

# Introdução a Álgebra Linear para Ciência de Dados

Prof. Wagner Hugo Bonat

Curso de Especialização em  
Data Science & Big Data  
Universidade Federal do Paraná

12 de março de 2020



# Conteúdo

# Conteúdo

## 1. Vetores

- 1.1 Definição;
- 1.2 Operações com vetores;
- 1.3 Dependência e independência linear;
- 1.4 Vetores em R.

## 2. Matrizes

- 2.1 Definição;
- 2.2 Operações com matrizes;
- 2.3 Matrizes especiais;
- 2.4 Inversa de uma matriz;
- 2.5 Determinante de uma matriz;
- 2.6 Propriedades de matrizes.

## 3. Aplicação: Regressão linear múltipla.

## 4. Matrizes esparsas

- 4.1 Definição;
- 4.2 Matrizes esparsas em R: O pacote Matrix.
- 4.3 Tempo computacional e uso de memória.



# Vetores

# Vetores

- ▶ 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.

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

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

# 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.

$$\mathbf{v} = [v_1 \quad \dots \quad v_n] \quad \text{ou} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}.$$

- ▶ 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  $\mathbf{x}$  e  $\mathbf{y}$  adequados e  $\alpha$  um escalar, as seguintes operações são bem definidas.
  1. Soma  $\mathbf{x} + \mathbf{y} = [x_i + y_i] = [x_1 + y_1, \dots, x_n + y_n]$ .
  2. Subtração  $\mathbf{x} - \mathbf{y} = [x_i - y_i] = [x_1 - y_1, \dots, x_n - y_n]$ .
  3. Multiplicação por escalar  $\alpha\mathbf{x} = [\alpha x_1, \dots, \alpha x_n]$ .
  4. Transposta de um vetor: A operação transposta transforma um **vetor coluna** em um **vetor linha** e vice-versa. Por exemplo,

$$\mathbf{x} = [x_1 \quad \dots \quad x_n] \quad \mathbf{x}^T = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$



# Multiplicação de dois vetores

- ▶ Produto interno ou escalar → resulta um escalar (número), i.e.

$$\mathbf{x} \cdot \mathbf{y} = [x_1y_1 + x_2y_2 + \dots + x_ny_n].$$

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

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \dots + \alpha_n\mathbf{v}_n = \mathbf{0}$$

é satisfeita se e somente se  $\alpha_1 = \alpha_2 = \dots = \alpha_n = 0$ .

- ▶ 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

# Multiplicação por escalar
alpha = 10
alpha*x

## [1] 40 50 60

alpha*y

## [1] 10 20 30

# 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 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:

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

- ▶ 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

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

# 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.

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



# 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  $\mathbf{B}$ .
- ▶  $\mathbf{C} = \mathbf{A} \mathbf{B}$ .  
 $m \times n \quad m \times q \quad q \times n$
- ▶ 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.

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

- ▶ Elementos  $a_{ij}$  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

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

- ▶ Matriz triangular superior

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

# 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  $\mathbf{AB} = \mathbf{I}$ .
- ▶ A matrix **B** é chamada inversa de **A** e denotada por  $\mathbf{A}^{-1}$ .
- ▶ Resumindo,

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

- ▶ Inversas são fundamentais em *Data Science*.
- ▶ Em geral não calculamos explicitamente.
- ▶ São extremamente 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  $(3 \times 3)$ .

# Determinante de uma matriz

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

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

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  $1, 2, \dots, n$ .

- Exemplo,

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



# Propriedades de matrizes

- Sendo **A**, **B** e **C** matrizes adequadas as seguintes propriedades são válidas.

1.  $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$ .
2.  $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$ .
3.  $\alpha(\mathbf{A} + \mathbf{B}) = \alpha\mathbf{A} + \alpha\mathbf{B}$ .
4.  $(\alpha + \beta)\mathbf{A} = \alpha\mathbf{A} + \beta\mathbf{A}$ .

- Sendo **A** e **B** quadradas em geral  $\mathbf{AB} \neq \mathbf{BA}$ .

1.  $(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$ .
2.  $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$ .
3.  $\alpha(\mathbf{AB}) = (\alpha\mathbf{A})\mathbf{B} = \mathbf{A}(\alpha\mathbf{B})$ .
4.  $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$ .
5.  $(\mathbf{A}^T)^T = \mathbf{A}$ .
6.  $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$ .
7.  $(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$ .

# Matrizes em R

## ► Iniciando matrizes em 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
```

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

# Subtração
A - B

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

# 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

```
# 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

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

# Determinante e Inversa

## ► Determinante e inversa

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

## [1] 0.36

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

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

A%%inv_A # Matriz identidade

##           [,1] [,2]
## [1,]      1   0
## [2,]      0   1
```



# Aplicação: Regressão Linear múltipla

# Regressão linear simples

- ▶ Seja  $y_i (i = 1, \dots, n)$  observações de alguma variável de interesse.
- ▶ Seja  $x_i$  uma outra variável que queremos relacionar com  $y_i$  através de um reta, i.e.

$$y_i = \beta_0 + \beta_1 x_i.$$

- ▶ Problema: Encontrar  $\beta_0$  e  $\beta_1$  tal que

$$SQ = f(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_i))^2,$$

seja a menor possível.

- ▶ Processo extremamente tedioso.

# Regressão linear múltipla

- ▶ Suponha que ao invés de uma única  $x_i$  temos um vetor de dimensão  $p$  possivelmente grande.
- ▶ O modelo fica dado por

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots \beta_p x_{ip}.$$

- ▶ Como temos  $i = 1, \dots, n$  observações, temos

$$y_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \dots \beta_p x_{1p}$$

$$y_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \dots \beta_p x_{2p}$$

$$\vdots$$

$$y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots \beta_p x_{np}$$



# Regressão linear múltipla

- Matricialmente, temos

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} 1 & x_{11} & \dots & x_{p1} \\ 1 & x_{12} & \dots & x_{p1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & \dots & x_{pn} \end{bmatrix}_{n \times p} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_p \end{bmatrix}_{p \times 1}$$

- Usando uma notação mais compacta,

$$\begin{matrix} \mathbf{y} \\ n \times 1 \end{matrix} = \begin{matrix} \mathbf{X} \\ n \times p \end{matrix} \begin{matrix} \boldsymbol{\beta} \\ p \times 1 \end{matrix}.$$

# Regressão linear múltipla

- Objetivo: Encontrar o vetor  $\hat{\beta}$ , tal que

$$SQ(\beta) = (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta),$$

seja a menor possível.

- Derivando em  $\beta$ , temos

$$\begin{aligned}\frac{\partial SQ(\beta)}{\partial \beta} &= \frac{\partial}{\partial \beta} (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta) \\ &= \frac{\partial}{\partial \beta} \left( (\mathbf{y} - \mathbf{X}\beta)^\top \right) (\mathbf{y} - \mathbf{X}\beta) + (\mathbf{y} - \mathbf{X}\beta)^\top \frac{\partial}{\partial \beta} (\mathbf{y} - \mathbf{X}\beta) \\ &= -\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\beta) + (\mathbf{y} - \mathbf{X}\beta)^\top (-\mathbf{X}) \\ &= -2\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\beta).\end{aligned}$$

- Resolvendo o sistema,

$$\begin{aligned}\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\hat{\beta}) &= 0 \\ \mathbf{X}^\top \mathbf{y} - \mathbf{X}^\top \mathbf{X}\hat{\beta} &= 0 \\ \mathbf{X}^\top \mathbf{X}\hat{\beta} &= \mathbf{X}^\top \mathbf{y} \\ \hat{\beta} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}\end{aligned}$$

# Implementação em R

## ► Matriz de delineamento.

```
# Número de covariáveis
p = 100
# Matriz para armazenar as covariáveis
X <- matrix(NA, ncol = 100, nrow = 1000)
for(i in 1:1000) {X[i,] <- rbinom(100, size = 1, p = 0.1)}
X <- model.matrix(~ X) # Incluir a coluna de 1's (Intercepto)
```

## ► Vetor de parâmetros e valores simulados para y.

```
beta <- c(10, seq(-1,1, l = 100))
mu = X%*%beta
y <- rnorm(1000, mean = mu, sd = 1)
```

## ► Obtendo $\hat{\beta}$ .

```
# Abordagem ingênua
hat_beta = solve(t(X)%*%X)%*%t(X)%*%y
# Nunca use o solve() diretamente
hat_beta1 <- solve(t(X)%*%X, t(X)%*%y)
# Tempo computacional
system.time(replicate(100, solve(t(X)%*%X)%*%t(X)%*%y))

##      user      system elapsed
##    0.640    0.864    0.406

system.time(replicate(100, hat_beta1 <- solve(t(X)%*%X, t(X)%*%y)))

##      user      system elapsed
##    0.387    0.495    0.239
```

# Matrizes esparsas

# Matrizes esparsas

- ▶ Matrizes vão aparecer em todos os tipos de aplicação de *Data Science*.
- ▶ Modelos estatísticos, *machine learning*, análise de texto, análise de agrupamento, etc.
- ▶ Muitas vezes a matriz tem uma grande quantidade de zeros.
- ▶ Lembre-se no exemplo de regressão linear múltiplas nossas covariáveis eram binárias.
- ▶ Quando uma matriz tem uma quantidade considerável de 0's, dizemos que ela é **esparsa**, caso contrário dizemos que a matriz é **densa**.
- ▶ Saber que uma matriz é esparsa é útil em dois sentidos:
  1. Planejar formas de armazenar a matriz em memória.
  2. Economizar cálculos em algoritmos numéricos (multiplicação, inversa, determinante, decomposições, etc).

# Matrizes esparsas

- ▶ Todas as propriedades que vimos para matrizes em geral valem para matrizes esparsas.
- ▶ R tem um conjunto de métodos altamente eficiente para lidar com matrizes esparsas.
- ▶ Implementação através do pacote Matrix.
- ▶ Comparando quantidade de memória utilizada.

```
library('Matrix')  
m1 <- matrix(0, nrow = 1000, ncol = 1000)  
m2 <- Matrix(0, nrow = 1000, ncol = 1000, sparse = TRUE)  
object.size(m1)  
  
## 8000200 bytes  
  
object.size(m2)  
  
## 5632 bytes
```

# Matrizes esparsas: Regressão linear múltipla

- Implementação usando matrizes esparsas.

```
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
##    0.264    0.000    0.263
```

- O ganho em tempo computacional e em armazenamento será tanto maior quanto for a quantidade de zeros na matriz.

# Matrizes esparsas: Regressão linear múltipla

## ► Tamanho dos objetos.

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

## 20000200 bytes

object.size(X)

## 596872 bytes
```



# Matrizes esparsas: Regressão linear múltipla

## ► Diferentes formas de fazer as operações matriciais.

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

##      user      system elapsed
## 0.494    0.264    0.305

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

##      user      system elapsed
## 0.174    0.103    0.079

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

##      user      system elapsed
## 0.090    0.029    0.091

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

##      user      system elapsed
## 0.021    0.003    0.016
```

# Matrizes esparsas: Regressão linear múltipla

## ► Implementação usando o glmnet.

```
library(glmnet)
system.time(b <- coef(lm(y~x)))

##      user  system elapsed
##  2.776   3.430   1.913

system.time(glmnet(x, y, nlambda=1, lambda=0, standardize=FALSE))

##      user  system elapsed
##  0.079   0.007   0.087

system.time(g <- glmnet(X, y, nlambda=1, lambda=0, standardize=FALSE))

##      user  system elapsed
##  0.119   0.007   0.127
```