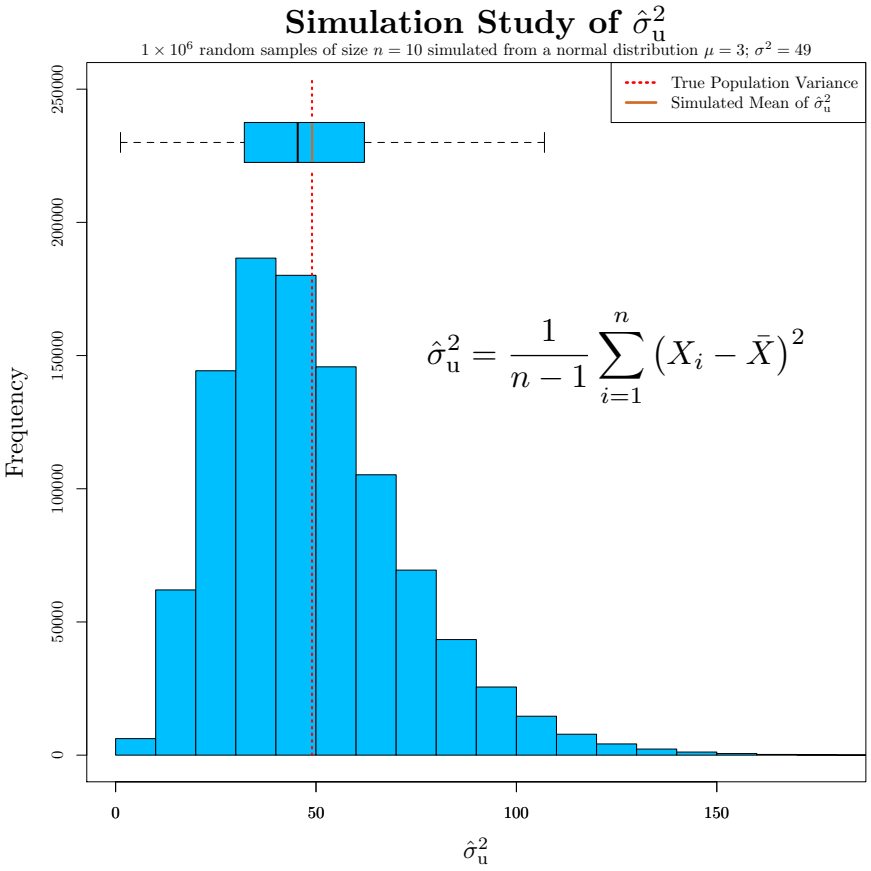
Q: What is the commonly used unbiased estimator of σ^2 given $\mathcal{S}$?

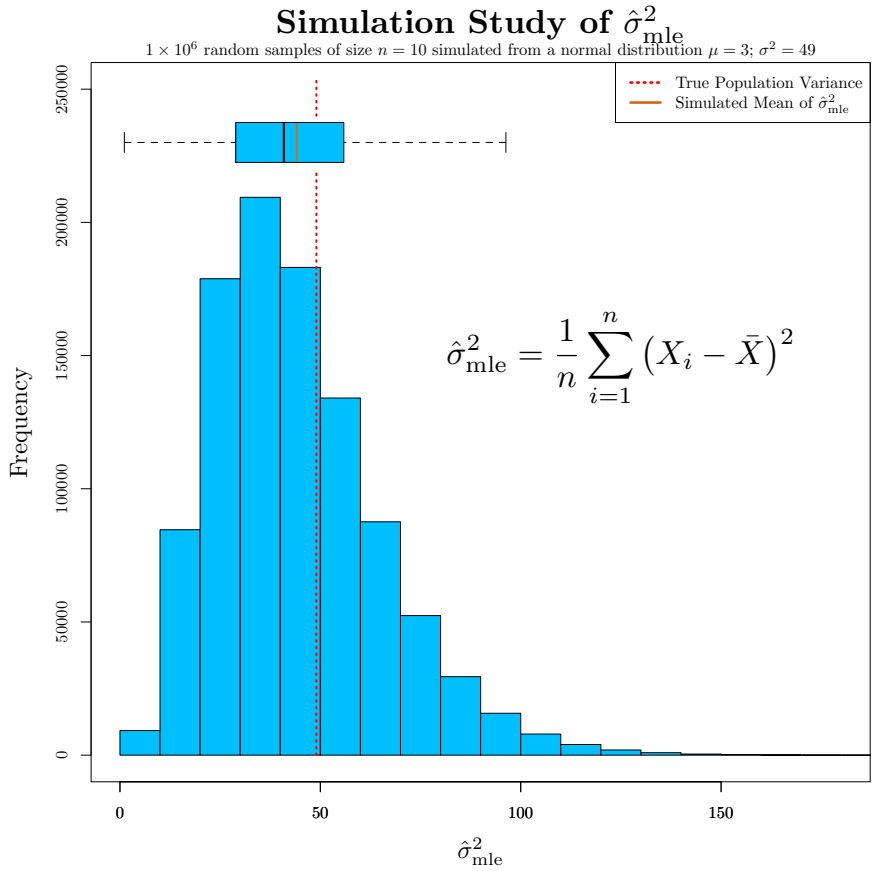
$$\hat{\sigma}^2_{\mathrm{u}} = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2 \qquad \text{where} \quad \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

- This estimator $\hat{\sigma}^2_u$ can be shown to be consistent as well as being unbiased.
- Recall the maximum likelihood estimator for σ^2 is also consistent but biased

$$\hat{\sigma}^2_{\mathrm{MLE}} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2$$

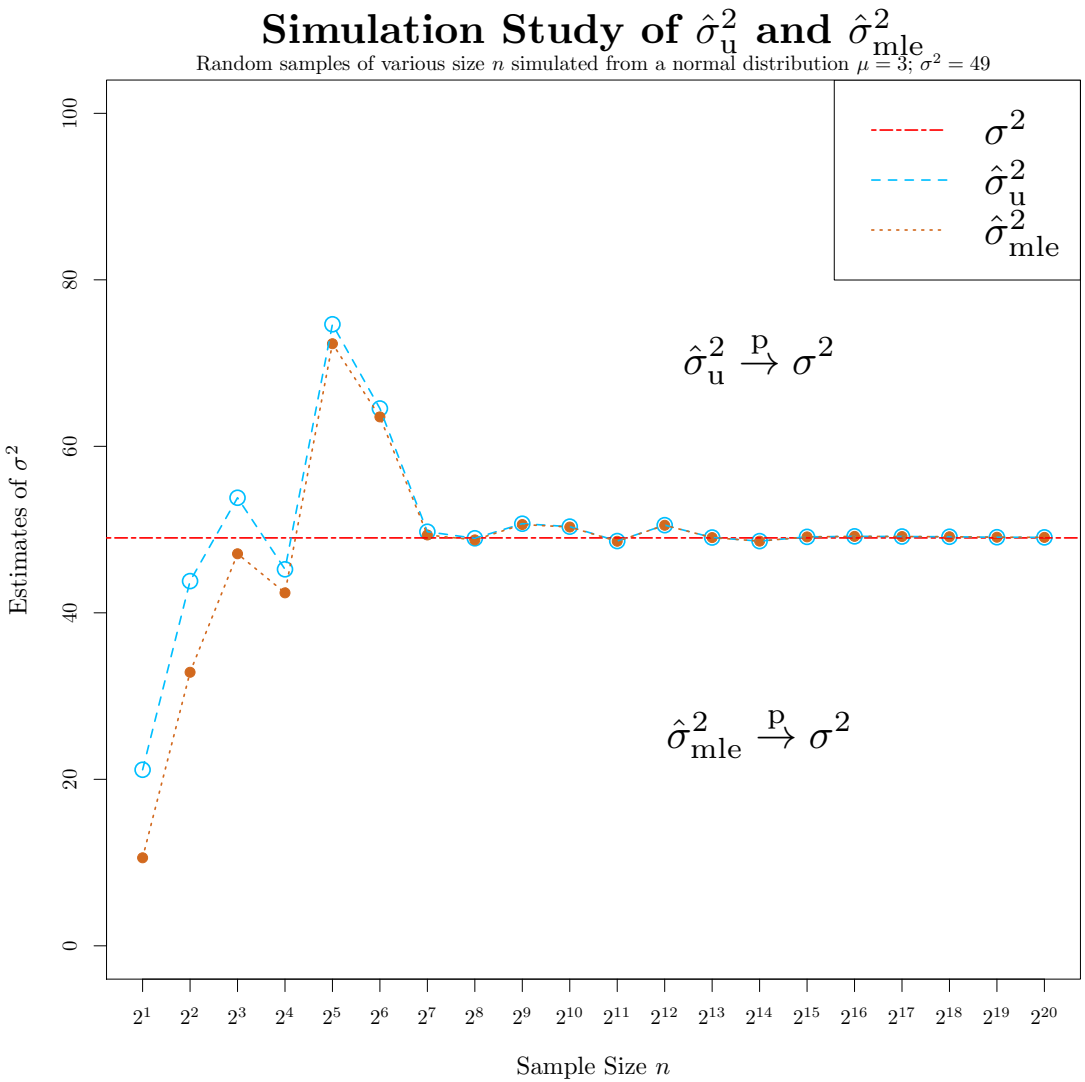
- On average an unbiased estimator is clearly better for small datasets.





Q: Should we still obsess about the unbiased estimator when data is large?

- Unbiasedness is *not* relevant for large datasets as long as $\hat{\sigma}^2$ is consistent.



Q: Should we use $\hat{\sigma}^2_{\mathrm{u}}$ or $\hat{\sigma}^2_{\mathrm{mle}}$ to estimate σ^2?

Definition

An estimator $\hat{\theta}$ that achieves optimality on estimating θ according to a particular *loss function* is said to be efficient with respect to the *loss function*.

- Traditionally, the *loss function* is chosen to be quadratic, that is,

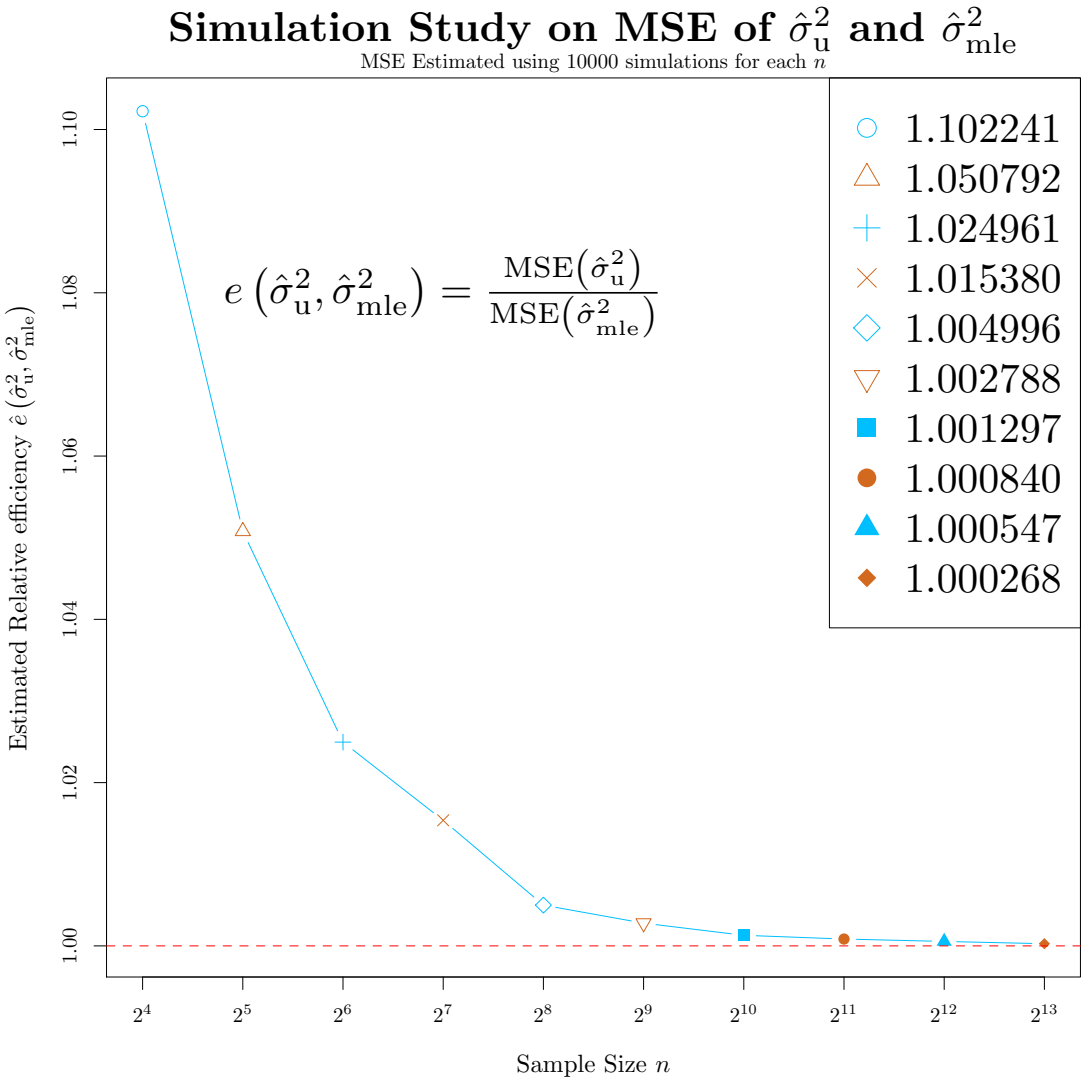
$$\mathrm{MSE}(\hat{\theta}) = \mathbb{E}\left[\left(\hat{\theta} - \theta\right)^2\right] = \mathrm{Var}\left[\hat{\theta}\right] + \Big(\underbrace{\mathbb{E}\left[\hat{\theta}\right] - \theta}_{\text{Bias}}\Big)^2$$

mean square error

  is used to judge the quality of an estimator, i.e. the smaller MSE, the better.

- Classically, only unbiased estimators are considered and judged, which means we choose the unbiased estimator that has the smallest variance.
- However, the property of unbiasedness is not that relevant for large datasets.

- The MSE of $\hat{\sigma}^2_{\text{mle}}$ is always smaller than $\hat{\sigma}^2_{\text{u}}$, thus better despite being biased.



- Consider k independent variables, X_j, values of which are stored in a matrix

$$\mathbf{X}_{n\times(k+1)} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1j} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2j} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \cdots & x_{ij} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nj} & \cdots & x_{nk} \end{bmatrix}$$

where each row of $\mathbf{X}$ corresponds to a case, and a vector

$$\mathbf{y}_{n\times 1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

is used to store the value of the corresponding dependent variable Y.

- In traditional regression analysis, the following linear model is often used

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{k+1}$ are fixed but unknown, while $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is random such that

$$\mathbb{E}[\boldsymbol{\varepsilon}] = 0 \qquad \text{and} \qquad \mathrm{Var}[\boldsymbol{\varepsilon}] = \sigma^2 \mathbf{I}$$

- One estimate of $\boldsymbol{\beta}$ is the vector $\mathbf{b}$ that minimises the sum of squared "errors"

$$\sum_{i=1}^{n} (y_i - \mathbf{X}_{i,}\mathbf{b})^2$$

where $\mathbf{X}_{i,}\mathbf{b}$ is essentially the dot product between the ith row of $\mathbf{X}$ and $\mathbf{b}$.

- This estimate is known as the least squares estimate of $\boldsymbol{\beta}$, denoted by $\mathbf{b}_{\text{lse}}$.
- The corresponding estimator of $\mathbf{b}_{\text{lse}}$ is often denoted by $\hat{\boldsymbol{\beta}}_{\text{lse}}$.

- It is useful to realise and understand $\mathbf{b}_{\text{lse}}$ as the solution to the following

$$\arg\min_{\mathbf{b}\in\mathbb{R}^{k+1}} \|\mathbf{y} - \mathbf{X}\mathbf{b}\|^2$$

  where $\|\cdot\|$ denote the usual ℓ_2 norm in $\mathbb{R}^n$.
- In other words, $\mathbf{b}_{\text{lse}}$ is the minimiser of the squared distance between

$$\mathbf{y}$$

  and the image of $\mathbf{b}$ under the linear transformation represented by $\mathbf{X}$

$$\mathbf{X}\mathbf{b}$$

Theorem 0.1

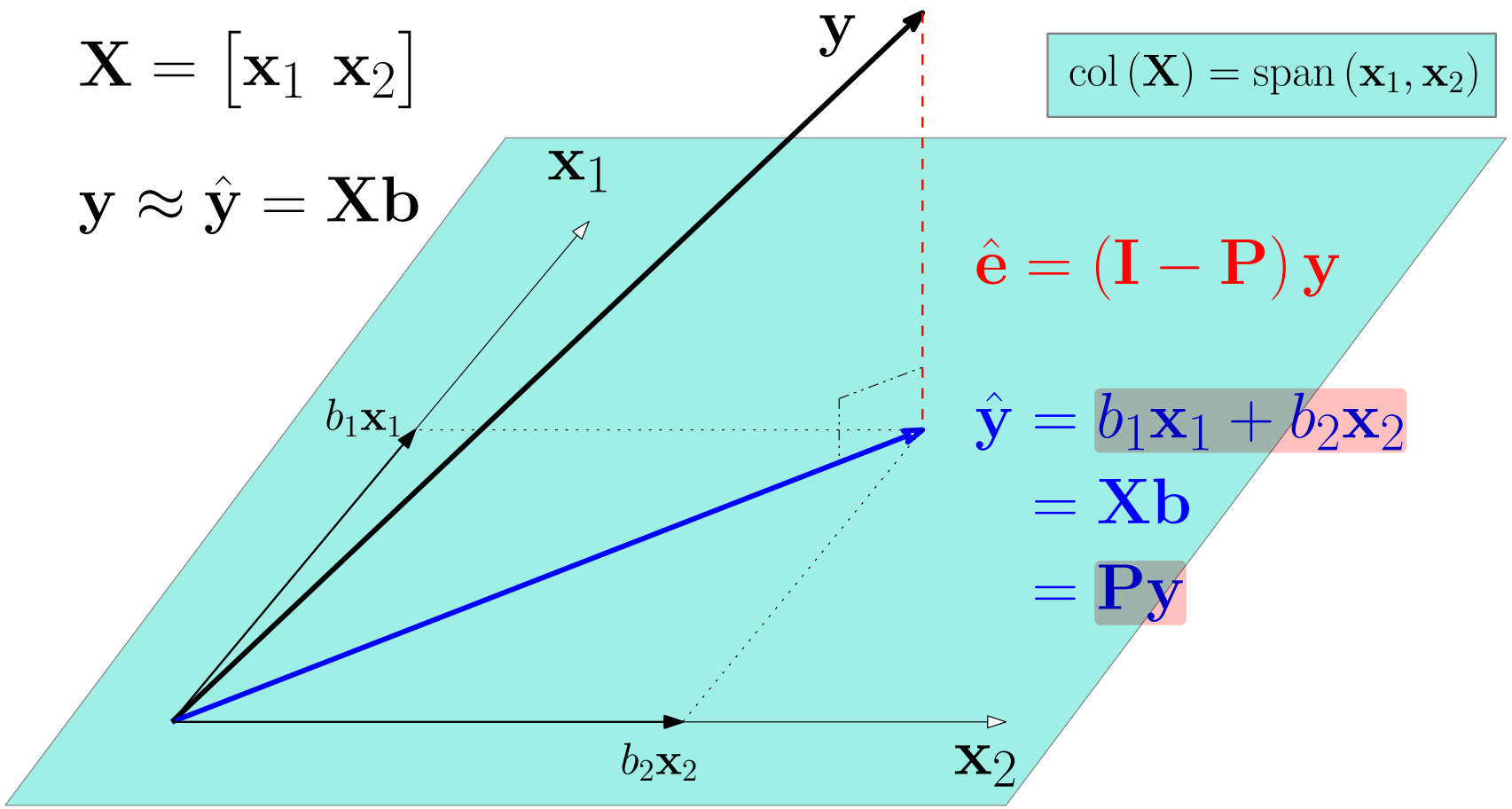
Suppose $\mathbf{b} \in \mathbb{R}^{k+1}$, then $\mathbf{b}$ is the least squares estimate of β if and only if $\mathbf{X}\mathbf{b}$ is the orthogonal projection of $\mathbf{y}$ onto the column space of the matrix $\mathbf{X}$,

$$\text{col}\,(\mathbf{X})$$

Least squares fit as projection

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{bmatrix}$$

$$\mathbf{y} \approx \hat{\mathbf{y}} = \mathbf{X}\mathbf{b}$$



$$\text{col}(\mathbf{X}) = \text{span}(\mathbf{x}_1, \mathbf{x}_2)$$

$$\hat{\mathbf{e}} = (\mathbf{I} - \mathbf{P})\,\mathbf{y}$$

$$\begin{aligned} \hat{\mathbf{y}} &= b_1\mathbf{x}_1 + b_2\mathbf{x}_2 \\ &= \mathbf{X}\mathbf{b} \\ &= \mathbf{P}\mathbf{y} \end{aligned}$$

Q: How to find the projection $\hat{\mathbf{y}}$? Is $\hat{\mathbf{y}}$ unique? How to find $\mathbf{b}$? Is $\mathbf{b}$ unique?

projection matrix yes no

Theorem 0.2

If $\mathbf{X}$ *is full rank, then the orthogonal projection of* $\mathbf{y}$ *onto* $\mathrm{col}(\mathbf{X})$ *is given by*

n >> k (row >> column)
columns are linearly independent

$$\mathbf{P}\mathbf{y}$$

幂等的

where $\mathbf{P}$ *is known as the projection matrix, which is symmetric and idempotent,*

$$\mathbf{P} = \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}$$

- One can reach the same matrix $\mathbf{P}$ without the concept of projection, but the concept of projection will provide insights into ways of dealing with big data.
- Theoretically, the last two theorems give us the least squares estimator

$$\mathbf{X}\mathbf{b} = \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \implies \underbrace{\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{b} = \mathbf{X}^{\mathrm{T}}\mathbf{y}}_{\text{normal equation}} \implies \mathbf{b} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

- However, unless $\mathbf{X}$ is full rank, the inverse will not exist.

b is unique iff X^TX is invertible
otherwise, inf number of b solutions

- Even if it exists, it is too slow for a big dataset and shall never be used.
- Directly solving the normal equation without explicitly computing the inverse

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{b} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

known as Gaussian elimination with pivoting, is a faster alternative.

```
>> n = 1000000; X = rand(n, 100); y = randn(n, 1);
>> XX = transpose(X)*X; yy = transpose(X)*y;
tic; for i = 1:10000
     bhat = inv(XX)*yy;
end; toc;
```

finding inverse -> slow

```
tic; for i = 1:10000
     [L, U] = lu(XX);
     bhat = U\(L\yy);
end; toc;
```

```
Elapsed time is 3.130701 seconds.
Elapsed time is 0.884935 seconds.
```

choleskey decomposition

Q: Do you know anything that might even be faster for arbitrary data matrices?

```
>> n = 1000000; X = rand(n, 100); y = randn(n, 1);
>> XX = transpose(X)*X; yy = transpose(X)*y;
tic; for i = 1:10000
     bhat = inv(XX)*yy;
end; toc;
tic; for i = 1:10000
     [L, U] = lu(XX);
     bhat = U\(L\yy);
end; toc;
tic; for i = 1:10000
     C = chol(XX, 'lower');
     bhat =  transpose(C)\(C\yy);
end; toc;
```

lu is much slower than chol

roughly the same cost

```
Elapsed time is 3.130701 seconds.
Elapsed time is 0.884935 seconds.
Elapsed time is 0.367677 seconds.
```