統計計算與模擬

政治大學統計系余清祥

2023年3月21日～27日

第三單元：矩陣運算

http://csyue.nccu.edu.tw

矩陣運算(Matrix Computation)

矩陣運算在統計分析上扮演非常重要的角色，包括以下方法：

- Multiple Regression
- Generalized Linear Model
- Multivariate Analysis
- Time Series
- Other topics (Random variables)

註：建議同學複習線性代數，可以幫助瞭解本次講義。

Multiple Regression

- In a multiple regression problem, we want to approximate vector Y by fitted values $\hat{Y}$ (a linear function of a set *p* predictors), i.e.,

$$Y = X\beta + \varepsilon, \;\; \varepsilon \sim N(0, \sigma^2 I_{n\times n})$$

$$\Rightarrow (X'X)\hat{\beta} = X'Y \quad (\textit{Normal equation})$$

$$\Rightarrow \hat{\beta} = (X'X)^{-1}X'Y \equiv AY.$$

(if the inverse can be solved.)

Note: The normal equation is hardly solved directly, unless it is necessary.

- In other words, we need to be familiar with matrix computation, such as matrix product and matrix inverse, i.e., $X'Y$ and $(X'X)^{-1}$.

- If the left hand side matrix $(X'X)^{-1}$ is a upper (or lower) triangular matrix, then the estimate $\hat{\beta}$ can be solved easily,

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{n1} \\ 0 & a_{22} & a_{23} & \cdots & a_{n2} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & a_{p-1,n-1} & a_{p,n-1} \\ 0 & 0 & \cdots & 0 & a_{pn} \end{pmatrix} \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_{p-1} \\ \hat{\beta}_p \end{bmatrix} = \begin{bmatrix} (X'Y)_1 \\ (X'Y)_2 \\ \vdots \\ (X'Y)_{n-1} \\ (X'Y)_n \end{bmatrix}$$

矩陣與向量空間
(Matrix and Vector Space)

- 如果$A = ZX$，A: $m\times n$、Z: $m\times k$、X: $k\times n$。以向量的角度而言，A矩陣的行向量是Z矩陣行向量的線性組合： $\tilde{A}_j = \sum_{i=1}^{k} X_{ij}\tilde{Z}_i$

$$A = \begin{bmatrix} | & | & & | \\ A_1 & A_2 & \cdots & A_n \\ | & | & & | \end{bmatrix}, \quad Z = \begin{bmatrix} | & | & & | \\ Z_1 & Z_2 & \cdots & Z_k \\ | & | & & | \end{bmatrix},$$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{kn} \end{bmatrix}$$

Gauss Elimination

- Gauss Elimination is a process of "row" operation, such as adding multiples of rows and interchanging rows, that produces a triangular system in the end.

$$\left[(X'X)^{-1} \mid X'Y\right] \rightarrow \cdots \rightarrow [U \mid (X'Y)^{*}]$$

- Gauss Elimination can be used to construct matrix inverse as well.

$$\left[(X'X) \mid I\right] \rightarrow \cdots \rightarrow \left[I \mid (X'X)^{-1}\right]$$

- An Example of Gauss Elimination:

$$\begin{pmatrix} 1 & 1 & 2 & 7 \\ 2 & 5 & -1 & -4 \\ 2 & 1 & -1 & 0 \end{pmatrix} \to \begin{pmatrix} 2 & 5 & -1 & -4 \\ 1 & 1 & 2 & 7 \\ 2 & 1 & -1 & 0 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & 5/2 & -1/2 & -2 \\ 1 & 1 & 2 & 7 \\ 2 & 1 & -1 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 5/2 & -1/2 & -2 \\ 0 & -3/2 & 5/2 & 9 \\ 0 & -4 & 0 & 4 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & 5/2 & -1/2 & -2 \\ 0 & -4 & 0 & 4 \\ 0 & -3/2 & 5/2 & 9 \end{pmatrix} \to \begin{pmatrix} 1 & 5/2 & -1/2 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & -3/2 & 5/2 & 9 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & 5/2 & -1/2 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 5/2 & 15/2 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix}.$$

- The idea of Gauss elimination is fine, but it would require a lot of computations.
→ For example, let X be an $n \times p$ matrix. Then we need to compute X'X first and then find the upper triangular matrix for (X'X). This requires a number of $n^2 \times p$ multiplications for (X'X) and about another $p \times p(p-1)/2 \cong O(p^3)$ multiplications for the upper triangular matrix.

何謂大O及小O？

- 大O (big o)及小O (little o)

→常見的定義為

$$a_n = O(n^{\lambda}) \Leftrightarrow n^{-\lambda} a_n \text{ is bounded}$$

$$a_n = o(n^{\lambda}) \Leftrightarrow n^{-\lambda} a_n \to 0 \text{ as } n \to \infty$$

註：大O代表同樣的收斂速度，小O代表較快的收斂的速度。

Cholesky Decomposition

- If A is positive semi-definite, there exists an lower triangular matrix L such that

$$LL' = A,$$

$$i.e.,\ a_{ij} = \sum_{k=1}^{p} l_{ik} l_{jk} \quad and\ so$$

$$a_{ij} = \sum_{k=1}^{i} l_{ik} l_{jk} = \sum_{k=1}^{i-1} l_{ik} l_{jk} + l_{ii} l_{ij},$$

since $l_{cr} = 0$ for $r > c$.

- Thus, the elements of L equal to

$$l_{ii} = (a_{ii} - \sum_{k=1}^{i-1} l_{ik} l_{jk})^{1/2};$$

$$l_{ij} = (a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}) / l_{ii}.$$

Notes: (1) If A is "positive semi-definite" if for all vectors $v \in R^p$, we have $v'Av \geq 0$, where A is a $p \times p$ matrix. If $v'Av = 0$ if and only if $v = 0$, then A is "positive definite."

(2) In R, Cholesky decomposition requires symmetric and positive definite.

Applications of Cholesky decomposition:

- Simulation of correlated random variables (and also multivariate distributions).
- Regression

$$X'X\hat{\beta} = X'y \Rightarrow LL'\hat{\beta} = X'y$$

$$solve\ L\theta = X'y\ \ for\ \ \theta = L'\hat{\beta}$$

$$and\ backsolve\ L'\hat{\beta} = \theta\ \ for\ \ \hat{\beta}.$$

- Determinant of a symmetric matrix, $A = LL'$

$$\det(A) = \det(LL') = \det(L)^2 = \prod_i l_{ii}^2$$

- Example of Cholesky decomposition

→We want to generate two random variables,

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix} \right)$$

Let ρ=0.5. We generate X_1 & X_2 from N(0,1).

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0.5 & \sqrt{3}/2 \end{pmatrix} = chol(A) = chol\left(\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \right)$$

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0.5 & \sqrt{3}/2 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

- Cholesky decomposition (R-code)

→We shall demonstrate using R.

```
x=matrix(rnorm(2000),ncol=1000)
  apply(x,1,var)
    apply(x,1,summary)
      cor(x[1,],x[2,])
A=matrix(c(1,0.5,0.5,1),ncol=2)
  a=t(chol(A))
   x1=a%*%x
     apply(x1,1,var)
       apply(x1,1,summary)
         cor(x1[1,],x1[2,])
           ks.test(x1[1,],"pnorm")
            ks.test(x1[2,],"pnorm")
```

- Cholesky decomposition (Conti.)

→Another way to generate is via

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

and we have
$$\begin{cases} a^2 + b^2 = 1 \\ c^2 + d^2 = 1 \\ ad + bc = 1/2 \end{cases}$$

Possible solution includes

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \dfrac{1}{\sqrt{2}} & \dfrac{1}{\sqrt{2}} \\ \dfrac{\sqrt{2}+\sqrt{6}}{4} & \dfrac{\sqrt{2}-\sqrt{6}}{4} \end{pmatrix}$$

QR Decomposition

- If we can find matrices Q and R such that X = QR, where Q is orthogonal (Q'Q = I) and R is upper triangular matrices, then

$$(X'X)\hat{\beta} = X'Y$$

$$\Leftrightarrow (QR)'(QR)\hat{\beta} = (QR)'Y$$

$$\Leftrightarrow R'Q'QR\hat{\beta} = R'Q'Y$$

$$\Leftrightarrow (R'R)\hat{\beta} = R'Q'Y$$

$$\Leftrightarrow R\hat{\beta} = Q'Y \quad (if\ R\ is\ full\ rank)$$

Gram-Schmidt Algorithm

- Gram-Schmidt algorithm is a famous algorithm for doing QR decomposition.
- <u>Algorithm:</u> (Q is $n \times p$ and R is $p \times p$.)

```
for (j in 1:p) {
  r[j,j] ← sqrt(sum(x[,j]^2))
    x[,j] ← x[,j]/r[j,j]
      if (j < p) for (k in (j+1):p) {
            r[j,k] ← sum(x[,j]*x[,k])
            x[,k] ← x[,k] – x[,j]*r[j,k]
      }
}
```

Note: Check the function "*qr*" in R and S-Plus.

QR指令在R的操作(QR in R)

- QR矩陣分解在 R 的指令為「qr」，R會提供矩陣的秩數(Rank)及其它相關訊息；如果需要求出Q及R矩陣，則要透過指令「qr.Q」及「qr.R」。

A=matrix(c(1:9),ncol=3)

qr(A)

qr.Q(qr(A))%%qr.R(qr(A))*

qr.Q(qr(A))%%t(qr.Q(qr(A)))*

t(qr.Q(qr(A)))%%qr.Q(qr(A))*

- Notes:

(1) Since R is upper triangular, i.e., R^{-1} is easy to obtain, $R\hat{\beta} = Q'y$

$$\Leftrightarrow \hat{\beta} = (R'R)^{-1} R'Q'y = R^{-1}(R^{-1})'X'y.$$

(2) The idea of the preceding algorithm is

$$X = QR$$

$$\Leftrightarrow X'X = R'Q'QR = R'R.$$

- Notes: (continued)

(3) If X is not of full rank, one of the columns will be very close to 0. Thus, $r_{jj} \approx 0$ and so there will be a divide-by-zero error.

(4) If we apply Gram-Schmidt algorithm to the augmented matrix $(X : y)$, the last column will become the residuals of $\hat{\varepsilon} = y - \hat{y}$.

$$Y = X\beta + \varepsilon \Leftrightarrow Q'Y = Q'X\beta + Q'\varepsilon$$

$$\Leftrightarrow \begin{pmatrix} Y_1^* \\ Y_2^* \end{pmatrix} = \begin{pmatrix} X_1^* \\ 0 \end{pmatrix}\beta + \begin{pmatrix} \varepsilon_1^* \\ \varepsilon_2^* \end{pmatrix}$$

$$Thus,\ |Y - X\beta|^2 = |Q'(Y - X\beta)|^2 = |Q'Y - Q'X\beta|^2$$

$$= \left| \begin{pmatrix} Y_1^* \\ Y_2^* \end{pmatrix} - \begin{pmatrix} X_1^*\beta \\ X_2^*\beta \end{pmatrix} \right|^2$$

$$= |Y_1^* - X_1^*\beta|^2 + |Y_2^* - X_2^*\beta|^2$$

$$= |Y_1^* - X_1^*\beta|^2 + |Y_2^*|^2,$$

$$i.e.,\ \hat{\beta} = (X_1^*)^{-1}Y_1^*\ \ and\ \ RSS = |Y_2^*|^2.$$

Other orthogonalization methods:

- Householder Transformation is a computationally efficient and numerically stable method for QR decomposition. The Householder transformation we have constructed are $n \times n$ matrices, and the transformation Q is the product of p such matrices.
- Given's rotation: The matrix X is reduced to upper triangular form $\begin{pmatrix} R \\ 0 \end{pmatrix}$ by making exactly subdiagonal element equal to zero at each step.

Sweep Operator

- The normal equation is not solved directly in the preceding methods. But sometimes we need to compute sums of squares and cross-products (SSCP) matrix.
- This is particularly useful in stepwise regression, since we need to compute the residual sum of squares (RSS) before and after a certain variable is added or removed.

- Sweep algorithm is also known as "Gauss-Jordan" algorithm.
- Consider the SSCP matrix

$$A = \begin{pmatrix} X'X & X'y \\ y'X & y'y \end{pmatrix}$$

where X is $n \times p$ and y is $p \times 1$.

- Applications of the Sweep operator to columns 1 through p of A results in the matrix

$$\tilde{A} = \begin{pmatrix} -(X'X)^{-1} & \hat{\beta} \\ \hat{\beta}' & RSS \end{pmatrix}.$$

Details of Sweep algorithm

Step 1: Row 1 $(times\ (X'X)^{-1})$

$$\begin{bmatrix} X'X & X'y & I_p \end{bmatrix} \rightarrow \begin{bmatrix} I_p & \hat{\beta} & (X'X)^{-1} \end{bmatrix}$$

Step 2: Row 2

$$\begin{bmatrix} y'X & y'y & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & y'(I-P_X)y & -\hat{\beta}' \end{bmatrix}$$

The operation of Row 2 is done via minus the right term of Row 1 times $y'X$.

- Notes:

(1) If you apply the Sweep operator to columns $i_1, i_2, \ldots, i_k$, you'll receive the results from regressing *y* on $X_{i_1}, X_{i_2}, \ldots, X_{i_k}$ — the corresponding elements in the last column will be the estimated regression coefficients, the $(p+1, p+1)$ element will contain RSS, and so forth.

- Notes: (continued)

(2) The Sweep operator has a simple inverse; the two together make it very easy to do stepwise regression. The SSCP matrix is symmetric, and any application of Sweep or its inverse result in a symmetric matrix, so one may take advantage of symmetric storage.

General Least Square (GLS)

- Consider the model $y = X\beta + \varepsilon,$

  where $\varepsilon \sim N(0, \sigma^2 V)$ with V unknown.
- Consider the Cholesky decomposition of V,

  $V = LL'$, and let $S' = (L)^{-1} = (L^{-1})$.

  let $y^* = S' y,\ X^* = S' X,\ and\ \varepsilon^* = S' \varepsilon.$

  Then $y^* = X^*\beta + \varepsilon^*$ with $\varepsilon^* \sim N(0, \sigma^2)$,

  and we may proceed with before.
- A special case (WLS): $V = diag\{v_1, \ldots, v_n\}$.

Eigenvalues, Eigenvectors, and Principal Component Analysis

- The notion of *principal components* refers to a collection of uncorrelated r.v.'s formed by linear combinations of a set of possibly correlated r.v.'s.
- Idea: From eigenvalues and eigenvectors, i.e., if x and λ are eigenvector and eigenvector of a symmetric positive semidefinite matrix A, then

$$Ax = \lambda x.$$

- If A is a symmetric positive semi-definite matrix ($p \times p$), then we can find orthogonal matrix $\mathit{\Gamma}$ such that $A = \mathit{\Gamma} \mathit{\Lambda} \mathit{\Gamma}'$, where

$$\Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{pmatrix}, \; with\ \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0.$$

- Note: This is called the *spectral decomposition* of A. The ith row of $\mathit{\Gamma}$ is the eigenvector of A which corresponds to the ith eigenvalue λ_i.

Write $\hat{\beta}^* = V\Lambda^+ U' y$ where

$$\Lambda^+ = \begin{pmatrix} \lambda_1^{-1} & & & \\ & \ddots & & 0 \\ & & \lambda_k^{-1} & \\ & 0 & & 0 \end{pmatrix}$$

We should note that the normal equation $X' X\hat{\beta} = X' y$ does not have a unique solution. $\hat{\beta}^*$ is the solution of the normal equations for which $\left\|\hat{\beta}\right\|^2 = \sum_j \hat{\beta}_j^2$ is minimized.

Power Method

- A naïve method for finding the eigenvalues of a matrix is via the fact that

$$A\underset{\sim}{x} = A\sum_{i=1}^{k} c_i \underset{\sim}{v_i} = \sum_{i=1}^{k} c_i \lambda_i \underset{\sim}{v_i}$$

$$\Rightarrow A^n \underset{\sim}{x} = \sum_{i=1}^{k} c_i \lambda_i^n \underset{\sim}{v_i} \cong c_1 \lambda_1^n \underset{\sim}{v_1}$$

In other words, the largest eigenvalue will dominate the product and eventually we can get approximate values of λ_1 and $\underset{\sim}{v_1}$.

Singular Value Decomposition

- In regression problem, we have

$$Y = X\beta + \varepsilon \Rightarrow U'Y = U'X\beta + U'\varepsilon$$

or equivalently, $(\theta = V'\beta \;\; \& \;\; X_1^* V = D)$

$$Y^* = \begin{pmatrix} X_1^* \\ 0 \end{pmatrix}\beta + \varepsilon^* = \begin{pmatrix} X_1^* \\ 0 \end{pmatrix} VV'\beta + \varepsilon^*$$

$$= \begin{pmatrix} D \\ 0 \end{pmatrix}\theta + \varepsilon^*$$

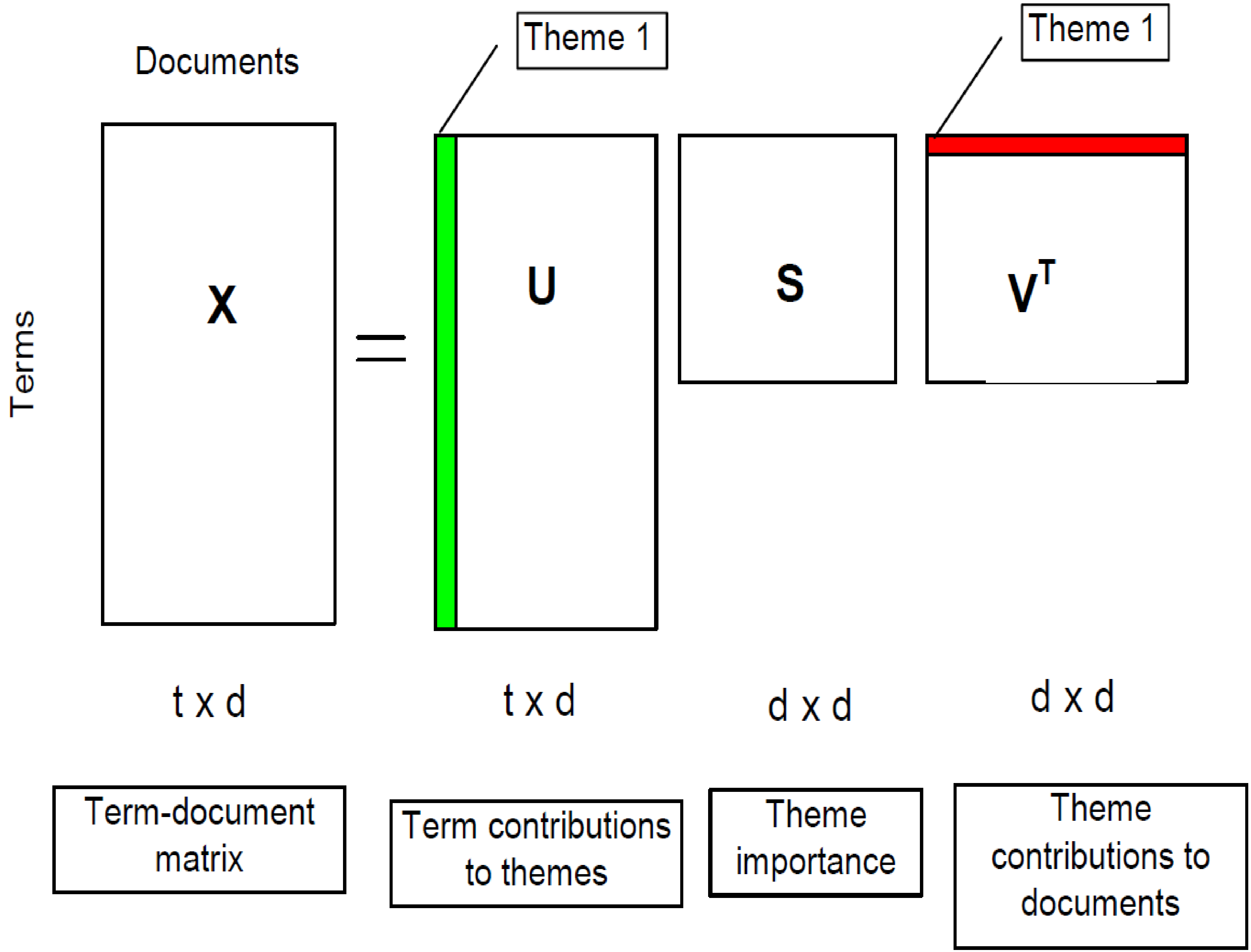
Note: SVD is often used for regression diagnostics, data reduction, and graphical clustering.

- The two orthogonal matrices *U* and *V* are associated with the following result:

<u>The Singular-Value Decomposition</u>

→Let *X* be an arbitrary $n \times p$ matrix with $n \geq p$. Then there exists orthogonal matrices U: $n \times n$ and V: $p \times p$ such that $U'XV = \tilde{D} = \begin{pmatrix} D \\ 0 \end{pmatrix}$,

$$\text{where } D = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_p \end{pmatrix} \text{ with } d_1 \geq d_2 \geq \cdots \geq d_p \geq 0.$$


Documents
Terms
X
=
U
S
V^T
Theme 1
Theme 1
t x d
t x d
d x d
d x d
Term-document matrix
Term contributions to themes
Theme importance
Theme contributions to documents

- Example of SVD:

→You can check if the command “svd” in R returns correct outputs.

```
A=matrix(c(1:12),ncol=3)
 aa=svd(A)
  t(aa$u)%*%A%*%aa$v     # Diagonal!
    aa$u%*%diag(c(aa$d))%*%t(aa$v)
  t(aa$u)%*%A     # Upper triangular!
```

Application of SVD (Lee-Carter Model)

- Lee and Carter (1992) proposed a model to forecast the mortality rates of U.S.：

$$\ln(m_{xt}) = \alpha_x + \beta_x \kappa_t + \varepsilon_{xt}$$

where

$\kappa_t \rightarrow$ change of mortality intensity

$\alpha_x \rightarrow$ average mortality of each age group

$\beta_x \rightarrow$ relative change rate of each age group

Singular Value Decomposition (SVD)

The parameters of Lee-Carter model can be estimated via

$$\text{Minimize} \quad \sum_{xt} (\ln(m_{xt}) - \alpha_x - \beta_x \kappa_t)^2$$

→This is done via decomposing the matrix

$$(\ln(m_{xt}) - \alpha_x) = UPV^T$$

→We can also use the approximation method, or the PCA to achieve similar estimation.

SVD Interpretation of Lee-Carter Model

Applying the SVD, i.e., $(\ln(m_{xt}) - \alpha_x) = UPV_t^T$

the matrix U represents the time component,

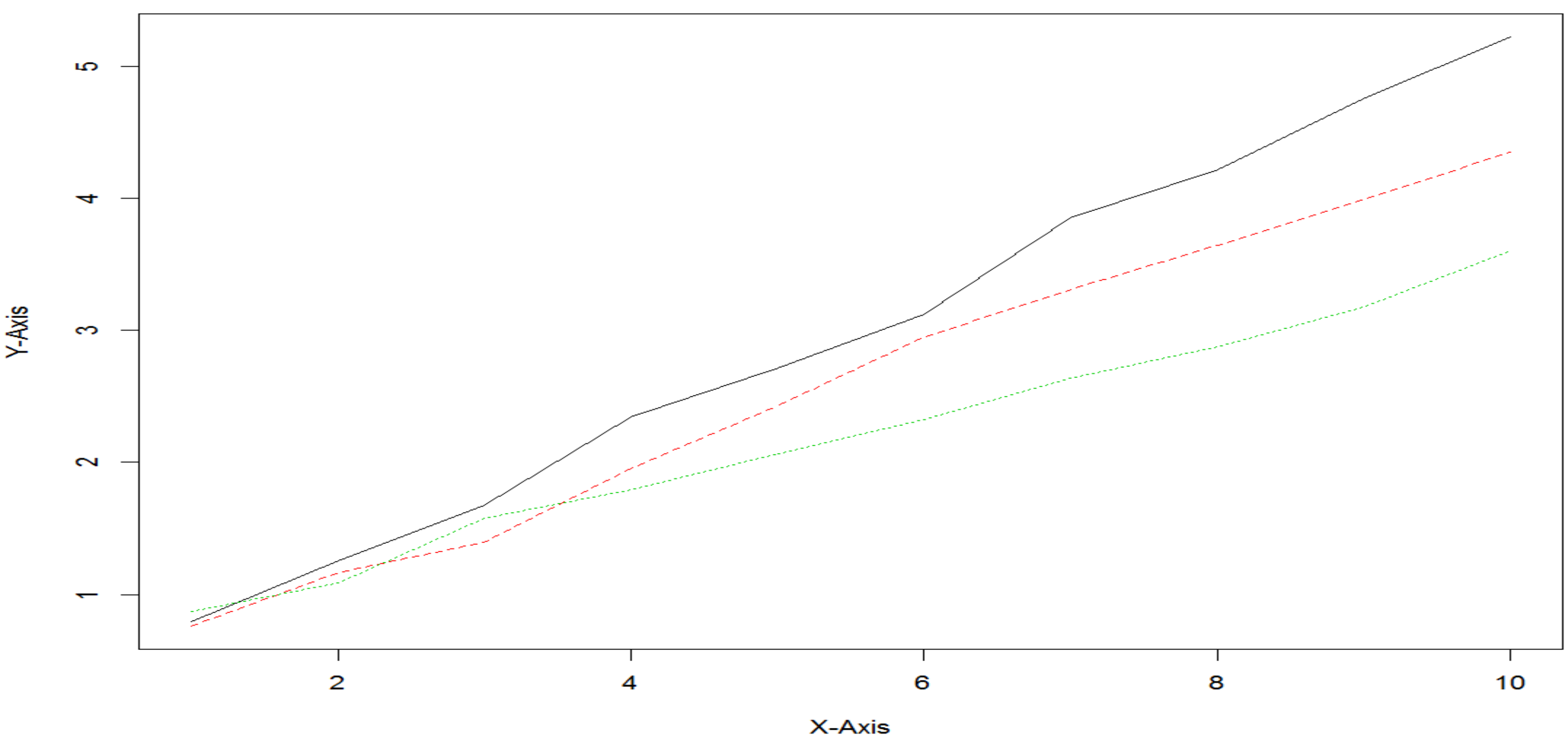
P is the singular values, and V is the age

component.

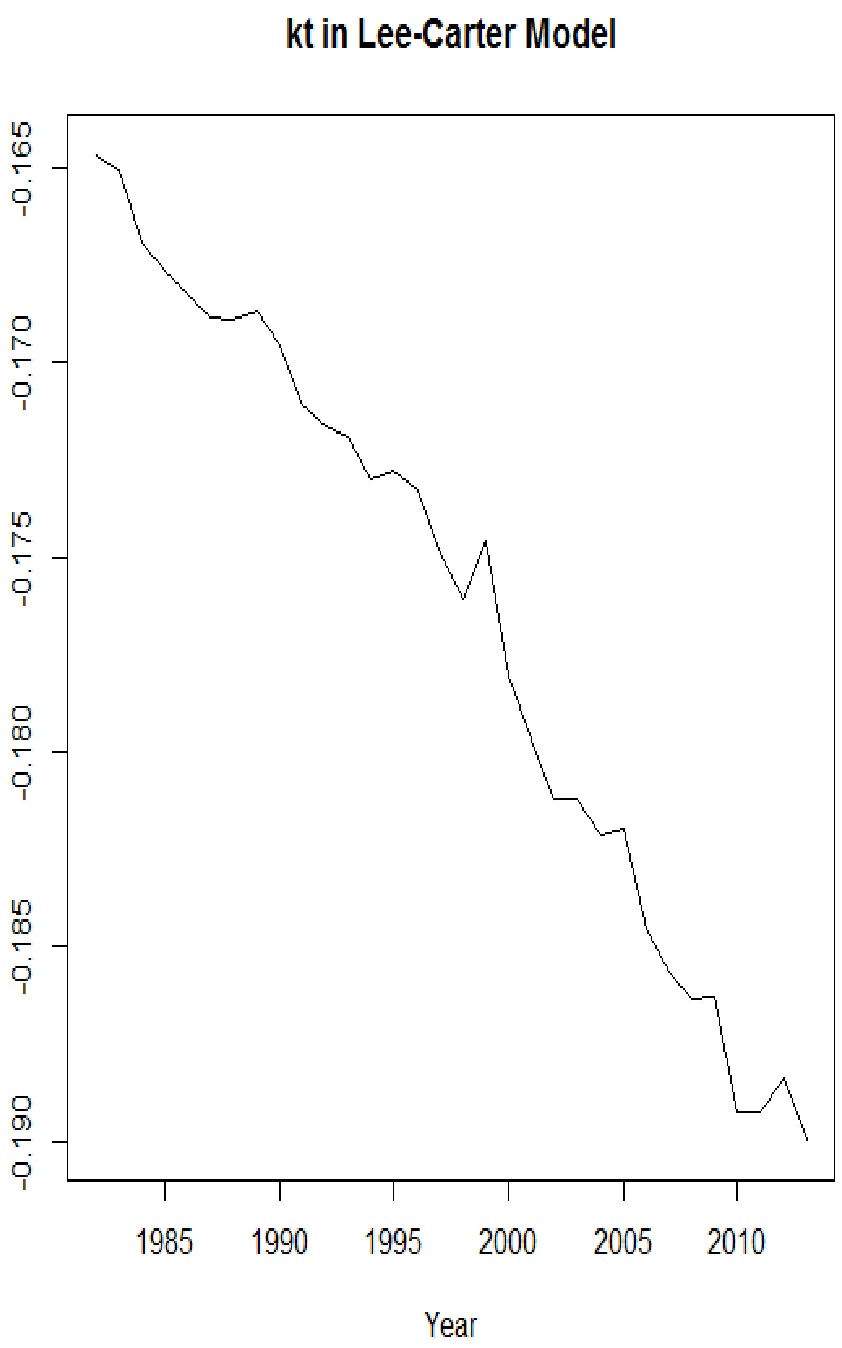
→ κ_t is derived from the first vector of the time-component matrix and the first singular value, and β_x is from the first vector of the age-component matrix. Other vectors correspond to the residuals.

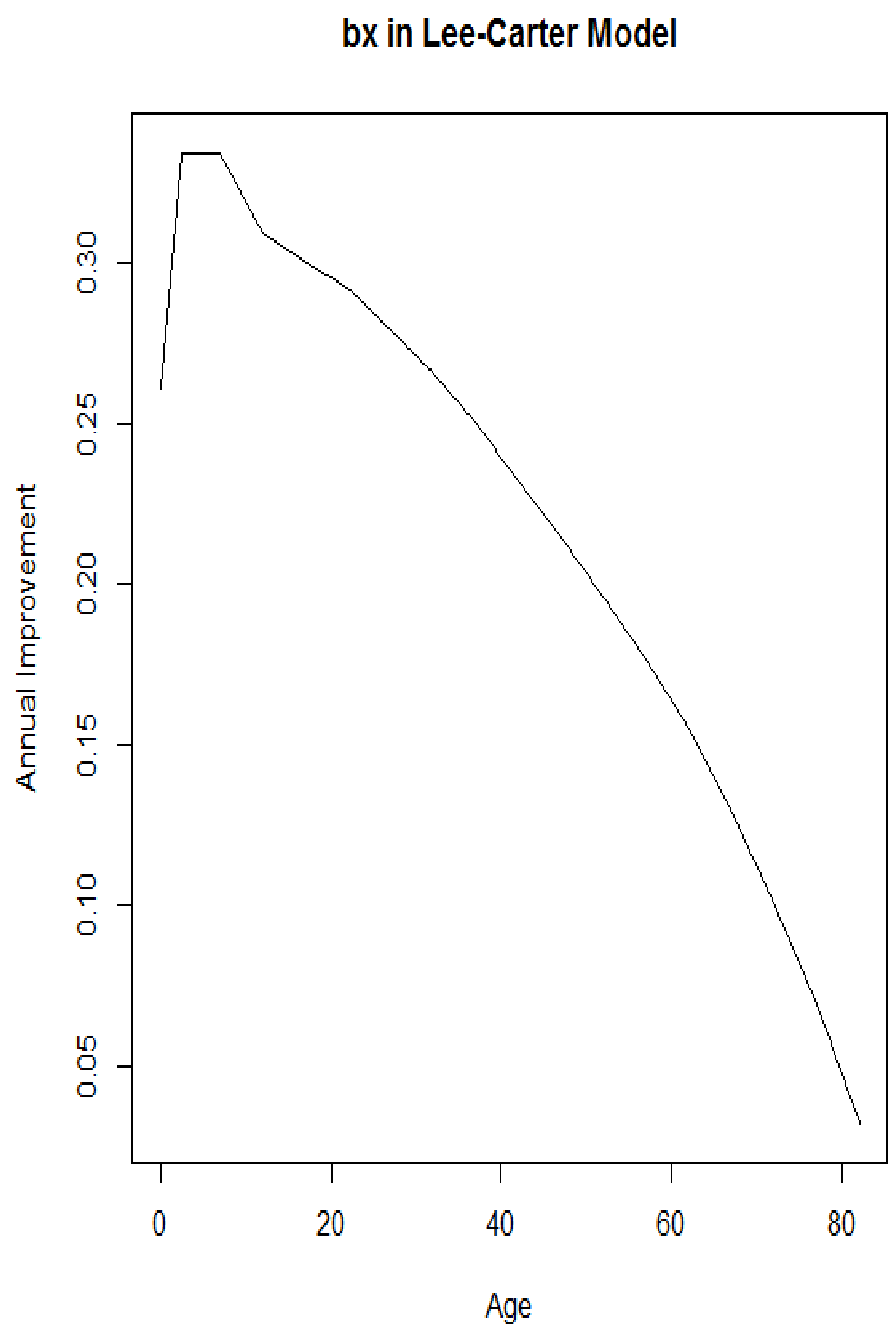
- Example of SVD (Lee-Carter model)

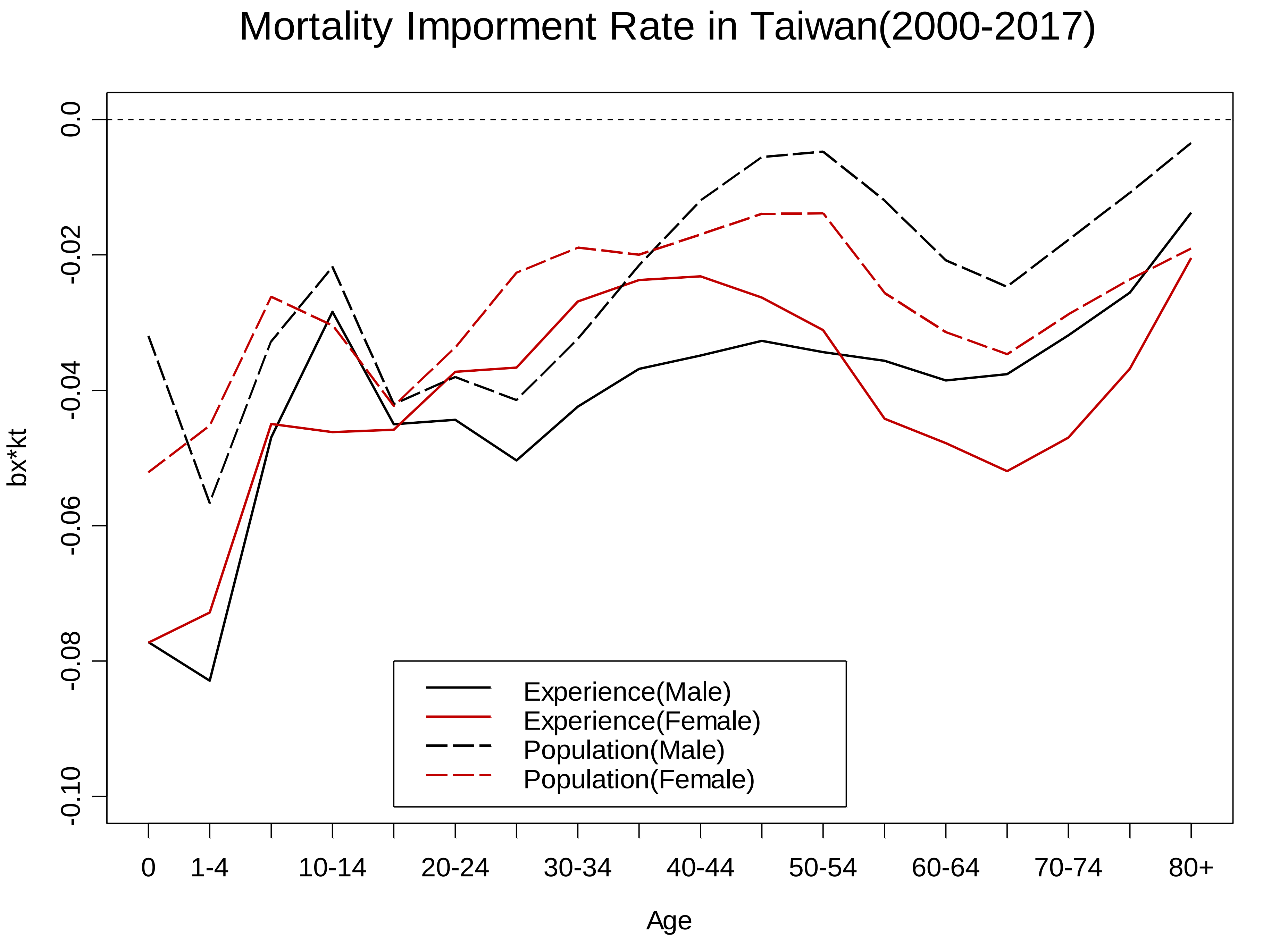
→Suppose there are three vectors and their relationships to X-axis are similar. We want to use only one vector to express the common pattern in these three vectors. (Data Reduction!)



```
a1=0.2+0.5*c(1:10)+0.1*rnorm(10)
 a2=0.4+0.4*c(1:10)+0.1*rnorm(10)
  a3=0.6+0.3*c(1:10)+0.1*rnorm(10)
   A=cbind(a1,a2,a3)
    x0=cbind(1:10,1:10,1:10)
   matplot(x0,A,type="l",xlab="X-Axis",ylab="Y-Axis")
aa=svd(A)
 A
  A1=aa$u%*%diag(c(aa$d[1],0,0))%*%t(aa$v)
   A1
    A2=A-A1
   mean(abs(A2/A))
```


kt in Lee-Carter Model
-0.165
-0.170
-0.175
-0.180
-0.185
-0.190
1985
1990
1995
2000
2005
2010
Year


bx in Lee-Carter Model
Annual Improvement
0.30
0.25
0.20
0.15
0.10
0.05
0
20
40
60
80
Age



Mortality Imporment Rate in Taiwan(2000-2017)
0.0
-0.02
-0.04
-0.06
-0.08
-0.10
bx*kt
0
1-4
10-14
20-24
30-34
40-44
50-54
60-64
70-74
80+
Age
Experience(Male)
Experience(Female)
Population(Male)
Population(Female)

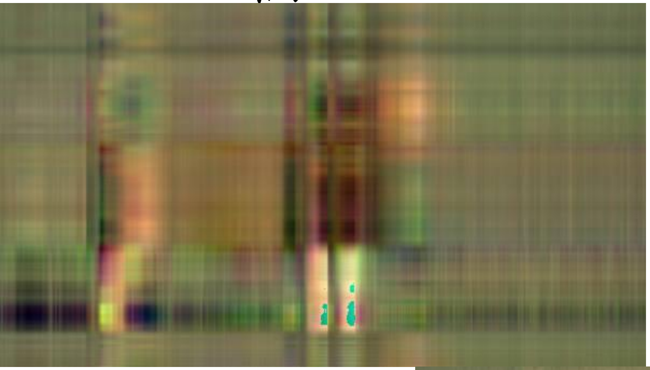
SVD與影像處理
(Image Processing)

- 左邊為原圖

→比較3~300個SVD的影像顯示差異。

https://rpubs.com/aaronsc32/image-compression-svd

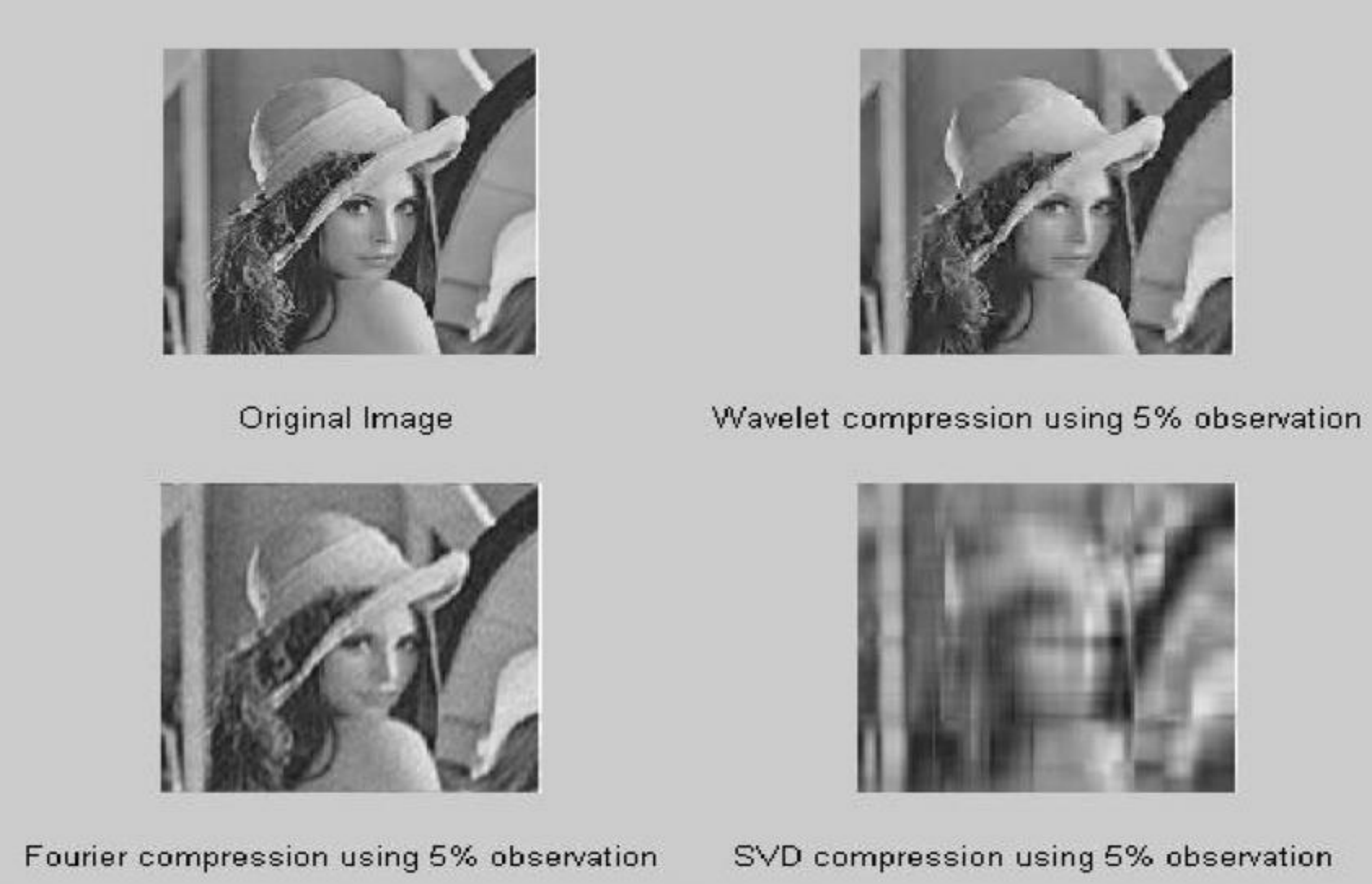
3個SVD



88個SVD

45個SVD

300個SVD



資料壓縮方法的比較(Comparing Dimension Reduction)

https://www.researchgate.net/profile/Md_Hossain402/publication/335134919/figure/fig4/AS:826535288250368@1574072777453/Compression-performance-among-wavelet-Fourier-and-SVD.png https://ars.els-cdn.com/content/image/1-s2.0-S0262885606002083-gr1.jpg

Time Series

- The time series analysis considered usually is ARIMA model, consisting of Auto-regressive (AR) and Moving Average (MA).

→AR(1) model

$$Z_t = \phi Z_{t-1} + e_t,\ e_t \sim N(0, \sigma^2),\ t = 1, 2, \ldots, n$$

Then the correlation coefficient of Z_i and Z_j is $\gamma_{|i-j|}$, where $\gamma_k = \phi^k$ and k=|i-j|.

→General form AR(p):

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + \cdots + \phi_p Z_{t-p} + e_t$$

$$\Rightarrow \gamma_k = \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2} + \cdots + \phi_p \gamma_{k-p}$$

Time Series (conti.)

Note: The coefficients of AR(p) can be solved by the ordinary regression, with some minor adjustments of variables.

→MA(1) model

$$Z_t = e_t - \theta e_{t-1},\ e_t \sim N(0,\sigma^2),\ t = 1,2,\ldots,n$$

Then the covariance of Z_i and Z_j is $\gamma_{|i-j|}$,

$$\gamma_k = \begin{cases} 1+\theta^2, & k = 0 \\ -\theta, & k = 1 \\ 0, & k \geq 2 \end{cases}$$

Time Series (conti.)

- The general form of ARMA(p,q) is.

$$\phi(B)Z_t = \theta(B)e_t$$

where $\phi(B)$ and $\theta(B)$ are polynomials of the backward operator B, i.e., $B(Z_t)=Z_{t-1}$.

→e.g., MA(1) model

Then the covariance of Z_i and Z_j is $\gamma_{|i-j|}$, or

$$\mathbf{A} = \begin{bmatrix} 1+\theta^2 & -\theta & 0 & 0 \\ -\theta & 1+\theta^2 & -\theta & 0 \\ 0 & -\theta & 1+\theta^2 & -\theta \\ 0 & 0 & -\theta & 1+\theta^2 \end{bmatrix}$$

Time Series Estimation

- The parameters in AR(p) model can be solved using the OLS, since

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + \cdots + \phi_p Z_{t-p} + e_t$$

$$\Rightarrow E(Z_t \mid Z_{t-1}, \ldots, Z_{t-p}) = \phi_1 Z_{t-1} + \cdots + \phi_p Z_{t-p}$$

Then the parameters are derived by minimizing (errors are normally distributed)

$$S(\mu, \tilde{\phi}) = \sum_{t=1}^{n} \left[z_t - \mu - \phi_1 (z_{t-1} - \mu) - \cdots - \phi_p (z_{t-p} - \mu) \right]^2$$

Estimation of AR(p) model

- The parameter estimation of AR model is easier (MA model requires iterations, since the “error” is not observable.)

→AR(2) model

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + e_t, \; e_t \sim N(0, \sigma^2).$$

The coefficients ϕ_1 and ϕ_2 can be solved by

(1) OLS (Z_{t-1} and Z_{t-2} as independent variables)

(2) Plugging the related numbers (Moment?)

(3) Exact likelihood functions (usually are very complicated)

Example: Estimation of AR(2) model

- Fit AR(2) model to the data "lynx" in R.

(1) OLS: $(\hat{\beta}_0, \hat{\phi}_1, \hat{\phi}_2) = (710.11, 1.1542, -0.6062)$

and by $\mu = \beta_0 / (1 - \phi_1 - \phi_2)$ to give $\hat{\mu}$.

(2) Moment:

$$corr(Z_i, Z_{i+1}) = \frac{\phi_1}{1-\phi_2}, \quad corr(Z_i, Z_{i+2}) = \phi_2 + \frac{\phi_1^2}{1-\phi_2}$$

→ $(\hat{\rho}_1, \hat{\rho}_2, \hat{\phi}_1, \hat{\phi}_2) = (0.7159, 0.2177, 1.1486, -0.6046)$

(3) Compare to the output from software

→Minitab: $(\hat{\mu}, \hat{\beta}_0, \hat{\phi}_1, \hat{\phi}_2) = (1545.4, 699.84, 1.1575, -0.6106)$

→R(arima): $(\hat{\mu}, \hat{\phi}_1, \hat{\phi}_2) = (1545.45, 1.1474, -0.5997)$

Example: AR(2) model (conti.)

- R also has a lot of options.

→Several choices in the function "AR"

ar(lynx, method="ols", 2)

(1) OLS: $(\hat{\phi}_1, \hat{\phi}_2) = (1.0320, -0.6288)$

(2) MLE: $(\hat{\phi}_1, \hat{\phi}_2) = (1.0555, -0.6298)$

(3) Default: $(\hat{\phi}_1, \hat{\phi}_2) = (1.0379, -0.6063)$

(4) Burg: $(\hat{\phi}_1, \hat{\phi}_2) = (1.0634, -0.6379)$

(5) YW: $(\hat{\phi}_1, \hat{\phi}_2) = (1.0379, -0.6063)$

Note: There are some handouts from the references regarding “Matrix Computation”

- “Numerical Linear Algebra” from Internet
- Chapters 3 to 6 in Monohan (2001)
- Chapter 3 in Thisted (1988)
- Chapter 2 in Gentle et al. (2004)
- Chapter 2 in Numerical Recipes in Fortran 77

Students are required to read and understand all these materials, in addition to the powerpoint notes.