Álgebra Linear

Prof. Wagner H. Bonat

Laboratório de Estatística e Geoinformação
Departamento de Estatística
Universidade Federal do Paraná

Vetores

Definição

- Vetores são grandezas (matemáticas ou físicas) com módulo e direção.
- Notações usuais: $\vec{v}$ e $\mathbf{v}$.
- Um vetor é escrito listando seus componentes em uma linha ou coluna.

$$\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \quad \text{ou} \quad \mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}.$$

- Vetor pode ser representado por v_i, onde $i = 1, 2, 3$.
- Módulo de um vetor é o seu comprimento

$$|\mathbf{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}.$$

- Vetor unitário $\hat{v} = \frac{\mathbf{v}}{|\mathbf{v}|}$.

Vetores

- Em situações físicas os vetores são restritos a três dimensões.
- A idéia pode ser generalizada.
- Um vetor é uma lista de n números (elementos ou componentes) escritos em linha ou coluna.

$$\boldsymbol{v} = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix} \quad \text{ou} \quad \boldsymbol{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}.$$

- Um elemento de um vetor é chamado v_i, onde o subscrito denota a posição do elemento na linha ou coluna.
- Elementos em linha **vetor linha**.
- Elementos em coluna **vetor coluna**.

Operações com vetores

- Dois vetores são iguais se tem o mesmo tamanho e se todos os seus elementos em posições equivalentes são iguais.
- Nem todas as operações fundamentais em matemática são definidas para vetores.
- Vetores podem ser somados, subtraídos e multiplicados (de certa forma).
- Vetores não podem ser divididos.
- Existem operações especiais para vetores, como produto interno e cruzado.

Operações com vetores

- Dois vetores podem ser somados ou subtraídos apenas se forem do mesmo tipo e do mesmo tamanho.
- Sejam dois vetores $\boldsymbol{x}$ e $\boldsymbol{y}$ adequados e α um escalar, as seguintes operações são bem definidas.
  - Soma $\boldsymbol{x} + \boldsymbol{y} = [x_i + y_i] = [x_1 + y_1, \ldots, x_n + y_n]$.
  - Subtração $\boldsymbol{x} - \boldsymbol{y} = [x_i - y_i] = [x_1 - y_1, \ldots, x_n - y_n]$.
  - Multiplicação por escalar $\alpha \boldsymbol{x} = [\alpha x_1, \ldots, \alpha x_n]$.
  - Transposta de um vetor: A operação transposta transforma um **vetor coluna** em um **vetor linha** e vice-versa. Por exemplo,

$$\boldsymbol{x} = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix} \qquad \boldsymbol{x}^\top = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$

Multiplicação de dois vetores

- Produto interno ou escalar $\rightarrow$ resulta um escalar (número), i.e.

$$\boldsymbol{x} \cdot \boldsymbol{y} = [x_1 y_1 + x_2 y_2 + \ldots + x_n y_n].$$

- Dependência e independência linear de um conjunto de vetores.
- Diz-se que um conjunto de vetores é **linearmente independente** se

$$\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} + \ldots \alpha_n \boldsymbol{v_n} = 0$$

  é satisfeita se e somente se $\alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$.
- Do contrário, diz-se que os vetores são **linearmente dependentes.**

Operações com vetores em R

- Todas as operações são trivialmente definidas em R.

```
# Definindo vetores
x <- c(4,5,6)
y <- c(1,2,3)
# Soma
x + y

## [1] 5 7 9

# Subtração
x-y

## [1] 3 3 3
```

Operações com vetores em R

```r
# Multiplicação por escalar
alpha = 10
alpha*x
```

```
## [1] 40 50 60
```

```r
alpha*y
```

```
## [1] 10 20 30
```

```r
# Produto interno
x%*%y
```

```
##      [,1]
## [1,]   32
```

Cuidado com a lei da reciclagem!!

- O R usa a lei da reciclagem o que pode trazer resultados inesperado quando fazendo operações em vetores.

```
# Definindo vetores de tamanhos diferentes
x <- c(4,5,6,5,6)
y <- c(1,2,3) # Note que o 1 e 2 de y foram reciclados
# Soma
x + y
```

```
## [1] 5 7 9 6 8
```

Cuidado!!

- Cuidado com o operador $*$ quando trabalhando com vetores a multiplicação é feita usando o operador $\% * \%$.

```
x <- c(4,5,6)
y <- c(1,2,3)
x*y # Não é o produto escalar
```

```
## [1]  4 10 18
```

```
x%*%y # Produto escalar
```

```
##      [,1]
## [1,]   32
```

Matrizes

Matrizes

- Uma **matriz** é um arranjo retangular de números.
- O tamanho de uma matriz refere-se ao seu número de linhas e colunas.
- Uma matrix $(m \times n)$ tem m linhas e n colunas:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \ddots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{bmatrix}.$$

- Um **vetor linha** é uma matriz com uma linha e várias colunas.
- Um **vetor coluna** é uma matriz com uma coluna e várias linhas.

Operações com matrizes

- Multiplicação por um escalar

$$\alpha A = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} & \dots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \ddots & \alpha a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha a_{m1} & \dots & \dots & \alpha a_{mn} \end{bmatrix}.$$

Soma e subtração de duas matrizes

- Duas matrizes podem ser somadas ou subtraídas somente se tiverem o mesmo tamanho.
- A soma ou subtração de duas matrizes A e B ambas $(m \times n)$ é uma matriz C cujos elementos são dados por:
  1. Soma $c_{ij} = a_{ij} + b_{ij}$.
  2. Subtração $c_{ij} = a_{ij} - b_{ij}$.
- Exemplo

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 20 \\ 30 & 40 \\ 50 & 60 \end{bmatrix} = \begin{bmatrix} 11 & 22 \\ 33 & 44 \\ 55 & 66 \end{bmatrix}.$$

Transposta de uma matriz

- Operação de transposição rearranja uma matriz de forma que suas linhas são transformadas em colunas e vice-versa.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^\top = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

Multiplicação de matrizes

- Multiplicação $\mathbf{C} = \mathbf{AB}$ é definida apenas quando o número de colunas de $\mathbf{A}$ é igual ao número de linhas de $\mathbf{B}$.
- $\underset{m \times n}{\mathbf{C}} = \underset{m \times q}{\mathbf{A}} \underset{q \times n}{\mathbf{B}}$.
- Cada elemento $c_{ij} = \sum_{k=1}^{q} a_{ik} b_{kj}$.
- Exemplo,

$$\begin{bmatrix} 2 & -1 \\ 8 & 3 \\ 6 & 7 \end{bmatrix} \begin{bmatrix} 4 & 9 & 1 & -3 \\ -5 & 2 & 4 & 6 \end{bmatrix} =$$

$$\begin{bmatrix} (2 \cdot 4 + -1 \cdot -5) & (2 \cdot 9 + -1 \cdot 2) & (2 \cdot 1 + -1 \cdot 4) & (2 \cdot -3 + -1 \cdot 6) \\ (8 \cdot 4 + 3 \cdot -5) & (8 \cdot 9 + 3 \cdot 2) & (8 \cdot 1 + 3 \cdot 4) & (8 \cdot -3 + 3 \cdot 6) \\ (6 \cdot 4 + 7 \cdot -5) & (6 \cdot 9 + 7 \cdot 2) & (6 \cdot 1 + 7 \cdot 4) & (6 \cdot -3 + 7 \cdot 6) \end{bmatrix} =$$

$$\begin{bmatrix} 13 & 16 & -2 & -12 \\ 17 & 78 & 20 & -6 \\ -11 & 68 & 34 & 24 \end{bmatrix}.$$

Matrizes especiais

- Matriz quadrada: mesmo número de linhas e colunas.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

- Elementos a_{ii} são elementos **diagonais**.
- Elementos a_{ij} para $i \neq j$ são elementos **fora da diagonal**.
- Elementos a_{ij} para $j > i$ são elementos **acima da diagonal**.
- Elementos a_{ij} para $i > j$ são elementos **abaixo da diagonal**.

Matrizes especiais

- Matriz diagonal

$$D = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

- Matriz triangular superior

$$U = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

Matrizes especiais

- Matriz triangular inferior

$$\mathbf{L} = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

- Matriz identidade

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Matrizes especiais

- Matriz zero

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

- Matriz simétrica é quadrada na qual $a_{ij} = a_{ji}$.

$$A = \begin{bmatrix} 1 & 0.8 & 0.6 & 0.4 \\ 0.8 & 1 & 0.2 & 0.4 \\ 0.6 & 0.2 & 1 & 0.1 \\ 0.4 & 0.4 & 0.1 & 1 \end{bmatrix}.$$

Inversa de uma matriz

- Divisão é uma importante operação não definida para matrizes.
- A inversa serve a um propósito equivalente.
- Uma matriz quadrada pode ser invertida desde que exista uma matriz B de mesmo tamanho tal que $AB = I$.
- A matrix B é chamada inversa de A e denotada por A^{-1}.
- Resumindo,

$$AA^{-1} = I.$$

- Inversas são fundamentais em *Estatística*.
- Em geral não calculamos explicitamente.
- São extremamentes caras computacionalmente.
- Vamos ver como obter a inversa na aula sobre métodos numéricos.

Determinante de uma matriz

- O **determinante** é um número e definido apenas para matrizes quadradas.
- Fundamental para o cálculo da inversa.
- Fornece informações sobre a existência ou não de soluções para um conjunto de equações simultâneas (lembre-se regressão linear).
- Difícil de obter para matrizes maiores que (3×3).

Determinante de uma matriz

- Formalmente o **determinante** de uma matriz A é o número

$$\text{det}(A) = \sum_j (-1)^k a_{1j_1} a_{2j_2}, \ldots, a_{nj_n},$$

onde a soma é realizada para todas as $n!$ permutações de grau n e k é o número de mudanças necessárias para que os segundos subscritos sejam colocados na ordem $1, 2, \ldots, n$.

- Exemplo,

$$\text{det}(A) = \text{det}\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = (-1)^0 a_{11}a_{22} + (-1)^1 a_{12}a_{21}.$$

Propriedades de matrizes

- Sendo A, B e C matrizes adequadas as seguintes propriedades são válidas.
  1. $A + B = B + A$.
  2. $(A + B) + C = A + (B + C)$.
  3. $\alpha(A + B) = \alpha A + \alpha B$.
  4. $(\alpha + \beta)A = \alpha A + \beta A$.
- Sendo A e B quadradas em geral $AB \neq BA$.
  1. $(A + B)C = AC + BC$.
  2. $A(B + C) = AB + AC$.
  3. $\alpha(AB) = (\alpha A)B = A(\alpha B)$.
  4. $(AB)^\top = B^\top A^\top$.
  5. $(A^\top)^\top = A$.
  6. $(A^{-1})^{-1} = A$.
  7. $(AB)^{-1} = B^{-1}A^{-1}$.

Matrizes em R

- Iniciando matrizes em R.

```r
# Definindo duas matrizes
A <- matrix(c(2,5,6,-1,3,1), ncol = 2, nrow = 3)
B <- matrix(c(1,-5,3,3,2,7), ncol = 2, nrow = 3)
A
```

```
##      [,1] [,2]
## [1,]    2   -1
## [2,]    5    3
## [3,]    6    1
```

```r
B
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]   -5    2
## [3,]    3    7
```

Operações com matrizes em R

- Operações básicas com matrizes.

```
# Tamanho
dim(A)
```

```
## [1] 3 2
```

```
# Soma
A + B
```

```
##      [,1] [,2]
## [1,]    3    2
## [2,]    0    5
## [3,]    9    8
```

Operações com matrizes em R

```r
# Subtração
A - B
```

```
##      [,1] [,2]
## [1,]    1   -4
## [2,]   10    1
## [3,]    3   -6
```

```r
# Multiplicação por escalar
alpha = 10
alpha*A
```

```
##      [,1] [,2]
## [1,]   20  -10
## [2,]   50   30
## [3,]   60   10
```

Operações com matrizes em R

- Multiplicação matricial

```r
# A%*%B # Matrices não compatíveis
A <- matrix(c(2,8,6,-1,3,7),3,2)
B <- matrix(c(4,-5,9,2,1,4,-3,6),2,4)
A%*%B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   -2  -12
## [2,]   17   78   20   -6
## [3,]  -11   68   34   24
```

```r
# B%*%A
# Error in B %*% A : non-conformable arguments
```

Determinante

- Determinante

```r
# Matriz quadrada
A <- matrix(c(1,0.8,0.8,1),2,2)
# Determinante de A
det(A)
```

```
## [1] 0.36
```

Inversa

```r
# Inversa de A
inv_A <- solve(A)
inv_A
```

```
##             [,1]      [,2]
## [1,]  2.777778 -2.222222
## [2,] -2.222222  2.777778
```

```r
A%*%inv_A # Matriz identidade
```

```
##                 [,1] [,2]
## [1,]  1.000000e+00    0
## [2,] -4.440892e-16    1
```

Sistemas de equações lineares

Sistemas de equações

- Sistema com duas equações:

$$\begin{aligned} f_1(x_1, x_2) &= 0 \\ f_2(x_1, x_2) &= 0. \end{aligned}$$

- Solução consiste em encontrar $\hat{x}_1$ e $\hat{x}_2$ que satisfaça o sistema.
- Sistema com n equações

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0. \end{aligned}$$

- Genericamente, tem-se

$$\mathbf{f}(\boldsymbol{x}) = \mathbf{0}.$$

Sistemas de equações lineares

- Cada equação é linear na incógnita.
- Solução analítica em geral é possível.
- Exemplo:

$$\begin{aligned} 7x_1 + 3x_2 &= 45 \\ 4x_1 + 5x_2 &= 29. \end{aligned}$$

- Solução analítica: $x_1 = 6$ e $x_2 = 1$.
- Resolver no quadro (tedioso!!).
- Três possíveis casos:
  1. Uma única solução (sistema não singular).
  2. Infinitas soluções (sistema singular).
  3. Nenhuma solução (sistema impossível).

Sistemas de equações lineares

- Representação matricial do sistema de equações lineares:

$$A = \begin{bmatrix} 7 & 3 \\ 4 & 5 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{e} \quad \mathbf{b} = \begin{bmatrix} 45 \\ 29 \end{bmatrix}.$$

- De forma geral, tem-se

$$A\mathbf{x} = \mathbf{b}.$$

Operações com linhas

- Sem qualquer alteração na relação linear, é possível
  1. Trocar a posição de linhas:

$$\begin{aligned} 4x_1 + 5x_2 &= 29 \\ 7x_1 + 3x_2 &= 45. \end{aligned}$$

  2. Multiplicar qualquer linha por uma constante, aqui $4x_1 + 5x_2$ por $\frac{1}{4}$, obtendo

$$x_1 + \frac{5}{4}x_2 = \frac{29}{4} \qquad (1)$$

$$7x_1 + 3x_2 = 45. \qquad (2)$$

Operações com linhas

3. Subtrair um múltiplo de uma linha de uma outra, aqui $7 * Eq.(1)$ menos Eq. (2), obtendo

$$
\begin{aligned}
x_1 + \frac{5}{4}x_2 &= \frac{29}{4} \\
0x_1 + (\frac{35}{4} - 3)x_2 &= \frac{203}{4} - 45.
\end{aligned}
$$

- Fazendo as contas, tem-se

$$
0x_1 + \frac{23}{4}x_2 = \frac{23}{4}.
$$

Solução de sistemas lineares

- Forma geral de um sistema com n equações lineares:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned}$$

- Matricialmente, tem-se

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- Métodos diretos e métodos iterativos.

Métodos diretos

- O sistema de equações é manipulado até se transformar em um sistema equivalente de fácil resolução.
- Triangular superior:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

- Substituição regressiva

$$x_n = \frac{b_n}{a_{nn}} \quad x_i = \frac{b_i - \sum_{j=i+1}^{j=n} a_{ij} x_j}{a_{ii}}, \quad i = n-1, n-2, \ldots, 1.$$

Métodos diretos

- Triangular inferior:

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

- Substituição progressiva

$$x_1 = \frac{b_1}{a_{11}} \quad x_i = \frac{b_i - \sum_{j=i}^{j=i-1} a_{ij}x_j}{a_{ii}}, \quad i = 2, 3, \ldots, n.$$

Métodos diretos

- Diagonal:

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

Método de Eliminação de Gauss

Métodos diretos: Eliminação de Gauss

- Método de eliminação de Gauss consiste em manipular o sistema original usando operações de linha até obter um sistema triangular superior.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{23} & a_{33} & a_{34} \\ a_{41} & a_{24} & a_{34} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \rightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b'_3 \\ b'_4 \end{bmatrix}$$

- Usar eliminação progressiva no novo sistema para obter a solução.
- Resolva o seguinte sistema usando Eliminação de Gauss.

$$\begin{bmatrix} 3 & 2 & 6 \\ 2 & 4 & 3 \\ 5 & 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 24 \\ 23 \\ 33 \end{bmatrix}$$

Métodos diretos: Eliminação de Gauss

- Passo 1: Encontrar o pivô e eliminar os elementos abaixo dele usando operações de linha.

$$\begin{bmatrix} [3] & 2 & 6 \\ 2-\frac{2}{3}3 & 4-\frac{2}{3}2 & 3-\frac{2}{3}6 \\ 5-\frac{5}{3}3 & 3-\frac{5}{3}2 & 4-\frac{5}{3}6 \end{bmatrix} \begin{bmatrix} 24 \\ 23-\frac{2}{3}24 \\ 33-\frac{5}{3}24 \end{bmatrix} \longrightarrow \begin{bmatrix} [3] & 2 & 6 \\ 0 & \frac{8}{3} & -1 \\ 0 & -\frac{1}{3} & -6 \end{bmatrix} \begin{bmatrix} 24 \\ 7 \\ -7 \end{bmatrix}$$

- Passo 2: Encontrar o segundo pivô e eliminar os elementos abaixo dele usando operações de linha.

$$\begin{bmatrix} 3 & 2 & 6 \\ 0 & [\frac{8}{3}] & -1 \\ 0 & -\frac{1}{3}-\left(-\frac{3}{24}\right)\left(\frac{8}{3}\right) & -6-(-\frac{3}{24})(-1) \end{bmatrix} \begin{bmatrix} 24 \\ 7 \\ -7-(-\frac{3}{24})(7) \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 2 & 6 \\ 0 & [\frac{8}{3}] & -1 \\ 0 & 0 & -\frac{147}{24} \end{bmatrix} \begin{bmatrix} 24 \\ 7 \\ -\frac{147}{24} \end{bmatrix}$$

- Passo 3: Substituição regressiva.

Métodos diretos: Eliminação de Gauss

- Usando a fórmula de substituição regressiva temos:
  1. $x_3 = \frac{b_3}{a_{33}} = 1.$
  2. $x_2 = \frac{b_2 - a_{23}x_3}{a_{22}} = 3.$
  3. $x_1 = \frac{(b_1 - (a_{12}x_2 + a_{13}x_3)}{a_{11}} = 4.$
- A extensão do procedimento para um sistema com n equações é trivial.
  1. Transforme o sistema em triangular superior usando operações linhas.
  2. Resolva o novo sistema usando substituição regressiva.
- Potenciais problemas do método de eliminação de Gauss:
  1. O elemento pivô é zero.
  2. O elemento pivô é pequeno em relação aos demais termos.

Eliminação de Gauss com pivotação

- Considere o sistema

$$\begin{aligned} 0x_1 + 2x_2 + 3x_2 &= 46 \\ 4x_1 - 3x_2 + 2x_3 &= 16 \\ 2x_1 + 4x_2 - 3x_3 &= 12 \end{aligned}$$

- Neste caso o pivô é zero e o procedimento não pode começar.
- Pivotação - trocar a ordem das linhas.
  1. Evitar pivôs zero.
  2. Diminuir o número de operações necessárias para triangular o sistema.

$$\begin{aligned} 4x_1 - 3x_2 + 2x_3 &= 16 \\ 2x_1 + 4x_2 - 3x_3 &= 12 \\ 0x_1 + 2x_2 + 3x_2 &= 46 \end{aligned}$$

Eliminação de Gauss com pivotação

- Se durante o procedimento uma equação pivô tiver um elemento nulo e o sistema tiver solução, uma equação com um elemento pivô diferente de zero sempre existirá.
- Cálculos numéricos são menos propensos a erros e apresentam menores erros de arredondamento se o elemento pivô for grande em valor absoluto.
- É usual ordenar as linhas para que o maior valor seja o primeiro pivô.

Implementação: Eliminação de Gauss sem pivotação

- Passo 1: Obtendo uma matriz triangular superior.

```r
gauss <- function(A, b) {
  # Sistema aumentado
  Ae <- cbind(A, b)
  n_row <- nrow(Ae)
  n_col <- ncol(Ae)
  # Matriz para receber os resultados
  SOL <- matrix(NA, n_row, n_col)
  # Pivotação
  #Ae <- Ae[order(Ae[,1], decreasing = TRUE),]
  SOL[1,] <- Ae[1,]
  pivo <- matrix(0, n_col, n_row)
  for(j in 1:c(n_row-1)) {
    for(i in c(j+1):c(n_row)) {
      pivo[i,j] <- Ae[i,j]/SOL[j,j]
      SOL[i,] <- Ae[i,] - pivo[i,j]*SOL[j,]
```

Implementação: Eliminação de Gauss sem pivotação

- Passo 2: Substituição regressiva.

```
sub_reg <- function(SOL) {
  n_row <- nrow(SOL)
  n_col <- ncol(SOL)
  A <- SOL[1:n_row,1:n_row]
  b <- SOL[,n_col]
  n <- length(b)
  x <- c()
  x[n] <- b[n]/A[n,n]
  for(i in (n-1):1) {
    x[i] <- (b[i] - sum(A[i,c(i+1):n]*x[c(i+1):n] ))/A[i,i]
  }
  return(x)
}
```

Aplicação: Eliminação de Gauss sem pivotação

- Resolva o sistema:

$$\begin{bmatrix} 3 & 2 & 6 \\ 2 & 4 & 3 \\ 5 & 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 24 \\ 23 \\ 33 \end{bmatrix}.$$

```
A <- matrix(c(3,2,5,2,4,3,6,3,4),3,3)
b <- c(24,23,33)
# Passo 1: Triangularização
S <- gauss(A, b)
S
```

```
##      [,1]          [,2]   [,3]   [,4]
## [1,]    3  2.000000e+00  6.000 24.000
## [2,]    0  2.666667e+00 -1.000  7.000
## [3,]    0 -5.551115e-17 -6.125 -6.125
```

Aplicação: Eliminação de Gauss sem pivotação

```
# Passo 2: Substituição regressiva
sol = sub_reg(SOL = S)
sol
```

```
## [1] 4 3 1
```

```
# Verificando a solução
A%*%sol
```

```
##       [,1]
## [1,]   24
## [2,]   23
## [3,]   33
```

Decomposição LU

Decomposição LU

- Nos métodos de eliminação de Gauss e Gauss-Jordan resolvemos sistemas do tipo

$$Ax = \boldsymbol{b}.$$

- Sendo dois sistemas

$$Ax = \boldsymbol{b}_1, \quad \text{e} \quad Ax = \boldsymbol{b}_2.$$

- Cálculos do primeiro não ajudam a resolver o segundo.
- IDEAL! - Operações realizadas em A fossem dissociadas das operações em $\boldsymbol{b}$.

Decomposição LU

- Suponha que precisamos resolver vários sistemas do tipo
$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b}.$$
para diferentes $\boldsymbol{b}'s$.
- Opção 1 - Calcular a inversa A^{-1}, assim a solução
$$\boldsymbol{x} = \mathrm{A}^{-1}\boldsymbol{b}.$$
- Cálculo da inversa é computacionalmente ineficiente.

Algoritmo: Decomposição LU

- Decomponha (fatore) a matriz $\mathbf{A}$ em um produto de duas matrizes

$$\mathbf{A} = \mathbf{L}\mathbf{U},$$

onde $\mathbf{L}$ é triangular inferior e $\mathbf{U}$ é triangular superior.

- Baseado na decomposição o sistema tem a forma:

$$\mathbf{L}\mathbf{U}\boldsymbol{x} = \boldsymbol{b}. \tag{3}$$

- Defina $\mathbf{U}\boldsymbol{x} = \boldsymbol{y}$.
- Substituindo acima tem-se

$$\mathbf{L}\boldsymbol{y} = \boldsymbol{b}. \tag{4}$$

- Solução é obtida em dois passos
  1. Resolva Eq.(4) para obter $\boldsymbol{y}$ usando substituição progressiva.
  2. Resolva Eq.(3) para obter $\boldsymbol{x}$ usando substituição regressiva.

Obtendo as matrizes L e U

- Método de eliminação de Gauss e método de Crout.
- Dentro do processo de eliminação de Gauss as matrizes L e U são obtidas como um subproduto, i.e.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{41} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ m_{21} & 1 & & \\ m_{31} & m_{32} & 1 & \\ m_{41} & m_{42} & m_{43} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a'_{22} & a'_{23} & a'_{24} \\ 0 & 0 & a'_{33} & a'_{34} \\ 0 & 0 & 0 & a'_{44} \end{bmatrix}.$$

- Os elementos $m'_{ij}s$ são os multiplicadores que multiplicam a equação pivô.

Obtendo as matrizes **L** e **U**

- Relembre o exemplo de eliminação de Gauss.

$$\begin{bmatrix} [3] & 2 & 6 \\ 2-\frac{2}{3}3 & 4-\frac{2}{3}2 & 3-\frac{2}{3}6 \\ 5-\frac{5}{3}3 & 3-\frac{5}{3}2 & 4-\frac{5}{3}6 \end{bmatrix} \begin{bmatrix} 24 \\ 23-\frac{2}{3}24 \\ 33-\frac{5}{3}24 \end{bmatrix} \rightarrow \begin{bmatrix} [3] & 2 & 6 \\ 0 & \frac{8}{3} & -1 \\ 0 & -\frac{1}{3} & -6 \end{bmatrix} \begin{bmatrix} 24 \\ 7 \\ -7 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 2 & 6 \\ 0 & [\frac{8}{3}] & -1 \\ 0 & -\frac{1}{3}-\left(-\frac{3}{24}\right)\left(\frac{8}{3}\right) & -6-(-\frac{3}{24})(-1) \end{bmatrix} \begin{bmatrix} 24 \\ 7 \\ -7-(-\frac{3}{24})(7) \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 2 & 6 \\ 0 & [\frac{8}{3}] & -1 \\ 0 & 0 & -\frac{147}{24} \end{bmatrix} \begin{bmatrix} 24 \\ 7 \\ -\frac{147}{24} \end{bmatrix}$$

- Neste caso, tem-se

$$\mathbf{L} = \begin{bmatrix} 1 & & \\ \frac{2}{3} & 1 & \\ \frac{5}{3} & -\frac{3}{24} & 1 \end{bmatrix} \quad \text{e} \quad \mathbf{U} = \begin{bmatrix} 3 & 2 & 6 \\ 0 & \frac{8}{3} & -1 \\ 0 & 0 & -\frac{147}{24} \end{bmatrix}.$$

Decomposição LU com pivotação

- O método de eliminação de Gauss foi realizado sem pivotação.
- Como discutido a pivotação pode ser necessária.
- Quando realizada a pivotação as mudanças feitas devem ser armazenadas, tal que

$$\mathrm{PA} = \mathrm{LU}.$$

- P é uma matriz de permutação.
- Se as matrizes LU forem usadas para resolver o sistema

$$\mathrm{A}\boldsymbol{x} = \boldsymbol{b},$$

então a ordem das linhas de $\boldsymbol{b}$ deve ser alterada de forma consistente com a pivotação, i.e. $\mathrm{P}\boldsymbol{b}$.

Implementação: Decomposição LU

- Podemos facilmente modificar a função **gauss()** para obter a decomposição LU.

```
my_lu <- function(A) {
  n_row <- nrow(A)
  n_col <- ncol(A)
  # Matriz para receber os resultados
  SOL <- matrix(NA, n_row, n_col)
  SOL[1,] <- A[1,]
  pivo <- matrix(0, n_col, n_row)
  for(j in 1:c(n_row-1)) {
    for(i in c(j+1):c(n_row)) {
      pivo[i,j] <- A[i,j]/SOL[j,j]
      SOL[i,] <- A[i,] - pivo[i,j]*SOL[j,]
      A[i,] <- SOL[i,]
    }
  }
  diag(pivo) <- 1
```

Aplicação: Decomposição LU

- Fazendo a decomposição.

```r
LU <- my_lu(A) # Decomposição
LU
```

```
## $L
##           [,1]   [,2] [,3]
## [1,] 1.0000000  0.000    0
## [2,] 0.6666667  1.000    0
## [3,] 1.6666667 -0.125    1
##
## $U
##      [,1]          [,2]   [,3]
## [1,]    3  2.000000e+00  6.000
## [2,]    0  2.666667e+00 -1.000
## [3,]    0 -5.551115e-17 -6.125
```

Aplicação: Decomposição LU

- Verificando a solução.

```
LU$L %*% LU$U # Verificando a solução
```

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

Aplicação: Decomposição LU

- Resolvendo o sistema de equações.

```
# Passo 1: Substituição progressiva
y = forwardsolve(LU$L, b)
y

## [1] 24.000  7.000 -6.125

# Passo 2: Substituição regressiva
x = backsolve(LU$U, y)
x

## [1] 4 3 1

A%*%x # Verificando a solução

##      [,1]
## [1,]   24
## [2,]   23
## [3,]   33
```

Aplicação: Decomposição LU

- Função `lu()` do `Matrix` fornece a decomposição LU.

```
require(Matrix)
```

```
## Loading required package: Matrix
```

```
LU_M <- lu(A) # Calcula mas não retorna
LU_M <- expand(LU_M) # Captura as matrizes L U e P
# Substituição progressiva. NOTE MATRIZ DE PERMUTAÇÃO
y <- forwardsolve(LU_M$L, LU_M$P%*%b)
x = backsolve(LU_M$U, y) # Substituição regressiva
x
```

```
## [1] 4 3 1
```

Matriz inversa

Obtendo a inversa via decomposição LU

- O método LU é especialmente adequado para o cálculo da inversa.
- Lembre-se que a inversa de $\mathbf{A}$ é tal que

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}.$$

- O procedimento de cálculo da inversa é essencialmente o mesmo da solução de um sistema de equações lineares, porém com mais incognitas.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Três sistemas de equações diferentes, em cada sistema, uma coluna da matriz $\mathbf{X}$ é a incognita.

Implementação: Inversa via decomposição LU}

- Função para resolver o sistema usando decomposição LU.

```r
solve_lu <- function(LU, b) {
  y <- forwardsolve(LU_M$L, LU_M$P%*%b)
  x = backsolve(LU_M$U, y)
  return(x)
}
```

- Resolvendo vários sistemas

```r
my_solve <- function(LU, B) {
  n_col <- ncol(B)
  n_row <- nrow(B)
  inv <- matrix(NA, n_col, n_row)
  for(i in 1:n_col) {
    inv[,i] <- solve_lu(LU, B[,i])
  }
  return(inv)
}
```

Aplicação: Inversa via decomposição LU}

- Calcule a inversa de

$$A = \begin{bmatrix} 3 & 2 & 6 \\ 2 & 4 & 3 \\ 5 & 3 & 4 \end{bmatrix}$$

```
A <- matrix(c(3,2,5,2,4,3,6,3,4),3,3)
I <- Diagonal(3, 1)
# Decomposição LU
LU <- my_lu(A)
# Obtendo a inversa
inv_A <- my_solve(LU = LU, B = I)
inv_A
```

```
##            [,1]        [,2]        [,3]
## [1,] -0.1428571 -0.20408163  0.36734694
## [2,] -0.1428571  0.36734694 -0.06122449
## [3,]  0.2857143 -0.02040816 -0.16326531
```

Aplicação: Inversa via decomposição LU

```r
# Verificando o resultado
A%*%inv_A
```

```
##      [,1]          [,2]          [,3]
## [1,]    1  6.938894e-17  0.000000e+00
## [2,]    0  1.000000e+00 -5.551115e-17
## [3,]    0 -2.775558e-17  1.000000e+00
```

Decomposição de matrizes

- Uma infinidade de estratégias para decompor uma matriz em outras mais simples estão disponíveis.
- As decomposições mais usadas em estatística são:
  1. Decomposição em autovalores e autovetores (`eigen()`).
  2. Decomposição QR (`qr()`).
  3. Decomposição de Cholesky (`chol()`).
  4. Entre outras.

Matrizes esparsas

Matrizes esparsas (tópico adicional)

- Matrizes aparecem em todos os tipos de aplicação em ciência de dados.
- Modelos estatísticos, *machine learning*, análise de texto, análise de *cluster*, etc.
- Muitas vezes as matrizes usadas têm uma grande quantidade de zeros.
- Quando uma matriz tem uma quantidade considerável de zeros, dizemos que ela é **esparsa**, caso contrário dizemos que a matriz é **densa**.
- Todas as propriedades que vimos para matrizes em geral valem para matrizes esparsas.
- O `R` tem um conjunto de métodos altamente eficiente por meio do pacote `Matrix`.
- Saber que uma matriz é esparsa é útil pois permite:
  - Planejar formas de armazenar a matriz em memória.
  - Economizar cálculos em algoritmos numéricos (multiplicação, inversa, determinante, decomposições, etc).

Matrizes esparsas

- Comparando a quantidade de memória utilizada.

```r
library('Matrix')
m1 <- matrix(0, nrow = 1000, ncol = 1000)
m2 <- Matrix(0, nrow = 1000, ncol = 1000, sparse = TRUE)
object.size(m1)
```

```
## 8000216 bytes
```

```r
object.size(m2)
```

```
## 9240 bytes
```

Comparando o tempo computacional

- Matriz esparsa

```
y <- rnorm(1000)
X <- Matrix(NA, ncol = 100, nrow = 1000)
for(i in 1:1000) {X[i,] <- rbinom(100, size = 1, p = 0.1)}
X <- Matrix(X, sparse = TRUE)
system.time(replicate(100, solve(t(X)%*%X, t(X)%*%y)))
```

```
##    user  system elapsed
##   0.257   0.011   0.269
```

- Matriz densa

```
y <- rnorm(1000)
X <- matrix(NA, ncol = 100, nrow = 1000)
for(i in 1:1000) {X[i,] <- rbinom(100, size = 1, p = 0.1)}
system.time(replicate(100, solve(t(X)%*%X, t(X)%*%y)))
```

```
##    user  system elapsed
##   0.926   0.023   0.952
```

Diferentes formas de implementar as operações matriciais

- Criando a base de dados para a comparação

```
library(Matrix)
n <- 10000; p <- 500
x <- matrix(rbinom(n*p, 1, 0.01), nrow=n, ncol=p)
X <- Matrix(x)
object.size(x)
```

```
## 20000216 bytes
```

```
object.size(X)
```

```
## 602304 bytes
```

Diferentes formas de implementar as operações matriciais

- Diferentes implementações

```r
y <- rnorm(n)
system.time(solve(t(x)%*%x, t(x)%*%y))
```

```
##    user  system elapsed
##   2.327   0.028   2.358
```

```r
system.time(solve(crossprod(x), crossprod(x, y)))
```

```
##    user  system elapsed
##   1.224   0.024   1.250
```

```r
system.time(solve(t(X)%*%X, t(X)%*%y))
```

```
##    user  system elapsed
##   0.087   0.000   0.088
```

```r
system.time(solve(crossprod(X), crossprod(X,y)))
```

```
##    user  system elapsed
##   0.032   0.000   0.076
```