第 13 章: 線性代數與矩陣運算

13: Matrix Algebra

在統計計算上, 許多時候會利用矩陣運算, 簡化統計模型的呈現方式, 並增加計算效率, R 如同 C 語言, 是一個程式語言, R 也有許多內建的矩陣運算函式, 寫作 R 的程式, 盡量以向量或矩陣型式寫作, 避免使用迴圈 (loop) 計算, 可以加快執行速度. R 程式語言的寫作分常類似 C 語言, 使用者可將 R 可以順利執行的程式, 輕鬆地改寫成 C 語言, 然後再從 R 呼叫 C 程式, 更加快 R 的計算執行速度.

13.1 矩陣函式: 總和與平均值 colSums(), colMeans(), rowSums(), rowMeans()

使用函式 `colSums()`, `colMeans()`, `rowSums()`, `rowMeans()` 等可以從從矩陣物件, 計算 欄 (行) 或 列 之 邊際總合 (marginal total) 與 邊際平均值 (marginal mean).

```
> colSums(x, na.rm = FALSE, dims = 1)
> rowSums(x, na.rm = FALSE, dims = 1)
> colMeans(x, na.rm = FALSE, dims = 1)
> rowMeans(x, na.rm = FALSE, dims = 1)
```

其中引數爲

- `x`: 矩陣或陣列, 非文字模式.
- `na.rm = FALSE`: 邏輯變數, 是否自動移除缺失值 (`NA`).
- `dims = 1`: 設定矩陣或陣列的維度視爲 `row` 或 `column` 進行計算.

```
> ## colSums(), colMeans(), rowSums(), rowMeans()
> ## matrix
> xvar = c(1:4)
> yvar = c(5:8)
> zvar = c(11:14)
> xyz.mat = cbind(xvar, yvar, zvar)
> class(xyz.mat)
[1] "matrix"
> xyz.mat
     xvar yvar zvar
[1,]    1    5   11
[2,]    2    6   12
[3,]    3    7   13
[4,]    4    8   14
> #
> colSums(xyz.mat, na.rm = FALSE, dims = 1)
xvar yvar zvar
  10   26   50
> rowSums(xyz.mat, na.rm = FALSE, dims = 1)
[1] 17 20 23 26
> colMeans(xyz.mat, na.rm = FALSE, dims = 1)
xvar yvar zvar
 2.5  6.5 12.5
> rowMeans(xyz.mat, na.rm = FALSE, dims = 1)
[1] 5.666667 6.666667 7.666667 8.666667
> round(rowMeans(xyz.mat, na.rm = FALSE, dims = 1), digits = 2)
[1] 5.67 6.67 7.67 8.67
> #
> ## colSums(), colMeans(), rowSums(), rowMeans()
> ## data frame
> xvar = c(1:4)
> yvar = c(5:8)
> zvar = c(11:14)
> xyz.df <- data.frame(xvar = c(1:4), yvar = c(5:8), zvar = c(11:14))
> xyz.df
  xvar yvar zvar
1    1    5   11
2    2    6   12
3    3    7   13
4    4    8   14
> #
> rowSums(xyz.df, zvar)
[1] 17 20 23 26
```

```
> rowSums(xyz.df, yvar)
[1] 17 20 23 26
> rowSums(xyz.df, xvar)
[1] 17 20 23 26
> #
> colSums(xyz.df, na.rm = FALSE, dims = 1)
xvar yvar zvar
  10   26   50
> rowSums(xyz.df, na.rm = FALSE, dims = 1)
[1] 17 20 23 26
> colMeans(xyz.df, na.rm = FALSE, dims = 1)
xvar yvar zvar
 2.5  6.5 12.5
> rowMeans(xyz.df, na.rm = FALSE, dims = 1)
[1] 5.666667 6.666667 7.666667 8.666667
> round(rowMeans(xyz.df, na.rm = FALSE, dims = 1), digits = 2)
[1] 5.67 6.67 7.67 8.67
```

另一個類似的 函式 `rowsum()` 則從矩陣或資料框架物件 x 中, R 依照 `group` 分組, 橫跨 欄位 來計算 列位 (row) 之總合.

```
> rowsum(x, group, reorder = TRUE, na.rm = FALSE, ...)
```

其中引數爲

- `x`: 向量, 矩陣 或 資料框架, 非文字模式.
- `group`: 因子變數 (factor), 對應 `x` 列位 (row) 之類別水準.
- `reorder = TRUE`: 輸出依照 `group` 排序, `sort(unique(group))`.
- `na.rm = FALSE`: 邏輯變數, 是否自動移除缺失值 (`NA`).

```
> ## rowsum()
> data(Puromycin)
> head(Puromycin)
  conc rate   state
1 0.02   76 treated
2 0.02   47 treated
3 0.06   97 treated
4 0.06  107 treated
5 0.11  123 treated
6 0.11  139 treated
> rowsum(Puromycin[, -3], group = Puromycin$state)
          conc rate
treated   4.14 1699
untreated 3.04 1218
```

13.2 加法與減法

矩陣 或 向量 的加法與減法, 可以一般算數指令 `+`, `-` 等進行 元素-元素 計算 (element-by-element calculation).

```
> ## matrix + and -
> (A <- matrix(c(1:12), nrow = 3, byrow = T))
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> (B <- matrix(c(1:12), nrow = 3))
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> A+B
     [,1] [,2] [,3] [,4]
[1,]    2    6   10   14
[2,]    7   11   15   19
[3,]   12   16   20   24
> A-B
     [,1] [,2] [,3] [,4]
[1,]    0   -2   -4   -6
[2,]    3    1   -1   -3
[3,]    6    4    2    0
```

13.3 轉置矩陣

計算 向量 或 矩陣 的 轉置 (transpose) 可以使用函式指令 `t()`. 函式 `t()` 也可以對 資料框架 進行轉置. 令 $m \times n$ 矩陣 A 的元素為 $a_{i,j}$, A 的 轉置矩陣 為 $\mathsf{B} = t(\mathsf{A})$, 維度是 $n \times m$, 且 B 的元素為 $b_{i,j}$, 定義轉置矩陣操作為

$$\mathsf{B} = \mathsf{A}^T = t(\mathsf{A}), b_{i,j} = a_{j,i}. \tag{13.3.1}$$

```
> t(x)
```

其中引數為:

- x: 向量, 矩陣 或 資料框架.
- 若 x 爲向量, 則先視爲 行向量 (column vector), 然後轉置成 列向量 (row vector).

```
> ## transpose
> ## vector transpose
> (x = c(1, 2, 3))
[1] 1 2 3
> t(x)
     [,1] [,2] [,3]
[1,]    1    2    3
> #
> ## matrix transpose
> (A <- matrix(c(1:12), nrow = 3, byrow = T))
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> t(A)
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> #
> ## data frame transpose
> (xyz.df <- data.frame(xvar = c(1:4), yvar = c(5:8), zvar = c(11:14)))
  xvar yvar zvar
1    1    5   11
2    2    6   12
3    3    7   13
4    4    8   14
> t(xyz.df)
     [,1] [,2] [,3] [,4]
xvar    1    2    3    4
yvar    5    6    7    8
zvar   11   12   13   14
```

使用 函式 aperm() 可以對一個 陣列 (array) 進行所謂的廣義轉置. 函式指令爲

```
> aperm(a, perm = NULL, resize = TRUE, keep.class = TRUE, ...)
```

其中的主要引數分別爲:

- a.arr: 要進行廣義轉置的 陣列 (array).

- perm: 一個向量, 可以是 $\{1, \ldots, k\}$ 的一種 排列組合 (permutation) 情形, 其中 k 是 陣列 a.arr 的長度之下標數目.

這個函式 aperm(a.arr, perm) 將產生一個和 原始陣列 a.arr 大小一致的陣列, 不過 原始陣列 a.arr 的第 perm[j] 維度, 將會變成 轉置陣列 的第 j 個維度, 這種操作實際上是對 陣列 或 矩陣 的一種廣義轉置. 實際上, 如果 **A** 是一個 矩陣, 那麼執行 Bmat <- aperm(A, c(2, 1)), 則 **B** 是矩陣 **A** 的轉置矩陣, 若矩陣 **B** 僅僅是 矩陣 **A** 的一個 (一般) 轉置矩陣, 這種情況下, 則簡單的函式 t() 就可以使用.

```
> ## aperm()
> ## aperm + matrix
> (A <- matrix(c(1:12), nrow = 3, byrow = T))
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> (B.aperm <- aperm(A, c(2, 1))) # = t(A) = matrix transpose
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
>
> ## aperm array
> b.array <- array(c(1:24), dim = c(4, 3, 2),
+                  dimnames = list(x = letters[1:4],
+                                  y = LETTERS[1:3],
+                                  z = c("i", "ii")))
> b.array
, , z = i

   y
x   A B  C
  a 1 5  9
  b 2 6 10
  c 3 7 11
  d 4 8 12

, , z = ii

   y
x    A  B  C
  a 13 17 21
  b 14 18 22
  c 15 19 23
```

```
  d 16 20 24

>
> ## interchange the first two subscripts on a 3-way array
> ## aperm() + resize = TRUE
> ## change dim 1st and 2nd
> aperm(b.array, perm = c(2, 1, 3), resize = TRUE)
, , z = i

   x
y   a  b  c  d
  A 1  2  3  4
  B 5  6  7  8
  C 9 10 11 12

, , z = ii

   x
y    a  b  c  d
  A 13 14 15 16
  B 17 18 19 20
  C 21 22 23 24

>
> ## aperm() + resize = FALSE
> aperm(b.array, perm = c(2, 1, 3), resize = FALSE)
, , 1

     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    5   10    4
[3,]    9    3    8
[4,]    2    7   12

, , 2

     [,1] [,2] [,3]
[1,]   13   18   23
[2,]   17   22   16
[3,]   21   15   20
[4,]   14   19   24

> #
> aperm(b.array, perm = c(3, 1, 2))
, , y = A

    x
z     a  b  c  d
  i   1  2  3  4
  ii 13 14 15 16
```

```

, , y = B

   x
z     a  b  c  d
  i   5  6  7  8
  ii 17 18 19 20

, , y = C

   x
z     a  b  c  d
  i   9 10 11 12
  ii 21 22 23 24
```

13.4 矩陣乘法

矩陣 (或向量) 的乘法, 有許多不同的定義, 例如, 對單一純量的乘法, 元素-與-元素 乘積, 矩陣內積, 矩陣外積, 矩陣 Konecker 乘積等. R 對不同的矩陣的乘法, 有不同的函式指令.

13.4.1 矩陣乘法: 純量

矩陣 (或向量) 的乘法, 若是將一個 矩陣 與 單一數值, 或 '純量 (scalar) 相乘, 可以使用算數乘法 * 的指令.

$$\mathbf{B}_{m\times n}[i,j] = s \times \mathbf{A}_{m\times n}[i,j] \tag{13.4.1}$$

$$\mathbf{B} = s \times \mathbf{A}_{m\times n} = s \star \mathbf{A} = \begin{pmatrix} sa_{1,1} & sa_{1,2} & \dots \\ sa_{2,1} & \ddots & \\ \vdots & & \end{pmatrix} \tag{13.4.2}$$

```
> ## product * : scalar
> (A <- matrix(c(1:12), nrow = 3, byrow = T))  # A_(3x4)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> 2*A # same as A*2
     [,1] [,2] [,3] [,4]
[1,]    2    4    6    8
[2,]   10   12   14   16
[3,]   18   20   22   24
> A*2
     [,1] [,2] [,3] [,4]
[1,]    2    4    6    8
[2,]   10   12   14   16
[3,]   18   20   22   24
```

矩陣 (或向量) 的除法, 若是將一個 矩陣 與 單一數值, 或 純量 (scalar) 相乘, 可以使用算數除法 / 的指令.

```
> ## divide / : scalar
> (A <- matrix(c(1:12), nrow = 3, byrow = T))  # A_(3x4)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> A/2
     [,1] [,2] [,3] [,4]
[1,]  0.5    1  1.5    2
[2,]  2.5    3  3.5    4
[3,]  4.5    5  5.5    6
> 2/A
          [,1]      [,2]      [,3]      [,4]
[1,] 2.0000000 1.0000000 0.6666667 0.5000000
[2,] 0.4000000 0.3333333 0.2857143 0.2500000
[3,] 0.2222222 0.2000000 0.1818182 0.1666667
```

13.4.2 矩陣乘法: 元素-與-元素 乘積

若 2 個矩陣分別爲 $\mathbf{A}_{m\times n}$ 與 矩陣 $\mathbf{B}_{m\times n}$, 且這 2 個矩陣的 列位數 與 行位數 (欄位數) 都相等, 計算 元素–與–元素 乘積 (element-by-element product), 矩陣 $\mathbf{A}_{m\times n}$ 內每一個元素, 與 矩陣 $\mathbf{B}_{m\times n}$ 內每一個元素, 個別相乘, 可以使用算數乘法 * 的指令.

$$C_{m\times n}[i,j] = A_{m\times n}[i,j] \times B_{m\times n}[i,j] \tag{13.4.3}$$

$$\Rightarrow \quad c_{i,j} = a_{i,j} * b_{i,j} \tag{13.4.4}$$

$$\begin{aligned} \mathbf{C}_{m\times n} &= \mathbf{A}_{m\times n} \star B_{m\times n} \\ &= \begin{pmatrix} c_{1,1} = a_{1,1} \star b_{1,1} & c_{1,2} = a_{1,2} \star b_{1,2} & \cdots & \cdots \\ c_{2,1} = a_{2,1} \star b_{2,1} & \vdots & \ddots & \vdots \\ \vdots & \vdots & c_{i,j} = a_{i,j} \star b_{i,j} & \vdots \\ \cdots & \cdots & \cdots & c_{m,n} = a_{m,n} \star b_{m,n} \end{pmatrix} \end{aligned} \tag{13.4.5}$$

```
> ## product * : element by element
> (A <- matrix(c(1:12), nrow = 3, byrow = T)) # A_(3x4)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> (B <- matrix(c(1:12), nrow = 3)) # B_(3x4)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> A*B
     [,1] [,2] [,3] [,4]
[1,]    1    8   21   40
[2,]   10   30   56   88
[3,]   27   60   99  144
> B*A
     [,1] [,2] [,3] [,4]
[1,]    1    8   21   40
```

```
[2,]   10   30   56   88
[3,]   27   60   99  144
```

若 2 個矩陣分別爲 $\mathsf{A}_{m\times n}$ 與 矩陣 $\mathsf{B}_{m\times n}$, 且這 2 個矩陣的 列位數 與 行位數 (欄位數) 都相等, 計算 元素-與-元素 除法, 矩陣 $\mathsf{A}_{m\times n}$ 內每一個元素, 與矩陣 $\mathsf{B}_{m\times n}$ 內每一個元素, 個別相除的除法 $\frac{a_{i,j}}{b_{i.j}}$, 可以使用算數除法 / 的指令執行,

```
> ## divide / : elelemt-by-element division
> (A <- matrix(c(1:12), nrow = 3, byrow = T)) # A_(3x4)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> (B <- matrix(c(1:12), nrow = 3)) # B_(3x4)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> A/B
     [,1]     [,2]      [,3]      [,4]
[1,]  1.0 0.500000 0.4285714 0.4000000
[2,]  2.5 1.200000 0.8750000 0.7272727
[3,]  3.0 1.666667 1.2222222 1.0000000
> B/A
          [,1]      [,2]      [,3]  [,4]
[1,] 1.0000000 2.0000000 2.3333333 2.500
[2,] 0.4000000 0.8333333 1.1428571 1.375
[3,] 0.3333333 0.6000000 0.8181818 1.000
```

13.4.3 矩陣乘法: 內積

習慣上, 矩陣乘法的基本定義爲 內積 (matrix product, inner product, dot product), 矩陣 $\mathsf{A}_{m\times n}$ 與 矩陣 $\mathsf{B}_{n\times p}$ 的 內積 (inner product), 以矩陣內積乘法 %*% 的指令進行,

$$\mathsf{C}_{m\times p}[i,j] = \sum_{k=1}^{n} \mathsf{A}_{m\times n}[i,k] \times \mathsf{B}_{n\times p}[k,j] \tag{13.4.6}$$

$$\Rightarrow \quad c_{i,j} = \sum_{k=1}^{n} a_{i,k} \star b_{k,j} \tag{13.4.7}$$

$$\mathsf{C}_{m\times p} = \mathsf{A}_{m\times n} \ \%{*}\% \ \mathsf{B}_{n\times p} \tag{13.4.8}$$

$$= \begin{pmatrix} c_{1,1} = \sum_{k=1}^{n} a_{1,k} \star b_{k,1} & c_{1,2} = \sum_{k=1}^{n} a_{1,k} \star b_{k,2} & \cdots & \cdots \\ c_{2,1} = \sum_{k=1}^{n} a_{2,k} \star b_{k,1} & \ddots & \cdots & \cdots \\ \vdots & & c_{i,j} = \sum_{k=1}^{n} a_{i,k} \star b_{k,j} & \\ \vdots & & & \\ & & & c_{m,p} = \sum_{k=1}^{n} a_{m,k} \star b_{k,p} \end{pmatrix} \tag{13.4.9}$$

```r
> ## product %*% : inner product
> (A <- matrix(c(1:12), nrow = 3, byrow = T)) # A_(3x4)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> (B <- matrix(c(1:8),nrow = 4)) # B_(4x2)
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> A%*%B # C_{3x2}
     [,1] [,2]
[1,]   30   70
[2,]   70  174
[3,]  110  278
```

13.4.4 矩陣乘法: 外積

向量 $\underline{\mathbf{a}}$ 與 向量 $\underline{\mathbf{b}}$ 的 外積 (outer product), 定義爲

$$\underline{\mathbf{c}} = \underline{\mathbf{a}} \ \%o\% \ \underline{\mathbf{b}} = \underline{\mathbf{a}} \star \underline{\mathbf{b}}^T. \tag{13.4.10}$$

一般向量的 外積 爲 Kronecker 乘積 (Kronecker product) 的一個特例. 同樣的, R 定義矩陣 $\mathsf{A}_{m\times n}$ 與 矩陣 $\mathsf{B}_{p\times q}$ 廣義的 外積 (outer product), 爲一種多維度陣列的 Tensor 乘法 (Tensor multiplication) 的一種情形. R 進行矩陣外積乘法的指令函式爲 %o%, 並回傳一個 列表物件 (list). R 將矩陣外積乘法回傳 列表 (list), 不容易理解.

$$\mathsf{C} = \mathsf{A}_{m\times n} \ \%o\% \ \mathsf{B}_{p\times q} \tag{13.4.11}$$

$$= \begin{pmatrix} \mathsf{A} \star b_{1,1} & \ldots & \mathsf{A} \star b_{1,q} \\ \vdots & \ddots & \vdots \\ \mathsf{A} \star b_{p,1} & \ldots & \mathsf{A} \star b_{p,q} \end{pmatrix} \tag{13.4.12}$$

$$\mathsf{D} = \mathsf{B}_{p\times q} \ \%o\% \ \mathsf{A}_{m\times n} \tag{13.4.13}$$

$$= \begin{pmatrix} \mathsf{B} \star a_{1,1} & \ldots & \mathsf{B} \star b_{1,n} \\ \vdots & \ddots & \vdots \\ \mathsf{B} \star a_{m,1} & \ldots & \mathsf{B} \star a_{m,n} \end{pmatrix} \tag{13.4.14}$$

習慣上使用 Kronecker 乘積 指令較爲方便.

```
> ## product %o% : outer product
> (A <- matrix(c(1:6), nrow = 3)) # A_(3x2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> (B <- matrix(c(1:4), nrow = 2)) # B_(2x2)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> A%o%B # return a list
```

```
, , 1, 1

     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

, , 2, 1

     [,1] [,2]
[1,]    2    8
[2,]    4   10
[3,]    6   12

, , 1, 2

     [,1] [,2]
[1,]    3   12
[2,]    6   15
[3,]    9   18

, , 2, 2

     [,1] [,2]
[1,]    4   16
[2,]    8   20
[3,]   12   24

> B%o%A
, , 1, 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2, 1

     [,1] [,2]
[1,]    2    6
[2,]    4    8

, , 3, 1

     [,1] [,2]
[1,]    3    9
[2,]    6   12

, , 1, 2

     [,1] [,2]
```

```
[1,]    4   12
[2,]    8   16

, , 2, 2

     [,1] [,2]
[1,]    5   15
[2,]   10   20

, , 3, 2

     [,1] [,2]
[1,]    6   18
[2,]   12   24

> ## product %o% : outer product
> (A <- matrix(c(1:6), nrow = 3)) # A_(3x2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> (B <- matrix(c(1:4), nrow = 2)) # B_(2x2)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> A%x%B # return a list
     [,1] [,2] [,3] [,4]
[1,]    1    3    4   12
[2,]    2    4    8   16
[3,]    2    6    5   15
[4,]    4    8   10   20
[5,]    3    9    6   18
[6,]    6   12   12   24
> B%x%A
     [,1] [,2] [,3] [,4]
[1,]    1    4    3   12
[2,]    2    5    6   15
[3,]    3    6    9   18
[4,]    2    8    4   16
[5,]    4   10    8   20
[6,]    6   12   12   24
```

13.4.5 矩陣乘法: Kronecker 乘積

在縱貫資料分析中最常用的矩陣乘法爲 Kronecker 乘積 (Kronecker product), 矩陣 $\mathbf{A}_{m\times n}$ 與 矩陣 $\mathbf{B}_{p\times q}$ 的 Kronecker product 可以矩陣外積指令 `%x%` 或以矩陣函式 `kronecker()` 執行, 並傳回矩陣. 函式的指令爲

```
> kronecker(X, Y, FUN = "*", make.dimnames = FALSE, ...)
X %x% Y
```

其中 `X` 與 `Y` 分別爲 矩陣 $\mathbf{A}_{m\times n}$ 與 矩陣 $\mathbf{B}_{p\times q}$.

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \tag{13.4.15}$$

$$\mathbf{C}_{m\times p, n\times q} = A_{m\times n} \text{ %x% } B_{p\times q}$$

$$= \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \ldots & a_{1,n}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1}\mathbf{B} & \ldots & \ldots & a_{m,n}\mathbf{B} \end{pmatrix} \tag{13.4.16}$$

$$= \begin{pmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & \ldots & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} & \ldots \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & & a_{1,2}b_{1,1} & a_{1,2}b_{2,2} & \\ \vdots & & \ddots & & & \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & & & & \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & & & & \\ \vdots & & & & & \end{pmatrix} \tag{13.4.17}$$

```
> ## product: kronecker product (outer product)
> (A <- matrix(c(1:6), nrow = 3)) # A_(3x2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

```
> (B <- matrix(c(1:4), nrow = 2)) # B_(2x2)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> kronecker(A, B) # return matrix
     [,1] [,2] [,3] [,4]
[1,]    1    3    4   12
[2,]    2    4    8   16
[3,]    2    6    5   15
[4,]    4    8   10   20
[5,]    3    9    6   18
[6,]    6   12   12   24
> kronecker(B, A)
     [,1] [,2] [,3] [,4]
[1,]    1    4    3   12
[2,]    2    5    6   15
[3,]    3    6    9   18
[4,]    2    8    4   16
[5,]    4   10    8   20
[6,]    6   12   12   24
```

13.4.6 矩陣乘法: 交叉乘積

在線性模型分析中最常用的矩陣乘法爲 交叉乘積 或 矢積, (cross-product), 定義 矩陣 $\mathbf{A}_{p\times n}$ 與 矩陣 $\mathbf{B}_{p\times q}$ 的 交叉乘積 爲

$$\mathbf{C}_{n\times q} = \text{crossprod}\left(\mathbf{A}_{p\times n}, \mathbf{B}_{p\times q}\right) = \mathbf{A}^T \bullet \mathbf{B}. \tag{13.4.18}$$

R 使用矩陣乘法函式 `crossprod()` 可以執行 交叉乘積 的矩陣運算, 函式指令爲

```
> crossprod(x, y = NULL) # = t(x) %*% y
> tcrossprod(x, y = NULL) # = x %*% t(y)
```

其中 `x` 與 `y` 分別爲 矩陣 $\mathbf{A}_{p\times n}$ 與 矩陣 $\mathbf{B}_{p\times q}$, 如果 `crossprod()` 第二個引數省略了, 它將和第一個引數一樣. 使用矩陣乘法函式 `crossprod()` 在運算上更有效率.

```
> ## product: crossproduct()
> (A <- matrix(c(1:6), nrow = 3)) # A_(3x2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> (B <- matrix(c(1:6), nrow = 3)) # B_(3x2)
```

```
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> crossprod(A, B) # = t(A) %*% B
     [,1] [,2]
[1,]   14   32
[2,]   32   77
> t(A) %*% B # = crossprod(A, B)
     [,1] [,2]
[1,]   14   32
[2,]   32   77
```

13.4.7 矩陣乘法: 元素-與-元素 乘冪

定義 方塊矩陣 或 方陣 (square matrix) $A_{m\times m}$ 的 元素-與-元素 之 k 次 乘冪 或 k 次方 (power), $\mathbf{A}^k$,

$$\mathbf{A}^k = = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & \dots & \dots & a_{m,m} \end{pmatrix}^k = \begin{pmatrix} a^k_{1,1} & a^k_{1,2} & \dots & a^k_{1,m} \\ a^k_{2,1} & a^k_{2,2} & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a^k_{m,1} & \dots & \dots & a^k_{m,m} \end{pmatrix} \tag{13.4.19}$$

R 可以直接使用算數乘冪指令 ^ 計算. 注意: 只有方陣 (square matrix) 才能求出其 元素-與-元素 之乘冪計算, 且 方陣 元素-與-元素 之乘冪 與 方陣 的矩陣內積計算不同.

```
> ## product ^ : power
> (A <- matrix(c(1:9), nrow = 3)) # square matrix A_(3x3)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> A^3
     [,1] [,2] [,3]
[1,]    1   64  343
[2,]    8  125  512
[3,]   27  216  729
> A*A*A
     [,1] [,2] [,3]
```

```
[1,]    1   64  343
[2,]    8  125  512
[3,]   27  216  729

## inner product: different
> A%*%A%*%A
     [,1] [,2] [,3]
[1,]  468 1062 1656
[2,]  576 1305 2034
[3,]  684 1548 2412
```

13.5 矩陣行列式值

假設 **A** 是一個 方塊矩陣 或 方陣 (square matrix), 則定義其 行列式值 (determinant) 爲

$$\det(\mathbf{A}) = |\mathbf{A}| \tag{13.5.1}$$

使用矩陣函式 det() 與 determinant() 可以計算矩陣行列式值, 並回傳行列式值, 函式指令爲

```
> det(x, ...)
> determinant(x, logarithm = TRUE, ...)
```

其中的主要引數分別爲:

- x: 方陣 **A**.
- logarithm = TRUE 表示 R 自動內建計算矩陣行列式值 的 對數值, 在使用矩陣函式 determinant() 時, 若要取得 矩陣行列式值並取絕對值, 則設定引數 logarithm = FALSE.
- 使用矩陣函式 determinant() 可以計算矩陣行列式值, 且回傳一個列表物件 (list). 包含 2 個成分:
  - modulus: $= \log(|\det(\mathbf{A})|)$, 計算矩陣行列式值, 取絕對值之後的行列式值 或 取絕對值之後的的行列式值再取對數值.

- sign: = sign(det(**A**)), 矩陣行列式值的原始正負號.

```
> ## det() and determinant()
> (A <- matrix(c(1,2,3, 2,3,4, 3,4,1), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    1
> det(A)
[1] 4
> determinant(A, logarithm = FALSE)
$modulus
[1] 4
attr(,"logarithm")
[1] FALSE

$sign
[1] 1

attr(,"class")
[1] "det"
> (B <- matrix(c(3, 1, 4, -2), nrow = 2, byrow = T))
     [,1] [,2]
[1,]    3    1
[2,]    4   -2
> det(B)
[1] -10
> determinant(B, logarithm = FALSE)
$modulus
[1] 10
attr(,"logarithm")
[1] FALSE

$sign
[1] -1

attr(,"class")
[1] "det"
> determinant(B, logarithm = TRUE)
$modulus
[1] 2.302585
attr(,"logarithm")
[1] TRUE

$sign
[1] -1

attr(,"class")
[1] "det"
```

13.6 主對角矩陣

假設 **A** 是一個 '方塊矩陣 或 方陣 (square matrix), 則矩陣 $\mathbf{A}_{k\times k}$ 之 主對角元素向量, $\underline{\mathbf{x}}_{k\times 1}$, (diagonal elements) 爲

$$\underline{\mathbf{x}} = \mathbf{Diag}(\mathbf{A}) = (a_{1,1}, a_{2,2}, \ldots, a_{k,k})^T. \tag{13.6.1}$$

R 的矩陣函式 diag() 可以取得方陣的主對角元素向量. 函式指令爲

```
> diag(x = 1, nrow, ncol)
> diag(x) <- value
```

其中其中的主要引數差別爲:

- 若 x 是 方陣 **A**, d <- diag(x) 回傳 方陣 **A** 的主對角元素向量 $\underline{\mathbf{d}}$. 若矩陣 **A** 不是 方陣, 則 R 仍會回傳矩陣 **A** 的主對角元素向量, 但計算上必須小心.
- 若 x 缺失, 但 nrow = k 存在, 則 I <- diag(x) 回傳 $k \times k$ 的單位矩陣 (identity matrix), $\mathbf{I}_{k\times k}$.
- 若 x 是 純量 (scalar), x = k, 且是唯一的引數, 則 I <- diag(x) 回傳 $k \times k$ 的 單位方陣 (square identity matrix), $\mathbf{I}_{k\times k}$.
- 若 x 是 向量 長度 $k \geq 2$, D <- diag(x) 回傳 主對角元素爲 向量 x 的 主對角方陣 $\mathbf{D}_{k\times k}$.

```
> ## diag()
> (A <- matrix(c(1:9), nrow = 3)) # square matrix A_(3x3)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> diag(x = A) # retrun diagonal vector
[1] 1 5 9
> diag(nrow = 3)  # retrun identity matrix
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

```
> diag(x = 3) # retrun identity matrix
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> diag(x = c(1, 2, 3)) # retrun diagonal matrix
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
>
> # not a square matrix
> (B <- matrix(c(1:6), nrow = 3)) # B_(3x2)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> diag(B)
[1] 1 5
> # not a square matrix
> (B <- matrix(c(1:6), nrow = 2)) # B_(2x3)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> diag(B)
[1] 1 4
```

13.7 反矩陣 與 線性方程式

在統計計算或許多數學的線性方程式, 必須求線性方程式的解, 例如,

$$\underline{\mathbf{b}} = \mathbf{A}\underline{\mathbf{x}}, \qquad (13.7.1)$$

上述線性方程式已知 $\mathbf{A}$ 與 $\underline{\mathbf{b}}$, 欲求 $\underline{\mathbf{x}}$ 的解, 則必須能夠計算 矩陣 $\mathbf{A}$ 的 反矩陣 (matrix inverse). 在 R 可以使用函式 `solve()` 求得矩陣之 反矩陣 (matrix inverse) 並求得線性方程式之解. 函式指令爲

```
> solve(a, b, tol, LINPACK = FALSE, ...)
```

其中的主要引數分別爲:

- `a`: 方塊矩陣 $\mathbf{A}$, `solve(a)` 回傳 矩陣 $\mathbf{A}$ 的 反矩陣.

- `b`: 向量 $\underline{\mathbf{b}}$, `solve(a, b)` 可以求得解線性方程式 $\underline{\mathbf{b}} = \mathbf{A}\underline{\mathbf{x}}$ 之解 $\underline{\mathbf{x}}$.
- 若要計算矩陣二次型式 $\underline{\mathbf{x}}^T\mathbf{A}^{-1}\underline{\mathbf{x}}$ 的結果, 在 R 中, 以 `x %*% solve(A, x)` 較有效率.

$$\underline{\mathbf{b}} = \mathbf{A}\underline{\mathbf{x}} \tag{13.7.2}$$

$$\underline{\mathbf{x}} = \mathbf{A}^{-1}\underline{\mathbf{b}} \tag{13.7.3}$$

$$\underline{\mathbf{x}} = \text{solve}(\mathbf{A}) \ \%*\% \ \underline{\mathbf{b}} \tag{13.7.4}$$

$$\underline{\mathbf{x}} = \text{solve}(\mathbf{A}, \underline{\mathbf{b}}) \tag{13.7.5}$$

考慮以下線性方程式

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 3 \\ 2x_1 + 3x_2 + 4x_3 &= 0 \\ 3x_1 + 4x_2 + \ \ x_3 &= 1 \end{aligned} \tag{13.7.6}$$

```
> ## solve(): inverse, linear equation
> ## solve(): inverse
> (A <- matrix(c(1,2,3, 2,3,4, 3,4,1), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    1
> solve(A)
      [,1] [,2]  [,3]
[1,] -3.25  2.5 -0.25
[2,]  2.50 -2.0  0.50
[3,] -0.25  0.5 -0.25
> round(solve(A)%*%A, 2) # reverse
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
>
> ## solve(): linear equation
> (A <- matrix(c(1,2,3, 2,3,4, 3,4,1), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
```

```
[3,]    3    4    1
> (b <- c(3, 0, 1))
[1] 3 0 1
> (x <- solve(A, b))
[1] -10   8  -1
> round(A%*%x, 6) # reverse
     [,1]
[1,]    3
[2,]    0
[3,]    1
```

13.8 矩陣特徵值與特徵向量

假設 $\mathbf{A}_{n\times n}$ 是一個 方塊矩陣 或 方陣 (square matrix), 若存在非 0 的 向量 $\underline{\mathbf{x}}$ 與 純量 (scalar) λ, 使得

$$\mathbf{A}\underline{\mathbf{x}} = \underline{\mathbf{x}}\lambda, \tag{13.8.1}$$

則純量 λ 爲 方陣 $\mathbf{A}_{n\times n}$ 的 特徵值 (eigenvalue), 向量 $\underline{\mathbf{x}}$ 爲 方陣 $\mathbf{A}_{n\times n}$ 對應於 特徵值 λ 的 特徵向量 (eigenvector). 求取 特徵值 與 特徵向量, 令 $\mathbf{I}$ 爲 單位矩陣 (identity matrix), 將 $\mathbf{A}\underline{\mathbf{x}} = \underline{\mathbf{x}}\lambda$ 改寫成

$$\mathbf{A}\underline{\mathbf{x}} = \underline{\mathbf{x}}\mathbf{I}\lambda \tag{13.8.2}$$

$$\Rightarrow \quad \mathbf{det}(\lambda\mathbf{I} - \mathbf{A})\underline{\mathbf{x}} = \underline{\mathbf{0}}. \tag{13.8.3}$$

上述方程式爲 方陣 $\mathbf{A}_{n\times n}$ 的 特徵方程式 (characteristic equation).

在 R 使用矩陣函式 `eign()` 可以計算矩陣的 特徵值 與 特徵向量. 函式指令爲

```
> eigen(x, symmetric, only.values = FALSE)
```

其中的主要引數分別爲:

- `x`: 方陣 $\mathbf{A}_{n\times n}$.
- `symmetric = TRUE`: 假設爲對稱矩陣.

- `only.values = TRUE`: 只回傳特徵值 λ 形成的列表 (`EV$value`).
- `EV <- eigen(x)` 回傳一個含有兩個成分的列表 (list), `EV`.
  - 特徵值所形成的向量 `EV$val`.
  - 特徵向量所組成的矩陣 `EV$vec`.
- `prod(as.vector(eigen(x)$values))` 可以計算 **A** 的矩陣行列式值.

```
> ## eigen(): eigen valves and vectors
> (A <- matrix(c(1,2,3, 2,3,4, 3,4,1), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    1
> eigen(x = A, only.values = TRUE)
$values
[1]  7.862717 -0.190368 -2.672349

$vectors
NULL

> A.eig <- eigen(x = A)
> A.eig$values
[1]  7.862717 -0.190368 -2.672349
> A.eig$vectors
          [,1]      [,2]      [,3]
[1,] -0.452500  0.781340 -0.429827
[2,] -0.670108 -0.615944 -0.414208
[3,] -0.588387  0.100601  0.802297
> prod(A.eig$values)
[1] 4
>
> ## determinant
> prod(as.vector(eigen(A)$values))
[1] 4
> det(A)
[1] 4
```

13.9 矩陣的奇異值分解

假設 $\mathbf{X}_{m\times n}$ 是一個矩陣, 且矩陣 $\mathbf{X}$ 的秩 (rank) 爲 $\text{rank}(\mathbf{X}) = r$, 則矩陣 $\mathbf{X}$ 的 奇異值分解 (SVD, Singular Value Decomposition) 定義爲

$$\mathbf{X}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{D}_{m\times n}\mathbf{V}^T_{n\times n} \tag{13.9.1}$$

其中矩陣 $\mathbf{U}_{m\times m}$, $\mathbf{U}^{-1} = \mathbf{U}^T$, 矩陣 $\mathbf{V}_{n\times n}$, $\mathbf{V}^{-1} = \mathbf{V}^T$, $\mathbf{D}_{m\times n}$, 爲主對角矩陣.

$$\mathbf{D}_{m\times n} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \ldots 0 \\ 0 & \ddots & 0 & \vdots \ \ldots 0 \\ 0 & 0 & \sigma_r & 0 \ldots 0 \\ 0 & \ldots & 0 & 0 \ \ddots \ 0 \\ 0 & \ldots & 0 & 0 \ldots 0 \end{bmatrix} \tag{13.9.2}$$

σ_i 爲 奇異值 (singular value), 其中 $\sigma_i \neq 0, i = 1, 2, \ldots, r$, 若 $i > r$, 則 $\sigma_i = 0$. SVD 的分解結果並非唯一的, 習慣上, 奇異值由大至小排序, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$. 矩陣 $\mathbf{X}$ 的 奇異值分解 表示 矩陣 $\mathbf{X}$ 可由矩陣 $\mathbf{U}$ 的前 r 個行向量 (column vector), 矩陣 $\mathbf{V}$ 的前 r 個列向量 (row vector), 與 矩陣 $\mathbf{D}$ 的左上方 $r \times r$ 主對角矩陣 之乘積表示. 以 R 函式表示爲

$$\mathbf{X} = \mathbf{U} \ \texttt{\%*\%} \ \mathbf{D} \ \texttt{\%*\%} \ \mathbf{V}^T = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T \tag{13.9.3}$$

$$\texttt{X} = \texttt{U} \ \ \texttt{\%*\%} \ \ \texttt{D} \ \ \texttt{\%*\%} \ \ \texttt{t(V)} \tag{13.9.4}$$

R 使用函式 `svd()` 可以對矩陣 $\mathbf{X}$ 執行奇異值分解. 函式指令爲

```
> svd(x, nu = min(n, p), nv = min(n, p), LINPACK = FALSE)
> La.svd(x, nu = min(n, p), nv = min(n, p))
```

其中的主要引數分別爲:

- `x`: 矩陣 $\mathbf{X}$, 需要進行奇異值分解.

- `svd(x)` 回傳 `d`, `u` 和 `v` 構成的一個列表, 計算如同 `X = U %*% D %*% t(V)`.
  - `$d`: 向量 **d**, 一個和 **X** 維度一至的一個僅具主對角正值元素之向量. **D** 實際上以對角元素向量 **d** 的形式傳回.
  - `$u`: 矩陣 **U**, 一個和 **X** 列空間 (row) 一致的正交列的矩陣.
  - `$v`: 矩陣 **V**, 一個和 **X** 行空間 (欄空間, column) 一致的正交列的矩陣.
- 如果 **X** 是一個方塊矩陣, 則使用函式指令 `absdetM <- prod(svd(X)$d)`, 可以計算 **X** 的矩陣行列式值.
- 主對角矩陣 **D** 之 主對角線元素和 (trace), 相當於特徵值和, 可以利用 主對角向量 **d** 得到.

```
> ## SVD decomposition
> (A <- matrix(c(1,2,3, 2,3,4, 3,4,1), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    1
> svd(A)
$d
[1] 7.862717 2.672349 0.190368

$u
          [,1]      [,2]      [,3]
[1,] -0.452500 -0.429827 -0.781340
[2,] -0.670108 -0.414208  0.615944
[3,] -0.588387  0.802297 -0.100601

$v
          [,1]      [,2]      [,3]
[1,] -0.452500  0.429827  0.781340
[2,] -0.670108  0.414208 -0.615944
[3,] -0.588387 -0.802297  0.100601

> #
> b.svd <- svd(A)
> b.svd$d
[1] 7.862717 2.672349 0.190368
> b.svd$u
          [,1]      [,2]      [,3]
[1,] -0.452500 -0.429827 -0.781340
[2,] -0.670108 -0.414208  0.615944
```

```
[3,] -0.588387  0.802297 -0.100601
> b.svd$v
           [,1]      [,2]      [,3]
[1,] -0.452500  0.429827  0.781340
[2,] -0.670108  0.414208 -0.615944
[3,] -0.588387 -0.802297  0.100601
>
> # determinant
> det(A)
[1] 4
> prod(svd(A)$d)
[1] 4

> # SVD
> (A <- matrix(c(1,2,3, 2,3,4, 3,4,1), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    1
> b.svd <- svd(A)
> b.svd$d
[1] 7.862717 2.672349 0.190368
> b.svd$u
           [,1]      [,2]      [,3]
[1,] -0.452500 -0.429827 -0.781340
[2,] -0.670108 -0.414208  0.615944
[3,] -0.588387  0.802297 -0.100601
> b.svd$v
           [,1]      [,2]      [,3]
[1,] -0.452500  0.429827  0.781340
[2,] -0.670108  0.414208 -0.615944
[3,] -0.588387 -0.802297  0.100601
>
> # determinant
> det(A)
[1] 4
> prod(svd(A)$d)
[1] 4
```

13.10 矩陣的 QR 分解

矩陣除了奇異值分解分解外, 尚有 QR 與 Cholski 分解, QR 分解是將 矩陣 $\mathbf{X}$ 分解成 一個 $\mathbf{Q}$ 矩陣 與 一個 $\mathbf{R}$ 矩陣 的乘積, QR 分解的定義爲

$$\mathbf{X} = \mathbf{Q}\mathbf{R} \tag{13.10.1}$$

其中 $\mathbf{R}$ 是 上三角矩陣 (upper triangular matrix), $\mathbf{Q}$ 是 直交矩陣 (orthogonal matrix), $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. 或者 $\mathbf{Q}$ 是擁有互相垂直的單位行向量 (欄向量, column vector) 之矩陣, 而且此單位行向量之集合, 形成矩陣 $\mathbf{X}$ 之行向量空間 (欄向量空間, column space). 函式 `qr()` 指令爲

```
> qr(x, tol = 1e-07 , LAPACK = FALSE, ...)
> qr.solve(a, b, tol = 1e-7)
```

其中的主要引數分別爲:

- `x` 爲矩陣 $\mathbf{X}$, 將執行 QR 分解.
- `a` 爲矩陣.
- `b` 爲向量.
- `B.qr.list <- qr(x)` 可進行 QR 分解, 回傳一列表 `B.qr.list`. 其中 列表 `B.qr.list` 中一個元素爲 矩陣 `B.qr.list$qr`, R 是將 $\mathbf{Q}$ 矩陣與 $\mathbf{R}$ 矩陣放在一起.
- R 函式指令 `qr.X(B.qr.list)`, `qr.Q(B.qr.list)` 與 `qr.R(B.qr.list)` 可以將 陣列 `B.qr.list` 還原成 $\mathbf{X}$, 以及 $\mathbf{Q}$ 與 $\mathbf{R}$.
- R 的 QR 分解在計算矩陣行列式值 (determinant) 較 `eigen()` 要有效率.

```
> ## qr(): QR decomposition
> (X <- matrix(c(1,2,3, 2,3,4, 3,4,1), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
```

```
[3,]    3    4    1
> (Xqr <- qr(X))
$qr
          [,1]      [,2]     [,3]
[1,] -3.741657 -5.345225 -3.74166
[2,]  0.534522  0.654654  3.05505
[3,]  0.801784  0.988693 -1.63299

$rank
[1] 3

$qraux
[1] 1.26726 1.14995 1.63299

$pivot
[1] 1 2 3

attr(,"class")
[1] "qr"
> (X.back <- qr.X(Xqr))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    1
> (R.back <- qr.R(Xqr))
         [,1]      [,2]     [,3]
[1,] -3.74166 -5.345225 -3.74166
[2,]  0.00000  0.654654  3.05505
[3,]  0.00000  0.000000 -1.63299
> (Q.back <- qr.Q(Xqr))
          [,1]      [,2]      [,3]
[1,] -0.267261  0.872872  0.408248
[2,] -0.534522  0.218218 -0.816497
[3,] -0.801784 -0.436436  0.408248
> Q.back%*%R.back
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    3    4
[3,]    3    4    1
```

13.11 矩陣的 Cholski 分解

矩陣 $\mathbf{X}$ 是 對稱 (symmetric) 且 正向 或 正定 (positive definite) 的實數矩陣, 可以利用 Cholski 分解,

$$\mathbf{X} = \mathbf{U}^T\mathbf{U} = \mathbf{L}\mathbf{L}^T, \tag{13.11.1}$$

即 $\mathbf{X}$ 可分解成 上三角矩陣 (upper triangular matrix), $\mathbf{U}^T$, 與 $\mathbf{U}$ 的乘積; 或是 下三角矩陣 (lower triangular matrix), $\mathbf{L}$ 與 $\mathbf{L}^T$ 的乘積.

上三角矩陣 $\mathbf{U}$ 的主對角線上的項皆爲正數, 且分解的方式是唯一的. 且 上三角矩陣 $\mathbf{U}$ 的主對角上的元素爲

$$\mathbf{U}_{1,1} = \sqrt{\mathbf{X}_{1,1}}, \tag{13.11.2}$$

$$\mathbf{U}_{i,i} = \sqrt{X_{i,i}} - \sum_k \mathbf{U}^2_{k,i}, \; k = 1, \ldots, i-1. \tag{13.11.3}$$

在線性方程式中

$$\mathbf{A}\underline{\mathbf{x}} = \underline{\mathbf{b}}, \tag{13.11.4}$$

若 $\mathbf{A}$ 是對稱正定的實數矩陣, 可以利用 Cholski 分解, $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, 先解線性方程式

$$\mathbf{L}\underline{\mathbf{y}} = \underline{\mathbf{b}}, \tag{13.11.5}$$

最後解線性方程式

$$\mathbf{L}^T\underline{\mathbf{x}} = \underline{\mathbf{y}}. \tag{13.11.6}$$

使用矩陣函式 chol() 可以執行 Cholski 分解. 函式指令爲

```
> chol(x, pivot = FALSE,  LINPACK = FALSE, tol = -1, ...)
```

其中的主要引數 x 爲對稱正定的實數矩陣 $\mathbf{X}$, 回傳上三角矩陣 $\mathbf{R}$. 當函式 chol(X) 執行 Cholski 分解, 若矩陣 $\mathbf{X}$ 不是對稱正向矩陣, 則函式無法執行.

```
> ## chol(): Choleski Decomposition
> (X <- matrix(c(2,-1,-3, -1,2,4, -3,4,9), nrow = 3, byrow = T))
     [,1] [,2] [,3]
[1,]    2   -1   -3
[2,]   -1    2    4
[3,]   -3    4    9
> (XU.chol <- chol(X))
        [,1]      [,2]     [,3]
[1,] 1.41421 -0.707107 -2.12132
[2,] 0.00000  1.224745  2.04124
[3,] 0.00000  0.000000  0.57735
> t(XU.chol)%*%XU.chol
     [,1] [,2] [,3]
[1,]    2   -1   -3
[2,]   -1    2    4
[3,]   -3    4    9
> crossprod(XU.chol)
     [,1] [,2] [,3]
[1,]    2   -1   -3
[2,]   -1    2    4
[3,]   -3    4    9
> (X.chol <- chol(X, pivot = TRUE))
     [,1] [,2]     [,3]
[1,]    3   -1 1.333333
[2,]    0    1 0.333333
[3,]    0    0 0.333333
attr(,"pivot")
[1] 3 1 2
attr(,"rank")
[1] 3
```